



Andrew S. Tanenbaum
David J. Wetherall

Computernetzwerke

5., aktualisierte Auflage

Computernetzwerke



**Andrew S. Tanenbaum
David J. Wetherall**

Computernetzwerke

5., aktualisierte Auflage

PEARSON

Higher Education

München • Harlow • Amsterdam • Madrid • Boston
San Francisco • Don Mills • Mexico City • Sydney
a part of Pearson plc worldwide

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.dnb.de> abrufbar.

Die Informationen in diesem Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.

Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.

Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht ausgeschlossen werden.

Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen. Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Autor dankbar.

Authorized translation from the English language edition, entitled COMPUTER NETWORKS by ANDREW S. TANENBAUM, DAVID J. WETHERALL, published by Pearson, Inc., publishing as Prentice Hall, Copyright © 2012, GERMAN Language edition published by Pearson Deutschland GmbH, Copyright © 2012.

Es konnten nicht alle Rechteinhaber von Abbildungen ermittelt werden. Sollte dem Verlag gegenüber der Nachweis der Rechtsinhaberschaft geführt werden, wird das branchenübliche Honorar nachträglich gezahlt.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.

Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Fast alle Produktbezeichnungen und weitere Stichworte und sonstige Angaben, die in diesem Buch verwendet werden, sind als eingetragene Marken geschützt. Da es nicht möglich ist, in allen Fällen zeitnah zu ermitteln, ob ein Markenschutz besteht, wird das ®-Symbol in diesem Buch nicht verwendet.

10 9 8 7 6 5 4 3 2 1

13 12

ISBN 978-3-86894-137-1

© 2012 by Pearson Deutschland GmbH
Martin-Kollar-Straße 10-12, D-81829 München/Germany
Alle Rechte vorbehalten
www.pearson.de

A part of Pearson plc worldwide
Programmleitung: Birger Peil, bpeil@pearson.de
Development: Alice Kachnij, akachnij@pearson.de
Fachlektorat: Professor Dr. Falko Dressler
Übersetzung: Katharina Pieper, Berlin; Petra Alm, Saarbrücken
Korrektorat: Katharina Pieper, Berlin
Einbandgestaltung: Thomas Arlt, tarlt@adesso21.net
Herstellung: Monika Weiher, mweiher@pearson.de
Satz: mediaService, Siegen (www.media-service.tv)
Druck und Verarbeitung: Drukarnia Dimograf, Bielsko-Biala

Printed in Poland

Inhaltsübersicht

| | |
|--|------|
| Vorwort | 15 |
| Kapitel 1 Einleitung | 21 |
| Kapitel 2 Die Bitübertragungsschicht | 119 |
| Kapitel 3 Die Sicherungsschicht | 233 |
| Kapitel 4 Die MAC-Teilschicht (Medium Access Control) | 303 |
| Kapitel 5 Die Vermittlungsschicht | 411 |
| Kapitel 6 Die Transportschicht | 565 |
| Kapitel 7 Die Anwendungsschicht | 693 |
| Kapitel 8 Sicherheit in Netzen | 863 |
| Kapitel 9 Leseempfehlungen und Bibliografie | 989 |
| Register | 1015 |

Inhaltsverzeichnis

| | |
|--|-----|
| Vorwort | 15 |
| Kapitel 1 Einleitung | 21 |
| 1.1 Einsatz von Rechnernetzen | 24 |
| 1.1.1 Geschäftsanwendungen | 24 |
| 1.1.2 Anwendungen im Privatbereich | 27 |
| 1.1.3 Mobile Benutzer | 32 |
| 1.1.4 Gesellschaftliche Aspekte | 36 |
| 1.2 Netzhardware | 39 |
| 1.2.1 Personal Area Network | 41 |
| 1.2.2 Lokale Netze (LANs) | 42 |
| 1.2.3 Stadtnetze (MANs) | 46 |
| 1.2.4 Fernnetze (WANs) | 47 |
| 1.2.5 Internetworks | 51 |
| 1.3 Netzsoftware | 53 |
| 1.3.1 Protokollhierarchien | 53 |
| 1.3.2 Entwurfsaspekte bei Schichten | 58 |
| 1.3.3 Verbindungsorientierte und verbindungslose Dienste | 60 |
| 1.3.4 Basisoperationen von Diensten | 63 |
| 1.3.5 Beziehung zwischen Diensten und Protokollen | 65 |
| 1.4 Referenzmodelle | 66 |
| 1.4.1 Das OSI-Referenzmodell | 66 |
| 1.4.2 Das TCP/IP-Referenzmodell | 70 |
| 1.4.3 Das in diesem Buch benutzte Modell | 73 |
| 1.4.4 Vergleich: OSI- und TCP/IP-Referenzmodell* | 74 |
| 1.4.5 Kritik am OSI-Modell und OSI-Protokollen* | 76 |
| 1.4.6 Kritik am TCP/IP-Referenzmodell* | 79 |
| 1.5 Beispielnetze | 80 |
| 1.5.1 Das Internet | 80 |
| 1.5.2 Mobilfunknetze der dritten Generation* | 91 |
| 1.5.3 Drahtlose LANs: IEEE 802.11* | 96 |
| 1.5.4 RFID und Sensornetze* | 100 |
| 1.6 Standardisierung von Netzen | 103 |
| 1.6.1 Who's who in der Welt der Telekommunikation | 104 |
| 1.6.2 Who's who in der Welt der internationalen Normen | 106 |
| 1.6.3 Who's who in der Welt der Internetstandards | 108 |
| 1.7 Metrische Einheiten | 110 |
| 1.8 Überblick über das restliche Buch | 111 |
| Zusammenfassung | 113 |
| Übungsaufgaben | 114 |

| | | |
|------------------|---|-----|
| Kapitel 2 | Die Bitübertragungsschicht | 119 |
| 2.1 | Theoretische Grundlagen der Datenübertragung..... | 121 |
| 2.1.1 | Fourieranalyse..... | 121 |
| 2.1.2 | Signale mit begrenzter Bandbreite | 122 |
| 2.1.3 | Maximale Datenübertragungsrate eines Kanals..... | 125 |
| 2.2 | Gerichtete Übertragungsmedien..... | 127 |
| 2.2.1 | Magnetische Medien..... | 127 |
| 2.2.2 | Twisted-Pair-Kabel | 128 |
| 2.2.3 | Koaxialkabel | 130 |
| 2.2.4 | Trägerfrequenzanlagen | 130 |
| 2.2.5 | Glasfaserleiter | 132 |
| 2.3 | Drahtlose Übertragung | 137 |
| 2.3.1 | Das elektromagnetische Spektrum..... | 138 |
| 2.3.2 | Funkübertragung..... | 141 |
| 2.3.3 | Mikrowellenübertragung | 143 |
| 2.3.4 | Infrarotübertragung..... | 147 |
| 2.3.5 | Lichtübertragung..... | 147 |
| 2.4 | Kommunikationssatelliten* | 149 |
| 2.4.1 | Geostationäre Satelliten | 150 |
| 2.4.2 | MEO-Satelliten | 154 |
| 2.4.3 | LEO-Satelliten | 154 |
| 2.4.4 | Vergleich: Satelliten und Glasfaser | 157 |
| 2.5 | Digitale Modulation und Multiplexing | 159 |
| 2.5.1 | Basisbandübertragung..... | 159 |
| 2.5.2 | Übertragung im Durchlassbereich | 164 |
| 2.5.3 | Frequenzmultiplexverfahren | 167 |
| 2.5.4 | Zeitmultiplexverfahren | 169 |
| 2.5.5 | Codemultiplexverfahren | 170 |
| 2.6 | Das öffentliche Telefonnetz | 173 |
| 2.6.1 | Aufbau des Telefonsystems | 174 |
| 2.6.2 | Politik im Bereich Telefonie..... | 177 |
| 2.6.3 | Teilnehmeranschlüsse: Modems, ADSL und Glasfaser..... | 180 |
| 2.6.4 | Verbindungsleitungen und Multiplexverfahren | 189 |
| 2.6.5 | Vermittlung | 197 |
| 2.7 | Das Mobiltelefonsystem* | 202 |
| 2.7.1 | Die erste Generation der Mobiltelefone: Analoge Sprache | 204 |
| 2.7.2 | Die zweite Generation der Mobiltelefone: Digitale Sprache | 208 |
| 2.7.3 | Die dritte Generation der Mobiltelefone: Digitale Sprache und Daten | 212 |
| 2.8 | Kabelfernsehen* | 217 |
| 2.8.1 | Gemeinschaftsantennen-Fernsehen | 217 |
| 2.8.2 | Internet über Kabel | 218 |
| 2.8.3 | Zuweisung der Frequenzbereiche | 220 |
| 2.8.4 | Kabelmodems | 221 |
| 2.8.5 | Vergleich: ADSL und Kabel | 224 |
| | Zusammenfassung | 226 |
| | Übungsaufgaben | 227 |

| | | |
|------------------|--|-----|
| Kapitel 3 | Die Sicherungsschicht | 233 |
| 3.1 | Entwurfsaspekte der Sicherungsschicht | 235 |
| 3.1.1 | Dienste für die Vermittlungsschicht. | 235 |
| 3.1.2 | Rahmenbildung | 238 |
| 3.1.3 | Fehlerüberwachung. | 242 |
| 3.1.4 | Flusskontrolle | 243 |
| 3.2 | Fehlererkennung und -korrektur | 244 |
| 3.2.1 | Fehlerkorrekturcodes | 246 |
| 3.2.2 | Fehlererkennungscodes. | 252 |
| 3.3 | Grundlegende Protokolle der Sicherungsschicht | 258 |
| 3.3.1 | Ein utopisches Simplexprotokoll | 263 |
| 3.3.2 | Ein Simplexprotokoll mit Stop-and-Wait für einen fehlerfreien Kanal | 265 |
| 3.3.3 | Ein Stop-and-Wait-Simplexprotokoll für verrauschte Kanäle. | 267 |
| 3.4 | Schiebefensterprotokolle | 271 |
| 3.4.1 | 1-Bit-Schiebefensterprotokoll | 274 |
| 3.4.2 | Protokoll mit Go-back-N | 277 |
| 3.4.3 | Protokoll mit selektiver Wiederholung | 284 |
| 3.5 | Beispiele für Protokolle der Sicherungsschicht. | 291 |
| 3.5.1 | Packet over SONET | 291 |
| 3.5.2 | ADSL | 295 |
| | Zusammenfassung. | 298 |
| | Übungsaufgaben | 299 |
| Kapitel 4 | Die MAC-Teilschicht (Medium Access Control) | 303 |
| 4.1 | Die Kanalzuordnung. | 305 |
| 4.1.1 | Statische Kanalzuordnung | 305 |
| 4.1.2 | Voraussetzungen für dynamische Kanalzuordnung | 307 |
| 4.2 | Mehrfachzugriffsprotokolle | 309 |
| 4.2.1 | ALOHA | 309 |
| 4.2.2 | CSMA-Protokolle (Carrier Sense Multiple Access) | 314 |
| 4.2.3 | Kollisionsfreie Protokolle | 317 |
| 4.2.4 | Protokolle mit eingeschränkter Konkurrenz | 322 |
| 4.2.5 | Protokolle für drahtlose LANs | 325 |
| 4.3 | Ethernet | 328 |
| 4.3.1 | Bitübertragungsschicht von klassischem Ethernet | 329 |
| 4.3.2 | Ethernet-MAC-Teilschichtprotokoll. | 331 |
| 4.3.3 | Leistungsaspekte bei Ethernet. | 335 |
| 4.3.4 | Switched Ethernet. | 337 |
| 4.3.5 | Fast Ethernet | 339 |
| 4.3.6 | Gigabit-Ethernet. | 342 |
| 4.3.7 | 10-Gigabit-Ethernet | 346 |
| 4.3.8 | Das Ethernet – ein Rückblick | 347 |
| 4.4 | Drahtlose LANs | 349 |
| 4.4.1 | IEEE-802.11-Architektur und -Protokollstapel. | 349 |
| 4.4.2 | Die IEEE-802.11-Bitübertragungsschicht | 351 |
| 4.4.3 | Das IEEE-802.11-MAC-Teilschichtprotokoll | 353 |

| | | |
|------------------|--|-----|
| 4.4.4 | IEEE-802.11-Rahmenstruktur | 360 |
| 4.4.5 | Dienste | 362 |
| 4.5 | Drahtloses Breitband* | 364 |
| 4.5.1 | Vergleich von IEEE 802.16 mit IEEE 802.11 und 3G | 365 |
| 4.5.2 | IEEE-802.16-Architektur und -Protokollstapel | 366 |
| 4.5.3 | IEEE-802.16-Bitübertragungsschicht | 367 |
| 4.5.4 | IEEE-802.16-MAC-Teilschichtprotokoll | 369 |
| 4.5.5 | IEEE-802.16-Rahmenstruktur | 371 |
| 4.6 | Bluetooth* | 372 |
| 4.6.1 | Architektur von Bluetooth | 373 |
| 4.6.2 | Bluetooth-Anwendungen | 374 |
| 4.6.3 | Bluetooth-Protokollstapel | 375 |
| 4.6.4 | Bluetooth-Funkschicht | 376 |
| 4.6.5 | Bluetooth-Verbindungsschicht | 377 |
| 4.6.6 | Bluetooth-Rahmenstruktur | 378 |
| 4.7 | RFID* | 380 |
| 4.7.1 | Architektur von EPC Gen 2 | 381 |
| 4.7.2 | Bitübertragungsschicht von EPC Gen 2 | 382 |
| 4.7.3 | Tag-Identifizierungsschicht von EPC Gen 2 | 383 |
| 4.7.4 | Nachrichtenformate der Tag-Identifizierung | 385 |
| 4.8 | Switches der Sicherungsschicht | 386 |
| 4.8.1 | Verwendung von Bridges | 386 |
| 4.8.2 | Learning-Bridges | 388 |
| 4.8.3 | Spannbäume und Bridges | 391 |
| 4.8.4 | Repeater, Hubs, Bridges, Switches, Router und Gateways | 394 |
| 4.8.5 | Virtuelle LANs | 397 |
| | Zusammenfassung | 405 |
| | Übungsaufgaben | 406 |
| Kapitel 5 | Die Vermittlungsschicht | 411 |
| 5.1 | Entwurfsaspekte der Vermittlungsschicht | 413 |
| 5.1.1 | Paketvermittlung unter Verwendung des Store-and-forward-Verfahrens | 413 |
| 5.1.2 | Dienste für die Transportschicht | 414 |
| 5.1.3 | Implementierung eines verbindungslosen Dienstes | 415 |
| 5.1.4 | Implementierung eines verbindungsorientierten Dienstes | 417 |
| 5.1.5 | Vergleich von VC-Netzen und Datagrammnetzen | 418 |
| 5.2 | Routing-Algorithmen | 420 |
| 5.2.1 | Das Optimalitätsprinzip | 422 |
| 5.2.2 | Routing nach dem kürzesten Pfad | 423 |
| 5.2.3 | Fluten | 427 |
| 5.2.4 | Distanzvektoralgorithmus | 428 |
| 5.2.5 | Link-State-Routing | 431 |
| 5.2.6 | Hierarchisches Routing | 437 |
| 5.2.7 | Broadcast-Routing | 439 |
| 5.2.8 | Multicast-Routing | 441 |
| 5.2.9 | Anycast-Routing | 444 |

| | | |
|------------------|---|-----|
| 5.2.10 | Routing für mobile Hosts | 445 |
| 5.2.11 | Routing in Ad-hoc-Netzen | 448 |
| 5.3 | Algorithmen zur Überlastungsüberwachung | 452 |
| 5.3.1 | Prinzipien der Überlastungsüberwachung | 454 |
| 5.3.2 | Routing unter Berücksichtigung des Verkehrs | 455 |
| 5.3.3 | Zugangssteuerung | 457 |
| 5.3.4 | Drosseln des Verkehrs | 458 |
| 5.3.5 | Lastabwurf | 463 |
| 5.4 | Dienstgüte | 465 |
| 5.4.1 | Anforderungen der Anwendungen | 466 |
| 5.4.2 | Traffic-Shaping | 468 |
| 5.4.3 | Scheduling der Pakete | 473 |
| 5.4.4 | Zugangssteuerung | 477 |
| 5.4.5 | Integrierte Dienste | 481 |
| 5.4.6 | Differenzierte Dienste | 484 |
| 5.5 | Internetworking | 487 |
| 5.5.1 | Unterscheidungsmerkmale von Netzen | 488 |
| 5.5.2 | Verbindung von Netzen | 490 |
| 5.5.3 | Tunneling | 493 |
| 5.5.4 | Internetwerk-Routing | 494 |
| 5.5.5 | Fragmentierung der Pakete | 496 |
| 5.6 | Vermittlungsschicht im Internet | 500 |
| 5.6.1 | IPv4 | 503 |
| 5.6.2 | IP-Adressen | 507 |
| 5.6.3 | IP Version 6 | 520 |
| 5.6.4 | Internetsteuerprotokolle | 531 |
| 5.6.5 | Label Switching und MPLS | 536 |
| 5.6.6 | OSPF-Protokoll | 540 |
| 5.6.7 | BGP | 546 |
| 5.6.8 | Internet-Multicasting | 551 |
| 5.6.9 | Mobiles IP | 553 |
| | Zusammenfassung | 557 |
| | Übungsaufgaben | 558 |
| Kapitel 6 | Die Transportschicht | 565 |
| 6.1 | Dienste der Transportschicht | 567 |
| 6.1.1 | Dienste für die oberen Schichten | 567 |
| 6.1.2 | Dienstprimitive der Transportschicht | 569 |
| 6.1.3 | Berkeley-Sockets | 573 |
| 6.1.4 | Beispiel für Socket-Programmierung: Ein Internetdateiserver | 575 |
| 6.2 | Elemente von Transportprotokollen | 580 |
| 6.2.1 | Adressierung | 581 |
| 6.2.2 | Verbindungsaufbau | 584 |
| 6.2.3 | Freigabe von Verbindungen | 591 |
| 6.2.4 | Fehlerüberwachung und Flusskontrolle | 595 |
| 6.2.5 | Multiplexing | 601 |
| 6.2.6 | Systemwiederherstellung | 602 |

| | | |
|------------------|--|-----|
| 6.3 | Überlastungsüberwachung | 604 |
| 6.3.1 | Gewünschte Bandbreitenzuordnung. | 605 |
| 6.3.2 | Regulierung der Senderate. | 609 |
| 6.3.3 | Probleme mit der drahtlosen Übertragung | 614 |
| 6.4 | Internettransportprotokolle: UDP | 616 |
| 6.4.1 | Einführung in UDP. | 616 |
| 6.4.2 | Entfernte Prozeduraufrufe | 618 |
| 6.4.3 | Echtzeittransportprotokolle. | 621 |
| 6.5 | Internettransportprotokolle: TCP | 628 |
| 6.5.1 | Einführung in TCP. | 628 |
| 6.5.2 | TCP-Dienstmodell | 629 |
| 6.5.3 | TCP-Protokoll. | 632 |
| 6.5.4 | TCP-Header | 633 |
| 6.5.5 | Verbindungsaufbau in TCP | 637 |
| 6.5.6 | Verbindungsfreigabe in TCP | 638 |
| 6.5.7 | Modellierung der Verwaltung von TCP-Verbindungen | 639 |
| 6.5.8 | TCP-Schiebefenster | 642 |
| 6.5.9 | Verwaltung von Timern in TCP | 646 |
| 6.5.10 | TCP-Überlastungsüberwachung | 649 |
| 6.5.11 | Die Zukunft von TCP | 659 |
| 6.6 | Leistungsaspekte* | 660 |
| 6.6.1 | Leistungsprobleme in Rechnernetzen. | 661 |
| 6.6.2 | Messung der Netzwerkleistung | 662 |
| 6.6.3 | Hostdesign für eine Optimierung der Leistung | 666 |
| 6.6.4 | Schnelle Segmentverarbeitung | 669 |
| 6.6.5 | Header-Komprimierung. | 672 |
| 6.6.6 | Protokolle für Long Fat Networks | 674 |
| 6.7 | Verzögerungstolerante Netze*. | 679 |
| 6.7.1 | DTN-Architektur | 680 |
| 6.7.2 | Bündel-Protokoll | 683 |
| | Zusammenfassung. | 686 |
| | Übungsaufgaben. | 687 |
| Kapitel 7 | Die Anwendungsschicht | 693 |
| 7.1 | DNS – Domain Name System | 695 |
| 7.1.1 | DNS-Namensraum | 696 |
| 7.1.2 | Ressourcendatensätze | 699 |
| 7.1.3 | Nameserver. | 703 |
| 7.2 | E-Mail*. | 708 |
| 7.2.1 | Architektur und Dienste | 709 |
| 7.2.2 | Benutzeragenten. | 711 |
| 7.2.3 | Nachrichtenformate | 716 |
| 7.2.4 | Nachrichtenübertragung | 725 |
| 7.3 | World Wide Web | 734 |
| 7.3.1 | Übersicht über die Architektur | 735 |
| 7.3.2 | Statische Webdokumente | 752 |
| 7.3.3 | Dynamische Webseiten und Webanwendungen | 763 |

| | | |
|------------------|--|-----|
| 7.3.4 | HTTP – HyperText Transfer Protocol | 775 |
| 7.3.5 | Das mobile Web | 787 |
| 7.3.6 | Suchen im Web | 789 |
| 7.4 | Streaming Audio und Video | 792 |
| 7.4.1 | Digitales Audio | 794 |
| 7.4.2 | Digitales Video | 799 |
| 7.4.3 | Streaming von gespeicherten Medien | 808 |
| 7.4.4 | Streaming von Live-Medien | 817 |
| 7.4.5 | Echtzeitkonferenzen | 821 |
| 7.5 | Content Delivery | 832 |
| 7.5.1 | Content und Internetverkehr | 833 |
| 7.5.2 | Serverfarmen und Webproxys | 836 |
| 7.5.3 | Content-Delivery-Netze | 841 |
| 7.5.4 | Peer-to-Peer-Netze | 846 |
| | Zusammenfassung | 857 |
| | Übungsaufgaben | 858 |
| Kapitel 8 | Sicherheit in Netzen | 863 |
| 8.1 | Kryptografie | 868 |
| 8.1.1 | Einführung in die Kryptografie | 868 |
| 8.1.2 | Substitutionschiffren | 871 |
| 8.1.3 | Transpositionschiffren | 873 |
| 8.1.4 | One-Time Pads | 874 |
| 8.1.5 | Zwei Grundprinzipien der Verschlüsselung | 879 |
| 8.2 | Algorithmen für die symmetrische Verschlüsselung | 881 |
| 8.2.1 | DES – Data Encryption Standard | 883 |
| 8.2.2 | AES – Advanced Encryption Standard | 886 |
| 8.2.3 | Chiffriermodi | 890 |
| 8.2.4 | Weitere Chiffren | 895 |
| 8.2.5 | Kryptoanalyse | 896 |
| 8.3 | Algorithmen für öffentliche Schlüssel | 897 |
| 8.3.1 | RSA | 898 |
| 8.3.2 | Weitere Algorithmen für öffentliche Schlüssel | 900 |
| 8.4 | Digitale Signaturen | 901 |
| 8.4.1 | Signaturen mit symmetrischen Schlüsseln | 902 |
| 8.4.2 | Signaturen mit öffentlichen Schlüsseln | 903 |
| 8.4.3 | Message Digests | 905 |
| 8.4.4 | Die Geburtstagsattacke | 909 |
| 8.5 | Verwaltung öffentlicher Schlüssel | 911 |
| 8.5.1 | Zertifikate | 912 |
| 8.5.2 | X.509 | 914 |
| 8.5.3 | PKI – Infrastruktur für öffentliche Schlüssel | 915 |
| 8.6 | Kommunikationssicherheit | 919 |
| 8.6.1 | IPsec | 919 |
| 8.6.2 | Firewalls | 924 |
| 8.6.3 | Virtuelle private Netze | 927 |
| 8.6.4 | Drahtlose Sicherheit | 929 |

| | | |
|------------|--|------|
| 8.7 | Authentifizierungsprotokolle | 934 |
| 8.7.1 | Authentifizierung auf der Basis eines gemeinsamen geheimen Schlüssels | 935 |
| 8.7.2 | Einrichten eines gemeinsamen Schlüssels: Das Schlüsselaustauschprotokoll von Diffie und Hellman | 940 |
| 8.7.3 | Authentifizierung mithilfe eines Schlüsselverteilungszentrums .. | 942 |
| 8.7.4 | Authentifizierung mit Kerberos | 945 |
| 8.7.5 | Authentifizierung mithilfe öffentlicher Verschlüsselung | 948 |
| 8.8 | E-Mail-Sicherheit* | 949 |
| 8.8.1 | PGP – Pretty Good Privacy | 949 |
| 8.8.2 | S/MIME | 954 |
| 8.9 | Sicherheit im Web | 954 |
| 8.9.1 | Bedrohungen der Sicherheit | 954 |
| 8.9.2 | Sichere Namensvergabe | 955 |
| 8.9.3 | SSL – Secure Sockets Layer | 961 |
| 8.9.4 | Sicherheit bei mobilem Code | 965 |
| 8.10 | Soziale Themen | 969 |
| 8.10.1 | Datenschutz | 969 |
| 8.10.2 | Redefreiheit | 972 |
| 8.10.3 | Urheberrechte | 976 |
| | Zusammenfassung | 979 |
| | Übungsaufgaben | 981 |
| 9.1 | Kapitel 9 Leseempfehlungen und Bibliografie | 989 |
| 9.1.1 | Empfehlungen für weiterführende Literatur* | 990 |
| 9.1.2 | Einführung und allgemeine Werke | 990 |
| 9.1.3 | Bitübertragungsschicht | 991 |
| 9.1.4 | Sicherungsschicht | 992 |
| 9.1.5 | MAC-Teilschicht | 993 |
| 9.1.6 | Vermittlungsschicht | 993 |
| 9.1.7 | Transportschicht | 995 |
| 9.1.8 | Anwendungsschicht | 995 |
| 9.2 | Netzsicherheit | 996 |
| | Alphabetische Bibliografie* | 997 |
| | Register | 1015 |

Vorwort

Dieses Buch liegt nun in der fünften überarbeiteten Auflage vor. Es hat sich parallel mit den verschiedenen Entwicklungs- und Nutzungsphasen von Rechnernetzen verändert. Als 1980 die erste Ausgabe erschien, waren Netze eine akademische Kuriosität. 1988, als die zweite Auflage erschien, wurden Netze fast nur an Universitäten und in Großunternehmen eingesetzt. Die dritte Auflage erschien 1996, zu einer Zeit, in der Rechnernetze und insbesondere das Internet bereits für Millionen von Menschen in aller Welt zu einem festen Bestandteil des Alltags geworden waren. Bei Erscheinen der vierten Auflage im Jahr 2003 waren drahtlose Netze und mobile Computer gang und gäbe geworden, um auf das Web und das Internet zuzugreifen. Nun, da es die fünfte Auflage gibt, geht es bei Netzen um die Verteilung von Inhalten (insbesondere von Videos unter Benutzung von CDNs und Peer-to-Peer-Netzen) und Mobiltelefone sind jetzt kleine Computer mit Internetzugang.

Neu in der fünften Auflage

Unter den vielen Änderungen in diesem Buch ist die wichtigste, dass Prof. David J. Wetherall als Mitautor hinzugekommen ist. David besitzt einen reichhaltigen Hintergrund im Bereich Networking, er hat schon vor mehr als 20 Jahren beim Entwurf von MANs seine ersten eigenen Erfahrungen gesammelt. Seitdem hat er mit dem Internet und drahtlosen Netzen gearbeitet und ist Professor an der Universität von Washington, wo er in den letzten zehn Jahren im Bereich Rechnernetze und zu verwandten Themen gelehrt und geforscht hat.

Natürlich finden sich im Buch außerdem viele Änderungen, um mit der sich stetig wandelnden Welt der Rechnernetze Schritt zu halten. Darunter ist überarbeitetes und neues Material zu

- drahtlosen Netzen (IEEE 802.12 und 802.16)
- den 3G-Netzen, die von Smartphones eingesetzt werden
- RFID und Sensornetzen
- Verteilung von Inhalten mithilfe von CDNs
- Peer-to-Peer-Netzen
- Echtzeitmedien (von gespeicherten, Streaming- und Live-Quellen)
- Internettelefonie (Voice-over-IP)
- verzögerungstoleranten Netzen

Hier folgt eine ausführlichere Auflistung nach Kapiteln:

Kapitel 1 hat zwar den gleichen einführenden Charakter wie in der vierten Auflage, jedoch wurde der Inhalt völlig überarbeitet und aktualisiert. Das Internet, Mobilfunknetze, IEEE 802.11 sowie RFID und Sensornetze werden als Beispiele für Rechnernetze vorgestellt. Lehrmaterial über das originale Ethernet – mit seinen Vampirklemmen – wurde ebenso wie ATM herausgenommen.

Kapitel 2 behandelt die Bitübertragungsschicht und befasst sich dabei ausführlich mit digitaler Modulation (einschließlich OFDM als häufig genutzte Methode in drahtlosen Netzen) und 3G-Netzen (die auf CDMA basieren). Außerdem werden neue Technologien diskutiert, darunter *Fiber to the Home* und Trägerfrequenzanlagen.

Kapitel 3 über Punkt-zu-Punkt-Verbindungen wurde auf zwei Arten verbessert. Die Inhalte über Codes zur Fehlererkennung und -korrektur wurden aktualisiert und enthalten nun außerdem eine kurze Beschreibung der modernen Codes, die in der Praxis wichtig sind (z.B. Faltungscodes und LDPC-Codes). Die Beispiele von Protokollen verwenden jetzt Packet over SONET und ADSL. Leider wurde das Lehrmaterial zur Protokollverifizierung entfernt, da diese wenig eingesetzt wird.

In **Kapitel 4** geht es um die MAC-Teilschicht. Hier sind die Prinzipien zeitlos, aber die Technologien haben sich geändert. Die Abschnitte zu den Beispielnetzen wurden entsprechend überarbeitet, einschließlich Gigabit-Ethernet, IEEE 802.11, Bluetooth und RFID. Ebenfalls aktualisiert wurde die Behandlung von LAN-Switching, einschließlich VLAN.

In **Kapitel 5** wird die Vermittlungsschicht behandelt, das Kapitel hat die gleichen Schwerpunkte wie in der vierten Auflage. Die Überarbeitungen bestanden hier darin, die Inhalte zu aktualisieren und zu vertiefen, insbesondere die Themen Dienstgüte (relevant für Echtzeitmedien) und Internetworking. Die Abschnitte über BGP, OSPF und CIDR wurden ausgeweitet, ebenso wie die Behandlung von Multicast-Routing. Anycast-Routing ist neu hinzugekommen.

In **Kapitel 6**, in dem es um die Transportschicht geht, wurde Lehrstoff hinzugefügt, überarbeitet und entfernt. Neue Inhalte beschreiben verzögerungstolerante Netzwerke und Überlastungsüberwachung im Allgemeinen. Das überarbeitete Material aktualisiert und erweitert die Behandlung von TCP-Überlastungsüberwachung. Das entfernte Material beschrieb verbindungsorientierte Vermittlungsschichten, welche heutzutage kaum noch verwendet werden.

Kapitel 7 über die Anwendungsschicht wurde ebenfalls aktualisiert und erweitert. Während sich bei DNS und E-Mail seit der vierten Auflage nicht viel geändert hat, gab es in den letzten paar Jahren viele Entwicklungen in der Benutzung von Web, Streaming Media und Content Delivery. Entsprechend wurden die Abschnitte über das Web und Streaming Media auf den neuesten Stand gebracht. Ein neuer Abschnitt behandelt die Verbreitung von Inhalten, einschließlich CDNs und Peer-to-Peer-Netze.

Kapitel 8 zum Thema Sicherheit behandelt immer noch sowohl symmetrische als auch Public-Key-Kryptografie für Vertraulichkeit und Authentifizierung. Informationen zu Techniken, die in der Praxis verwendet werden, wozu Firewalls und VPNs gehören, wurden aktualisiert und mit neuem Material zur IEEE-802.11-Sicherheit und Kerberos V5 angereichert.

Kapitel 9 enthält eine erneuerte Liste mit Leseempfehlungen und eine umfassende Bibliografie mit über 300 Literaturhinweisen aus der aktuellen Fachliteratur. Mehr als die Hälfte dieser Artikel oder Bücher wurden nach dem Jahr 2000 geschrieben, der Rest sind Hinweise auf klassische Arbeiten.

Sprach- und Wortwahl in der deutschen Übersetzung

Eine Herausforderung in der Übersetzung technischer Begrifflichkeiten ins Deutsche ist immer die grundlegende Verständlichkeit. In älteren Ausgaben wurden möglichst viele Begriffe übersetzt. Dies führte ganz offensichtlich manchmal zu Verwirrungen, insbesondere, wenn man Sekundärliteratur in das Studium der Rechnernetze einbeziehen wollte. Beispiele sind „Keller“ vs. „Stack“ oder „Schnittstelle“ vs. Interface. In den letzten Jahren wurden immer mehr Begriffe „eingedeutscht“ bzw. werden auch in Vorlesungen und Vorträgen meist in ihrer englischen Form genutzt. Beispiele dafür sind Peer-to-Peer-Netze, Streaming Media, Proxies oder das Web. Daher haben wir versucht, in der aktuellen Auflage einen besseren Kompromiss zu erzielen. Grundsätzlich werden alle Fachbegriffe sowohl englisch als auch deutsch eingeführt. Je nach typischem Gebrauch haben wir uns dann für das bekanntere Wort entschieden. Weiterhin sind alle standardisierten Protokollelemente konsequent englisch benannt, wohl aber deutsch erklärt.

Liste der Akronyme

Fachbücher über Computer sind normalerweise voller Akronyme. Das vorliegende bildet keine Ausnahme. Wenn Sie dieses Buch durchgearbeitet haben, sollten Ihnen die folgenden Abkürzungen etwas sagen: ADSL, AES, AJAX, AODV, AP, ARP, ARQ, AS, BGP, BOC, CDMA, CDN, CGI, CIDR, CRL, CSMA, CSS, DCT, DES, DHCP, DHT, DIFS, DMCA, DMT, DMZ, DNS, DOCSIS, DOM, DSLAM, DTN, FCFS, FDD, FDDI, FDM, FEC, FIFO, FSK, FTP, GPRS, GSM, HDTV, HFC, HMAC, HTTP, IAB, ICANN, ICMP, IDEA, IETF, IMAP, IMP, IP, IPTV, IRTF, ISO, ISP, ITU, JPEG, JSP, JVM, LAN, LATA, LEC, LEO, LLC, LSR, LTE, MAN, MFJ, MIME, MPEG, MPLS, MSC, MTSO, MTU, NAP, NAT, NRZ, NSAP, OFDM, OSI, OSPF, PAWS, PCM, PGP, PIM, PKI, POP, POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RFID, RPC, RSA, RTSP, SHA, SIP, SMTP, SNR, SOAP, SONET, SPE, SSL, TCP, TDD, TDM, TSAP, UDP, UMTS, URL, VLAN, VSAT, WAN, WDM und XML. Aber keine Sorge, jedes Akronym wird in **Fettschrift** eingeführt und sorgfältig erläutert, bevor wir es verwenden. Machen Sie doch zum Spaß einmal den Test, um zu sehen, wie viele der Abkürzungen Sie identifizieren können, *bevor* Sie das Buch lesen. Schreiben Sie sich die Anzahl an den Rand, dann versuchen Sie es noch einmal, *nachdem* Sie das Buch gelesen haben.

Hinweise zur Benutzung des Buchs

Um Dozenten mit unterschiedlich langen Unterrichtseinheiten – je nachdem ob es sich um Quartale oder Semester handelt – die Arbeit mit diesem Buch zu erleichtern, haben wir die Kapitel in Kernlehrstoff und optionales Material eingeteilt. Die Abschnitte, die im Inhaltsverzeichnis mit einem „*“ gekennzeichnet sind, sind die optionalen Teile. Falls ein größerer Abschnitt markiert ist (z.B. 2.7), dann sind alle dazugehörigen Unterabschnitte ebenfalls optional. Sie enthalten Material zu Netztechnologien, die nützlich sind, aber in einem kurzen Kurs ausgelassen werden können, ohne den Zusammenhang zu verlieren. Natürlich sollten Studenten ermutigt werden, diese Abschnitte auch zu lesen, in dem Maße, wie sie Zeit haben, da alles Material aktuell und wertvoll ist.

Hilfsmaterialien für Dozenten



Die folgenden geschützten Hilfsmaterialien für Dozenten sind auf der Website des Verlags
www.pearson-studium.de/4137

verfügbar. Um einen Benutzernamen und ein Passwort zu erhalten, kontaktieren Sie bitte Ihren lokalen Pearson-Vertreter.

- Lösungshandbuch
- PowerPoint-Vortragsfolien

Hilfsmaterialien für Studenten



Materialien für Studenten sind über die frei zugängliche Companion-Website
www.pearson-studium.de/4137

erhältlich. Dort finden Sie:

- Webseite mit Links zu Tutorenkursen, Organisationen, FAQs und vieles mehr
- die Abbildungen, Tabellen und Programme aus dem Buch
- Demo zur Steganografie
- Protokollsimulatoren

Danksagung

Unzählige Leute haben uns bei der Entstehung der fünften Auflage unterstützt. Unser besonderer Dank gilt Emmanuel Agu (Worcester Polytechnic Institute), Yoris Au (Universität von Texas/Antonio), Nikhil Bhargava (Aircom International, Inc.), Michael Buettner (Universität von Washington), John Day (Universität Boston), Kevin Fall (Intel Labs), Ronald Fulle (Rochester Institute of Technology), Ben Greenstein (Intel Labs), Daniel Halperin (Universität von Washington), Bob Kinicki (Worcester Polytechnic Institute), Tadayoshi Kohno (Universität von Washington), Sarvish Kulkarni (Universität Villanova), Hank Levy (Universität von Washington), Ratul Mahajan (Microsoft Research), Craig Partridge (BBN), Michael Piatek (Universität von Washington), Joshua Smith (Intel Labs), Neil Spring (Universität von Maryland), David Teneyuca (University of Texas/Antonio), Tammy VanDegrift (Universität von Portland) and Bo Yuan (Rochester Institute of Technology). Sie alle haben uns mit ihren Ideen und ihrem Feedback geholfen. Melody Kadenko und Julie Svendsen haben David bei Verwaltungsaufgaben unterstützt.

Shivakant Mishra (Universität von Colorado/Boulder) und Paul Nagin (Chimborazo Publishing, Inc.) haben sich viele neue und anspruchsvolle Aufgaben ausgedacht, die am Ende jedes Kapitels zu finden sind. Unsere Lektorin bei Pearson, Tracy Dunkelberger, war wie immer in den vielen großen und kleinen Dingen auf die ihr eigene Art hilfreich. Melinda Haggerty und Jeff Holcomb haben dafür gesorgt, dass alles reibungslos verläuft. Steve Armstrong (Universität LeTourneau) hat die PowerPoint-Folien vorbereitet. Stephen Turner (Universität von Michigan/Flint) hat geschickt die Webquellen und die Simulatoren überarbeitet, die den Text begleiten. Unsere Redakteurin, Rachel

Head, ist eine außerordentliche Person: Sie hat das Auge eines Adlers und das Gedächtnis eines Elefanten. Nachdem wir all ihre Korrekturen gelesen hatten, haben wir uns beide gefragt, wie wir jemals über die dritte Klasse hinausgekommen sind.

Zum Schluss kommen wir zu den wichtigsten Menschen. Suzanne hat dies nun 19 Male mitgemacht und bringt immer noch endlose Geduld und Liebe auf. Barbara und Marvin kennen heute den Unterschied zwischen guten und schlechten Lehrbüchern und sind immer eine Inspiration, ein gutes Lehrbuch herzustellen. Daniel und Matilde sind ein willkommener Zuwachs zu unserer Familie. Aron wird wahrscheinlich dieses Buch nicht so bald lesen, aber er mag die hübschen Bilder auf Seite 866 (AST). Katrin und Lucy haben mich unendlich unterstützt und es immer vermocht, ein Lächeln auf mein Gesicht zu zaubern. Danke (DJW).

Andrew S. Tanenbaum

David J. Wetherall

Über die Autoren

Andrew S. Tanenbaum hat einen Senior-Bachelor-Abschluss des MIT und die Doktorwürde der kalifornischen Universität in Berkeley. Er ist derzeit Professor für Informatik an der Vrije Universiteit in Amsterdam, wo er seit über 30 Jahren in den Bereichen Betriebssysteme, Netzwerke und verwandten Themen lehrt. In seiner aktuellen Forschungstätigkeit widmet er sich vor allem hoch zuverlässigen Betriebssystemen, doch er hat im Laufe der Jahre bereits in den Bereichen Compiler, verteilte Systeme, Sicherheit und zu verwandten Themen gearbeitet. Die Ergebnisse seiner Forschungsprojekte sind in über 150 Artikeln in Fachzeitschriften und Tagungsbänden dokumentiert.

Prof. Tanenbaum war außerdem (Mit-)Autor von fünf Büchern, die mittlerweile in 19 Ausgaben erschienen sind. Die Bücher wurden in 21 Sprachen übersetzt, von baskisch bis thailändisch, und werden an Universitäten auf der ganzen Welt eingesetzt. Alles in allem gibt es 159 Versionen (Kombinationen aus Sprache und Ausgabe), die aufgeführt sind unter www.cs.vu.nl/~ast/publications.

Prof. Tanenbaum ist außerdem Autor einer bedeutenden Anzahl von Softwareanwendungen, darunter das Amsterdam Compiler Kit (ein retargierbarer portable Compiler), Amoeba (ein frühes verteiltes System, das in LANs eingesetzt wird) und Globe (ein überregionales verteiltes System).

Er ist darüber hinaus der Autor von MINIX, einem kleinen Unix-Klon, das ursprünglich zum Einsatz in studentischen Programmierlaborene gedacht war. MINIX war die direkte Inspiration für Linux und war die Plattform, auf der Linux anfangs entwickelt wurde. Die aktuelle Version – MINIX 3 – legt den Schwerpunkt darauf, ein außerordentlich zuverlässiges und sicheres Betriebssystem zu sein. Prof. Tanenbaum wird seine Arbeit dann als getan ansehen, wenn kein Computer mehr mit einem Reset-Knopf ausgestattet ist und niemand mehr die Erfahrung eines Systemabsturzes erleben muss. MINIX 3 ist ein laufendes Open-Source-Projekt. Sie sind eingeladen, daran mitzuwirken. Besuchen Sie www.minix3.org, um eine kostenlose Kopie herunterzuladen und sich über den neuesten Stand zu informieren.

Prof. Tanenbaum ist Mitglied der ACM und der IEEE sowie Mitglied der Königlich-Niederländischen Akademie der Wissenschaften. Er hat außerdem zahlreiche wissenschaftliche Preise verliehen bekommen, darunter:

2010 TAA McGuffy Award für Lehrbücher im Bereich Informatik und Ingenieurwissenschaft

2007 IEEE James H. Mulligan, Jr. Education Medal

2002 TAA Texty Award für Lehrbücher im Bereich Informatik und Ingenieurwissenschaft

1997 ACM/SIGCSE Award for Outstanding Contributions to Computer Science Education

1994 ACM Karl V. Karlstrom Outstanding Educator Award

Seine Homepage im World Wide Web findet sich unter der URL

<http://www.cs.vu.nl/~ast/>.

David J. Wetherall ist außerordentlicher Professor der Informatik und Ingenieurwissenschaft an der Universität Washington in Seattle und Berater der Intel Labs in Seattle. Er stammt aus Australien, dort hat er seinen Bachelorabschluss in Elektrotechnik an der Universität von West-Australien erlangt. Er promovierte in Informatik am MIT.

Prof. Wetherall arbeitet seit zwei Jahrzehnten im Bereich Netzwerke. Seine Forschung konzentriert sich auf Netzsysteme, insbesondere drahtlose Netze und mobile Computer, den Entwurf von Internetprotokollen sowie Netzmessungen.

Er erhielt den ACM SIGCOMM Test-of-Time Award für Forschung, die den Weg für aktive Netze bereitete – eine Architektur für das Internet Mapping. Seine Forschung wurde 2002 mit einem NSF CAREER Award ausgezeichnet. Im Jahr 2004 wurde er ein Sloan Fellow.

Neben der Lehrtätigkeit zu Netzwerken ist Prof. Wetherall auch Mitglied der Netzforschungsgemeinde. Er war Co-Chair der Programmkomitees von SIGCOMM, NSDI und MobSys und Mitbegründer des ACM HotNets Workshop. Er hat bei unzähligen Programmkomitees für Netzkonferenzen mitgewirkt und ist Lektor für ACM Computer Communication Review.

Seine Homepage im World Wide Web finden Sie unter <http://djh.cs.washington.edu>.

Fachlektor **Falko Dressler** ist Professor der Informatik und Leiter des Lehrstuhls für Technische Informatik an der Universität Innsbruck. Seine Forschung und Lehre konzentriert sich auf verschiedenste Aspekte der Rechnernetze mit einem starken Fokus auf drahtlose Kommunikation und Selbstorganisationsmechanismen in massiv verteilten Systemen. Als IEEE Distinguished Lecturer und Autor des Lehrbuchs *Self-Organization in Sensor and Actor Networks* (Wiley) gibt er weltweit Tutorien zu aktuellen Themen moderner Rechnernetze.

Einleitung

| | | |
|------------|--|-----|
| 1.1 | Einsatz von Rechnernetzen | 24 |
| 1.2 | Netzhardware | 39 |
| 1.3 | Netzsoftware | 53 |
| 1.4 | Referenzmodelle | 66 |
| 1.5 | Beispielnetze | 80 |
| 1.6 | Standardisierung von Netzen | 103 |
| 1.7 | Metrische Einheiten | 110 |
| 1.8 | Überblick über das restliche Buch | 111 |

» Jedes der vergangenen drei Jahrhunderte wurde von einer einzelnen neuen Technologie beherrscht. Das 18. Jahrhundert war die Blütezeit der großen mechanischen Systeme, die die Industrielle Revolution begleiteten. Das 19. Jahrhundert war das Zeitalter der Dampfmaschine. Im 20. Jahrhundert spielten das Sammeln, Verarbeiten und Verteilen von Information die technologische Schlüsselrolle. Wir haben unter anderem die Einrichtung weltweiter Telefonnetze, die Erfindung von Radio und Fernsehen, die Geburt und den beispiellosen Aufstieg der Computerindustrie, den Start von Kommunikationssatelliten und – natürlich – das Aufkommen des Internets miterlebt.

Durch den rasanten technologischen Fortschritt verschmelzen diese Gebiete im 21. Jahrhundert mit zunehmender Geschwindigkeit und die Unterschiede zwischen dem Erfassen, Übertragen, Speichern und Verarbeiten von Informationen verschwinden immer mehr. Für Unternehmen mit Hunderten von weit voneinander entfernten Standorten ist es heute eine Selbstverständlichkeit, den augenblicklichen Geschäftsstand von jedem Standort – und sei er noch so fern – per Tastendruck abrufen zu können. Schneller noch als unsere Fähigkeiten auf dem Gebiet des Informationsmanagements zunehmen, entstehen neue Bedürfnisse nach noch besseren Datenverarbeitungssystemen.

Obwohl die Computerindustrie im Vergleich zu anderen Industrien (z.B. der Kfz- oder Luftfahrtindustrie) immer noch relativ jung ist, haben die Computer doch in kurzer Zeit spektakuläre Fortschritte gemacht. In den ersten zwei Jahrzehnten ihrer Existenz waren Computersysteme völlig zentralisiert, normalerweise in einem einzigen großen Raum. Nicht selten hatten diese Räume Glaswände, durch die Besucher das großartige elektronische Wunder bestaunen konnten. Ein Unternehmen mittlerer Größe oder eine Universität hatten zu dieser Zeit vielleicht einen oder zwei Computer und große Institutionen hatten höchstens ein paar Dutzend. Die Vorstellung, dass vierzig Jahre später Computer mit einer erheblich höheren Leistungsfähigkeit und einem geringeren Platzbedarf als eine Briefmarke in milliardenfacher Massenproduktion hergestellt würden, war reine Zukunftsmusik.

Die Verschmelzung von Computern und Kommunikation hat die Art, wie Computersysteme organisiert sind, grundlegend beeinflusst. Das einst vorherrschende Konzept eines Rechenzentrums in einem großen Raum mit nur einem Großrechner, in das alle Benutzer ihre Arbeiten zum Verarbeiten bringen, gehört heute zum alten Eisen (obwohl Datenzentren, die Tausende von Internetservern beherbergen, gerade üblich werden). Das alte Modell, bei dem ein Großrechner für ein Unternehmen die gesamte Nachfrage nach Rechenleistung befriedigte, wurde durch ein neues ersetzt, bei dem eine große Anzahl einzelner, miteinander verbundener Computer die Arbeit übernehmen. Diese Systeme nennt man **Rechnernetze**. Die Entwicklung und der Aufbau dieser Rechnernetze sind Gegenstand dieses Buchs.

In diesem Buch wird der Begriff „Rechnernetze“ für mehrere miteinander mit einer bestimmten Technologie verbundene autonome Computer verwendet. Zwei Computer gelten als miteinander verbunden, wenn sie Informationen austauschen können. Die Verbindung muss nicht aus einem Kupferdraht bestehen – es können auch Glasfaserkabel, Mikrowellen, Infrarotsignale und Kommunikationssatelliten verwendet werden. In diesem Buch werden Sie noch eine Vielzahl an Netzen mit verschiedenen

Ausdehnungen, Strukturen und Formen kennenlernen. Die Netze sind in der Regel miteinander verbunden, um größere Netze zu bilden. Das **Internet** ist das bekannteste Beispiel solch eines Netzes von Netzen.

In der einschlägigen Literatur werden die Begriffe Rechnernetz und **verteiltes System** (*distributed system*) häufig miteinander verwechselt. Das zentrale Unterscheidungskriterium ist, dass bei einem verteilten System die Benutzer mehrere unabhängige Computer als ein einheitliches, zusammenhängendes System wahrnehmen. In der Regel folgt es einer bestimmten Sichtweise oder einem Paradigma, unter der bzw. dem es sich den Benutzern präsentiert. Oftmals ist eine Software, die auf dem Betriebssystem aufsetzt, eine sogenannte **Middleware**, für die Implementierung dieses Paradigmas verantwortlich. Ein sehr bekanntes Beispiel eines verteilten Systems ist das **World Wide Web**. Dies setzt auf dem Internet auf und stellt eine Sicht zur Verfügung, in der alles wie ein Dokument (Webseite) aussieht.

In Rechnernetzen fehlen diese Kohärenz, die einheitliche Sichtweise und die zugehörige Software. Der Benutzer¹ muss mit den konkreten, eine bestimmte Funktionalität bereitstellenden Rechnern interagieren. Das System stellt hier keine Funktionalität bereit, um den Benutzern ein kohärentes Aussehen und Verhalten bereitzustellen. Haben die Rechner verschiedene Hardware und unterschiedliche Betriebssysteme, so ist dies für die Benutzer in vollem Umfang ersichtlich. Möchte ein Benutzer ein Programm auf einem entfernten Rechner ausführen, dann muss er sich bei dem entfernten Rechner anmelden und das Programm dort ausführen.

Ein verteiltes System ist im Grunde ein Softwaresystem, das auf ein Netz aufgesetzt ist. Durch die Software erhält es ein hohes Maß an Zusammenhalt und Transparenz. Somit liegt der Unterschied zwischen einem verteilten System und einem Rechnernetz bei der Software (speziell im Betriebssystem) und weniger bei der Hardware.

Trotzdem gibt es zwischen den beiden Themengebieten viele Überschneidungen. So müssen beispielsweise bei verteilten Systemen und Rechnernetzen Dateien transferiert werden. Die Frage ist nur, ob das System oder der Benutzer den Transfer veranlasst.

Obwohl sich dieses Buch hauptsächlich mit Netzen beschäftigt, sind viele der Themen auch für verteilte Systeme interessant. Weitere Informationen zu verteilten Systemen finden Sie in Tanenbaum und van Steen (2007).



¹ „Benutzer“ steht in diesem Buch immer für Benutzerin und Benutzer.

1.1 Einsatz von Rechnernetzen

Bevor wir uns den technischen Aspekten im Einzelnen zuwenden, sollten wir uns zunächst die Zeit nehmen aufzuzeigen, warum die Menschen an Rechnernetzen interessiert sind und wozu sie benutzt werden können. Denn wenn niemand an Rechnernetzen interessiert wäre, würden nur wenige installiert werden. Wir beginnen mit dem traditionellen Einsatz bei Unternehmen, gehen dann weiter zu Heimnetzwerken und neueren Entwicklungen wie mobilen Nutzern und beschließen diesen Abschnitt mit sozialen Aspekten.

1.1.1 Geschäftsanwendungen

Viele Unternehmen setzen bereits eine erhebliche Anzahl an Computern ein. Ein Unternehmen hat beispielsweise einen Rechner für jeden Angestellten, um Produkte zu entwerfen, Broschüren zu schreiben und die Gehaltsabrechnungen vorzunehmen. Ursprünglich wurden einige dieser Rechner vielleicht unabhängig voneinander betrieben, aber an einem gewissen Punkt hat sich das Management dazu entschlossen, sie alle zu verbinden, um so Informationen im ganzen Unternehmen verbreiten zu können.

Allgemein geht es hier um die **gemeinsame Nutzung von Ressourcen** (*resource sharing*) mit dem Ziel, alle in einem Netzwerk verfügbaren Programme, Geräte und insbesondere Daten allen Benutzern, ungeachtet des Standorts von Ressourcen oder Benutzern, verfügbar zu machen. Ein gängiges Beispiel ist eine Gruppe von Büroangestellten, die einen Drucker gemeinsam nutzen. Keiner der Angestellten benötigt einen Drucker nur für sich. Ein vernetzter Drucker mit hoher Durchsatzleistung ist oftmals billiger, schneller und einfacher zu warten als viele einzelne Drucker.

Neben der gemeinsamen Nutzung von physikalischen Betriebsmitteln wie Druckern oder Bandsystemen zur Datensicherung ist es aber meistens viel wichtiger, Informationen gemeinsam zu nutzen. Jedes Unternehmen, ob klein oder groß, ist entscheidend von elektronisch aufbereiteten Informationen abhängig. Die meisten Unternehmen verfügen über elektronisch gespeicherte Kundendatensätze, Produktinformationen, Außenstände, Jahresabschlüsse, Steuererklärungen und viele andere wichtige Informationen. Wenn bei einer Bank plötzlich alle Computer ausfallen würden, könnte diese keine fünf Minuten länger existieren. Ein modernes Fertigungsunternehmen mit einem computergesteuerten Montageband könnte nicht einmal fünf Sekunden überstehen. Selbst ein kleines Reisebüro oder eine Anwaltskanzlei mit drei Personen ist heutzutage in hohem Maß von Rechnernetzwerken abhängig, damit die Mitarbeiter sofort auf wichtige Informationen und Dokumente zugreifen können.

In kleineren Unternehmen befinden sich die Rechner häufig in einem Büro oder in einem Gebäude, aber bei großen Unternehmen können die Computer und Mitarbeiter über Dutzende von Büros in vielen Ländern versprengt sein. So muss ein Vertriebsmitarbeiter in New York manchmal auf eine Produktbestandsdatenbank in Singapur zugreifen. **Virtuelle private Netze (VPN)**, *(Virtual Private Network)* können eingesetzt werden, um die individuellen Netzwerke von verschiedenen Orten zu einem erweiterten

Netz zu verbinden. Anders ausgedrückt, soll ein Benutzer allein aufgrund der Tatsache, dass er sich 15 000 Kilometer von seinen Daten entfernt befindet, nicht daran gehindert werden, die Daten so zu nutzen, als wären sie lokal vorhanden. Dieses Ziel kann als Versuch aufgefasst werden, die „Tyrannei der Geografie“ zu beenden.

Sehr vereinfacht erscheint das Informationssystem eines Unternehmens als eine Sammlung von einer oder mehreren Datenbanken mit Unternehmensdaten sowie einigen Mitarbeitern, die auch von entfernten Standorten aus darauf zugreifen wollen. In diesem Modell werden die Daten in leistungsstarken Computern gespeichert, die **Server** genannt werden. Diese sind oftmals an einem zentralen Standort aufgestellt und werden von einem Systemadministrator verwaltet. Die Mitarbeiter haben dagegen einfache Rechner, sogenannte **Clients**, auf ihren Schreibtischen stehen, mit denen sie auf die entfernten Daten zugreifen, beispielsweise um diese in die Tabellenkalkulation aufzunehmen, die sie gerade erstellen. (Manchmal bezeichnet man auch den Benutzer des Client-Rechners als Client. Es sollte jedoch aus dem Kontext ersichtlich sein, ob es sich um einen Menschen oder seinen Rechner handelt.) Der Client- und der Server-Rechner sind über ein Netzwerk verbunden (►Abbildung 1.1). Beachten Sie, dass das Netzwerk als einfaches Oval ohne nähere Einzelheiten dargestellt wird. Diese Form wird im vorliegenden Buch immer verwendet, wenn wir ganz abstrakt von einem Netzwerk sprechen. Sind weitere Einzelheiten erforderlich, werden diese angegeben.

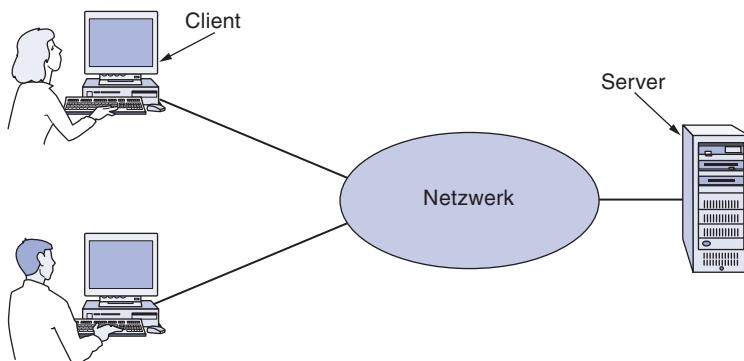


Abbildung 1.1: Ein Netzwerk mit zwei Clients und einem Server.

Diese gesamte Anordnung wird als **Client-Server-Modell** bezeichnet. Es ist stark verbreitet und bildet die Basis für viele Einsatzbereiche von Netzen. Die bekannteste Umsetzung dieses Prinzips sind **Webanwendungen**, bei denen ein Server Webseiten anhand seiner Datenbank als Antwort von Client-Anfragen erzeugt, welche die Datenbank aktualisieren können. Das Client-Server-Modell ist anwendbar, wenn sich Client und Server im gleichen Gebäude befinden (und zum gleichen Unternehmen gehören), es ist aber genauso anwendbar, wenn Client und Server räumlich weit voneinander entfernt sind. Wenn beispielsweise jemand von zu Hause auf das World Wide Web zugreift, gilt das gleiche Modell, wobei der entfernte Webserver als Server und der PC des Benutzers als Client fungiert. In den meisten Fällen kann ein Server eine große Anzahl (Hunderte oder Tausende) an Clients gleichzeitig bedienen.

Wenn wir uns das Client-Server-Modell genauer ansehen, entdecken wir zwei Prozesse (das heißt laufende Programme): einen auf dem Client-Rechner und einen auf dem Server-Rechner. Die Kommunikation findet statt, indem der Client-Prozess über das Netzwerk eine Nachricht an den Server-Prozess schickt. Der Client-Prozess wartet dann auf eine Antwort. Wenn der Server die Anforderung erhält, führt er die angeforderte Aufgabe aus oder sucht die angeforderten Daten heraus und sendet eine Antwort zurück. Diese Nachrichten sind in ▶Abbildung 1.2 dargestellt.

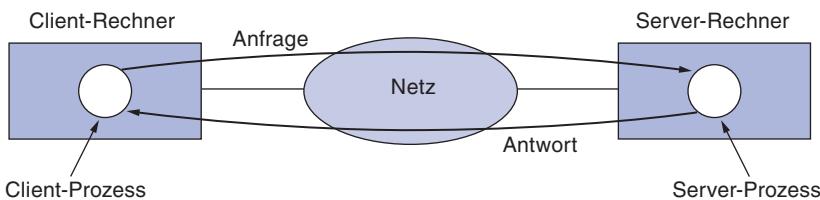


Abbildung 1.2: Das Client-Server-Modell enthält Anforderungen und Antworten.

Ein zweites Ziel bei der Einrichtung eines Rechnernetzes hat mehr mit den Personen als mit den Informationen oder gar Computern zu tun. Ein Rechnernetz kann für die Mitarbeiter ein leistungsstarkes **Kommunikationsmedium** zur Verfügung stellen. Praktisch jedes Unternehmen, das mehr als zwei Computer besitzt, hat heutzutage **E-Mail** (*Electronic Mail*), mit der die Mitarbeiter in der Regel einen Großteil ihrer täglichen Kommunikation abwickeln. Aber in der Kaffeeküche wird unisono über die Flut von E-Mails geklagt, die täglich zu bearbeiten ist, zumal viele dieser Nachrichten ziemlich bedeutungslos sind, weil der Chef entdeckt hat, dass er per Tastendruck ein und dieselbe (oft inhaltlose) Nachricht gleich an alle Mitarbeiter senden kann.

Telefonanrufe zwischen Mitarbeitern können über das Computernetz anstatt über die Telefongesellschaft getätigt werden. Diese Technologie heißt **IP-Telefonie**, **Internet-telefonie** bzw. **Voice-over-IP (VoIP)**, wenn Internettechnologie eingesetzt wird. Mikrofon und Lautsprecher am jeweiligen Ende können zu einem VoIP-fähigen Telefon gehören oder im Computer integriert sein. Unternehmen halten dies für eine wunderbare Art, ihre Telefonrechnungen niedrig zu halten.

Aber auch noch andere, reichere Formen der Kommunikation werden durch Rechnernetze ermöglicht. Video kann den Audiodaten hinzugefügt werden, sodass sich Mitarbeiter sehen und hören können, als würden sie eine normale Besprechung abhalten, auch wenn sie sich an weit voneinander entfernten Orten befinden. Diese Technik ist ein leistungsstarkes Mittel, um Reisezeit und -kosten enorm zu senken. Mithilfe von **Desktop-Sharing** können entfernte Mitarbeiter miteinander über einen grafischen Computerbildschirm interagieren. Dadurch ist es für zwei oder mehr räumlich getrennte Menschen leicht, ein gemeinsames Blackboard zu lesen und zu schreiben oder zusammen einen Bericht zu verfassen. Wenn einer der Autoren online eine Änderung an einem gemeinsamen Dokument durchführt, können die anderen dies sofort sehen, statt tagelang auf einen Brief warten zu müssen. Durch diese Beschleunigung können weitverstreute Personengruppen auf bisher nicht da gewesene Art zusammenarbeiten. Noch ambitioniertere Formen der entfernten Koordination wie Telemedizin stehen

erst am Anfang (wie die Fernüberwachung von Patienten), können aber noch sehr an Bedeutung gewinnen. Es wird manchmal behauptet, dass sich Transport und Kommunikation ein Wettrennen liefern. Eine Technik wird sich durchsetzen und die andere aussterben lassen.

Ein drittes Ziel für viele Unternehmen ist es, Geschäfte mit anderen Unternehmen elektronisch abzuwickeln, speziell mit Kunden und Lieferanten. Dieses neue Modell heißt **E-Commerce** (*Electronic Commerce*) und ist in den letzten Jahren rasant gewachsen. Fluglinien, Buchläden und andere Händler haben entdeckt, dass viele Kunden am bequemen Einkaufen von zu Hause aus Gefallen finden. In der Folge bieten viele Unternehmen Produktkataloge und Dienstleistungen online an und nehmen Bestellungen auch online entgegen. Zum Beispiel kaufen Hersteller von Kraftfahrzeugen, Flugzeugen oder Computern Teilsysteme von verschiedenen Zulieferern und montieren die Bauteile. Mithilfe von Netzen können die Hersteller Aufträge nach Bedarf in elektronischer Form erteilen. Es müssen keine hohen Lagerbestände mehr verwaltet werden und die Effizienz wird gesteigert.

1.1.2 Anwendungen im Privatbereich

Im Jahr 1977 war Ken Olsen Präsident der Digital Equipment Corporation, damals weltweit die Nummer zwei unter den Computeranbietern (nach IBM). Als er gefragt wurde, warum sich Digital nicht im großen Stil im PC-Markt engagiere, antwortete er: „Es gibt keinen Grund dafür, dass irgendjemand einen Computer bei sich zu Hause haben will.“ Die Geschichte hat das Gegenteil bewiesen – und Digital existiert heute nicht mehr. Anfangs haben Leute Computer zur Textverarbeitung und für Spiele gekauft. In jüngster Zeit ist der Internetzugang wahrscheinlich ein Hauptgrund, sich privat einen Rechner anzuschaffen. Heutzutage sind viele Geräte der Unterhaltungselektronik wie Set-Top-Boxen, Spielkonsolen und Radiowecker mit eingebetteten Computern zum Anschluss an Rechnernetze ausgerüstet, speziell für kabellose Netze. Heimnetze werden häufig zu Unterhaltungszwecken eingesetzt. Dazu gehört auch das Hören, Ansehen und Erzeugen von Musik, Fotos und Videos.

Der Internetzugang bietet privaten Nutzern **Konnektivität** (*connectivity*) zu entfernten Rechnern. Ebenso wie Unternehmen können Heimanwender auf Informationen zugreifen, mit anderen Personen kommunizieren und Produkte und Dienstleistungen mithilfe von E-Commerce kaufen. Der größte Gewinn ist heute die Verbindung mit der Außenwelt. Bob Metcalfe, der Erfinder des Ethernets, hat die Hypothese aufgestellt, dass der Wert eines Netzwerks proportional zum Quadrat der Anzahl seiner Nutzer ist, da dies grob die Anzahl der möglichen unterschiedlichen Verbindungen ist (Gilder, 1993). Diese Hypothese ist als „Metcalfes Gesetz“ bekannt und hilft zu erklären, warum die sagenhafte Beliebtheit des Internets aus seiner Größe abgeleitet werden kann.

Der Zugriff auf entfernte Informationen ist in vielen Formen möglich, beispielsweise beim Surfen durch das World Wide Web, um nach Informationen zu suchen, oder nur zum Zeitvertreib. Informationen über die verschiedensten Bereiche wie Kunst, Geschäfte, Kochen, Politik, Gesundheit, Geschichte, Hobbys, Freizeitgestaltung, Wissen-

schaft, Sport, Reise etc. stehen hier zur Verfügung. Möglichkeiten zur Unterhaltung gibt es auf so viele Arten, dass wir sie hier nicht alle erwähnen können, ganz zu schweigen von einigen Formen, die überhaupt unerwähnt bleiben.

Viele Zeitungen publizieren heute online und können personalisiert werden. In manchen Fällen kann der Leser einer Zeitung beispielsweise mitteilen, dass er alles über korrupte Politiker, große Waldbrände, die neuesten Skandale von Filmstars und Epidemien lesen will – aber bitte nichts über Fußball. Manchmal kann man auch die gewünschten Artikel in der Nacht, während man schlafst, herunterladen. Wenn sich diese Entwicklung fortsetzt, müssen wir mit einer massiven Arbeitslosigkeit unter den 12-jährigen Zeitungsasträgern rechnen. Doch den Zeitungsverlagen wird dies recht sein, da die Zustellung in der gesamten Produktionskette immer das schwächste Glied war. Um dieses Modell anwendbar zu machen, müssen die Verlage natürlich erst einmal herausfinden, wie man in dieser neuen Zeitungswelt Geld machen kann – dies ist ein Aspekt, der nicht völlig offensichtlich ist, da Internetnutzer davon ausgehen, dass alles kostenlos ist.

Der nächste Schritt über Zeitungen (sowie Zeitschriften und Fachjournale) hinaus ist die digitale Onlinebibliothek. Viele Fachorganisationen wie die ACM (www.acm.org, Association for Computing Machinery) und die IEEE Computer Society (www.computer.org, Institute of Electrical and Electronics Engineers) stellen bereits all ihre Journale und Konferenzberichte online zur Verfügung. Lesegeräte für elektronische Bücher und Onlinebüchereien könnten gedruckte Bücher demnächst überflüssig machen. Skeptiker mögen bedenken, welche Wirkung die Druckerpresse auf das mittelalterliche illuminierte Manuscript hatte.

Auf viele dieser Informationen wird über das Client-Server-Modell zugegriffen, doch es gibt noch ein weiteres beliebtes Modell dafür, die **Peer-to-Peer-Kommunikation** (Parameswaran et al., 2001). In dieser Form können Personen, die eine lose Gruppe bilden, mit anderen in der Gruppe kommunizieren, wie in ▶ Abbildung 1.3 dargestellt. Jede Person kann im Grunde mit einer oder mehreren Personen kommunizieren. Es besteht keine feste Aufteilung in Clients und Server.

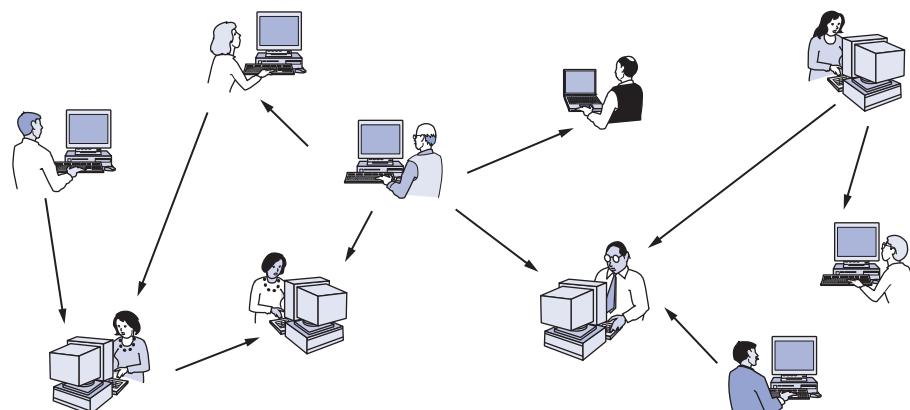


Abbildung 1.3: In einem Peer-to-Peer-System gibt es keine festen Clients und Server.

Viele Peer-to-Peer-Systeme wie BitTorrent (Cohen, 2003) haben keine zentrale Datenbank mit Inhalten, sondern jeder Benutzer verwaltet seine Datenbank lokal und besitzt eine Liste von in der Nähe lebenden Personen, die auch Mitglied in diesem System sind. Ein neuer Benutzer kann dann zu einem Mitglied gehen und nachsehen, was dieser hat, sowie die Namen der anderen Mitglieder bekommen, um diese wiederum auf weitere Inhalte und weitere Namen zu prüfen. Dieser Nachschlageprozess kann unbegrenzt wiederholt werden, um über die einzelnen Mitglieder eine große lokale Datenbank zu erstellen. Diese Aktivität wäre für Menschen sehr mühselig, aber Computer meistern dies hervorragend.

Peer-to-Peer-Kommunikation wird häufig zum Austausch von Musik und Videos eingesetzt. Einen Höhepunkt gab es um das Jahr 2000 mit der US-amerikanischen Musiktauschbörse Napster, die nach dem vermutlich größten Gerichtsverfahren zur Urheberrechtsverletzung in der Geschichtsschreibung geschlossen wurde (Lam und Tan, 2001; Macedonia, 2000). Es gibt aber auch legale Anwendungen der Peer-to-Peer-Kommunikation. Dazu gehören Fans, die Public-Domain-Musik gemeinsam nutzen, Familien, die Fotos und Filme austauschen, und Anwender, die öffentliche Softwarepakete herunterladen. Tatsächlich ist eine der beliebtesten Internetanwendungen überhaupt – das Versenden von E-Mails – eine Peer-to-Peer-Anwendung. Diese Kommunikationsform wird wahrscheinlich in der Zukunft noch beträchtlich wachsen.

Bei all den oben genannten Anwendungen interagiert eine Person mit einer entfernten Datenbank, die eine Vielzahl von Informationen enthält. Die zweite breite Kategorie der Nutzung von Rechnernetzen ist die interpersonelle Kommunikation. Sie ist gewissermaßen die Antwort des 21. Jahrhunderts auf das Telefon des 19. Jahrhunderts. E-Mail wird bereits von Millionen von Menschen in der ganzen Welt täglich verwendet und die Verbreitung nimmt rapide zu. Audio- und Videoinformationen sind hier genauso möglich wie Text und Abbildungen. Die Übertragung von Gerüchen wird wohl noch ein wenig auf sich warten lassen.

Jeder Teenager, der nur ein bisschen auf sich hält, ist glühender Anhänger des Austauschs von Sofortnachrichten ([Instant Messaging](#)). Diese Einrichtung, die vom Unix-Programm *talk* stammt und seit 1970 im Gebrauch ist, ermöglicht zwei Personen, in Echtzeit eingegebene Nachrichten auszutauschen. Es gibt auch Multi-Personen-Nachrichtendienste wie [Twitter](#), die es ermöglichen, kurze Textnachrichten, „Tweets“ genannt, an seinen Freundeskreis oder anderes interessiertes Publikum zu senden.

Das Internet kann von Anwendungen benutzt werden, um Audio- (z.B. Internetradio-sender) oder Videodaten (z.B. YouTube) zu übertragen. Abgesehen davon, dass es eine billige Art ist, entfernte Freunde anzurufen, können diese Anwendungen wertvolle Möglichkeiten bieten, wie zum Beispiel Telelearning. Damit kann man am Unterricht um 8 Uhr morgens teilnehmen, ohne hierfür aus dem Bett steigen zu müssen. Langfristig mögen bei der Nutzung von Netzwerken die Aspekte, die die zwischenmenschliche Kommunikation fördern, sogar wichtiger werden als viele andere. Dies könnte für Personen immens wichtig werden, die in strukturschwachen Regionen leben, da sie hierdurch auf die gleichen Dienste zugreifen können wie Personen, die in einer großen Stadt wohnen.

Zwischen der interpersonellen Kommunikation und dem Zugriff auf Informationen liegen die **sozialen Netzwerke**. Hier wird der Informationsfluss über die Beziehungen gesteuert, die die Mitglieder des Netzes untereinander haben. Eines der populärsten sozialen Netzwerke ist Facebook. Die Mitglieder können hier ihre persönlichen Profile aktualisieren und diese Aktualisierungen mit anderen Leuten teilen, die sie zu ihren Freunden erklärt haben. In anderen sozialen Netzwerken können neue Mitglieder über Freunde von Freunden eingeführt werden, neue Nachrichten an Freunde versandt werden, wie beim oben erwähnten Twitter, und vieles mehr.

Auf eine noch ungebundene Weise können Gruppen von Leuten zusammenarbeiten, um Inhalte zu erzeugen. Ein **Wiki** ist beispielsweise eine gemeinsame Website, die die Mitglieder einer Gemeinschaft bearbeiten. Das berühmteste Wiki ist **Wikipedia**, eine Enzyklopädie, an der jeder mitschreiben kann, doch es gibt auch Tausende anderer Wikis.

Unsere dritte Kategorie ist E-Commerce im weitesten Sinne. Homeshopping ist bereits sehr beliebt. Die Benutzer können die Onlinekataloge von Tausenden von Unternehmen durchforsten. Einige dieser Kataloge sind interaktiv, zeigen Produkte aus unterschiedlichen Blickwinkeln und in Anordnungen, die personalisiert werden können. Wenn ein Kunde nach dem Onlinekauf eines Produkts Probleme mit der Bedienung oder Benutzung hat, kann er sich gegebenenfalls an den technischen Online-Support wenden.

Ein weiterer Bereich, in dem E-Commerce viel genutzt wird, ist der Zugriff auf Finanzinstitute. Viele Leute zahlen ihre Rechnungen, verwalten ihre Bankkonten und tätigen ihre Anlagegeschäfte bereits auf elektronischem Weg. Dieser Trend wird sicherlich noch anhalten, wenn die Netze sicherer werden.

Ein Bereich, den eigentlich niemand vorhergesehen hat, sind elektronische Flohmärkte (E-Floh?). Onlineauktionen von gebrauchten Produkten sind zu einem riesigen Geschäft geworden. Im Unterschied zum herkömmlichen E-Commerce, der nach dem Client-Server-Modell abgewickelt wird, sind Onlineauktionen ein Peer-to-Peer-System in dem Sinne, dass Konsumenten sowohl als Verkäufer als auch als Käufer agieren können.

Einige Formen dieser elektronischen Geschäfte haben sich eigene putzige Abkürzungen geschaffen, die die Tatsache nutzen, dass im Englischen „to“ und „2“ gleich ausgesprochen werden. Die wichtigsten sind in ▶ Abbildung 1.4 aufgeführt.

| Kürzel | Vollständiger Name | Beispiel |
|--------|------------------------|---|
| B2C | Business-to-Consumer | Onlinebestellung von Büchern |
| B2B | Business-to-Business | Kfz-Hersteller bestellt Reifen beim Lieferanten |
| G2C | Government-to-Consumer | Finanzamt versendet elektronische Steuerformulare |
| C2C | Consumer-to-Consumer | Onlineauktionen von gebrauchten Artikeln |
| P2P | Peer-to-Peer | Gemeinsame Nutzung von Musik |

Abbildung 1.4: Einige E-Commerce-Modelle.

Unsere vierte Kategorie ist der Unterhaltungsbereich. Hier wurden in den letzten Jahren ungeheure Fortschritte erzielt und die Verbreitung von Musik, Radio- und Fernsehprogrammen sowie Filmen über das Internet beginnt, die traditionellen Vertriebswege auszustechen. Benutzer können MP3-Songs und Filme in DVD-Qualität finden, kaufen und herunterladen und ihrer persönlichen Sammlung hinzufügen. Fernsehsendungen erreichen heute viele Privathaushalte über **IPTV**-Systeme (**IP TeleVision**), die auf der IP-Technologie statt auf Kabelfernsehen oder Radioübertragung basieren. Anwendungen zum Streamen von Medien ermöglichen es Benutzern, Internetradiosendungen zu empfangen oder die letzte Episode ihrer Lieblingsserie anzusehen. Selbstverständlich können all diese Inhalte in Ihrem Haus zwischen verschiedenen Geräten, Bildschirmen und Lautsprechern herumwandern, in der Regel über ein drahtloses Netz.

Bald dürfte es möglich sein, nach jedem jemals für das Kino oder das Fernsehen produzierten Film in jedem beliebigen Land zu suchen und sofort auf dem Bildschirm abzuspielen. Neue Filme werden interaktiv, wobei der Benutzer gelegentlich aufgefordert wird, die Wendung der Geschichte zu bestimmen (soll Macbeth Duncan ermorden oder ihn nur mit Verachtung strafen?), wobei für alle Fälle alternative Szenarien geboten werden. Auch das Live-Fernsehen könnte interaktiv werden, was bedeutet, dass der Zuschauer vom Sessel aus an Quizshows mitmacht, zwischen Kandidaten auswählt usw.

Eine weitere Art der Unterhaltung sind Spiele. Schon jetzt gibt es Spiele für mehrere Spieler in Echtzeitsimulation, wie Verstecken in einem virtuellen Kerker und Flugsimulatoren, bei denen die Spieler eines Teams versuchen, die Spieler des gegnerischen Teams abzuschießen. Virtuelle Welten stellen einen permanenten Rahmen zur Verfügung, in dem Tausende von Nutzer eine gemeinsame Realität mit dreidimensionaler Grafik erfahren.

Unsere letzte Kategorie ist **Ubiquitous Computing** (auch **Rechnerallgegenwart**), bei dem die Datenverarbeitung, wie in der Vision von Mark Weiser (1991), in das tägliche Leben eingebettet ist. Viele Häuser sind schon mit Sicherheitssystemen ausgerüstet, die Tür- und Fenstersensoren enthalten, und es gibt noch mehr Sensoren, die in einem intelligenten Messgerät, z.B. zur Messung des Energieverbrauchs, untergebracht werden können. Ihre Strom-, Gas- und Wasserzähler könnten den Verbrauch auch über das Netz melden. Dadurch ließe sich Geld sparen, denn es müssten keine Ableser mehr in die Haushalte geschickt werden. Und Ihre Rauchmelder könnten direkt die Feuerwehr alarmieren, anstatt einen Riesenlärm zu veranstalten (was außerdem ziemlich sinnlos ist, wenn niemand zu Hause ist). Wenn die Kosten für Abtastung und Kommunikation weiter sinken, werden Messungen und Berichterstattung verstärkt über Netze abgewickelt werden.

Auch Geräte der Unterhaltungselektronik sind zunehmend vernetzt. Zum Beispiel besitzen einige hochwertige Kameras schon kabellose Netzwerkfähigkeit, um Fotos an einen in der Nähe stehenden Bildschirm zum Betrachten zu senden. Professionelle Sportfotografen können ebenfalls ihre Fotos an Redakteure in Echtzeit senden: zuerst per Funk zu einem Zugangspunkt und dann weiter über das Internet. Geräte wie Fernseher, die einen Wandanschluss besitzen, können **Trägerfrequenzanlagen** (*power-line network*) benutzen, um Informationen durch das Haus über die Stromkabel zu senden. Es mag nicht sehr

überraschen, diese Art von Objekten im Netzwerk zu finden, doch auch Objekte, die wir nicht als Computer ansehen, können Informationen aufnehmen und weiterleiten. Zum Beispiel könnte Ihre Dusche den Wasserverbrauch aufzeichnen, Ihnen eine visuelle Rückmeldung darüber geben, während Sie sich einseifen, und anschließend einen Bericht an ein Programm zur Verbrauchsdatenmessung im häuslichen Umfeld übermitteln, um Sie dabei zu unterstützen, Ihre Wasserrechnung niedrig zu halten.

Eine Technologie namens **RFID** (*Radio Frequency IDentification*) wird dieses Konzept sogar noch weiter in die Zukunft vorantreiben. RFID-Tags sind passive Chips (d.h., sie haben keine Batterie) in der Größe von Briefmarken und sie können bereits auf bzw. an Büchern, Reisepässen, Haustieren, Kreditkarten und anderen Objekten im Haus und außerhalb angebracht sein. RFID-Lesegeräte können diese Objekte dann lokalisieren und mit ihnen über eine Entfernung von bis zu mehreren Metern kommunizieren, je nach RFID-Typ. Ursprünglich wurde RFID kommerzialisiert, um Barcodes zu ersetzen. Dies war jedoch bisher noch nicht erfolgreich, da Barcodes kostenlos sind, während RFID-Tags ein paar Cent kosten. Natürlich bieten RFID-Tags viel mehr und ihr Preis fällt rapide. Sie könnten die Realwelt in das „Internet der Dinge“ verwandeln (ITU, 2005).

1.1.3 Mobile Benutzer

Mobile Computer, z.B. Notebooks und Handhelds, gehören zu den Segmenten in der Computerindustrie mit dem höchsten Wachstum. Ihre Verkaufszahlen haben bereits die der Desktop-Rechner überholt. Warum sollte irgendjemand so etwas wollen? Personen, die viel unterwegs sind, möchten mit ihrer mobilen Ausrüstung E-Mails und Tweets lesen und schreiben, Filme ansehen, Musik herunterladen, spielen oder einfach nur im Web nach Informationen surfen. Sie möchten all diese Dinge zu Hause und im Büro tun. Und natürlich wollen sie dies von jedem Ort aus tun, ob zu Land, zu Wasser oder in der Luft.

Konnektivität (*connectivity*) zum Internet ermöglicht viele dieser mobilen Nutzungen. Da eine Kabelverbindung in Autos, Flugzeugen und auf Booten unmöglich ist, besteht ein großes Interesse an Funknetzen. Mobilfunknetze, die von Telefongesellschaften betrieben werden, sind eine vertraute Art von drahtlosem Netzwerk, das uns mobiles Telefonieren ermöglicht. Kabellose **Hotspots**, die auf dem IEEE-802.11-Standard beruhen, sind eine weitere Art von kabellosen Netzen für mobile Rechner. Sie sind überall dort aus dem Boden geschossen, wo Menschen hingehen, was zu einer Art Abdeckungspatchwork in Cafés, Hotels, Flughäfen, Schulen, Zügen und Flugzeugen geführt hat. Jeder mit einem Laptop und einem schnurlosen Modem kann einfach seinen Rechner starten und ist über den Hotspot genau so mit dem Internet verbunden, als wäre der Computer über Kabel mit einem Netz verbunden.

Drahtlose Netze sind von unschätzbarem Wert für Fahrzeugflotten, z.B. Speditionen, Taxiunternehmen, Busunternehmen und alle Arten von Reparaturfirmen mit vielen Mechanikern im Außendienst, um den Kontakt mit ihrem Heimatstandort aufrechtzuhalten. Beispielsweise sind Taxifahrer in vielen Städten eigenständige Unternehmer und nicht Angestellte eines Taxiunternehmens. In einigen Städten sind die Taxis mit

einem Bildschirm im Sichtbereich des Fahrers ausgestattet. Wenn ein Kunde anruft, gibt ein zentraler Verteiler den Abhol- und den Zielort ein. Diese Informationen werden auf dem Bildschirm des Fahrers angezeigt und es wird ein Klingelton erzeugt. Der erste Fahrer, der eine Taste auf der Anzeige drückt, erhält den Anruf.

Drahtlose Netze sind auch für das Militär wichtig. Will man in der Lage sein, irgendwo auf der Welt kurzfristig einen Krieg zu führen, sollte man sich sicher nicht auf die örtliche Netzinfrastruktur verlassen. Da ist es schon besser, man bringt sein eigenes Netz mit.

Nun hängen drahtlose Netze und mobile Computer zwar oft zusammen, sind aber nicht gleichzusetzen (►Abbildung 1.5). Wir unterscheiden zwischen **festen drahtlosen** und **mobilien drahtlosen** Netzen. Selbst bei Notebooks werden manchmal Kabel verwendet. Wenn Sie beispielsweise auf Reisen Ihr Notebook in die verkabelte Netzan schlussbuchse im Hotelzimmer einstecken, sind Sie mobil, aber ohne drahtloses Netz.

| Drahtlos | Mobil | Typische Anwendungen |
|----------|-------|--|
| Nein | Nein | Arbeitsplatzrechner in Büros |
| Nein | Ja | Im Hotelzimmer benutztes Notebook |
| Ja | Nein | Netze in Gebäuden ohne Verkabelung |
| Ja | Ja | Erfassung des Lagerbestands mit einem Handheld |

Abbildung 1.5: Kombinationen von drahtlosen Netzen und mobilen Computern.

Umgekehrt sind einige drahtlose Computer nicht mobil. Zu Hause, in Büros oder Hotels, die keine passende Verkabelung haben, kann es zweckmäßiger sein, Desktop Rechner oder Mediaplayer drahtlos zu verbinden statt Kabel zu installieren. Für die Installation eines drahtlosen Netzes muss man unter Umständen nur ein kleines Kästchen mit etwas Elektronik kaufen, dieses auspacken und es einstecken. Diese Lösung kann viel billiger sein als Handwerker zu beauftragen, die Kabelführungen für die Netzwerkkabel durch das Gebäude legen.

Zu guter Letzt gibt es auch echte mobile drahtlose Anwendungen, zum Beispiel wenn Leute in Lagerhäusern mit Handhelds umherlaufen, um Lagerbestände aufzunehmen. In den Parkhäusern stark frequentierter Flughäfen arbeiten Mitarbeiter von Mietwagenfirmen mit drahtlosen mobilen Computern. Sie scannen den Barcode oder den RFID Chip des zurückgegebenen Autos ein und ihr mobiles Gerät, das einen eingebauten Drucker hat, holt sich die Daten vom Hauptrechner, stellt die Mietdaten zusammen und druckt vor Ort gleich eine Rechnung aus.

Der Hauptmotor der mobilen kabellosen Anwendungen ist vielleicht das Mobiltelefon. Das Senden von **Textnachrichten** (*text message*) ist unheimlich beliebt. Ein Mobiltelefonbenutzer tippt dazu eine kurze Nachricht ein, die dann über das Mobilfunknetz an einen anderen mobilen Teilnehmer übertragen wird. Nur wenige Leute hätten wohl vor zehn Jahren vorhergesagt, dass Teenager, die mühsam kurze Textnachrichten in ihre

Handys eintippen, eine enorme Einnahmequelle für Telefongesellschaften sein werden. Aber die **SMS**-Dienste (*Short Message Service*) sind sehr profitabel, da es den Netzbetreiber nur den Bruchteil eines Cents kostet, eine Textnachricht zu übermitteln – ein Dienst, den er sich weitaus höher bezahlen lässt.

Die lang erwartete Annäherung von Telefonen und dem Internet hat endlich stattgefunden und wird das Wachstum der mobilen Anwendungen weiter beschleunigen. Smartphones wie das bekannte iPhone vereinen Aspekte von Mobiltelefonen und mobilen Computern. Die benutzten (3G- und 4G-)Funknetze können sowohl schnellen Datenservice für den Internetzugriff bereitstellen als auch Telefonanrufe abwickeln. Viele fortschrittliche Telefone verbinden sich auch mit kabellosen Hotspots und wechseln automatisch das Netz, um dem Nutzer die beste Option zu bieten.

Auch andere Geräte der Unterhaltungselektronik nutzen mobile Netze und Hotspots, um die Verbindung mit entfernten Computern aufrechtzuerhalten. Lesegeräte für E-Books können ein gerade erschienenes Buch, die nächste Ausgabe einer Zeitschrift oder die heutige Tageszeitung herunterladen, wo auch immer sie sich befinden. Elektronische Bilderrahmen können ihre Anzeigen auf Zuruf mit neuen Bildern aktualisieren.

Da Mobiltelefone ihren Standort kennen – häufig, weil sie mit **GPS**-Empfängern (*Global Positioning System*) ausgerüstet sind –, können einige Dienste bewusst standortbezogen angeboten werden. Mobile Navigations- und Kartensysteme eignen sich für diese Serviceart offensichtlich besonders, da Ihr GPS-fähiges Telefon bzw. Auto wahrscheinlich eher weiß, wo Sie gerade sind, als Sie selbst. Dasselbe gilt für das Suchen nach dem nächsten Buchladen, dem nächsten chinesischen Restaurant oder die lokale Wettervorhersage. Andere Dienste könnten den Ort aufzeichnen, z.B. indem Fotos und Videos mit dem Ortsnamen der Aufzeichnung beschriftet werden. Diese Beschriftung wird als „Geotagging“ bezeichnet.

Ein Bereich, in dem Mobiltelefone immer mehr genutzt werden, ist **M-Commerce** (*Mobile Commerce*; Senn, 2000). Anstelle von Bargeld oder Kreditkarten werden kurze Textnachrichten vom Mobiltelefon aus verschickt, um die Bezahlung für Speisen aus Verkaufsautomaten, Kinokarten und andere kleine Posten zu autorisieren. Die Belastung erfolgt dann über die Mobiltelefonrechnung. Wenn ein Handy mit **NFC**-Technologie (*Near Field Communication*) ausgestattet ist, kann es als eine RFID-Smartcard fungieren und mit einem nahen Lesegerät zur Bezahlung eingesetzt werden. Die treibenden Kräfte hinter diesem Phänomen sind die Hersteller von mobilen Geräten und die Netzbetreiber, die intensiv an Ideen arbeiten, wie sie ein Stück vom E-Commerce-Kuchen abbekommen können. Vom Händlerstandpunkt aus betrachtet spart dies die Gebühren für die Kreditkartenunternehmen, die sich auf mehrere Prozent belaufen können. Natürlich kann dies aber auch nach hinten losgehen, weil die Kunden in einem Laden die RFID- oder Barcodelesegeräte in ihren mobilen Geräten einsetzen könnten, um vor dem Kauf die Preise der Mitbewerber zu prüfen und einen detaillierten Bericht darüber zu erhalten, wo dieses Produkt in der Nähe noch erworben werden kann und zu welchem Preis.

Ein großer Vorteil von M-Commerce ist, dass Handy-Benutzer gewohnt sind, für alles zu zahlen (im Gegensatz zu Internetnutzern, die erwarten, dass alles kostenlos ist).

Würde eine Website im Internet eine Gebühr erheben, die die Kunden per Kreditkarte zahlen können, gäbe es einen lauten Aufschrei der Benutzer. Wenn dagegen ein Mobilfunkbetreiber seinen Kunden ermöglicht, in Geschäften zu zahlen, indem er mit seinem Telefon an der Kasse winkt, und dann eine zusätzliche Gebühr für diese Annehmlichkeit berechnet, würde dies höchstwahrscheinlich als normal akzeptiert werden. Die Zeit wird es zeigen.

Ohne Zweifel wird mit sinkender Rechnergröße der Einsatz von mobilen und drahtlosen Geräten in der Zukunft rapide wachsen, und dies wahrscheinlich auf Arten, die niemand vorhersehen kann. Lassen Sie uns einen kurzen Blick auf einige der Möglichkeiten werfen. **Sensornetze** sind aus Knoten aufgebaut, die Informationen über den Zustand der physischen Welt sammeln und kabellos weiterübertragen. Diese Knoten können in vertrauten Dingen wie Autos oder Telefonen eingebaut oder kleine, eigenständige Geräte sein. Zum Beispiel könnte Ihr Auto Daten über den Standort, Geschwindigkeit, Vibration und Kraftstoffverbrauch von seinem eingebauten Diagnosesystem sammeln und diese Informationen an eine Datenbank übermitteln (Hull et al., 2006). Diese Daten können beim Auffinden von Schlaglöchern oder Ausweichrouten für überfüllte Straßen helfen und Ihnen mitteilen, wenn Sie – im Vergleich zu anderen Fahrern auf demselben Straßenabschnitt – ein „Spritfresser“ sind.

Sensornetze revolutionieren die Wissenschaft, indem sie eine Fülle von Daten über Verhalten liefern, das vorher nicht beobachtet werden konnte. Ein Beispiel dafür ist das Aufzeichnen der Wanderung von einzelnen Zebras, bei dem ein kleiner Sensor an jedem Tier angebracht wird (Juang et al., 2002). Forscher haben einen drahtlosen Computer in einen Würfel von 1 mm Kantenlänge gepackt (Warneke et al., 2001). Mit derart winzigen Computern können sogar kleine Vögel, Nagetiere und Insekten verfolgt werden (Warneke et al., 2001).

Selbst in ganz banalen Anwendungen wie beispielsweise in Parkuhren können Sensornetze aufgrund neu verfügbarer Daten eine wichtige Rolle spielen. Funkparkuhren können Kredit- oder Geldkarten akzeptieren, wobei die Karte über die Funkverbindung sofort verifiziert wird. Die Parkuhren können außerdem über das kabellose Netz melden, wenn sie gerade benutzt werden. Dann könnten Fahrer die neueste Parkplatzübersicht in ihre Autos herunterladen, sodass diese viel leichter einen freien Parkplatz finden können. Natürlich könnte die Parkuhr auch prüfen, ob ein Auto auf dem Parkplatz steht (durch das Aussenden eines Signals), wenn die Uhr abläuft, und den Ablauf der Parkzeit an das Ordnungsamt melden. Nach Schätzungen könnten die Stadtverwaltungen in den USA alleine hierdurch zusätzlich 10 Milliarden US-Dollar einnehmen (Harte et al., 2000).

Tragbare Computer (*wearable computer*) sind eine weitere vielversprechende Anwendung. Intelligente Uhren mit Radio gehören seit dem ersten Auftauchen im Dick-Tracy-Comic im Jahre 1946 zu unserer Vorstellungswelt – heute können Sie sie kaufen. Andere Geräte könnten implantiert werden, wie Schrittmacher und Insulinpumpen. Einige davon können über ein drahtloses Netz gesteuert werden. Dadurch können Ärzte sie leichter testen und neu einstellen. Es könnte aber auch zu einigen hässlichen Problemen führen, wenn diese Geräte – wie der Standard-PC – unsicher sind und leicht geknackt werden können (Halperin et al., 2008).

1.1.4 Gesellschaftliche Aspekte

Wie die Druckerpresse vor 500 Jahren ermöglichen Rechnernetze dem Normalbürger, Inhalte auf Arten zu verbreiten und anzusehen, die bisher nicht möglich waren. Doch wo Licht ist, ist auch Schatten: Diese neu entdeckte Freiheit wirft viele ungelöste gesellschaftliche, ethische und politische Fragen auf. Wir wollen hier einige kurz anreißen; eine ausführliche Abhandlung würde sicherlich ein dickes Buch füllen.

Soziale Netzwerke, Foren, Content-Sharing-Sites und viele andere Anwendungen erlauben es den Nutzern, ihre Ansichten mit Gleichgesinnten auszutauschen. Solange sich die Themen auf rein technische Gebiete oder Hobbys wie Gartenpflege beschränken, können nicht viele Probleme entstehen.

Der Ärger beginnt, wenn Newsgroups zu Themen eingerichtet werden, die die Gemüter erregen, wie Politik, Religion oder Sex. Die in solchen Gruppen veröffentlichten Ansichten können einige Personen tief verletzen. Schlimmer noch, sie können politisch nicht korrekt sein. Darüber hinaus müssen die Nachrichten nicht unbedingt auf Text beschränkt sein – hochauflösende Farbfotos und Videoclips können leicht über Rechnernetze übertragen werden. Manche Leute vertreten die Ansicht „Leben und leben lassen“, während für andere die globale elektronische Bereitstellung von bestimmten Materialien (z.B. verbale Angriffe auf bestimmte Länder oder Religionen, Pornografie etc.) absolut inakzeptabel ist und censiert werden sollte. Verschiedene Länder haben in diesem Bereich eine unterschiedliche und widersprüchliche Gesetzgebung. Somit erhitzt sich die Debatte.

In der Vergangenheit wurden Klagen gegen Netzbetreiber mit dem Vorwurf eingereicht, sie seien für die Verbreitung des Inhalts über ihre Einrichtungen ebenso verantwortlich wie die Herausgeber von Zeitungen und Zeitschriften. Die unausweichliche Antwort ist, dass ein Netzbetreiber mit einer Telefongesellschaft oder der Post vergleichbar ist und es nicht erwartet werden kann, dass sie die Aussagen ihrer Benutzer überwachen.

Die Tatsache, dass einige Netzbetreiber Inhalte aus eigenen Gründen blockieren, sollte jetzt nur wenig überraschen. So mancher Benutzer von Peer-to-Peer-Anwendungen wurde von seinem Netzdienst abgeschnitten, weil die Netzbetreiber es nicht profitabel fanden, die großen Mengen von Datenverkehr zu übertragen, der mit diesen Anwendungen zusammenhängt. Dieselben Betreiber würden wahrscheinlich gerne verschiedene Unternehmen unterschiedlich behandeln. Falls Sie ein großes Unternehmen sind und gut bezahlen, dann bekommen Sie auch guten Service. Aber wenn Sie ein kleiner Unternehmer sind, werden Sie schlecht bedient. Gegner dieser Praxis fordern, Peer-to-Peer- und andere Inhalte auf die gleiche Weise zu behandeln, da sie für das Netz nichts weiter als Bits sind. Dieses Argument, Kommunikationen nicht über ihren Inhalt oder ihre Quelle oder über denjenigen zu differenzieren, der diesen Inhalt zur Verfügung stellt, ist als **Netzneutralität** (*network neutrality*; Wu, 2003) bekannt. Man kann mit großer Sicherheit sagen, dass diese Debatte noch eine Weile anhalten wird.

Viele andere Parteien sind in das Gerangel über Inhalte einbezogen. Beispielsweise heizen Musik- und Filmpiraterie das ungeheure Wachstum der Peer-to-Peer-Netze an, was die Rechteinhaber nicht erfreut, die mit rechtlichen Schritten drohen (und diese

manchmal auch unternehmen). Es werden jetzt automatisierte Systeme eingesetzt, die Peer-to-Peer-Netze durchsuchen und Warnmeldungen an Netzbetreiber und Nutzer abgeben, wenn diese möglicherweise Urheberrechte verletzen. In den Vereinigten Staaten sind diese Warnungen als **DMCA Takedown Notices** bekannt (nach dem **Digital Millennium Copyright Act**). Dieses Durchsuchen gleicht einem Rüstungswettlauf, weil es schwer ist, zuverlässig Urheberrechtsverletzungen zu fassen. Selbst Ihr Drucker könnte irrtümlich für einen Schuldigen gehalten werden (Piatek et al., 2008).

Rechnernetze vereinfachen die Kommunikation. Sie erleichtern es auch denjenigen, die das Netzwerk zur Verfügung stellen, im Datenverkehr zu schnüffeln. Dies wirft Konflikte zu Themen wie Arbeitnehmerrechte versus Arbeitgeberrechte auf. Viele Leute lesen und verfassen E-Mails während der Arbeit. Nun gibt es Arbeitgeber, die behaupten, das Recht zu haben, die Nachrichten der Mitarbeiter zu lesen und eventuell zu zensieren, selbst die Mails, die nach der Arbeit am PC zu Hause außerhalb der Arbeitszeit versendet wurden. Nicht alle Arbeitnehmer sind damit einverstanden, besonders nicht mit dem letztgenannten Teil.

Bei einem weiteren Konfliktthema geht es um Staatsmacht versus Bürgerrechte. In den USA hat das FBI bei vielen Internetdienstanbietern (*Internet Service Provider, ISP*) ein System installiert, das die gesamte eingehende und ausgehende E-Mail nach für das FBI interessanten Informationsbröckchen ausspioniert. Ein frühes System wurde anfangs „Carnivore“ (Fleischfresser) genannt, aber wegen seines schlechten Rufs in der Öffentlichkeit wurde es in ein unschuldiger klingendes „DCS1000“ umbenannt (Blaze und Bellovin, 2000; Sobel, 2001; Zacks, 2001). Das Ziel solcher Systeme ist es, Millionen von Menschen auszuspionieren – in der Hoffnung, eventuell Informationen über illegale Aktivitäten zu erhalten. Sehr zum Leidwesen für die Spione verbietet die Vierte Ergänzung zur amerikanischen Verfassung staatliche Durchsuchungen ohne eine Durchsuchungsvollmacht, doch die Regierung ignoriert dies häufig.

Natürlich hat der Staat kein Monopol auf die Bedrohung der Privatsphäre. Der Privatsektor tut das Seinige dazu, indem **Profile** von Benutzern angelegt werden. So ermöglichen beispielsweise kleine Dateien, sogenannte **Cookies**, die vom Webbrowser auf dem Rechner eines Benutzers abgelegt werden, den Computerunternehmen die Aktivitäten von Benutzern im Cyberspace zu verfolgen. Cookies könnten außerdem dazu beitragen, dass sich Kreditkartenzahlen, Sozialversicherungsnummern und andere vertrauliche Informationen über das gesamte Internet verbreiten (Berghel, 2001). Unternehmen, die webbasierte Dienste anbieten, sammeln möglicherweise große Mengen von persönlichen Informationen über ihre Nutzer, die es ihnen ermöglichen, die Aktivitäten ihrer Anwender direkt zu beobachten. Zum Beispiel kann Google Ihre E-Mails lesen und dann Werbungen anzeigen, die auf Ihren Interessen beruhen, falls Sie den E-Mail-Dienst **Gmail** von Google benutzen.

Ein neuer Aspekt im Zusammenhang mit mobilen Geräten ist das Problem, Informationen über den Aufenthaltsort zu schützen, das als **Location Privacy** bekannt ist (Beresford und Stajano, 2003). Wenn ein Netzbetreiber Ihnen Dienste für Ihr mobiles Gerät anbietet, erfährt er damit gleichzeitig, wo Sie sich zu unterschiedlichen Tageszeiten

aufhalten. Dadurch ist er in der Lage, all Ihre Bewegungen nachzuvollziehen. Er könnte also wissen, welchen Nachtclub und welches Ärztezentrum Sie besuchen.

Rechnernetze bieten auch das Potenzial, die Privatsphäre durch das Versenden von anonymen Nachrichten zu schützen. In manchen Situationen ist diese Fähigkeit sicherlich wünschenswert. Abgesehen davon, dass Anonymität Unternehmen davon abhält, etwas über Ihre Gewohnheiten zu erfahren, ermöglicht sie es z.B. Studenten, Soldaten, Angestellten und Bürgern bei illegalem Verhalten ihre Professoren, Offiziere, Vorgesetzte und Politiker ohne Furcht vor Repressalien zu verpfeifen. Andererseits gewährt die Rechtsprechung in den USA und den meisten anderen Demokratien einer beschuldigten Person das Recht, die Kläger zu Rede und Antwort vor Gericht zu zitieren, sodass anonyme Anschuldigungen nicht als Beweis zugelassen sind.

Im Internet können zwar sehr schnell Informationen gefunden werden, aber sehr vieles davon ist unbedacht, irreführend oder schlichtweg falsch. Der medizinische Rat zu dem Schmerz in Ihrer Brust, den Sie im Internet aufgelesen haben, kann von einem Nobelpreisträger oder einem Studienabbrecher stammen.

Eine andere Art der Information ist häufig unerwünscht: Elektronische **Junkmail** (Spam) ist Teil unseres täglichen Lebens geworden, weil Spammer Millionen von E-Mail-Adressen gesammelt haben, sodass nun Mächtigern-Marketingprofis computergenerierte Nachrichten billig an diese Adressen schicken können. Die resultierende Spamflut konkurriert mit dem Nachrichtenfluss von echten Personen. Zum Glück gibt es Filtersoftware, die die von anderen Computern erzeugten Spams lesen und löschen kann – mit mehr oder weniger großem Erfolg.

Eine weitere Art von Inhalt leistet kriminellem Verhalten Vorschub. Webseiten und E-Mail-Nachrichten, die aktive Inhalte enthalten (im Grunde Programme oder Makros, die auf dem Rechner des Empfängers ausgeführt werden), können Viren enthalten, die die Kontrolle Ihres Computer übernehmen. Diese Viren können eingesetzt werden, um das Passwort Ihres Bankkontos zu stehlen oder um Ihren Rechner dazu zu bringen, als Teil eines Botnetzes oder Pools von kompromittierten Maschinen Spams zu versenden.

Phishing-Nachrichten geben sich als von einem vertrauenswürdigen Absender stammend – zum Beispiel Ihrer Bank – aus und versuchen Sie dazu zu bringen, sensible Informationen wie Kreditkartenzahlen offenzulegen. Identitätsdiebstahl wird zum ernsthaften Problem, wenn die Diebe ausreichend Informationen über ein Opfer sammeln, um Kreditkarten und andere Dokumente im Namen des Opfers zu erhalten.

Es kann schwierig sein, Computer davon abzuhalten, sich im Internet als eine Person auszugeben. Dieses Problem führte zur Entwicklung von **CAPTCHAs**. Hierbei fordert der Rechner den Anwender zur Lösung einer kleinen Aufgabe auf, zum Beispiel die Buchstaben einzugeben, die in einem verzerrten Bild zu sehen sind. Dieser Test dient als Bestätigung, dass der aktuelle Anwender tatsächlich ein Mensch ist (von Ahn, 2001). Es handelt sich hier um eine Variation des berühmten Turing-Tests, bei dem eine Person Fragen über ein Netzwerk stellt um herauszufinden, ob die antwortende Entität menschlich ist.

Viele dieser Probleme könnten gelöst werden, wenn die Computerindustrie das Thema Computersicherheit ernst nähme. Wenn alle Nachrichten verschlüsselt und authentifiziert würden, wäre der Missbrauch nicht so einfach. Eine solche Technologie ist gut etabliert und wird in *Kapitel 8* genauer erläutert. Das Problem ist, dass Hardware- und Softwareanbieter wissen, dass die Integration einer Sicherheitsfunktion Geld kostet und die Kunden diese Funktion nicht fordern. Darüber hinaus werden sehr viele Probleme durch fehlerhafte Software verursacht. Dies ist darauf zurückzuführen, dass Anbieter immer mehr Funktionen in ihre Programme aufnehmen, was notwendigerweise mehr Code und auch mehr Fehler (Bugs) bedeutet. Abhilfe könnte hier eine Steuer auf neue Funktionen schaffen, doch dies ließe sich vermutlich einigen Lagern nur schwer vermitteln. Rückzahlungen für fehlerhafte Software wären eine nette Sache – leider würde die gesamte Softwareindustrie innerhalb eines Jahres daran bankrott gehen.

Rechnernetze lassen neue rechtliche Probleme entstehen, wenn sie auf alte Gesetze treffen. Elektronische Glücksspiele sind dafür ein Beispiel. Seit Jahrzehnten simulieren Computer viele Dinge – warum sollten sie dann nicht Münzspieler, Rouletteräder, Blackjack-Geber und andere Glücksspielkomponenten simulieren? Nun, weil Glücksspiele in vielen Ländern verboten sind. Das Dumme ist nur: Glücksspiele sind in vielen anderen Ländern (zum Beispiel in England) erlaubt und Casinobetreiber haben die Chance ergriffen, Glücksspiele im Internet anzubieten. Was passiert nun, wenn der Spieler, das Casino und der Server sich alle in verschiedenen Ländern mit sich widersprechenden Gesetzen befinden? Gute Frage.

1.2 Netzhardware

Nun ist es an der Zeit, unsere Aufmerksamkeit von den Anwendungen und den sozialen Aspekten der Netztechnologie (dem Dessert) auf die technischen Aspekte (den Spinat) zu verlagern. Es gibt keine allgemeingültige Systematik, in die alle Rechnernetze passen. Man kann aber zwei wichtige Klassifikationskategorien unterscheiden: Übertragungstechnik und Ausdehnung. Diese beiden werden in den folgenden Abschnitten beschrieben.

Allgemein betrachtet, gibt es zwei Arten von gängigen Übertragungstechniken: **Broadcast**- und **Punkt-zu-Punkt**-Verbindungen (*point-to-point*).

Punkt-zu-Punkt-Verbindungen vernetzen einzelne Paare von Rechnern miteinander. Um in einem Netzwerk, das aus Punkt-zu-Punkt-Verbindungen aufgebaut ist, von der Quelle ans Ziel zu gelangen, muss eine kurze Nachricht (in bestimmten Zusammenhängen **Paket** genannt) eventuell zuvor einen oder mehrere dazwischenliegende Rechner kontaktieren. Meist sind mehrere Routen mit unterschiedlicher Länge möglich, sodass bei Punkt-zu-Punkt-Netzwerken die Ermittlung einer guten Route entscheidend ist. Die Punkt-zu-Punkt-Übertragung von genau einem Sender zu genau einem Empfänger nennt man **Unicasting**.

Im Gegensatz dazu wird in einem Broadcast-Netz ein Kommunikationskanal von allen am Netz angeschlossenen Maschinen gemeinsam genutzt. Pakete werden von einer

Maschine versendet und von allen anderen empfangen. Ein Adressfeld in jedem Paket gibt den Empfänger an. Beim Empfang eines Pakets prüft ein Rechner das Adressfeld. Ist das Paket für ihn bestimmt, verarbeitet der Rechner es. Ist es für einen anderen Rechner bestimmt, wird es einfach ignoriert.

Ein drahtloses Netzwerk ist ein bekanntes Beispiel für eine Broadcast-Verbindung, bei der die Kommunikation über einen Abdeckungsbereich läuft, welcher von dem drahtlosen Kanal und der übertragenden Maschine abhängt. Als Analogie stelle man sich jemanden vor, der in einem Besprechungszimmer steht und ruft: „Watson, komm hierher, ich brauche dich.“ Auch wenn das Paket von vielen empfangen (gehört) wird, reagiert nur Watson. Die anderen ignorieren es einfach.

Broadcast-Systeme unterstützen im Allgemeinen auch die Möglichkeit, ein Paket an *alle* Ziele zu richten, indem im Adressfeld ein spezieller Code verwendet wird. Wird ein Paket mit diesem Code übertragen, nimmt es jeder Rechner im Netz entgegen und verarbeitet es. Diese Betriebsart nennt man **Broadcasting**. Einige Broadcast-Systeme unterstützen auch die Übertragung an eine Teilmenge der angeschlossenen Rechner. Das nennt man **Multicasting**.

Ein alternatives Kriterium zur Klassifizierung von Netzen ist deren Ausdehnung. Entfernung ist als Klassifizierungsgröße wichtig, weil für die unterschiedlichen Ausdehnungen verschiedene Techniken angewendet werden.

In ►Abbildung 1.6 werden mehrere Rechnersysteme grob nach der physikalischen Größe klassifiziert. Ganz oben finden sich persönliche Netze (*Personal Area Network*, PAN), die sich auf Einzelpersonen beziehen. Darunter kommen Netze mit weiterer Ausdehnung. Diese werden in lokale Netze (*Local Area Network*, LAN), Stadtnetze (*Metropolitan Area Network*, MAN) und Fernnetze (*Wide Area Network*, WAN) unterteilt, jeweils wieder nach steigender Ausdehnung sortiert. Werden schließlich zwei oder mehr Netze miteinander verbunden, spricht man von einem Internetwork. Das weltweite Internet ist sicherlich das bekannteste (aber nicht das einzige) Beispiel für ein Internetwork. Bald werden wir noch größere Internetworks mit dem **interplanetarischen Internet** (*Interplanetary Internet*) bekommen, das Netze im Weltall verbindet (Burleigh et al., 2003).

In diesem Buch werden wir Netze mit all den aufgeführten Ausdehnungen behandeln. In den folgenden Abschnitten geben wir zunächst eine kurze Einführung in die Netzhardware, geordnet nach Ausdehnungsgröße.

| Entfernung der Prozessoren | Prozessoren im gleichen ... | Beispiel |
|----------------------------|-----------------------------|----------|
| 1 m | Quadratmeter | PAN |
| 10 m | Raum | LAN |
| 100 m | Gebäude | |
| 1 km | Campus | MAN |
| 10 km | Stadt | |
| 100 km | Land | WAN |
| 1000 km | Kontinent | |
| 10.000 km | Planet | Internet |

Abbildung 1.6: Klassifizierung von Netzen nach ihrer Ausdehnung.

1.2.1 Personal Area Network

Mithilfe von **PANs** (*Personal Area Network*) können Geräte über die Reichweite einer Person kommunizieren. Ein bekanntes Beispiel ist ein kabelloses Netz, das einen Rechner mit seinen Peripheriegeräten verbindet. Jeder Computer verfügt in der Regel über einen Bildschirm, eine Tastatur, eine Maus und einen Drucker. Wenn diese Verbindungen nicht drahtlos sind, werden hierfür Kabel benötigt. Viele Einsteiger tun sich sehr schwer, die richtigen Kabel zu finden und sie in die richtigen kleinen Löcher zu stecken (selbst wenn diese normalerweise farbig gekennzeichnet sind). Daher bieten die Computerhersteller meist die Option an, einen Techniker nach Hause kommen zu lassen, der die Installation vornimmt. Um diesen Nutzern anderweitig zu helfen, haben sich einige Unternehmen zusammengetan und ein Kurzstreckenfunknetz namens **Bluetooth** entwickelt, sodass diese Komponenten ohne Kabel angeschlossen werden können. Wenn Ihre Geräte also mit Bluetooth ausgestattet sind, dann brauchen Sie keine Kabel. Einfach nur Aufstellen, Einschalten und alles funktioniert. Dieses vereinfachte Verfahren ist für viele ein großes Plus.

In der einfachsten Form verwendet ein Bluetooth-Netz das Master-Slave-Paradigma (►Abbildung 1.7). Die Systemeinheit (der PC) ist in der Regel der Master, der mit der Maus, der Tastatur etc. als den Slaves kommuniziert. Der Master gibt den Slaves an, welche Adressen verwendet werden, wann sie senden können, wie lange sie übertragen können, welche Frequenzen sie verwenden können usw.

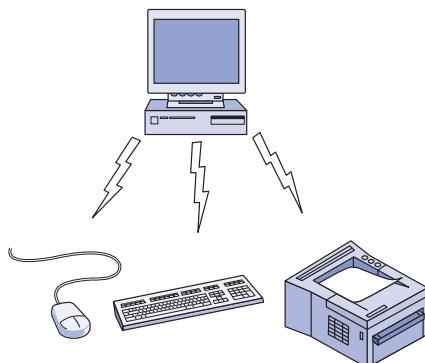


Abbildung 1.7: Bluetooth-PAN-Konfiguration.

Bluetooth kann auch in anderen Umgebungen eingesetzt werden. Es wird häufig benutzt, um ein Headset schnurlos mit einem Mobiltelefon zu verbinden, und es kann Ihren digitalen Musikplayer befähigen, sich mit Ihrem Auto zu verbinden, sobald sich dieses innerhalb der Reichweite befindet. Ein ganz andere Art von PAN entsteht, wenn ein eingebettetes medizinisches Gerät wie ein Schrittmacher, eine Insulinpumpe oder ein Hörgerät mit einer benutzerbedienten Fernsteuerung spricht. Bluetooth wird ausführlich in *Kapitel 4* behandelt.

PANs können außerdem mit anderen Technologien eingesetzt werden, die über kurze Distanz kommunizieren, wie RFID auf Smartcards und in Bibliotheksbüchern. Wir betrachten RFID in *Kapitel 4*.

1.2.2 Lokale Netze (LANs)

Der nächste Schritt sind **lokale Netze** (*Local Area Network, LAN*). Ein LAN ist ein privates Netz, das innerhalb oder in der Nähe eines einzelnen Gebäudes wie ein Privathaus, Büro oder eine Fabrik arbeitet. LANs werden zur Verbindung von PCs und Unterhaltungselektronik benutzt, um Informationen auszutauschen und Betriebsmittel (z.B. Drucker) gemeinsam zu nutzen. Wenn LANs in einer Firma eingesetzt werden, nennt man sie **Unternehmensnetzwerke** (*enterprise network*).

Drahtlose LANs sind heutzutage sehr beliebt, besonders in Privathaushalten, älteren Bürogebäuden, Cafeterias und anderen Orten, wo es zu aufwendig wäre, Kabel zu verlegen. Bei diesen Systemen hat jeder Computer ein Funkmodem und eine Antenne, um mit anderen Rechnern zu kommunizieren. In den meisten Fällen kommuniziert jeder Computer mit einem Gerät, das an der Wand angebracht ist (►Abbildung 1.8a). Dieses Gerät, genannt **Zugangspunkt** (*Access Point, AP*) oder **kabelloser Router** oder **Basisstation**, übermittelt Pakete zwischen kabellosen Computern und ebenso zwischen diesen und dem Internet. Mit dem Zugangspunkt ist es wie beim beliebtesten Kind in der Schule, mit dem jeder sprechen möchte. Wenn andere Rechner aber nahe genug beieinanderstehen, können sie auch direkt miteinander in einer Peer-to-Peer-Konfiguration kommunizieren.

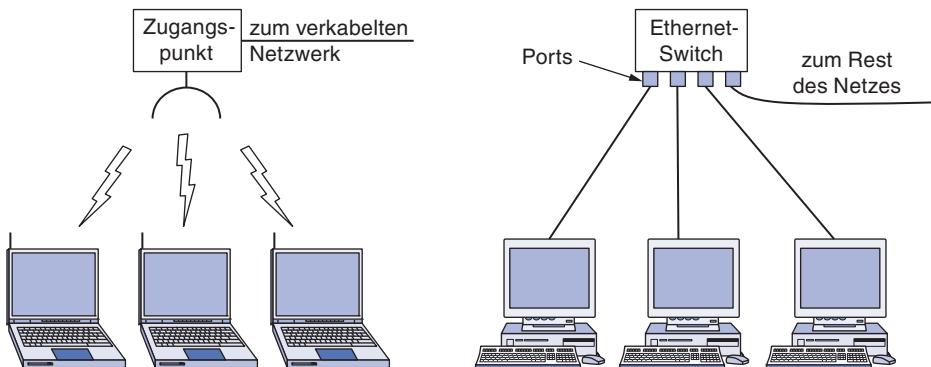


Abbildung 1.8: Drahtlose und verkabelte LANs. (a) IEEE 802.11. (b) Switched Ethernet.

Für kabellose LANs gibt es einen Standard, **IEEE 802.11**, allgemein bekannt als **WiFi**, der heute sehr verbreitet ist. Er arbeitet mit Geschwindigkeiten irgendwo zwischen 11 und Hunderten von Mbit/s. (In diesem Buch wollen wir die Tradition wahren und messen die Übertragungsrate in Mbit/s, wobei 1 Mbit/s 1 000 000 Bit/s sind, und Gbit/s, wobei 1 Gbit/s 1 000 000 000 Bit/s sind.)

Verkabelte LANs benutzen eine Reihe verschiedener Übertragungstechniken. Die meisten setzen Kupferkabel ein, aber einige verwenden Glasfaserkabel. LANs sind hinsichtlich der Größe begrenzt, was bedeutet, dass die ungünstigste Übertragungszeit nach oben beschränkt und im Voraus bekannt ist. Die Kenntnis dieser Grenze hilft bei der Aufgabe, Netzprotokolle zu entwerfen. Typischerweise arbeiten LANs mit Geschwindigkeiten zwischen 100 Mbit/s und 1 Gbit/s, haben eine geringe Übertragungsverzögerung (Mikro- oder Nanosekunden) und machen sehr wenige Fehler. Neuere LANs können mit bis zu 10 Gbit/s arbeiten. Verglichen mit Funknetzen übertreffen verkabelte LANs diese in allen Dimensionen der Performanz. Es ist eben einfacher, Signale über ein Kabel oder durch eine Faser als durch die Luft zu schicken.

Die Topologie von vielen verkabelten LANs wird mit Punkt-zu-Punkt-Verbindungen aufgebaut. IEEE 802.3, bekannt als **Ethernet**, ist die bekannteste LAN-Art. ►Abbildung 1.8b zeigt eine Beispieltopologie von **Switched Ethernet**. Jeder Computer spricht das Ethernet-Protokoll und verbindet sich mit einer Box, die **Switch** (Schalter) genannt wird, über eine Punkt-zu-Punkt-Verbindung. Daher der Name. Ein Switch hat mehrere **Ports**, jeder davon kann sich mit einem Rechner verbinden. Die Aufgabe des Switch ist es, Pakete zwischen den angeschlossenen Computern weiterzuleiten, indem die Adressen in jedem Paket benutzt werden, um zu entscheiden, an welchen Rechner es gesendet werden soll.

Um größere LANs aufzubauen, können Switches über ihre Ports ineinander gesteckt werden. Was passiert aber, wenn man auf diese Art eine Schleife zusammensteckt? Wird das Netz dann immer noch arbeiten? Glücklicherweise haben die Entwickler diesen Fall bedacht. Es ist die Aufgabe des Protokolls herauszufinden, auf welchen Pfaden die Pakete reisen sollten, um sicher den Zielcomputer zu erreichen. Wie dies funktioniert, sehen wir in *Kapitel 4*.

Es ist auch möglich, ein großes physisches LAN in zwei kleinere logische LANs aufzuteilen. Sie wundern sich vielleicht, warum dies sinnvoll sein sollte. Manchmal passt die Anordnung der Netzwerkausstattung nicht zur Struktur der Organisation. Zum Beispiel benutzen die Rechner der Technik- und der Finanzabteilung einer Firma das gleiche physische LAN, weil sie im selben Teil des Gebäudes untergebracht sind, doch die Systemverwaltung könnte einfacher sein, wenn Technik und Finanzen logisch jeweils ihr eigenes Netz hätten. Dies führt zu **virtuellen LANs (VLAN, Virtual LAN)**. Bei diesem Entwurf ist jeder Port mit einer „Farbe“ gekennzeichnet, zum Beispiel „grün“ für Technik und „rot“ für Finanzen. Der Switch leitet die Pakete dann so weiter, dass Computer an den grünen Ports getrennt von den Rechnern an den roten Ports behandelt werden. Broadcast-Pakete, die an einen roten Port gesendet wurden, werden an den grünen Ports nicht empfangen, genau so, als wären es zwei verschiedene LANs. Wir behandeln VLANs am Ende von *Kapitel 4*.

Es gibt noch andere verkabelte LAN-Topologien. In der Tat ist Switched Ethernet eine moderne Version des originalen Ethernet-Entwurfs, bei dem alle Pakete über ein einziges lineares Kabel geschickt werden. Höchstens eine Maschine konnte jeweils erfolgreich übermitteln und ein verteilter Mechanismus wurde zur Schlichtung eingesetzt, um Konflikte zu lösen. Dazu wurde ein einfacher Algorithmus benutzt: Computer können nichts senden, wenn das Kabel belegt ist. Wenn zwei oder mehr Pakete kollidieren, dann wartet jeder Rechner eine zufällig festgelegte Zeitspanne und versucht es später noch einmal. Diese Version bezeichnen wir aus Gründen der Eindeutigkeit als **klassisches Ethernet** und – wie Sie sicher schon erraten haben – erfahren Sie mehr darüber in *Kapitel 4*.

Sowohl kabellose als auch verkabelte Broadcast-Netze können in Netze mit statischem und dynamischem Entwurf, je nach Art der Kanalzuteilung, unterteilt werden. Typischerweise wird für die statische Zuweisung die Zeit in periodisch wiederkehrende, diskrete Intervalle eingeteilt und ein Round-Robin-Algorithmus verwendet, der jeder Maschine eine Übertragung immer nur innerhalb ihrer Zeitscheibe erlaubt. Bei der statischen Zuweisung wird Kanalkapazität verschwendet, wenn ein Rechner innerhalb der ihm zugeordneten Zeitscheibe nichts zu übertragen hat. Daher versuchen die meisten Systeme, einen Kanal dynamisch (d.h. bei Bedarf) zuzuweisen.

Die dynamischen Kanalzuweisungsmethoden sind entweder zentral oder dezentral. Bei der zentralen Kanalzuweisungsmethode gibt es eine einzelne Einheit, beispielsweise die Basisstation in Funknetzen, die bestimmt, wer als Nächstes an der Reihe ist. Diese kann mehrere Pakete entgegennehmen und nach einem internen Algorithmus diesen Prioritäten zuweisen. Bei der dezentralen Kanalzuteilungsmethode gibt es keine zentrale Einheit. Jeder Rechner muss selbst entscheiden, ob er übertragen kann. Wer nun annimmt, dass dieser Ansatz zwangsläufig zu Chaos führen muss, irrt sich. Später werden noch mehrere Algorithmen untersucht, die entwickelt wurden, um Ordnung in dieses potenzielle Chaos zu bringen.

Es lohnt sich, noch ein wenig Zeit in die Diskussion von LANs im Privathaushalt zu investieren. In der Zukunft werden wahrscheinlich alle Haushaltsgeräte in einem privaten Haushalt miteinander kommunizieren können und auf alle Geräte kann über

das Internet zugegriffen werden. Diese Entwicklung ist vermutlich eines dieser visionären Konzepte, nach denen niemand konkret verlangt hat (wie TV-Fernbedienungen oder Mobiltelefone), aber wenn es sie erst einmal gibt, kann man sich ein Leben ohne sie nicht mehr vorstellen.

Viele Geräte können bereits an ein Netz angeschlossen werden. Zu diesen Geräten gehören Computer, Unterhaltungsgeräte (Fernseher, DVD-Spieler etc.) und andere Unterhaltungselektronik (wie Kameras), Haushaltsgeräte (z.B. Radiowecker) sowie Infrastruktur (Verbrauchszähler und Thermostate). Zum Beispiel hat der Durchschnittshaushalt wahrscheinlich ein Dutzend Uhren (u.a. in Haushaltsgeräten), die alle zweimal pro Jahr automatisch auf Sommer- bzw. Winterzeit umgestellt werden könnten, wenn die Uhren mit dem Internet verbunden wären. Die Fernüberwachung eines Hauses wird aller Voraussicht nach zu den Gewinnern gehören, da viele erwachsene Kinder gerne etwas Geld investieren, um ihren älter werdenden Eltern zu helfen, sicher in ihren eigenen Häusern zu leben.

Auch wenn wir uns ein Heimnetz lediglich als weiteres LAN vorstellen, unterscheidet es sich wahrscheinlich grundlegend von anderen Netztypen. Als Erstes müssen die vernetzten Geräte sehr einfach zu installieren sein. Kabellose Router sind das Produkt unter den Elektronikgeräten, das am häufigsten zurückgebracht wird. Die Leute kaufen einen Router, weil sie sich zu Hause ein Funknetz einrichten möchten, und stellen fest, dass es nicht ohne weiteres Zutun funktioniert. Dann bringen sie es lieber zurück, anstatt länger der Warteschleifenmusik in der technischen Hotline zuzuhören.

Zweitens müssen das Netz und die Geräte idiotensicher in der Anwendung sein. Klimaanlagen haben in der Regel immer einen Schalter mit vier Einstellungen: AUS, NIEDRIG, MITTEL und HOCH. Das Handbuch dazu hat heute über 30 Seiten. Wenn die Klimaanlagen einmal vernetzt werden können, wird alleine das Kapitel über die Sicherheit 30 Seiten lang sein. Dies ist ein Problem, denn nur Computerbenutzer sind daran gewöhnt, mit nicht funktionierenden Produkten umzugehen – die Käufer von Autos, Fernsehern oder Kühlschränken sind weit weniger tolerant. Sie erwarten, dass die Produkte hundertprozentig funktionieren – ohne dafür einen Computerfreak anheuern zu müssen.

Drittens ist der niedrige Preis ausschlaggebend für den Erfolg. Niemand wird 50 Euro für ein Internetthermometer zahlen, da es nur für wenige Leute wichtig ist, die heimische Raumtemperatur von der Arbeit aus zu überwachen. Bei 5 Euro sieht die Sache aber schon wieder ganz anders aus.

Viertens muss es möglich sein, mit ein oder zwei Geräten zu beginnen und das Netz dann schrittweise zu erweitern. Das heißt: keine Formatkriege. Wenn Verbrauchern empfohlen wird, Peripheriegeräte mit IEEE-1394-Schnittstellen (FireWire) zu kaufen, und ein paar Jahre später dies dann widerrufen und USB 2.0 als Schnittstelle des Monats deklariert wird, um dann umzusteigen auf IEEE 802.11g – ups, nein, machen Sie IEEE 802.11n daraus – ich meine, IEEE 802.16 (unterschiedliche Funknetze) –, dann verunsichert dies die Verbraucher sehr. Die Netzschaltung muss über Jahrzehnte stabil bleiben, wie die Standards bei der Fernsehübertragung.

Fünftens werden Sicherheit und Zuverlässigkeit enorme Bedeutung erlangen. Der Verlust von einigen Dateien aufgrund eines E-Mail-Virus ist eine Sache, eine andere Sache ist es, wenn ein Einbrecher über seinen mobilen Computer das Sicherheitssystem außer Kraft setzt und Ihr Haus ausraubt.

Eine interessante Frage ist, ob Heimnetze verkabelt oder drahtlos sein werden. Bequemlichkeit und Kosten sprechen für drahtlose Netze, weil dann keine Kabel anzubringen oder – was noch schlimmer wäre – umzurüsten sind. Die Sicherheit spricht für Kabelnetze, weil die Funkwellen der kabellosen Netze sehr gut darin sind, Wände zu durchdringen. Nicht jeder ist hocherfreut bei dem Gedanken, dass die Nachbarn bei ihm munter mitsurfen und vielleicht sogar seine E-Mails lesen. In *Kapitel 8* befassen wir uns näher mit Verschlüsselungstechniken, um die Sicherheit zu gewährleisten, doch für unerfahrene Benutzer ist das leichter gesagt als getan.

Eine dritte Option, die möglicherweise reizvoll ist, ist die Wiederverwendung von Netzen, die bereits im Haus vorhanden sind. Dafür kommen in erster Linie die elektrischen Leitungen infrage, die im gesamten Haus installiert sind. Mithilfe von **Trägerfrequenzanlagen** (*power-line network*) können Geräte, die mit Steckdosen verbunden sind, Informationen durch das ganze Haus schicken. Man muss das Fernsehgerät sowieso an das Stromnetz anschließen und auf diese Art hat man gleichzeitig Internetzugang. Die Schwierigkeit hierbei ist, sowohl Strom als auch Datensignale zur gleichen Zeit zu transportieren. Teilweise kann dies durch die Verwendung von unterschiedlichen Frequenzbändern gelöst werden.

Kurzum: LAN-Heimnetze bieten viele Möglichkeiten und Herausforderungen. Die meisten der Herausforderungen beziehen sich auf die Anforderung an das Netz, das leicht verwaltbar, verlässlich und sicher sein soll, vor allem in den Händen von technisch nicht bewanderten Anwendern, ebenso wie auf niedrige Kosten.

1.2.3 Stadtnetze (MANs)

Ein **Stadtnetz** (*Metropolitan Area Network, MAN*) erstreckt sich über ein Gebiet von der Größe einer Stadt. Das bekannteste Beispiel von MANs ist das Kabelfernsehnetz in vielen Städten. Dieses System ist aus den früheren Gemeinschaftsantennensystemen entstanden, die in Gegenden mit schlechtem Fernsehempfang installiert wurden. Bei diesen frühen Systemen wurde eine große Antenne auf dem nächsten Hügel installiert und das Signal dann zu den Häusern der Teilnehmer übertragen.

Zuerst waren dies lokal konzipierte Ad-hoc-Systeme. Dann begannen Unternehmen, das Geschäft für sich zu entdecken. Sie schlossen mit den lokalen Verwaltungen Verträge und verkabelten ganze Städte. Der nächste Schritt war die Erstellung von speziellen Fernsehprogrammen und dann der Aufbau reiner Kabelkanäle. Oftmals sind diese Kabelkanäle thematisch sehr spezialisiert, wie Nachrichten, Sport, Kochen, Gartenpflege etc. Aber von der Einführung bis Ende der 1990er Jahre waren sie nur für den Fernsehempfang gedacht.

Als das Internet für ein Massenpublikum attraktiv wurde, stellten die TV-Netzbetreiber fest, dass sie mit ein paar Systemänderungen einen Zweiwege-Internetdienst in noch

nicht genutzten Frequenzbereichen anbieten können. An diesem Punkt begann sich das Kabelfernsehensystem von einem einfachen Fernsehnetz hin zu einem Stadtnetz zu verwandeln. Auf den ersten Blick sieht ein Stadtnetz in etwa wie in ▶ Abbildung 1.9 aus. In dieser Abbildung werden Fernsehsignale und Internetdaten zu einer zentralen **Kabelkopfstelle** (*cable headend*) gesendet, die diese dann an die Privathaushalte verteilt. Dieses Thema werden wir in *Kapitel 2* noch ausführlicher behandeln.

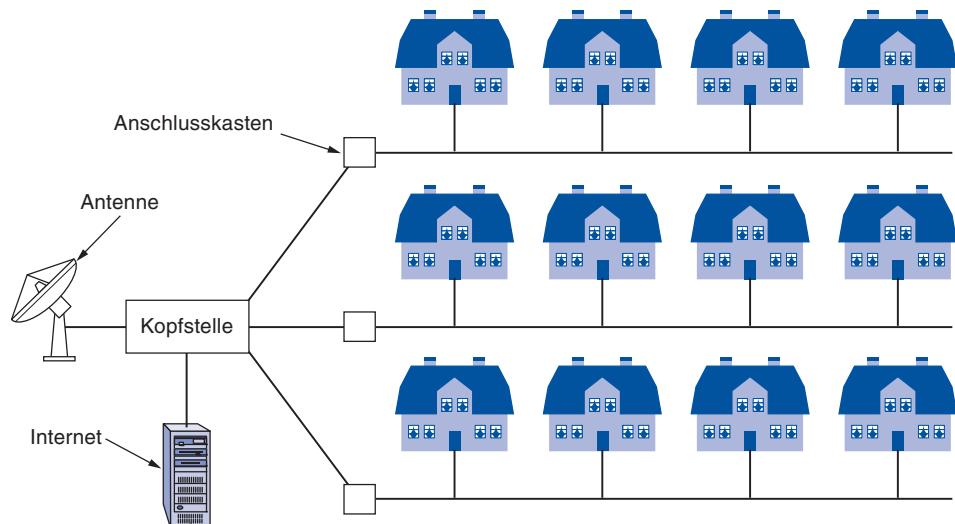


Abbildung 1.9: Ein MAN, das auf Kabelfernsehen basiert.

Kabelfernsehen ist jedoch nicht das einzige MAN. Neuere Entwicklungen im drahtlosen Internet-Hochgeschwindigkeitszugang führten zu einem weiteren Stadtnetz, das als IEEE 802.16 standardisiert wurde und als **WiMAX** bekannt ist. Wir werden in *Kapitel 4* einen Blick darauf werfen.

1.2.4 Fernnetze (WANs)

Ein **Fernnetz** (*Wide Area Network, WAN*) erstreckt sich über ein großes geografisches Gebiet, meist ein Land oder einen Kontinent. Wir beginnen unsere Darstellung mit verkabelten WANs, dazu benutzen wir das Beispiel eines Unternehmens mit Zweigstellenbüros in verschiedenen Städten.

Das WAN in ▶ Abbildung 1.10 ist ein Netzwerk, das Büros in Perth, Melbourne und Brisbane verbindet. Jedes dieser Büros besitzt Computer, auf denen Benutzerprogramme (Anwendungen) ausgeführt werden. Wir halten uns an die Konvention und nennen diese Maschinen **Hosts**. Der Rest des Netzes, das diese Hosts verbindet, wird dann **Kommunikationssubnetz** (*communication subnet*) oder einfach nur **Subnetz** genannt. Die Aufgabe eines Subnetzes ist es, Nachrichten von Host zu Host zu übertragen, so wie das Telefonsystem gesprochene Wörter (eigentlich nur Geräusche) vom sprechenden zum zuhörenden Teilnehmer überträgt.

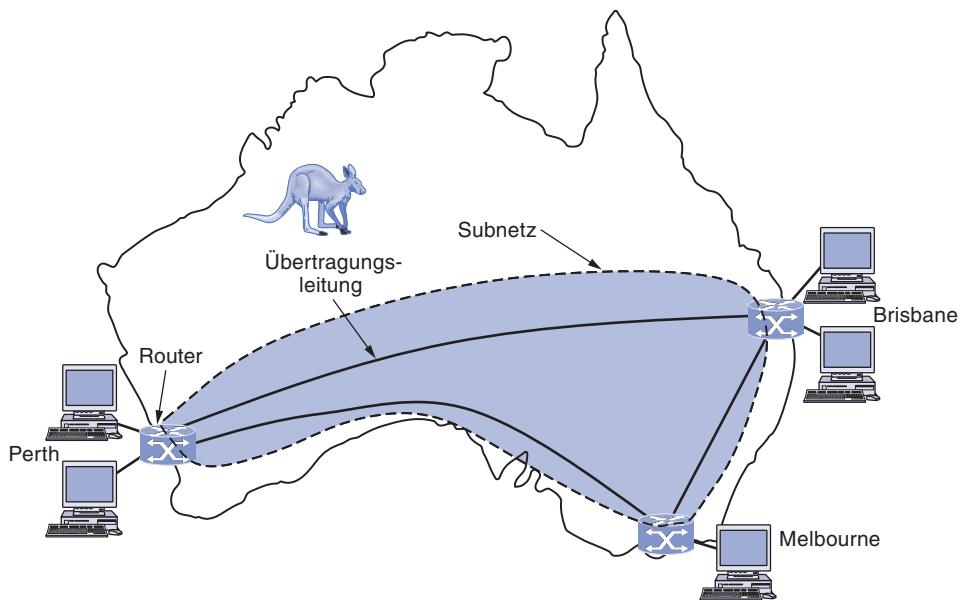


Abbildung 1.10: WAN, das drei Zweigstellenbüros in Australien miteinander verbindet.

In den meisten WANs besteht das Subnetz aus zwei getrennten Komponenten: Übertragungsleitungen und Vermittlungselementen. **Übertragungsleitungen** (*transmission line*) übertragen die Bits zwischen den Rechnern. Die Leitungen können Kupferdrähte oder Glasfasern oder sogar Funkverbindungen sein. Die meisten Unternehmen haben keine eigenen Übertragungsleitungen, also mieten sie diese stattdessen von einer Telefongesellschaft. **Vermittlungselemente** (*switching element*) sind spezielle Computer, die zwei oder mehr Übertragungsleitungen verbinden. Wenn Daten auf einer Eingangsleitung ankommen, muss ein Vermittlungselement eine Ausgangsleitung wählen, auf der die Daten weiterzuleiten sind. Diese Vermittlungsrechner hatten in der Vergangenheit verschiedene Namen, wobei heute die Bezeichnung **Router** am gebräuchlichsten ist. Unglücklicherweise sprechen dies manche Leute wie „Ruter“, andere wie „Rauter“ aus. Die richtige Aussprache herauszufinden, bleibt dem Leser als Übung überlassen. (Hinweis: Die als richtig empfundene Antwort hängt unter Umständen davon ab, in welcher Region Sie wohnen.)

Ein kurzer Kommentar über den Begriff „Subnetz“ ist hier angebracht. Ursprünglich war seine *einige* Bedeutung die Ansammlung von Routern und Kommunikationsleitungen, die Pakete vom Quell-Host zum Ziel-Host bewegen haben. Die Leser sollten sich darüber bewusst sein, dass eine zweite, jüngere Bedeutung im Zusammenhang mit Netzadressierung hinzugekommen ist. Wir werden auf diese Bedeutung in *Kapitel 5* stoßen und bleiben bis dahin bei der Originalbedeutung (nämlich eine Zusammenstellung von Leitungen und Routern).

Das WAN, wie wir es beschrieben haben, sieht einem großen, verkabelten LAN sehr ähnlich, doch es gibt einige wichtige Unterschiede, die jenseits der Drähte liegen. Bei einem WAN gehören die Hosts und das Subnetz normalerweise unterschiedlichen

Personen und werden von unterschiedlichen Personen bedient. In unserem Beispiel könnten die Arbeitnehmer für ihre eigenen Rechner verantwortlich sein, während die IT-Abteilung des Unternehmens die Aufsicht für den Rest des Netzes innehat. In den nächsten Beispielen werden wir klarere Grenzen sehen, in denen die Netzbetreiber oder Telefongesellschaften das Subnetz betreiben. Die Trennung der reinen Kommunikationsaspekte des Netzwerks (das Subnetz) von den Anwendungsspekten (den Hosts) vereinfacht den übergreifenden Netzentwurf.

Ein zweiter Unterschied ist, dass die Router in der Regel unterschiedliche Arten von Netzwerktechnologie miteinander verbinden. Das Netz innerhalb der Büros könnte beispielsweise Switched Ethernet sein, während die Übertragungsleitungen für größere Entfernungen SONET-Verbindungen sind (welche wir in *Kapitel 2* behandeln werden). Es muss also irgendeine Vorrichtung geben, die diese zusammenfügt. Der scharfsinnige Leser wird bemerken, dass dies über unsere Definition eines Netzwerks hinausgeht. Das heißt, viele WANs wären tatsächlich eher **Internetworks** oder zusammengesetzte Netze, die aus mehr als einem Netzwerk aufgebaut sind. Auf Internetworks werden wir im nächsten Abschnitt noch ausführlicher eingehen.

Ein letzter Unterschied liegt darin, was mit dem Subnetz verbunden wird. Dies können einzelne Rechner, wie es bei LANs der Fall war, oder vollständige LANs sein. Auf diese Weise werden größere Netzwerke aus kleineren aufgebaut. Für das Subnetz macht dies keinen Unterschied, es erledigt immer dieselbe Aufgabe.

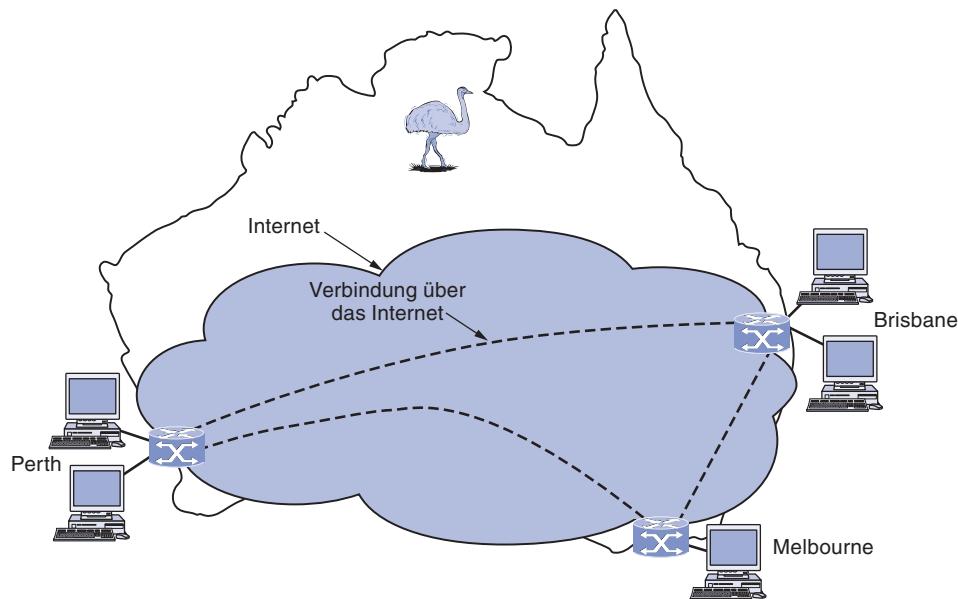


Abbildung 1.11: WAN, das ein virtuelles privates Netzwerk benutzt.

Wir sind nun in der Lage, uns zwei Varianten von WANs anzusehen. Bei der ersten verbindet ein Unternehmen seine Büros mit dem Internet, statt spezielle Übertragungsleitungen zu mieten. Dadurch können Verbindungen zwischen den Büros als

virtuelle Verbindungen hergestellt werden, die das Internet als Basis benutzen. Dieses Arrangement, das in ►Abbildung 1.11 gezeigt ist, wird **virtuelles privates Netzwerk (VPN, Virtual Private Network)** genannt. Im Vergleich mit den speziell dafür vorgesehenen Einrichtungen hat ein VPN den üblichen Vorteil der Virtualisierung, nämlich die flexible Wiederverwendung einer Ressource (Internetverbindung). Denken Sie daran, wie einfach es beispielsweise ist, ein viertes Büro hinzuzufügen. Ein VPN hat auch die üblichen Nachteile der Virtualisierung: Mangel an Kontrolle über die zugrunde liegenden Ressourcen. Bei einer Standleitungen wissen Sie, über welche Kapazität Sie verfügen können. Bei einem VPN können Ihre gewonnenen Vorteile je nach verwendetem Internetdienst variieren.

Die zweite Variante ist, dass das Subnetz von einem anderen Unternehmen betrieben wird. Dieser Subnetzbetreiber ist als **Netzdienstanbieter** (*network service provider*) bekannt und die Büros sind seine Kunden. Diese Struktur ist in ►Abbildung 1.12 gezeigt. Der Subnetzbetreiber wird auch Verbindungen zu anderen Kunden bereitstellen, solange diese bezahlen können. Der Netzdienst wäre recht unbefriedigend, wenn die Kunden sich nur gegenseitig Pakete schicken könnten, daher wird der Subnetzbetreiber auch Verbindungen zu anderen Netzwerken des Internets herstellen. Solch ein Subnetzbetreiber heißt **Internetdienstanbieter** (*Internet Service Provider, ISP*) und das Subnetz ist ein **ISP-Netz**. Die mit dem ISP verbundenen Kunden erhalten Internetdienst.

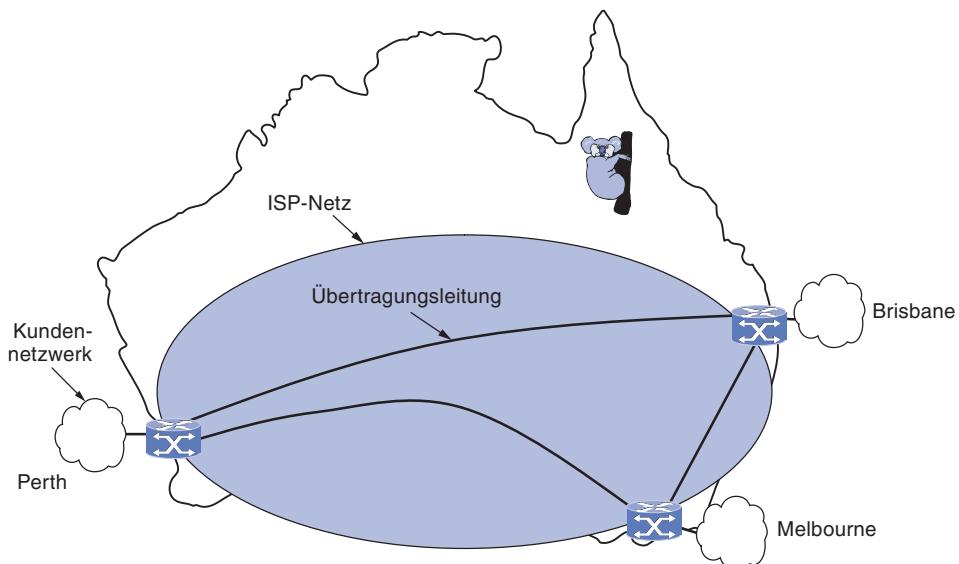


Abbildung 1.12: WAN, das ein ISP-Netz benutzt.

Wir können das ISP-Netz verwenden, um eine Vorschau auf einige der Schlüssel- aspekte zu geben, die wir in späteren Kapiteln noch untersuchen. In den meisten WANs enthält das Netz zahlreiche Übertragungsleitungen, die jeweils ein Router-Paar verbinden. Möchten zwei Router, die nicht die gleiche Übertragungsleitung nutzen, miteinander kommunizieren, so müssen sie dies indirekt über andere Router tun. Es

kann viele Pfade in dem Netz geben, die diese beiden Router verbinden. Das Netz trifft die Entscheidung darüber, welcher Pfad benutzt werden soll, mithilfe eines **Routing-Algorithmus**. Es gibt viele solcher Algorithmen. Der Router trifft die Entscheidung, wohin ein Paket als Nächstes geschickt werden soll, mithilfe eines **Forwarding-Algorithmus**. Auch hiervon gibt es viele. Einige von beiden Arten werden wir detailliert in *Kapitel 5* untersuchen.

Andere Arten von WANs machen regen Gebrauch von kabellosen Technologien. In Satellitensystemen hat jeder Computer am Boden eine Antenne, über die er Daten an einen Satelliten senden bzw. von einem Satelliten im Orbit empfangen kann. Alle Rechner können „hören“, was vom Satelliten kommt, und manchmal auch das, was von anderen Computern an den Satelliten gesendet wird. Satellitennetze sind grundsätzlich Broadcast-Netze und am nützlichsten, wenn Broadcasting wichtig ist.

Das Funknetz für Mobiltelefone ist ein weiteres Beispiel für ein drahtloses WAN. Dieses System befindet sich bereits in der dritten Generation und eine vierte taucht gerade am Horizont auf. Die erste Generation war analog und nur für Sprache ausgelegt. Die zweite Generation war digital und nur für Sprache ausgelegt. Die dritte Generation ist digital und sowohl für Sprache als auch für Daten ausgelegt. Die Entfernung, die jede einzelne Basisstation abdeckt, ist bedeutend größer als bei einem drahtlosen LAN, wobei hier die Reichweiten in Kilometern statt in einigen Metern gemessen werden. Die Basisstationen sind miteinander über ein Backbonenetz verbunden, das in der Regel verkabelt ist. Die Datenraten von Funknetzwerken liegen häufig in der Größenordnung von 1 Mbit/s und sind damit viel geringer als bei einem drahtlosem LAN, das bis zu Größenordnungen von 100 Mbit/s reichen kann. Über diese Netze wird in *Kapitel 2* noch viel zu sagen sein.

1.2.5 Internetworks

Die Welt ist voller Netze, die oft unterschiedliche Hard- und Software verwenden. Personen, die an ein Netz angeschlossen sind, möchten mit anderen, die an anderen Netzen angeschlossen sind, kommunizieren. Die Erfüllung dieses Wunsches setzt voraus, dass verschiedene, meist miteinander nicht kompatible Netze zusammenge schlossen werden. Eine Gruppe miteinander verbundener Netze nennt man **Internet work** oder **Internet**. Diese Begriffe werden nicht nur zur Bezeichnung des weltweiten Internets benutzt (welches ein spezielles Internetwork ist), sondern auch in einem allgemeinen Sinne. Hier wird der Terminus „Internetwork“ benutzt, wenn im allgemeinen Sinne von einem Zusammenschluss von Netzen die Rede ist, und der Terminus „Internet“, wenn das weltweite Internet gemeint ist. Das Internet benutzt ISP-Netze, um Unternehmensnetze, private und viele andere Netzwerke zu verbinden. Wir werden später auf das Internet noch sehr ausführlich eingehen.

Subnetze, Netze und Internetworks werden oft verwechselt. Der Begriff „Subnetz“ wird in Zusammenhang mit einem Fernnetz sinnvollerweise dort verwendet, wo er sich auf die Sammlung von Routern und Übertragungsleitungen des Netzbetreibers bezieht. Entsprechend besteht das Telefonsystem aus Vermittlungsstellen verschiede-

ner Ebenen, die untereinander über Hochgeschwindigkeitsleitungen verbunden und an Privathaushalte und Unternehmen über Leitungen mit niedriger Übertragungsgeschwindigkeit angeschlossen sind. Diese der Telefongesellschaft gehörenden Leitungen und Geräte bilden das Subnetz des Telefonsystems. Die Telefonapparate (analog zu den Hosts) gehören nicht zum Subnetz.

Ein Netzwerk wird aus der Kombination eines Subnetzes und seiner Hosts gebildet. Das Wort „Netzwerk“ wird jedoch häufig auch in einem lockeren Sinne gebraucht. Ein Subnetz könnte als ein Netzwerk beschrieben werden, wie im Fall des „ISP-Netzwerks“ in Abbildung 1.12. Ein Internetwork könnte auch als ein Netz beschrieben werden, wie im Fall des WAN in Abbildung 1.10. Wie werden dies ähnlich handhaben und wenn wir ein Netz von anderen Anordnungen unterscheiden, werden wir bei unserer ursprünglichen Definition von einer Gruppe von Computern bleiben, die durch eine einzige Technologie miteinander verbunden sind.

Lassen Sie uns noch ein wenig mehr darüber sagen, woraus ein Internetwork besteht. Wir wissen, dass ein Internetwork aus der Zusammenlegung verschiedener Netze gebildet wird. Aus unserer Sicht entsteht ein Internetwork in der Regel durch die Verbindung eines LAN mit einem WAN oder zweier LANs, aber über die Terminologie besteht in der Branche in diesem Bereich keine Einheitlichkeit. Es gibt hier zwei nützliche Faustregeln. Wenn erstens verschiedene Organisationen für die Errichtung eines Teils eines Netzes bezahlt haben und dieses auch warten, dann hat man ein Internetwork und kein einheitliches Netz. Wenn zweitens die zugrunde liegende Technologie in verschiedenen Bereichen unterschiedlich ist (wie Broadcast- und Punkt-zu-Punkt-Übertragungen oder auch verkabelt und drahtlos), so haben wir wahrscheinlich ein Internetwork.

Um tiefer zu gehen, müssen wir darüber sprechen, wie zwei unterschiedliche Netze miteinander verbunden werden können. Der allgemeine Name für eine Maschine, die eine Verbindung zwischen zwei oder mehreren Netzen herstellt und die notwendige Übersetzung bereitstellt – sowohl hinsichtlich der Hard- als auch der Software – ist **Gateway**. Gateways werden unterschieden durch die Schicht in der Protokollhierarchie, auf der sie operieren. Wir werden uns mit Schichten und Protokollhierarchien noch viel beschäftigen und beginnen damit im nächsten Abschnitt. Für den Moment stellen Sie sich vor, dass höhere Schichten eher an Anwendungen wie das Web gebunden sind und tiefere Schichten eher an Übertragungsverbindungen wie Ethernet angebunden sind.

Da der Vorteil, ein Internetwork zu bilden, darin liegt, Rechner über mehrere Netzwerke hinweg zu verbinden, wollen wir kein Gateway auf einer zu weit unten liegenden Ebene benutzen, denn damit wären wir nicht in der Lage, Verbindungen zwischen verschiedenen Netzwerkarten herzustellen. Wir wollen auch kein Gateway der zu weit oben liegenden Ebenen verwenden, denn damit würde die Verbindung nur für bestimmte Anwendungen funktionieren. Die Ebene in der Mitte, die „genau richtig“ ist, wird oft die Netzsicht genannt, und ein Router ist ein Gateway, das Pakete auf der Netzsicht schaltet. Wir können also ein Internetwork ausfindig machen, wenn wir ein Netz mit Routern finden.

1.3 Netzsoftware

Bei den ersten Rechnernetzen lag das Hauptaugenmerk beim Entwurf auf der Hardware, die Software spielte nur eine Nebenrolle. Diese Herangehensweise funktioniert nicht mehr. Netzsoftware ist heute hochgradig strukturiert. In den folgenden Abschnitten untersuchen wir die Strukturierungstechnik für Netzsoftware ausführlicher. Der hier beschriebene Ansatz bildet den Eckpfeiler des gesamten Buchs und wird wiederholt aufgegriffen.

1.3.1 Protokollhierarchien

Um ihre Komplexität zu verringern, sind die meisten Netze als mehrere übereinanderliegende **Schichten** oder **Ebenen** aufgebaut. Anzahl, Bezeichnung, Inhalt und Funktion der einzelnen Schichten unterscheiden sich je nach Netz. In allen Netzen haben Schichten den Zweck, den jeweils höheren Schichten bestimmte Dienste zur Verfügung zu stellen, diese Schichten aber von Einzelheiten, wie die Dienste implementiert werden, abzuschirmen. So ist eine Schicht gewissermaßen eine virtuelle Maschine, die bestimmte Dienste für die darüberliegende Schicht anbietet.

Dieses Konzept ist uns vertraut und wird in der gesamten Informatik verwendet. Hier ist es oftmals als Verbergen von Informationen, abstrakte Datentypen, Datenkapselung und objektorientierte Programmierung bekannt. Die Grundidee ist, dass ein bestimmtes Stück Software (oder Hardware) einen Dienst für die Benutzer zur Verfügung stellt, aber die Einzelheiten über seinen inneren Zustand und die Algorithmen vor ihnen verbirgt.

Wenn Schicht n auf einem Rechner mit Schicht n eines anderen Rechners kommuniziert, so werden die Regeln und Konventionen, auf denen diese Kommunikation basiert, insgesamt das Schicht- n -Protokoll genannt. Im Grunde ist ein **Protokoll** eine Vereinbarung zwischen kommunizierenden Parteien über den Ablauf der Kommunikation. Analog kann eine Frau, die einem Mann vorgestellt wird, ihre Hand ausstrecken. Der Mann kann sich seinerseits für einen kräftigen Händedruck oder einen charmanten Handkuss entscheiden, je nachdem, ob sie eine amerikanische Anwältin auf Geschäftsreise oder eine europäische Prinzessin auf einem Ball ist. Die Verletzung des Protokolls erschwert die Kommunikation oder macht sie sogar völlig unmöglich.

► Abbildung 1.13 zeigt ein Netz mit fünf Schichten. Die Einheiten, die die entsprechenden Schichten auf verschiedenen Rechnern bilden, nennt man **Peers**. Diese Peers können Softwareprozesse, Hardwaregeräte oder auch Menschen sein. Anders ausgedrückt, es sind die Peers, die mithilfe des Protokolls miteinander kommunizieren.

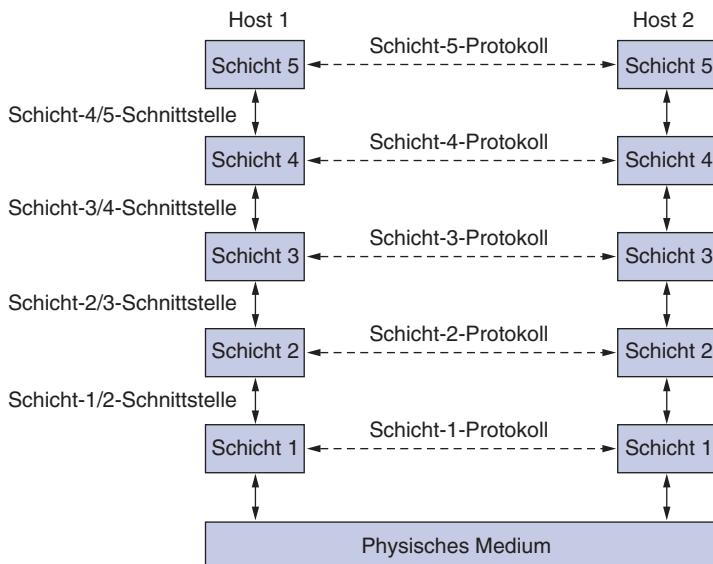


Abbildung 1.13: Schichten, Protokolle und Schnittstellen.

In Wirklichkeit werden Daten nicht direkt von Schicht n des einen Rechners zu Schicht n eines anderen übertragen. Vielmehr leitet jede Schicht Daten und Steuerinformationen an die unmittelbar darunterliegende Schicht weiter, bis die unterste Schicht erreicht ist. Unter Schicht 1 liegt das **physische Medium**, über das die Kommunikation stattfindet. In Abbildung 1.13 ist die virtuelle Kommunikation durch gestrichelte und die physische durch durchgezogene Linien dargestellt.

Zwischen je zwei angrenzenden Schichten befindet sich eine **Schnittstelle** (*interface*). Die Schnittstelle definiert, welche Basisoperationen und -dienste die untere Schicht der oberen Schicht anbietet. Wenn Netzentwickler über die Anzahl der in ein Netz einzubindenden Schichten und darüber entscheiden, was diese Schichten alles machen sollen, dann ist die Definition sauberer Schnittstellen einer der wichtigsten Aspekte. Das setzt wiederum voraus, dass jede Schicht eine bestimmte Anzahl von wohlverstandenen Funktionen ausführt. Zusätzlich zur Minimierung der Informationsmenge, die zwischen den Schichten weiterzugeben ist, vereinfachen saubere Schnittstellen auch die Ablösung einer Schicht durch ein völlig anderes Protokoll oder eine andere Implementierung (z.B. alle Telefonleitungen durch Satellitenkanäle ersetzen), weil für das neue Protokoll bzw. die neue Implementierung nur erforderlich ist, dass sie ihren Nachbarn ein Stockwerk höher genau die gleichen Dienste bietet wie die alte. Es ist üblich, dass verschiedene Hosts auch unterschiedliche Implementierungen desselben Protokolls (oft von unterschiedlichen Firmen geschrieben) verwenden. Tatsächlich kann sich das Protokoll selbst innerhalb einer Schicht ändern, ohne dass die Schichten darüber und darunter es überhaupt bemerken.

Eine Menge von Schichten und Protokollen nennt man **Netzarchitektur**. Die Spezifikation einer Architektur muss genügend Informationen enthalten, damit ein Entwickler

das Programm oder die Hardware für jede Schicht so implementieren kann, dass es das vorgesehene Protokoll korrekt befolgt. Weder die Einzelheiten der Implementierung noch die Spezifikation der Schnittstellen sind Teil der Architektur, da diese im Rechner versteckt und nicht von außen sichtbar sind. Die Schnittstellen aller Rechner in einem Netz müssen nicht einmal gleich sein, solange jeder Rechner alle Protokolle korrekt verwenden kann. Eine Liste der Protokolle, eines für jede Schicht, die ein bestimmtes System verwenden kann, nennt man **Protokollstapel** (*protocol stack*). Netzarchitekturen, Protokollstapel sowie die Protokolle selbst zählen zu den Schwerpunktthemen dieses Buchs.

Mit einer Analogie lässt sich der Grundgedanke der mehrschichtigen Kommunikation verdeutlichen. Stellen Sie sich zwei Philosophen (Peer-Prozesse auf Schicht 3) vor, von denen einer Urdu und Englisch und der andere Chinesisch und Französisch spricht. Die beiden wollen miteinander kommunizieren. Da sie keine gemeinsame Sprache sprechen, engagiert jeder einen Dolmetscher (Peer-Prozesse auf Schicht 2), von denen sich wiederum jeder an eine Sekretärin (Peer-Prozesse auf Schicht 1) wendet. Philosoph 1 möchte seinem Partner seine Vorliebe für *oryctolagus cuniculus* mitteilen. Hierfür gibt er eine Nachricht mit dem Wortlaut „I like rabbits“ (in Englisch) über Schnittstelle 2/3 an seinen Übersetzer (► Abbildung 1.14). Die Übersetzer haben sich auf eine neutrale Sprache, in diesem Fall Holländisch, geeinigt, sodass die Nachricht in „Ik vind konijnen leuk“ konvertiert wird. Die Wahl der Sprache ist das Schicht-2-Protokoll und wird durch Peer-Prozesse der Schicht 2 geregelt.

Der Übersetzer gibt die Nachricht an eine Sekretärin zur Übertragung weiter, z.B. per E-Mail (Schicht-1-Protokoll). Wenn die Nachricht bei der anderen Sekretärin ankommt, wird sie dem dortigen Übersetzer gegeben, der sie ins Französische übersetzt und über die Schnittstelle 2/3 an den zweiten Philosophen weitergibt. Jedes Protokoll ist völlig unabhängig vom anderen – solange sich die Schnittstellen nicht ändern. Die Übersetzer können beispielsweise von Holländisch auf Finnisch wechseln, wenn sich beide einig sind und keine Änderungen an den Schnittstellen zu Schicht 1 und 3 vorgenommen werden. Ähnlich können die Sekretärinnen von E-Mail auf Telefon umsteigen, ohne dass die anderen Schichten gestört (bzw. nicht einmal informiert) werden. Jeder Prozess kann Informationen hinzufügen, die nur für seinen Peer gedacht sind. Diese Information wird nicht an die darüberliegende Schicht weitergegeben.

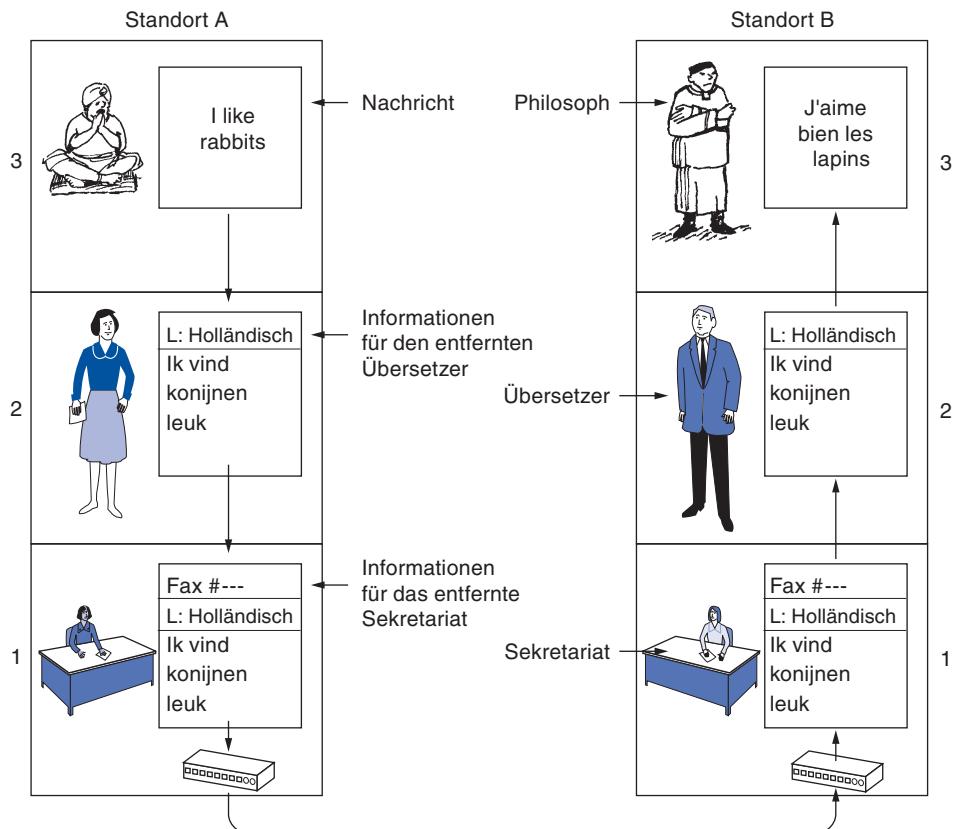


Abbildung 1.14: Die Architektur „Philosoph-Übersetzer-Sekretärin“.

Betrachten wir nun ein etwas technischeres Beispiel: Wie wird die Kommunikation mit der obersten Schicht in dem fünfschichtigen Netzwerk aus ►Abbildung 1.15 bereitgestellt? Eine Nachricht M wird von einem Anwendungsprozess auf Schicht 5 erzeugt und an Schicht 4 zur Übertragung weitergegeben. Die Schicht 4 setzt zur eindeutigen Bestimmung einen **Header** (Protokollkopf) an den Anfang der Nachricht und übergibt das Ergebnis an Schicht 3. Der Header enthält Steuerungsinformationen wie Adressen, damit Schicht 4 am Ziel die Nachrichten richtig zustellen kann. Andere Steuerungsinformationen, die in einigen Schichten verwendet werden, sind zum Beispiel fortlaufende Nummern (für den Fall, dass die darunterliegenden Schichten nicht die Reihenfolge der Nachrichten beibehalten), Größe und die Uhrzeit.

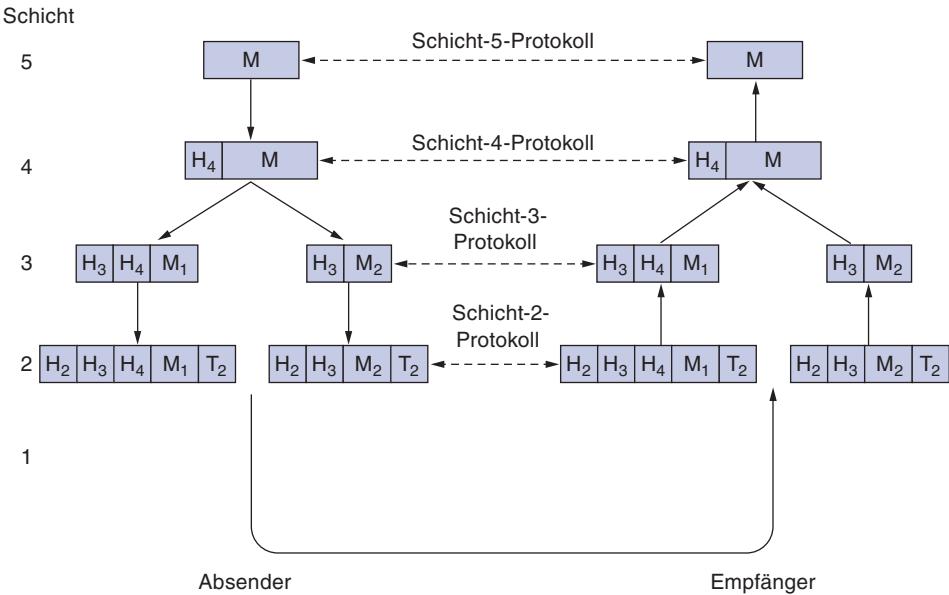


Abbildung 1.15: Informationsfluss, der die virtuelle Kommunikation auf Schicht 5 unterstützt.

In vielen Netzen gibt es keine Beschränkung für die Größe der mit dem Protokoll der Schicht 4 zu übertragenden Nachrichten, wohingegen das Schicht-3-Protokoll fast immer eine Größenbeschränkung auferlegt. Daher muss Schicht 3 die eingehenden Nachrichten in kleinere Einheiten, die Pakete, aufteilen, wobei jedes Paket mit einem Schicht-3-Header versehen wird. In dem Beispiel wird M in M_1 und M_2 aufgeteilt, die unabhängig voneinander übertragen werden.

Schicht 3 entscheidet, welche der abgehenden Leitungen verwendet wird, und über gibt die Pakete an Schicht 2. Schicht 2 fügt nicht nur zu jedem Teil einen Header, sondern auch einen **Trailer** (Nachspann) hinzu und übergibt die daraus resultierende Einheit an Schicht 1 zur eigentlichen Datenübertragung. Im empfangenden Rechner wird die Nachricht Schicht für Schicht nach oben transportiert, wobei die Header einer nach dem anderen wieder entfernt werden. Kein Header von Schichten unterhalb von n wird an die Schicht n übergeben.

Wichtig am Beispiel in Abbildung 1.15 ist die Beziehung zwischen der virtuellen und der tatsächlichen Kommunikation und der Unterschied zwischen den Protokollen und Schnittstellen. Die Peer-Prozesse auf Schicht 4 beispielsweise halten ihre Kommunikation konzeptionell für horizontal und verwenden das Protokoll der Schicht 4. Jeder hat wahrscheinlich eine Prozedur wie *SchickeAnAndereSeite* und *HoleVonAndererSeite*, selbst wenn diese Prozeduren nicht mit der anderen Seite, sondern mit den niedrigeren Schichten über die 3/4-Schnittstelle kommunizieren.

Die Abstraktion hin zu Peer-Prozessen ist für jeden Netzentwurf von elementarer Wichtigkeit. Dadurch kann die schwierige Aufgabe der Entwicklung eines kompletten Netzes in mehrere kleinere handhabbare Entwurfsprobleme aufgeteilt werden, d.h. in den Entwurf der einzelnen Schichten.

Obwohl Abschnitt 1.3 die Überschrift „Netzsoftware“ trägt, sei darauf hingewiesen, dass die unteren Schichten einer Protokollhierarchie häufig in Hardware oder Firmware implementiert werden. Dennoch sind hier komplexe Protokollalgorithmen involviert, auch wenn diese (ganz oder teilweise) in Hardware implementiert sind.

1.3.2 Entwurfsaspekte bei Schichten

Einige der grundlegenden Aspekte des Entwurfs von Rechnernetzen tauchen Schicht um Schicht auf. Die wichtigsten davon werden in diesem Abschnitt kurz beschrieben.

Zuverlässigkeit ist der Entwurfsaspekt, der sich damit befasst, ein Netz so konstruieren, dass es selbst dann korrekt funktioniert, wenn es aus unzuverlässigen Komponenten aufgebaut ist. Denken Sie an die Bits eines Pakets, die durch das Netzwerk reisen. Es könnte sein, dass einige dieser Bits beschädigt (invertiert) empfangen werden, ausgelöst durch zufällige elektrische Störungen, zufällige Funksignale, Hardwarefehler, Softwarefehler und so weiter. Wie ist es möglich, diese Fehler zu finden und zu beheben?

Ein Mechanismus zum Auffinden von Fehlern in erhaltener Information benutzt Code zu **Fehlererkennung** (*error detection*). Information, die falsch empfangen wurde, kann dann noch einmal übertragen werden, bis sie korrekt ankommt. Noch stärkere Codes ermöglichen die **Fehlerbehebung** (*error correction*). Hierbei wird die korrekte Nachricht aus den ursprünglich empfangenen und möglicherweise inkorrekten Bits wiederhergestellt. Bei beiden Verfahren wird redundante Information hinzugefügt. Auf den unteren Schichten werden sie eingesetzt, um Pakete zu schützen, die über individuelle Verbindungen gesendet werden, und auf den hohen Schichten, um zu überprüfen, ob die richtigen Inhalte empfangen wurden.

Ein weiteres Problem im Zusammenhang mit Zuverlässigkeit ist das Auffinden eines Arbeitspfads durch ein Netzwerk. Oft gibt es mehrere Pfade zwischen einer Quelle und einem Ziel und in einem größeren Netz kann es vorkommen, dass einige Verbindungen oder Router außer Betrieb sind. Stellen Sie sich vor, das Netz in Deutschland ist zusammengebrochen. Pakete, die von London nach Rom über Deutschland geschickt werden, können nicht weitergeleitet werden, doch wir könnten stattdessen Pakete von London nach Rom über Paris senden. Das Netz sollte diese Entscheidung automatisch treffen. Diesen Sachverhalt nennt man **Routing**.

Ein zweites Entwurfsproblem betrifft die Weiterentwicklung des Netzes. Mit der Zeit werden Netze größer und neue Entwürfe tauchen auf, die mit dem vorhandenen Netzwerk verbunden werden müssen. Wir haben eben den wichtigsten Mechanismus kennengelernt, der eingesetzt wird, um Änderungen zu unterstützen, indem das übergeordnete Problem aufgeteilt wird und die Implementierungsdetails versteckt werden: **Protokollsichten**. Es gibt aber auch noch viele andere Strategien.

Da in ein Netz viele Computer eingebunden sind, benötigt jede Schicht einen Mechanismus zur Identifikation des Senders und des Empfängers, die an dem Versenden dieser bestimmten Nachricht beteiligt sind. Dieser Mechanismus heißt **Adressierung** (*addressing*) in den unteren Schichten bzw. **Namensgebung** (*naming*) in den oberen Schichten.

Ein Aspekt des Wachstums ist, dass unterschiedliche Netztechnologien häufig verschiedene Einschränkungen haben. Zum Beispiel behalten nicht alle Kommunikationskanäle die Reihenfolge der übertragenen Nachrichten bei, was durch eine Nummerierung der Nachrichten gelöst werden kann. Ein anderes Beispiel sind die Unterschiede bei der maximalen Größe einer Nachricht, die das Netz übermitteln kann. Daher muss die Nachricht in Teile zerlegt, übermittelt und dann wieder neu zusammengesetzt werden. Dieses übergeordnete Thema heißt **Internetworking**.

Wenn Netze größer werden, entstehen neue Probleme. Städte können Verkehrsstaus haben, die Telefonnummern könnten knapp werden und man verläuft sich leicht. Die meisten Menschen haben diese Probleme in ihrer eigenen Nachbarschaft nicht, aber auf die gesamte Stadt bezogen könnte das von Bedeutung sein. Entwürfe, die auch dann noch gut funktionieren, wenn das Netz größer wird, bezeichnet man als **skalierbar (scalable)**.

Ein drittes Entwurfsproblem ist die Ressourcenzuteilung. Netze stellen einen Dienst an Hosts zur Verfügung auf Basis der ihnen zugrunde liegenden Ressourcen, wie zum Beispiel die Kapazität der Übertragungsleitungen. Dazu benötigen sie Mechanismen zur Aufteilung dieser Ressourcen, sodass sich die Hosts untereinander nicht zu sehr beeinträchtigen.

Viele Entwürfe teilen die Netzbandbreiten dynamisch, abhängig vom kurzfristigen Bedarf des Hosts, anstatt jedem Host einen festen Anteil der Bandbreite zuzuweisen, die er nutzen kann oder nicht. Dieser Entwurf heißt **statistisches Multiplexen** und bedeutet, dass die gemeinsame Nutzung auf der statistischen Nachfrage basiert. Statistisches Multiplexen kann auf den unteren Schichten für eine einzelne Verbindung angewandt werden oder auf höheren Schichten für ein Netzwerk oder sogar für Anwendungen, die das Netz benutzen.

Ein Zuweisungsproblem auf jeder Ebene besteht darin, einen schnellen Sender daran zu hindern, einen langsamen Empfänger mit Daten zu überschwemmen. Häufig wird eine Rückmeldung vom Empfänger an den Sender benutzt. Dieses Themengebiet nennt man **Flusskontrolle (flow control)**. Manchmal ist das Netz überladen, weil zu viele Rechner zu viel Datenverkehr verursachen und das Netz nicht alle bedienen kann. Dieses Problem nennt man **Netzüberlastung (congestion)**. Eine Strategie dagegen sieht vor, dass jeder Computer seine Anfragen verringert, sobald er die Überlastung wahrnimmt. Auch dies kann in allen Schichten angewandt werden.

Es ist interessant zu beobachten, dass das Netz mehr Ressourcen als nur die Bandbreite anzubieten hat. Bei Anwendungen wie der Übertragung von Live-Video spielen zeitliche Aspekte eine große Rolle. Die meisten Netze müssen sowohl Anwendungen bedienen, die dieses Senden in **Echtzeit (real-time)** verlangen, als auch gleichzeitig solche, die einen hohen Durchsatz wünschen. Unter dem Begriff **Dienstgüte (quality of service)** werden die Anstrengungen zusammengefasst, diese konkurrierenden Ansprüche auszugleichen.

Das letzte große Entwurfsproblem ist, das Netz sicher zu machen und es gegen verschiedene Bedrohungarten zu verteidigen. Eine dieser Bedrohungen hatten wir

bereits erwähnt: das Abhören der Kommunikation. Schutz vor dieser Bedrohung bieten Mechanismen, die **Vertraulichkeit** (*confidentiality*) garantieren, sie werden in mehreren Schichten eingesetzt. Einrichtungen zur **Authentifizierung** (*authentication*) sorgen dafür, dass sich niemand als eine andere Person ausgeben kann. Mit ihrer Hilfe können gefälschte Webseiten einer Bank von den echten unterschieden werden, Mobilfunknetze können feststellen, ob ein Anruf tatsächlich von Ihrem Telefon kommt und Sie somit die Rechnung dafür bezahlen. Andere Mechanismen schützen die **Integrität** der Daten und bewahren den Nutzer damit vor betrügerischen Änderungen, zum Beispiel dem Verändern der Nachricht „Belaste mein Konto mit 10 Euro“ zu „Belaste mein Konto mit 1 000 Euro“. All diese Entwürfe basieren auf Kryptografie, die wir in *Kapitel 8* behandeln.

1.3.3 Verbindungsorientierte und verbindungslose Dienste

Jede Schicht kann der jeweils übergeordneten Schicht zwei verschiedene Dienstarten zur Verfügung stellen: verbindungsorientierte und verbindungslose. In diesem Abschnitt werden wir einen Blick auf diese zwei Typen und deren Unterschiede werfen.

Verbindungsorientierte Dienste (*connection-oriented service*) sind in der Modellierung dem Telefonsystem nachempfunden. Um mit jemandem zu sprechen, nimmt man den Hörer in die Hand, wählt die Nummer, spricht und legt dann wieder auf. Ähnlich verläuft das bei einem verbindungsorientierten Netzdienst: Der Dienstnutzer baut zuerst eine Verbindung auf, benutzt diese und löst sie schließlich auf. Der wesentliche Aspekt einer Verbindung ist, dass sie wie ein Rohr funktioniert: Der Sender schiebt Objekte (Bits) an einem Ende hinein und der Empfänger entnimmt sie am anderen Ende in der gleichen Reihenfolge. In den meisten Fällen bleibt die Reihenfolge erhalten, sodass die Bits in der Reihenfolge eintreffen, in der sie gesendet wurden.

In einigen Fällen führen der Sender, der Empfänger und das Subnetz eine **Verhandlung** (*negotiation*) über die zu verwendenden Parameter, wie die maximale Nachrichtengröße, die Güte des erforderlichen Dienstes und andere Dinge. Im Normalfall macht eine Seite einen Vorschlag und die andere Seite kann dies akzeptieren, zurückweisen oder einen Gegenvorschlag machen. Eine **Schaltung** (*circuit*) ist ein anderer Name für eine Verbindung mit den dazugehörigen Ressourcen wie eine festgelegte Bandbreite. Diese Bezeichnung stammt aus der Terminologie des Telefonnetzes, wo eine Schaltung ein Pfad über eine Kupferleitung war, welche das Telefongespräch übertrug.

Im Gegensatz zum verbindungsorientierten Dienst ist der **verbindungslose Dienst** (*connection less service*) dem Postwesen nachempfunden. Jede Nachricht (Brief) enthält die vollständige Adresse und wird unabhängig von allen folgenden Nachrichten durch die dazwischenliegenden Knoten innerhalb des Systems geleitet. Es gibt unterschiedliche Namen für Nachrichten in unterschiedlichen Zusammenhängen. Ein **Paket** (*packet*) ist eine Nachricht in der Netzsicht. Wenn die dazwischenliegenden Knoten eine Nachricht im Ganzen empfangen, bevor sie zum nächsten Knoten weitergeleitet wurde, wird dies **Store-and-forward-Schaltung** genannt. Die andere Variante ist, bei der die Weiterleitung der Nachricht an den nächsten Knoten beginnt, bevor sie vollständig

am ersten Knoten angekommen ist, heißt **Cut-through-Schaltung**. Werden zwei Nachrichten an die gleiche Adresse gesandt, so wird normalerweise diejenige als Erste ankommen, die früher gesendet wurde. Zuweilen kann sich die erste aber auch verzögern, sodass die zweite vor der ersten ankommt.

Jeder Dienst kann weiter nach seiner Zuverlässigkeit charakterisiert werden. Einige Dienste sind in dem Sinn zuverlässig, dass sie nie Daten verlieren. Normalerweise wird ein zuverlässiger Dienst so implementiert, dass der Empfänger den Erhalt jeder Nachricht bestätigen muss, sodass sich der Sender über die Ankunft der Nachricht sicher sein kann. Dieser Bestätigungsprozess bringt zusätzlichen Aufwand und Verzögerungen mit sich, die sich oft lohnen, vielfach aber äußerst unerwünscht sind.

Eine Dateiübertragung ist beispielsweise eine typische Situation, für die ein zuverlässiger verbindungsorientierter Dienst angemessen ist. Der Eigentümer der Datei möchte sicher sein, dass alle Bits richtig und in der Reihenfolge ankommen, in der sie abgesendet wurden. Nur sehr wenige Kunden würden einen Dienst vorziehen, der gelegentlich ein paar Bits durcheinanderwirft oder verliert, auch wenn er viel schneller wäre.

Ein zuverlässiger verbindungsorientierter Dienst hat zwei Varianten, die sich geringfügig unterscheiden: Nachrichtensequenzen und Byteströme. Bei der ersten Form werden die Nachrichtengrenzen gewahrt. Wenn zwei 1 024-Byte-Nachrichten gesendet werden, dann kommen sie als zwei getrennte 1 024-Byte-Nachrichten und nicht als eine 2 048-Byte-Nachricht an. Bei der zweiten Form ist die Verbindung einfach ein Strom von Bytes ohne Nachrichtengrenzen. Kommen 2 048 Byte beim Empfänger an, dann kann nicht festgestellt werden, ob sie als eine 2 048-Byte-Nachricht, zwei 1 024-Byte-Nachrichten oder 2 048 1-Byte-Nachrichten gesendet wurden. Werden die Seiten eines Buchs über ein Netz an einen Setzer als getrennte Nachrichten gesendet, dann kann es wichtig sein, die Nachrichtengrenzen einzuhalten. Andererseits ist beim Download eines DVD-Films ein Bytestrom vom Server zum Rechner des Benutzers absolut ausreichend. Nachrichtengrenzen innerhalb des Films sind hier nicht relevant.

Bei manchen Anwendungen sind Verzögerungen, die durch die Bestätigungen entstehen, nicht akzeptabel. Ein Beispiel dafür ist die digitale Sprachübertragung für **Voice-over-IP**. Die Telefonbenutzer werden es als weniger störend empfinden, von Zeit zu Zeit etwas Rauschen in der Leitung zu hören, als ständig auf Rückmeldungen warten zu müssen. Ebenso ist es bei der Übertragung eines Videofilms nicht so tragisch, wenn ein paar Pixel falsch sind, während ein Film, der durch laufende Fehlerkorrekturen ständig ins Stocken kommt, sehr irritierend ist.

Nicht bei allen Anwendungen sind Verbindungen nötig. Beispielsweise senden Spammer elektronische Junkmail an viele Empfänger. Der Spammer wird sich wahrscheinlich nicht die Mühe machen, eine Verbindung zu einem Empfänger auf- und später wieder abzubauen, nur um ihm einmal etwas zu senden. Außerdem ist eine hundertprozentig zuverlässige Zustellung nicht entscheidend, insbesondere wenn dies mit Mehrkosten verbunden ist. Nötig ist lediglich eine Möglichkeit, eine einzelne Nachricht so zu versenden, dass sie mit hoher Wahrscheinlichkeit, aber ohne Garantie ankommt. Ein unzuverlässiger (das heißt unbestätigter) verbindungsloser Dienst wird **Datagrammdienst** (*data-*

gram service) genannt – analog zum Telegrammdienst, bei dem der Absender ebenfalls keine Empfangsbestätigung erhält. Trotz dieser Unzuverlässigkeit ist dies die vorherrschende Form in den meisten Netzen aus Gründen, die später klar werden.

In anderen Situationen ist zwar die Annehmlichkeit wünschenswert, keine Verbindung aufzubauen zu müssen, um eine kurze Nachricht zu schicken. Zuverlässigkeit ist dennoch unentbehrlich. Für solche Anwendungen kann ein **bestätigter Datagrammdienst** (*acknowledged datagram service*) bereitgestellt werden. Dieser Dienst lässt sich mit einem Einschreiben mit Rückschein vergleichen: Kommt der Rückschein zurück, so kann der Absender absolut sicher sein, dass der Brief dem gewünschten Empfänger zugestellt wurde und nicht unterwegs verloren gegangen ist. Textübermittlung von Mobiltelefonen aus ist ein Beispiel hierfür.

Ein weiterer Dienst ist der **Anfrage-Antwort-Dienst** (*request-reply service*). Bei diesem Dienst übermittelt der Sender ein einzelnes Datagramm mit einer Anfrage; die zurückgesandte Nachricht enthält die Antwort. Das Anfrage-Antwort-Modell wird üblicherweise verwendet, um Kommunikation in einer Client-Server-Umgebung zu implementieren: Der Client setzt eine Anfrage ab und der Server reagiert darauf. Zum Beispiel könnte ein Mobiltelefon-Client eine Anfrage an einen Karten-Server schicken, um Kartendaten für seinen aktuellen Aufenthaltsort abzurufen. ►Abbildung 1.16 fasst die bisher beschriebenen Dienstarten noch einmal zusammen.

| | Dienst | Beispiel |
|-----------------------|--------------------------------|----------------------|
| verbindungsorientiert | Zuverlässiger Nachrichtenstrom | Folge von Seiten |
| | Zuverlässiger Bytestrom | Download eines Films |
| | Unzuverlässige Verbindung | Voice-over-IP |
| verbindungslos | Unzuverlässiges Datagramm | Junk-E-Mail |
| | Bestätigtes Datagramm | Textnachrichten |
| | Anforderung/Antwort | Datenbankabfrage |

Abbildung 1.16: Sechs verschiedene Dienstarten.

Das Konzept, unzuverlässige Kommunikation zu verwenden, mag auf den ersten Blick seltsam erscheinen. Denn warum sollte jemand unzuverlässige Kommunikation der zuverlässigen vorziehen? Vor allem dann, wenn zuverlässige Kommunikation – das heißt in unserem Fall mit Bestätigungen – in einer bestimmten Schicht nicht verfügbar ist. So stellt Ethernet beispielsweise keine zuverlässige Kommunikation zur Verfügung. Pakete können gelegentlich bei der Übertragung beschädigt werden. Die höheren Protokollschichten müssen dieses Problem dann beheben. Viele zuverlässige Dienste setzen nämlich auf einem unzuverlässigen Datagrammdienst auf. Zweitens kann die Verzögerung inakzeptabel sein, die zur Bereitstellung eines zuverlässigen Dienstes erforderlich ist, insbesondere bei Echtzeitanwendungen wie Multimedia. Aus diesen Gründen gibt es sowohl zuverlässige wie auch unzuverlässige Kommunikation.

1.3.4 Basisoperationen von Diensten

Ein Dienst wird formal durch eine Gruppe von **Basisoperationen** (*primitive*) spezifiziert, mit denen ein Benutzerprozess auf den Dienst zugreifen kann. Diese Operationen weisen den Dienst an, eine Aktion auszuführen, oder berichten über eine Aktion, die von einer Peer-Einheit ausgeführt wurde. Befindet sich der Protokollstapel im Betriebssystem, dann sind die Basisoperationen normalerweise Systemaufrufe. Diese Aufrufe verursachen einen Sprung in den Kern des Betriebssystems, der dann die Steuerung des Rechners an das Betriebssystem übergibt, um die erforderlichen Pakete zu senden.

Die Menge der verfügbaren Operationen hängt von der Art des bereitgestellten Dienstes ab. Die Basisoperationen für verbindungsorientierte Dienste unterscheiden sich von denen für verbindungslose Dienste. Ein kleiner Ausschnitt an Dienstoperationen, die für einen zuverlässigen Bytestrom bereitgestellt werden müssen, ist in ►Abbildung 1.17 dargestellt. Diese werden den Fans der Berkeley-Socket-Schnittstelle vertraut sein, da sie eine vereinfachte Version dieser Schnittstelle sind.

| Basisoperation | Bedeutung |
|----------------|--|
| LISTEN | Blockiert, während auf eingehende Verbindung gewartet wird |
| CONNECT | Aufbau einer Verbindung zu einem wartenden Peer |
| ACCEPT | Ankommende Verbindung von einem Peer akzeptieren |
| RECEIVE | Blockiert, während auf eingehende Nachricht gewartet wird |
| SEND | Nachricht an einen Peer senden |
| DISCONNECT | Verbindung beenden |

Abbildung 1.17: Sechs Basisoperationen für Dienste, die einen einfachen verbindungsorientierten Dienst anbieten.

Diese Operationen können für eine Anfrage-Antwort-Interaktion in einer Client-Server-Umgebung verwendet werden. Um dies zu veranschaulichen, skizzieren wir ein einfaches Protokoll, welches den Dienst mithilfe von bestätigten Datagrammen implementiert.

Als Erstes führt der Server LISTEN aus um anzuzeigen, dass er für den Empfang eingehender Verbindungen bereit ist. Eine gängige Implementierungsweise von LISTEN ist ein blockierender Systemaufruf. Nach Ausführung dieser Operation wird der Server-Prozess blockiert, bis eine Verbindungsanforderung eingeht.

Als Nächstes führt der Client-Prozess CONNECT aus, um eine Verbindung zum Server herzustellen. Der CONNECT-Aufruf muss angeben, zu wem die Verbindung aufgebaut werden soll, sodass dieser vermutlich einen Parameter mit der Adresse des Servers enthalten wird. Das Betriebssystem sendet dann in der Regel ein Paket an den Peer und fordert ihn zur Anmeldung auf, wie in (1) in ►Abbildung 1.18 gezeigt. Der Client-Prozess wird unterbrochen, bis eine Antwort eingeht.

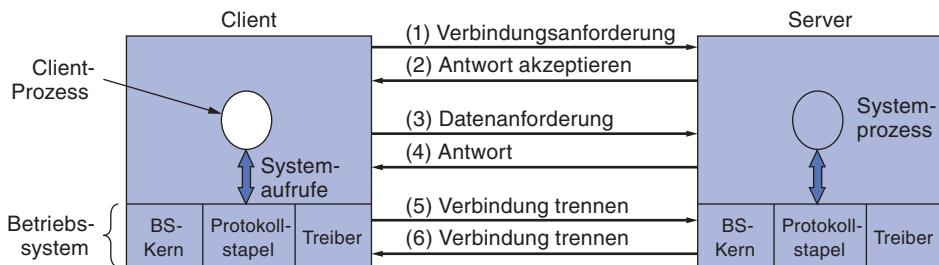


Abbildung 1.18: Eine einfache Client-Server-Interaktion, die bestätigte Datagramme verwendet.

Wenn das Paket beim Server ankommt, merkt das Betriebssystem, dass das Paket eine Verbindung anfordert. Es prüft, ob ein empfangsbereiter Prozess (Listener) vorhanden ist, in dem Fall wird die Blockade des Listeners aufgehoben. Der Server-Prozess kann dann die Verbindung über den ACCEPT-Aufruf einrichten. Dieser sendet eine Antwort (2) zurück zum Client-Prozess, um die Verbindung anzunehmen. Die Ankunft dieser Antwort gibt dann den Client frei. An diesem Punkt laufen sowohl der Client als auch der Server, und eine Verbindung ist aufgebaut.

Die offensichtliche Analogie zwischen diesem Protokoll und der realen Lebenswelt ist ein Kunde (Client), der den Kundendienstbeauftragten eines Unternehmens anruft. Am Tagesbeginn sitzt der Kundendienstbeauftragte neben seinem Telefon, für den Fall, dass dieses läutet. Irgendwann ruft dann ein Kunde an. Der Sachbearbeiter nimmt den Hörer ab und die Verbindung ist hergestellt.

Der nächste Schritt für den Server ist die Ausführung von RECEIVE, um sich für die Annahme der ersten Anfrage zu rüsten. Normalerweise macht dies der Server sofort, nachdem er aus dem LISTEN geweckt wurde, bevor die Bestätigung zurück zum Client gelangen kann. Der RECEIVE-Aufruf blockiert den Server.

Der Client führt dann SEND aus, um seine Anfrage (3) zu übertragen, gefolgt von der Ausführung von RECEIVE, um die Antwort zu erhalten. Die Ankunft des Anfrage-pakets beim Server gibt den Server frei, sodass dieser die Anfrage verarbeiten kann. Nach getaner Arbeit sendet dieser mit SEND die Antwort an den Client zurück (4). Die Ankunft der Antwort hebt die Blockade beim Client auf, der die Antwort nun untersuchen kann. Wenn der Client weitere Anfragen hat, kann er diese nun starten.

Wenn der Client fertig ist, trennt er die Verbindung mit DISCONNECT (5). In der Regel ist ein erstes DISCONNECT ein blockierender Aufruf, der den Client vorübergehend stilllegt und ein Paket mit der Information an den Server sendet, dass die Verbindung nicht länger benötigt wird. Wenn der Server das Paket erhält, gibt dieser auch ein DIS-CONNECT als Bestätigung für den Client aus und gibt die Verbindung frei (6). Wenn das Paket des Servers zurück zum Client gelangt, wird der Client-Prozess freigegeben und die Verbindung aufgelöst. Dies ist – kurz und knapp – das Prinzip der verbindungsorientierten Kommunikation.

Natürlich ist das Leben nicht so einfach. Vieles kann hier schiefgehen. Das Timing kann falsch sein (z.B. CONNECT wird vor LISTEN ausgeführt), Pakete können verlo-

ren gehen und vieles mehr. Wir werden diese Probleme später sehr detailliert behandeln, doch für den Augenblick reicht die Kurzfassung in Abbildung 1.18, um zu zeigen, wie die Kommunikation zwischen Client und Server mithilfe von bestätigten Datagrammen funktioniert, sodass wir verlorene Pakete ignorieren können.

Es werden hier für das vollständige Protokoll sechs Pakete benötigt – da kann man sich fragen, warum kein verbindungsloses Protokoll verwendet wird. Die Antwort ist, dass in einer perfekten Welt tatsächlich nur die zwei folgenden Pakete erforderlich wären: eines für die Anfrage und eines für die Antwort. Wenn jedoch große Nachrichten in beiden Richtungen versendet werden (z.B. eine Datei in Megabyte-Größe), sieht die Situation – angesichts von Übertragungsfehlern und verlorenen Paketen – schon anders aus. Wenn nun die Antwort aus Hunderten von Paketen besteht, von denen einige bei der Übertragung verloren gingen – wie soll der Client dann wissen, welche Teile fehlen? Wie weiß der Client, ob das zuletzt erhaltene Paket auch wirklich das zuletzt gesendete Paket ist? Angenommen, der Client benötigt eine zweite Datei. Wie kann er dann Paket 1 der zweiten Datei von einem verloren gegangenen Paket 1 aus der ersten Datei unterscheiden, das plötzlich doch noch beim Client angekommen ist? Kurz gesagt, in der realen Welt ist ein einfaches Anfrage-Antwort-Protokoll über ein unzuverlässiges Netzwerk oftmals ungeeignet. In *Kapitel 3* behandeln wir verschiedene Protokolle, die diese und andere Probleme beheben. Für den Moment genügt die Feststellung, dass es manchmal sehr angenehm sein kann, einen zuverlässigen, geordneten Bytestrom zu haben.

1.3.5 Beziehung zwischen Diensten und Protokollen

Dienste und Protokolle basieren auf verschiedenen Konzepten. Ihre Unterscheidung ist so wichtig, dass wir sie hier nochmals betonen wollen. Ein *Dienst* ist eine Gruppe von Basisoperationen, die eine Schicht der über ihr liegenden Schicht zur Verfügung stellt. Der Dienst definiert, welche Operationen diese Schicht für ihre Benutzer ausführen kann. Dies sagt aber nichts darüber aus, wie diese Operationen implementiert werden. Ein Dienst fungiert als Schnittstelle zwischen zwei Schichten, wobei die untere Schicht der Dienstanbieter und die obere der Dienstnutzer ist.

Ein *Protokoll* hingegen ist eine Menge von Regeln, die das Format und die Bedeutung der von den Peer-Einheiten innerhalb einer Schicht ausgetauschten Pakete oder Nachrichten festlegt. Mit Protokollen führen die Einheiten ihre definierten Dienste aus. Sie können ihre Protokolle nach Belieben verändern, solange sie nichts an den für den Dienstnutzer sichtbaren Diensten ändern. Das bedeutet, dass Dienst und Protokoll völlig voneinander getrennt sind. Dies ist ein wesentliches Konzept, das jeder Netzentwickler gut verstanden haben sollte.

Um diesen Kernpunkt noch einmal zu wiederholen: Dienste beziehen sich auf die Schnittstellen zwischen den Schichten (►Abbildung 1.19). Im Unterschied dazu beziehen sich Protokolle auf die Pakete, die zwischen Peer-Einheiten auf verschiedenen Rechnern versendet werden. Diese beiden Konzepte dürfen auf keinen Fall verwechselt werden.

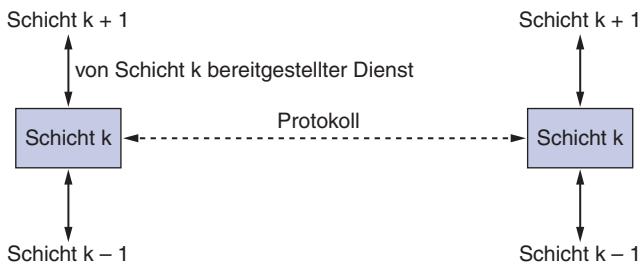


Abbildung 1.19: Die Beziehung zwischen Dienst und Protokoll.

Hier lohnt sich eine Analogie mit Programmiersprachen. Ein Dienst entspricht einem abstrakten Datentyp oder einem Objekt in einer objektorientierten Sprache. Er definiert Operationen, die für ein Objekt ausgeführt werden können, gibt aber nicht an, wie diese Operationen implementiert werden. Im Gegensatz dazu bezieht sich ein Protokoll auf die *Implementierung* des Dienstes und ist als solches für den Dienstnutzer nicht sichtbar.

Viele ältere Modelle haben nicht zwischen Dienst und Protokoll unterschieden. Eine typische Schicht verwendete beispielsweise eine Dienstoperation namens SEND PACKET, wobei der Benutzer einen Zeiger auf ein vollständig zusammengesetztes Paket bereitstellt. Diese Anordnung bedeutete, dass alle Änderungen des Protokolls unmittelbar für den Benutzer sichtbar waren. Die meisten Netzentwickler betrachten ein solches Design heute als ernsthaften Fehler.

1.4 Referenzmodelle

Nachdem wir uns mit der abstrakten Beschreibung von Netzsichten beschäftigt haben, kommen wir jetzt zu einigen Beispielen. Wir behandeln zwei wichtige Netzarchitekturen ausführlich: das OSI- und das TCP/IP-Referenzmodell. Auch wenn die mit dem OSI-Modell verknüpften *Protokolle* heute nicht mehr verwendet werden – das *Modell* selbst ist sehr allgemein formuliert und immer noch gültig. Die für jede Schicht erörterten Funktionen sind daher immer noch sehr wichtig. Beim TCP/IP-Modell ist es genau umgekehrt: Das Modell selbst wird nicht viel benutzt, doch die Protokolle sind weitverbreitet. Aus diesem Grund gehen wir auf beide ausführlich ein. Außerdem kann man manchmal aus Fehlern mehr lernen als aus Erfolgen.

1.4.1 Das OSI-Referenzmodell

Das OSI-Modell (ohne das physische Medium) wird in ►Abbildung 1.20 dargestellt. Dieses Modell basiert auf einem Vorschlag, der von der International Standards Organization (ISO) entwickelt wurde und den ersten Schritt auf dem Weg zur internationalen Standardisierung der verschiedenen Protokolle darstellt (Day und Zimmermann, 1983). Es wurde 1995 überarbeitet (Day, 1995). Dieses Modell trägt den Namen **ISO/OSI-Referenzmodell** (*Open Systems Interconnection*), weil es die Verbindung offener Systeme behandelt, d.h. Systeme, die für die Kommunikation mit anderen Systemen offen sind. Wir nennen es im weiteren Verlauf kurz **OSI-Modell**.

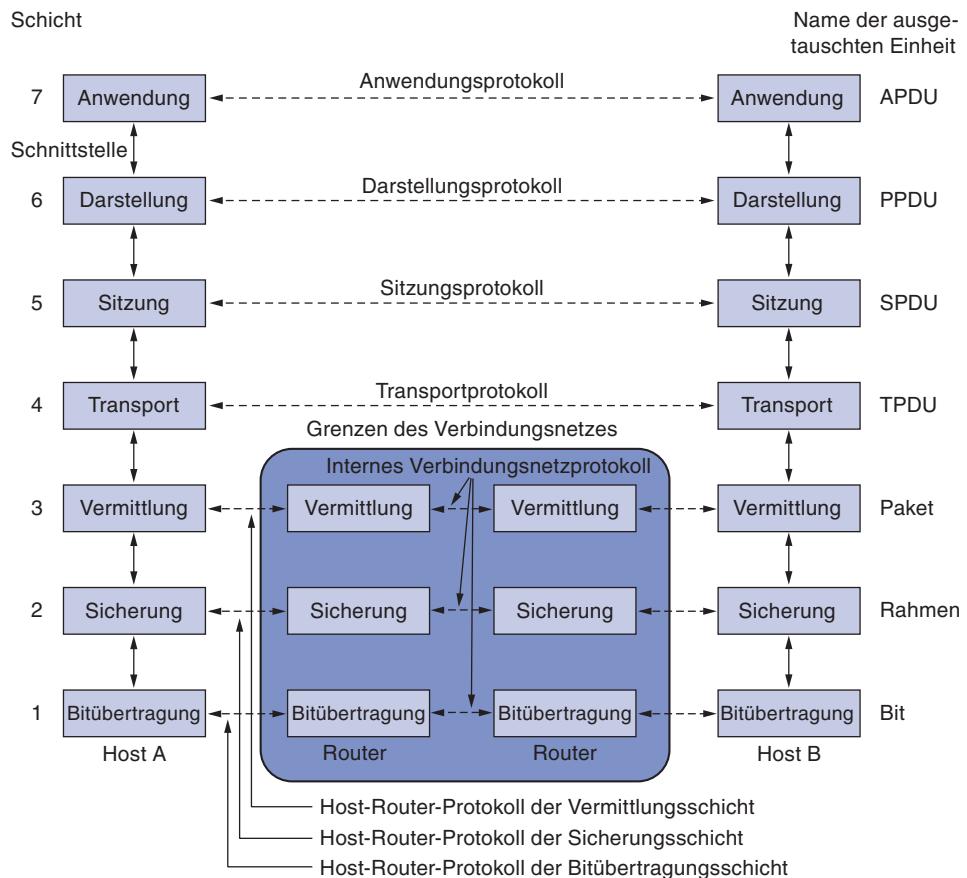


Abbildung 1.20: Das OSI-Referenzmodell.

Das OSI-Modell hat sieben Schichten. Folgende Grundgedanken haben zum Sieben-Schichten-Entwurf geführt:

1. Eine neue Schicht sollte dort erstellt werden, wo ein neuer Abstraktionsgrad benötigt wird.
2. Jede Schicht sollte eine genau definierte Funktion erfüllen.
3. Die Funktionen jeder Schicht sollten mit Blick auf die Definition international genormter Protokolle gewählt werden.
4. Die Grenzen zwischen den einzelnen Schichten sollten so gewählt werden, dass der Informationsfluss über die Schnittstellen möglichst gering ist.
5. Die Anzahl der Schichten sollte so groß sein, dass unterschiedliche Funktionen nicht in einer Schicht zusammengekürtzt werden müssen, aber so klein, dass die gesamte Architektur nicht unhandlich wird.

Die einzelnen Schichten des Modells werden nacheinander in den folgenden Abschnitten beschrieben. Wir beginnen dabei mit der untersten Schicht. Beachten Sie, dass das OSI-Modell selbst keine Netzarchitektur ist, weil es die in den Schichten zu benutzenden Dienste und Protokolle nicht im Detail angibt. Es sagt lediglich etwas darüber aus, welche Aufgaben die einzelnen Schichten ausführen sollen. ISO hat daneben auch Normen für alle Schichten ausgearbeitet, die aber nicht Teil des Referenzmodells sind. Diese Definitionen wurden als eigener internationaler Standard veröffentlicht. Das *Modell* wird (in Teilen) häufig eingesetzt, auch wenn die zugehörigen Protokolle längst vergessen sind.

Die Bitübertragungsschicht

Die **Bitübertragungsschicht** (*physical layer*) betrifft die Übertragung von reinen Bits über einen Kommunikationskanal. Beim Entwurf muss Folgendes beachtet werden: Wenn eine Seite ein 1-Bit schickt, muss dies auf der anderen Seite auch als ein 1-Bit und nicht als ein 0-Bit empfangen werden. Typische Fragen hier sind: Welche elektrischen Signale sollen zur Darstellung einer logischen 1 und einer 0 verwendet werden? Wie viele Nanosekunden soll ein Bit dauern? Soll die Übertragung in beide Richtungen gleichzeitig erfolgen? Wie kommt die Erstverbindung zustande? Wie wird sie wieder gelöst, wenn beide Seiten fertig sind? Wie viele Pins hat der Netzwerkstecker und wofür werden diese verwendet? Die Entwurfsprobleme betreffen hier weitgehend mechanische, elektrische und zeitorientierte Schnittstellen sowie das physikalische Übertragungsmedium, das sich unterhalb der Bitübertragungsschicht befindet.

Die Sicherungsschicht

Die Hauptaufgabe der **Sicherungsschicht** (*data link layer*) besteht darin, eine reine Übertragungseinrichtung in eine Leitung zu verwandeln, die frei von unerkannten Übertragungsfehlern erscheint. Erreicht wird dies durch Maskieren der realen Fehler, sodass die Vermittlungsschicht diese nicht wahrnimmt. Dies wiederum geschieht dadurch, dass der Sender die Eingangsdaten in **Datenrahmen** (*data frame*) aufteilt (typisch sind einige Hundert oder ein paar Tausend Byte) und diese Rahmen sequenziell überträgt. Ist der Dienst zuverlässig, bestätigt der Empfänger den korrekten Empfang, indem er einen **Bestätigungsrahmen** (*acknowledgement frame*) zurücksendet.

Ein weiteres Problem, das auf der Sicherungsschicht (und auf den meisten höheren Schichten) auftaucht, ist, wie man einen schnellen Übermittler davon abhalten kann, einen langsamen Empfänger mit Daten zu überschwemmen. Hierfür wird eine Art Verkehrsregelung benötigt, die den Übertragenden wissen lässt, wann der Empfänger mehr Daten annehmen kann.

Broadcast-Netze werfen in der Sicherungsschicht eine zusätzliche Frage auf: Wie kann der Zugriff auf den gemeinsamen Kanal gesteuert werden? Gelöst wird dieses Problem durch eine spezielle Zwischenschicht der Sicherungsschicht, der **MAC**-Teilschicht (*Medium Access Control*, Medienzugriffssteuerung).

Die Vermittlungsschicht

Die **Vermittlungsschicht** (*network layer*) steuert den Betrieb des Subnetzes. Eine der wichtigsten Entwurfsaufgaben ist hier die Auswahl der Paketrouten vom Ursprungsort zum Bestimmungsort. Die Routen können auf statischen Tabellen beruhen, die im Netz „verdrahtet“ sind und sich nur selten ändern, oder sie können (was häufiger passiert) automatisch aktualisiert werden, um ausgefallene Komponenten zu umgehen. Die Routen können auch zu Beginn jeder Kommunikation festgelegt werden, z.B. einer Terminalsitzung wie der Anmeldung bei einem entfernten Rechner. Schließlich können sie aber auch hochdynamisch sein und für jedes Paket neu bestimmt werden, um die optimale Netzauslastung zu gewährleisten.

Befinden sich im Subnetz zu viele Pakete gleichzeitig, stehen sie sich gegenseitig im Weg und es kommt zu Engpässen. Auch für den Umgang mit solch einer Überlastungssituation ist die Vermittlungsschicht verantwortlich, zusammen mit höheren Schichten, welche die auf das Netz gebrachte Last anpassen. Allgemein gehört die Qualität des bereitgestellten Dienstes (Verzögerung, Übertragungszeit, Jitter etc.) ebenfalls zu den Aufgaben der Vermittlungsschicht.

Ferner können viele Probleme entstehen, wenn ein Paket auf seiner Reise zum Bestimmungsort mehrere Netze durchquert. Es könnte z.B. sein, dass die Adressierung beim zweiten Netz anders ist als beim ersten. Das zweite Netz könnte das Paket überhaupt ablehnen, weil es zu groß ist. Die Protokolle können verschieden sein und anderes mehr. Alle Probleme dieser Art müssen auf der Vermittlungsschicht gelöst werden, damit heterogene Netze miteinander verbunden werden können.

Bei Broadcast-Netzen ist das Routing-Problem einfach, sodass die Vermittlungsschicht hier dünn oder gar nicht vorhanden ist.

Die Transportschicht

Die **Transportschicht** (*transport layer*) hat die grundlegende Aufgabe, Daten von der darüberliegenden Schicht zu übernehmen, diese gegebenenfalls in kleinere Einheiten zu zerlegen, sie an die Vermittlungsschicht zu übergeben und sicherzustellen, dass alle Teile richtig am anderen Ende ankommen. Dies alles muss außerdem effizient und in einer Weise erfolgen, dass die oberen Schichten von Änderungen der Hardwaretechnik abgeschirmt werden, die unvermeidlich im Laufe der Zeit anfallen.

Die Transportschicht bestimmt auch die Art des Dienstes, der der Sitzungsschicht – und letztendlich dem Netzbenutzer – zur Verfügung gestellt wird. Die gebräuchlichste Art von Transportverbindung ist ein fehlerfreier Punkt-zu-Punkt-Kanal, über den Nachrichten in der Sendereihenfolge übertragen werden. Es gibt aber auch andere Möglichkeiten für den Transportdienst, z.B. die Beförderung einzelner Nachrichten ohne Gewähr für die Einhaltung der ursprünglichen Reihenfolge oder die Rundsendung einer Nachricht an zahlreiche Empfänger (Broadcasting). Die Dienstart wird beim Aufbau der Verbindung festgelegt. (Nebenbei bemerkt kann ein völlig fehlerfreier Kanal nie erreicht werden. Eigentlich soll damit ausgedrückt werden, dass die Fehlerrate so gering ist, dass sie in der Praxis ignoriert werden kann.)

Die Transportschicht ist eine echte Ende-zu-Ende-Schicht; sie transportiert Daten den gesamten Weg von der Quelle bis zum Ziel. Anders ausgedrückt, kommuniziert ein Programm auf der Quelle mit einem analogen Programm auf dem Ziel und verwendet dabei die Nachrichten-Header und die Steuerdaten. Auf den niedrigeren Schichten agiert jedes Protokoll zwischen einem Rechner und dessen unmittelbarem Nachbarn und nicht zwischen der eigentlichen Quelle und dem eigentlichen Ziel, die durch viele Router voneinander getrennt sein können. Den Unterschied zwischen den vertakteten Schichten 1 bis 3 und den (Ende-zu-Ende-)Schichten 4 bis 7 wird in Abbildung 1.20 veranschaulicht.

Die Sitzungsschicht

Die **Sitzungsschicht** (*session layer*) ermöglicht es Benutzern an verschiedenen Rechnern, Sitzungen untereinander aufzubauen. Sitzungen stellen verschiedene Dienste zur Verfügung, wie **Dialogsteuerung** (*dialog control*), die verfolgt, wer gerade Daten übertragen darf; **Token-Verwaltung** (*token management*), die unterbindet, dass zwei Parteien eine wichtige Operation zur gleichen Zeit durchführen; und **Synchronisation** (*synchronization*), bei der bei langen Übertragungen Fixpunkte gesetzt werden, ab denen die Übertragung nach einem Absturz und der nachfolgenden Wiederherstellung fortgesetzt werden kann.

Die Darstellungsschicht

Im Unterschied zu den unteren Schichten, die vor allem mit der Übertragung von Bits beschäftigt sind, hat die **Darstellungsschicht** (*presentation layer*) die Syntax und Semantik der übertragenen Informationen zum Inhalt. Damit Computer mit intern unterschiedlichen Datendarstellungen kommunizieren können, müssen die ausgetauschten Datenstrukturen auf abstrakte Weise definiert werden, zusammen mit einer Standardcodierung, die „in der Leitung“ verwendet wird. Die Darstellungsschicht verwaltet diese abstrakten Datenstrukturen und ermöglicht die Definition und den Austausch von Datenstrukturen auf höheren Ebenen (wie Buchungsdatensätze).

Die Anwendungsschicht

Die **Anwendungsschicht** (*application layer*) enthält eine Vielzahl von Protokollen, die häufig von den Benutzern benötigt werden. Ein weitverbreitetes Anwendungsprotokoll ist **HTTP** (*HyperText Transfer Protocol*), das die Grundlage für das World Wide Web bildet. Möchte ein Browser auf eine Webseite zugreifen, sendet er den Namen der gewünschten Seite unter Verwendung des HTTP-Protokolls an den Host-Server dieser Seite. Der Server sendet daraufhin die Seite zurück. Andere Anwendungsprotokolle werden für Dateiübertragungen, E-Mail und Netznachrichten verwendet.

1.4.2 Das TCP/IP-Referenzmodell

Wir wollen uns nun vom OSI-Referenzmodell dem Referenzmodell zuwenden, das im ARPANET – dem Urahn aller Rechnernetze – und seinem Nachfolger – dem weltweiten Internet – eingesetzt wird. Wir werden zwar später noch einen kleinen Einblick in

die Geschichte von ARPANET geben, dennoch ist es sinnvoll, hier schon ein paar wichtige Punkte vorwegzunehmen. Das ARPANET war ein Forschungsnetz, das vom US-Verteidigungsministerium gefördert wurde. Es verband zunächst Hunderte von Universitäten und Regierungsbehörden über gemietete Telefonstandleitungen. Als später Satelliten- und Funknetze hinzukamen, warfen die verwendeten Protokolle bei der Zusammenarbeit der Netze Schwierigkeiten auf. Deshalb musste eine neue Referenzarchitektur gefunden werden. So war also fast von Anfang an die Fähigkeit, mehrere Netze nahtlos miteinander zu verbinden, eines der wichtigsten Entwurfsziele. Diese Architektur wurde später nach den zwei primären Protokollen das **TCP/IP-Referenzmodell** genannt. Es wurde zuerst von Cerf und Kahn beschrieben (1974) und später verfeinert und als ein Standard in der Internetgemeinde definiert (Braden, 1989). Die Entwurfsphilosophie des Modells wird von Clark (1988) diskutiert.

Aufgrund der Besorgnis des US-Verteidigungsministeriums, einige seiner wertvollsten Hosts, Router und netzübergreifenden Gateways könnten im Ernstfall in Sekunden durch einen Angriff der Sowjetunion in Stücke zerfetzt werden, war es ein weiteres wichtiges Ziel, das Netz bei einem Hardwareausfall von Subnetzen überlebensfähig zu halten, ohne die bestehenden Gespräche abzubrechen. Mit anderen Worten, das Verteidigungsministerium wollte, dass Verbindungen intakt bleiben, solange die Quell- und Zielrechner funktionierten, auch falls einige der dazwischenliegenden Geräte oder Übertragungsleitungen plötzlich zusammenbrechen sollten. Da man sich außerdem Anwendungen mit völlig unterschiedlichen Anforderungen ausmalte – von Dateiübertragungen bis hin zur Sprachübertragung in Echtzeit –, war eine flexible Architektur erforderlich.

Die Netzzugangsschicht

All diese Anforderungen führten zur Wahl eines paketvermittelten Netzes auf der Grundlage einer verbindungslosen Schicht, die über viele unterschiedliche Netze lief. Die unterste Schicht in dem Modell, die **Netzzugangsschicht** (*link layer*), beschreibt, was die Verbindungen wie serielle Leitungen und klassisches Ethernet tun müssen, um die Anforderungen dieser verbindungslosen Internetschicht zu erfüllen. Die Netzzugangsschicht ist eigentlich keine Schicht im üblichen Sinn des Begriffs, sondern eher eine Schnittstelle zwischen Hosts und Übertragungsleitungen. In frühen Veröffentlichungen über das TCP/IP-Modell findet man wenig darüber.

Die Internetschicht

Die **Internetschicht** (*internet layer*) ist die Sicherheitsnadel, die die gesamte Architektur zusammenhält. Anhand von ▶ Abbildung 1.21 können Sie sehen, dass die Internetschicht grob der OSI-Vermittlungsschicht entspricht. Ihre Aufgabe ist es, den Hosts zu ermöglichen, Pakete in jedes beliebige Netz einzuspeisen und unabhängig an das (potenziell in einem anderen Netz befindliche) Ziel zu befördern. Sie können sogar in einer völlig anderen Reihenfolge ankommen, als sie abgeschickt wurden. In diesem Fall ist es die Aufgabe der übergeordneten Schichten, sie erneut anzugeben, wenn die exakte Sendereihenfolge benötigt wird. Beachten Sie, dass „Internet“ hier in einem allgemeinen Sinn verwendet wird, auch wenn diese Schicht im Internet ebenfalls vorhanden ist.

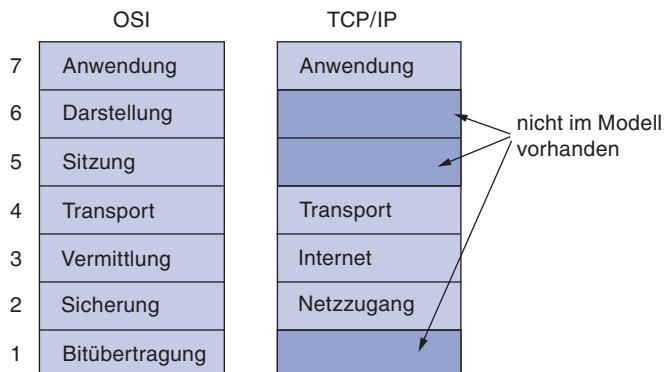


Abbildung 1.21: Das TCP/IP-Referenzmodell.

Eine Analogie dazu wäre die („gelbe“) Post. Man kann eine Reihe von Auslandsbriefen in einem Land in einen Briefkasten werfen und mit ein wenig Glück werden die meisten an die richtige Adresse im Zielland zugestellt. Die Briefe bereisen wahrscheinlich eine oder mehrere internationale Sammelstellen (Gateways) auf ihrem Weg, was für die Benutzer allerdings transparent ist. Auch dass jedes Land (d.h. Netz) seine eigenen Briefmarken, bevorzugten Umschlagformate und Zustellregeln hat, ist den Benutzern egal.

Die Internetschicht definiert ein offizielles Paketformat und Protokoll namens **IP** (*Internet Protocol*), außerdem ein dazu gehöriges Protokoll namens **ICMP** (*Internet Control Message Protocol*) zur Unterstützung. Die Internetschicht hat die Aufgabe, IP-Pakete richtig zuzustellen. Paket-Routing ist hier offensichtlich eines der Hauptprobleme, ebenso wie Überlastungen (obwohl sich IP als nicht effektiv beim Vermeiden von Überlastungen erwiesen hat).

Die Transportschicht

Die Schicht über der Internetschicht im TCP/IP-Modell wird heutzutage in der Regel als **Transportschicht** (*transport layer*) bezeichnet. Wie die OSI-Transportschicht soll sie Peer-Einheiten auf den Quell- und Ziel-Hosts die Kommunikation ermöglichen. Hier wurden anfangs zwei Ende-zu-Ende-Übertragungsprotokolle definiert. **TCP** (*Transmission Control Protocol*) ist ein zuverlässiges, verbindungsorientiertes Protokoll, über das ein Bytestrom von einem Rechner im Internet fehlerfrei einem anderen Rechner zugestellt wird. Es teilt den eingehenden Bytestrom in einzelne Nachrichten auf und leitet diese an die Internetschicht weiter. Am Ziel werden die Nachrichten vom empfangenden TCP-Prozess wieder zum Ausgabestrom zusammengesetzt. TCP verwaltet auch die Flusskontrolle, um sicherzustellen, dass ein langsamer Empfänger nicht von einem schnellen Sender mit Nachrichten überschwemmt wird.

Das zweite Protokoll auf dieser Schicht ist **UDP** (*User Datagram Protocol*). Dies ist ein unzuverlässiges, verbindungsloses Protokoll für Anwendungen, die anstelle der Sicherstellung der Reihenfolge oder der Flusskontrolle von TCP diese Aufgaben lieber selbst bereitstellen. Dieses Protokoll wird vorwiegend für einmalige Anfragen und Anwen-

dungen in Client-Server-Umgebungen verwendet, in denen die Schnelligkeit der Zustellung wichtiger ist als ihre Genauigkeit, z.B. bei der Übertragung von Sprache oder Video. Das Verhältnis zwischen IP, TCP und UDP geht aus ▶ Abbildung 1.22 hervor. Seit der Entwicklung dieses Modells wurde IP in vielen anderen Netzen implementiert.

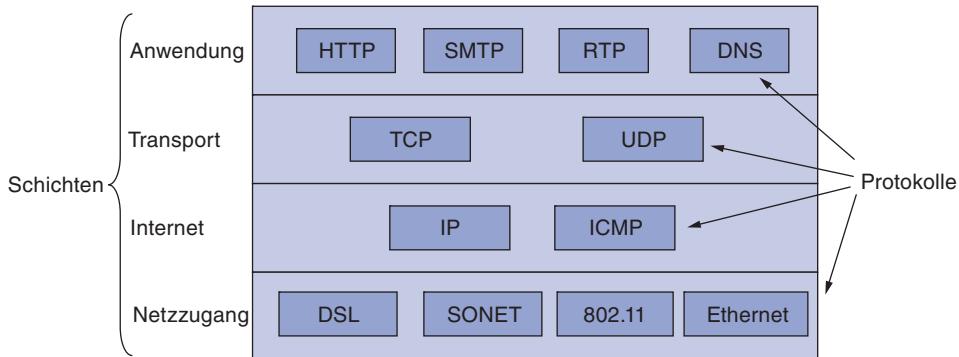


Abbildung 1.22: Ursprüngliche Protokolle und Netzwerke im TCP/IP-Modell.

Die Anwendungsschicht

Das TCP/IP-Modell hat keine Sitzungs- oder Darstellungsschichten. Man sah dafür einfach keine Notwendigkeit. Stattdessen binden Anwendungen einfach jede Sitzungs- und Darstellungsfunktion ein, die sie benötigen. Die Erfahrungen mit dem OSI-Modell haben gezeigt, dass diese Entscheidung richtig war: Für diese Schichten besteht in den meisten Anwendungen kein Bedarf.

Über der Transportschicht befindet sich die **Anwendungsschicht** (*application layer*). Sie umfasst alle Protokolle der höheren Schichten. Zu den frühen Entwicklungen gehören TELNET (virtuelles Terminal), FTP (*File Transfer Protocol*, Dateiübertragungsprotokoll) und SMTP (*Simple Mail Transfer Protocol*, einfaches E-Mail-Übertragungsprotokoll). Im Laufe der Jahre sind viele weitere Protokolle hinzugekommen. Einige wichtige, die wir uns noch genauer ansehen werden, sind in Abbildung 1.22 zusammengefasst, darunter DNS (*Domain Name System*, Domänennamenssystem) für die Zuordnung von Hostnamen an deren Netzadressen, HTTP (*HyperText Transfer Protocol*, Hypertext-Übertragungsprotokoll) zum Abruf von Seiten aus dem World Wide Web und RTP (*Real-time Transport Protocol*, Echtzeitübertragungsprotokoll), das Protokoll zum Zustellen von Echtzeitmedien, wie beispielsweise Sprache oder Filme.

1.4.3 Das in diesem Buch benutzte Modell

Wie bereits erwähnt, ist die Stärke des OSI-Referenzmodells das *Modell* selbst (abzüglich der Sitzungs- und Darstellungsschicht), das sich als außergewöhnlich nützlich bei der Diskussion von Rechnernetzen erwiesen hat. Demgegenüber liegt die Stärke des TCP/IP-Referenzmodells in den *Protokollen*, die seit vielen Jahren häufig angewendet werden. Da Informatiker ja immer alles auf einmal haben wollen, benutzen wir als Grundlage für dieses Buch das in ▶ Abbildung 1.23 dargestellte Hybridmodell.

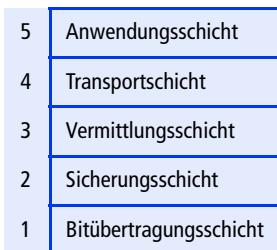


Abbildung 1.23: Das Referenzmodell, das wir in diesem Buch benutzen.

Dieses Modell hat fünf Schichten, von der Bitübertragungsschicht durch Sicherungs-, Vermittlungs- und Transportschichten bis hin zu der Anwendungsschicht. Die Bitübertragungsschicht legt fest, wie Bits über unterschiedliche Medienarten wie elektrische (oder andere analoge) Signale übertragen werden. Die Sicherungsschicht kümmert sich darum, wie Nachrichten endlicher Länge zwischen direkt verbundenen Rechnern mit festgelegten Zuverlässigkeitssstufen gesendet werden. Ethernet und IEEE 802.11 sind Beispiele von Protokollen der Sicherungsschicht.

Die Vermittlungsschicht befasst sich damit, wie mehrere Verbindungen in Netze und Netze von Netzen in Internetworks kombiniert werden, sodass wir Pakete zwischen entfernten Computern senden können. Hierzu gehört die Aufgabe, den Pfad zu finden, auf dem die Pakete gesendet werden. IP ist das Hauptbeispiel, das wir für diese Schicht untersuchen werden. Die Transportschicht verstärkt die Auslieferungsgarantie der Vermittlungsschicht, in der Regel mit gesteigerter Zuverlässigkeit, und stellt Zustellungsabstraktionen zur Verfügung, wie einen zuverlässigen Bytestrom, der die Bedürfnisse von verschiedenen Anwendungen erfüllt. TCP ist ein wichtiges Beispiel eines Protokolls der Transportschicht.

Schließlich enthält die Anwendungsschicht Programme, die das Netzwerk benutzen. Viele, nicht alle, Netzanwendungen haben Benutzerschnittstellen wie einen Webbrowser. Unser Interesse gilt jedoch dem Teil des Programms, der das Netz benutzt. Im Fall eines Webbrowsers ist dies das HTTP-Protokoll. Es gibt auch noch andere wichtige Unterstützungsprogramme in der Anwendungsschicht, wie das DNS, das von vielen Anwendungen benutzt wird.

Unsere Kapitelfolge basiert auf diesem Modell. Auf diese Art behalten wir den Wert des OSI-Modells zum Verständnis der Netzarchitekturen bei, doch konzentrieren uns hauptsächlich auf Protokolle, die in der Praxis wichtig sind, von TCP/IP und verwandten Protokollen bis zu neueren wie IEEE 802.11, SONET und Bluetooth.

1.4.4 Vergleich: OSI- und TCP/IP-Referenzmodell

Das OSI- und das TCP/IP-Referenzmodell haben viel gemeinsam. Beide basieren auf dem Konzept eines Stapels unabhängiger Protokolle. Auch die Funktionalität der Schichten ist recht ähnlich. So dienen z.B. die Schichten bis einschließlich zur Transportschicht bei beiden Modellen zur Bereitstellung eines netzunabhängigen Ende-zu-

Ende-Transportdienstes für kommunikationswillige Prozesse. Diese Schichten bilden den Transportanbieter. Die Schichten oberhalb der Transportschicht sind bei beiden Modellen anwendungsorientierte Nutzer des Transportdienstes.

Trotz dieser grundlegenden Ähnlichkeiten weisen die beiden Modelle auch viele Unterschiede auf. In diesem Abschnitt konzentrieren wir uns auf die wichtigsten Unterschiede zwischen den beiden Referenzmodellen. Beachten Sie aber, dass wir hier die *Referenzmodelle* und nicht die entsprechenden *Protokollstapel* vergleichen. Die Protokolle werden später ausführlich beschrieben. Ein ganzes Buch über Vergleich und Gegenüberstellung von TCP/IP und OSI ist Piscitello und Chapin (1993).

Das OSI-Modell basiert auf drei Konzepten:

- 1.** Dienste
- 2.** Schnittstellen
- 3.** Protokolle

Der wahrscheinlich größte Beitrag des OSI-Modells ist die explizite Unterscheidung dieser drei Konzepte. Jede Schicht erbringt bestimmte *Dienste* für die jeweils darüberliegende Schicht. Die Definition eines Dienstes gibt an, was die Schicht macht, nicht wie die darüberliegenden Einheiten darauf zugreifen oder wie die Schicht funktioniert. Anders ausgedrückt definiert sie die Semantik der Schicht.

Die *Schnittstelle* einer Schicht teilt den darüberliegenden Prozessen mit, wie sie darauf zugreifen können. Sie definiert die Parameter und welche Ergebnisse erwartet werden können. Sie sagt ebenfalls nichts darüber aus, wie die Schicht intern funktioniert.

Die *Protokolle* einer Schicht schließlich sind die Privatangelegenheit der betreffenden Schicht. Sie kann Protokolle nach eigenem Belieben benutzen, solange die Aufgabe ausgeführt werden kann (d.h. die angebotenen Dienste bereitgestellt werden können). Sie kann die Protokolle auch willkürlich ändern, ohne dass die Software in den höheren Schichten davon betroffen wird.

Diese Konzepte passen gut zu den modernen Vorstellungen über objektorientierte Programmierung. Wie eine Schicht hat ein Objekt eine Gruppe von Methoden (Operationen), die Prozesse außerhalb des Objekts aufrufen können. Die Semantik dieser Methoden definiert die Dienste, die das Objekt anbietet. Die Parameter und Ergebnisse von Methoden bilden die Schnittstelle des Objekts. Der objektinterne Code ist sein Protokoll. Es ist außerhalb des Objekts nicht sichtbar und auch nicht von Interesse.

Das TCP/IP-Modell unterschied ursprünglich nicht deutlich zwischen Dienst, Schnittstelle und Protokoll, trotz intensiver Bemühungen, es nachträglich mehr nach OSI-Art auszulegen. So waren z.B. die einzigen echten Dienste der Internetschicht SEND IP PACKET und RECEIVE IP PACKET. Als Folge sind die Protokolle im OSI-Modell besser verborgen als im TCP/IP-Modell und können im Zuge technologischer Änderungen relativ leicht ersetzt werden. Die Möglichkeit solcher transparenter Änderungen ist einer der Hauptzwecke, warum Protokolle von Anfang an in Schichten ausgelegt wurden.

Das OSI-Referenzmodell wurde vor der Erfindung der entsprechenden Protokolle entwickelt. Diese Vorgehensweise bedeutet, dass das Modell nicht auf eine bestimmte Menge von Protokollen ausgelegt, sondern sehr allgemein war. Die Schattenseite dieser Vorgehensweise ist, dass die Entwickler nicht viel Erfahrung mit dem Thema und auch keine ausgereifte Vorstellung davon hatten, welche Schicht mit welcher Funktionalität ausgestattet werden sollte.

So verwaltete die Sicherungsschicht anfänglich z.B. nur Punkt-zu-Punkt-Netze. Als Broadcast-Netze entstanden, musste eine neue Teilschicht in das Modell eingefügt werden. Als die Leute außerdem begannen, echte Netze auf der Grundlage des OSI-Modells und vorhandener Protokolle zu entwickeln, stellte man (oh Wunder!) fest, dass sie nicht zu den erforderlichen Dienstspezifikationen passten. Aus diesem Grund mussten Konvergenzteilschichten auf das Modell aufgepropft werden, um hiermit die Unterschiede ausgleichen zu können. Außerdem erwartete das Gremium anfänglich, dass jedes Land ein Netz haben würde, das von der Regierung unter Verwendung der OSI-Protokolle verwaltet wird. Deshalb wurde an die Verbindung verschiedener Netze kein Gedanke verschwendet. Kurz: Die Dinge haben sich, wie wir wissen, völlig anders entwickelt.

Bei TCP/IP passierte das Gegenteil: Zuerst kamen die Protokolle, während das Modell eigentlich nur eine Beschreibung der vorhandenen Protokolle war. Mit der Einpassung der Protokolle in das Modell gab es keine Schwierigkeiten. Sie passten natürlich perfekt. Der einzige Ärger war, dass das *Modell* zu keinem anderen Protokollstapel passte. Folglich war es nicht besonders sinnvoll, damit TCP/IP-fremde Netze zu beschreiben.

Um von der philosophischen Betrachtung auf eine spezifischere zu wechseln, fällt als offensichtlicher Unterschied zwischen den zwei Modellen die Zahl der Schichten auf: Das OSI-Modell hat sieben, das TCP/IP-Modell fünf Schichten. Beide haben eine Vermittlungs- (bzw. Internet-), eine Transport- und eine Anwendungsschicht; der Rest ist unterschiedlich.

Ein weiterer Unterschied liegt in der verbindungslosen gegenüber der verbindungsorientierten Kommunikation. Das OSI-Modell unterstützt in der Vermittlungsschicht sowohl die verbindungslose als auch die verbindungsorientierte Kommunikation, auf der Transportschicht jedoch nur die verbindungsorientierte, weil das dort wichtig ist (da der Transportdienst für die Benutzer sichtbar ist). Das TCP/IP-Modell unterstützt in der Vermittlungsschicht nur einen Modus (verbindungslos), auf der Transportschicht aber beide Modi, sodass die Benutzer die Wahl haben. Diese Wahl ist besonders für einfache Anfrage-Antwort-Protokolle wichtig.

1.4.5 Kritik am OSI-Modell und OSI-Protokollen

Weder das OSI-Modell und seine Protokolle noch das TCP/IP-Modell und dessen Protokolle sind perfekt. Beide sind einer gewissen Kritik ausgesetzt. In diesem und dem folgenden Abschnitt betrachten wir einige dieser Kritiken. Wir beginnen mit OSI und prüfen danach das TCP/IP-Modell.

Als die zweite Auflage dieses Buchs erschien (1989), hatten die meisten Fachleute auf diesem Gebiet den Eindruck, dass das OSI-Modell und seine Protokolle die Welt beherrschen und alles andere verdrängen würden. Doch das ist nicht geschehen. Warum? Ein Blick zurück auf die Gründe kann hier einige Antworten liefern. Diese können folgendermaßen zusammengefasst werden:

- 1.** Schlechtes Timing
- 2.** Schlechte Technologie
- 3.** Schlechte Implementierungen
- 4.** Schlechte Politik

Schlechtes Timing

Betrachten wir zunächst den ersten Grund: schlechtes Timing. Der Zeitpunkt, zu dem ein Standard etabliert wird, ist für dessen Erfolg absolut entscheidend. David Clark vom MIT hat eine Theorie über Standards, die er *Apokalypse der zwei Elefanten* nennt und die in ► Abbildung 1.24 dargestellt wird.

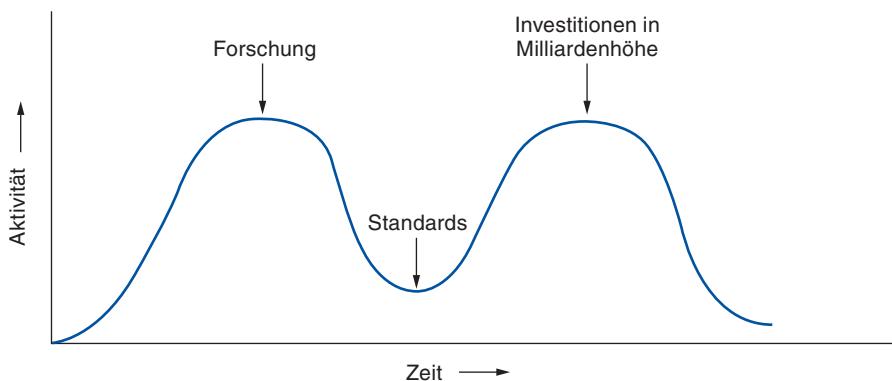


Abbildung 1.24: Die Apokalypse der zwei Elefanten.

Diese Abbildung zeigt den Umfang an Aktivitäten rund um ein neues Projekt. Zu Projektbeginn ist die Forschungsaktivität in Form von Diskussionen, Artikeln und Besprechungen am größten. Nach einer Weile lassen diese Aktivitäten nach, die Unternehmen realisieren das Projekt, und die Investitionen in Milliardenhöhe geraten ins Rollen.

Es ist überaus wichtig, dass die Standards in der Senke zwischen den zwei „Elefanten“ definiert werden. Wenn sie zu früh festgelegt werden (noch bevor die Forschungsergebnisse gut eingeführt sind), besteht eventuell ein zu geringes Verständnis für das Projekt, was zu schlechten Standards führt. Werden sie zu spät festgelegt, haben viele Unternehmen eventuell bereits auf eine andere Weise größere Investitionen in die Umsetzung bestimmter Vorhaben investiert, sodass die Standards dadurch übergangen werden. Ist der Abstand zwischen den zwei Elefanten sehr kurz (weil sich jeder beeilt, loszulegen), werden die Leute, die die Standards entwickeln, überrollt.

Es scheint, dass die OSI-Standardprotokolle überrollt wurden. Die konkurrierenden TCP/IP-Protokolle waren bei Forschungsinstituten und Universitäten bereits weitverbreitet, als die OSI-Protokolle erschienen. Während die Milliardeninvestitionen noch nicht in Bewegung geraten waren, war der akademische Markt doch groß genug, sodass viele Anbieter vorsichtig damit begannen, TCP/IP-Produkte anzubieten. Als OSI auftauchte, wollten sie einen zweiten Protokollstapel so lange nicht unterstützen, bis sie dazu gezwungen würden; also gab es keine anfänglichen Angebote. Da jedes Unternehmen wartete, dass ein anderes damit anfing, machte kein Unternehmen den Anfang, und OSI wurde nie Wirklichkeit.

Schlechte Technologie

Der zweite Grund, warum OSI nie Fuß fasste, ist der, dass sowohl das Modell als auch die Protokolle Schwachpunkte haben. Die Entscheidung zugunsten von sieben Schichten war mehr politisch als technisch. Zwei Schichten (Sitzungs- und Darstellungsschicht) sind nahezu leer, wohingegen andere (Sicherungs- und Vermittlungsschicht) übervoll sind.

Zusammen mit den relevanten Dienstdefinitionen und Protokollen ist das OSI-Modell außerordentlich komplex. Aufeinandergestapelt sind die gedruckten Standards ungefähr einen Meter hoch. Sie sind auch schwer zu implementieren und im Betrieb ineffizient. In diesem Zusammenhang wurde von Paul Mockapetris (zitiert in Rose, 1993) ein Rätsel aufgegeben:

F: Was erhalten Sie, wenn Sie einen Mafioso mit einem internationalen Standard kreuzen?

A: Jemand, der Ihnen ein Angebot macht, das Sie nicht verstehen.

Abgesehen davon, dass OSI schwer zu verstehen ist, hat OSI den weiteren Nachteil, dass einige Funktionen wie Adressierung, Flusskontrolle und Fehlerüberwachung in jeder Schicht erneut auftauchen. Saltzer et al. (1984) haben beispielsweise darauf hingewiesen, dass Fehlerüberwachung auf der höchsten Schicht greifen muss, um effektiv zu sein, sodass sich die Wiederholung auf allen darunterliegenden Schichten als unnötig und ineffizient erweist.

Schlechte Implementierungen

Angesichts der enormen Komplexität des Modells und seiner Protokolle muss es nicht verwundern, dass die ersten Implementierungen riesig, unhandlich und langsam waren. Jeder, der sich an ihnen versuchte, verbrannte sich die Finger. Es dauert nicht lange, bis die Leute „OSI“ mit „schlechter Qualität“ gleichsetzen. Die Produkte wurden zwar mit der Zeit besser, aber ein einmal gefestigtes Image lässt sich nur schwer korrigieren.

Demgegenüber war eine der ersten Implementierungen von TCP/IP Teil von Berkeley-Unix und ziemlich gut (ganz zu schweigen davon, dass das Produkt kostenlos war). Die Leute benutzten es bereitwillig. Dies führte zu einer großen Benutzergemeinde, was wiederum zu laufenden Verbesserungen führte, die eine noch größere Benutzergemeinde nach sich zogen. Hier verlief die Spirale nach oben, nicht nach unten.

Schlechte Politik

Aufgrund der ersten Implementierung betrachteten viele Leute, insbesondere in der akademischen Welt, TCP/IP als einen Teil von Unix – und eine Vorliebe für Unix gehörte in den 1980er Jahren unter den Akademikern schließlich zum guten Ton.

OSI galt demgegenüber als Geschöpf der europäischen Telekommunikationsministerien, der Europäischen Gemeinschaft und später der US-Regierung. Diese Ansicht stimmte nur teilweise. Doch allein die Vorstellung, dass ein Haufen von Regierungsbürokraten versuchte, den armen Entwicklern und Programmierern unten in den Schützengräben einen technisch unterlegenen Standard aufzudrängen, hat die Lage noch verschlimmert. Einige Leute betrachteten diese Entwicklung im gleichen Licht wie IBMs Ankündigung in den sechziger Jahren, dass PL/I die Sprache der Zukunft sei, oder die spätere Ankündigung vom US-Verteidigungsministerium, dass eigentlich Ada diese Rolle zukomme.

1.4.6 Kritik am TCP/IP-Referenzmodell

Das TCP/IP-Modell und die zugehörigen Protokolle haben auch gewisse Probleme. Erstens unterscheidet das Modell nicht deutlich zwischen den Konzepten Dienst, Schnittstelle und Protokoll. Gute Geplogenheiten bei der Softwareentwicklung setzen voraus, dass zwischen der Spezifikation und der Implementierung unterschieden wird. Teilweise macht das OSI sehr sorgfältig, TCP/IP aber überhaupt nicht. Folglich ist das TCP/IP-Modell kein guter Leitfaden zur Entwicklung neuer Netze mit neuen Technologien.

Zweitens ist das TCP/IP-Modell überhaupt nicht allgemein gehalten und schlecht zur Beschreibung eines Protokollstapels außer TCP/IP geeignet. Es ist beispielsweise völlig unmöglich, Bluetooth mit dem TCP/IP-Modell zu beschreiben.

Drittens ist die Netzzugangsschicht eigentlich keine Schicht im üblichen Sinn des Begriffs, wie dieser bei Protokollen mit Schichtenaufbau verwendet wird. Es handelt sich vielmehr um eine Schnittstelle (zwischen Vermittlung- und Sicherungsschichten). Die Unterscheidung zwischen einer Schnittstelle und einer Schicht ist aber wichtig und keiner sollte es sich leisten, salopp damit umzugehen.

Viertens unterscheidet das TCP/IP-Modell nicht zwischen der Bitübertragungs- und der Sicherungsschicht. Diese Schichten sind völlig unterschiedlich. Die Bitübertragungsschicht hat mit den Übertragungsmerkmalen von Kupferdraht, Glasfaser und drahtlosen Kommunikationsmedien zu tun. Die Aufgabe der Sicherungsschicht ist es, den Anfang und das Ende von Rahmen abzugrenzen und sie mit der gewünschten Zuverlässigkeit von einem Ende zum anderen zu befördern. Ein korrektes Modell sollte beide als separate Schichten beinhalten. Das TCP/IP-Modell tut das nicht.

Obwohl das IP- und TCP-Protokoll sorgfältig konzipiert und gut implementiert wurden, wurden viele andere Protokolle spontan produziert, meist von ein paar Studenten, die vor sich hin hackten, bis sie müde waren. Die Protokollimplementierungen wurden dann kostenlos verteilt, was dazu führte, dass sie viel benutzt wurden, inzwi-

schen fest eingebunkert sind und somit kaum ersetzt werden können. Einige davon sind heute eigentlich peinlich. Das virtuelle Terminalprotokoll TELNET wurde beispielsweise für ein mechanisches Telexterminal mit zehn Zeichen/Sekunde entwickelt. Es weiß nichts über grafische Benutzeroberflächen und Mäuse. Trotzdem ist es 30 Jahre später immer noch viel in Gebrauch.

1.5 Beispielnetze

Das Thema Computernetze bezieht sich auf viele verschiedene Netze, große und kleine, bekannte und weniger bekannte. Sie verfügen über verschiedene Ziele, Größen und Technologien. In den folgenden Abschnitten betrachten wir einige Beispiele, um eine Vorstellung von der Bandbreite zu erhalten, die im Bereich Computernetze zu finden ist.

Wir beginnen mit dem Internet, dem vermutlich bekanntesten Netzwerk, und betrachten dessen Geschichte, Evolution und Technologie. Dann beschäftigen wir uns mit dem mobilen Telefonnetz. Technisch gesehen unterscheidet es sich vom Internet vollkommen, sodass dies ein hübscher Gegensatz ist. Als Nächstes befassen wir uns mit IEEE 802.11, dem vorherrschenden Standard für drahtlose LANs. Schließlich werfen wir einen Blick auf RFID und Sensornetze – Technologien, die den Bereich der Netzwerke erweitern, um die physische Welt und alltägliche Objekte einzubeziehen.

1.5.1 Das Internet

Das Internet ist nicht wirklich ein Netz, sondern eine riesige Ansammlung von verschiedenen Netzen, die bestimmte gängige Protokolle nutzen und bestimmte allgemeine Dienste zur Verfügung stellen. Es ist ein ungewöhnliches System, das von niemandem geplant war und auch von niemandem kontrolliert wird. Um es besser zu verstehen, beginnen wir von vorne und betrachten, wie es entwickelt wurde und warum. Eine wundervolle Darstellung der Geschichte des Internets findet man bei John Naughton (2000). Es ist eines der seltenen Bücher, die man nicht nur mit Vergnügen liest, sondern das auch für den ernsthaften Historiker 20 Seiten mit Literatur- und Quellverweisen enthält. Einige der Informationen in diesem Abschnitt basieren auf diesem Buch.

Natürlich wurden über das Internet und seine Protokolle unzählige Bücher geschrieben. Weitere Informationen finden Sie beispielsweise in Maufer (1999).

Das ARPANET

Die Geschichte begann Ende der späten 1950er Jahre. Auf der Höhe des kalten Kriegs wollte das US-Verteidigungsministerium ein Kommando- und Steuernetz, das imstande sein sollte, einen Atomkrieg zu überdauern. Zu dieser Zeit nutzte die militärische Kommunikation das öffentliche Telefonnetz, das als verletzbar angesehen wurde. Der Grund hierfür ist aus ► Abbildung 1.25a ersichtlich. Hier stehen die schwarzen Punkte für die Telefonvermittlungsstellen, von denen jeder Tausende von Telefonen verband. Diese Vermittlungsstellen waren wiederum mit übergeordneten

Vermittlungsstellen (Fernvermittlungsstellen) verbunden und bildeten so eine landesweite Hierarchie mit nur geringer Redundanz. Die Verletzbarkeit dieses Systems lag darin, dass die Zerstörung einiger weniger Fernvermittlungsstellen das System in viele isolierte Inseln fragmentieren würde.

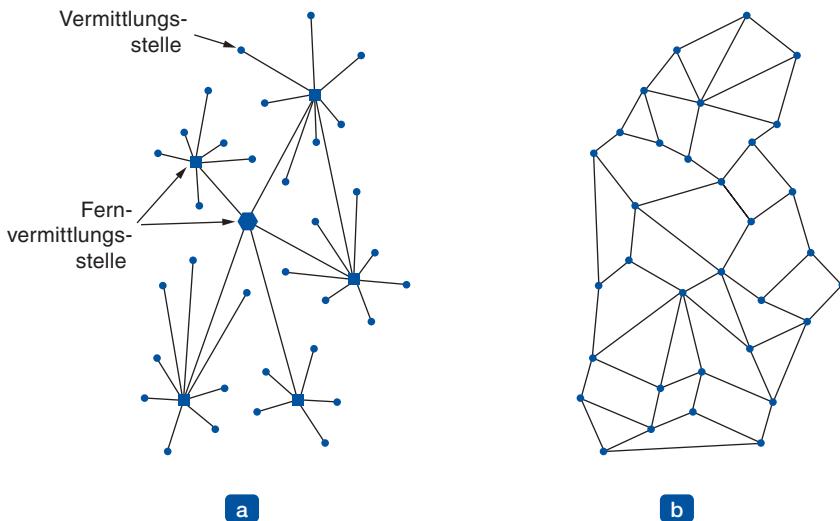


Abbildung 1.25: (a) Struktur des Telefonsystems. (b) Das von Baran vorgeschlagene verteilte Vermittlungssystem.

Um 1960 beauftragte das US-Verteidigungsministerium die RAND Corporation mit der Erarbeitung einer Lösung. Einer der Mitarbeiter, Paul Baran, stellte den hochgradig verteilten und fehlertoleranten Entwurf von ►Abbildung 1.25b vor. Da die Pfade zwischen zwei Vermittlungsstellen jetzt nicht sehr viel länger waren, als die analogen Signale ohne Störung fließen können, schlug Baran vor, digitale Paketvermittlung zu verwenden. Baran schrieb verschiedene Bereiche für das US-Verteidigungsministerium, in denen er seine Ideen im Detail darstellte (Baran, 1964). Den Vertretern des Pentagons gefiel das Konzept und man beauftragte AT&T – zu dieser Zeit der nationale amerikanische Telefonmonopolist – einen Prototyp zu erstellen. AT&T tat die Ideen von Baran kurzerhand ab. Das größte und reichste Unternehmen der Welt wollte sich nicht von einem jungen Klugscheißer sagen lassen, wie man ein Telefonnetz aufbaut. Man erklärte, das Telefonnetz von Baran könne nicht gebaut werden, und die Idee war damit gestorben.

Es vergingen einige Jahre und das amerikanische Verteidigungsministerium hatte immer noch kein besseres System. Um die nächste Entwicklung zu verstehen, müssen wir ganz zurückgehen, bis zum Oktober 1957, als die Sowjetunion die USA mit dem Start von Sputnik, dem ersten künstlichen Satelliten, im Weltall schlug. Als Präsident Eisenhower herauszufinden versuchte, wer hier mit offenen Augen geschlafen hatte, war er entsetzt, herauszufinden, dass sich Army, Navy und Air Force um das Forschungsbudget des Pentagons stritten. Seine sofortige Antwort war, eine einzige Forschungsorganisation für Verteidigungsprojekte zu gründen, die sogenannte **ARPA** (*Advanced Research Projects Agency*). ARPA hatte weder Wissenschaftler noch Labors. Sie bestand aus nichts als einem Büro und einem (nach Pentagon-Maßstäben)

kleinen Budget. Sie bestritt seine Aufgabe durch die Vergabe von Stipendien und Aufträgen von Verträgen an Universitäten und Unternehmen, deren Ideen vielversprechend aussahen.

In den ersten Jahren war ARPA noch damit beschäftigt, seine Aufgaben zu definieren. 1967 versuchte Larry Roberts, ein Programm-Manager von ARPA, herauszufinden, wie entfernter Zugriff auf Computer zur Verfügung gestellt werden könnte, und wandte sich dem Thema Netze zu. Er kontaktierte mehrere Experten und fragte sie um Rat. Einer davon, Wesley Clark, schlug vor, ein paketvermitteltes Subnetz zu errichten, das jeden Host mit seinem eigenen Router verbindet.

Nach anfänglicher Skepsis kaufte Roberts die Idee und präsentierte einen etwas vagen Artikel darüber im ACM SIGOPS Symposium on Operating System Principles, das in Gatlinburg, Tennessee, Ende 1967 stattfand (Roberts, 1967). Zu Roberts großem Erstaunen beschrieb ein anderer Artikel in der Konferenz ein ähnliches System, das nicht nur entworfen, sondern tatsächlich unter der Leitung von Donald Davies am National Physical Laboratory (NPL) in Großbritannien vollständig implementiert worden war. Das NPL-System war zwar kein landesweites System (es verband nur ein paar Rechner auf dem NPL-Campus), aber es zeigte, dass die Paketvermittlung praxistauglich war. Darauf hinaus zitierte dieser Artikel nun Barans frühere, verworfene Arbeit. Roberts kam von Gatlinburg mit dem Entschluss zurück, das System zu errichten, das später als **ARPANET** bekannt wurde.

Das Subnetz sollte aus Minicomputern namens **IMPs** (*Interface Message Processor*) bestehen, die über Übertragungsleitungen mit einer Kapazität von 56 kbit/s verbunden werden sollten. Um hohe Zuverlässigkeit sicherzustellen, sollte jeder IMP an mindestens zwei weitere IMPs angeschlossen werden. Das Subnetz sollte ein Datagrammnetz sein, sodass Nachrichten automatisch erneut über alternative Wege gesendet werden konnten, falls einige Leitungen und IMPs zerstört würden.

Jeder Netzknoten sollte aus einem IMP und einem Host bestehen, die im gleichen Raum stehen und über eine kurze Leitung verbunden werden sollten. Ein Host würde Nachrichten mit bis zu 8 063 Bit an seinen IMP senden, der diese in Pakete von höchstens 1 008 Bit aufteilen und unabhängig in Richtung Ziel befördern sollte. Jedes Paket würde vollständig vor der Weiterleitung empfangen werden, sodass das Subnetz das erste elektronische paketvermittelte Speichervermittlungsnetz (Store-and-forward-Paketvermittlungsnetz) der Welt werden sollte.

ARPA veranstaltete eine Ausschreibung zur Entwicklung des Subnetzes. Zwölf Unternehmen reichten ihre Angebote ein. Nach der Auswertung aller Angebote entschied sich ARPA für BBN, eine in Cambridge im US-Bundesstaat Massachusetts ansässige Beratungsfirma. Im Dezember 1968 wurde der Zuschlag für die Entwicklung des Subnetzes und der dazugehörenden Software erteilt. BBN wählte für diesen Auftrag speziell modifizierte Honeywell-Minicomputer vom Typ DDP-316 mit 12 000 16-Bit-Wörtern Kernspeicher als IMPs. Die IMPs hatten keine Platten, da auswechselbare Teile als unzuverlässig galten. Die IMPs wurden über 56-kbit/s-Leitungen, die von Telefongesellschaften angemietet wurden, verbunden. Obwohl heutzutage 56 kbit/s nur noch

in Ausnahmefällen verwendet werden, war es damals das Beste, was man für Geld bekommen konnte.

Die Software wurde zweigeteilt: Subnetz und Host. Die Software für das Subnetz bestand aus dem IMP-Teil der Host-IMP-Verbindung, dem IMP/IMP-Protokoll und einem Protokoll vom Quell- zum Ziel-IMP, um die Zuverlässigkeit zu erhöhen. Das ARPANET in seinem ersten Entwurf ist in ►Abbildung 1.26 dargestellt.

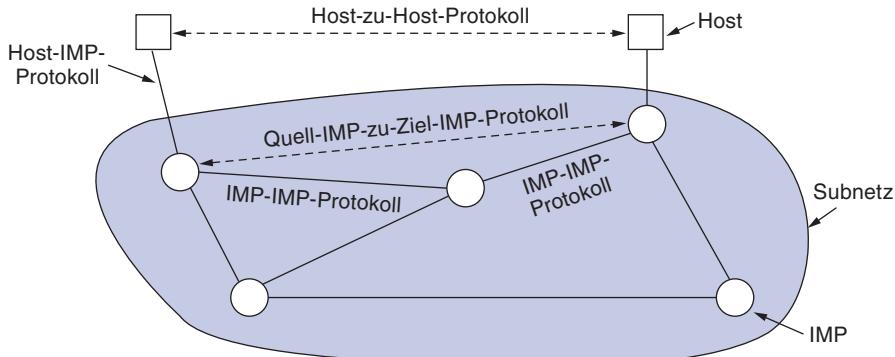


Abbildung 1.26: Der ursprüngliche Entwurf von ARPANET.

Außerhalb des Subnetzes war auch Software nötig: für den Host-Endpunkt der Host-IMP-Verbindung das Host-Host-Protokoll und die Anwendungssoftware. Bald wurde klar, dass BBN der Meinung war, dass seine Aufgabe getan sei, sobald eine Nachricht auf einer Host-IMP-Leitung angenommen und auf die Host-IMP-Leitung am Ziel zugestellt war.

Roberts hatte ein Problem: Auch der Host benötigte Software. Um das Problem mit der Host-Software zu lösen, berief er im Sommer 1969 in Snowbird in Utah eine Besprechung von Netzforschern, vorwiegend graduierten Studenten, ein. Die Studenten erwarteten einen Netzfachmann, der ihnen den Entwurf des Netzes und seine Software erklären und dann jedem seinen Teil an dem Projekt zuweisen würde. Sie waren verblüfft, als sie feststellten, dass es weder einen Netzfachmann noch einen nennenswerten Entwurf gab. Sie sollten sich selbst etwas einfallen lassen.

So wurde im Dezember 1969 ein experimentelles Netzwerk mit vier Knoten in Betrieb genommen: an der Universität Los Angeles (UCLA), der Universität Santa Barbara (UCSB), am Stanford Research Institute (SRI) und an der Universität von Utah. Diese vier wurden ausgewählt, weil alle eine große Anzahl von ARPA-Verträgen besaßen und auch verschiedene und völlig inkompatible Hostrechner im Einsatz hatten (um das Projekt spaßiger zu gestalten). Die erste Host-zu-Host-Nachricht war zwei Monate früher von dem UCLA-Knoten von einem Team, das von Len Kleinrock (einem Pionier der Paketvermittlungstheorie) angeführt wurde, zum SRI-Knoten gesendet worden. Das Netz wuchs schnell und immer mehr IMPs wurden ausgeliefert und installiert. Bald überspannte es die Vereinigten Staaten. ►Abbildung 1.27 zeigt, wie schnell das ARPANET in den ersten drei Jahren wuchs.

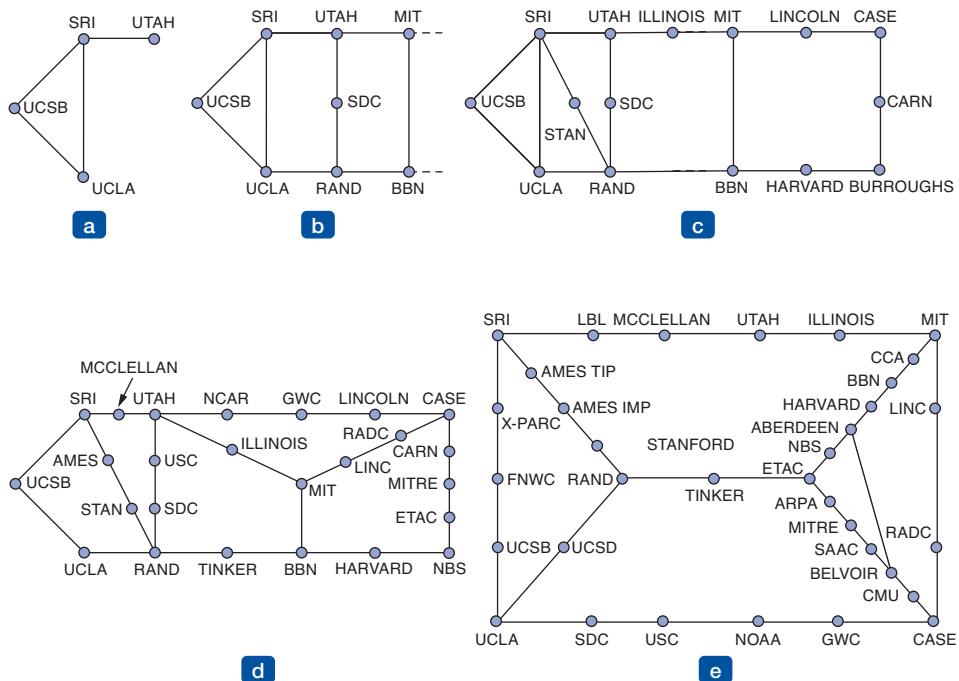


Abbildung 1.27: Wachstum des ARPANET: a) Dezember 1969; (b) Juli 1970; (c) März 1971; (d) April 1972; (e) September 1972.

Abgesehen von der Unterstützung des flügig gewordenen ARPANET, bezuschusste ARPA auch die Forschung von satellitengestützten und mobilen paketvermittelten Funknetzen. In einer spektakulären Demonstration fuhr ein Lastwagen durch ganz Kalifornien und nutzte das paketvermittelte Funknetz, um Nachrichten an SRI zu senden, die dann über das ARPANET an die Ostküste weitergeleitet wurden, von wo sie über Satellitennetz an das University College London (UCL) übertragen wurden. Dabei konnte ein im Lkw mitfahrender wissenschaftlicher Mitarbeiter einen Computer in London auf seiner Fahrt in Kalifornien nutzen.

Dieses Experiment zeigte auch auf, dass die vorhandenen ARPANET-Protokolle nicht für den Betrieb über unterschiedliche Netze geeignet waren. Diese Beobachtung führte zu weiteren Forschungsarbeiten über Protokolle, die schließlich in der Erfindung des TCP/IP-Modells und der entsprechenden Protokolle gipfelte (Cerf und Kahn, 1974). TCP/IP wurde speziell zur Verwaltung der Kommunikation über Internetworks entwickelt, was immer wichtiger wurde, da mehr und mehr Netze an das ARPANET angegeschlossen wurden.

Um die Übernahme dieser neuen Protokolle voranzutreiben, erteilte ARPA mehrere Aufträge, TCP/IP auf verschiedenen Computerplattformen zu implementieren, unter anderem für IBM-, DEC- und HP-Systeme sowie für Berkeley-UNIX. Wissenschaftliche Mitarbeiter an der kalifornischen Universität in Berkeley schrieben TCP/IP mit einer neuen Programmierschnittstelle ([Socket](#)) für die aufkommende 4.2BSD-Version von Berkeley-UNIX um.

Sie schrieben außerdem viele Anwendungen, Hilfs- und Verwaltungsprogramme, um zu zeigen, wie bequem es war, das Netz mithilfe der Sockets zu nutzen.

Das war perfektes Timing. Viele Universitäten hatten eben erst einen zweiten oder dritten VAX-Rechner angeschafft sowie ein LAN, um diese miteinander zu verbinden, hatten aber noch keine Netzsoftware. Als 4.2BSD mit TCP/IP, Sockets und vielen Netzdienstprogrammen herauskam, wurde das Komplett Paket mit offenen Armen aufgenommen. Mit TCP/IP erwies es sich außerdem als einfach, LANs an das ARPANET anzuschließen, was viele auch taten.

In den 1980er Jahren wurden weitere Netze, vor allem LANs, an das ARPANET angeschlossen. Je mehr der Umfang zunahm, umso aufwendiger wurde es, die Hosts zu finden. Daher wurde das **DNS** (*Domain Name System*) entwickelt, damit Rechner in Domänen organisiert und Host-Namen IP-Adressen zugeordnet werden konnten. Seitdem hat sich DNS zu einem allgemeinen, verteilten Datenbanksystem zum Speichern verschiedener Informationen in Bezug auf Namen entwickelt. Dies wird ausführlich in *Kapitel 7* untersucht.

NSFNET

Gegen Ende der 1970er Jahre erkannte die NSF (U.S. National Science Foundation) die enorme Wirkung des ARPANET auf Forschungen an Universitäten, denn Wissenschaftler konnten kreuz und quer über das ganze Land Daten austauschen und an Forschungsprojekten zusammenarbeiten. Um sich allerdings in das ARPANET einzuklinken, musste eine Universität einen Forschungsvertrag mit dem US-Verteidigungsministerium haben. Viele hatten keinen Vertrag. Die anfängliche Reaktion der NSF war 1981 die Bezuschussung des **CSNET** (*Computer Science Network*). Dieses Netz verband Informatikfachbereiche und industrielle Forschungszentren mit dem ARPANET über Einwahlstandleitungen. In den späten 1980er Jahren ging die NSF einen Schritt weiter und entschied sich, einen Nachfolger zum ARPANET zu entwickeln, der für alle Forschungsgruppen an Universitäten offen sein sollte.

Um für den Anfang etwas Konkretes zur Hand zu haben, baute die NSF ein Backbone-Netz, um ihre sechs Superrechenzentren in San Diego, Boulder, Champaign, Pittsburgh, Ithaca und Princeton zu verbinden. Jeder Supercomputer erhielt einen kleinen Bruder, der aus einem Mikrocomputer LSI-11 bestand, den man **Fuzzball** nannte. Diese Fuzzballs wurden über 56-kbit/s-Standleitungen verbunden und bildeten das Subnetz. Hardwareseitig wurde also die gleiche Technologie angewandt wie beim ARPANET. Die Software war jedoch völlig anders: Die Fuzzballs sprachen von Anfang an TCP/IP und bildeten damit das erste TCP/IP-WAN.

Die NSF bezuschusste außerdem einige (zum Schluss 20) regionale Netze, die an das Backbone angeschlossen wurden, um Benutzern an Tausenden von Universitäten, Forschungseinrichtungen, Bibliotheken und Museen Zugriff auf jeden der Supercomputer zu ermöglichen und untereinander zu kommunizieren. Das komplexe Netz, einschließlich des Backbones und der regionalen Netze, wurde **NSFNET** genannt. Es bot Zugang zum ARPANET über eine Verbindung zu einem IMP und einem Fuzzball im

Rechenzentrum der Carnegie-Mellon-Universität. Das erste NSFNET-Backbone ist in ▶ Abbildung 1.28 dargestellt.

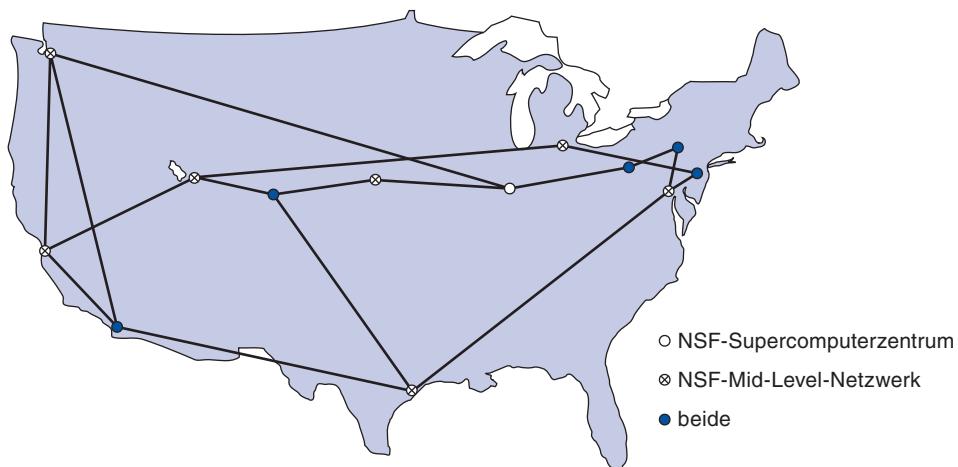


Abbildung 1.28: Das NSFNET-Backbone, USA, im Jahr 1988.

NSFNET war sofort ein durchschlagender Erfolg – und von Anfang an überladen. Die NSF begann unverzüglich mit der Planung eines Nachfolgers und erteilte nach einer Ausschreibung dem in Michigan ansässigen MERIT-Konsortium den Zuschlag als Betreiber des Netzes. Glasfaserkanäle mit 448 kbit/s wurden von MCI (einer amerikanischen Telefongesellschaft, die mit WorldCom fusionierte) angemietet, um das Backbone Version 2 bereitzustellen. Als Router wurden PC-RTs von IBM eingesetzt. Auch dieses Netz war bald überladen, sodass bis 1990 das zweite Backbone auf 1,5 Mbit/s aufgerüstet wurde.

Das Wachstum setzte sich in raschem Tempo fort und die NSF erkannte, dass die Regierung nicht für immer und ewig Netze finanzieren konnte. Außerdem wollten Privatunternehmen in den Bereich einsteigen, wurden aber durch das Monopol der NSF, das durch deren Finanzierung der Projekte entstand, daran gehindert. Als Folge regte die NSF MERIT, MCI und IBM an, ein gemeinnütziges Unternehmen zu gründen. Es entstand **ANS** (*Advanced Networks and Services*) als erster Schritt auf dem Weg zur Kommerzialisierung. 1990 übernahm ANS das NSFNET und rüstete die 1,5-Mbit/s-Leitungen auf 45 Mbit/s auf. Damit war das **ANSNET** geboren. Dieses Netzwerk war fünf Jahre in Betrieb und wurde dann an America Online verkauft. Aber zu dieser Zeit boten bereits verschiedene Unternehmen kommerzielle IP-Dienste an und es war klar, dass sich der Staat nun aus dem Netzgeschäft zurückziehen sollte.

Um den Übergang zu erleichtern und sicherzustellen, dass jedes regionale Netz mit jedem anderen regionalen Netz kommunizieren konnte, erteilte die NSF Aufträge an vier verschiedene Netzbetreiber für die Errichtung eines **Netzzugangspunkts (NAP)**, *Network Access Point*. Diese Betreiber waren PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, DC) und Sprint (New York City, wobei für NAP-Zwecke der Ort Pennsauken in New Jersey als Teil von New York City zählte). Jeder Netz-

betreiber, der Backbonedienste für die regionalen NSF-Netze bereitstellen wollte, musste sich an alle NAPs anschließen.

Diese Anordnung bedeutete, dass ein Paket, das von irgendeinem regionalen Netz ausging, den Backbonebetreiber aussuchen konnte. Folglich standen die Backbonebetreiber im Wettbewerb um ein regionales Netzgeschäft, der auf der Grundlage von Service und Preis ausgetragen wurde – was natürlich der Grundgedanke war. In der Folge wurde das Konzept eines einzigen Standard-Backbones durch eine kommerziell ausgerichtete wettbewerbsorientierte Infrastruktur ersetzt. Viele Leute kritisieren die US-Regierung gerne, dass sie nicht innovativ genug sei, aber beim Betrieb von Netzen waren das US-Verteidigungsministerium und die NSF federführend bei der Erstellung der Infrastruktur, die die Basis für das Internet bildete, und man übergab dies dann an die Industrie zum weiteren Betrieb.

In den 1990er Jahren haben viele Länder und Regionen nationale Forschungsnetze aufgebaut, die oftmals nach dem Muster von ARPANET und NSFNET ausgelegt waren. Hierzu gehörten EuropaNET und EBONE in Europa, die mit 2-Mbit/s-Standleitungen begannen und dann auf Leitungen mit einer Kapazität von 34 Mbit/s aufgerüstet wurden. Auch in Europa ging die Netzinfrastruktur in die Hände der Industrie über.

Das Internet hat sich seit diesen frühen Tagen deutlich gewandelt. Mit dem Aufkommen des World Wide Web (WWW) in den frühen 1990er Jahren ist seine Größe nahezu explodiert. Aktuelle Zahlen des Internet Systems Consortium gehen von über 600 Millionen sichtbaren Internethosts aus. Diese Schätzung ist bewusst zu niedrig veranschlagt, aber die Zahl übersteigt bei Weitem die paar Millionen Hosts von 1994, als die erste WWW-Konferenz am CERN abgehalten wurde.

Auch die Art und Weise, wie wir das Internet nutzen, hat sich radikal geändert. Anfangs herrschten Anwendungen wie E-Mail-für-Akademiker, Newsgroups, entferntes Einloggen und Dateiübertragungen vor. Später verschob sich der Fokus zu E-Mail-für-Alle, dann zu Web- und Peer-to-Peer-Anwendungen zum Verbreiten von Inhalten (wie das heute abgeschaltete Napster). Heute sind das Verbreiten von Media in Echtzeit, soziale Netze (z.B. Facebook) und Mikroblogging (z.B. Twitter) gängige Anwendungen. Dieser Wechsel brachte reichhaltigere Medienarten und daher sehr viel mehr Datenverkehr mit sich. Tatsächlich scheint sich die Art des jeweils vorherrschenden Datentransports im Internet mit einer gewissen Regelmäßigkeit zu ändern, beispielsweise wenn neue und bessere Möglichkeiten der Musik- oder Filmbearbeitung sehr beliebt werden.

Die Internetarchitektur

Die Architektur des Internets hat sich im Zuge des explosionsartigen Wachstums ebenfalls stark gewandelt. In diesem Abschnitt versuchen wir, eine kurze Übersicht darüber zu geben, wie das Internet heute aussieht. Dieses Bild verkompliziert sich durch die ständigen Umwälzungen im Bereich der Telefongesellschaften, Kabelunternehmen und ISPs, wodurch es oftmals schwer zu sagen ist, wer was macht. Eine treibende Kraft dieser Umwälzungen ist das Zusammenwachsen in der Telekommunikation in dem Sinne, dass Netze eingesetzt werden, die vorher anderen Benutzungsarten

dienten. Beispielsweise kaufen Sie von einem Unternehmen in einem „Triple-Play“-Paket Telefon-, Fernsehen- und Internetdienste über dieselbe Netzverbindung in der Annahme, dass Sie damit Geld sparen. Daher ist die Beschreibung, die wir hier geben, etwas einfacher als die realen Verhältnisse. Und was heute gilt, kann morgen schon nicht mehr wahr sein. ►Abbildung 1.29 zeigt die große Übersicht. Lassen Sie uns diese Darstellung einmal Stück für Stück untersuchen, angefangen bei einem Computer zu Hause (an den Rändern der Abbildung). Um sich an das Internet anzuschließen, ist der Computer mit einem **Internetdienstanbieter (ISP, Internet Service Provider)** verbunden, von dem der Benutzer **Internetzugang (access)** oder **-konnektivität (connectivity)** erwirbt. Damit ist der Rechner in der Lage, Pakete mit allen anderen erreichbaren Hosts im Internet auszutauschen. Der Nutzer könnte Pakete senden, um im Web zu surfen oder für eine der tausend anderen Nutzungsmöglichkeiten, das spielt keine Rolle. Es gibt viele Arten von Internetzugriff und sie werden in der Regel danach unterschieden, wie viel Bandbreite sie zur Verfügung stellen und wie viel sie kosten, aber die wichtigste Eigenschaft ist die Verbindungsfähigkeit.

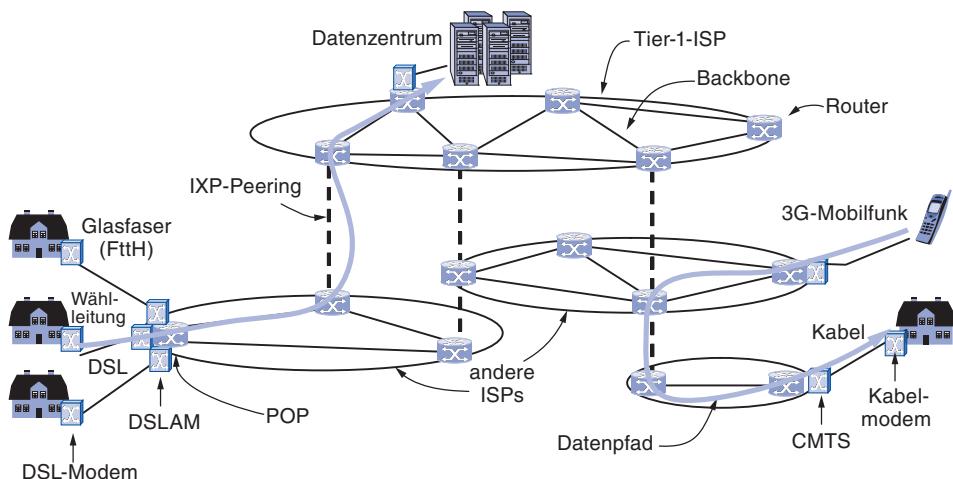


Abbildung 1.29: Überblick über die Internetarchitektur.

Häufig wird zum Anschluss an einen ISP die Telefonleitung zu Ihrem Haus genutzt, in diesem Fall ist Ihre Telefongesellschaft auch Ihr ISP. **DSL (Digital Subscriber Line)** benutzt die Telefonleitungen, die Ihr Haus verbindet, zur digitalen Datenübertragung. Der Rechner ist mit einem **DSL-Modem** verbunden, einem Gerät zur Umwandlung von digitalen Paketen in analoge Signale, welche ungehindert die Telefonleitung passieren können. Am anderen Ende konvertiert ein Gerät namens **DSLAM (Digital Subscriber Line Access Multiplexer)** die Signale wieder in Pakete.

In Abbildung 1.29 sind mehrere andere Wege dargestellt, sich an einen ISP anzuschließen. DSL benutzt ein Frequenzband der lokalen Telefonleitung, das oberhalb des für Sprachübertragung verwendeten Bands liegt. Somit werden nicht einfach – wie bei der **Wählleitung (dial-up)** – Bits anstelle von Sprache über eine traditionelle Telefonverbindung gesendet. Bei der Wählleitung wird eine andere Art von Modem

an beiden Enden eingesetzt. Das Wort **Modem** ist ein Akronym aus „*Modulator Demodulator*“ und bezieht sich auf jedes Gerät, das zwischen digitalen Bits und analogen Signalen konvertiert.

Eine weitere Methode ist das Senden von Signalen über das Kabelfernsehsystem. Wie DSL ist dies eine Möglichkeit, vorhandene Infrastrukturen zu nutzen, in diesem Fall ansonsten ungenutzte Kabelfernsehkanäle. Das Endgerät im Haus heißt **Kabelmodem** und das Gerät am Kabelkopfende heißt **Kabelmodemabschlusssystem (CMTS, Cable Modem Termination System)**.

DSL und Kabel bieten Internetanschluss zu Übertragungsraten von einem kleinen Bruchteil eines Megabit/s bis hin zu mehreren Megabit/s, je nach System. Diese Übertragungsraten sind deutlich größer als bei Wählleitungen, welche aufgrund ihrer geringen Bandbreite, die für Sprachübertragungen benutzt wird, auf 56 kbit/s begrenzt sind. Ein Internetzugang mit deutlich größeren Übertragungsraten als bei Wählleitungen wird **Breitbandzugang** genannt. Der Name bezieht sich auf die breitere Bandbreite, die für schnellere Netze eingesetzt wird, und nicht auf eine bestimmte Geschwindigkeit.

Die bisher erwähnten Zugangsmethoden sind durch die Bandbreite der „letzten Meile“ oder letzten Etappe der Übertragung begrenzt. Wenn Glasfaserkabel zu Wohnhäusern verlegt werden, dann kann schneller Internetzugang zu Übertragungsraten in der Größenordnung von 10 bis 100 Mbit/s angeboten werden. Dieser Entwurf heißt **FttH (Fiber to the Home)**. Für Betriebe in Gewerbegebieten kann es sinnvoll sein, eine Leitung mit hoher Übertragungsgeschwindigkeit vom Büro zum nächstgelegenen ISP zu mieten. In Nordamerika läuft beispielsweise eine T3-Leitung mit ungefähr 45 Mbit/s.

Auch beim Internetzugang gibt es drahtlose Anwendungen. Ein Beispiel, das wir (im folgenden Abschnitt noch) kurz untersuchen werden, sind die 3G-Mobilfunknetze. Diese können Datenübertragungsraten von 1 Mbit/s oder höher zu Mobiltelefonen und stationären Teilnehmern im Abdeckungsbereich bieten.

Wir können jetzt also Pakete zwischen dem Haus und dem ISP übertragen. Der Ort, an dem Kundenpakete das ISP-Netz betreten, heißt **Übergabepunkt (POP, Point of Presence)** des ISP. Als Nächstes werden wir erklären, wie Pakete zwischen den POPs der verschiedenen ISPs übertragen werden. Ab diesem Punkt ist das System vollständig digital und paketvermittelt.

ISP-Netze können sich regional, national oder international ausdehnen. Wir haben bereits gesehen, dass ihre Architektur aus Fernübertragungsleitungen aufgebaut ist, welche Router an POPs in den verschiedenen Städten, die dieser ISP bedient, miteinander verbinden. Diese Anlage heißt das **Backbone** des ISP. Wenn ein Paket für einen Host bestimmt ist, der direkt von dem ISP bedient wird, dann wird dieses Paket über das Backbone geleitet und dem Host zugestellt. Ansonsten muss es einem anderen ISP übergeben werden.

ISPs verbinden ihre Netze, um Datenverkehr an **Internetknoten (IXP, Internet eXchange Point)** auszutauschen. Das Verbinden der ISPs untereinander wird **Peering** genannt. Es gibt auf der ganzen Welt viele IXPs in Städten. Sie sind in Abbildung 1.29 vertikal ein-

gezeichnet, da sich ISP-Netze geografisch überlappen. Im Prinzip ist ein IXP ein Raum voll mit Routern, pro ISP mindestens einer. Ein LAN in dem Raum verbindet all die Router, sodass Pakete von jedem ISP-Backbone zu jedem anderen ISP-Backbone weitergeleitet werden können. IXPs können große Einrichtungen sein, die voneinander unabhängige Eigentümer haben. Eines der größten IXPs ist die Amsterdam Internet Exchange, mit der Hunderte von ISPs verbunden sind und über die Hunderte von Gigabyte/s an Datenverkehr ausgetauscht werden.

Wie das Peering an den IXPs genau vorstatten geht, hängt von den Geschäftsbeziehungen zwischen den ISPs ab. Er gibt viele mögliche Beziehungen. Zum Beispiel könnte ein kleiner ISP einen größeren für Internetanschluss bezahlen, um entfernte Hosts zu erreichen, so wie ein Kunde für den Dienst eines Internetproviders bezahlt. In diesem Fall sagt man, dass der kleine ISP für den **Transit** bezahlt. Zwei große ISPs könnten dagegen den gegenseitigen Datenaustausch so organisieren, dass jeder ISP einen gewissen Datenverkehr an den anderen ISP liefert, ohne für den Transit zu bezahlen. Eines der vielen Paradoxe des Internets ist, dass ISPs, die öffentlich um Kunden konkurrieren, im nicht öffentlichen Bereich als Peering-Partner zusammenarbeiten (Metz, 2001).

Der Pfad eines Pakets durch das Internet hängt von der Wahl der Peering-Partner des ISP ab. Falls der ausliefernde ISP ein Peering-Partner vom Ziel-ISP ist, so kann das Paket direkt zu dem Peer geliefert werden. Andernfalls muss das Paket an den nächsten Ort geleitet werden, an dem es eine Verbindung zu einem bezahlten Transit-Provider gibt, der wiederum das Paket ausliefern kann. Zwei ISP-Beispelpfade sind in Abbildung 1.29 eingezeichnet. Oftmals ist der Pfad, den ein Paket nimmt, nicht der kürzeste Weg durch das Internet.

Am oberen Ende der Nahrungskette sitzt eine kleine Handvoll Unternehmen wie AT&T und Sprint, die große internationale Backbone netze mit Tausenden von Routern betreiben, welche über Glasfaserkabeln mit hoher Bandbreite miteinander verbunden sind. Diese ISPs bezahlen nicht für den Transit. Sie werden in der Regel **Tier-1**-ISPs genannt und werden als das Backbone des Internets bezeichnet, da sich jeder mit ihnen verbinden muss, um das gesamte Internet erreichen zu können.

Unternehmen, die wie z.B. Google und Yahoo! viel Inhalt zur Verfügung stellen, stellen ihre Rechner in **Datenzentren** auf, die eine gute Verbindung zum Rest des Internets aufweisen. Diese Datenzentren sind für Computer, nicht für Menschen entworfen worden. In einigen Zentren stapeln sich die Maschinen in vielen Regalen, diese werden **Serverfarmen** genannt. Datenzentren mit **Server-Housing** (*colocation*) oder **Hosting** ermöglichen ihren Kunden, Server an ISP-POPs aufzustellen, sodass kurze, schnelle Verbindungen zwischen den Servern und dem ISP-Backbone hergestellt werden können. Der Bereich des Internet-Hostings ist immer mehr virtualisiert worden, sodass es heute üblich ist, eine virtuelle Maschine zu mieten, die in einer Serverfarm läuft, anstatt einen physischen Rechner zu installieren. Diese Datenzentren sind so groß (Zigtausende von Maschinen), dass Strom ein Hauptkostenfaktor ist. Deshalb werden Datenzentren manchmal in Gegenden gebaut, in denen Strom billig ist.

Damit endet unsere kurze Betrachtung des Internets. In den folgenden Kapiteln wird noch ausführlich auf die einzelnen Komponenten und deren Entwurf, die Algorithmen und Protokolle eingegangen. Ein weiterer erwähnenswerter Punkt hier ist, dass sich auch die Bedeutung dessen verändert, was es heißt, im Internet zu sein. Früher war ein Rechner dann im Internet, wenn er (1) den TCP/IP-Protokollstapel ausführte, (2) eine IP-Adresse hatte und (3) IP-Pakete an alle anderen Maschinen im Internet schicken konnte. ISPs verwenden jedoch häufig IP-Adressen erneut, je nachdem welche Rechner zurzeit benutzt werden, und Heimnetze verwenden häufig eine IP-Adresse für mehrere Computer. Diese Praxis untergräbt also die zweite Bedingung. Sicherheitsmaßnahmen wie Firewalls können unter Umständen den Empfang von Paketen blockieren, was die dritte Bedingung untergräbt. Trotz dieser Schwierigkeiten ist es sinnvoll, auch bei solchen Rechnern davon zu sprechen, dass sie im Internet sind, solange sie mit ihren ISPs verbunden sind.

Auch sollte noch nebenbei erwähnt werden, dass manche Unternehmen all ihre vorhandenen internen Netze miteinander verbunden haben und dabei oftmals die gleiche Technologie wie das Internet verwenden. Auf diese **Intranets** kann normalerweise nur innerhalb des Firmengeländes oder von Firmenrechnern aus zugegriffen werden. Sie funktionieren aber nach dem gleichen Prinzip wie das Internet.

1.5.2 Mobilfunknetze der dritten Generation

Die Leute lieben es, am Telefon miteinander zu sprechen, sogar mehr noch als im Internet zu surfen – und deshalb wurde das Mobilfunknetz zum erfolgreichsten Netz der Welt. Es hat weltweit mehr als vier Milliarden Teilnehmer. Um diese Zahl in ein Verhältnis zu setzen: das ist ungefähr 60 % der Weltbevölkerung und mehr als die Anzahl der Internethosts und Festnetzanschlüsse zusammen (ITU, 2009).

Parallel zum unglaublichen Wachstum der Mobilfunknetze hat sich im Laufe der letzten 40 Jahre auch deren Architektur stark gewandelt. Mobiltelefone der ersten Generation übertrugen Sprache als ständig variierende (analoge) Signale, nicht als (digitale) Bitfolgen. **AMPS** (*Advanced Mobile Phone System*), welches in den USA im Jahr 1982 eingesetzt wurde, war ein weitverbreitetes System der ersten Generation. Mobiltelefonsysteme der zweiten Generation (2G) wechselten für die Sprachübertragung auf die digitale Form, um die Kapazität zu erhöhen, die Sicherheit zu verbessern und Textnachrichten zu ermöglichen. **GSM** (*Global System for Mobile communications*), welches ab 1991 eingesetzt wurde und das am weitesten verbreitete Mobiltelefonsystem in der Welt darstellt, ist ein 2G-System.

Die Systeme der dritten Generation (oder 3G-Systeme) werden seit dem Jahr 2001 eingesetzt und bieten sowohl digitalen Sprach- als auch digitalen Datenservice. Sie brachten außerdem viel Fachsprache und viele unterschiedlichen Standards mit, aus denen man auswählen konnte. Die ITU (ein internationales Standardisierungsgremium, welches wir im nächsten Abschnitt besprechen werden) definiert 3G lose so, dass Übertragungsraten von mindestens 2 Mbit/s für stationäre oder zu Fuß gehende Benutzer und 384 kbit/s in einem sich bewegenden Fahrzeug bereitgestellt werden. **UMTS** (*Universal*

Mobile Telecommunications System), auch WCDMA (Wideband Code Division Multiple Access) genannt, ist das wichtigste 3G-System, das schnell weltweiten Einsatz gefunden hat. Es kann bis zu 14 Mbit/s auf dem Downlink und fast 6 Mbit/s auf dem Uplink bereitstellen. Zukünftige Versionen werden mehrere Antennen und Funkwellen verwenden, um den Nutzern noch größere Geschwindigkeit zu bieten.

Die knappe Ressource in 3G-Systemen ist – wie vorher in den 2G- und 1G-Systemen – das Funkwellenspektrum. Regierungen vergeben den Mobiltelefonbetreibern Lizenzen für das Recht, Teile des Spektrums zu verwenden. Häufig geschieht dies über eine Auktion, in der Netzbetreiber Gebote abgeben. Der Besitz eines Teils von einem lizenzierten Spektrum vereinfacht den Entwurf und den Betrieb von Systemen, da niemand anderes auf diesem Spektrum senden darf, doch oft kostet dies eine erhebliche Geldsumme. Beispielsweise wurden im Jahr 2000 in Großbritannien fünf 3G-Lizenzen für zusammen ungefähr 38 Milliarden Euro versteigert.

Diese Knappheit an Spektrum führte zu dem **zellulären Netzentwurf** (►Abbildung 1.30), der heute für Mobilfunknetze verwendet wird. Um die Funkinterferenzen zwischen Nutzern zu handhaben, wird der Abdeckungsbereich in Zellen unterteilt. Innerhalb einer Zelle werden den Nutzern Kanäle zugewiesen, die sich nicht gegenseitig stören und die nicht zu viele Interferenzen für die benachbarten Zellen verursachen. Dies ermöglicht eine gute Wiederverwendung des Spektrums oder **Frequenzwiederholung** in den Nachbarzellen, wodurch die Kapazität des Netzes erhöht wird. In 1G-Systemen, bei denen jede Sprachübertragung auf einem bestimmten Frequenzband lag, waren die Frequenzen sorgfältig ausgewählt, sodass sie nicht mit benachbarten Zellen in Konflikt kamen. Auf diese Weise konnte eine gegebene Frequenz nur jeweils einmal in mehreren Zellen wiederverwendet werden. Bei modernen 3G-Systemen kann jede Zelle alle Frequenzen benutzen, und zwar so, dass die Interferenzen in den Nachbarzellen tolerierbar bleiben. Variationen des zellulären Entwurfs umfassen den Einsatz von direktionalen oder in Sektoren eingeteilte Antennen auf Mobilfunkmasten, um Interferenzen weitergehend zu reduzieren, doch die Grundidee bleibt dieselbe.

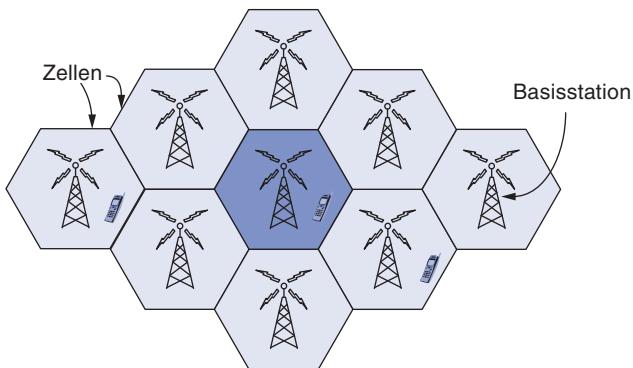


Abbildung 1.30: Zellulärer Entwurf der Mobilfunknetze.

Die Architektur des Mobilfunknetzes unterscheidet sich stark von der Internetarchitektur. Sie besteht aus mehreren Teilen, wie in einer vereinfachten Version der UMTS-

Architektur in ► Abbildung 1.31 gezeigt ist. Zunächst gibt es die **Luftschnittstelle** (*air interface*). Dieser Begriff ist ein Fantasiename für das Funkkommunikationsprotokoll, das in der Luft zwischen dem mobilen Gerät (z.B. dem Handy) und der **zellulären Basisstation** benutzt wird. Fortschritte bei der Luftschnittstelle in den letzten Jahrzehnten konnten die drahtlosen Datenraten erheblich steigern. Die UMTS-Luftschnittstelle basiert auf dem **Codemultiplexverfahren (CDMA, Code Division Multiple Access)**, eine Technik, die wir in Kapitel 2 untersuchen werden.

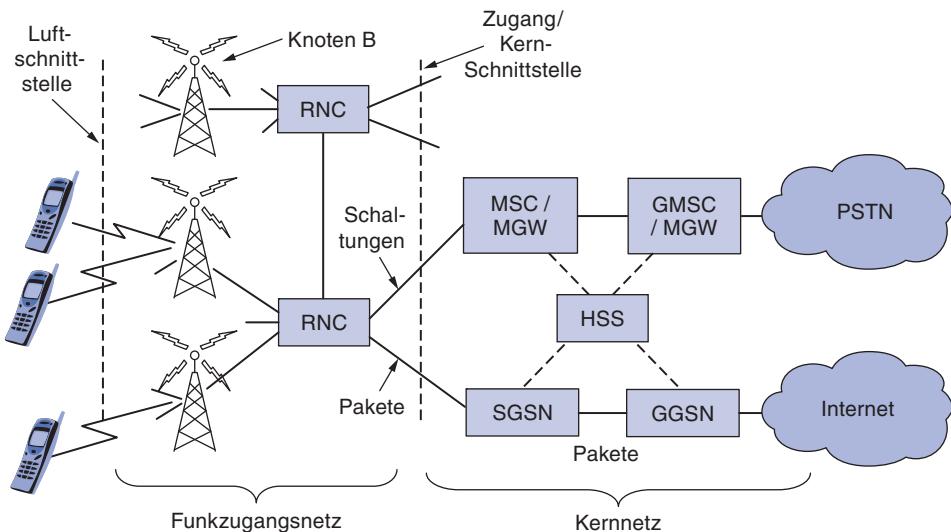


Abbildung 1.31: Architektur des 3G-Mobilfunknetzes UMTS.

Die zelluläre Basisstation zusammen mit seinen Steuerungseinheiten bildet das **Funkzugangsnetz** (*radio access network*). Dieser Teil stellt die drahtlose Seite des Mobilfunknetzes dar. Der Steuerungsknoten oder **RNC** (*Radio Network Controller*) kontrolliert die Verwendung des Spektrums. Die Basisstation implementiert die Luftschnittstelle. Sie wird **Node B** genannt – eigentlich ein provisorischer Name, der geblieben ist.

Der Rest des Mobilfunknetzes, das **Kernnetz** (*core network*), transportiert den Datenverkehr für das Funkzugangsnetz. Das UMTS-Kernnetz hat sich aus dem Kernnetz entwickelt, welches für den Vorgänger, das 2G-GSM-Netz, benutzt wurde. Und doch passiert etwas Überraschendes im UMTS-Kernnetz.

Seit den Anfängen der Vernetzungstechnologie fand ein Krieg zwischen den Befürwortern der Paketnetze (d.h. verbindungslose Subnetze) und den Anhängern der Schaltnetzwerke (d.h. verbindungsorientierte Subnetze) statt. Die Hauptvertreter der Paketnetze kommen aus der Internetgemeinde. In einem verbindungslosen Netzentwurf wird jedes Paket unabhängig von allen anderen Paketen weitergeleitet. Falls also einige Router während einer Sitzung zusammenbrechen, wird kein Schaden entstehen, solange sich das System selbst dynamisch neu konfigurieren kann. Somit können die nachfolgenden Pakete an ihrem Ziel ankommen, selbst wenn sie dabei einen anderen Pfad als die vorigen Pakete nehmen.

Das Lager der Schaltungsanhänger kommt aus der Welt der Telefongesellschaften. Im Telefonystem muss ein Anrufer die Nummer des angerufenen Teilnehmers wählen und auf eine Verbindung warten, bevor er sprechen oder Daten senden kann. Dieser Verbindungsauflauf baut einen Weg durch das Telefonystem auf, der so lange erhalten bleibt, bis der Anruf beendet wird. Alle Wörter oder Pakete folgen derselben Route. Falls eine Leitung oder Schaltung auf dem Pfad zusammenbricht, wird der Anruf abgebrochen. Dieser Ansatz ist somit weniger fehlertolerant als ein verbindungsloser Entwurf.

Der Vorteil von Schaltkreisen ist, dass sie Dienstgüte besser unterstützen können. Wenn eine Verbindung im Voraus aufgebaut wird, kann das Subnetz Ressourcen wie Verbindungsbandbreite, Switch-Puffer und CPU reservieren. Falls der Versuch eines Anrufaufbaus unternommen wird, doch keine ausreichenden Ressourcen verfügbar sind, so wird der Anruf abgelehnt und der Anrufer erhält eine Art Besetztzeichen. Auf diese Weise kann die Qualität der Verbindung, sobald diese einmal zustande gekommen ist, garantiert werden.

Falls bei einem verbindungslosen Netzwerk zu viele Pakete gleichzeitig am gleichen Router ankommen, wird der Router quasi ersticken und wahrscheinlich Pakete verlieren. Irgendwann wird der Absender dies bemerken und die Pakete noch einmal senden, aber die Dienstgüte wird schwanken und – außer bei schwacher Netzauslastung – für Audio- oder Videoübertragung nicht ausreichend sein. Es ist unnötig zu sagen, dass die Bereitstellung von angemessener Audioqualität etwas ist, was für Telefongesellschaften sehr wichtig ist, daher ihre Präferenz für Verbindungen.

Die Überraschung in Abbildung 1.31 ist, dass im Kernnetz sowohl Pakete als auch Schaltungen vorkommen. Hier wird das Mobilfunknetz in einer Übergangssituation dargestellt, in der Mobilfunkgesellschaften eine oder manchmal beide der Möglichkeiten implementieren können. Ältere Mobilfunknetze verwendeten einen leitungsvermittelten Kern im Stil der traditionellen Telefonnetze, um Sprachanrufe zu übertragen. Diese Altlast sieht man im UMTS-Netz mit der **mobilen Vermittlungsstelle (MSC, Mobile Switching Center)**, Elementen von GMSC (*Gateway Mobile Switching Center*) und MGW (*Media Gateway*), die Verbindungen über ein leitungsvermitteltes Kernnetz wie das PSTN (*Public Switched Telephone Network*) aufbauen.

Datendienste sind heute ein viel wichtigerer Teil des Mobilfunknetzes als früher. Es begann mit Textnachrichten und frühen Paketdatendiensten wie **GPRS** (*General Packet Radio Service*) im GSM-System. Diese älteren Datendienste liefen im kbit-Bereich, doch die Nutzer verlangten nach mehr. Neuere Mobilfunknetze übertragen Datenpakete mit mehreren Mbit/s. Zum Vergleich: ein Sprachanruf wird mit einer Rate von 64 kbit/s übertragen, bei Kompression in der Regel mit einer 3–4-mal kleineren Rate.

Zur Datenübertragung verbinden sich die Knoten des UMTS-Kernnetzes direkt mit einem paketvermittelnden Netz. Das **SGSN** (*Serving GPRS Support Node*) und das **GGSN** (*Gateway GPRS Support Node*) liefern Datenpakete zu und von mobilen Geräten und Schnittstellen von externen Paketnetzen wie dem Internet.

Dieser Übergang wird sich in den Mobilfunknetzen fortsetzen, die heute geplant und eingesetzt werden. Internetprotokolle werden sogar auf Mobilgeräten verwendet, um Verbindungen für Sprachanrufe über ein Paketdatennetz in der Art von Voice-over-IP aufzubauen. IP und Pakete werden für den gesamten Weg vom Funkzugangsnetz durch das Kernnetz verwendet. Natürlich ändert sich auch die Art, wie IP-Netze entworfen werden, um bessere Dienstgüte zu unterstützen. Dies ist auch notwendig, denn Probleme mit zerhackten Audio- und stockenden Videodaten schrecken zahlende Kunden eher ab. Auf dieses Thema werden wir in *Kapitel 5* zurückkommen.

Ein weiterer Unterschied zwischen Mobilfunknetzen und dem traditionellen Internet ist die Mobilität. Wenn ein Nutzer sich aus dem Bereich einer zellulären Basisstation heraus- und in den Bereich eines anderen hineinbewegt, dann muss der Datenfluss von der alten zur neuen Basisstation umgeleitet werden. Diese Technik ist als **Handover** oder **Handoff** bekannt (►Abbildung 1.32).

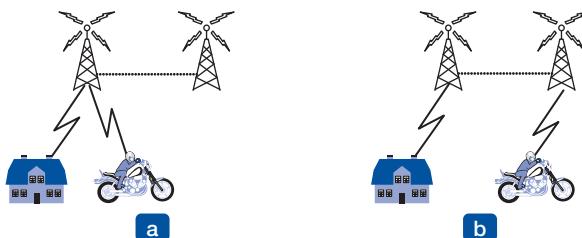


Abbildung 1.32: Handover im Mobilfunk: (a) vorher, (b) danach.

Ein Handover wird entweder vom mobilen Gerät oder der Basisstation angefordert, sobald die Qualität des Signals abfällt. In einigen zellulären Netzen – in der Regel in solchen, die auf CDMA-Technologie basieren – ist es möglich, die Verbindung zur neuen Basisstation aufzubauen, bevor die Verbindung zur alten Basisstation getrennt wird. Dadurch wird die Verbindungsqualität für das mobile Gerät verbessert, weil es keinen Bruch im Dienst gibt; tatsächlich ist das mobile Gerät für kurze Zeit mit zwei Basisstationen verbunden. Diese Art, ein Handover durchzuführen, wird **Soft Handover** genannt, um es vom **Hard Handover** zu unterscheiden, bei dem das mobile Gerät von der alten Basisstation trennt wird, bevor es mit der neuen verbunden wird.

Ein verwandtes Problem ist, wie ein mobiles Gerät bei einem ankommenden Anruf überhaupt gefunden wird. Jedes Mobiltelefon hat einen **HSS** (*Home Subscriber Server*) im Kernnetz, das den Aufenthaltsort jedes Teilnehmers kennt und ebenso andere Profilinformationen, die zur Authentifizierung und Autorisierung benutzt werden. Durch Kontaktieren des HSS kann also jedes mobile Gerät gefunden werden.

Ein letzter Bereich, den wir besprechen wollen, ist Sicherheit. Historisch gesehen haben Telefongesellschaften das Thema Sicherheit viel ernster genommen als es die Internetbetreiber eine lange Zeit taten, da sie ihre Dienste in Rechnung stellen und Betrügereien bei der Bezahlung vermeiden mussten. Leider sagt das noch nicht sehr viel aus. Dennoch waren bei der Weiterentwicklung von 1G- zu 3G-Technologien die Mobiltelefongesellschaften in der Lage, einige grundlegende Sicherheitsmechanismen für mobile Geräte herauszubringen.

Seit dem 2G-GSM-System war das Mobiltelefon in das Telefongerät selbst und einen entfernbaren Chip aufgeteilt, der die Identität des Teilnehmers und weitere Informationen enthält. Der Chip wird informell **SIM-Karte** (für *Subscriber Identity Module*) genannt. SIM-Karten können in verschiedene Handygeräte zur Aktivierung eingesetzt werden und sie bieten eine Basis für die Sicherheit. Wenn GSM-Kunden im Urlaub oder geschäftlich in andere Länder reisen, bringen sie häufig ihre Handys mit und kaufen bei ihrer Ankunft für ein paar Euro eine neue SIM-Karte, um ohne Roaming-Gebühren lokal zu telefonieren.

Zur Reduzierung von Betrügereien nutzt das Mobilfunknetz die Information auf SIM-Karten auch, um Teilnehmer zu authentifizieren und um zu überprüfen, ob diese das Netz benutzen dürfen. Bei UMTS verwendet das mobile Gerät die Information auf der SIM-Karte um zu prüfen, ob die Verbindung mit einem zulässigen Netz hergestellt wird.

Ein anderer Aspekt von Sicherheit ist Datenschutz. Funksignale werden an alle nahe gelegenen Empfänger gesendet. Daher werden kryptografische Schlüssel auf der SIM-Karte zum Entschlüsseln der Übermittlung eingesetzt, um das Mithören von Gesprächen zu erschweren. Dieser Ansatz bietet weit besseren Datenschutz als dies in 1G-Systemen der Fall war, die leicht angezapft werden konnten, dennoch stellt es aufgrund der Schwäche in den Entschlüsselungsschemata kein Allheilmittel dar.

Mobilfunknetze werden in zukünftigen Netzen eine zentrale Rolle spielen. Sie haben heute mehr mit mobilen Breitbandanwendungen als mit Sprachanrufen zu tun, was bedeutende Auswirkungen auf die Luftschnittstellen, die Architektur der Kernnetze und die Sicherheit von künftigen Netzen hat. Die noch schnelleren und besseren 4G-Technologien sind unter dem Namen **LTE** (*Long Term Evolution*) in der Entwicklung, dennoch geht der 3G-Entwurf und -Einsatz weiter. Andere drahtlose Technologien bieten ebenfalls Breitbandinternetzugang für stationäre und mobile Kunden, allen voran die IEEE-802.16-Netze, die unter dem Namen **WiMAX** bekannt sind. Es ist durchaus möglich, dass sich LTE und WiMAX auf einem Kollisionskurs miteinander befinden und man kann schwer vorhersagen, was mit ihnen geschehen wird.

1.5.3 Drahtlose LANs: IEEE 802.11

Mit dem Aufkommen der Laptops kam auch die Vorstellung auf, dass man mit dem Laptop nur in ein Büro gehen muss und der Laptop stellt automatisch die Verbindung zum Internet her. Daher begannen mehrere Gruppen an Lösungen hierfür zu arbeiten. Der praktischste Ansatz war, sowohl das Büro als auch den Laptop mit Kurzstreckenfunktionsendern und -empfängern auszurüsten, über die sie kommunizieren können.

Fortschritte auf diesem Gebiet führten schnell zu drahtlosen LANs, die von verschiedenen Unternehmen vermarktet wurden. Das Problem war nur, dass keine zwei davon kompatibel waren. Die Vielzahl der Standards bedeutete, dass ein Computer, der mit einem Funkgerät der Marke X ausgestattet war, nicht in einem Raum funktionierte, in dem eine Basisstation der Marke Y aufgestellt war. Mitte der 1990er Jahre befand die Industrie, dass ein Standard für drahtlose LANs eine gute Idee sei. Daher wurde das

IEEE-Komitee, welches das kabelbasierte LAN standardisiert hatte, beauftragt, einen Standard für drahtlose LANs zu entwickeln.

Die erste Entscheidung war am einfachsten: Welchen Namen sollte es erhalten? Alle anderen LAN-Standards bestanden aus Ziffern wie 802.1, 802.2, 802.3 bis hin zu 802.10. Daher wurde dem drahtlosen LAN der Name „802.11“ gegeben. Eine gängige umgangssprachliche Bezeichnung dafür ist **WiFi**, doch da es ein wichtiger Standard ist, der Achtung verdient, wollen wir seinen korrekten Namen IEEE 802.11 verwenden.

Der Rest war schwieriger. Das erste Problem war, ein passendes Frequenzband zu finden, das – vorzugsweise weltweit – verfügbar war. Der gewählte Ansatz war das Gegen teil von dem Vorgehen in Mobilfunknetzen. Anstatt ein teures, lizenziertes Spektrum zu benutzen, operieren IEEE-802.11-Systeme in unlizenzierten Bändern wie dem **ISM-Band** (die Abkürzung steht für *Industrial, Scientific, Medical*), das von ITU-R definiert wurde (z.B. 902–928 MHz, 2,4–2,5 GHz, 5,725–5,825 GHz). Alle Geräte dürfen dieses Spektrum benutzen, vorausgesetzt sie begrenzen ihre Sendeleistung, damit unterschiedliche Geräte nebeneinander betrieben werden können. Dies könnte natürlich bedeuten, dass sich IEEE-802.11-Funkwellen in Konkurrenz mit schnurlosen Telefonen, Garagentoröffnern und Mikrowellenöfen wiederfinden.

IEEE-802.11-Netze bestehen aus Clients wie Laptops und Mobiltelefonen sowie einer Infrastruktur namens **Zugangspunkt (AP, Access Point)**, die in Gebäuden installiert ist. Zugangspunkte werden manchmal auch **Basisstationen** genannt. Sie stellen die Verbindung mit dem verkabelten Netzwerk her; die gesamte Kommunikation zwischen Clients läuft über einen Zugangspunkt. Für Clients in Funkreichweite ist auch direkte Kommunikation möglich, ohne einen Zugangspunkt, beispielsweise zwei Rechner in einem Büro. Dieses Arrangement wird ein **Ad-hoc-Netzwerk** genannt. Es wird deutlich seltener als der Zugangspunktmodus eingesetzt. Beide Modi sind in ► Abbildung 1.33 gezeigt.

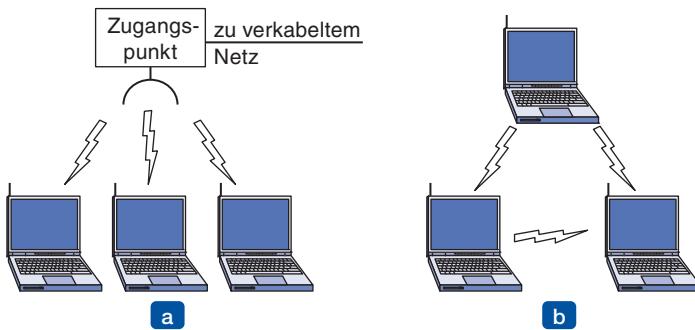


Abbildung 1.33: (a) Drahtloses Netz mit einem Zugangspunkt. (b) Ad-hoc-Netzwerk.

IEEE-802.11-Übertragungen werden durch drahtlose Bedingungen, die selbst mit kleinen Änderungen in der Umgebung variieren, komplizierter. Bei den für IEEE 802.11 benutzten Frequenzen können Funksignale von festen Gegenständen reflektiert werden, sodass bei einem Empfänger mehrere Echosignale einer Übertragung auf unterschiedlichen Pfaden ankommen können. Die Echosignale können sich gegenseitig aufheben oder verstärken, dadurch ist das empfangene Signal einer hohen Fluktuation ausgesetzt. Die-

ses Phänomen heißt **Mehrwegeempfang** (*multipath fading*), es ist in ► Abbildung 1.34 illustriert.

Zur Überwindung dieser schwankenden drahtlosen Bedingungen wird vor allem das Konzept der **Mehrwegeführung** (*path diversity*) – das Senden von Information über viele unabhängige Pfade – herangezogen. Auf diese Art wird die Information vermutlich auch dann ankommen, wenn einer der Pfade schwächer wird. Diese unabhängigen Pfade werden typischerweise in das digitale Modulationsschema in der Bitübertragungsschicht eingebaut. Optionen umfassen die Benutzung unterschiedlicher Frequenzen über dem erlaubten Band, das Verfolgen von unterschiedlichen räumlichen Pfaden zwischen verschiedenen Antennenpaaren oder das Wiederholen von Bits über unterschiedliche Zeiträume.

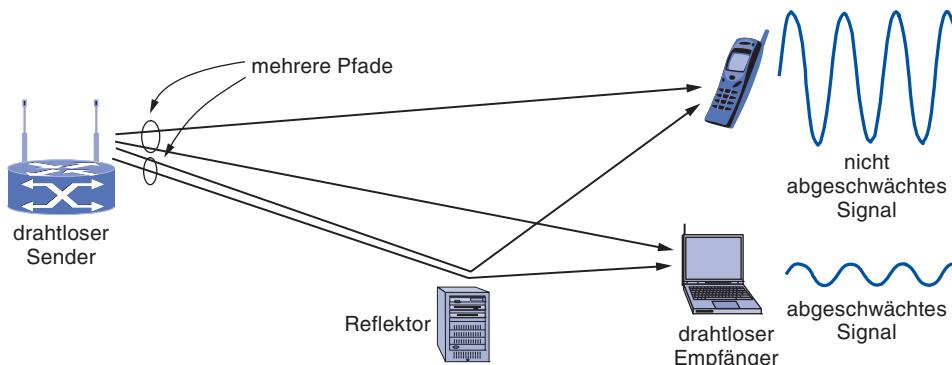


Abbildung 1.34: Mehrwegeempfang.

Verschiedene Versionen von IEEE 802.11 haben all diese Techniken eingesetzt. Der anfängliche Standard (von 1997) definierte ein drahtloses LAN, das entweder mit 1 Mbit/s oder 2 Mbit/s sendete und zwischen Frequenzen hin- und herwechselte oder bei dem das Signal über das erlaubte Spektrum verstreu wurde. Fast unmittelbar beklagten sich Leute, dass dies zu langsam sei, also wurden schnellere Standards erarbeitet. Die Frequenzspreizung wurde im IEEE-802.11b-Standard (1999) erweitert und lieferte Datenraten bis zu 11 Mbit/s. Die IEEE-Standards 802.11a (1999) und 802.11g (2003) wechselten zu einem unterschiedlichen Modulationsschema, dem **orthogonalen Frequenzmultiplexverfahren (OFDM, Orthogonal Frequency Division Multiplexing)**. Dieses Verfahren teilt ein weites Spektrumsband in viele enge Scheiben, über die verschiedene Bits parallel gesendet werden. Dieses verbesserte Schema, welches wir in Kapitel 2 untersuchen werden, hob die IEEE-802.11a/g-Bitraten auf bis zu 54 Mbit/s an. Das ist eine bedeutende Steigerung, doch die Nutzer verlangten mehr Datendurchsatz, um anspruchsvollere Anwendungen zu unterstützen. Die neueste Version ist IEEE 802.11n (2009). Diese verwendet weitere Frequenzbänder und bis zu vier Antennen pro Computer, um Übertragungsraten bis zu 450 Mbit/s zu erzielen.

Da ein drahtloses Netz grundsätzlich ein Broadcast-Medium ist, muss ein IEEE-802.11-Funknetz auch das Problem lösen, das entsteht, wenn mehrere Übertragungen gleichzeitig gesendet werden. Dies würde zu einer Kollision führen, wodurch der

Empfang beeinträchtigt wird. Deshalb benutzt IEEE 802.11 ein Schema, das als **CSMA** (*Carrier Sense Multiple Access*) bezeichnet wird. Es greift auf Konzepte des klassischen verdrahteten Ethernets zurück, welche – ironischerweise – wiederum von einem frühen drahtlosen Netz stammen, das in Hawaii entwickelt wurde und **ALOHA** heißt. Der Computer wartet eine kurze, zufällige Zeitspanne, bevor er mit seiner Übermittlung beginnt, beziehungsweise stellt seine Übertragung zurück, wenn er mitbekommt, dass bereits jemand anderes Daten sendet. Durch dieses Vorgehen wird es unwahrscheinlicher, dass zwei Rechner gleichzeitig senden. Im Fall von verkabelten Netzen funktioniert dies allerdings nicht so gut. Der Grund hierfür ist in ► Abbildung 1.35 dargestellt. Angenommen Computer A überträgt Daten zu Computer B, aber die Funkreichweite des Sendegeräts von A ist zu kurz, um Computer C zu erreichen. Wenn C Daten an B übertragen möchte, kann er vor dem Start auf das Vorhandensein einer Übertragung prüfen, aber auch wenn er nichts wahrnimmt, bedeutet dies noch nicht, dass die Übertragung auch gelingt. Da C nicht in der Lage ist, vor dem Beginn seiner Übertragung Computer A zu überprüfen, können Kollisionen auftreten. Nach einer Kollision wartet der Sender dann eine weitere, diesmal etwas längere Zeitspanne und überträgt das Paket erneut. Trotz dieser und einiger anderer Probleme funktioniert das Schema in der Praxis ganz gut.

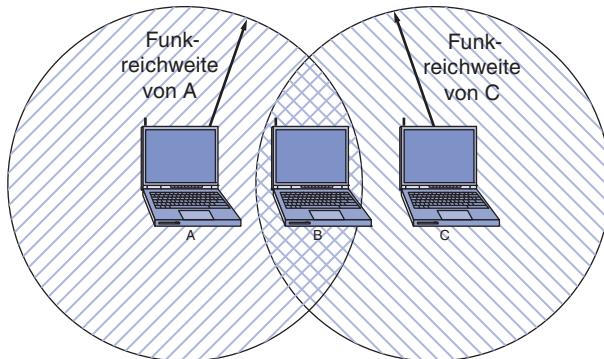


Abbildung 1.35: Die Reichweite eines einzelnen Funksenders deckt unter Umständen nicht das gesamte System ab.

Ein anderes Problem ist die Mobilität. Wenn ein mobiler Client vom verwendeten Zugangspunkt wegbewegt wird und in den Bereich eines anderen Zugangspunkts kommt, muss irgendwie mit dieser Situation umgegangen werden. Die Lösung sieht vor, ein IEEE-802.11-Netz aus mehreren Zellen, von denen jede über einen eigenen Zugangspunkt verfügt, und einem Verteilernetz aufzubauen, das die Zellen verbindet. Das Verteilernetz ist häufig Switched Ethernet, aber es könnte auch jede andere Technologie eingesetzt werden. Wenn sich die Clients bewegen, finden sie eventuell einen anderen Zugangspunkt mit einem besseren Signal als den aktuell benutzten, und die Clients können ihre Zuordnung ändern. Von außen betrachtet sieht das Gesamtsystem wie ein einziges verdrahtetes LAN aus.

Abgesehen davon war Mobilität bei IEEE 802.11 – verglichen mit der Mobilität im Mobilfunknetz – bislang nur von untergeordneter Bedeutung. In der Regel wird IEEE

802.11 von nomadenhaften Clients benutzt, die sich von einem festen Standort zum nächsten bewegen, und weniger von Clients, die ständig unterwegs sind. Mobilität ist für eine nomadenhafte Benutzung nicht wirklich nötig. Selbst wenn die IEEE-802.11-Mobilität ausgenutzt wird, bleibt sie auf ein einzelnes IEEE-802.11-Netz begrenzt, erstreckt sich also höchstens auf ein großes Gebäude. Zukünftig werden Verfahren benötigt, damit die Mobilität auf unterschiedliche Netze und unterschiedliche Technologien (z.B. IEEE 802.21) ausgeweitet werden kann.

Schließlich gibt es noch das Problem mit der Sicherheit. Da kabellose Übertragungen Broadcast-Verbindungen sind, ist es für nahe stehende Computer ein Leichtes, Informationspakete abzufangen, die nicht für sie bestimmt sind. Um dies zu verhindern, enthält der IEEE-802.11-Standard ein Verschlüsselungsschema, das als **WEP** (*Wired Equivalent Privacy*) bekannt ist. Die Idee dahinter war, drahtlose Sicherheit entsprechend der verdrahteten Sicherheit zu konzipieren. Der Grundgedanke ist zwar gut, doch unglücklicherweise war das Schema fehlerhaft und wurde bald gebrochen (Borisov et al., 2001). WEP ist inzwischen von einem neueren Schema ersetzt worden, **WiFi Protected Access**, welches andere kryptografische Details als der IEEE-802.11i-Standard hat. Die erste Methode, **WPA**, ist mittlerweile durch **WPA2** ersetzt worden.

IEEE 802.11 hat eine Revolution im Bereich der drahtlosen Netze ausgelöst, die sich auch weiterhin fortsetzt. Zusätzlich zu Gebäuden wird es zunehmend in Zügen, Flugzeugen, Booten und Autos installiert, sodass man im Internet surfen kann, wo immer man ist. Mobiltelefone und alle Arten von Unterhaltungselektronik – von Spielkonsolen bis zu Digitalkameras – können damit kommunizieren. Wie werden auf dieses Thema in *Kapitel 4* noch ausführlich zu sprechen kommen.

1.5.4 RFID und Sensornetze

Die Netze, die wir bisher untersucht haben, waren aus Geräten aufgebaut, die irgendwie als rechnerartig zu erkennen sind, vom Computer bis zu Mobiltelefonen. Mit **RFID** (*Radio Frequency IDentification*) können auch alltägliche Gegenstände Teil eines Rechnernetzes sein.

Ein RFID-Tag sieht wie ein briefmarkengroßer Aufkleber aus, der an einem Objekt befestigt (oder in ein Objekt eingebettet) werden kann, sodass man dieses verfolgen kann. Das Objekt könnte eine Kuh sein, ein Reisepass, ein Buch oder eine Versandpalette. Das Tag besteht aus einem kleinen Mikrochip mit einer eindeutigen Kennung und einer Antenne, die Funkübertragung empfängt. Fest installierte RFID-Lesegeräte entdecken alle Tags, die in ihre Funkreichweite kommen, und lesen deren Informationen aus (►Abbildung 1.36). Zu den Anwendungen gehören das Überprüfen von Identitäten, das Verwalten von Lieferketten, die Zeiterfassung bei Wettkräften und die Ersetzung von Barcodes.

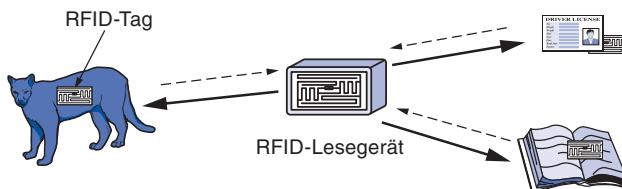


Abbildung 1.36: Einsatz von RFID bei der Vernetzung von alltäglichen Objekten.

Es gibt viele RFID-Arten, die alle unterschiedliche Eigenschaften haben, aber der vielleicht faszinierendste Aspekt der RFID-Technologie ist, dass die meisten RFID-Tags weder einen Stromanschluss noch eine Batterie besitzen. Stattdessen wird die gesamte Energie, die zum Betrieb nötig ist, in Form von Funkwellen von den RFID-Lesegeräten zur Verfügung gestellt. Diese Technologie heißt **passives RFID**, um es vom (weniger gebräuchlichen) **aktiven RFID** zu unterscheiden. Beim aktiven RFID ist eine Energiequelle auf dem Tag vorhanden.

Eine übliche RFID-Form ist **UHF-RFID** (*Ultra-High Frequency RFID*). Diese wird beim Versenden von Paletten und einigen Führerscheinarten eingesetzt. In den Vereinigten Staaten senden Lesegeräte Signale im 902–928-Mhz-Band. Tags kommunizieren über Entferungen von mehreren Metern, indem sie die Art verändern, wie sie die Signale des Lesegeräts reflektieren. Das Lesegerät ist in der Lage, diese Reflexionen aufzufangen. Diese Betriebsart wird **Rückstreuung** (*backscatter*) genannt.

Eine weitere bekannte RFID-Art ist **HF-RFID** (*High Frequency RFID*). Diese arbeitet mit 13,56 MHz und findet sich wahrscheinlich in Ihrem Reisepass, in Kreditkarten, Büchern und kontaktlosen Bezahlsystemen. HF-RFID hat eine kurze Reichweite, in der Regel einen Meter oder weniger, weil der physische Mechanismus auf Induktion statt auf Rückstreuung basiert. Es gibt auch noch andere Formen von RFIDs, die andere Frequenzen benutzen, wie **LF-RFID** (*Low Frequency RFID*), welche vor HF-RFID entwickelt wurde und zum Verfolgen von Tieren eingesetzt wird. Das ist die Art von RFID, die sich vermutlich in Ihrer Katze befindet.

RFID-Lesegeräte sind in der Regel mit dem Problem konfrontiert, dass sich mehrere Tags innerhalb ihres Leseradius befinden. Wenn jedes Tag sofort reagiert, sobald es etwas vom Lesegerät empfängt, dann könnten die Signale von mehreren Tags miteinander kollidieren. Die Lösung hierfür ist ähnlich dem bei IEEE 802.11 gewählten Ansatz: Tags warten eine kurze, zufällige Zeitspanne, bevor sie mit ihrer Identifikation reagieren, wodurch das Lesegerät individuelle Tags eingrenzen und diese weiter befragen kann.

Sicherheit ist ein weiteres Problem. Die Fähigkeit von RFID-Lesegeräten, ein Objekt einfach zu verfolgen – und damit die Person, die dieses benutzt – kann eine Verletzung der Privatsphäre bedeuten. Leider ist es schwierig, Sicherheit in RFID-Tags einzubauen, da ihnen die Rechen- und Kommunikationsleistung fehlt, um starke kryptografische Algorithmen auszuführen. Stattdessen werden schwache Maßnahmen wie Passwörter (welche leicht geknackt werden können) benutzt. Wenn es möglich ist, dass ein Personalausweis von einem Beamten an der Grenze aus einer gewissen Entfernung gelesen werden kann, was kann man dann gegen das heimliche Verfolgen desselben Ausweises durch Unbefugte unternehmen? Nicht viel.

RFID-Tags begannen als Identifikationschips, aber sie verwandeln sich schnell in vollwertige Rechner. Viele Tags haben beispielsweise einen Speicher, der aktualisiert und später abgefragt werden kann. Somit können Informationen darüber, was mit dem Tagging-Objekt passiert ist, ebenfalls gespeichert werden. Rieback et al. (2006) haben demonstriert, dass damit auch all die üblichen Probleme mit Computer-Malware auftreten: jetzt könnte Ihre Katze oder Ihr Reisepass benutzt werden, um einen RFID-Virus zu verbreiten.

Ein Schritt voran in Bezug auf Leistungsvermögen von RFID ist das **Sensornetz**. Sensornetze werden installiert, um Aspekte der physischen Welt zu überwachen. Bisher sind sie hauptsächlich für wissenschaftliche Versuche wie das Beobachten von Vogelhabitaten, Vulkanaktivitäten und Zebrawanderungen eingesetzt worden. Doch geschäftliche Anwendungen einschließlich Gesundheitswesen, Überwachungsausrüstung für Vibratoren und das Verfolgen von gefrorenen, tiefgekühlten oder anderweitig verderblichen Gütern sind nicht mehr in allzu weiter Ferne.

Sensorknoten sind kleine Rechner, häufig in der Größe eines Schlüsselanhängers, die Temperatur-, Vibrations- oder andere Sensoren besitzen. Viele Knoten sind in der Umgebung untergebracht, die beobachtet werden soll. In der Regel besitzen die Knoten Batterien, obwohl sie auch Energie aus Vibratoren oder der Sonne beziehen können. Wie bei RFID ist es eine der größten Herausforderungen, genug Energie zu bekommen, und die Knoten müssen ihre Kommunikation sorgfältig planen, um ihre Sensorinformationen an einen externen Sammelpunkt übermitteln zu können. Eine übliche Strategie ist, die Knoten so zu organisieren, dass jeder seiner Nachrichten über andere Knoten weitergeben kann, wie in ►Abbildung 1.37 gezeigt ist. Dieser Entwurf wird **Multi-Hop-Netz** genannt.

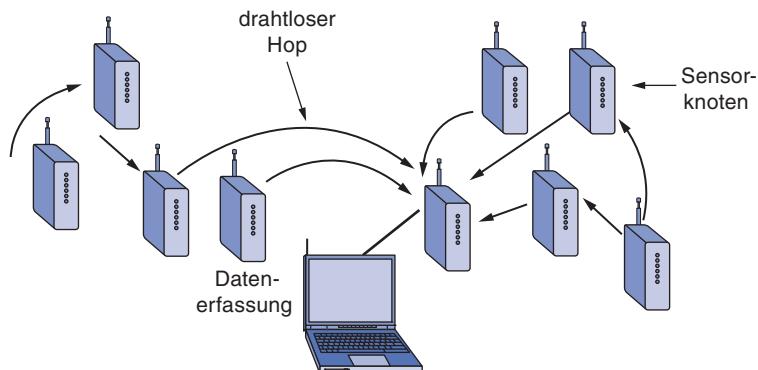


Abbildung 1.37: Multi-Hop-Topologie eines Sensornetzes.

RFID und Sensornetz werden in der Zukunft wahrscheinlich viel leistungsfähiger werden und sich weiter verbreiten. Forscher haben bereits das Beste von beiden Technologien kombiniert, indem sie den Prototyp von programmierbaren RFID-Tags mit Licht-, Bewegungs- und anderen Sensoren entwickelt haben (Sample et al., 2008).

1.6 Standardisierung von Netzen

Es gibt viele Netzbetreiber und -anbieter, die alle ihre eigenen Vorstellungen davon haben, wie es laufen sollte. Ohne Koordination würde völliges Chaos herrschen und die Benutzer könnten nicht mehr vernünftig arbeiten. Die einzige Möglichkeit aus diesem Dilemma ist die Vereinbarung bestimmter Netzstandards. Standards ermöglichen nicht nur die Kommunikation verschiedener Computer, sondern erweitern auch den Markt für Produkte, die dem Standard entsprechen. Ein größerer Markt führt zur Massenproduktion, wirtschaftlicheren Fertigung, besseren VLSI-Implementierungen und anderen Vorteilen, die den Preis senken und die Akzeptanz weiter erhöhen.

In diesem Abschnitt werfen wir einen kurzen Blick auf die so wichtige, aber dennoch so wenig bekannte Welt der internationalen Standardisierung. Doch lassen Sie uns zunächst sehen, was in einen Standard hineingehört. Ein vernünftiger Mensch könnte vielleicht annehmen, dass ein Standard ihm sagt, wie ein Protokoll arbeiten sollte, sodass die Implementierung gut gelingt. Dieser Mensch läge falsch.

Standards legen fest, was für die Zusammenarbeit benötigt wird: nicht mehr, nicht weniger. Dadurch können sich größere Märkte herausbilden und gleichzeitig Unternehmen auf der Basis ihrer Produktqualität konkurrieren. Beispielsweise definiert der IEEE-802.11-Standard viele Übertragungsraten, doch er sagt nichts darüber aus, wann ein Sender welche Rate verwenden sollte, was ein entscheidender Faktor von guter Performance ist. Dies wird demjenigen überlassen, der das Produkt herstellt. Häufig ist es schwierig, auf diese Weise zur Interoperabilität zu gelangen, da viele Implementierungsentscheidungen getroffen werden müssen und Standards in der Regel viele Möglichkeiten definieren. Bei IEEE 802.11 gab es so viele Probleme, dass eine Handelsgruppe namens [WiFi Alliance](#) gestartet wurde, um an der Interoperabilität innerhalb des IEEE-802.11-Standards zu arbeiten. Dieses Vorgehen ist inzwischen übliche Praxis geworden.

Ähnlich definiert ein Protokollstandard, wie das Protokoll auf den Leitungen auszusehen hat, aber nicht wie die internen Dienstschnittstellen beschaffen sind, außer um bei der Erklärung des Protokolls zu helfen. Tatsächlich sind Dienstschnittstellen oft proprietär. Zum Beispiel spielt die Art und Weise, wie die Schnittstelle von TCP zu IP innerhalb eines Computers aussieht, bei der Kommunikation mit einem entfernten Host keine Rolle. Es ist nur wichtig, dass der entfernte Host TCP/IP spricht. Tatsächlich werden TCP und IP häufig zusammen implementiert, ohne eine klar definierte Schnittstelle. Abgesehen davon sind gute Dienstschnittstellen ebenso wie gute APIs wertvoll bei der Einführung eines Protokolls, und richtig gute Schnittstellen (wie Berkeley-Sockets) können sehr populär werden.

Standards werden in zwei Kategorien unterteilt: de facto und de jure. [**De-facto-Standards**](#) (*de facto* ist lateinisch für „aufgrund der Tatsache“) sind jene, die sich einfach aufgrund der hohen Akzeptanz eines Produkts oder einer Produktreihe ergeben haben, ohne dass es dazu einen formellen Plan gab. Das Webprotokoll HTTP begann seine Existenz als ein De-facto-Standard. HTTP war Teil des frühen WWW-Browsers, den Tim Berners-Lee am CERN entwickelt hatte, und seine Beliebtheit wuchs zusam-

men mit dem Wachstum des Webs. Bluetooth ist ein weiteres Beispiel. Es wurde ursprünglich von Ericsson entwickelt, doch heute benutzt es jeder.

De-jure-Standards (lateinisch für „von Rechts wegen“) hingegen sind mithilfe der Regeln eines formalen Standardisierungsgremiums entwickelt worden. Die internationalen Standardisierungsgremien können ebenfalls in zwei Gruppen unterteilt werden: diejenigen, die aufgrund von Verträgen zwischen Staatsregierungen entstanden sind, und die freiwilligen Organisationen. Auf dem Gebiet der Rechnernetze gibt es in jeder Gruppe mehrere Organisationen, namentlich ITU, ISO, IETF und IEEE, die wir alle im Folgenden beschreiben werden.

In der Praxis sind die Beziehungen zwischen Standards, Unternehmen und Standardisierungsgremien kompliziert. De-facto-Standards entwickeln sich oft zu De-jure-Standards, besonders wenn sie erfolgreich sind. Dies passierte im Fall von HTTP, das schnell von der IETF aufgegriffen wurde. Standardisierungsgremien ratifizieren sich häufig gegenseitig ihre Standards – wodurch der Eindruck erweckt wird, man klopfe sich einander auf die Schulter, um den Markt für eine Technologie zu steigern. Zur Zeit spielen viele Ad-hoc-Geschäftsallianzen, die sich um bestimmte Technologien herum bilden, ebenfalls eine bedeutende Rolle bei der Entwicklung und dem Verfeinern von Netzstandards. Zum Beispiel ist **3GPP** (*Third Generation Partnership Project*) eine Zusammenarbeit zwischen Telekommunikationsvereinigungen, die den UMTS-3G-Mobilfunkstandard vorantreiben.

1.6.1 Who's who in der Welt der Telekommunikation

Der rechtliche Status der Telefongesellschaften dieser Welt ist von Land zu Land sehr verschieden. An dem einen Ende stehen die Vereinigten Staaten, wo es über 2 000 eigenständige (meist sehr kleine) private Telefongesellschaften gibt. Einige kamen im Zuge der Spaltung von AT&T im Jahr 1984 hinzu (AT&T war bis dahin die größte Gesellschaft der Welt, die ungefähr 80 % aller amerikanischen Telefonanschlüsse versorgte), andere mit dem Telecommunications Act von 1996, mit dem die bisherige Regulierung überholt wurde, um den Wettbewerb fördern.

Das andere Extrem sind die Länder, in denen die Landesregierung das Monopol auf alle Kommunikationsmedien, z.B. Post, Telegramm, Telefon und oft sogar Radio und Fernsehen, besitzt. Hierunter fällt ein großer Teil der restlichen Welt. In manchen Fällen ist die Behörde für Telekommunikation ein staatliches Unternehmen, in anderen ist sie einfach eine Regierungsabteilung, die häufig den Namen **PTT** (*Post, Telegraph & Telephone*, Post- und Fernmeldeverwaltungen) trägt. Weltweit zeichnet sich ein Trend ab, dieses Monopol abzubauen, d.h., den Telekommunikationsmarkt für private Unternehmen zugänglich zu machen und damit dem Wettbewerb auszusetzen. Die meisten europäischen Länder haben nun ihre Post- und Fernmeldeverwaltung (teil-) privatisiert, aber in anderen Gebieten steckt dies noch in den Kinderschuhen.

Angesichts dieser verschiedenen Betreiber und Anbieter liegt die Notwendigkeit einer weltweiten Kompatibilität auf der Hand, damit Menschen (und Computer) in allen Ländern miteinander in Kontakt treten können. Dieser Bedarf besteht schon sehr

lange. Bereits 1865 setzten sich Vertreter vieler europäischer Regierungen zusammen, um den Vorläufer der heutigen **ITU** (*International Telecommunications Union*) zu gründen. Aufgabe der ITU war die Standardisierung der internationalen Telekommunikation, was in jenen Tagen Telegrafie bedeutete. Schon damals war klar, dass Probleme ins Haus stehen, wenn die Hälfte der Länder mit Morsecode und die andere Hälfte mit einem anderen Code sendet. Als der internationale Telefondienst in Betrieb genommen wurde, übernahm die ITU auch die Aufgabe der Standardisierung des Telefons. 1947 wurde die ITU eine Behörde der Vereinten Nationen.

Derzeit sind über 200 Regierungen Mitglied bei der ITU; hierzu gehört auch fast jeder Mitgliedsstaat der Vereinten Nationen. Da es in den USA keine PTT gibt, muss jemand anderer das Land in der ITU vertreten. Mit dieser Aufgabe wurde das State Department (US-Außenministerium) betraut, wahrscheinlich aufgrund der Tatsache, dass die ITU mit vielen Ländern zu tun hat. Die ITU hat außerdem mehr als 700 Sektormitglieder und assoziierte Mitglieder. Dazu gehören Telefongesellschaften (z.B. AT&T, Vodafone, Sprint), Hersteller von Telekommunikationsgeräten (z.B. Cisco, Nokia, Nortel), Computerhersteller (z.B. HP, Sun, Toshiba), Chiphersteller (z.B. Intel, Motorola, TI) sowie andere interessierte Unternehmen (z.B. Boeing, CBS, VeriSign).

Die ITU umfasst drei Hauptsektoren. Wir konzentrieren uns hier auf **ITU-T** (*Telecommunication Standardization Sector*), der sich mit Telefon- und Datenkommunikationssystemen befasst. Vor 1993 hieß dieser Sektor **CCITT**, ein Akronym der französischen Bezeichnung „Comité Consultatif International Télégraphique et Téléphonique“. **ITU-R**, der Radiokommunikationssektor, befasst sich mit der weltweiten Koordination von Funkfrequenzen an die im Wettbewerb stehenden Interessengruppen. Der andere Sektor ist **ITU-D**, der Entwicklungssektor. Dieser unterstützt die Entwicklung von Informations- und Kommunikationstechnologien, um die „digitale Spaltung“ zwischen Ländern mit effektivem Zugang zu Informationstechnologien und Ländern mit beschränktem Zugang zu verringern.

Die Aufgabe der ITU-T ist es, technische Empfehlungen für Telefon-, Telegraf- und Datenkommunikationsschnittstellen auszuarbeiten. Aus diesen Empfehlungen werden oft international anerkannte Standards, obwohl diese Empfehlungen technisch gesehen nur Vorschläge sind, die die einzelnen Regierungen – ganz nach Belieben – annehmen oder ignorieren können (denn Regierungen sind wie 13-jährige Kinder: sie mögen es nicht, Anweisungen zu bekommen). In der Praxis sieht das so, dass es jedem Land freisteht, einen völlig anderen Telefonstandard als der Rest der Welt anzunehmen – jedoch zu dem Preis der völligen Isolation. Das funktioniert vielleicht für Nordkorea, aber anderswo wäre das eine Katastrophe.

Die wirkliche Arbeit der ITU-T findet in den Arbeitsgruppen (*Study Group*) statt. Derzeit gibt es 10 Arbeitsgruppen, die nicht selten mehr als 400 Leute umfassen und die sich mit Themen von der Telefonabrechnung über Multimediadienste bis hin zu Sicherheitsfragen befassen. SG 15 ist zum Beispiel ein Standard für die DSL-Technologie, die gerne für Internetverbindungen benutzt wird. Damit überhaupt etwas erledigt werden kann, werden die Arbeitsgruppen in kleinere Arbeitsteams aufgeteilt, die

ihrerseits in Expertenteams unterteilt werden, die wiederum aus projektbezogenen Gruppen zusammengestellt werden. Einmal bürokratisch, immer bürokratisch.

Trotz alledem ist die ITU-T produktiv. Seit der Gründung wurden mehr als 3 000 Empfehlungen erarbeitet, wovon viele in der Praxis häufig zum Einsatz kommen. So wird beispielsweise Empfehlung H.264 (welche auch ein ISO-Standard ist, bekannt als MPEG-4 AVC) oft für Videokompression benutzt, und X.509-Public-Key-Zertifikate werden für sicheres Webbrowsing und digital signierte E-Mails verwendet.

Im Zuge des Übergangs des Telekommunikationsbereichs, der in den 1980er Jahren einsetzte, von einem vollständig nationalen zu einem vollständig globalen Gebiet gewinnen Standards an Bedeutung. Künftig werden mehr und mehr Organisationen danach streben, an der Ausarbeitung dieser Standards aktiv teilzunehmen. Weitere Informationen über die ITU finden Sie bei Irmer (1994).

1.6.2 Who's who in der Welt der internationalen Normen

Internationale Normen werden von der **ISO** (*International Standards Organization*)² ausgegeben, einer freiwilligen, nicht per Staatsvertrag geregelten Organisation, die 1946 gegründet wurde. Ihre Mitglieder sind die nationalen Normungsinstitute der 157 Mitgliedsstaaten. Dazu gehören ANSI (USA), BSI (Großbritannien), AFNOR (Frankreich), DIN (Deutschland) und 153 weitere.

Die ISO legt für eine wahre Unmenge verschiedener Sachgebiete Normen fest, angefangen bei Schrauben und Muttern bis hin zur Beschichtung von Telefonleitungsmasten [ganz zu schweigen von Kakaobohnen (ISO 2451), Fischernetzen (ISO 1530), Damenunterwäsche (ISO 4416) und einer ganzen Reihe weiterer Dinge, von denen man nie glauben würde, dass sie normiert werden]. Bei Standards bezüglich Telekommunikation arbeiten ISO und ITU-T oft zusammen (ISO ist Mitglied der ITU-T), um die Blamage zu vermeiden, dass zwei offizielle Organe nicht kompatible internationale Standards ausarbeiten.

Insgesamt wurden über 17 000 Normen formuliert, darunter die OSI-Standards. ISO hat fast 200 technische Fachausschüsse (TC, *Technical Committee*), die nach der Reihenfolge ihrer Entstehung nummeriert sind und sich mit je einem Fachgebiet befassen. TC1 beschäftigt sich z.B. mit Schrauben und Muttern (und der Standardisierung der Gewindesteigungen). JTC1 befasst sich mit Informationstechnologie, einschließlich Netzwerke, Computer und Software. Es ist der erste (und bisher einzige) gemeinsame Fachausschuss (*Joint Technical Committee*). JTC1 wurde 1987 gegründet, als TC97 mit den Aktivitäten von IEC, einem weiteren Standardisierungsgremium, zusammengelegt wurde. Jeder technische Fachausschuss hat Unterausschüsse (SC, *Subcommittee*), die in Arbeitsgruppen (WG, *working group*) aufgeteilt sind.

Die eigentliche Arbeit wird weltweit in den Arbeitsgruppen von über 100 000 Freiwilligen geleistet. Viele dieser „Freiwilligen“ werden von ihren Arbeitgebern, deren Produkte normiert werden sollen, für die Arbeit in Sachen ISO abgestellt. Andere sind Regierungs-

2 Für alle, die es genau nehmen: Die richtige Bezeichnung der ISO ist „International Organization for Standardization“.

beamte, die bestrebt sind, eine Norm ihres Landes international durchzusetzen. In vielen Arbeitsgruppen sind auch Wissenschaftler aus verschiedenen Fachgebieten vertreten.

Das Standardisierungsverfahren der ISO ist auf breitestmöglichen Konsens ausgelegt. Der Prozess beginnt, wenn eines der nationalen Normungsinstitute auf einem Gebiet die Notwendigkeit einer internationalen Norm sieht. Dann wird eine Arbeitsgruppe gebildet, die einen **Ausschussentwurf (CD, Committee Draft)** ausarbeitet. Dieser Entwurf wird an alle Mitglieder verteilt. In den darauffolgenden sechs Monaten darf Kritik daran geübt werden. Wenn die überwiegende Mehrheit dafür ist, wird der Entwurf überarbeitet und als **internationaler Standardentwurf (DIS, Draft International Standard)** ausgearbeitet und zur Diskussion und Abstimmung verteilt. Der endgültige Wortlaut des **internationalen Standards (IS, International Standard)** wird basierend auf den Ergebnissen dieser Runde vorbereitet, bestätigt und veröffentlicht. Auf Gebieten, in denen viele Meinungen kontrovers aufeinandertreffen, müssen Entwürfe und DIS eventuell mehrmals umformuliert und überarbeitet werden, bis sie mit genügender Stimmenanzahl verabschiedet werden. Dieses Standardisierungsverfahren kann unter Umständen Jahre dauern.

NIST (*National Institute of Standards and Technology*) ist eine Behörde des US-Handelsministeriums. Es wurde vormals NBS (*National Bureau of Standards*) genannt. Diese Behörde gibt Standards heraus, die für das gesamte Beschaffungswesen der US-Regierungsbehörden gelten (mit Ausnahme des Verteidigungsministeriums, das seine eigenen Normen festlegt) und zwingend eingehalten werden müssen.

Ein weiteres wichtiges Mitglied auf der Standardisierungsbühne ist **IEEE** (*Institute of Electrical and Electronics Engineers*), der größte Fachverband der Welt. Außer der Veröffentlichung von Testberichten in Fachzeitschriften und der Veranstaltung vieler jährlich stattfindender Konferenzen beschäftigt sich die IEEE in einer Standardisierungsgruppe auch mit der Entwicklung von Normen im Bereich der Elektrotechnik und Informatik. Das 802-Komitee der IEEE hat viele LAN-Typen standardisiert. Einige werden später noch ausführlicher behandelt. Die eigentliche Arbeit wird in mehreren Arbeitsgruppen geleistet, die in ► Abbildung 1.38 aufgeführt sind. Die Erfolgsrate der verschiedenen IEEE-802-Arbeitsgruppen war gering – eine 802.x-Nummer ist noch kein Garant für den Erfolg. Dennoch waren die Auswirkungen der Erfolgsgeschichten, vor allem von IEEE 802.3 und IEEE 802.11, auf Industrie und die Welt enorm.

| Nummer | Thema |
|---------|---|
| 802.1 | Übersicht und Architektur von LANs |
| 802.2 ↓ | Logische Verbindungssteuerung |
| 802.3 * | Ethernet |
| 802.4 ↓ | Token-Bus (wurde kurze Zeit in Fertigungsbetrieben verwendet) |
| 802.5 | Token-Ring (IBMs Einstieg in die LAN-Welt) |

Abbildung 1.38: Die IEEE-802-Arbeitsgruppen. Die wichtigen sind mit * gekennzeichnet. Die Tätigkeit der mit ↓ gekennzeichneten Arbeitsgruppen ruht gerade. Die mit einem † gekennzeichnete Gruppe hat ihre Arbeit beendet und sich aufgelöst.

| Nummer | Thema |
|----------|---|
| 802.6 ↓ | Dual Queue Dual Bus (frühe MANs) |
| 802.7 ↓ | Technische Beratungsgruppe für Breitbandtechnologien |
| 802.8 † | Technische Beratungsgruppe für Glasfasertechnologien |
| 802.9 ↓ | Isochrone LANs (für Echtzeitanwendungen) |
| 802.10 ↓ | Virtuelle LANs und Sicherheit |
| 802.11 * | Drahtlose LANs (WiFi) |
| 802.12 ↓ | Bedarfsgesteuerte Priorität (AnyLAN von Hewlett-Packard) |
| 802.13 | (Unglückszahl, die keiner wollte) |
| 802.14 ↓ | Kabelmodems (besteht nicht mehr: ein Industiekonsortium war hier schneller) |
| 802.15 * | Persönliche Netze (Bluetooth, Zigbee) |
| 802.16 * | Drahtlose Breitbandnetze |
| 802.17 | Resilient Packet Ring |
| 802.18 | Technische Beratungsgruppe für Fragen zur Funkregulierung |
| 802.19 | Technische Beratungsgruppe zur Koexistenz all dieser Standards |
| 802.20 | Drahtlose Breitbandnetze (ähnlich 802.16e) |
| 802.21 | Medienunabhängiges Handover (zum Übergang zwischen Technologien) |
| 802.22 | Drahtlose Regionalnetze |

Abbildung 1.38: Die IEEE-802-Arbeitsgruppen. Die wichtigen sind mit * gekennzeichnet. Die Tätigkeit der mit ↓ gekennzeichneten Arbeitsgruppen ruht gerade. Die mit einem † gekennzeichnete Gruppe hat ihre Arbeit beendet und sich aufgelöst. (*Forts.*)

1.6.3 Who's who in der Welt der Internetstandards

Das weltweite Internet verfügt über eigene Standardisierungsmechanismen, die sich stark von denen der ITU-T und ISO unterscheiden. Zusammenfassend kann man sagen, dass die Leute, die zu den ITU- und ISO-Standardisierungskonferenzen kommen, Anzug und Krawatte tragen, während man in den entsprechenden Internetkonferenzen eher Leuten in Jeans begegnet (außer wenn die Treffen in San Diego stattfinden, dann werden Shorts und T-Shirt getragen).

In den Konferenzen der ITU-T und ISO wimmelt es von Führungskräften aus Unternehmen und Beamten verschiedener Regierungen. Sie betrachten Standardisierung als gute Sache, der sie ihr Leben widmen. Internetvertreter legen demgegenüber eine eher anarchische Haltung an den Tag. Wenn jedoch viele Millionen Personen nur nach ihrem Gutdünken verfahren, ist wenig Kommunikation möglich. Somit sieht man

auch in diesen Kreisen mit mehr oder weniger großem Bedauern ein, dass hier und da Standards nötig sind. In diesem Zusammenhang hat David Clark vom MIT die heute berühmte Bemerkung gemacht, dass Internetstandardisierung aus „grobem Konsens und lauffähigem Code“ besteht.

Als das ARPANET eingerichtet wurde, schuf das US-Verteidigungsministerium einen informellen Ausschuss zu dessen Überwachung. 1983 wurde der Ausschuss in **IAB** (*Internet Activities Board*) umbenannt. Gleichzeitig erhielt er eine etwas breitere Mission, nämlich die am ARPANET und Internet beteiligten Wissenschaftler mehr oder weniger in einer Richtung zu halten. Diese Aufgabe kam in etwa dem Zureiten von Wildpferden gleich. Die Bedeutung des Akryoms „IAB“ wurde später in **Internet Architecture Board** umbenannt.

Jedes der ungefähr zehn Mitglieder des IAB stand einem Expertenteam vor. Das IAB traf sich mehrmals im Jahr, um die Ergebnisse zu besprechen und um das US-Verteidigungsministerium und die NSF (die damals beide die meisten Finanzmittel zur Verfügung stellten) mit Rückmeldungen zu versorgen. Wurde ein Standard benötigt (z.B. ein neuer Routing-Algorithmus), setzten sich die IAB-Mitglieder hin, um etwas Passendes zu entwickeln, und machten die Arbeit dann publik, damit Studenten der höheren Semester, die den Kern der Softwareentwicklung bildeten, die neue Technik implementieren konnten. Die Kommunikation erfolgte durch eine Reihe technischer Berichte namens **RFCs** (*Request For Comments*). RFCs werden online gespeichert und können von jedem Interessierten unter der Webadresse www.ietf.org/rfc abgerufen werden. Sie sind in chronologischer Reihenfolge nach dem Datum ihres Entstehens nummeriert. Derzeit gibt es über 6 000. Wir werden in diesem Buch auf viele RFCs Bezug nehmen.

Bis 1989 nahm das Internet derart an Umfang zu, dass dieser höchst informelle Stil nicht mehr funktionierte. Viele Anbieter vertrieben bereits TCP/IP-Produkte und wollten sie nicht ändern, nur weil sich zehn Entwickler etwas Besseres ausdachten. Im Sommer 1989 wurde das IAB erneut umorganisiert. Die Entwickler wurden der **IRTF** (*Internet Research Task Force*) zugeordnet, die als Teilbereich in das IAB eingegliedert wurde, ebenso wie die **IETF** (*Internet Engineering Task Force*). Das IAB setzte sich aus Leuten zusammen, die außer die Forschungsgemeinde auch andere Organisationen vertraten. Anfangs war das eine rotierende Gruppe, deren Mitglieder sich alle zwei Jahre abwechselten. Später wurde die **Internet Society** ins Leben gerufen. Sie setzte sich aus Leuten zusammen, die am Internet interessiert waren. Die Internet Society ist somit in gewissem Sinn mit ACM oder IEEE vergleichbar. Sie wird von gewählten Treuhändern beaufsichtigt, die die IAB-Mitglieder benennen.

Mit dieser Aufteilung wurde bezweckt, dass sich die IRTF auf langfristige Forschungsprojekte konzentriert, während sich die IETF mit kurzfristigen technischen Fragen befassen sollte. Die IETF wurde in Arbeitsgruppen aufgeteilt, die sich jeweils mit einer bestimmten Problemlösung befassten. Die Vorsitzenden dieser Arbeitsgruppen trafen sich anfangs als Lenkungsausschuss des jeweiligen Projekts. Zu den Sachgebieten der Arbeitsgruppen zählen neue Anwendungen, Benutzerinformationen, OSI-Integration, Routing und Adressierung, Sicherheit, Netzverwaltung und Standards. Schließlich gab

es derart viele Arbeitsgruppen (über 70), dass sie wiederum in Bereiche unterteilt wurden. Diesen Bereichen stehen Bereichsleiter vor, die dem Lenkungsausschuss berichten.

Darüber hinaus wurde ein formelles Standardisierungsverfahren nach dem ISO-Muster umgesetzt. Um als **Standardvorschlag** (*Proposed Standard*) abgesegnet zu werden, muss eine Grundidee in einem RFC erläutert werden. Ferner muss sie in der Gemeinde ausreichend Interesse gewinnen, um überhaupt in Betracht gezogen zu werden. Ein Standardvorschlag avanciert zum **Standardentwurf** (*Draft Standard*), wenn es eine funktionierende Implementierung gibt, die sorgfältig an zwei unabhängigen Standorten mindestens vier Monate lang getestet wurde. Ist das IAB davon überzeugt, dass die Idee ausreichend fundiert ist und die Software fehlerfrei läuft, dann kann es den RFC als **Internetstandard** deklarieren. Einige Internetstandards wurden vom US-Verteidigungsministerium übernommen (MIL-STD). Dadurch wird ein solcher Standard für die Zulieferer des Ministeriums zwingend.

Für Webstandards entwickelt das **World Wide Web Consortium** (**W3C**) Protokolle und Richtlinien, um das langfristige Wachstum des Webs zu unterstützen. Dies ist ein Industriekonsortium, das von Tim Berners-Lee geführt wird und 1994 eingerichtet wurde, als das Web gerade seinen Siegeszug antrat. W3C hat heute mehr als 300 Mitglieder aus der ganzen Welt und hat mehr als 100 W3C-Empfehlungen, wie die Standards hier heißen, herausgegeben, welche Themen wie HTML und Webanonymität behandeln.

1.7 Metrische Einheiten

Um etwaige Verwirrungen zu vermeiden, weisen wir hier explizit darauf hin, dass in dem vorliegenden Buch, wie in der Informatik im Allgemeinen, das metrische Maßsystem anstelle der traditionellen angelsächsischen Einheiten verwendet wird. Die grundlegenden metrischen Präfixe werden in ► Abbildung 1.39 aufgelistet. Die Präfixe werden in der Regel mit den ersten Buchstaben abgekürzt, wobei Einheiten größer als 1 mit Großbuchstaben geschrieben werden (KB, MB etc.). Eine Ausnahme ist aus historischen Gründen kbit/s. So überträgt eine Übertragungsleitung mit 1 Mbit/s 10^6 Bit/s und eine 100-ps-Uhr tickt alle 10^{-10} Sekunden. Da Milli und Mikro beide mit dem Buchstaben „M“ beginnen, musste man sich hier entscheiden. Normalerweise wird „m“ für Milli und „μ“ (der griechische Buchstabe mü) für Mikro verwendet.

Weiterhin muss beachtet werden, dass bei den in der Industrie üblichen Angaben für Speicher-, Festplatten-, Datei- und Datenbankgrößen die Einheiten eine etwas andere Bedeutung haben. Kilo bedeutet hier 2^{10} (1 024) anstatt 10^3 (1 000), weil Speichergrößen immer eine Potenz von 2 sind. So enthält 1 KB Speicher 1 024 Byte und nicht 1 000 Byte. Beachten Sie außerdem, dass der Großbuchstabe „B“ in dieser Verwendung „Byte“ bedeutet (Einheiten von 8 Bit), wohingegen der Kleinbuchstabe „b“ für „Bit“ steht. Ebenso enthält 1 MB Speicher 2^{20} (1 048 576) Byte, 1 GB Speicher enthält 2^{30} (1 073 741 824) Byte und eine Datenbank mit 1 TB enthält 2^{40} (1 099 511 627 776) Byte. Eine Übertragungsleitung mit 1 kbit/s überträgt jedoch 1000 Bit pro Sekunde und ein 10-Mbit/s-LAN operiert mit 10 000 000 Bit/s, da diese Geschwindigkeiten keine Potenz von 2 sind. Leider bringen viele diese beiden Systeme durcheinander,

vor allem bei der Größe von Festplatten. Um Mehrdeutigkeiten zu vermeiden, verwenden wir die Bezeichnungen KB, MB, GB und TB für 2^{10} , 2^{20} , 2^{30} und 2^{40} Byte sowie die Bezeichnungen kbit/s, Mbit/s, Gbit/s und Tbps für 10^3 , 10^6 , 10^9 und 10^{12} Bit/s.

Abbildung 1.39: Die wichtigsten metrischen Präfixe.³

1.8 Überblick über das restliche Buch

In diesem Buch werden sowohl die Grundlagen sowie der praktische Einsatz von Rechnernetzen behandelt. Die meisten Kapitel beginnen mit einer Diskussion der relevanten Prinzipien, gefolgt von mehreren Beispielen, die diese Prinzipien verdeutlichen. Die Beispiele stammen in der Regel aus dem Internet oder drahtlosen Netzen wie mobilen Funknetzen, da diese beide wichtig, aber sehr verschieden sind. Soweit relevant, werden andere Beispiele angeführt.

Das Buch folgt in seinem Aufbau dem in Abbildung 1.23 dargestellten Hybridmodell. In *Kapitel 2* beginnen wir, uns von unten durch die Protokollhierarchie hochzuarbeiten. Wir präsentieren Hintergrundinformationen aus dem Bereich der Datenkommunikation, sowohl für kabelbasierte als auch für drahtlose Übertragungssysteme. Diese Themen betreffen die Sendung von Informationen über physische Kanäle. Allerdings werden keine Hardware-, sondern nur Architekturaspekte behandelt. Einige Beispiele der Bitübertragungsschicht wie öffentliche Telefonnetze, Mobilfunknetze und Kabelfernsehen werden zudem erörtert.

Kapitel 3 und *4* behandeln die Sicherungsschicht in zwei Teilen. In *Kapitel 3* sehen wir uns an, wie Pakete über eine Verbindung gesendet werden, einschließlich der Themen Fehlererkennung und -behebung. Wie werfen einen Blick auf DSL (welches

3 Für die binären Einheiten wird heute zunehmend die Notation Kibibyte (KiB, 2^{10} Byte = 1.024 Byte), Mebibyte (MiB, 2^{20} Byte = 1.048.576 Byte), Gigabyte (GiB, 2^{30} Byte = 1.073.741.824 Byte) etc. verwendet.

für den Breitbandinternetzugang über Telefonleitungen verwendet wird) als ein Beispiel aus der realen Welt eines Protokolls für die Sicherungsschicht.

In *Kapitel 4* untersuchen wir die MAC-Teilschicht. Diese ist der Teil der Sicherungsschicht, welche die gemeinsame Nutzung eines Kanals von mehreren Rechnern regelt. Unter den vorgestellten Beispielen sind drahtlose LANs wie IEEE 802.11 und RFID sowie verdrahtete LANs wie klassisches Ethernet. Sicherungsschicht-Switches zum Verbinden von LANs, wie beispielsweise Switched Ethernet, werden ebenfalls hier behandelt.

In *Kapitel 5* geht es um die Vermittlungsschicht mit dem besonderen Schwerpunkt Routing. Es werden viele statische und dynamische Routing-Algorithmen behandelt. Tritt mehr Netzverkehr auf, als das Netz verwalten kann, werden einige Pakete verspätet oder überhaupt nicht ankommen, selbst wenn ein guter Routing-Algorithmus verwendet wird. Wir werden dieses Problem vom Verhindern der Überlastung bis hin zur Garantie einer bestimmten Dienstgüte erörtern. Auch die Verbindung heterogener Netze zu Internetworks bringt unzählige Probleme mit sich, die hier besprochen werden. Die Vermittlungsschicht im Internet wird ausführlich dargestellt.

Kapitel 6 behandelt die Transportschicht. Dabei wird großes Gewicht auf verbindungsorientierte Protokolle und auf Zuverlässigkeit gelegt, da dies von vielen Anwendungen benötigt wird. Beide Internettransportprotokolle, UDP und TCP, werden im Detail unter Berücksichtigung von Leistungsaspekten vorgestellt.

Kapitel 7 befasst sich mit der Anwendungsschicht, ihren Protokollen und Anwendungen. Als Erstes wird DNS – das Telefonbuch des Internets – behandelt. Als Nächstes werden E-Mails mit einer Erörterung der Protokolle untersucht. Dann wenden wir uns dem Web zu. Wir erörtern ausführlich statische und dynamische Inhalte, sowie die Vorgänge auf der Client- und der Serverseite. Anschließend werfen wir einen Blick auf vernetzte Multimedia-Anwendungen, darunter Audio- und Videostreaming. Zum Schluss besprechen wir Content-Delivery-Netze und Peer-to-Peer-Technologie.

Kapitel 8 behandelt die Sicherheitsaspekte im Netz. Dieses Themengebiet bezieht sich auf alle Schichten; daher wird es am Ende nach der ausführlichen Darstellung aller Schichten behandelt. Das Kapitel beginnt mit einer Einführung in die Kryptografie. Im Folgenden wird dann gezeigt, wie man mit Verschlüsselungstechniken die Kommunikation, E-Mails und das Web sicher gestalten kann. Das Kapitel schließt mit einer Erörterung von Bereichen, in denen die Sicherheit mit der Privatsphäre, der Meinungsfreiheit, Zensur und anderen sozialen Themen kollidiert.

Kapitel 9 enthält eine kommentierte Liste mit Leseempfehlungen zu den einzelnen Kapiteln. Dies ist für die Leser gedacht, die ihre Studien zum Thema Netze weiter vertiefen möchten. Dieses Kapitel enthält auch ein alphabetisches Literaturverzeichnis aller Quellen, die in dem vorliegenden Buch zitiert wurden.

Die Website des Autors bei Pearson

<http://www.pearsonhighered.com/tanenbaum>

enthält eine Seite mit Links auf viele Tutorials, häufig gestellte Fragen (FAQs), Unternehmen, Fachverbände, Standardisierungsgremien, Technologien, Artikel und vieles mehr.

Zusammenfassung

Rechnernetze haben viele Einsatzmöglichkeiten, sowohl für Unternehmen als auch für Privatpersonen, zu Hause oder unterwegs. Unternehmen verwenden Netze aus Computern, um Unternehmensinformationen gemeinsam zu nutzen. In der Regel sind diese nach dem Client-Server-Modell aufgebaut, wobei die Desktop-Rechner auf den Schreibtischen der Mitarbeiter als Clients auf leistungsstarke Server im Rechenzentrum zugreifen. Für Privatleute bieten Netze Zugang zu einer Vielfalt an Informationen und Unterhaltungsressourcen, ebenso wie einen Weg, Produkte und Dienstleistungen zu kaufen und zu verkaufen. Privatpersonen greifen zu Hause häufig über Ihren Telefon- oder Kabelbetreiber auf das Internet zu, obwohl zunehmend drahtloser Zugang für Laptops und Telefone verwendet wird. Technologische Fortschritte ermöglichen neue Arten von mobilen Anwendungen und Netzwerken, wobei Rechner in Haushalts- und anderen Endgeräten eingebettet sind. Durch diese Fortschritte tauchen gleichzeitig gesellschaftliche Probleme auf, die zum Beispiel die Privatsphäre betreffen.

Netze können grob in LANs, MANs, WANs und Internetworks unterteilt werden. LANs sind in der Regel auf Gebäude beschränkt und arbeiten in hoher Geschwindigkeit. MANs erstrecken sich gewöhnlich über eine Stadt. Ein Beispiel ist das Kabelfernsehsystem, das heutzutage auch von vielen Personen für den Zugang ins Internet genutzt wird. WANs können sich über ein Land oder einen Kontinent erstrecken. Einige der zum Aufbau dieser Netze eingesetzten Technologien sind Punkt-zu-Punkt-Verbindungen (z.B. ein Kabel), während andere Broadcast-Netze (z.B. drahtlos) sind. Netzwerke können mit Routern miteinander verbunden werden und bilden somit Internetworks, von denen das Internet das größte und bekannteste Beispiel ist. Drahtlose Netze, z.B. IEEE-802.11-LANs und 3G-Mobilfunk, gewinnen ebenfalls zunehmend an Beliebtheit.

Netzsoftware ist um Protokolle herum aufgebaut. Dies sind die Regeln, durch die Prozesse kommunizieren können. Die meisten Netze unterstützen Protokollhierarchien, wobei jede Schicht der nächsten Schicht gewisse Dienste bietet und sie von den Einzelheiten der Protokolle der unteren Schichten abschirmt. Protokollstapel basieren normalerweise auf dem OSI- oder dem TCP/IP-Modell. Beide Modelle haben eine Netzzugangs- bzw. Sicherungs-, eine Vermittlungs-, eine Transport- und eine Anwendungsschicht, unterscheiden sich aber in den anderen Schichten. Zu den Entwurfsaspekten gehören Zuverlässigkeit, Ressourcenzuweisung, Wachstum, Sicherheit und anderes mehr. Ein großer Teil dieses Buches beschäftigt sich mit Protokollen und deren Entwurf.

Netze stellen für ihre Benutzer verschiedene **Dienste** zur Verfügung. Diese Dienste reichen von der verbindungslosen Best-Effort-Paketzustellung bis zu verbindungsorientierter Sendung mit Dienstgütegarantien. In einigen Netzen wird auf einer Schicht ein verbindungsloser Dienst bereitgestellt und in der Schicht darüber ein verbindungsorientierter Dienst.

Bekannte Netze sind das Internet, das 3G-Mobilfunknetz und drahtlose IEEE-802.11-LANs. Das Internet ist aus dem ARPANET hervorgegangen, das durch die weitere Anbindung von anderen Netzwerken zu einem Internetwork wurde. Das aktuelle Internet ist eigentlich eine Ansammlung von vielen Tausenden von Netzen, die den TCP/IP-Protokollstapel verwenden. Das 3G-Mobilfunknetz stellt drahtlosen und mobilen Internetzugang zu Geschwindigkeiten vom mehreren Mbit/s zur Verfügung und überträgt natürlich auch Sprachanrufe. Drahtlose LANs, die auf dem 802.11-Standard der IEEE basieren, sind in vielen Haushalten und Cafés installiert und können Übertragungsraten von mehr als 100 Mbit/s zur Verfügung stellen. Auch neue Netzwerkarten entstehen, wie eingebettete Sensornetze und Netze, die auf der RFID-Technologie beruhen.

Damit mehrere Computer miteinander kommunizieren können, ist ein erheblicher **Standardisierungsaufwand** bei Hardware und Software vonnöten. Organisationen wie ITU-T, ISO, IEEE und IAB betreuen verschiedene Teile des Standardisierungsprozesses.



Lösungs-
hinweise

Übungsaufgaben

- 1** Stellen Sie sich vor, Sie hätten Ihren Bernhardiner Bernie darauf abgerichtet, anstelle einer Flasche Schnaps eine Schachtel mit drei 8-mm-Bändern zu tragen. (Wenn Ihre Festplatte überläuft, halten Sie dies für einen Notfall.) Diese Bänder fassen je 7 Gbyte. Der Hund kann Ihren Aufenthaltsort mit einer Geschwindigkeit von 18 km/Stunde erreichen, egal wo Sie sich befinden. Bei welcher Entfernung erreicht Bernie eine höhere Datenrate als eine Übertragungsleitung mit 150 Mbit/s? Wie ändert sich Ihre Antwort, wenn
 - a. Bernie seine Geschwindigkeit verdoppelt?
 - b. jede Bandkapazität verdoppelt wird?
 - c. die Datenrate der Übertragungsleitung verdoppelt wird?
- 2** Eine Alternative zu einem LAN ist ein großes Timesharing-System mit Terminals für alle Benutzer, die darauf zugreifen können. Nennen Sie zwei Vorteile eines Client-Server-Systems mit einem LAN.
- 3** Die Leistung eines Client-Server-Systems wird von zwei Netzfaktoren stark beeinflusst: der Bandbreite des Netzes (wie viele Bit/s übertragen werden können) und der Latenz (wie viele Sekunden es dauert, bis das erste Bit vom Client zum Server gelangt). Nennen Sie ein Beispiel für ein Netz, das eine hohe Bandbreite, aber auch eine hohe Latenz aufweist. Geben Sie ein weiteres Beispiel für ein Netz, das sowohl eine niedrige Bandbreite als auch eine geringe Latenz hat.
- 4** Welcher weitere Parameter ist neben Bandbreite und Latenz erforderlich, um eine gute Charakterisierung der Dienstgüte eines Netzes für die Übertragung von
 - a. digitalisierter Sprache
 - b. Video
 - c. finanziellen Transaktionen
 zu geben?

- 5** Ein Faktor bei der Verzögerung eines Store-and-forward-Paketvermittlungssystems ist, wie lange es dauert, um ein Paket über eine Schaltung (Switch) zu speichern und weiterzuleiten. Wenn die Umschaltzeit $10 \mu\text{s}$ beträgt, ist dies dann für die Antwortzeit in einem Client-Server-System ein kritischer Faktor, bei dem der Client sich in New York und der Server sich in Kalifornien befindet? Gehen Sie davon aus, dass die Ausbreitungsgeschwindigkeit im Kabel $2/3$ der Lichtgeschwindigkeit im Vakuum beträgt.
- 6** Ein Client-Server-System verwendet ein Satellitennetz mit einem Satelliten auf $40\,000$ km Höhe. Wie sieht im besten Fall die Verzögerung einer Antwort auf eine Anforderung aus?
- 7** In der Zukunft, wenn jeder mit seinem Heim-PC an ein Rechnernetz angeschlossen ist, werden auch öffentliche Befragungen oder Entscheide zu wichtigen anstehenden Gesetzgebungen möglich. Eventuell könnten bestehende Gesetze geändert oder abgeschafft werden, wenn die Bürger ihren Willen und ihre Meinung in großer Zahl direkt ausdrücken können. Die positiven Aspekte dieser Art von direkter Demokratie liegen auf der Hand. Welche negativen Aspekte könnten sich Ihrer Meinung nach dadurch ergeben?
- 8** Fünf Router sollen an ein Punkt-zu-Punkt-Subnetz angeschlossen werden. Zwischen jedem Router-Paar kann der Entwickler eine Leitung mit hoher, mittlerer oder niedriger Geschwindigkeit oder keine Leitung verwenden. Wenn es 100 ms an Rechenzeit dauert, um jede Topologie zu erzeugen und zu prüfen, wie lange dauert es dann, um alle zu prüfen?
- 9** Ein Nachteil bei Broadcast-Subnetzen ist die Verschwendungen von Netzkapazität, wenn mehrere Hosts gleichzeitig auf den Kanal zugreifen wollen. Stellen Sie sich als einfaches Beispiel vor, dass die Zeit in diskrete Abschnitte (Zeitscheiben) aufgeteilt wird und jeder der n Hosts mit einer Wahrscheinlichkeit von p während einer Zeitscheibe versucht, auf den Kanal zuzugreifen. Wie hoch ist der Anteil der Zeitscheiben, der aufgrund von Kollisionen nicht genutzt wird?
- 10** Nennen Sie zwei Gründe, die für die Verwendung von Schichtprotokollen sprechen. Welches ist ein möglicher Nachteil von Schichtprotokollen?
- 11** Der Geschäftsführer des Unternehmens Sonderfarbe möchte mit einer örtlichen Brauerei zusammenarbeiten, um eine unsichtbare Bierdose (als Maßnahme gegen den Verpackungsmüll) zu produzieren. Er weist seine Rechtsabteilung an, sich der Angelegenheit anzunehmen. Diese wiederum wendet sich an die Konstruktionsabteilung um Hilfe. Als Ergebnis ruft der leitende Ingenieur seinen Kollegen in der Brauerei an, um technische Aspekte des Projekts zu besprechen. Die beiden Ingenieure berichten das Ergebnis ihrer jeweiligen Rechtsabteilung. Diese beginnen, die rechtlichen Grundlagen der Zusammenarbeit per Telefon miteinander auszuarbeiten. Schließlich diskutieren die Geschäftsführer der beiden Unternehmen die finanziellen Aspekte der Zusammenarbeit. Welche Prinzipien eines mehrschichtigen Protokolls im Sinne des OSI-Modells verletzt diese Kommunikationsstruktur?
- 12** Zwei Netze bieten zuverlässigen verbindungsorientierten Dienst. Eines davon bietet einen zuverlässigen Bytestrom und das andere einen zuverlässigen Nachrichtenstrom. Sind diese identisch? Falls ja, warum wird diese Unterscheidung gemacht? Falls nicht, geben Sie ein Beispiel an, um den Unterschied zu verdeutlichen.
- 13** Was bedeutet „Verhandlung“ bei Netzprotokollen? Führen Sie ein Beispiel an.
- 14** In Abbildung 1.19 wird ein Dienst dargestellt. Sind in dieser Abbildung noch weitere Dienste implizit vorhanden? Falls ja, wo? Wenn nicht, warum nicht?

- 15** Bei den meisten Netzen behandelt die Sicherungsschicht Übertragungsfehler, indem sie fordert, fehlerhafte Rahmen erneut zu übertragen. Wie oft muss ein Rahmen durchschnittlich erneut übertragen werden, wenn die Wahrscheinlichkeit eines fehlerhaften Rahmens p ist? Gehen Sie davon aus, dass die Bestätigungen nie verloren gehen.
- 16** Ein System hat eine Protokollhierarchie mit n Schichten. Anwendungen erzeugen Nachrichten in einer Länge von M Byte. Auf jeder Schicht wird ein Header mit h Byte hinzugefügt. Welcher Anteil der Netzbänderbreite wird von den Headern belegt?
- 17** Was ist der hauptsächliche Unterschied zwischen TCP und UDP?
- 18** Das Subnetz in Abbildung 1.25b wurde so konzipiert, dass es einen Atomkrieg überdauert. Wie viele Bomben sind nötig, um die Knoten in zwei getrennte Gruppen aufzuteilen? Gehen Sie davon aus, dass eine Bombe einen Knoten und alle damit verbundenen Links auslöscht.
- 19** Die Größe des Internets verdoppelt sich ungefähr alle 18 Monate. Obwohl es keiner mit Sicherheit weiß, gehen vorsichtige Schätzungen davon aus, dass die Anzahl der Hosts im Internet bis 2009 auf 600 Millionen gestiegen ist. Berechnen Sie anhand dieser Daten die bis zum Jahr 2018 erwartete Anzahl an Hosts im Internet. Glauben Sie dies? Erklären Sie Ihre Antwort.
- 20** Bei der Übertragung einer Datei zwischen zwei Rechnern sind zwei Bestätigungsstrategien möglich. Bei der ersten wird die Datei in Pakete zerlegt, die einzeln vom Empfänger bestätigt werden. Der Dateitransfer als Ganzes wird aber nicht bestätigt. Bei der zweiten werden die Pakete nicht einzeln bestätigt. Vielmehr wird die gesamte Datei bei Ankunft bestätigt. Diskutieren Sie die zwei Ansätze.
- 21** Betreiber von mobilen Funknetzen müssen wissen, wo sich die Mobiltelefone ihrer Teilnehmer (also ihre Benutzer) aufhalten. Erklären Sie, warum dies ungünstig für die Nutzer ist. Geben Sie dann Gründe dafür an, warum es gut für die Nutzer ist.
- 22** Wie lang ist ein Bit beim ursprünglichen IEEE-802.3-Standard in Metern? Gehen Sie von einer Übertragungsgeschwindigkeit von 10 Mbit/s sowie davon aus, dass die Geschwindigkeit im Koaxialkabel $2/3$ der Lichtgeschwindigkeit im Vakuum beträgt.
- 23** Eine Grafik ist 1 600 1 200 Pixel mit 3 Byte/Pixel groß. Die Grafik ist nicht komprimiert. Wie lange dauert die Übertragung über einen Modemkanal mit 56 kbit/s? Über ein Kabelmodem mit 1 Mbit/s? Über ein Ethernet mit 10 Mbit/s? Über ein Ethernet mit 100 Mbit/s? Über ein Gigabit-Ethernet?
- 24** Ethernet und drahtlose Netze weisen einige Ähnlichkeiten und Unterschiede auf. Eine Eigenschaft des Ethernets ist, dass im Ethernet zu einem Zeitpunkt nur ein Rahmen übertragen werden kann. Gilt diese Ethernet-Eigenschaft auch für IEEE 802.11? Erläutern Sie Ihre Antwort.
- 25** Führen Sie zwei Vor- und zwei Nachteile der Verfügbarkeit internationaler Standards für Netzprotokolle auf.
- 26** Bei einem System mit einer festen und einer austauschbaren Komponente, z. B. einem CD-Laufwerk und einer CD, ist es wichtig, dass das System standardisiert ist, damit verschiedene Hersteller beide Teile produzieren können und alles miteinander kompatibel ist. Führen Sie drei Beispiele von solchen internationalen Standards außerhalb der Computerindustrie auf. Dann nennen Sie drei Bereiche, ebenfalls außerhalb der Computerindustrie, in denen solche Standards nicht vorhanden sind.

- 27** Angenommen der Algorithmus, der für die Implementierung der Operationen in Schicht k eingesetzt wird, wird geändert. Welche Auswirkungen hat die auf Operationen in den Schichten $k-1$ und $k+1$?
- 28** Angenommen es gibt eine Änderung im Dienst (der Menge der Operationen), die von Schicht k angeboten wird. Welche Auswirkungen hat die auf Dienste in den Schichten $k-1$ und $k+1$?
- 29** Erstellen Sie eine Liste von Begründungen, warum die Antwortzeit eines Clients größer sein kann als die Verzögerung im besten Fall.
- 30** Erstellen Sie eine Liste Ihrer täglichen Aktivitäten, bei denen Sie Computernetze verwenden. Wie würde sich Ihr Leben ändern, wenn diese Netze plötzlich abgeschaltet würden?
- 31** Finden Sie heraus, wofür Netze an Ihrer Ausbildungsstätte oder Ihrem Arbeitsplatz verwendet werden. Beschreiben Sie die hier verwendeten Netztypen, Topologien und Switching-Methoden.
- 32** Mit dem *ping*-Programm können Sie ein Testpaket an einen bestimmten Ort senden und prüfen, wie lange es dorthin und wieder zurück braucht. Testen Sie mit *ping*, wie lange eine Datenübertragung von Ihrem Standort zu anderen bekannten Standorten benötigt. Zeichnen Sie anhand der Daten die Übertragungszeit über das Internet als Funktion der Entfernung auf. Am besten verwenden Sie Universitäten, da der Standort der Server ganz genau bekannt ist. So befindet sich *berkeley.edu* beispielsweise in Berkeley in Kalifornien, *mit.edu* in Cambridge in Massachusetts, *vu.nl* in den Niederlanden in Amsterdam, *www.usyd.edu.au* im australischen Sydney und *www.uct.ac.za* in Kapstadt in Südafrika.
- 33** Gehen Sie zur Website von IETF, *www.ietf.org*, und sehen Sie sich die Aktivitäten der Organisation genauer an. Wählen Sie ein Projekt aus, und schreiben Sie einen halbseitigen Bericht über das Problem und die vorgeschlagene Lösung.
- 34** Das Internet setzt sich aus vielen Netzen zusammen. Deren Anordnung legt die Topologie des Internets fest. Sehr viele Informationen über die Topologie des Internets sind online verfügbar: Finden Sie unter Verwendung einer Suchmaschine Genaueres über die Internettopologie heraus und erstellen Sie einen kurzen Bericht über die gefundenen Ergebnisse.
- 35** Durchsuchen Sie das Internet, um einige wichtige Peering-Punkte herauszufinden, die aktuell zur Weiterleitung von Paketen im Internet benutzt werden.
- 36** Schreiben Sie ein Programm, das den Nachrichtenfluss von der obersten Schicht zur untersten Schicht des 7-Schichten-Protokollmodells implementiert. Ihr Programm sollte eine separate Protokollfunktion für jede Schicht enthalten. Protokoll-Header sind Folgen aus bis zu 64 Zeichen. Jede Protokollfunktion hat zwei Parameter: eine Nachricht, die vom Protokoll der höheren Schicht (ein Zeichenpuffer) übergeben wird, und die Größe der Nachricht. Diese Funktion hängt ihren Header vor der Nachricht an, druckt die neue Nachricht auf der Standardausgabe und ruft dann die Protokollfunktion des Protokolls der darunterliegenden Schicht auf. Die Programmeingabe ist eine Nachricht der Anwendung (eine Folge von 80 Zeichen oder weniger).

Die Bitübertragungsschicht

| | | |
|------------|---|-----------|
| 2.1 | Theoretische Grundlagen der Datenübertragung | . 121 |
| 2.2 | Gerichtete Übertragungsmedien | 127 |
| 2.3 | Drahtlose Übertragung | 137 |
| 2.4 | Kommunikationssatelliten | 149 |
| 2.5 | Digitale Modulation und Multiplexing | 159 |
| 2.6 | Das öffentliche Telefonnetz | 173 |
| 2.7 | Das Mobiltelefonsystem | 202 |
| 2.8 | Kabelfernsehen | 217 |

» In diesem Kapitel beschäftigen wir uns mit der untersten Schicht in unserem Protokollmodell. Diese definiert die elektrischen, zeitbezogenen und anderen Schnittstellen zum Senden von Bits in Form von Signalen über die Kanäle. Die Bitübertragungsschicht ist die Grundlage, auf der das Netz aufgebaut ist. Die Eigenschaften von verschiedenen Arten von physischen Kanälen bestimmen die Leistungsfähigkeit (z.B. Durchsatz, Latenz- und Fehlerrate). Dies ist also ein guter Ausgangspunkt für unsere Reise in das Land der Netzwerke.

Wir beginnen mit einer theoretischen Analyse der Datenübertragung und werden dabei entdecken, dass Mutter (Eltern?) Natur dem, was über einen Kanal gesendet werden kann, Grenzen auferlegt. Dann sehen wir uns drei verschiedene Übertragungsmedien etwas genauer an: gerichtete Medien (Kupferdraht und Glasfaser), drahtlose (Radio- bzw. Funkübertragung) und Satellitenkommunikation. Jede dieser Technologien hat andere Eigenschaften, die den Entwurf und die Leistungsfähigkeit des jeweiligen Netzes beeinflussen. Dieses Material bildet unser Hintergrundwissen zu den wichtigsten Übertragungstechniken, die heute in modernen Netzen verwendet werden.

Das nächste Thema ist digitale Modulation, bei der es darum geht, wie analoge Signale in digitale Bits umgewandelt werden und umgekehrt. Danach sehen wir uns Multiplexing-Schemata an und erkunden, wie mehrere Gespräche gleichzeitig auf dasselbe Übertragungsmedium gepackt werden können, ohne sich gegenseitig zu stören.

Schließlich werden wir einen Blick auf drei Beispiele von Kommunikationssystemen werfen, die in der Praxis bei Weitverkehrsnetzen eingesetzt werden: das Telefonsystem (Festnetz), das Mobilfunksystem und das Kabelfernsehsystem. Diese sind alle in der Praxis wichtig, deshalb werden wir jedem einen angemessenen Raum geben.



2.1 Theoretische Grundlagen der Datenübertragung

Informationen können über Kabel durch die Variation bestimmter physikalischer Eigenschaften, z.B. Spannung oder Strom, übertragen werden. Durch Darstellung des Wertes dieser Spannung oder des Stroms als einwertige Funktion der Zeit $f(t)$ können wir das Verhalten des Signals modellieren und mathematisch analysieren. Diese Analyse ist Thema der folgenden Abschnitte.

2.1.1 Fourieranalyse

Im frühen 19. Jahrhundert bewies der französische Mathematiker Jean-Baptiste Fourier, dass jede periodische Funktion $g(t)$, die sich einigermaßen vernünftig verhält, mit der Periode T als (möglicherweise unendliche) Summe einer Anzahl von Sinus- und Kosinusfunktionen gebildet werden kann:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft) \quad (2.1)$$

wobei $f=1/T$ die Grundfrequenz ist und a_n und b_n die Sinus- und Kosinusamplituden der n -ten **Harmonischen** sind und c eine Konstante ist. Eine solche Zerlegung nennt man **Fourierreihe**. Aus der Fourierreihe kann die Funktion rekonstruiert werden. Das heißt, wenn die Periode T bekannt ist und die Amplituden gegeben sind, kann die ursprüngliche Funktion der Zeit durch die Berechnung der Summe in Gleichung (2.1) gefunden werden.

Ein Datensignal, das eine endliche Dauer hat (was für alle Datensignale zutrifft), kann man auch so betrachten, dass das gleiche Muster immer wiederholt wird (d.h., das Intervall von T bis $2T$ entspricht dem von 0 bis T usw.).

Die a_n -Amplituden können für jedes gegebene $g(t)$ berechnet werden, indem beide Seiten der Gleichung (2.1) mit $\sin(2\pi kft)$ multipliziert und dann von 0 bis T integriert werden.

Da

$$\int_0^T \sin(2\pi kft) \sin(2\pi nft) dt = \begin{cases} 0 & \text{für } k \neq n \\ T/2 & \text{für } k = n \end{cases}$$

bleibt bei der Summation nur ein Term übrig: a_n . Die Summation mit den b_n fällt ganz weg. Ebenso können wir b_n durch Multiplikation der Gleichung (2.1) mit $\cos(2\pi kft)$ und Integrieren von 0 und T herleiten. Indem wir beide Seiten der Gleichung, wie sie oben dargestellt ist, integrieren, erhalten wir c . Die Ergebnisse dieser Rechnungen lauten wie folgt:

$$a_n = \frac{2}{T} \int_0^T g(t) \sin(2\pi nft) dt \quad b_n = \frac{2}{T} \int_0^T g(t) \cos(2\pi nft) dt \quad c = \frac{2}{T} \int_0^T g(t) dt$$

2.1.2 Signale mit begrenzter Bandbreite

Diese Berechnungen sind für die Datenübertragung deshalb relevant, weil reale Kanäle auf verschiedene Frequenzsignale unterschiedliche Auswirkungen haben. Dazu betrachten wir am besten ein Beispiel: die Übertragung des ASCII-Zeichens „b“, das in einem 8-Bit-Byte codiert ist. Das zu übertragende Bitmuster ist 01100010. Der linke Teil von ► Abbildung 2.1a zeigt die vom übertragenden Rechner erzeugte Spannung. Die Fourieranalyse dieser Spannung ergibt folgende Koeffizienten:

$$a_n = \frac{1}{\pi n} [\cos(\pi n / 4) - \cos(3\pi n / 4) + \cos(6\pi n / 4) - \cos(7\pi n / 4)]$$

$$b_n = \frac{1}{\pi n} [\sin(3\pi n / 4) - \sin(\pi n / 4) + \sin(7\pi n / 4) - \sin(6\pi n / 4)]$$

$$c = 3/4$$

Die Effektivamplituden $\sqrt{a_n^2 + b_n^2}$ für die ersten Terme stehen auf der rechten Seite von Abbildung 2.1a. Diese Werte sind deshalb interessant, weil ihre Quadrate proportional zu der Energie sind, die bei der zugehörigen Frequenz übertragen wird.

Kein Übertragungsgerät kann Signale übermitteln, ohne hierbei etwas Energie zu verlieren. Würden alle Fourierkomponenten gleichermaßen verringert, hätte das entstehende Signal zwar eine geringere Amplitude, wäre aber nicht verzerrt (d.h., es hätte die gleiche nette rechtwinklige Gestalt wie Abbildung 2.1a). Leider verringern alle Übertragungseinrichtungen die verschiedenen Fourierkomponenten um unterschiedliche Beträge und erzeugen dadurch Verzerrungen. Normalerweise werden die Amplituden für ein Kabel zwischen den Frequenzen 0 und f_c (gemessen in Schwingungen/Sekunde oder Hertz (Hz)) fast unverändert übertragen, während alle Frequenzen über dieser Grenzfrequenz abgedämpft werden. Die Breite des Frequenzbereichs, in dem ohne größere Dämpfung übertragen wird, wird als **Bandbreite** (*bandwidth*) bezeichnet. In der Praxis ist die Trennung nicht so streng, sodass die angegebene Bandbreite oftmals von 0 bis zu der Frequenz reicht, bei der die Leistung auf die Hälfte abgefallen ist.

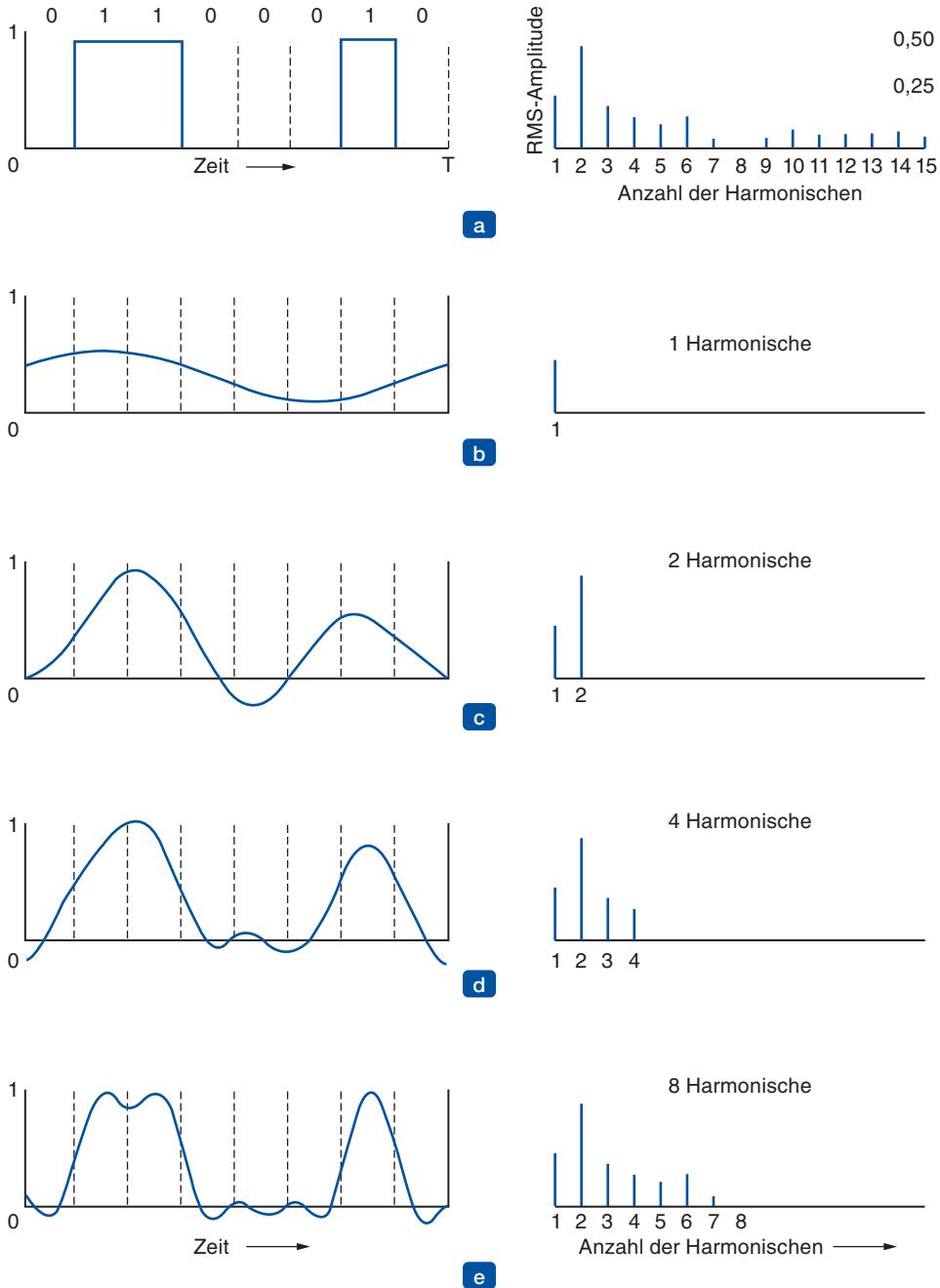


Abbildung 2.1: (a) Binäres Signal und die quadratischen Mittelwerte (RMS, Root Mean Square) seiner Fourieramplituden. (b)–(e) Aufeinanderfolgende Näherungen des ursprünglichen Signals.

Die Bandbreite ist eine physikalische Eigenschaft des Übertragungsmediums, die in der Regel z.B. von dessen Aufbau, der Dicke und der Länge des Kabels oder der Glasfaser abhängt. Häufig werden Filter eingesetzt, um die Bandbreite eines Signals weiter zu begrenzen. Kanäle für drahtloses IEEE 802.11 (WiFi) können beispielsweise bis zu circa 20 MHz benutzen, sodass 802.11-Wellen die Signalbandbreite auf diese Größe filtern. Ein weiteres Beispiel sind traditionelle (analoge) Fernsehkanäle, die jeweils 6 MHz besetzen, auf einem Kabel oder in der Luft. Mithilfe dieses Filterns können mehr Signale einen vorgegebenen Bereich des Spektrums gemeinsam nutzen, was die Gesamteffizienz des Systems verbessert. Dies bedeutet, dass der Frequenzbereich für einige Signale nicht bei null starten wird, aber das macht nichts. Die Bandbreite ist immer noch die Breite des Bandes der durchlaufenen Frequenzen; und die Information, die übertragen werden kann, hängt nur von dieser Breite und nicht von den Anfangs- und Endfrequenzen ab. Signale innerhalb des Bereichs von 0 bis zu einer maximalen Frequenz werden **Basisbandsignale** (*baseband signal*) genannt. Signale, die verschoben werden, um einen höheren Frequenzbereich zu belegen – wie bei allen drahtlosen Übertragungen –, heißen **Durchlassbandsignale** (*passband signal*).

Betrachten wir nun, wie das Signal in Abbildung 2.1a aussehen würde, falls die Bandbreite nur für eine Übertragung der niedrigsten Frequenzen ausreichend wäre (d.h., falls die Funktion durch die ersten Terme von Gleichung (2.1) angenähert würde). ►Abbildung 2.1b zeigt das Signal, das bei einem Kanal entsteht, der nur die erste Harmonische (Grundschwingung f) durchgehen lässt. Abbildung 2.1c–e zeigen die Spektren und rekonstruierten Funktionen für Kanäle mit höheren Bandbreiten. Bei der digitalen Übertragung ist es das Ziel, ein Signal mit gerade so viel Genauigkeit zu empfangen, um die gesendete Bitfolge zu rekonstruieren. Bei dem Signal in Abbildung 2.1e ist dies schon einfach möglich, also wäre es verschwenderisch, mehr Harmonische zu benutzen, um eine noch akkurate Kopie zu empfangen.

Bei einer Bitrate von b Bit/s ist die Zeit, die in unserem Beispiel für die Übertragung von 8 Bit benötigt wird, $8/b$ Sekunden. Folglich beträgt die Frequenz der ersten Harmonischen dieses Signals $b/8$ Hz. Eine normale Telefonleitung mit Sprachqualität (*voice-grade line*) hat eine künstlich eingeführte Grenzfrequenz bei etwa 3 000 Hz. Durch diese Einschränkung liegt die Zahl der höchsten übertragenen Harmonischen bei etwa $3\,000/(b/8)$ oder $24\,000/b$ (die Grenze ist nicht strikt).

Bei einigen Datenübertragungsraten ergeben sich die in ►Abbildung 2.2 aufgeführten Zahlen. Aus diesen Zahlen geht hervor, dass der Versuch, mit 9 600 Bit/s über eine Telefonleitung in Sprachqualität zu übertragen, Abbildung 2.1a in etwas umwandelt, das aussieht wie Abbildung 2.1c, was den korrekten Empfang des binären Bitstroms etwas schwieriger macht. Es ist offensichtlich, dass bei Datenübertragungsraten, die weit über 38,4 kbit/s liegen, für binäre Signale keinerlei Hoffnung besteht, auch wenn die Übertragungseinrichtung absolut rauschfrei ist. Mit anderen Worten, die Einschränkung der Bandbreite begrenzt die Datenübertragungsraten auch bei perfekten Kanälen. Allerdings gibt es ausgefeilte Codierverfahren mit mehreren Spannungsstufen, die höhere Datenübertragungsraten ermöglichen. Dies wird später in diesem Kapitel ausführlicher behandelt.

| Bit/s | T (ms) | 1. Harmonische (Hz) | Anzahl gesendeter Harmonischer |
|-------|--------|---------------------|--------------------------------|
| 300 | 26,67 | 37,5 | 80 |
| 600 | 13,33 | 75 | 40 |
| 1200 | 6,67 | 150 | 20 |
| 2400 | 3,33 | 300 | 10 |
| 4800 | 1,67 | 600 | 5 |
| 9600 | 0,83 | 1200 | 2 |
| 19200 | 0,42 | 2400 | 1 |
| 38400 | 0,21 | 4800 | 0 |

Abbildung 2.2: Beziehung zwischen Datenraten und Harmonischen für unser Beispiel.

Es herrscht viel Verwirrung, wenn es um Bandbreite geht, da dieser Begriff bei Elektroingenieuren und Informatikern unterschiedliche Bedeutung hat. Für Elektroingenieure ist die (analoge) Bandbreite (so wie wir es oben beschrieben haben) eine Quantität, die in Hz gemessen wird. Für Informatiker ist die (digitale) Bandbreite die maximale Datenrate eines Kanals, eine Quantität, die in Bit/s gemessen wird. Diese Datenrate ist das Endresultat, das sich aus Benutzung der analogen Bandbreite eines physischen Kanals für digitale Übertragung ergibt, und die Beziehungen dieser beiden werden wir als Nächstes diskutieren. In diesem Buch wird immer aus dem Kontext hervorgehen, ob wir die analoge Bandbreite (Hz) oder die digitale Bandbreite (Bit/s) meinen.

2.1.3 Maximale Datenübertragungsrate eines Kanals

Schon 1924 hat Henry Nyquist, Ingenieur bei AT&T, erkannt, dass auch ein perfekter Kanal eine begrenzte Übertragungskapazität aufweist. Er leitete einen Ausdruck ab, der die maximale Datenübertragungsrate für einen rauschfreien Kanal mit begrenzter Bandbreite darstellt. 1948 führte Claude Shannon die Arbeit von Nyquist fort und dehnte die Gleichung auf einen Kanal aus, der zufälligem (thermodynamischem) Rauschen unterliegt (Shannon, 1948). Dieser Artikel gilt als der wichtigste Aufsatz in der gesamten Informationstheorie. Wir fassen seine inzwischen klassischen Ergebnisse hier kurz zusammen.

Nyquist bewies: Wenn ein beliebiges Signal durch einen Tiefpassfilter der Bandbreite B geführt wird, kann das gefilterte Signal durch nur $2B$ (exakte) Abtastwerte pro Sekunde vollständig rekonstruiert werden. Mehr als $2B$ Abtastwerte pro Sekunde sind nutzlos, da Anteile mit höherer Frequenz, die durch eine höhere Abtastrate entdeckt werden könnten, bereits ausgefiltert wurden. Besteht ein Signal aus V diskreten Stufen, lautet das Nyquist-Theorem:

$$\text{maximale Datenübertragungsrate} = 2B \log_2 V \text{ Bit/s} \quad (2.2)$$

Ein rauschfreier 3-kHz-Kanal beispielsweise kann binäre (also zweistufige) Signale nicht mit mehr als 6 000 Bit/s übertragen.

Bis jetzt haben wir nur rauschfreie Kanäle betrachtet. Ist zufälliges Rauschen vorhanden, verschlechtert sich die Situation sehr schnell. Zufälliges (thermisches) Rauschen ist aufgrund der Bewegung der Moleküle im System immer vorhanden. Die Intensität des vorhandenen thermischen Rauschens wird durch das Verhältnis der Signalstärke zur Rauschstärke – das **Signal-Rausch-Verhältnis (SNR, Signal-to-Noise Ratio)**, auch als **Rauschabstand** bezeichnet – gemessen. Ist die Signalstärke S und die Rauschstärke N , dann beträgt das Signal-Rausch-Verhältnis S/N . Normalerweise wird dieses Verhältnis auf einer logarithmischen Skala als Wert $10\log_{10} S/N$ angegeben, weil es über einem extrem großen Bereich variieren kann. Die Einheiten dieser logarithmischen Skala werden **Dezibel (dB)** genannt, wobei „dezi“ von 10 kommt und „bel“ zu Ehren von Alexander Graham Bell, dem Erfinder des Telefons, gewählt wurde. Ein S/N -Verhältnis von 10 ist 10 dB, ein Verhältnis von 100 ist 20 dB, das Verhältnis von 1 000 ist 30 dB usw. Die Hersteller von Hifi-Verstärkern geben oft die Bandbreite (den Frequenzbereich) an, über die sich ihr Produkt linear verhält, indem sie die beiden Grenzfrequenzen mit 3 dB Abweichung angeben. An diesen Punkten wird die Verstärkung ungefähr halbiert (da $10 \log_{10} 0,5 \approx -3$).

Shannons wichtigstes Ergebnis ist, dass die maximale Datenübertragungsrate oder **Kapazität** eines rauschenden Kanals mit einer Bandbreite von B Hz und einem Signal-Rausch-Verhältnis von S/N folgendermaßen gegeben ist:

$$\text{maximale Anzahl von Bit/Sekunde} = B \log_2 (1 + S/N) \quad (2.3)$$

Daraus können wir ablesen, welches die besten Kapazitäten sind, die reale Kanäle aufweisen können. Beispielsweise benutzt ADSL (*Asymmetric Digital Subscriber Line*) für den Internetzugang über normale Telefonleitungen eine Bandbreite von ca. 1 MHz. Das Signal-Rausch-Verhältnis hängt stark von der Entfernung des Hauses vom Fernsprechamt ab, und ein SNR von ungefähr 40 dB für kurze Leitungen von ein bis zwei Kilometern ist sehr gut. Mit diesen Charakteristiken kann der Kanal nie mehr als 13 Mbit/s übertragen, gleichgültig, wie viele oder wenige Signalstufen benutzt und wie oft oder selten Abtastwerte abgenommen werden. In der Praxis ist ADSL auf bis zu 12 Mbit/s festgelegt, obwohl Benutzer häufig niedrigere Datenraten bekommen. Diese Datenrate ist in der Tat sehr gut – über 60 Jahre der Kommunikationstechnik haben die Lücke zwischen theoretischer Shannon-Kapazität und der Kapazität von realen Systemen stark verkleinert.

Shannons Ergebnis wurde aus Argumenten der Informationstheorie abgeleitet und ist auf jeden Kanal, bei dem thermisches Rauschen auftritt, anwendbar. Gegenbeispiele fallen in die gleiche Kategorie wie das Perpetuum mobile. Um ADSL auf 13 Mbit/s zu erweitern, muss entweder das SNR verbessert werden (zum Beispiel indem digitale Verstärker in die Leitungen eingefügt werden, die näher an den Kunden sind) oder es muss mehr Bandbreite benutzt werden, so wie bei der Weiterentwicklung ADSL2+.

2.2 Gerichtete Übertragungsmedien

Die Aufgabe der Bitübertragungsschicht ist die Beförderung von Bits von einem Rechner zum anderen. Für die eigentliche Übertragung können verschiedene physikalische Medien eingesetzt werden. Jedes hat in Bezug auf Bandbreite, Verzögerung, Kosten sowie Installationsfreundlichkeit und Wartung seine eigenen Stärken und Schwächen. Medien werden grob in gerichtete (Kupferkabel, Glasfaserkabel) und ungerichtete Medien (Bodenfunk, Satellit und Laserstrahlen) unterteilt. Wir betrachten gerichtete Medien in diesem Abschnitt und ungerichtete in den nächsten Abschnitten.

2.2.1 Magnetische Medien

Eine der gebräuchlichsten Arten, Daten von einem Computer zum anderen zu übertragen, ist immer noch, sie auf Magnetband oder andere transportierbare Medien (wie beschreibbare DVDs) zu schreiben, das Band oder die DVDs zum Zielgerät zu tragen und sie dort wieder einzulesen. Selbst wenn diese Methode nicht so elegant ist wie die Übertragung über einen geostationären Satelliten, ist sie doch oft viel kostengünstiger, vor allem in Anwendungsbereichen, bei denen eine hohe Bandbreite oder die Kosten pro übertragenem Bit von Bedeutung sind.

Das wird an einer einfachen Rechnung deutlich: Ein branchenübliches Ultrium-Band hat eine Speicherkapazität von 200 GB. In einen Karton mit der Abmessung 60×60×60 cm passen etwa 1 000 solcher Bänder. Das ergibt eine Gesamtkapazität von 800 TB bzw. 6 400 Tbit (6,4 Pbit). Dieser Karton kann in den USA innerhalb von 24 Stunden durch Federal Express oder einen anderen Kurierdienst an einen beliebigen Ort befördert werden. Die effektive Bandbreite dieser Übertragung ist 6 400 Tbit/86 400 s, also ein wenig mehr als 70 Gbit/s. Ist das Ziel auf dem Landweg nur eine Stunde entfernt, erhöht sich die Bandbreite auf über 1 700 Gbit/s. Kein Rechnernetz kann dies auch nur näherungsweise erreichen. Natürlich werden Netze schneller, doch die Bandaufzeichnungsdichte steigt ebenfalls.

Betrachtet man die Kosten, ergibt sich ein ähnliches Bild: Der Preis für ein Ultrium-Band liegt bei etwa 40 US-Dollar, wenn man sie in größeren Mengen einkauft. Ein Band kann mindestens zehnmal wiederverwendet werden, sodass sich ein Preis von etwa 4 000 US-Dollar pro Karton und Nutzung ergibt. Addiert man 1 000 US-Dollar für den Transport (vermutlich viel weniger), ergeben sich ungefähr 5 000 US-Dollar für die Beförderung von 800 TB. Dies ergibt Lieferkosten von etwas weniger als einem halben US-Cent pro GB. Dies kann kein Netz schlagen. Und die Moral von der Geschichte:

Unterschätze nie die Bandbreite eines mit Magnetbändern voll beladenen Lkws, der über die Autobahn braust.

2.2.2 Twisted-Pair-Kabel

Obwohl die Bandbreiteneigenschaften von Magnetbändern hervorragend sind, lassen die Verzögerungszeiten zu wünschen übrig. Sie werden in Stunden und Minuten gemessen, nicht in Millisekunden. Für viele Anwendungen ist eine Onlineverbindung unabdingbar. Das älteste und immer noch gebräuchlichste Übertragungsmedium ist das **Twisted-Pair-Kabel** (verdrilltes Leitungspaar). Ein in der Regel 1 mm dickes Twisted-Pair-Kabel besteht aus zwei isolierten Kupferdrähten. Diese sind wie ein DNA-Molekül spiralförmig miteinander verdrillt. Zwei parallele Kabel würden eine gute Antenne bilden, daher sind die Kabel verdrillt. Diese verdrillte Form bewirkt, dass die Wellen durch die Verwindungen gebrochen werden und das Kabel in der Folge weniger Störungen ausstrahlt. Ein Signal wird gewöhnlich als Spannungsdifferenz zwischen den zwei Kabeln übertragen. Dadurch wird bessere Abschirmung gegenüber äußeren Störungen (Rauschen) erreicht, da diese in der Regel beide Kabel gleichzeitig beeinträchtigen, die Differenz aber unverändert lässt.

Die häufigste Verwendung von Twisted-Pair-Kabeln ist im Telefonnetz. Fast alle Telefone sind auf diese Weise an die Vermittlungsstellen angeschlossen. Sowohl Telefonanrufe als auch der ADSL-Internetanschluss laufen über diese Leitungen. Twisted-Pair-Kabel können über mehrere Kilometer hinweg ohne Verstärkung arbeiten, doch bei größeren Entfernen wird das Signal zu sehr gedämpft und es müssen Repeater eingesetzt werden. Laufen viele Twisted-Pair-Kabel über eine längere Strecke parallel, wie z.B. aus einem Hochhaus kommende Leitungen zur Vermittlungsstelle, werden sie in einem Schutzmantel gebündelt. Diese Kabel würden sich hier gegenseitig stören, wenn sie nicht paarweise verdrillt wären. In Teilen der Welt, wo die Telefonleitungen über Masten oberhalb der Erde geführt werden, erreichen diese Bündel einen Durchmesser von einigen Zentimetern.

Twisted-Pair-Kabel können sowohl für analoge als auch für digitale Informationen genutzt werden. Die Bandbreite hängt von der Stärke der Drähte und der Entfernung ab. In vielen Fällen sind einige Mbit/s über Entfernen von wenigen Kilometern möglich. Aufgrund ihrer oft ausreichenden Leistungsfähigkeit und der geringen Kosten werden Twisted-Pair-Kabel derzeit sehr häufig verwendet und dies wird sich in den nächsten Jahren wohl kaum ändern.

Twisted-Pair-Kabel gibt es in verschiedenen Varianten. Der Feld-, Wald- und Wiesentyp, der in vielen Bürogebäuden installiert ist, gehören zur **Kategorie 5** (oder kurz „Cat-5“). Ein Twisted-Pair-Kabel der Kategorie 5 besteht aus zwei Isoliertdrähten, die leicht miteinander verdrillt sein. In der Regel werden vier Paare in einer Plastikummantelung untergebracht, um die Kabel zu schützen und zusammenzuhalten. Dies ist in ►Abbildung 2.3 dargestellt.

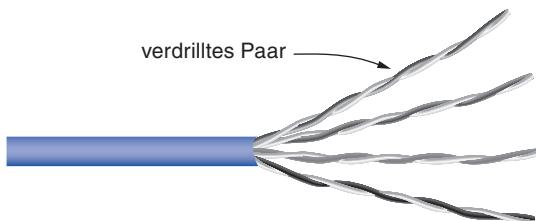


Abbildung 2.3: Kategorie-5-UTP-Kabel mit vier verdrillten Kabeln.

Verschiedene LAN-Standards verwenden Twisted-Pair-Kabel möglicherweise unterschiedlich. Beispielsweise benutzt 100-Mbit/s-Ethernet zwei (von vier) Paaren: ein Paar pro Richtung. Um höhere Geschwindigkeiten zu erreichen, verwendet Gigabit-Ethernet alle vier Paare in beiden Richtungen gleichzeitig. Hier muss der Empfänger das Signal herausfiltern, das lokal übertragen wird.

Ein paar Worte zur allgemeinen Terminologie: Verbindungen, die wie eine zweispurige Straße gleichzeitig in beide Richtungen benutzt werden können, heißen **Voll-duplexverbindungen**. Im Gegensatz dazu heißen Verbindungen, die zwar prinzipiell in beide Richtungen benutzt werden, doch wie eine enge einspurige Straße jeweils immer nur in eine Richtung, **Halbduplexverbindungen**. Zu einer dritten Kategorie gehörten Verbindungen, die Datenverkehr wie Einbahnstraßen nur in einer Richtung zulassen; diese werden **Simplexverbindungen** genannt.

Kommen wir zu den verdrillten Paaren zurück. Cat-5-Kabel haben die bisherigen Kabel der **Kategorie 3** (Cat-3-Kabel) ersetzt, welche ähnlich sind und dieselbe Steckerverbindung besitzen, aber auf einem Meter ein wenig öfter verdrillt sind. Häufigeres Verdrillen führt zu weniger Übersprechen und über größere Entfernung zu einem qualitativ besseren Signal. Daher sind diese Kabel für schnelle Rechnerkommunikation besser geeignet, speziell für 100-Mbit/s- und Gigabit-Ethernet.

Neuere Verkabelung wird wahrscheinlich mit Kabeln der **Kategorie 6** oder sogar **Kategorie 7** durchgeführt. Diese Kategorien haben strengere Spezifikationen für die Behandlung von Signalen mit größeren Bandbreiten. Einige Kabel der Kategorie 6 oder höher sind für Signale von 500 MHz ausgelegt und können 10-Gbit/s-Verbindungen unterstützen, die demnächst installiert werden.

Bis einschließlich Kategorie 6 werden diese Kabeltypen als **UTP-Kabel** (*Unshielded Twisted Pair*) bezeichnet, da sie nur aus Kabeln und Isolierung bestehen. Im Gegensatz dazu haben Kabel der Kategorie 7 sowohl eine Ummantelung um jedes einzelne Paar als auch um das Gesamtkabel (aber innerhalb der Plastikummantelung). Die Abschirmung verringert die Anfälligkeit für externe Interferenzen und das Übersprechen mit anderen Kabeln in der Nähe, um die geforderten Leistungsvorgaben einzuhalten. Cat-7-Kabel erinnern an die hochwertigen, aber sperrigen und teuren geschirmten Twisted-Pair-Kabel, die IBM in den frühen 1980er Jahren eingeführt hat, die seinerzeit aber außerhalb von IBM-Installationen wenig Anklang gefunden haben. Offensichtlich ist es an der Zeit, einen neuen Versuch zu starten.

2.2.3 Koaxialkabel

Ein ebenso weitverbreitetes Übertragungsmedium ist das **Koaxialkabel** (von seinen vielen Anhängern auch liebevoll „Koax“ genannt). Dieses Kabel ist besser abgeschirmt und besitzt eine größere Bandbreite als ungeschirmte verdrillte Kabelpaare, sodass es sich für größere Entfernung und höhere Geschwindigkeiten eignet. Derzeit sind zwei Arten von Koaxialkabeln gebräuchlich. Das 50-Ohm-Kabel wird häufig für digitale Übertragungen verwendet. Der zweite Typ, ein 75-Ohm-Kabel, wird vorwiegend für analoge Übertragungen und Kabelfernsehen eingesetzt. Diese Unterscheidung basiert eher auf historischen als auf technischen Faktoren (so hatten die ersten zweipoligen Antennen z.B. eine Impedanz von 300 Ohm und es war einfach, passende Transformatoren mit einer Impedanz von 4:1 herzustellen). Ab Mitte der 1990er Jahre haben Fernsehbetreiber damit begonnen, Internetzugang über Kabel anzubieten, wodurch das 75-Ohm-Kabel wichtiger für die Datenübertragung wurde.

Ein Koaxialkabel besteht aus einem starren Kupferdraht als Kern, der mit einem Isoliermaterial ummantelt ist. Der Isolator wird wiederum von einer zylindrischen Leiter umschlossen, oftmals ein eng geflochtenes Netz. Der äußere Leiter wird mit einem Plastikschatzmantel abgeschirmt. Ein Querschnitt eines Koaxialkabels ist in ► Abbildung 2.4 dargestellt.

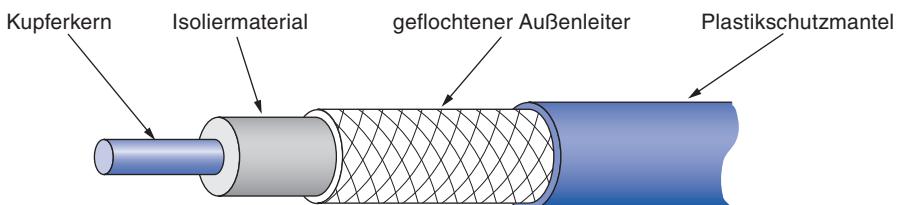


Abbildung 2.4: Koaxialkabel.

Aufbau und Abschirmung des Koaxialkabels verleihen ihm eine gute Kombination aus hoher Bandbreite und ausgezeichneter Rauschunempfindlichkeit. Die mögliche Bandbreite hängt von der Qualität und der Länge des Kabels ab. Moderne Kabel weisen eine Bandbreite von einigen GHz auf. Koaxialkabel wurden früher häufig für Fernleitungen im Telefonnetz eingesetzt, wurden aber inzwischen in Fernnetzen größtenteils von Glasfaserkabeln abgelöst. Koaxialkabel werden jedoch nach wie vor häufig für das Kabelfernsehen und Stadtnetze verwendet.

2.2.4 Trägerfrequenzanlagen

Das Telefon- und das Kabelfernsehnetz sind nicht die einzigen Kabelquellen, die zur Datenkommunikation wiederverwendet werden können. Es gibt eine noch geläufigere Art der Verkabelung: elektrische Stromleitungen. Starkstromleitungen transportieren den elektrischen Strom in Häuser, innerhalb der Häuser sorgt die elektrische Verkabelung für die Verteilung des Stroms auf die Steckdosen.

Stromleitungen zur Datenkommunikation zu verwenden – im Deutschen als **Trägerfrequenzanlage** bezeichnet – ist eine alte Idee. Stromkonzerne benutzen seit vielen Jah-

ren Stromleitungen zur Übertragung mit geringen Datenraten wie entferntes Ablesen von Messdaten, auch die Steuerung von Geräten in Privathaushalten (z.B. mit dem X10-Standard) kann über Stromleitungen erfolgen. In jüngerer Zeit gibt es ein neu erwachtes Interesse an der Kommunikation mit höheren Datenraten über diese Leitungen, sowohl innerhalb von Haushalten für LAN (PowerLAN) als auch außerhalb für Breitbandinternetzugang. Wir konzentrieren uns hier auf das häufigste Szenario: die Verwendung von elektrischen Leitungen innerhalb von Häusern.

Der Vorteil, Stromleitungen für Netze zu benutzen, liegt auf der Hand: Einfach ein Fernsehergerät und einen Receiver an die Steckdose anschließen – was man sowieso tun muss, weil diese Geräte Strom brauchen –, und schon können Filme über die Stromleitungen gesendet und empfangen werden. Diese Konstellation ist in ► Abbildung 2.5 zu sehen. Es gibt keine weiteren Anschlüsse oder Funkwellen. Das Datensignal wird auf das niedrigfrequente Stromsignal aufgesetzt (auf dem aktiven oder „heißen“ Kabel), da beide Signale das Kabel gleichzeitig benutzen.

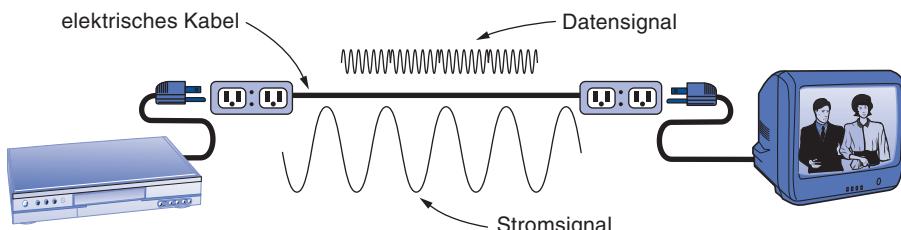


Abbildung 2.5: Netzwerk, das die elektrischen Leitungen im Haushalt benutzt.

Die Schwierigkeit bei der Verwendung von elektrischen Leitungen des Haushalts für ein Netz ist, dass die Leitungen entworfen wurden, um Strom zu transportieren. Diese Aufgabe ist etwas ganz anderes als die Verbreitung von Datensignalen, wofür die Hausverkabelung fürchterlich schlecht geeignet ist. Elektrische Signale werden mit 50–60 Hz gesendet und die Kabel dämpfen die Signale der viel höheren Frequenz (MHz), die für die Datenkommunikation nötig sind. Die elektrischen Eigenschaften der Verkabelung variieren von Haus zu Haus und ändern sich, wenn Haushaltsgeschäfte an- und abgestellt werden, was dazu führt, dass Datensignale in den Kabeln herumspringen. Transiente Ströme, die entstehen, wenn Geräte an- und abgeschaltet werden, erzeugen elektrische Störgeräusche über einen breiten Frequenzbereich. Und ohne die vorsichtige Verdrillung der Twisted-Pair-Kabel fungieren elektrische Kabel als eine feine Antenne, die externe Signale auffängt sowie eigene aussendet. Dies bedeutet, dass zur Erfüllung von regulatorischen Anforderungen das Datensignal lizenzierte Frequenzen wie die Amateurfunkbänder ausschließen muss.

Trotz dieser Schwierigkeiten ist es praktisch möglich, mindestens 100 Mbit/s über die normalen Stromleitungen im Haus zu schicken, indem Kommunikationsmuster benutzt werden, die gestörte Frequenzen und eine Häufung von Fehlern abfangen. Viele Produkte benutzen unterschiedliche proprietäre Standards für Trägerfrequenzanlagen, deshalb wird intensiv an der Entwicklung internationalen Standards gearbeitet.

2.2.5 Glasfaserleiter

Viele Leute in der Computerindustrie sind sehr stolz darauf, wie schnell sich die Technologien laut dem Gesetz von Moore weiterentwickeln. Dieses Gesetz besagt, dass sich die Anzahl an Transistoren pro Chip rund alle zwei Jahre verdoppelt (Schaller, 1997). Der ursprüngliche IBM-PC (1981) lief mit einer Taktfrequenz von 4,77 MHz. Achtundzwanzig Jahre später konnte ein PC mit Vierkernprozessor mit über 3 GHz laufen. Diese Steigerung ist eine Verbesserung ungefähr um das 2 500-Fache – bzw. um das 16-Fache pro Jahrzehnt. Beeindruckend.

In der gleichen Zeit entwickelten sich die überregionalen Kommunikationsverbindungen von 45 Mbit/s (eine T3-Leitung im Telefonsystem) auf 100 Gbit/s (eine moderne Fernleitung). Dieser Zuwachs ist ähnlich beeindruckend, um mehr als den Faktor 2 000 und fast um 16 pro Jahrzehnt – während gleichzeitig die Fehlerrate von 10^{-5} pro Bit auf fast null sank. Außerdem nähern sich Einzelkernprozessoren allmählich physikalischen Grenzen, weshalb heute die Anzahl der Prozessoren auf einem Chip erhöht wird. Demgegenüber beträgt die erreichbare Bandbreite bei der Glasfasertechnik mehr als 50 000 Gbit/s (50 Tbit/s) und wir sind weit davon entfernt, diese Grenzen zu erreichen. Die heutige Beschränkung in der Praxis von rund 100 Gbit/s ist unserem Unvermögen zuzuschreiben, elektrische und optische Signale schneller ineinander zu konvertieren. Um Verbindungen mit höherer Kapazität aufzubauen, werden viele Kanäle einfach parallel über eine einzige Glasfaser geführt.

In diesem Abschnitt untersuchen wir Glasfaserleiter, um die Funktionsweise dieser Übertragungstechnik kennenzulernen. Im andauernden Wettlauf zwischen Rechengeschwindigkeit und Kommunikation könnte die Kommunikation aufgrund der Glasfasernetze schon gewonnen haben. Die Konsequenzen wären hauptsächlich unbegrenzte Bandbreite und die neue Allerweltsweisheit, dass Computer hoffnungslos langsam sind, sodass Netze versuchen sollten, Berechnungen um jeden Preis zu vermeiden. Diese Veränderung wird noch eine Weile brauchen, bis sie zu einer Generation von Informatikern und Ingenieuren durchgedrungen ist, die noch in Begriffen von niedrigen Shannon-Grenzen denken, welche von Kupferkabeln auferlegt wurden.

Natürlich erzählt dieses Szenario nicht die ganze Geschichte, weil es nichts über die Kosten sagt. Die Kosten, Glasfaser auf der letzten Meile zu installieren, um Konsumenten zu erreichen und die niedrige Bandbreite von Kabeln und begrenzte Verfügbarkeit des Frequenzspektrums zu umgehen, sind enorm. Außerdem kostet es mehr Energie, Bits zu bewegen als zu rechnen. Wir haben möglicherweise immer Inseln der Ungerechtigkeit, wo entweder Berechnung oder Kommunikation im Wesentlichen kostenlos ist. Beispielsweise reagieren wir in den Randbereichen des Internets mit Berechnung und Speicherung auf das Problem der Komprimierung und Zwischen speicherung von Inhalten. Innerhalb des Internets passiert unter Umständen das Gegenteil, nämlich wenn Unternehmen wie Google riesige Datenmengen über das Netz dorthin verschieben, wo die Speicherung oder Berechnung billiger ist.

Glasfaserleiter werden für Langstreckenübertragungen im Backbone des Netzes, in Hochgeschwindigkeits-LANs (auch wenn bisher Kupfer letzten Endes immer noch aufholen konnte) und Hochgeschwindigkeitsinternetzugang wie [Fiber to the Home](#)

(FttH) eingesetzt. Ein optisches Übertragungssystem hat drei Komponenten: die Lichtquelle, das Übertragungsmedium und den Detektor. Konventionell wird ein Lichtimpuls als ein 1-Bit und die Abwesenheit eines Lichtimpulses als ein 0-Bit interpretiert. Das Übertragungsmedium ist eine hauchdünne Glasfaser. Der Detektor erzeugt einen elektrischen Impuls, wenn Licht auf ihn fällt. Durch das Anschließen einer Lichtquelle an einem Ende eines Glasfaserleiters und eines Detektors am anderen Ende erhalten wir ein unidirektionales Datenübertragungssystem, das ein elektrisches Signal annimmt, in Lichtimpulse konvertiert und diese überträgt und die Ausgabe dann beim Empfänger wieder in ein elektrisches Signal umwandelt.

Dieses Übertragungssystem würde Licht verlieren und wäre in der Praxis nutzlos, gäbe es da nicht ein interessantes physikalisches Prinzip: Wenn ein Lichtstrahl von einem Medium in ein anderes übertritt – z.B. von Quarzglas in die Luft –, so wird er an der Grenzfläche zwischen Quarzglas und Luft gebrochen (►Abbildung 2.6a). Hier sehen wir einen Lichtstrahl, der mit dem Winkel α_1 auf die Grenzfläche trifft und sie im Winkel β_1 verlässt. Die Brechungsstärke hängt von den Eigenschaften der beiden Medien ab (genauer gesagt, von den Brechungsindizes). Liegt der Einfallswinkel oberhalb eines bestimmten kritischen Werts, dann wird das gesamte Licht vollständig in das Quarzglas zurückgeworfen und nichts entkommt in die Luft. Somit ist ein Lichteinfall im oder oberhalb des kritischen Winkels in der Faser gefangen (siehe ►Abbildung 2.6b) und kann sich praktisch ohne Verlust über viele Kilometer ausbreiten.

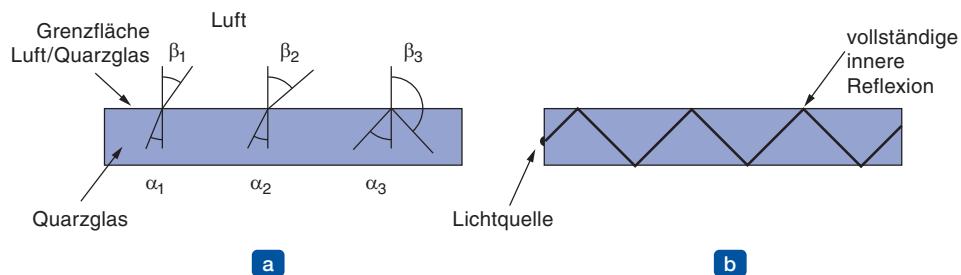


Abbildung 2.6: (a) Drei Beispiele eines Lichtstrahls in einer Glasfaser, der in verschiedenen Winkeln auf die Luft-Quarzglas-Grenzfläche trifft. (b) Durch vollständige interne Reflexion eingeschlossener Lichtstrahl.

Die Skizze in Abbildung 2.6b zeigt nur einen eingeschlossenen Strahl. Da aber jeder Lichtstrahl oberhalb des kritischen Winkels nach innen reflektiert wird, prallen in der Glasfaser viele Lichtstrahlen in verschiedenen Winkeln auf die Grenzfläche. Jeder Strahl hat einen anderen sogenannten Modus, sodass eine Glasfaser mit dieser Eigenschaft **Multimode-Faser** genannt wird.

Wird jedoch der Durchmesser der Glasfaser auf wenige Wellenlängen reduziert, dann verhält sich die Faser wie ein Wellenleiter. Das Licht breitet sich ohne Aufprall an der Grenzfläche entlang einer geraden Linie aus. Diese Fasern werden **Singlemode-Fasern** genannt. Singlemode-Fasern sind teurer, werden aber häufig für Langstrecken verwendet. Derzeit erhältliche Singlemode-Fasern können Daten über 100 km mit 100 Gbit/s ohne Verstärkung übertragen. Bei kürzeren Entferungen wurden im Labor sogar noch höhere Datenübertragungsraten erzielt.

Übertragung von Licht über Glasfaser

Lichtwellenleiter werden aus Glas hergestellt. Für Glas ist der Rohstoff wiederum Sand, ein kostengünstiges Rohmaterial, das in nahezu unbegrenzten Mengen vorhanden ist. Die Herstellung von Glas war schon im alten Ägypten bekannt. Damals durfte Glas aber nicht stärker als 1 mm sein, sonst wäre es lichtundurchlässig gewesen. Glas, das ausreichend durchsichtig war, um für Fenster benutzt zu werden, wurde erstmals in der Renaissance entwickelt. Das für moderne optische Fasern verwendete Glas ist so transparent, dass man hier, wenn die Meere statt mit Wasser mit diesem Glas gefüllt wären, von der Oberfläche oder einem Flugzeug aus bis auf den Grund sehen könnte.

Die Dämpfung des Lichts durch Glas hängt von der Wellenlänge des Lichts ab (wie auch von einigen physikalischen Eigenschaften des Glases). Für die in Glasfasern verwendete Glasart wird in ▶ Abbildung 2.7 die Dämpfung in Einheiten von Dezibel pro linearem Kilometer Glasfaser dargestellt. Ein Verlustfaktor der Signalstärke von zwei ergibt beispielsweise eine Dämpfung von $10\log_{10} 2 = 3$ dB. Die Abbildung zeigt den nahen infraroten Teil der Frequenzbereiche, der in der Praxis verwendet wird. Sichtbares Licht hat geringfügig kürzere Wellenlängen von 0,4 bis 0,7 µm (1 µm entspricht 10^{-6} m). Ein echter metrischer Purist würde diese Wellenlänge mit 400 nm bis 700 nm angeben, aber wir bleiben hier bei der traditionellen Bezeichnung.

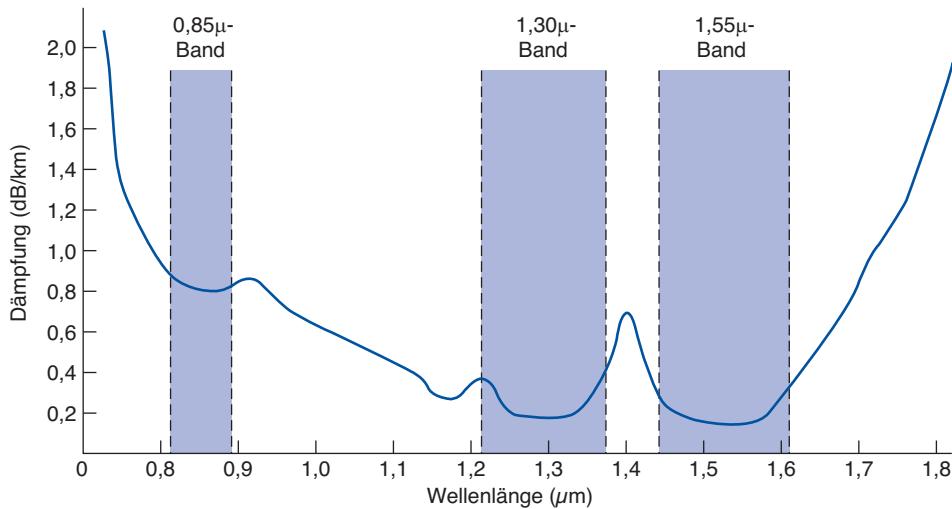


Abbildung 2.7: Dämpfung von Licht durch Glasfaser im Infrarotbereich.

Im optischen Übertragungsbereich werden zurzeit drei Wellenlängenbänder am häufigsten genutzt. Ihre jeweilige Mitte liegt bei 0,85, 1,30 bzw. 1,55 µm. Alle drei Bänder sind 25 000 bis 30 000 GHz breit. Das 0,85-µm-Band wurde zuerst eingesetzt. Es hat eine höhere Dämpfung und wird daher für kürzere Entferungen verwendet, aber bei dieser Wellenlänge können Laser und Elektronik aus dem gleichen Material (Galliumarsenid) hergestellt werden. Die letzten beiden Bänder haben gute Dämpfungseigenschaften (weniger als 5 % Verlust pro Kilometer). Das 1,55-µm-Band wird heute verbreitet mit Erbium-dotierten Verstärkern eingesetzt, die direkt im optischen Bereich arbeiten.

Lichtimpulse, die durch eine Glasfaser gesendet werden, dehnen sich bei der Übertragung in der Länge aus. Diese Ausbreitung nennt man **chromatische Dispersion**. Der Umfang hängt von der Wellenlänge ab. Eine Möglichkeit, diese ausgedehnten Impulse daran zu hindern sich zu überlappen, ist die Erhöhung der Entfernung zwischen ihnen. Das ist aber nur durch Reduzierung der Signalrate möglich. Zum Glück wurde inzwischen entdeckt, dass fast alle Dispersionseffekte verschwinden, wenn die Impulse eine spezielle Form aufweisen, die etwas mit dem Kehrwert des Kosinus Hyperbolicus zu tun hat. So ist es möglich, Impulse Tausende von Kilometern zu senden, ohne dass eine nennenswerte Formverzerrung entsteht. Diese Impulse nennt man **Solitons**. Derzeit sind beträchtliche Forschungsarbeiten im Gange, um Solitonimpulse vom Labor in die Praxis zu überführen.

Glasfaserkabel

Glasfaserkabel sind mit Koaxialkabeln vergleichbar, haben aber keinen geflochtenen Schutzmantel. ►Abbildung 2.8a zeigt die Seitenansicht einer einzelnen Glasfaser. In der Mitte befindet sich der Glaskern, durch den sich das Licht ausbreitet. Bei Multi-mode-Fasern hat der Kern einen Durchmesser von 50 µm. Das ist etwa die Stärke eines menschlichen Haars. Singlemode-Fasern haben einen Kern von 8 bis 10 µm.

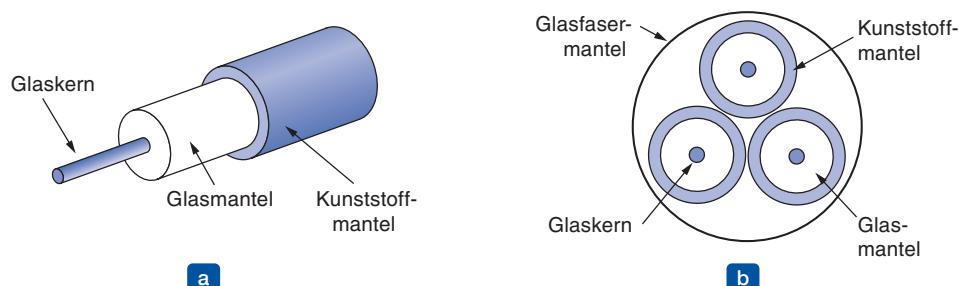


Abbildung 2.8: (a) Seitenansicht einer Glasfaser. (b) Ende eines Kabelmantels mit drei Glasfasern.

Der Glaskern ist mit einem Glasmantel verkleidet, der einen niedrigeren Brechungsindex hat als der Glaskern, um das gesamte Licht im Kern zu halten. Darüber liegt ein dünner Kunststoffmantel zum Schutz des Glasmantels. Glasfaserleiter werden in der Regel in Bündeln gruppiert, die von einem schützenden Außenmantel umhüllt sind. ►Abbildung 2.8b zeigt einen Kabelmantel mit drei Glasfaserleitern.

Terrestrische Fasermäntel werden normalerweise unterirdisch – etwa einen Meter unter der Oberfläche – verlegt, wo sie gelegentlich den Angriffen von Iltissen oder Mardern ausgesetzt sind. In Küstennähe werden Überseeleitungen aus Glasfaser mit einer Art Meerespflug in Gräben verlegt. Im tieferen Gewässer liegen sie nur auf dem Meerest Grund, wo sie Opfer von Schleppnetzfischern oder Riesentintenfischen werden können.

Glasfaserleiter können auf drei Arten verbunden werden. Sie können erstens am Ende in Steckern münden und in Glasfaserbuchsen eingesteckt werden. Die Stecker verlieren etwa 10 bis 20 % Licht, vereinfachen aber die Rekonfiguration des Systems.

Zweitens können sie mechanisch verbunden (verspleißt) werden. Beim mechanischen Spleißen werden einfach zwei sauber abgeschnittene Enden nebeneinander in einen speziellen Mantel gelegt und festgeklemmt. Die Ausrichtung kann verbessert werden, indem Licht durch die Verbindungsstelle geleitet wird und dann kleine Korrekturen vorgenommen werden, um das Signal zu optimieren. Mechanische Spleiße können von einem geschulten Fachmann in etwa fünf Minuten eingerichtet werden und führen zu einem Lichtverlust von 10 %.

Drittens können zwei Faserstücke verschmolzen werden, um eine feste Verbindung zu bilden. Solch ein Fusionsspleiß ist fast so gut wie eine einzeln gezogene Faser, aber auch hier tritt eine geringe Dämpfung auf.

Bei allen drei Spleißarten können an der Spleißstelle Reflexionen auftreten und die reflektierte Energie kann das Signal stören.

Für die Signale werden in der Regel zwei Arten von Lichtquellen verwendet: LEDs (lichtemittierende Dioden) und Halbleiterlaser. Wie in ►Abbildung 2.9 gezeigt, besitzen sie verschiedene Eigenschaften. Indem man Interferometer (Fabry-Perot oder Mach-Zehnder) zwischen Quelle und Faser einfügt, kann man ihre Wellenlänge einstellen. Bei Fabry-Perot-Interferometern handelt es sich um einfache Hohlraumresonatoren, die aus zwei parallelen Spiegeln bestehen. Das Licht fällt senkrecht auf die Spiegel ein. Die Länge des Hohlraums wählt jene Wellenlängen aus, die in ein ganzzahliges Vielfaches hineinpassen. Mach-Zehnder-Interferometer teilen das Licht in zwei Lichtbündel. Die zwei Lichtbündel durchlaufen leicht unterschiedliche Strecken. Sie werden am Ende wieder gebündelt und sind nur für bestimmte Wellenlängen phasengleich.

| Eigenschaft | LED | Halbleiterlaser |
|---------------------------|-----------|---------------------------|
| Datenübertragungsrate | Niedrig | Hoch |
| Glasfasertyp | Multimode | Multimode oder Singlemode |
| Entfernung | Kurz | Lang |
| Lebensdauer | Langlebig | Kurzlebig |
| Temperaturempfindlichkeit | Gering | Beträchtlich |
| Kosten | Niedrig | Hoch |

Abbildung 2.9: Vergleich von Halbleiterdioden und LEDs als Lichtquellen.

Das Empfangsstück einer optischen Faser besteht aus einer Photodiode, die einen elektrischen Impuls abgibt, wenn Licht auf sie fällt. Die Reaktionszeit einer Photodiode, welche das Signal vom optischen in den elektrischen Bereich überführt, begrenzt die Datenübertragungsraten auf etwa 100 Gbit/s. Darüber hinaus ist hier thermisches Rauschen ein Problem, sodass ein Lichtimpuls ausreichend Energie führen muss, um erkannt zu werden. Die Fehlerrate kann beliebig weit reduziert werden, indem man sicherstellt, dass die Impulse stark genug sind.

Vergleich: Glasfaser und Kupferdraht

Ein Vergleich von Glasfaser und Kupfer lohnt sich. Glasfaser hat viele Vorteile. Sie kann zum einen viel höhere Bandbreiten unterstützen als Kupfer. Das allein würde schon genügen, um sie als bevorzugtes Medium in leistungsstarken Netzen zu verwenden. Aufgrund der geringen Dämpfung ist auf Langstrecken nur ungefähr alle 50 km ein Repeater erforderlich, während bei Kupferdraht etwa alle 5 km ein Repeater installiert werden muss. Das bedeutet große Kosteneinsparungen. Glasfaser hat den weiteren Vorteil, dass sie von Stromstößen, elektromagnetischen Störungen und Stromausfällen nicht beeinflusst wird. Auch korrodierende Schadstoffe in der Luft können ihr nichts anhaben, was wichtig beim Einsatz in rauen Fabrikumgebungen ist.

Seltsamerweise bevorzugen Telefongesellschaften die Glasfaser aus einem völlig anderen Grund: Sie ist dünn und wiegt wenig. Viele vorhandene Kabelkanäle sind voll, deshalb ist für Kapazitätserweiterungen kein Platz vorhanden. Die Entfernung sämtlicher Kupferkabel und deren Ablösung durch Glasfaser schafft in den Kabelkanälen wieder Platz und Kupfer hat einen ausgezeichneten Wiederverkaufswert für Kupferaffinerien, die es als sehr hochwertiges Erz betrachten. Glasfaser ist darüber hinaus sehr viel leichter als Kupfer. Eintausend Twisted-Pair-Kabel mit einer Länge von 1 km wiegen 8 000 kg. Zwei Glasfaserkabel in der gleichen Länge haben eine höhere Kapazität und wiegen nur 100 kg. Das bedeutet auch geringere Kosten für mechanische Stützsysteme. Bei der Verlegung neuer Strecken gewinnt die Glasfaser um Längen aufgrund der wesentlich niedrigeren Installationskosten. Glasfasern verlieren auch kein Licht und lassen sich weniger leicht anzapfen. Dadurch bieten sie guten Schutz gegen unbefugtes Abhören.

Ein Nachteil ist, dass Glasfaser eine sehr viel weniger bekannte Technologie ist, die Kenntnisse erfordert, über die nicht alle Techniker verfügen. Darüber hinaus kann Glasfaser sehr leicht beschädigt werden, wenn sie zu stark gebogen wird. Da die optische Übertragung von Haus aus unidirektional ist, setzt eine bidirektionale Kommunikation entweder zwei Glasfaserkabel oder zwei Frequenzbänder für ein Glasfaserkabel voraus. Außerdem kosten Glasfaserschnittstellen mehr als elektrische. Dennoch liegt die Zukunft aller festen Datenkommunikationseinrichtungen für mehr als kurze Entferungen eindeutig bei der Glasfaser. Eine ausführliche Diskussion aller Aspekte der Glasfaser und Glasfasernetze finden Sie in Hecht (2005).

2.3 Drahtlose Übertragung

Unser Zeitalter hat eine ganz besondere Spezies von Informations-Junkies mit sich gebracht: Leute, die den ganzen Tag und oft auch nachts online sein müssen. Für diese mobilen Benutzer sind Twisted-Pair-, Koaxial- und Glasfaserkabel nutzlos. Sie brauchen ihren „Datenschuss“ für den Laptop, das Notebook, den Pocket-PC, Palmtop oder Armbanduhr-PC, ohne an die terrestrische Kommunikationsinfrastruktur angebunden zu sein. Für diese Benutzer ist die drahtlose Kommunikation die einzige Lösung.

In den folgenden Abschnitten wird drahtlose Kommunikation im Allgemeinen behandelt. Es gibt noch viele andere wichtige Anwendungsbereiche dafür als lediglich am

Strand liegend im Web surfen zu können. Drahtlosigkeit hat aber auch für fest aufgestellte Geräte unter gewissen Umständen einen Vorteil. Ist beispielsweise die Verlegung einer Glasfaserleitung zu einem Gebäude aufgrund geologischer Gegebenheiten (Berge, Dschungel, Sümpfe usw.) schwierig, bietet sich die drahtlose Verbindung an. Interessant ist, dass die drahtlose digitale Kommunikation auf den Inseln von Hawaii anfing, wo die Benutzer durch den Pazifik von ihren Datenzentren getrennt sind und das Telefonystem ungenügend war.

2.3.1 Das elektromagnetische Spektrum

Wenn sich Elektronen bewegen, erzeugen sie elektromagnetische Wellen, die sich im Raum (auch im Vakuum) ausbreiten können. Diese Wellen wurden 1865 von dem britischen Physiker James Clerk Maxwell vorhergesagt und das erste Mal 1887 von dem deutschen Physiker Heinrich Hertz beobachtet. Die Anzahl der Schwingungen einer Welle pro Sekunde wird als ihre **Frequenz** f bezeichnet und in **Hz** (zu Ehren von Heinrich Hertz) gemessen. Die Entfernung zwischen zwei aufeinanderfolgenden Maxima (oder Minima) wird als **Wellenlänge** bezeichnet. Sie wird mit dem griechischen Buchstaben Lambda (λ) angegeben.

Schließt man eine Antenne der richtigen Größe an einen Stromkreis an, können elektromagnetische Wellen effizient ausgestrahlt und vom Empfänger in einiger Entfernung empfangen werden. Alle drahtlosen Kommunikationsarten basieren auf diesem Prinzip.

Im Vakuum bewegen sich alle elektromagnetischen Wellen ungeachtet ihrer Frequenz mit derselben Geschwindigkeit. Diese wird als **Lichtgeschwindigkeit** c bezeichnet, die ungefähr 3×10^8 m/s bzw. etwa 30 cm/ns beträgt. Bei Kupferdraht oder Glasfaser verlangsamt sich die Geschwindigkeit auf etwa 2/3 dieses Wertes und ist in geringem Umfang von der Frequenz abhängig. Lichtgeschwindigkeit ist die ultimative Geschwindigkeitsbegrenzung. Kein Objekt oder Signal kann jemals schneller sein.

Die grundlegende Beziehung zwischen f , λ und c (im Vakuum) lautet:

$$\lambda f = c \quad (2.4)$$

Da c eine Konstante ist, erhalten wir λ , wenn wir f kennen, und umgekehrt. Als Faustregel gilt: Wenn λ in Meter angegeben wird und f in MHz, dann ist $\lambda f \approx 300$. Ein Beispiel: 100-MHz-Wellen sind z.B. etwa 3 m lang, 1 000-MHz-Wellen sind z.B. etwa 0,3 m und 0,1 m lange Wellen haben eine Frequenz von 3 000 MHz.

Das elektromagnetische Spektrum ist in ► Abbildung 2.10 dargestellt. Funkwellen, Mikrowellen, Infrarotwellen und sichtbares Licht sind die Teile der Frequenzbereiche, die zur Übertragung von Informationen genutzt werden können, indem die Amplitude, Frequenz oder Phase der Wellen moduliert wird. Ultraviolettes Licht, Röntgen- und Gammastrahlen wären aufgrund ihrer höheren Frequenzen noch besser, sind aber schwer zu erzeugen und zu modulieren, breiten sich nicht so gut in Gebäuden aus und sind für alle lebenden Geschöpfe gefährlich. Die in Abbildung 2.10 unten aufgeföhrten Bänder sind die offiziellen ITU-Bezeichnungen. Sie basieren auf den Wellenlängen, was

bedeutet, dass das LF-Band von 1 bis 10 km (ungefähr 30 bis 300 kHz) reicht. Die Bezeichnungen LF, MF und HF bezeichnen niedrige, mittlere und hohe Frequenz. Als die Bezeichnungen festgelegt wurden, hat niemand geahnt, dass man einmal über 10 MHz hinauskommen würde. Deshalb wurden die höheren Bänder später mit „Very“, „Ultra“, „Super“, „Extremely“ und „Tremendously High Frequency“ benannt. Darüber hinaus gibt es keine Bezeichnungen, aber „Incredibly“, „Astonishingly“ und „Prodigiously High Frequency“ – abgekürzt IHF, AHF und PHF – klingen doch ganz hübsch.

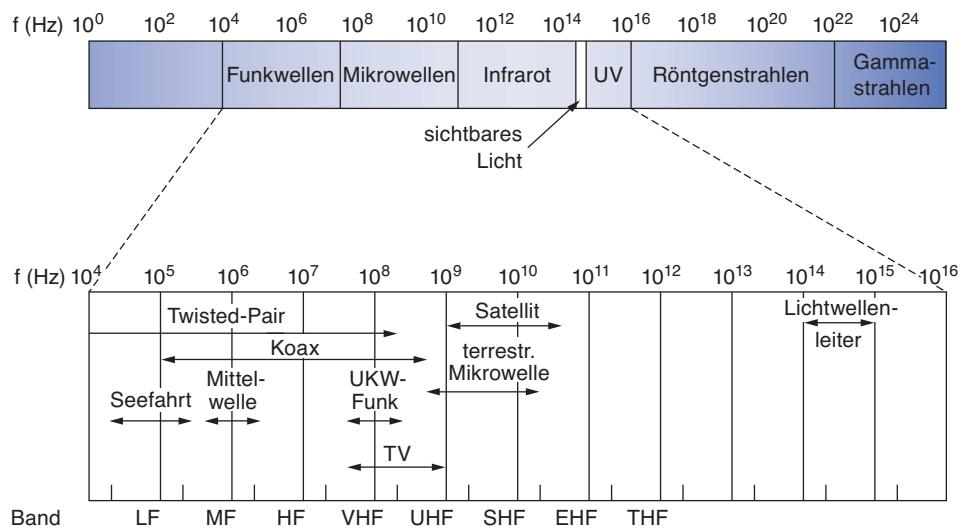


Abbildung 2.10: Das elektromagnetische Spektrum und seine Verwendung in der Telekommunikation.

Wir wissen aus der Gleichung von Shannon (Gleichung (2.3)), dass die Informationsmenge, die ein Signal wie eine elektromagnetische Welle führen kann, von der empfangenen Stärke abhängt und proportional zur Bandbreite ist. Aus Abbildung 2.10 sollte nun auch ersichtlich sein, warum Netzspezialisten die Glasfaser so sehr schätzen. Im Mikrowellenband könnten viele GHz der Bandbreite für die Datenübertragung abgegriffen werden, im Glasfaserband sogar noch mehr, da es weiter rechts auf unserer logarithmischen Skala steht. Sehen Sie sich beispielsweise das 1,30-µm-Band in Abbildung 2.7 an, welches eine Breite von 0,17 µm hat. Benutzen wir Gleichung (2.4), um die Anfangs- und Endfrequenzen von den Anfangs- und Endwellenlängen zu suchen, so stellen wir fest, dass der Frequenzbereich über 30 000 GHz liegt. Bei einem realistischen Signal-Rausch-Verhältnis von 10 dB sind dies 300 Tbit/s.

Die meisten Übertragungen nutzen ein relativ schmales Frequenzband (d.h. $\Delta f / f \ll 1$). Sie konzentrieren ihre Signale in diesem schmalen Band, um das Spektrum effizient auszunutzen und um annehmbare Datenraten zu erhalten, wenn mit ausreichender Leistung übertragen wird. In einigen Fällen wird aber ein breiteres Band verwendet, und zwar in drei Variationen. Beim **Frequency Hopping Spread Spectrum (FHSS)**, Frequenzsprungtechnik mit Spektrumsspreizung) wechselt der Sender mehrere Hundert Mal pro Sekunde von Frequenz zu Frequenz. Das ist besonders im Militärbereich

beliebt, weil Übertragungen dadurch schwer aufzudecken sind und es fast unmöglich ist, sie zu stören. Es bietet auch einen guten Schutz gegen den Mehrwegeempfang und Niederfrequenzbandstörungen, da der Empfänger nie so lange auf einer gestörten Frequenz bleiben wird, dass es zum Abbruch der Kommunikation kommt. Diese Robustheit macht den Einsatz für überbelegte Teile des Spektrums sinnvoll, wie die ISM-Bänder, die wir in Kürze beschreiben. Diese Technik wird kommerziell genutzt, beispielsweise bei Bluetooth oder älteren Versionen von IEEE 802.11.

Eine kleine Anekdote am Rande: Diese Technik wurde von der in Österreich geborenen Sexgöttin Hedy Lamarr miterfunden, die als erste Frau nackt in einem Film auftrat (1933 im tschechischen Film *Extase*). Ihr erster Ehemann war ein Waffenfabrikant, von dem sie lernte, wie einfach die Funksignale blockiert werden konnten, mit denen damals Torpedos gesteuert wurden. Als sie entdeckte, dass ihr Mann Waffen an Hitler verkaufte, war sie entsetzt. Sie verkleidete sich als Dienstmädchen und floh nach Hollywood, um ihre Karriere als Schauspielerin fortzusetzen. In ihrer Freizeit entwickelte sie das Frequenzsprungverfahren, um die Alliierten beim Kampf gegen Hitler zu unterstützen. In ihrem Schema wurden 88 Frequenzen verwendet, was der Anzahl der Tasten (und Frequenzen) eines Klaviers entspricht. Für ihre Erfindung erhielten sie und ihr Freund, der Komponist George Antheil, das US-Patent 2.292.387. Es gelang ihnen aber nicht, die US-amerikanische Marine von dem praktischen Nutzen ihrer Erfindung zu überzeugen, und sie erhielten somit nie Lizenzennahmen dafür. Nur ein paar Jahre nach Ablauf des Patents wurde das Verfahren populär.

Eine zweite Form des Streuspektrums, die **Spreizbandtechnik (DSSS)**, *Direct Sequence Spread Spectrum*, benutzt eine Codefolge, um das Datensignal über ein breiteres Frequenzband zu verteilen. Dieses wird im kommerziellen Bereich häufig als effektive Möglichkeit eingesetzt, damit mehrere Signale ein Frequenzband gemeinsam nutzen können. Diesen Signalen können unterschiedliche Codes zugeordnet werden, eine Methode, die **Codemultiplexverfahren (CDMA)**, *Code Division Multiple Access* heißt und die wir später in diesem Kapitel noch besprechen werden. ► Abbildung 2.11 stellt diese Methode dem Frequenzsprungverfahren gegenüber. DSSS bildet die Grundlage des 3G-Mobilfunknetzes und wird außerdem für GPS (*Global Positioning System*) eingesetzt. Selbst ohne verschiedene Codes kann ein DSSS-Verfahren ebenso wie FHHS Niederfrequenzbandstörungen und Mehrwegeempfang tolerieren, da nur ein Bruchteil des gewünschten Signals verloren ist. Es wird in dieser Rolle in älteren drahtlosen IEEE-802.11b-LANs verwendet. Eine faszinierende und ausführliche Geschichte der Streuspektrumskommunikation findet der Leser in Scholtz (1982).

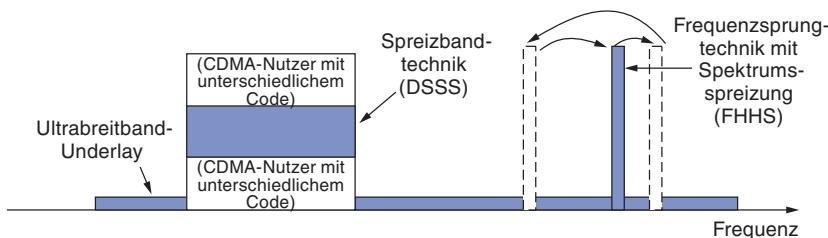


Abbildung 2.11: Streuspektrum- und Ultrabreitbandkommunikation.

Eine dritte Kommunikationsmethode mit einem breiteren Band ist die **Ultrabreitbandtechnik (UWB, Ultra-WideBand)**. UWB sendet zur Informationsübertragung eine Reihe von schnellen Impulsen, die ihre Positionen variieren. Die schnellen Übergänge führen zu einem Signal, das dünn über eine sehr breite Frequenz verteilt ist. Bei UWB haben die Signale eine Bandbreite von mindestens 500 MHz oder mindestens 20 % der Mittenfrequenz ihres Frequenzbandes. UWB ist ebenfalls in Abbildung 2.11 dargestellt. Aufgrund dieser großen Bandbreite kann UWB mit hohen Datenraten übertragen. Durch die Verteilung über ein breites Frequenzband kann eine beträchtliche Menge von verhältnismäßig starken Interferenzen von andern Schmalbandsignalen toleriert werden. Da UWB bei Kurzstreckenübertragungen auf jeder Frequenz sehr wenig Energie hat, ist es ebenso wichtig, dass UWB keine nachteiligen Interferenzen mit den anderen schmalbandigen Funksignalen verursacht – man spricht hier von **Underlay**-Technik. Diese friedliche Koexistenz führte zu Anwendungen in drahtlosen PANs mit bis zu 1 Gbit/s, obwohl der kommerzielle Erfolg eher durchwachsen war. Es kann auch zum Durchleuchten von festen Objekten (Boden, Mauern und Körper) oder als Teil eines präzisen Lokalisierungssystems eingesetzt werden.

Wir werden uns jetzt ansehen, wie die verschiedenen Teile des elektromagnetischen Spektrums von Abbildung 2.11 benutzt werden, dabei beginnen wir mit Funkwellen. Wenn nichts anderes gesagt wird, setzen wir voraus, dass alle Übertragungen ein schmales Frequenzband nutzen.

2.3.2 Funkübertragung

Funkwellen (Radiowellen) sind leicht zu erzeugen, legen große Entfernungen zurück und dringen mühelos in Gebäude ein. Deshalb werden sie häufig für die Kommunikation in Räumen und im Freien benutzt. Funkwellen sind außerdem rundstrahlend (omnidirektional), das heißt, sie strahlen von der Quelle in alle Richtungen aus, sodass Sender und Empfänger nicht sorgfältig physikalisch ausgerichtet werden müssen.

Rundstrahlende Funkwellen sind oft nützlich – aber nicht immer. In den 1970er Jahren entschloss sich General Motors, alle neuen Cadillac-Modelle mit computergesteuerten Antiblockierbremsen auszustatten. Trat der Fahrer auf das Bremspedal, schaltete der Computer die Bremsen in kurzen Abständen ein und aus, um hartes Bremsen zu vermeiden. Eines schönen Tages setzte ein Streifenpolizist auf dem Highway in Ohio sein neues mobiles Funkgerät ein, um das Revier anzurufen, und plötzlich verhielt sich der Cadillac neben ihm wie ein buckelndes Wildpferd. Nachdem der Polizist den Fahrer an den Seitenstreifen geordert hatte, erklärte dieser ihm aufgereggt, er könne nichts dazu, der Wagen sei einfach verrückt geworden.

Schließlich zeichnete sich ein Muster ab: Die Cadillacs spielten manchmal verrückt, aber nur auf den Ohio-Highways und nur, wenn Streifenwagen in der Nähe waren. Über lange Zeit hinweg konnte man bei General Motors einfach nicht verstehen, warum dies weder in anderen Bundesstaaten noch auf kleineren Straßen in Ohio auftrat. Nach umfassenden Untersuchungen stellte sich heraus, dass die Verkabelung des Antiblockiersystems im neuen Cadillac eine fantastische Antenne für die Frequenz der Ohio-Highway-Streifenwagen bildete, die mit einem neuen Funksystem ausgestattet waren.

Die Eigenschaften von Funkwellen sind von der Frequenz abhängig. In niedrigen Frequenzen können Funkwellen leicht durch Hindernisse hindurchdringen, jedoch fällt die Leistung mit zunehmender Entfernung von der Quelle stark ab – mindestens so schnell wie $1/r^2$ in der Luft – da sich die Signalenergie dünner über eine größere Oberfläche verteilt. Diese Abschwächung heißt **Pfadverlust** (*path loss*). Bei hohen Frequenzen verlaufen Funkwellen eher in geraden Linien und prallen an Hindernissen ab. Pfadverlust verringert die Energie noch, obwohl das empfangene Signal ebenfalls stark von Reflexionen abhängt. Außerdem werden Hochfrequenzfunkwellen in größerem Ausmaß von Regen und anderen Hindernissen absorbiert als Wellen der niedrigeren Frequenzen. Auf allen Frequenzen sind Funkwellen Störungen von Motoren und Elektroanlagen ausgesetzt.

Es ist interessant, die Dämpfung von Funkwellen mit denen von Signalen in gerichteten Medien zu vergleichen. Bei Glasfaser-, Koaxial- und Twisted-Pair-Kabeln fällt das Signal mit dem gleichen Bruchteil pro Entfernungseinheit, zum Beispiel 20 dB pro 100 m für Twisted-Pair-Kabel. Bei Funkwellen, fällt das Signal mit dem gleichen Bruchteil, mit dem sich die Entfernung verdoppelt, zum Beispiel 6 dB pro Verdopplung im freien Raum. Dieses Verhalten bedeutet, dass Funkwellen große Entfernung zurücklegen können, und Überlagerungen der verschiedenen Benutzer sind ein Problem. Aus diesem Grund wird die Nutzung von Funkwellensendern in allen Ländern sehr streng reguliert, mit einigen Ausnahmen, die später in diesem Kapitel noch besprochen werden.

In den VLF-, LF- und MF-Bändern folgen Funkwellen der Erdoberfläche (► Abbildung 2.12a). Diese Wellen können bei niedrigen Frequenzen über etwa 1 000 km, bei höheren über kürzere Entfernen erkannt werden. Das Mittelwellenradio läuft über das MF-Band. Aus diesem Grund kann man z.B. die Hamburger Mittelwellen-Radiostationen in München nicht gut empfangen. Radiowellen in diesen Bändern durchdringen mühelos Gebäude; deshalb funktionieren tragbare Radios in Innenräumen. Das Hauptproblem bei der Verwendung dieser Bänder für die Datenkommunikation ist die geringe Bandbreite (siehe Gleichung (2.4)).

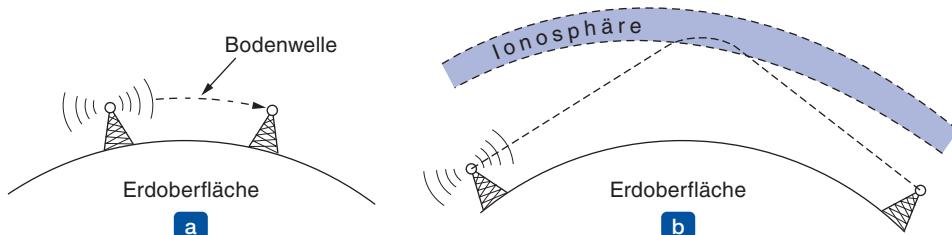


Abbildung 2.12: In den VLF-, LF- und MF-Bändern folgen Funkwellen der Krümmung der Erdoberfläche. Beim HF-Band prallen sie an der Ionosphäre ab.

Bei den HF- und VHF-Bändern werden die Bodenwellen leichter von der Erde absorbiert. Die Wellen, die die Ionosphäre, eine mit Partikeln aufgeladene Schicht, die in einer Höhe von 100 bis 500 km die Erde umschließt, erreichen, werden hier gebrochen und auf die Erde zurückgeschickt (siehe ► Abbildung 2.12b). Unter bestimmten atmo-

sphärischen Bedingungen schnellen die Signale eventuell mehrmals hin und her. Amateurfunker benutzen diese Bänder für Fernübertragungen. Auch das Militär benutzt die HF- und VHF-Bänder.

2.3.3 Mikrowellenübertragung

Über 100 MHz verlaufen die Wellen in nahezu geraden Linien und können daher eng gebündelt werden. Die Konzentration der gesamten Energie in einen kleinen Strahl mithilfe einer Parabolantenne (die üblichen Fernsehschüsseln auf den Hausdächern) ergeben ein viel höheres Signal-Rausch-Verhältnis. Die Sende- und Empfangsantennen müssen aber genau aufeinander ausgerichtet werden. Diese Ausrichtung ermöglicht außerdem, dass mehrere reihenweise angeordnete Sender mit mehreren ebenfalls reihenweise angeordneten Empfängern ohne Störeinflüsse kommunizieren können – unter der Voraussetzung, dass ein gewisser Mindestabstand eingehalten wird. Bevor die Glasfaser eingeführt wurde, bildeten diese Mikrowellen jahrzehntelang den Kern des Telefonnetzes für Ferngespräche. So hat beispielsweise MCI, einer der ersten Mitbewerber von AT&T, nach der Deregulierung sein gesamtes System auf Mikrowellenkommunikation aufgebaut, wobei die Übertragung zwischen den Masten über zig Kilometer ablief. Die verwendete Technologie kam selbst im Firmennamen zum Ausdruck (MCI stand für Microwave Communications, Inc.). MCI ist dann auf Glasfaser umgestiegen und wurde nach einer langen Reihe von Firmenzusammenschlüssen und Konkursen im Telekommunikationskarussell ein Teil von Verizon.

Mikrowellen verlaufen in einer geraden Linie, daher kommt ihnen manchmal die Erde in die Quere, wenn die Sendemasten zu weit auseinanderstehen (etwa bei der Verbindung zwischen Seattle und Amsterdam). Folglich sind in gewissen Abständen Repeater erforderlich. Je höher die Sendemasten sind, umso weiter können sie voneinander entfernt sein. Die Entfernung zwischen Repeatern steigt in etwa mit der Quadratwurzel der Höhe der Sendemasten. Bei einem 100 m hohen Sendemast können sich die Repeater 80 km weit entfernt befinden.

Im Gegensatz zu Funkwellen mit niedrigeren Frequenzen können Mikrowellen Gebäude nicht durchdringen. Darüber hinaus besteht, auch wenn der Strahl am Sender gut ausgerichtet ist, eine gewisse räumliche Abweichung. Einige Wellen werden eventuell durch niedrige atmosphärische Schichten gebrochen und kommen dadurch etwas später an als direkte Wellen. Die verzögerten Wellen kommen eventuell phasenverschoben bei den direkten Wellen an und löschen damit das Signal aus. Diesen Effekt nennt man **Mehrwegeempfang** (*multipath fading*), oft ein ernstes Problem. Es hängt vom Wetter und von der Frequenz ab. Einige Betreiber halten 10 % ihrer Kanäle ungenutzt als Reserve, um umschalten zu können, wenn ein Frequenzband vorübergehend durch Mehrwegeempfang unbrauchbar ist.

Die Forderungen nach immer mehr Frequenzbereichen zwingen die Betreiber zu noch höheren Frequenzen. Heute werden in der Praxis Bänder mit bis zu 10 GHz eingesetzt. Bei etwa 4 GHz setzt allerdings ein neues Problem ein: die Absorption durch Wasser. Diese Wellen sind nur ein paar Zentimeter lang und werden vom Regen absorbiert.

Dieser Effekt wäre prima, wenn man planen würde, einen riesigen Mikrowellherd im Garten aufzubauen, um vorbeifliegende Vögel zu grillen – im Kommunikationsbereich ist dies aber ein schwerwiegendes Problem. Wie beim Mehrwegeempfang ist die einzige Lösung auch hier die Abschaltung der Verbindungen, die dem Regen ausgesetzt sind, und deren Umlegung auf einen anderen Weg.

Zusammenfassend kann man sagen, dass die Mikrowellenkommunikation heute im Bereich der Telefonfernverbindungen, Mobiltelefone, der Ausstrahlung von Fernsehsendungen und anderer Anwendungen derart stark verbreitet ist, dass ein ernsthafter Mangel an Frequenzbereichen herrscht. Die Mikrowelle hat gegenüber der Glasfaser einige entscheidende Vorteile aufzuweisen. Der Hauptvorteil ist, dass keine Wege-rechte benötigt werden, um Kabel zu verlegen. Indem man alle 50 km ein kleines Grundstück erwirbt und darauf einen Mikrowellensendemast aufstellt, kann man das Telefonnetz völlig umgehen. Auf diese Weise hat es MCI geschafft, sich schnell erfolgreich als Anbieter von Telefonfernverbindungen auf dem Markt zu etablieren. (Sprint, ein anderer früher Konkurrent der deregulierten AT&T, ging einen völlig anderen Weg: Die Firma wurde von Southern Pacific Railroad gegründet, die bereits Eigentümerin eines umfangreichen Wegnutzungsrechtes war und einfach entlang ihrer Eisenbahnschienen Glasfaser verlegte.)

Mikrowellen sind auch verhältnismäßig preisgünstig. Die Aufstellung von zwei einfachen Sendemasten (was auch einfach nur hohe Stangen mit vier Spanndrähten sein können) und je einer Antenne kann billiger sein als das Einbuddeln von Glasfaserkabeln über eine Strecke von 50 km quer durch dicht besiedelte Wohngebiete oder Berge. Es kann auch billiger sein als das Anmieten von Glasfaseranschlüssen von einer Telefongesellschaft, insbesondere wenn die Telefongesellschaft noch nicht einmal die Kupferkabel voll abbezahlt hat, die zugunsten der Glasfaser herausgerissen wurden.

Politische Regelungen zu den elektromagnetischen Frequenzbereichen

Um das totale Chaos zu vermeiden, gibt es nationale und internationale Vereinbarungen über die Nutzung der Frequenzen. Da jeder eine höhere Datenübertragungsrate anstrebt, möchte jeder mehr Frequenzbereiche. Die Landesregierungen vergeben die Frequenzbereiche für Mittelwellen und UKW-Radio, Fernsehen und Mobiltelefone sowie für Telefongesellschaften, Polizei, Seefahrt, Navigation, Militär, Regierung und viele andere Anwendungsbereiche. International versucht eine Behörde der ITU-R (WRC) diese Zuweisungen zu koordinieren, damit Geräte hergestellt werden können, die in verschiedenen Ländern funktionieren. Die Länder sind jedoch nicht an die Empfehlungen der ITU-R gebunden. So hat die Telekommunikationsbehörde FCC (Federal Communication Commission), die in den Vereinigten Staaten für die Zuweisung verantwortlich ist, die Empfehlungen von ITU-R in einigen Fällen zurückgewiesen (in der Regel weil eine politisch einflussreiche Gruppe hierzu einen Frequenzbereich hätte aufgeben müssen).

Selbst wenn ein Frequenzbereich einer bestimmten Verwendung zugewiesen ist, wie etwa Mobiltelefonen, besteht weiterhin die Frage, welcher Betreiber welche Frequenzen nutzen darf. In der Vergangenheit wurden oftmals drei Verfahren angewendet. Das

älteste Verfahren, häufig als **Schönheitswettbewerb** bezeichnet, erfordert, dass jeder Betreiber begründet, warum sein Vorschlag dem öffentlichen Interesse am besten gerecht wird. Regierungsvertreter entscheiden dann, welche der netten Geschichten ihnen am besten gefällt. Wenn einige Staatsdiener Güter, die Milliarden wert sind, an ihre Lieblingsfirma vergeben, führt dies oft zu Bestechung, Korruption, Vetternwirtschaft und noch Schlimmerem. Darüber hinaus kann auch ein ehrenhafter Staatsdiener in Erklärungsnot kommen, wenn er der Ansicht ist, dass ein ausländisches Unternehmen die Aufgabe besser löst als alle einheimischen Firmen.

Diese Beobachtungen führten zum Verfahren Nummer zwei: ein **Losverfahren** für alle Interessenten durchzuführen. Das Problem bei dieser Idee ist, dass auch Unternehmen, die den Frequenzbereich nicht nutzen möchten, an der Auslosung teilnehmen können. Wenn beispielsweise eine Fast-Food- oder Schuhkette gewinnt, kann diese den Frequenzbereich an einen Betreiber ohne Risiko mit Riesengewinnen weiterverkaufen. Diese Möglichkeit der zufälligen und unerwarteten Gewinnmitnahme hat den Unternehmen viel Kritik eingebracht, sodass man zu Verfahren Nummer drei überging: die **Versteigerung** der Bandbreite an den Höchstbietenden. Als die britische Regierung im Jahr 2000 die Frequenzen für die dritte Generation der Mobiltelefone versteigerte, ging man davon aus, dass dies knapp 4 Milliarden Euro einbringen würde. Tatsächlich wurden aber 38 Milliarden Euro erzielt, weil sich die Betreiber eine Bieterschlacht lieferten und jeder in Todesangst befürchtete, die nächste Welle bei der Mobilfunkkommunikation zu verpassen. Diese interessante Einnahmequelle ließ auch andere Regierungen hellhörig werden und in verschiedenen anderen Ländern wurden die Frequenzen ebenso versteigert. Es funktionierte auch hier, aber einige der Betreiber hatten sich so hoch verschuldet, dass sie nahezu bankrott waren. Selbst in den besten Fällen wird es viele Jahre dauern, bis die Lizenzgebühren wieder hereingeholt sind.

Ein völlig anderer Ansatz bei der Zuteilung von Frequenzen ist, diese überhaupt nicht zuzuweisen. Stattdessen darf jeder nach Belieben übertragen, aber die verwendete Leistung wird reguliert, sodass die Stationen eine geringe Reichweite haben und einander nicht stören. Dementsprechend haben die meisten Regierungen einige Frequenzbänder, die sogenannten **ISM-Bänder** (was für *Industrial, Scientific, Medical* steht), für unlizenzierten Gebrauch reserviert. ISM-Bänder werden bei automatischen Garagentüröffnern, schnurlosen Telefonen, funkgesteuerten Spielzeugen, Funkmäusen und einer Vielzahl anderer drahtloser Haushaltsgeräte eingesetzt. Um die Störungen zwischen nicht aufeinander abgestimmten Geräten zu minimieren, fordert das FCC, dass alle Geräte, die die ISM-Bänder nutzen, ihre Übertragungsleistung begrenzen (z.B. auf 1 Watt) und weitere Techniken einsetzen, um ihre Signale über einen Frequenzbereich zu verstreuhen. Es kann auch nötig sein, dass die Geräte selbst darauf achten, Interferenzen mit Radaranlagen zu vermeiden.

Die Zuordnung der ISM-Bänder ist in den verschiedenen Ländern etwas unterschiedlich. In den Vereinigten Staaten beispielsweise nutzen Netzgeräte in der Praxis die in ► Abbildung 2.13 dargestellten Bänder, ohne dass hierfür eine FCC-Lizenz erforderlich ist. Das 900-MHz-Band wurde für frühe Versionen von IEEE 802.11 eingesetzt, ist aber überbelegt. Das 2,4-GHz-Band ist in den meisten Ländern verfügbar und wird häufig

für IEEE 802.11b/g und Bluetooth verwendet, obwohl es Störungen von Mikrowellenherden und Radaranlagen ausgesetzt ist. Der 5-GHz-Bereich des Spektrums umfasst **U-NII-Bänder** (*Unlicensed National Information Infrastructure*). Die 5-GHz-Bänder sind relativ unentwickelt, sie gewinnen jedoch – da sie die größte Bandbreite haben und von IEEE 802.11a eingesetzt werden – schnell an Beliebtheit.

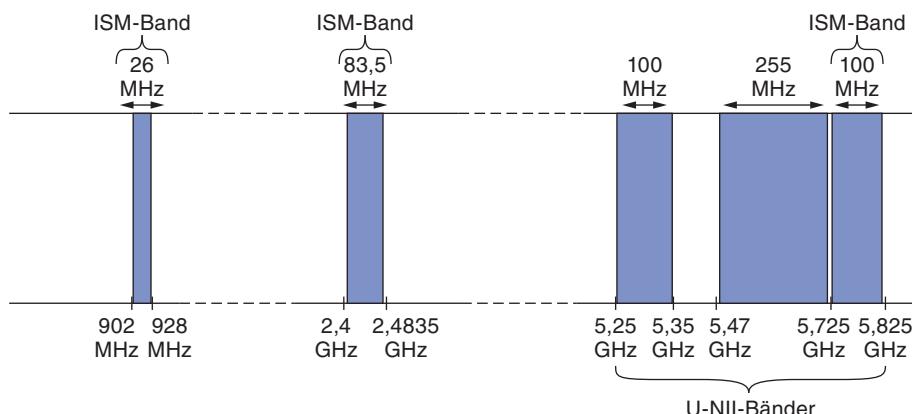


Abbildung 2.13: Die ISM- und U-NII-Bänder, die in den Vereinigten Staaten von drahtlosen Geräten genutzt werden.

Die unlizenzierten Bänder waren im letzten Jahrzehnt ein Bombenerfolg. Die Fähigkeit, das Spektrum frei zu nutzen, hat eine Welle an Innovationen bei drahtlosen LANs und PANs entfesselt, was durch die Verbreitung von Technologien wie IEEE 802.11 und Bluetooth bewiesen wird. Um diese Innovationen fortzusetzen, wird mehr Spektrum benötigt. Eine spannende Entwicklung in den USA ist die FCC-Entscheidung im Jahr 2009, die unlizenzierte Nutzung der **White Spaces** im Bereich um 700 MHz zu erlauben. White Spaces sind Frequenzbänder, die zwar zugewiesen wurden, aber lokal nicht benutzt werden. Durch den Übergang von der analogen zur vollständig digitalen Fernsehübertragung in den USA im Jahr 2010 wurden die White Spaces im Bereich um 700 MHz frei. Die einzige Schwierigkeit bei der Benutzung der White Spaces ist, dass unlizenzierte Geräte in der Lage sein müssen, jeden lizenzierten Sender in der Nähe zu entdecken. Dazu gehören auch schnurlose Mikrofone, die das Frequenzband vorrangig nutzen dürfen.

Eine andere hektische Betriebsamkeit spielt sich um das 60-GHz-Band ab. 2001 öffnete das FCC den Bereich 57 GHz bis 64 GHz für den unlizenzierten Betrieb. Dieser Bereich stellt einen enormen Abschnitt des Spektrums dar, der größer als alle anderen ISM-Bänder zusammen ist – ausreichend also, um die Art von Hochgeschwindigkeitsnetzen zu unterstützen, die für die Übertragung von hochauflösendem Fernsehen (HDTV) über die Luft quer durch Ihr Wohnzimmer nötig wäre. Bei 60 MHz werden Funkwellen von Sauerstoff absorbiert. Dies bedeutet, Signale pflanzen sich nicht sehr weit fort, daher eignen sie sich gut für Kurzstreckennetze. Die hohen Frequenzen (60 GHz liegt im EHF-Bereich (*Extreme High Frequency*) oder Millimeterband, direkt unterhalb der Infrarotstrahlen) stellten anfangs eine Herausforderung für die Gerätehersteller dar, inzwischen sind jedoch Produkte auf dem Markt.

2.3.4 Infrarotübertragung

Ungerichtete Infrarotwellen werden heute viel in der Kurzstreckenkommunikation benutzt. Die Fernsteuerung für Fernsehergeräte, Videorekorder und Stereoanlagen läuft z.B. über Infrarot. Sie sind einigermaßen gerichtet, preisgünstig und einfach herzustellen, haben aber einen entscheidenden Nachteil: Sie können keine festen Gegenstände durchdringen (stellen Sie sich einmal zwischen Ihren Fernseher und die Fernsteuerung, dann geht mit Knöpfchendrücken nichts mehr). Im Allgemeinen gilt: Je weiter man sich vom Langwellenfunk hin zum sichtbaren Licht bewegt, umso mehr verhalten sich die Wellen wie Licht und umso weniger wie Funkwellen.

Andererseits hat aber die Tatsache, dass Infrarotwellen keine Mauern durchdringen können, auch Vorteile. Es bedeutet, dass ein Infrarotsystem in einem Raum eines Gebäudes keine ähnlichen Systeme in benachbarten Räumen stören kann – Sie können mit Ihrer Fernsteuerung nicht den Fernseher Ihres Nachbarn steuern. Außerdem ist die Sicherheit gegen Abhören viel höher als bei Funksystemen. Deshalb ist für das Betreiben eines Infrarotsystems im Gegensatz zu Funksystemen, die außerhalb der ISM-Bänder lizenziert werden müssen, keine amtliche Zulassung erforderlich. Die infrarotbasierte Kommunikation hat im Desktop-Bereich eine begrenzte Verwendung, um beispielsweise Notebooks und Drucker über den **IrDA**-Standard (*Infrared Data Association*) zu verbinden, spielt aber im Kommunikationsbereich keine herausragende Rolle.

2.3.5 Lichtübertragung

Die ungerichtete optische Signalübertragung, auch **optischer Richtfunk** (*free-space optics*) genannt, wird schon seit Jahrhunderten benutzt. Paul Revere¹ benutzte die binäre optische Signalisierung von der Old North Church aus, kurz bevor er zu seinem berühmten Ritt aufbrach. Ein modernerer Anwendungsbereich ist die Verbindung der LANs zweier Gebäude über Laser, die auf den Dächern montiert werden. Die optische Signalübertragung über Laser ist prinzipiell unidirektional, sodass beide Enden mit einem Laser und einem Fotodetektor ausgestattet werden müssen. Diese Technik bietet eine sehr hohe Bandbreite zu geringen Kosten und ist relativ sicher, weil es schwierig ist, einen schmalen Laserstrahl anzuzapfen. Außerdem ist sie relativ leicht zu installieren und setzt im Gegensatz zur Mikrowellenübertragung keine FCC-Lizenz voraus.

Die Stärke des Lasers, ein sehr schmaler Strahl, ist gleichzeitig seine Schwäche. Die Ausrichtung eines Laserstrahls mit einer Breite von 1 mm auf einen Nadelkopf, der 500 Meter entfernt ist, erfordert die Geschicklichkeit einer modernen Annie Oakley². Normalerweise werden Linsen in das System eingefügt, um den Strahl leicht zu defokussieren. Die Schwierigkeiten werden dadurch vergrößert, dass Wind- und Temperaturänderungen den Strahl verfälschen können und Laserstrahlen auch Regen oder dichten Nebel nicht durchdringen können, auch wenn sie an sonnigen Tagen gut

¹ (Anm. d. Übers.) Paul Revere (1735–1818), US-amerikanischer Freiheitskämpfer. 1775 ritt Revere von Boston nach Lexington, um eine Warnung über die Truppenbewegung der britischen Armee zu überbringen.

² (Anm. d. Übers.) Annie Oakley (1860–1926), US-amerikanische Kunstschiützin.

funktionieren. Viele dieser Faktoren sind jedoch kein Thema, wenn es darum geht, zwei Raumschiffe miteinander zu verbinden.

Einer der Autoren dieses Buchs (ATS) hat einmal an einer Konferenz in einem modernen europäischen Hotel teilgenommen, in dem die Organisatoren der Konferenz wohlmeinend einen Raum mit Terminals ausstatteten, damit die Tagungsteilnehmer während langweiliger Vorträge ihre E-Mails lesen konnten. Da die örtliche Telefongesellschaft nicht bereit war, für nur drei Tage zahlreiche Telefonleitungen zu installieren, stellten die Organisatoren auf dem Dach des Hotels einen Laser auf und richteten ihn auf das Informatikgebäude ihrer ein paar Kilometer entfernten Universität aus. Sie testeten die Anlage am Abend vor Beginn der Veranstaltung; sie funktionierte prima. Um 9 Uhr früh, einem sonnigen Morgen, versagte die Verbindung völlig und war den ganzen Tag über nicht wiederherstellbar. Dies wiederholte sich an den nächsten zwei Tagen. Erst nach der Konferenz entdeckten die Veranstalter das Problem: Die Wärme der Sonneneinstrahlung während des Tages verursachte Konvektionsströme, die vom Dach des Gebäudes aufstiegen (► Abbildung 2.14). Dieser Luftwirbel lenkte den Strahl ab, sodass er praktisch um den Detektor tanzte, ähnlich wie bei flimmernden Straßen an heißen Sommertagen. Hieraus lernen wir: damit ungerichtete optische Verbindungen sowohl unter schwierigen als auch unter guten Bedingungen zuverlässig funktionieren, müssen sie mit einem ausreichenden Spielraum für Fehler konstruiert sein.

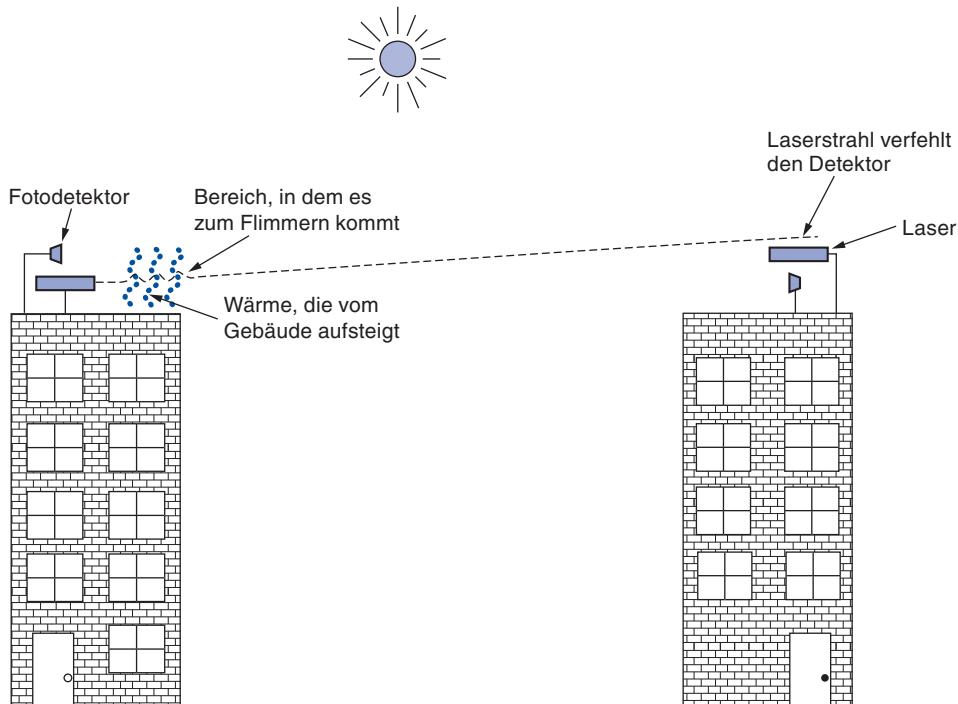


Abbildung 2.14: Konvektionsströme können Störungen bei Laserkommunikationssystemen verursachen. Die Abbildung zeigt ein bidirektionales System mit zwei Lasern.

Ungerichtete optische Kommunikation mag heute aussehen wie eine exotische Netzwerktechnologie, doch diese Technologie könnte sich bald mehr durchsetzen. Wir sind umgeben von Kameras (die Licht wahrnehmen) und Displays (die Licht mithilfe von LEDs und anderen Technologien aussenden). Die Datenkommunikation kann als obere Schicht auf diese Displays aufgesetzt werden, indem Information in das An- und Ausschaltmuster der LEDs codiert wird, welches unterhalb der menschlichen Wahrnehmungsschwelle liegt. Solch eine Kommunikation mit sichtbarem Licht ist prinzipiell sicher und erzeugt ein langsames Netz in der unmittelbaren Umgebung des Displays. Dies könnte allerlei abstruse Ubiquitous-Computing-Szenarien ermöglichen. Das Blaulicht von Rettungsfahrzeugen könnte Ampeln und Autos in der Nähe alarmieren, um zu helfen, den Weg freizugeben. Informationsschilder könnten Landkarten aussenden. Und festliche Lichter könnten Lieder ausstrahlen, die mit ihren Displays synchronisiert sind.

2.4 Kommunikationssatelliten

In den 1950er und Anfang der 1960er Jahre versuchte man, Kommunikationssysteme einzurichten, bei denen Signale an metallisierten Wetterballons abprallten. Leider waren die empfangenen Signale zu schwach, um für irgendetwas tauglich zu sein. Dann entdeckte die US-Marine einen permanenten Wetterballon am Himmel – den Mond – und baute ein betriebsfähiges System zur Kommunikation zwischen Schiff und Land, das den Mond als Abprallpunkt nutzen sollte.

Weitere Fortschritte auf diesem himmlischen Kommunikationsgebiet mussten allerdings warten, bis der erste Satellit gestartet wurde. Der wichtigste Unterschied zwischen einem künstlichen und einem echten Satelliten liegt darin, dass der künstliche Satellit Signale verstärken kann, bevor er sie zurückschickt – somit wurde aus einer kuriosen Idee ein mächtiges Kommunikationssystem.

Kommunikationssatelliten verfügen über einige interessante Eigenschaften, die sie für bestimmte Anwendungen interessant machen. Einen Kommunikationssatelliten kann man sich als großen Mikrowellenverstärker am Himmel vorstellen. Er enthält einen oder mehrere **Transponder** für verschiedene Frequenzbereiche, die das eingehende Signal verstärken und auf einer anderen Frequenz zurücksenden, um Störungen mit dem eingehenden Signal zu vermeiden. Diese Betriebsart wird als **Bent Pipe** (gebogenes Rohr) bezeichnet. Um Datenströme einzeln zu behandeln oder innerhalb des gesamten Bands umzuleiten, kann zusätzlich digitale Verarbeitung eingesetzt werden oder der Satellit kann sogar digitale Informationen empfangen und zurücksenden. Eine derartige Rückkopplung von Signalen verbessert die Leistungsfähigkeit im Vergleich zu einer Bent Pipe, weil der Satellit das Rauschen im hochkommenden Signal nicht mitverstärkt. Die abwärtsgerichteten Strahlen können breit sein und über einen großen Teil der Erdoberfläche gehen, oder sie sind schmal und bedecken nur eine Fläche mit einem Durchmesser von einigen 100 km.

Nach dem Kepler'schen Gesetz hängt die Umlaufzeit eines Satelliten vom Orbitalradius hoch $3/2$ ab. Je höher der Satellit, umso länger die Umlaufzeit. Nahe der Erd-

oberfläche beträgt die Umlaufzeit 90 Minuten. Satelliten mit niedrigen Umlaufbahnen verliert man sehr schnell aus den Augen, sodass hier zur flächendeckenden Versorgung viele benötigt werden. Außerdem müssen sie von Bodenantennen nachverfolgt werden. In einer Höhe von 35 800 km beträgt die Umlaufzeit 24 Stunden. In einer Höhe von 384 000 km beträgt die Umlaufzeit ungefähr einen Monat, wie jeder bestätigen kann, der regelmäßig den Mond beobachtet.

Die Umlaufzeit eines Satelliten ist wichtig, aber es ist nicht das einzige Kriterium, um seinen Standort festzulegen. Ein weiterer Punkt ist die Existenz des Van-Allen-Strahlungsgürtels. Dies sind Schichten mit stark geladenen Partikeln, die durch das Magnetfeld der Erde gefangen sind. Ein Satellit, der hier durchfliegt, würde sehr schnell durch die Partikel zerstört werden. Aus diesen Gründen gibt es drei Bereiche, in denen sich Satelliten sicher aufhalten können. Diese Bereiche und einige ihrer Eigenschaften sind in ▶ Abbildung 2.15 dargestellt. Im Folgenden beschreiben wir kurz die Satelliten, die sich in diesen Bereichen befinden.

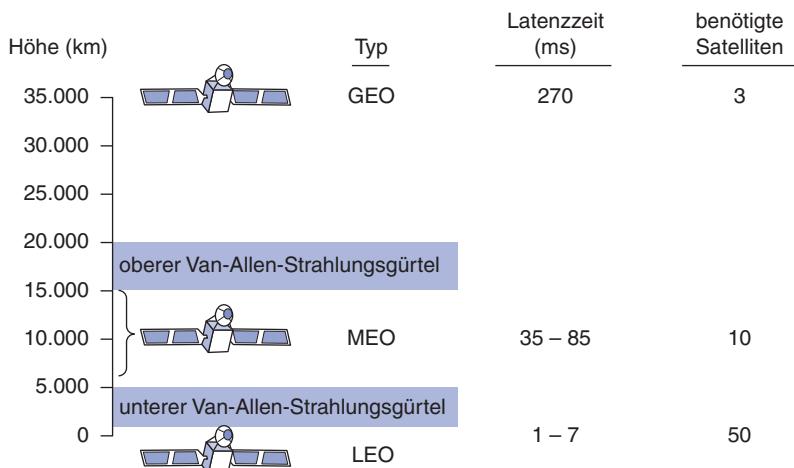


Abbildung 2.15: Kommunikationssatelliten mit einigen Eigenschaften, wie Entfernung von der Erde, Umlaufverzögerung und Anzahl der Satelliten, die für eine weltweite Abdeckung erforderlich ist.

2.4.1 Geostationäre Satelliten

1945 berechnete der Science-Fiction-Autor Arthur C. Clarke, dass ein Satellit in einer Höhe von 35 800 km in einer kreisförmigen, äquatornahen Umlaufbahn im Himmel als bewegungslos wahrgenommen und daher nicht nachverfolgt werden müsste (Clarke, 1945). Er beschrieb ein vollständiges Kommunikationssystem, das diese (bemannten) **geostationären Satelliten** verwendet, einschließlich der Umlaufbahnen, Solaranlagen, Funkfrequenzen und dem Ablauf des Starts. Leider traf er die Schlussfolgerung, dass Satelliten nicht möglich sind, da leistungshungrige, zerbrechliche Vakuumröhrenverstärker nicht in die Umlaufbahn gebracht werden könnten, sodass er die Idee nicht weiterverfolgte, obwohl er einige Science-Fiction-Romane darüber schrieb.

Die Erfindung des Transistors änderte dies grundlegend und im Juli 1962 wurde der erste künstliche Kommunikationssatellit Telstar gestartet. Seither sind Kommunikationssatelliten ein Geschäft in Milliardenhöhe und die einzige Entwicklung, die das Weltall nutzt und dabei hochprofitabel ist. Die in großen Höhen fliegenden Satelliten werden oft als **GEO-Satelliten** (*Geostationary Earth Orbit*, geostationäre Umlaufbahn) bezeichnet.

Mit der heutigen Technologie ist es nicht sinnvoll, geostationäre Satelliten sehr viel enger als 2 Grad nebeneinander in der 360-Grad-Äquatorialebene zu platzieren, um Störungen zu vermeiden. Bei einem Abstand von 2 Grad können sich nur $360/2 = 180$ geostationäre Satelliten auf dieser Umlaufbahn befinden. Jeder Transponder kann aber mehrere Frequenzen und Polarisationen verwenden, um die verfügbare Bandbreite zu erhöhen.

Um das totale Chaos am Himmel zu vermeiden, wird die Umlaufbahnposition von der ITU zugewiesen. Dieser Prozess ist hochpolitisch, weil auch Länder, die gerade erst die Steinzeit hinter sich gelassen haben, „ihre“ Umlaufposition fordern (damit sie diese an den höchsten Bieter vermieten können). Andere Länder vertreten wiederum die Ansicht, dass die nationalen Besitzrechte sich nicht bis zum Mond erstrecken und kein Land gesetzliche Ansprüche auf Umlaufpositionen über dem Landesgebiet hat. Erschwerend kommt noch hinzu, dass die kommerzielle Telekommunikation nicht das einzige Anwendungsgebiet ist. Fernsehgesellschaften, Regierungen und das Militär möchten auch ein Stück vom Umlaufbahnkuchen.

Moderne Satelliten können ziemlich groß sein, bis zu 5 000 kg wiegen und einige Kilowatt Strom konsumieren, den die Solargeneratoren erzeugen. Durch die Einwirkungen von Sonnen-, Mond- und Planetengravitation tendieren sie dazu, von den zugewiesenen Umlaufbahnen und -richtungen abzuweichen. Dem wird mit den für diese Zwecke montierten Raketenmotoren des Satelliten entgegengesteuert. Diese Feinabstimmung wird als **Positionsbeibehaltung** (*station keeping*) bezeichnet. Wenn aber der Brennstoff für die Motoren erschöpft ist (in der Regel nach zehn Jahren), dann treibt der Satellit hilflos umher, sodass er abgeschaltet werden muss. Mit der Zeit verlässt er seine Umlaufbahn und tritt wieder in die Erdatmosphäre ein, wo er verbrennt oder – sehr selten – auf die Erde aufschlägt.

Umlaufpositionen sind nicht der einzige Streitpunkt. Auch um Frequenzen wird gekämpft, da die Downlink-Übertragung mit den verwendeten Mikrowellen in Konflikt gerät. Deshalb hat die ITU bestimmte Frequenzbänder für Satellitennutzer zugeteilt. Die wichtigsten sind in ▶ Abbildung 2.16 aufgeführt. Das C-Band war das erste Band, das für den kommerziellen Satellitenverkehr ausgelegt war. Zwei Frequenzbereiche können darauf zugewiesen werden. Der untere Bereich ist für Downlink-Verkehr, der vom Satelliten zur Erde fließt, und der obere für den Uplink-Verkehr zum Satelliten. Damit der Datenverkehr gleichzeitig in beide Richtungen ablaufen kann, sind zwei Kanäle erforderlich. Diese Kanäle sind bereits überlastet, weil sie auch von den Betreibern terrestrischer Mikrowellenanschlüsse benutzt werden. Die L- und S-Bänder wurden im Jahr 2000 im Rahmen eines internationalen Abkommens hinzugefügt. Sie sind aber eng und ebenfalls sehr gut belegt.

| Band | Downlink | Uplink | Bandbreite | Probleme |
|------|----------|---------|------------|-------------------------------|
| L | 1,5 GHz | 1,6 GHz | 15 MHz | Geringe Bandbreite; überfüllt |
| S | 1,9 GHz | 2,2 GHz | 70 MHz | Geringe Bandbreite; überfüllt |
| C | 4,0 GHz | 6,0 GHz | 500 MHz | Bodeninterferenzen |
| Ku | 11 GHz | 14 GHz | 500 MHz | Regen |
| Ka | 20 GHz | 30 GHz | 3500 MHz | Regen, Gerätekosten |

Abbildung 2.16: Die wichtigsten Satellitenbänder.

Das nächsthöhere für die Telekommunikation verfügbare Band ist das Ku-Band (von *K under*). Dieses Band ist (noch) nicht überlastet. Außerdem können Satelliten, die in den höheren Frequenzbereich dieses Bandes arbeiten, ein Grad voneinander entfernt positioniert werden. Allerdings tritt hier ein anderes Problem auf: Regen. Wasser kann diese kurzen Mikrowellen leicht absorbieren. Normalerweise sind große Stürme aber lokal begrenzt, sodass sich das Problem mit mehreren weit auseinanderliegenden Erdstationen lösen lässt. Dabei fallen aber Zusatzkosten für weitere Antennen, Kabel und Elektronikbauteile an, um zwischen den Stationen schnell umschalten zu können. Die Bandbreite im Ka-Band (von *K above*) wurde ebenfalls für kommerziellen Satellitenverkehr reserviert, aber die dafür erforderlichen Anlagen sind sehr teuer. Außer diesen kommerziellen Bändern gibt es noch viele staatliche und militärische Bänder.

Ein moderner Satellit hat etwa 40 Transponder, meistens mit einer Bandbreite von 36 MHz. In der Regel arbeitet ein Transponder im Bent-Pipe-Prinzip, aber neuere Satelliten verfügen auch über eigene Rechenkapazitäten, die einen ausgefilterten Betrieb ermöglichen. Bei den ersten Satelliten war die Aufteilung der Transponder in Kanäle statisch: Die Bandbreite war einfach in feste Frequenzbänder unterteilt. Heutzutage wird jeder Transponderstrahl in verschiedene Zeitscheiben unterteilt, wobei die verschiedenen Benutzer der Reihe nach drankommen. Wir behandeln diese beiden Verfahren (Frequenzmultiplexing und Zeitmultiplexing) später in diesem Kapitel genauer.

Die ersten geostationären Satelliten hatten einen einzigen Strahl, der etwa 1/3 der Erdoberfläche abdeckte und als **Ausleuchtzone** (*footprint*) bezeichnet wird. Da bei Mikroelektronikkomponenten Preis, Größe und Strombedarf enorm gesunken sind, wurden anspruchsvollere Übertragungstechniken möglich. Ein Satellit ist mit mehreren Antennen und Transpondern ausgerüstet. Jeder abwärtsgerichtete Strahl kann auf ein kleines geografisches Gebiet ausgerichtet werden. Dadurch sind gleichzeitig mehrere auf- und abwärtsgerichtete Übertragungen möglich. Diese sogenannten **scharf gebündelten Funkstrahlen** (*spot beam*) sind normalerweise elliptisch und können einen Durchmesser von nur einigen Hundert Kilometern haben. Ein Kommunikationssatellit für die USA hat in der Regel einen breiten Strahl für die 48 zusammenhängenden Bundesstaaten sowie scharf gebündelte Funkstrahlen für Alaska und Hawaii.

Eine Innovation in der Welt der Kommunikationssatelliten ist die Entwicklung kostengünstiger Mikrostationen, die auch **VSATs** (*Very Small Aperture Terminal*; Abramson,

2000) genannt werden. Diese winzigen Terminals haben Antennen von 1 m Durchmesser (im Unterschied zu 10 m für eine GEO-Standardantenne) und können eine Leistung von etwa 1 Watt erbringen. Uplink unterstützt im Allgemeinen bis zu 1 Mbit/s, während die Abwärtsverbindung meist bis zu einigen Mbit/s aufweist. Direkt sendendes Satellitenfernsehen verwendet diese Technologie für unidirektionale Übertragungen.

In vielen VSAT-Systemen haben die Mikrostationen nicht genügend Leistung, um direkt (selbstverständlich über den Satelliten) miteinander zu kommunizieren. Hierfür ist eine spezielle Bodenstation, ein **Hub**, mit einer großen leistungsstarken Antenne erforderlich, um den Verkehr zwischen den VSATs zu vermitteln (► Abbildung 2.17). In diesem Betriebsmodus hat entweder der Sender oder der Empfänger eine große Antenne und einen leistungsstarken Verstärker. Für die preisgünstigeren Endbenutzerstationen muss eine längere Übertragungszeit in Kauf genommen werden.

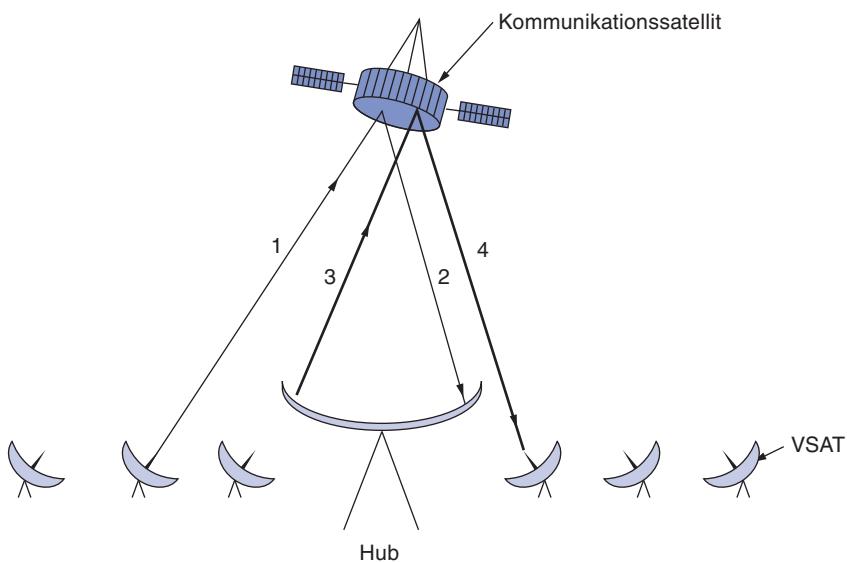


Abbildung 2.17: VSAT-Schüsseln, die einen Hub verwenden.

VSATs sind in ländlichen Gebieten sehr interessant. Es ist zwar nicht allgemein bekannt, aber über die Hälfte der Weltbevölkerung lebt über eine Stunde Gehzeit vom nächsten Telefon entfernt. Das Verlegen von Telefonleitungen in Tausende kleiner Dörfer ist für viele Länder der dritten Welt einfach unerschwinglich. Die Installation einer solarbetriebenen VSAT-Schüssel mit einem Meter Durchmesser ist aber oftmals machbar. VSAT-Schüsseln stellen die Technologie bereit, die die Welt verbinden wird.

Kommunikationssatelliten haben verschiedene Eigenschaften, durch die sie sich völlig von terrestrischen Punkt-zu-Punkt-Verbindungen unterscheiden. Auch wenn die Signale von und zu einem Satelliten in Lichtgeschwindigkeit (fast 300 000 km/s) übertragen werden, ergibt sich durch die enorme Entfernung eine beträchtliche Verzögerung für GEO-Satelliten. Je nach Entfernung zwischen Benutzer und Bodenstation und der Erhebung des Satelliten über dem Horizont kann die Übertragungszeit von

Endpunkt zu Endpunkt bei 250 bis 300 ms liegen. Ein typischer Mittelwert ist 270 ms (540 ms bei einem VSAT-System mit einem Hub).

Im Vergleich dazu haben terrestrische Mikrowellenverbindungen eine Signalausbreitungsverzögerung von ungefähr 3 µs/km und Koaxial- oder Glasfaserkabel eine Verzögerung von ca. 5 µs/km (elektromagnetische Signale bewegen sich in der Luft schneller als in Feststoffen).

Eine weitere wichtige Eigenschaft von Satelliten ist, dass sie eigentlich Broadcasting-Medien sind. Es kostet nicht mehr, eine Nachricht an Tausende von Stationen innerhalb der Reichweite des Transponders zu versenden als an eine einzige. Für manche Anwendungsbereiche ist diese Eigenschaft sehr nützlich. Stellen Sie sich beispielsweise einen Satelliten vor, der normale Webseiten an die Cache-Speicher einer großen Zahl von Computern überträgt, die über ein weites Gebiet verteilt sind. Broadcasting kann zwar durch eine Punkt-zu-Punkt-Leitung simuliert werden, ist aber über Satellit wesentlich kostengünstiger. Andererseits sind Satelliten in Bezug auf Datenschutz ein völliges Desaster. Jeder kann alles mithören. Wenn Sicherheit gefordert ist, ist die Verschlüsselung daher eine absolute Bedingung.

Bei Satelliten sind die Kosten der Übertragung einer Nachricht außerdem von der zurückgelegten Entfernung unabhängig. Die Bereitstellung eines Ferngesprächs über den Atlantik kostet nicht mehr als ein Gespräch auf die andere Straßenseite. Satellitenanschlüsse weisen auch ausgezeichnete Fehlerraten auf und können ohne viel Aufwand installiert werden, was besonders für die Katastrophenhilfe und den militärischen Bereich interessant ist.

2.4.2 MEO-Satelliten

In viel geringeren Höhen, zwischen den beiden Van-Allen-Strahlungsgürteln, befinden sich die **MEO**-Satelliten (*Medium-Earth Orbit*). Von der Erde aus betrachtet wandern diese langsam an den Längengraden entlang und benötigen für eine Erdumrundung etwa sechs Stunden. Dementsprechend müssen sie bei ihrer Reise über den Himmel verfolgt werden. Da sie eine sehr viel niedrigere Umlaufbahn als die GEO-Satelliten aufweisen, haben sie eine geringere Ausleuchtzone und es werden weniger leistungsstarke Sender benötigt, um sie zu kontaktieren. Derzeit werden sie eher für Navigationssysteme als für die Telekommunikation verwendet, daher gehen wir hier nicht weiter auf sie ein. Die Konstellation der rund 30 **GPS**-Satelliten (*Global Positioning System*), die in etwa 20 200 km Höhe die Erde umkreisen, sind MEO-Satelliten.

2.4.3 LEO-Satelliten

Verringern wir die Höhe noch ein wenig, so treffen wir auf die **LEO**-Satelliten (*Low-Earth Orbit*). Aufgrund ihrer schnellen Bewegung werden hier für ein vollständiges System sehr viele Satelliten benötigt. Andererseits müssen die Bodenstationen nicht leistungsstark sein, weil die Satelliten so nahe über der Erde kreisen und die Signalverzögerung nur ein paar Millisekunden beträgt. Auch sind die Kosten für den Start

wesentlich niedriger. In diesem Abschnitt betrachten wir zwei Beispiele von Satellitenkonstellationen für die Sprachkommunikation: Iridium und Globalstar.

In den ersten 30 Jahren des Satellitenzeitalters wurden Satelliten in niedriger Umlaufbahn selten für Kommunikationszwecke benutzt, weil sie wie der Blitz im Sichtfeld auftauchen und genauso schnell wieder verschwinden. 1990 hat Motorola neue Wege eingeschlagen und als erstes Unternehmen beim FCC einen Antrag zur Genehmigung des Starts von 77 Satelliten in die niedrige Umlaufbahn für das Projekt **Iridium** gestellt (Element 77 ist Iridium). Der Plan wurde später dahingehend geändert, dass nur noch 66 Satelliten eingesetzt werden sollten – eigentlich hätte das Projekt in Dysprosium (Element 66) umbenannt werden müssen, aber das hörte sich wahrscheinlich zu sehr nach einer Krankheit an. Die Idee in diesem Projekt war, dass ein Satellit durch den nächsten ersetzt wird, sobald der erste aus dem Sichtfeld verschwindet. Dieser Antrag löste bei anderen Kommunikationsunternehmen eine wahre Hysterie aus. Plötzlich wollte jeder eine Reihe von Satelliten in die niedrige Umlaufbahn schießen.

Nach mehreren Jahren des Kampfes um Partner und Finanzierung begann im November 1998 der Kommunikationsdienst. Unglücklicherweise war der kommerzielle Bedarf für die großen, schweren Satellitentelefone vernachlässigbar, da das Mobilfunknetz seit 1990 auf eindrucksvolle Weise gewachsen war. Daher war Iridium nicht profitabel, die Betreiber mussten im August 1999 Konkurs anmelden. Dies war eines der spektakulärsten Fiaskos eines Unternehmens in der Geschichte. Die Satelliten und andere Sachwerte (im Wert von 5 Milliarden Dollar) wurden in der Folge von einem Investor als extraterrestrisches Schnäppchen für 25 Millionen Dollar erworben.

Der Iridiumdienst startete im März 2001 erneut und ist seitdem gewachsen. Es werden Sprach-, Daten-, Pager-, Fax- und Navigationsdienste weltweit an jedem Ort zu Land, zu Wasser und in der Luft zur Verfügung gestellt – über Handhelds, die direkt mit dem Iridiumsatelliten kommunizieren. Zu den Kunden gehören die Marine, die Luftfahrt, Ölförderindustrien wie auch Personen, die in abgelegene Gebiete ohne Telekommunikationsinfrastruktur reisen (z.B. Wüsten, Berge, der Südpol und einige Dritte-Welt-Länder).

Die Satelliten werden in einer Höhe von 750 km in kreisförmigen polaren Umlaufbahnen positioniert. Sie werden in einer Nord-Süd-Kette angeordnet, wobei die Satelliten jeweils 32 Breitengrade voneinander entfernt sind (►Abbildung 2.18). Jeder Satellit hat maximal 48 scharf gebündelte Punktstrahlen und eine Kapazität von 3 840 Kanälen, wovon einige für Pager- und Navigationsdienste, andere für Daten und Sprache verwendet werden.

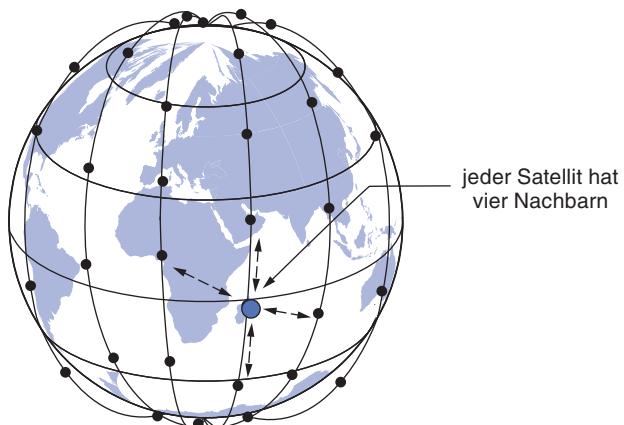


Abbildung 2.18: Die Iridiumsatelliten bilden sechs Ketten rund um die Erde.

Mit sechs Satellitenketten kann die ganze Erde abgedeckt werden (siehe Abbildung 2.18). Eine interessante Eigenschaft von Iridium ist, dass die Kommunikation zwischen entfernten Kunden im All stattfindet, wie in ►Abbildung 2.19a gezeigt. Hier sehen wir einen Anrufer am Nordpol, der den Satelliten direkt über ihm kontaktiert. Jeder Satellit hat vier Nachbarn, mit denen er kommunizieren kann, zwei davon in derselben Kette (abgebildet) und zwei in benachbarten Ketten (nicht abgebildet). Die Satelliten leiten den Anruf über dieses Gitter weiter, bis er schließlich zum Angerufenen am Südpol hinunter gesendet wird.

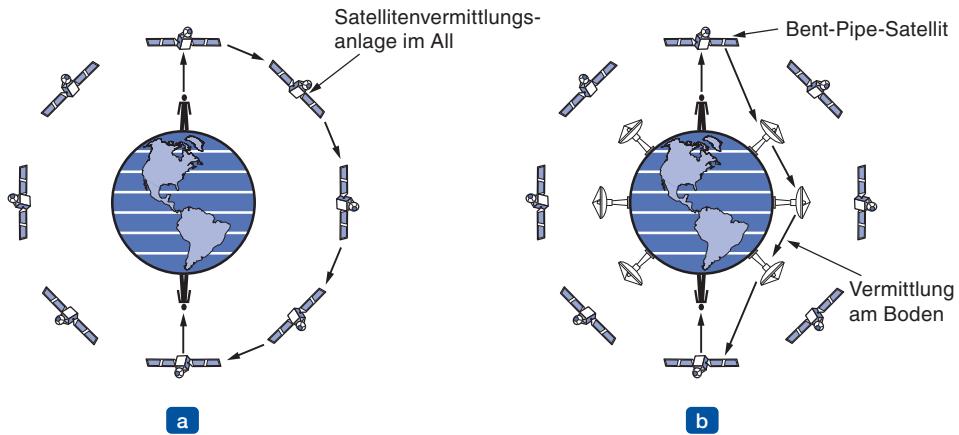


Abbildung 2.19: (a) Weitergabe von Daten im All. (b) Weitergabe von Daten am Boden.

Eine Alternative zu Iridium ist **Globalstar**. Globalstar basiert auf 48 LEO-Satelliten, verwendet aber ein anderes Vermittlungsschema als Iridium. Während Iridium Anrufe von Satellit zu Satellit weitergibt, was eine ausgefeilte Vermittlungsausrüstung in den Satelliten erforderlich macht, verwendet Globalstar den traditionellen Bent-Pipe-Entwurf. Der Anruf, der in ► Abbildung 2.19b am Nordpol begonnen wird, wird zur Erde zurückgesendet und von einer großen Bodenstation aufgenommen. Der Anruf wird

dann über ein terrestrisches Netz an die für den Angerufenen nächstliegende Bodenstation weitergeleitet und über eine Bent-Pipe-Verbindung dem Empfänger wie dargestellt übermittelt. Der Vorteil dieses Schemas ist, dass damit vieles der Komplexität auf den Boden zurückgeholt wird, wo diese einfacher zu verwalten ist. Auch bedeutet die Verwendung von großen Bodenstationsantennen, die starke Signale aussenden und schwache empfangen können, dass hier Telefone mit niedriger Leistung verwendet werden können. Das Telefon gibt nur ein paar Milliwatt Leistung ab, sodass das zur Bodenstation zurückkommende Signal ziemlich schwach ist, selbst wenn es durch den Satelliten verstärkt wurde.

Es werden weiterhin rund 20 Satelliten im Jahr gestartet, darunter auch größere Satelliten, die heute über 5 000 Kilogramm wiegen. Doch für eher kostenbewusste Organisationen gibt es auch sehr kleine Satelliten. Um die Weltraumforschung erschwinglicher zu machen, haben sich im Jahr 1999 Wissenschaftler von Cal Poly und Stanford zusammengetan, um einen Standard für Miniatursatelliten und eine gemeinsame Startvorrichtung festzulegen, wodurch sich die Startkosten erheblich senken ließen (Nugent et al., 2008). **CubeSats** sind Satelliten in der Form von $10 \times 10 \times 10$ -cm-Würfeln, von denen jeder nicht mehr als 1 Kilogramm wiegt und die für nur 40 000 US-Dollar abgeschossen werden können. Die Startvorrichtung fliegt als Sekundärnutzlast von kommerziellen Weltraummissionen. Im Prinzip ist die Vorrichtung eine Röhre, die bis zu drei Einheiten von CubeSats aufnehmen kann und die mithilfe von Sprungfedern die Satelliten auf eine Umlaufbahn bringt. Mehr als 20 CubeSats wurden bisher gestartet, viele weitere sind in Vorbereitung. Die meisten kommunizieren mit Bodenstationen auf den UHF- und VHF-Bändern.

2.4.4 Vergleich: Satelliten und Glasfaser

Ein Vergleich zwischen der Satelliten- und der terrestrischen Kommunikation ist ganz aufschlussreich. Vor nur 25 Jahren konnte man noch behaupten, dass die Zukunft der Telekommunikation bei den Kommunikationssatelliten liege. Schließlich hatte sich das Telefonystem in den vorhergehenden hundert Jahren kaum geändert und es gab auch keine Anzeichen, dass es sich in den nächsten hundert Jahren ändern würde. Dieses Schnekkentempo war zu einem nicht geringen Anteil durch die Regulierungsbestimmungen verursacht worden, in deren Rahmen man erwartete, dass die Telefongesellschaften gute Sprachdienste zu vernünftigen Preisen bereitstellten (was sie taten) und dafür wiederum einen garantierten Gewinn für ihre Investitionen erhielten. Für diejenigen, die Daten übertragen mussten, waren 1 200-Bit/s-Modems verfügbar. Mehr gab es nicht.

Die Einführung des Wettbewerbs in den USA im Jahre 1984 und etwas später in Europa änderte dies radikal. Die Telefongesellschaften begannen, ihre alten Fernnetze durch Glasfaserkabel zu ersetzen, und führten Dienste mit hoher Bandbreite wie ADSL (*Asymmetric Digital Subscriber Line*) ein. Sie beendeten ihre lange gepflegte Praxis, für Ferngespräche künstlich überzogene Tarife zu berechnen, um den lokalen Dienst zu subventionieren. Plötzlich entstand der Eindruck, dass die terrestrischen Glasfaserverbindungen das bevorzugte Medium sein würden.

Dennoch konnten sich Kommunikationssatelliten bestimmte Nischenmärkte erobern, die Glasfaser nicht berücksichtigt (und manchmal auch nicht berücksichtigen kann). Zunächst einmal liegen die Satelliten immer dann vorne, wenn schnelle Installation entscheidend ist. Eine schnelle Reaktion ist für militärische Kommunikationssysteme in Kriegszeiten und bei der Katastrophenhilfe in Friedenszeiten wichtig. Beispielsweise konnten Kommunikationssatelliten nach dem schweren Sumatra-Erdbeben im Dezember 2004 und dem darauffolgenden Tsunami die Verbindung zu Notfallhelfern innerhalb von 24 Stunden wiederherstellen. Diese schnelle Reaktion war möglich, weil es einen entwickelten Markt für Satellitendienstanbieter gibt, in dem große Akteure, wie Intelsat mit über 50 Satelliten, so ziemlich überall Kapazität vermieten können, wo es nötig ist. Für Kunden, die von bestehenden Satellitennetzen bedient werden, kann ein VSAT leicht und schnell eingerichtet werden, um eine Mbit/s-Verbindung weltweit zur Verfügung zu stellen.

Eine zweite Nische ist die Kommunikation an Orten mit schlechter oder keiner terrestrischen Infrastruktur. Viele Leute wollen heutzutage von überall aus kommunizieren können. Mobilfunknetze decken Orte mit guter Bevölkerungsdichte ab, doch an anderen Stellen (z.B. auf See oder in einer Wüste) sind sie weniger geeignet. Im Gegensatz dazu stellt Iridium überall auf der Erde Sprachdienste zur Verfügung, sogar am Südpol. Die Installation von terrestrischer Infrastruktur kann außerdem teuer sein, je nach Gelände und erforderlichen Wegerechten. Indonesien hat zum Beispiel einen eigenen Satelliten für den inländischen Telefonverkehr. Der Start eines Satelliten war kostengünstiger, als Tausende von Unterwasserkabeln zwischen den 13 677 Inseln des Archipels zu verlegen.

Eine dritte Nische sind Anwendungsbereiche, in denen Broadcasting eine wesentliche Rolle spielt. Eine über einen Satelliten übertragene Nachricht kann von Tausenden von Bodenstationen gleichzeitig empfangen werden. Satelliten werden daher häufig eingesetzt, um TV-Programme zu lokalen Stationen zu übertragen. Es gibt heute einen großen Markt für die direkte Satellitenübertragung von digitalem Fernsehen und Radio zu Endnutzern, die Satellitenempfänger in ihren Häusern und Autos haben. Prinzipiell können aber alle Arten von Inhalt via Broadcasting gesendet werden. Beispielsweise könnte ein Unternehmen, das Aktienkurse an Tausende von Börsenhändlern überträgt, ein Satellitensystem viel kostengünstiger finden als die Simulation von Broadcasting auf dem Boden.

Kurz gesagt, es sieht ganz danach aus, als ob in Zukunft die Hauptkommunikationstechnologie eine Kombination von terrestrischen Glasfaserkabeln und Zellulärfunk sein wird, für spezielle Anwendungen sind jedoch Satelliten am besten geeignet. Bei all diesen Möglichkeiten werden die Kosten leicht übersehen. Obwohl Glasfaser mehr Bandbreite bietet, ist es vorstellbar, dass aggressive Preiskämpfe zwischen der terrestrischen und der Satellitenkommunikation stattfinden könnten. Falls technologische Fortschritte die Kosten für den Einsatz eines Satelliten radikal senken (z.B. wenn ein zukünftiges Raumfahrzeug bei einem Start gleich Dutzende von Satelliten ins All schleudern könnte) oder falls sich Satelliten auf niedriger Umlaufbahn im großen Stil durchsetzen, dann ist nicht mehr ganz so sicher, ob die Glasfaser in allen Märkten als Gewinner hervorgehen wird.

2.5 Digitale Modulation und Multiplexing

Nachdem wir nun die Eigenschaften von verkabelten und drahtlosen Kanälen untersucht haben, können wir unsere Aufmerksamkeit dem Problem zuwenden, digitale Informationen zu senden. Kabelgebundene und drahtlose Kanäle übertragen analoge Signale als kontinuierlich variierende Spannung, Lichtstärke oder Schallstärke. Um digitale Informationen zu senden, müssen wir uns überlegen, wie analoge Signale als Bits dargestellt werden. Der Vorgang des Konvertierens zwischen Bits und den dazugehörigen Signalen wird **digitale Modulation** genannt.

Wir beginnen mit Schemata, die Bits direkt in ein Signal konvertieren. Diese führen zur **Basisbandübertragung** (*baseband transmission*), bei der das Signal Frequenzen von null bis zu einem Maximum besetzt. Dieses Maximum hängt von der Signalrate ab. Dieses Vorgehen ist üblich für Kabel. Dann werden wir Schemata betrachten, die die Amplitude, Phase oder Frequenz eines Trägersignals regulieren, um Bits zu transportieren. Daraus ergibt sich die **Übertragung im Durchlassbereich** (*passband transmission*), bei der das Signal ein Frequenzband um die Frequenz des Trägersignals herum besetzt. Dieses Vorgehen ist für drahtlose und optische Kanäle üblich, bei denen die Signale in einem vorgegebenen Frequenzband liegen müssen.

Kanäle werden häufig von mehreren Signalen gemeinsam benutzt. Schließlich ist es sehr viel bequemer, ein einziges Kabel zu verwenden, statt für jedes Signal ein eigenes Kabel zu verlegen. Diese Art der gemeinsamen Nutzung heißt **Multiplexing**. Es kann auf mehrere Weisen durchgeführt werden. Wir werden Methoden für Zeit-, Frequenz- und Codemultiplexing vorstellen.

Die Modulations- und Multiplexverfahren, die wir in diesem Abschnitt beschreiben, sind weitverbreitet für Kabel, Glasfaser, terrestrische drahtlose und Satellitenkanäle. In den folgenden Abschnitten werden wir Beispielnetze sehen, die diese Techniken im Betrieb zeigen.

2.5.1 Basisbandübertragung

Die direkteste Form der digitalen Modulation ist, eine positive Spannung zu nehmen, um eine 1 darzustellen, und eine negative Spannung, um eine 0 darzustellen. Für Glasfaserkabel könnte die Anwesenheit von Licht eine 1 und die Abwesenheit von Licht eine 0 repräsentieren. Dieses Schema heißt **NRZ** (*Non-Return-to-Zero*). Der seltsame Name hat historische Gründe und bedeutet einfach nur, dass das Signal den Daten folgt. Ein Beispiel ist in ► Abbildung 2.20b zu sehen.

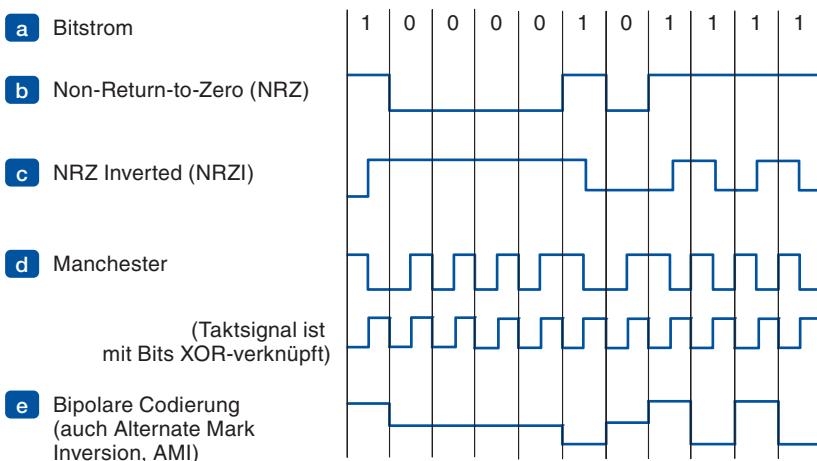


Abbildung 2.20: Leitungscodes: (a) Bits, (b) NRZ, (c) NRZI, (d) Manchester, (e) bipolar oder AMI.

Einmal gesendet, pflanzt sich das NRZ-Signal durch das Kabel weiter fort. Am anderen Ende konvertiert der Empfänger es in Bits, indem das Signal in regelmäßigen Zeitintervallen abgetastet wird. Dieses Signal entspricht nicht exakt dem gesendeten Signal, da es vermutlich durch den Kanal gedämpft wird und durch Rauschen auf der Empfängerseite gestört ist. Um die Bits zu decodieren, bildet der Empfänger die Signalabtastungen auf die am nächsten kommenden Symbole ab. Bei NRZ wird man eine positive Spannung nehmen, um anzusehen, dass eine 1 gesendet wurde, und eine negative Spannung, um anzusehen, dass eine 0 gesendet wurde.

NRZ ist ein guter Ausgangspunkt für unsere Betrachtungen, weil es ein einfaches Verfahren ist, jedoch wird es in der Praxis selten allein benutzt. Komplexere Schemata werden **Leitungscodes** (*line code*) genannt. Im Folgenden beschreiben wir Leitungscodes, die Bandbreiteneffizienz, Taktrückgewinnung und Gleichstromsymmetrie unterstützen.

Bandbreiteneffizienz

Bei NRZ kann das Signal zwischen den positiven und negativen Spannungsniveaus bis hin zu jedem zweiten Bit (wenn Einsen und Nullen abwechselnd vorkommen) springen. Dies bedeutet, dass wir bei einer Bitrate von B Bit/s mindestens eine Bandbreite von $B/2$ Hz benötigen. Diese Relation resultiert aus der Nyquist-Gleichung (2.2). Es ist eine prinzipielle Grenze, wir können also NRZ nicht schneller ausführen, ohne mehr Bandbreite zu benutzen. Bandbreite ist oft eine begrenzte Ressource, selbst für verkabelte Kanäle. Hochfrequente Signale werden zunehmend gedämpft – was sie weniger nützlich macht – und benötigen außerdem schnellere Elektronik.

Eine Vorgehensweise zur effizienteren Ausnutzung der begrenzten Bandbreite ist, mehr als zwei Signalebenen zu verwenden. Wenn beispielsweise vier Spannungsniveaus benutzt werden, können wir zwei Bits auf einmal als ein **Symbol** senden. Dieses Konzept funktioniert so lange, wie das Signal beim Empfänger ausreichend stark ist, um die vier Niveaus zu unterscheiden. Die Rate, mit der das Signal sich ändert, ist dann die Hälfte der Bitrate, die benötigte Bandbreite ist also verringert worden.

Wir nennen die Rate, mit der sich das Signal ändert, die **Symbolrate**, um es von der **Bitrate** zu unterscheiden. Die Bitrate entspricht der Symbolrate multipliziert mit der Anzahl der Bits pro Symbol. Ein älterer Name für die Symbolrate, insbesondere im Kontext mit Telefonmodems, welche digitale Daten über Telefonleitungen transportieren, ist **Baudrate**. In der Fachliteratur werden die Begriffe „Bitrate“ und „Baudrate“ oft falsch benutzt.

Die Anzahl der Signalebenen muss nicht unbedingt eine Potenz von zwei sein. Dies kommt häufig vor, wenn einige Ebenen zum Schutz vor Fehlern und zur Vereinfachung des Empfängerentwurfs benutzt werden.

Taktrückgewinnung

Bei allen Schemata zur Codierung von Bits in Symbole muss der Empfänger wissen, wann ein Symbol endet und das nächste beginnt, um die Bits korrekt zu decodieren. Bei NRZ sind die Symbole einfach nur Spannungsniveaus – eine lange Reihe von Nullen oder Einsen lässt das Signal also unverändert. Nach einer Weile ist es schwer, die einzelnen Bits auseinanderzuhalten, da eine Reihe von 15 Nullen einer Reihe von 16 Nullen recht ähnlich sieht, außer man hat einen extrem akkurate Taktgeber.

Akkurate Taktgeber würden bei diesem Problem helfen, aber sie sind eine teure Lösung für Allerweltsgeräte. Schließlich messen wir die Bits zeitlich auf Verbindungen, die mit vielen Megabit/s laufen, daher dürfte der Taktgeber nur weniger als den Bruchteil einer Mikrosekunde über der längsten erlaubten Reihe abweichen. Dies könnte für langsame oder kurze Verbindungen noch sinnvoll sein, doch es ist keine generelle Lösung.

Eine Strategie wäre, dem Empfänger ein separates Taktignal zu senden. Eine weitere Taktleitung ist für Computerbusse oder kurze Kabel, in denen es viele parallele Leitungen gibt, keine große Sache, aber bei den meisten Netzverbindungen wäre das Verschwendungen: wenn wir eine zusätzliche Leitung haben, um ein Signal zu senden, dann könnten wir diese Leitung auch nutzen, um Daten zu übertragen. Ein raffinierter Trick hier ist, das Taktignal mit dem Datensignal durch eine XOR-Verknüpfung zu mischen, sodass keine zusätzliche Leitung benötigt wird. Die Ergebnisse sind in ▶ Abbildung 2.20d zu sehen. Der Taktgeber macht in jeder Bitzeit einen Taktübergang, er läuft also mit doppelter Bitrate. Wenn das Taktignal mit dem 0-Pegel XOR-verknüpft ist, so wird ein Übergang von niedrig zu hoch (steigende Flanke) durchgeführt, welcher einfach dem Taktignal entspricht. Dieser Übergang entspricht einer logischen 0. Wenn es mit dem 1-Pegel XOR-verknüpft ist, dann ist das Signal invertiert und macht einen Übergang von hoch zu niedrig (fallende Flanke). Dieser Übergang entspricht einer logischen 1. Das Schema heißt **Manchester**-Code und wurde im klassischen Ethernet eingesetzt.

Der Nachteil des Manchester-Codes ist, dass aufgrund des Taktsignals doppelt so viel Bandbreite wie bei NRZ benötigt wird, und wir haben bereits gesehen, dass Bandbreite oft eine wichtige Rolle spielt. Eine andere Strategie basiert auf der Idee, die Daten zu codieren um sicherzustellen, dass das Signal genügend Übergänge aufweist. Schließlich hat NRZ nur für lange Reihen von Nullen und Einsen Taktrückgewin-

nungsprobleme. Falls es häufige Übergänge gibt, wird es für den Empfänger einfach, mit dem ankommenden Symbolstrom synchronisiert zu bleiben.

Als Schritt in die richtige Richtung können wir die Situation vereinfachen, indem wir eine 1 als einen Übergang und eine 0 als keinen Übergang (oder umgekehrt) codieren. Diese Codierung wird **NRZI (Non-Return-to-Zero Inverted)** genannt, eine Variation von NRZ. Ein Beispiel ist in ► Abbildung 2.20c gezeigt. Der beliebte **USB**-Standard (*Universal Serial Bus*) für die Verbindung von Rechnerperipheriegeräten benutzt NRZI. Somit gibt es kein Problem mit langen Reihen von Einsen.

Nun haben wir natürlich noch das Problem der langen Reihen von Nullen zu lösen. Wenn wir eine Telefongesellschaft wären, könnten wir einfach fordern, dass der Sender nicht zu viele Nullen übermittelt. Ältere digitale Telefonleitungen in den USA, sogenannte **T1-Leitungen**, verlangten in der Tat, dass nicht mehr als 15 hintereinanderfolgende Nullen gesendet wurden, damit die Leitungen korrekt funktionierten. Um dieses Problem völlig zu beheben, können wir eine Folge von Nullen aufbrechen, indem wir eine Abbildung definieren, die kleine Gruppen der zu sendenden Bits jeweils anderen, leicht längeren Gruppen zuweist, welche nicht zu viele aufeinanderfolgende Nullen haben.

Ein bekannter Code hierfür ist **4B/5B**. Jeweils 4 Bits werden auf ein 5-Bit-Muster nach einer festgelegten Übersetzungstabelle abgebildet. Die 5-Bit-Muster sind so ausgewählt, dass niemals eine Folge von mehr als drei aufeinanderfolgenden Nullen vorkommt. Die Übersetzungstabelle ist in ► Abbildung 2.21 gezeigt. Dieses Schema fügt 25 % Overhead hinzu – was besser ist als 100 % Overhead beim Manchester-Code. Da es 16 Eingabekombinationen und 32 Ausgabekombinationen gibt, werden einige der Ausgabekombinationen nicht verwendet. Nimmt man die Kombinationen mit zu vielen Nullen heraus, dann sind immer noch einige Codes übrig. Als Bonus können wir diese Nicht-Daten-Codes benutzen, um Steuersignale der Bitübertragungsschicht zu repräsentieren. Zum Beispiel stellt in einigen Verwendungen „11111“ eine unbeschäftigte Leitung dar und „11000“ repräsentiert den Beginn eines Rahmens.

| Daten (4B) | Codewort (5B) | Daten (4B) | Codewort (5B) |
|------------|---------------|------------|---------------|
| 0000 | 11110 | 1000 | 10010 |
| 0001 | 01001 | 1001 | 10011 |
| 0010 | 10100 | 1010 | 10110 |
| 0011 | 10101 | 1011 | 10111 |
| 0100 | 01010 | 1100 | 11010 |
| 0101 | 01011 | 1101 | 11011 |
| 0110 | 01110 | 1110 | 11100 |
| 0111 | 01111 | 1111 | 11101 |

Abbildung 2.21: 4B/5B-Abbildung.

Ein alternativer Ansatz sieht vor, die Daten zufällig erzeugt aussehen zu lassen, auch bekannt als Scrambling. In diesem Fall ist es sehr wahrscheinlich, dass es häufige Übergänge geben wird. Ein **Scrambler** führt vor der Übertragung eine XOR-Verknüpfung der Daten mit einer pseudozufälligen Folge aus. Durch dieses Mischen werden die Daten so zufällig wie die pseudozufällige Folge selbst (vorausgesetzt, dass sie von der pseudozufälligen Folge unabhängig sind). Der Empfänger nimmt dann eine XOR-Verknüpfung der ankommenden Bits mit derselben pseudozufälligen Folge vor, um die ursprünglichen Daten wiederherzustellen. Damit dieses Verfahren praktikabel bleibt, muss die pseudozufällige Folge leicht zu erzeugen sein. Sie wird in der Regel von einem einfachen Zufallszahlengenerator ausgegeben.

Scrambling ist attraktiv, weil keine zusätzliche Bandbreite oder Zeit erforderlich ist. Tatsächlich ist es häufig hilfreich, das Signal aufzubereiten, sodass es seine Energie nicht in dominanten Frequenzkomponenten (verursacht durch sich wiederholende Datenmuster) hat, die elektromagnetische Interferenzen ausstrahlen können. Scrambling kann hier helfen, da zufällige Signale tendenziell „weiß“ sind oder ihre Energie über die Frequenzkomponenten verstreuhen.

Scrambling garantiert jedoch nicht, dass es keine langen Reihen geben wird. Es ist auch hier möglich, gelegentlich Pech zu haben, und zwar wenn die Daten dieselben sind wie die pseudozufällige Folge. In diesem Fall bewirkt die XOR-Verknüpfung, dass eine Reihe von Nullen entsteht. Dieses Ergebnis tritt nicht allgemein bei einer langen pseudozufälligen Folge auf, welche schwierig vorherzusagen ist. Bei kurzen oder vorhersehbaren Folgen kann es jedoch für bösartige Benutzer einfach sein, Bitmuster zu senden, die nach dem Scrambling eine lange Reihen von Nullen erzeugen und somit Verbindungsfehler verursachen. Frühe Versionen des Standards zum Senden von IP-Paketen über SONET-Verbindungen im Telefonsystem besaßen diesen Defekt (Malis und Simpson, 1999). Damit konnten Benutzer bestimmte „Killerpakete“ senden, die garantiert Probleme verursachten.

Symmetrische Signale

Signale, die sogar innerhalb kurzer Zeitintervalle genauso viel positive wie negative Spannung haben, heißen **symmetrische Signale** (*balanced signal*). Ihr Mittelwert beträgt null, was bedeutet, dass diese Signale keine elektrische Gleichstromkomponente haben. Dieses Fehlen ist ein Vorteil, weil einige Kanäle wie Koaxialkabel oder Leitungen mit Transformatoren eine Gleichstromkomponente aufgrund ihrer physischen Eigenschaften stark dämpfen. Eine weitere Methode, den Empfänger mit dem Kanal zu verbinden, ist die **kapazitive Kopplung** (*capacitive coupling*), bei der nur der Wechselstromanteil eines Signals weitergeleitet wird. In beiden Fällen verschwenden wir Energie beim Senden eines Signals, dessen Mittelwert nicht null ist, da die Gleichstromkomponente herausgefiltert wird.

Symmetrische Signalübertragung stellt Übergänge für Taktrückgewinnung zur Verfügung, da es hier eine Mischung aus positiven und negativen Spannungen gibt. Außerdem können Empfänger auf eine einfache Art kalibriert werden, weil der Mittelwert des Signals gemessen und als eine Entscheidungsschwelle zum Decodieren von Symbolen

genutzt werden kann. Bei asymmetrischen Signalen kann der Mittelwert zum Beispiel aufgrund von vielen Einsen von der echten Entscheidungsebene abweichen. Dies könnte beispielsweise durch viele Einsen verursacht werden, was das Decodieren von mehr Symbolen mit Fehlern nach sich ziehen würde.

Eine direkte Möglichkeit, einen symmetrischen Code zu konstruieren, ist die Verwendung von zwei Spannungsebenen, um eine logische Eins darzustellen (z.B. +1 V oder -1 V). Zur Repräsentation der logischen Null wird dann 0 V benutzt. Um eine 1 zu senden, wechselt der Sender zwischen der +1-V- und der -1-V-Ebene, sodass immer ein Ausgleich stattfindet. Dieses Schema heißt **bipolare Codierung** (*bipolar encoding*). Im Bereich von Telefonnetzen hat das Verfahren den Namen **AMI** (*Alternate Mark Inversion*), welcher von einer veralteten Terminologie herröhrt, in der eine 1 als „mark“ (Zeichen) und eine 0 als „space“ (Zwischenraum) bezeichnet wurde. Ein Beispiel ist in ► Abbildung 2.20e zu sehen.

Bipolare Codierung fügt ein Spannungs niveau hinzu, um Symmetrie zu erhalten. Alternativ können wir dazu eine Abbildung wie 4B/5B benutzen (wodurch außerdem Übergänge zur Taktrückgewinnung gewonnen werden). Ein Beispiel dieser Art von symmetrischem Code ist der **8B/10B**-Leitungscode. Hierbei werden 8 Eingabebits auf 10 Ausgabebits abgebildet, damit hat dieser Code genau wie der 4B/5B-Leitungscode eine Effizienz von 80 %. Die 8 Bits werden in zwei Gruppen von 5 Bits und von 3 Bits aufgeteilt, wobei die 5-Bit-Gruppe auf 6 Bits und die 3-Bit-Gruppe auf 4 Bits abgebildet wird. Die 6-Bit- und 4-Bit-Symbole werden anschließend aneinandergehängt. In jeder Gruppe können einige Eingabemuster auf symmetrische Ausgabemuster abgebildet werden, die dieselbe Anzahl von Nullen und Einsen besitzen. Beispielsweise wird „001“ auf „1001“ abgebildet, was symmetrisch ist. Doch es gibt nicht genügend Kombinationen, damit alle Ausgabemuster symmetrisch sind. In diesen Fällen wird jedes Eingabemuster auf zwei Ausgabemuster abgebildet. Das erste Muster bekommt eine zusätzliche 1, das zweite eine zusätzliche 0. Beispielsweise wird „000“ sowohl auf „1011“ als auch auf das Komplement „0100“ abgebildet. Wenn Eingabebits auf Ausgabebits abgebildet werden, erinnert sich der Codierer an die **Disparität** (*disparity*) des vorhergehenden Symbols. Die Disparität ist die Gesamtzahl der Nullen oder Einsen, bei denen das Signal seine Symmetrie verliert. Der Codierer wählt dann entweder ein Ausgabemuster oder das Komplement, um die Disparität zu verringern. Bei 8B/10B wird die Disparität höchstens 2 Bits betragen. Somit ist das Signal niemals weit von der Symmetrie entfernt. Es gibt auch nie mehr als fünf aufeinanderfolgende Einsen oder Nullen, was die Taktrückgewinnung unterstützt.

2.5.2 Übertragung im Durchlassbereich

Häufig möchten wir zum Senden über einen Kanal einen Frequenzbereich benutzen, der nicht bei Null beginnt. Bei drahtlosen Kanälen ist es nicht praktikabel, sehr niederfrequente Signale zu senden, weil die Größe der Antenne ein Bruchteil der Signalwellenlänge sein muss, welche groß werden kann. Auf jeden Fall wird die Wahl der Frequenzen gewöhnlich durch regulatorische Beschränkungen und die Notwendigkeit diktiert, Inter-

ferenzen zu vermeiden. Selbst bei Kabeln kann es sinnvoll sein, ein Signal in einem gegebenen Frequenzband so zu platzieren, dass verschiedene Arten von Signalen auf dem Kanal vorhanden sein können. Diese Art der Übertragung heißt Durchlassbandübertragung, da ein beliebiges Frequenzband benutzt wird, um das Signal weiterzuleiten.

Glücklicherweise beziehen sich unsere grundlegenden Ergebnisse am Anfang dieses Kapitels allesamt auf die Bandbreite oder die Breite des Frequenzbandes. Die absoluten Frequenzwerte spielen für die Kapazität keine Rolle. Dies bedeutet, wir können ein **Basisbandsignal** nehmen, das die Frequenzen von 0 bis B Hz besetzt, und es in einen sogenannten **Durchlassbereich** (*passband*) von S bis $S+B$ Hz verschieben. Dabei bleibt die übertragene Informationsmenge unverändert, auch wenn das Signal anders aussieht. Um das Signal beim Empfänger zu verarbeiten, kann es wieder zurück zum Basisband verschoben werden, wo die Erkennung von Symbolen günstiger ist.

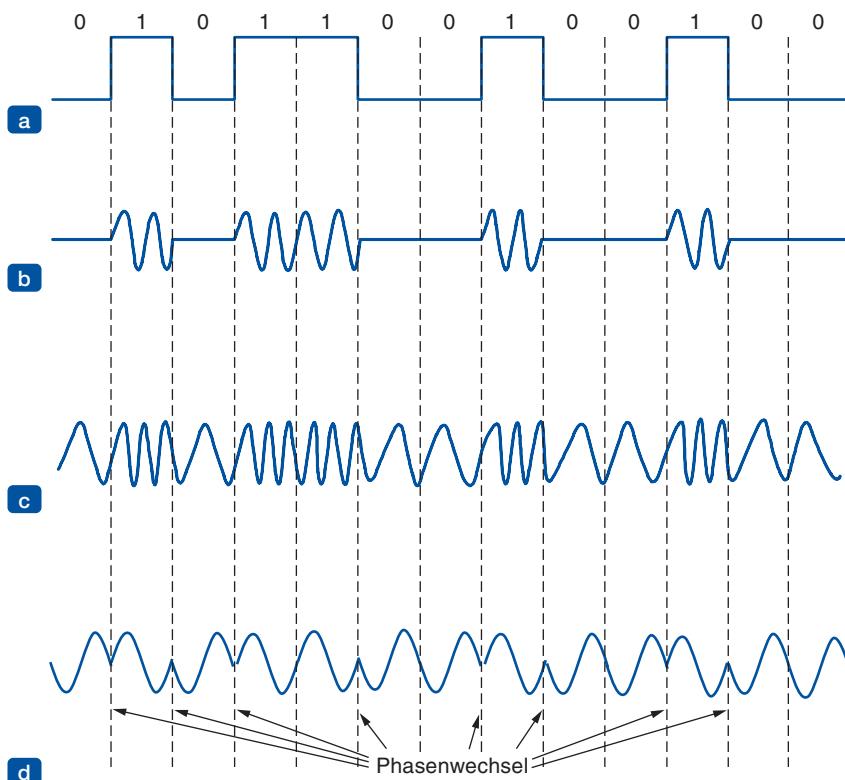


Abbildung 2.22: (a) Binärsignal. (b) Amplitudenmodulation. (c) Frequenzmodulation. (d) Phasenmodulation.

Digitale Modulation wird bei der Übertragung im Durchlassbereich durchgeführt, indem ein Trägersignal im Durchlassband reguliert oder moduliert wird. Wir können die Amplitude, Frequenz oder Phase des Trägersignals modulieren. Jedes dieser Methoden hat einen eigenen Namen. Bei der **Amplitudenmodulation (ASK, Amplitude Shift Keying)** werden zwei unterschiedliche Amplituden benutzt, um 0 und 1 zu repräsentieren. Ein Beispiel dafür ist in ►Abbildung 2.22b gezeigt. Wenn mehr Sym-

bole dargestellt werden sollen, können weitere Spannungsniveaus eingesetzt werden. Ähnlich werden bei der **Frequenzmodulation (FSK, Frequency Shift Keying)** zwei oder mehr unterschiedliche Tonhöhen verwendet. Das Beispiel in ►Abbildung 2.22c benutzt nur zwei Frequenzen. In der einfachsten Form der **Phasenmodulation (PSK, Phase Shift Keying)** wird die Trägerwelle bei jeder Symbolperiode systematisch um 0 oder 180 Grad verschoben. Da es zwei Phasen gibt, wird dieses Verfahren auch als **BPSK (Binary Phase Shift Keying)** bezeichnet. „Binary“ hier bezieht sich auf die zwei Symbole und nicht darauf, dass die Symbole zwei Bits repräsentieren. Ein Beispiel ist in Abbildung 2.22c zu sehen. Ein besseres Schema zur effizienteren Ausnutzung der Kanalbandbreite ist die Verwendung von vier Verschiebungen, z.B. 45, 135, 225 oder 315 Grad, um 2 Bits an Information per Symbol zu übertragen. Diese Version heißt **QPSK (Quadrature Phase Shift Keying)**.

Wir können diese Schemata kombinieren und mehr Spannungslagen benutzen, um mehr Bits pro Symbol zu übertragen. Aufgrund ihrer Beziehung – Frequenz ist die Rate der Phasenveränderung über der Zeit – kann zu einem Zeitpunkt immer nur entweder die Frequenz oder die Phase moduliert werden. In der Regel werden Amplitude und Phase zusammen moduliert. In ►Abbildung 2.23 sind drei Beispiele dafür zu sehen. In jedem Beispiel geben die Punkte die erlaubten Amplituden- und Phasenkombinationen jedes Symbols an. In Abbildung 2.23a haben die Punkte den gleichen Abstand voneinander und befinden sich auf 45, 135, 225 bzw. 315 Grad. Die Phase eines Punkts wird durch den Winkel angegeben, den eine Linie von diesem Punkt zum Koordinatenursprung mit der positiven x-Achse bildet. Die Amplitude eines Punkts ist der Abstand vom Ursprung. Dieses Bild ist eine Darstellung von QPSK.

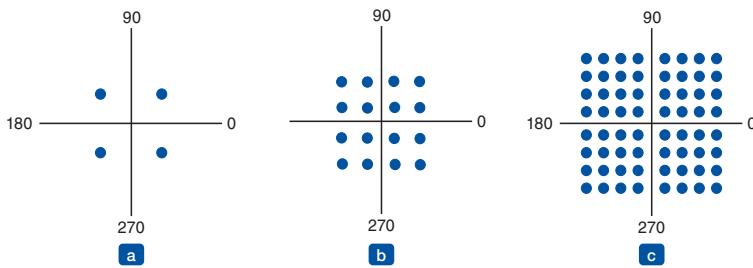


Abbildung 2.23: (a) QPSK. (b) QAM-16. (c) QAM-64.

Diese Art von Diagramm heißt **Konstellationsdiagramm**. In Abbildung 2.23b sehen wir ein Modulationsschema mit einer dichtenen Konstellation. Sechzehn Kombinationen von Amplituden und Phasen werden benutzt, sodass das Modulationsschema verwendet werden kann, um 4 Bits pro Symbol zu übermitteln. Es wird **QAM-16** genannt, wobei QAM für **Quadraturamplitudenmodulation (Quadrature Amplitude Modulation)** steht. Abbildung 2.23c ist ein noch dichteres Modulationsschema mit 64 unterschiedlichen Kombinationen, sodass 6 Bits pro Symbol übermittelt werden können. Es heißt **QAM-64**. Es werden sogar auch QAMs von noch höherer Ordnung eingesetzt. Wie Sie vielleicht anhand dieser Konstellationen vermuten können, ist es einfacher, elektronische Geräte so zu bauen, dass sie Symbole als eine Kombination von Werten auf jeder

Achse erzeugen, anstatt als Kombination von Amplituden- und Phasenwerten. Daher sehen die Muster eher wie Quadrate als wie konzentrische Kreise aus.

Die Konstellationen, die wir bisher gesehen haben, zeigen nicht, wie die Bits den Symbolen zugewiesen werden. Eine wichtige Überlegung dabei ist, dass eine kleine Anhäufung von Rauschen beim Empfänger nicht in zu vielen Bitfehlern resultieren sollte. Dies könnte passieren, wenn wir aufeinanderfolgenden Bitwerten benachbarte Symbole zuweisen. Wenn bei QAM-16 zum Beispiel ein Symbol für 0111 und das benachbarte Symbol für 1000 steht und der Empfänger versehentlich das benachbarte Symbol auswählt, dann werden als Folge davon alle Bits falsch sein. Eine bessere Lösung ist, Bits so auf Symbole abzubilden, dass benachbarte Symbole sich in nur einer Bitposition unterscheiden. Diese Abbildung heißt **Gray-Code**. ►Abbildung 2.24 zeigt eine QAM-16-Konstellation, die Gray-codiert ist. Wenn nun der Empfänger das Symbol fehlerhaft decodiert, so gibt es nur einen einzigen Bitfehler in dem erwarteten Fall, dass das decodierte Symbol nahe an dem übermittelten Symbol ist.

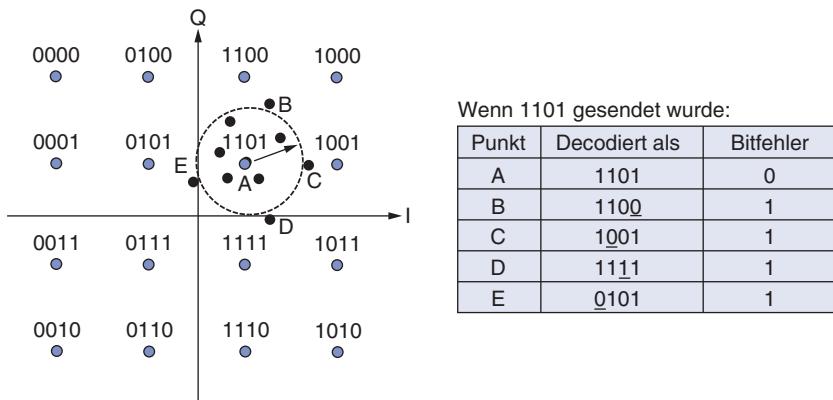


Abbildung 2.24: Gray-codiertes QAM-16.

2.5.3 Frequenzmultiplexverfahren

Die bisher vorgestellten Modulationsschemata erlauben das Senden eines Signals, um Bits über eine verkabelte oder drahtlose Verbindung zu transportieren. Skaleneffekte spielen eine große Rolle dabei, wie wir Netzwerke benutzen. Eine Übertragungsleitung mit hoher statt mit niedriger Bandbreite zwischen zwei Büros zu installieren und zu pflegen, kostet im Wesentlichen dasselbe (d.h., die Kosten kommen vom Ausheben des Grabens und nicht davon, welche Art von Kabel oder Glasfaser dort hinein kommt). Folglich wurden Multiplex-Schemata entwickelt, um eine gemeinsame Nutzung einer Leitung durch viele Signale zu ermöglichen.

Frequenzmultiplexing (FDM, Frequency Division Multiplexing) verwendet die Übertragung im Durchlassbereich zur gemeinsamen Nutzung eines Kanals. Das Spektrum wird in Frequenzbänder aufgeteilt und jeder Benutzer erhält exklusive Besitzrechte auf ein bestimmtes Band zum Senden. Die Übertragung von Mittelwellenradio ist ein Beispiel für FDM. Das zugeteilte Spektrum umfasst ungefähr 1 MHz, ca. 500 bis

1 500 kHz. Jedem logischen Kanal (Station) wird ein Frequenzbereich zugeordnet, in dem er ausgeführt werden kann, wobei der Abstand zwischen den Kanälen groß genug sein muss, um Interferenzen zu verhindern.

Als ausführlicheres Beispiel sind in ▶ Abbildung 2.25 drei Telefonkanäle mit Sprachqualität dargestellt, die mithilfe von FDM mehrfach genutzt werden. Filter begrenzen die nutzbare Bandbreite auf etwa 3 100 Hz pro Sprachkanal. Werden viele Kanäle mehrfach genutzt, erhält jeder Kanal 4 000 Hz. Der überschüssige Frequenzbereich wird **Sicherheitsband** (*guard band*) genannt. Dadurch wird die Trennung der Kanäle gewährleistet. Zuerst werden die Sprachkanäle in der Frequenz angehoben, wobei die Höhe der Anhebung für jeden Sprachkanal anders aussieht. Dann werden sie zusammengeführt, weil jetzt nie zwei Kanäle den gleichen Frequenzbereich nutzen. Obwohl sich dank der Sicherheitsbänder Lücken zwischen den Kanälen befinden, überlappen sich benachbarte Kanäle teilweise. Diese Überlappung kommt daher, weil reale Filter keine idealen scharfen Grenzen haben. Das bedeutet, dass eine starke Signalspitze am Rand eines Kanals im benachbarten Kanal ein nicht thermisches Rauschen verursachen kann.

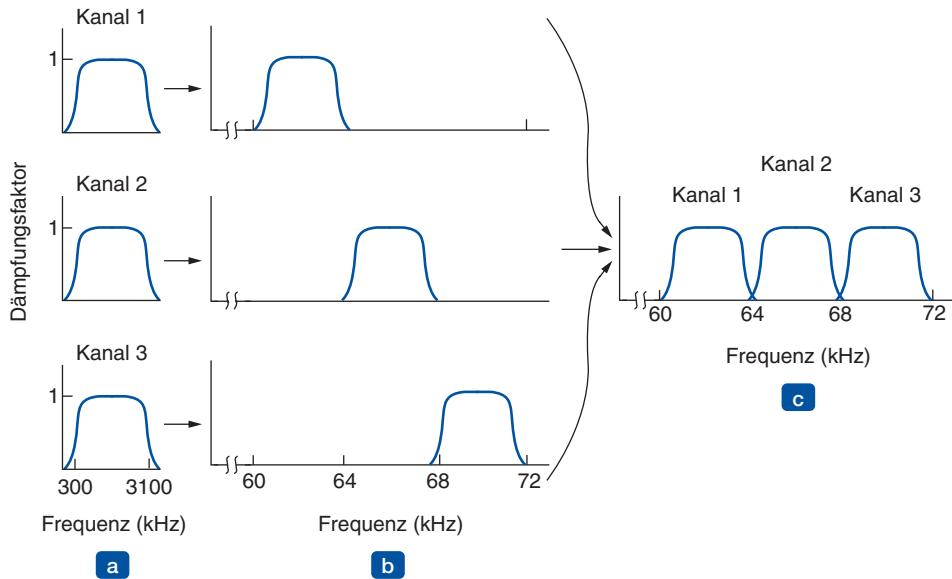


Abbildung 2.25: Frequenzmultiplexing. (a) Ursprüngliche Bandbreite. (b) Bandbreiten mit erhöhter Frequenz. (c) Multiplexkanal.

Dieses Schema wurde über viele Jahre im Telefonsystem eingesetzt, doch heute werden Zeitmultiplexverfahren bevorzugt. FDM wird jedoch weiterhin in Telefonnetzen sowie in zellulären, terrestrisch-drahtlosen und Satellitennetzen auf einer höheren Detailebene verwendet.

Wenn digitale Daten gesendet werden, ist es möglich, das Spektrum effizient ohne Benutzung von Sicherheitsbändern aufzuteilen. Bei **orthogonalem Frequenzmultiplexing (OFDM, Orthogonal Frequency Division Multiplexing)** wird die Kanalbandbreite in viele Unterträger eingeteilt, die unabhängig Daten senden (z.B. mit QAM). Die Unterträger

werden eng im Frequenzbereich zusammengepackt. Somit reichen die Signale von einem Unterträger zu dem benachbarten Träger. Wie man allerdings in ▶ Abbildung 2.26 sieht, ist die Frequenzantwort jedes Unterträgers so konzipiert, dass diese in der Mitte der benachbarten Unterträger null ist. Die Unterträger können daher an ihren Mittelfrequenzen ohne Interferenzen von ihren Nachbarn abgetastet werden. Dazu wird eine Sicherheitszeit benötigt, sodass ein Teil des Symbolsignals rechtzeitig wiederholt werden kann, um die gewünschte Frequenzantwort zu erhalten. Dieser Mehraufwand ist jedoch sehr viel geringer als mit vielen Sicherheitsbändern.

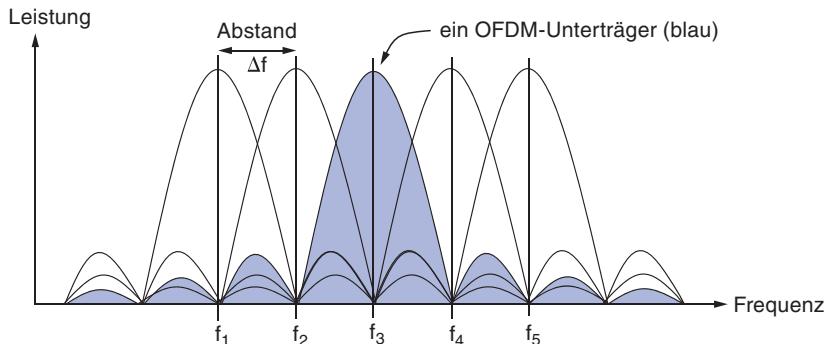


Abbildung 2.26: Orthogonales Frequenzmultiplexing (OFDM).

Die Idee von OFDM gibt es schon eine ganze Weile, doch erst im letzten Jahrzehnt hat es sich weithin durchgesetzt – im Zuge der Erkenntnis, dass es möglich ist, OFDM effizient hinsichtlich einer Fouriertransformation von digitalen Daten über allen Unterträgern zu implementieren (anstatt jeden Unterträger separat zu modulieren). OFDM wird bei IEEE 802.11, in Kabelnetzen und Trägerfrequenzanlagen verwendet und ist für zelluläre Systeme der vierten Generation in Planung. In der Regel wird ein einzelner digitaler Informationsstrom mit hoher Datenrate in viele Ströme mit niedriger Datenrate aufgespalten, die auf den Unterträgern parallel übermittelt werden. Diese Aufteilung ist sinnvoll, weil Qualitätsverluste des Kanals auf der Unterträgerebene einfacher zu behandeln sind; einige Unterträger können sehr hohe Qualitätsverluste aufweisen, diese werden dann zugunsten von Unterträgern ausgesondert, auf denen der Empfang gut ist.

2.5.4 Zeitmultiplexverfahren

Eine Alternative zu FDM ist **Zeitmultiplexing (TDM, Time Division Multiplexing)**. Hier wechseln sich die Benutzer (in Round-Robin-Manier) ab, jeder bekommt periodisch die gesamte Bandbreite für eine kurze Zeit. In ▶ Abbildung 2.27 sind drei Ströme dargestellt, die mittels TDM mehrfach genutzt werden. Die Bits aus den Eingabeströmen werden in eine feste **Zeitscheibe** (*time slot*) aufgenommen und in den aggregierten Strom ausgegeben. Dieser Strom läuft mit der Summenrate der individuellen Ströme. Dazu müssen die Ströme zeitlich synchronisiert werden. Analog zum Sicherheitsfrequenzband können kurze Intervalle von **Sicherheitszeiten** (*guard time*) hinzugefügt werden, um kleine zeitliche Variationen auszugleichen.

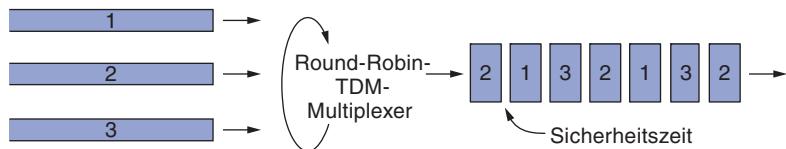


Abbildung 2.27: Zeitmultiplexing (TDM).

TDM wird weitverbreitet innerhalb des Telefon- und des zellulären Netzes eingesetzt. Um Verwirrung zu vermeiden, wollen wir klarstellen, dass TDM sich grundsätzlich vom alternativen **statistischen Zeitmultiplexing (STDM)**, *Statistical Time Division Multiplexing*) unterscheidet. Das Attribut „statistisch“ deutet darauf hin, dass die einzelnen Ströme des Multiplexstroms *nicht* nach einem festgelegten Plan, sondern aufgrund der Statistik ihrer Nachfrage hinzukommen. STDM ist Paketvermittlung unter einem anderen Namen.

2.5.5 Codemultiplexverfahren

Es gibt noch eine dritte Variante des Multiplexing, die völlig anders funktioniert als FDM und TDM. **Codemultiplexing (CDM)**, *Code Division Multiplexing*) ist eine Form der **Frequenzspreizungskommunikation**, in der ein Schmalbandsignal über ein weiteres Frequenzband gestreut wird. Dadurch kann das Verfahren toleranter gegenüber Interferenzen sein und es außerdem ermöglichen, dass mehrerer Signale von unterschiedlichen Benutzern dasselbe Frequenzband gemeinsam nutzen. Da Codemultiplexing hauptsächlich für den letztgenannten Zweck eingesetzt wird, nennt man es häufig **CDMA** (*Code Division Multiple Access*).

CDMA erlaubt es jeder Station, die ganze Zeit über das gesamte Frequenzspektrum zu übermitteln. Mehrere gleichzeitige Übertragungen werden mithilfe der Codierungstheorie voneinander getrennt. Bevor wir uns den Algorithmus ansehen, hier eine Analogie: eine Abflughalle mit vielen Personenpaaren, die sich unterhalten. TDM ist mit Personenpaaren in diesem Raum vergleichbar, die nacheinander sprechen. FDM ist mit Personenpaaren vergleichbar, die auf verschiedenen Höhenpositionen miteinander sprechen, einige stehen höher und andere tiefer, sodass jedes Paar zwar gleichzeitig, aber unabhängig von den anderen Paaren sein eigenes Gespräch führen kann. CDMA entspricht der Situation, dass jedes Personenpaar gleichzeitig spricht, aber jeweils in einer anderen Sprache. Das französische Paar rattert in Französisch los und empfindet alles Nicht-Französische lediglich als Geräuschkulisse. Der Kern von CDMA ist also die Möglichkeit, das gewünschte Signal herauszufiltern und alles andere als zufälliges Rauschen zu werten. Im Folgenden nun eine etwas vereinfachte Beschreibung von CDMA.

In CDMA wird jede Bitübertragungszeit in m kurze Intervalle namens **Chips** unterteilt. Normalerweise gibt es 64 oder 128 Chips pro Bit, aber im folgenden Beispiel werden der Einfachheit halber nur 8 Chips pro Bit verwendet. Jeder Station wird ein eindeutiger m -Bit-Code, der als **Chipfolge** bezeichnet wird, zugewiesen. Aus pädagogischen Gründen ist die Verwendung einer bipolaren Notation einfacher, wobei diese Codes als Folgen von -1 und $+1$ zu schreiben. Chipfolgen werden in Klammern aufgeführt.

Um ein 1-Bit zu übertragen, sendet die Station ihre Chipfolge. Um ein 0-Bit zu übertragen, sendet sie die Negation ihrer Chipfolge. Kein anderes Muster ist zulässig. Für $m=8$ gilt: Wenn der Station A die Chipfolge $(-1 -1 -1 +1 +1 -1 +1 +1)$ zugewiesen wurde, dann kann sie ein 1-Bit durch die Übertragung der Chipfolge und ein 0-Bit durch die Übertragung von $(+1 +1 +1 -1 -1 +1 -1 -1)$ senden. Es werden tatsächlich Signale dieser Spannungen gesendet, doch für uns ist es ausreichend, sich Folgen vorzustellen.

Die Erhöhung der zu sendenden Informationsmenge für jede Station von b Bit/s auf mb Chip/s bedeutet, dass muss die für CDMA benötigte Bandbreite um den Faktor m größer ist als die Bandbreite, die für eine Station benötigt wird, die CDMA nicht benutzt (unter der Annahme, dass keine Änderungen in der Modulation oder der Codiertechnik vorgenommen werden). Ist ein 1-MHz-Band für 100 Stationen verfügbar, hätte jede Station bei FDM 10 kHz und könnte mit 10 kbit/s senden (bei einem Bit pro Hz). Bei CDMA nutzt jede Station 1 MHz voll, sodass die Chiprate 100 Chip pro Bit beträgt, um die Bitrate der Station von 10 kbit/s über den Kanal zu verstreuen.

In ►Abbildung 2.28a und b sind die Chipfolgen zu sehen, die den vier Beispielstationen zugewiesen sind, sowie die Signale, die sie repräsentieren. Jede Station hat ihre eindeutige Chipfolge. Wir verwenden das Symbol S zur Bezeichnung des m -Chipvektors für Station S und \bar{S} für seine Negation. Alle Chipfolgen sind paarweise **orthogonal**, was bedeutet, dass das normalisierte innere Produkt von zwei beliebigen verschiedenen Chipfolgen S und T (geschrieben als $S \cdot T = 0$) 0 ist. Solche orthogonalen Chipfolgen kann man mit einer Methode namens **Walsh-Code** erzeugen. Mathematisch kann die Orthogonalität der Chipfolgen wie folgt ausgedrückt werden:

$$S \cdot T \equiv \frac{1}{m} \sum_{i=1}^m S_i T_i = 0 \quad (2.5)$$

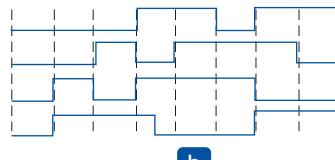
In einfachem Deutsch: Es sind genauso viele Paare gleich wie verschieden. Diese Orthogonalitätseigenschaft erweist sich später noch als sehr wichtig. Beachten Sie, dass, wenn $S \cdot T = 0$, dann gilt auch $S \cdot \bar{T} = 0$. Das normalisierte innere Produkt einer Chipfolge mit sich selbst ist 1:

$$S \cdot S = \frac{1}{m} \sum_{i=1}^m S_i S_i = \frac{1}{m} \sum_{i=1}^m S_i^2 = \frac{1}{m} \sum_{i=1}^m (\pm 1)^2 = 1$$

Der Grund liegt darin, dass alle m Terme im inneren Produkt 1 sind, also ist die Summe m . Beachten Sie auch, dass $S \cdot \bar{S} = -1$ ist.

$$\begin{aligned}
 A &= (-1 -1 -1 +1 +1 -1 +1 +1) \\
 B &= (-1 -1 +1 -1 +1 +1 +1 -1) \\
 C &= (-1 +1 -1 +1 +1 +1 -1 -1) \\
 D &= (-1 +1 -1 -1 -1 -1 +1 -1)
 \end{aligned}$$

a

**c**

$$\begin{aligned}
 S_1 = C &= (-1 +1 -1 +1 +1 +1 -1 -1) \\
 S_2 = B+C &= (-2 \ 0 \ 0 \ 0 +2 +2 \ 0 -2) \\
 S_3 = A+\bar{B} &= (\ 0 \ 0 -2 +2 \ 0 -2 \ 0 +2) \\
 S_4 = A+\bar{B}+C &= (-1 +1 -3 +3 +1 -1 -1 +1) \\
 S_5 = A+B+C+D &= (-4 \ 0 -2 \ 0 +2 \ 0 +2 -2) \\
 S_6 = A+B+\bar{C}+D &= (-2 -2 \ 0 -2 \ 0 -2 +4 \ 0)
 \end{aligned}$$

d

Abbildung 2.28: (a) Chipfolgen für vier Stationen. (b) Signale, welche die Folgen repräsentieren.
(c) Sechs Beispiele für Übertragungen. (d) Wiederherstellung des Signals von Station C.

Während der Übertragungszeit für ein Bit kann eine Station eine 1 übertragen (indem sie ihre Chipfolge sendet) oder sie kann eine 0 übertragen (indem sie das Negative ihrer Chipfolge sendet). Sie kann aber auch schweigen und nichts übertragen. Für den Moment nehmen wir an, dass alle Stationen zeitlich synchronisiert sind; deshalb beginnen alle Chipfolgen im gleichen Augenblick. Übertragen zwei oder mehr Stationen gleichzeitig, addieren sich ihre bipolaren Folgen linear. Wenn beispielsweise in einer Chipperiode drei Stationen +1 ausgeben und eine Station -1 ausgibt, dann wird +2 empfangen. Man kann sich das so vorstellen, dass Signale als Spannungen zu dem Kanal addiert werden: Drei Stationen geben +1 Volt und eine gibt -1 Volt aus, sodass 2 Volt empfangen wird. In ► Abbildung 2.28c sehen wir sechs Beispiele der gleichzeitigen Übertragung von einem 1-Bit durch eine oder mehrere Stationen. Im ersten Beispiel überträgt C ein 1-Bit, sodass wir hier nur die Chipfolge von C erhalten. Im zweiten Beispiel übertragen B und C 1-Bits, sodass wir hier die Summe ihrer bipolaren Chipfolgen erhalten, konkret:

$$(-1 -1 +1 -1 +1 +1 -1) + (-1 +1 -1 +1 +1 +1 -1) = (-2 \ 0 \ 0 \ 0 +2 +2 \ 0 -2)$$

Um den Bitstrom einer einzelnen Station wiederherzustellen, muss der Empfänger die Chipfolgen der betreffenden Station im Voraus kennen. Er führt die Wiederherstellung durch Berechnung des normalisierten inneren Produkts der empfangenen Chipfolge und der Chipfolge der Station durch, deren Bitstrom wiederherzustellen ist. Ist die erhaltene Chipfolge **c** und der Empfänger versucht eine Station abzuhören, deren Chipfolge C ist, so berechnet er nur das normalisierte innere Produkt **S•C**.

Um dies zu verstehen, stellen Sie sich zwei Stationen A und C vor, die beide ein 1-Bit zu dem Zeitpunkt übertragen, an dem B ein 0-Bit überträgt. Der Empfänger sieht die Summe $S = A+\bar{B}+C$ und rechnet wie folgt:

$$S \cdot C = (A+\bar{B}+C) \cdot C = A \cdot C + \bar{B} \cdot C + C \cdot C = 0+0+1=1$$

Die ersten zwei Terme verschwinden, weil alle Paare von Chipfolgen sorgfältig so ausgewählt wurden, dass sie alle orthogonal sind (Gleichung (2.5)). Nun dürfte klar sein, warum diese Eigenschaft den Chipfolgen auferlegt werden muss.

Um den Decodierprozess konkreter darzustellen, zeigen wir sechs Beispiele in ▶ Abbildung 2.28d. Nehmen wir an, der Empfänger ist daran interessiert, das von Station C gesendete Bit aus den sechs Summen S_1 bis S_6 zu extrahieren. Er berechnet das Bit durch Summieren der paarweisen Produkte des empfangenen S- und C-Vektors von ▶ Abbildung 2.28a. Dann nimmt er 1/8 des Ergebnisses (da hier $m=8$ ist). Die Beispiele umfassen Fälle, in denen C schweigt, ein 1-Bit sendet und ein 0-Bit sendet, einzeln und in Kombination mit anderen Übertragungen. Wie man sieht, wird jedes Mal das richtige Bit decodiert. Es ist wie Französisch zu sprechen.

Wenn genügend Rechenkapazität vorhanden ist, kann der Empfänger im Prinzip allen Sendern gleichzeitig zuhören, indem der Decodieralgorithmus für jeden Sender parallel ausgeführt wird. Für das wahre Leben sei angemerkt: Dies ist leichter gesagt als getan, und es ist nützlich zu wissen, welche Sender übertragen könnten.

In dem idealen rauschfreien CDMA-System, das wir hier betrachtet haben, kann die Anzahl der Stationen, die parallel senden, beliebig vergrößert werden, indem längere Chipfolgen benutzt werden. Für 2^n Stationen können Walsh-Codes 2^n orthogonale Chipfolgen der Länge 2^n zur Verfügung stellen. Eine bedeutende Begrenzung ist jedoch unsere Annahme, dass alle Chips beim Empfänger rechtzeitig synchronisiert werden. Diese Synchronisation ist bei einigen Anwendungen wie zellulären Netzen (bei denen CDMA seit den 1990er Jahren verbreitet eingesetzt wird) nicht einmal annähernd realistisch. Dies führt zu unterschiedlichen Entwürfen. Wir werden auf dieses Thema später in diesem Kapitel noch einmal zurückkommen und beschreiben, wie sich asynchrones CDMA von synchronem CDMA unterscheidet.

CDMA wird außer in zellulären Netzen auch in Satelliten- und Kabelnetzen eingesetzt. Wir haben in dieser kurzen Einleitung viele komplizierende Faktoren unter den Teppich gekehrt. Leser, die noch tiefer in die CDMA-Materie einsteigen möchten, sollten Viterbi (1995) und Lee und Miller (1998) lesen. Für diese Literaturtipps ist jedoch ein gewisser Hintergrund in der Telekommunikationstechnik von Vorteil.

2.6 Das öffentliche Telefonnetz

Wenn zwei Computer des gleichen Unternehmens miteinander kommunizieren wollen und nicht zu weit voneinander entfernt sind, ist es meist am einfachsten, sie mit einem Kabel zu verbinden. Dieses Prinzip wird bei LANs angewendet. Steigt die Entfernung oder die Zahl der Computer oder müsste die Leitung eine öffentliche Straße überqueren, erhöhen sich die Kosten für die Verlegung eigener Kabel ganz beträchtlich. Darüber hinaus ist es in fast jedem Land verboten, Verbindungsleitungen über oder unter öffentlichem Eigentum zu verlegen. Daher müssen Netzentwickler auf bereits existierende Telekommunikationsanlagen zurückgreifen.

Diese Anlagen, insbesondere das **öffentliche Telefonnetz** (*Public Switched Telephone Network, PSTN*), wurden vor vielen Jahren mit einem völlig anderen Ziel entwickelt und installiert: Übertragung der menschlichen Sprache in mehr oder weniger klarer Form. Ihre Eignung für die Kommunikation zwischen Rechnern ist eher bescheiden. Zur Verdeutlichung der Größe dieses Problems: Ein billiges handelsübliches Kabel, das zwischen zwei Rechnern verläuft, kann Daten mit 1 Gbit/s oder mehr übertragen. Im Gegensatz dazu arbeitet typisches ADSL – die unglaublich schnelle Alternative zu einem Telefonmodem – mit rund 1 Mbit/s. Die Differenz zwischen diesen beiden entspricht etwa dem Unterschied zwischen der Fortbewegung mit einem Flugzeug und einem gemütlichen Spaziergang.

Dennoch ist das Telefonnetz eng mit Rechnernetzen (über weite Entfernung) verflochten, sodass es sich lohnt, sich damit noch ausführlich zu befassen. Es hat sich gezeigt, dass der begrenzende Faktor für Netzwerkzwecke die „letzte Meile“ zum Kunden ist, nicht die Anschlüsse und Schalter innerhalb des Telefonnetzes. Diese Situation ändert sich mit der allmählichen Einführung von Glasfaser- und Digitaltechniken an den Randbereichen des Netzes, aber es wird noch Zeit und Geld kosten. Während der langen Wartezeit haben sich Computerentwickler daran gewöhnt, mit Systemen zu arbeiten, die eine mindestens dreimal höhere Leistung erbrachten, und haben viel Zeit und Mühe aufgewendet, um herauszufinden, wie das Telefonnetz effizient genutzt werden kann.

In den folgenden Abschnitten beschäftigen wir uns eingehender mit dem Telefonsystem und damit, wie es funktioniert. Das Telefonsystem wird ausführlich in Bellamy (2000) beschrieben.

2.6.1 Aufbau des Telefonsystems

Kurz nachdem Alexander Graham Bell im Jahre 1876 (nur ein paar Stunden vor seinem Rivalen Elisha Gray) das Telefon patentieren ließ, bestand eine enorme Nachfrage nach seiner neuen Erfindung. Telefone wurden damals paarweise verkauft. Der Kunde musste die Apparate selbst mit einem Kabel verbinden. Wollte ein Telefonbesitzer mit n anderen Telefonbesitzern sprechen, musste je ein Kabel zu allen n Häusern verlegt werden. Innerhalb eines Jahres waren die Städte mit einem wilden Gewirr von Drähten überzogen, die über Häuser und Bäume gespannt waren. An diesem Punkt wurde jedem klar, dass das Modell, jedes Telefon mit jedem anderen Telefon zu verbinden (► Abbildung 2.29a), nicht funktionieren konnte.

Man muss Bell zugute halten, dass er dieses Problem frühzeitig voraussah. Er gründete die Bell Telephone Company, die 1878 ihr erstes Vermittlungsamt in New Haven, Connecticut, eröffnete. Das Unternehmen verlegte ein Kabel zum Haus oder Büro jedes Kunden. Um einen Anruf zu tätigen, musste der Kunde am Telefon kurbeln, um im Vermittlungsamt ein Klingelzeichen zu erzeugen und somit einen Mitarbeiter aufmerksam zu machen, der daraufhin den Anrufer mit dem Angerufenen mithilfe einer kurzen Steckverbindung manuell verband. Das Modell einer Vermittlungszentrale zeigt ► Abbildung 2.29b.

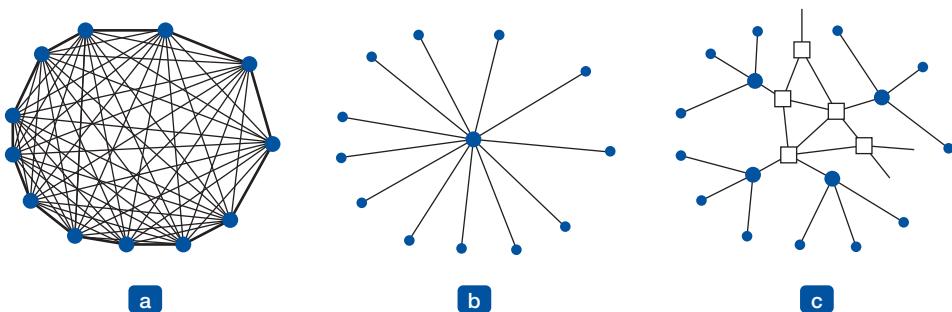


Abbildung 2.29: (a) Netz, in dem alle Teilnehmer direkt miteinander verbunden sind. (b) Zentrale Vermittlungsstelle. (c) Zweistufige Hierarchie.

Innerhalb kürzester Zeit schlossen Vermittlungsbüros von Bell Systems wie Pilze aus dem Boden und die Leute wollten nun auch Ferngespräche führen, sodass Bell anfing, die Vermittlungsbüros zu verbinden. Das ursprüngliche Problem trat auch hier wieder auf: Die Verbindung der einzelnen Vermittlungsbüros untereinander mit einem Kabel von einem zum anderen war bald nicht mehr handhabbar, sodass eine zweite Ebene von Vermittlungsbüros eingeführt wurde. Nach einer gewissen Zeit wurden mehrere Vermittlungsbüros der zweiten Ebene benötigt (siehe ▶ Abbildung 2.29c). Schließlich wuchs die Hierarchie auf fünf Ebenen an.

Im Jahr 1890 standen die drei wichtigsten Komponenten des Telefonsystems: die Vermittlungsbüros, die Kabel zwischen den Kunden und den Vermittlungsbüros (in Form symmetrischer, isolierter Twisted-Pair-Kabel anstelle von offenen Kabeln mit einer Masserückleitung) und die Fernverbindungen zwischen den Vermittlungsbüros. Ein kurzer geschichtlicher Abriss des Telefonsystems ist in Hawley (1991) zu finden.

Seither gab es in allen drei Bereichen beträchtliche Verbesserungen, doch das grundlegende Bell-Modell ist im Wesentlichen seit über 100 Jahren gleich geblieben. Die folgende Beschreibung ist stark vereinfacht, vermittelt aber dennoch die wesentlichen Prinzipien. Jedes Telefon hat zwei ausgehende Kupferkabel, die direkt an die nächste lokale **Teilnehmervermittlungsstelle** (*end office* oder auch *local central office*) angeschlossen werden. Die Entfernung beträgt in der Regel 1 bis 10 km und ist in Stadtgebieten kürzer als auf dem Land. Allein in den USA gibt es ungefähr 22 000 Teilnehmervermittlungsstellen. Die Verbindungen über zwei Kabel zwischen dem Telefon eines Teilnehmers und der lokalen Vermittlungsstelle nennt man **Teilnehmeranschlussleitung** oder **Amtsleitung** (*local loop*). Würde man alle Teilnehmeranschlussleitungen der Welt aneinanderreihen, so wäre dies 1 000-mal die Strecke bis zum Mond und wieder zurück.

Zu einem bestimmten Zeitpunkt bestanden 80 % des gesamten Kapitalvermögens von AT&T aus den für die Teilnehmeranschlussleitungen verlegten Kupferkabeln. AT&T war damals quasi die größte Kupfermine der Welt. Zum Glück wurde diese Tatsache nicht unter den Anlagespekulanten bekannt. Andernfalls hätte ein Spekulant möglicherweise AT&T aufgekauft, den gesamten Telefondienst der USA durch Rausreißen der Kupferkabel zum Erliegen gebracht und die Kupferdrähte anderweitig profitabler verwertet.

Ruft ein an einer bestimmten Teilnehmervermittlungsstelle angeschlossener Teilnehmer einen anderen Teilnehmer an, der an der gleichen Teilnehmervermittlungsstelle angeschlossen ist, wird durch den Vermittlungsmechanismus im Amt eine direkte elektrische Verbindung zwischen den zwei Teilnehmeranschlussleitungen hergestellt. Diese Verbindung bleibt für die Dauer des Gesprächs aufrechterhalten.

Ist der angerufene Teilnehmer an einer anderen lokalen Vermittlungsstelle angeschlossen, muss ein anderes Verfahren angewendet werden. Jede Teilnehmervermittlungsstelle hat eine Reihe von Ausgangsleitungen zu einer oder mehreren nahe gelegenen Vermittlungsstellen, die als **Fernvermittlungsstellen** (*toll office*) oder, falls sie sich im gleichen lokalen Gebiet befinden, als **Durchgangsvermittlung** (*tandem office*) bezeichnet werden. Diese Leitungen nennt man **Fernleitungen** (*toll connecting trunk*). Die Anzahl der unterschiedlichen Arten von Vermittlungsstellen und ihre Topologie variiert von Land zu Land, abhängig von der dort herrschenden Dichte des Telefonnetzes.

Haben die Teilnehmervermittlungsstellen beider gesprächsbereiten Teilnehmer zufällig eine Fernleitung zur gleichen Fernvermittlungsstelle (was wahrscheinlich ist, wenn sie sich in einer geringen Entfernung voneinander befinden), kann die Verbindung innerhalb der Fernvermittlungsstelle aufgebaut werden. In Abbildung 2.29c ist ein Telefonnetz dargestellt, das nur aus Telefonen (den kleinen blauen Punkten), Teilnehmervermittlungsstellen (den großen Punkten) und Fernvermittlungsstellen (den Vierecken) besteht.

Hängen die beiden Teilnehmer nicht an der gleichen Fernvermittlungsstelle, muss ein Pfad zwischen zwei Fernvermittlungsstellen aufgebaut werden. Zwei Fernvermittlungsstellen kommunizieren miteinander über **Verbindungsleitungen** (*intertoll trunk* oder *interoffice trunk*) mit hoher Bandbreite. Vor dem Auseinanderfall von AT&T im Jahre 1984 wurde im US-Telefonsystem ein hierarchisches Routing zum Auffinden eines Pfads durchgeführt, wobei so lange auf eine höhere Hierarchieebene gewechselt wurde, bis man eine gemeinsame Vermittlungsstelle fand. Später ging man dann zu einem flexibleren, nicht hierarchischen Routing über. ►Abbildung 2.30 zeigt, wie eine Verbindung über eine längere Entfernung verlaufen könnte.

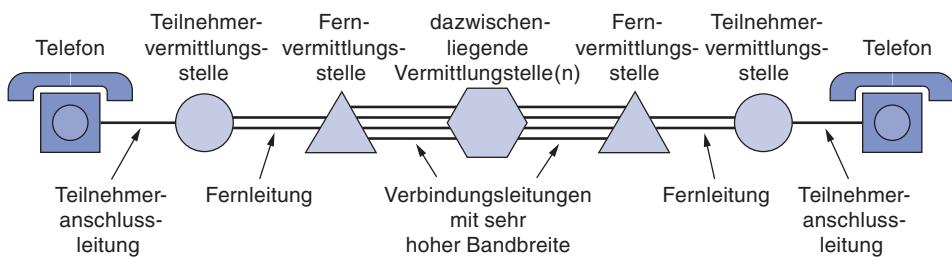


Abbildung 2.30: Ein typischer Leitungsweg für ein Ferngespräch.

In der Telekommunikation werden verschiedene Übertragungsmedien benutzt. Anders als in modernen Bürogebäuden, wo üblicherweise Kabel der Kategorie 5 verlegt sind, bestehen Teilnehmeranschlussleitungen zu Privathaushalten aus Twisted-Pair-Kabeln der Kategorie 3 oder seit Neuestem auch aus Glasfaserkabeln. Zwischen

den Vermittlungsstellen werden Koaxialkabel, Mikrowellen und insbesondere Glasfaserkabel verwendet.

In der Vergangenheit wurden die Daten im Telefonsystem analog übertragen. Das Sprachsignal wurde als elektrische Spannung von der Quelle zum Ziel gesendet. Mit dem Aufkommen von Glasfaserkabeln, Digitalelektronik und Computern wurden alle Verbindungsleitungen und Schaltstellen durch digitale Komponenten ersetzt, sodass die Teilnehmeranschlussleitung die letzte analoge Komponente im System ist. Die digitale Übertragung wird bevorzugt, da hier keine exakte analoge Wellenform erzeugt werden muss, nachdem das Signal bei einem Ferngespräch viele Verstärker durchlaufen hat. Es genügt, genau zwischen einer 0 und einer 1 unterscheiden zu können. Durch diese Eigenschaft wird die digitale Übertragung viel zuverlässiger als die analoge. Außerdem ist sie kostengünstiger und einfacher zu warten.

Die drei wichtigsten Komponenten des Telefonsystems können wie folgt zusammengefasst werden:

- 1.** Teilnehmeranschlussleitungen (analoge Twisted-Pair-Kabel, die in Privatwohnungen und gewerblich genutzte Gebäude gehen)
- 2.** Verbindungsleitungen (digitale Glasfaserkabel, die die Vermittlungsstellen verbinden)
- 3.** Vermittlungsstellen (in denen die Anrufe von einer Verbindungsleitung zur nächsten weitergestellt werden)

Nach einem kleinen Exkurs in die Politik im Bereich Telefonie werden wir uns den oben aufgeführten Komponenten im Detail zuwenden. Die Teilnehmeranschlussleitungen ermöglichen den allgemeinen Zugriff auf das gesamte System, daher sind sie von entscheidender Bedeutung. Leider sind sie auch das schwächste Glied im System. Bei Fernverbindungen ist das Hauptproblem, wie man mehrere Anrufe zusammenstellt und sie über die gleiche Leitung aussendet. Dies ist ein Anwendungsbereich für Multiplexing, in diesem Fall FDM und TDM. Schließlich gibt es noch zwei grundlegend verschiedene Arten der Vermittlung, die wir beide behandeln werden.

2.6.2 Politik im Bereich Telefonie

Bell System bot bis zum Jahr 1984 jahrzehntelang Orts- und Ferndienste in fast dem gesamten Gebiet der USA an. In den 1970er Jahren gelangte die US-Regierung zu der Überzeugung, dass das ein ungesetzliches Monopol sei, und strengte ein Gerichtsverfahren mit dem Bestreben an, dieses aufzubrechen. Die US-Regierung gewann den Prozess und am 1. Januar 1984 wurde AT&T in AT&T Long Lines, 23 **BOCs** (*Bell Operating Company*) und ein paar weitere Unternehmen aufgeteilt. Die 23 BOCs wurden zu sieben regionalen RBOCs zusammengefasst, um sie wirtschaftlich überlebensfähig zu machen. Die Telekommunikation in den USA hatte sich über Nacht per Gerichtsbeschluss (*nicht* durch Begehren des Kongresses) grundlegend geändert.

Die Einzelheiten der Aufteilung wurden im sogenannten **MFJ** (*Modified Final Judgment*, modifiziertes endgültiges Urteil) beschrieben. Das ist ein Oxymoron in seiner reinsten Form – wenn das Urteil modifiziert werden konnte, war es sicherlich nicht endgültig. Dies führte zu mehr Wettbewerb, besseren Dienstleistungen und niedrigeren Preisen für Ferngespräche für Privatpersonen und Unternehmen. Die Preise für die lokalen Dienstleistungen stiegen jedoch, da die Quersubventionierung für Ferngespräche abgeschafft wurde und die lokalen Dienstleistungen sich selber tragen mussten. Viele Länder haben nun auf ähnliche Weise mehr Wettbewerb im Telekommunikationsbereich eingeführt.

Von direkter Relevanz für unsere Betrachtung ist, dass der neue Wettbewerbsrahmen eine Schlüsseltechnologie hervorgebracht hat, welche der Architektur des Telefonnetzes hinzugefügt werden muss. Um unmissverständlich festzuhalten, wer was machen darf, wurde die USA in etwa 160 **Ortsnetzbereiche** (*Local Access and Transport Area, LATA*) aufgegliedert. Ein Ortsnetzbereich entspricht in der Ausdehnung etwa dem Bereich einer Vorwahlnummer. Innerhalb eines Ortsnetzbereiches gibt es normalerweise einen **Ortsnetzbetreiber** (*Local Exchange Carrier, LEC*), der das Monopol auf die konventionellen Telefondienste innerhalb des Ortsnetzbereiches hat. Die meisten größeren Ortsnetzbetreiber sind die verschiedenen örtlichen BOCs. In manchen Ortsnetzbereichen gibt es aber auch eine oder mehr der 1 500 unabhängigen Telefongesellschaften, die als Ortsnetzbetreiber operieren.

Die neue Schlüsseleigenschaft war, dass der gesamte ortsnetzübergreifende Verkehr von einer anderen Gesellschaftsart, einem **Fernnetzbetreiber** (*Interexchange Carrier, IXC*), verwaltet wird. Ursprünglich war AT&T Long Lines der einzige große Fernnetzbetreiber, heute sind aber gut etablierte Mitbewerber wie Verizon und Sprint dazugekommen. Mit der Aufteilung des Telefonriesen sollte unter anderem sichergestellt werden, dass alle Fernnetzbetreiber in Bezug auf Leitungsqualität, Tarife und Anzahl der den Kunden zugewiesenen Telefonnummern gleich behandelt werden. Die Struktur und die Arbeitsweise werden in ► Abbildung 2.31 aufgezeigt. Die Abbildung enthält drei Ortsnetzbereiche, von denen jeder verschiedene Teilnehmervermittlungsstellen aufweist. Ortsnetzbereich 2 und 3 haben auch eine kleine Hierarchie mit Durchgangsämtern.

Möchte ein Fernnetzbetreiber Verbindungen abwickeln, die von einem Ortsnetzbereich ausgehen, so kann dort er eine Vermittlungsstelle namens **POP** (*Point of Presence*) einrichten. Der Ortsnetzbetreiber muss jeden Fernnetzbetreiber an jede Teilnehmervermittlungsstelle anbinden, entweder direkt wie bei den Ortsnetzbereichen 1 und 3 oder indirekt wie bei Ortsnetzbereich 2. Darüber hinaus müssen die technischen und finanziellen Rahmenbedingungen für alle Fernnetzbetreiber gleich sein. Diese Bedingungen ermöglichen es einem Teilnehmer beispielsweise in Ortsnetzbereich 1 zu wählen, welcher Fernnetzbetreiber für den Anruf eines Teilnehmers in Ortsnetzbereich 3 verwendet wird.

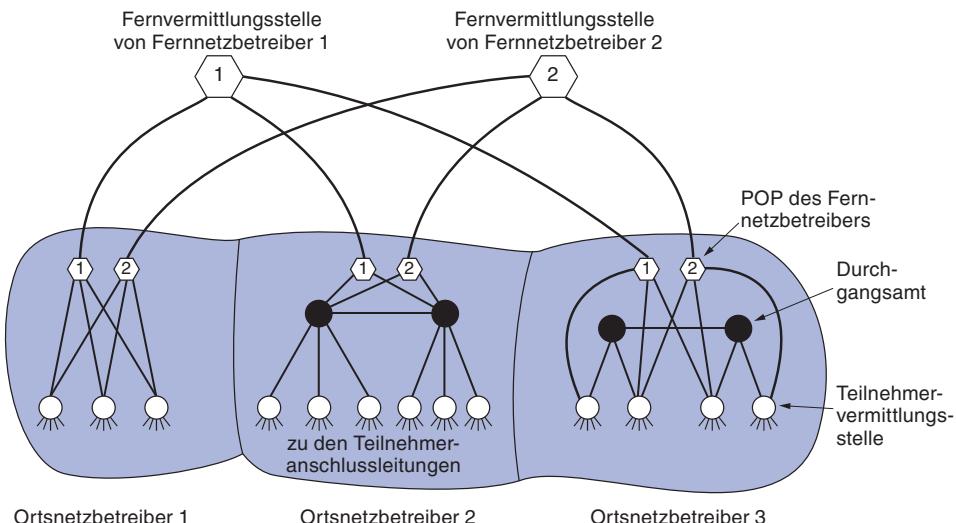


Abbildung 2.31: Die Beziehungen zwischen Ortsnetzbereichen, Ortsnetzbetreibern und Fernnetzbetreibern. Die Kreise stehen für Vermittlungsstellen der Ortsnetzbetreiber, jedes Sechseck für den Fernnetzbetreiber, mit dessen Nummer es beschriftet ist.

Im Rahmen der MFJ war es den Fernnetzbetreibern nicht erlaubt, lokale Telefondienste anzubieten, und die Ortsnetzbetreiber durften keine Telefondienste über den Ortsnetzbereich hinaus anbieten, obwohl beide in jedes andere Geschäft wie den Betrieb von Fast-Food-Restaurants einsteigen durften. Im Jahr 1984 war das eine eindeutige Aussage. Unglücklicherweise hat die Technologie einen lustigen Weg gefunden, das Gesetz überflüssig zu machen. So enthält die Vereinbarung weder für das Kabelfernsehen noch für Mobiltelefone irgendwelche Regelungen. Als sich das Kabelfernsehen von der uni- zu einer bidirektionalen Technik entwickelte und Mobiltelefone ungeheuer an Beliebtheit gewannen, begannen Ortsnetzbetreiber und Fernnetzbetreiber gleichermaßen, Kabel- und Mobilfunkbetreiber zu übernehmen oder mit ihnen zu fusionieren.

1995 stellte der Kongress fest, dass der Versuch, weiterhin zwischen den verschiedenen Gesellschaftsarten zu unterscheiden, nicht mehr länger aufrechterhalten werden konnte. Es wurde ein Gesetzesentwurf eingereicht, um den Zugang zum Wettbewerb beizubehalten und es dennoch Kabelfernsehgesellschaften, örtlichen Telefongesellschaften, Fernverbindungsanbietern und Mobilfunkbetreibern zu erlauben, in die gegenseitigen Geschäftsbereiche einzutreten. Damit wurde das Konzept verfolgt, dass jede Gesellschaft ihren Kunden ein integriertes Gesamtpaket anbieten kann, das Kabelfernsehen, Telefon und Informationsdienste enthält, die bis dato durch verschiedene Gesellschaften monopolistisch angeboten wurden. Dadurch sollte der Wettbewerb auf der Dienst- und Preisschiene erhöht werden. Das Gesetz trat im Februar 1996 als Generalüberholung der Regulierungen im Telekommunikationsbereich in Kraft. In der Folge wurden aus einigen BOCs Fernnetzbetreiber und einige andere Unternehmen wie Kabelfernsehbetreiber begannen lokale Telefondienste im Wettbewerb mit den Ortsnetzbetreibern anzubieten.

Eine interessante Eigenschaft des Gesetzes von 1996 ist die Anforderung, dass die Ortsnetzbetreiber die **Mitnahme der lokalen Telefonnummer** ermöglichen müssen. Dies bedeutet, dass ein Kunde die lokalen Telefongesellschaften wechseln kann, ohne hierbei eine neue Telefonnummer beantragen zu müssen. Die Mitnahme der Mobiltelefonnummer (und zwischen Fest- und Mobileitungen) folgte 2003. Durch diese Bestimmung entfällt für viele Leute eine riesige Hürde und sie sind sehr viel eher geneigt, den Ortsnetzbetreiber zu wechseln. Daraufhin wurde die US-amerikanische Telekommunikationslandschaft noch wettbewerbsorientierter und andere Länder zogen nach. Oft warten andere Länder einmal ab, wie das Experiment in den Vereinigten Staaten funktioniert. Wenn es gut geht, machen sie dasselbe, wenn nicht, probieren sie etwas Neues aus.

2.6.3 Teilnehmeranschlüsse: Modems, ADSL und Glasfaser

Jetzt ist es an der Zeit, uns einmal genauer anzusehen, wie das Telefonsystem funktioniert. Beginnen wir mit dem Teil, mit dem die meisten Menschen vertraut sind: die zweidrige Teilnehmeranschlussleitung, die von der Telefongesellschaft in Häuser gelegt wird. Diese Teilnehmeranschlussleitung wird auch häufig als die „letzte Meile“ bezeichnet, obwohl sich die Länge auf mehrere Kilometer belaufen kann. Hier werden seit über 100 Jahren analoge Informationen übertragen und dies wird auch in den nächsten Jahren noch so sein, da die Umstellungskosten auf die digitale Betriebsart hoch sind.

Es wurden große Anstrengungen unternommen, um das letzte Quäntchen Datenübertragungskapazität aus den vorhandenen Kupferanschlüssen herauszuquetschen. Telefonmodems senden digitale Daten zwischen Rechnern über den schmalen Kanal, den die Telefonnetzbetreiber für einen Sprachanruf zur Verfügung stellen. Sie waren einst weitverbreitet, sind heute aber größtenteils von Breitbandtechnologien wie beispielsweise ADSL ersetzt worden, die den Teilnehmeranschluss wiederverwenden, um digitale Daten von einem Kunden zur Vermittlungsstelle zu senden, von wo aus sie ins Internet übertragen werden. Sowohl Modems als auch ADSL müssen mit den Begrenzungen der alten Teilnehmeranschlüsse zurecht kommen: relativ schmale Bandbreite, Dämpfung und Verzerrung von Signalen und Anfälligkeit für elektrische Störgeräusche wie Übersprechen.

In einigen Orten wurde der Teilnehmeranschluss modernisiert, indem Glasfaserkabel zum (oder bis sehr nah zum) Wohnhaus installiert werden. Glasfasern sind hier die Technologie der Zukunft. Diese unterstützen Rechnernetze von Grund auf, da der Teilnehmeranschluss damit genügend Bandbreite für Datendienste bekommt. Der begrenzende Faktor sind hier nicht die physikalischen Gegebenheiten des Teilnehmeranschlusses, sondern die Frage, was die Kunden bereit sind, dafür zu bezahlen.

In diesem Abschnitt werden wir uns sowohl die alten als auch die neuen Teilnehmeranschlussleitungen ansehen. Wir behandeln Telefonmodems, ADSL und FttH (*Fiber to the Home*).

Telefonmodems

Um Bits über die Teilnehmeranschlussleitung oder einen anderen physischen Kanal zu senden, müssen diese in analoge Signale konvertiert werden, die über den Kanal übertragen werden können. Diese Konvertierung wird mithilfe von Methoden der digitalen Modulation durchgeführt, die wir im vorigen Abschnitt betrachtet haben. Am anderen Ende des Kanals wird das analoge Signal zurück in Bits konvertiert.

Ein Gerät, das zwischen einem Strom von digitalen Bits und einem analogen Signal, das die Bits repräsentiert, konvertiert, heißt **Modem** (kurz für „Modulator Demodulator“). Modems gibt es in einer großen Vielfalt: Telefonmodems, DSL-Modems, Kabelmodems, drahtlose Modems usw. Das Modem kann in den Rechner eingebaut sein (was heutzutage für Telefonmodems üblich ist) oder eine eigenständiger Kasten sein (was für DSL- und Kabelmodems gebräuchlich ist). Logisch befindet sich das Modem zwischen dem (digitalen) Computer und dem (analogen) Telefonsystem (► Abbildung 2.32).

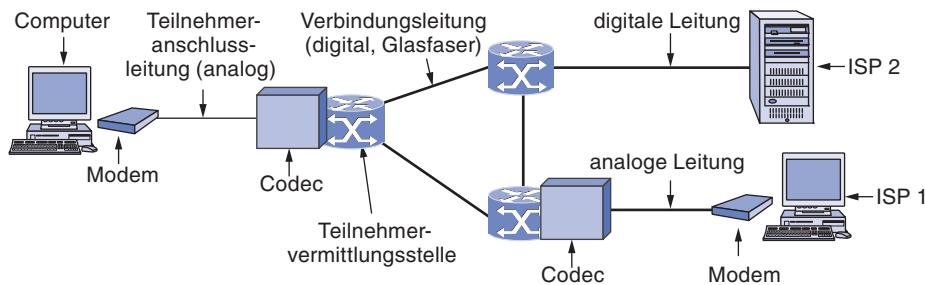


Abbildung 2.32: Der Einsatz von analoger und digitaler Übertragung für einen Anruf von Computer zu Computer. Die Konvertierung wird durch Modems und Codecs vorgenommen.

Telefonmodems werden benutzt, um Bits zwischen zwei Computern über eine Sprachtelefonleitung zu senden, die normalerweise für Gespräche verwendet wird. Die Hauptschwierigkeit dabei ist, dass eine Sprachleitung auf 3 100 Hz begrenzt ist, was für die Gesprächsübertragung ausreichend ist. Diese Bandbreite ist mehr als vier Größenordnungen geringer als die Bandbreite von Ethernet oder IEEE 802.11 (WiFi). Es überrascht daher nicht, dass die Datenraten von Telefonmodems um Größenordnungen geringer als die von Ethernet und WiFi sind.

Werfen wir kurz einen Blick darauf, wie diese Zahl zustande kommt. Das Nyquist-Theorem besagt, dass es sogar mit einer perfekten 3 000-Hz-Leitung (was eine Telefonleitung ganz sicher nicht ist) keinen Sinn hat, Symbole schneller als mit 6 000 Baud zu senden. In der Praxis senden die meisten Modems mit einer Rate von 2 400 Symbole/Sekunde bzw. 2 400 Baud und erhalten mehrere Bits pro Symbol, während Datenverkehr in beiden Richtungen gleichzeitig möglich ist (indem verschiedene Frequenzen für verschiedene Richtungen benutzt werden).

Das bescheidene 2 400-Bit/s-Modem benutzt 0 Volt für eine logische 0 und 1 Volt für eine logische 1, wobei 1 Bit pro Symbol verwendet wird. Ein Schritt weiter wäre die Verwendung von vier unterschiedlichen Symbolen, wie z.B. in den vier Phasen von QPSK. Somit kann mit 2 Bit/Symbol eine Datenrate von 4 800 Bit/s erreicht werden.

Mit der Weiterentwicklung der Technologie wurden auch höhere Datenraten erreicht. Höhere Raten erfordern eine größere Menge von Symbolen oder **Konstellationen**. Bei vielen Symbolen kann selbst ein wenig Rauschen in der erkannten Amplitude oder Phase zu einem Fehler führen. Um die Wahrscheinlichkeit von Fehlern zu reduzieren, verwenden Standards für Modems mit hoher Geschwindigkeit einige der Symbole zur Fehlerbehebung. Die Schemata sind unter dem Namen **Trellis Coded Modulation (TCM)** bekannt (Ungerboeck, 1987).

Der **V.32**-Modemstandard benutzt 32 Konstellationspunkte, um 4 Datenbits und 1 Prüfbit pro Symbol mit 2 400 Baud zu übertragen, womit 9 600 Bit/s einschließlich Fehlererkennung erreicht werden. Der nächste Schritt über 9 600 Bit/s hinaus sind 14 400 Bit/s. Dieser Standard heißt **V.32bis** und definiert die Übertragung von 6 Datenbits und 1 Prüfbit pro Symbol bei 2 400 Baud. Als Nächstes kommt **V.34**, welches 28 800 Bit/s erreicht, indem 12 Datenbits/Symbol bei 2 400 Baud übertragen werden. Die Konstellation hier besteht aus Tausenden von Punkten. Das letzte Modem in dieser Serie ist **V.34bis**, welches 14 Datenbits/Symbol zu 2 400 Baud sendet, um 33 600 Bit/s zu erreichen.

Doch warum hier aufhören? Standardmodems gehen nicht über 33 600 Bit/s hinaus, weil die Shannon-Grenze für das Telefonsystem bei ca. 35 kbit/s liegt, basierend auf der durchschnittlichen Länge der Teilnehmeranschlüsse und der Qualität dieser Leitungen. Diese Grenze zu überschreiten, würde die physikalischen Gesetze verletzen (Abteilung für Thermodynamik).

Es gibt jedoch eine Möglichkeit, diese Situation zu ändern. Bei der Teilnehmervermittlungsstelle der Telefongesellschaft werden die Daten für die Übertragung innerhalb des Telefonnetzes in eine digitale Form konvertiert (der Kern des Telefonnetzes wurde vor langer Zeit von analog zu digital konvertiert). Die 35-kbit/s-Grenze gilt für Situationen, in denen es zwei Teilnehmeranschlussleitungen gibt, eine an jedem Ende. Durch jede dieser Leitungen wird dem Signal Rauschen hinzugefügt. Wenn wir einen dieser Teilnehmeranschlüsse loswerden könnten, würden wir das Signalaus-Rausch-Verhältnis erhöhen und die maximale Datenrate könnte verdoppelt werden.

Dies ist genau das Vorgehen bei 56-kbit/s-Modems. Das eine Ende, in der Regel ein ISP, bekommt eine hochqualitative digitale Einspeisung von der nächsten Teilnehmervermittlungsstelle. Wenn ein Ende der Verbindung ein hochqualitatives Signal ist, wie es bei den meisten ISPs heute der Fall ist, kann die maximale Datenrate somit bis zu 70 kbit/s betragen. Zwischen zwei privaten Endnutzern mit Modems und analogen Leitungen liegt das Maximum noch bei 33,6 kbit/s.

Der Grund dafür, dass 56-kbit/s-Modems (statt 70-kbit/s-Modems) in Gebrauch sind, hat mit dem Nyquist-Theorem zu tun. Ein Sprachkanal wird innerhalb des Telefonsystems als digitales Signalmuster übertragen. Jeder Kanal ist 4 000 Hz breit, wenn man das Sicherheitsband dazu nimmt. Die Anzahl der Abtastungen pro Sekunden, die zur Rekonstruktion benötigt werden, beträgt somit 8 000. In den USA ist die Anzahl der Bits pro Abtastung 8, ein Bit davon kann zu Kontrollzwecken benutzt werden, womit 56 000 Bit/s für Benutzerdaten zur Verfügung stehen. In Europa sind alle 8 Bits für die

Anwender verfügbar, sodass theoretisch Modems mit 64 000 Bit/s verwendet werden könnten, doch um internationale Vereinbarungen über Standards zu erreichen, wurde 56 000 Bit/s gewählt.

Das Endergebnis sind der **V.90**- und der **V.92**-Modemstandard. Diese stellen für einen 56-kbit/s-Empfangskanal (vom ISP zum Anwender) bzw. einen 33,6-kbit/s- und 48-kbit/s-Rückkanal (vom Anwender zum ISP) zur Verfügung. Die Asymmetrie liegt daran, dass es in der Regel mehr Datentransport vom ISP zum Benutzer statt andersherum gibt. Dies bedeutet auch, dass mehr von der begrenzten Bandbreite dem Empfangskanal zugewiesen werden kann, um damit die Chance zu erhöhen, dass das Modem tatsächlich 56 kbit/s erreicht.

DSL

Als die Telefonbranche schließlich 56 kbit/s erreicht hatte, klopfte sie sich selbst auf die Schulter dafür, die Aufgabe so gut gelöst zu haben. In der Zwischenzeit bot die Kabelfernsehbranche Geschwindigkeiten mit bis zu 10 Mbit/s auf gemeinsam genutzten Kabeln an. Der Internetzugang gewann für das Geschäft der Telekommunikationsgesellschaften zunehmend an Bedeutung und man erkannte, dass man ein wettbewerbsfähigeres Produkt benötigte. Die Antwort darauf war das Angebot neuer digitaler Dienste über die Teilnehmeranschlussleitung.

Anfangs gab es viele einander überschneidende Hochgeschwindigkeitsangebote, die alle unter dem Namen **xDSL** (*Digital Subscriber Line*) für verschiedene x liefen. Dienste mit einer höheren Bandbreite als der Standardtelefondienst werden manchmal als **Breitbanddienste** bezeichnet, obwohl der Begriff eigentlich mehr ein Marketingkonzept als ein besonderes technisches Konzept ist. Später besprechen wir das, was zum bekanntesten dieser Dienste geworden ist: **ADSL** (*Asymmetric DSL*). Wir werden den Begriff „DSL“ oder „xDSL“ als Abkürzung für alle Spielarten verwenden.

Modems sind deshalb so langsam, weil Telefone für die Übertragung der menschlichen Stimme entwickelt wurden und das gesamte System sorgfältig darauf abgestimmt wurde. Daten wurden immer als Stiefkinder behandelt. An dem Punkt, an dem eine Teilnehmeranschlussleitung in der Teilnehmervermittlungsstelle endet, läuft das Kabel durch einen Filter, der alle Frequenzen unter 300 Hz und über 3 400 Hz dämpft. Dies kann man verschmerzen. 300 Hz und 3 400 Hz sind die 3-dB-Punkte, sodass die Bandbreite in der Regel mit 4 000 Hz angegeben wird, selbst wenn der Abstand zwischen den 3-dB-Punkten bei 3 100 Hz liegt. Die Daten auf dem Kabel sind daher auf dieses schmale Band beschränkt.

Der Trick, der bei xDSL angewendet wird, ist folgender: Wenn ein Teilnehmer xDSL abonniert, wird die eingehende Leitung an eine andere Vermittlungsstelle (Switch) angeschlossen, die keinen Filter hat, sodass die gesamte Kapazität der Teilnehmeranschlussleitung verfügbar wird. Der einschränkende Faktor sind nun die physischen Merkmale der Teilnehmeranschlussleitung, die ca. 1 MHz unterstützt, und nicht die künstliche, durch den Filter erzeugte Bandbreite von 3 100 Hz.

Bedauerlicherweise fällt die Kapazität der Teilnehmeranschlussleitung recht schnell mit der Entfernung von der Vermittlungsstelle, wobei das Signal entlang des Kabels zunehmend schwächer wird. Es hängt außerdem von der Dicke und der allgemeinen Qualität der Twisted-Pair-Kabel ab. Eine Darstellung der möglichen Bandbreite als Funktion der Entfernung wird in ► Abbildung 2.33 gezeigt. In der Abbildung wird davon ausgegangen, dass alle anderen Faktoren optimal sind (neue Kabel, mittlere Kabelbündel etc.).

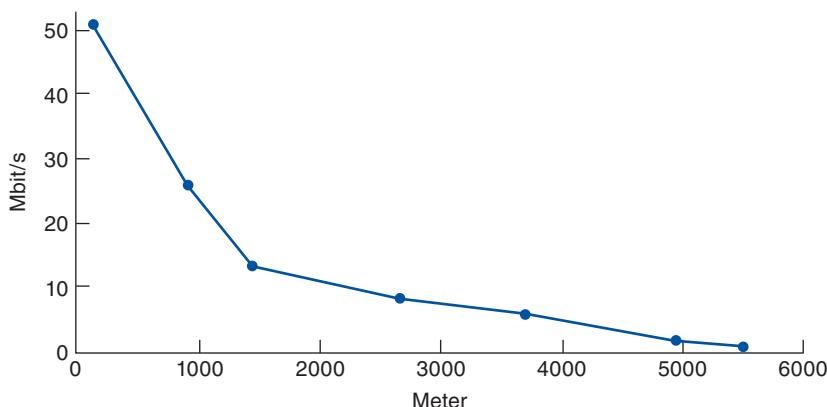


Abbildung 2.33: Bandbreite im Verhältnis zur Entfernung für DSL über UTP der Kategorie 3.

Die Auswirkungen dieser Abbildung führen für die Telefongesellschaft zu einem Problem. Wenn die Telefongesellschaft eine bestimmte Geschwindigkeit auswählt, wählt sie gleichzeitig einen Radius aus, über den hinaus dieser Dienst nicht bereitgestellt werden kann. Dies bedeutet, wenn Kunden an eher abgelegenen Orten versuchen, sich für den Dienst anzumelden, müssen sie unter Umständen mit folgendem Satz rechnen: „Vielen Dank für Ihr Interesse, aber Sie sind 100 m zu weit von der nächsten Teilnehmervermittlungsstelle entfernt und wir können Ihnen den Dienst nicht bereitstellen. Könnten Sie vielleicht umziehen?“ Je niedriger die gewählte Geschwindigkeit ist, umso größer wird der Einzugskreis und damit auch der Kundenkreis. Aber je geringer die Geschwindigkeit, umso weniger attraktiv ist der Dienst und umso weniger Leute sind bereit, dafür zu zahlen. Hier treffen Geschäft und Technologie aufeinander.

Die xDSL-Dienste wurden mit einer bestimmten Zielsetzung konzipiert. Als Erstes muss xDSL über die bestehenden Twisted-Pair-Anschlussleitungen der Kategorie 3 funktionieren. Zweitens dürfen die bei den Kunden vorhandenen Telefone und Faxe nicht beeinträchtigt werden. Drittens müssen sie sehr viel schneller als 56 kbit/s sein. Viertens sollten sie immer aktiviert sein, wobei eine monatliche Abrechnung und keine nach Minuten zugrunde gelegt wird.

Um die technischen Ziele zu erreichen, wird das verfügbare 1,1-MHz-Spektrum auf der Teilnehmeranschlussleitung in 256 unabhängige Kanäle von jeweils 4 312,5 Hz aufgeteilt. Diese Anordnung ist in ► Abbildung 2.34 dargestellt. Das OFDM-Schema, das wir im vorigen Abschnitt besprochen haben, wird zum Senden von Daten über diese Kanäle benutzt, im Kontext von ADSL wird es allerdings häufig **Mehrtonverfahren** genannt.

ren (**DMT**, *Discrete MultiTone*) genannt. Kanal 0 wird für den „guten alten Telefon-dienst“ (**POTS**, *Plain Old Telephone Service*) verwendet. Die Kanäle 1–5 werden nicht genutzt, um Störungen zwischen den Sprachsignalen und den Datensignalen zu vermeiden. Von den restlichen 250 Kanälen wird einer für die Upstream-Überwachung und einer für die Downstream-Überwachung eingesetzt. Der Rest steht für Benutzerdaten zur Verfügung.

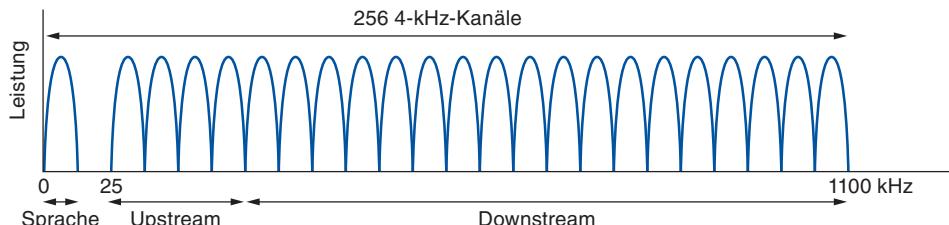


Abbildung 2.34: Betrieb von ADSL mit diskreter Mehrtonmodulation.

Im Grunde ist dann jeder der restlichen Kanäle für einen Duplexdatenstrom verfügbar, aber durch die Harmonischen, Übersprechen und andere Effekte bleiben die in der Praxis eingesetzten Systeme weit unter dieser theoretischen Grenze. Es ist Sache des Anbieters festzulegen, wie viele Kanäle für das Senden und wie viele für den Empfang reserviert sind. Eine 50/50-Kombination von Sendekapazität und Empfangskapazität ist theoretisch möglich, aber die meisten Provider weisen etwa 80 % – 90 % der Bandbreite dem Empfangskanal zu, weil die meisten Benutzer mehr Daten herunterladen als übertragen. Diese Festlegung führt zum „A“ in ADSL. Eine gängige Aufteilung sind 32 Kanäle für das Senden von Daten, der Rest ist für den Empfang von Daten reserviert. Einige der höchsten Rückkanäle können auch zur Erhöhung der Bandbreite bidirektional gemacht werden. Diese Optimierung erfordert allerdings den Einbau einer speziellen Leitung, um Echos zu unterbinden.

Der internationale ADSL-Standard, bekannt als **G.dmt**, wurde 1999 herausgegeben. Es werden Geschwindigkeiten bis 8 Mbit/s Downstream und 1 Mbit/s Upstream unterstützt. Der Standard wurde 2002 von einer zweiten Generation namens ADSL2 abgelöst, der verschiedene Verbesserungen enthielt, um Geschwindigkeiten wie 12 Mbit/s Downstream und 1 Mbit/s Upstream zu ermöglichen. Mittlerweile sind wir bei ADSL2+, bei dem die Downstream-Geschwindigkeit auf 24 Mbit/s erhöht wird, indem die Bandbreite verdoppelt wird, um 2,2 MHz auf den Twisted-Pair-Kabeln zu nutzen.

Die hier angegebenen Zahlen stellen jedoch die besten Geschwindigkeiten für gute Leitungen dar, die nah (d.h. innerhalb von 1–2 km) an der Vermittlungsstelle sind. Nur wenige Leitungen unterstützen diese Datenraten und nur wenige Anbieter stellen diese Geschwindigkeiten zur Verfügung. In der Regel bietet ein Provider um die 1 Mbit/s für den Empfang und 256 kbit/s zum Senden (als Standarddienst), 4 Mbit/s für den Empfang und 1 Mbit/s zum Senden (als verbesserter Service) sowie 8 Mbit/s für den Empfang und 2 Mbit/s zum Senden (als Premiumdienst).

In jedem Kanal wird QAM-Modulation mit einer Rate von rund 4 000 Symbolen pro Sekunde verwendet. Die Leitungsqualität jedes Kanals wird fortlaufend überwacht und die Datenübertragungsrate kontinuierlich angepasst, indem wie in Abbildung 2.23 eine größere oder kleinere Konstellation benutzt wird. Verschiedene Kanäle können ganz unterschiedliche Datenraten aufweisen, je nach Standard von maximal 15 Bit pro Symbol bei einem Kanal mit einem hohen Signal-Rausch-Verhältnis bis zu 2, 1 oder keinem Bit pro Symbol bei einem Kanal mit einem niedrigen Signal-Rausch-Verhältnis.

Eine typische ADSL-Konfiguration ist in ►Abbildung 2.35 dargestellt. In diesem Schema muss der Techniker einer Telefongesellschaft ein **NID** (*Network Interface Device*) auf dem Grundstück des Kunden installieren. Diese kleine Plastikbox markiert das Ende des Eigentums der Telefongesellschaft und den Beginn des Kundeneigentums. Neben dem NID (manchmal auch in einem Gerät) befindet sich der **Splitter**, ein analoger Filter, der das von POTS genutzte 0- bis 4 000-Hz-Band von den Daten trennt. Das POTS-Signal wird an das vorhandene Telefon oder Fax weitergeleitet. Das Daten-Signal wird zu einem ADSL-Modem geleitet, welches digitale Signalverarbeitung benutzt, um OFDM zu implementieren. Da die meisten ADSL-Modems externe Geräte sind, muss der Computer mit diesen über eine Hochgeschwindigkeitsleitung verbunden werden. In der Regel wird hierzu Ethernet, ein USB-Kabel oder WiFi benutzt.

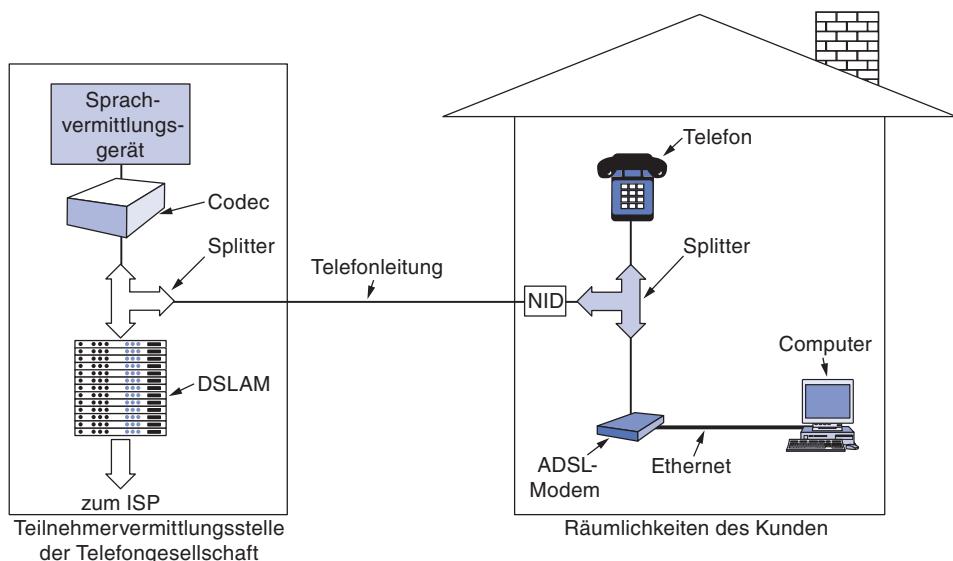


Abbildung 2.35: Eine typische ADSL-Gerätekonfiguration.

Am anderen Ende des Kabels, an der Teilnehmervermittlungsstelle, ist ebenfalls ein entsprechender Splitter installiert. Hier wird der Sprachanteil des Signals herausgefiltert und an eine normale Sprachvermittlung weitergeleitet. Das Signal über 26 kHz wird an ein neuartiges Gerät namens **DSLAM** (*Digital Subscriber Line Access Multiplexer*) weitergeleitet, das den gleichen digitalen Signalprozessortyp wie das ADSL-Modem enthält. Nachdem die Bits aus dem Signal wiederhergestellt sind, werden Pakete gebildet und an den ISP gesendet.

Durch die komplette Trennung von Sprachsystem und ADSL ist es für die Telefongesellschaft relativ einfach, ADSL einzusetzen. Hierzu müssen nur ein DSLAM und ein Splitter erworben werden und die ADSL-Teilnehmer mit dem Splitter verbunden werden. Andere Dienste mit hoher Bandbreite (wie ISDN) erfordern sehr viel umfangreichere Änderungen an den bestehenden Vermittlungsgeräten.

Ein Nachteil des Entwurfs von Abbildung 2.35 ist die Voraussetzung, dass sich NID und Splitter im Privatbereich des Kunden befinden. Die Installation kann hier nur von einem Techniker der Telefongesellschaft vorgenommen werden, was aufwendig und teuer ist. Deshalb wurde auch ein anderes Design ohne Splitter standardisiert, das informell als **G-lite** bezeichnet wird. Die Konfiguration ist die gleiche wie in Abbildung 2.35, nur ohne den Splitter beim Kunden. Stattdessen wird die bestehende Telefonleitung genommen. Der einzige Unterschied ist, dass ein Mikrofilter in jede Telefonbuchse zwischen dem Telefon oder ADSL-Modem und der Leitung eingefügt werden muss. Der Mikrofilter für das Telefon ist ein Tiefpassfilter, der Frequenzen über 3 400 Hz eliminiert; der Mikrofilter für das ADSL-Modem ist ein Hochpassfilter, der Frequenzen unter 26 kHz eliminiert. Dieses System ist aber nicht so zuverlässig wie ein Splitter, sodass G-lite nur bis zu 1,5 Mbit/s (im Unterschied zu 8 Mbit/s bei ADSL mit Splitter) zum Einsatz kommt. Weitere Informationen über ADSL finden Sie in Starr (2003).

FttH – Fiber to the Home

Die vorhandenen Teilnehmeranschlüsse aus Kupfer begrenzen die Leistung von ADSL und Telefonmodems. Um einen schnelleren und besseren Netzwerkdienst anzubieten, erneuern Telefongesellschaften die Teilnehmeranschlussleitungen bei jeder Gelegenheit und installieren Glasfaser auf dem gesamten Weg zu Haushalten und Büros. Das Ergebnis wird **FttH** (*Fiber to the Home*) genannt. Obwohl die FttH-Technologie schon seit einiger Zeit verfügbar ist, wurde mit der Installation erst im Jahr 2005 begonnen, als die Nachfrage von Kunden nach Hochgeschwindigkeitsinternet wuchs, die an DSL und Kabel gewöhnt waren und Filme herunterladen wollten. Rund 4 % der Haushalte in den USA sind heute an FttH mit Internetzugangsgeschwindigkeiten von bis zu 100 Mbit/s angebunden.

Es gibt mehrere Variationen der Form „FttX“ (wobei X für *Basement*, *Curb* und *Neighborhood* steht). Sie werden eingesetzt, um zu vermerken, wie nah die Glasfaserinstallation an das Haus reicht. In diesem Fall ist die Geschwindigkeit, die Kupferleitungen (Twisted-Pair- oder Koaxialkabel) bieten können, schnell genug für die letzte kurze Distanz. Bei der Entscheidung, wie weit Glasfaser verlegt wird, spielen ökonomische Überlegungen eine Rolle, da die Kosten mit dem erwarteten Gewinn aufzurechnen sind. Der springende Punkt hier ist, dass Glasfaser auf jeden Fall die traditionelle Barriere der „letzten Meile“ überschritten hat. Wir werden uns in unserer Diskussion auf FttH konzentrieren.

Wie der Vorgänger aus Kupfer ist der Teilnehmeranschluss aus Glasfaser passiv. Dies bedeutet, es ist keine strombetriebene Ausrüstung notwendig, um Signale zu verstärken oder anderweitig zu verarbeiten. Die Glasfaser trägt einfach Signale zwischen dem

Haus und der Vermittlungsstelle. Dies verringert wiederum die Kosten und verbessert die Zuverlässigkeit.

In der Regel werden die Glasfasern von mehreren Häusern gebündelt, sodass nur eine einzige Glasfaser pro Gruppe von maximal 100 Häusern in der Vermittlungsstelle ankommt. In Downstream-Richtung teilen optische Splitter das Signal von der Vermittlungsstelle, sodass es alle Häuser erreicht. Aus Sicherheitsgründen ist eine Verschlüsselung notwendig, wenn vorgesehen ist, dass nur ein Haus das Signal decodieren darf. In der Upstream-Richtung verschmelzen optische Kombinatoren die Signale aus den verschiedenen Häusern zu einem einzigen Signal, das an der Vermittlungsstelle empfangen wird.

Diese Architektur heißt **passives optisches Netz (PON, Passive Optical Network)** und ist in ▶ Abbildung 2.36 dargestellt. Es ist üblich, eine Wellenlänge zu benutzen, die von allen Häusern für Downstream-Übertragungen gemeinsam verwendet wird, und eine weitere Wellenlänge für Upstream-Übertragungen.

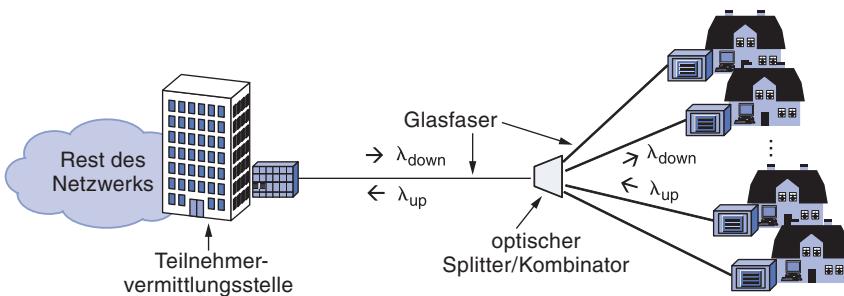


Abbildung 2.36: Passives optisches Netz (PON) für FttH.

Selbst mit dieser Aufteilung bringen es die ungeheure Bandbreite und die geringe Dämpfung der Glasfaser mit sich, dass PONs den Benutzern über Entfernungen von bis zu 20 km hohe Datenraten zur Verfügung stellen kann. Die aktuellen Datenraten und andere Einzelheiten hängen vom Typ des PON ab. Zwei Arten sind üblich. **GPONs (Gigabit-capable PON)** kommen aus der Welt der Telekommunikation, sie werden also durch einen ITU-Standard festgelegt. **EPONs (Ethernet PON)** sind mehr auf die Welt der Netze abgestimmt, sie werden also durch einen IEEE-Standard definiert. Beide laufen mit rund einem Gigabit und können Datenverkehr für unterschiedliche Dienste übertragen, einschließlich Internet, Video und Sprache. Zum Beispiel stellen GPONs 2,4 Gbit/s für den Empfang und 1,2 oder 2,4 Gbit/s zum Hochladen.

Für die gemeinsame Nutzung der Kapazität einer einzelnen Glasfaser in der Vermittlungsstelle durch verschiedene Häuser ist außerdem ein Protokoll notwendig. Die Downstream-Richtung ist einfach. Die Vermittlungsstelle kann Nachrichten zu jedem einzelnen Haus in beliebiger Reihenfolge senden. Andersherum können jedoch Nachrichten von verschiedenen Endkunden nicht gleichzeitig gesendet werden, ohne dass die Gefahr von Kollisionen besteht. Da die Teilnehmer auch nicht die Übertragungen der jeweils anderen mithören können, ist vor dem Senden auch kein Abhorchen der Leitung möglich. Die Lösung besteht darin, dass das Endkundengerät Zeitscheiben in

der Vermittlungsstelle anfordert und zugeteilt bekommt. Dazu wird ein Verfahren zur Distanzmessung verwendet, um die Übertragungszeiten vom Haus so anzupassen, dass alle Signale, die an der Vermittlungsstelle ankommen, synchronisiert sind. Der Entwurf ist ähnlich den Kabelmodems, die wir später in diesem Kapitel behandeln. Weitere Informationen über die Zukunft von PONs finden Sie bei Grobe und Elbers (2008).

2.6.4 Verbindungsleitungen und Multiplexverfahren

Verbindungsleitungen im Telefonnetz sind nicht nur viel schneller als die Teilnehmeranschlussleitungen, sie unterscheiden sich noch in zwei weiteren Beziehungen. Der Kern des Telefonnetzes trägt digitale, keine analoge Information, das heißt: Bits statt Sprache. Dies erfordert eine Umwandlung in der Vermittlungsstelle in die digitale Form zur Übertragung über Langstreckenverbindungen. Die Verbindungsleitungen tragen Tausende, sogar Millionen von Gesprächen gleichzeitig. Diese gemeinsame Nutzung ist wichtig für das Erreichen von Skaleneffekten, da es im Wesentlichen die gleiche Menge Geld kostet, eine Verbindungsleitung mit hoher Bandbreite zu installieren und unterhalten wie eine Verbindungsleitung mit niedriger Bandbreite zwischen zwei Vermittlungsstellen. Dies wird mit Varianten von TDM- und FDM-Multiplexing erreicht.

Im Folgenden werden wir kurz untersuchen, wie Sprachsignale so digitalisiert werden, dass sie vom Telefonnetz transportiert werden können. Danach betrachten wir, wie TDM im Lichtwellenleiterbereich eingesetzt werden kann, um Bits über Verbindungsleitungen zu übertragen, einschließlich des TDM-Systems für Lichtwellenleiter (SONET). Dann sehen wir, wie FDM für Glasfaser eingesetzt wird, was Wellenlängenmultiplexing (*wavelength division multiplexing*) genannt wird.

Digitalisieren von Sprachsignalen

In den Anfängen der Telefonnetzentwicklung behandelte der Kern Sprachsignale als analoge Information. FDM-Techniken wurden über viele Jahre eingesetzt, um 4 000-Hz-Sprachkanäle (bestehend aus 3 100 Hz plus Sicherheitsbänder) in immer größeren Einheiten mehrfach zu nutzen. Beispielsweise heißen 12 Anrufe im 60–108-kHz-Band eine **Gruppe** und fünf Gruppen (60 Anrufe) sind eine **Supergruppe** und so weiter. Diese FDM-Methoden werden noch auf einigen Kupferleitungen und Mikrowellenkanälen eingesetzt. FDM benötigt jedoch analoge Schaltkreise und kann nicht von einem Computer ausgeführt werden. Im Gegensatz dazu kann TDM vollständig von der digitalen Elektronik abgewickelt werden, sodass es in den letzten Jahren weit mehr verwendet wird. Da TDM nur für digitale Daten benutzt werden kann und die Teilnehmeranschlussleitungen analoge Signale erzeugen, muss eine Umwandlung von analog zu digital in der Vermittlungsstelle vorgenommen werden. Dabei werden alle einzelnen Teilnehmeranschlüsse auf Ausgangsverbindungsleitungen zusammengefasst.

Die Analogsignale werden im Fernmeldeamt von einem Gerät namens **Codec** (Kurzform für „Coder-Decoder“) digitalisiert. Der Codec nimmt 8 000 Abtastungen pro Sekunde (125 µs/Abtastung) vor, weil das laut Nyquist-Theorem ausreicht, um alle Informationen von einem Telefonkanal mit einer Bandbreite von 4 kHz zu erfassen.

Mit einer niedrigeren Abtastrate würden Informationen verloren gehen. Bei einer höheren könnten keine zusätzlichen Informationen gewonnen werden. Jede Abtastung der Amplitude eines Signals wird zu einer 8-Bit-Zahl quantisiert.

Diese Technik heißt **Pulse Modulation (PCM, Pulse Code Modulation)**. PCM bildet den Kern eines modernen Telefonsystems. Praktisch alle Zeitintervalle im Telefonsystem sind Vielfache von 125 µs. Die Standardrate für unkomprimierte Daten bei einem Sprachanruf über die Telefonleitung ist daher 8 Bits jede 125 µs bzw. 64 kbit/s.

Am anderen Ende des Anrufs wird aus den quantisierten Abtastungen wieder ein analoges Signal erzeugt, indem diese über die Zeit abgespielt (und geglättet) werden. Das neue Signal wird nicht exakt mit dem ursprünglichen analogen Signal übereinstimmen, selbst wenn wir mit Nyquist-Rate abgetastet haben, da die Abtastungen quantisiert wurden. Um die Fehler zu reduzieren, die im Zuge der Quantisierung auftreten können, werden die Quantisierungsstufen ungleichmäßig getrennt. Es wird eine logarithmische Skalierung benutzt, die verhältnismäßig viele Bits für kleinere Signalamplituden und wenig Bits für große Signalamplituden verwendet. Auf diese Weise ist der Fehler proportional zur Signalamplitude.

Zwei Quantisierungsmethoden werden üblicherweise benutzt: **μ -law** (in Nordamerika und Japan) und **A-law** (in Europa und der restlichen Welt). Beide Versionen sind im ITU-Standard G.711 festgelegt. Alternativ kann man sich diesen Prozess so vorstellen, dass der dynamische Bereich des Signals (oder das Verhältnis zwischen den größten und den kleinsten möglichen Werten) komprimiert wird, bevor es (gleichmäßig) quantisiert wird, und danach erweitert wird, wenn das analoge Signal wiederhergestellt wird. Dieses System aus Kompressor und Expander wird **Komander** genannt. Es ist auch möglich, die Abtastungen nach der Digitalisierung zu komprimieren, sodass deutlich weniger als 64 kbit/s erforderlich sind. Wir werden auf dieses Thema zurückkommen, wenn wir Audioanwendungen wie VoIP untersuchen.

Zeitmultiplexing

Das auf PCM basierende Zeitmultiplexing (TDM) wird eingesetzt, um mehrere Sprachanrufe über Verbindungsleitungen zu übertragen, indem alle 125 µs eine Abtastung von jedem Anruf gesendet wird. Als sich die digitale Übertragung als machbare Technologie abzeichnete, war ITU (damals noch unter dem Namen CCITT) nicht in der Lage, für PCM einen internationalen Standard herbeizuführen. Folglich gibt es heute in den verschiedenen Ländern mehrere untereinander nicht kompatible Verfahren.

Die in Nordamerika und Japan verwendete Methode trägt den Namen **T1-Träger** und ist in ► Abbildung 2.37 dargestellt. (Bei einer rein technischen Betrachtungsweise muss man unterscheiden, dass das Format als DS1 und der Träger als T1 bezeichnet wird. Wir halten uns hier aber an eine weitverbreitete Sprachregelung in der Branche und verzichten auf die Feinunterscheidung.) Der T1-Träger besteht aus 24 Sprachkanälen, die mehrfach genutzt werden. Normalerweise werden die Analogsignale reihum abgetastet. Jeder der 24 Kanäle darf reihum 8 Bits in den Ausgangstrom einfügen.

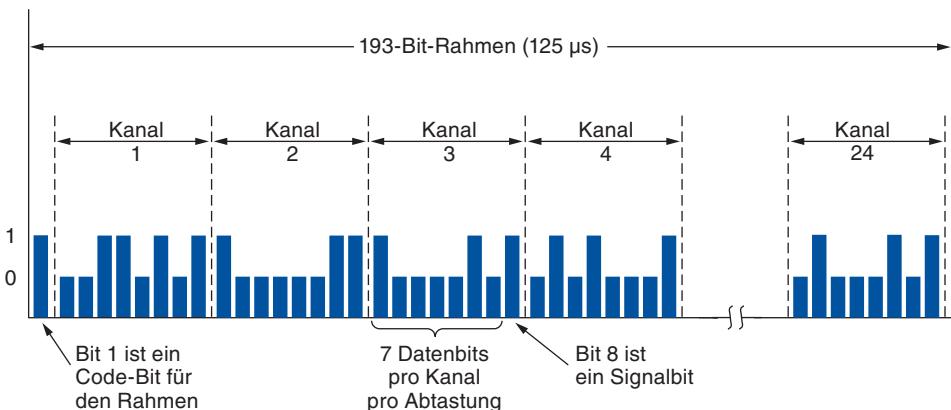


Abbildung 2.37: Der T1-Träger (1,544 Mbit/s).

Ein Rahmen besteht aus $24 \times 8 = 192$ Bit, zuzüglich einem Bit zu Steuerzwecken, sodass wir auf 193 Bit alle 125 µs kommen. Dies ergibt insgesamt eine Datenübertragungsrate von 1,544 Mbit/s, davon sind 8 kbit/s für die Signalübertragung. Das 193. Bit wird zur Rahmensynchronisation und Signalübertragung benutzt. Es gibt eine Variation, bei der das 193. Bit über einen Gruppe von 24 Rahmen (**erweiterter Superraum** genannt) verwendet wird. Sechs der Bits – in der 4., 8., 12., 16., 20. und 24. Position – nehmen abwechselnd das Muster 001011 an. Normalerweise prüft der Empfänger dieses Muster fortlaufend, um sicherzustellen, dass er noch synchron ist. Sechs weitere Bits werden benutzt, um einen Fehlerprüfcode zu senden, der den Empfänger bei der Bestätigung der Synchronisation unterstützt. Verliert er die Synchronisation, kann der Empfänger das Muster abtasten und den Fehlerprüfcode validieren, um sich erneut zu synchronisieren. Die verbleibenden Bits werden für Informationen zur Steuerung von Betrieb und Wartung des Netzes benutzt, wie beispielsweise Leistungsberichte vom entfernten Ende.

Das T1-Format hat mehrere Varianten. Die früheren Versionen senden Signalinformationen **bandintern**, das heißt in dem gleichen Kanal wie die Daten, indem einige der Datenbits benutzt werden. Dieser Entwurf ist eine Form von **kanalgebundener Zeichengabe** (*channel-associated signaling*), weil jeder Kanal seinen eigenen privaten Subkanal für die Signialisierung besitzt. Es gibt eine Konstellation, bei der das niedrigwertigste Bit einer 8-Bit-Abtastung von jedem Kanal in jedem sechsten Rahmen benutzt wird. Diese trägt die bildliche Bezeichnung **Robbed-Bit Signaling**. Die Idee dabei ist, dass ein paar gestohlene Bits für Sprachanrufe keine Rolle spielen. Niemand wird den Unterschied hören.

Bei Daten ist das jedoch eine andere Geschichte. Die falschen Bits zu senden ist – gelinde gesagt – nicht hilfreich. Falls ältere Versionen von T1 zur Datenübertragung benutzt werden, können nur 7 oder 8 Bits bzw. 56 kbit/s in jedem der 24 Kanäle verwendet werden. Neuere Versionen von T1 hingegen stellen freie Kanäle zur Verfügung, in denen alle Bits zum Senden von Daten benutzt werden können. Unternehmen, die eine T1-Leitung mieten, wollen genau diese freien Kanäle, wenn sie Daten über das Telefonnetz anstatt Sprachmuster senden. Die Signalgebung für Sprach-

anrufe wird dann **bandextern** verarbeitet, das heißt in einem von den Daten getrennten Kanal. Häufig wird die Signalisierung mit **Zentralkanal-Zeichengabe** (*common-channel signaling*) durchgeführt, bei der es einen gemeinsamen Kanal zur Signalgebung gibt. Einer der 24 Kanäle kann für diesen Zweck eingesetzt werden.

Außerhalb von Nordamerika und Japan ist der **E1**-Träger mit 2 048 Mbit/s im Einsatz. Dieser Träger hat 32 8-Bit-Datenabtastwerte, die in dem 125- μ s-Basisrahmen gepackt werden. Dreißig Kanäle werden für Informationen und bis zu zwei für die Signalübertragung benutzt. Jede Gruppe von vier Rahmen liefert 64 Signalisierungsbits. Die Hälfte davon wird zur Signalisierung benutzt (entweder kanalgebundene oder Zentralkanal-Zeichengabe), die andere Hälfte dient zur Rahmensynchronisation bzw. ist für die einzelnen Länder zur Belegung nach eigenem Ermessen reserviert.

Zeitmultiplexverfahren ermöglichen das Multiplexing mehrerer T1-Träger in höherwertige Träger. Wie das realisiert werden kann, ist aus ▶ Abbildung 2.38 ersichtlich. Auf der linken Seite werden vier T1-Kanäle auf einen T2-Kanal mehrfach genutzt. Das Multiplexing auf T2 und darüber erfolgt bitweise und nicht byteweise in den 24 Sprachkanälen, aus denen ein T1-Rahmen besteht. Vier T1-Ströme mit 1,544 Mbit/s sollten 6,176 Mbit/s erzeugen, aber T2 enthält tatsächlich 6,312 Mbit/s. Die Zusatzbits werden für die Rahmenbildung und -wiederherstellung verwendet, wenn das Trägersignal sich verschiebt. T1 und T3 werden häufig im Endkundenbereich eingesetzt, T2 und T4 aber nur innerhalb des Telefonsystems verwendet, sodass sie nicht so bekannt sind.

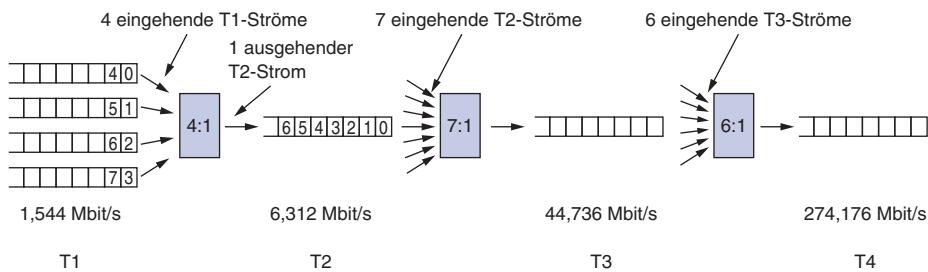


Abbildung 2.38: Multiplexing von T1-Strömen auf höherwertige Träger.

Auf der nächsten Ebene werden sieben T2-Ströme bitweise zusammengefasst, um einen T3-Strom zu bilden. Dann werden sechs T3-Ströme zu einem T4-Strom zusammengefasst. In jedem Schritt wird etwas Overhead für die Rahmenbildung und Wiederherstellung hinzugefügt, falls die Synchronisation zwischen Sender und Empfänger verloren geht.

Ebenso wie es kaum Vereinbarungen über den Basisträger zwischen den USA und dem Rest der Welt gibt, besteht auch keine Vereinbarung darüber, wie das Multiplexing in Trägern höherer Bandbreite erfolgen soll. Die in USA verbreitete Methode der 4er-, 6er- und 7er-Schritte konnte niemanden als künftige Lösung begeistern, und der ITU-Standard definiert Multiplexing auf jeder Ebene von vier Strömen in einen Strom. Außerdem unterscheiden sich die Daten für die Rahmenbildung und Wiederherstellung bei dem US-amerikanischen und den ITU-Standards. Die ITU-Hierarchie für 32,

128, 512, 2 048 und 8 192 Kanäle läuft in Geschwindigkeiten von 2,048, 8,848, 34,304, 139,264 und 565,148 Mbit/s.

SONET/SDH

In den Anfängen der Glasfasertechnik hatte jede Telefongesellschaft ihr eigenes optisches TDM-System. Nachdem AT&T 1984 aufgeteilt wurde, mussten sich die örtlichen Telefongesellschaften an mehrere Fernnetzbetreiber anschließen, die alle mit unterschiedlichen optischen TDM-Systemen arbeiteten. Somit entstand ein dringender Bedarf an Standardisierung. 1985 begann Bellcore, die Forschungseinrichtung von RBOC, mit der Ausarbeitung eines Standards namens **SONET** (*Synchronous Optical NETwork*).

Später stieg die ITU mit ein. Die gemeinsamen Anstrengungen führten 1989 zum SONET-Standard und einer Menge paralleler ITU-Empfehlungen (G.707, G.708 und G.709). Die ITU-Empfehlungen werden als **SDH** (*Synchronous Digital Hierarchy*) bezeichnet, unterscheiden sich aber nur in Kleinigkeiten von SONET. Praktisch der gesamte Telefonfernverkehr in den USA und in großen Teilen der Welt basiert heutzutage auf Verbindungsleitungen mit SONET auf der Bitübertragungsschicht. Weiterführende Informationen zu SONET finden Sie in Bellamy (2000), Goralski (2002) und Shepard (2001).

Das SONET-Design hatte vier wichtige Ziele. Erstens sollte SONET die Zusammenarbeit verschiedener Netzbetreiber ermöglichen. Um dieses Ziel zu erreichen, musste ein gemeinsamer Signalübertragungsstandard in Bezug auf Wellenlänge, Zeitverhalten, Rahmenstruktur usw. definiert werden.

Zweitens sollten die digitalen Systeme der USA, Europas und Japans vereinheitlicht werden, die alle auf PCM-Kanälen mit 64 kbit/s basierten, diese Kanäle aber auf unterschiedliche (nicht kompatible) Weise zusammenfassten.

Drittens sollte SONET eine Möglichkeit bieten, mehrere digitale Kanäle mehrfach zu nutzen. Als SONET entwickelt wurde, war die höchste Geschwindigkeit in den USA bei T3 44,736 Mbit/s. T4 gab es zwar, wurde aber nicht häufig verwendet, und oberhalb der Geschwindigkeit von T4 war nichts mehr definiert. Eine Aufgabe von SONET war, die Hierarchie auf Gbit/s und darüber hinaus fortzusetzen. Deshalb war auch ein Standard erforderlich, um langsamere Kanäle in einen SONET-Kanal mehrfach zu nutzen.

Viertens sollte SONET Unterstützung für Betrieb, Verwaltung und Wartung (*OAM – Operations, Administration and Management*) bereitstellen, was zum Management des Netzes notwendig ist. Bei den bisherigen Systemen waren diese Aufgaben nicht optimal gelöst worden.

Einer frühen Entscheidung zufolge sollte SONET ein herkömmliches TDM-System werden, bei dem die gesamte Bandbreite der Glasfaser einem Kanal zugeordnet wurde, welcher Zeitscheiben für die verschiedenen Unterkanäle besaß. So gesehen ist SONET ein synchrones System. Sender und Empfänger sind an einen gemeinsamen Taktgeber angeschlossen. Dieser Taktgeber, der das System steuert, arbeitet mit einer Genauigkeit von 1 zu 10⁹.

Der SONET-Basisrahmen ist ein Block von 810 Byte, der alle 125 µs gesendet wird. Da SONET synchron ist, werden Rahmen unabhängig davon ausgegeben, ob es Nutzdaten zu senden gibt oder nicht. Die 8 000 Rahmen pro Sekunde passen genau zur Abtastrate von PCM-Kanälen, die in allen digitalen Telefonsystemen eingesetzt werden.

Die 810 Byte großen SONET-Rahmen kann man sich als Tabelle von Bytes vorstellen, die 90 Spalten breit und 9 Zeilen hoch ist. Somit werden $8 \times 810 = 6\,480$ Bit 8 000-mal pro Sekunde übertragen. Die gesamte Datenübertragungsrate beträgt 51,84 Mbit/s. Dieses Layout ist der SONET-Basiskanal, der **STS-1** (*Synchronous Transport Signal-1*) heißt. Alle SONET-Verbindungsleitungen sind Vielfache von STS-1.

Die ersten drei Spalten eines jeden Rahmens sind für Systemverwaltungsinformationen reserviert, wie in ▶ Abbildung 2.39 dargestellt. In diesem Block enthalten die ersten drei Zeilen den Abschnittsoverhead, die nächsten sechs den Leitungsoverhead. Der Abschnittsoverhead wird am Anfang und Ende eines jeden Abschnitts erzeugt und geprüft, während der Leitungsoverhead am Anfang und Ende jeder Leitung erzeugt und geprüft wird.

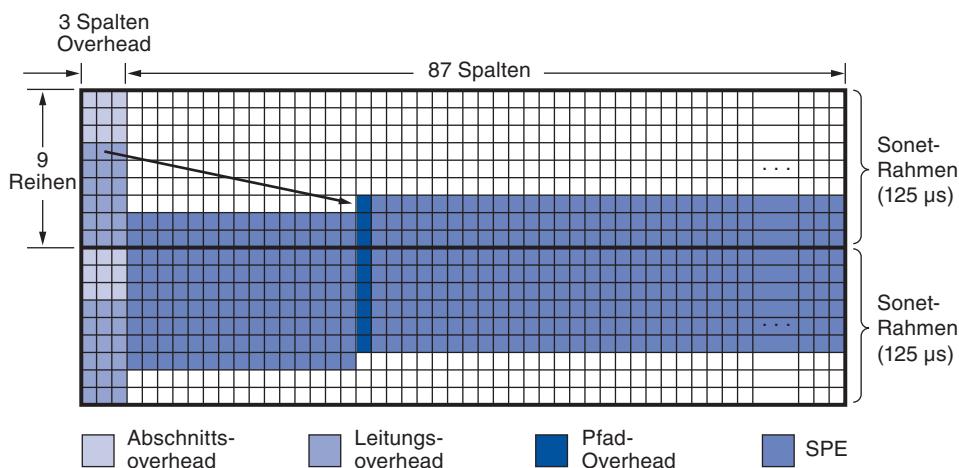


Abbildung 2.39: Zwei unmittelbar aufeinanderfolgende SONET-Rahmen.

Ein SONET-Sender sendet unmittelbar nacheinander und ohne einen Abstand dazwischen 810-Byte-Rahmen, selbst wenn keine Daten darin enthalten sind (in diesem Fall werden Leerdaten gesendet). Für den Empfänger stellt sich dies als ein fortlaufender Bitstrom dar. Wie soll er also wissen, wo ein Rahmen beginnt? Die Antwort lautet, dass die ersten beiden Bytes eines Rahmens ein festes Muster enthalten, nach dem der Empfänger sucht. Wenn er dieses Muster an der gleichen Stelle einer großen Anzahl aufeinanderfolgender Rahmen findet, geht er davon aus, dass er mit dem Sender synchronisiert ist. Theoretisch könnte ein Benutzer dieses Muster auch regulär in die Nutzdaten einfügen, aber in der Praxis ist dies nicht möglich. Dies liegt unter anderem daran, dass Multiplexing auf mehrere Benutzer im gleichen Rahmen angewandt wird.

Die verbleibenden 87 Spalten fassen $879 \times 8 \times 8 = 50,112$ Mbit/s für Nutzdaten. Diese Nutzdaten können Sprachabtastungen – von T1- und anderen Trägern im Ganzen übernommen – oder Pakete sein. SONET ist einfach ein bequemer Behälter zum Transportieren von Bits. Der **SPE**-Block (*Synchronous Payload Envelope*), welches die Nutzdaten trägt, beginnt nicht immer in Zeile 1, Spalte 4. Der SPE-Block kann an beliebiger Stelle innerhalb des Rahmens beginnen. Ein Zeiger auf das erste Byte ist in der ersten Zeile des Leitungsoverheads enthalten. Die erste Spalte des SPE-Blocks ist der Pfad-Overhead (d.h. Header für das Ende-zu-Ende-Pfad-Teilschichtprotokoll).

Da SPEs an einer beliebigen Stelle im SONET-Rahmen beginnen und sich auch auf zwei Rahmen erstrecken können, erhöht dies die Flexibilität des Systems (siehe Abbildung 2.39). Wenn beispielsweise Nutzdaten an der Quelle ankommen, während ein SONET-Leerrahmen gebildet wird, können diese in den aktuellen Rahmen eingefügt werden und müssen nicht bis zum Start des nächsten zurückgehalten werden.

Die Multiplex-Hierarchie von SONET/SDH wird in ►Abbildung 2.40 dargestellt. Es wurden Übertragungsarten von STS-1 bis STS-768 festgelegt, angefangen von Raten, die ungefähr einer T3-Leitung entsprechen, bis zu 40 Gbit/s. Sicherlich werden sogar noch höhere Raten im Laufe der Zeit definiert, wobei OC-3072 mit 160 Gbit/s als Nächstes ansteht – falls und wenn dies technisch möglich sein wird. Der optische Träger, der STS-*n* entspricht, wird als OC-*n* bezeichnet. Diese entsprechen sich ganz genau, bis auf eine geringe Neuordnung der Bits, die zur Synchronisation erforderlich ist. Die Bezeichnungen von SDH sind verschieden; sie beginnen bei OC-3, da ITU-basierte Systeme keine Übertragungsrate nahe 51,84 Mbit/s aufweisen. Wir haben die üblichen Raten angegeben, die ausgehend von OC-3 als Vielfache von vier auftreten. Die Gesamtdatenrate beinhaltet den Overhead. Die SPE-Datenübertragungsrate enthält keinen Leitungs- und Abschnittsoverhead. Die Benutzerdatenübertragungsrate enthält keinerlei Overhead und verfügt nur über 87 Nutzdatenspalten.

| SONET | | SDH | Datenübertragungsrate (Mbit/s) | | |
|------------|---------|---------|--------------------------------|-----------|-----------|
| Elektrisch | Optisch | Optisch | Insgesamt | SPE | Benutzer |
| STS-1 | OC-1 | | 51,84 | 50,112 | 49,536 |
| STS-3 | OC-3 | STM-1 | 155,52 | 150,336 | 148,608 |
| STS-12 | OC-12 | STM-4 | 622,08 | 601,344 | 594,432 |
| STS-48 | OC-48 | STM-16 | 2488,32 | 2405,376 | 2377,728 |
| STS-192 | OC-192 | STM-64 | 9953,28 | 9621,504 | 9510,912 |
| STS-768 | OC-768 | STM-256 | 39813,12 | 38486,016 | 38043,648 |

Abbildung 2.40: Multiplexraten von SONET und SDH.

Wird ein Träger wie OC-3 nicht mehrfach genutzt, sondern befördert nur die Daten von einer Quelle, wird der Buchstabe *c* (Abkürzung für *concatenated*, verkettet) an die

Bezeichnung angehängt. OC-3 bezeichnet einen Träger mit 155,52 Mbit/s, der aus drei getrennten OC-1-Trägern besteht, während OC-3c bedeutet, dass es sich um einen Datenstrom von einer einzigen Quelle mit 155,52 Mbit/s handelt. Die drei OC-1-Ströme in einem OC-3c-Strom werden nach Spalten verschachtelt – 1. Spalte von Strom 1, dann 1. Spalte von Strom 2, dann 1. Spalte von Strom 3, gefolgt von 2. Spalte von Strom 1 usw. –, sodass man am Ende einen Rahmen mit 270 Spalten und 9 Zeilen erhält.

Wellenlängenmultiplexing

Eine Form von Frequenzmultiplexing wird ebenso wie TDM angewandt, um sich die enorme Bandbreite der Glasfaserkanäle zunutze zu machen: **Wellenlängenmultiplexing (WDM, Wavelength Division Multiplexing)**. Das Grundprinzip beim Einsatz von WDM bei Glasfasern ist in ► Abbildung 2.41 aufgezeigt. Hier laufen vier Glasfasern in einem optischen Kombinator zusammen, wobei jede Glasfaser eine andere Wellenlänge verwendet. Die vier Bündel werden zu einer gemeinsamen Glasfaser zusammengefasst, um die Signale an ein entferntes Ziel zu übertragen. Am Ziel wird das Bündel auf die ursprüngliche Anzahl von Glasfasern wieder aufgeteilt. Jede Ausgangsglasfaser enthält einen kurzen, besonders dafür entwickelten Kern, der alle Wellenlängen bis auf die eine richtige ausfiltert. Die daraus resultierenden Signale können an das Ziel weitergeleitet oder auf verschiedenen Arten für einen Multiplextransport neu kombiniert werden.

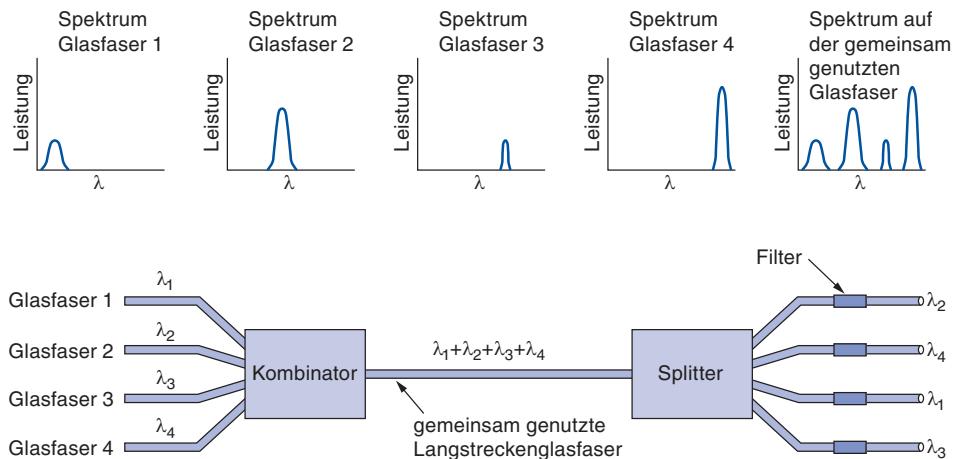


Abbildung 2.41: Wellenlängenmultiplexing.

An diesem Aufbau ist eigentlich nichts Neues. Diese Betriebsart ist nur ein Frequenzmultiplexing auf sehr hohen Frequenzen, wobei sich der Begriff WDM darauf bezieht, dass Glasfaserkanäle durch ihre Wellenlänge oder „Farbe“ anstatt durch ihre Frequenz beschrieben werden. Solange jeder Kanal seinen eigenen Frequenzbereich (d.h. Wellenlängenbereich) hat und alle Bereiche getrennt sind, können sie in der Langstreckenglasfaser mehrfach genutzt werden. Das Verfahren unterscheidet sich vom elektrischen FDM lediglich dadurch, dass ein optisches System mit einem Beugungsgitter völlig passiv und damit absolut zuverlässig ist.

Die Beliebtheit von WDM ist darauf zurückzuführen, dass die Energie in einem einzelnen Kanal normalerweise nur ein paar Gigahertz breit ist, denn dies ist die derzeitige Grenze dafür, wie schnell zwischen elektrischen und optischen Signalen konvertiert werden kann. Indem viele Kanäle parallel auf verschiedenen Wellenlängen betrieben werden, erhöht sich die gesamte Bandbreite linear mit der Anzahl der Kanäle. Da die Bandbreite eines einzelnen Glasfaserbandes etwa 25 000 GHz beträgt (siehe Abbildung 2.7), ist theoretisch selbst bei 1 Bit/Hz (auch höhere Übertragungsraten sind möglich) Platz für 2 500 10-Gbit/s-Kanäle vorhanden.

Die WDM-Technologie hat sich in einer Geschwindigkeit weiterentwickelt, welche die der Computertechnologie verblassen lässt. WDM wurde um das Jahr 1990 erfunden. Die ersten kommerziellen Systeme verfügten über acht Kanäle mit 2,5 Gbit/s pro Kanal. 1998 waren Systeme mit 40 Kanälen mit 2,5 Gbit/s auf dem Markt. 2006 gab es Modelle mit 192 Kanälen mit 10 Gbit/s und 64 Kanäle mit 40 Gbit/s, was potenziell bis auf 2,56 Tbit/s gesteigert werden kann. Diese Bandbreite reicht aus, um pro Sekunde 80 DVD-Filme in voller Länge zu übertragen. Die Kanäle sind auf Glasfaser ebenfalls dicht gepackt, wobei 200, 100 oder sogar weniger als 50 GHz für die Trennung benutzt werden. Großspurige Technologiedemonstrationen von Unternehmen haben das 10-Fache dieser Kapazität im Labor gezeigt, doch der Weg vom Labor in die Praxis dauert in der Regel mindestens ein paar Jahre. Ist die Anzahl der Kanäle sehr groß und liegen die Wellenlängen eng beieinander, dann wird das System als **DWDM** (*Dense WDM*, dichtes Wellenlängenmultiplexing) bezeichnet.

Einer der Treiber der WDM-Technologie ist die Entwicklung von optischen Verstärkern. Bisher mussten sämtliche Kanäle alle 100 km aufgeteilt und jeder in ein elektrisches Signal konvertiert werden, um einzeln verstärkt zu werden, bevor man sie wieder in optische Signale zurückverwandelt und die einzelnen Signale wieder zusammenführte. Heutzutage können optische Verstärker das gesamte Signal alle 1 000 km auffrischen, ohne dass hierzu mehrere optisch-elektrische Konvertierungen erforderlich sind.

Das in Abbildung 2.41 gezeigte Beispiel ist ein System mit fester Wellenlänge. Bits von der Eingangsglasfaser 1 gehen zur Ausgangsglasfaser 3, Bits von der Eingangsglasfaser 2 gehen zur Ausgangsglasfaser 1 usw. Es besteht aber die Möglichkeit, WDM-Systeme zu errichten, die im optischen Bereich mit Vermittlung (*switching*) arbeiten. Bei einem solchen Gerät lassen sich die optischen Filter mit Fabry-Perot- oder Mach-Zehnder-Interferometern einstellen. Diese Geräte ermöglichen die dynamische Veränderung der gewählten Frequenzen mithilfe eines Steuercomputers. Dadurch entsteht ein hohes Maß an Flexibilität, sodass viele unterschiedliche Wellenlängenpfade durch das Telefonnetz aus einer festen Menge von Glasfasern versorgt werden. Weitere Informationen zu optischen Netzen und WDM finden Sie bei Ramaswamit et al. (2009).

2.6.5 Vermittlung

Aus Sicht des normalen Fernmeldetechnikers ist das Telefonsystem in zwei Teile geteilt: Außenanlagen (Teilnehmeranschlussleitungen und Verbindungsleitungen, die sich außerhalb der Vermittlungsstellen befinden) und Innenanlagen (die Switches, also

Vermittlungsgeräte), die sich in den Vermittlungsstellen befinden. Wir haben bislang immer den äußeren Bereich betrachtet. Betrachten wir nun einmal die Sache von innen.

Heutzutage werden zwei verschiedene Vermittlungstechniken für das Netzwerk eingesetzt: Leitungsvermittlung und Paketvermittlung. Das traditionelle Telefonsystem basiert auf Leitungsvermittlung, doch mit dem Aufkommen der VoIP-Technologie holt die Paketvermittlung auf. Wir werden uns Leitungsvermittlung detailliert ansehen und mit Paketvermittlung vergleichen. Beide Vermittlungsarten sind so wichtig, dass wir darauf noch einmal zurückkommen, wenn wir die Vermittlungsschicht behandeln.

Leitungsvermittlung

Wenn Sie (oder Ihr Computer) eine Telefonverbindung aufbauen, suchen die Vermittlungseinrichtungen im Telefonsystem einen durchgehenden physikalischen Pfad von Ihrem Telefon zu dem des anderen Teilnehmers. Diese Technik heißt **Leitungsvermittlung** (*circuit switching*). Sie ist in ▶ Abbildung 2.42 schematisch dargestellt. Jedes der sechs Rechtecke stellt eine Vermittlungsstelle des Betreibers (Teilnehmervermittlungsstelle, Fernvermittlungsstelle usw.) dar. In diesem Beispiel hat jedes Amt drei eingehende und drei ausgehende Leitungen. Wenn ein Anruf eine Vermittlungsstelle durchläuft, wird eine physikalische Verbindung zwischen der Leitung, auf der der Anruf ankommt, und einer Ausgangsleitung hergestellt. Das ist in der Abbildung durch die gestrichelten Linien dargestellt.

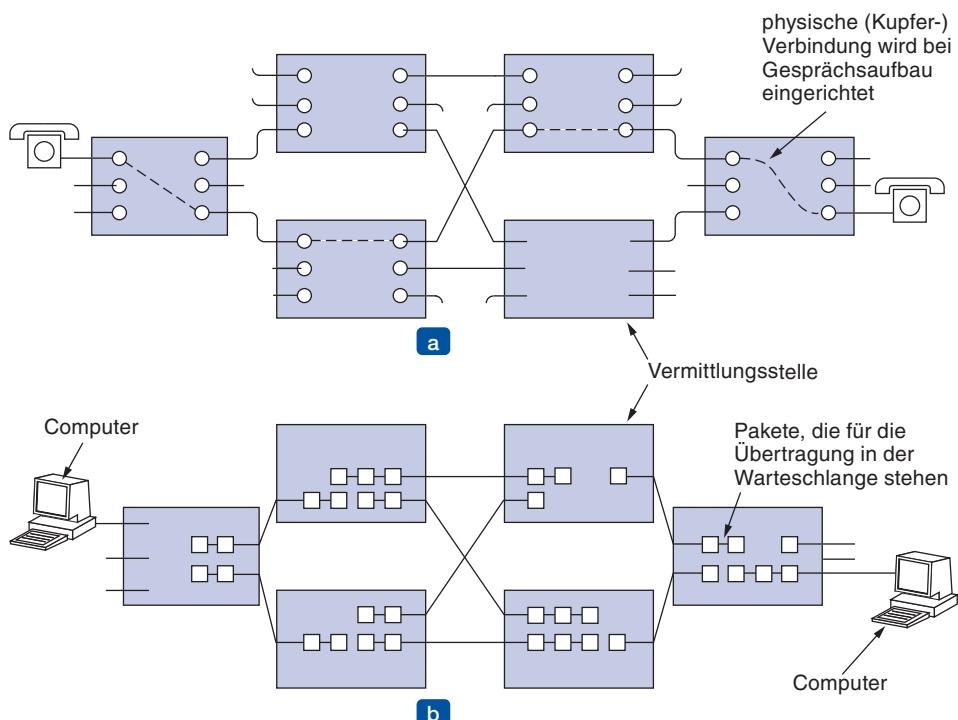


Abbildung 2.42: (a) Leitungsvermittlung. (b) Paketvermittlung.

In den Kindertagen des Telefons wurde die Verbindung durch das „Fräulein vom Amt“ hergestellt, das mit einem Verbindungsstecker die Eingangs- und Ausgangsbuchsen verband. Die Erfindung der automatischen Leitungsvermittlung hängt mit einer kuriosen Geschichte zusammen. Die Entdeckung machte ein Bestattungsunternehmer namens Almon B. Strowger im 19. Jahrhundert. Seit der Erfindung des Telefons konnten sich die Hinterbliebenen eines Verstorbenen über das städtische Telefonamt eine Verbindung zu einem Bestatter herstellen lassen. Zum Bedauern von Herrn Strowger gab es außer ihm in seiner Stadt noch ein anderes Bestattungsunternehmen. Wie es der Zufall wollte, war die Frau seines Konkurrenten die Dame vom Amt, die alle Telefonverbindungen der Stadt stöpselte. Er sah sich vor die Wahl gestellt, entweder eine automatische Telefonvermittlung zu erfinden oder sein Unternehmen dicht zu machen. Er entschied sich für die erste Möglichkeit. Nahezu 100 Jahre lang wurden die Leitungsvermittlungsgeräte als **Strowger-Schaltung** bezeichnet. (Aus der Überlieferung der Geschichte geht nicht hervor, ob die durch Strowgers Erfindung arbeitslos gewordene Dame eventuell in der Telefonauskunft weiterbeschäftigt wurde und Fragen wie „Wie lautet die Telefonnummer des Bestatters?“ beantwortete.)

Das Modell in ► Abbildung 2.42a ist natürlich stark vereinfacht, da Teile des physikalischen Weges zwischen zwei Telefonen Mikrowellen oder Glasfaserleitungen sein können, auf denen Tausende von Anrufen gebündelt werden. Dennoch gilt der Grundgedanke: Ist ein Anruf einmal eingerichtet, besteht ein dedizierter Weg zwischen den beiden Teilnehmern, bis der Anruf beendet wird.

Ein wichtiges Merkmal der Leitungsvermittlung ist die Notwendigkeit, einen Ende-zu-Ende-Pfad einzurichten, *bevor* Daten übertragen werden können. Die zwischen dem Ende des Wählvorgangs und dem Beginn des Klingelzeichens verstrichene Zeit kann gut zehn Sekunden betragen, bei Ferngesprächen sogar mehr. Während dieser Zeit sucht das Telefonsystem nach einem möglichen Pfad (► Abbildung 2.43a). Ebenfalls vor Beginn der Datenübertragung muss das Rufanforderungssignal den ganzen Weg bis zum Ziel gesendet und bestätigt werden. Bei vielen Computeranwendungen (z.B. Kreditrahmenüberprüfungen an Ladenkassen usw.) sind lange Aufbauzeiten nicht sehr erstrebenswert.

Eine Folge des reservierten Weges zwischen zwei Teilnehmern ist, dass nach erfolgreichem Einrichten der Verbindung als einzige Datenverzögerung nur die Ausbreitungszeit des elektromagnetischen Signals (ca. 5 ms je 1 000 km) anfällt. Bei einem vorab errichteten Übertragungsweg tritt daher kein Datenrückstau an – d.h., wenn der Anruf einmal durchgestellt wurde, erhalten Sie nie ein Besetztzeichen. Natürlich können Sie ein Besetztzeichen vor dem erfolgreichen Aufbau der Verbindung erhalten, wenn keine entsprechende Vermittlungs- oder Leitungskapazität vorhanden ist.

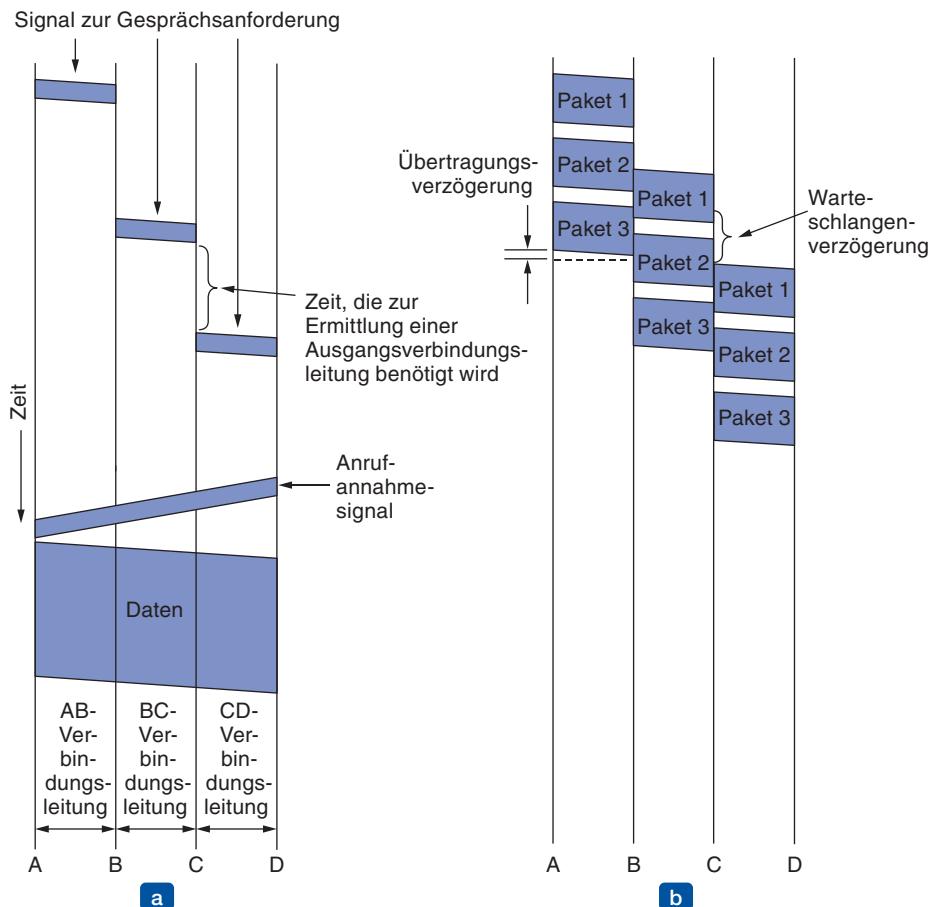


Abbildung 2.43: Zeitliches Verhalten der Ereignisse bei (a) Leitungsvermittlung, (b) Paketvermittlung.

Paketvermittlung

Die Alternative zur Leitungsvermittlung ist die **Paketvermittlung** (*packet switching*), die in ► Abbildung 2.42b dargestellt ist und in Kapitel 1 beschrieben wurde. Bei dieser Technologie werden einzelne Pakete gesendet, sobald sie verfügbar sind. Anders als bei der Leitungsvermittlung ist es hier nicht notwendig, einen dedizierten Pfad vorab einzurichten. Es ist die Aufgabe der Router, mittels Store-and-forward-Übertragungen jedes Paket für sich allein auf seinen Weg zum Ziel zu senden. Im Gegensatz dazu besteht bei der Leitungsvermittlung der Verbindungsaufbau darin, die Bandbreite auf dem gesamten Weg vom Sender zum Empfänger zu reservieren. Alle Daten auf der Leitung folgen diesem Pfad. Unter anderem bringt es diese Eigenschaft mit sich, dass die Daten immer in derselben Reihenfolge ankommen. Bei der Paketvermittlung gibt es keinen festgelegten Pfad, unterschiedliche Pakete können also verschiedenen Pfaden folgen, je nach Netzbedingung zum Zeitpunkt der Übermittlung, und die Daten können in einer anderen Reihenfolge ankommen.

Paketvermittelte Netzwerke legen eine strenge Obergrenze für die Paketgröße fest. Damit wird sichergestellt, dass kein Nutzer eine Übertragungsleitung sehr lange (z.B. viele Millisekunden) in Beschlag nehmen kann, sodass paketvermittelte Netze interaktiven Datenverkehr verarbeiten können. Außerdem wird dadurch die Verzögerungszeit verringert, da das erste Paket einer langen Nachricht weitergeleitet werden kann, bevor das zweite Paket vollständig angekommen ist. Die Store-and-forward-Verzögerung, die entsteht, wenn ein Paket im Speicher des Routers zwischengespeichert wird, bevor es zum nächsten Router weitergeschickt wird, ist jedoch größer als die Verzögerung bei der Leitungsvermittlung (► Abbildung 2.43b). Dort fließen die Bits einfach kontinuierlich durch die Leitung.

Paket- und Leitungsvermittlung unterscheiden sich noch auf andere Arten. Da bei der Paketvermittlung keine Bandbreite reserviert wird, müssen Pakete unter Umständen warten, bis sie weitergeleitet werden. Dies führt zu **Verzögerungen durch Warteschlangenbildung** (*queuing delay*) und Netzüberlastung, falls viele Pakete gleichzeitig gesendet werden. Auf der anderen Seite besteht keine Gefahr, ein Besetzzeichen zu bekommen und somit das Netz nicht nutzen zu können. Netzüberlastung tritt also zu unterschiedlichen Zeitpunkten auf: bei Leitungsvermittlung während der Aufbauzeit und bei Paketvermittlung, wenn Pakete gesendet werden.

Wenn eine Leitung für einen bestimmten Benutzer reserviert wurde und es keinen Datenverkehr gibt, wird dessen Bandbreite verschwendet; sie kann nicht für anderen Datenverkehr verwendet werden. Paketvermittlung verschwendet keine Bandbreite und ist somit aus einer Systemperspektive betrachtet effizienter. Um den Unterschied zwischen Leitungs- und Paketvermittlung nachzuvollziehen, ist es ganz wesentlich, diesen Zielkonflikt zu verstehen: Er besteht zwischen garantierter Service und der Verschwendungen von Ressourcen auf der einen Seite und nicht garantierter Service und keine Ressourcenverschwendungen auf der anderen Seite.

Die Paketvermittlung ist fehlertoleranter als die Leitungsvermittlung. Dies war der eigentliche Grund, warum sie erfunden wurde. Fällt ein Vermittlungselement aus, werden alle Verbindungen, die es gerade nutzen, beendet und es können keine Daten mehr darüber übertragen werden. Bei der Paketvermittlung können Pakete um ausgefallene Vermittlungselemente herum gesendet werden.

Ein letztes Unterscheidungsmerkmal ist die Gebührenberechnung. Bei der Leitungsvermittlung werden die Gebühren seit Einführung des Telefons nach Entfernung und Zeit berechnet. Bei Mobiltelefonen spielt die Entfernung in der Regel keine Rolle, außer bei internationalen Ferngesprächen. Zeit spielt auch nur eine untergeordnete Rolle (z.B. kostet ein Tarif mit 2 000 Freiminuten mehr als einer mit 1 000 Freiminuten und manchmal sind Abend- und Wochenendgespräche billig). Bei der Paketvermittlung spielt die Verbindungszeit keine Rolle, in manchen Fällen aber das Volumen des Datenverkehrs. Bei privaten Nutzern berechnen ISPs in der Regel eine monatliche Flatrate, da dies für sie und ihre Kunden weniger Arbeit bedeutet und das Abrechnungsmodell für ihre Kunden verständlich ist. Die Backbonebetreiber stellen die regionalen Netze jedoch nach dem anfallenden Volumen des Datenverkehrs in Rechnung.

Die Unterschiede werden in ▶ Abbildung 2.44 aufgeführt. Traditionell haben Telefonnetze die Leitungsvermittlung benutzt, um Telefongespräche mit hoher Qualität zu bieten, und Rechnernetze haben Paketvermittlung aus Gründen der Einfachheit und Effizienz verwendet. Es gibt jedoch auch bemerkenswerte Ausnahmen. Einige ältere Rechnernetze haben unter der Oberfläche Leitungsvermittlung eingesetzt (z.B. X25) und einige neuere Telefonnetze benutzen Paketvermittlung bei VoIP. Dies sieht nach außen für den Anwender ganz wie ein normaler Telefonanruf aus, aber innerhalb des Netzes werden die Sprachdaten in Paketen weitervermittelt. Dieser Ansatz hat es möglich gemacht, dass Emporkömmlinge im Telekommunikationsgeschäft billige internationale Anrufe über Telefonkarten verkaufen, auch wenn diese vielleicht eine schlechtere Gesprächsqualität als solche von etablierten Unternehmen aufweisen.

| Eigenschaft | Leitungsvermittlung | Paketvermittlung |
|---|------------------------|------------------|
| Gesprächsaufbau | Notwendig | Nicht notwendig |
| Dedizierter physischer Pfad | Ja | Nein |
| Alle Pakete verlaufen auf der gleichen Route | Ja | Nein |
| Pakete kommen der Reihe nach an | Ja | Nein |
| Ausfall einer Vermittlungsstelle ist fatal | Ja | Nein |
| Verfügbare Bandbreite | Fest | Dynamisch |
| Zeitpunkt möglicher Überlastung | Beim Verbindungsaufbau | Bei jedem Paket |
| Potenziell verschwendete Bandbreite | Ja | Nein |
| Übertragung mit Zwischenspeicherung (Store-and-forward) | Nein | Ja |
| Gebührenberechnung | Pro Minute | Pro Paket |

Abbildung 2.44: Vergleich von leitungsvermittelten und paketvermittelten Netzen.

2.7 Das Mobilfunksystem

Das herkömmliche Telefonnetz ist (auch wenn es eines Tages Glasfasern mit mehreren Gigabit zwischen zwei Teilnehmerstellen bereitstellt) nicht in der Lage, die Anforderungen der Benutzergruppe mit der größten Zuwachsrate zu erfüllen: der mobilen Benutzer. Viele Leute möchten heutzutage in Flugzeugen, Autos, Swimmingpools und beim Joggen im Park telefonieren, ihre E-Mails lesen und im Web surfen. Somit ist das Interesse an der drahtlosen Telefonie sehr groß. Im Folgenden werden wir dieses Thema im Detail betrachten.

Das Mobilfunksystem wird für überregionale Sprach- und Datenkommunikation genutzt. **Mobiltelefone** (*mobile phone, cell phone*) haben drei unterschiedliche Generationen durchlaufen, die allgemein **1G**, **2G** und **3G** genannt werden. Die Generationen sind:

- 1.** Analoge Sprache
- 2.** Digitale Sprache
- 3.** Digitale Sprache und Daten (Internet, E-Mail etc.)

(Mobiltelefone sollten nicht mit **schnurlosen Telefonen** verwechselt werden, die aus einer Basisstation und einem Handapparat bestehen, das als Komplettsystem für die Nutzung im Haus verkauft wird. Sie werden nie für den Netzbetrieb verwendet, daher gehen wir hier nicht weiter darauf ein.)

Obwohl wir uns hier vor allem mit der Technologie dieser Systeme befassen, ist es interessant, welch großen Einfluss hier politische Entscheidungen wie auch winzige Marketingentscheidungen haben können. Das erste mobile System wurde in den Vereinigten Staaten von AT&T entwickelt und für das ganze Land von der FCC freigegeben. In der Folge war in den gesamten Vereinigten Staaten ein einziges (analoges) System im Einsatz und ein Mobiltelefon, das man in Kalifornien erworben hatte, funktionierte auch in New York. Als aber Mobiltelefone sich auch in Europa etablierten, entwickelte jedes Land sein eigenes System, was zu einem Fiasko führte.

Europa lernte aus dem Fehler und als das digitale Mobiltelefon vorgestellt wurde, taten sich die einzelnen Aufsichtsbehörden zusammen und erarbeiteten als einheitlichen Standard GSM, sodass nun ein europäisches Handy in ganz Europa funktioniert. Zu diesem Zeitpunkt hatte man in den Vereinigten Staaten beschlossen, dass die Standardisierung nicht Sache des Staates ist, und man überließ die Standardisierung der digitalen Handys dem Markt. Diese Entscheidung führte dazu, dass verschiedene Gerätehersteller verschiedene Arten von Mobiltelefonen produzierten. So wurden in den USA zwei größere – und vollständig inkompatible – digitale Mobilfunksysteme sowie weitere kleinere Systeme installiert.

Obwohl anfangs in den Vereinigten Staaten mehr Leute Handys besaßen, liegt nun Europa vorne. Ein Grund hierfür ist, dass ein einheitliches System existiert, das überall in Europa und mit jedem Provider funktioniert. Aber es gibt noch weitere Gründe. Ein zweiter Punkt, in dem sich die USA und Europa unterscheiden, ist die unscheinbare Sache mit den Telefonnummern. In den USA werden die Mobiltelefone nicht von den normalen (Festnetz-)Telefonen unterschieden. Daher kann ein Anrufer nicht erkennen, ob beispielsweise hinter (212) 234-5678 ein Festanschluss (ein günstiger oder kostenfreier Anruf) oder ein Mobiltelefon (ein teurer Anruf) steht. Damit die Leute nicht zu nervös wurden, wenn sie jemanden anrufen wollten, entschieden die Telefongesellschaften, dass die Besitzer des Mobiltelefons für die eingehenden Anrufe bezahlen müssen. In der Folge zögerten viele Leute bei der Anschaffung eines Handys, weil sie fürchteten, dass hier nur aufgrund der eingehenden Anrufe riesige Rechnungen auflaufen würden. In Europa haben Mobiltelefonnummern eine spezielle Vorwahl, sodass man sie sofort erkennen kann. So gilt in Europa die übliche Regel „der Anrufende zahlt“ auch für Mobiltelefone, außer bei internationalen Anrufen, wo die Kosten aufgeteilt werden.

Ein dritter Punkt, der für die Akzeptanz von Mobiltelefonen in Europa sehr wichtig ist, ist die weitverbreitete Nutzung von im Voraus bezahlten (*prepaid*) Handykarten (in manchen Gebieten sind dies bis zu 75 %). Sie können in vielen Läden ohne viel mehr Formalitäten als beim Kauf einer Digitalkamera erworben werden. Man bezahlt und kann telefonieren. Sie sind beispielsweise mit 20 oder 50 Euro aufgeladen und können unter Verwendung eines geheimen Pin-Codes wieder neu aufgeladen werden, wenn das Guthaben aufgebraucht ist. In der Folge hat in Europa praktisch jeder Teenager oder auch viele kleinere Kinder Mobiltelefone (in der Regel mit Handy-Guthabekarten), sodass ihre Eltern sie schnell finden können, ohne dass hierfür riesige Telefongebühren entstehen. Wenn ein Mobiltelefon nur gelegentlich genutzt wird, ist die Nutzung sehr billig bzw. ganz kostenlos, da keine Monatsgebühren oder Gebühren für eingehende Anrufe erhoben werden.

2.7.1 Die erste Generation der Mobiltelefone: Analoge Sprache

Wenden wir uns jetzt nach diesem kurzen Exkurs über Politik und Marketing wieder der Technologie zu. Wir beginnen mit den ersten Mobilfunksystemen. Mobile Funktelefone wurden in den ersten Jahrzehnten des 20. Jahrhunderts sporadisch in der Seefahrt, besonders der Marine, und im Militär eingesetzt. 1946 wurde in St. Louis das erste System für Autotelefone eingerichtet. Dieses System nutzte einen einzelnen großen Sender auf einem Hochhaus und hatte einen einzigen Kanal, der sowohl zum Senden als auch zum Empfangen diente. Um sprechen zu können, musste der Benutzer eine Taste drücken, durch die der Sender ein- und der Empfänger ausgeschaltet wurde. Solche Systeme nannte man **Push-to-Talk-Systeme**. Sie wurden Ende der 1950er Jahre in mehreren Städten der USA installiert. CB-Funk, Taxis und Streifenwagen der Polizei benutzten diese Technologie sehr oft.

In den 1960er Jahren wurde **IMTS** (*Improved Mobile Telephone System*) eingeführt. Es basierte ebenfalls auf einem leistungsstarken Sender (200 Watt), der auf einem Hügel oder Berg stand, aber zwei Frequenzen hatte – eine zum Senden und eine zum Empfangen. Somit war die Umschalttaste nicht mehr länger nötig. Da die gesamte Kommunikation von den mobilen Telefonen auf einem anderen Kanal einging als die ausgehenden Signale, konnten die mobilen Benutzer einander nicht hören (im Gegensatz zum Push-to-Talk-System in Taxis).

Das IMTS unterstützte 23 Kanäle, die sich über ein Band von 150 bis 450 MHz verteilten. Aufgrund der kleinen Zahl an Kanälen mussten die Benutzer oft lange Zeit warten, bis sie einen Freiton erhielten. Wegen seiner hohen Leistung musste der Sender einige Hundert Kilometer von anderen Sendern entfernt aufgestellt werden, um Störungen zu vermeiden. Insgesamt war das System aufgrund seiner begrenzten Kapazität unpraktisch.

AMPS – Advanced Mobile Phone System

Diese Situation änderte sich drastisch mit dem von Bell Labs entwickelten und erstmals in den USA 1982 installierten **AMPS** (*Advanced Mobile Phone System*). Dieses System wird auch in Großbritannien eingesetzt, wo es TACS, und in Japan, wo es

MCS-L1 heißtt. AMPS wurde 2008 formal in den Ruhestand geschickt, dennoch werden wir einen Blick darauf werfen, um die 2G- und 3G-Systeme im Zusammenhang zu verstehen, da diese eine Verbesserung von AMPS darstellen.

Bei allen Mobiltelefonsystemen wird ein geografisches Gebiet in **Zellen** unterteilt. Bei AMPS haben die Zellen in der Regel einen Durchmesser von 10 bis 20 km, bei digitalen Systemen sind die Zellen kleiner. Jede Zelle verwendet eine Menge von Frequenzen, die von keiner benachbarten Zelle verwendet werden. Das Grundkonzept, durch das Zellulärsysteme eine viel größere Kapazität haben als alle vorherigen Systeme, ist die Nutzung relativ kleiner Zellen und die Wiederverwendung von Übertragungsfrequenzen in nahe gelegenen (aber nicht nebeneinanderliegenden) Zellen. Während ein IMTS-System im Umkreis von 100 km nur einen Anruf auf einer Frequenz unterstützt, können bei einem AMPS-System mit hundert 10 km großen Zellen in einem Gebiet fünf bis zehn Anrufe auf jeder Frequenz in weit auseinanderliegenden Zellen stattfinden. Daher erhöht das zelluläre Design die Systemkapazität um mindestens eine Größenordnung oder mehr, wenn die Zellen kleiner werden. Außerdem bedeuten kleinere Zellen auch einen geringeren Leistungsbedarf, was zu kleineren und kostengünstigeren Geräten führt.

Das Konzept der Wiederholung von Frequenzen ist in ►Abbildung 2.45a dargestellt. Die Zellen sind normalerweise in etwa kreisförmig, können aber als Sechsecke leichter modelliert werden. In Abbildung 2.45a haben alle Zellen die gleiche Größe. Sie werden in Einheiten von je sieben Zellen gruppiert. Jeder Buchstabe bezeichnet eine Frequenzgruppe. Für jede Frequenzgruppe gibt es einen Puffer, der eine Breite von etwa zwei Zellen hat, in dem diese Frequenz nicht wiederverwendet wird, wodurch eine gute Trennung und Abschirmung gegen Störungen erreicht wird.

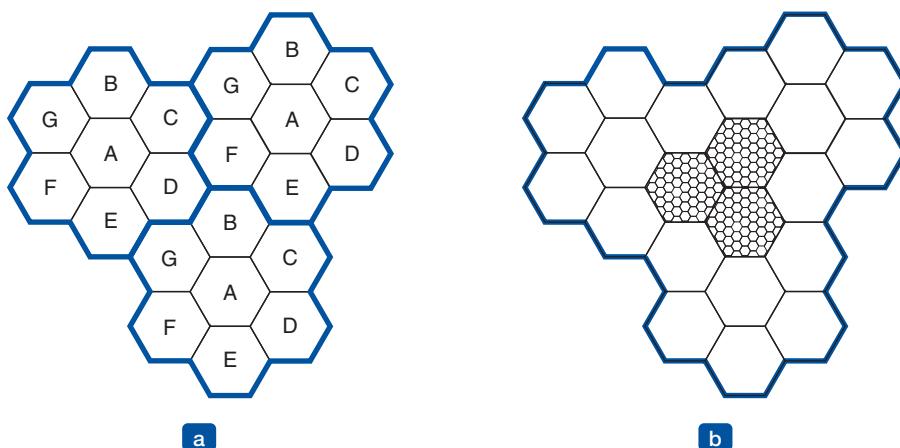


Abbildung 2.45: (a) Frequenzen werden in Nachbarzellen nicht wiederverwendet.
 (b) Um mehr Benutzer zu unterstützen, können kleinere Zellen verwendet werden.

Eine wichtige Frage ist das Finden von Standorten, die sehr hoch liegen, um die Antennen für die Basisstationen aufzustellen. Dieses Problem hat einige Telekommunikationsbetreiber bereits dazu geführt, mit der römisch-katholischen Kirche ein

Bündnis einzugehen, da diese Eigentümerin zahlreicher exponierter Standorte in aller Welt ist, die sich potenziell zur Aufstellung von Antennen gut eignen.

In einem Gebiet, in dem die Zahl der Benutzer bis zu einem Punkt angestiegen ist, an dem das System überlastet ist, wird die Leistung reduziert und die überlasteten Zellen werden in kleinere, sogenannte **Mikrozellen** aufgeteilt, damit die Frequenzwiederholung erhöht werden kann (siehe ► Abbildung 2.45b). Telefongesellschaften erzeugen manchmal temporäre Mikrozellen mithilfe von tragbaren Funktürmen mit Satellitenverbindungen bei Sportveranstaltungen, Rockkonzerten und an anderen Stellen, an denen sich für ein paar Stunden eine große Anzahl von Benutzern versammelt.

Im Zentrum jeder Zelle befindet sich eine Basisstation, an die alle Telefone innerhalb der Zelle übertragen. Die Basisstation besteht aus einem Computer und einem an eine Antenne angeschlossenen Sender/Empfänger. Bei einem kleinen System werden alle Basisstationen an ein Gerät namens **Mobilfunkvermittlungsstelle (MSC, Mobile Switching Center)** oder **MTSO, Mobile Telephone Switching Office**) angeschlossen. Bei größeren Systemen sind eventuell mehrere MSCs erforderlich, die alle an eine übergeordnete MSC angeschlossen werden usw. Die MSCs sind im Grunde Teilnehmervermittlungsstellen wie beim Telefonnetz. Sie sind auch mindestens an eine Teilnehmervermittlungsstelle eines Telefonnetzes angeschlossen. Die Funkvermittlungsstellen kommunizieren mit den Basisstationen, untereinander und mit dem öffentlichen Fernsprechnetz über ein paketvermitteltes Netz.

Jedes Mobiltelefon befindet sich logisch zu jedem bestimmten Zeitpunkt in einer bestimmten Zelle und unter der Kontrolle der Basisstation dieser Zelle. Verlässt ein Mobiltelefon eine Zelle, stellt deren Basisstation fest, dass sich das Signal des Telefons entfernt, und fordert alle benachbarten Basisstationen um Mitteilung auf, wie viel Leistung sie von ihm erhalten. Wenn die Antwort zurückkommt, übergibt die Basisstation die Zuständigkeit an die Zelle, die das stärkste Signal empfängt, d.h. meistens diejenige, in deren unmittelbarer Nähe sich das Telefon jetzt befindet. Anschließend wird das Telefon über seinen neuen Chef informiert. Wenn gerade über das Telefon telefoniert wird, so wird dieses aufgefordert, auf einen neuen Kanal zu wechseln (weil der alte von benachbarten Zellen nicht verwendet wird). Dieser Vorgang ist das sogenannte **Handover (Verbindungsübergabe)**, das etwa 300 ms dauert. Die Kanalzuweisung nimmt das MSC vor. Die Basisstationen sind im Grunde einfach stupide Richtfunkgeräte.

Kanäle

Das AMPS-System benutzt FDM, um die Kanäle zu trennen. Das System arbeitet mit 832 Vollduplexkanälen, die je aus einem Paar von Simplexkanälen bestehen. Diese Anordnung wird **FDD (Frequency Division Duplex)** genannt. Die 832 Simplexsendekanäle von 824 bis 849 MHz werden für Übertragungen vom Mobiltelefon zur Basisstation verwendet, die 832 Simplexempfangskanäle von 869 bis 894 MHz für Übertragungen von der Basisstation zum mobilen Gerät. Jeder dieser Simplexkanäle ist 30 kHz breit.

Die 832 Kanäle sind in vier Kategorien unterteilt. Steuerungskanäle (Basisstation zum Handy) werden zur Verwaltung des Systems benutzt. Paging-Kanäle (Basisstation zum

Handy) machen mobile Benutzer auf eingehende Anrufe aufmerksam. (Bidirektionale) Zugangskanäle werden für den Gesprächsaufbau und die Kanalzuweisung eingesetzt. Schließlich sind die (bidirektionale) Datenkanäle für die Übermittlung von Sprache, Fax oder Daten zuständig. Da Frequenzen in benachbarten Zellen nicht wiederholt werden können und 21 Kanäle in jeder Zelle für die Steuerung reserviert sind, ist die tatsächlich pro Zelle verfügbare Zahl an Sprachkanälen viel geringer als 832, normalerweise nur 45.

Rufverwaltung

Jedes Mobiltelefon im AMPS-System hat eine 32 Bit große Seriennummer und eine zehnstellige Telefonnummer im PROM (*Programmable Read-Only Memory*). Die Telefonnummer setzt sich aus einer dreistelligen Vorwahl von 10 Bit und einer siebensteligen Teilnehmernummer von 24 Bit zusammen. Beim Einschalten tastet das Mobiltelefon eine vorprogrammierte Liste mit 21 Steuerkanälen ab, um das stärkste Signal zu ermitteln. Dann sendet das Telefon seine 32 Bit große Seriennummer und die 34 Bit große Telefonnummer. Wie alle Steuerinformationen im AMPS-System wird auch dieses Paket digital mehrmals und mit einem Fehlerkorrekturcode versendet, obwohl die Sprachkanäle selbst analog sind.

Hört die Basisstation die Ankündigung, informiert sie das MSC, die das Vorhandensein eines neuen Kunden vermerkt, sowie das Heimat-MSC des Kunden über dessen derzeitigen Standort. Im Normalbetrieb registriert sich das Mobiltelefon etwa alle 15 Minuten erneut.

Um einen Anruf zu tätigen, schaltet der Benutzer das Mobiltelefon ein, wählt die Nummer des gewünschten Teilnehmers und drückt die SENDE-Taste. Das Telefon sendet die gewählte Nummer und seine eigene Identität an den Zugriffskanal. Tritt eine Kollision auf, versucht es später noch einmal, die Verbindung herzustellen. Erhält die Basisstation die Anfrage, dann informiert sie das MSC. Wenn der Anrufer ein Kunde des MSC-Unternehmens (oder einer seiner Partner) ist, sucht das MSC nach einem freien Kanal für den Anruf. Ist einer vorhanden, so wird die Kanalnummer über den Steuerkanal zurückgeschickt. Das Mobiltelefon schaltet dann automatisch auf den gewählten Sprachkanal und wartet, bis der gerufene Teilnehmer abnimmt.

Ankommende Anrufe werden anders abgewickelt. Zunächst hören alle betriebsbereiten Telefone ständig den Paging-Kanal ab, um eventuell an sie gerichtete Nachrichten zu erkennen. Wird ein Mobiltelefon (entweder von einem Festnetztelefon oder einem anderen Mobiltelefon) angerufen, dann wird ein Paket an das Heimat-MSC des gerufenen Teilnehmers gesendet, um festzustellen, wo es sich befindet. Danach wird ein Paket an die Basisstation seiner momentanen Zelle gesendet, die dann auf dem Paging-Kanal einen Rundruf in der Form „Einheit 14, sind Sie da?“ überträgt. Das gerufene Telefon antwortet mit „Ja“ über den Zugangskanal. Die Basis meldet dann den Anruf etwa so: „Einheit 14, ein Anruf für Sie auf Kanal 3.“ An diesem Punkt schaltet das gerufene Telefon auf Kanal 3 und aktiviert das Klingelzeichen (oder eine Melodie, die der Besitzer als Geburtstagsgeschenk erhalten hat).

2.7.2 Die zweite Generation der Mobiltelefone: Digitale Sprache

Die erste Generation der Mobiltelefone war analog, die zweite ist digital. Der Wechsel zu digital hat mehrere Vorteile. Er bietet Kapazitätszuwachs, da Sprachsignale digitalisiert und komprimiert werden können. Die Sicherheit wird erhöht, weil die Verschlüsselung von Sprach- und Steuersignalen möglich ist. Damit werden umgekehrt Betrügereien und Lauschangriffe verhindert, und zwar sowohl das absichtliche Absuchen als auch die Echosignale von anderen Anrufen aufgrund der Funkwellenweiterleitung. Schließlich werden neue Dienste wie Textnachrichten ermöglicht.

Genau wie bei der ersten Generation erfolgte auch bei der zweiten keine internationale Standardisierung. Es wurden mehrere verschiedene Systeme entwickelt, drei davon werden weitverbreitet eingesetzt. **D-AMPS** (*Digital Advanced Mobile Phone System*) ist eine digitale Version von AMPS, die neben AMPS existiert und TDM benutzt, um mehrere Anrufe auf dem gleichen Frequenzkanal zu platzieren. D-AMPS wird im internationalen Standard IS-54 und dem Nachfolger IS-136 beschrieben. **GSM** (*Global System for Mobile communication*) hat sich als das vorherrschende System herauskristallisiert, und obwohl sich GSM in den USA nur langsam durchsetzte, wird es heute fast überall auf der Welt verwendet. Wie D-AMPS basiert es auf einer Mischung aus FDM und TDM. **Codemultiplexverfahren (CDMA)**, *Code Division Multiple Access*, beschrieben im internationalen Standard **IS-95**) ist eine völlig andere Systemart, die weder auf FDM noch auf TDM beruht. Obwohl CDMA nicht zum vorherrschenden 2G-System wurde, so ist es doch die Grundlage für 3G-Systeme geworden.

Auch der Name **PCS** (*Personal Communications Services*) wird manchmal in der Marketingliteratur verwendet, um ein (digitales) System der zweiten Generation zu bezeichnen. Ursprünglich war damit ein Mobiltelefon mit einem 1 900-MHz-Band gemeint, aber diese Unterscheidung wird heute nur noch selten getroffen.

Im Folgenden beschreiben wir GSM, da es das vorherrschende 2G-System ist. Im nächsten Abschnitt werden uns CDMA detaillierter ansehen, wenn wir uns den 3G-Systemen zuwenden.

GSM – Global System for Mobile Communications

GSM startete in den 1980er Jahren als ein Versuch, einen einheitlichen europäischen 2G-Standard hervorzu bringen. Diese Aufgabe wurde an eine Telekommunikationsgruppe delegiert, die den französischen Namen „Groupe Spécial Mobile“ trug. Die ersten GSM-Systeme wurden ab 1991 installiert und waren schnell erfolgreich. Es wurde bald klar, dass GSM auf dem Weg war, mehr als nur ein europäischer Erfolg zu werden, sondern sich auf weit entfernte Länder wie Australien ausdehnte. Deshalb wurde der Standard in GSM umbenannt, um weltweit Anklang zu finden.

GSM und die anderen Mobilfunksysteme, die wir untersuchen, haben von 1G-Systemen einen Entwurf übernommen, der auf Zellen, der Frequenzwiederholung zwischen Zellen und der Mobilität mithilfe von Handovers bei Fortbewegung des Teilnehmers basiert. Die wichtigsten Eigenschaften von GSM werden im folgenden Abschnitt kurz beschrieben. Der GSM-Standard ist über 5 000 (!) Seiten lang. Ein

Großteil dieses Materials bezieht sich auf technische Aspekte des Systems, insbesondere das Design von Empfängern zur Handhabung der Signalausbreitung über mehrere Wege und der Synchronisation von Sendern und Empfängern. All das werden wir hier nicht einmal erwähnen.

► Abbildung 2.46 zeigt, dass die GSM-Architektur der AMPS-Architektur ähnelt, auch wenn die Komponenten unterschiedliche Namen haben. Das Handy selbst wird aufgeteilt in den Handapparat und einen entfernabaren Chip mit Teilnehmer- und Kontoinformationen, **SIM-Karte** genannt (kurz für *Subscriber Identity Module*). Die SIM-Karte aktiviert den Handapparat und sie enthält Geheimnisse, die zur Identifizierung zwischen Handy und Netzwerk und zur Verschlüsselung von Gesprächen benötigt werden. Eine SIM-Karte kann herausgenommen und in einen anderen Apparat eingesetzt werden – und verwandelt so diesen Apparat in Ihr Handy, jedenfalls soweit es das Netz betrifft.

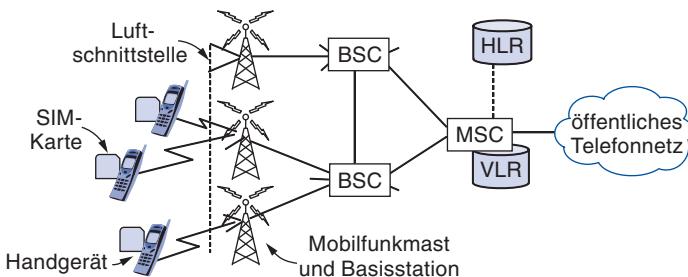


Abbildung 2.46: Architektur des GSM-Funknetzes.

Das Handy spricht zur Zellbasisstation über eine **Luftschnittstelle** (*air interface*), die wir gleich beschreiben werden. Die Zellbasisstationen sind jeweils mit einer **BSC** (*Base Station Controller*) verbunden, die die Funkwellenressourcen der Zellen steuert und die Handovers durchführt. Das BSC wiederum ist (wie bei AMPS) an ein MSC angebunden, welches Anrufe weiterleitet und mit einem öffentlichen Telefonnetz verbindet.

Um Anrufe weiterleiten zu können, muss das MSC wissen, wo die Mobiltelefone aktuell gefunden werden können. Dazu unterhält das MSC eine Datenbank mit allen Handys in der Nähe, die mit den vom MSC verwalteten Zellen verbunden sind. Diese Datenbank heißt **VLR** (*Visitor Location Register*). Außerdem gibt es eine Datenbank im Funknetz, die den letzten bekannten Aufenthaltsort jedes Mobiltelefons angibt, das **HLR** (*Home Location Register*). Diese Datenbank wird benutzt, um ankommende Anrufe an die richtigen Orte zu leiten. Beide Datenbanken müssen aktuell gehalten werden, wenn Handys von Zelle zu Zelle wandern.

Wir werden nun die Luftschnittstelle detaillierter beschreiben. GSM läuft weltweit auf einem Frequenzbereich, der 900, 1 800 und 1 900 MHz umfasst. Das belegte Spektrum ist größer als für AMPS, um sehr viel mehr Nutzer unterstützen zu können. GSM ist genau wie AMPS ein Frequenzduplex-Zellulärsystem. Das heißt, jedes Handy übermittelt auf der einen Frequenz und empfängt auf einer anderen, höheren Frequenz (bei GSM 55 MHz höher gegenüber 80 MHz bei AMPS). Anders als bei AMPS wird bei

GSM jedoch ein einzelnes Frequenzpaar mittels Zeitmultiplexing in Zeitscheiben aufgeteilt. Auf diese Art wird es von mehreren Mobiltelefonen genutzt.

Um mehrere Mobiltelefone handhaben zu können, sind GSM-Kanäle viel breiter als die AMPS-Kanäle (200 kHz gegenüber 30 kHz). Ein 200-kHz-Kanal ist in ►Abbildung 2.47 dargestellt. Ein GSM-System, das im 900-kHz-Bereich arbeitet, verfügt über 124 Simplexkanalpaare. Jeder der Simplexkanäle ist 200 kHz breit und unterstützt acht getrennte Verbindungen mittels Zeitmultiplexing. Jede aktuell aktive Station erhält auf einem Kanalpaar eine Zeitscheibe. Theoretisch können 992 Kanäle pro Zelle unterstützt werden, aber viele davon sind nicht verfügbar, um Frequenzkonflikte mit benachbarten Zellen zu vermeiden. In Abbildung 2.47 gehören die acht blau schattierten Zeitscheiben zur gleichen Verbindung, je vier in eine Richtung. Senden und Empfangen geschehen nicht in der gleichen Zeitscheibe, weil die GSM-Funkwellen nicht gleichzeitig senden und empfangen können und das Umschalten etwas Zeit erfordert. Wenn die mobile Station, die der Frequenz von 890,4/935,4 MHz und der Zeitscheibe 2 zugewiesen ist, an die Basisstation übertragen möchte, dann müsste sie die unteren vier blau schattierten Zeitscheiben (und die zeitlich darauffolgenden) benutzen. Sie würde in jeder Zeitscheibe einen Teil der Daten übertragen, bis die gesamten Daten versendet sind.

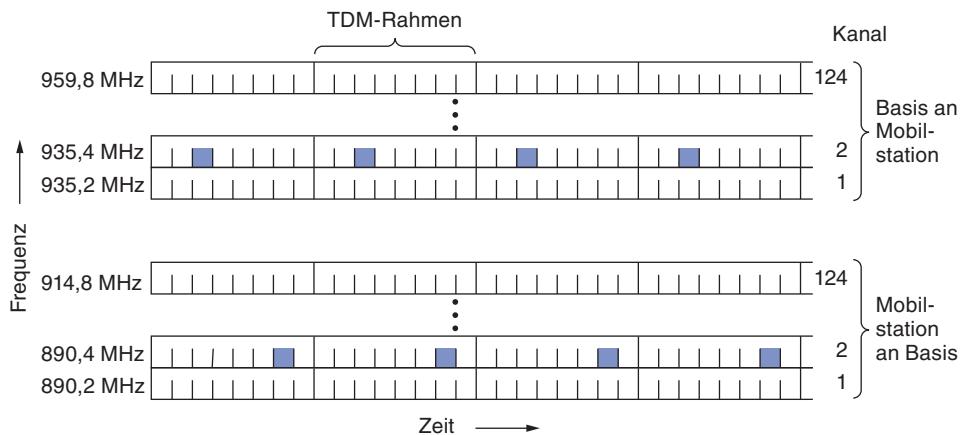


Abbildung 2.47: GSM nutzt 124 Frequenzkanäle, wobei jeder Kanal ein TDM-System mit acht Zeitscheiben verwendet.

Die in Abbildung 2.47 dargestellten TDM-Einträge (d.h. die Daten, die in einer Zeitscheibe übertragen werden) sind Teil einer komplexen Rahmenhierarchie. Jeder TDM-Eintrag hat eine eigene Struktur. Gruppen von TDM-Einträgen bilden Mehrfachrahmen, ebenfalls mit einer besonderen Struktur. Eine vereinfachte Version dieser Hierarchie ist in ►Abbildung 2.48 dargestellt. Hier ist ersichtlich, dass jeder TDM-Eintrag aus einem 148-Bit-Datenrahmen besteht, der den Kanal für 577 µs (einschließlich einer 30-µs-Sicherheitszeit nach jeder Zeitscheibe) belegt. Jeder Datenrahmen beginnt und endet aus Gründen der Rahmenabgrenzung mit drei 0-Bit. Er enthält auch zwei 57 Bit große *Information*-Felder mit je einem Steuerbit, das angibt, ob das folgende *Information*-Feld für Sprache oder Daten gilt. Zwischen den *Information*-Feldern befindet sich ein *Sync*-Feld von 26 Bit, das vom Empfänger benutzt wird, um sich mit den Rahmengrenzen des Senders zu synchronisieren.

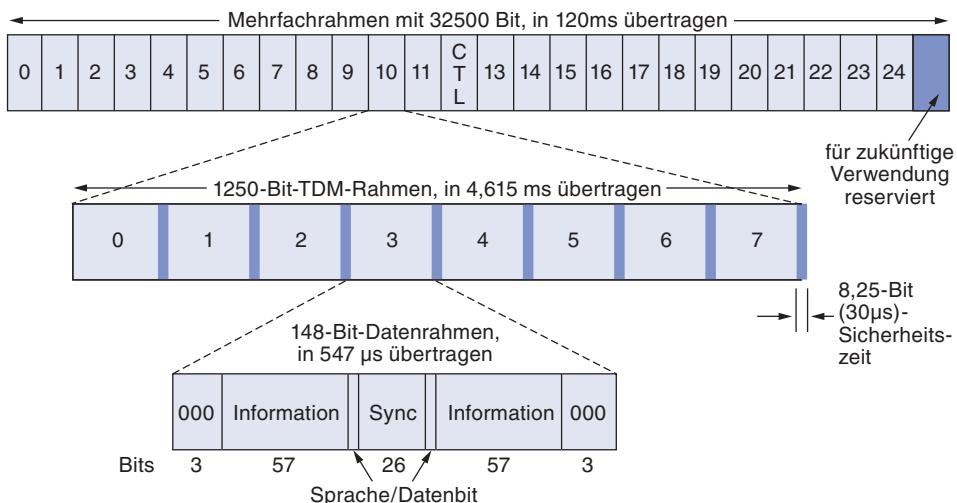


Abbildung 2.48: Teil der GSM-Rahmenstruktur.

Ein Datenrahmen wird in 547 µs übertragen. Einem Sender ist es aber nur gestattet, alle 4,615 µs einen Datenrahmen zu senden, da er den Kanal mit sieben anderen Stationen teilt. Die Gesamtübertragungsrate jedes Kanals beträgt 270 833 Bit/s, aufgeteilt auf acht Benutzer. Doch wie bei AMPS frisst der Overhead einen Großteil der Bandbreite, sodass letztlich nur 24,7 kbit/s für die Nutzdaten pro Benutzer vor der Fehlerkorrektur übrig bleiben. Nach der Fehlerkorrektur bleiben 13 kbit/s für die Sprache übrig. Obwohl dies bedeutend weniger als 64 kbit/s PCM für unkomprimierte Sprachsignale im Telefonfestnetz sind, kann die Kompression auf dem mobilen Gerät dies mit wenig Qualitätsverlust erreichen.

Aus Abbildung 2.48 ist ersichtlich, dass acht Datenrahmen einen TDM-Rahmen und 26 TDM-Rahmen einen Mehrfachrahmen mit 120 ms bilden. Von den 26 TDM-Rahmen in einem Mehrfachrahmen wird Rahmen 12 zur Steuerung benutzt und Rahmen 25 ist für künftige Benutzung reserviert; somit sind für den Benutzerverkehr nur 24 Rahmen verfügbar.

Zusätzlich zu dem in Abbildung 2.48 aufgezeigten Mehrfachrahmen mit 26 Einträgen wird auch ein (hier nicht dargestellter) Mehrfachrahmen mit 51 Einträgen benutzt. Einige dieser Einträge werden für verschiedene Steuerkanäle zur Systemverwaltung verwendet. Der sogenannte **BCCH** (*Broadcast Control Channel*) ist ein kontinuierlicher Ausgabestrom von der Basisstation, der ihre Identität und den Kanalstatus enthält. Alle Mobiltelefone überwachen ihre Signalstärke, um festzustellen, wann sie sich in einer neuen Zelle befinden.

Der **DCCH** (*Dedicated Control Channel*) wird zum Aktualisieren, Registrieren des Standorts und zum Gesprächsaufbau benutzt. Insbesondere verwaltet jedes BSC eine Datenbank der momentan in ihrem Hoheitsgebiet befindlichen Mobiltelefone, das VLR. Informationen, die zur Verwaltung des VLR benötigt werden, werden über den DCCH gesendet.

Schließlich gibt es noch den **CCCH** (*Common Control Channel*), der in drei logische Teilkanäle aufgeteilt ist. Der erste dieser Teilkanäle ist der **Paging Channel**, den die Basisstation benutzt, um eingehende Anrufe anzukündigen. Jedes Mobiltelefon überwacht ihn kontinuierlich, um eventuelle Anrufe festzustellen, die es beantworten muss. Der zweite ist der **Random Access Channel**, mit dem Benutzer eine Zeitscheibe auf dem DCCH anfordern können. Wenn zwei Anforderungen gleichzeitig eingehen, gibt es einen Konflikt und die Anforderungen müssen zu einem späteren Zeitpunkt wiederholt werden. Durch Verwendung von DCCH-Zeitscheiben kann die Station ein Gespräch aufbauen. Die zugeteilte Zeitscheibe wird auf dem dritten Teilkanal, dem **Access Grant Channel**, angekündigt.

Schließlich unterscheiden sich GSM und AMPS darin, wie ein Handover durchgeführt wird. Bei AMPS führt das MSC dieses ganz ohne Hilfe des mobilen Geräts durch. Durch die Zeitscheiben bei GSM kann das Handy die meiste Zeit weder senden noch empfangen. Während der inaktiven Zeitscheiben hat das Mobiltelefon Gelegenheit, die Signalqualität zu anderen Basisstationen in der Nähe zu messen. Diese Information wird an das BSC gesendet, welche diese nutzen kann um festzulegen, wann ein Handy eine Zelle verlässt und eine andere betritt, sodass das Handover durchgeführt werden kann. Dieser Entwurf heißt **MAHO** (*Mobile Assisted HandOff*).

2.7.3 Die dritte Generation der Mobiltelefone: Digitale Sprache und Daten

Die erste Generation der Mobiltelefone war analoge Sprache und die zweite Generation war digitale Sprache. Bei der dritten Generation der Mobiltelefon (**3G**) stehen digitale Sprache *und* Daten im Mittelpunkt.

Die Weiterentwicklung in der Branche wird von verschiedenen Faktoren angetrieben. Bereits jetzt übersteigt der Datenverkehr im Festnetz den Sprachverkehr und wächst exponentiell, wohingegen der Sprachverkehr im Grunde gleich bleibt. Viele Experten erwarten, dass auch bei den mobilen Geräten der Datenverkehr die Sprache bald übersteigt. Darüber hinaus sind das Telefon, die Unterhaltungs- und Computerindustrie alle sehr schnell digital geworden und wachsen immer mehr zusammen. Viele Leute sind ganz wild auf ein leichtes, tragbares Gerät, das als Telefon, Musik- und Videorekorder, E-Mail-Terminal, Webschnittstelle, Spielkonsole und vieles mehr fungieren kann, wobei der weltweite drahtlose Internetzugang in hoher Bandbreite vorausgesetzt wird.

Das iPhone von Apple ist ein gutes Beispiel für diese Art von 3G-Gerät. Durch das iPhone sind die Leute süchtig nach drahtlosen Datendiensten geworden und das drahtlose Datenvolumen von AT&T stieg mit der Beliebtheit des iPhones steil an. Das Problem ist, dass das iPhone ein 2,5G-Netz benutzt (ein verbessertes 2G-Netzwerk, aber noch kein richtiges 3G-Netz) und nicht genügend Datenkapazität vorhanden ist, um die Benutzer glücklich zu machen. Bei der 3G-Funktelefonie dreht sich alles darum, genügend drahtlose Bandbreite zur Verfügung zu stellen, um diese zukünftigen Anwender zufriedenzustellen.

Im Jahre 1992 versuchte die ITU diesen Traum etwas zu konkretisieren und gab eine vorläufige Fassung des sogenannten **IMT-2000** heraus, wobei IMT für **International Mobile Telecommunications** stand. Das IMT-2000-Netz sollte die folgenden Grunddienste für die Benutzer bereitstellen:

1. Sprachübertragung in höchster Qualität
2. Nachrichtenübertragung (als Ersatz für E-Mail, Fax, SMS, Chats etc.)
3. Multimedia (Musik abspielen, Video und Filme abspielen, Fernsehen etc.)
4. Internetzugang (Surfen im Web, wobei Audio und Video eingeschlossen sind)

Weitere Dienste können Videokonferenzen, Telepräsenz, das Spielen von Spielen in Gruppen und M-Commerce (Bezahlen mit dem Handy in einem Laden) sein. Darüber hinaus geht man davon aus, dass diese Dienste jederzeit weltweit verfügbar sind (mit einer automatischen Verbindung über Satellit, wenn kein terrestrisches Netz vorhanden ist) und die Dienstgüte garantiert ist.

ITU hatte eine einzige weltweite Technologie für IMT-2000 im Auge, sodass die Hersteller ein einzelnes Gerät bauen können, das überall auf der Welt verkauft werden kann (wie CD-Player und Computer im Unterschied zu Mobiltelefonen und Fernsehern). Eine einzige Technologie würde auch das Leben von Netzbetreibern viel einfacher machen und mehr Leute dazu ermutigen, diese Dienste zu nutzen. Formatkriege, wie Betamax gegen VHS bei den Videorekordern, sind nicht gut fürs Geschäft.

Wie sich herausstellte, war dies ein wenig zu optimistisch. Die Zahl 2000 stand für drei Dinge: (1) das Jahr, in dem es seinen Dienst aufnehmen sollte, (2) die angenommene Betriebsfrequenz (in MHz) und (3) die Bandbreite des Dienstes (in kbit/s). Keine der drei Annahmen trat ein. Im Jahre 2000 wurde nichts implementiert. ITU empfahl, dass alle Staaten Frequenzen bei 2 GHz reservieren sollten, sodass Geräte nahtlos von Land zu Land einsatzfähig sind. China hat die erforderliche Bandbreite reserviert, sonst aber niemand. Schließlich wurde erkannt, dass 2 Mbit/s derzeit für Benutzer nicht machbar ist, die *zu* mobil sind (aufgrund der Schwierigkeit, die Gesprächsweitergabe entsprechend schnell auszuführen). Realistischer sind 2 Mbit/s für stationäre Benutzer in Räumen (was mit ADSL konkurriert), 384 kbit/s für Leute, die zu Fuß unterwegs sind, sowie 144 kbit/s für Verbindungen in Kraftfahrzeugen.

Trotz dieser anfänglichen Rückschläge ist seither viel verwirklicht worden. Es wurden verschiedene ITU-Vorschläge gemacht und nach einigem Hin und Her kristallisierten sich zwei Hauptentwicklungen heraus. Die erste, **WCDMA** (*Wideband CDMA*), wurde von Ericsson vorgeschlagen und von der Europäischen Union stark vorangetrieben. Es wurde hier mit dem Namen **UMTS** (*Universal Mobile Telecommunications System*) versehen. Der andere Mitbewerber war **CDMA2000**, das von Qualcomm vorgestellt wurde.

Beide Systeme haben mehr Ähnlichkeiten als Unterschiede in dem Sinne, dass sie auf Breitband-CDMA beruhen; WCDMA verwendet 5-MHz-Kanäle und CDMA2000 1,25-MHz-Kanäle. Wenn die Ingenieure von Ericsson und Qualcomm mit der Aufgabe in einen Raum eingesperrt würden, ein einheitliches Verfahren zu entwickeln, würden sie

ganz schnell eines finden. Das wirkliche Problem ist hier nicht die Technik, sondern – wie üblich – die Politik. Europa wollte ein System, das mit GSM kompatibel ist, die USA wollten ein System, das mit einem in den USA weitverbreiteten System kompatibel ist (IS-95). Jede Seite unterstützte das lokale Unternehmen (Ericsson hat seinen Hauptsitz in Schweden, Qualcomm in Kalifornien). Am Ende waren Ericsson und Qualcomm in verschiedene Rechtsstreite wegen der jeweiligen CDMA-Patente verwickelt.

Weltweit werden 3G-Technologien bereits von 10–15 % der mobilen Teilnehmer benutzt; in Nordamerika und Europa sind es rund ein Drittel. Japan ist früh auf diesen Zug aufgesprungen und heute gehören fast alle Mobiltelefone in Japan zur 3. Generation. Diese Zahlen beinhalten sowohl UMTS als auch CDMA2000. Selbst wenn sich der Markt gesundschrumpft, bleibt 3G weiterhin ein großer Hexenkessel an Aktivitäten. Zur allgemeinen Verwirrung beigetragen hat die Festlegung von UMTS zum einzigen 3G-Standard mit mehreren inkompatiblen Optionen, wozu auch CDMA2000 gehörte. Dieser Wechsel ist als Versuch zu verstehen, die verschiedenen Lager zu vereinen, doch damit wurden nur die technischen Unterschiede vertuscht und die Ausrichtung auf die anhaltenden Bemühungen verschleiert. Wir benutzen den Begriff UMTS, wenn wir WCDMA meinen, im Unterschied zu CDMA2000.

Wir werden unsere Diskussion auf die Verwendung von CDMA in zellulären Netzen konzentrieren, da dies das charakteristische Merkmal der beiden Systeme ist. CDMA ist weder FDM noch TDM, sondern eine Art Mischung von beiden, bei der jeder Nutzer gleichzeitig auf demselben Frequenzband sendet. Als CDMA erstmals für Zellulärsysteme vorgeschlagen wurde, kam von der Industrie ungefähr die gleiche Reaktion wie von Königin Isabella, als Kolumbus vorschlug, Indien dadurch zu erreichen, indem man in die falsche Richtung segelte. Durch die Beharrlichkeit eines einzelnen Unternehmens, Qualcomm, wurde CDMA als 2G-System (IS-95) erfolgreich und es entwickelte sich so weit, dass es zur technischen Basis für 3G wurde.

Um CDMA im Mobiltelefonumfeld einsetzen zu können, ist mehr als die grundlegende CDMA-Technik erforderlich, die wir im vorherigen Abschnitt beschrieben haben. Dort haben wir synchrones CDMA betrachtet, bei dem die Chipfolgen exakt orthogonal sind. Dieser Entwurf funktioniert, wenn alle Benutzer zur Startzeit ihrer Chipfolgen synchronisiert sind, wie es bei der Basisstation der Fall ist, die an ein Handy überträgt. Die Basisstation kann die Chipfolgen übermitteln, die gleichzeitig beginnen, sodass die Signale orthogonal sind und separiert werden können. Es ist jedoch schwierig, die Übertragungen von unabhängigen Mobiltelefonen zu synchronisieren. Ohne Vorsichtsmaßnahmen würden die Übertragungen an der Basisstation zu verschiedenen Zeiten ankommen und es könnte keine Orthogonalität garantiert werden. Damit Handys zur Basisstation ohne Synchronisation senden können, benötigen wir Codefolgen, die jeweils orthogonal zueinander bei allen möglichen Verschiebungen sind und nicht nur, wenn sie am Anfang ausgerichtet wurden.

Obwohl es nicht möglich ist, Folgen zu finden, die exakt orthogonal für diesen allgemeinen Fall sind, kommen lange pseudozufällige Folgen dieser Forderung nah genug. Solche Folgen haben die Eigenschaft, dass sie untereinander mit hoher Wahrscheinlichkeit eine geringe **Kreuzkorrelation** bei jeder Verschiebung haben. Dies heißt, wenn

eine Folge von einer anderen Folge zur Berechnung des inneren Produkts multipliziert und aufsummiert wird, dann wird das Ergebnis klein sein; wenn die Folgen orthogonal zueinander wären, dann wäre das Ergebnis null. (Intuitiv sollten sich Zufallsfolgen immer voneinander unterscheiden. Die Multiplikation dieser Folgen sollte dann ein zufälliges Signal erzeugen, welches zu einem kleinen Ergebnis aufsummiert wird.) Damit kann ein Empfänger unerwünschte Übertragungen aus dem empfangenen Signal herausfiltern. Die **Autokorrelation** von pseudozufälligen Folgen ist mit hoher Wahrscheinlichkeit ebenfalls klein, außer bei einer Null-Verschiebung. Das heißt, wenn eine Folge von einer verzögerten Kopie der Folge selbst multipliziert und aufsummiert wird, so ist das Ergebnis klein, außer die Verzögerung ist null. (Intuitiv sieht eine verzögerte Zufallsfolge wie eine andere Zufallsfolge aus und wir sind wieder beim Kreuzkorrelationsfall.) Dadurch wird ein Empfänger am Anfang der gewünschten Übertragung im empfangenen Signal gesperrt.

Durch den Einsatz von pseudozufälligen Folgen kann die Basisstation CDMA-Nachrichten von unsynchronisierten Handys empfangen. Eine implizite Annahme in unserer Diskussion von CDMA ist jedoch, dass die Leistungspegel aller Mobiltelefone mit denen des Empfängers übereinstimmen. Falls dies nicht zutrifft, könnte eine kleine Kreuzkorrelation mit einem leistungsstarken Signal eine große Autokorrelation mit einem schwachen Signal übertreffen. Die Übertragungsleistung von Handys muss also gesteuert werden, um die Interferenz zwischen konkurrierenden Signalen zu minimieren. Diese Interferenz begrenzt die Kapazität von CDMA-Systemen.

Die Leistungspegel, die an einer Basisstation empfangen werden, sind davon abhängig, wie weit die Sender entfernt sind und wie viel Leistung diese übermitteln. Möglicherweise gibt es viele mobile Stationen mit schwankenden Entfernung von der Basisstation. Eine gute Heuristik, die empfangene Leistung anzulegen, besteht darin, dass jede mobile Station an die Basisstation mit dem inversen Leistungspegel übermittelt, die sie von der Basisstation empfängt. Mit anderen Worten, eine mobile Station, die ein schwaches Signal von der Basisstation empfängt, wird mehr Leistung brauchen als eine Station, die ein starkes Signal erhält. Um größere Genauigkeit zu erzielen, gibt die Basisstation außerdem jedem Mobiltelefon ein Feedback, ob die Übertragungsleistung erhöht, verringert oder beibehalten werden soll. Dieses Feedback wird häufig (1 500-mal pro Sekunde) abgegeben, da eine gute Leistungssteuerung zur Minimierung der Interferenz wichtig ist.

Eine weitere Verbesserung des grundlegenden CDMA-Schemas, das wir weiter vorne beschrieben haben, ermöglicht es unterschiedlichen Benutzern, Daten zu verschiedenen Raten zu senden. Dieser Trick besteht bei CDMA einfach darin, dass die Raten, mit denen Chips übertragen werden, festgelegt und den Nutzern Chipfolgen mit unterschiedlicher Länge zugewiesen werden. Bei WCDMA beträgt die Chiprate zum Beispiel 3,84 MChip/Sekunde und die Spreizcodes variieren zwischen 4 und 256 Chips. Bei einem 256-Chip-Code bleiben nach Fehlerbehebung rund 12 kbit/s, diese Kapazität reicht für eine Sprachübertragung aus. Bei einem 4-Chip-Code ist die Benutzerdatenrate nah an 1 Mbit/s. Die Raten von Codes mittlerer Längen liegen dazwischen; um mehrere Mbps zu erreichen, muss das Handy mehr als einen 5-MHz-Kanal auf einmal benutzen.

Nun kommen wir zu den Vorteilen von CDMA – vorausgesetzt, wir konnten die anfänglichen Probleme lösen. CDMA hat drei Hauptvorteile. Erstens kann CDMA die Kapazität verbessern, indem die kleinen Perioden ausgenutzt werden, wenn einige der Sender stumm sind. In einem höflichen Telefongespräch schweigt die eine Seite, während die andere spricht. Im Durchschnitt ist die Leitung nur 40 % der Zeit belegt. Die Pausen können jedoch klein und schwierig vorherzusagen sein. Bei TDM- oder FDM-Systemen ist es nicht möglich, Zeitscheiben oder Frequenzkanäle schnell genug neu zuzuweisen, um von diesen kleinen Sprechpausen zu profitieren. Bei CDMA verringert jedoch ein Benutzer die Interferenz für andere Nutzer, indem einfach nichts gesendet wird, und es ist wahrscheinlich, dass zu jeder Zeit ein gewisser Teil der Nutzer zu einer ausgelasteten Zelle nichts übertragen wird. CDMA nutzt diese erwarteten Sprechpausen, um mehr Anrufe gleichzeitig zuzulassen.

Der zweite Vorteil ist, dass bei CDMA jede Zelle dieselben Frequenzen verwendet. Anders als bei GSM und AMPS wird kein FDM benötigt, um die Übermittlungen von verschiedenen Nutzern zu trennen. Dies beseitigt komplizierte Frequenzplanungsaufgaben und erhöht die Kapazität. Außerdem wird es dadurch für eine Basisstation einfach, mehrere direktionale oder **sektorisierte Antennen** anstatt einer omnidirekionalen Antenne zu benutzen. Direktionale Antennen richten ein Signal auf die gewünschte Richtung aus, in die anderen Richtungen wird das Signal – und somit Interferenzen – abgeschwächt. Dies wiederum erhöht die Kapazität. Drei Sektorentwürfe sind weitverbreitet. Die Basisstation muss das Handy verfolgen, wenn es sich von Sektor zu Sektor bewegt. Dieses Verfolgen ist bei CDMA einfach, weil alle Frequenzen in allen Sektoren benutzt werden.

Der dritte Pluspunkt von CDMA ist, dass es das **Soft Handover** erleichtert, bei dem das Handy von der neuen Basisstation aufgenommen wird, bevor sich die alte abmeldet. Auf diese Weise ist die Fortsetzung des laufenden Gesprächs gewährleistet. Soft Handover wird in ►Abbildung 2.49 gezeigt. Bei CDMA ist dies einfach, da alle Frequenzen in jeder Zelle benutzt werden. Die Alternative ist ein **Hard Handover**, bei dem die alte Basisstation das Gespräch abmeldet, bevor die neue Basisstation es übernimmt. Wenn die neue Basisstation es nicht übernehmen kann (weil beispielsweise keine Frequenz verfügbar ist), dann wird das Gespräch abrupt beendet. Die Benutzer merken dies natürlich gelegentlich, aber bei diesem Verfahren ist dies unvermeidbar. Beim FDM-Entwurf ist Hard Handover die Norm, um die Kosten zu vermeiden, die entstehen, wenn das Mobiltelefon auf zwei Frequenzen gleichzeitig sendet oder empfängt.

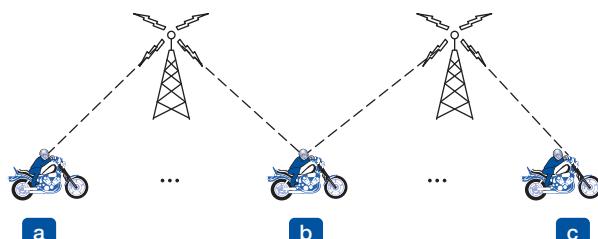


Abbildung 2.49: (a) Vor, (b) während, (c) nach einem Soft Handover.

Es wurde viel über 3G-Systeme geschrieben, wobei sie meistens als das Größte seit Erfindung der Bratkartoffel gepriesen wurden. Inzwischen haben viele Betreiber vorsichtige Schritte in Richtung 3G gewagt, indem sie etwas namens **2,5G** anbieten, wobei aber eigentlich „2,1G“ passender wäre. Ein solches System ist **EDGE** (*Enhanced Data rates for GSM Evolution*), im Grunde ein GSM-System mit mehr Bits pro Symbol. Das Problem ist hier, dass mehr Bits pro Symbol auch mehr Fehler pro Baud bedeuten, sodass EDGE neun verschiedene Schemata für die Modulation und Fehlerkorrektur besitzt, je nachdem wie viel Bandbreite für die Behebung von Fehlern verwendet wird, die durch die höhere Geschwindigkeit verursacht werden. EDGE ist ein Schritt entlang eines Entwicklungswegs von GSM zu WCDMA. Genauso gibt es einen festgelegten Entwicklungsweg für Betreiber, die von IS-95- auf CDMA2000-Netzen aufrüsten.

Selbst wenn 3G-Netze noch nicht vollständig im Einsatz sind, betrachten einige Forscher 3G als abgeschlossene Sache. Diese Leute arbeiten bereits an 4G-Systemen unter dem Namen **LTE** (*Long Term Evolution*). Zu den vorgeschlagenen Merkmalen von 4G-Systemen gehören hohe Bandbreite, Ubiquität (an jedem Ort mögliche Verbindung), nahtlose Integration von kabelgebundenen Netzen und drahtlosen IP-Netzen (einschließlich IEEE-802.11-Zugangspunkte), adaptive Ressourcen- und Frequenzverwaltung und hohe Dienstgüte für Multimedia-Anwendungen. Für weitere Informationen, siehe Astely et al. (2009) und Larmo et al. (2009).

Mittlerweile sind drahtlose Netze mit 4G-Leistungsniveau bereits verfügbar. Das beste Beispiel dafür ist **IEEE 802.16**, auch bekannt als **WiMAX**. Einen Überblick über mobiles WiMAX finden Sie in Ahmadi (2009). Wenn man behauptet, dass die Branche in einem Wandel begriffen ist, so ist dies eine riesige Untertreibung. Schauen wir in einigen Jahren nochmals zurück, dann werden wir es wissen.

2.8 Kabelfernsehen

Wir haben nun die Festnetz- und die Mobiltelefone ziemlich ausführlich untersucht. Beide werden ganz eindeutig in zukünftigen Netzen eine entscheidende Rolle spielen. Doch es gibt noch einen weiteren wichtigen Mitspieler beim Internetzugang, der sich im letzten Jahrzehnt herauskristallisiert hat: Kabelfernsehnetze. Viele Leute beziehen heutzutage Telefon- und Internetdienste über Kabel. In den folgenden Abschnitten untersuchen wir die Eignung des Kabelfernsehens als Netzsystem und stellen es den gerade untersuchten Telefonsystemen gegenüber. Maßgebliche Literatur für weitere Informationen sind Donaldson und Jones (2001), Dutta-Roy (2001) und Fellows und Jones (2001).

2.8.1 Gemeinschaftsantennen-Fernsehen

Kabelfernsehen wurde Ende der 1940er Jahre entwickelt, um für Bewohner ländlicher oder hügeliger Gebiete einen besseren Empfang zu ermöglichen. Das System bestand ursprünglich aus einer großen Antenne auf einem Hügel, um das Fernsehsignal aus der Luft zu empfangen, einem Verstärker, der sogenannten Kopfstelle (*headend*), zur

Verstärkung des Signals und einem Koaxialkabel zu den jeweiligen Häusern, wie in ▶ Abbildung 2.50 dargestellt.

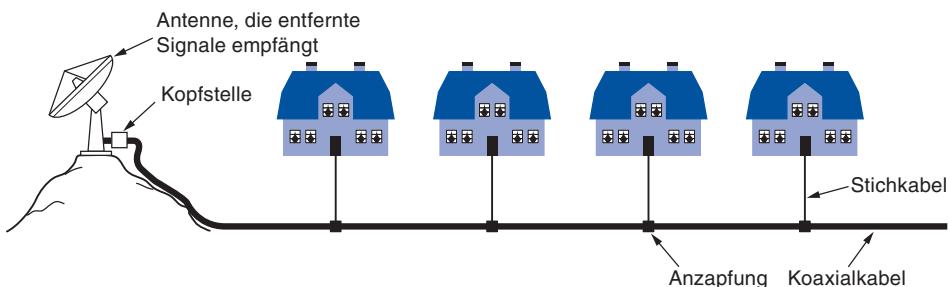


Abbildung 2.50: Ein frühes Kabelfernsehensystem.

In der Anfangszeit bezeichnete man dies als **Gemeinschaftsantennen-Fernsehen** (*Community Antenna Television*). Das Ganze hatte große Ähnlichkeit mit Tante-Emma-Läden: Jeder, der sich ein bisschen mit Elektronik auskannte, konnte für seine Stadt einen Dienst anbieten, und die Benutzer drängten sich herein, um die Kosten zu begleichen. Als die Anzahl der Teilnehmer zunahm, wurden mehr Kabel auf das ursprüngliche Kabel gespleißt und Verstärker bei Bedarf hinzugefügt. Die Übertragung ging in eine Richtung, von der Empfangsstelle zu den Benutzern. Im Jahre 1970 existierten Tausende von einzelnen Systemen.

1974 startete Time Inc. einen neuen Kanal namens Home Box Office mit neuen Inhalten (Filme), der nur über Kabel verteilt wurde. Andere reine Kabelkanäle folgten mit The-menschwarpunkten wie Nachrichten, Sport, Kochen und vielen anderen Themen. Diese Entwicklung führte zu zwei Veränderungen. Erstens begannen große Unternehmen, vorhandene Kabelsysteme zu kaufen und neue Kabel zu legen, um neue Teilnehmer zu gewinnen. Zweitens war es nun erforderlich, mehrere Systeme zu verbinden, oftmals in weit auseinanderliegenden Städten, um die neuen Kabelkänele zu verbreiten. Die Kabelunternehmen begannen, Kabel zwischen ihren Städten zu legen, um alle zu einem System zu verbinden. Dies entsprach der Entwicklung in der Telefonbranche etwa 80 Jahre früher, als die isolierten Teilnehmervermittlungssämter verbunden wurden, um Ferngespräche zu ermöglichen.

2.8.2 Internet über Kabel

Im Laufe der Jahre wuchs das Kabelsystem und die Kabel zwischen den verschiedenen Städten wurden ähnlich wie beim Telefonsystem durch Glasfaser mit hoher Bandbreite ersetzt. Ein Glasfaser-System für Langstreckenübertragungen mit einem Koaxialkabel bis zu den einzelnen Häusern wird als **HFC-System** (*Hybrid Fiber Coax*) bezeichnet. Die elektrooptischen Konverter, die die Schnittstelle zwischen den optischen und elektrischen Systemkomponenten bilden, werden als **Glasfaserknoten** bezeichnet. Da die Bandbreite von Glasfaser sehr viel größer ist als die von Koaxialkabeln, kann ein Glasfaserknoten mehrere Koaxialkabel versorgen. Ein Ausschnitt eines modernen HFC-Systems wird in ▶ Abbildung 2.51a gezeigt.

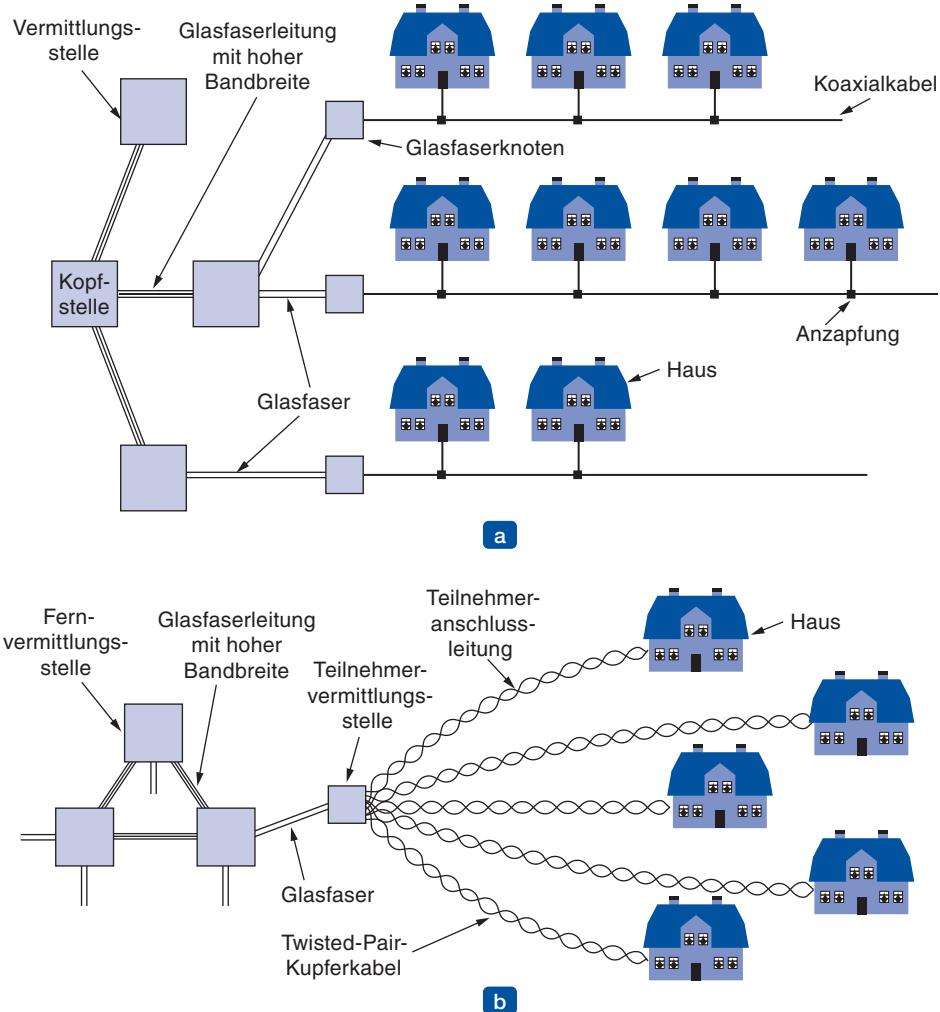


Abbildung 2.51: (a) Kabelfernsehen. (b) Telefonfestnetz.

Im Laufe des vergangenen Jahrzehnts haben sich viele Kabelbetreiber entschlossen, in das Geschäft der Bereitstellung von Internetzugängen und auch in das Telefongeschäft einzusteigen. Die vielen technischen Unterschiede zwischen einer Kabelanlage und einer Telefonanlage wirkten sich auf die Maßnahmen aus, die zur Verwirklichung dieser Ziele ergriffen werden mussten. So mussten beispielsweise alle Einwegverstärker durch bidirektionale Verstärker ersetzt werden, um sowohl Upstream- als auch Downstream-Übertragungen zu unterstützen. Währenddessen verwendeten frühe Internet-über-Kabel-Systeme das Kabelfernsehnetz für den Downstream und eine Wähleleitung über das Telefonnetz für den Upstream. Dies war zwar eine clevere Notlösung, hatte aber nicht viel mit einem Netzwerk im eigentlichen Sinne zu tun.

Es besteht aber noch ein weiterer Unterschied zwischen dem HFC-System in Abbildung 2.51a und dem Telefonsystem von ►Abbildung 2.51b, der sehr viel schwieriger behoben werden kann. In den Wohnvierteln wird ein Kabel von vielen Häusern gemeinsam genutzt, wohingegen bei Telefonsystemen jedes Haus seine private Teilnehmeranschlussleitung hat. Bei der Ausstrahlung von Fernsehsendungen ist diese gemeinsame Nutzung ein logischer Schritt. Alle Programme werden über das Kabel übertragen und es spielt keine Rolle, ob es 10 oder 10 000 Fernsehzuschauer sind. Wird das gleiche Kabel jedoch für den Internetzugang verwendet, spielt es dagegen eine große Rolle, ob 10 oder 10 000 Benutzer daran hängen. Wenn ein Benutzer sehr große Dateien herunterlädt, wird hierdurch potenziell Bandbreite von anderen Benutzern weggenommen. Je mehr Benutzer aktiv sind, umso stärker ist die Bandbreite umkämpft. Beim Telefonsystem ist es hingegen so, dass das Herunterladen einer großen Datei über ADSL nicht die Bandbreite der Nachbarn reduziert. Andererseits ist die Bandbreite des Koaxialkabels viel höher als die von Twisted-Pair-Kabeln – Sie können sich also glücklich schätzen, wenn Ihr Nachbar das Internet nur wenig nutzt.

Die Kabelindustrie hat das Problem so gelöst, dass sie lange Kabel aufgeteilt und jedes direkt mit einem Glasfaserknoten verbunden hat. Die Bandbreite von der Kopfstelle zu jedem Glasfaserknoten ist so gut wie unbegrenzt, sodass also – solange es nicht zu viele Teilnehmer pro Kabelsegment sind – das Verkehrsaufkommen verwaltbar ist. Typische Kabel versorgen heutzutage 500–2 000 Häuser, doch wenn immer mehr Personen das Internet über Kabel abonnieren, kann die Last zu groß werden und die Glasfaserknoten müssen in noch mehr Kabel und Glasfaserknoten aufgeteilt werden.

2.8.3 Zuweisung der Frequenzbereiche

Wenn man alle Fernsehkanäle hinauswirft und die Kabelinfrastruktur nur für den Internetzugang verwendet, würde dies vermutlich zu einer großen Menge erzürnter Kunden führen, sodass die Kabelunternehmen dies lieber lassen. Darüber hinaus regulieren die meisten Städte sehr stark, was über Kabel übertragen wird, sodass die Kabelbetreiber dies nicht tun dürften, selbst wenn sie es wirklich wollten. Sie müssen also einen Weg finden, um Fernsehen und Internet auf dem gleichen Kabel friedlich nebeneinander betreiben zu können.

Die Lösung beruht auf Frequenzmultiplexing. Kabelfernsehkanäle in Nordamerika senden im Frequenzbereich von 54–550 MHz (außer UKW-Radio von 88 bis 108 MHz). Fernsehkanäle haben eine Bandbreite von 6 MHz, einschließlich der Sicherheitsfrequenzbänder, und können einen traditionellen analogen Fernsehkanal oder mehrere digitale Fernsehkanäle transportieren. In Europa ist die untere Grenze in der Regel 65 MHz und die Kanäle sind 6–8 MHz breit aufgrund der höheren Auflösung, die von PAL und SECAM benötigt wird, aber ansonsten ist das Zuweisungsschema ähnlich. Der untere Teil des Bands wird nicht verwendet. Moderne Kabel können auch gut über 550 MHz arbeiten, oftmals sogar bis 750 MHz oder mehr. Mit der gewählten Lösung sollten Rückkanäle in das 5- bis 42-MHz-Band (etwas höher in Europa) eingeführt und die Frequenzen am oberen Ende für die Downstream-Datenübertragung verwendet werden. Der Frequenzbereich des Kabels ist in ►Abbildung 2.52 dargestellt.

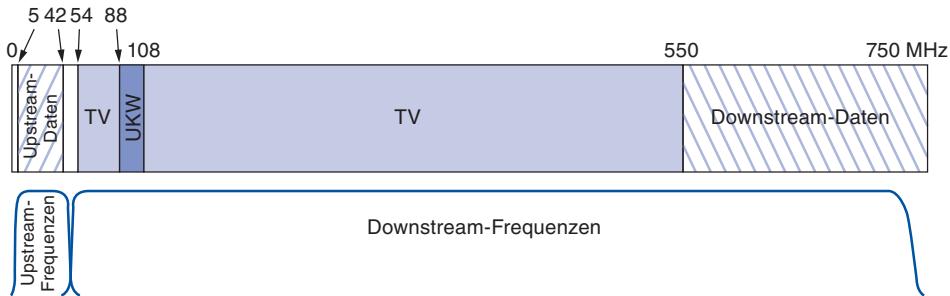


Abbildung 2.52: Frequenzzuweisung bei einem typischen Kabelfernsehsystem, das für den Internetzugang verwendet wird.

Da die Fernsehsignale in Downstream-Richtung übertragen werden, kann man hier Upstream-Verstärker verwenden, die nur im 5- bis 42-MHz-Bereich arbeiten, sowie Downstream-Verstärker, die nur mit 54 MHz und höher arbeiten (siehe Abbildung 2.52). Wir erhalten daher eine Asymmetrie zwischen den Upstream- und Downstream-Bandbreiten, da ein größerer Frequenzbereich oberhalb der für TV reservierten Frequenzen als darunter verfügbar ist. Andererseits wünschen die Benutzer mehr Downstream-Datenverkehr, sodass Kabelbetreiber hierüber nicht unglücklich sind. Telefongesellschaften setzen oftmals asymmetrisches DSL ein, selbst wenn hierfür kein technischer Grund besteht.

Zusätzlich zum Aufrüsten der Verstärker muss der Betreiber auch die Kopfstelle aufrüsten, von einem dummen Verstärker auf ein intelligentes digitales Computersystem mit einer Glasfaserschnittstelle mit hoher Bandbreite zu einem ISP. Oftmals wird auch der Name aufgerüstet, von „Kopfstelle“ auf **CMTS** (*Cable Modem Termination System*). Wir werden jedoch auf diese Namensaufwertung verzichten und bleiben bei dem herkömmlichen Begriff „Kopfstelle“.

2.8.4 Kabelmodems

Der Internzugang erfordert ein Kabelmodem – ein Gerät, das mit zwei Schnittstellen ausgestattet ist: eine zum Computer und eine zum Kabelnetz. In den ersten Jahren des kabelgebundenen Internetzugangs hatte jeder Betreiber ein proprietäres Kabelmodem, das von einem Techniker des jeweiligen Kabelunternehmens installiert wurde. Es zeigte sich aber bald, dass ein offener Standard zu mehr Wettbewerb und einer Senkung der Preise führen würde, sodass mehr Benutzer auf diesen Dienst zugreifen werden. Wenn darüber hinaus die Kunden Kabelmodems in Läden erwerben und selbst installieren (wie sie dies mit den drahtlosen Zugangspunkten machen), müssten nicht so viele Servicetechniker eingesetzt werden.

So taten sich die großen Kabelbetreiber mit einem Unternehmen namens CableLabs zusammen, um einen Kabelmodemstandard zu erarbeiten und die Produkte auf Kompatibilität zu testen. Dieser Standard namens **DOCSIS** (*Data Over Cable Service Interface Specification*) hat größtenteils die proprietären Modems ersetzt. Die Version 1.0 von DOCSIS kam 1997 heraus, im Jahr 2001 folgte bereits DOCSIS 2.0. Diese Version

bot höhere Upstream-Datenraten, um symmetrische Dienste wie IP-Telefonie besser zu unterstützen. Die aktuelle Version des Standards ist DOCSIS 3.0, die 2006 herauskam. Diese benutzt mehr Bandbreite, um die Datenraten in beiden Richtungen zu erhöhen. Die europäische Version dieser Standards heißt **EuroDOCSIS**. Nicht alle Kabelhersteller unterstützen die Standardisierungsbestrebungen, da viele mit dem Vermieten von Modems an ihre Kunden gute Geschäfte machen. Ein offener Standard mit Dutzenden von Herstellern, die Kabelmodems in Läden verkaufen, bedeutet natürlich das Ende dieser lukrativen Einnahmequelle.

Die Modem-Computer-Schnittstelle ist ganz einfach. Es ist derzeit in der Regel Ethernet oder manchmal USB. Die Schnittstelle auf der anderen Seite ist komplizierter, da FDM, TDM und CDMA eingesetzt werden, um die Bandbreite des Kabels zwischen den Teilnehmern aufzuteilen.

Wenn ein Kabelmodem eingesteckt und angeschaltet wird, prüft es den Empfangskanal in periodischen Abständen auf ein spezielles Paket, das von der Kopfstelle gesendet wird, um die Systemparameter für Modems bereitzustellen, die gerade online gegangen sind. Entdeckt ein Medium dieses Paket, gibt es sein Vorhandensein auf einem der Rückkanäle bekannt. Die Kopfstelle antwortet, indem dem Modem die Rück- und Empfangskanäle zugeordnet werden. Diese Zuweisungen können später geändert werden, wenn die Kopfstelle es für erforderlich hält, die Last auszugleichen.

Die 6-MHz- oder 8-MHz-Kanäle sind für FDM vorgesehen. Jedes Kabelmodem sendet Daten auf einem Rück- und einem Empfangskanal bzw. mehreren Kanälen unter DOCSIS 3.0. Das übliche Schema besteht darin, jeden 6- oder 8-MHz-Empfangskanal mit QAM-64 zu modulieren, oder QAM-256, wenn die Kabelqualität außergewöhnlich gut ist. Bei einem 6 MHz-Kanal und QAM-64 erhalten wir etwa 36 Mbit/s. Wenn man den Overhead abzieht, liegen die verbleibenden Nutzdaten um 27 Mbit/s. Bei QAM-256 beträgt die verbleibende Nutzlast etwa 39 Mbit/s. Die europäischen Werte liegen um ein Drittel höher.

Beim Rückkanal gibt es mehr Hochfrequenzrauschen, da das System ursprünglich nicht für Daten ausgelegt war, außerdem kommt das Rauschen von mehreren Teilnehmern an der Kopfstelle an, sodass ein konservativeres Schema benutzt wird. Dieses reicht von QPSK bis QAM-128, wobei einige der Symbole zur Fehlerschutz mit Trellis Coded Modulation verwendet werden. Bei weniger Bits auf dem Rückkanal ist die Asymmetrie zwischen Upstream- und Downstream-Datenraten viel größer als in Abbildung 2.52 dargestellt.

TDM wird dann eingesetzt, um die Bandbreite auf dem Rückkanal mit mehreren Teilnehmern gemeinsam zu nutzen. Andernfalls würden deren Übertragungen an der Kopfstelle kollidieren. Die Zeit wird in sogenannte **Minizeitscheiben** (*minislot*) aufgeteilt und die unterschiedlichen Teilnehmer senden in diesen verschiedenen Minizeitscheiben. Dazu bestimmt das Modem die Entfernung bis zur Kopfstelle, indem ein spezielles Paket gesendet und daraufhin geprüft wird, wie lange die Antwort braucht. Diesen Prozess nennt man **Bereichswahl** (*ranging*). Das Modem muss die korrekte Entfernung kennen, um das zeitliche Verhalten richtig anzupassen. Jedes Upstream-Paket

muss in einen oder mehrere aufeinanderfolgende Minizeitscheiben an der Kopfstelle passen, wenn sie ankommen. Die Kopfstelle kündigt den Beginn einer neuen Runde von Minizeitscheiben in periodischen Abständen an, aber der Startschuss wird aufgrund der Signalübertragungsverzögerung nicht von allen Modems am Kabel gleichzeitig gehört. Wenn ein Modem weiß, wie weit es bis zur Kopfstelle ist, kann es berechnen, wann die erste Minizeitscheibe wirklich gestartet wurde. Die Länge einer Minizeitscheibe hängt vom Netz ab. Die typischen Nutzdaten betragen 8 Byte.

Während der Initialisierung weist die Kopfstelle jedem Modem eine Minizeitscheibe zu, der für die Anforderung der Rückkanal-Bandbreite verwendet wird. Möchte ein Computer ein Paket senden, überträgt er das Paket zum Modem, das dann dafür die erforderliche Anzahl von Minizeitscheiben anfordert. Wird die Anforderung akzeptiert, schickt die Kopfstelle über den Empfangskanal eine Bestätigung, die dem Modem angibt, wie viele Minizeitscheiben für das Paket reserviert wurden. Das Paket wird dann beginnend mit der zugewiesenen Minizeitscheibe gesendet. Weitere Pakete können über ein Feld im Header angefordert werden.

In der Regel wird mehreren Modems dieselbe Minizeitscheibe zugewiesen, was zu Konkurrenz führt. Es gibt zwei unterschiedliche Möglichkeiten, um diesem Problem zu begegnen. Die erste ist, CDMA einzusetzen, um die Minizeitscheiben unter den Teilnehmern aufzuteilen. Damit ist das Konkurrenzproblem gelöst, da alle Teilnehmer mit einer CDMA-Codefolge gleichzeitig senden können, wenn auch mit verringrigerer Datenrate. Die zweite Option ist, CDMA nicht zu verwenden. In diesem Fall kann es passieren, dass es aufgrund einer Kollision keine Bestätigung für die Anfrage gibt. Sollte dies passieren, so wartet das Modem eine zufällige Zeitspanne und versucht es dann erneut. Treten hintereinander mehrere Fehlversuche auf, wird die Zufallszeit verdoppelt. (Für Leser, die mit Netzen vertrauter sind: Dieser Algorithmus ist einfach S-ALOHA mit binärer exponentieller Übertragungsverzögerung (Binary Exponential Backoff). Ethernet kann für Kabel nicht verwendet werden, weil die Stationen das Medium nicht abhören können. Dieses Thema wird später in *Kapitel 4* noch einmal aufgegriffen.)

Die Empfangskanäle werden anders gehandhabt als die Rückkanäle. Einerseits gibt es nur einen Sender (die Kopfstelle), sodass es keine Konkurrenz und keinen Bedarf an Minizeitscheiben gibt, was im Grunde nur ein statistisches Zeitmultiplexverfahren ist. Andererseits ist die Menge des Datenverkehrs in Richtung auf das Ziel in der Regel viel größer, sodass eine feste Paketgröße von 204 Byte verwendet wird. Ein Teil davon ist der Reed-Solomon-Fehlerkorrekturcode und ein gewisser Overhead, sodass für die Nutzdaten des Benutzers 184 Byte übrig sind. Diese Zahlen wurden zur Sicherstellung der Kompatibilität zum Digitalfernsehen mit MPEG-2 verwendet, sodass die TV- und Empfangskanäle auf die gleiche Weise formatiert sind. Logisch sehen die Verbindungen wie in ► Abbildung 2.53 dargestellt aus.

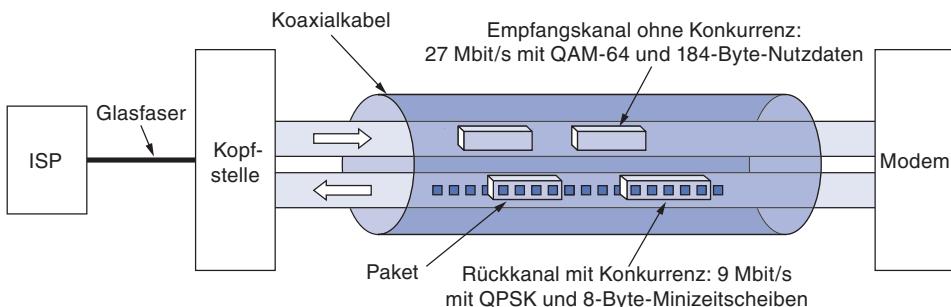


Abbildung 2.53: Typischer Aufbau der Rück- und Empfangskanäle in Nordamerika.

2.8.5 Vergleich: ADSL und Kabel

Was ist besser, ADSL oder Kabel? Diese Frage ist wie die Frage, welches Betriebssystem besser ist. Oder welche Sprache die bessere ist. Oder welche Religion. Die Antwort hängt immer davon ab, wen Sie fragen. Vergleichen wir ADSL und Kabel anhand einiger Gesichtspunkte. Beide verwenden Glasfaser im Backbone, unterscheiden sich aber im vorderen Bereich. Kabel verwendet ein Koaxialkabel, ADSL verwendet Twisted-Pair-Kabel. Die theoretische Übertragungskapazität eines Koaxialkabels ist hundertmal höher als die von Twisted-Pair-Kabeln. Es ist aber nicht die gesamte Kapazität des Kabels für Datennutzer verfügbar, weil ein Großteil der Kabelbandbreite für unnütze Dinge wie Fernsehprogramme verschwendet wird.

In der Praxis ist es schwierig, eine allgemeine Aussage über die effektive Kapazität zu treffen. ADSL-Anbieter machen genaue Aussagen zur Bandbreite (wie 1 Mbit/s für Downstream, 256 kbit/s für Upstream) und erzielen in der Regel davon 80 % konsistent. Kabelanbieter kappen eventuell die Bandbreite jedes Benutzers, um Leistungsvorhersagen machen zu können. Trotzdem können keine Garantien gegeben werden, da die effektive Kapazität davon abhängt, wie viele Personen zu einem Zeitpunkt in einem Kabelsegment aktiv sind. Manchmal kann es besser als ADSL sein, manchmal auch schlechter. Diese Unvorhersagbarkeit kann sehr ärgerlich sein. Wenn in einem Augenblick eine prima Leistung vorhanden ist, so ist dies keine Garantie für die Leistung in der nächsten Minute – weil vielleicht gerade eben der größte Bandbreitenfresser der Stadt seinen Computer eingeschaltet hat.

Wenn bei einem ADSL-System mehr Benutzer einsteigen, haben hier die steigenden Zahlen wenig Auswirkung auf die vorhandenen Benutzer, da jeder eine eigene Verbindung besitzt. Beim Kabel fällt die Leistung für die vorhandenen aktiven Nutzer, wenn mehr Teilnehmer ins Internet gehen. Die einzige Rettung für den Kabelbetreiber ist, belegte Kabel aufzuteilen und jedes direkt mit einem Glasfaserknoten zu verbinden. Dies kostet Zeit und Geld, sodass hier geschäftliche Gründe dagegen sprechen.

Wir haben ja bereits ein anderes System mit einem gemeinsam genutzten Kanal wie das Kabel untersucht: das Mobilfunksystem. Auch hier nutzt eine Gruppe von Teilnehmern – wie könnten sie Zellgenossen nennen – eine feste Bandbreite gemeinsam. Beim Sprachverkehr, der ziemlich gleichmäßig ist, wird die Bandbreite unter den akti-

ven Nutzern unter Verwendung von FDM und TDM in feste Teile unterteilt. Beim Datenverkehr ist diese feste Aufteilung jedoch ineffizient, weil die Datennutzer oftmals nichts machen und der Kanal daher im Leerlaufzustand ist, sodass die reservierte Bandbreite verschwendet wird. Wie beim Kabelzugriff wird ein dynamischeres Mittel verwendet, um die gemeinsam benutzte Bandbreite zuzuweisen.

ADSL und Kabel unterscheiden sich auch hinsichtlich der Verfügbarkeit. Jeder hat ein Telefon, aber nicht alle Benutzer befinden sich so nahe an einer Teilnehmervermittlungsstelle, dass sie ADSL beziehen können. Andererseits hat nicht jeder Kabel; wenn Sie aber Kabel haben und das Unternehmen einen Internetzugang anbietet, können Sie dieses nutzen. Die Entfernung zum Glasfaserknoten oder zur Kopfstelle spielt keine Rolle. In diesem Zusammenhang ist auch wichtig, dass, seit Kabel als Medium für den Fernsehempfang genutzt wurden, Kabel nur in wenigen Unternehmen installiert wurden.

Das Telefonsystem ist in der Regel zuverlässiger als der Kabelzugang. So verfügt es beispielsweise über Notstromversorgung und funktioniert selbst bei Stromausfall normal weiter. Wenn beim Kabel bei einem Verstärker in der Kette der Strom ausfällt, werden alle nachfolgenden Benutzer sofort abgeschnitten.

Schließlich bieten die meisten ADSL-Anbieter verschiedene ISPs an. Manchmal sind sie auch von Gesetzes wegen dazu verpflichtet. Dies ist bei Kabelbetreibern nicht immer der Fall. Die Schlussfolgerung ist die, dass ADSL und Kabel mehr Gemeinsamkeiten als Unterschiede enthalten. Sie bieten vergleichbare Dienstleistungen und mit zunehmendem Wettbewerb auch vergleichbare Preise.

Zusammenfassung

Die **Bitübertragungsschicht** ist die Grundlage aller Netze. Die Natur stellt bei allen Kanalarten zwei grundlegende Grenzen auf und diese Grenzen bestimmen die Bandbreite. Das sind die Nyquist-Grenze in Bezug auf rauschfreie Kanäle und die Shannon-Grenze bei Kanälen, die Rauscheinwirkungen unterliegen.

Übertragungsmedien können gerichtet oder ungerichtet sein. Zur ersten Kategorie zählen Twisted-Pair-Kabel, Koaxialkabel und Lichtwellenleiter. Ungerichtete Medien sind u. a. terrestrische Funkwellen, Mikrowellen, Infrarot, Laser und Satelliten.

Digitale **Modulationsmethoden** senden Bits als analoge Signale über gerichtete und ungerichtete Medien. Leitungscodes arbeiten im Basisband und Signale können in einem Durchlassbereich abgelegt werden, indem die Amplitude, die Frequenz und die Phase eines Trägers moduliert wird. Mit Zeit-, Frequenz- und Codemultiplexing können Kanäle von mehreren Anwendern gemeinsam genutzt werden.

Ein wichtiges Element bei den meisten Fernnetzen (WANs) ist das **Telefonsystem**. Seine wesentlichen Komponenten sind Teilnehmeranschlussleitungen, Verbindungsleitungen und Vermittlungsgeräte (Switches). ADSL unterstützt Geschwindigkeiten von bis 40 Mbit/s über die Teilnehmeranschlussleitung, indem diese in viele parallele Unterträger aufgeteilt wird. Dies übertrifft die Datenraten von Telefonmodems weit. PONs bringen FttH für sogar noch größere Zugangsrraten als ADSL.

Verbindungsleitungen übertragen digitale Informationen. Multiplexing kann zum einen mithilfe von WDM durchgeführt werden, um sehr hohe Kapazitätsverbindungen über einzelne Glasfasern zu versorgen, zum anderen aber auch mithilfe von TDM zur gemeinsamen Nutzung von hochratigen Verbindungen benutzt werden. Sowohl Leitungs- als auch Paketvermittlung haben ihre Bedeutung.

Für **mobile Anwendungen** ist das kabelgebundene Telefonfestnetz nicht geeignet. Mobiltelefone werden derzeit viel für die Sprachübertragung und zunehmend auch für die Übertragung von Daten verwendet. Das Mobilfunknetz hat bisher drei Generationen durchlaufen. Die erste Generation (1G) war analog und wurde von AMPS dominiert. Die zweite Generation (2G) war digital, wobei GSM zurzeit das weltweit am meisten eingesetzte Mobilfunknetz ist. Die dritte Generation (3G) ist digital und basiert auf Breitband-CDMA, wobei heute WCDMA und auch CDMA2000 eingesetzt werden.

Ein alternatives System für den Netzzugriff ist das **Kabelfernsehsystem**. Dieses hat sich im Laufe der Zeit von Koaxialkabel zu einem hybriden Glasfaser-Koaxial-System und von Fernsehen zu Fernsehen und Internet entwickelt. Es bietet potenziell eine sehr viel höhere Bandbreite, aber die Bandbreite hängt in der Praxis sehr stark von den anderen Benutzern ab, da die Bandbreite gemeinsam benutzt wird.



Übungsaufgaben

- 1** Berechnen Sie die Fourierkoeffizienten für die Funktion $f(t)=t$ ($0 \leq t \leq 1$).
- 2** Ein rauschfreier 4-kHz-Kanal wird jede Millisekunde abgefragt. Wie hoch ist die maximale Übertragungsrate? Wie ändert sich die Datenrate, wenn der Kanal bei einem Signal-Rausch-Verhältnis von 30 dB verrauscht ist?
- 3** Fernsehkanäle haben eine Bandbreite von 6 MHz. Wie viele Bit/s können bei der Verwendung von vierstufigen Digitalsignalen verwendet werden? Gehen Sie von einem rauschfreien Kanal aus.
- 4** Wie hoch ist die maximal erreichbare Datenübertragungsrate, wenn ein binäres Signal über einen 3-kHz-Kanal mit einem Signal-Rausch-Verhältnis von 20 dB gesendet wird?
- 5** Welches Signal-Rausch-Verhältnis ist erforderlich, um einen T1-Träger auf eine 50-kHz-Leitung zu legen?
- 6** Was sind die Vorteile von Glasfaser gegenüber Kupfer als Übertragungsmedium? Gibt es auch Nachteile dabei?
- 7** Wie viel Bandbreite besteht in einem Frequenzbereich von $0,1 \mu\text{m}$ bei einer Wellenlänge von $1 \mu\text{m}$?
- 8** Eine Bildfolge eines Computerbildschirms soll über einen Lichtwellenleiter übertragen werden. Der Bildschirm ist $2\,560 \times 1\,600$ Pixel groß und jedes Pixel hat 24 Bit. Insgesamt sind 60 Bilder pro Sekunde vorhanden. Wie viel Bandbreite ist erforderlich und welche Wellenlänge (in μm) wird für dieses Band bei $1,30 \mu\text{m}$ benötigt?
- 9** Trifft das Nyquist-Theorem nur auf Kupferdraht oder auch auf hochqualitative Singlenmode-Glasfaser zu?
- 10** Radioantennen funktionieren oft am besten, wenn der Durchmesser der Antenne der Wellenlänge der Radiowelle entspricht. Einsetzbare Antennen können einen Durchmesser von 1 cm bis 5 m haben. Welchen Frequenzbereich decken sie ab?
- 11** Ein 1 mm großer Laserstrahl wird auf einen 1 mm großen Detektor gerichtet, der sich 100 m entfernt auf dem Dach eines Gebäudes befindet. Ab welcher Winkelabweichung (in Grad) verfehlt der Laserstrahl den Detektor?
- 12** Die 66 Satelliten in niedriger Umlaufbahn des Iridiumprojekts werden in sechs Ketten um den Erdball angeordnet. In der Höhe, in der sie sich befinden, beträgt die Umlaufzeit 90 Minuten. Wie groß ist das durchschnittliche Handover-Intervall bei einem stationären Sender?
- 13** Berechnen Sie die Übertragungszeit von Endpunkt zu Endpunkt eines Pakets für GEO- (Höhe: 35 800 km), MEO- (Höhe: 18 000 km) und LEO-Satelliten (Höhe: 750 km).
- 14** Wie ist die Latenzzeit eines Anrufs, der seinen Ursprung im Nordpol hat, um den Südpol zu erreichen, wenn der Anruf über Iridiumsatelliten geleitet wird. Gehen Sie davon aus, dass die Umschaltzeit an den Satelliten $10 \mu\text{s}$ dauert und der Erdradius 6 371 km beträgt.

- 15** Was ist die minimale Bandbreite, die benötigt wird, um eine Datenrate von B Bit/s zu erreichen, wenn das Signal mittels NRZ, MLT-3 und Manchester-Codierung übertragen wird? Erläutern Sie Ihre Antwort.
- 16** Zeigen Sie, dass bei der 4B/5B-Codierung spätestens nach jedem vierten Bit ein Signalübergang stattfindet.
- 17** Wie viele Teilnehmervermittlungsstellen-Codes gab es vor 1984 in den USA, als jede Teilnehmervermittlungsstelle durch eine dreistellige Ortsvorwahl und die ersten drei Ziffern der lokalen Telefonnummer gekennzeichnet wurde? Ortsvorwahlen begannen mit einer Ziffer zwischen 2 und 9, hatten als zweite Ziffer 0 oder 1 und endeten mit einer beliebigen Ziffer. Die ersten beiden Ziffern einer lokalen Telefonnummer lagen immer zwischen 2 und 9. Die dritte Ziffer konnte eine beliebige Zahl sein.
- 18** Ein einfaches Telefonsystem besteht aus zwei Teilnehmervermittlungsstellen und einer einzelnen Fernvermittlungsstelle, die mit jeder Teilnehmervermittlungsstelle über einen 1-MHz-Vollduplexanschluss verbunden ist. Im Durchschnitt werden auf jedem Telefon an einem achtstündigen Arbeitstag vier Anrufe getätigt. Die mittlere Dauer eines Telefonats liegt bei sechs Minuten. Zehn Prozent aller Gespräche sind Ferngespräche (d. h., sie durchlaufen die Fernvermittlungsstelle). Wie viele Telefone kann eine Teilnehmervermittlungsstelle maximal unterstützen? (Gehen Sie von 4 kHz je Leitung aus.) Erklären Sie, warum sich eine Telefongesellschaft entscheiden könnte, eine geringere Anzahl Telefone als dieses Maximum an der Vermittlungsstelle zu unterstützen.
- 19** Eine regionale Telefongesellschaft hat 10 Millionen Teilnehmer. Die Apparate aller Teilnehmer sind über eine Kupferleitung (Twisted Pair) an eine zentrale Vermittlungsstelle angeschlossen. Diese Twisted-Pair-Leitungen haben eine durchschnittliche Länge von 10 km. Wie viel ist der Kupferanteil in den Teilnehmeranschlussleitungen wert? Gehen Sie bei jedem Strang von einem Querschnitt von 1 mm², von einer spezifischen Dichte für Kupfer von 9,0 g/cm³ und von einem Kupferpreis von 6 US-Dollar pro kg aus.
- 20** Ist eine Öl-Pipeline ein Simplex-, ein Halbduplex-, ein Vollduplexsystem oder keines der hier genannten? Wie sieht es bei einem Fluss oder einer Kommunikation im Walkie-Talkie-Stil aus?
- 21** Der Preis für einen leistungsstarken Mikroprozessor ist derart stark gesunken, dass es heute möglich ist, Modems damit auszustatten. Welche Wirkung hat das auf die Handhabung von Fehlern in der Telefonleitung? Entfällt damit die Notwendigkeit zur Fehlererkennung/-behebung in Schicht 2?
- 22** Ein Modemkonstellationsdiagramm wie das in Abbildung 2.23 hat Datenpunkte mit den folgenden Koordinaten: (1,1), (1,-1), (-1,1) und (-1,-1). Wie viele Bit/s kann ein Modem mit diesen Parametern bei 1 200 Symbole/Sekunde erreichen?
- 23** Welches ist die maximale Bitrate, die in einem V.32-Standardmodem erreicht werden kann, wenn die Baudrate 1 200 beträgt und es keine Fehlerbehebung gibt?
- 24** Wie viele Frequenzen verwendet ein QAM-64-Duplexmodem?
- 25** Zehn Signale, von denen jedes 4 000 Hz erfordert, werden auf einem Kanal mit FDM gemultplexiert. Wie groß ist die minimale Bandbreite, die für den gemultiplexten Kanal erforderlich ist? Gehen Sie davon aus, dass die Sicherheitsfrequenzbänder 400 Hz breit sind.

- 26** Warum wurde bei PCM die Abtastrate auf $125 \mu\text{s}$ festgelegt?
- 27** Wie hoch ist der prozentuale Overhead auf einem T1-Träger? Das heißt, wie viel Prozent der 1,544 Mbit/s werden dem Endanwender vorerhalten? Wie hängt dies mit dem prozentualen Overhead in OC-1- oder OC-758-Leitungen zusammen?
- 28** Vergleichen Sie die maximale Datenrate eines rauschfreien 4-kHz-Kanals mit
 - analoger Codierung (wie QPSK) mit 2 Bit pro Abtastwert;
 - dem T1/PCM-System.
- 29** Unterläuft einem T1-Trägersystem ein Fehler und gerät aus der Synchronisation, versucht es, sich anhand des ersten Bits in jedem Rahmen wieder zu synchronisieren. Wie viele Rahmen müssen im Durchschnitt geprüft werden, damit die erneute Synchronisation mit einer Fehlerwahrscheinlichkeit von 0,001 klappt?
- 30** Was ist der Unterschied (falls es einen gibt) zwischen der Demodulationskomponente eines Modems und dem Coderteil eines Codecs? (Beide konvertieren schließlich analoge in digitale Signale.)
- 31** SONET-Uhren haben eine Abweichungszeit von 1 zu 109. Wie lange dauert es, bis die Abweichung der Breite eines Bits entspricht? Sehen Sie irgendwelche praktischen Auswirkungen dieser Berechnung? Falls ja, welche?
- 32** Wie lange wird es dauern, eine 1-GB-Datei von einem VSAT zum nächsten wie in Abbildung 2.17 mithilfe eines Hubs zu übertragen? Nehmen Sie an, dass der Uplink 1 Mbit/s, der Downlink 7 Mbit/s beträgt und Leitungsvermittlung mit 1,2 s Schaltungsaufbauzeit benutzt wird.
- 33** Berechnen Sie die Übertragungszeit in der vorigen Aufgabe, wenn stattdessen Paketvermittlung eingesetzt wird. Nehmen Sie an, dass das Paket 64 KB groß ist, die Schaltverzögerung im Satelliten und Hub $10 \mu\text{s}$ beträgt und die Größe des Paket-Headers 32 Byte ist.
- 34** In Abbildung 2.40 wird die Nutzdatenrate für OC-3 mit 148,608 Mbit/s angegeben. Zeigen Sie, wie diese Zahl aus den OC-3-Parametern des SONET-Systems abgeleitet werden kann. Welches sind die Gesamt-, SPE- und Nutzerdatenraten einer OC-3072-Leitung?
- 35** Um geringere Übertragungsraten als STS-1 zu unterstützen, verwendet SONET ein System von Virtual Tributaries (VT, virtueller Container). Ein VT enthält einen Teil der Nutzdaten und kann in einen STS-1-Rahmen eingefügt und dann mit anderen Nutzdatenanteilen kombiniert werden, um den Rahmen aufzufüllen. VT1.5 verwendet drei Spalten, VT2 vier Spalten, VT3 sechs Spalten und VT6 zwölf Spalten eines STS-1-Rahmens. Welcher Typ von VT kann Folgendes unterstützen:
 - einen DS-1-Dienst (1,544 Mbit/s)?
 - den europäischen CEPT-1-Dienst (2,048 Mbit/s)?
 - einen DS-2-Dienst (6,312 Mbit/s)?
- 36** Welche Bandbreite steht dem Benutzer in einer OC-12c-Verbindung zur Verfügung?

- 37** Drei paketvermittelte Netze enthalten je n Knoten. Das erste Netz hat eine Sterntopologie mit einem zentralen Vermittler, das zweite ist ein (bidirektonaler) Ring und beim dritten ist jeder Knoten mit jedem anderen Knoten durch ein Kabel verbunden. Ermitteln Sie die Anzahl der Teilstrecken im Übertragungsweg im bestmöglichen Fall, im Durchschnitt und im schlimmsten Fall.
- 38** Vergleichen Sie die Verzögerung, die bei der Übertragung einer Nachricht mit x Bit über ein leitungsvermitteltes Netz mit k Teilstrecken (Hops) und über ein gering ausgelastetes paketvermitteltes Netz entsteht. Die Leitungsaufbauzeit liegt bei s Sekunden, die Signalübertragungsverzögerung beträgt d Sekunden pro Teilstrecke, die Pakete sind p Bit groß und die Übertragungsrate beträgt b Bit/s. Unter welchen Voraussetzungen hat das paketvermittelte Netz eine geringere Verzögerung? Erläutern Sie auch die Bedingungen, unter denen ein paketvermitteltes Netzwerk einem leitungsvermittelten Netz vorzuziehen ist.
- 39** In einem paketvermittelten Netz sollen Nutzdaten mit x Bit über einen Pfad mit k Teilstrecken als Paketfolge übertragen werden. Jedes Paket enthält p Datenbit und h Header-Bit, wobei $x \gg p + h$ gilt. Die Übertragungsrate der Leitungen liegt bei b Bit/s, die Signalausbreitungsverzögerung ist verschwindend gering. Durch welchen Wert von p wird die Gesamtverzögerung minimiert?
- 40** In einem typischen Mobilfunksystem mit hexagonalen Zellen darf das Frequenzband einer benachbarten Zelle nicht wiederverwendet werden. Wie viele Frequenzen können in einer bestimmten Zelle benutzt werden, wenn insgesamt 840 Frequenzen verfügbar sind?
- 41** Das tatsächliche Layout einer Zelle ist selten so regelmäßig wie in Abbildung 2.45 dargestellt. Selbst die Formen einzelner Zellen sind normalerweise unregelmäßig. Geben Sie einen möglichen Grund hierfür an. Wie beeinflussen diese unregelmäßigen Formen die Frequenzzuweisung an jede Zelle?
- 42** Stellen Sie eine grobe Schätzung der Anzahl von PCS-Mikrozellen auf, die bei einer Reichweite von 100 m Durchmesser notwendig sind, um San Francisco (120 Quadratkilometer) abzudecken?
- 43** Überschreitet der Benutzer eines Mobiltelefons die Grenze von einer Zelle zu einer anderen, kann es vorkommen, dass die Verbindung abrupt abgebrochen wird, auch wenn alle Sender und Empfänger einwandfrei funktionieren. Warum?
- 44** Nehmen Sie an, dass A , B und C in einem CDMA-System mit den Chipfolgen von Abbildung 2.28a gleichzeitig 0-Bits übertragen. Welche Chipfolge ergibt sich?
- 45** Betrachten Sie das Merkmal der Orthogonalität von CDMA-Chipfolgen unter einem anderen Gesichtspunkt. Jedes Bit in einem Paar von Chipfolgen kann übereinstimmen oder nicht. Drücken Sie die Orthogonalität in Bezug auf Übereinstimmungen und Nichtübereinstimmungen aus.
- 46** Ein CDMA-Empfänger erhält folgende Chips: $(-1 + 1 - 3 + 1 - 1 - 3 + 1 + 1)$. Welche Stationen haben übertragen, und welche Bits hat jede davon gesendet, wenn man von den in Abbildung 2.28a definierten Chipfolgen ausgeht?
- 47** In Abbildung 2.28 gibt es vier Stationen, die übertragen können. Angenommen, es werden vier weitere Stationen hinzugefügt. Geben Sie die Chipfolgen für diese Stationen an.

- 48** In Richtung auf die Teilnehmer ist das Telefonsystem sternförmig, wobei alle Teilnehmeranschlussleitungen in einem Einzugsgebiet in einer Teilnehmervermittlungsstelle zusammenlaufen. Demgegenüber hat das Kabelfernsehen ein einzelnes langes Kabel, das sich durch alle Gebäude im gleichen Einzugsgebiet schlängelt. Nehmen Sie an, dass die Fernsehkabel der Zukunft aus einem Glasfaserkabel von 10 Gbit/s bestehen werden und nicht aus Kupfer. Könnte man hiermit das Telefonmodell simulieren, sodass jeder seine eigene Privatleitung zur Teilnehmervermittlungsstelle erhält? Falls ja, wie viele Häuser mit je einem Telefonanschluss könnten an einziges Glasfaserkabel angeschlossen werden?
- 49** Eine Kabelgesellschaft entscheidet sich, in einem Wohnviertel mit 5 000 Häusern den Internetzugang über Kabel bereitzustellen. Das Unternehmen verwendet ein Koaxialkabel und eine Frequenzbereichszuweisung, mit der 100 Mbit/s Downstream-Bandbreite möglich sind. Um Kunden anzuziehen, entscheidet das Unternehmen, mindestens 2 Mbit/s Bandbreite für den Empfang für jedes Haus zu jedem Zeitpunkt zu garantieren. Geben Sie an, was das Kabelunternehmen tun muss, um dies garantieren zu können.
- 50** Wenn man die Frequenzbereichszuweisung von Abbildung 2.52 und die Informationen aus dem Text zugrunde legt, wie viele Mbit/s muss ein Kabelsystem dann für die Upstream-Übertragung und wie viele für die Downstream-Übertragung zuweisen?
- 51** Wie schnell kann ein Kabelbenutzer Daten empfangen, wenn das Netz ansonsten im Leerlaufzustand ist? Nehmen Sie an, dass die Benutzerschnittstelle
- 10-Mbit/s-Ethernet
 - 100-Mbit/s-Ethernet
 - 54-Mbit/s drahtlos
- ist.
- 52** In SONET spielt das Multiplexing von mehreren STS-1-Datenströmen, sogenannten Tributaries, eine bedeutende Rolle. Ein 3:1-Multiplexer multiplext drei STS-1-Eingangs-Tributaries auf einen STS-3-Ausgangstrom. Das Multiplexing wird byteweise durchgeführt, also sind die ersten drei Ausgangsbytes die ersten Bytes der Tributaries 1, 2, und 3. Die nächsten drei Ausgangsbytes sind die zweiten Bytes der Tributaries 1, 2 und 3 und so weiter. Erstellen Sie ein Programm, das diesen 3:1-Multiplexer simuliert. Ihr Programm sollte aus fünf Prozessen bestehen. Der Hauptprozess erstellt vier Prozesse, je einen für die drei STS-1-Tributaries und einen für den Multiplexer. Jeder Tributary-Prozess liest einen STS-1-Rahmen aus einer Eingabedatei als Folge von 810 Byte ein. Sie senden ihre Rahmen (byteweise) an den Multiplexer-Prozess. Der Multiplexer-Prozess erhält diese Bytes und gibt einen STS-3-Rahmen (byteweise) aus, indem er ihn auf der Standardausgabe schreibt. Verwenden Sie für die Kommunikation zwischen den Prozessen Pipes.

- 53** Schreiben Sie ein Programm zur Implementierung von CDMA. Nehmen Sie an, dass die Länge einer Chipfolge acht und die Anzahl der übertragenden Stationen vier ist. Ihr Programm sollte aus drei Prozessmengen bestehen: vier Übertragungsprozesse (t_0, t_1, t_2 und t_3), ein Verbindungsprozess und vier Empfängerprozesse (r_0, r_1, r_2 und r_3). Das Hauptprogramm, das außerdem als Verbindungsprozess fungiert, liest zunächst vier Chipfolgen (binäre Notation) aus der Standardeingabe und eine Folge von 4 Bits (1 Bit pro Übertragungsprozess, der gesendet werden soll) ein. Dann werden vier Paare von Übertragungs- und Empfängerprozessen abgezweigt. Jedem Paar von Übertragungs-/Empfängerprozessen ($t_0, r_0; t_1, r_1; t_2, r_2; t_3, r_3$) wird eine Chipfolge zugewiesen und jedem Übertragungsprozess wird 1 Bit zugewiesen (erstes Bit zu t_0 , zweites Bit zu t_1 und so weiter). Als Nächstes berechnet jeder Übertragungsprozess das zu übermittelnde Signal (eine Folge von 8 Bits) und sendet es zum Verbindungsprozess. Nachdem Signale von allen vier Übertragungsprozessen empfangen wurden, fasst der Verbindungsprozess die Signale zusammen und sendet das kombinierte Signal an die vier Empfängerprozesse. Jeder Empfängerprozess berechnet dann das Bit, das es empfangen hat und sendet es zur Standardausgabe. Benutzen Sie Pipes zur Kommunikation zwischen Prozessen.

Die Sicherungsschicht

| | |
|---|-----|
| 3.1 Entwurfsaspekte der Sicherungsschicht | 235 |
| 3.2 Fehlererkennung und -korrektur | 244 |
| 3.3 Grundlegende Protokolle der Sicherungsschicht .. | 258 |
| 3.4 Schiebefensterprotokolle | 271 |
| 3.5 Beispiele für Protokolle der Sicherungsschicht .. | 291 |

» In diesem Kapitel befassen wir uns mit den Entwurfsprinzipien der zweiten Schicht unseres Modells, der Sicherungsschicht (*data link layer*). Wir untersuchen Algorithmen, die eine effiziente, zuverlässige Kommunikation unterstützen. Hier bewegen wir ganze Informationseinheiten, sogenannte Rahmen (anstelle von einzelnen Bits wie in der Bitübertragungsschicht), zwischen zwei benachbarten Rechnern. Der Begriff „benachbart“ bezieht sich auf zwei Rechner, die physikalisch über einen Übertragungskanal miteinander verbunden sind, der konzeptionell wie ein Kabel funktioniert (z.B. ein Koaxialkabel, eine Telefonleitung oder eine Funkverbindung). Das wesentliche Merkmal, damit sich ein Kanal wie ein Kabel verhält, ist die Zustellung der Bits in genau der gleichen Reihenfolge, in der sie gesendet wurden.

Zunächst könnte man denken, dieses Problem sei so unbedeutend, dass es überhaupt nichts zu untersuchen gibt – Rechner A stellt die Bits auf die Leitung und Rechner B entnimmt sie. Leider machen Übertragungskanäle zuweilen auch Fehler. Außerdem haben sie nur eine begrenzte Datenübertragungsrate und die Übertragungsverzögerung zwischen dem Zeitpunkt, an dem ein Bit gesendet wird, und dem Zeitpunkt, an dem es empfangen wird, kann nie null sein. Diese Einschränkungen haben entscheidende Auswirkungen auf die Effizienz des Datentransfers. Die für die Kommunikation verwendeten Protokolle müssen diese Faktoren berücksichtigen. Diese Protokolle sind Thema dieses Kapitels.

Nach einer Einführung in die wichtigsten Entwurfsfragen im Zusammenhang mit der Sicherungsschicht beginnen wir mit der Untersuchung der dazugehörigen Protokolle. Dazu betrachten wir die Art der Fehler und wie sie erkannt und korrigiert werden können. Dann behandeln wir eine Reihe von zunehmend komplexeren Protokollen, wobei jedes auch zunehmend mehr Probleme auf dieser Schicht löst. Wir beenden das Kapitel mit einigen Beispielen von Schicht-2-Protokollen.



3.1 Entwurfsaspekte der Sicherungsschicht

Die Sicherungsschicht benutzt die Dienste der Bitübertragungsschicht, um Bits über die Kommunikationskanäle zu senden und zu empfangen. Zu den Funktionen dieser Schicht gehören:

1. Bereitstellung einer wohldefinierten Dienstschnittstelle für die Vermittlungsschicht
2. Behandlung von Übertragungsfehlern
3. Regulierung des Datenflusses, damit langsame Empfänger nicht von schnellen Sendern mit Daten überschwemmt werden

Hierzu kapselt die Sicherungsschicht die von der Vermittlungsschicht erhaltenen Pakete für die Übertragung in **Rahmen** (frame). Jeder Rahmen enthält einen Rahmen-Header, ein Nutzdatenfeld, welches das Paket enthält, und einen Trailer (►Abbildung 3.1). Die zentrale Aufgabe der Sicherungsschicht ist die Verwaltung dieser Rahmen. Im Folgenden wollen wir uns diese Themen im Detail ansehen.

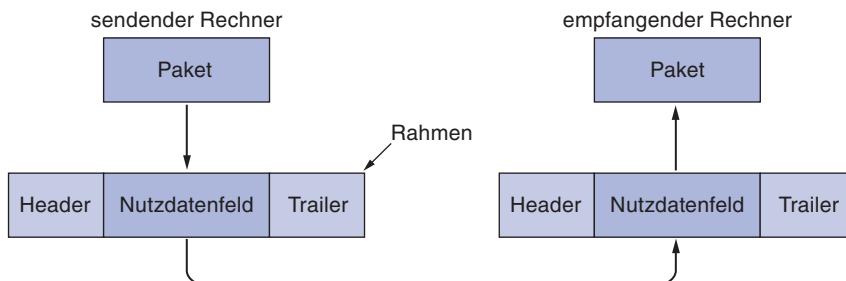


Abbildung 3.1: Beziehung zwischen Paketen und Rahmen.

Obwohl dieses Kapitel ausdrücklich die Sicherungsschicht und deren Protokolle betrifft, sind viele der hier untersuchten Prinzipien, z.B. Fehlerüberwachung und Flusskontrolle, auch auf die Transportschicht und auf andere Protokolle anwendbar. Das liegt daran, dass Zuverlässigkeit ein übergeordnetes Ziel ist, das erreicht wird, wenn alle Schichten zusammenarbeiten. Tatsächlich befinden sich in vielen Netzen diese Funktionen hauptsächlich in den oberen Schichten, wobei die Sicherungsschicht gerade so viel Funktionalität anbietet, wie unbedingt nötig ist. Unabhängig von ihrer tatsächlichen Position sind aber die Grundkonzepte nahezu identisch. Sie tauchen in ihrer einfachsten und reinsten Form oft in der Sicherungsschicht auf, hier ist also ein guter Ausgangspunkt für eine detaillierte Darstellung.

3.1.1 Dienste für die Vermittlungsschicht

Die Aufgabe der Sicherungsschicht ist es, der Vermittlungsschicht Dienste zur Verfügung zu stellen. Der Hauptdienst besteht in der Übertragung von Daten von der Vermittlungsschicht des Quellrechners zur Vermittlungsschicht des Zielrechners. Auf dem Quellrechner gibt es in der Vermittlungsschicht eine Instanz, die wir als Prozess bezeichnen, die einige Bits an die Sicherungsschicht zur Übertragung an das Ziel wei-

tergiert. Die Aufgabe der Sicherungsschicht ist die Übertragung von Bits zum Ziel, sodass sie dort wieder an die Vermittlungsschicht übergeben werden können (► Abbildung 3.2a). Die konkrete Datenübertragung geht wie in ► Abbildung 3.2b vonstatten. Einfacher kann man sich den Vorgang so vorstellen, dass zwei Sicherungsschichtprozesse über ein Schicht-2-Protokoll miteinander kommunizieren. Daher werden wir in diesem Kapitel implizit das Modell von Abbildung 3.2a verwenden.

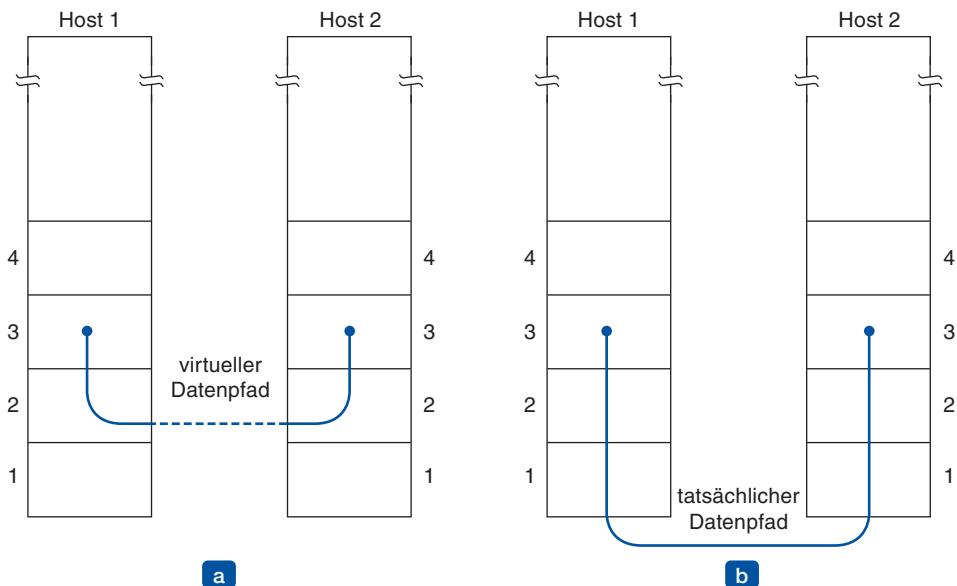


Abbildung 3.2: (a) Virtuelle Kommunikation. (b) Tatsächliche Kommunikation.

Die Sicherungsschicht kann so ausgelegt werden, dass sie verschiedene Dienste anbietet. Die tatsächlich gebotenen Dienste unterscheiden sich von einem System zum anderen. Drei sinnvolle Optionen, die wir nacheinander betrachten werden, sind:

1. unbestätigter verbindungsloser Dienst
2. bestätigter verbindungsloser Dienst
3. bestätigter verbindungsorientierter Dienst

Bei unbestätigten verbindungslosen Diensten sendet der Quellrechner unabhängige Rahmen an den Zielrechner, ohne dass dieser den Empfang bestätigen muss. Ethernet ist ein gutes Beispiel einer Sicherungsschicht, die diese Dienstklasse zur Verfügung stellt. Es wird weder vorher eine logische Verbindung aufgebaut noch anschließend wieder freigegeben. Wenn aufgrund von Rauschen ein Rahmen in der Leitung verloren geht, wird in der Sicherungsschicht nicht versucht, diesen Verlust zu entdecken und die Daten wiederherzustellen. Diese Dienstklasse ist angemessen, wenn die Fehlerquote sehr niedrig ist, die Datenwiederherstellung kann also höheren Schichten überlassen werden. Sie wird auch beim Echtzeitdatenverkehr (z.B. Sprachübertragung) angewendet, wo verspätete Daten schlimmer sind als zerstörte Daten.

Bestätigte verbindungslose Dienste sind hinsichtlich der Zuverlässigkeit der nächste Schritt. Wird dieser Dienst zur Verfügung gestellt, so werden zwar immer noch keine logischen Verbindungen verwendet, aber der Empfang jedes gesendeten Rahmens wird einzeln bestätigt. So weiß der Sender, ob sein Rahmen sicher angekommen ist oder verloren ging. Kommt der Rahmen nicht innerhalb eines festgesetzten Zeitraums an, kann er noch einmal gesendet werden. Der Dienst ist bei unzuverlässigen Kanälen wie drahtlosen Systemen nützlich. Ein gutes Beispiel für diese Dienstklasse ist IEEE 802.11 (WiFi).

Hier sollte vielleicht betont werden, dass das Bereitstellen von Bestätigungen auf der Sicherungsschicht nie erforderlich ist, sondern nur der Optimierung dient. Die Vermittlungsschicht kann immer eine Nachricht senden und warten, bis diese vom Peer auf dem entfernten Rechner bestätigt wird. Geht die Bestätigung nicht innerhalb einer bestimmten Zeit ein, kann der Sender die gesamte Nachricht erneut senden. Diese Strategie hat den Nachteil, dass sie eventuell ineffizient ist. Verbindungen haben in der Regel eine feste maximale Rahmenlänge, die über die Hardware festgelegt ist, und kennen Übertragungsverzögerungen. Die Vermittlungsschicht kennt diese Parameter jedoch nicht. Sie könnte ein großes Paket senden, das z.B. in zehn Rahmen zerlegt wird, von denen durchschnittlich zwei verloren gehen. Dann würde es sehr lange dauern, bis die Nachricht durchkommt. Wenn stattdessen einzelne Rahmen bestätigt und erneut übertragen werden, dann können Fehler direkter und schneller korrigiert werden. Auf zuverlässigen Kanälen, wie beispielsweise Glasfaser, kann der Overhead eines komplexen Sicherungsschichtprotokolls unnötig sein, wohingegen sich bei (inhärent unzuverlässigen) drahtlosen Kanälen die Investition lohnt.

Kommen wir zu unseren Diensten zurück. Verbindungsorientierte Dienste sind die ausgereiftesten Dienste, die der Vermittlungsschicht von der Sicherungsschicht zur Verfügung gestellt werden können. Kommen verbindungsorientierte Dienste zum Einsatz, so bauen Quell- und Zielmaschine vor jeder Datenübertragung eine Verbindung zueinander auf. Jeder der über diese Verbindung gesendeten Rahmen ist nummeriert und die Sicherungsschicht garantiert, dass jeder gesendete Rahmen auch ankommt. Sie garantiert ferner, dass jeder Rahmen genau einmal empfangen und die Reihenfolge bei der Übermittlung beibehalten wird. Der verbindungsorientierte Dienst bietet somit den Vermittlungsschichtprozessen einen Datenfluss in Form eines zuverlässigen Bitstroms. Wenn bestätigte verbindungslose Dienste verwendet werden, kann es vorkommen, dass verloren gegangene Bestätigungen das wiederholte Senden und Empfangen des Rahmens bewirken können, wodurch Bandbreite verschwendet wird.

Beim verbindungsorientierten Dienst ist die Datenübertragung in drei verschiedene Phasen eingeteilt. In der ersten Phase wird die Verbindung aufgebaut, indem Sender und Empfänger Variablen und Zähler initialisieren, um zu verfolgen, welche Rahmen angekommen sind und welche nicht. In der zweiten Phase werden Rahmen übertragen. In der dritten Phase wird die Verbindung getrennt und die Variablen, Puffer und anderen Ressourcen, durch die die Verbindung aufrechterhalten wurde, werden freigegeben.

3.1.2 Rahmenbildung

Damit die Sicherungsschicht der Vermittlungsschicht Dienste anbieten kann, muss sie selbst zuerst die Dienste der Bitübertragungsschicht in Anspruch nehmen. Die Bitübertragungsschicht nimmt einen rohen Bitstrom auf und versucht, ihn an einen Empfänger weiterzuleiten. Wenn der Kanal verrauscht ist, wie es bei den meisten drahtlosen und einigen Kabelverbindungen der Fall ist, dann wird die Bitübertragungsschicht den Signalen des Bitstroms Redundanz hinzufügen, um die Bitfehlerrate auf ein tolerables Maß zu reduzieren. Der Bitstrom, der von der Sicherungsschicht empfangen wird, ist jedoch nicht unbedingt fehlerfrei. Einige Bits können verschiedene Werte haben und die Anzahl der empfangenen Bits kann kleiner, gleich oder größer als die der gesendeten Bits sein. Es ist Aufgabe der Sicherungsschicht, Fehler aufzudecken und notfalls zu korrigieren.

Die übliche Vorgehensweise ist wie folgt: Die Sicherungsschicht unterteilt den Bitstrom in diskrete Rahmen, berechnet ein kurzes Token – die Prüfsumme – für jeden Rahmen und fügt die Prüfsumme in den Rahmen ein, wenn dieser übertragen wird. (Prüfsummenalgorithmen werden später in diesem Kapitel behandelt.) Sobald ein Rahmen seinen Bestimmungsort erreicht, wird die Prüfsumme neu errechnet. Wenn die neu berechnete Prüfsumme nicht mit der im Rahmen angegebenen übereinstimmt, so weiß die Sicherungsschicht, dass ein Fehler aufgetreten sein muss, und unternimmt die zur Behebung nötigen Schritte (z.B. Verwerfen des fehlerhaften Rahmens und Zurücksenden einer Fehlermeldung).

Die Aufteilung eines Bitstroms in Rahmen ist schwieriger, als es zunächst aussieht. Ein guter Entwurf muss es einem Empfänger leicht machen, den Anfang eines neuen Rahmens zu finden, ohne allzu viel der Kanalbandbreite zu benutzen. In diesem Abschnitt betrachten wir vier Methoden:

1. Bytezahl
2. Verwendung von Flagbytes mit Bytestopfen
3. Flagbits mit Bitstopfen
4. Codierungsverletzungen auf der Bitübertragungsschicht

Die erste Rahmenbildungsmethode verwendet ein Feld im Header, um die Bytezahl des Rahmens anzugeben. Sobald die Sicherungsschicht auf der Empfängerseite die Bytezahl liest, weiß sie, wie viele Bytes ankommen werden und wo folglich das Ende des Rahmens sein wird. Diese Technik wird in ▶ Abbildung 3.3a für vier Rahmen der Größen 5, 5, 8 und 8 Bytes aufgezeigt.

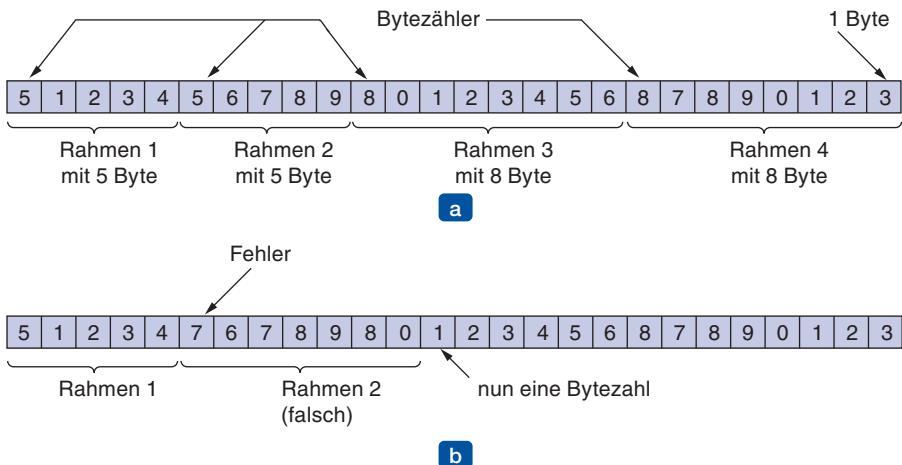


Abbildung 3.3: Ein Bytestrom: (a) ohne Fehler, (b) mit einem Fehler.

Das Problem bei diesem Algorithmus besteht darin, dass das Bytezählfeld durch einen Übertragungsfehler verändert werden kann. Wenn also z.B. die Anzahl 5 im zweiten Rahmen in ► Abbildung 3.3b bei der Übertragung infolge einer einzigen Bitvertauschung in eine 7 abgeändert wird, käme der Empfänger mit der Synchronisation durcheinander und könnte dann den korrekten Anfang des nächsten Rahmens nicht finden. Der Empfänger weiß zwar aufgrund der Prüfsumme, dass der Rahmen falsch ist, kann aber den Anfang des nächsten Rahmens nicht erkennen. Das Zurücksenden des Rahmens an den Sender mit der Aufforderung, ihn noch einmal zu übertragen, ist ebenfalls nutzlos, denn die Zielmaschine weiß nicht, wie viele Bytes ausgelassen werden müssen, um den Anfang der Wiederholungsübertragung zu finden. Aus diesem Grund wird heute die Bytezählmethode kaum noch alleine angewendet.

Bei der zweiten Rahmenbildungsmethode wird das Problem mit der erneuten Synchronisation nach einem Fehler vermieden, indem spezielle Bytes zu Beginn und am Ende eines Rahmens verwendet werden. Oft wird das gleiche Byte, ein sogenanntes **Flagbyte**, als Begrenzungszeichen sowohl am Anfang als auch am Ende des Rahmens verwendet. Dieses Byte wird in ► Abbildung 3.4a als FLAG bezeichnet. Zwei aufeinanderfolgende Flagbytes markieren das Ende eines Rahmens und den Anfang des nächsten. Wenn der Empfänger also jemals aus der Synchronisation gerät, muss er nur nach zwei Flagbytes suchen, um das Ende des aktuellen und den Anfang des nächsten Rahmens zu finden.

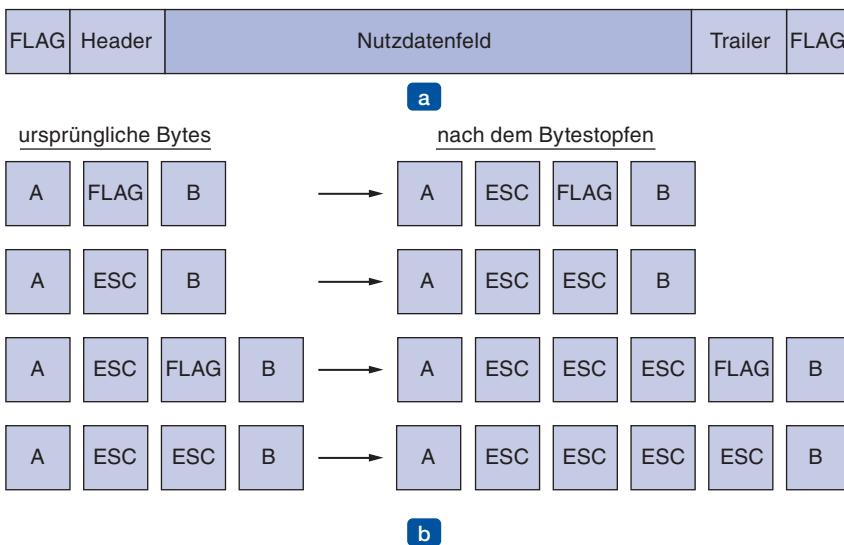


Abbildung 3.4: (a) Mit Flagbytes markierter Rahmen. (b) Vier Beispiele von Bytefolgen vor und nach dem Bytestopfen.

Es gibt jedoch noch ein ernstes Problem, das wir zu lösen haben. Es kann vorkommen, dass das Flagbyte in den Daten vorkommt, insbesondere wenn Binärdaten wie Fotos oder Musik übertragen werden. Das würde natürlich mit der Rahmenbildung kollidieren. Eine Lösung wäre, dass die Sicherungsschicht des Absenders ein spezielles Escape-Byte (ESC) vor jedem „zufälligen“ Flagbyte in den Daten einfügt. Ein Flagbyte zur Kennzeichnung eines Rahmens kann dann von einem Flagbyte in den Daten dadurch unterschieden werden, dass bei Letzterem ein Escape-Byte davor steht. Die Sicherungsschicht auf der Empfängerseite nimmt das Escape-Byte heraus, bevor es die Daten an die Vermittlungsschicht übergibt. Diese Technik wird **Bytestopfen** (*byte stuffing*) genannt.

Natürlich stellt sich die nächste Frage: Was geschieht, wenn eine Escape-Byte mitten in den Daten vorkommt? Die Antwort ist, dass es ebenfalls mit einem Escape-Byte aufgefüllt wird. Der Empfänger entfernt das erste Escape-Byte und belässt das folgende Datenbyte (welches ein weiteres Escape-Byte oder das Flagbyte sein könnte). Einige Beispiele hierfür finden Sie in ► Abbildung 3.4b. In jedem Fall ist die Bytefolge nach dem Entfernen der Füllbytes genau die gleiche wie in der ursprünglichen Bytefolge. Wir können weiterhin nach einer Rahmengrenze suchen, indem wir nach zwei Flagbytes hintereinander Ausschau halten, und müssen uns nicht darum kümmern, Escapes-Bytes zu löschen.

Die Methode des Bytestopfens in ► Abbildung 3.4 ist eine leichte Vereinfachung des Schemas, das im Protokoll **PPP** (*Point-to-Point Protocol*) verwendet wird, welches zur Übertragung von Paketen über Kommunikationsverbindungen eingesetzt wird. Auf PPP kommen wir gegen Ende dieses Kapitels noch einmal zu sprechen.

Die dritte Methode zur Begrenzung des Bitstroms umgeht einen Nachteil von Bytestopfen, nämlich dass dieser an die Verwendung von 8-Bit-Bytes gebunden ist. Rahmenbildung kann auch auf Bitebene stattfinden, sodass Rahmen eine beliebige Anzahl von

Bits enthalten können, die aus Einheiten jeder Größe zusammengesetzt sind. Dieses Vorgehen wurde für das früher häufig eingesetzte **HDLC**-Protokoll (*Highlevel Data Link Control*) entwickelt. Jeder Rahmen beginnt und endet mit einem speziellen Bitmuster, nämlich 01111110 bzw. 0x7E in hexadezimaler Schreibweise. Dieses Muster ist ein Flagbyte. Sobald die Sicherungsschicht des Senders fünf aufeinanderfolgende Einsen im Datenstrom entdeckt, füllt sie den abzusendenden Bitstrom automatisch mit einer Null auf. Dieses **Bitstopfen** (*bit stuffing*) entspricht dem Bytestopfen, bei dem im ausgehenden Datenstrom vor dem Flagbyte ein Escape-Byte eingefügt wird. Dadurch wird außerdem sichergestellt, dass eine minimale Dichte von Übergängen vorhanden ist, welche der Bitübertragungsschicht bei der Aufrechterhaltung der Synchronisation helfen. USB (*Universal Serial Bus*) setzt Bitstopfen aus diesem Grund ein.

Sobald der Empfänger eine Folge von fünf Einsen gefolgt von einer Null im ankommenden Datenstrom erkennt, nimmt er das Nullbit automatisch aus dem Datenstrom heraus. Genau wie das Bytestopfen in der Vermittlungsschicht beider Rechner ist auch das Bitstopfen völlig transparent. Wenn die Benutzerdaten das Flagmuster 01111110 enthalten, wird das Flag als 01111101 übertragen, aber im Speicher des Empfängers als 01111110 abgelegt. ►Abbildung 3.5 zeigt ein Beispiel für Bitstopfen.

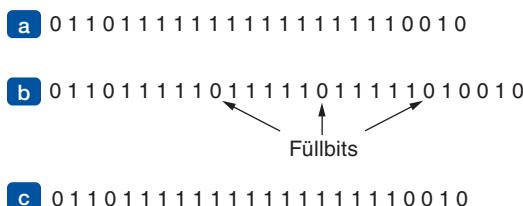


Abbildung 3.5: Bitstopfen: (a) Originaldaten. (b) Daten, wie sie während der Übertragung dargestellt werden. (c) Daten, wie sie nach dem Entfernen der Füllbits im Speicher des Empfängers gespeichert werden.

Beim Bitstopfen können die Rahmengrenzen ganz eindeutig anhand des Flagmusters erkannt werden. Wenn also der Empfänger einmal seine Position verliert, kann er ganz einfach die ankommenden Daten nach bestimmten Flagfolgen absuchen. Diese können sich nämlich nur an den Rahmengrenzen, nie innerhalb der Daten befinden.

Sowohl beim Bit- als auch beim Bytestopfen gibt es den Nebeneffekt, dass die Länge eines Rahmens nun von den Inhalten der übertragenen Daten abhängt. Wenn es beispielsweise keine Flagbytes in den Daten gibt, dann könnten 100 Bytes in einem rund 100 Byte großen Rahmen übertragen werden. Falls jedoch die Daten ausschließlich aus Flagbytes bestehen, wird für jedes Flagbyte ein Escape-Byte eingefügt und der Rahmen wird ungefähr 200 Byte lang sein. Beim Bitstopfen werden die Daten um rund 12,5 % verlängert, da zu jedem Byte jeweils 1 Bit hinzugefügt wird.

Bei der letzten Methode der Rahmenbildung nutzen wir einen Nebeneffekt, den wir direkt aus der Bitübertragungsschicht erhalten. Wir haben in *Kapitel 2* gesehen, dass beim Codieren von Bits zu Signalen häufig Redundanzen eingefügt werden, um den Empfang zu unterstützen. Diese Redundanz bedeutet, einige Signale tauchen nicht in regulären Daten auf. Zum Beispiel werden im 4B/5B-Leitungscode 4 Datenbits auf

5 Signalbits abgebildet, um ausreichende Bitübergänge sicherzustellen. Das heißt, dass 16 von den 32 möglichen Signalen nicht genutzt werden. Wir können nun einige dieser reservierten Signale verwenden, um Rahmenanfang und -ende zu markieren. Eigentlich handelt es sich hierbei um „Codierungsverletzungen“, die wir zur Begrenzung von Rahmen einsetzen. Der Reiz dieses Schemas liegt darin, dass es einfach ist, den Anfang und das Ende von Rahmen zu finden, weil es sich um reservierte Signale handelt, und es ist nicht notwendig, die Daten zu „stopfen“.

Viele Protokolle in der Sicherungsschicht verwenden aus Sicherheitsgründen eine Kombination dieser Methoden. Ein übliches Muster, das für Ethernet und WiFi eingesetzt wird, ist es, an den Anfang eines Rahmens ein wohldefiniertes Muster, die sogenannte **Präambel**, zu stellen. Dieses Muster kann recht lang sein (72 Bits ist die Regel bei WiFi), sodass sich der Empfänger auf ein ankommendes Paket vorbereiten kann. Nach der Präambel kommt dann ein Längenfeld (z.B. Zählfeld) im Header, welches benutzt wird, um das Ende des Rahmens zu ermitteln.

3.1.3 Fehlerüberwachung

Nachdem das Problem der Markierung von Anfang und Ende des Rahmens gelöst wurde, kommen wir zum nächsten Problem: Wie wird sichergestellt, dass alle Rahmen wirklich an die Vermittlungsschicht des Empfängers übertragen werden und dass die Reihenfolge eingehalten wird? Nehmen wir für den Moment an, dass der Empfänger entscheiden kann, ob ein ankommender Rahmen korrekte oder fehlerhafte Information enthält (wir werden uns die Codes zur Erkennung und Korrektur von Übertragungsfehlern in Abschnitt 3.2 ansehen).

Für einen unbestätigten verbindungslosen Dienst mag es vielleicht reichen, wenn der Sender einfach weiterhin Rahmen abschickt, ohne zu prüfen, ob sie richtig ankommen. Aber für einen zuverlässigen verbindungsorientierten Dienst wäre dies ganz und gar nicht ausreichend.

Im Normalfall wird die zuverlässige Übermittlung dadurch gewährleistet, dass der Sender eine Bestätigung über die Vorgänge am anderen Ende der Leitung erhält. Grundsätzlich erfordert das Protokoll, dass der Empfänger spezielle Steuerrahmen an den Sender zurücksendet, die positive oder negative Bestätigungen über die ankommenden Rahmen enthalten. Wenn also der Sender eine positive Bestätigung über den gesendeten Rahmen erhält, weiß er, dass dieser sicher angekommen ist. Erhält er andererseits eine negative Bestätigung, bedeutet das, dass etwas schiefgelaufen ist und der Rahmen noch einmal übertragen werden muss.

Zusätzliche Komplikationen treten dadurch auf, dass Hardwarestörungen einen Rahmen verschwinden lassen können (z.B. in einem länger anhaltenden Rauschen). In einem solchen Fall reagiert der Empfänger überhaupt nicht. Er weiß ja nicht, dass er reagieren sollte. Außerdem wird der Sender nicht wissen, wie er weiter verfahren soll, wenn der bestätigte Rahmen verloren geht. Damit dürfte klar sein, dass ein Protokoll, in dem ein Absender einen Rahmen gesendet hat und dann auf eine Bestätigung (positiv

oder negativ) wartet, für immer blockiert wäre, wenn ein Rahmen z.B. aufgrund eines Hardwarefehlers oder eines fehlerhaften Übertragungskanals vollständig verloren ginge.

Dieser Fall wird durch die Einführung von Timern in der Sicherungsschicht abgedeckt. Wenn der Sender einen Rahmen überträgt, wird der Timer gestartet. Der Timer wird so eingestellt, dass er nach einer Zeit abläuft, die lang genug dafür ist, dass der Rahmen sein Ziel erreichen, dort verarbeitet werden und eine Rückmeldung den Sender erreichen kann. Normalerweise wird der Rahmen korrekt empfangen und die Bestätigung vor Ablauf des Timers zurückgesendet. In diesem Fall wird der Timer abgeschaltet.

Falls jedoch der Rahmen oder die Bestätigung verloren gehen, läuft der Timer ab und macht den Sender auf mögliche Probleme aufmerksam. Die einfachste Lösung ist, den Rahmen noch einmal zu übertragen. Wenn jedoch ein Rahmen mehrmals gesendet wird, besteht die Gefahr, dass der Empfänger den gleichen Rahmen mehrmals annimmt und damit auch mehrmals an die Vermittlungsschicht weitergibt. Um dieser Situation vorzubeugen, vergibt man normalerweise Sequenznummern für die zu sendenden Rahmen, damit der Empfänger Originale von Wiederholungen unterscheiden kann.

Die Verwaltung von Timern und Sequenznummern zur Sicherstellung, dass jeder Rahmen schlussendlich genau einmal – nicht mehr und nicht weniger – an die Vermittlungsschicht des Ziels weitergegeben wird, ist eine wichtige Aufgabe der Sicherungsschicht (und höherer Schichten). Später in diesem Kapitel wird die Verwaltung anhand zunehmend komplexer werdender Beispiele genauer betrachtet.

3.1.4 Flusskontrolle

Ein weiteres wichtiges Entwurfskriterium der Sicherungsschicht (und ebenso der darüberliegenden Schichten) ist, wie man mit einem Sender umgehen soll, der die Rahmen ständig schneller übertragen möchte, als sie der Empfänger aufnehmen kann. Diese Situation kann auftreten, wenn der Sender ein schneller, leistungsstarker Rechner und der Empfänger ein langsamer, einfacher Rechner ist. Ein typisches Szenario ist, wenn ein Smartphone eine Webseite von einem sehr viel leistungsstärkeren Server aufruft, der daraufhin seine Schleusen öffnet und die Daten auf das arme, hilflose Telefon loslässt, bis dieses förmlich in einer Datenwelle ertrinkt. Auch wenn die Übertragung fehlerfrei abläuft, könnte der Empfänger nicht mehr in der Lage sein, alle ankommenden Rahmen sofort abzuarbeiten, und verliert deshalb einige. Für solche Fälle müssen natürlich Vorkehrungen getroffen werden. Hier werden in der Regel zwei Ansätze verwendet. Im ersten Ansatz, der **feedbackbasierten Flusskontrolle** (*feedback-based flow control*), sendet der Empfänger Informationen an den Sender zurück und erteilt damit die Erlaubnis, weitere Daten zu senden, oder informiert den Sender zumindest, wie es dem Empfänger geht. Beim zweiten Ansatz, der **Flusskontrolle basierend auf der Übertragungsrate** (*rate-based flow control*), besitzt das Protokoll einen integrierten Mechanismus, der die Übertragungsgeschwindigkeit beschränkt, mit der der Sender Daten überträgt, ohne dass hierzu eine Rückmeldung vom Empfänger erforderlich ist.

In diesem Kapitel beschäftigen wird uns mit der feedbackbasierten Flusskontrolle, hauptsächlich deshalb, weil Schemata, die auf der Übertragungsrate basieren, nur als Teil der Transportschicht verwendet werden (*Kapitel 5*). Feedbackbasierte Verfahren kommen sowohl in Sicherungsschichten als auch in höheren Schichten vor. Letzteres ist heutzutage üblicher. In diesem Fall wird die Hardware der Sicherungsschichten so konzipiert, dass sie schnell genug ist, um keinen Ausfall zu verursachen. Wenn beispielsweise die Hardware der Sicherungsschicht als **Netzwerkkarte (NIC, Network Interface Card)** implementiert wird, sagt man, dass die Hardware mit „Kabelgeschwindigkeit“ läuft. Das bedeutet, sie kann die Rahmen so schnell bearbeiten, wie diese auf der Leitung ankommen. Überläufe sind dann kein Verbindungsproblem, sie können also von höheren Schichten behandelt werden.

Es gibt verschiedene feedbackbasierte Flusskontrollverfahren. Die meisten funktionieren jedoch nach dem gleichen Grundprinzip. Das Protokoll besitzt wohldefinierte Regeln, wann ein Sender den nächsten Rahmen senden darf. Diese Regeln verbieten es dem Sender, Rahmen ohne ausdrückliche oder implizite Erlaubnis des Empfängers zu senden. Wenn z.B. eine Verbindung aufgebaut wird, könnte der Empfänger sagen: „Du kannst mir jetzt n Rahmen senden, aber danach sende so lange nichts mehr, bis ich dir mitteile, dass du wieder senden kannst.“ Die Details dazu untersuchen wir in Kürze.

3.2 Fehlererkennung und -korrektur

Wir haben in *Kapitel 2* gesehen, dass Kommunikationskanäle eine Reihe von Charakteristika haben. Einige Kanäle wie Glasfaser in Telekommunikationsnetzen haben winzige Fehlerraten, sodass Übertragungsfehler nur sehr selten vorkommen. Doch andere Kanäle, insbesondere drahtlose Verbindungen und alte Teilnehmeranschlussleitungen haben Fehlerraten, die um ein Vielfaches höher sind. Für solche Verbindungen sind Übertragungsfehler die Norm. Diese Fehler können nur mit hohem Aufwand oder Kosten hinsichtlich der Leistungsfähigkeit vermieden werden. Die Folge davon ist, dass Übertragungsfehler hier eine dauerhafte Erscheinung sind. Wir müssen lernen, mit ihnen umzugehen.

Netzentwickler haben zwei Grundstrategien für die Fehlerbehandlung entwickelt. Beide fügen den gesendeten Daten redundante Information hinzu. Eine Strategie ist, genug redundante Informationen zu senden, dass der Empfänger daraus ableiten kann, wie die übertragenen Daten ausgesehen haben müssen. Bei der anderen Methode wird nur so viel Redundanz mitgesendet, dass der Empfänger das Auftreten eines Fehlers feststellen (nicht aber die Art des Fehlers) und eine erneute Übertragung anfordern kann. Die erste Strategie verwendet **Fehlerkorrekturcodes**, die zweite **Fehlererkennungscodes**. Die Verwendung von Fehlerkorrekturcodes wird häufig auch als **Vorwärtsfehlerkorrektur (FEC, Forward Error Correction)** bezeichnet.

Jede dieser Techniken belegt eine eigene Nische. Bei extrem zuverlässigen Kanälen mit niedrigen Übertragungslatenzen wie Glasfaser ist es günstiger, einen Fehlererkennungscode zu verwenden und die gelegentlich fehlerhaften Blöcke erneut zu übertra-

gen. Bei Kanälen mit vielen Fehlern wie Funkverbindungen ist es jedoch besser, jeden Block mit Redundanz auszustatten, sodass der Empfänger ermitteln kann, wie der ursprünglich übermittelte Block ausgesehen hat. FEC wird auf rauschbehafteten Kanälen benutzt, da hier eine erneute Übertragung mit der gleichen Wahrscheinlichkeit fehlerhaft ist wie die erste Übertragung.

Bei diesen Codes müssen wir uns als Erstes überlegen, welcher Art die Fehler sind, die wahrscheinlich auftreten werden. Weder Fehlerkorrekturcodes noch Fehlererkennungscodes können alle möglichen Arten von Fehlern behandeln, da die redundanten Bits, die als Schutzmaßnahme eingefügt werden, mit derselben Wahrscheinlichkeit fehlerhaft empfangen werden wie die Datenbits (wodurch deren Schutz beeinträchtigt werden kann). Es wäre nett, wenn der Kanal redundante Bits anders als Datenbits behandeln würde, doch er tut es nicht. Für den Kanal sind es alles einfach nur Bits. Das heißt, wenn man verhindern möchte, dass Fehler unentdeckt bleiben, muss der Code stark genug sein, um die erwarteten Fehler zu behandeln.

Ein Szenario besteht darin, dass Fehler durch extreme Werte von thermischem Rauschen verursacht werden, wodurch das Signal von Zeit zu Zeit jeweils kurz überlagert wird, was zu isolierten Einzelbitfehlern führt. In einem anderen Fall gibt es die Tendenz, dass Fehler eher gehäuft, in sogenannten Bursts, statt einzeln vorkommen. Dieses Modell leitet sich aus den physikalischen Prozessen ab, welche die Fehler erzeugen – so wie starke Schwankungen auf einem drahtlosen Kanal oder transiente elektrische Interferenzen auf einem verkabelten Kanal.

Beide Herangehensweisen haben praktische Bedeutung und basieren auf unterschiedlichen Abwägungen. Wenn die Fehler in Bursts und nicht als einzelne Bitfehler auftreten, hat dies Vor- und Nachteile. Der Vorteil ist: Computerdaten werden immer in Bitblöcken übertragen. Nimmt man eine Blockgröße von 1 000 Bit und eine Fehlerrate von 0,001 pro Bit an, dann wäre in den meisten Blöcken ein Fehler enthalten, wenn die Fehler einzeln auftreten würden. Treten die Fehler beispielsweise in Hunderthaufen auf, wären im Durchschnitt höchstens zwei Blöcke von 100 betroffen. Der Nachteil des gruppenweisen Auftretens von Fehlern liegt darin, dass sie im Vergleich zu vereinzelten Fehlern weniger leicht erkannt und korrigiert werden können.

Darüber hinaus gibt es noch andere Fehlerarten. Manchmal ist die Position des Fehlers bekannt, vielleicht weil die Bitübertragungsschicht ein analoges Signal empfangen hat, das weit vom erwarteten Wert für eine 0 oder 1 abwich, und daher das Bit als verloren erklärt hat. Diese Situation wird **Auslöschungskanal** (*erasure channel*) genannt. Es ist einfacher, Fehler in Auslöschungskanälen zu beheben als in Kanälen, die Bits kippen. Das liegt daran, dass selbst wenn der Wert der Bits verloren ist, wir wenigstens wissen, welches Bit fehlerhaft ist. Leider haben wir diesen Vorteil der Auslöschungskanäle nicht häufig.

Als Nächstes werden wir sowohl Fehlerkorrekturcodes als auch Fehlererkennungscodes untersuchen. Bitte behalten Sie dabei zwei Punkte im Hinterkopf. Erstens behandeln wir diese Codes in der Sicherungsschicht, weil dies die erste Stelle ist, an der wir das Problem der zuverlässigen Übertragung von Bitgruppen angehen. Die Codes wer-

den weitverbreitet eingesetzt, weil Zuverlässigkeit ein übergreifendes Anliegen ist. Fehlerkorrekturcodes kommen auch auf der Bitübertragungsschicht vor, besonders für rauschbehaftete Kanäle, ebenso wie in höheren Schichten, insbesondere bei Echtzeitmedien und bei der Verbreitung von Inhalten. Fehlererkennungscodes werden üblicherweise in Sicherungs-, Vermittlungs- und Transportschichten eingesetzt.

Der zweite Punkt, den es zu berücksichtigen gilt, ist die Tatsache, dass Fehlercodes angewandte Mathematik sind. Sofern Sie nicht besonders versiert mit Galois-Körpern oder den Eigenschaften von dünnbesetzten Matrizen sind, sollten Sie es vorziehen, Codes mit guten Eigenschaften von einer zuverlässigen Quelle zu beziehen anstatt Ihren eigenen Code zu entwickeln. Dies ist in der Tat die Vorgehensweise vieler Protokollstandards, daher tauchen die gleichen Codes immer und immer wieder auf. Im weiteren Verlauf des Kapitels werden wir einen einfachen Code ausführlich untersuchen und anschließend verschiedene Erweiterungen davon jeweils kurz beschreiben. Auf diese Art können wir die oben erwähnten Abwägungen anhand des einfachen Codes nachvollziehen und über die Eigenschaften von praxisrelevanten Codes anhand der Erweiterungen sprechen.

3.2.1 Fehlerkorrekturcodes

Wir werden vier unterschiedliche Fehlerkorrekturcodes untersuchen:

1. Hamming-Codes
2. Binäre Faltungscodes
3. Reed-Solomon-Codes
4. LDPC-Codes (*Low-Density Parity Check*)

Alle diese Codes fügen den gesendeten Bits redundante Informationen hinzu. Ein Rahmen besteht aus m Datenbits (Nachrichtenbits) und r redundanten Bits (Prüfbits). In einem **Blockcode** werden r Prüfbits allein als Funktion der m zugehörigen Datenbits berechnet, so als würden die m Bits in einer großen Tabelle nachgeschlagen, um die entsprechenden r Prüfbits zu finden. In einem **systematischen Code** werden die m Bits direkt, ohne weitere Codierung zusammen mit den Prüfbits gesendet. Bei einem **linearen Code** werden die r Prüfbits als lineare Funktion der m Datenbits berechnet. Hierzu wird gerne die XOR-Verknüpfung oder die Modulo-2-Addition gewählt. Das heißt, die Codierung kann mit Operationen wie der Matrixmultiplikation oder mithilfe einfacher logischer Schaltkreise durchgeführt werden. Die Codes, die wir uns in diesem Abschnitt anschauen, sind – soweit nichts anderes gesagt wird – lineare systematische Blockcodes.

Die Gesamtlänge eines Blocks sei n (d.h. $n=m+r$). Wir werden dies als einen (n,m) -Code bezeichnen. Eine n -Bit-Einheit, die Daten- und Prüfbits enthält, wird **n -Bit-Codewort** genannt. Die **Codrate** oder kurz Rate ist der Teil des Codeworts, das die nicht redundante Information trägt, oder m/n . Die in der Praxis verwendeten Raten variieren beträchtlich. Sie könnten $1/2$ bei einem verrauschten Kanal sein – hier ist also die

Hälften der empfangenen Information redundant – oder fast 1 für hochwertige Kanäle, bei denen nur eine kleine Anzahl von Prüfbits einer langen Nachricht hinzugefügt werden.

Zum Verständnis der Fehlerbehandlung muss man erst einmal genau betrachten, was ein Fehler ist. Vergleicht man zwei Codewörter, die übertragen oder gesendet werden können, beispielsweise 10001001 und 10110001, so kann man feststellen, wie viele Bits nicht übereinstimmen. In diesem Fall sind drei Bits verschieden. Um festzustellen, wie viele Bits nicht übereinstimmen, verknüpft man die zwei Codewörter mit einem exklusiven ODER (XOR) und zählt, wie oft ein 1-Bit im Ergebnis vorkommt. Zum Beispiel:

$$\begin{array}{r} 10001001 \\ \text{XOR} \quad \underline{10110001} \\ 00111000 \end{array}$$

Die Anzahl der Bitpositionen, in denen sich zwei Codewörter unterscheiden, wird als **Hamming-Abstand** bezeichnet (Hamming, 1950). Haben zwei Codewörter einen Hamming-Abstand von d , müssen d Einzelbitfehler vorkommen, um ein Wort in das andere umzuwandeln.

Mithilfe des Algorithmus zur Berechnung der Prüfbits ist es möglich, eine komplette Liste aller zulässigen Codewörter zu erstellen. Dieser Liste können nun die beiden Codewörter mit dem kleinsten Hamming-Abstand entnommen werden. Dieser Abstand ist der Hamming-Abstand des gesamten Codes.

Bei den meisten Datenübertragungsanwendungen sind alle 2^m möglichen Datennachrichten zulässig. Aufgrund der Berechnungsweise der Prüfbits werden aber nicht alle 2^n möglichen Codewörter verwendet. Bei r Prüfbits wird in der Tat nur ein Bruchteil von $2^m/2^n$ oder $1/2^r$ der möglichen Nachrichten ein zulässiges Codewort sein. Aufgrund dieser geringen Dichte, mit der die Nachricht in den Raum der Codewörter eingebettet ist, kann der Empfänger Fehler erkennen und beheben.

Die Fehlererkennungs- und -korrektureigenschaften eines Blockcodes hängen von seinem Hamming-Abstand ab. Zum zuverlässigen Auffinden von d Fehlern benötigt man einen Code mit Abstand $d+1$. Nur so kann sichergestellt werden, dass d Einzelbitfehler kein gültiges Codewort in ein anderes gültiges umwandeln. Findet der Empfänger ein ungültiges Codewort, kann er sofort erkennen, dass ein Übertragungsfehler aufgetreten sein muss. Zur Behebung von d Fehlern braucht man einen Code mit Abstand von $2d+1$. Damit sind die zulässigen Codewörter so weit voneinander entfernt, dass das Originalcodewort auch im Falle von d Änderungen immer noch näher liegt als jedes andere. Vorausgesetzt, eine größere Fehleranzahl ist weniger wahrscheinlich, so bedeutet dies, dass das ursprüngliche Codewort eindeutig erkannt werden kann.

Als einfaches Beispiel für einen Fehlerkorrekturcode betrachten wir einen Code, der nur vier gültige Codewörter hat:

0000000000, 0000011111, 1111100000 und 1111111111

Dieser Code hat einen Abstand von 5, kann also Doppelfehler korrigieren oder Vierfachfehler erkennen. Wenn das Codewort 0000000111 ankommt und wir Einfach- oder Doppelbitfehler erwarten, dann weiß der Empfänger, dass das Original 0000011111 gewesen sein muss. Wenn aber ein dreifacher Fehler 0000000000 in 0000000111 abändert, wird der Fehler nicht richtig korrigiert. Wenn wir andererseits alle diese Fehler erwarten, dann können wir sie entdecken. Keines der empfangenen Codewörter ist zulässig, deshalb muss ein Fehler aufgetreten sein. Es sollte offensichtlich sein, dass wir in diesem Beispiel nicht gleichzeitig Doppelfehler korrigieren und Vierfachfehler entdecken können. Dies hieße, dass wir ein empfangenes Codewort auf zwei unterschiedliche Arten interpretieren müssten.

Die Aufgabe des Decodierens durch Auffinden eines zulässigen Codeworts, das dem empfangenen Codewort am nächsten kommt, kann in unserem Beispiel einfach abgelesen werden. Unglücklicherweise kann diese Aufgabe im allgemeinen Fall, in dem alle Codewörter als mögliche Kandidaten infrage kommen und bewertet werden müssen, zu einer zeitaufwendigen Suche werden. Daher werden Codes in der Praxis so entworfen, dass Abkürzungen zugelassen sind, um das Codewort zu finden, welches vermutlich das ursprüngliche war.

Angenommen, wir wollen einen Code mit m Nachrichtenbits und r Prüfbits so zusammenstellen, dass alle Einzelfehler korrigiert werden können. Jede der 2^m erlaubten Nachrichten beinhaltet n unerlaubte Codewörter mit einem Abstand von 1. Diese werden erstellt, indem systematisch jedes der n Bits in dem aus ihm erstellten n -Bit-Codewort invertiert wird. Folglich benötigt jede der 2^m erlaubten Nachrichten $n+1$ zugeteilte Bitmuster. Da die Gesamtzahl der Bitmuster 2^n ist, brauchen wir $(n+1) \leq 2^m \leq 2^n$. Mit $n=m+r$ wird diese Anforderung zu

$$(m+r+1) \leq 2^r \quad (3.1)$$

Mit gegebenem m ist dies eine Untergrenze für die Zahl der Prüfbits, die zur Korrektur von Einzelfehlern benötigt werden.

Diese theoretische Untergrenze kann in der Praxis mit einer Methode nach Hamming (1950) erreicht werden. Bei **Hamming-Codes** werden die Bits des Codewortes beginnend mit Bit 1 auf der linken Seite, Bit 2 rechts daneben etc. durchnummieriert. Die Bits, die Potenzen von 2 sind (1, 2, 4, 8, 16 usw.), sind Prüfbits. Die restlichen (3, 5, 6, 7, 9 usw.) werden mit den m Datenbits aufgefüllt. Dieses Muster ist für einen (11,7)-Hamming-Code mit 7 Datenbits und 4 Prüfbits in ▶ Abbildung 3.6 dargestellt. Jedes Prüfbit erzwingt die Modulo-2-Summe oder Parität einer bestimmten Bitreihe (zu der auch das Prüfbit selbst gehört) auf gerade (oder ungerade). Ein Bit kann in mehrere Prüfbitberechnungen aufgenommen werden. Um zu sehen, zu welchem Prüfbit das Datenbit in Position k gehört, wird k als Summe von Zweierpotenzen umgeschrieben, z.B. $11=1+2+8$ und $29=1+4+8+16$. Ein Bit wird überprüft, indem nur die Prüfbits berechnet werden, die in seiner Erweiterung vorkommen (z.B. wird Bit 11 durch Bits 1, 2 und 8 überprüft). In diesem Beispiel werden die Prüfbits für gerade Paritätssummen für eine Nachricht berechnet, die den ASCII-Buchstaben „A“ enthält.

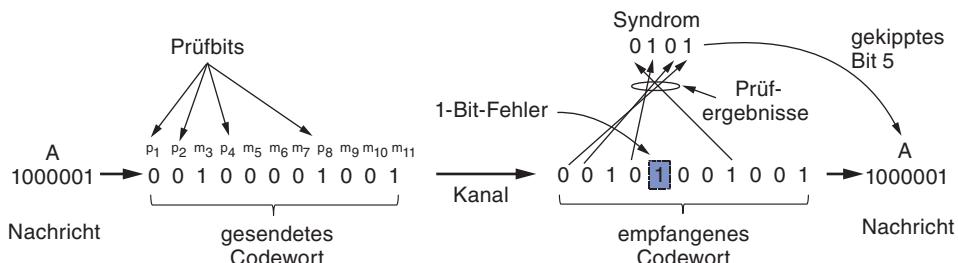


Abbildung 3.6: Beispiel eines (11,7)-Hamming-Codes zur Korrektur eines Einzelbitfehlers.

Diese Konstruktion liefert einen Code mit dem Hamming-Abstand 3, was bedeutet, dass Einzelfehler korrigiert (oder Doppelfehler entdeckt) werden können. Der Grund für die sehr sorgfältige Nummerierung von Nachrichten- und Prüfbits wird beim Decodierprozess ersichtlich. Wenn ein Codewort ankommt, führt der Empfänger die Prüfberechnung noch einmal durch, einschließlich der Werte der empfangenen Prüfbits. Wir nennen diese die Prüfergebnisse. Wenn die Prüfbits – sogar für Paritätssummen – korrekt sind, dann sollte jedes Prüfergebnis null sein. In diesem Fall wird das Codewort als gültig angenommen.

Falls die Prüfergebnisse nicht überall null sind, dann ist ein Fehler entdeckt worden. Die Menge der Prüfergebnisse bilden das sogenannte **Fehlersyndrom** (*error syndrome*), das benutzt wird, um den Fehler genau zu lokalisieren und zu korrigieren. In Abbildung 3.6 kommt auf dem Kanal ein Einzelbitfehler vor, die Prüfergebnisse sind also 0, 1, 0 bzw. 1 für $k = 8, 4, 2$ bzw. 1. Dies ergibt ein Syndrom von 0101 oder $4+1=5$. Dies bedeutet, dass fünfte Bit ist fehlerhaft. Durch Kippen des fehlerhaften Bits (welches ein Prüf- oder ein Datenbit sein kann) und Verwerfen der Prüfbits erhält man die korrekte Nachricht, ein „A“ in ASCII.

Hamming-Abstände sind für das Verständnis von Blockcodes sehr wertvoll und Hamming-Codes werden bei fehlerkorrigierenden Speichern eingesetzt. Die meisten Netze benutzen jedoch stärkere Codes. Der zweite Code, den wir uns ansehen wollen, ist ein **Faltungscode** (*convolutional code*). Dieser ist unter den Codes, die wir hier behandeln, der einzige, der kein Blockcode ist. In einem Faltungscode verarbeitet ein Codierer eine Folge von Eingabebits und erzeugt eine Folge von Ausgabebits. Es gibt keine natürliche Nachrichtengröße oder Codierungsgrenze wie bei Blockcodes. Die Ausgabe hängt von den aktuellen und den vorherigen Eingabebits ab. Das bedeutet, der Codierer hat einen Speicher. Die Anzahl der vorherigen Bits, von denen die Ausgabe abhängt, wird die **Einflusslänge** (*constraint length*) des Codes genannt. Faltungscodes sind bezüglich ihrer Rate und Einflusslänge festgelegt.

Faltungscodes werden in Netzen häufig eingesetzt, zum Beispiel als Teil des GSM-Mobilfunksystems, bei der Satellitenkommunikation und bei WiFi. Als ein Beispiel wird ein bekannter Faltungscode in ►Abbildung 3.7 gezeigt. Dieser Code ist als der NASA-Faltungscode von $r=1/2$ und $k=7$ bekannt, da er zuerst für die Voyager-Weltaummisionen eingesetzt wurde, die 1997 begannen. Seitdem wurde der Code großzügig wiederbenutzt, zum Beispiel als Teil von IEEE 802.11.

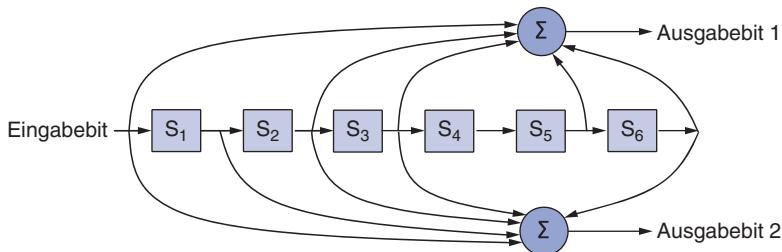


Abbildung 3.7: Der binäre Faltungscode der NASA, der in WiFi benutzt wird.

In ▶ Abbildung 3.7 erzeugt jedes Bit auf der linken Seite zwei Ausgabebits auf der rechten Seite, welche XOR-Summen der Eingabe und des internen Zustands sind. Da dieser Code mit Bits arbeitet und lineare Operationen durchführt, ist es ein binärer linearer Faltungscode. Da ein Eingabebit zwei Ausgabebits erzeugt, beträgt die Coderate 1/2. Es ist kein systematischer Code, da keines der Ausgabebits dem Eingabebit entspricht.

Der interne Zustand wird in sechs Speicherregistern festgehalten. Jedes Mal, wenn ein weiteres Bit eingegeben wird, werden die Werte in den Registern nach rechts geschoben. Wenn beispielsweise 111 die Eingabe ist und der Anfangszustand aus lauter Nullen besteht, dann wird der interne Zustand – geschrieben von links nach rechts – zu 100000, 110000 und 111000, nachdem jeweils das erste, zweite und dritte Bit eingegeben wurde. Die Ausgabebits sind dann 11, gefolgt von 10 und 01. Man benötigt sieben Verschiebungen, um eine Eingabe vollständig zu löschen, sodass sie keinen Einfluss mehr auf die Ausgabe hat. Die Einflusslänge dieses Codes ist somit $k=7$.

Ein Faltungscode wird decodiert, indem man die Folge der Eingabebits findet, die mit der größten Wahrscheinlichkeit die beobachtete Folge von Ausgabebits erzeugt hat (mit allen Fehlern). Für kleine Werte von k wird dies mit einem häufig benutzten Algorithmus durchgeführt, der von Viterbi entwickelt wurde (Forney, 1973). Der Algorithmus durchläuft die beobachtete Folge und speichert für jeden Schritt und jeden möglichen internen Zustand die Eingabefolge, die die beobachtete Folge mit den wenigsten Fehlern erzeugt hätte. Am Ende entspricht die Eingabefolge mit den wenigsten Fehlern wahrscheinlich der ursprünglichen Nachricht.

Faltungscodes waren in der Praxis sehr beliebt, da die Unsicherheit, ob ein Bit eine 0 oder eine 1 ist, einfach in die Decodierung einbezogen werden kann. Wenn beispielsweise $-1V$ die logische 0-Ebene und $+1V$ die logische 1-Ebene darstellt, könnten wir für zwei Bits $0,9V$ und $-0,1V$ empfangen. Anstatt diese Signale direkt auf 1 und 0 abzubilden, möchten wir $0,9V$ gerne als „sehr wahrscheinlich eine 1“ und $-0,1V$ als „vielleicht eine 0“ ansehen und die Folge als Ganzes korrigieren. Erweiterungen des Viterbi-Algorithmus können mit diesen Unsicherheiten arbeiten und somit eine stärkere Fehlerbehebung bieten. Dieser Ansatz, mit der Unsicherheit eines Bits umzugehen, wird **Soft-Decision-Decodierung** genannt. Umgekehrt wird die Methode, vor der nachfolgenden Fehlerbehebung für jedes Bit zu entscheiden, ob es sich um eine 0 oder eine 1 handelt, **Hard-Decision-Decodierung** genannt.

Die dritte Art von Fehlerrichturcode, den wir beschreiben wollen, ist der **Reed-Solomon-Code**. Wie Hamming-Codes sind Reed-Solomon-Codes lineare Blockcodes und oft sind sie auch systematisch. Anders als Hamming-Codes, die auf einzelnen Bits arbeiten, operieren Reed-Solomon-Codes auf m Bitsymbolen. Naturgemäß ist die Mathematik hier mehr beteiligt, daher werden wir die Operationen dieser Codes sinngemäß beschreiben.

Reed-Solomon-Codes basieren auf der Tatsache, dass jedes n -Grad-Polynom allein durch $n+1$ Punkte bestimmt wird. Zum Beispiel wird eine Gerade der Form $ax+b$ durch zwei Punkte festgelegt. Zusätzliche Punkte auf der gleichen Geraden sind redundant, was hilfreich für die Fehlerrichtur ist. Angenommen, wir haben zwei Datenpunkte, die eine Gerade repräsentieren, und wir senden diese zwei Punkte sowie zusätzlich zwei Prüfpunkte, die so gewählt sind, dass sie auf denselben Geraden liegen. Wenn nun einer der Punkte fehlerhaft empfangen wird, so können wir dennoch die Datenpunkte wiederherstellen, indem wir eine Gerade durch die empfangenen Punkte ziehen. Drei dieser Punkte werden auf der Geraden liegen und ein Punkt – der fehlerhafte – nicht. Durch das Auffinden der Geraden haben wir den Fehler korrigiert.

Reed-Solomon-Codes werden tatsächlich als Polynome definiert, die über endlichen Feldern operieren, doch sie funktionieren ähnlich. Für m Bitsymbole sind die Codewörter 2^m-1 Symbole lang. Eine beliebte Wahl ist $m=8$, somit entsprechen Symbole Bytes. Ein Codewort ist dann 255 Byte lang. Der (255,233)-Code wird weitverbreitet verwendet; es werden hier 32 redundante Symbole zu 233 Datensymbolen hinzugefügt. Das Decodieren mit Fehlerrichtur wird mit einem Algorithmus durchgeführt, der von Berlekamp und Massey entwickelt wurde. Mithilfe dieses Algorithmus kann das Anpassen der Codes mittlerer Länge effizient ausgeführt werden (Massey, 1969).

Reed-Solomon-Codes werden in der Praxis wegen ihrer starken Fehlerbehebungseigenschaften häufig eingesetzt, speziell für Burstfehler. Sie werden für DSL, Datenübertragung per Kabel, Satellitenkommunikation und – vielleicht am allgegenwärtigsten – für CDs, DVDs und Blu-Ray-Discs angewandt. Da die Codes auf m Bitsymbolen basieren, werden sowohl Einzelbitfehler als auch m -Bit-Burstfehler einfach als ein Symbolfehler behandelt. Wenn $2t$ redundante Symbole hinzugefügt werden, ist ein Reed-Solomon-Code in der Lage, bis zu t Fehler in jedem der übermittelten Symbole zu korrigieren. Dies bedeutet, dass beispielsweise der (255,233)-Code, der 32 redundante Symbole besitzt, bis zu 16 Symbolfehler korrigieren kann. Da die Symbole aufeinanderfolgen und jeweils 8 Bit lang sind, kann ein Burstfehler von bis zu 128 Bits korrigiert werden. Die Situation ist sogar noch besser, wenn das Fehlermodell Auslöschen beinhaltet (z.B. ein Kratzer auf einer CD, wodurch einige Symbole unlesbar werden). In diesem Fall können bis zu $2t$ Fehler korrigiert werden.

Reed-Solomon-Codes werden häufig in Kombination mit anderen Codes eingesetzt, beispielsweise mit einem Faltungscode. Die dahinterstehende Idee ist wie folgt. Faltungscodes sind bei der Behandlung von isolierten Bitfehlern effektiv, doch sie versagen wahrscheinlich bei Burstfehlern, wenn es zu viele Fehler im empfangenen Bitstrom gibt. Durch das Einfügen eines Reed-Solomon-Codes in den Faltungscode kann der Reed-Solomon-Decodierer den Burstfehler abfangen – eine Aufgabe, für die er sehr

gut geeignet ist. Der gesamte Code bietet damit eine gute Schutzmaßnahme sowohl gegen Einzelfehler als auch gegen Burstfehler.

Den letzten Fehlerkorrekturcode, den wir behandeln werden, ist der **LDPC-Code** (*Low-Density Parity Check*). LDPC-Codes sind lineare Blockcodes, die von Robert Gallager in seiner Doktorarbeit entwickelt wurden (Gallagher, 1962). Wie die meisten Promotionen versank auch diese sofort in Vergessenheit – nur um im Jahr 1995 wiederentdeckt zu werden, als durch Fortschritte bei der Rechnerleistung diese Codes praktikabel wurden.

In einem LDPC-Code wird jede Ausgabe nur aus einem Teil der Eingabebits gebildet. Dies führte zu einer Matrixdarstellung des Codes, wobei die Matrix nur dünn mit Einsen besetzt ist, daher der Name für den Code. Das empfangene Codewort wird mit einem Approximationsalgorithmus decodiert, welcher iterativ die empfangenen Daten zu einem zulässigen Codewort verbessert. Damit werden Fehler korrigiert.

LDPC-Codes sind praktisch für große Blockgrößen und haben exzellente Fehlerkorrektureigenschaften, die viele andere Codes (einschließlich derjenigen, die wir behandelt haben) in der Praxis übertreffen. Aus diesem Grund werden sie schnell in neue Protokolle integriert. Sie sind Teil des Standards für digitales Video-Broadcasting, 10-Giga-bit-Ethernet, Trägerfrequenzanlagen und der letzten Version von IEEE 802.11. Sie dürfen erwarten, in zukünftigen Netzen mehr davon anzutreffen.

3.2.2 Fehlererkennungscodes

Fehlerkorrekturcodes werden bei drahtlosen Verbindungen häufig verwendet, da diese im Vergleich zur Glasfaser fast immer mit Störrauschen behaftet und fehleranfällig sind. Ohne Fehlerkorrekturcodes würden schwerlich Daten durchkommen. Bei Glasfaser oder hochwertigem Kupfer ist aber die Fehlerrate sehr viel niedriger, sodass die Fehlererkennung und die erneute Übertragung hier effizienter ist, um die nur gelegentlich auftretenden Fehler zu behandeln.

Wir werden drei unterschiedliche Fehlererkennungscodes untersuchen. Es sind alles lineare systematische Blockcodes:

1. Parität
2. Prüfsummen
3. Zyklische Redundanzprüfung (CRC, *Cyclic Redundancy Check*)

Um zu verstehen, warum diese effizienter als fehlerkorrigierende Codes sein können, betrachten wir den ersten Fehlererkennungscode, bei dem ein einzelnes **Paritätsbit** an die Daten angehängt wird. Das Paritätsbit wird so gewählt, dass die Anzahl der 1-Bits im Codewort gerade (oder ungerade) ist. Dieses Vorgehen ist äquivalent zum Berechnen des (geraden) Paritätsbits als die Modulo-2-Summe oder die XOR-Verknüpfung der Datenbits. Wenn beispielsweise 1011010 in gerader Parität gesendet wird, wird am Ende ein Bit angehängt, um 10110100 zu erhalten.

Bei einer ungeraden Parität wird aus 1011010 dann 10110101. Ein Code mit einem Paritätsbit hat den Abstand 2, da jeder Einzelbitfehler ein Codewort mit der falschen Parität erzeugt. Das heißt, dass es Einzelbitfehler aufdecken kann.

Wir wollen einen Kanal untersuchen, auf dem Fehler nur isoliert auftreten und dessen Fehlerquote bei 10^{-6} pro Bit liegt. Dies mag wie eine winzige Fehlerrate aussehen, doch für ein langes Anschlusskabel ist dies bestenfalls eine mittelmäßige Rate, die für die Fehlererkennung eine Herausforderung darstellt. Typische LAN-Verbindungen liefern Bitfehlerraten von 10^{-10} . Als Blockgröße nehmen wir 1 000 Bit an. Um eine Fehlererkennung für Blöcke von 1 000 Bit zu erreichen, wissen wir aus Gleichung (3.1), dass dafür 10 Prüfbits benötigt werden. Somit würde eine Datenmenge von einem Megabit 10 000 Prüfbits erfordern. Zum bloßen Auffinden eines Blocks mit einem einzigen 1-Bit-Fehler genügt ein einziges Paritätsbit pro Block. Nach jeweils 1 000 Blöcken wird ein fehlerhafter Block gefunden und ein Extrablock (1 001 Bit) muss übertragen werden, um den Fehler zu reparieren. Der zusätzliche Platzbedarf für diese Methode der Fehlererkennung plus Sendewiederholung ist nur 2 001 Bit pro Megabit, im Gegensatz zu 10 000 Bit beim Hamming-Code.

Eine Schwierigkeit bei diesem Schema ist, dass ein einzelnes Paritätsbit zuverlässig nur einen Einzelbitfehler in dem Block entdecken kann. Wenn der Block durch einen Burstfehler verstümmelt wurde, liegt die Wahrscheinlichkeit, dass der Fehler gefunden wird, bei nur 0,5. Das ist kaum akzeptabel. Eine deutliche Verbesserung wird erreicht, wenn jeder gesendete Block als rechtwinklige Matrix, n Bit breit und k Bit hoch, betrachtet wird. Wenn wir nun ein Paritätsbit für jede Reihe berechnet und gesendet haben, dann werden bis zu k Bitfehler zuverlässig erkannt, solange es höchstens einen Fehler pro Reihe gibt.

Es gibt jedoch etwas anderes, was wir zu einem verbesserten Schutz gegen Burstfehler unternehmen können: Wir können die Paritätsbits über die Daten in einer anderen Reihenfolge berechnen als in Reihenfolge, in der die Datenbits übertragen werden. Dies wird **Versatz** oder **Interleaving** genannt. In unserem Fall werden wir ein Paritätsbit für jede der n Spalten berechnen und alle Datenbits als k Reihen senden. Hierbei werden die Reihen von oben nach unten und die Bits in jeder Reihe von links nach rechts auf die übliche Art und Weise gesendet. In der letzten Reihe senden wir die n Paritätsbits. Die Übertragungsreihenfolge wird in ▶ Abbildung 3.8 für $n=7$ und $k=7$ gezeigt.

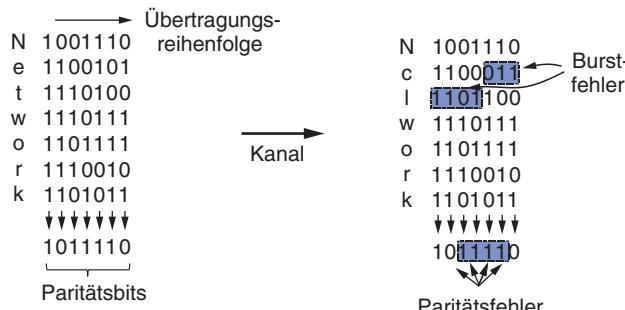


Abbildung 3.8: Interleaving von Paritätsbits zum Aufdecken eines Burstfehlers.

Interleaving ist eine allgemeine Technik, einen Code, der isolierte Fehler entdeckt (oder korrigiert) in einen Code umzuwandeln, der Burstfehler entdeckt (oder korrigiert). Wenn in Abbildung 3.8 ein Burstfehler der Länge $n=7$ auftritt, dann werden die fehlerhaften Bits über mehrere Spalten verstreut. (Ein Burstfehler impliziert nicht, dass alle Bits falsch sind; es bedeutet lediglich, dass mindestens das erste und das letzte Bit falsch sind. In Abbildung 3.8 wurden innerhalb eines Bereichs von 7 Bits 4 Bits gekippt.) In jeder der n Spalten wird höchstens 1 Bit betroffen sein, die Paritätsbits von diesen Spalten werden daher den Fehler entdecken. Diese Methode benutzt n Paritätsbits auf Blöcken von kn Datenbits, um einen einzelnen Burstfehler der Länge n oder kleiner zu entdecken.

Ein Burst der Länge $n+1$ wird dann unentdeckt durchgehen, wenn das erste und das letzte Bit invertiert und alle anderen Bits korrekt sind. Wird ein Block durch einen langen oder viele kurze Bursts verstümmelt, liegt für jede einzelne der n Spalten die Wahrscheinlichkeit, dass diese die richtige Parität hat, bei 0,5. Folglich ist die Wahrscheinlichkeit, dass ein fehlerhafter Block dennoch angenommen wird, 2^{-n} .

Die zweite Art von Fehlererkennungscode, die **Prüfsumme** (*checksum*), ist eng mit Gruppen von Paritätsbits verwandt. Das Wort „Prüfsumme“ wird häufig benutzt, um eine Gruppe von Prüfbits zu bezeichnen, die mit einer Nachricht verbunden sind, unabhängig vom Berechnungsverfahren. Eine Gruppe von Paritätsbits ist ein Beispiel für eine Prüfsumme. Es gibt jedoch andere, stärkere Prüfsummen, die auf einer laufenden Summe von Datenbits der Nachricht basieren. Die Prüfsumme wird normalerweise am Ende der Nachricht als Komplement der Summenfunktion eingefügt. Dann können Fehler entdeckt werden, indem das gesamte empfangene Codewort aufsummiert wird, sowohl Datenbits als auch die Prüfsumme. Wenn das Ergebnis null ist, dann ist kein Fehler entdeckt worden.

Ein Beispiel einer Prüfsumme ist die 16-Bit-Internetprüfsumme, die für alle Internetpakete als Teil des IP-Protokolls benutzt wird (Braden et al., 1988). Diese Prüfsumme addiert die Nachrichtenbits auf, die in 16-Bit-Wörter aufgeteilt sind. Weil diese Methode auf Wörtern statt wie bei Parität auf Bits operiert, können Fehler, die die Parität unverändert lassen würden, dennoch die Summe verändern und somit entdeckt werden. Wenn beispielsweise das niederwertigste Bit in zwei verschiedenen Wörtern von 0 auf 1 gekippt ist, dann würde eine Paritätsprüfung über diese Bits den Fehler nicht entdecken und somit versagen. Wenn allerdings zwei Einsen zur 16-Bit-Prüfsumme hinzuaddiert werden, so wird ein unterschiedliches Ergebnis erzielt. Der Fehler kann also entdeckt werden.

Die Internetprüfsumme wird in Einerkomplement-Arithmetik anstatt als Modulo- 2^{16} -Summe berechnet. In Einerkomplement-Arithmetik ist eine negative Zahl das bitweise Komplement seines positiven Gegenparts. Moderne Rechner verwenden Zweierkomplement-Arithmetik, bei der eine negative Zahl das Einerkomplement plus eins ist. Auf einem Zweierkomplement-Rechner ist die Einerkomplement-Summe äquivalent damit, die Modulo- 2^{16} -Summe zu nehmen und den Überlauf des höchswertigen Bits zu den niederwertigen Bits zu addieren. Dieser Algorithmus liefert eine eher gleichmäßige Abdeckung der Daten durch die Prüfsummenbits. Sonst können zwei

höherwertige Bits hinzugefügt werden, überlaufen und verloren gehen, ohne dass sich die Summe ändert. Es gibt noch einen weiteren Vorteil: In der Einerkomplement-Arithmetik gibt es zwei Darstellungen der Null – alles Nullen und alles Einsen. Damit kann einer der Werte (z.B. alles Nullen) benutzt werden um anzulegen, dass es keine Prüfsumme gibt, ohne dass dafür ein weiteres Feld nötig wäre.

Jahrzehntelang wurde immer vorausgesetzt, dass Rahmen, die per Prüfsumme getestet werden, zufällige Bits enthalten. Alle Analysen von Prüfsummenalgorithmen sind unter dieser Voraussetzung durchgeführt worden. Die Untersuchung von realen Daten durch Partridge et al. (1995) hat gezeigt, dass diese Annahme schlichtweg falsch ist. Das heißt, in einigen Fällen kommen unentdeckte Fehler viel häufiger vor, als man ursprünglich dachte.

Speziell die Internetprüfsumme ist effizient und einfach, bietet jedoch in einigen Fällen nur schwachen Schutz, gerade weil es eine einfache Summe ist. Weder die Löschung oder das Hinzufügen von Nulldaten noch der Austausch von Teilen der Nachricht werden entdeckt. Außerdem bietet das Verfahren nur schwachen Schutz gegen das Verbinden von Nachrichten, bei dem Teile von zwei Nachrichten aneinandergehängt werden. Es mag sehr unwahrscheinlich erscheinen, dass diese Fehler bei beliebigen Prozessen auftreten, doch es ist genau diese Sorte von Fehlern, die sich bei beschädigter Hardware einstellen können.

Eine bessere Wahl ist die **Fletcher-Prüfsumme** (Fletcher, 1982). Sie enthält eine Lagekomponente, die das Produkt der Daten und seine Position zur laufenden Summe hinzufügt. Damit können Veränderungen in der Position der Daten besser entdeckt werden.

Obwohl die beiden vorangehenden Verfahren in höheren Schichten manchmal angemessen sein können, wird in der Praxis häufig eine dritte und stärkere Art von fehlererkennendem Code in der Sicherungsschicht benutzt: die **zyklische Redundanzprüfung (CRC, Cyclic Redundancy Check)**, auch **Polynomcode** genannt. Polynomcodes basieren darauf, dass man Bitketten als Darstellungen von Polynomen mit den Koeffizienten 0 und 1 behandelt. Ein Rahmen mit k Bits wird als Koeffizientenliste für ein Polynom mit k Termen von x^{k-1} bis x^0 betrachtet. Dieses Polynom hat den Grad $k-1$. Das höchstwertige Bit (das am weitesten links stehende) ist der Koeffizient von x^{k-1} , das nächste Bit ist der Koeffizient von x^{k-2} und so weiter. So besteht beispielsweise 110001 aus sechs Bit und stellt daher ein sechsgliedriges Polynom mit den Koeffizienten 1, 1, 0, 0, 0 und 1 dar: $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x + 1x^0$.

Rechnungen mit Polynomen modulo 2 werden nach den Regeln der algebraischen Körpertheorie durchgeführt. Es gibt keinen Übertrag bei der Addition und kein Borgen bei der Subtraktion. Addition und Subtraktion sind identisch mit dem exklusiven ODER (XOR).

Ein Beispiel:

$$\begin{array}{r}
 10011011 & 00110011 & 11110000 & 01010101 \\
 + \underline{11001010} & + \underline{11001101} & - \underline{10100110} & - \underline{10101111} \\
 01010001 & 11111110 & 01010110 & 11111010
 \end{array}$$

Die lange Division wird auf exakt die gleiche Weise ausgeführt wie im Binärsystem, außer dass dabei die Subtraktion wieder modulo 2 ausgeführt wird. Ein Divisor „passt“ in einen Dividenden, wenn der Dividend genauso viele Bits hat wie der Divisor.

Wird die Polynomcode-Methode angewandt, dann müssen sich Sender und Empfänger vorher auf ein sogenanntes **Generatorpolynom** einigen. Sowohl das höchstwertige als auch die niederwertigste Bit müssen 1 sein. Um den CRC-Wert für einen Rahmen mit m Bit, der einem Polynom $M(x)$ entspricht, ausrechnen zu können, muss der Rahmen länger als das Generatorpolynom sein. Die Grundidee ist, den CRC-Wert am Ende des Rahmens anzuhängen, und zwar so, dass das Polynom, das durch um eine Prüfsumme ergänzten Rahmen dargestellt wird, durch $G(x)$ teilbar ist. Erhält der Empfänger den Rahmen, teilt er ihn durch $G(x)$. Bleibt ein Rest, ist ein Übertragungsfehler vorhanden.

Der Algorithmus zur Berechnung des CRC-Werts lautet folgendermaßen:

1. Sei r der Grad von $G(x)$. Hänge r 0-Bits an der niederwertigen Seite des Rahmens an, sodass er nun $m+r$ Bits enthält und dem Polynom $x^rM(x)$ entspricht.
2. Teile die Bitkette, die $x^rM(x)$ entspricht, anhand der Modulo-2-Division durch die Bitkette, die $G(x)$ entspricht.
3. Subtrahiere den Rest (immer r Bits oder weniger) mit der Modulo-2-Subtraktion von der Bitkette, die $x^rM(x)$ entspricht. Das Ergebnis ist der um die Prüfsumme ergänzte Rahmen, der übertragen wird. Das zugehörige Polynom heiße $T(x)$.

► Abbildung 3.9 veranschaulicht die Berechnung für den Rahmen 1101011011 unter Verwendung des Generatorpolynoms $G(x)=x^4+x+1$.

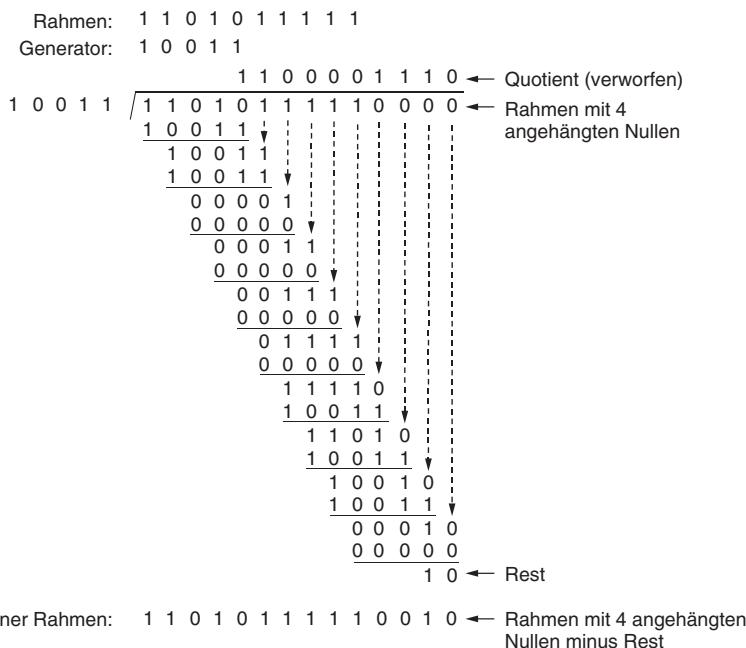


Abbildung 3.9: Beispielberechnung des CRC-Werts.

Es sollte klar sein, dass $T(x)$ durch $G(x)$ teilbar ist (modulo 2). Bei jeder Division ist das, was übrig bleibt, wenn man vom Dividenden den Rest abzieht, durch den Divisor teilbar. Wenn Sie beispielsweise bei der Basis 10 210 278 durch 10 941 teilen, so beträgt der Rest 2 399. Wenn Sie 2 399 von 210 278 abziehen, dann ist der Rest (207 879) teilbar durch 10 941.

Untersuchen wir nun einmal, was diese Methode leistet. Welche Arten von Fehlern werden erkannt? Stellen Sie sich einen Übertragungsfehler vor, bei dem anstatt des Polynoms $T(x)$ das Polynom $T(x)+E(x)$ ankommt. Jedes 1-Bit in $E(x)$ entspricht einem Bit, das bei der Übertragung invertiert wurde. Wenn $E(x)$ k 1-Bits enthält, haben sich k Einzelbitfehler ereignet. Ein einzelner Burstfehler sieht wie folgt aus: eine Eins am Anfang, eine Mischung von Nullen und Einsen in der Mitte und eine Eins am Schluss. Alle anderen Bits sind Nullen.

Wenn der Empfänger den Rahmen mit der angehängten Prüfsumme empfängt, teilt er ihn durch $G(x)$. Er berechnet also $[T(x)+E(x)]/G(x)$. $T(x)/G(x)$ ist immer 0. Folglich ist das Ergebnis der Berechnung einfach $E(x)/G(x)$. Die Fehler, die zufällig Polynomen mit $G(x)$ als Faktor (d.h. Vielfachen von $G(x)$) entsprechen, werden nicht entdeckt, alle anderen aber schon.

Bei einem Einzelbitfehler ist $E(x)=x^i$, wobei i angibt, welches Bit fehlerhaft ist. Wenn $G(x)$ zwei oder mehr Terme enthält, kann $E(x)$ nie durch $G(x)$ geteilt werden. Folglich können alle Einzelbitfehler erkannt werden.

Treten zwei isolierte Einzelbitfehler auf, ergibt sich $E(x)=x^i+x^j$, wobei $i>j$ ist. Diese Formel kann auch folgendermaßen geschrieben werden: $E(x)=x^i(x^{i-j}+1)$. Nimmt man an, dass $G(x)$ nicht durch x teilbar ist, reicht zur Entdeckung aller Doppelfehler aus, dass $G(x)$ nicht Teiler von x^{k+1} ist, und zwar für alle k bis zu einem Maximalwert von $i-j$ (d.h. bis zur maximalen Rahmengröße). Es sind einfache Polynome niedrigen Grades bekannt, die lange Rahmen schützen. Zum Beispiel ist $x^{15}+x^{14}+1$ nicht Teiler von x^{k+1} für alle k kleiner 32 768.

Ist die Zahl der fehlerhaften Bits ungerade, enthält $E(x)$ eine ungerade Anzahl von Termen (z.B. x^5+x^2+1 , aber nicht x^2+1). Interessant ist, dass es kein Polynom mit einer ungeraden Zahl von Termen gibt, das $x+1$ als Faktor im Modulo-2-System hat. Durch die Verwendung von $x+1$ als Faktor von $G(x)$ werden alle Fehler mit einer ungeraden Zahl von invertierten Bits gefunden.

Schließlich, und auch am wichtigsten, ist jedoch, dass man mit einem Polynomcode mit r Prüfbits alle Burstfehler der Länge $\leq r$ erkennt. Ein Burstfehler der Länge k kann als $x^i(x^{k-1}+\dots+1)$ dargestellt werden, wobei i angibt, wie weit der Burstfehler vom rechten Ende des empfangenen Rahmens entfernt ist. Wenn $G(x)$ einen x^0 -Term enthält, hat es keinen x^i -Term als Faktor. Wenn also der Grad des Ausdrucks in Klammern kleiner ist als der Grad von $G(x)$, kann der Rest nie null sein.

Ist die Länge des Burstfehlers $r+1$, so ist der nach der Division durch $G(x)$ verbleibende Rest ausschließlich dann null, wenn der Burstfehler mit $G(x)$ identisch ist. Bei einem Burstfehler ist das erste und letzte Bit per Definition 1. Folglich hängt die Über-

einstimmung von den $r-1$ dazwischenliegenden Bits ab. Sind alle Kombinationen gleich wahrscheinlich, liegt die Wahrscheinlichkeit, dass ein nicht korrekter Rahmen akzeptiert wird, bei $1/2^{r-1}$.

Es kann auch gezeigt werden, dass bei einem Burstfehler, der länger als $r+1$ Bit ist, oder bei mehreren kleineren Bursts die Wahrscheinlichkeit des Nicherkennens eines fehlerhaften Rahmens gleich $1/2^r$ ist, unter der Annahme, dass alle Bitmuster gleich wahrscheinlich sind.

Bestimmte Polynome wurden zu internationalen Standards erhoben. Das Polynom, das im Standard IEEE 802 verwendet wird und dem Beispiel von Ethernet folgt, lautet:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

Neben anderen wünschenswerten Eigenschaften zeichnet sich dieses Polynom dadurch aus, dass es alle Bursts bis zur Länge 32 entdeckt sowie alle Bursts, die eine ungerade Anzahl von Bits betreffen. Es wurde seit den 1980er Jahren weitverbreitet eingesetzt. Dies bedeutet jedoch nicht, dass es die beste Wahl ist. Unter Anwendung einer erschöpfenden, rechenintensiven Suche haben Castagnoli et al. (1993) und Koopman (2002) die besten CRC-Codes gefunden. Diese CRCs haben einen Hamming-Abstand von 6 für typische Nachrichtengrößen, während der IEEE-Standard CRC-32 einen Hamming-Abstand von nur 4 hat.

Obwohl die Berechnung des CRC-Werts kompliziert aussieht, ist es auf Hardware-ebene mit einfachen Schieberegisterschaltungen leicht zu berechnen und zu verifizieren (Peterson und Brown, 1961). In der Praxis wird diese Hardware fast immer eingesetzt. Dutzende von Netzstandards enthalten verschiedene CRCs, unter anderem fast alle LANs (z.B. Ethernet) sowie Punkt-zu-Punkt-Leitungen (z.B. Packet over SONET).

3.3 Grundlegende Protokolle der Sicherungsschicht

Als Einführung in dieses Thema sehen wir uns drei Protokolle mit zunehmender Komplexität an. Für interessierte Leser sind diese und die noch folgenden Protokolle im Web ein Simulator verfügbar (siehe Vorwort). Vorher ist es jedoch sinnvoll, einige dem Kommunikationsmodell zugrunde liegende Annahmen ausführlich darzustellen.

Erstens nehmen wir an, dass die Bitübertragungs-, die Sicherungs- und die Vermittlungsschicht unabhängige Prozesse sind, die miteinander kommunizieren, indem sie Nachrichten austauschen. Eine weitverbreitete Implementierung ist in ► Abbildung 3.10 zu sehen. Die Prozesse der Bitübertragungsschicht und einige Prozesse der Sicherungsschicht laufen auf spezieller Hardware, der **Netzwerkkarte (NIC, Network Interface Card)**. Der Rest der Sicherungsschichtprozesse und die Prozesse der Vermittlungsschicht laufen meist auf der Haupt-CPU als Teil des Betriebssystems, wobei die Software für die Sicherungsschichtprozesse häufig in Form von Gerätetreibern auftritt. Andere Implementierungen sind jedoch ebenfalls möglich (z.B. drei Prozesse einer dafür vorgesehenen Hardware – dem **Netzbeschleuniger (network accelerator)** – aufladen oder drei Prozesse auf der Haupt-CPU mit einem Verhältnis, das softwareseitig

definiert ist, ausführen). Auf alle Fälle vereinfacht die Behandlung der drei Schichten als separate Prozesse die Diskussion des Konzeptes grundsätzlich und die Unabhängigkeit der einzelnen Schichten wird unterstrichen.

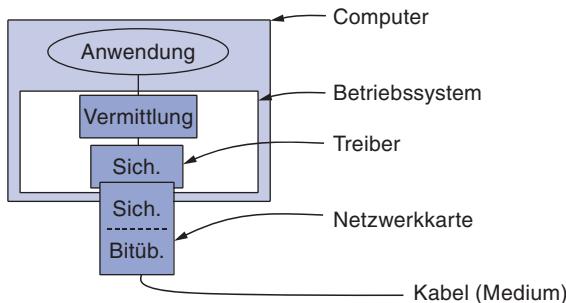


Abbildung 3.10: Implementierung der Bitübertragungs-, Sicherungs- und Vermittlungsschichten.

Eine weitere wichtige Annahme ist, dass Rechner A eine lange Datenfolge an Rechner B senden und dabei einen zuverlässigen verbindungsorientierten Dienst verwenden will. Später wird der Fall behandelt, in dem auch B gleichzeitig Daten an A senden will. Wir gehen davon aus, dass A einen unerschöpflichen Vorrat an sendebereiten Daten hat und nie auf die Erstellung der Daten warten muss. Fordert die Sicherungsschicht von A Daten an, so kann die Vermittlungsschicht sofort welche zur Verfügung stellen (auch diese Annahme werden wir später fallen lassen).

Wir gehen weiter davon aus, dass die Rechner nicht abstürzen. Diese Protokolle behandeln also Kommunikationsfehler, nicht aber die Probleme, die beim Absturz und erneutem Hochfahren eines Rechners entstehen.

Für die Sicherungsschicht besteht das Paket, das über die Schnittstelle von der Vermittlungsschicht kommt, nur aus Daten, von denen jedes einzelne Bit an die Vermittlungsschicht des Zielrechners gesendet werden muss. Die Tatsache, dass die Vermittlungsschicht am Zielort einen Teil des Pakets als Nachrichten-Header interpretiert, ist für die Sicherungsschicht nicht von Belang.

Die Sicherungsschicht nimmt ein Paket an und kapselt es in einem Rahmen, indem ein spezieller Nachrichten-Header und ein Trailer hinzugefügt werden (siehe Abbildung 3.1). Folglich besteht ein Rahmen aus einem eingeschlossenen Paket und einigen Steuerdaten (im Nachrichten-Header) sowie einer Prüfsumme (im Trailer). Der Rahmen wird dann zur Sicherungsschicht des anderen Rechners übertragen. Nehmen wir an, es gibt die Bibliotheksprozeduren `to_physical_layer`, um Rahmen zu senden, und `from_physical_layer`, um Rahmen zu empfangen. Diese Prozeduren berechnen die Prüfsumme und hängen sie an den Code an oder überprüfen diese (was in der Regel hardwaremäßig durchgeführt wird), sodass wir uns im Zusammenhang mit den in diesem Abschnitt zu entwickelnden Protokollen nicht darum kümmern müssen. Es könnte beispielsweise der CRC-Algorithmus verwendet werden, den wir im vorigen Abschnitt besprochen haben.

Anfangs hat der Empfänger nichts zu tun. Er wartet nur darauf, dass etwas geschieht. In den Protokollbeispielen in diesem Kapitel stellen wir die Tatsache, dass die Sicherungsschicht darauf wartet, dass etwas passiert, durch den Prozedurauftrag *wait_for_event(&event)* dar. Diese Prozedur kehrt nur dann zurück, wenn etwas passiert ist (z.B. wenn ein Rahmen angekommen ist). Bei der Rückkehr gibt die Variable *event* an, was genau geschehen ist. Die Menge der möglichen Ereignisse ist bei verschiedenen Protokollarten jeweils unterschiedlich und wird für jedes Protokoll gesondert festgelegt. In der Realität sieht das anders aus: Die Sicherungsschicht wartet nicht tatenlos auf ein Ereignis (wie wir jetzt hier annehmen), sondern erhält ein Interrupt, welches sie ihre momentane Arbeit unterbrechen lässt, um den eingegangenen Rahmen zu verarbeiten. Der Einfachheit halber werden wir trotzdem die parallelen Aktivitäten der Sicherungsschicht nicht weiter berücksichtigen, sondern davon ausgehen, dass sie ihre gesamte Bearbeitungszeit einem einzigen Kanal widmet.

Kommt ein Rahmen beim Empfänger an, dann wird die Prüfsumme neu berechnet. Falls die Prüfsumme nicht korrekt ist (d.h. ein Übertragungsfehler aufgetreten ist), so wird die Sicherungsschicht darüber informiert (*event = cksum_err*). Kommt der Rahmen korrekt an, wird die Sicherungsschicht ebenfalls informiert (*event = frame_arrival*), sodass sie den Rahmen mit *from_physical_layer* zur Überprüfung annehmen kann. Hat die Sicherungsschicht der Empfängerseite einen fehlerfreien Rahmen erhalten, dann prüft sie die Informationen im Nachrichten-Header. Ist alles richtig, dann wird der Paketeil an die Vermittlungsschicht übertragen. Der Rahmen-Header wird unter keinen Umständen an die Vermittlungsschicht weitergegeben.

Es gibt einen guten Grund, dass die Vermittlungsschicht nie auch nur einen Teil des Rahmen-Headers erhalten darf: die Protokolle der Vermittlungs- und der Sicherungsschicht absolut getrennt zu halten. Solange die Vermittlungsschicht nichts über die Protokolle oder das Rahmenformat der Sicherungsschicht weiß, können Änderungen in der Sicherungsschicht vorgenommen werden, ohne dass die Software der Vermittlungsschicht ebenfalls modifiziert werden muss. Dies passiert jedes Mal, wenn eine neue Netzwerkkarte in einem Rechner eingebaut wird. Durch eine vorgegebene Schnittstelle zwischen Vermittlungs- und Sicherungsschicht wird der Entwurf wesentlich vereinfacht, weil die Kommunikationsprotokolle in den verschiedenen Schichten unabhängig voneinander entwickelt werden können.

► Abbildung 3.11 zeigt einige Deklarationen (in C), die in vielen Protokollen verwendet werden, die später beschrieben werden. Fünf Datenstrukturen sind hier definiert: *boolean*, *seq_nr*, *packet*, *frame_kind* und *frame*. Ein *boolean* ist ein Typ mit zwei Werten *true* (wahr) oder *false* (falsch). Bei *seq_nr* handelt es sich um eine kleine Ganzzahl zur Nummerierung der Rahmen, damit wir sie unterscheiden können. Die Nummerierung geht von 0 bis einschließlich *MAX_SEQ*, das in jedem Protokoll definiert ist, das diesen Maximalwert braucht. Ein *packet* (Paket) ist die Informationseinheit, die zwischen der Vermittlungs- und der Sicherungsschicht eines Rechners oder zwischen den Vermittlungsschichten gleichgestellter Rechner ausgetauscht wird. In unserem Modell enthält es immer *MAX_PKT* Byte. Wirklichkeitsnah wäre aber ein Paket mit variabler Länge.

```

#define MAX_PKT 1024           /* bestimmt die Paketgröße in Byte */
typedef enum {false, true} boolean; /* boolescher Typ */
typedef unsigned int seq_nr;    /* Sequenz- oder Bestätigungsnummer */
typedef struct {unsigned char data[MAX_PKT];} packet; /* Paketdefinition */
typedef enum {data, ack, nak} frame_kind; /* Definition der Rahmenart */
typedef struct {                         /* Rahmen, die auf dieser Schicht
    frame_kind kind;                   /* übertragen werden */
    seq_nr seq;                       /* welche Art von Rahmen? */
    seq_nr ack;                      /* Sequenznummer */
    packet info;                     /* Bestätigungsnummer */
} frame;                                /* Paket der Vermittlungsschicht */

/* Auf Ereignis warten; Typ in event zurückgeben */
void wait_for_event(event_type *event);

/* Paket von der Vermittlungsschicht zur Übertragung holen */
void from_network_layer(packet *p);

/* Informationen eines eingegangenen Rahmens zur Vermittlungsschicht senden */
void to_network_layer(packet *p);

/* Eingegangenen Rahmen von der Bitübertragungsschicht holen und auf
   r kopieren */
void from_physical_layer(frame *r);

/* Rahmen an Bitübertragungsschicht zur Übertragung weitergeben */
void to_physical_layer(frame *s);

/* Uhr starten und Timeout-Ereignis aktivieren */
void start_timer(seq_nr k);

/* Uhr anhalten und Timeout-Ereignis deaktivieren */
void stop_timer(seq_nr k);

/* Start eines Hilfstimers und Ereignis ack_timeout aktivieren */
void start_ack_timer(void);

/* Hilfstimer anhalten und Ereignis ack_timeout deaktivieren */
void stop_ack_timer(void);

/* Der Vermittlungsschicht erlauben, Ereignis network_layer_ready auszulösen */
void enable_network_layer(void);

/* Der Vermittlungsschicht verbieten, Ereignis network_layer_ready auszulösen */
void disable_network_layer(void);

/* Makro inc wird in der Zeile erweitert: k wird zyklisch inkrementiert */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0

```

Abbildung 3.11: Definitionen, die in nachfolgenden Protokollen benötigt werden.
 Diese Definitionen befinden sich in der Datei *protocol.h*.

Ein *frame* setzt sich aus vier Feldern zusammen: *kind*, *seq*, *ack* und *info*. Die ersten drei enthalten Steuerinformationen und das letzte kann zu übertragende Daten enthalten. Die Steuerfelder werden zusammen **Rahmen-Header** genannt.

Das *kind*-Feld gibt an, ob der Rahmen Daten enthält, weil Rahmen von einigen Protokollen danach unterschieden werden, ob sie nur Steuerinformationen oder auch Daten enthalten. Die Felder *seq* und *ack* werden für Sequenznummern bzw. Bestätigungsnummern verwendet. Ihre Verwendung wird später ausführlich beschrieben. Das *info*-Feld eines Datenrahmens enthält ein einzelnes Paket. Das *info*-Feld eines Steuerrahmens wird nicht verwendet. Eine realistischere Implementierung hätte ein *info*-Feld mit variabler Länge, das bei Steuerrahmen ganz weggelassen würde.

Wichtig ist auch hier wieder, die Beziehung zwischen einem Paket und einem Rahmen zu verstehen. Die Vermittlungsschicht erstellt ein Paket, indem eine Nachricht von der Transportschicht übernommen und dieser der Vermittlungsschicht-Header angehängt wird. Dieses Paket wird an die Sicherungsschicht übergeben, die es wiederum in das *info*-Feld eines ausgehenden Rahmens einbaut. Wenn ein Rahmen sein Ziel erreicht, extrahiert die Sicherungsschicht das Paket aus dem Rahmen und über gibt es an die Vermittlungsschicht. Auf diese Weise kann die Vermittlungsschicht so arbeiten, als ob die Rechner die Pakete direkt austauschen würden.

In Abbildung 3.11 sind auch eine Reihe von Prozeduren aufgelistet. Dies sind Bibliotheks routinen, deren Einzelheiten von der Implementierung abhängen und deren innere Abläufe uns in der folgenden Diskussion nicht weiter interessieren. Die Prozedur *wait_for_event* durchläuft eine Schleife und wartet, bis etwas passiert. Die Prozeduren *to_network_layer* und *from_network_layer* werden von der Sicherungsschicht verwendet, um Pakete an die Vermittlungsschicht weiterzugeben und von ihr Pakete entgegenzunehmen. Beachten Sie, dass *from_physical_layer* und *to_physical_layer* Rahmen zwischen der Sicherungsschicht und der Bitübertragungsschicht übertragen. Anders ausgedrückt beziehen sich *to_network_layer* und *from_network_layer* auf die Schnittstelle zwischen den Schichten 2 und 3, während *from_physical_layer* und *to_physical_layer* die Schnittstelle zwischen den Schichten 1 und 2 betreffen.

Bei den meisten Protokollen gehen wir von einem unzuverlässigen Kanal aus, der gelegentlich ganze Rahmen verliert. Aus diesem Grund muss die sendende Sicherungsschicht einen internen Timer oder eine Uhr starten, wenn sie einen Rahmen sendet. Geht innerhalb einer bestimmten vordefinierten Zeit keine Bestätigung ein, läuft die Uhr ab und die Sicherungsschicht erhält ein Interrupt-Signal.

Bei unseren Protokollen wird das folgendermaßen gehandhabt: Die Prozedur *wait_for_event* gibt *event = timeout* zurück. Die Prozeduren *start_timer* und *stop_timer* schalten den Timer an und aus. Timeout-Ereignisse können nur eintreten, wenn der Timer läuft und bevor *stop_timer* aufgerufen wird. Der Aufruf von *start_timer* ist ausdrücklich auch zulässig, während der Timer läuft. Durch einen solchen Aufruf wird die Uhr zurückgesetzt, um die Beziehung zwischen einem Paket und einem Rahmen zu verstehen (sofern der Timer nicht zurückgesetzt oder ausgeschaltet wird).

Die Prozeduren *start_ack_timer* und *stop_ack_timer* werden zur Steuerung eines Hilfstimers benutzt, um unter bestimmten Bedingungen Bestätigungen zu erzeugen.

Die Prozeduren *enable_network_layer* und *disable_network_layer* werden in anspruchsvoller Protokollen benutzt, bei denen wir nicht mehr davon ausgehen, dass die Vermittlungsschicht immer Pakete zum Versenden hat. Wird die Vermittlungsschicht von der Sicherungsschicht aktiviert, kann die Vermittlungsschicht unterbrechen, wenn sie ein Paket zu versenden hat. Das wird im Beispiel durch die Prozedur *event = network_layer_ready* angegeben. Ist eine Vermittlungsschicht deaktiviert, darf sie solche Ereignisse nicht erzeugen. Durch sorgfältige Vorgehensweise, wann sie ihre Vermittlungsschicht aktiviert und deaktiviert, kann die Sicherungsschicht verhindern, dass die Vermittlungsschicht sie mit Paketen überschwemmt, wenn nicht mehr ausreichend Pufferplatz zur Verfügung steht.

Die Sequenznummern der Rahmen liegen immer im Bereich von 0 bis einschließlich *MAX_SEQ*, wobei *MAX_SEQ* je nach Protokoll unterschiedlich ist. Häufig muss die Sequenznummer modulo *MAX_SEQ* um 1 erhöht werden (d.h., nach *MAX_SEQ* kommt 0). Diese schrittweise Erhöhung wird vom Makro *inc* ausgeführt. Es wurde als Makro definiert, weil es innerhalb des kritischen Pfades verwendet wird. Wie wir später noch sehen werden, ist die Protokollverarbeitung häufig der entscheidende Faktor, der die Netzwerkleistung limitiert. Durch Definition von einfachen Operationen wie dieser als Makros wird die Leistung gesteigert, ohne die Lesbarkeit des Codes zu beeinträchtigen.

Die Deklarationen in Abbildung 3.11 sind Teil aller Protokolle, die wir kurz besprechen werden. Um Platz zu sparen und eine bequeme Nachschlagemöglichkeit bereitzustellen, wurden sie extrahiert und in einer Liste zusammengestellt, aber vom Konzept her sollten sie bei den einzelnen Protokollen stehen. In C erfolgt die Zusammenführung durch Einfügen der Definitionen in eine gesonderte Header-Datei, in diesem Fall *protocol.h*, und durch Verwenden der `#include`-Anweisung des C-Präprozessors, um sie in die Protokolldateien einzubeziehen.

3.3.1 Ein utopisches Simplexprotokoll

Als erstes Beispiel wollen wir ein Protokoll betrachten, das einfacher nicht sein könnte, weil es sich nicht um die Möglichkeit kümmert, dass irgendetwas schiefgehen könnte. Die Daten werden nur in eine Richtung übertragen. Sowohl die sendende als auch die empfangende Vermittlungsschicht ist jederzeit bereit. Eventuelle Verarbeitungszeiten werden ignoriert. Der Puffer ist unendlich groß. Als absolute Krönung verliert der Übertragungskanal zwischen den Sicherungsschichten nie Rahmen und beschädigt sie auch nicht. Dieses völlig unrealistische Protokoll, das wir „Utopia“ nennen, ist nur dazu da, die Grundstruktur darzustellen, auf der wir aufbauen werden. Die Implementierung ist in ►Abbildung 3.12 zu sehen.

```

/* Protokoll 1 (Utopia) lässt die Datenübertragung nur in einer Richtung zu:
vom Sender zum Empfänger. Laut Voraussetzung ist der Kommunikationskanal
fehlerfrei und der Empfänger kann alle eingehenden Daten unendlich schnell
verarbeiten. Deshalb befindet sich der Sender einfach in einer Schleife
und füttert die Leitung mit Daten, so schnell er kann. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void senderl(void)
{
    frame s;                                /* Puffer für ausgehende Rahmen */
    packet buffer;                          /* Puffer für ausgehendes Paket */
    while (true) {
        from_network_layer(&buffer);       /* hole etwas zum Senden */
        s.info = buffer;                   /* kopiere es in s zum Übertragen */
        to_physical_layer(&s);           /* schicke es ab */
    }                                         /* Morgen, und Morgen, und dann wieder
                                                morgen, Kriecht so mit kleinem
                                                Schritt von Tag zu Tag, Zur letzten
                                                Silb? auf unserem Lebensblatt
                                                - Macbeth, V, v */
}

void receiverl(void)
{
    frame r;
    event_type event;                      /* wird von wait ausgefüllt,
                                                aber hier nicht verwendet */

    while (true) {
        wait_for_event(&event);          /* einzige Möglichkeit ist
                                                frame_arrival */
        from_physical_layer(&r);        /* hole ankommenden Rahmen */
        to_network_layer(&r.info);      /* gib Daten an Vermittlungsschicht
                                                weiter */
    }
}

```

Abbildung 3.12: Utopisches Simplexprotokoll.

Das Protokoll besteht aus zwei unterschiedlichen Prozeduren, einem Sender und einem Empfänger. Der Sender läuft auf der Sicherungsschicht der Quellmaschine, der Empfänger auf der Sicherungsschicht der Zielmaschine. Es werden keine Sequenznummern oder Bestätigungen verwendet, deshalb wird *MAX_SEQ* nicht benötigt. Die einzige mögliche Ereignisart ist *frame_arrival* (d.h. die Ankunft eines fehlerfreien Rahmens).

Der Sender befindet sich in einer *while*-Endlosschleife und macht nichts anderes, als Daten in die Leitung zu pumpen, so schnell er kann. Der Schleifenrumpf besteht aus drei Aktionen: Hole ein Paket von der (immer bereiten) Vermittlungsschicht, baue in der Variablen *s* einen abgehenden Rahmen auf und schicke den Rahmen ab. Dieses Protokoll nutzt nur das Rahmenfeld *info*, da die anderen Felder Fehlerüberwachung und Flusskontrolle betreffen und es hier ja weder Fehler noch Flusskontrolle gibt.

Der Empfänger ist ähnlich einfach aufgebaut: Zuerst wartet er, bis etwas passiert, wobei die einzige Möglichkeit die Ankunft eines fehlerfreien Rahmens ist. Irgendwann kommt der Rahmen an und die Prozedur `wait_for_event` übergibt in `event` den Wert `frame_arrival` (was aber ohnehin ignoriert wird). Der Aufruf von `from_physical_layer` nimmt den neu angekommenen Rahmen aus dem Hardwarepuffer und legt ihn in der Variablen `r` ab, auf die der Empfängercode zugreifen kann. Schließlich werden die Daten an die Vermittlungsschicht übergeben und die Sicherungsschicht lehnt sich entspannt zurück, um auf den nächsten Rahmen zu warten (sie suspendiert sich sozusagen selbst, bis wieder ein Rahmen eintrifft).

Das Utopia-Protokoll ist unrealistisch, da es weder Flusskontrolle noch Fehlerkorrektur bietet. Die Verarbeitung ähnelt hier der eines unbestätigten verbindungslosen Dienstes, der die Lösung dieser Probleme an höhere Schichten abgibt, obwohl selbst ein unbestätigter verbindungsloser Dienst etwas in puncto Fehlererkennung unternehmen würde.

3.3.2 Ein Simplexprotokoll mit Stop-and-Wait für einen fehlerfreien Kanal

Als Nächstes wollen wir das Problem angehen, wie der Sender davon abgehalten werden kann, den Empfänger schneller mit Rahmen zu überschwemmen als dieser verarbeiten kann. Diese Situation kann in der Praxis leicht entstehen. Die Fähigkeit, dies zu verhindern, ist also von großer Wichtigkeit. Wir nehmen weiterhin an, dass der Kommunikationskanal fehlerfrei ist und der Datenverkehr immer noch in nur eine Richtung geht.

Eine Lösung könnte sein, den Empfänger so leistungsstark zu konstruieren, dass er einen kontinuierlichen Strom von Back-to-Back-Rahmen verarbeiten kann (oder gleichwertig, die Sicherungsschicht so langsam einzurichten, dass der Empfänger durchhalten kann). Der Empfänger muss ausreichend Pufferungsmöglichkeiten und Verarbeitungsfähigkeiten haben, um mit Leitungsrate zu laufen, außerdem muss er die ankommenden Rahmen schnell genug zur Vermittlungsschicht weiterreichen können. Die ist jedoch eine Worst-Case-Lösung, die dedizierte Hardware benötigt und verschwenderisch mit Ressourcen umgeht, falls die Benutzung der Verbindung in der Regel niedrig ist. Mehr noch, die Bewältigung des Problems des zu schnellen Senders wird lediglich an einen anderen Ort verschoben – in diesem Fall zur Vermittlungsschicht.

Eine allgemeinere Lösung dieses Problems erhält man, indem man den Empfänger eine Bestätigung an den Sender schicken lässt. Nach der Weitergabe eines Pakets an die Vermittlungsschicht sendet der Empfänger einen kleinen Leerrahmen an den Sender, der diesem die Erlaubnis erteilt, den nächsten Rahmen zu senden. Nach der Übertragung eines Rahmens lässt das Protokoll den Sender warten, bis der kleine Leerrahmen (d.h. der Bestätigungsrahmen) des Empfängers ankommt. Diese Verzögerung ist ein einfaches Beispiel für ein Flusskontrollprotokoll.

Protokolle, in denen der Sender einen Rahmen sendet und dann auf eine Bestätigung wartet, bevor er weitermacht, werden als **Stop-and-Wait-Protokolle** bezeichnet.

► Abbildung 3.13 zeigt ein Beispiel eines Stop-and-Wait-Simplexprotokolls.

```
/* Protokoll 2 (Stop-and-Wait) definiert ebenfalls einen unidirektionalen
   Datenfluss vom Sender zum Empfänger. Wie in Protokoll 1 wird auch hier
   angenommen, dass der Kommunikationskanal fehlerfrei ist. Diesmal hat der
   Empfänger jedoch nur einen begrenzten Puffer zur Verfügung. Deshalb muss
   das Protokoll dafür sorgen, dass der Sender den Empfänger nicht schneller
   mit Daten überschwemmt, als dieser verarbeiten kann. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
    frame s;                      /* Puffer für einen ausgehenden Rahmen */
    packet buffer;                /* Puffer für ein ausgehendes Paket */
    event_type event;              /* frame_arrival ist die einzige Möglichkeit */

    while (true) {
        from_network_layer(&buffer); /* hole Daten zum Senden */
        s.info = buffer;           /* kopiere diese zum Übertragen in s */
        to_physical_layer(&s);     /* tschüss, kleiner Rahmen */
        wait_for_event(&event);    /* nicht weitermachen, bis grünes Licht
                                       kommt */
    }
}

void receiver2(void)
{
    frame r, s;                  /* Rahmenpuffer */
    event_type event;             /* frame_arrival ist die einzige
                                   Möglichkeit */

    while (true) {
        wait_for_event(&event);  /* einzige Möglichkeit ist frame_arrival */
        from_physical_layer(&r); /* hole den ankommenden Rahmen */
        to_network_layer(&r.info); /* gib die Daten an die Vermittlungs-
                                      schicht weiter */
        to_physical_layer(&s);    /* sende ein Leerrahmen, um den Sender
                                      zu wecken */
    }
}
```

Abbildung 3.13: Simplexprotokoll mit Stop-and-Wait.

Obwohl es sich bei diesem Beispiel um einen Datenverkehr im Simplexmodus handelt, der nur vom Sender zum Empfänger stattfindet, bewegen sich die Rahmen doch in beide Richtungen. Deshalb muss der Kommunikationskanal zwischen den zwei Sicherungsschichten die Datenübertragung in beide Richtungen zulassen. Dieses Protokoll legt jedoch eine strenge Reihenfolge hinsichtlich der Änderung der Flussrichtung fest: Erst schickt der Sender einen Rahmen, dann schickt der Empfänger einen Rahmen, danach schickt der Sender wieder einen Rahmen und dann wieder der Empfänger usw. Ein Halbduplexkanal würde hier also genügen.

Wie bei Protokoll 1 beginnt der Sender damit, ein Paket von der Vermittlungsschicht zu holen, daraus einen Rahmen zu erstellen und ihn abzusenden. Im Gegensatz zu Protokoll 1 muss der Sender jetzt aber warten, bis ein Bestätigungsrahmen ankommt. Erst dann kann er fortfahren und das nächste Paket von der Vermittlungsschicht holen. Die sendende Sicherungsschicht muss den ankommenden Rahmen nicht einmal prüfen, weil es nur eine Möglichkeit gibt. Der eingehende Rahmen ist immer eine Bestätigung.

Der einzige Unterschied zwischen *receiver1* und *receiver2* ist, dass *receiver2* nach Abgabe des Pakets an die Vermittlungsschicht eine Bestätigung an den Sender richtet, bevor er wieder in die Warteschleife eintritt. Da hier nur die Ankunft eines Rahmens beim Sender wichtig ist und nicht dessen Inhalt, braucht der Empfänger keine besonderen Informationen in den Rahmen einzufügen.

3.3.3 Ein Stop-and-Wait-Simplexprotokoll für verrauschte Kanäle

Nun wollen wir uns mit der üblichen Situation eines Kommunikationskanals befassen, der Fehler macht. Rahmen können beschädigt werden oder völlig verloren gehen. Wir nehmen hier an, dass die Hardware auf der Empfängerseite erkennt, wenn ein Rahmen während seiner Übertragung beschädigt wurde, sobald sie die Prüfsumme errechnet. Ist der Rahmen so beschädigt, dass die Prüfsumme dennoch korrekt ist – was äußerst selten der Fall ist –, so kann dieses Protokoll (und alle anderen) versagen (d.h., es wird ein fehlerhaftes Paket an die Vermittlungsschicht übertragen).

Auf den ersten Blick sieht es so aus, als ob eine kleinere Veränderung von Protokoll 2 schon ausreichen würde: Hinzufügen eines Timers. In diesem Fall überträgt der Sender einen Rahmen, der Empfänger sendet aber nur dann eine Bestätigung, wenn die Daten korrekt angekommen sind. Kommt beim Empfänger ein zerstörter Rahmen an, wird er verworfen. Nach einer bestimmten Zeit läuft der Timer des Senders ab und der Rahmen wird noch einmal gesendet. Dieser Vorgang wird so oft wiederholt, bis der Rahmen in einwandfreiem Zustand ankommt.

Dieses Schema enthält allerdings einen fatalen Fehler. Denken Sie noch einmal über das Problem nach und versuchen Sie herauszufinden, was schiefgehen könnte, bevor Sie weiterlesen.

Vergessen Sie nicht, dass das Ziel der Sicherungsschicht die Bereitstellung einer fehlerfreien transparenten Kommunikation zwischen den Prozessen der Vermittlungsschichten ist. Die Vermittlungsschicht auf Rechner A gibt eine Reihe von Paketen an die Sicherungsschicht ab. Diese wiederum muss gewährleisten, dass auf Rechner B genau die gleiche Reihe von Paketen durch die Sicherungsschicht an die Vermittlungsschicht weitergegeben wird. Insbesondere hat die Vermittlungsschicht auf Rechner B keine Möglichkeit zu erkennen, dass ein Paket verloren gegangen ist oder doppelt vor kommt. Die Sicherungsschicht muss also garantieren, dass es keine Kombination von Übertragungsfehlern gibt, die dazu führen können, dass Pakete doppelt an die Vermittlungsschicht gesendet werden, egal wie unwahrscheinlich das sein mag.

Betrachten Sie folgendes Szenario:

1. Die Vermittlungsschicht auf A sendet Paket 1 an ihre Sicherungsschicht. B erhält das Paket korrekt und gibt es an die Vermittlungsschicht auf der B-Seite ab. B sendet einen Bestätigungsrahmen zurück an A.
2. Der Bestätigungsrahmen geht vollständig verloren. Er kommt einfach nie an. Das Leben wäre um sehr vieles einfacher, wenn der Kommunikationskanal nur Datenrahmen, aber keine Steuerrahmen demolieren und verlieren würde – bedauerlicherweise ist der Kanal jedoch nicht sehr wählerisch.
3. Auf der Sicherungsschicht von A läuft schließlich der Timer ab. Da keine Bestätigung angekommen ist, nimmt die Sicherungsschicht fälschlicherweise an, dass der Datenrahmen verloren gegangen oder beschädigt worden ist, und sendet den Rahmen mit Paket 1 noch einmal.
4. Das Duplikat kommt bei der Sicherungsschicht von B unbeschädigt an und wird von der ahnungslosen Sicherungsschicht an die Vermittlungsschicht weitergegeben. Sendet A eine Datei an B, wird ein Teil der Datei verdoppelt (d.h. die von B angefertigte Kopie der Datei ist fehlerhaft, was aber nicht entdeckt wird). Mit anderen Worten: das Protokoll hat versagt.

Es ist also offensichtlich nötig, dass der Empfänger Rahmen, die er zum ersten Mal sieht, von erneut übertragenen Rahmen unterscheiden kann. Der einfachste Weg, das zu erreichen, ist der, dass der Sender eine Sequenznummer im Nachrichten-Header jedes gesendeten Rahmens angibt. Der Empfänger kann dann die Sequenznummer jedes ankommenden Rahmens prüfen, um festzustellen, ob der Rahmen neu oder ein Duplikat ist, das verworfen werden kann.

Da das Protokoll korrekt sein muss und das Sequenznummerfeld im Header wahrscheinlich zu klein ist, um die Verbindung effizient zu benutzen, taucht die Frage auf: Was ist die für Sequenznummern erforderliche Mindestzahl an Bits? Der Header könnte, je nach Protokoll, 1 Bit, ein paar Bits, 1 Byte oder mehrere Bytes für eine Sequenznummer bereitstellen. Der springende Punkt ist, dass Sequenznummern übertragen werden müssen, die groß genug sind, damit das Protokoll korrekt arbeiten kann, oder es ist kein richtiges Protokoll.

Die einzige Mehrdeutigkeit in diesem Protokoll besteht zwischen einem Rahmen m und seinem direkten Nachfolger $m+1$. Wenn Rahmen m verloren geht oder beschädigt wird, bestätigt der Empfänger dessen Erhalt nicht und der Sender versucht weiter, den Rahmen zu übertragen. Sobald dieser korrekt angekommen ist, sendet der Empfänger eine Bestätigung an den Sender. Genau an dieser Stelle könnten wir in Schwierigkeiten kommen. Je nachdem, ob der Bestätigungsrahmen wieder korrekt beim Empfänger ankommt, sendet der Sender unter Umständen m oder $m+1$.

Das Ereignis, das beim Sender die Übertragung des Rahmens $m+1$ auslöst, ist die Ankunft der Bestätigung für den Rahmen m . Diese Situation impliziert aber, dass $m-1$ korrekt empfangen wurde und dass außerdem die dazugehörige Bestätigung beim Sen-

der korrekt angekommen ist. Andernfalls hätte der Sender nicht mit m begonnen, geschweige denn das Senden von $m+1$ in Betracht gezogen. Deshalb kann die einzige Verwechslung zwischen einem Rahmen und seinem unmittelbaren Vorgänger oder Nachfolger auftreten, nicht aber zwischen dem Vorgänger und dem Nachfolger des Rahmens.

Es genügt also eine Sequenznummer mit einem Bit (0 oder 1). Der Empfänger erwartet zu jedem Zeitpunkt eine bestimmte Sequenznummer. Kommt ein Rahmen mit der korrekten Sequenznummer an, wird er akzeptiert und an die Vermittlungsschicht übergeben, dann bestätigt. Daraufhin wird die erwartete Sequenznummer modulo 2 inkrementiert (d.h. 0 wird 1 und 1 wird 0). Jeder Rahmen, der mit der falschen Sequenznummer ankommt, wird als Duplikat zurückgewiesen. Die letzte gültige Bestätigung wird jedoch wiederholt, sodass der Sender letzten Endes mitbekommt, dass der Rahmen empfangen wurde.

Ein Beispiel für diesen Protokolltyp zeigt ►Abbildung 3.14. Protokolle, bei denen der Sender auf eine positive Bestätigung wartet, bevor er die nächsten Daten übermittelt, werden häufig **ARQ**-Protokolle (*Automatic Repeat reQuest*, automatische Wiederholungsanforderung) oder **PAR**-Protokolle (*Positive Acknowledgement with Retransmission*, positive Bestätigung mit erneuter Übertragung) genannt. Wie Protokoll 2 übermittelt auch diese Protokollart die Daten nur in eine Richtung.

```
/* Mit Protokoll 3 (PAR) können Daten unidirektional über einen unzuverlässigen
   Kommunikationskanal übertragen werden. */

#define MAX_SEQ 1           /* für Protokoll 3 muss das 1 sein */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
    seq_nr next_frame_to_send;      /* Sequenznummer des nächsten
                                      ausgehenden Rahmens */
    frame s;                      /* Hilfsvariable */
    packet buffer;                /* Puffer für ein ausgehendes Paket */
    event_type event;

    next_frame_to_send = 0;         /* initialisiere Ausgangssequenz-
                                    nummer */
    from_network_layer(&buffer);  /* hole erstes Paket */

    while (true) {
        s.info = buffer;
        s.seq = next_frame_to_send;
        to_physical_layer(&s);
        start_timer(s.seq);

        wait_for_event(&event);
        if (event == frame_arrival) {
            from_physical_layer(&s);
            if (s.ack == next_frame_to_send) {
                stop_timer(s.ack);      /* Timer deaktivieren */
                from_network_layer

```

```

        (&buffer);
        /* hole die nächsten zu sendenden
           Daten */
        inc(next_frame_to_send); /* invertiere next_frame_to_send */
    }
}

void receiver3(void)
{
    seq_nr frame_expected;
    frame r, s;
    event_type event;

    frame_expected = 0;
    while (true) {
        wait_for_event(&event);
        if (event == frame_arrival) {
            from_physical_layer(&r);
            if (r.seq == frame_expected) {
                to_network_layer(&r.info);
                /* Möglichkeiten: frame_arrival,
                   cksum_err */
                /* ein gültiger Rahmen ist
                   angekommen */
                /* hole neu angekommenen Rahmen */
                /* darauf haben wir gewartet */
                /* gib die Daten an die Vermittlungs-
                   schicht weiter */
                /* beim nächsten Mal wird die andere
                   Sequenznummer erwartet */
                inc(frame_expected);
                /* bestimme, welcher Rahmen
                   bestätigt wird */
                /* sende Bestätigung */
                s.ack = 1 + frame_expected;
                to_physical_layer(&s);
            }
        }
    }
}

```

Abbildung 3.14: PAR-Protokoll.

Protokoll 3 unterscheidet sich von seinen Vorgängern darin, dass Sender und Empfänger eine Variable haben, deren Wert gespeichert wird, während sich die Sicherungsschicht im Wartezustand befindet. Der Sender speichert die Sequenznummer des nächsten Rahmens in *next_frame_to_send*. Der Empfänger speichert die Sequenznummer des nächsten erwarteten Rahmens in *frame_expected*. Jedes Protokoll hat eine kurze Initialisierungsphase, bevor es in seine Endlosschleife eintritt.

Nach der Übertragung eines Rahmens schaltet der Sender den Timer ein. War der Timer bereits aktiv, wird er wieder zurückgesetzt, sodass sich ein volles Timerintervall ergibt. Die Zeitspanne muss so gewählt werden, dass der Rahmen beim Empfänger ankommen, der Empfänger den Rahmen auch im schlechtesten Fall verarbeiten und der Bestätigungsrahmen beim Sender ankommen kann. Erst nach Ablauf dieses Intervalls kann man sicher sein, dass entweder der übermittelte Rahmen oder seine Bestätigung verloren gegangen ist und ein Duplikat gesendet werden muss. Ist das Timeout-Intervall zu kurz, dann überträgt der Sender unnötig Rahmen. Wenngleich diese zusätzlichen Rahmen auch die Korrektheit des Protokolls nicht beeinträchtigen, so mindern sie doch die Leistung.

Nach der Übertragung des Rahmens und dem Starten des Timers wartet der Sender auf das nächste große Ereignis. Es gibt nur drei Möglichkeiten: Ein Bestätigungsrahmen kommt unbeschädigt an, ein Bestätigungsrahmen kommt fehlerhaft an oder der Timer läuft ab. Kommt ein gültiger Bestätigungsrahmen beim Sender an, dann holt dieser das nächste Paket von der Vermittlungsschicht und speichert es im Puffer, wodurch das vorhergehende Paket gelöscht wird. Außerdem erhöht er die Sequenznummer. Wenn ein beschädigter Rahmen ankommt oder der Timer abläuft, werden weder die Informationen im Puffer noch die Sequenznummer geändert, sodass ein Duplikat gesendet werden kann. Auf jeden Fall wird dann der Inhalt des Puffers gesendet (entweder das nächste Paket oder ein Duplikat).

Kommt ein korrekter Rahmen beim Empfänger an, dann wird die Sequenznummer des Rahmens überprüft, um zu sehen, ob der Rahmen ein Duplikat ist. Falls nicht, wird er angenommen, an die Vermittlungsschicht weitergegeben und es wird eine Bestätigung erstellt. Duplikeate und beschädigte Rahmen werden nicht an die Vermittlungsschicht weitergegeben, aber sie bewirken, dass der letzte korrekt empfangene Rahmen bestätigt wird, um damit dem Sender zu signalisieren, mit dem nächsten Rahmen fortzufahren oder einen beschädigten Rahmen noch einmal zu übertragen.

3.4 Schiebefensterprotokolle

In den bisher besprochenen Protokollen wurden Datenrahmen in eine einzige Richtung übertragen. In der Praxis werden jedoch Übertragungsmöglichkeiten in beide Richtungen benötigt. Für Datenübertragungen im Vollduplexmodus könnte man zwei Instanzen von einem der vorherigen Protokolle ausführen, wobei jedes eine separate Verbindung für den Simplexdatenverkehr verwendet (in verschiedene Richtungen). Jede Verbindung besteht dann aus einem Hauptkanal (für Daten) und einem Rückkanal (für Bestätigungen). In beiden Fällen wäre die Kapazität des Rückkanals fast vollständig verschwendet.

Eine bessere Lösung ist der Einsatz der gleichen Verbindung für die Datenübertragung in beide Richtungen. Schließlich wurde in den Protokollen 2 und 3 bereits eine Leitung für die Rahmenübertragung in beide Richtungen verwendet, wobei der Rückkanal normalerweise die gleiche Kapazität hat wie der Hauptkanal. Bei diesem Modell werden die Daten- und Bestätigungsrahmen von *A* nach *B* gemischt. Inspiziert der Empfänger das Feld *kind* im Nachrichten-Header des ankommenden Rahmens, erkennt er sofort, ob der Rahmen Daten enthält oder ein Bestätigungsrahmen ist.

Obwohl die Überlappung von Daten- und Steuerrahmen auf der gleichen Verbindung gegenüber dem Einsatz von zwei getrennten Verbindungen schon ein großer Vorteil ist, gibt es noch eine zusätzliche Verbesserungsmöglichkeit: Wenn ein Datenrahmen ankommt, schickt der Empfänger nicht sofort einen Bestätigungsrahmen, sondern hält sich damit zurück, bis die Vermittlungsschicht das nächste Paket übermittelt. Die Bestätigung wird an den ausgehenden Datenrahmen angehängt (im Feld *ack* im Rahmen-Header). So bekommt die Bestätigung also eine Freifahrt auf dem nächsten

Datenrahmen, der gesendet wird. Die Methode, ausgehende Bestätigungen kurzzeitig zu verzögern, sodass sie an dem nächsten ausgehenden Datenrahmen auf den Rücken geschnallt werden können, wird **Huckepacktransport** (*piggybacking*) genannt.

Der wesentliche Vorteil des Huckepacktransports gegenüber der gesonderten Sendung von Bestätigungsrahmen ist der, dass die verfügbare Bandbreite des Kanals besser genutzt wird. Das *ack*-Feld im Rahmen-Header kostet nur wenige Bits, während für einen separaten Rahmen ein Nachrichten-Header, die Bestätigung und eine Prüfsumme benötigt werden. Ferner bedeutet eine geringere Anzahl von übertragenen Rahmen in der Regel eine schwächere Verarbeitungslast beim Empfänger. In unserem nächsten Protokoll kostet uns das Feld für den Huckepacktransport nur ein Bit im Rahmen-Header. Selten werden mehr als ein paar Bits benötigt.

Der Huckepacktransport bringt jedoch eine Komplikation mit sich, die wir bei der getrennten Übertragung nicht haben. Wie lange sollte die Sicherungsschicht auf ein Paket warten, an das sie die Bestätigung anhängen kann? Ist die Wartezeit länger als das Timeout-Intervall des Senders, dann wird der Rahmen noch einmal übertragen – was dem Zweck der Bestätigungsprozedur zuwider läuft. Hätte die Sicherungsschicht prophetische Kräfte und könnte die Zukunft vorhersagen, dann wüsste sie, wann das nächste Paket der Vermittlungsschicht ankommt. Sie könnte sich somit entscheiden, ob sie darauf warten oder die Bestätigung getrennt senden soll, je nachdem, wie lange die vorhergesehene Wartezeit wäre. Natürlich ist unsere Sicherungsschicht keine Hellseherin, sondern muss Zuflucht zu Ad-hoc-Schemata nehmen, wie z.B. eine bestimmte Dauer in Millisekunden zu warten. Kommt ein neues Paket schnell an, wird ihm die Bestätigung huckepack aufgeladen. Wenn andererseits bis zum Ende des Zeitintervalls kein neues Paket angekommen ist, sendet die Sicherungsschicht einen separaten Bestätigungsrahmen.

Die nächsten drei Protokolle sind bidirektionale Protokolle, die zur Klasse der sogenannten **Schiebefensterprotokolle** (*sliding windows protocol*) gehören. Sie unterscheiden sich in Bezug auf Effizienz, Komplexität und Pufferanforderungen, was später genauer erörtert wird. Bei allen Schiebefensterprotokollen enthält jeder zu sendende Rahmen eine Sequenznummer von 0 bis zu einem bestimmten Maximum. Das Maximum ist normalerweise $2^n - 1$, damit die Sequenznummer in ein n -Bit-Feld passt. Das Stop-and-Wait-Schiebefensterprotokoll verwendet $n=1$, wobei die Sequenznummern auf 0 oder 1 beschränkt sind. Ausgefaltete Versionen können ein beliebiges n verwenden.

Das Hauptmerkmal aller Schiebefensterprotokolle ist, dass der Sender ständig eine Liste von aufeinanderfolgenden Sequenznummern führt, die der Anzahl der Rahmen entspricht, die er senden darf. Man sagt, dass diese Rahmen genau in das sogenannte **Sendefenster** fallen. Ähnlich benutzt der Empfänger ein **Empfangsfenster**, das den Rahmen entspricht, die er annehmen darf. Die Fenster des Senders und Empfängers müssen nicht die gleichen Unter- und Obergrenzen, noch nicht einmal die gleiche Größe haben. In einigen Protokollen ist die Größe festgelegt, in anderen aber kann sie im Laufe der Zeit mit dem Senden oder Empfangen von Rahmen größer oder kleiner werden.

Obwohl diese Protokolle der Sicherungsschicht größere Freiheiten bezüglich der Reihenfolge lassen, in der sie Rahmen senden und empfangen darf, haben wir ganz bewusst folgende Bedingung nicht aufgehoben: Das Protokoll muss Pakete an die Vermittlungsschicht der Zielmaschine in genau der Reihenfolge übertragen, in der die Pakete an die Sicherungsschicht auf der Quellmaschine übergeben wurden. Zudem haben wir beibehalten, dass das eigentliche Verhalten des Kommunikationskanals der kabelgebundenen Übertragung entspricht (d.h., die Rahmen müssen in der Reihenfolge, in der sie gesendet werden, beim Empfänger abgeliefert werden).

Die Sequenznummern im Sendefenster stellen Rahmen dar, die versendet wurden oder noch versendet werden können, aber noch nicht bestätigt wurden. Sobald aus der Vermittlungsschicht ein neues Paket ankommt, wird ihm die nächsthöhere Sequenznummer zugeteilt und die Obergrenze des Fensters wird um eins verschoben. Kommt eine Bestätigung an, wird die untere Begrenzung des Fensters um eins weitergerückt. So enthält das Fenster immer eine Liste der unbestätigten Rahmen.

► Abbildung 3.15 zeigt dies an einem Beispiel.

Da Rahmen, die sich gerade im Sendefenster befinden, schließlich doch noch verloren gehen oder beschädigt werden können, muss der Sender alle diese Rahmen in seinem Speicher halten, um sie eventuell noch einmal übertragen zu können. Wenn also die maximale Größe des Fensters n beträgt, braucht der Sender n Puffer, um alle unbestätigten Rahmen aufnehmen zu können. Wächst das Fenster bis zu seiner Höchstgrenze an, dann muss die Sicherungsschicht auf der Senderseite die Vermittlungsschicht zwangsweise blockieren, bis wieder ein Puffer frei wird.

Das Fenster der Sicherungsschicht auf der Empfängerseite entspricht der Anzahl der Rahmen, die sie aufnehmen kann. Jeder Rahmen, der innerhalb des Fensters fällt, wird im Puffer des Empfängers gespeichert. Kommt ein Rahmen an, dessen Sequenznummer der Untergrenze des Fensters entspricht, wird der Rahmen an die Vermittlungsschicht weitergegeben und das Fenster wird um eins weiterbewegt. Jeder Rahmen, der nicht in das Fenster fällt, wird kommentarlos verworfen. In all diesen Fällen wird eine nachfolgende Bestätigung erzeugt, sodass der Sender herausfinden kann, wie er weiter verfahren soll. Beachten Sie, dass eine Fenstergröße von 1 bedeutet, dass die Sicherungsschicht Rahmen nur der Reihe nach annimmt. Dies gilt aber nicht für größere Fenster. Die Vermittlungsschicht andererseits erhält ihre Daten immer in der richtigen Reihenfolge, unabhängig davon, welche Größe das Fenster der Sicherungsschicht hat.

Abbildung 3.15 ist ein Beispiel mit einer maximalen Fenstergröße von 1. Anfangs stehen keine Rahmen aus, folglich sind Ober- und Untergrenze des Sendefensters gleich. Im Laufe der Zeit entsteht aber die hier dargestellte Situation. Anders als das Sendefenster behält das Empfangsfenster immer seine ursprüngliche Größe und rotiert, wenn der nächste Rahmen angenommen und zur Vermittlungsschicht weitergereicht wird.

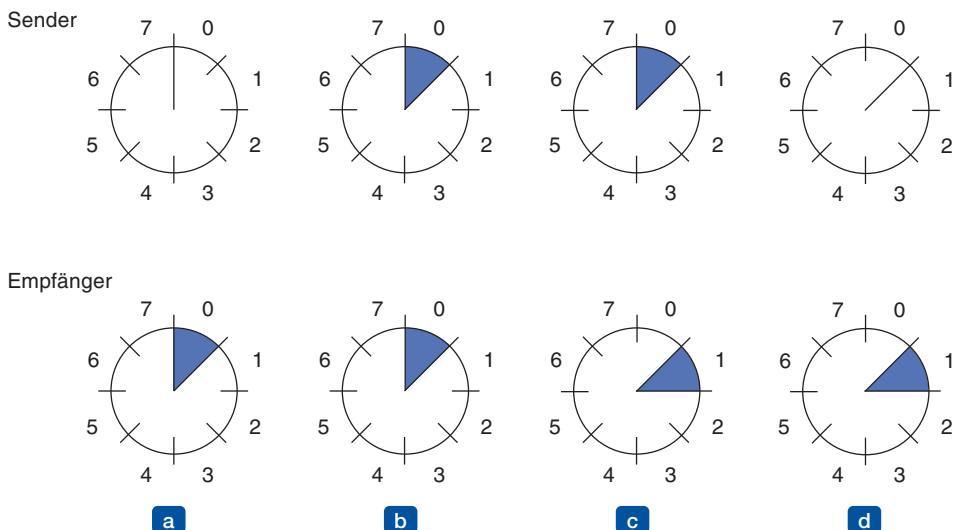


Abbildung 3.15: Schiebefenster der Größe 1 mit einer 3-Bit-Sequenznummer: (a) Anfangszustand, (b) nach Senden des ersten Rahmens, (c) nach Empfang des ersten Rahmens, (d) nach Empfang der ersten Bestätigung.

3.4.1 1-Bit-Schiebefensterprotokoll

Bevor wir uns dem allgemeinen Fall zuwenden, wollen wir ein Schiebefensterprotokoll mit einer Fenstergröße von 1 untersuchen. Ein solches Protokoll geht nach dem Stop-and-Wait-Verfahren vor: Der Sender übermittelt einen Rahmen und wartet auf die zugehörige Bestätigung, bevor er den nächsten abschickt.

► Abbildung 3.16 zeigt ein solches Protokoll. Wie bei den anderen beginnt es mit der Definition einiger Variablen. Mit *next_frame_to_send* wird bezeichnet, welchen Rahmen der Sender als Nächstes zu übertragen versucht, und *frame_expected* sagt aus, welchen Rahmen der Empfänger erwartet. In beiden Fällen sind 0 und 1 die einzigen Möglichkeiten.

```
/* Protokoll 4 (Schiebefenster) ist bidirektional. */

#define MAX_SEQ 1                                /* für Protokoll 4 muss das 1
                                                 sein*/
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void protocol4 (void)
{
    seq_nr next_frame_to_send;                  /* nur 0 oder 1 */
    seq_nr frame_expected;                      /* nur 0 oder 1 */
    frame r, s;                                /* Hilfsvariablen */
    packet buffer;                             /* aktuell zu übertragendes
                                                 Paket */
    event_type event;
```

```

next_frame_to_send = 0;           /* nächster Rahmen des
frame_expected = 0;             Ausgangsstroms */
from_network_layer(&buffer);    /* Nummer des nächsten erwarteten
s.info = buffer;                Rahmens */
s.seq = next_frame_to_send;      /* hole Paket von der
s.ack = 1 - frame_expected;    Vermittlungsschicht */
to_physical_layer(&s);          /* bereite einen Rahmen zum
start_timer(s.seq);            Senden vor */
/* füge Sequenznummer in Rahmen
   ein */
/* Huckepackbestätigung */
/* übertrage den Rahmen */
/* starte den Timer */

while (true) {
    wait_for_event(&event);

    if (event == frame_arrival) {
        from_physical_layer(&r);
        if (r.seq == frame_expected) {
            to_network_layer(&r.info);
            inc(frame_expected);
        }
        if (r.ack == next_frame_to_send) {
            stop_timer(r.ack);
            from_network_layer(&buffer);
            inc(next_frame_to_send);
        }
        s.info = buffer;
        s.seq = next_frame_to_send;
        s.ack = 1 - frame_expected;
        to_physical_layer(&s);
        start_timer(s.seq);
    }
}

```

/* nächste Rahmen des
Ausgangsstroms */
/* Nummer des nächsten erwarteten
Rahmens */
/* hole Paket von der
Vermittlungsschicht */
/* bereite einen Rahmen zum
Senden vor */
/* füge Sequenznummer in Rahmen
ein */
/* Huckepackbestätigung */
/* übertrage den Rahmen */
/* starte den Timer */

/* frame_arrival, cksum_err,
oder timeout */
/* ein Rahmen ist unbeschädigt
angekommen */
/* hole ihn */
/* bearbeite ankommenden
Rahmenstrom */
/* gib das Paket weiter zur
Vermittlungsschicht */
/* invertiere nächste erwartete
Sequenznummer */

/* bearbeite ausgehenden
Rahmenstrom */
/* deaktiviere Timer */
/* hole neues Paket von
Vermittlungsschicht */
/* invertiere Sequenznummer
des Senders */

/* erstelle den Ausgangsrahmen */
/* füge Sequenznummer ein */
/* Sequenznummer des zuletzt
eingegangenen Rahmens */
/* übertrage einen Rahmen */
/* starte den Timer */

Abbildung 3.16: Ein 1-Bit-Schiebefensterprotokoll.

Normalerweise beginnt eine der beiden Sicherungsschichten mit der Übertragung des ersten Rahmens. Anders ausgedrückt, sollte nur eines der Programme der Sicherungsschicht die Prozeduraufrufe *to_physical_layer* und *start_timer* außerhalb der Hauptschleife aufweisen. Der startende Rechner holt das erste Paket von der Vermittlungsschicht, erstellt einen Rahmen und sendet diesen. Sobald dieser (oder ein anderer) Rahmen ankommt, prüft die empfangende Sicherungsschicht genau wie bei Protokoll 3,

ob der Rahmen ein Duplikat ist. Ist es der erwartete Rahmen, so wird er an die Vermittlungsschicht weitergegeben und das Fenster des Empfängers wird nach oben verschoben.

Das Bestätigungsfeld enthält die Nummer des letzten fehlerfrei empfangenen Rahmens. Stimmt diese Nummer mit der Sequenznummer des Rahmens überein, der gesendet werden soll, dann weiß der Sender, dass der Rahmen in *buffer* erledigt ist, und kann das nächste Paket von seiner Vermittlungsschicht holen. Stimmt die Sequenznummer nicht überein, muss der Sender weiter versuchen, den gleichen Rahmen zu senden. Jedes Mal wenn ein Rahmen angekommen ist, wird auch einer zurückgesendet.

Nun wollen wir Protokoll 4 dahingehend untersuchen, wie belastbar es unter ungünstigen Bedingungen ist. Nehmen wir an, A versucht, seinen Rahmen 0 an B zu senden, und B versucht, seinen Rahmen 0 an A zu verschicken. A sendet seinen Rahmen an B, aber der Timer von A ist ein wenig zu kurz eingestellt. Also unterbricht A immer wieder und sendet eine Reihe von identischen Rahmen, alle mit $seq=0$ und $ack=1$.

Der erste gültige Rahmen, der bei B ankommt, wird angenommen und *frame_expected* wird auf den Wert 1 gesetzt. Alle nachfolgenden Rahmen werden nicht angenommen, weil B nun Rahmen mit der Sequenznummer 1 und nicht 0 erwartet. Außerdem holt B kein neues Paket von der Vermittlungsschicht, da alle Duplikate $ack=1$ aufweisen und B immer noch auf die Bestätigung von Rahmen 0 wartet.

Nach der Ankunft jedes zurückgewiesenen Duplikats schickt B einen Rahmen an A, der $seq=0$ und $ack=0$ enthält. Irgendwann kommt einer dieser Rahmen korrekt bei A an und veranlasst A, das nächste Paket zu senden. Es ist durch keine Kombination aus verlorenen Rahmen und vorzeitigem Timeout möglich, dass das Protokoll Duplikatpakete an eine der Vermittlungsschichten sendet, ein Paket überspringt oder in einen Deadlock gerät. Das Protokoll ist korrekt.

Um jedoch zu zeigen, wie subtil Protokollinteraktionen sein können, betrachten wir die Situation, wenn beide Seiten gleichzeitig ein Anfangspaket senden – hier kommt es zu einem Sonderfall. Diese Schwierigkeit bei der Synchronisation ist in ►Abbildung 3.17 gezeigt. In ►Abbildung 3.17a wird der normale Ablauf des Protokolls geschildert. In ►Abbildung 3.17b sehen wir den Sonderfall. Wenn B auf den ersten Rahmen von A wartet, bevor B seinen eigenen ersten sendet, sieht das Ganze wie in Abbildung 3.17a aus und jeder Rahmen wird angenommen.

Fangen jedoch A und B gleichzeitig zu senden an, kreuzen sich ihre ersten Rahmen und die Sicherungsschichten kommen in die in Abbildung 3.17b gezeigte Lage. In Abbildung 3.17a enthält jeder ankommende Rahmen ein neues Paket für die Vermittlungsschicht – es gibt keine Duplikate. In Abbildung 3.17b ist die Hälfte der Rahmen Duplikate, obwohl keine Übertragungsfehler vorkommen. Ähnliche Situationen können durch vorzeitige Timeouts entstehen, auch wenn eindeutig eine der beiden Seiten als Erstes zu übertragen beginnt. Finden mehrere vorzeitige Timeouts statt, werden Rahmen eventuell dreimal oder noch häufiger gesendet, wodurch wertvolle Bandbreite verschwendet wird.

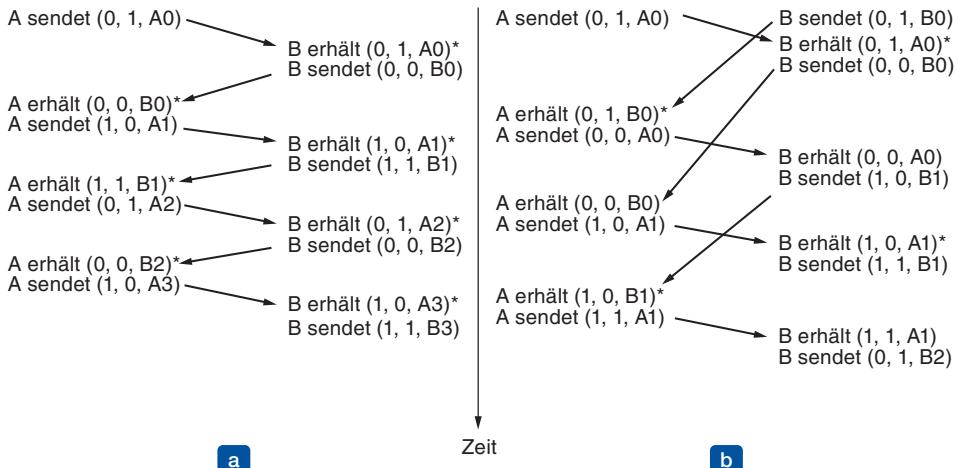


Abbildung 3.17: Zwei Szenarien für Protokoll 4: (a) Normalfall, (b) Sonderfall. Die Schreibweise lautet *(seq, ack, Paketnummer)*. Ein Sternchen gibt an, wo eine Vermittlungsschicht ein Paket annimmt.

3.4.2 Protokoll mit Go-back-N

Bis jetzt sind wir stillschweigend davon ausgegangen, dass die Übertragungszeit, die ein Rahmen braucht, bis er beim Empfänger ankommt, sowie die zum Senden der Bestätigung erforderliche Zeit vernachlässigt werden können. Manchmal ist diese Annahme allerdings schlichtweg falsch. In solchen Fällen kann eine lange Übertragungszeit gravierende Auswirkungen auf die effiziente Nutzung der Bandbreite haben. Betrachten wir z.B. einen Satellitenkanal mit 50 kbit/s und einer Signalausbreitungsverzögerung pro Paketumlaufzeit (*round-trip delay*) von 500 ms. Angenommen, wir benutzen Protokoll 4 zur Übertragung eines 1 000-Bit-Rahmens über Satellit. Bei $t=0$ beginnt der Sender mit der Übertragung des ersten Rahmens. Bei $t=20$ ms ist die Übertragung beendet. Frühestens bei $t=270$ ms ist der Rahmen vollständig beim Empfänger angekommen und frühestens bei $t=520$ ms erreicht die Bestätigung den Sender unter den besten Umständen (d.h. ohne Wartezeit beim Empfänger und bei einem kleinen Bestätigungsrahmen). Dies bedeutet, dass der Sender zwischen 500/520 bzw. 96 % der Zeit blockiert war. Anders ausgedrückt, wurden nur 4 % der verfügbaren Bandbreite genutzt. Dies verdeutlicht, dass die Kombination von langen Übertragungszeiten, großer Bandbreite und kleinen Rahmengrößen katastrophale Auswirkungen auf die Effizienz hat.

Das oben geschilderte Problem kann als Folge davon angesehen werden, dass der Sender auf eine Bestätigung warten muss, bevor er den nächsten Rahmen senden kann. Durch Lockerung dieser Vorgabe kann die Effizienz gesteigert werden. Grundsätzlich können wir das Problem lösen, wenn der Sender bis zu w Rahmen abschicken darf (statt nur einen), bevor er blockiert. Wird w groß genug gewählt, so kann der Sender fortlaufend Rahmen senden, da die Bestätigungen für vorhergehende Rahmen ankommen, bevor das Fenster voll wird, wodurch die Blockierung des Senders verhindert wird.

Um einen angemessenen Wert für w zu finden, müssen wir wissen, wie viele Rahmen in den Kanal passen, wenn sie vom Sender zum Empfänger wandern. Diese Kapazität wird bestimmt durch die Bandbreite in Bit/s multipliziert mit der einfachen Übertragungsdauer, auch **Bandbreite-Verzögerung-Produkt** (*bandwidth-delay product*) der Verbindung genannt. Wir können diese Größe durch die Anzahl der Bits in einem Rahmen dividieren, um diese als Anzahl von Rahmen auszudrücken. Wir wollen diese Größe BD nennen. Dann sollte w auf $2BD+1$ gesetzt werden. Betrachtet man die gesamte Paketumlaufzeit zum Erhalt einer Bestätigung, dann entspricht die doppelte Bandbreite-Verzögerung der Anzahl von Rahmen, die noch ausstehen können, falls der Sender kontinuierlich Rahmen sendet. Die „+1“ kommt daher, dass kein Bestätigungsrahmen gesendet wird, bevor nicht der vollständige Rahmen angekommen ist.

Für die Beispielverbindung beträgt bei einer Bandbreite von 50 kbit/s und einer einfachen Übertragungszeit von 250 ms das Bandbreite-Verzögerung-Produkt 12,5 kbit bzw. 12,5 Rahmen von je 1 000 Bits. $2BD+1$ entspricht dann 26 Rahmen. Angenommen, der Sender fängt wie vorher mit der Übertragung von Rahmen 0 an und sendet alle 20 ms einen neuen Rahmen. Ist er bei $t=520$ ms mit der Übertragung von 26 Rahmen fertig, dann ist gerade die Bestätigung für Rahmen 0 eingetroffen. Danach treffen alle 20 ms Bestätigungen ein und der Sender erhält immer dann die Erlaubnis weiterzusenden, wenn er sie braucht. Von jetzt an sind also laufend 25 oder 26 Rahmen nicht bestätigt. Anders ausgedrückt, ist die maximale Fenstergröße des Senders (sein Zeitfenster) 26.

Für kleinere Fenstergrößen wird die Ausnutzung der Verbindung weniger als 100 % betragen, da der Sender manchmal blockiert ist. Wir können die Ausnutzung als den Anteil der Zeit schreiben, in welcher der Sender nicht blockiert ist:

$$\text{Verbindungsausnutzung} \leq \frac{w}{1+2BD}$$

Dieser Wert ist eine obere Grenze, weil keine Verarbeitungszeit für die Rahmen vorgesehen ist und der Bestätigungsrahmen so behandelt wird, als hätte er die Länge null, da er üblicherweise kurz ist. Diese Ungleichung zeigt, dass ein großes Fenster w immer dann notwendig ist, wenn das Bandbreite-Verzögerung-Produkt groß ist. Ist die Verzögerungszeit hoch, dann erschöpft der Sender schnell sein Fenster, selbst bei moderater Bandbreite wie im Satellitenbeispiel. Ist die Bandbreite hoch, wird der Sender selbst bei kleinen Verzögerungen sein Fenster schnell ausschöpfen, außer das Fenster ist sehr groß (z.B. hält eine 1-Gbit/s-Verbindung mit einer 1-ms-Verzögerungszeit 1 Mbit). Bei Stop-and-Wait ($w=1$) wird die Effizienz weniger als 50 % betragen, wenn die Verzögerung mindestens eine Rahmensexpeditionszeit beträgt.

Diese Technik, mehrere Rahmen im Fluss zu halten, ist ein Beispiel für **Pipelining**. Die Anwendung von Pipelining zur Rahmenübertragung über einen unzuverlässigen Kommunikationskanal wirft einige wichtige Fragen auf. Was passiert beispielsweise, wenn ein Rahmen in der Mitte eines langen Datenflusses zerstört wird oder verloren geht? Sehr viele nachfolgende Rahmen kommen beim Empfänger an, bevor der Sender herausfindet, dass etwas schiefgegangen ist. Sobald ein zerstörter Rahmen beim Empfänger ankommt, sollte er logischerweise verworfen werden. Doch was soll der Empfänger

mit all den darauffolgenden korrekten Rahmen tun? Wir dürfen nicht vergessen, dass die Sicherungsschicht auf der Empfängerseite dazu verpflichtet ist, die Pakete in einer ganz bestimmten Reihenfolge an die Vermittlungsschicht weiterzugeben.

Bei der Fehlerbehandlung beim Pipelining gibt es zwei grundlegende Ansätze, die beide in ▶ Abbildung 3.18 dargestellt sind.

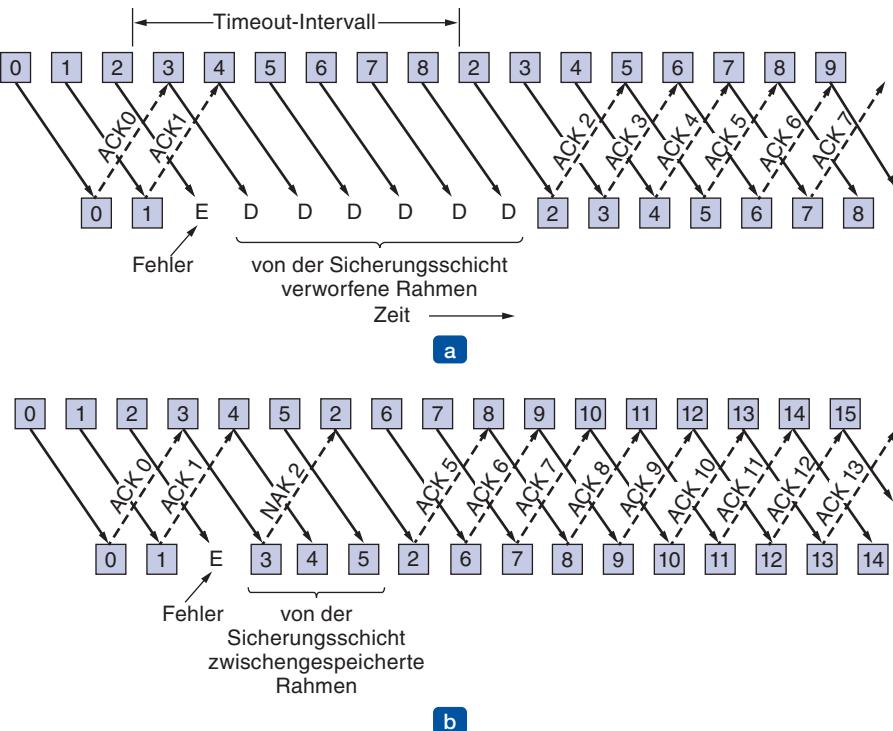


Abbildung 3.18: Pipelining und Wiederherstellung nach einem Fehler. Auswirkungen eines Fehlers, wenn (a) das Empfangsfenster die Größe 1 hat, (b) bei einem großen Empfangsfenster.

Eine Möglichkeit ist das sogenannte **Go-back-N**. Hierbei verwirft der Empfänger alle nachfolgenden Rahmen und sendet für diese keine Bestätigungen. Diese Strategie entspricht einem Empfangsfenster der Größe 1. Mit anderen Worten, die Sicherungsschicht nimmt nur den Rahmen an, den sie als nächsten an die Vermittlungsschicht weitergeben muss. Ist das Sendefenster voll, bevor der Timer des Senders abläuft, beginnt sich die Pipeline zu leeren. Irgendwann läuft der Timer des Senders ab; also überträgt dieser alle unbestätigten Rahmen in der richtigen Reihenfolge noch einmal, wobei er mit dem verlorenen oder zerstörten anfängt. Dieser Ansatz kann einen Großteil der verfügbaren Bandbreite verschwenden, wenn die Fehlerrate hoch ist.

▶ Abbildung 3.18a zeigt den Ansatz Go-back-N für den Fall, dass das Sendefenster groß ist. Die Rahmen 0 und 1 werden korrekt empfangen und bestätigt. Rahmen 2 wird jedoch beschädigt oder geht verloren. Der Sender, dem dies nicht bekannt ist, sendet

weiterhin Rahmen, bis der Timer für Rahmen 2 abläuft. Dann geht er zurück zu Rahmen 2 und beginnt erneut mit dem Senden der Rahmen 2, 3, 4 usw.

Die andere allgemeine Fehlerbehandlungsstrategie beim Pipelining von Rahmen wird als **selektive Wiederholung** (*selective repeat*) bezeichnet. Hier wird ein fehlerhafter Rahmen verworfen, aber die danach erhaltenen fehlerfreien Rahmen werden in einem Puffer abgelegt. Wenn beim Sender die Zeit abgelaufen ist, wird nur der älteste nicht bestätigte Rahmen erneut übertragen. Wenn dieser Rahmen korrekt ankommt, kann der Empfänger in der Folge alle im Puffer gespeicherten Rahmen an die Vermittlungsschicht übertragen. Die selektive Wiederholung entspricht einem Empfangsfenster, das größer als 1 ist. Das kann in der Sicherungsschicht viel Speicherplatz in Anspruch nehmen, falls das Fenster groß ist.

Die selektive Wiederholung wird oftmals damit kombiniert, dass der Empfänger eine negative Bestätigung (NAK, *negative acknowledgement*) sendet, wenn er einen Fehler wie beispielsweise einen Prüfsummenfehler oder einen Rahmen außerhalb der Reihenfolge entdeckt. NAKs stoßen die erneute Übertragung an, bevor der entsprechende Timer abläuft, und verbessern daher die Leistung.

In ► Abbildung 3.18b werden die Rahmen 0 und 1 wieder korrekt empfangen und bestätigt, doch Rahmen 2 geht verloren. Wenn Rahmen 3 beim Empfänger ankommt, bemerkt die Sicherungsschicht, dass ein Rahmen fehlt, und sendet ein NAK für Rahmen 2 zurück, legt aber 3 im Puffer ab. Wenn die Rahmen 4 und 5 ankommen, werden sie ebenfalls von der Sicherungsschicht im Puffer abgelegt, anstatt an die Vermittlungsschicht weitergegeben zu werden. Irgendwann gelangt NAK 2 zum Sender zurück, der sofort Rahmen 2 neu sendet. Wenn dieser ankommt, verfügt die Sicherungsschicht nun über die Rahmen 2, 3, 4 und 5 und kann alle in der korrekten Reihenfolge an die Vermittlungsschicht übergeben. Ebenso können nun alle Rahmen bis einschließlich Rahmen 5 bestätigt werden, wie in der Abbildung dargestellt. Geht das NAK verloren, wartet der Sender auf das Timeout für Rahmen 2 und sendet ihn (und zwar nur ihn) auf eigene Veranlassung noch einmal. Dies kann eine ganze Weile später passieren.

Bei diesen zwei alternativen Ansätzen wird zwischen der effizienten Benutzung der Bandbreite und der Größe des Pufferspeichers der Sicherungsschicht ein Kompromiss eingegangen. Je nachdem, welcher Aspekt wichtiger ist, kann man das eine oder das andere verwenden. ► Abbildung 3.19 zeigt ein Go-back-N-Protokoll, bei dem die empfangende Sicherungsschicht nur Daten in der richtigen Reihenfolge annimmt. Rahmen, die auf einen Fehler folgen, werden verworfen. Bei diesem Protokoll haben wir nun zum ersten Mal die Annahme fallen gelassen, dass die Vermittlungsschicht immer einen unerschöpflichen Vorrat an zu sendenden Paketen hat. Sobald die Vermittlungsschicht ein Paket übertragen möchte, ruft sie das Ereignis *network_layer_ready* auf. Um die Flusskontrollgrenze einzuhalten, die das Sendefenster oder die Anzahl der unbestätigten Rahmen begrenzt, die zu jedem Zeitpunkt ausstehen können, muss die Sicherungsschicht in der Lage sein, ein Übermaß an Arbeit von der Vermittlungsschicht abzuwehren. Hierfür werden die Bibliotheksprozeduren *enable_network_layer* und *disable_network_layer* verwendet.

```

/* Protokoll 5 (Go-back-N) lässt mehrere ausstehende Rahmen zu. Der Sender
 kann bis zu MAX_SEQ Rahmen übertragen, ohne auf eine Bestätigung warten zu
 müssen. Ferner wird (im Gegensatz zu den vorherigen Protokollen) von der
 Vermittlungsschicht nicht angenommen, dass sie jederzeit neue Pakete ver-
 fügbar hat. Stattdessen löst die Vermittlungsschicht immer, wenn sie ein
 Paket zu verschicken hat, ein network_layer_ready-Ereignis aus. */

#define MAX_SEQ 7
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready}
event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Gib true aus, wenn a <= b < c (modulo gerechnet); false sonst */
    if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
        return(true);
    else
        return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
/* Erstelle und sende einen Datenrahmen */
    frame s;                                /* Hilfsvariable */

    s.info = buffer[frame_nr];                /* füge Paket in Rahmen ein */
    s.seq = frame_nr;                        /* füge Sequenznummer Rahmen ein */
    s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1); /* Huckepackbestätigung */
    to_physical_layer(&s);                  /* übertrage den Rahmen */
    start_timer(frame_nr);                  /* starte den Timer */
}

void protocol5(void)
{
    seq_nr next_frame_to_send;               /* MAX_SEQ > 1 für Ausgangstrom */
    seq_nr ack_expected;                   /* ältester noch nicht bestätigter
                                                Rahmen */
    seq_nr frame_expected;                 /* nächster erwarteter Rahmen im
                                                Eingangstrom */
    frame r;                                /* Hilfsvariable */
    packet buffer[MAX_SEQ + 1];            /* Puffer für Ausgangstrom */
    seq_nr nbuffered;                     /* Anzahl der in Gebrauch befind-
                                                lichen Ausgangspuffer */
    seq_nr i;                                /* Index in das Puffer-Array */

    event_type event;

    enable_network_layer();                /* erlaube Ereignisse des Typs
                                                network_layer_ready */

    ack_expected = 0;                      /* nächste erwartete Bestätigung */
    next_frame_to_send = 0;                /* nächster ausgehender Rahmen */
    frame_expected = 0;                   /* Nummer des erwarteten
                                                Eingangsrahmens */
    nbuffered = 0;                        /* anfangs wird kein Paket
                                                zwischengespeichert */
}

```

```

while (true) {
    wait_for_event(&event); /* vier Möglichkeiten: siehe
                           event_type oben */

    switch(event) {
        case network_layer_ready: /* Vermittlungsschicht möchte Paket
                                     senden */
            /* Einen neuen Rahmen annehmen, speichern und übertragen */
            from_network_layer(&buffer[next_frame_to_send]); /* hole neues
                           Paket */
            nbuffered = nbuffered + 1; /* erweitere Sendefenster */
            send_data(next_frame_to_send, frame_expected, buffer); /* über-
                           trage den Rahmen */
            inc(next_frame_to_send); /* inkrementiere oberes Ende des
                           Sendefensters */
            break;

        case frame_arrival: /* ein Daten- oder Steuerrahmen
                           ist angekommen */
            from_physical_layer(&r); /* hole ankommenden Rahmen von der
                           Bitübertragungsschicht */

            if (r.seq == frame_expected) {
                /* Rahmen werden nur in richtiger Reihenfolge angenommen. */
                to_network_layer(&r.info); /* Übergib Paket an
                                              Vermittlungsschicht */
                inc(frame_expected); /* inkrementiere unteres Ende vom
                           Empfangsfenster */
            }

            /* Ack n impliziert n - 1, n - 2 usw. Überprüfe dies. */
            while (between(ack_expected, r.ack, next_frame_to_send)) {
                /* Bearbeiten der Huckepackbestätigung */
                nbuffered = nbuffered - 1; /* ein Rahmen weniger
                                              zwischengespeichert */
                stop_timer(ack_expected); /* Rahmen intakt angekommen;
                                              stoppe Timer */
                inc(ack_expected); /* verkleinere Sendefenster */
            }
            break;

        case cksum_err: break; /* fehlerhafte Rahmen einfach
                           ignorieren */

        case timeout: /* Fehler; alle ausstehenden Rahmen
                      erneut übertragen */
            next_frame_to_send = ack_expected; /* beginne hier mit
                           Neuübertragung */
            for (i = 1; i <= nbuffered; i++) {
                send_data(next_frame_to_send, frame_expected, buffer);
                /* sende Rahmen erneut */
                inc(next_frame_to_send); /* bereite nächsten Rahmen zum
                               Senden vor */
            }
    }
}

```

```

    if (nbuffered < MAX_SEQ)
        enable_network_layer();
    else
        disable_network_layer();
}
}

```

Abbildung 3.19: Schiebefensterprotokoll mit Go-back-N.

Die maximale Anzahl an Rahmen, die zu jedem Zeitpunkt ausstehen können, entspricht nicht der Größe des Sequenznummernbereichs. Bei Go-back-N können MAX_SEQ Rahmen zu jeder Zeit ausstehen, obwohl es für MAX_SEQ+1 unterschiedliche Sequenznummern gibt ($0, 1, 2, \dots, MAX_SEQ$). Für das nächste Protokoll, die selektive Wiederholung, werden wir sogar eine noch stärkere Einschränkung kennenlernen. Um einzusehen, wozu diese Einschränkung nötig ist, betrachten Sie folgenden Ablauf mit $MAX_SEQ=7$:

1. Der Sender schickt die Rahmen 0 bis 7.
2. Eine Huckepackbestätigung für Rahmen 7 kommt zum Sender zurück.
3. Der Sender schickt weitere acht Rahmen, wieder mit den Sequenznummern 0 bis 7.
4. Anschließend geht wieder eine Huckepackbestätigung für Rahmen 7 ein.

Die Frage ist nun: Kamen alle acht Rahmen der zweiten Sendung erfolgreich an oder sind alle acht verloren gegangen (wobei die nach einem Fehler verworfenen als verloren gelten)? In beiden Fällen würde der Empfänger den Rahmen 7 als Bestätigungsrahmen senden. Der Sender kann das nicht wissen. Aus diesem Grund muss die zulässige Höchstzahl an ausstehenden Rahmen auf MAX_SEQ beschränkt werden.

Obwohl Protokoll 5 die nach einem Fehler ankommenden Rahmen nicht zwischenspeichert, kann es den Einsatz von Puffern doch nicht ganz vermeiden. Da ein Sender möglicherweise alle nicht bestätigten Rahmen zu einem späteren Zeitpunkt noch einmal übertragen muss, muss er sie im Speicher behalten, bis er ganz sicher weiß, dass die Rahmen vom Empfänger angenommen worden sind. Sobald die Bestätigung für n eingeht, gelten die Rahmen $n-1, n-2$ usw. automatisch ebenfalls als bestätigt. Diese Art der Bestätigung wird **kumulative Bestätigung** genannt. Diese Eigenschaft ist besonders wichtig, wenn einige der früheren Rahmen mit Bestätigungen verloren gegangen sind. Jedes Mal, wenn eine Bestätigung ankommt, prüft die Sicherungsschicht, ob irgendein Puffer geleert werden kann. Ist das der Fall (d.h., im Fenster ist etwas Platz frei), erhält die bis dahin blockierte Vermittlungsschicht die Erlaubnis, erneut *network_layer_ready*-Ereignisse auszulösen.

Bei diesem Protokoll gehen wir davon aus, dass es immer Rückverkehr gibt, auf dem die Bestätigung im Huckepackverfahren gesendet werden kann. Bei Protokoll 4 ist diese Annahme nicht erforderlich, da es für jeden erhaltenen Rahmen einen Rahmen zurücksendet, selbst wenn es einen Rahmen gesendet hat. Im nächsten Protokoll lösen wir das Problem des Einbahnverkehrs auf elegante Weise.

Da bei Protokoll 5 mehrere Rahmen unterwegs sein können, werden auch mehrere Timer (einer für jeden ausstehenden Rahmen) benötigt. Für jeden Rahmen läuft die Zeit unabhängig von allen anderen ab. Man kann alle diese Timer jedoch ganz einfach softwareseitig simulieren, indem man einen Hardwaretaktgeber verwendet, der periodisch Interrupts auslöst. Die ausstehenden Timeouts bilden eine nach der Ablaufzeit des Timers geordnete verkettete Liste. Jeder Knoten der Liste enthält die Information, wie viele Takte vom Ablauf des vorherigen Timers bis zu diesem Timer noch gewartet werden muss und um welchen Rahmen es sich handelt. Außerdem ist ein Zeiger auf den nächsten Knoten vorhanden.

Ein Beispiel für die Implementierung von Timern enthält ►Abbildung 3.20a. Nehmen wir an, dass genau alle 1 ms ein Taktsignal erfolgt. Anfangs ist die absolute Zeit 10:00:00,000 und es stehen noch drei Timeouts aus – bei 10:00:00,005, 10:00:00,013 und 10:00:00,019. Bei jedem Takt der Hardwareuhr wird die Realzeit nachgestellt und der Taktzähler am Anfang der Liste um eins verringert. Sobald der Taktzähler auf null steht, wird ein Timeout veranlasst und der Knoten aus der Liste entfernt (siehe ►Abbildung 3.20b). Obwohl bei diesem Verfahren die Liste durchsucht werden muss, wenn *start_timer* oder *stop_timer* aufgerufen wird, ist doch der Aufwand pro Takt insgesamt relativ gering. Bei Protokoll 5 haben beide Routinen Parameter erhalten, die angeben, für welchen Rahmen sie aufgerufen werden.

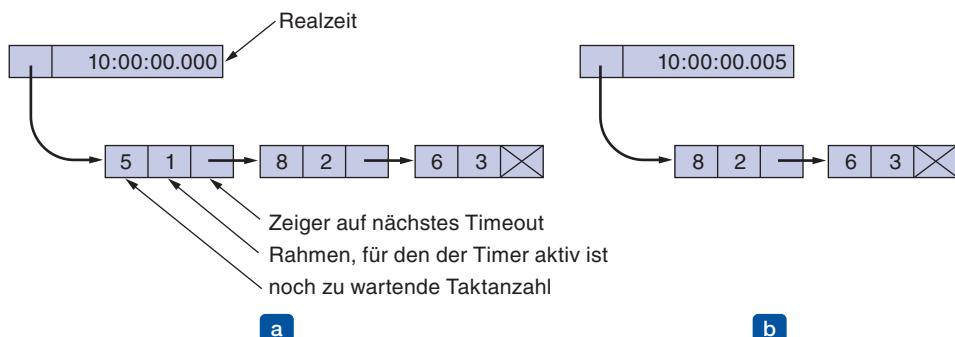


Abbildung 3.20: Softwareseitige Simulation mehrerer Timer. (a) Die Warteschlange der Timeouts. (b) Situation, nachdem das erste Timeout abgelaufen ist.

3.4.3 Protokoll mit selektiver Wiederholung

Das Go-back-N-Protokoll funktioniert gut, solange wenig Fehler auftreten. Ist die Leitung jedoch schlecht, dann wird viel Bandbreite dafür verschwendet, dass Rahmen wiederholt übertragen werden müssen. Eine alternative Strategie – die selektive Wiederholung – erlaubt es dem Empfänger, die Rahmen, die auf einen zerstörten oder verlorenen Rahmen folgen, anzunehmen und zwischenzuspeichern.

```

/* Protokoll 6 (selektive Wiederholung) nimmt Rahmen ungeordnet an, gibt die
Pakete aber geordnet an die Vermittlungsschicht weiter. Zu jedem ausstehenden
Rahmen gehört ein Timer. Wenn der Timer abläuft, dann wird nur dieser
Rahmen erneut übertragen, nicht alle ausstehenden Rahmen wie bei Protokoll 5. */

#define MAX_SEQ 7           /* sollte 2^n-1 sein */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready,
ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;          /* es wurde noch kein NAK gesendet */
seq_nr oldest_frame = MAX_SEQ + 1; /* der Anfangswert ist nur für den
Simulator */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Wie bei Protokoll 5, aber kürzer und etwas obskurer. */
    return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected,
packet buffer[])
{
/* Erstelle und sende Daten-, Bestätigungs- oder NAK-Rahmen. */
    frame s;                  /* Hilfsvariable */

    s.kind = fk;                /* Möglichkeiten: data, ack oder
                                nak */
    if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
    s.seq = frame_nr;           /* nur für Datenrahmen von Bedeutung */
    s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
    if (fk == nak) no_nak = false; /* ein NAK pro Rahmen, bitte */
    to_physical_layer(&s);      /* übertrage den Rahmen */
    if (fk == data) start_timer(frame_nr % NR_BUFS);
    stop_ack_timer();           /* separater Bestätigungsrahmen nicht
                                nötig */
}

void protocol6(void)
{
    seq_nr ack_expected;        /* unteres Ende des Sendefensters */
    seq_nr next_frame_to_send;   /* oberes Ende des Sendefensters + 1 */
    seq_nr frame_expected;       /* unteres Ende des Empfangsfensters */
    seq_nr too_far;             /* oberes Ende des Empfangsfensters + 1 */
    int i;                      /* Index in Puffer */
    frame r;                   /* Hilfsvariable */
    packet out_buf[NR_BUFS];     /* Puffer für Ausgangsstrom */
    packet in_buf[NR_BUFS];      /* Puffer für Eingangsstrom */
    boolean arrived[NR_BUFS];    /* eingehendes Bitmap */
    seq_nr nbuffered;           /* Zahl der aktuell verwendeten
                                Ausgangspuffer */
    event_type event;

    enable_network_layer();      /* initialisieren */
}

```

```

ack_expected = 0;                      /* nächste erwartete Bestätigung im
                                         Eingangsstrom */
next_frame_to_send = 0;                 /* Nummer des nächsten Ausgangsrahmens */
frame_expected = 0;
too_far = NR_BUFS;
nbuffered = 0;                         /* anfangs sind keine Pakete
                                         zwischengespeichert */
for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
while (true) {
    wait_for_event(&event);           /* fünf Möglichkeiten: siehe event_type
                                         oben */
    switch(event) {
        case network_layer_ready:    /* akzeptiere, speichere und übertrage
                                         neuen Rahmen */
            nbuffered = nbuffered + 1; /* erweitere das Fenster */
            from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]);
                                         /* hole neues Paket */
            send_frame(data, next_frame_to_send, frame_expected, out_buf);
                                         /* übertrage den Rahmen */
            inc(next_frame_to_send);   /* inkrementiere oberes Fensterende */
            break;

        case frame_arrival:          /* ein Daten- oder Steuerrahmen ist
                                         angekommen */
            from_physical_layer(&r); /* hole Eingangsrahmen von Bitüber-
                                         tragungsschicht */
            if (r.kind == data) {
                /* Ein unbeschädigter Rahmen ist angekommen. */
                if ((r.seq != frame_expected) && no_nak)
                    send_frame(nak, 0, frame_expected, out_buf); else
                        start_ack_timer();
                if (between(frame_expected, r.seq, too_far) &&
                    (arrived[r.seq % NR_BUFS] == false)) {
                    /* Rahmen können in jeder Reihenfolge angenommen
                     werden. */
                    arrived[r.seq % NR_BUFS] = true; /* markiere Puffer als voll */
                    in_buf[r.seq % NR_BUFS] = r.info; /* füge Daten in Puffer ein */
                    while (arrived[frame_expected % NR_BUFS]) {
                        /* Übergib Rahmen und schiebe das Fenster eine Einheit
                         weiter. */
                        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                        no_nak = true;
                        arrived[frame_expected % NR_BUFS] = false;
                        inc(frame_expected); /* inkrementiere unteres Ende
                                         des Empfangsfensters */
                        inc(too_far);       /* inkrementiere oberes Ende des
                                         Empfangsfensters */
                        start_ack_timer(); /* um zu prüfen, ob eine eigene
                                         Bestätigung nötig ist */
                    }
                }
            }
            if ((r.kind == nak) && between(ack_expected, (r.ack+1)%(MAX_SEQ+1),
                                              next_frame_to_send))
                send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected,
                           out_buf);
    }
}

```

```

        while (between(ack_expected, r.ack, next_frame_to_send)) {
            nbuffered = nbuffered - 1; /* bearbeite Huckepackbestätigung */
            stop_timer(ack_expected % NR_BUFS); /* Rahmen intakt
                                                angekommen */
            inc(ack_expected); /* inkrementiere untere Ecke des
                                Sendefensters */
        }
        break;

    case cksum_err:
        if (no_nak) send_frame(nak, 0, frame_expected, out_buf);
                    /* Rahmen beschädigt */
        break;

    case timeout:
        send_frame(data, oldest_frame, frame_expected, out_buf);
                    /* Zeit abgelaufen */
        break;

    case ack_timeout:
        send_frame(ack, 0, frame_expected, out_buf); /* ack-Timer abgelaufen;
                                                       sende Bestätigung */
    }

    if (nbuffered < NR_BUFS) enable_network_layer(); else
        disable_network_layer();
}
}

```

Abbildung 3.21: Schiebefensterprotokoll mit selektiver Wiederholung.

Bei diesem Protokoll haben sowohl der Sender als auch der Empfänger ein Fenster mit ausstehenden bzw. zulässigen Sequenznummern. Zuerst hat das Sendefenster die Größe 0 und kann bis zu einem vorgegebenen Maximum anwachsen. Das Empfangsfenster hat andererseits immer eine festgelegte Größe, die dem vorgegebenen Maximum entspricht. Der Empfänger reserviert für jede Sequenznummer in seinem Fenster einen Puffer. Für jeden Puffer gibt es ein Bit (*arrived*), das angibt, ob der Puffer voll oder leer ist. Sobald ein Rahmen ankommt, wird über seine Sequenznummer mithilfe der Funktion *between* geprüft, ob er in das Fenster passt. Ist das der Fall und wurde er nicht schon einmal übertragen, dann wird er angenommen und gespeichert. Hierbei wird nicht darauf geachtet, ob der Rahmen das nächste von der Vermittlungsschicht erwartete Paket enthält oder nicht. Natürlich muss der Rahmen in der Sicherungsschicht verbleiben und darf nicht an die Vermittlungsschicht weitergegeben werden, bis alle Rahmen mit niedrigerer Sequenznummer in der richtigen Reihenfolge an die Vermittlungsschicht weitergegeben wurden. ► Abbildung 3.21 zeigt ein Protokoll, bei dem dieser Algorithmus angewendet wird.

Das Empfangen ohne Beachtung der Reihenfolge erzeugt weitere Beschränkungen auf Rahmensequenznummern im Vergleich mit Protokollen, die Rahmen nur in der richtigen Reihenfolge akzeptieren. Solche Problemfälle können am besten anhand eines Beispiels aufgezeigt werden. Nehmen wir den Fall an, dass die Sequenznummer aus 3 Bits besteht, sodass der Sender bis zu sieben Rahmen versenden kann, bevor er auf

Bestätigungen warten muss. Zunächst sehen die Fenster des Senders und Empfängers so aus wie in ► Abbildung 3.22a. Der Sender schickt zuerst Rahmen 0 bis 6 ab. Das Empfangsfenster lässt den Eingang aller Rahmen mit den Sequenznummern von 0 bis einschließlich 6 zu. Alle sieben Rahmen kommen korrekt an; daher sendet der Empfänger die Bestätigungen ab und schiebt sein Fenster so weiter, dass wieder die Rahmen 7, 0, 1, 2, 3, 4 oder 5 angenommen werden können (siehe ► Abbildung 3.22b). Alle sieben Puffer werden als leer gekennzeichnet.

Genau jetzt kommt es zu einem Unglück: Der Blitz schlägt in einen Telefonmast ein und alle Bestätigungen werden ausgelöscht. Das Protokoll sollte trotz dieses Unglücks korrekt arbeiten. Der Timer des Senders läuft nach einer bestimmten Zeit ab und der Sender schickt Rahmen 0 noch einmal. Sobald dieser Rahmen beim Empfänger ankommt, wird geprüft, ob der Rahmen in das Empfangsfenster fällt. Leider ist er im neuen Fenster enthalten (siehe Abbildung 3.22b) und wird deshalb als ein neuer Rahmen angenommen. Der Empfänger sendet eine (Huckepack-)Bestätigung für Rahmen 6, da 0 bis 6 angekommen sind.

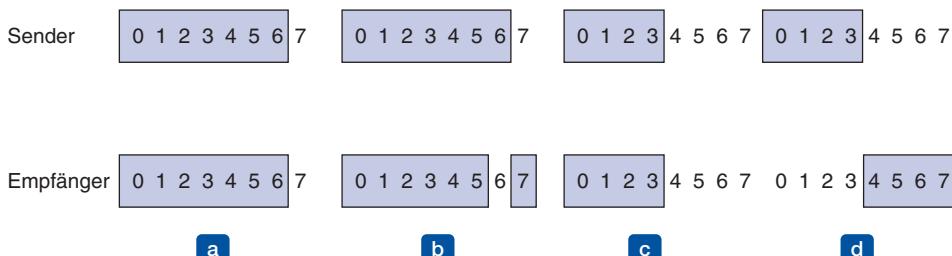


Abbildung 3.22: (a) Ausgangssituation mit Fenster der Größe 7. (b) Nach der Übertragung und dem Empfang von sieben Rahmen ohne Bestätigung. (c) Ausgangssituation mit Fenster der Größe 4. (d) Nach der Übertragung und dem Empfang von vier Rahmen ohne Bestätigung.

Der Sender freut sich zu hören, dass alle seine Rahmen korrekt angekommen sind, und schiebt sein Fenster weiter. Er schickt dann sofort Rahmen 7, 0, 1, 2, 3, 4 und 5. Rahmen 7 wird vom Empfänger angenommen und sein Paket wird direkt an die Vermittlungsschicht weitergegeben. Gleich danach prüft die empfangende Sicherungsschicht, ob sie schon einen gültigen Rahmen 0 erhalten hat, und stellt fest, dass das der Fall ist. Das (alte zwischengespeicherte) Paket wird daraufhin so an die Vermittlungsschicht weitergegeben, als wäre es ein neues Paket. Folglich erhält die Vermittlungsschicht ein falsches Paket und das Protokoll hat einen Fehler erzeugt.

Der Kern des Problems liegt darin, dass sich die neue Reihe der gültigen Sequenznummern mit der alten überschneidet, nachdem der Empfänger sein Fenster weitergeschoben hat. Die nächste Menge Rahmen kann nun entweder aus Duplikaten bestehen (wenn alle Bestätigungen verloren gegangen sind) oder aus neu gesendeten Rahmen (wenn alle Bestätigungen angekommen sind). Der bedauernswerte Empfänger hat keine Möglichkeit, die zwei Fälle auseinanderzuhalten.

Ein Ausweg aus diesem Dilemma ist, dafür Sorge zu tragen, dass es keine Überschneidung mit dem Originalfenster gibt, nachdem der Empfänger sein Fenster weiterge-

schoben hat. Um dies zu gewährleisten, darf das Fenster höchstens so groß sein, dass die Hälfte der Sequenznummern hineinpasst. Diese Situation ist in ▶ Abbildung 3.22c und ▶ Abbildung 3.22d gezeigt. Bei drei Bits wird ein Bereich von 0 bis 7 abgedeckt. Es sollten immer maximal vier unbestätigte Rahmen ausstehen. So kann der Empfänger, wenn er gerade Rahmen 0 bis 3 angenommen und sein Fenster für den Empfang der Rahmen 4 bis 7 weitergeschoben hat, eindeutig sagen, ob die nächsten empfangenen Rahmen noch einmal gesendet wurden (0 bis 3) oder neu sind (4 bis 7). Im Allgemeinen haben wir bei Protokoll 6 eine Fenstergröße von $(MAX_SEQ+1)/2$.

Nun stellt sich die interessante Frage, wie viele Puffer der Empfänger haben muss. Er wird unter keinen Umständen Rahmen annehmen, deren Sequenznummern nicht zwischen der Unter- und Obergrenze des Fensters liegen. Folglich ist die benötigte Zahl von Puffern gleich der Fenstergröße und nicht gleich der Zahl der Sequenznummern. In dem obigen Beispiel mit 3-Bit-Sequenznummern brauchen wir also vier Puffer, die wir mit 0 bis 3 nummerieren. Wenn Rahmen i ankommt, wird er in Puffer $i \bmod 8$ gespeichert. Beachten Sie, dass i und $(i+8) \bmod 8$ niemals zur gleichen Zeit im Fenster sein können, obwohl sie um den gleichen Puffer „kämpfen“, weil dafür eine Fenstergröße von mindestens 5 nötig wäre.

Aus dem gleichen Grund ist die Zahl der benötigten Timer gleich der Zahl der Puffer, nicht gleich der Größe des Sequenznummernbereichs. Jedem Puffer wird je ein Timer zugeordnet. Läuft der Timer ab, wird der Inhalt des Puffers noch einmal gesendet.

Protokoll 6 lockert außerdem die implizite Annahme, dass die Leitung stark ausgelastet ist. Wir hatten diese Annahme in Protokoll 5 aufgenommen, als wir uns auf Rahmen bezogen, die in umgekehrter Richtung gesendet werden und dabei die Huckepack-bestätigungen mitnehmen. Gibt es nur wenig Rückverkehr, so könnte die Bestätigung lange aufgehalten werden, was Probleme verursachen kann. Im Extremfall ist der Verkehr in eine Richtung sehr stark und in die andere sehr gering, dann blockiert das Protokoll, wenn das Sendefenster sein Maximum erreicht.

Um diese Annahme abzuschwächen, wird ein Hilfstimer durch *start_ack_timer* nach dem Empfang des in der richtigen Reihenfolge nächsten Rahmens gestartet. Kam es vor dem Ablauen des Timers zu keinem Rückverkehr, so wird eine separate Bestätigung gesendet. Eine vom Hilfstimer hervorgerufene Unterbrechung wird *ack_timeout* genannt. Mit dieser Methode ist Datenfluss nur in eine Richtung möglich, weil ein eventueller Mangel an zurückzusendenden Datenrahmen, die Bestätigungen huckepack mitnehmen können, kein Problem mehr darstellt. Es gibt nur einen einzigen Hilfstimer und wenn *start_ack_timer* aufgerufen wird, während der Timer läuft, hat dies keine Auswirkung. Der Timer wird nicht zurückgesetzt oder verlängert, da es seine Aufgabe ist, eine minimale Rate an Bestätigungen zur Verfügung zu stellen.

Das Timeout des Hilfstimers muss erheblich kürzer sein als das Timeout, welches zur zeitlichen Begrenzung der Datenrahmen verwendet wird. Diese Bedingung ist erforderlich um sicherzustellen, dass ein korrekt empfangener Rahmen früh genug bestätigt wird, sodass der Timer für die erneute Übertragung noch nicht abgelaufen ist und der Rahmen erneut gesendet wird.

Protokoll 6 hat eine effizientere Fehlerbehandlungsmethode als Protokoll 5. Sobald der Empfänger Grund zu der Annahme hat, dass ein Fehler aufgetreten ist, sendet er einen negativen Bestätigungsrahmen (NAK) an den Sender. Dieser Rahmen ist eine Aufforderung zur erneuten Übertragung des im NAK angegebenen Rahmens. In zwei Fällen sollte der Empfänger misstrauisch werden: wenn entweder ein zerstörter oder ein anderer Rahmen als der erwartete angekommen ist (ein Rahmen ist möglicherweise verloren gegangen). Um zu vermeiden, dass die Neuübertragung des gleichen (verlorenen) Rahmens immer wieder angefordert wird, sollte der Empfänger genau überwachen, ob schon ein NAK für einen bestimmten Rahmen gesendet wurde. Die Variable *no_nak* in Protokoll 6 hat den Wert *true*, wenn kein NAK für *frame_expected* angekommen ist. Wird NAK beschädigt oder geht verloren, ist das nicht so schlimm, weil das Timeout für den Sender schließlich abläuft und der Sender den fehlenden Rahmen auf jeden Fall erneut übertragen wird. Kommt der fehlerhafte Rahmen nach einem beschädigten oder verlorenen NAK an, hat *no_nak* den Wert *true*. Demzufolge wird der Hilfstimer gestartet. Läuft er ab, wird ein ACK gesendet, um den Sender mit dem aktuellen Status des Empfängers zu synchronisieren.

In einigen Fällen ist die Zeit, die der Rahmen braucht, um an sein Ziel zu gelangen und dort verarbeitet zu werden, sowie die Rückübertragung der Bestätigung (fast) konstant. In solchen Fällen kann der Sender seinen Timer „knapp“, das heißt genau so einstellen, dass er ein bisschen länger läuft, als Zeit benötigt wird, um den Rahmen zu senden und die Bestätigung abzuwarten. In diesem Fall sind NAKs nicht sinnvoll.

In anderen Situationen kann die Zeit jedoch sehr variabel gehandhabt werden. Zum Beispiel ist bei sporadischem Rückverkehr die Zeitspanne bis zum Erhalt der Bestätigungen kürzer, wenn es Rückverkehr gibt, und länger, wenn es keinen Rückverkehr gibt. Der Sender steht vor der Entscheidung, entweder die Zeitspanne auf einen kleinen Wert festzusetzen (und dadurch unnötige Neuübertragungen zu riskieren) oder einen großen Wert zu wählen (und damit eventuell nach einem Fehler für eine längere Zeit unbeschäftigt zu sein). Beide Optionen verschwenden Bandbreite. Grundsätzlich gilt: Ist die Standardabweichung des Rückmeldeintervalls groß im Vergleich zum Intervall selbst, dann wird der Timer vorsichtshalber großzügig eingestellt. NAKs können die Neuübertragungen von verlorenen oder zerstörten Rahmen stark beschleunigen.

In engem Zusammenhang mit Timeouts und NAKs steht die Frage, wie festgestellt werden kann, welcher Rahmen das Timeout hervorgerufen hat. In Protokoll 5 ist das *ack_expected*, weil es immer der älteste Rahmen ist. In Protokoll 6 ist es nicht so einfach festzustellen, wer das Timeout ausgelöst hat. Nehmen wir an, Rahmen 0 bis 4 sind übertragen worden, was bedeutet, dass die Liste der ausstehenden Rahmen 01234 lautet, angefangen bei dem am längsten ausstehenden Rahmen. Jetzt ist die Zeitspanne für 0 abgelaufen, 5 (ein neuer Rahmen) wird übertragen, für 1 läuft die Zeit ab, für 2 auch, und 6 (ebenfalls ein neuer Rahmen) wird gesendet. Zu diesem Zeitpunkt lautet die Liste der noch ausstehenden Rahmen (vom ältesten zum jüngsten) 3 405 126. Wenn eine Zeitlang überhaupt kein Eingangsverkehr fließt (beispielsweise Rahmen mit Bestätigung), laufen die Timer für die sieben ausstehenden Rahmen in genau dieser Reihenfolge ab.

Um das Beispiel nicht noch komplizierter zu machen, wurde die Verwaltung des Timers weggelassen. Stattdessen gehen wir davon aus, dass die Variable *oldest_frame* beim Auftreten eines Timeouts gesetzt wird um anzugeben, für welchen Rahmen die Zeit abgelaufen ist.

3.5 Beispiele für Protokolle der Sicherungsschicht

Innerhalb eines einzelnen Gebäudes werden häufig LANs zur Vernetzung eingesetzt, doch eine WAN-Infrastruktur wird meistens aus Punkt-zu-Punkt-Leitungen aufgebaut. In *Kapitel 4* werden wir uns mit LANs beschäftigen. Hier wollen wir die Sicherungsschichtprotokolle untersuchen, die auf Punkt-zu-Punkt-Leitungen im Internet in zwei typischen Situationen angetroffen werden. Die erste Situation entsteht beim Senden von Paketen über SONET-Glasfaserverbindungen in WANs. Diese Verbindungen werden häufig eingesetzt, um zum Beispiel Router in den verschiedenen Standorten eines ISP-Netzes anzuschließen.

Die zweite Situation betrifft ADSL-Verbindungen, die auf der Teilnehmeranschlussleitung des Telefonnetzes am Rand des Internets verwendet werden. Diese Verbindungen verbinden Millionen von Einzelpersonen und Unternehmen mit dem Internet.

Das Internet benötigt Punkt-zu-Punkt-Verbindungen für diese Nutzungsarten, ebenso wie Einwahlmodems, Standleitungen, Kabelmodems usw. Das Standardprotokoll **PPP** (*Point-to-Point Protocol*) wird benutzt, um Pakete über diese Verbindungen zu schicken. PPP wird in RFC 1661 definiert und in RFC 1662 und anderen RFCs weiter ausgearbeitet (Simpson, 1994a, 1994b). SONET- und ADSL-Verbindungen wenden beide PPP an, jedoch auf unterschiedliche Weisen.

3.5.1 Packet over SONET

SONET (siehe Abschnitt 2.6.4) ist das Protokoll der Bitübertragungsschicht, das am häufigsten auf den WAN-Glasfaserverbindungen benutzt wird, die das Backbone von Kommunikationsnetzen, darunter das des Telefonsystems, bilden. SONET stellt einen Bitstrom zu einer genau festgelegten Rate bereit, zum Beispiel 2,4 Gbit/s für eine OC-48-Verbindung. Dieser Bitstrom ist aus Nutzdaten fester Bytegröße aufgebaut, die nach jeweils 125 µs wiederholt werden, unabhängig davon, ob Benutzerdaten zu senden sind oder nicht.

Die Übertragung von Paketen über diese Verbindungen setzt einen Mechanismus voraus, der es ermöglicht, die vereinzelt auftretenden Pakete von dem kontinuierlichen Bitstrom zu unterscheiden, in dem die Pakete transportiert werden. Auf IP-Routern wird PPP eingesetzt, um diesen Mechanismus zur Verfügung zu stellen (► Abbildung 3.23).

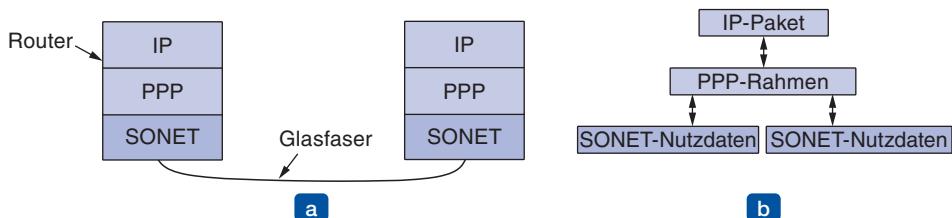


Abbildung 3.23: Packet over SONET. (a) Ein Protokollstapel. (b) Beziehungen zwischen Rahmen.

PPP ist die Verbesserung eines früheren, einfacheren Protokolls namens **SLIP** (*Serial Line Internet Protocol*). Es wird zur Verbindungskonfiguration bei der Fehlererkennung benutzt, unterstützt mehrere Protokolle, ermöglicht Authentifizierung und einiges mehr. PPP bietet eine große Bandbreite an Optionen und zeichnet sich durch drei Hauptmerkmale aus:

1. Eine Rahmenbildungsmethode, die das Ende eines Rahmens und den Anfang des nächsten eindeutig kennzeichnet. Das Rahmenformat übernimmt auch die Fehlererkennung.
2. Ein Verbindungssteuerungsprotokoll zum Aktivieren und Testen von Leitungen, Verhandeln von Optionen und Beenden von Verbindungen, wenn sie nicht mehr gebraucht werden. Dieses Protokoll heißt **LCP** (*Link Control Protocol*).
3. Eine Methode zum Aushandeln von Optionen auf der Vermittlungsschicht, die von dem auf der Vermittlungsschicht benutzten Protokoll unabhängig sind. Die gewählte Methode sieht vor, dass auf jeder unterstützten Vermittlungsschicht ein anderes **NCP** (*Network Control Protocol*) läuft.

Das PPP-Rahmenformat ist dem Rahmenformat **HDLC** (*High-level Data Link Control*) sehr ähnlich, einem häufig genutzten Vertreter einer frühen Protokollfamilie, weil keine Notwendigkeit bestand, das Rad neu zu erfinden.

Der wesentliche Unterschied zwischen PPP und HDLC liegt darin, dass PPP byteorientiert und HDLC bitorientiert ist. Insbesondere nutzt PPP das Bytestopfen und alle Rahmen enthalten eine ganzzahlige Anzahl an Bytes. HDLC verwendet Bitstopfen, sodass Rahmen mit beispielsweise 30,25 Byte möglich sind.

In der Praxis gibt es jedoch einen zweiten großen Unterschied. HDLC stellt zuverlässige Übertragungen mit einem Schiebefenster, Bestätigungen und Timeouts in der Art bereit, wie wir es untersucht haben. Auch PPP kann zuverlässige Übertragungen in verrauschten Umgebungen wie drahtlosen Netzen bieten; die Details sind in RFC 1663 definiert. In der Praxis wird dies jedoch selten umgesetzt. Stattdessen wird im Internet fast immer ein „unnummerierter Modus“ benutzt, um verbindungslosen unbestätigten Dienst zu liefern.

Das PPP-Rahmenformat ist in ▶ Abbildung 3.24 dargestellt. Alle PPP-Rahmen beginnen mit dem Flagbyte des HDLC-Standards von 0xE (0111110). Das Flagbyte wird gestopft, wenn es innerhalb des *Payload*-Felds auftritt, indem das Escape-Byte 0x7D benutzt wird. Das folgende Byte ist das Escape-Byte, das mit 0x20 XOR-verknüpft

wurde, wodurch das 5. Bit gekippt wird. Beispielsweise ist 0x7D 0x5E die Escape-Folge für das Flagbyte 0x7E. Dies bedeutet, nach Rahmenanfang und -ende kann gesucht werden, indem einfach nach dem Byte 0x7E gescannt wird, da dies nirgendwo sonst vorkommt. Die Regel zum Entstopfen beim Empfang eines Rahmens sieht vor, nach 0x7D zu suchen, dieses löschen und das darauffolgende Byte mit 0x20 XOR-verknüpfen. Auch hier wird nur ein Flagbyte zwischen zwei Rahmen benötigt. Mehrere Flagbytes können zum Auffüllen benutzt werden, wenn keine Rahmen zum Senden vorhanden sind.

Nach dem Start-des-Rahmens-Flag kommt das Feld *Address*. Dieses Feld wird immer auf den Binärwert 11111111 gesetzt, um anzugeben, dass alle Stationen den Rahmen akzeptieren sollen. Durch diesen Wert wird vermieden, dass Verbindungsadressen zugewiesen werden müssen.

| Bytes | 1 | 1 | 1 | 1 oder 2 | variabel | 2 oder 4 | 1 |
|-------|------------------|---------------------|---------------------|----------|----------|----------|------------------|
| | Flag 01111110 | Address 11111111 | Control 00000011 | Protocol | Payload | Checksum | Flag 01111110 |

Abbildung 3.24: Format eines vollständigen PPP-Rahmens für den Betrieb im unnummerierten Modus.

Dem *Address*-Feld folgt das Feld *Control*, dessen Standardwert 00000011 ist. Dieser Wert zeigt einen unnummerierten Rahmen an.

Da die Felder *Address* und *Control* in der Standardkonfiguration immer konstant sind, stellt LCP den nötigen Mechanismus bereit, damit zwei Parteien eine Option aushandeln können, mit der beide Felder weggelassen und dadurch zwei Byte pro Rahmen eingespart werden.

Das vierte PPP-Feld ist *Protocol*. Dies gibt an, welche Paketart sich im Feld *Payload* befindet. Codes, die mit einem 0-Bit beginnen, werden für IP-Version 4, IP-Version 6 und andere Protokolle der Vermittlungsschicht definiert, die eventuell genutzt werden, wie IPX und AppleTalk. Codes, die mit einem 1-Bit beginnen, werden für PPP-Konfigurationsprotokolle verwendet. Hierzu zählen LCP und ein weiteres NCP für jedes unterstützte Protokoll der Vermittlungsschicht. Die Standardgröße des *Protocol*-Feldes ist 2 Byte, über LCP kann aber auch 1 Byte vereinbart werden. Die Entwickler waren vermutlich etwas übervorsichtig mit ihrer Einschätzung, dass eines Tages mehr als 256 Protokolle im Einsatz sein könnten.

Das Feld *Payload* hat eine variable Länge bis zu einem vereinbarten Maximum. Wird die Länge nicht beim Verbindungsaufbau über LCP vereinbart, dann wird eine Standardlänge von 1 500 Byte verwendet. Die Nutzdaten können bei Bedarf auch aufgefüllt werden (Padding).

Nach diesem Feld kommt das Feld *Checksum* (Prüfsumme), das normalerweise 2 Byte groß ist. Alternativ kann eine Prüfsumme von 4 Byte vereinbart werden. Die 4-Byte-Prüfsumme entspricht in der Tat dem 32-Bit-CRC-Wert, dessen Generatorpolynom wir am Ende von Abschnitt 3.2.2 angegeben haben. Die 2-Byte-Prüfsumme ist ebenfalls ein Standard-CRC.

PPP ist ein Rahmenmechanismus zur Übertragung von Paketen mehrerer Protokolle über viele Arten von Bitübertragungsschichten. Wenn PPP über SONET verwendet werden soll, dann werden die dabei zu treffenden Entscheidungen in RFC 2615 vorgegeben (Malis und Simpson, 1999). Eine 4-Byte-Prüfsumme wird als Hauptinstrument zum Entdecken von Übertragungsfehlern über den Bitübertragungs-, Sicherungs- und Vermittlungsschichten benutzt. Es wird empfohlen, die *Address*-, *Control*- und *Protocol*-Felder nicht zu komprimieren, da SONET-Verbindungen immer mit relativ hohen Raten laufen.

PPP hat auch noch ein ungewöhnliches Merkmal. Auf die PPP-Nutzdaten wird Scrambling angewandt (siehe Abschnitt 2.5.1), bevor diese in die SONET-Nutzdaten einge-fügt werden. Scrambling verknüpft vor der Übertragung die Nutzdaten mit einer lan-gen pseudozufälligen Folge über XOR. Das Problem dabei ist, dass der SONET-Bitstrom häufige Bitübergänge zur Synchronisation benötigt. Diese Übergänge treten bei Sprachsignalen durch Variationen ganz natürlich auf, doch bei der Datenkommu-nikation wählt der Benutzer die zu sendende Information aus und überträgt eventuell ein Paket mit einer langen Folge von Nullen. Durch Scrambling verringert sich die Wahrscheinlichkeit enorm, dass Probleme entstehen, wenn ein Benutzer eine lange Folge von Nullen sendet.

Bevor PPP-Rahmen über SONET-Leitungen übertragen werden können, muss die PPP-Verbindung aufgebaut und konfiguriert werden. Die Phasen, die eine Verbindung vom Aufbau über Benutzung bis zum Trennen durchläuft, ist in ► Abbildung 3.25 gezeigt.

Die Verbindung beginnt im *DEAD*-Zustand, was bedeutet, dass keine Verbindung auf der Bitübertragungsschicht vorhanden ist. Wenn eine Verbindung auf der Bitübertragungsschicht aufgebaut wird, wechselt die Verbindung zum Zustand *ESTABLISH*. An diesem Punkt tauschen die PPP-Peers eine Reihe von LCP-Paketen aus, die jeweils im *Payload*-Feld eines PPP-Rahmens übertragen werden, um die PPP-Optionen für die Verbindung aus den oben erwähnten Möglichkeiten zu wählen. Der initiiierende Peer schlägt Optionen vor und der antwortende Peer akzeptiert diese oder weist sie – voll-ständig oder teilweise – zurück. Er kann auch alternative Vorschläge unterbreiten.

Wenn die Verhandlung der LCP-Option erfolgreich ist, erreicht die Verbindung den *AUTHENTICATE*-Zustand. Nun können die zwei Parteien auf Wunsch gegenseitig ihre Identitäten prüfen. Falls die Authentifizierung erfolgreich ist, wird der *NET-WORK*-Zustand betreten und zur Konfiguration der Vermittlungsschicht wird eine Reihe von NCP-Paketen gesendet. Man kann kaum Verallgemeinerungen zu NCP-Pro-tokollen treffen, weil jedes auf ein bestimmtes Protokoll der Vermittlungsschicht aus-gelegt ist und Konfigurationsanfragen zulässt, die spezifisch für dieses Protokoll sind. Für IP ist zum Beispiel die Zuweisung von IP-Adressen zu beiden Verbindungsenden die wichtigste Möglichkeit.

Sobald *OPEN* erreicht wird, kann die Datenübertragung beginnen. In diesem Zustand werden IP-Pakete in PPP-Rahmen über die SONET-Verbindung übertragen. Ist die Datenübertragung beendet, wechselt die Verbindung in den *TERMINATE*-Zustand und geht von dort wieder in den *DEAD*-Zustand zurück, wenn die Verbindung der Bitüber-tragungsschicht freigegeben wird.

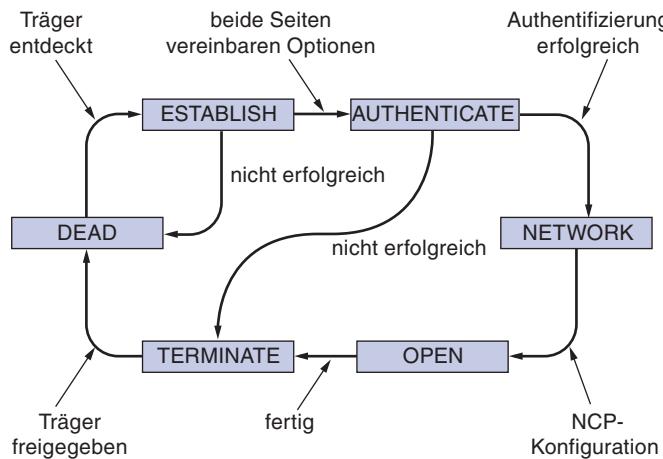


Abbildung 3.25: Zustandsdiagramm zum Auf- und Abbau von PPP-Verbindungen.

3.5.2 ADSL

ADSL verbindet Millionen von Privathaushalten mit dem Internet mit Datenraten von mehreren Megabit/Sekunde über dieselbe Telefonleitung, die für den guten alten Telefondienst benutzt wird. In Abschnitt 2.5.3 haben wir beschrieben, wie ein DSL-Modem auf der Teilnehmeranschlussseite angeschlossen wird. Das Modem sendet Bits über die Anschlussleitung zu einem Gerät namens DSLAM (*DSL Access Multiplexer*), ausgesprochen „di-släm“, das sich im lokalen Büro des Telefonunternehmens befindet. Nun werden wir ausführlicher untersuchen, wie Pakete über ADSL-Verbindungen übertragen werden.

Das Gesamtbild der Protokolle und Geräte, die bei ADSL benutzt werden, ist in ▶ Abbildung 3.26 zu sehen. Verschiedene Protokolle werden in unterschiedlichen Netzen eingesetzt, deshalb haben wir uns entschieden, das am weitesten verbreitete Szenario darzustellen. Innerhalb eines Hauses sendet ein Rechner, z.B. ein PC, IP-Pakete zu dem DSL-Modem unter Verwendung einer Sicherheitsschicht wie Ethernet. Das DSL-Modem sendet dann IP-Pakete über die Teilnehmeranschlussleitung zum DSLAM mithilfe der Protokolle, die wir jetzt untersuchen wollen. Am DSLAM (oder einem Router, der mit diesem verbunden ist, je nach Implementierung) werden die IP-Pakete extrahiert und treten in ein ISP-Netz ein, sodass sie jedes Ziel im Internet erreichen können.

Die Protokolle, die in Abbildung 3.26 auf der ADSL-Verbindung zu sehen sind, beginnen im unteren Teil mit der ADSL-Bitübertragungsschicht. Sie basieren auf einem digitalen Modulationsschema, dem orthogonalen Frequenzmultiplexing (auch bekannt als Mehrtonverfahren, siehe Abschnitt 2.5.3). Am oberen Ende des Stapsels, direkt unterhalb der IP-Vermittlungsschicht, finden wir PPP. Hierbei handelt es sich um dasselbe PPP, welches wir gerade für Packet-over-SONET-Übertragungen betrachtet haben. Verbindungsaufbau und -konfiguration werden auf die gleiche Weise durchgeführt, um IP-Pakete zu übertragen.

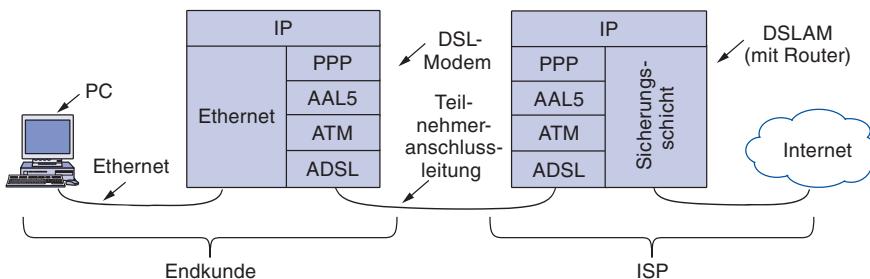


Abbildung 3.26: ADSL-Protokollstapel.

Zwischen ADSL und PPP befinden sich ATM und AAL5. Dies sind neue Protokolle, die wir bisher noch nicht kennengelernt haben. **ATM** (*Asynchronous Transfer Mode*) wurde in den frühen 1990er Jahren entwickelt und startete mit unglaublichem Tam-tam. Es versprach eine Netzwerktechnologie, die alle Telekommunikationsprobleme dieser Welt lösen würde, indem Sprache, Daten, Kabelfernsehen, Fernschreiber, Brieftauben, Dosentelefone an Schnüren, Trommeln und alles andere zu einem einzigen großen System verbunden werden, welches alles für jeden tun könnte. Dies trat nicht ein. Größtenteils ähnelten die Probleme von ATM denen, die wir für die OSI-Protokolle aufgezeigt haben, das heißt: schlechtes Timing, schlechte Technologie, schlechte Implementierung und schlechte Politik. Nichtsdestotrotz war ATM viel erfolgreicher als OSI. Auch wenn es nicht die Weltherrschaft übernommen hat, so findet es doch in einigen Nischen breite Anwendung, wozu Breitbandzugangsleitungen wie DSL und WAN-Verbindungen innerhalb von Telefonnetzen gehören.

ATM ist eine Sicherheitsschicht, die auf der Übertragung von **Zellen** fester Länge mit Informationen basiert. Das „asynchronous“ im Namen deutet darauf hin, dass die Zellen anders als bei SONET nicht immer als kontinuierlicher Bitstrom über synchrone Leitungen übertragen werden müssen. Zellen müssen nur dann gesendet werden, wenn auch Information zum Übertragen vorhanden ist. ATM ist eine verbindungsorientierte Technologie. Jede Zelle trägt einen Identifikator der **virtuellen Verbindung** (*virtual circuit*) in ihrem Header. Dieser Identifikator wird benutzt, um die Zellen über den Pfad der aufgebauten Verbindungen zu leiten.

Die Zellen haben jeweils eine Länge von 53 Byte, sie bestehen aus einer 48-Byte-Nutzlast und einem 5-Byte-Header. Die Verwendung von Zellen ermöglicht es ATM, die Bandbreite einer Verbindung der Bitübertragungsschicht flexibel zwischen unterschiedlichen Benutzern in Form von dünnen Scheiben aufzuteilen. Diese Eigenschaft ist z.B. dann nützlich, wenn sowohl Sprache als auch Daten über die gleiche Verbindung gesendet werden sollen und man keine langen Datenpakete haben möchte, die große Variationen in der Verzögerung von Sprachabtastungen verursachen würden. Die ungewöhnliche Wahl der Zelllänge (verglichen beispielsweise mit der eher natürlichen Wahl einer Potenz von 2) ist ein Anzeichen genau dafür, wie politisch der Entwurf von ATM war. Die 48-Byte-Größe für die Nutzdaten war ein Kompromiss, um die festgefahrenen Verhandlungen zwischen Europa, das 32-Byte-Zellen forderte, und den USA, die 64-Byte-Zellen bevorzugten, aufzulösen. Einen kurzen Überblick über ATM finden Sie bei Siu und Jain (1995).

Um Daten über ein ATM-Netz zu senden, müssen diese auf eine Folge von Zellen abgebildet werden. Diese Abbildung wird mithilfe einer ATM-Adaptionsschicht in einem Prozess durchgeführt, der Segmentierung und Reassemblierung genannt wird. Mehrere Adaptionsschichten wurden für unterschiedliche Dienste definiert, angefangen von periodischen Sprachabtastungen bis zu Paketdaten. Die Adaptionsschicht, die vorwiegend für Paketdaten benutzt wird, heißt **AAL5** (*ATM Adaption Layer 5*).

Ein AAL5-Rahmen ist in ▶ Abbildung 3.27 gezeigt. Anstatt eines Headers gibt es hier einen Trailer, der die Länge sowie einen 4-Byte-CRC-Wert zur Fehlererkennung enthält. Normalerweise ist dieser CRC-Wert derselbe, der für PPP und IEEE-802-LANs wie Ethernet benutzt wird. Wang und Crowcroft (1992) haben gezeigt, dass dieser stark genug ist, um unübliche Fehler wie Änderung der Zellreihenfolge zu entdecken. Außer den Nutzdaten enthält der AAL5-Rahmen noch Fülldaten (Padding). Dies vervollständigt die Gesamtlänge auf ein Vielfaches von 48 Byte, sodass der Rahmen gleichmäßig in Zellen aufgeteilt werden kann. Der Rahmen benötigt keine Adressen, da der Identifikator der virtuellen Verbindung, der mit jeder Zelle übertragen wird, dafür sorgt, den Rahmen zum richtigen Ziel bringen.

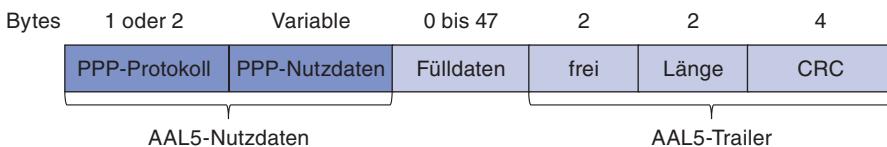


Abbildung 3.27: AAL5-Rahmen, der PPP-Daten überträgt.

Nachdem wir nun ATM beschrieben haben, müssen wir uns nur noch ansehen, wie ATM von PPP im Fall von ADSL benutzt wird. Dazu wird ein weiterer Standard, **PPPoA** (**PPP over ATM**), herangezogen. Dieser Standard ist eigentlich kein Protokoll (daher erscheint es nicht in Abbildung 3.26), sondern eher eine Spezifikation, wie mit PPP- und mit AAL5-Rahmen umzugehen ist. PPPoA wird in RFC 2364 beschrieben (Gross et al., 1998).

Lediglich die PPP-Protokoll- und PPP-Nutzdatenfelder werden in die AAL5-Nutzlast übernommen (siehe Abbildung 3.27). Das Protokollfeld unterrichtet den DSLAM am anderen Ende darüber, ob die Nutzlast ein IP-Paket oder ein Paket eines anderen Protokolls wie LCP ist. Dieses andere Ende weiß, dass die Zellen PPP-Informationen enthalten, weil eine virtuelle ATM-Verbindung zu diesem Zweck eingerichtet ist.

Innerhalb des AAL5-Rahmens wird keine PPP-Rahmenbildung benötigt, da es hier keinen Zweck erfüllt; ATM und AAL5 bieten schon Rahmenbildung. Eine weitere Rahmenbildung wäre sinnlos. Der PPP-CRC-Wert wird ebenfalls nicht gebraucht, weil AAL5 bereits genau diesen CRC-Wert umfasst. Dieser Fehlererkennungsmechanismus ergänzt die Codierung der ADSL-Bitübertragungsschicht eines Reed-Solomon-Codes zur Fehlerkorrektur und fügt einen 1-Byte-CRC zur Erkennung von Fehlern hinzu, die anderweitig nicht abgefangen werden. Dieses Schema hat einen sehr viel anspruchsvollereren Mechanismus für den Wiederanlauf im Fehlerfall als bei der Übertragung von Paketen über eine SONET-Leitung, da ADSL ein störanfälliger Kanal ist.

Zusammenfassung

Die **Sicherungsschicht** hat die Aufgabe, den reinen Bitstrom aus der Bitübertragungsschicht für die Weiterverarbeitung durch die Vermittlungsschicht in einen Rahmenstrom zu konvertieren. Die Sicherungsschicht kann diesen Strom mit unterschiedlichen Zuverlässigkeitsebenen anbieten, vom verbindungslosen unbestätigten Dienst bis hin zum zuverlässigen verbindungsorientierten Dienst.

Bei der **Rahmenbildung** gibt es verschiedene Methoden, z. B. Bytezählverfahren, Byte- und Bitstopfen. Protokolle der Sicherungsschicht implementieren Fehlerüberwachungsmethoden, sodass beschädigte Rahmen erkannt und korrigiert oder verlorene Rahmen erneut übertragen werden können. Um einen schnellen Sender davon abzuhalten, einen langsamen Empfänger mit Daten zu überschütten, setzen die Protokolle der Sicherungsschicht Flusskontrollmechanismen ein. Mit dem Schiebefenstermechanismus können Fehlerüberwachung und Flusskontrolle einfach koordiniert werden. Wenn die Fenstergröße 1 Paket beträgt, dann wird ein Stop-and-Wait-Protokoll verwendet.

Codes zur **Fehlerkorrektur** und -entdeckung fügen den Nachrichten redundante Information hinzu, indem eine Reihe von mathematischen Techniken verwendet wird. Faltungscodes und Reed-Solomon-Codes werden zur Fehlerkorrektur weitverbreitet eingesetzt, wobei Paritätsprüfcodes mit geringer Dichte zunehmend beliebter werden. Zu den Codes zur Fehlererkennung, die in der Praxis eingesetzt werden, gehören zyklische Redundanzprüfung und Prüfsummen. Diese Codes können sowohl in der Sicherungsschicht als auch in der Bitübertragungsschicht und höheren Schichten eingesetzt werden.

Wir haben verschiedene **Protokolle** untersucht, die eine zuverlässige Sicherungsschicht mithilfe von Bestätigungen und Neuübertragungen bzw. – unter realistischeren Annahmen – ARQ (*Automatic Repeat reQuest*) unterstützen. Angefangen bei einer fehlerfreien Umgebung, in der der Empfänger jeden an ihn gesendeten Rahmen verarbeiten kann, haben wir Flusskontrolle eingeführt, gefolgt von Fehlerkontrolle mit Sequenznummern und den Stop-and-Wait-Algorithmus. Dann haben wir den Schiebefensteralgorithmus verwendet, um bidirektionale Kommunikation zu ermöglichen, und haben das Konzept des Huckepacktransports eingeführt. Die letzten beiden Protokolle leiten die Übertragung von mehreren Rahmen, um den Sender davor zu bewahren, auf einer Verbindung mit einer langen Übertragungsverzögerung blockiert zu werden. Der Empfänger kann entweder alle Rahmen bis auf den Rahmen verwerfen, der in der Folge der nächste wäre, oder die Rahmen zwischenspeichern, die außerhalb der Reihenfolge ankommen, und negative Bestätigungen senden, um die Bandbreiteeffizienz zu erhöhen. Die erste Strategie ist ein Go-Back-N-Protokoll und die zweite ein Protokoll mit selektiver Wiederholung.

Das **Internet** verwendet vorwiegend PPP als Schicht-2-Protokoll auf Punkt-zu-Punkt-Leitungen. PPP stellt einen verbindungslosen unbestätigten Dienst zur Verfügung, benutzt Flagbytes zur Rahmenbegrenzung und ein CRC zur Fehlererkennung. PPP wird zur Paketübertragung für eine Vielzahl von Verbindungen benutzt, dazu gehören SONET-Verbindungen in WANs und ADSL-Verbindungen für Privathaushalte.



Übungsaufgaben

- 1** Ein Paket einer oberen Schicht ist in zehn Rahmen unterteilt, die je eine Chance von 80 % haben, unbeschädigt anzukommen. Wie oft muss die Nachricht durchschnittlich übertragen werden, um vollständig und korrekt anzukommen, wenn das Protokoll der Sicherungsschicht keine Fehlerüberwachung ausführt?
- 2** In einem Schicht-2-Protokoll wird die folgende Zeichencodierung verwendet:
A.: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000
Zeigen Sie, welche Bitfolge (binär) für den folgenden vier Zeichen langen Rahmenübertragen wird: A B ESC FLAG, wenn die folgenden Rahmenbildungsmethoden verwendet werden:
 - a. Bytezählung
 - b. Flagbytes mit Bytestopfen
 - c. Start- und Endflagbytes mit Bitstopfen
- 3** Das folgende Datenfragment erscheint inmitten eines Datenflusses, für den der im Text beschriebene Bytestopfen-Algorithmus verwendet wird: A B ESC C ESC FLAG FLAG D. Welches Ergebnis erhält man nach dem Auffüllen?
- 4** Welches ist der maximale Overhead beim Bytestopfen-Algorithmus?
- 5** Einer Ihrer Kommilitonen, Dagobert, hat darauf hingewiesen, dass es Verschwendungen ist, jeden Rahmen mit einem Flagbyte zu beenden und dann den nächsten wieder mit einem zweiten Flagbyte zu beginnen. Dies könnte man auch mit einem Flagbyte erledigen und sich das zweite Flagbyte sparen. Stimmen Sie dem zu?
- 6** Die Bitfolge 0111101111101111110 muss über die Sicherungsschicht übertragen werden. Welche Bitfolge wird nach dem Bitstopfen tatsächlich übertragen?
- 7** Können Sie sich Situationen vorstellen, in denen Protokolle mit offenen Schleifen (z.B. ein Hamming-Code) besser geeignet sind als die in diesem Kapitel beschriebenen Protokolle mit Bestätigungsfunction?
- 8** Um eine höhere Zuverlässigkeit bereitzustellen, als ein einzelnes Paritätsbit bieten kann, verwendet ein Codierschema zur Fehlererkennung ein Paritätsbit zum Überprüfen aller Bits mit ungerader und ein zweites Paritätsbit für alle Bits mit gerader Nummer. Wie lautet der Hamming-Abstand dieses Codes?
- 9** 16-Bit-Nachrichten werden unter Verwendung eines Hamming-Codes übertragen. Wie viele Prüfbits sind erforderlich um sicherzustellen, dass der Empfänger Einzelbitfehler entdecken und korrigieren kann? Geben Sie das übertragene Bitmuster für die Nachricht 1101001100110101 an. Gehen Sie davon aus, dass im Hamming-Code die gerade Parität verwendet wird.
- 10** Ein 12-Bit-Hamming-Code mit dem hexadezimalen Wert 0xE4F kommt beim Empfänger an. Wie lautet der Ursprungswert in hexadezimaler Darstellung? Gehen Sie davon aus, dass nicht mehr als ein Bit fehlerhaft ist.

- 11** Eine Möglichkeit der Erkennung von Fehlern ist, Daten als einen Block mit n Zeilen und k Bit pro Zeile zu übertragen und jeder Zeile und jeder Spalte ein Paritätsbit anzufügen. Das Bit rechts unten ist ein Paritätsbit, das seine Zeile und seine Spalte prüft. Können mit diesem Schema alle Einzelfehler aufgedeckt werden? Doppelte Fehler? Dreifache Fehler? Zeigen Sie, dass dieses Schema einige Vierbitfehler nicht entdecken kann.
- 12** Angenommen, Daten werden in Blöcken von 1 000 Bits übertragen. Welches ist die maximale Fehlerrate, bei der Fehlererkennungs- und Neuübertragungsmechanismen (1 Paritätsbit pro Block) besser als die Benutzung von Hamming-Code sind? Setzen Sie voraus, dass Bitfehler unabhängig voneinander sind und kein Bitfehler während einer Neuübertragung auftritt.
- 13** Ein aus n Zeilen und k Spalten bestehender Bitblock benutzt horizontale und vertikale Paritätsbits zur Fehlererkennung. Nehmen Sie an, dass aufgrund von Übertragungsfehlern genau vier Bit invertiert werden. Leiten Sie einen Ausdruck für die Wahrscheinlichkeit ab, dass dieser Fehler nicht entdeckt wird.
- 14** Welche Ausgabefolge wird bei Anwendung des Faltungscodes von Abbildung 3.7 erzeugt, wenn die Eingabefolge 10101010 (von links nach rechts) ist und der interne Status anfangs aus lauter Nullen besteht?
- 15** Nehmen Sie an, dass die Nachricht 1001 1100 1010 0011 unter Benutzung der Internetprüfsumme (4-Bit-Wort) übertragen wird. Welchen Wert hat die Prüfsumme?
- 16** Welchen Rest erhält man, wenn man $x^7 + x^5 + 1$ durch das Generatorpolynom $x^3 + 1$ teilt?
- 17** Der Bitstrom 10011101 wird mit der im Text beschriebenen CRC-Standardmethode übertragen. Das Generatorpolynom ist $x^3 + 1$. Geben Sie die tatsächlich übertragene Bitfolge an. Gehen Sie davon aus, dass das dritte Bit von links bei der Übertragung invertiert wird. Zeigen Sie, dass dieser Fehler vom Empfänger entdeckt wird. Geben Sie ein Beispiel von Bitfehlern in der übertragenen Bitfolge, die nicht vom Empfänger entdeckt wird.
- 18** Eine 1 024-Bit-Nachricht, die aus 992 Datenbits und 32 CRC-Bits besteht, wird gesendet. Der CRC-Wert wird mithilfe des durch IEEE 802 standardisierten Polynoms vom Grad 32 berechnet. Erklären Sie für jede der folgenden Situationen, ob die Fehler während Nachrichtenübertragung vom Empfänger entdeckt werden.
- Es gab einen Einzelbitfehler.
 - Es gab zwei isolierte Bitfehler.
 - Es gab 18 isolierte Bitfehler.
 - Es gab 47 isolierte Bitfehler.
 - Es gab einen 24 Bit langen Burstfehler.
 - Es gab einen 35 Bit langen Burstfehler.
- 19** Bei der Besprechung des ARQ-Protokolls in Abschnitt 3.3.3 wurde ein Szenario entworfen, das damit endete, dass der Empfänger zwei Kopien desselben Rahmens akzeptierte, weil der Bestätigungsrahmen verloren ging. Ist es möglich, dass ein Empfänger mehrere Kopien des selben Rahmens auch dann akzeptiert, wenn keiner der Rahmen (Nachrichten oder Bestätigung) verloren wurde?

- 20** Ein Kanal hat eine Übertragungsgeschwindigkeit von 4 kbit/s und eine Übertragungszeit von 20 ms. Bei welchen Rahmengrößen ergibt das Stop-and-Wait-Protokoll eine Effizienz von mindestens 50 %?
- 21** Kann ein Sender in Protokoll 3 den Timer starten, wenn dieser bereits läuft? Falls ja, warum kann es dazu kommen? Wenn nicht, warum ist dies unmöglich?
- 22** Über eine 3 000 km lange T1-Leitung werden mit Protokoll 5 64-Byte-Rahmen übertragen. Wie viele Bits sollte die Sequenznummer haben, wenn die Übertragungsgeschwindigkeit 6 $\mu\text{s}/\text{km}$ beträgt?
- 23** Nehmen Sie ein Schiebefensterprotokoll, das so viele Bits für Sequenznummern verwendet, dass die Sequenznummern nie wieder bei 0 anfangen müssen. Welche Beziehungen müssen zwischen den vier Fensterenden und der Fenstergröße bestehen, die konstant und für Sender und Empfänger gleich sei?
- 24** Würden sich Auswirkungen auf die Richtigkeit des Protokolls oder seine Effizienz ergeben, wenn die Prozedur *between* in Protokoll 5 die Bedingung $a \leq b \leq c$ anstatt der Bedingung $a \leq b < c$ überprüfen würde? Erläutern Sie Ihre Antwort.
- 25** In Protokoll 6 wird bei Ankunft eines Datenrahmens geprüft, ob die Sequenznummer von der erwarteten abweicht und *no_nak* wahr (*true*) ist. Falls beide Bedingungen zutreffen, wird ein NAK gesendet. Andernfalls wird der Hilfstimer gestartet. Angenommen, die *else*-Klausel würde weggelassen. Würde sich diese Änderung auf die Korrektheit des Protokolls auswirken?
- 26** Angenommen, die *while*-Schleife mit drei Anweisungen am Ende von Protokoll 6 würde entfernt. Wirkt sich dies auf die Korrektheit des Protokolls aus oder nur auf die Leistung? Erläutern Sie Ihre Antwort.
- 27** Die Entfernung von der Erde zu einem entfernten Planeten beträgt ungefähr 9×10^{10} m. Wie groß ist die Kanalauslastung, wenn ein Stop-and-Wait-Protokoll zur Rahmenübertragung auf einer Punkt-zu-Punkt-Verbindung mit 64 Mbit/s benutzt wird? Nehmen Sie an, dass die Rahmengröße 32 KB beträgt, die Lichtgeschwindigkeit ist 3×10^8 m/s.
- 28** Nehmen Sie an, dass in der vorherigen Aufgabe anstelle von Stop-and-Wait ein Schiebefensterprotokoll benutzt wird. Für welche gesendete Fenstergröße beträgt die Verbindungsausnutzung 100 %? Sie dürfen die Protokollverarbeitungszeiten beim Sender und Empfänger vernachlässigen.
- 29** In Protokoll 6 hat der Code für *frame_arrival* einen Abschnitt für NAKs. Dieser Abschnitt wird aufgerufen, wenn der ankommende Rahmen ein NAK ist und eine weitere Bedingung erfüllt wird. Beschreiben Sie einen Ablauf, bei dem die Erfüllung dieser anderen Bedingung wichtig ist.
- 30** Betrachten Sie den Betrieb von Protokoll 6 über eine perfekte (d. h. fehlerfreie) 1-Mbit/s-Leitung. Die maximale Rahmengröße ist 1 000 Bit. Neue Pakete werden im Abstand von einer Sekunde erzeugt. Das Timeout-Intervall steht auf 10 ms. Würde man den speziellen Bestätigungs timer weglassen, entstünden unnötige Timeouts. Wie oft würde die durchschnittliche Nachricht übertragen werden?
- 31** In Protokoll 6 gilt $\text{MAX_SEQ} = 2^n - 1$. Diese Bedingung ist sicher wünschenswert, um Header-Bits effizient zu verwenden. Wir haben aber nicht gezeigt, dass sie notwendig ist. Funktioniert das Protokoll noch richtig mit beispielsweise $\text{MAX_SEQ} = 4$?

- 32** Rahmen mit 1 000 Bit werden über einen 1-Mbit/s-Kanal über einen geostationären Satelliten versendet, dessen Übertragungszeit von der Erde weg 270 ms beträgt. Die Bestätigungen erfolgen immer huckepack auf Datenrahmen. Die Header sind sehr kurz. Es werden 3-Bit-Sequenznummern verwendet. Welche maximale Kanalauslastung ergibt sich bei
- Stop-and-Wait?
 - Protokoll 5?
 - Protokoll 6?
- 33** Berechnen Sie den Teil der Bandbreite, der für den zusätzlichen Overhead (Header und erneute Übertragungen) von Protokoll 6 auf einem stark benutzen 50-kbit/s-Satellitenkanal verschwendet wird. Es werden Datenrahmen mit 40 Header- und 3960 Datenbits verwendet. Gehen Sie davon aus, dass die Signalübertragungszeit von der Erde zum Satelliten 270 ms beträgt. ACK-Rahmen kommen nicht vor. NAK-Rahmen bestehen aus 40 Bits. Die Fehlerrate liegt für Datenrahmen bei 1 % und die von NAK-Rahmen ist verschwindend gering. Die Sequenznummern bestehen aus 8 Bits.
- 34** Betrachten Sie einen fehlerfreien Satellitenkanal mit 64 kbit/s zur Übertragung von Rahmen mit 512 Byte in eine Richtung. Die mit dem Rückverkehr ankommenden Bestätigungen sind sehr klein. Wie groß ist der maximale Durchsatz für die Fenstergrößen 1, 7, 15 und 127? Die Übertragungszeit Erde–Satellit beträgt 270 ms.
- 35** Ein 100 km langes Kabel läuft mit T1-Geschwindigkeit. Die Übertragungsgeschwindigkeit im Kabel beträgt 2/3 der Lichtgeschwindigkeit. Wie viele Bits passen in das Kabel?
- 36** Geben Sie mindestens einen Grund an, warum PPP Bytestopfen anstelle von Bitstopfen verwendet, um unabsichtliche Flagbytes innerhalb der Nutzdaten daran zu hindern, Konfusion zu verursachen.
- 37** Wie hoch ist der minimale Overhead beim Senden eines IP-Pakets über PPP? Berücksichtigen Sie nur den von PPP selbst eingebrachten Overhead, nicht den des IP-Headers. Wie hoch ist der maximale Overhead?
- 38** Ein 100-Byte-IP-Paket wird über eine Teilnehmeranschlussleitung mithilfe des ADSL-Protokollstacks übertragen. Wie viele ATM-Zellen werden übertragen? Beschreiben Sie kurz deren Inhalte.
- 39** Das Ziel dieser Praxisübung ist, einen Fehlererkennungsmechanismus unter Verwendung des im Text beschriebenen CRC-Standardalgorithmus zu implementieren. Schreiben Sie zwei Programme, *Generator* und *Verifikation*. Das *Generator*-Programm liest von der Standardeingabe eine Zeile ASCII-Text ein, die eine n -Bit-Nachricht als Folge von Nullen und Einsen enthält. Die zweite Zeile ist das k -Bit-Polynom, ebenfalls in ASCII. Die Ausgabe erfolgt auf der Standardausgabe als Zeile mit ASCII-Text mit $n+k$ Nullen und Einsen, die die zu übertragende Nachricht darstellen. Dann wird das Polynom wie eingelesen ausgegeben. Das *Verifikation*-Programm liest die Ausgabe des *Generator*-Programms und gibt eine Meldung aus, ob diese korrekt ist oder nicht. Schreiben Sie dann ein Programm *Änderungen*, das in der ersten Zeile das im Argument des Programms angegebene Bit verändert (das am weitesten links stehende Bit habe die Nummer 1), den Rest der beiden Zeilen aber korrekt kopiert. Durch die Eingabe von
`generator <file | verifikation`
sollten Sie feststellen können, dass die Nachricht korrekt ist; durch die Eingabe von
`generator <file | alter arg | verifikation`
sollten Sie eine Fehlermeldung erhalten.

Die MAC-Teilschicht (Medium Access Control)

| | |
|--|-----|
| 4.1 Die Kanalzuordnung | 305 |
| 4.2 Mehrfachzugriffsprotokolle | 309 |
| 4.3 Ethernet | 328 |
| 4.4 Drahtlose LANs | 349 |
| 4.5 Drahtloses Breitband | 364 |
| 4.6 Bluetooth | 372 |
| 4.7 RFID | 380 |
| 4.8 Switches der Sicherungsschicht | 386 |

» Netze können in zwei Kategorien unterteilt werden: Netze mit Punkt-zu-Punkt-Verbindungen und Netze mit Broadcast-Kanälen. In *Kapitel 2* haben wir Punkt-zu-Punkt-Verbindungen untersucht; in diesem Kapitel geht es um Broadcast-Verbindungen und ihre Protokolle.

Der wichtigste Aspekt in jedem Broadcast-Netz ist die Festlegung, wer den Kanal benutzen darf, falls es mehrere Bewerber für den Kanal gibt. Zur Verdeutlichung stellen Sie sich am besten eine Konferenzschaltung vor, in der sechs Personen an sechs verschiedenen Telefonen miteinander verbunden sind, sodass jeder jeden hören und jeder mit jedem sprechen kann. Es führt sehr wahrscheinlich zu einem Chaos, wenn eine Person zu sprechen aufhört und sofort zwei oder mehrere andere zu sprechen beginnen. Sitzen sich die Personen direkt gegenüber, wird das Chaos anderweitig vermieden. Bei einem Meeting kann man beispielsweise durch Handzeichen mitteilen, dass man sprechen will. Ist nur ein einziger Kanal vorhanden, dann ist die Festlegung, wer als Nächster an die Reihe kommt, viel schwieriger. Es gibt viele Protokolle, die dieses Problem lösen. Sie sind das Thema dieses Kapitels. In der Literatur werden die Broadcast-Kanäle auch **Mehrfachzugriffskanäle** bzw. **Multiaccess Channels** oder **Random Access Channels** genannt.

Protokolle, mit denen bestimmt werden kann, wer wann in einem Mehrfachzugriffskanal an die Reihe kommt, gehören zu einer Teilschicht der Sicherungsschicht namens **MAC** (*Medium Access Control*, Zugriffskontrolle). Die MAC-Teilschicht ist besonders in LANs wichtig, speziell in drahtlosen, da diese natürlicherweise Broadcast-Kanäle sind. Abgesehen von Satellitennetzen arbeiten WANs demgegenüber vorwiegend mit Punkt-zu-Punkt-Verbindungen. Da Mehrfachzugriffskanäle und LANs in engem Zusammenhang stehen, werden in diesem Kapitel sowohl LANs im Allgemeinen als auch einige Probleme behandelt, die streng genommen nicht unbedingt zur MAC-Teilschicht gehören, doch das Hauptthema hier wird die Steuerung des Kanals sein.

Vom technischen Gesichtspunkt her ist die MAC-Teilschicht der untere Teil der Sicherungsschicht; deshalb sollte sie logisch betrachtet eigentlich vor den in *Kapitel 3* beschriebenen Punkt-zu-Punkt-Protokollen behandelt werden. Aber für die meisten Leute ist das Verständnis von Protokollen einfacher, bei denen mehrere Parteien beteiligt sind, wenn Protokolle mit zwei Parteien bekannt sind. Aus diesem Grund sind wir etwas von der strengen Anordnung von unten nach oben abgewichen. <<

4.1 Die Kanalzuordnung

Das zentrale Thema dieses Kapitels ist, wie ein einzelner Broadcast-Kanal mehreren Benutzern gleichzeitig zur Verfügung gestellt werden kann. Der Kanal kann Teil des drahtlosen Spektrums innerhalb eines geografischen Bereichs sein oder ein einzelnes Kabel oder Glasfaserkabel, mit dem mehrere Knoten verbunden sind. Das spielt hier keine Rolle. In beiden Fällen verbindet der Kanal jeden Benutzer mit allen anderen Nutzern und jeder, der den Kanal ausgiebig verwendet, gerät mit anderen Nutzern, welche ebenfalls den Kanal verwenden wollen, in Konflikt.

Zunächst werfen wir einen Blick auf die Mängel statischer Zuordnungsverfahren bei Datenverkehr, der in Bursts auftritt. Dann werden wir die wesentlichen Voraussetzungen bereitstellen, die wir zur Modellierung der dynamischen Verfahren benötigen, welche wir in den folgenden Abschnitten untersuchen.

4.1.1 Statische Kanalzuordnung

Der normale Weg, einen einzelnen Kanal, etwa eine Telefonleitung, unter mehreren Benutzern aufzuteilen, besteht darin, seine Kapazität aufzuspalten, indem eines der Multiplexing-Verfahren eingesetzt werden, die wir im Abschnitt 2.5 besprochen haben, beispielsweise FDM (Frequenzmultiplexing). Bei N Benutzern wird die verfügbare Bandbreite in N gleich große Teile zerlegt und jedem Benutzer wird ein Teil zugewiesen. Da jeder Benutzer nun sein eigenes Frequenzband hat, gibt es keine Überlagerungen zwischen den Nutzern. Wenn nur eine kleine, gleichbleibende Anzahl von Benutzern einen ständigen Datenstrom oder ein relativ hohes Datenverkehrsaufkommen hat, ist diese Aufteilung ein einfacher und effizienter Zuordnungsalgorithmus. Ein drahtloses Beispiel sind UKW-Radiostationen. Jede Station erhält einen Teil des UKW-Bands, das er meistens zum Senden seiner Signale verwendet.

Wenn jedoch die Anzahl der einzelnen Sender groß ist und sich verändert oder der Datenverkehr aus Bursts besteht, dann wirft FDM einige Probleme auf. Ist das verfügbare Spektrum in N Bereiche unterteilt, sind momentan aber weniger als N Benutzer an einer Übertragung interessiert, dann wird wertvolle Kapazität verschenkt. Und wollen andererseits mehr als N Benutzer kommunizieren, so wird einigen von ihnen mangels Bandbreite der Zugang verweigert, obwohl andere Benutzer, denen ein Frequenzband zugeteilt wurde, kaum etwas senden oder empfangen.

Selbst wenn man davon ausgeht, dass die Anzahl der Benutzer im Großen und Ganzen stets bei N liegt, so ist die Aufteilung eines verfügbaren Kanals in statische Unterkanäle dennoch ineffizient. Das Grundproblem ist hier, dass die Bandbreite einfach verloren geht, wenn Benutzer nichts übertragen. Sie nutzen ihren Unterkanal nicht selbst, er wird aber auch nicht anderen zur Verfügung gestellt. Eine statische Zuordnung ist für die meisten Computersysteme dann ungeeignet, wenn der Datenverkehr extrem starken Schwankungen unterworfen ist, häufig mit einem Verhältnis zwischen Spitzen- und durchschnittlichen Übertragungsraten von 1 000:1. Daher sind die meisten Kanäle einen Großteil der Zeit im Leerlaufzustand.

Die geringe Leistung des statischen FDM ist leicht anhand einer einfachen Berechnung aus der Warteschlangentheorie nachvollziehbar. Beginnen wir damit, die mittlere Verzögerungszeit T zu finden, um einen Rahmen über einen Kanal der Kapazität von C Bit/s zu senden. Wir nehmen an, dass die Rahmen zufällig mit einer durchschnittlichen Ankunftsrate von λ Rahmen/Sekunde ankommen und dass die Rahmenlänge durchschnittlich um $1/\mu$ Bit/Rahmen schwankt. Mit diesen Parametern beträgt die Dienstrate μC Rahmen/Sekunde. Ein Standardergebnis aus der Warteschlangentheorie ist:

$$T = \frac{1}{\mu C - \lambda}$$

(Für die Wissbegierigen unter den Lesern: Dieses Ergebnis gilt für eine „M/M/1“-Warteschlange. Diese setzt voraus, dass die Zufälligkeit der Zeiten zwischen der Ankunft von Rahmen und der Rahmenlänge einer Exponentialverteilung folgt bzw. dass sie das Ergebnis eines Poisson-Prozesses ist.)

Wenn in unserem Beispiel C gleich 100 Mbit/s, die mittlere Rahmenlänge $1/\mu$ 10 000 Bit und die Rahmenankunftsrate λ 5 000 Rahmen/Sekunde ist, dann gilt $T=200$ µs. Wenn wir die Übertragungsverzögerung durch die Warteschlange ignorieren und nur gefragt hätten, wie lange das Senden eines 10 000-Bit-Rahmens in einem 100-Mbit/s-Netz dauert, dann hätten wir die (falsche) Antwort 100 µs erhalten. Dieses Ergebnis gilt nur, wenn für den Kanal keine Konkurrenz besteht.

Lassen Sie uns nun den ganzen Kanal in N unabhängige Unterkanäle mit einer Kapazität von je C/N Bit/s unterteilen. Die mittlere Rahmenankunftsrate für jeden Unterkanal beträgt jetzt λ/N . Wenn wir nun T erneut berechnen, erhalten wir:

$$T_N = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \quad (4.1)$$

Die mittlere Verzögerung für den geteilten Kanal ist N -mal schlechter, als wenn alle Rahmen ordentlich in einer großen zentralen Warteschlange angeordnet wären. Dies ist vergleichbar mit dem Vorraum einer Bank mit mehreren Geldautomaten. Auch hier ist es besser, wenn eine einzige Schlange gebildet wird, die sich dann auf alle Automaten verteilt, als wenn sich vor jedem Automaten eine eigene Schlange bildet.

Genau die gleichen Argumente, die für FDM gelten, sind auch auf andere Arten der statischen Kanalaufteilung anwendbar. Wird beispielsweise Zeitmultiplexing benutzt (TDM), dann wird jedem Benutzer statisch jede N -te Zeitscheibe zugeordnet. Falls ein Benutzer seine Zeitscheibe nicht beansprucht, liegt diese brach. Das Gleiche gilt auch, wenn das Netz physikalisch aufgeteilt wird. Würden wir in obigem Beispiel das 100-Mbit/s-Netz durch zehn Netze mit je 10 Mbit/s ersetzen und jeden Benutzer statisch einem dieser Netze zuordnen, so würde die mittlere Übertragungsverzögerung von 200 µs auf 2 ms verändert.

Da keine der herkömmlichen statischen Kanalzuordnungsmethoden bei Burst-Datenverkehr gut funktioniert, untersuchen wir jetzt die dynamischen Zuordnungsverfahren.

4.1.2 Voraussetzungen für dynamische Kanalzuordnung

Bevor wir zur ersten der vielen Kanalzuordnungsmethoden in diesem Kapitel kommen, ist es sinnvoll, das Problem der Zuordnung genau zu formulieren. Alle Überlegungen auf diesem Gebiet bauen auf den folgenden fünf Grundannahmen auf:

- 1. Unabhängiger Datenverkehr:** Das Modell besteht aus N unabhängigen **Stationen** (wie Computer, Telefone), jede mit einem Programm (oder Benutzer), welches Rahmen zur Übertragung erzeugt. Die erwartete Anzahl der Rahmen, die innerhalb eines Intervalls der Länge Δt erzeugt werden, ist $\lambda \Delta t$, wobei λ eine Konstante ist (die Ankunftshäufigkeit von neuen Rahmen). Ist ein Rahmen einmal erstellt worden, so ist die Station blockiert, bis der Rahmen erfolgreich übertragen wurde.
- 2. Einzelkanal:** Für die gesamte Kommunikation ist ein einzelner Kanal verfügbar. Alle Stationen können über ihn übertragen und empfangen. Es wird davon ausgegangen, dass alle Stationen die gleiche Kapazität aufweisen, auch wenn Protokolle ihnen unterschiedliche Rollen (z.B. Prioritäten) zuweisen können.
- 3. Beobachtbare Kollisionen:** Wenn zwei Rahmen gleichzeitig übertragen werden, überschneiden sie sich eine gewisse Zeit und das resultierende Signal ist gestört. Dieses Ereignis wird **Kollision** genannt. Alle Stationen können erkennen, dass eine Kollision stattgefunden hat. Ein kollidierter Rahmen muss später erneut übertragen werden. Es gibt keine anderen Fehler als jene, die durch Kollisionen verursacht werden.
- 4. Kontinuierlicher oder in Zeitscheiben unterteilter Zeitverlauf:** Zeit kann als ununterbrochen angenommen werden, in diesem Fall kann die Übertragung eines Rahmens zu jedem Zeitpunkt beginnen. Alternativ dazu könnte die Zeit auch in einzelne Intervalle (sogenannte Zeitscheiben) unterteilt werden. Die Übertragung eines Rahmens muss dann immer am Anfang einer Zeitscheibe beginnen. Eine Zeitscheibe kann 0, 1 oder mehrere Rahmen enthalten, was einem Kanal im Leerlaufzustand, einer erfolgreichen Übertragung oder einer Kollision entspricht.
- 5. Trägerprüfung oder keine Trägerprüfung:** Unter der Voraussetzung, dass Trägerprüfung stattfindet, können Stationen feststellen, ob der Kanal genutzt wird, bevor sie versuchen, ihn zu verwenden. Keine Station unternimmt einen Nutzungsversuch, solange der Kanal als besetzt erkannt wird. Wenn es keine Trägerprüfung gibt, können Stationen den Kanal nicht überprüfen, bevor sie versuchen, ihn zu benutzen. Sie fangen einfach mit der Übertragung an. Erst hinterher können sie feststellen, ob die Übertragung erfolgreich war oder nicht.

Schauen wir uns die zugrunde liegenden Annahmen etwas genauer an. Die erste Annahme besagt, dass die Ankunft der Rahmen unabhängig ist – sowohl über mehrere Stationen hinweg als auch an einer bestimmten Station – und dass Rahmen auf nicht vorhersehbare Weise, aber kontinuierlich erzeugt werden. Eigentlich ist diese Annahme kein besonders gutes Modell für Netzverkehr, da allgemein bekannt ist, dass Pakete in Bursts über viele Zeiträume hinweg ankommen (Paxson und Floyd, 1995; Leland et al., 1994). Dennoch sind **Poisson-Modelle** – wie diese häufig genannt werden

– sinnvoll, da sie mathematisch beherrschbar sind. Sie helfen bei der Analyse von Protokollen, um eine grobe Vorstellung davon zu bekommen, wie sich die Performanz innerhalb eines Einsatzbereichs ändert und wie sich diese im Vergleich zu anderen Entwürfen verhält.

Die Annahme eines einzelnen Kanals ist das Herz des ganzen Modells. Es gibt keine externen Kommunikationsmöglichkeiten. Stationen können nicht ihre Hand heben, um dem Lehrer anzudeuten, dass er sie aufrufen soll, daher werden wir wohl nach besseren Lösungen suchen müssen.

Die verbleibenden drei Annahmen hängen von der Konstruktion des Systems ab. Wenn wir ein bestimmtes Protokoll untersuchen, werden wir jeweils angeben, welche Annahmen gelten.

Die Kollisionsannahme ist grundlegend. Stationen müssen für den Fall, dass Rahmen neu übertragen werden sollen, die Möglichkeit haben, Kollisionen zu entdecken. Für verkabelte Kanäle kann eine Knotenhardware entworfen werden, die möglicherweise auftretende Kollisionen entdeckt. Die Stationen können dann ihre Übertragungen vorzeitig beenden und somit die Verschwendungen von Kapazität vermeiden. Für drahtlose Kanäle ist diese Kollisionserkennung viel schwieriger. Hier wird in der Regel im Nachhinein, wenn ein erwarteter Bestätigungsrahmen ausbleibt, gefolgert, dass eine Kollision aufgetreten ist. Es kann auch vorkommen, dass einige der an einer Kollision beteiligten Rahmen erfolgreich empfangen werden, abhängig von den Signalen und der empfangenen Hardware. Dies ist jedoch gewöhnlich nicht der Fall, wir gehen deshalb davon aus, dass alle an einer Kollision beteiligten Rahmen verloren sind. Wir werden auch Protokollen begegnen, die verhindern, dass Kollisionen überhaupt auftreten.

Der Grund für die beiden Annahmen hinsichtlich der Zeit ist, dass in Zeitscheiben aufgespaltene Zeit eingesetzt werden kann, um die Leistung zu verbessern. Dies macht es jedoch erforderlich, dass die Stationen einem Haupttaktgeber folgen oder ihre Aktionen miteinander synchronisieren, um die Zeit in diskrete Intervalle aufzuteilen. Somit ist diese Möglichkeit nicht immer gegeben. Wir werden Systeme mit beiden Arten von Zeit besprechen und untersuchen. Für ein bestimmtes System trifft selbstverständlich immer nur eine zu.

Ebenso kann in einem Netz eine Trägerprüfung (*carrier sense*) stattfinden oder nicht, aber nicht beides zugleich. Kabelnetze unterstützen in der Regel die Prüfung des Trägers. Drahtlose Netze können dies nicht immer effizient tun, weil möglicherweise nicht jede Station im Funkbereich jeder anderen Station liegt. Ebenso ist Trägerprüfung nicht in Umgebungen verfügbar, in denen eine Station nicht direkt mit anderen Stationen kommunizieren kann. Ein Beispiel dafür ist ein Kabelmodem, bei dem Stationen über das Kabelkopfende kommunizieren müssen.

Um jegliches Missverständnis zu vermeiden, sollte hier noch erwähnt werden, dass kein Mehrfachzugriffsprotokoll eine zuverlässige Sendung garantieren kann. Selbst wenn es keine Kollisionen gibt, könnte der Empfänger aus unterschiedlichen Gründen einige der Rahmen falsch kopiert haben. Zuverlässigkeit bieten andere Teile der Sicherungsschicht oder höherer Schichten.

4.2 Mehrfachzugriffsprotokolle

Viele Algorithmen zur Zuordnung eines Mehrfachzugriffskanals sind heute im Einsatz. In den folgenden Abschnitten werden wir einen kleinen Ausschnitt der interessanteren Protokolle behandeln und Beispiele dafür anführen, wie diese in der Praxis häufig eingesetzt werden.

4.2.1 ALOHA

Die Geschichte unseres ersten MAC begann im unberührten Hawaii in den frühen 1970er Jahren. In diesem Fall kann „unberührt“ als „hat kein funktionierendes Telefonsystem“ interpretiert werden. Dies hat das Leben für die Forscher Norman Abramson und seine Kollegen an der Universität von Hawaii nicht angenehmer gemacht, die versuchten, Benutzer auf entfernten Inseln mit dem Hauptrechner in Honolulu zu verbinden. Ihre eigenen Kabel unter dem Pazifischen Ozean her zu verlegen war nicht wirklich eine Option, also hielten sie Ausschau nach einer anderen Lösung.

Diese Lösung wurde auch gefunden: die Verwendung von Kurzstreckenfunk, wobei alle Benutzerterminals dieselbe Upstream-Frequenz benutzten, um Rahmen an den Zentralrechner zu senden. Dazu gehörte auch eine einfache und elegante Methode, um das Problem der Kanalzuordnung zu lösen. Diese Ergebnisse wurden seitdem von vielen Forschern erweitert (Schwarz und Abramson, 2009). Obwohl Abramsons Arbeit – das ALOHA-System – den Bodenfunk verwendete, ist die Grundidee auf jedes System übertragbar, in dem nicht koordinierte Benutzer um die Benutzung eines einzelnen gemeinsamen Kanals konkurrieren.

Im Folgenden werden zwei ALOHA-Versionen beschrieben: reines ALOHA und S-ALOHA (Slotted ALOHA). Sie unterscheiden sich darin, ob wie bei der reinen Version die Zeit kontinuierlich ist oder ob sie in einzelne Zeitscheiben aufgeteilt wird, in die alle Rahmen passen müssen.

Reines ALOHA

Das grundlegende Konzept eines ALOHA-Systems ist einfach: Benutzer dürfen jederzeit, wenn sie Daten senden wollen, übertragen. Es treten selbstverständlich Kollisionen auf und die kollidierenden Rahmen werden beschädigt. Ein Sender muss herausfinden, ob dies der Fall ist. Wenn im ALOHA-System jede Station ihren Rahmen an den Zentralrechner gesendet hat, überträgt dieser den Rahmen via Broadcast zu allen Stationen. Eine sendende Station kann diese Übertragung somit am Hub abhören, um festzustellen, ob ihr Rahmen durchgekommen ist. In anderen Systemen wie kabelbehafteten LANs kann der Sender eventuell während der Übertragung auf Kollisionen abhören.

Falls der Rahmen zerstört wurde, wartet der Absender eine zufällig gewählte Zeitspanne und sendet ihn dann nochmals. Die Wartezeit muss zufällig sein, sonst kollidieren die gleichen Rahmen immer wieder, bis in alle Ewigkeit. Systeme, in denen Benutzer einen gemeinsamen Kanal so benutzen, dass es zu Konflikten kommen kann, sind allgemein als **Konkurrenzsysteme** (*contention system*) bekannt.

► Abbildung 4.1 zeigt die prinzipielle Darstellung der Erzeugung von Rahmen in einem ALOHA-System. Die Rahmen haben alle die gleiche Länge, da der Datendurchsatz von ALOHA-Systemen dadurch maximiert wird, dass eine feste Rahmengröße vorgeschrieben ist.

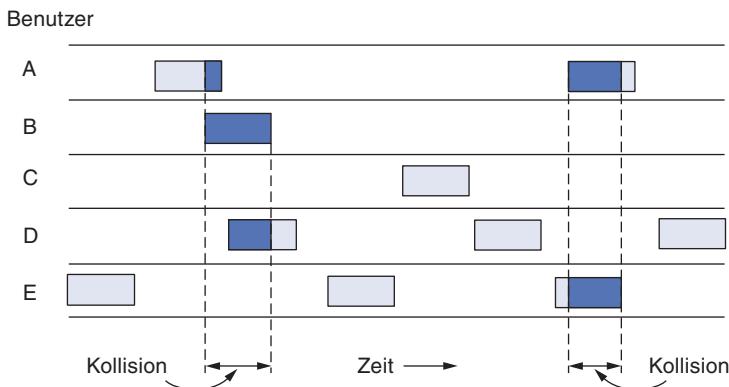


Abbildung 4.1: Beim reinen ALOHA werden Rahmen zu beliebigen Zeiten übertragen.

Jedes Mal, wenn zwei Rahmen zur gleichen Zeit versuchen, den Kanal zu besetzen, entsteht eine Kollision (siehe Abbildung 4.1) und beide werden verstümmelt. Falls auch nur das erste Bit eines neuen Rahmens das letzte Bit eines fast beendeten Rahmens überschneidet, werden beide Rahmen völlig zerstört (d.h. haben falsche Prüfsummen) und müssen später erneut übertragen werden. Eine Prüfsumme kann nicht (und sollte auch nicht) zwischen einem totalen und einem teilweisen Verlust unterscheiden.

Eine interessante Frage ist: Wie sieht die Effizienz eines ALOHA-Systems aus? Mit anderen Worten, welcher Teil aller übermittelten Rahmen entgeht unter diesen chaotischen Bedingungen einer Kollision? Betrachten wir zunächst eine unendliche Menge von Benutzern, die etwas in ihre Terminals (Stationen) eingeben. Ein Benutzer befindet sich immer in einem von zwei Zuständen: Er schreibt oder er wartet. Zu Beginn befinden sich alle Benutzer im schreibenden Zustand. Wenn eine Zeile fertig geschrieben ist, hört der Benutzer zu schreiben auf und wartet auf eine Antwort. Dann überträgt die Station einen Rahmen, in dem sich die Zeile befindet, über den gemeinsamen Kanal zum Zentralrechner und prüft den Kanal, um zu sehen, ob die Übertragung erfolgreich war. Trifft das zu, sieht der Benutzer die Antwort und fährt mit dem Schreiben fort. Andernfalls wartet der Benutzer, während die Station den Rahmen immer wieder neu überträgt, bis er erfolgreich angekommen ist.

Die Rahmenübertragungszeit sei die Zeit, die benötigt wird, um den Standardrahmen mit fester Länge zu übermitteln (d.h. die Länge des Rahmens geteilt durch die Bitübertragungsrate). Wir gehen hier davon aus, dass neue Rahmen, die von den Stationen erzeugt werden, durch eine Poisson-Verteilung mit einer mittleren Anzahl von N Rahmen pro Rahmenübertragungszeit wohlmodelliert sind. (Die Annahme einer unbegrenzten Benutzerzahl ist erforderlich, um sicherzustellen, dass N nicht abnimmt, wenn die Benutzer gerade blockiert sind.) Falls $N > 1$ ist, erzeugen die Benutzer die

Rahmen schneller als der Kanal sie verarbeiten kann, also erleidet fast jeder Rahmen eine Kollision. Für einen vernünftigen Durchsatz gehen wir von $0 < N < 1$ aus.

Abgesehen von den neuen Rahmen generieren die Stationen auch Neuübertragungen von vorher verunglückten Rahmen. Wir nehmen an, dass die alten und die neuen Rahmen zusammen durch eine Poisson-Verteilung wohlmodelliert sind, mit einem mittleren Wert von G Rahmen pro Rahmenübertragungszeit. Natürlich gilt $G \geq N$. Bei niedriger Belastung (d.h. $N \approx 0$) gibt es wenig Kollisionen, also auch wenig Neuübertragungen, sodass $G \approx N$ ist. Bei höheren Lasten entstehen viele Kollisionen, sodass $G > N$ ist. Für jede beliebige Last ist der Datendurchsatz S die gegebene Last G multipliziert mit der Wahrscheinlichkeit, dass eine Übertragung erfolgreich ist, d.h. $S = GP_0$, wobei P_0 die Wahrscheinlichkeit ist, dass ein Rahmen keine Kollision erleidet.

Ein Rahmen kollidiert nicht, wenn keine anderen Rahmen innerhalb einer Rahmenübertragungszeit nach seinem Start gesendet werden (► Abbildung 4.2). Unter welchen Bedingungen kommt der graue Rahmen unbeschädigt an? Angenommen, t ist die zum Senden eines Rahmens benötigte Zeit. Wenn ein anderer Benutzer in der Zeit zwischen t_0 und t_0+t einen Rahmen erstellt, kollidiert das Ende dieses Rahmens mit dem Anfang des grauen Rahmens. In Wirklichkeit war das Schicksal des grauen Rahmens schon vor der Übertragung des ersten Bits besiegelt. Da beim reinen ALOHA eine Station den Kanal vor der Übertragung nicht prüft, hat sie keine Möglichkeit zu erfahren, dass ein anderer Rahmen schon unterwegs ist. Ebenso wird jeder andere Rahmen, der zwischen t_0+t und t_0+2t abgesendet wird, mit dem Ende des dunkelblauen Rahmens zusammenstoßen.

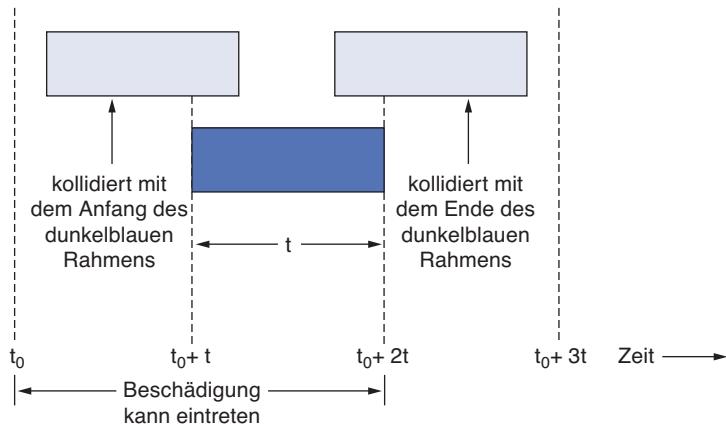


Abbildung 4.2: Gefährliche Zeitspanne für den dunkelblauen Rahmen.

Die Wahrscheinlichkeit, dass k Rahmen innerhalb einer gegebenen Rahmenübertragungszeit produziert werden, in der G Rahmen erwartet werden, ist durch die Poisson-Verteilung

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \quad (4.2)$$

gegeben. Die Wahrscheinlichkeit für 0 Rahmen ist daher genau e^{-G} . Innerhalb einer Zeitspanne, die zwei Rahmenübertragungszeiten lang ist, werden im Durchschnitt $2G$ Rahmen erzeugt. Die Wahrscheinlichkeit, dass innerhalb der gefährlichen Zeitspanne keine weiteren Rahmen begonnen werden, ist somit $P_0 = e^{-2G}$. Unter Verwendung von $S = GP_0$ erhalten wir

$$S = Ge^{-2G}$$

Das Verhältnis zwischen dem gegebenen Datenverkehr und dem Datendurchsatz wird in ▶ Abbildung 4.3 dargestellt. Der maximale Durchsatz erfolgt bei $G=0,5$ mit $S=1/2e$, was etwa 0,184 ergibt. Mit anderen Worten: Wir können bestenfalls mit einer Kanalnutzung von maximal 18 % rechnen. Dieses Ergebnis ist nicht sehr ermutigend. Wenn aber jeder Daten dann überträgt, wann er will, ist kaum eine Erfolgsrate von 100 % zu erwarten.

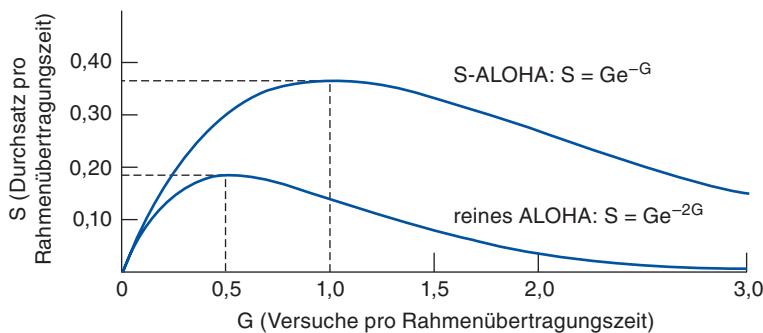


Abbildung 4.3: Durchsatz gegenüber angebotenem Datenverkehr bei ALOHA-Systemen.

S-ALOHA

Kurz nachdem ALOHA in der Szene auftauchte, veröffentlichte Roberts (1972) eine Methode zur Verdoppelung der Kapazität eines ALOHA-Systems. Er schlug vor, die Zeit in einzelne Intervalle – sogenannte **Zeitscheiben** (*slot*) – aufzuteilen, wobei jedes Intervall einem Rahmen entspricht. Bei diesem Ansatz müssen sich die Benutzer bezüglich der Grenzen der Zeitscheiben abstimmen. Eine Möglichkeit, eine gewisse Synchronisation zwischen den Benutzern zu erreichen, wäre die, dass eine Station zu Beginn eines jeden Intervalls wie eine Uhr einen Piepton aussendet.

Bei der Methode von Roberts, die als **S-ALOHA** (Slotted ALOHA) bekannt wurde, darf im Gegensatz zum **reinem ALOHA** von Abramson eine Station nicht sofort senden, wenn der Benutzer eine Zeile eingibt. Stattdessen muss auf die nächste Zeitscheibe gewartet werden. So wird der zeitlich ununterbrochene Fluss des reinen ALOHA in eine Variante mit diskreten Zeitabschnitten abgewandelt. Dies halbiert die gefährliche Zeitspanne. Betrachten Sie dazu Abbildung 4.3 und überlegen Sie sich die nun möglichen Kollisionen. Die Wahrscheinlichkeit, dass kein anderer Datenverkehr in der gleichen Zeitscheibe wie unser Testrahmen stattfindet, ist e^{-G} , woraus sich

$$S = Ge^{-G} \tag{4.3}$$

ergibt. Wie man in Gleichung (4.3) sieht, hat S-ALOHA seinen Spitzenwert bei $G=1$ mit einem Durchsatz von $S=1/e$ bzw. ungefähr 0,368. Dies ist also eine Verdoppelung gegenüber dem reinen ALOHA. Wenn das System mit $G=1$ arbeitet, ist die Wahrscheinlichkeit für eine leere Zeitscheibe 0,368 (aus Gleichung (4.2)). Im besten Fall liefert S-ALOHA 37 % leere Zeitscheiben, 37 % erfolgreiche Übertragungen und 26 % Kollisionen. Ein Betrieb mit höheren Werten für G verringert zwar die Zahl der leeren Zeitscheiben, erhöht aber die Zahl der Kollisionen exponentiell. Um zu verstehen, wie dieser schnelle Zuwachs von Kollisionen mit G zusammenhängt, betrachten wir als Beispiel die Übertragung eines Testrahmens. Die Wahrscheinlichkeit, dass er eine Kollision vermeiden kann, ist e^{-G} , welches die Wahrscheinlichkeit ist, dass alle anderen Stationen nicht in der gleichen Zeitscheibe übertragen. In diesem Fall beträgt die Wahrscheinlichkeit einer Kollision nur $1-e^{-G}$. Die Wahrscheinlichkeit, dass eine Übertragung genau k Versuche benötigt (d.h. $k-1$ Kollisionen, gefolgt von einem Erfolg) ist

$$P_k = e^{-G}(1-e^{-G})^{k-1}$$

Die erwartete Anzahl an Übertragungen E pro am Terminal eingegebener Zeile ist dann

$$E = \sum_{k=1}^{\infty} k P_k = \sum_{k=1}^{\infty} k e^{-G}(1-e^{-G})^{k-1} = e^G$$

Als Folge der exponentiellen Abhängigkeit von E von G können bereits geringe Erhöhungen der Kanallast die Leistung erheblich verringern.

S-ALOHA ist darüber hinaus aus einem Grund bemerkenswert, der nicht auf den ersten Blick ersichtlich ist. Es wurde in den 1970er Jahren entwickelt, in ein paar experimentellen Systemen eingesetzt und geriet dann überwiegend in Vergessenheit. Als der Internetzugang über Kabel erfunden wurde, entstand plötzlich das Problem, wie man einen gemeinsam genutzten Kanal unter mehreren konkurrierenden Benutzern zuweisen sollte. Hier wurde dann S-ALOHA zur Rettung der Lage wieder aus der Versenkung geholt. Die Situation, dass mehrere RFID-Tags mit demselben RFID-Lesegerät kommunizieren wollen, stellt eine weitere Variation desselben Problems dar. S-ALOHA, gemischt mit einem Spritzer weiterer Ideen, wurde erneut zur Rettung herangezogen. Es ist schon öfter vorgekommen, dass vollkommen gültige Protokolle aus politischen Gründen nicht mehr verwendet werden (beispielsweise weil ein großes Unternehmen möchte, dass alle nach seinem Verfahren arbeiten) oder aufgrund von sich ständig verändernden Technologietrends. Doch dann, Jahre später, erkennt jemand, dass das eigentlich vergessene Protokoll sein aktuelles Problem löst. Aus diesem Grunde beschäftigen wir uns in diesem Kapitel mit mehreren eleganten Protokollen, die zwar zurzeit keine große Verbreitung aufweisen, aber vielleicht in zukünftigen Anwendungen einen Einsatz finden – vorausgesetzt, dass viele Netzentwickler sie kennen. Natürlich behandeln wir auch viele Protokolle, die aktuell benutzt werden.

4.2.2 CSMA-Protokolle (Carrier Sense Multiple Access)

Bei S-ALOHA liegt die bestmögliche Kanalauslastung bei $1/e$. Dieses niedrige Ergebnis ist nicht erstaunlich, da hier die Stationen senden, wann sie wollen, und die Aktivitäten der anderen Stationen nicht kennen, sodass viele Kollisionen zu erwarten sind. In LANs können die Stationen demgegenüber häufig erkennen, was andere Stationen tun, und somit ihr eigenes Verhalten anpassen. Diese Netze können eine viel bessere Auslastung als $1/e$ erreichen. In diesem Abschnitt werden einige Protokolle mit Hinblick auf die Leistungsverbesserung behandelt.

Protokolle, bei denen Stationen einen Träger (d.h. eine Übertragung) abhören und dementsprechend handeln, heißen **Protokolle mit Trägerprüfung** (*carrier sense protocol*). Hier wurden inzwischen mehrere Varianten vorgeschlagen und schon vor langer Zeit detailliert analysiert, siehe zum Beispiel Kleinrock und Tobagi (1975). Im Folgenden werden wir uns verschiedene Varianten von Protokollen mit Trägerprüfung ansehen.

Persistentes CSMA und nicht persistentes CSMA

Als erstes Protokoll mit Trägerprüfung untersuchen wir das **1-persistente CSMA** (*Carrier Sense Multiple Access, Mehrfachzugriff mit Trägerprüfung*). Dieser Name ist ein wenig sperrig für das einfachste CSMA-Verfahren. Wenn eine sendewillige Station Daten übertragen möchte, hört sie erst den Kanal ab, ob bereits jemand sendet. Ist der Kanal frei, dann sendet die Station ihre Daten. Wenn der Kanal dagegen besetzt ist, so wartet die Station einfach, bis er frei wird, dann überträgt sie einen Rahmen. Falls eine Kollision auftritt, wartet die Station eine zufällige Zeitspanne und beginnt von vorn. Dieses Protokoll heißt 1-persistent, weil die Station mit einer Wahrscheinlichkeit von 1 sendet, wenn der Kanal im Leerlaufzustand ist.

Man könnte nun erwarten, dass dieses Verfahren Kollisionen bis auf den seltenen Fall von gleichzeitigem Senden vermeidet, doch tatsächlich tut es das nicht. Wenn zwei Stationen sendebereit werden, während gerade eine dritte Station sendet, so warten beide höflich, bis diese Übertragung beendet ist, und dann beginnen beide genau gleichzeitig zu senden, was zu einer Kollision führt. Wenn sie nicht so ungeduldig wären, gäbe es weniger Kollisionen.

Außerdem hat auch die Ausbreitungsverzögerung einen großen Einfluss auf Kollisionen. Es besteht die Möglichkeit, dass kurz nachdem eine Station zu senden begonnen hat, eine andere Station auch senden will und den Kanal überprüft. Wenn das Signal der ersten Station die zweite noch nicht erreicht hat, findet die zweite einen Kanal im Leerlaufzustand vor und beginnt ebenfalls zu senden, was zu einer Kollision führt. Diese Möglichkeit hängt von der Anzahl der Rahmen ab, die sich auf dem Kanal aufhalten können, bzw. dem **Bandbreite-Verzögerung-Produkt** des Kanals. Wenn nur ein kleiner Teil eines Rahmens auf den Kanal passt, wie dies für die meisten LANs der Fall ist, da hier die Ausbreitungsverzögerung gering ist, dann tritt selten eine Kollision auf. Je größer das Bandbreite-Verzögerung-Produkt ist, umso wichtiger wird dieser Effekt und umso geringer wird die Leistungsfähigkeit des Protokolls.

Trotz allem hat dieses Protokoll eine bessere Performanz als das reine ALOHA, da beide Stationen genug Anstand besitzen, nicht mit dem Rahmen der dritten Station in Konflikt zu geraten. Genau das Gleiche gilt auch für S-ALOHA.

Die zweite Variante eines Protokolls mit Trägerprüfung ist **nicht-persistentes CSMA**. Bei diesem Protokoll mit Trägerprüfung wird bewusst der Versuch unternommen, etwas weniger gierig zu sein als im vorherigen Protokoll. Wie vorher überprüft eine Station den Kanal, wenn es einen Rahmen senden möchte, und falls niemand anderes sendet, fängt die Station selbst damit an. Ist allerdings der Kanal bereits belegt, dann überprüft ihn die Station nicht andauernd mit dem Hintergedanken, ihn sofort an sich zu reißen, sobald das Ende der vorherigen Übertragung erkannt wird. Stattdessen wird eine zufällige Zeitspanne gewartet und der Vorgang dann wiederholt. Dieser Algorithmus führt zu einer besseren Kanalauslastung, aber längeren Wartezeiten als beim 1-persistenten CSMA.

Eine weitere Protokollvariante ist **p-persistentes CSMA**. Dieses Protokoll kann auf Kanäle mit Zeitscheiben angewandt werden und funktioniert wie folgt: Wenn eine Station senden will, überprüft sie den Kanal. Ist er frei, sendet die Station mit einer Wahrscheinlichkeit p . Mit einer Wahrscheinlichkeit von $q=1-p$ wartet sie bis zur nächsten Zeitscheibe. Ist diese Zeitscheibe auch frei, wird entweder gesendet oder gewartet, wiederum mit den Wahrscheinlichkeiten p und q . Dieser Vorgang wird wiederholt, bis entweder der Rahmen übertragen worden ist oder eine andere Station zu senden begonnen hat. Im letzteren Fall reagiert die glücklose Station, als ob eine Kollision stattgefunden hätte (d.h. eine zufällige Zeitspanne warten und dann alles von vorn beginnen). Wenn die Station erkennt, dass der Kanal belegt ist, wartet sie bis zur nächsten Zeitscheibe und wendet dann das obige Verfahren an. IEEE 802.11 benutzt eine Verfeinerung von p-persistentem CSMA, welches wir in Abschnitt 4.4 besprechen.

► Abbildung 4.4 zeigt den berechneten Durchsatz in Abhängigkeit vom anfallenden Datenverkehr für alle drei Protokolle sowie für reines ALOHA und S-ALOHA.

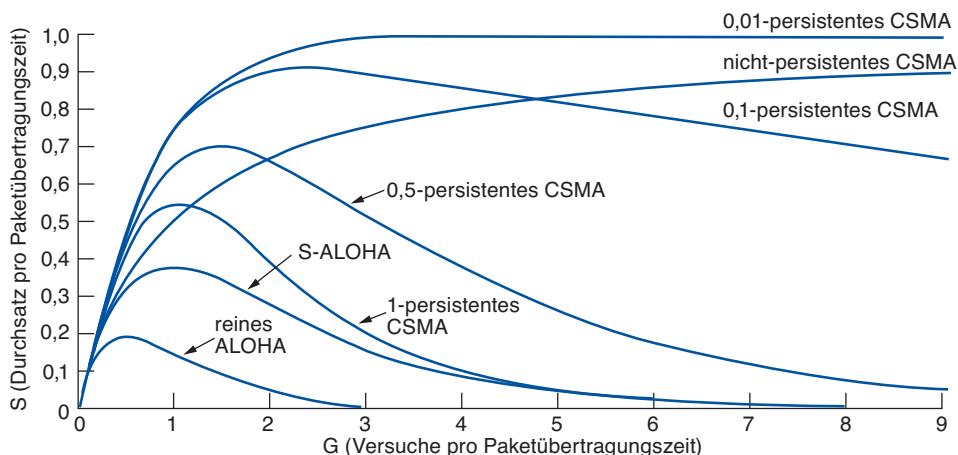


Abbildung 4.4: Vergleich der Kanalauslastung in Abhängigkeit vom Datenverkehr bei verschiedenen zufallsgesteuerten Protokollen.

CSMA mit Kollisionserkennung

Persistente und nicht persistente CSMA-Protokolle sind sicher eine Verbesserung gegenüber ALOHA, da sie gewährleisten, dass keine Station sendet, während der Kanal belegt ist. Wenn jedoch zwei Stationen den Kanal als frei erkennen und beide gleichzeitig zu senden beginnen, werden ihre Signale immer noch kollidieren. Eine weitere Verbesserung ist, dass die Station die Kollision schnell entdecken und die Übertragung abrupt abbrechen kann (statt sie normal zu beenden), da diese ohnehin irreparabel verstümmelt wäre. Diese Strategie spart Zeit und Bandbreite.

Dieses Protokoll heißt **CSMA/CD** (*Carrier Sense Multiple Access with Collision Detection*, Mehrfachzugang mit Trägerprüfung und Kollisionserkennung) und ist die Grundlage für das klassische Ethernet-LAN, sodass es hier durchaus eine nähere Betrachtung verdient. Es ist wichtig sich zu vergegenwärtigen, dass Kollisionserkennung ein analoger Prozess ist. Die Hardware der Station muss den Kanal während der Übertragung abhören. Wenn das Signal, das zurückkommt, sich von dem Signal unterscheidet, das ursprünglich auf den Kanal gelegt wurde, dann weiß die Station, dass eine Kollision aufgetreten ist. Dies impliziert, dass ein empfangenes Signal im Vergleich zum gesendeten Signal nicht winzig sein muss (was bei kabellosen Kanälen schwierig ist, da empfangene Signale 1 000 000-mal schwächer als gesendete Signale sein können) und dass die Modulation so gewählt werden muss, um Kollisionserkennung zu ermöglichen (z.B. könnte es gut sein, dass es unmöglich ist, eine Kollision von zwei 0-Volt-Signalen zu entdecken).

CSMA/CD verwendet genau wie viele andere LAN-Protokolle das Konzept aus ►Abbildung 4.5. Zum Zeitpunkt t_0 hat eine Station die Übertragung ihres Rahmens beendet. Jede andere Station, die einen Rahmen zu senden hat, kann das jetzt versuchen. Wenn sich zwei oder mehr Stationen gleichzeitig zum Senden entschließen, kommt es zu einer Kollision. Wenn eine Station die Kollision erkennt, unterbricht sie ihre Übertragung, wartet eine zufällige Zeitspanne und versucht es dann erneut (unter der Annahme, dass inzwischen keine andere Station zu übertragen begonnen hat). Deshalb besteht unser CSMA/CD-Modell aus abwechselnden Konkurrenz- und Übertragungsperioden, wobei Leerlauf entsteht, sobald keine Station sendet.

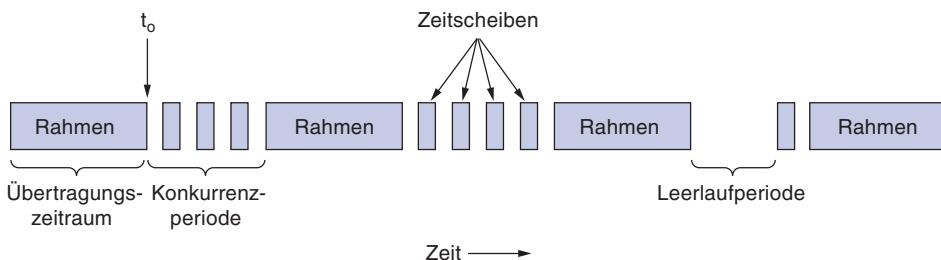


Abbildung 4.5: CSMA/CD kann sich im Konkurrenz-, Übertragungs- oder Leerlaufzustand befinden.

Wir wollen nun die Einzelheiten des Konkurrenzalgorithmus genauer untersuchen. Angenommen, zwei Stationen beginnen genau zur Zeit t_0 mit der Übertragung. Wie lange brauchen sie um zu erkennen, dass eine Kollision stattgefunden hat? Die Beant-

wortung dieser Frage ist grundlegend für die Ermittlung der Konkurrenzperiode, aus der Verzögerung und Datendurchsatz bestimmt werden.

Die minimale Zeit bis zur Entdeckung einer Kollision ist die Zeit, die ein Signal braucht, um von einer Station zur anderen zu gelangen. Aufgrund dieser Information könnte man folgern, dass eine Station, die seit Beginn der Übertragung für die Dauer der Signalausbreitung über das Kabel hinweg keine Kollision erkannt hat, sicher sein kann, sie verfüge über das Kabel. Mit „verfügen“ ist gemeint, dass alle anderen Stationen die Übertragung erkannt haben und demzufolge nicht stören. Diese Schlussfolgerung ist falsch.

Stellen Sie sich als schlechtesten Fall folgendes Szenario vor: Angenommen, τ ist die Zeit, die ein Signal für die Ausbreitung zwischen den beiden am weitesten entfernten Stationen benötigt. Bei t_0 beginnt eine Station mit der Übertragung. Zur Zeit $t_0 + \tau - \epsilon$, kurz bevor das Signal die entfernteste Station erreicht, beginnt auch diese zu senden. Natürlich entdeckt sie die Kollision fast im gleichen Augenblick und bricht ab, aber das kurze Signal, das durch die Kollision erzeugt wurde, gelangt zur ursprünglichen Station nicht vor der Zeit $2\tau - \epsilon$ zurück. Das bedeutet, dass eine Station im schlechtesten Fall nicht sicher sein kann, den Kanal zu belegen, bevor sie 2τ lang gesendet hat, ohne eine Kollision festzustellen.

Mit diesem Verständnis können wir uns CSMA/CD-Konkurrenz als S-ALOHA-System mit einer Zeitscheibengröße von 2τ vorstellen. In einem 1 km langen Koaxialkabel gilt $\tau \approx 5 \mu s$. Der Unterschied zwischen CSMA/CD und S-ALOHA ist, dass Zeitscheiben, in denen nur eine Station überträgt (d.h., in denen der Kanal belegt ist), vom Rest eines Rahmens gefolgt werden. Dieser Unterschied verbessert die Performanz bedeutend, falls die Rahmenzeit sehr viel länger als die Übertragungszeit ist.

4.2.3 Kollisionsfreie Protokolle

Obwohl Kollisionen bei CSMA/CD nicht mehr entstehen, nachdem eine Station eindeutig den Kanal in Besitz hält, können sie während der Konkurrenzperiode immer noch auftreten. Diese Kollisionen beeinflussen die Leistungsfähigkeit des Systems im negativen Sinn, vor allem wenn das Bandbreite-Verzögerung-Produkt sehr groß ist, z.B. wenn das Kabel lang ist (d.h. ein großer τ -Wert) und kurze Rahmen übertragen werden. Kollisionen verringern nicht nur die Bandbreite, darüber hinaus bewirken sie, dass die Zeit zum Senden eines Rahmens variiert, was für Echtzeitdatenverkehr wie VoIP keine gute Eigenschaft ist. CSMA/CD ist außerdem nicht universell einsetzbar.

In diesem Abschnitt werden wir einige Protokolle untersuchen, die die Konkurrenz um den Kanal auf eine Art lösen, bei der keine Kollisionen auftreten, nicht einmal während der Konkurrenzperiode. Die meisten dieser Protokolle werden derzeit nicht in praktisch bedeutsamen Systemen verwendet. Aber in einem sich sehr schnell wandelnden Bereich ist es oft sehr vorteilhaft, Protokolle mit exzellenten Eigenschaften für zukünftige Systeme bereits an der Hand zu haben.

Bei allen beschriebenen Protokollen nehmen wir an, dass es N Stationen gibt, jede mit einer eindeutigen Adresse von 0 bis $N-1$ programmiert. Es spielt keine Rolle, dass einige Stationen ab und zu inaktiv sind. Wir gehen weiter davon aus, dass die Signalausbreitungsverzögerung vernachlässigbar ist. Die Hauptfrage bleibt: Welche Station erhält nach einer erfolgreichen Übertragung den Kanal? Wir benutzen weiterhin das Modell von Abbildung 4.5 mit den diskreten Zeitscheiben für konkurrierende Übertragungsversuche.

Bitmusterprotokoll

In unserem ersten kollisionsfreien Protokoll, der **Bitmuster-Methode** (*basic bit-map method*), besteht jede Konkurrenzperiode aus genau N Zeitscheiben. Wenn Station 0 einen Rahmen senden will, überträgt sie in Zeitscheibe 0 ein 1-Bit. Keine andere Station darf in dieser Zeitscheibe übertragen. Unabhängig davon, was Station 0 macht, erhält Station 1 die Möglichkeit, in Zeitscheibe 1 ein 1-Bit zu übertragen, aber nur wenn sie einen zur Übertragung bereitstehenden Rahmens hat. Allgemein kann Station j die Tatsache eines zur Übertragung anstehenden Rahmens dadurch anzeigen, dass sie in Zeitscheibe j eine 1 einfügt. Nachdem alle N Zeitscheiben durchlaufen sind, hat jede Station den genauen Überblick, welche Stationen übertragen wollen. Jetzt beginnen sie der Reihe nach Rahmen zu senden (►Abbildung 4.6).

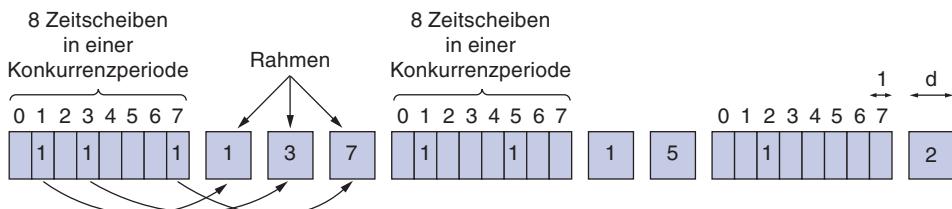


Abbildung 4.6: Das Bitmusterprotokoll.

Da sich bei jeder Übertragung alle darüber einigen, wer als Nächstes an die Reihe kommt, entstehen nie Kollisionen. Nachdem die letzte bereite Station ihren Rahmen übertragen hat, was alle Stationen leicht mitverfolgen können, beginnt eine weitere N -Bit-Konkurrenzperiode. Wenn eine Station erst kurz nach Ablauf ihrer Bitscheibe sendebereit wird, hat sie Pech gehabt und muss warten, bis jeder an der Reihe war und das Bitmuster wieder die Runde gemacht hat. Protokolle wie dieses, bei dem jeder Übertragungswunsch vor der tatsächlichen Übertragung allen Stationen bekannt gegeben wird, nennt man **Reservierungsprotokolle** (*reservation protocol*), weil sie Kanalbesitz im Voraus reservieren und Kollisionen verhindern. Nun wollen wir kurz die Leistungsfähigkeit dieses Protokolls analysieren. Der Einfachheit halber messen wir die Zeit in Einheiten der Größe einer Zeitscheibe in der Konkurrenzperiode, wobei Datenrahmen aus d Zeiteinheiten bestehen.

Im Zustand geringer Belastung wird einfach mangels Datenrahmen das Bitmuster immer wiederholt. Wir betrachten die Situation aus Sicht einer Station mit niedriger Nummer, z.B. 0 oder 1. Wenn sie senden will, wird die „aktuelle“ Zeitscheibe meist irgendwo in der Mitte des Bitmusters sein. Im Durchschnitt muss die Station $N/2$ Zeit-

scheiben warten, bis die aktuelle Abfrage vorbei ist, und weitere N Zeitscheiben bis zur Beendigung der nächsten Abfrage, bevor übertragen werden kann.

Die Aussichten für Stationen mit höheren Nummern sind besser. Normalerweise müssen diese, bevor sie mit einer Übertragung beginnen können, nur für die Dauer einer halben Prüfung ($N/2$ Bitscheiben) warten. Stationen mit höheren Nummern müssen selten auf die nächste Abfrage warten. Da Stationen mit niedrigen Nummern durchschnittlich $1,5N$ Zeitscheiben und solche mit hoher Nummer durchschnittlich $0,5N$ Zeitscheiben warten müssen, beträgt das Mittel für alle Stationen N Zeitscheiben.

Die Effizienz des Kanals bei niedriger Auslastung ist leicht zu berechnen. Der zusätzliche Platzbedarf pro Rahmen beträgt N Bit. Bei einer Datenmenge von d Bit ergibt sich eine Effizienz von $d/(N+d)$.

Bei hoher Belastung, wenn alle Stationen andauernd senden wollen, werden die N Bit der Konkurrenzperiode auf N Rahmen aufgeteilt. Dadurch erhöht sich der Platzbedarf nur um ein Bit pro Rahmen, was zu einer Effizienz von $d/(d+1)$ führt. Die durchschnittliche Übertragungsverzögerung eines Rahmens ist gleich der Zeit, die er innerhalb der Station in Warteposition steht, zuzüglich $(N-1)d+N$, wenn er einmal den Kopf der internen Warteschlange erreicht hat. Diese Zeitspanne gibt an, wie lange gewartet werden muss, bis alle anderen Stationen an der Reihe waren und einen Rahmen gesendet haben und ein weiteres Bitmuster durchgelaufen ist.

Tokenweitergabe

Das Wesentlich am Bitmusterprotokoll ist, dass es jeder Station der Reihe nach die Übertragung eines Rahmens in einer vorgegebenen Reihenfolge ermöglicht. Dasselbe kann auch auf einem anderen Weg erreicht werden, und zwar indem eine kleine Nachricht, ein **Token**, von einer Station zur nächsten in der gleichen Reihenfolge übergeben wird. Das Token repräsentiert die Erlaubnis zum Senden. Falls eine Station zu dem Zeitpunkt, wenn es das Token erhält, einen Rahmen in ihrer Warteschlange zur Übertragung bereit hat, dann kann die Station diesen Rahmen senden, bevor das Token zur nächsten Station weitergegeben wird. Falls kein Rahmen in der Warteschlange ist, wird das Token einfach weitergegeben.

Bei einem **Token-Ring**-Protokoll wird die Topologie des Netzwerks benutzt, um die Reihenfolge festzulegen, in der die Stationen senden. Die Stationen sind jeweils mit der nächsten zu einem Ring verbunden. Die Weitergabe des Token zur nächsten Station besteht dann einfach darin, das Token aus der einen Richtung zu empfangen und es in die andere Richtung weiterzusenden (►Abbildung 4.7). Die Übermittlung der Rahmen findet ebenfalls in der Richtung des Token statt. Auf diese Art kreisen die Rahmen auf dem Ring herum und erreichen immer ihre Zielstation. Um zu verhindern, dass Rahmen (bzw. das Token) unendlich im Ring kreisen, muss irgendeine Station den Rahmen wieder vom Ring entfernen. Diese Station könnte entweder diejenige sein, die den Rahmen ursprünglich ausgesandt hat, nachdem er einen vollständigen Zyklus durchlaufen hat, oder die Station, die der beabsichtigte Empfänger des Rahmens war.

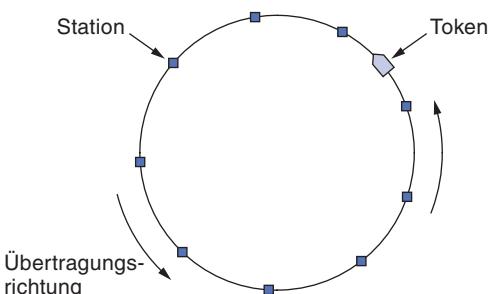


Abbildung 4.7: Token-Ring.

Beachten Sie, dass wir keinen physikalischen Ring benötigen, um Tokenweitergabe zu implementieren. Der Kanal, der die Stationen verbindet, könnte stattdessen ein einzelner langer Bus sein. Jede Station benutzt dann den Bus, um das Token in der vorgegebenen Reihenfolge an die nächste Station zu senden. Der Besitz des Token erlaubt wie vorher einer Station, den Bus zu benutzen, um einen Rahmen zu senden. Dieses Protokoll wird **Token-Bus** genannt.

Die Performanz der Tokenweiterleitung ähnelt der vom Bitmusterprotokoll, obwohl sich die Konkurrenzzeitscheiben und -rahmen eines Zyklus nun vermischen. Nach dem Senden eines Rahmens muss jede Station so lange warten, bis alle N Stationen (einschließlich der Station selbst) das Token an ihre Nachbarn weitergeleitet und die anderen $N-1$ Stationen ihren Rahmen – falls vorhanden – übertragen haben. Es gibt aber einen feinen Unterschied: Da alle Positionen im Kreis gleichwertig sind, werden keine Stationen aufgrund ihrer niedrigen bzw. hohen Nummerierung bevorzugt. Bei einem Token-Ring sendet jede Station außerdem das Token nur bis zur nächsten Station, bevor das Protokoll einen Schritt weiter geht. Es muss also nicht jedes Token zu allen Stationen weitergeleitet werden, bevor das Protokoll mit dem nächsten Schritt fortfährt.

Token-Ringe tauchten als MAC-Protokolle mit einer gewissen Beständigkeit auf. Ein frühes Token-Ring-Protokoll (unter dem Namen „Token Ring“ als IEEE 802.5 standardisiert) war in den 1980er Jahren als Alternative zum klassischen Ethernet weitverbreitet. In den 1990er Jahren wurde ein viel schnellerer Token-Ring mit Namen **FDDI** (*Fiber Distributed Data Interface*) von Switched Ethernet geschlagen. In den 2000er Jahren wurde ein Token-Ring namens **RPR** (*Resilient Packet Ring*) als IEEE 802.17 definiert, um die Mischung der MAN-Ringe, die von ISPs benutzt werden, zu standardisieren. Wir sind gespannt, was die 2010er Jahre zu bieten haben.

Binärer Countdown

Eine Schwierigkeit beim normalen Bitmusterprotokoll – und infolgedessen auch bei der Tokenweiterleitung – ist, dass der Overhead (d.h. der zusätzliche Aufwand) pro Station 1 Bit beträgt, sodass dies nicht gut auf Netze mit Tausenden von Stationen angepasst werden kann. Durch Verwendung von binären Stationsadressen mit einem Kanal, der Übertragungen kombiniert, lässt sich ein besseres Ergebnis erzielen. Eine Station, die den Kanal benutzen will, sendet dabei ihre Adresse als Binärzeichenfolge,

beginnend mit dem höchstwertigen Bit, an alle anderen Stationen. Bei allen Adressen wird von der gleichen Länge ausgegangen. Die Bits an derselben Adressposition der verschiedenen Stationen werden vom Kanal mittels booleschem ODER verknüpft, wenn sie zur selben Zeit gesendet werden. Man nennt dieses Protokoll **binärer Countdown** (*binary countdown*). Es wurde in Datakit (Fraser, 1987) verwendet. Man geht implizit davon aus, dass die Übertragungsverzögerungen vernachlässigbar sind, sodass alle Stationen die Bits unmittelbar wahrnehmen.

Um Konflikte zu vermeiden, muss eine Schiedsregel verwendet werden: Sobald eine Station erkennt, dass ein hochwertiges Bit, das in ihrer eigenen Adresse 0 ist, mit 1 überschrieben wurde, gibt sie auf. Versuchen z.B. die Stationen 0010, 0100, 1001 und 1010 gleichzeitig, den Kanal zu belegen, übertragen die Stationen in der ersten Bitzeit 0, 0, 1 und 1. Daraus wird mit ODER eine 1 gebildet. Die Stationen 0010 und 0100 erkennen an der 1, dass sich eine Station mit einer höheren Nummer um den Kanal bewirbt, und geben in dieser Runde den Kampf auf, während die Stationen 1001 und 1010 fortfahren.

Das nächste Bit ist 0 und beide Stationen fahren fort. Das nächste Bit ist 1, somit gibt Station 1001 auf. Siegerin ist Station 1010, weil sie die höchste Adresse hat. Sie kann nun einen Rahmen übertragen. Wenn sie fertig ist, beginnt der nächste Zyklus. Dieses Protokoll ist in ► Abbildung 4.8 dargestellt. Es weist die Eigenschaft auf, dass Stationen mit höheren Bitwerten eine höhere Priorität haben als die Stationen mit niedrigeren Werten, was je nach Kontext gut oder schlecht sein kann.

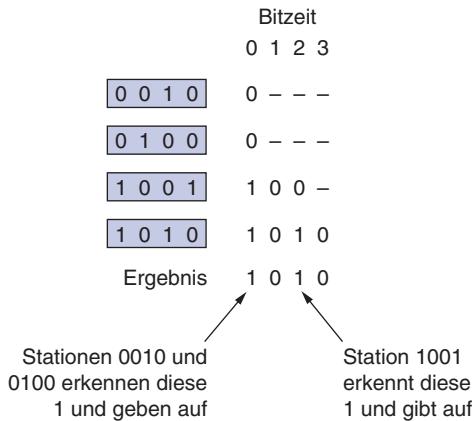


Abbildung 4.8: Das binäre Countdown-Protokoll. Ein Strich bedeutet Ruhe.

Die Effizienz des Kanals ist bei dieser Methode $d/(d+\log_2 N)$. Wird das Rahmenformat so klug gewählt, dass die Adresse des Senders das erste Feld im Rahmen ist, dann werden auch diese $\log_2 N$ Bits nicht verschwendet und die Effizienz beträgt 100 %.

Der binäre Countdown ist ein Beispiel für ein einfaches, elegantes und effizientes Protokoll, das wiederentdeckt werden muss. Hoffen wir, dass es eines Tages ein neues Zuhause finden wird.

4.2.4 Protokolle mit eingeschränkter Konkurrenz

Wir haben jetzt zwei Grundstrategien für die Inbesitznahme eines Kanals in einem Broadcast-Netz betrachtet: auf konkurrierenden Zugriffsversuchen basierende Methoden wie CSMA (Konkurrenzmethoden) und kollisionsfreie Methoden. Jede Strategie kann im Hinblick auf zwei wichtige Leistungsmerkmale bewertet werden: die Übertragungsverzögerung bei geringer Auslastung und die Effizienz bei hoher Auslastung. Bei geringer Belastung ist die Konkurrenzmethode (reines ALOHA oder S-ALOHA) aufgrund der kurzen Wartezeiten vorzuziehen (da Kollisionen selten sind). Wenn die Belastung steigt, wird Konkurrenz immer unattraktiver, da der Overhead, der durch die Kanalzuteilung erzeugt wird, ebenfalls steigt. Bei kollisionsfreien Protokollen ist genau das Gegenteil der Fall. Bei niedriger Auslastung erzeugen sie relativ hohe Verzögerungen. Nimmt die Auslastung aber zu, steigt die Kanaleffizienz (da der Overhead fest ist).

Jetzt wäre es natürlich nett, die besten Eigenschaften von Konkurrenz- und kollisionsfreien Protokollen in einem neuen Protokoll zu vereinen, das bei niedriger Auslastung das Konkurrenzverfahren nutzt, um kurze Wartezeiten zu gewährleisten, bei hoher Auslastung aber eine kollisionsfreie Technik anwendet, um hohe Kanaleffizienz zu garantieren. Solche Protokolle gibt es tatsächlich, sie heißen **Protokolle mit eingeschränkter Konkurrenz** (*limited-contention protocol*), mit denen wir unsere Betrachtung der Netze mit Trägerprüfung abschließen wollen.

Alle bisher beschriebenen Konkurrenzprotokolle sind symmetrisch, d.h., jede Station versucht mit einer Wahrscheinlichkeit von p den Kanal zu nutzen, wobei p für alle Stationen gleich ist. Interessanterweise kann die Leistungsfähigkeit des gesamten Systems zuweilen verbessert werden, wenn ein Protokoll benutzt wird, das den verschiedenen Stationen unterschiedliche Wahrscheinlichkeiten zuteilt.

Bevor wir uns den asymmetrischen Protokollen zuwenden, wollen wir noch einmal kurz die Leistungsmerkmale der symmetrischen Verfahren betrachten. Angenommen, k Stationen bewerben sich um den Zugriff auf den Kanal. Jede überträgt mit der Wahrscheinlichkeit p Daten in einer Zeitscheibe. Die Wahrscheinlichkeit, dass eine Station den Kanal während einer Zeitscheibe erfolgreich belegt, entspricht der Wahrscheinlichkeit, dass irgendeine Station mit Wahrscheinlichkeit p überträgt und alle anderen $k-1$ Stationen ihre Übertragung jeweils mit Wahrscheinlichkeit $1-p$ zurückstellen. Dieser Wert ist somit $kp(1-p)^{k-1}$. Um den optimalen Wert für p zu ermitteln, differenzieren wir nach p . Die erhaltene Ableitung setzen wir gleich 0 und lösen sie nach p auf. Dadurch finden wir heraus, dass der beste Wert von p $1/k$ beträgt. Setzt man $p=1/k$, erhält man

$$\Pr[\text{Erfolg bei optimalem } p] = \left[\frac{k-1}{k} \right]^{k-1} \quad (4.4)$$

Diese Wahrscheinlichkeit ist in ▶ Abbildung 4.9 dargestellt. Bei nur wenigen Stationen sind die Erfolgschancen hoch, aber sobald die Anzahl der Stationen auch nur fünf beträgt, sinkt die Wahrscheinlichkeit fast auf den asymptotischen Wert $1/e$.

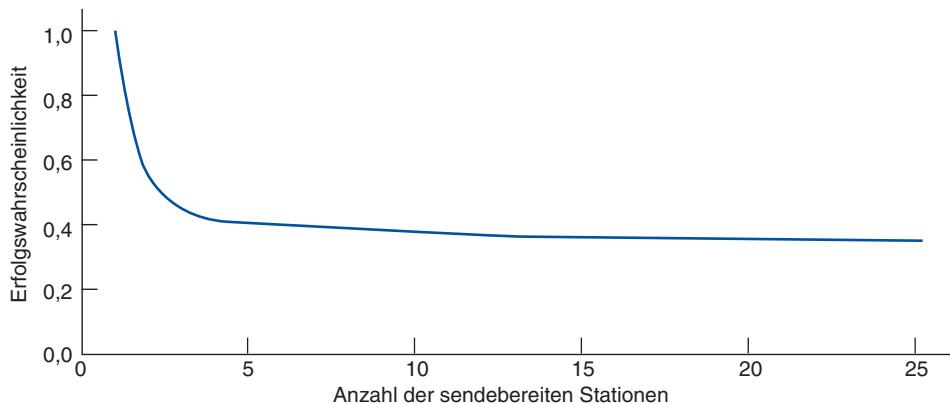


Abbildung 4.9: Belegungswahrscheinlichkeit für einen symmetrischen Konkurrenzkanal.

Aus Abbildung 4.9 ist gut zu erkennen, dass die Wahrscheinlichkeit, dass eine Station einen Kanal belegt, nur erhöht werden kann, wenn der Wettbewerb reduziert wird. Genau das machen Protokolle mit eingeschränkter Konkurrenz. Zuerst unterteilen sie die Stationen in (nicht unbedingt disjunkte) Gruppen. Nur die Mitglieder der Gruppe 0 können um Zeitscheibe 0 konkurrieren. Wer durchkommt, besetzt den Kanal und überträgt seinen Rahmen. Wenn die Zeitscheibe ungenutzt ist oder eine Kollision eintritt, konkurrieren die Mitglieder von Gruppe 1 um Zeitscheibe 1 usw. Indem eine Teilung der Stationen in Gruppen durchgeführt wird, kann die Konkurrenz um jede einzelne Zeitscheibe vermindert werden. Daher wird jede Zeitscheibe wie am linken Ende des Graphen in Abbildung 4.9 genutzt.

Der Trick dabei ist, wie man die Stationen den Zeitscheiben zuordnet. Bevor wir den allgemeinen Fall betrachten, sehen wir uns erst einige Spezialfälle an. Im Extremfall hat jede Gruppe nicht mehr als ein Mitglied. Diese Zuordnung garantiert, dass nie Kollisionen auftreten, sondern sich nur eine Station um eine Zeitscheibe bewirbt. Diese Art von Protokoll kennen wir bereits (z.B. den binären Countdown). Ein weiterer Sonderfall ist die Zusammenfassung von je zwei Stationen zu einer Gruppe. Die Wahrscheinlichkeit, dass beide innerhalb einer Zeitscheibe übertragen wollen, ist p^2 , was bei einem kleinen Wert von p vernachlässigt werden kann. Wenn der gleichen Zeitscheibe immer mehr Stationen zugewiesen werden, steigt die Wahrscheinlichkeit einer Kollision, aber gleichzeitig schrumpft die Länge des Bitmusters, das abgefragt werden muss, um jedem eine Chance zu geben. Der Grenzfall ist eine einzelne Gruppe, die alle Stationen enthält (d.h. S-ALOHA). Wir brauchen nun eine Lösung, um die Stationen dynamisch zuzuweisen, mit vielen Stationen pro Zeitscheibe bei niedriger Auslastung und wenigen Stationen (oder gar nur einer) pro Zeitscheibe bei hoher Auslastung.

Das adaptive Tree-Walk-Protokoll

Eine besonders einfache Art der erforderlichen Zuweisung ist der Algorithmus, den sich die US-amerikanische Armee ausdachte, um im zweiten Weltkrieg Soldaten auf Syphilis zu untersuchen (Dorfman, 1943). Hier das Verfahren kurzgefasst: die Armee nahm Blutproben von einer Stichprobe aus N Soldaten. Ein Teil jeder Stichprobe

wurde in ein Teströhrchen eingefüllt. Diese gemischte Probe wurde auf Antikörper getestet. Wurden keine gefunden, so waren alle Soldaten gesund. Waren Antikörper vorhanden, dann wurden zwei neue gemischte Stichproben erstellt, eine von den Soldaten 1 bis $N/2$ und eine vom Rest. Der Prozess wurde rekursiv wiederholt, bis die infizierten Soldaten ermittelt waren.

Für die Computerversion dieses Algorithmus (Capetanakis, 1979) ist es zweckmäßig, sich die Stationen in einem binären Baum angeordnet vorzustellen, wie in ►Abbildung 4.10 gezeigt. In der ersten konkurrierenden Zeitscheibe nach einer erfolgreichen Rahmenübertragung (Zeitscheibe 0) dürfen alle Stationen versuchen, sich den Kanal anzueignen. Wenn es einer gelingt, gut. Wenn eine Kollision auftritt, dann dürfen nur diejenigen Stationen in Zeitscheibe 1 konkurrieren, die im Baum zu Knoten 2 gehören. Belegt eine davon den Kanal, wird der Zeitscheibe, die auf den Rahmen folgt, für die Stationen unter Knoten 3 reserviert. Möchten andererseits zwei oder mehr Stationen unter Knoten 2 übertragen, gibt es in Zeitscheibe 1 eine Kollision. In diesem Fall kommt dann Knoten 4 in Zeitscheibe 2 an die Reihe.

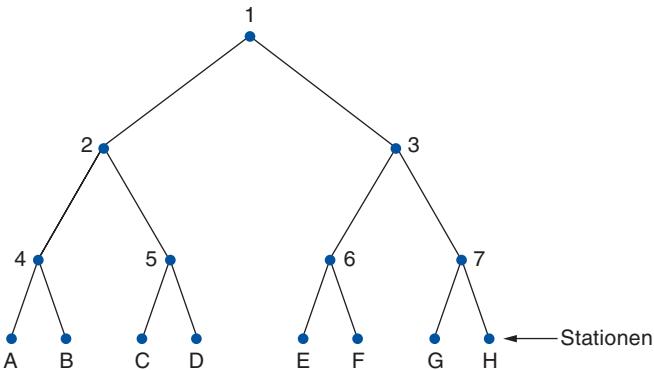


Abbildung 4.10: Ein Baum mit acht Stationen.

Im Wesentlichen passiert also Folgendes: wenn während Zeitscheibe 0 eine Kollision auftritt, wird der ganze Baum in vertikaler Richtung (Tiefe des Baumes) durchsucht, um alle sendebereiten Stationen zu ermitteln. Jede Zeitscheibe wird durch einen bestimmten Knoten im Baum dargestellt. Wenn eine Kollision auftritt, geht die Suche rekursiv mit dem linken und dann mit dem rechten Kindknoten dieses Knotens weiter. Ist eine Zeitscheibe im Leerlaufzustand oder überträgt nur eine Station darin, so kann die Suche nach diesem Knoten abgebrochen werden, weil dann alle sendebereiten Stationen entdeckt wurden (wären es mehr als eine, hätte eine Kollision stattgefunden).

Wenn das System stark ausgelastet ist, lohnt es sich kaum, Zeitscheibe 0 dem Knoten 1 zuzuweisen, da dies nur in der unwahrscheinlichen Situation sinnvoll ist, dass genau eine Station einen Rahmen zu übertragen hat. Ebenso könnte man argumentieren, dass die Knoten 2 und 3 aus dem gleichen Grund übersprungen werden sollten. Etwas allgemeiner ausgedrückt: Auf welcher Ebene des Baumes sollte die Suche beginnen? Je stärker die Auslastung, desto tiefer im Baum sollte die Suche natürlich beginnen. Wir neh-

men an, dass anhand der Überwachung des jüngsten Datenverkehrs jede Station über eine gute Schätzung der Anzahl q der sendebereiten Stationen verfügt.

Wir nummerieren nun die Ebenen des Baumes von oben herab, wobei Knoten 1 in Abbildung 4.10 auf Ebene 0 liegt, die Knoten 2 und 3 auf Ebene 1 usw. Beachten Sie, dass jeder Knoten auf der Ebene i noch den Bruchteil 2^{-i} der Stationen unter sich hat. Falls q sendebereite Stationen gleichmäßig im Baum verteilt sind, ist ihre erwartete Anzahl unter einem bestimmten Knoten der Ebene i genau $2^{-i}q$. Intuitiv würden wir die Ebene, ab der der Baum am besten durchsucht werden kann, dort erwarten, wo die durchschnittliche Anzahl der konkurrierenden Stationen pro Zeitscheibe gleich 1 ist, also die Ebene, auf der $2^{-i}q=1$ gilt. Die Lösung dieser Gleichung ergibt $i=\log_2 q$.

Zu diesem Grundalgorithmus wurden viele Verbesserungen erarbeitet und von Bertsekas und Gallagher (1992) beschrieben. Stellen Sie sich z.B. vor, dass nur die Stationen G und H senden möchten. Am Knoten 1 findet in dem Fall eine Kollision statt; also wird 2 getestet und im Leerlaufzustand vorgefunden. Wir müssen nun Knoten 3 nicht mehr testen, da hier auf jeden Fall eine Kollision stattfindet (wir wissen, dass unterhalb von 1 zwei oder mehr Stationen sendebereit sind und keine davon unter 2 liegt, also müssen alle unter 3 liegen). Die Abfrage von 3 kann übersprungen und bei 6 fortgesetzt werden. Falls diese Abfrage auch nichts bringt, kann wiederum 7 übersprungen und als Nächstes Knoten G überprüft werden.

4.2.5 Protokolle für drahtlose LANs

Ein System von Laptoprechnern, die über Funk miteinander kommunizieren, können als ein drahtloses LAN angesehen werden, wie wir es in Abschnitt 1.5.3 besprochen haben. Solch ein LAN ist ein Beispiel für einen Broadcast-Kanal. Es hat auch etwas andere Eigenschaften als ein verkabeltes LAN, was zu unterschiedlichen MAC-Protokollen führt. In diesem Abschnitt werden wir einige dieser Protokolle untersuchen. In Abschnitt 4.4 sehen wir uns IEEE 802.11 (WiFi) detailliert an.

Eine übliche Konfiguration für ein drahtloses LAN ist ein Bürogebäude mit Zugangspunkten (*Access Point, AP*), die strategisch überall im Gebäude verteilt aufgestellt sind. Die Zugangspunkte sind untereinander über Kupferkabel oder Glasfaser verbunden und stellen eine Verbindung zu den Stationen her, die mit ihnen kommunizieren. Wird die Übertragungsleistung der Zugangspunkte und Laptops so eingestellt, dass diese eine Reichweite von vielen Metern haben, so werden nahe gelegene Räume zu einzelnen Zellen und das Gebäude wird zu einem Zellulärtelefonsystem, wie wir in *Kapitel 2* beschrieben haben, mit dem Unterschied, dass jede Zelle nur einen Kanal hat. Dieser Kanal wird von allen Stationen innerhalb der Zelle gemeinsam benutzt, einschließlich dem Zugangspunkt. In der Regel wird Bandbreite im Bereich von Megabit/s zur Verfügung gestellt, was bis zu 600 Mbit/s reichen kann.

Wir haben bereits angemerkt, dass drahtlose Systeme normalerweise eine Kollision, während sie passiert, nicht erkennen können. Das an der Station empfangene Signal ist möglicherweise verschwindend klein, vielleicht eine Million Mal schwächer als

das Signal, das übermittelt wurde. Dieses Signal zu finden ist wie die Suche nach der Stecknadel im Heuhaufen. Stattdessen werden Bestätigungen benutzt, um Kollisionen und Folgefehler zu entdecken.

Es gibt einen sogar noch wichtigeren Unterschied zwischen drahtlosen und verkabelten LANs. Eine Station eines drahtlosen LAN ist möglicherweise aufgrund der begrenzten Funkreichweite der Stationen nicht in der Lage, Rahmen an alle anderen Stationen zu übertragen bzw. von allen anderen Stationen zu empfangen. Wenn bei verkabelten LANs eine Station einen Rahmen sendet, empfangen diesen alle anderen Stationen. Das Fehlen dieser Eigenschaft in drahtlosen LANs verursacht eine Reihe von Komplikationen.

Wir gehen von der vereinfachten Annahme aus, dass jeder Funksender eine feste Reichweite hat, die durch einen kreisförmigen Abdeckungsbereich dargestellt wird, innerhalb dessen eine andere Station die Übertragungen der Station abhören und empfangen kann. Es ist wichtig sich klarzumachen, dass reale Abdeckungsbereiche nicht annähernd so regelmäßig sind, weil die Weiterleitung von Funksignalen von der Umgebung abhängt. Wände und andere Hindernisse, die Signale dämpfen und reflektieren, können dazu führen, dass sich der Bereich in verschiedenen Richtungen deutlich unterscheidet. Doch für unsere Zwecke reicht die Vorstellung von einem einfachen Kreis.

Ein naiver Ansatz für ein drahtloses LAN könnte die Verwendung von CSMA sein: Man hört einfach auf andere Übertragungen und sendet nur, wenn kein anderer sendet. Die Schwierigkeit dabei ist, dass dieses Protokoll für drahtlose Netze nicht wirklich geeignet ist, weil die Interferenzen beim Empfänger und nicht beim Sender für den Empfang wichtig sind. Um die Natur des Problems aufzuzeigen, betrachten wir ▶Abbildung 4.11 mit vier drahtlosen Stationen. Für unsere Zwecke spielt es keine Rolle, was Basisstationen und was Laptops sind. Der Funkbereich ist so angelegt, dass sich *A* und *B* in der gegenseitigen Reichweite befinden und einander potenziell stören können. *C* kann auch *B* und *D*, nicht aber *A* stören.

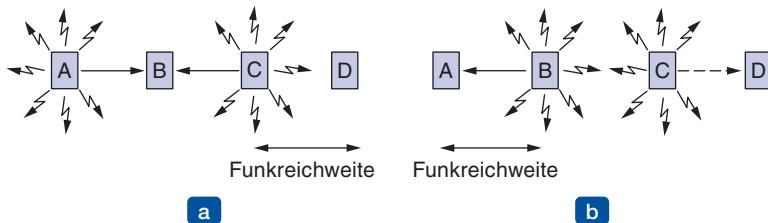


Abbildung 4.11: Drahtloses LAN. (a) *A* und *C* sind bei der Übertragung zu *B* versteckte Terminals (*hidden terminal*)
(b) *B* und *C* sind zu weit entfernte Terminals (*exposed terminal*) bei der Übertragung von *A* und *D*.

Zunächst betrachten wir, was passiert, wenn *A* und *C* an *B* übertragen, wie in ▶Abbildung 4.11a dargestellt. Wenn *A* sendet und dann Station *C* sofort das Medium prüft, kann *C* Station *A* nicht hören, da sich *A* außerhalb des Funkbereichs befindet. *C* nimmt also fälschlicherweise an, dass mit der Übertragung an *B* begonnen werden kann. Beginnt *C* mit der Übertragung, so werden Überlagerungen bei *B* erzeugt, d.h., *C* zerstört den Rahmen von *A*. (Wir setzen hier voraus, dass kein CDMA-artiges Schema

zur Bereitstellung mehrerer Kanäle benutzt wird, daher verstümmeln Kollisionen das Signal und zerstören beide Rahmen.) Wir möchten ein MAC-Protokoll haben, dass diese Art von Kollisionen verhindert, weil dadurch Bandbreite verschwendet wird. Die Schwierigkeit einer Station, einen potenziellen Mitbewerber um das Medium nicht zu erkennen, weil die Entferungen zu groß sind, nennt man auch **Hidden-Terminal-Problem** (Problem der verborgenen Endgeräte).

Betrachten wir nun eine andere Situation: *B* überträgt zur gleichen Zeit an *A*, zu der *C* an *D* übertragen möchte (siehe ► Abbildung 4.11b). Wenn *C* das Medium überprüft, hört *C* eine Übertragung und schließt irrtümlich daraus, dass nicht an *D* gesendet werden darf (in der Abbildung als gestrichelte Linie gezeigt). Tatsächlich würde eine solche Übertragung nur in der Zone zwischen *B* und *C* einen schlechten Empfang verursachen, wo sich aber keiner der Empfänger befindet. Wir wünschen uns ein MAC-Protokoll, das diese Art der Verschiebung verhindert, da dadurch Bandbreite verschwendet wird. Das Problem wird **Exposed-Terminal-Problem** (Problem eines zu weit entfernten Endgeräts) genannt.

Die Schwierigkeit ist, dass eine Station vor Beginn einer Übertragung im Grunde wissen will, ob es rund um den Empfänger Funkaktivitäten gibt oder nicht. CSMA teilt lediglich mit, ob nahe des Senders, der den Träger prüft, Aktivitäten vorhanden sind. Bei einem Kabel verteilen sich alle Signale an alle Stationen, diese Unterscheidung gibt es hier also nicht. Es kann nur jeweils eine Übertragung im System stattfinden. Bei einem auf Kurzstreckenfunk basierenden System können gleichzeitig mehrere Übertragungen laufen, sofern alle Übertragungen für ein anderes Ziel bestimmt sind und diese Ziele nicht innerhalb gegenseitiger Reichweite liegen. Wir möchten diese Nebenläufigkeit zulassen, wenn die Zelle zunehmend größer wird, ebenso wie Leute auf einer Party nicht darauf warten sollten, bis alle im Raum schweigen, bevor sie selbst beginnen zu sprechen – mehrere Gespräche können gleichzeitig in einem großen Raum stattfinden, solange sie sich nicht auf dieselbe Stelle richten.

Ein frühes und einflussreiches Protokoll, das diese Probleme für drahtlose LANs löst, ist **MACA** (*Multiple Access with Collision Avoidance*, Mehrfachzugriff mit Kollisionsvermeidung; Karn, 1990). Das zugrunde liegende Konzept ist hier, dass der Sender den Empfänger zur Ausgabe eines kurzen Rahmens veranlasst, sodass nahe gelegene Stationen diese Übertragung erkennen und für die Dauer des nachfolgenden (großen) Datenrahmens nichts übertragen. Diese Technik wird anstelle der Trägerprüfung eingesetzt.

MACA ist in ► Abbildung 4.12 dargestellt. Betrachten wir einen Fall, in dem *A* einen Rahmen an *B* sendet. *A* beginnt durch Aussenden eines **RTS**-Rahmens (*Request To Send*, Sendeanforderung) an *B*, siehe ► Abbildung 4.12a. Dieser kurze Rahmen (30 Byte) enthält die Länge des folgenden Datenrahmens. Dann antwortet *B* mit einem **CTS**-Rahmen (*Clear To Send*, Senden erlaubt), wie in ► Abbildung 4.12b dargestellt. Der CTS-Rahmen enthält die (aus dem RTS-Rahmen kopierte) Datenlänge. Nach Empfang des CTS-Rahmens beginnt *A* mit der Übertragung.

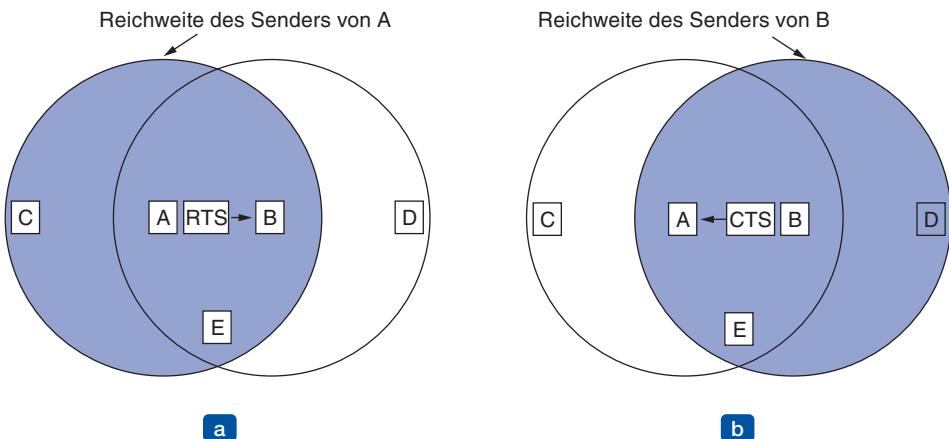


Abbildung 4.12: Das MACA-Protokoll: (a) *A* sendet ein RTS an *B*, (b) *B* antwortet *A* mit einem CTS.

Wie reagieren nun Stationen, die einen dieser Rahmen mithören? Stationen, die das RTS hören, befinden sich in der Nähe von *A* und müssen so lange stillschweigen, bis CTS konfliktfrei an *A* zurückgesendet wird. Stationen, die das CTS hören, befinden sich in der Nähe von *B* und müssen während der anstehenden Datenübertragung, deren Länge sie durch Prüfung des CTS-Rahmens ermitteln können, schweigen.

In Abbildung 4.12 befindet sich *C* innerhalb des Bereichs von *A*, nicht aber in dem von *B*. Deshalb hört *C* das RTS von *A*, aber nicht das CTS von *B*. Solange *C* das CTS nicht stört, kann *C* zur gleichen Zeit senden, in der der betreffende Datenrahmen übertragen wird. Demgegenüber liegt *D* nicht in der Reichweite von *A*, sondern in der von *B*. *D* hört nur das CTS, nicht aber das RTS. Daher folgert Station *D* aus diesem CTS, dass sie sich in der Nähe einer Station befindet, die jeden Augenblick einen Rahmen empfangen muss; deshalb hält sich *D* mit Senden zurück, bis der Rahmen erwartungsgemäß fertig übertragen wurde. Station *E* hört beide Steuernachrichten und muss wie *D* stillhalten, bis der Datenrahmen fertig ist.

Trotz dieser Vorsichtsmaßnahmen können Kollisionen entstehen. So könnten *B* und *C* z.B. gleichzeitig RTS-Rahmen an *A* senden. Diese kollidieren und gehen verloren. Im Falle einer Kollision wartet der erfolglose Sender (d.h. derjenige, der das CTS nicht innerhalb des erwarteten Zeitintervalls hört) eine zufallsgesteuerte Zeitspanne und versucht es später erneut.

4.3 Ethernet

Wir haben nun unsere Beschreibung der Protokolle für die Kanalzuordnung auf abstrakter Ebene abgeschlossen. Im Folgenden untersuchen wir, wie diese Prinzipien auf echte Systeme angewendet werden können. Viele der Entwürfe für PANs, LANs und MANs wurden unter der Bezeichnung IEEE 802 standardisiert. Wie in Abbildung 1.38 gezeigt, haben einige überlebt, viele andere sind jedoch in Vergessenheit geraten. Einige Leute, die an die Wiedergeburt glauben, denken, dass Charles Darwin als Mit-

glied der IEEE Standards Association wiedergeboren wurde, um die nicht Überlebensfähigen auszusondieren. Die wichtigsten Überlebenden sind IEEE 802.3 (Ethernet) und IEEE 802.11 (drahtloses LAN). Bluetooth (drahtloses PAN) wird häufig eingesetzt, ist aber heute außerhalb von IEEE 802.15 standardisiert. Bei IEEE 802.16 (drahtloses MAN) ist es noch zu früh, um Aussagen zu treffen. In der sechsten Ausgabe dieses Buches werden Sie dann etwas darüber erfahren.

Wir wollen unsere Untersuchung von realen Systeme mit Ethernet beginnen, das wahrscheinlich die allgegenwärtigste Art von Rechnernetzen auf der Welt darstellt. Es gibt zwei Typen von Ethernet: klassisches Ethernet, welches das Mehrfachzugriffsproblem mithilfe der Techniken löst, die wir in diesem Kapitel besprochen haben; und **Switched Ethernet**, bei welchem sogenannte Switches benutzt werden, um mehrere Computer zu verbinden. Es ist wichtig zu bemerken, dass beide Arten als Ethernet bezeichnet werden, obwohl sie ziemlich unterschiedlich sind. Klassisches Ethernet ist die ursprüngliche Form und läuft mit Raten von 3 bis 100 Mbit/s. Switched Ethernet ist das, wozu Ethernet geworden ist, und läuft mit 100, 1 000 und 10 000 Mbit/s, in Gestalt von Fast Ethernet, Gigabit-Ethernet und 10-Gigabit-Ethernet. In der Praxis wird heutzutage nur noch Switched Ethernet eingesetzt.

Wir werden diese historischen Formen von Ethernet in chronologischer Reihenfolge besprechen, um zu sehen, wie sie sich entwickelt haben. Da Ethernet und IEEE 802.3 bis auf eine kleinere Abweichung (die wir im Folgenden kurz behandeln) identisch sind, verwenden viele die Begriffe „Ethernet“ und „IEEE 802.3“ synonym. Wir schließen uns dem an. Weitere Informationen über das Ethernet finden Sie in Spurgeon (2000).

4.3.1 Bitübertragungsschicht von klassischem Ethernet

Die Geschichte des Ethernets begann zur gleichen Zeit wie die Geschichte von ALOHA, als ein Student namens Bob Metcalfe seinen Abschluss am MIT machte und dann nach Harvard ging, um dort zu promovieren. Während seiner Studien lernte er die Arbeit von Abramson kennen. Dies interessierte ihn so sehr, dass er nach seiner Promotion in Harvard einen Sommer nach Hawaii ging, um bei Abramson zu arbeiten, bevor er seinen Job beim Xerox PARC (Palo Alto Research Center) antrat. Als er zu PARC kam, hatten die Forscher dort ein Gerät entwickelt und gebaut, das später den Namen Personal Computer tragen sollte. Aber die Geräte waren isoliert. Mit der Kenntnis von Abramsons Arbeit entwickelte und implementierte er zusammen mit seinem Kollegen David Boggs das erste lokale Netz (Metcalfe und Boggs, 1976). Es verwendete ein einzelnes langes, dickes Koaxialkabel und lief mit 3 Mbit/s.

Dieses System wurde **Ethernet** genannt, nach dem *Lichtäther*, von dem einmal angenommen wurde, dass sich in ihm elektromagnetische Strahlung fortpflanzt. (Als im 19. Jahrhundert der britische Physiker James Clerk Maxwell entdeckte, dass sich elektromagnetische Strahlung durch eine Wellengleichung beschreiben lässt, nahmen die Wissenschaftler allgemein an, dass der Weltraum mit einem ätherischen Medium gefüllt sein müsse, in dem sich Strahlung fortpflanzen kann. Nach dem berühmten Experiment von Michelson-Morley im Jahre 1887 entdeckten die Physiker, dass sich elektromagnetische Strahlung im Vakuum ausbreiten kann.)

Das Xerox-Ethernet war so erfolgreich, dass DEC, Intel und Xerox 1978 den **DIX-Standard** für das 10-Mbit/s-Ethernet definierten. Mit einer kleineren Änderung wurde der DIX-Standard 1983 zum IEEE-802.3-Standard. Sehr zum Nachteil von Xerox, das bereits einige bahnbrechende Erfindungen (wie den Personal Computer) gemacht hatte und diese dann nicht vermarkten konnte – diese Geschichte wird ausführlicher in *Fumbling the Future* von Smith und Alexander (1988)¹ erzählt. Als Xerox außer seinem Beitrag zur Standardisierung kein Engagement zeigte, gründete Metcalfe sein eigenes Unternehmen, 3Com, um Ethernet-Adapter für PCs zu verkaufen. Seither hat 3COM viele Millionen davon verkauft.

Klassisches Ethernet schlängelt sich um das Gebäude wie ein einzelnes langes Kabel herum, an das alle Computer angeschlossen werden. Diese Architektur wird in ►Abbildung 4.13 gezeigt. Die erste Variante, allgemein **Thick Ethernet** genannt, ähnelt einem gelben Gartenschlauch, der im Abstand von 2,5 m an den Stellen Markierungen aufweist, wo die Computer angeschlossen werden können. (Der IEEE-802.3-Standard forderte zwar nicht, dass das Kabel gelb sein musste, hat dies jedoch vorgeschlagen.) Der Nachfolger war **Thin Ethernet**, das sich leichter biegen ließ und Verbindungen über Industriestandard-BNC-Stecker herstellte. Thin Ethernet war viel preisgünstiger und einfacher zu installieren, bot aber pro Segment nur eine Länge von 185 m (statt 500 m bei Thick Ethernet), wobei in einem Segment nur maximal 30 Rechner (statt 100) unterstützt werden konnten.

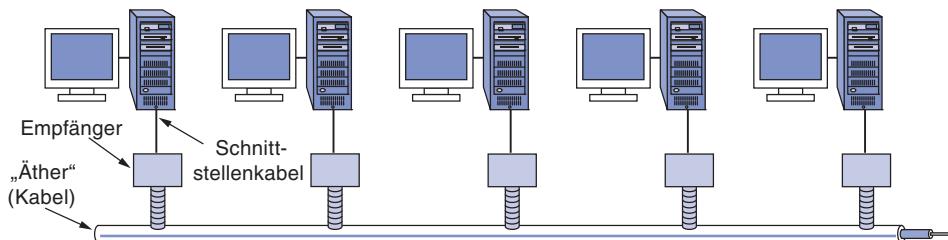


Abbildung 4.13: Architektur des klassischen Ethernets.

Jede Ethernet-Version hat eine maximale Kabellänge pro Segment (d.h. unverstärkte Länge), über die sich das Signal ausbreitet. Um das Netz auf größere Entfernen auszuweiten, können mehrere Kabelsegmente durch **Repeater** verbunden werden. Ein Repeater arbeitet auf der Ebene der Bitübertragungsschicht. Er empfängt, verstärkt (d.h. bereitst neu auf) und überträgt Signale in beide Richtungen. Aus Sicht der Software sind mehrere durch Repeater verbundene Kabel nichts anderes als ein einziger Kabelstrang, abgesehen von der kleinen Verzögerung, die durch Repeater verursacht wird.

Über jedes dieser Kabel wurde Information unter Verwendung der Manchester-Codierung (siehe Abschnitt 2.5) gesendet. Ein Ethernet konnte mehrere Kabelsegmente und mehrere Repeater enthalten, wobei zwei Transceiver nicht weiter als 2,5 km voneinander entfernt sein und zwischen zwei Transceivern nicht mehr als vier Repeater liegen

¹ In Deutsch erschienen unter dem Titel „Das Milliardenspiel. Xerox‘ Kampf um den ersten PC“.

durften. Diese Einschränkung wurde eingeführt, damit das MAC-Protokoll, welches wir uns als Nächstes ansehen wollen, korrekt funktionierte.

4.3.2 Ethernet-MAC-Teilschichtprotokoll

Das Format, das zum Senden von Rahmen eingesetzt wird, ist in ▶ Abbildung 4.14 dargestellt. Jeder Rahmen beginnt mit einer Präambel (*Preamble*) von 8 Byte, wobei jedes Byte das Bitmuster 10101010 enthält (mit Ausnahme des letzten Bytes, in dem die letzten 2 Bits auf 11 gesetzt sind). Dieses letzte Byte wird der *Start of Frame*-Begrenzer für IEEE 802.3 genannt. Die Manchester-Codierung dieses Musters erzeugt für 6,4 µs eine Rechteckwelle von 10 MHz, damit sich der Takt des Empfängers mit dem Takt des Senders synchronisieren kann. Die letzten zwei 1-Bits sagen dem Empfänger, dass der Rest des Rahmens für die Übertragung bereit ist.

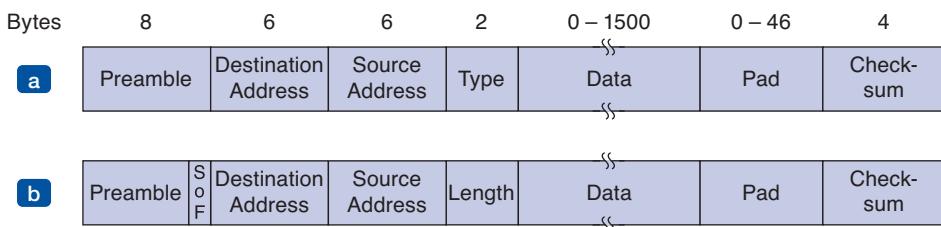


Abbildung 4.14: Rahmenformate: (a) Ethernet (DIX), (b) IEEE 802.3.

Als Nächstes kommen zwei Adressen: eine für das Ziel und eine für die Quelle. Sie sind jeweils 6 Byte lang. Das zuerst übertragene Bit der Zieladresse ist 0 für normale Adressen und 1 für Gruppenadressen. Durch Gruppenadressen können mehrere Stationen eine einzige Adresse abhören. Wird ein Rahmen an eine Gruppenadresse geschickt, empfangen ihn alle Stationen der Gruppe. Das Senden an eine Gruppe von Stationen wird **Multicasting** genannt. Die spezielle Adresse, die nur aus 1-Bits besteht, ist für **Broadcasting** reserviert. Ein Rahmen, der im Feld der Zieladresse nur 1-Bits enthält, wird von allen Stationen im Netz akzeptiert. Multicasting ist selektiver, erfordert aber die Verwaltung der Gruppen, um zu definieren, welche Stationen zur Gruppe gehören. Umgekehrt unterscheidet **Broadcasting** überhaupt nicht zwischen Stationen und braucht daher auch keine Gruppenverwaltung.

Eine interessante Eigenschaft der Stationsquelladressen ist, dass sie global eindeutig sind. Sie werden von der IEEE vergeben um sicherzustellen, dass nirgendwo auf der Welt zwei Stationen dieselbe Adresse haben. Die Idee dabei ist, dass jede Station eindeutig jede andere Station einfach durch die richtige 48-Bit-Zahl adressieren kann. Dazu werden die ersten drei Byte des Adressfelds für einen Identifikator namens **OUI** (*Organizationally Unique Identifier*) benutzt. Die Werte für dieses Feld werden von der IEEE zugewiesen und geben den Hersteller an. Den Herstellern werden Blöcke von 2^{24} Adressen zugewiesen. Der Hersteller weist die letzten drei Byte der Adresse zu und programmiert die vollständige Adresse vor dem Verkauf in die Netzwerkkarte ein.

Als Nächstes kommt das Feld *Type* bzw. *Length*, je nachdem, ob es sich um einen Ethernet- oder einen IEEE-802.3-Rahmen handelt. Ethernet verwendet das *Type*-Feld, um dem Empfänger mitzuteilen, was mit dem Rahmen geschehen soll. Mehrere Vermittlungsschichtprotokolle können auf einem Rechner zur gleichen Zeit in Verwendung sein, sodass bei Ankunft des Ethernet-Rahmens das Betriebssystem wissen muss, an wen der Rahmen übergeben werden soll. Das *Type*-Feld spezifiziert den Prozess, an den der Rahmen übergeben wird. Der *Type*-Code 0x0 800 bedeutet beispielsweise, dass die Daten ein IPv4-Paket beinhalten.

IEEE 802.3 hatte – in ihrer Weisheit – entschieden, dass dieses Feld die Länge des Rahmens enthalten sollte, da die Ethernet-Länge durch einen Blick auf die Daten festgestellt werden konnte – eine Schichtverletzung, falls es da ja eine gab. Dies bedeutete natürlich, dass der Empfänger nicht mehr feststellen konnte, was mit dem eingehenden Rahmen passieren sollte. Dieses Problem wurde durch das Hinzufügen eines weiteren Headers für das **LLC**-Protokoll (*Logical Link Control*) in den Datenteil gelöst. Es werden 8 Byte benutzt, um die 2 Byte von Protokolltypinformation zu transportieren.

Leider war bei der Veröffentlichung von IEEE 802.3 so viel Hardware und Software für DIX-Ethernet im Einsatz, dass nur wenige Hersteller und Benutzer die Neuverpackung der *Type*- und *Length*-Felder begeistert aufnahmen. 1997 warf die IEEE das Handtuch und erkannte beide Methoden an. Glücklicherweise hatten alle vor 1997 verwendeten *Type*-Felder Werte größer als 1 500, der damals gängigen maximalen Datengröße. Heute gilt die Regel, dass jede Zahl kleiner oder gleich 0x600 (1 536) als *Length* und jede Zahl über 0x600 als *Type* interpretiert werden kann. Nun kann IEEE mit Recht behaupten, dass ihre Standards allgemein verwendet werden, und alle anderen können so weitermachen wie bisher (und sich nicht an LCC stören), ohne dabei ein schlechtes Gewissen haben zu müssen.

Als Nächstes kommen die Daten mit bis zu 1 500 Byte. Diese Grenze wurde ein wenig willkürlich zu der Zeit festgelegt, als der Ethernet-Standard in Stein gemeißelt wurde. Ausschlaggebend hierfür war im Grunde, dass ein Transceiver ausreichend RAM benötigt, um einen ganzen Rahmen zu speichern, und RAM war im Jahr 1978 noch sehr teuer. Eine höhere Obergrenze hätte mehr RAM bedeutet, was den Transceiver verteuert hätte.

Darüber hinaus gibt es neben der maximalen Länge eines Datenrahmens auch eine Mindestlänge. Wenngleich ein Datenfeld mit 0 Byte manchmal nützlich ist, verursacht es auch ein Problem. Wenn ein Transceiver eine Kollision entdeckt, schneidet er den aktuellen Rahmen ab, sodass im Kabel andauernd einzelne Bits und Bruchteile von Rahmen umherwandern. Um die Unterscheidung von Müll und gültigen Rahmen zu erleichtern, müssen bei Ethernet gültige Rahmen von der Zieladresse bis einschließlich der Prüfsumme (jeweils inklusive) mindestens 64 Byte lang sein. Falls der Daten teil des Rahmens kleiner als 46 Byte ist, füllt das *Pad*-Feld (Füllung) den Rahmen bis zur Minimallänge auf.

Die Festlegung einer minimalen Rahmenlänge hat einen weiteren, noch wichtigeren Grund. Damit soll auch verhindert werden, dass eine Station die Übertragung eines

kurzen Rahmens beendet, bevor dessen erstes Bit überhaupt das andere Ende des Kabels erreicht hat, wo er dann eventuell mit einem anderen Rahmen kollidiert. Dieses Problem ist in ▶ Abbildung 4.15a-d dargestellt. Zum Zeitpunkt 0 sendet Station A an einem Ende des Netzes einen Rahmen. Gehen wir als Beispiel von einer Signalübertragungszeit von τ aus, bis dieser Rahmen das andere Ende erreicht. Kurz vor Erreichen des anderen Endes (d.h. zur Zeit $\tau - \epsilon$) beginnt B, die am weitesten entfernte Station, mit einer Übertragung. Erkennt B, dass sie mehr Leistung erhält als sie ausgibt, dann weiß sie, dass eine Kollision stattgefunden hat; deshalb bricht sie ihre Übertragung ab und erzeugt ein 48-Bit-Burst-Signal, um alle anderen Stationen zu warnen. Anders ausgedrückt, die Leitung wird blockiert um sicherzustellen, dass der Sender die Kollision nicht übersieht. Etwa zur Zeit 2τ sieht der Sender dieses Warnsignal und bricht seinerseits die Übertragung ab. Dann wartet er eine zufällige Zeitspanne, bevor er es erneut versucht.

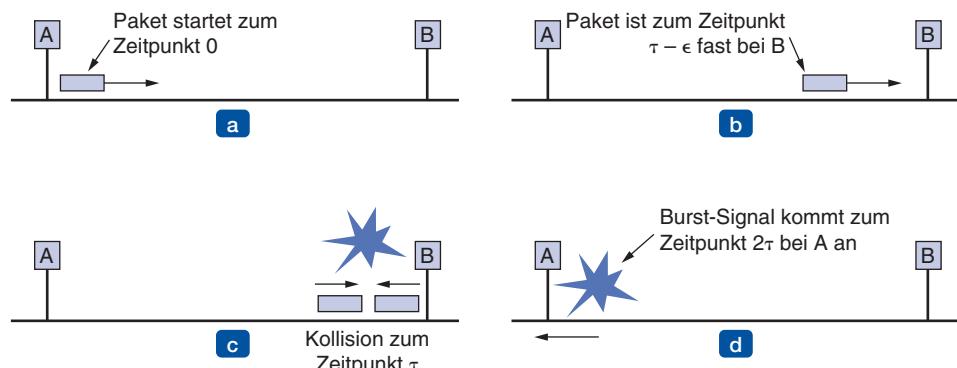


Abbildung 4.15: Die Kollisionserkennung kann 2τ dauern.

Versucht eine Station, einen sehr kurzen Rahmen zu übertragen, dann kann eine Kollision stattfinden, aber die Übertragung wird beendet sein, bevor das Warnsignal zum Zeitpunkt 2τ zur Station zurückkommt. Der Sender schließt daraus irrigerweise, dass der Rahmen erfolgreich gesendet wurde. Damit dieser Fall nicht eintritt, müssen alle Rahmen länger als 2τ senden, sodass die Übertragung immer noch stattfindet, wenn das Burst-Signal zum Sender zurückgelangt. Bei einem 10-Mbit/s-LAN mit einer maximalen Länge von 2 500 Metern und vier Repeatern (aus der IEEE-802.3-Spezifikation) liegt die Paketumlaufzeit (einschließlich der Verbreitungszeit über die vier Repeater) im schlechtesten Fall bei etwa 50 µs, einschließlich der Zeit für den Durchlauf durch die Repeater. Daher muss die Übertragung des kleinsten erlaubten Rahmens mindestens so lange dauern. Bei 10 Mbit/s benötigt ein Bit 100 ns, sodass der kleinste zu übertragende Rahmen 500 Bit enthalten muss. Um etwas Sicherheitsspielraum hinzuzufügen, wurde dies auf 512 Bit oder 64 Byte aufgerundet.

Das letzte Ethernet-Feld ist die Prüfsumme (*Checksum*). Es ist ein 32 Bit langer CRC-Wert von der Art, wie wir sie in Abschnitt 3.2 untersucht haben. In der Tat wird dieser Wert genau von dem Generatorpolynom definiert, das wir dort angegeben haben und das auch für PPP, ADSL und andere Verbindungen angewandt wird.

CSMA/CD mit binärem exponentiellen Backoff-Algorithmus

Klassisches Ethernet verwendet den 1-persistenten CSMA/CD-Algorithmus, den wir in Abschnitt 4.2 untersucht haben. Diese Benennung bedeutet nur, dass Stationen das Medium überprüfen, wenn sie einen Rahmen zur Übertragung haben, und den Rahmen senden, sobald das Medium verfügbar ist. Der Kanal wird von den Stationen auf Kollisionen hin überwacht, während sie senden. Falls eine Kollision auftritt, wird die Übertragung mit einem kurzen Signal abgebrochen und nach einer zufälligen Zeitspanne erneut gesendet.

Wir wollen uns nun ansehen, wie das Zufallsintervall im Falle einer Kollision bestimmt wird, da dies eine neue Methode ist. Das Modell ist noch dasselbe wie in Abbildung 4.5. Nach einer Kollision wird die Zeit in einzelne Zeitscheiben unterteilt, deren Länge im schlechtesten Fall der vollen Paketumlaufzeit (2τ) entspricht. Um den längsten nach Ethernet zulässigen Pfad zu verwenden (2,5 km und vier Repeater), wurden die Zeitscheiben auf 512 Bitzeiten bzw. 51,2 µs gesetzt.

Nach der ersten Kollision wartet die Station zufällig entweder 0 oder 1 Zeitscheibenzeit, bevor sie es wieder versucht. Wenn zwei Stationen kollidieren und die gleiche Zufallszahl nehmen, kollidieren sie wieder. Nach der zweiten Kollision nimmt sich jede zufällig eine der Zahlen 0, 1, 2 oder 3 und wartet diese Anzahl von Zeitscheiben ab. Falls eine dritte Kollision auftritt (die Wahrscheinlichkeit dafür ist 0,25), wird die nächste Wartezeit zufällig zwischen 0 und $2^3 - 1$ gewählt.

Allgemein wird nach i Kollisionen eine Zufallszahl zwischen 0 und $2^i - 1$ gewählt und diese Anzahl von Zeitscheiben wird übersprungen. Nachdem allerdings zehn Kollisionen erreicht worden sind, wird das maximale Zufallsintervall bei 1 023 Zeitscheiben eingefroren. Nach 16 Kollisionen wirft der Controller das Handtuch und berichtet dem Rechner den Fehler. Die weitere Behandlung liegt jetzt bei höheren Schichten.

Dieser Algorithmus heißt **binäres exponentielles Backoff** (*binary exponential backoff*). Er wurde gewählt, um die Anzahl von sendebereiten Stationen dynamisch anpassen zu können. Die Wahrscheinlichkeit, dass zwei Stationen bei einem festen Zufallsintervall von 1 023 ein zweites Mal kollidieren, wäre verschwindend gering, aber die durchschnittliche Wartezeit nach einer Kollision läge bei mehreren Hundert Zeitscheiben, was insgesamt zu einer spürbaren Verzögerung führen würde. Versuchten auf der anderen Seite 100 Stationen gleichzeitig zu senden und wartete jede nach einer Kollision entweder 1 oder 0 Zeitscheiben, dann würden sie so lange kollidieren, bis 99 Stationen 0 und eine Station 1 wählt oder umgekehrt. Das kann Jahre dauern. Indem das Zufallsintervall exponentiell mit der Anzahl der Kollisionen anwächst, garantiert der Algorithmus kurze Verzögerungen, falls wenige Stationen kollidieren, aber er garantiert auch eine einigermaßen rasche Auflösung, wenn viele Stationen kollidieren. Wenn man die Obergrenze für das Backoff auf 1 023 festsetzt, wird die Grenze nicht zu groß.

Falls es keine Kollision gibt, nimmt der Sender an, dass der Rahmen wahrscheinlich erfolgreich angekommen ist. Das heißt, weder CSMA/CD noch Ethernet bieten Bestätigungen. Diese Wahl ist für verkabelte und Glasfaserkanäle angemessen, da hier die Fehlerrate niedrig ist. Alle auftretenden Fehler müssen dann von dem CRC entdeckt

werden, die Wiederherstellung wird von höheren Schichten vorgenommen. Bei drahtlosen Kanälen, die fehleranfälliger sind, werden dagegen Bestätigungen eingesetzt, wie noch sehen werden.

4.3.3 Leistungsaspekte bei Ethernet

Wir wollen nun kurz die Leistungsfähigkeit von klassischem Ethernet unter hoher und konstanter Auslastung untersucht. Wir gehen immer davon aus, dass k Stationen sendebereit sind. Eine umfassende Analyse des binären exponentiellen Backoff-Algorithmus ist sehr komplex; also folgen wir dem Beispiel von Metcalfe und Boggs (1976) und gehen für jede Zeitscheibe von einer konstanten Wahrscheinlichkeit für eine nochmalige Übertragung aus. Wenn jede Station während einer Zeitscheibe mit der Wahrscheinlichkeit p überträgt, ergibt sich für die Wahrscheinlichkeit A , dass eine Station während dieser Zeitscheibe das Übertragungsmedium belegt:

$$A = kp(1-p)^{k-1} \quad (4.5)$$

A wird maximiert, wenn $p=1/k$, wobei $A \rightarrow 1/e$, wenn $k \rightarrow \infty$. Die Wahrscheinlichkeit, dass das Konkurrenzintervall genau j Zeitscheiben umfasst, ist $A(1-A)^{j-1}$, sodass die durchschnittliche Anzahl von Zeitscheiben pro Konkurrenzsituation gegeben ist durch

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}$$

Da jede Zeitscheibe 2τ dauert, ist das mittlere Konkurrenzintervall w gleich $2\tau/A$. Unter einem optimalen p beträgt die durchschnittliche Anzahl der Konkurrenzzeitscheiben nie mehr als e , sodass w höchstens $2te \approx 5,4\tau$ ist.

Wenn ein durchschnittlicher Rahmen P Sekunden zur Übertragung benötigt, kann die Kanaleffizienz, wenn viele Stationen senden, wie folgt angegeben werden:

$$\text{Kanaleffizienz} = \frac{P}{P + 2\tau / A} \quad (4.6)$$

Hieran kann man sehen, an welcher Stelle die maximale Kabellänge zwischen zwei Stationen die Leistungswerte beeinflusst. Je länger das Kabel ist, desto länger ist auch das Konkurrenzintervall, deshalb wird vom Ethernet-Standard eine maximale Kabellänge angegeben.

Es ist recht aufschlussreich, die Gleichung (4.6) in Bezug auf die Rahmenlänge F , die Bandbreite des Netzes B , die Kabellänge L und die Geschwindigkeit der Signalausbreitungsverzögerung c für den günstigsten Fall von e Konkurrenzzeitscheiben pro Rahmen auszudrücken. Mit $P=F/B$ wird aus Gleichung (4.6):

$$\text{Kanaleffizienz} = \frac{1}{1 + 2BLc / cF} \quad (4.7)$$

Ist der zweite Term im Nenner groß, dann ist die Effizienz des Netzes gering. Genauer ausgedrückt: Eine Vergrößerung der Netzbandbreite oder der Entfernung (also des Pro-

duktes BL) verringert bei gleichbleibender Rahmengröße die Leistungsstärke. Leider ist der Großteil der Forschungsarbeiten im Bereich der Netzhardware genau auf die Erhöhung dieses Produkts ausgerichtet. Man möchte hohe Bandbreiten über große Entfernungen (z.B. Glasfaser-MANs), jedoch ist klassisches Ethernet für diese Anwendungen nicht das beste System. Wir werden im nächsten Abschnitt noch andere Implementierungsmöglichkeiten für Ethernet kennenlernen.

In ▶ Abbildung 4.16 wird die Kanaleffizienz der Zahl der sendebereiten Stationen für $2\tau=51,2 \mu\text{s}$ und einer Datenrate von 10 Mbit/s unter Verwendung von Gleichung (4.7) dargestellt. Bei einer Zeitscheibe von 64 Byte sind 64-Byte-Rahmen selbstverständlich nicht effizient. Andererseits ergibt sich mit Rahmen von 1 024 Byte und einem asymptotischen Wert von e 64-Byte-Zeitscheiben pro Konkurrenzintervall eine Konkurrenzperiode von 174 Byte und eine Effizienz von 0,85 %. Dieses Ergebnis ist viel besser als die 37 %ige Effizienz von S-ALOHA.

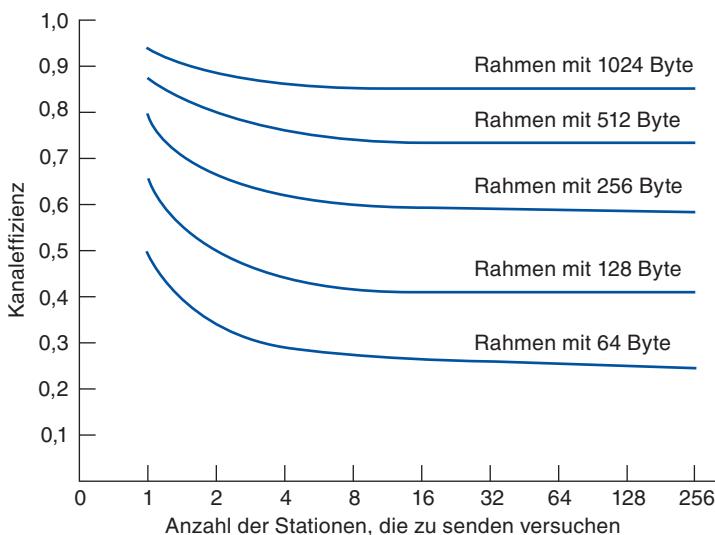


Abbildung 4.16: Effizienz von Ethernet bei 10 Mbit/s und Zeitscheiben von 512 Bit.

Man sollte wahrscheinlich erwähnen, dass für das Ethernet (und andere Netze) sehr viele theoretische Leistungsanalysen durchgeführt wurden. Die meisten der Ergebnisse sollten aus zwei Gründen mit größter Vorsicht aufgenommen werden. Zunächst wurden fast alle theoretischen Arbeiten unter der Annahme eines Poisson-verteilten Datenverkehrs ausgelegt. Nachdem man aber begonnen hat, echte Datenflüsse zu betrachten, erweist es sich jetzt, dass der Netzverkehr selten nach Poisson verläuft, sondern selbstähnlich oder Burst-anfällig über einer Reihe von Zeitskalen (Paxson und Floyd, 1995; Leland et al., 1994). Das bedeutet, dass die Ermittlung von Durchschnitten über lange Zeitspannen den Verkehr nicht glättet. Ebenso wie die Verwendung von fragwürdigen Modellen ist die Analyse von „interessanten“ Fällen unter Einsatz von unrealistisch hoher Last kritisch zu bewerten. Boggs et al. (1988) zeigten durch Experimente, dass Ethernet in der Realität sogar bei mäßig hoher Auslastung gut arbeitet.

4.3.4 Switched Ethernet

Ethernet begann sich bald von der Einzel-Kabelarchitektur des klassischen Ethernets wegzuentwickeln. Probleme in Verbindung mit dem Auffinden von Brüchen oder dem Verlieren von Verbindungen haben zur Entwicklung eines anderen Verkabelungsmusters geführt, bei dem von jeder Station aus ein dediziertes Kabel zu einem zentralen **Hub** führt. Ein Hub verbindet einfach all die angeschlossenen Kabel elektrisch, als ob sie zusammengelötet wären. Diese Konfiguration ist in ► Abbildung 4.17a zu sehen.

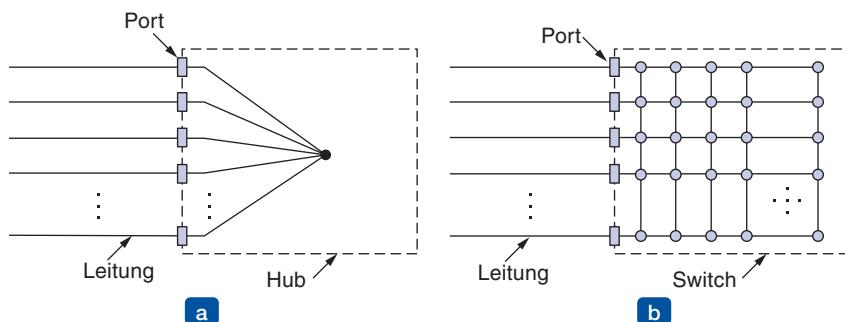


Abbildung 4.17: (a) Hub. (b) Switch.

Die Kabel waren Twisted-Pair-Kabel von der Telefongesellschaft, da die meisten Bürogebäude bereits verkabelt waren und normalerweise jede Menge Ersatzkabel übrig sind. Diese Wiederverwendung war ein Gewinn, doch dadurch wurde die maximale Kabellänge bis zum Hub auf 100 m verringert (bzw. 200 m, wenn qualitativ hochwertige Twisted-Pair-Kabel der Kategorie 5 verwendet wurden). Bei dieser Konfiguration ist das Hinzufügen weiterer oder das Entfernen vorhandener Stationen viel einfacher. Außerdem können Kabelbrüche schnell entdeckt werden. Aufgrund dieses Vorteils, die vorhandene Verkabelung zu nutzen, sowie der Wartungsfreundlichkeit wurden Twisted-Pair-Hubs schnell die vorherrschende Form des Ethernets.

Hubs erhöhen jedoch nicht die Kapazität, da sie logisch äquivalent zu einem einzelnen langen Kabel des klassischen Ethernets sind. Wenn immer mehr Stationen hinzugefügt werden, so bekommt jede Station einen abnehmenden Anteil der konstanten Kapazität. Zu guter Letzt wird das LAN gesättigt sein. Ein Ausweg aus dieser Situation ist es, die Geschwindigkeit zu erhöhen, beispielsweise von 10 Mbit/s auf 100 Mbit/s, 1 Gbit/s oder auf sogar noch größere Geschwindigkeiten. Doch mit dem Anwachsen von Multimedia und leistungsstarken Servern kann selbst ein 1-Gbit/s-Ethernet gesättigt werden.

Glücklicherweise gibt es noch eine weitere Methode, höhere Lasten zu verarbeiten: Switched Ethernet. Den Kern dieses Systems bildet ein **Switch**, der eine Backplane hat, die alle Ports verbindet (siehe ► Abbildung 4.17b). Von außen sieht ein Switch genau wie ein Hub aus. Beides sind Boxen, in der Regel mit 4 bis 32 Ports, jeder mit einer Standard-RJ-Verbindungsstecker für ein Twisted-Pair-Kabel. Jedes Kabel verbindet den Switch oder Hub mit einem einzelnen Rechner, wie in ► Abbildung 4.18 gezeigt. Ein Switch hat auch dieselben Vorteile wie ein Hub. Eine neue Station kann einfach hinzugefügt oder entfernt werden, indem ein Kabel angeschlossen bzw. ausgesteckt

wird, und die meisten Fehler sind einfach zu finden, da ein unzuverlässiges Kabel oder Port in der Regel nur jeweils eine Station betrifft. Es gibt noch eine gemeinsam benutzte Komponente, die ausfallen kann – der Switch selbst –, doch wenn alle Stationen die Verbindung verlieren, wissen die IT-Leute, was zu tun ist, um das Problem zu beheben: den gesamten Switch ersetzen.

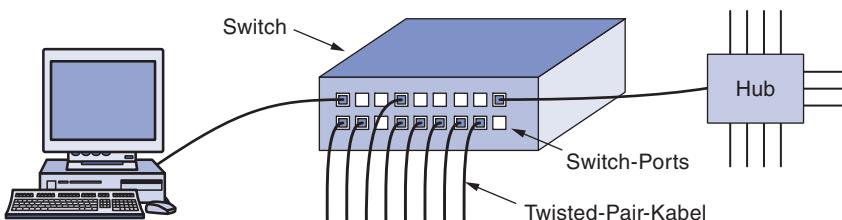


Abbildung 4.18: Ethernet-Switch.

Im Inneren des Switch passiert jedoch etwas völlig anderes. Switches legen Rahmen nur auf die Ports, für die diese Rahmen gedacht sind. Wenn ein Switch-Port einen Ethernet-Rahmen von einer Station empfängt, überprüft der Switch die Ethernet-Adresse um festzustellen, für welchen Port der Rahmen bestimmt ist. Dieser Schritt erfordert, dass der Switch herausfinden kann, welche Ports zu welcher Adresse gehören. Diesen Vorgang beschreiben wir in Abschnitt 4.8, wenn wir den allgemeinen Fall behandeln, dass Switches mit anderen Switches verbunden sind. Für den Moment nehmen wir einfach an, dass der Switch den Zielport des Rahmens kennt. Der Switch leitet den Rahmen dann über seine Hochgeschwindigkeitsbackplane zum Zielport weiter. Die Backplane läuft in der Regel mit vielen Gbit/s, unter Verwendung eines proprietären Protokolls, das nicht standardisiert werden muss, weil es vollständig innerhalb des Switch verborgen ist. Der Zielport überträgt dann den Rahmen auf die Leitung, sodass er die beabsichtigte Station erreicht. Die anderen Ports wissen nicht einmal, dass der Rahmen existiert.

Was passiert, wenn mehr als eine Station oder ein Port einen Rahmen zur gleichen Zeit senden wollen? Auch hier unterscheidet sich der Switch von Hubs. Bei einem Hub befinden sich alle Stationen in derselben **Kollisionsdomäne**. Sie müssen den CSMA/CD-Algorithmus verwenden, um ihre Übertragungen zeitlich festzulegen. Bei einem Switch ist jeder Port in seiner eigenen, unabhängigen Kollisionsdomäne. Im allgemeinen Fall eines Vollduplexkabels können die Station und der Port gleichzeitig einen Rahmen über das Kabel schicken, ohne sich um andere Ports oder Stationen kümmern zu müssen. Kollisionen sind hier unmöglich und CSMA/CD wird nicht benötigt. Wenn es sich jedoch um ein Halbduplexkabel handelt, dann müssen Station und Port auf die übliche Weise mithilfe von CSMA/CD um die Übertragung konkurrieren.

Ein Switch verbessert die Performanz gegenüber einem Hub auf zwei Arten. Zunächst wird die Kapazität effektiver genutzt, da es keine Kollisionen gibt. Noch viel wichtiger ist jedoch, dass mit einem Switch mehrere Rahmen gleichzeitig (von verschiedenen Stationen) gesendet werden können. Diese Rahmen werden die Switch-Ports erreichen und über die Backplane des Switch auf die richtigen Ports ausgegeben. Da zwei

Rahmen jedoch eventuell gleichzeitig auf denselben Ausgabeport gesendet werden könnten, muss der Switch eine Puffermöglichkeit haben, sodass er kurzzeitig einen ankommenden Rahmen zwischenspeichern kann, bis dieser zu dem Ausgabeport übertragen werden kann. Insgesamt bringen diese Verbesserungen einen großen Performanzgewinn, der mit einem Hub nicht möglich wäre. Der gesamte Systemdurchsatz kann häufig um eine Größenordnung erhöht werden, je nach Anzahl an Ports und Datenverkehrsmustern.

Die Veränderung in den Ports, auf denen Rahmen ausgegeben werden, haben auch Sicherheitsvorteile. Die meisten LAN-Schnittstellen haben einen **Gemischtmodus** (*promiscuous mode*), in dem *alle* Rahmen an jeden Computer übertragen werden, nicht nur die, die für ihn bestimmt sind. Mit einem Hub kann jeder angeschlossene Computer den Datenverkehr sehen, der zwischen allen anderen Rechnern gesendet wird. Spione und Wichtigtuer lieben diese Funktion. Bei einem Switch wird der Datenverkehr nur an die Ports weitergeleitet, für die er bestimmt ist. Diese Einschränkung bietet eine bessere Isolation, sodass die Daten nicht leicht entweichen und in die falschen Hände geraten können. Dennoch ist es besser, den Datenverkehr zu verschlüsseln, wenn Sicherheit ein wichtiger Aspekt ist.

Da der Switch auf allen Eingangssports nur Ethernet-Standardrahmen erwartet, können einige Ports als Konzentratoren fungieren. In Abbildung 4.18 ist der Port in der oberen rechten Ecke nicht an eine Station, sondern an einen Hub mit zwölf Ports angeschlossen. Gehen Rahmen auf dem Hub ein, können sie sich auf die übliche Weise mit Kollisionen und binärem Backoff um einen Übertragungskanal bewerben. Erfolgreiche Rahmen schaffen ihren Weg durch den Hub zum Switch und werden dort wie alle anderen Eingangsrahmen behandelt. Der Switch weiß nicht, dass sie sich ihren Weg erkämpfen mussten. Im Switch angekommen werden sie über die Hochgeschwindigkeitsplatine an die richtige Ausgangsleitung gesendet. Es ist auch möglich, dass das richtige Ziel auf den Leitungen war, die an den Hub angeschlossen waren, in welchem Fall der Rahmen bereits zugestellt ist, sodass der Switch ihn einfach fallen lässt. Hubs sind einfacher und kostengünstiger als Switches, aber da die Preise für Switches immer mehr sinken, sind sie eine vom Aussterben bedrohte Art geworden. Dennoch gibt es Hubs weiterhin.

4.3.5 Fast Ethernet

Zur gleichen Zeit, in der Switches beliebt wurden, kam die Geschwindigkeit von 10-Mbit/s-Ethernet unter Druck. Auf den ersten Blick sahen 10 Mbit/s wunderbar aus, so wie ein Kabelmodem für die Nutzer von Telefonmodems ein Traum war. Aber die Neuheit hat sich schnell verbraucht. Als eine Art Ergänzung zu dem Parkinson'schen Gesetz („Jede Arbeit dauert so lange, wie Zeit für sie zur Verfügung steht“) schienen sich die Daten immer mehr auszubreiten, um die zur Übertragung verfügbare Bandbreite auszufüllen.

Viele Installationen benötigten mehr Bandbreite und deshalb wurden die unzähligen 10-Mbit/s-LANs mit einem Gewirr aus Repeatern, Hubs und Switches verbunden –

obwohl viele Netzmanager mitunter das Gefühl hatten, dass diese Netze nur mit Kaugummi und Blumendraht zusammengehalten wurden. Doch selbst bei Ethernet-Switches wurde die maximale Bandbreite eines einzelnen Computers durch das Kabel begrenzt, welches ihn mit dem Switch-Port verband.

In diesem Umfeld betraute IEEE das 802.3-Komitee im Jahr 1992 mit der Aufgabe, ein schnelleres LAN zu erarbeiten. Ein Vorschlag lautete, IEEE 802.3 so zu belassen, wie es war, es lediglich schneller zu machen. Ein weiterer Vorschlag sah die völlige Umarbeitung des Standards mit einer Fülle neuer Merkmale vor, z.B. Echtzeitverkehr und digitalisierte Sprache. Im Grunde sollte bei dieser Lösung (aus Marketinggründen) nur der Name beibehalten werden. Nach einem Hin und Her entschloss sich der Ausschuss dafür, IEEE 802.3 zu lassen wie es war, es lediglich schneller zu machen. Diese Strategie erfüllte die Aufgabe, bevor sich die Technologie änderte und so unvorhergesehene Probleme mit einem nagelneuen Entwurf vermieden wurden. Der neue Entwurf musste außerdem abwärtskompatibel mit vorhandenen Ethernet-LANs sein. Die hinter dem abgelehnten Vorschlag stehenden Leute reagierten so, wie jeder anständige Mensch aus der Computerindustrie in dieser Lage reagiert hätte: Sie bildeten ihren eigenen Ausschuss und standardisierten ihr LAN allein (unter IEEE 802.12). Es wurde ein riesiger Flop.

Die Standardisierungsarbeiten wurden relativ schnell durchgezogen und das Ergebnis namens IEEE 802.3u wurde im Juni 1995 von der IEEE anerkannt. Unter technischen Gesichtspunkten ist IEEE 802.3u kein neuer Standard, sondern eine Erweiterung des bestehenden IEEE-802.3-Standards (um die Abwärtskompatibilität auch durch die Bezeichnung deutlich zu machen). Diese Strategie wird häufig eingesetzt. Da praktisch jeder **Fast Ethernet** zu IEEE 802.3u sagt, werden wir das hier auch so halten.

Das grundlegende Konzept hinter Fast Ethernet war einfach: Behalte alle alten Rahmenformate, Schnittstellen, Verfahrensregeln und senke die Bitzeit von 100 ns auf 10 ns. Technisch wäre es möglich gewesen, klassisches 10-Mbit/s-Ethernet zu kopieren und immer noch Kollisionen rechtzeitig festzustellen, indem man lediglich die maximale Kabellänge um einen Faktor 10 gekürzt hätte. Andererseits waren die Vorteile der Twisted-Pair-Verkabelung derart überwältigend, dass Fast Ethernet völlig auf diesem Design basiert. Folglich werden in allen Fast-Ethernet-Systemen Hubs verwendet; Multidrop-Kabel mit Vampirklemmen oder BNC-Stecker sind nicht zulässig.

Dennoch mussten einige Entscheidungen getroffen werden, von denen die wichtigste damit zu tun hat, welche Kabeltypen unterstützt werden sollen. Ein Bewerber war Twisted-Pair der Kategorie 3. Für dieses Kabel sprach, dass praktisch in jedem Büro der westlichen Welt vier Twisted-Pair-Kabel der Kategorie 3 (oder besser) vorhanden sind, die von einem Telefonkabelschrank maximal 100 m entfernt sind. Manchmal gibt es auch zwei solche Kabel. Auf diese Weise hätte man die vorhandene Infrastruktur der Twisted-Pair-Kabel Kategorie 3 zum enormen Vorteil unzähliger Unternehmen mitnutzen können, ohne die Gebäude neu zu verkabeln.

Der größte Nachteil von Twisted-Pair-Kabeln der Kategorie 3 ist die Unfähigkeit, Signale mit 100 Mbit/s über 100 m zu befördern. Das aber ist die maximale Rechner-Hub-Entfernung gemäß 10-Mbit/s-Spezifikation. Demgegenüber würden Twisted-Pair-Kabel der

Kategorie 5 locker 100 m abdecken und Lichtwellenleiter sogar noch mehr. Man entschied sich für den Kompromiss, alle drei Kabelarten zu unterstützen (► Abbildung 4.19), die Lösung der Kategorie 3 aber aufzupeppen, um ihr die nötige Kapazität zu verleihen.

| Name | Kabel | Max. Segment | Vorteile |
|------------|--------------|--------------|---|
| 100Base-T4 | Twisted-Pair | 100 m | Verwendet UTP Kategorie 3 |
| 100Base-TX | Twisted-Pair | 100 m | Vollduplex mit 100 Mbit/s |
| 100Base-FX | Glasfaser | 2000 m | Vollduplex mit 100 Mbit/s; weite Entfernung |

Abbildung 4.19: Ursprüngliche Fast-Ethernet-Verkabelung.

Das Modell mit UTP der Kategorie 3 hat die Bezeichnung **100Base-T4** und verwendete eine Signalübertragungsgeschwindigkeit von 25 MHz. Das sind nur 25 % mehr als die 20 MHz des Standard-Ethernets. (Die Manchester-Codierung, die wir in Abschnitt 2.5 besprochen haben, erfordert zwei Taktperioden für jedes der 10 Millionen Bit, die pro Sekunde gesendet werden.) Um die nötige Bitrate zu erreichen, setzt 100Base-T4 vier Twisted-Pair-Kabel voraus. Von den vier Paaren läuft eines immer zum Hub hin, eines immer vom Hub weg, und die übrigen zwei sind auf die gewünschte Übertragungsrichtung umschaltbar. Um 100 Mbit/s aus den drei Twisted-Pair-Kabeln in Übertragungsrichtung herauszuholen, wird ein recht verzwicktes Schema auf jedem einzelnen Kabel verwendet. Dazu gehört das Senden von ternären Signalen mit drei unterschiedlichen Spannungspegeln. Dieses Schema dürfte sicherlich keine Preise für seine Eleganz davontragen und wir werden die Einzelheiten auslassen. Da das Telefonnetz jedoch seit Jahrzehnten mit vier Twisted-Pair-Kabeln ausgelegt ist, können die meisten Büros die vorhandenen Kabelanlagen nutzen. Selbstverständlich bedeutet das, dass man auf den Telefonanschluss in diesen Büros verzichten muss, aber das ist sicherlich ein kleiner Preis für schnellere E-Mails.

100Base-T4 blieb auf der Strecke, als viele Bürogebäude mit Kategorie-5-Kabeln für **100Base-TX**-Ethernet, das den Markt zu beherrschen begann, neu verkabelt wurden. Dieser Entwurf ist einfacher, weil die Kabel Taktraten von bis zu 125 MHz und mehr handhaben können. Pro Station werden nur zwei Twisted-Pair-Kabel verwendet, eines zum Hub hin und eines vom Hub weg. Es wird weder die direkte Binärcodierung (d.h. NRZ) noch Manchester-Codierung verwendet. Stattdessen kommt die **4B/5B**-Codierung (siehe Abschnitt 2.5) zum Einsatz. Dabei werden 4 Datenbits auf 5 Signalbits abgebildet und mit 125 MHz gesendet, um 100 Mbit/s bereitzustellen. Dieses Schema ist einfach, bietet aber genügend Übergänge für die Synchronisation und nutzt die Bandbreite des Kabels relativ gut. Das 100Base-TX ist ein Vollduplexsystem. Die Stationen können gleichzeitig mit 100 Mbit/s auf einem Twisted-Pair-Kabel übertragen und mit 100 Mbit/s auf einem anderen Twisted-Pair-Kabel empfangen.

Die letzte Option heißt **100Base-FX**. Bei ihr werden zwei Stränge einer Multimode-Faser verwendet, je eine in jede Richtung. Somit ist auch diese Verkabelung Voll-duplex mit 100 Mbit/s in jeder Richtung. In dieser Konstellation kann die Entfernung zwischen einer Station und dem Switch bis zu 2 km betragen.

Fast Ethernet erlaubt Vernetzung wahlweise mittels Hubs oder Switches. Um sicherzustellen, dass der CSMA/CD-Algorithmus weiterhin korrekt funktioniert, muss das Verhältnis zwischen minimaler Rahmengröße und maximaler Kabellänge beibehalten werden, wenn sich die Netzgeschwindigkeit von 10 Mbit/s auf 100 Mbit/s erhöht. Im Verhältnis muss also entweder die minimale Rahmenlänge von 64 Byte erhöht oder die maximale Kabellänge von 2 500 m verringert werden. Die einfache Wahl war, die maximale Entfernung zwischen je zwei Stationen um den Faktor 10 zu verringern, da ein Hub mit 100-m-Kabeln schon innerhalb dieses neuen Maximums liegt. 100Base-FX-Kabel mit einer Länge von 2 km sind jedoch zu lang, um einen 100-Mbit/s-Hub mit dem normalen Ethernet-Kollisionsalgorithmus zuzulassen. Diese Kabel müssen stattdessen mit einem Switch verbunden werden und im Vollduplexmodus betrieben werden, damit keine Kollisionen auftreten.

Die Benutzer begannen schnell, Fast Ethernet einzusetzen, aber sie dachten nicht daran, ihre 10-Mbit/s-Ethernet-Karten auf älteren Rechnern wegzutwerfen. Als Folge können fast alle Fast-Ethernet-Switches eine Mischung aus 10-Mbit/s- und 100-Mbit/s-Stationen handhaben. Um das Aufrüsten zu vereinfachen, liefert der Standard selbst einen Mechanismus namens **Autonegotiation**, mit dessen Hilfe zwei Stationen automatisch die optimale Geschwindigkeit (10 oder 100 Mbit/s) und den Duplexmodus (voll oder halb) aushandeln können. Dies funktioniert meistens gut, führt aber zu Problemen bei der Duplexanpassung, wenn ein Verbindungsende Autonegotiation durchführt, das andere Ende jedoch nicht, sondern den Vollduplexmodus verwendet (Shalunov und Carlson, 2005). Die meisten Ethernet-Produkte benutzen diese Funktion, um sich selbst zu konfigurieren.

4.3.6 Gigabit-Ethernet

Die Tinte zur Definition des Fast-Ethernet-Standards war kaum trocken, als das IEEE-802-Komitee mit der Erarbeitung eines noch schnelleren Ethernet-Standards begann (1995), der ziemlich schnell auf den Namen **Gigabit-Ethernet** getauft wurde. IEEE genehmigte 1999 die bekannteste Form unter der Bezeichnung 802.3ab. Im Folgenden werden wir die wichtigsten Eigenschaften des Gigabit-Ethernets eingehender untersuchen. Weitere Informationen finden Sie in Spurgeon (2000).

Die Ziele des Komitees für Gigabit-Ethernet waren im Grunde die gleichen wie die des Komitees für Fast Ethernet: die Leistung verzehnfachen, während die Kompatibilität mit allen vorhandenen Ethernet-Standards erhalten bleibt. Insbesondere musste das Gigabit-Ethernet einen nicht bestätigten Datagrammdienst für Unicast und Broadcast zur Verfügung stellen, das gleiche bereits verwendete 48-Bit-Adressierschema unterstützen und das gleiche Rahmenformat einschließlich der Mindest- und Maximalgrößen der Rahmen aufweisen. Der erarbeitete Standard erfüllt alle diese Voraussetzungen.

Wie bei Fast Ethernet verwenden alle Konfigurationen des Gigabit-Ethernets Punkt-zu-Punkt-Verbindungen. Bei der einfachsten Konfiguration (► Abbildung 4.20a) sind zwei Computer direkt miteinander verbunden. In der Regel wird aber ein Switch oder Hub mit mehreren Computern und eventuell auch weiteren Switches oder Hubs ver-

bunden, wie in ▶ Abbildung 4.20b dargestellt. Bei beiden Konfigurationen sind an ein Ethernet-Kabel genau zwei Geräte angeschlossen, nicht mehr und nicht weniger.

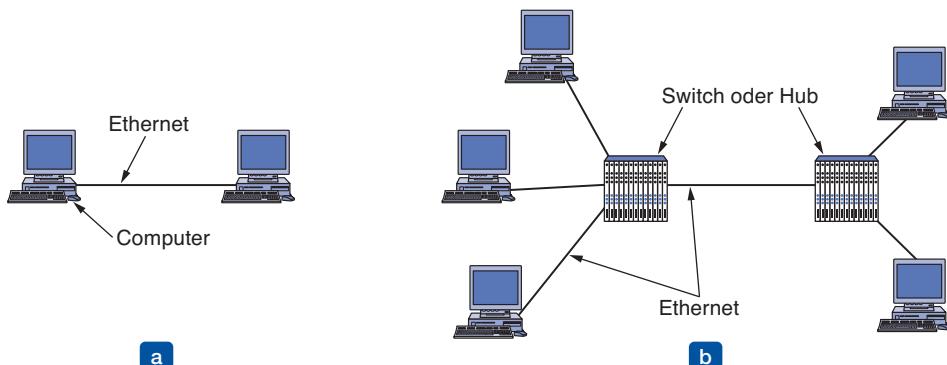


Abbildung 4.20: (a) Ethernet mit zwei Stationen. (b) Ethernet mit vielen Stationen.

Ebenso wie Fast Ethernet unterstützt Gigabit-Ethernet zwei Betriebsmodi: den Voll-duplex- und den Halbduplexmodus. Der „normale“ Modus ist der Vollduplexmodus, der Datenverkehr in beide Richtungen gleichzeitig zulässt. Dieser Modus wird verwendet, wenn ein zentraler Switch mit Rechnern (oder anderen Switches) verbunden ist. In dieser Konfiguration werden alle Leitungen gepuffert, damit jeder Switch und jeder Computer jederzeit Rahmen senden kann, wenn er dies möchte. Der Sender muss den Kanal nicht prüfen um herauszufinden, ob ihn eine andere Station belegt, da Konkurrenz unmöglich ist. Auf der Leitung zwischen einem Computer und einem Switch ist der Computer der einzige mögliche Sender zum Switch. Die Übertragung gelingt sogar dann, wenn der Switch aktuell einen Rahmen an den Computer sendet (weil die Leitung den Vollduplexbetrieb unterstützt). Da keine Konkurrenz möglich ist, wird auch kein CSMA/CD-Protokoll verwendet, sodass die maximale Kabellänge eher durch die Signalstärke als dadurch beschränkt wird, wie lange es im schlechtesten Fall dauert, dass ein Rausch-Burst zum Sender zurückübertragen wird. Switches können Geschwindigkeiten frei kombinieren und zuordnen. Autonegotiation wird genau wie bei Fast Ethernet unterstützt, nur dass hier zwischen 10, 100 und 1 000 Mbit/s gewählt werden kann.

Die andere Betriebsart, Halbduplex, wird verwendet, wenn die Computer mit einem Hub anstatt über einen Switch verbunden werden. Ein Hub puffert die eingehenden Rahmen nicht. Stattdessen verbindet er elektronisch alle Leitungen intern und simuliert so ein Multidrop-Kabel, wie es im klassischen Ethernet verwendet wird. Bei dieser Betriebsart können Kollisionen auftreten, sodass hier das CSMA/CD-Standardprotokoll erforderlich ist. Da ein 64-Byte-Rahmen (der kleinste erlaubte Rahmen) nun hundertmal schneller übertragen werden kann als beim klassischen Ethernet, muss die maximale Kabellänge hundertmal geringer (oder 25 m) betragen, um die wesentliche Eigenschaft zu wahren, dass der Sender immer noch überträgt, wenn ein Burst-Signal zurückkommt – selbst im schlechtesten Fall. Bei einem 2 500 m langen Kabel wäre der Sender eines 64-Byte-Rahmens bei 1 Gbit/s längst fertig, bevor der Rahmen auch nur ein Zehntel seines Weges zum Ziel zurückgelegt hat.

Diese Längenbeschränkung war so schmerhaft, dass dem Standard zwei Funktionen hinzugefügt wurden, um die maximale Kabellänge auf 200 m zu erhöhen, was für die meisten Büros vermutlich ausreichend ist. Die erste Funktion namens **Carrier Extension** (Erweiterung des Trägers) weist die Hardware an, ein eigenes Füllverfahren anzuwenden, um den normalen Rahmen auf 512 Byte zu erweitern. Da die sendende Hardware die Bytes auffüllt und die empfangende Hardware diese wieder entfernt, weiß die Software nichts darüber. Dies bedeutet, dass keine Änderungen an der vorhandenen Software erforderlich sind. Der Nachteil ist, dass die Übertragung von 46 Byte Nutzerdaten (die Nutzdaten in einem 64-Byte-Rahmen) unter Verwendung von 512 Byte Bandbreite eine Leistungseffizienz von nur 9 % hat.

Die zweite Funktion namens **Frame-Bursting** (Rahmen-Burst-Übertragung) erlaubt einem Sender eine miteinander verknüpfte Folge mehrerer Rahmen in einer Übertragung zu senden. Beträgt der gesamte Burst unter 512 Byte, füllt dies die Hardware wieder auf. Wenn ausreichend Rahmen auf eine Übertragung warten, ist dieses Schema sehr effizient und der Carrier Extension vorzuziehen.

Zugegebenermaßen ist es schwer vorstellbar, dass ein Unternehmen moderne Rechner mit Gigabit-Ethernet-Karten kauft und diese dann über einen almodischen Hub verbindet, um das klassische Ethernet mit allen Kollisionen zu simulieren. Gigabit-Ethernet-Schnittstellen und Switches waren früher teuer, doch die Preise dafür fallen schnell mit steigenden Verkaufszahlen. Dennoch ist die Abwärtskompatibilität in der Computerbranche immer noch eine heilige Kuh, sodass das Komitee dies berücksichtigen musste. Heute werden die meisten Rechner mit einer Ethernet-Schnittstelle ausgeliefert, die 10-, 100- und 1 000-Mbit/s-Betrieb zur Verfügung stellen kann und mit allen kompatibel ist.

Gigabit-Ethernet unterstützt sowohl Kupfer- als auch Glasfaserkabel, wie in ▶ Abbildung 4.21 aufgelistet. Die Signalübertragung mit nahezu 1 Gbit/s erfordert Codierung und das Senden von einem Bit pro Nanosekunde. Dieser Trick wurde anfangs mit kurzen, abgeschirmten Kupferkabeln (die 1000Base-CX-Version) und Glasfasern bewerkstelligt. Für Glasfaser sind zwei Wellenlängen zulässig, die zu zwei verschiedenen Versionen führen: 0,85 µm (kurz, für 1000Base-SX) und 1,3 µm (lang, für 1000Base-LX).

| Name | Kabel | Max. Segment | Vorteile |
|-------------|------------|--------------|--|
| 1000Base-SX | Glasfaser | 550 m | Multimode-Faser (50, 62,5 µm) |
| 1000Base-LX | Glasfaser | 5000 m | Single- (10 µm) oder Multimode-Faser (50, 62,5 µm) |
| 1000Base-CX | 2 Paar STP | 25 m | Abgeschirmtes Twisted-Pair-Kabel |
| 1000Base-T | 4 Paar UTP | 100 m | Standard-UTP der Kategorie 5 |

Abbildung 4.21: Kabelarten für Gigabit-Ethernet.

Die Signalgebung bei kurzer Wellenlänge kann mit billigeren LEDs erreicht werden. Diese werden mit Multimode-Fasern benutzt und sind bei Verbindungen innerhalb

eines Gebäudes hilfreich, da sie bis zu 500 m für 50-µm-Faser laufen können. Signalgebung bei langen Wellenlängen erfordert teurere Laser. Andererseits kann hier die Kabellänge bis zu 5 km betragen, wenn die Laser mit (10-µm-)Singlemode-Fasern kombiniert werden. Diese Begrenzung ermöglicht Verbindungen über große Entfernnungen zwischen Gebäuden als dedizierte Punkt-zu-Punkt-Verbindung, zum Beispiel bei einem Campus-Backbone. Spätere Variationen des Standards erlaubten sogar längere Verbindungen über Singlemode-Fasern.

Um Bits über diese Versionen von Gigabit-Ethernet zu senden, wurde die **8B/10B**-Codierung, die wir in Abschnitt 2.5 beschrieben haben, von einer weiteren Technologie namens Fibre Channel übernommen. Das Schema codiert 8 Datenbits in 10-Bit-Codewörter, die über das Kabel oder die Glasfaser gesendet werden, daher der Name 8B/10B. Die Codewörter werden so ausgewählt, dass sie balanciert werden können (d.h., sie haben die gleiche Anzahl an Nullen und Einsen), mit ausreichend Übergängen zur Taktrückgewinnung. Das Senden der codierten Bits mit NRZ erfordert eine Signalbandbreite von 25 % mehr als für die uncodierten Bits erforderlich ist, was eine große Verbesserung gegenüber der Bandbreitenerhöhung von 100 % bei der Manchester-Codierung bedeutet.

All diese Optionen erfordern neue Kupfer- oder Glasfaserkabel, um schnellere Signalgebung zu unterstützen. Keine davon benutzte die große Menge an Kategorie-5-UTP-Kabeln, die zusammen mit Fast Ethernet installiert wurden. Innerhalb eines Jahres tauchte 1000Base-T auf, um diese Lücke zu füllen, und wurde seitdem zur beliebtesten Art von Gigabit-Ethernet. Anscheinend mögen es die Leute nicht, ihre Gebäude neu zu verkabeln.

Kompliziertere Signalgebung wird benötigt, um Ethernet mit 1 000 Mbit/s über Kategorie-5-Kabel laufen zu lassen. Zunächst werden alle vier verdrillten Aderpaare innerhalb des Twisted-Pair-Kabels benutzt und jedes Paar wird gleichzeitig in beiden Richtungen eingesetzt, indem digitale Signalverarbeitung zum Trennen der Signale benutzt wird. Über jedes Kabel werden fünf Spannungspiegel, die 2 Bits tragen, zur Signalgebung mit 125 MSymbol/s benutzt. Die Erzeugung der Symbole aus den Bits ist nicht ganz unkompliziert. Dazu wird Scrambling (für die Übergänge) eingesetzt, gefolgt von einem Fehlerkorrekturcode, in dem vier Werte in fünf Signalebenen eingebettet werden.

Die Geschwindigkeit von 1 Gbit/s ist ziemlich schnell. Wenn beispielsweise ein Empfänger für nur 1 ms mit einer anderen Aufgabe belegt ist und den Eingangspuffer bei einer Leitung nicht leert, können sich in diesem Zwischenraum bis zu 1953 Rahmen angesammelt haben. Ebenso sind Pufferüberläufe sehr wahrscheinlich, wenn ein Computer in einem Gigabit-Ethernet Daten zu einem Computer in einem klassischen Ethernet sendet. Deshalb unterstützt Gigabit-Ethernet Flusskontrolle. Bei diesem Mechanismus sendet das eine Ende einen speziellen Steuerrahmen an das andere Ende, um diesem mitzuteilen, dass für eine gewisse Zeit eine Pause eingelegt werden soll. Diese PAUSE-Steuerrahmen sind normale Ethernet-Rahmen mit einer Typangabe von 0x8 808. Pausen werden in Vielfachen der Zeit für die Übertragung des kleinsten Rahmens angegeben. Bei Gigabit-Ethernet beträgt die Zeiteinheit 512 ns, sodass Pausen bis zu 33,6 ms lang sein können.

Es gibt noch eine andere Erweiterung, die zusammen mit Gigabit-Ethernet eingeführt wurde. **Jumborahmen** erlauben es, dass Rahmen länger als 1 500 Byte sein können, in der Regel bis zu 9 KB. Diese Erweiterung ist herstellerabhängig. Sie wird vom Standard nicht beachtet, denn falls sie benutzt wird, dann ist Ethernet nicht mehr kompatibel mit früheren Versionen. Die meisten Anbieter unterstützen diese Erweiterung dennoch. Die Begründung dafür ist, dass 1 500 Byte eine kleine Einheit bei Gigabit-Geschwindigkeiten ist. Indem größere Informationsblöcke behandelt werden, kann die Datenrate verringert werden. Gleichzeitig reduzieren sich auch die dazugehörigen Verarbeitungsschritte, wie beispielsweise dem Prozessor mitzuteilen, dass ein Rahmen angekommen ist, oder das Aufsplitten und Neukombinieren von Nachrichten, die zu lang waren, um in einen Ethernet-Rahmen zu passen.

4.3.7 10-Gigabit-Ethernet

Sobald das Gigabit-Ethernet einmal standardisiert war, langweilte sich das IEEE-802-Komitee und wollte wieder zur Tat schreiten. Die IEEE gab ihm den Auftrag, mit dem 10-Gigabit-Ethernet zu beginnen. Diese Arbeit folgte im Großen und Ganzen dem gleichen Muster wie die vorhergehenden Ethernet-Standards, wobei die Standards für Glasfaser- und abgeschirmte Kupferkabel zuerst erschienen (2002 und 2004), gefolgt vom Standard für Kupfer-Twisted-Pair-Kabel im Jahr 2006.

10 Gbit/s ist eine wahrhaft enorme Geschwindigkeit, 1 000-mal schneller als das ursprüngliche Ethernet. Wo könnte diese benötigt werden? Die Antwort ist: innerhalb von Datenzentren und Telefonvermittlungen, um High-End-Router, Switches und Server zu verbinden, ebenso in Fernverbindungsleitungen mit hoher Bandbreite zwischen Büros, wodurch ganze MANs realisiert werden können, die auf Ethernet und Glasfaser basieren. Die Fernverbindungen benutzen Glasfaser, während die kurzen Verbindungen Kupfer oder Glasfaser einsetzen können.

Alle Versionen von 10-Gigabit-Ethernet unterstützen Vollduplexbetrieb. CSMA/CD ist nicht länger Teil des Entwurfs und die Standards konzentrieren sich auf die Einzelheiten der Bitübertragungsschichten, die mit sehr hoher Geschwindigkeit laufen können. Kompatibilität ist immer noch ein Thema, daher führen 10-Gigabit-Ethernet-Schnittstellen Autonegotiation durch und fallen auf die Geschwindigkeit zurück, die von beiden Leitungsenden maximal unterstützt wird.

Die Hauptarten von 10-Gigabit-Ethernet sind in ► Abbildung 4.22 aufgeführt. Multi-mode-Fasern mit der (kurzen) Wellenlänge von 0,85 µm werden für mittlere Entfernungen und Singlemode-Fasern mit 1,3 µm (lang) und 1,5 µm (erweitert) für weitere Entfernungen benutzt. 10GBase-ER kann für Entfernungen bis zu 40 km betrieben werden, damit eignet es sich für überregionale Anwendungen. All diese Versionen senden einen seriellen Informationsstrom, der durch Scrambling der Datenbits und anschließender Codierung mit einem **64B/66B**-Code erzeugt wird. Diese Codierung hat weniger Overhead als ein 8B/10B-Code.

| Name | Kabel | Max. Segment | Vorteile |
|-------------|----------------|--------------|--|
| 10GBase-SR | Glasfaser | Bis zu 300 m | Multimode-Faser ($0,85 \mu\text{m}$) |
| 10GBase-LR | Glasfaser | 10 km | Singlemode-Faser ($1,3 \mu\text{m}$) |
| 10GBase-ER | Glasfaser | 40 km | Singlemode-Faser ($1,5 \mu\text{m}$) |
| 10GBase-CX4 | 4 Twinax-Paare | 15 m | Twinaxial-Kupfer |
| 10GBase-T | 4 UTP-Paare | 100 m | UTP der Kategorie 6a |

Abbildung 4.22: Kabelarten für 10-Gigabit-Ethernet.

Die erste definierte Kupferversion, 10GBase-CX4, verwendet ein Kabel mit vier Paaren verdrillter Kupferadern. Jedes Paar benutzt 8B/10B-Codierung und läuft mit 3,125 GSymbol/s, um 10 Gbit/s zu erreichen. Diese Version ist billiger als Glasfaser und wurde früh vermarktet, aber es bleibt abzuwarten, ob es sich auf lange Sicht gegen 10-Gigabit-Ethernet über eher gewöhnliche Twisted-Pair-Verkabelung durchsetzen wird.

10GBase-T ist die Version, die UTP-Kabel benutzt. Während Verkabelung der Kategorie 6a nötig ist, kann es für kürzere Strecken niedrigere Kategorien (einschließlich Kategorie 5) benutzen, um eine gewisse Wiederverwendung der installierten Kabel zu erlauben. Es ist keine Überraschung, dass die Bitübertragungsschicht einen großen Anteil daran hat, 10 Gbit/s über Twisted-Pair-Kabel zu erreichen. Wir werden daher nur einige der Details der höheren Schichten anreißen. Jedes der vier verdrillten Adern wird benutzt, um 2 500 Mbit/s in beiden Richtungen zu senden. Diese Geschwindigkeit wird erreicht, indem eine Signalrate von 800 MSymbol/s eingesetzt wird, wobei 16 Spannungspegel für die Symbole verwendet werden. Die Symbole werden durch Scrambling der Daten erzeugt, mit einem LDPC-Code (*Low Density Parity Check*) geschützt und zur Fehlerkorrektur weiter codiert.

10-Gigabit-Ethernet wird noch vom Markt sondiert, aber das IEEE-802.3-Komitee hat sich schon weiterbewegt. Ende 2007 hat IEEE eine Gruppe gegründet, um den Betrieb von Ethernet mit 40 Gbit/s und 100 Gbit/s zu standardisieren. Durch diese Verbesserung wird Ethernet auch in hochperformanten Umgebungen konkurrieren können, dazu gehören Fernverbindungen in Backbonenetzwerken und kurze Verbindungen über die Backplanes der Geräte. Der Standard ist zurzeit noch nicht vervollständigt, doch proprietäre Produkte sind bereits verfügbar.

4.3.8 Das Ethernet – ein Rückblick

Das Ethernet gibt es seit über 30 Jahren und es sind noch keine ernsthaften Konkurrenten in Sicht, sodass es uns auch noch einige Jahre erhalten bleiben wird. Nur wenige CPU-Architekturen, Betriebssysteme oder Programmiersprachen konnten sich so erfolgreich über drei Jahrzehnte halten – und Ethernet wächst weiter. Also hat Ethernet doch etwas richtig gemacht. Und was?

Der Hauptgrund für die Langlebigkeit von Ethernet ist höchstwahrscheinlich, dass es einfach und flexibel ist. In der Praxis bedeutet „einfach“: zuverlässig, kostengünstig und leicht zu warten. Nachdem die Hub- und Switch-Architektur angenommen wurde, waren Ausfälle sehr selten. Viele Leute zögern, etwas zu ersetzen, das prima funktioniert, besonders da ja allgemein bekannt ist, dass viele Dinge in der Computerindustrie mehr recht als schlecht funktionieren, sodass viele „Upgrades“ schlechter sind als das, was sie ersetzen.

„Einfach“ kann auch mit kostengünstig übersetzt werden. Twisted-Pair-Verkabelung ist relativ günstig, genauso wie die Hardwarekomponenten. Bei einem Wandel werden diese zunächst vermutlich teuer sein, zum Beispiel neue Gigabit-Ethernetkarten oder -Switches, aber das sind lediglich Ergänzungen zu einem gut etablierten Netz (keine Ersetzung desselben), und die Preise fallen schnell, sobald die Verkaufszahlen steigen.

Ethernet ist einfach zu warten. Man muss außer den Treibern keine Software installieren und nicht viele Konfigurationstabellen verwalten (und dabei Fehler machen). Darüber hinaus können auch neue Hosts problemlos – durch einfaches Einsticken – hinzugefügt werden.

Ein weiterer Punkt ist, dass Ethernet gut mit TCP/IP zusammenarbeitet, welches sich durchgesetzt hat. IP ist ein verbindungsloses Protokoll, sodass es hervorragend zu Ethernet passt, das ebenfalls verbindungslos ist. IP passt viel weniger gut zu verbindungsorientierten Alternativen wie ATM. Dies mindert die Chancen von ATM erheblich.

Schließlich, und das ist vielleicht das wichtigste, war Ethernet in der Lage, sich an bestimmten wichtigen Punkten weiterzuentwickeln. Die Geschwindigkeiten sind um mehrere Größenordnungen gestiegen, Hubs und Switches sind hinzugekommen, aber diese Veränderungen haben keine Änderung an der Software nach sich gezogen. Häufig war es möglich, die bestehende Verkabelung eine Zeitlang weiterzubenutzen. Wenn ein Verkaufsberater für Netze bei einer großen Installation sagt: „Ich habe hier ein fantastisches Netz für Sie. Sie müssen nur die gesamte Hardware ersetzen und die Software umschreiben“, dann hat er ein Problem.

Viele alternative Technologien, von denen Sie vermutlich noch nicht einmal gehört haben, waren bei ihrer Einführung schneller als Ethernet. Außer ATM enthält diese Liste auch FDDI (*Fiber Distributed Data Interface*) und Fibre Channel², zwei ringbasierte optische LANs. Beide waren nicht mit Ethernet kompatibel. Keiner der beiden hat es geschafft. Sie waren zu kompliziert, was zu komplexen Chips und hohen Preisen führte. Die Lektion, die hier gelernt werden sollte, lautete: KISS (Keep It Simple, Stupid). Im Laufe der Zeit legte Ethernet an Geschwindigkeit zu, häufig indem einige ihrer Technologien übernommen wurden, zum Beispiel die 4B/5B-Codierung von FDDI und die 8B/10B-Codierung von Fibre Channel. Damit hatten die anderen keine Vorteile mehr und starben leise weg oder fielen in spezialisierte Rollen.

2 Die Schreibweise „Fibre Channel“ statt „Fiber Channel“ kommt daher, weil der Lektor der Dokumentation ein Engländer war.

Es sieht so aus, als würde das Wachstum der Ethernet-Anwendungen noch eine Zeitlang andauern. 10-Gigabit-Ethernet brachte die Befreiung von den Entfernungs-einschränkungen durch CSMA/CD. Es wird viel Mühe in **Carrier-grade-Ethernet** gesteckt, damit Netzbetreiber ihren Kunden Ethernet-basierte Dienste für MANs und WANs anbieten können (Fouli und Maler, 2009). Diese Anwendung überträgt Ethernet-Rahmen über große Entfernnungen über Glasfaser und benötigt bessere Verwaltungsfunktionen, um Betreiber dabei zu unterstützen, zuverlässige, hoch qualitative Dienste anzubieten. Extrem schnelle Netze finden außerdem in Backplanes zur Verbindung von Komponenten in großen Routern oder Servern Anwendung. Beide Nutzungsarten werden zusätzlich zum Senden von Rahmen zwischen Rechnern in Büros verwendet.

4.4 Drahtlose LANs

Drahtlose LANs werden immer beliebter und Privathaushalte, Büros, Cafés, Bibliotheken, Flughäfen, Zoos und andere öffentliche Orte werden damit ausgerüstet, um Computer, PDAs und Smartphones mit dem Internet zu verbinden. Drahtlose LANs können auch benutzt werden, damit zwei oder mehrere nah beieinander stehende Rechner kommunizieren können, ohne das Internet benutzen zu müssen.

Der wichtigste LAN-Standard ist IEEE 802.11. Einige Hintergrundinformationen darüber haben wir bereits in Abschnitt 1.5.3 behandelt. Nun wollen wir uns die Technologie etwas genauer ansehen. In den folgenden Abschnitten untersuchen wir den Protokollstapel, die Funkübertragungsverfahren auf der Bitübertragungsschicht, das MAC-Teilschichtprotokoll, die Rahmenstruktur und die zur Verfügung gestellten Dienste. Weitere Informationen zu IEEE 802.11 finden Sie in Gast (2005). Wenn Sie Informationen aus erster Hand bekommen möchten, können Sie auch den veröffentlichten Standard IEEE-802.11-2007 selbst konsultieren.

4.4.1 IEEE-802.11-Architektur und -Protokollstapel

IEEE-802.11-Netze können in zwei Modi betrieben werden. Der bekannteste Modus ist, Clients wie Laptops und Smartphones mit anderen Netzen, zum Beispiel einem Firmenintranet oder dem Internet, zu verbinden. Dieser Modus ist in ►Abbildung 4.23a dargestellt. Im Infrastrukturmodus ist jeder Client einem **Zugangspunkt** (*Access Point, AP*) zugeordnet, der wiederum mit dem anderen Netz verbunden ist. Der Client sendet und empfängt seine Pakete über den Zugangspunkt. Mehrere Zugangspunkte können miteinander verbunden werden, in der Regel über ein verkabeltes **Verteilernetz** (*distribution system*), um ein erweitertes IEEE-802.11-Netz zu bilden. In diesem Fall können Clients Rahmen zu anderen Clients über ihre Zugangspunkte senden.

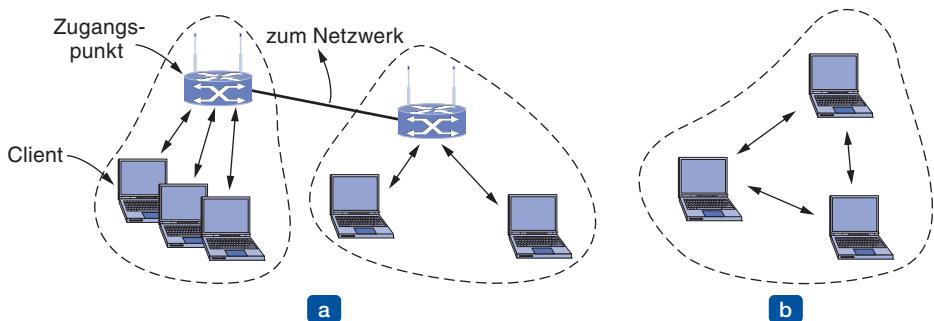


Abbildung 4.23: IEEE-802.11-Architektur. (a) Infrastrukturmodus. (b) Ad-hoc-Modus.

Der andere Modus (►Abbildung 4.23b) ist ein **Ad-hoc-Netzwerk**. Dieser Modus ist eine Ansammlung von Rechnern, die verbunden sind, sodass sie direkt Rahmen zueinander senden können. Es gibt keinen Zugangspunkt. Da Internetzugang die Killeranwendung für drahtlose Netze ist, sind Ad-hoc-Netze nicht sehr verbreitet.

Nun wenden wir uns den Protokollen zu. Alle IEEE-802-Protokolle – einschließlich IEEE 802.11 und Ethernet – haben eine gewisse Gemeinsamkeit in den Strukturen. ►Abbildung 4.24 zeigt eine teilweise Darstellung des IEEE-802.11-Protokollstapels. Der Stapel ist für Clients und Zugangspunkte derselbe. Die Bitübertragungsschicht entspricht im Großen und Ganzen der OSI-Bitübertragungsschicht, aber die Sicherungsschicht ist in allen IEEE-802-Protokollen in zwei oder mehr Teilschichten aufgeteilt. In IEEE 802.11 bestimmt die MAC-Teilschicht, wie der Kanal zugewiesen wird und wer als Nächster übertragen darf. Darüber liegt die LLC-Teilschicht (*Logical Link Control*), die die Unterschiede zwischen den verschiedenen IEEE-802-Varianten verbergen und sie für die Vermittlungsschicht gleich darstellen muss. Dies könnte eine bedeutende Verantwortung gewesen sein, aber heutzutage ist LLC eine Art Verbindungsschicht, welche das Protokoll (z.B. IP) identifiziert, das innerhalb des IEEE-802.11-Rahmens transportiert wird.



Abbildung 4.24: Teil des IEEE-802.11-Protokollstapels.

Verschiedene Übertragungstechniken wurden der Bitübertragungsschicht hinzugefügt, als IEEE 802.11 sich weiterentwickelte, seitdem es erstmals 1997 erschienen ist. Zwei der anfänglichen Techniken – Infrarot in der Art von Fernsehfernbedienungen und Frequenzspreizung im 2,4-GHz-Band – wurden erweitert, um zu Raten mit bis zu 11 Mbit/s zu laufen, und wurden schnell zum Hit. Heute kennt man diese Technik unter dem Namen IEEE 802.11b.

Um Drahtlos-Junkies den ersehnten Geschwindigkeitsschub zu geben, wurden in den Jahren 1999 und 2003 neue Übertragungstechniken eingeführt, die auf dem OFDM-Schema (*Orthogonal Frequency Division Multiplexing*) basieren, das wir in Abschnitt 2.5.3 beschrieben haben. Die erste Technik heißt IEEE 802.11a und benutzt ein unterschiedliches Frequenzband, 5 GHz. Die zweite Technik – IEEE 802.11g – hielt an 2,4 GHz und Kompatibilität fest. Beide liefern Datenraten bis zu 54 Mbit/s.

Zuletzt wurden im Oktober 2009 Übertragungstechniken, die gleichzeitig mehrere Antennen beim Sender und Empfänger für eine Steigerung der Geschwindigkeit benutzen, unter dem Namen IEEE 802.11n zum Abschluss gebracht. Mit vier Antennen und breiten Kanälen definiert der IEEE-802.11-Standard nun Datenraten bis zu verblüffenden 600 Mbit/s.

Wir wollen nun kurz jede dieser Übertragungstechniken untersuchen. Wir werden jedoch nur diejenigen behandeln, die tatsächlich verwendet werden, und die veralteten IEEE-802.11-Übertragungsmethoden auslassen. Von der technischen Seite her gehören diese Methoden eigentlich zur Bitübertragungsschicht und sollten daher in *Kapitel 2* behandelt werden. Da sie aber eng mit LANs im Allgemeinen und den IEEE-802.11-LANs im Besonderen verknüpft sind, behandeln wir sie stattdessen hier.

4.4.2 Die IEEE-802.11-Bitübertragungsschicht

Jede der Übertragungstechniken unterstützt das Senden eines MAC-Rahmens über den Äther von einer Station zur nächsten. Sie unterscheiden sich jedoch in der verwendeten Technologie und den verfügbaren Geschwindigkeiten. Eine detaillierte Erörterung dieser Technologien geht weit über den Rahmen dieses Buches hinaus. Dennoch wollen wir ein paar Worte zu jeder dieser Techniken verlieren, um sie mit den in Abschnitt 2.5 behandelten Themen zu verknüpfen und um den interessierten Lesern die Schlüsselbegriffe an die Hand zu geben, mit denen sie an anderen Stellen nach weiterführender Information suchen können.

All diese IEEE-802.11-Techniken verwenden Kurzstreckenfunk, um Signale entweder in 2,4-GHz- oder in 5-GHz-ISM-Frequenzbändern zu senden, die beide in Abschnitt 2.3.3 beschrieben wurden. Diese Bänder haben den Vorteil, unlizenziert und somit frei verfügbar für jeden Sender zu sein, der willens ist, ein paar Einschränkungen in Kauf zu nehmen, wie Strahlungsleistung von höchstens 1 W (obwohl 50 mW typischer für drahtlose WiFi-basierte LANs ist). Unglücklicherweise ist diese Tatsache auch den Herstellern von elektrischen Garagentoröffnern, schnurlosen Telefonen, Mikrowellenherden und unzähligen anderen Geräten bekannt, die alle mit Laptops um dasselbe Frequenzspektrum konkurrieren. Das 2,4-GHz-Band ist tendenziell über-

füllter als das 5-GHz-Band, sodass 5 GHz für einige Anwendungen besser sein kann, obwohl die Reichweite aufgrund der höheren Frequenz kürzer ist.

All diese Übertragungsmethoden definieren mehrere Datenraten. Die Idee ist, je nach aktueller Bedingung unterschiedliche Raten zu verwenden. Falls das drahtlose Signal schwach ist, kann eine niedrige Rate benutzt werden. Falls das Signal klar ist, kann die höchste Rate verwendet werden. Diese Justierung wird **Ratenanpassung (rate adaptation)** genannt. Da die Raten um den Faktor 10 oder mehr variieren, ist eine gute Ratenanpassung für gute Performanz wichtig. Da dies natürlich nicht für die Interoperabilität benötigt wird, sagen die Standards nichts darüber aus, wie die Ratenanpassung durchgeführt werden sollte.

Die erste Übertragungsmethode, die wir uns ansehen wollen, ist **IEEE 802.11b**. Dies ist ein Spreizspektrumverfahren, das Raten von 1, 2, 5,5 und 11 Mbit/s unterstützt, obwohl es in der Praxis fast immer mit 11 Mbit/s betrieben wird. Es ähnelt dem CDMA-System, das wir in Abschnitt 2.5 untersucht haben, bis auf die Tatsache, dass es sich hier um nur einen Spreizcode handelt, der von allen Nutzern gemeinsam verwendet wird. Frequenzspreizung wird eingesetzt, um die FCC-Anforderung zu erfüllen, dass die Leistung über das ISM-Band verteilt wird. Die Spreizfolge, die von IEEE 802.11b benutzt wird, ist eine **Barker-Folge**. Diese hat die Eigenschaft, dass ihre Autokorrelation niedrig ist, solange die Folgen nicht ausgerichtet sind. Diese Eigenschaft ermöglicht es einem Empfänger, am Beginn einer Übertragung zu sperren. Um mit einer Rate von 1 Mbit/s zu senden, wird die Barker-Folge mit BPSK-Modulation eingesetzt, um 1 Bit pro 11 Chips zu senden. Die Chips werden zu einer Rate von 11 MChips/s übertragen. Um mit 2 Mbit/s zu senden, wird die Barker-Folge mit QPSK-Modulation benutzt, um 2 Bits pro 11 Chips zu senden. Mit den höheren Raten verhält es sich anders. Diese Raten benutzen eine Technik namens **CCK** (*Complementary Code Keying*), um Codes anstelle der Barker-Folge zu erstellen. Die 5,5-Mbit/s-Rate sendet 4 Bits in jedem 8-Chip-Code und die 11-Mbit/s-Rate sendet 8 Bits in jedem 8-Chip-Code.

Als Nächstes kommen wir zu **IEEE 802.11a**, welches Raten bis zu 54 Mbit/s im 5-GHz-ISM-Band unterstützt. Man könnte annehmen, dass IEEE 802.11a vor IEEE 802.11b kommt, aber dies war nicht der Fall. Obwohl die IEEE-802.11a-Gruppe früher zusammengestellt wurde, wurde der IEEE-802.11b-Standard zuerst genehmigt und seine Produkte konnten weit vor den IEEE-802.11a-Produkten auf den Markt gebracht werden, was teilweise an der Schwierigkeit lag, im höheren 5-GHz-Band zu operieren.

Die IEEE-802.11a-Methode basiert auf **OFDM** (*Orthogonal Frequency Division Multiplexing*), weil OFDM das Spektrum effizient nutzt und Einbußen bei drahtloser Signalqualität wie Mehrwegeempfang abfängt. Bits werden parallel über 52 Unterträger gesendet, wobei 48 zur Datenübertragung und 4 zur Synchronisation benutzt werden. Die Symboldauer beträgt $4\mu s$ und jedes Symbol sendet 1, 2, 4 oder 6 Bits. Die Bits sind zur Fehlerkorrektur zuerst mit einem binären Faltungscode codiert, sodass nur 1/2, 2/3 oder 3/4 der Bits nicht redundant sind. Mit unterschiedlichen Kombinationen kann IEEE 802.11a mit acht verschiedenen Raten laufen, von 6 bis 54 Mbit/s. Diese Raten sind signifikant schneller als die IEEE-802.11b-Raten, und es gibt weniger Interferenz

im 5-GHz-Band. IEEE 802.11b hat jedoch eine Reichweite, die ungefähr siebenmal größer ist als die von IEEE 802.11a, was in vielen Situationen wichtiger ist.

Selbst mit der größeren Reichweite hatten die 802.11b-Leute nicht vor, diesen Emporkömmling das Geschwindigkeitsrennen gewinnen zu lassen. Glücklicherweise ließ das FCC im Mai 2002 seine lange bestehende Regel fallen, die forderte, dass jede drahtlose Kommunikationseinrichtung, die im ISM-Band in den USA betrieben wird, Frequenzspreizung benutzt. Also begann man, an IEEE 802.11g zu arbeiten, was von der IEEE im Jahr 2003 genehmigt wurde. Dieses kopierte die OFDM-Modulationsverfahren von IEEE 802.11a, aber operierte im schmalen 2,4-GHz-ISM-Band zusammen mit IEEE 802.11b. Es bot dieselben Raten wie IEEE 802.11a (6 bis 54 Mbit/s) plus natürlich Kompatibilität mit allen IEEE-802.11b-Geräten, die gerade in der Nähe sind. All diese unterschiedlichen Wahlmöglichkeiten können für Kunden verwirrend sein, deshalb ist es üblich, dass Produkte IEEE 802.11a/b/g in einer einzigen Netzwerkkarte unterstützen.

Damit konnte sich das IEEE-Komitee jedoch nicht begnügen und begann, an einer Bitübertragungsschicht mit hohem Durchsatz namens IEEE 802.11n zu arbeiten. Diese wurde 2009 genehmigt. Das Ziel für IEEE 802.11n war ein Durchsatz von mindestens 100 Mbit/s, nachdem alle drahtlosen Overheads entfernt waren. Dieses Ziel verlangte eine Geschwindigkeitssteigerung mindestens um das Vierfache. Dazu verdoppelte das Komitee die Kanäle von 20 MHz auf 40 MHz und verringerte den Overhead bei der Rahmenbildung, indem zugelassen wurde, eine Gruppe von Rahmen zusammen zu senden. Noch signifikanter jedoch benutzt IEEE 802.11n bis zu vier Antennen, um bis zu vier Informationsströme gleichzeitig zu übermitteln. Die Signale der Ströme überlagern sich beim Empfänger, aber sie können mithilfe von **MIMO**-Kommunikationstechniken (*Multiple Input Multiple Output*) wieder getrennt werden. Der Einsatz von mehreren Antennen bringt einen großen Geschwindigkeitsschub bzw. bessere Reichweite und Zuverlässigkeit. MIMO ist wie OFDM eine dieser cleveren Kommunikationsideen, die den drahtlosen Entwurf verändern und über die wir wahrscheinlich in der Zukunft noch viel hören werden. Eine kurze Einführung in die Verwendung mehrerer Antennen finden Sie bei Halperin et al. (2010).

4.4.3 Das IEEE-802.11-MAC-Teilschichtprotokoll

Kehren wir nun von der Elektrotechnik wieder in die Computergefilde zurück. Das IEEE-802.11-MAC-Teilschichtprotokoll unterscheidet sich erheblich von dem des Ethernet. Dies liegt an zwei Faktoren, die grundlegend für drahtlose Kommunikation sind.

Zunächst einmal sind Funkwellen fast immer halbduplex, was bedeutet, dass sie nicht gleichzeitig auf einer einzelnen Frequenz übertragen und auf Rausch-Bursts abhören können. Das empfangene Signal kann leicht eine Millionen Mal schwächer als das übertragene Signal sein, es kann also nicht zur gleichen Zeit gehört werden. Bei Ethernet wartet eine Station, bis der Träger schweigt, und beginnt dann zu übertragen. Wenn diese Station während der Übertragung der ersten 64 Byte keinen Rausch-Burst zurückerhält, kann mit großer Wahrscheinlichkeit davon ausgegangen werden, dass der Rahmen korrekt übertragen wurde. Bei drahtlosen Übertragungen funktioniert dieser Mechanismus zur Kollisionserkennung nicht.

IEEE 802.11 versucht stattdessen, Kollisionen mithilfe eines Protokolls namens **CSMA/CA** (*CSMA with Collision Avoidance*) zu vermeiden. Dieses Protokoll ähnelt konzeptionell dem CSMA/CD bei Ethernet, es prüft den Kanal vor dem Senden und verwendet nach Kollisionen exponentiellen Backoff. Eine Station mit einem zu sendenden Rahmen beginnt mit einem zufälligen Backoff (außer in dem Fall, dass die Station den Kanal in letzter Zeit nicht benutzt hat und der Kanal frei ist). Die Station wartet nicht auf eine Kollision. Die Anzahl der Zeitscheiben, die gewartet wird, liegt im Bereich von 0 bis beispielsweise 15 (im Fall der OFDM-Bitübertragungsschicht). Die Station wartet, bis der Kanal frei ist, indem sie prüft, dass es eine kurze Zeitspanne (die DIFS genannt wird, wie wir gleich noch erklären werden) kein Signal gibt. Dann zählt die Station die Zeitscheiben rückwärts, in denen nichts passiert, und pausiert, wenn Rahmen gesendet werden. Die Station sendet ihren Rahmen, wenn der Zähler auf 0 steht. Kommt der Rahmen durch, dann sendet die Zielstation direkt eine kurze Bestätigung zurück. Aus dem Fehlen einer Bestätigung wird gefolgert, dass ein Fehler vorliegt, entweder eine Kollision oder etwas anderes. In diesem Fall verdoppelt der Sender die Backoff-Zeitspanne und versucht es erneut, indem er mit exponentiellem Backoff wie bei Ethernet weitermacht, bis der Rahmen erfolgreich übertragen wurde oder die maximale Anzahl an Neuübertragungen erreicht ist.

In ► Abbildung 4.25 ist ein Beispiel für eine zeitliche Abfolge zu sehen. Zuerst sendet Station A einen Rahmen. Während A sendet, werden B und C sendebereit. Sie stellen fest, dass der Kanal belegt ist, und warten darauf, dass er wieder frei wird. Kurz danach enthält A eine Bestätigung und der Kanal wird frei. Anstatt jedoch direkt einen Rahmen zu senden, der dann kollidieren würde, führen B und C einen Backoff durch. C wählt eine kurze Zeitspanne für den Backoff und sendet somit zuerst. Station B unterbricht ihren Countdown, während C den Kanal belegt, und fährt fort, nachdem C eine Bestätigung bekommen hat. Station B beendet ihren Backoff und sendet ihren Rahmen.

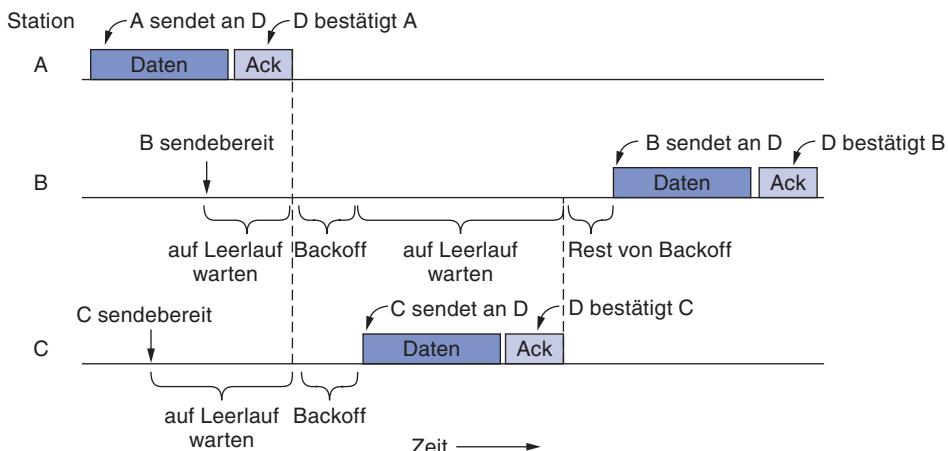


Abbildung 4.25: Senden eines Rahmens mit CSMA/CD.

Zu Ethernet gibt es zwei Hauptunterschiede. Zunächst einmal ist das frühe Starten von Backoffs hilfreich beim Vermeiden von Kollisionen. Dieses Vermeiden lohnt sich,

denn Kollisionen sind teuer, da der gesamte Rahmen übertragen wird, selbst wenn eine Kollision auftritt. Zweitens werden Bestätigungen benutzt, um auf Kollisionen zu schließen, da Kollisionen nicht entdeckt werden können.

Dieser Betriebsmodus heißt **DCF** (*Distributed Coordination Function*), weil jede Station unabhängig agiert, ohne eine Art von zentraler Steuerung. Der Standard enthält außerdem einen optionalen Betriebsmodus, **PCF** (*Point Coordination Function*), in dem der Zugangspunkt alle Aktivitäten in seiner Zelle steuert, genau wie eine zelluläre Basisstation. PCF wird jedoch in der Praxis nicht eingesetzt, weil es normalerweise keine Möglichkeit gibt, Stationen in anderen nahe gelegenen Netzen davon abzuhalten, konkurrierenden Datenverkehr zu senden.

Das zweite Problem ist, dass die Übertragungsbereiche von verschiedenen Stationen unterschiedlich sein können. Ein Kabelsystem ist so entworfen, dass alle Stationen sich gegenseitig hören können. Mit der Komplexität der RF-Weiterleitung gilt diese Situation nicht für drahtlose Stationen. Folglich können Situationen wie das bereits vorgestellte Hidden-Terminal-Problem auftreten, das noch einmal in ► Abbildung 4.26a dargestellt ist. Da nicht alle Stationen im gegenseitigen Funkbereich liegen, können Übertragungen, die in einem Teil einer Zelle stattfinden, an anderen Orten innerhalb derselben Zelle nicht empfangen werden. In diesem Beispiel sendet die Station C an Station B. Wenn A den Kanal prüft, hört A nichts und schließt fälschlicherweise daraus, dass nun mit der Übertragung zu B begonnen werden kann. Diese Entscheidung führt zu einer Kollision.

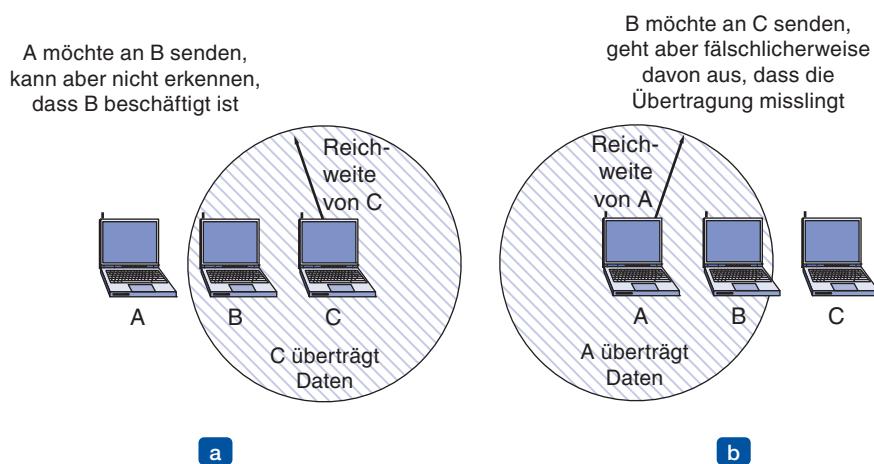


Abbildung 4.26: (a) Das Hidden-Terminal-Problem. (b) Das Exposed-Terminal-Problem.

Die umgekehrte Situation ist das Exposed-Terminal-Problem, das in ► Abbildung 4.26b dargestellt ist. Hier möchte B an C senden, daher prüft B den Kanal. Wenn eine Übertragung entdeckt wird, schließt B fälschlicherweise daraus, dass nicht an C gesendet werden darf, obwohl A vielleicht tatsächlich an D sendet (nicht dargestellt). Durch diese Entscheidung wird eine Übertragungsmöglichkeit verschwendet.

Um die Unklarheit, welche Station sendet, zu verringern, definiert IEEE 802.11, dass Kanalprüfung sowohl aus physischer als auch aus virtueller Prüfung besteht. Die physische Prüfung kontrolliert einfach das Medium, um festzustellen, ob es ein gültiges Signal gibt. Bei der virtuellen Prüfung verwaltet jede Station einen logischen Datensatz mit Informationen darüber, wann der Kanal benutzt wird, indem der sogenannte **Netzbelegungsvektor (NAV, Network Allocation Vector)** verfolgt wird. Jeder Rahmen trägt ein NAV-Feld, das angibt, wann die Folge, zu der dieser Rahmen gehört, beendet sein wird. Stationen, die diesen Rahmen mithören, wissen dann, dass der Kanal noch so lange belegt sein wird, wie durch den NAV angegeben ist – unabhängig davon, ob sie ein physisches Signal wahrnehmen können oder nicht. Zum Beispiel gibt der NAV eines Datenrahmens die Zeit an, die benötigt wird, um eine Bestätigung zu senden. Alle Stationen, die den Datenrahmen hören, werden die Bestätigungsphase abwarten, egal ob sie die Bestätigung hören können oder nicht.

Ein optionaler RTS/CTS-Mechanismus benutzt den NAV, um Stationen daran zu hindern, zur gleichen Zeit als verdeckte Terminals Rahmen zu senden. Dies ist in Abbildung 4.27 dargestellt. In diesem Beispiel möchte A Daten an B senden. C ist eine Station in der Reichweite von A (und eventuell in der Reichweite von B, dies spielt aber keine Rolle). D ist eine Station in der Reichweite von B, nicht aber in der Reichweite von A.

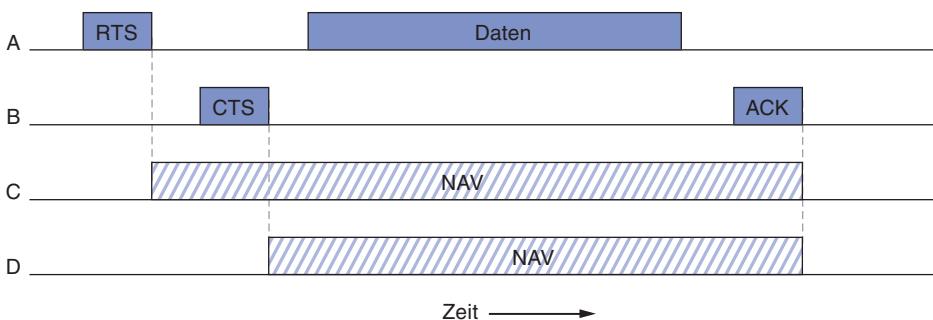


Abbildung 4.27: Virtuelle Kanalprüfung mit CSMA/CA.

Das Protokoll beginnt, wenn sich A entscheidet, Daten an B zu senden. A beginnt mit der Übertragung eines RTS-Rahmens an B, um die Erlaubnis anzufordern, einen Rahmen zu senden. Falls B diese Anforderung erhält, antwortet B mit einem CTS-Rahmen um anzugeben, dass der Kanal frei zum Senden ist. Nach Empfang des CTS sendet A den Rahmen und startet einen ACK-Timer. Wenn der Datenrahmen korrekt empfangen wurde, antwortet B mit einem ACK-Rahmen und vollendet den Austausch. Wenn der ACK-Timer von A abläuft, bevor der ACK-Rahmen zurückkommt, wird dies wie eine Kollision behandelt und das gesamte Protokoll wird nach einem Backoff erneut ausgeführt.

Betrachten wir nun diesen Informationsaustausch aus der Warte von C und D. C befindet sich in der Reichweite von A, sodass C den RTS-Rahmen empfangen kann. Wenn dies passiert, erkennt C, dass jemand in Kürze Daten senden möchte. Aus der Information in der RTS-Anforderung kann C ersehen, wie lange dies einschließlich der letzten

Bestätigung (ACK) dauern wird. Daher hält C sich zum Vorteil aller von sämtlichen Übertragungen zurück, bis der Austausch beendet ist. Dazu aktualisiert C seinen NAV und zeigt so an, dass der Kanal belegt ist (siehe Abbildung 4.27). D hört das RTS-Signal nicht, dafür aber das CTS, sodass D ebenfalls sein NAV aktualisiert. Beachten Sie, dass NAV-Signale nicht übertragen werden, sie dienen nur der internen Erinnerung, dass für eine gewisse Zeit Ruhe herrschen soll.

Obwohl sich RTS/CTS in der Theorie gut anhört, ist es eines der Entwürfe, die sich in der Praxis als wenig sinnvoll herausgestellt haben. Es gibt mehrere Gründe, warum dieses Verfahren selten benutzt wird. Es bringt wenig bei kurzen Rahmen (die anstelle des RTS gesendet werden) oder beim Zugangspunkt (den per definitionem jeder hören kann). In anderen Situationen verlangsamt es lediglich die Ausführung. RTS/CTS bei IEEE 802.11 funktioniert ein wenig anders als im MACA-Protokoll (das wir in Abschnitt 4.2 besprochen haben), weil jeder, der das RTS oder das CTS hört, während dieser Zeitspanne ruhig bleibt, um es dem ACK zu ermöglichen, ohne Kollisionen durchzukommen. Daher kann dieses Verfahren im Gegensatz zu MACA nicht zum Verhindern des Exposed-Terminal-Problems verwendet werden, es hilft nur bei versteckten Terminals. Meistens gibt es wenige versteckte Terminals und hier hilft bereits CSMA/CA, indem Stationen verlangsamt werden, die – aus welchem Grund auch immer – erfolglos übertragen, um die Wahrscheinlichkeit zu erhöhen, dass Übertragungen gelingen.

CSMA/CA mit physischer und virtueller Prüfung ist der Kern des IEEE-802.11-Protokolls. Es gibt jedoch mehrere andere Mechanismen, die dazu passend entwickelt wurden. Jeder dieser Mechanismen entstand aus Anforderungen aus dem Praxiseinsatz, daher werden wir diese kurz betrachten.

Die erste Praxisanforderung, die wir uns ansehen, ist Zuverlässigkeit. Im Gegensatz zu Kabelnetzen sind drahtlose Netze rauschbehaftet und unzuverlässig, was nicht unwe sentlich auf Interferenzen durch andere Gerätearten wie Mikrowellenherde zurückzuführen ist, die ebenfalls die unlizenzierten ISM-Bänder benutzen. Die Benutzung von Bestätigungen und Neuübertragungen ist wenig hilfreich, wenn die Wahrscheinlichkeit, einen Rahmen durchzubekommen, von Anfang an gering ist.

Die Hauptstrategie, die eingesetzt wird, um die Anzahl der erfolgreichen Übertragungen zu erhöhen, ist die Herabsetzung der Übertragungsrate. Langsamere Raten verwenden robustere Modulationen, für die die Wahrscheinlichkeit höher ist, dass sie bei einem gegebenen Signal-Rausch-Verhältnis korrekt empfangen werden. Wenn zu viele Rahmen verloren gehen, kann eine Station die Rate heruntersetzen. Wenn Rahmen mit wenig Verlust gesendet werden, kann eine Station von Zeit zu Zeit eine höhere Rate testen, um festzustellen, ob diese benutzt werden sollte.

Eine weitere Strategie, um die Wahrscheinlichkeit zu erhöhen, dass ein Rahmen unbeschädigt durchkommt, ist, kürzere Rahmen zu senden. Wenn die Wahrscheinlichkeit eines fehlerhaften Bits p ist, dann ist die Wahrscheinlichkeit, dass ein Rahmen mit n Bit völlig korrekt empfangen wird, $(1-p)^n$. Ein Beispiel: Für $p=10^{-4}$ liegt die Wahrscheinlichkeit, dass ein vollständiger Ethernet-Rahmen (12 144 Bit) korrekt ankommt, bei unter 30 %. Die meisten Rahmen gehen verloren. Aber falls die Rahmen nur ein Drittel

so lang sind (4 048 Bit), dann werden zwei Drittel davon korrekt empfangen. Nun kommen die meisten Rahmen durch und es sind weniger Neuübertragungen nötig.

Kürzere Rahmen können realisiert werden, indem die maximale Größe der Nachricht, die von der Vermittlungsschicht akzeptiert wird, reduziert wird. Alternativ können Rahmen bei IEEE 802.11 in kleinere Teile, sogenannte **Fragmente**, aufgeteilt werden, von denen jedes seine eigene Prüfsumme hat. Die Fragmentgröße ist durch den Standard nicht festgelegt, doch es ist ein Parameter, der vom Zugangspunkt eingestellt werden kann. Die Fragmente werden einzeln durchnummeriert und mit dem Stop-and-Wait-Protokoll bestätigt (d.h., der Sender darf Fragment $k+1$ erst dann übertragen, wenn er die Bestätigung für Fragment k erhalten hat). Wurde ein Kanal erst einmal belegt, dann können mehrere Fragmente als Burst gesendet werden. Einer nach dem anderen wird jeweils mit einer Bestätigung (und möglicherweise Neuübertragungen) dazwischen übertragen, bis entweder der ganze Rahmen erfolgreich gesendet wurde oder die Übertragungszeit die maximal erlaubte Zeit erreicht. Durch den NAV-Mechanismus werden die anderen Stationen nur bis zur nächsten Bestätigung ruhig gehalten, doch ein anderes Verfahren (siehe unten) wird benutzt, um das Senden eines Fragment-Bursts zu ermöglichen, ohne dass andere Stationen zwischendurch einen Rahmen übertragen können.

Die zweite Anforderung, die wir besprechen, ist Energiesparen. Die Lebensdauer von Batterien ist bei mobilen Geräten immer ein Thema. Beim IEEE-802.11-Standard wird auch die Stromversorgung berücksichtigt, sodass Clients keinen Strom verschwenden müssen, wenn sie weder Informationen zu senden noch zu empfangen haben.

Der Grundmechanismus zum Stromsparen basiert auf sogenannten **Beacon-Rahmen** (Leuchtfeuer). Beacons werden in regelmäßigen Abständen vom Zugangspunkt ausgesendet (z.B. alle 100 ms). Diese Rahmen kündigen allen Clients die Anwesenheit des Zugangspunkts an und übermitteln Systemparameter wie den Identifikator des Zugangspunkts, die Zeit bis zum nächsten Beacon sowie Sicherheitseinstellungen.

Clients können ein Bit zur Energieverwaltung in Rahmen setzen, die sie zum Zugangspunkt senden, um mitzuteilen, dass sie in den **Stromsparmodus** eintreten. In diesem Modus kann der Client dösen und der Zugangspunkt wird den Datenverkehr zwischenspeichern, der an diesen Client gerichtet ist. Um auf ankommenden Verkehr zu prüfen, wacht der Client für jedes Beacon auf und überprüft eine TIM (*Traffic Indication Map*), die als Teil des Beacons gesendet wird. Diese TIM teilt dem Client mit, ob es gespeicherte Daten für ihn gibt. Falls dies so ist, sendet der Client eine Polling-Nachricht an den Zugangspunkt, welcher dann den gespeicherten Datenverkehr sendet. Der Client kann sich dann wieder schlafen legen, bis das nächste Beacon gesendet wird.

Ein weiterer Mechanismus zum Energiesparen, **APSD** (*Automatic Power Save Delivery*), wurde 2005 ebenfalls zu IEEE 802.11 hinzugefügt. Mit diesem neuen Mechanismus puffert der Zugangspunkt Rahmen und sendet diese an einen Client, sobald der Client Rahmen zum Zugangspunkt sendet. Der Client kann dann schlafen gehen, bis er mehr Daten zum Senden (und Empfangen) hat. Dieser Mechanismus funktioniert gut für Anwendungen wie VoIP, bei denen viel Datenverkehr in beiden Richtungen

stattfindet. Zum Beispiel könnte ein schnurloses VoIP-Telefon dieses Verfahren einsetzen, um alle 20 ms Rahmen zu senden und zu empfangen, was sehr viel häufiger als das Beacon-Intervall mit 100 ms ist, während es in der Zwischenzeit schlummert.

Die dritte und letzte Anforderung, die wir untersuchen wollen, ist Dienstgüte. Konkurriert der VoIP-Verkehr im vorangehenden Beispiel mit Peer-to-Peer-Datenverkehr, dann wird der VoIP-Verkehr den Kürzeren ziehen. Aufgrund der Konkurrenz zum Peer-to-Peer-Datenverkehr mit hoher Bandbreite wird der VoIP-Verkehr verzögert, obgleich die VoIP-Bandbreite niedrig ist. Diese Verzögerungen mindern die Qualität der Sprachübertragung. Um dies zu verhindern, würden wir gerne dem VoIP- vor dem Peer-to-Peer-Datenverkehr den Vorzug geben, da dieser eine höhere Priorität hat.

IEEE 802.11 verfügt über einen intelligenten Mechanismus, um diese Art der Dienstgüte – im Jahr 2005 als eine Reihe von Erweiterungen unter dem Namen IEEE 802.11e eingeführt – zur Verfügung zu stellen. Dieses Verfahren erweitert CSMA/CA mittels vorsichtig defi nierter Intervalle zwischen den Rahmen. Nachdem ein Rahmen gesendet wurde, muss eine bestimmte Wartezeit eingehalten werden, bevor eine Station einen Rahmen zur Überprüfung senden darf, ob der Kanal wieder frei ist. Der Trick hierbei ist, unterschiedliche Zeitspannen für unterschiedliche Rahmenarten festzulegen.

In ► Abbildung 4.28 sind fünf Intervalle eingezeichnet. Das Intervall zwischen regulären Datenrahmen heißt DIFS (DCF InterFrame Spacing). Jede Station kann versuchen, den Kanal zu belegen, um einen neuen Rahmen zu senden, nachdem das Medium für eine DIFS-Länge im Leerlauf war. Es gelten hier die normalen Konkurrenzregeln und im Falle von Kollisionen kann das binäre exponentielle Backoff erforderlich sein. Das kürzeste Intervall ist SIFS (Short InterFrame Spacing). Es dient dazu, den Parteien in einem Dialog die Chance zu geben, zuerst zu senden. Zu den Beispielen gehört, dass der Empfänger ein ACK oder andere Steuerrahmenfolgen wie RTS und CTS senden kann, oder dass der Sender ein Fragment-Burst übertragen kann. Indem mit dem Senden des nächsten Fragments nur eine SIFS-Länge gewartet wird, werden die anderen Station daran gehindert, in den Datenaustausch hineinzuplatzen.

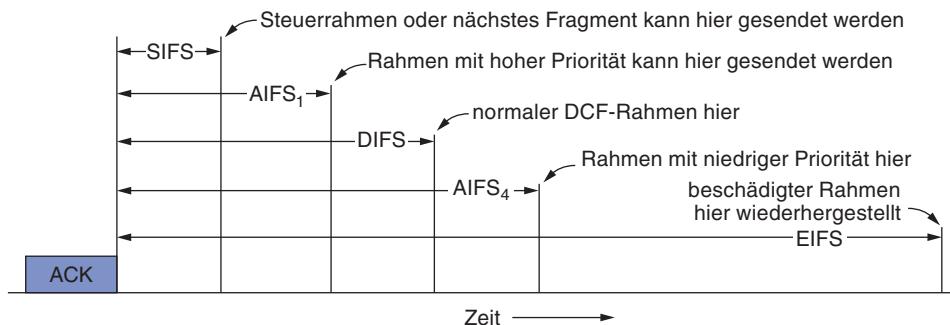


Abbildung 4.28: Abstand zwischen Rahmen bei IEEE 802.11.

Die zwei AIFS-Intervalle (Arbitration InterFrame Space) zeigen Beispiele von verschiedenen Prioritätsebenen. Das kurze Intervall, AIFS₁, ist kleiner als DIFS, aber länger als SIFS. Es kann vom Zugangspunkt benutzt werden, um Sprache oder anderen Datenver-

kehr mit hoher Priorität zum Kopf der Leitung zu bewegen. Der Zugangspunkt wird auf ein kürzeres Intervall warten, bevor er die Sprachübertragung beginnt, und diese somit vor dem regulären Datenverkehr senden. Das lange Intervall, AIFS₄, ist größer als DIFS. Es wird für Hintergrundverkehr benutzt, der bis nach dem regulären Datenverkehr hinausgezögert werden kann. Der Zugangspunkt wird auf ein längeres Intervall warten, bevor er diese Daten sendet, und gibt somit regulärem Verkehr die Möglichkeit, zuerst zu übertragen. Der vollständige Dienstgütemechanismus definiert vier unterschiedliche Prioritätsebenen, die jeweils verschiedene Backoff-Parameter sowie unterschiedliche Wartezeit-Parameter haben.

Das letzte Zeitintervall, EIFS (*Extended InterFrame Spacing*), wird nur von einer Station verwendet, die gerade einen fehlerhaften oder ungültigen Rahmen erhalten hat, um das Problem zu melden. Die Idee hierbei ist, dass der Empfänger, der eventuell nicht weiß, was gerade passiert, eine Weile warten soll, um keinen laufenden Dialog zwischen zwei Stationen zu stören.

Ein weiterer Teil der Dienstgüte-Erweiterungen ist der Begriff eines TXOP (*Transmit Opportunity*). Beim ursprünglichen CSMA/CA-Mechanismus können Stationen nur jeweils einen Rahmen senden. Dieser Entwurf war ausreichend, bis sich der Bereich der Raten vergrößerte. Bei IEEE 802.11a/g kann eine Station mit 6 Mbit/s senden und eine andere mit 54 Mbit/s. Sie senden jeweils einen Rahmen, aber die 6-Mbit/s-Station braucht zum Senden ihres Rahmens neunmal so lange (wenn man den festen Overhead ignoriert) wie die 54-Mbit/s-Station. Dieses Ungleichgewicht hat den unglücklichen Nebeneffekt, dass ein schneller Sender, der mit einem langsamen Sender konkurriert, ausgebremst wird, und zwar auf ungefähr die Rate des langsamen Senders. Wenn zum Beispiel – der feste Overhead wird wieder ignoriert – die 6-Mbit/s- und die 54-Mbit/s-Stationen jeweils alleine senden, werden sie ihre eigenen Raten bekommen, doch wenn sie zusammen senden, bekommen beide durchschnittlich 5,4 Mbit/s. Das ist eine harte Strafe für den schnellen Sender. Dieses Problem ist als **Ratenanomalität** bekannt (Heusse et al., 2003).

Mit TXOPs bekommt jede Station die gleiche Menge an Sendezeit, nicht die gleiche Anzahl an Rahmen. Stationen, die mit einer höheren Datenrate in ihrer Sendezeit senden, werden einen höheren Durchsatz erreichen. Wenn also in unserem Beispiel der 6-Mbit/s- und der 54-Mbit/s-Sender zusammen senden, bekommen sie nun 3 Mbit/s bzw. 27 Mbit/s.

4.4.4 IEEE-802.11-Rahmenstruktur

Der IEEE-802.11-Standard definiert drei verschiedene Rahmenklassen: Daten, Steuerung und Verwaltung. Jede dieser Klassen hat einen Header mit verschiedenen Feldern, die von der MAC-Teilschicht verwendet werden. Darüber hinaus werden auf der Bitübertragungsschicht verschiedene Header benutzt, wobei sich diese aber meistens mit verwendeten Modulationstechniken beschäftigen, sodass wir hier nicht näher darauf eingehen.

Zunächst wollen wir uns das Format des Datenrahmens als Beispiel ansehen. Es ist in Abbildung 4.29 dargestellt. Als Erstes kommt das Feld *Frame Control* (Rahmensteuerung), das aus 11 Unterfeldern besteht. Das erste ist hier *Protocol Version* (Protokollversion), das auf 00 gesetzt ist. Es ermöglicht zukünftigen Versionen von IEEE 802.11, gleichzeitig in der gleichen Zelle zu arbeiten. Dann folgt das Feld *Type* (Daten, Steuerung oder Verwaltung) und *Subtype* (z.B. RTS oder CTS). Bei einem regulären Datenrahmen (ohne Dienstgüte) werden diese Felder auf 10 und 0000 binär gesetzt. Die Bits *To DS* und *From DS* geben an, ob der Rahmen zu dem mit dem Zugangspunkt verbundenen Netz, dem Verteilungssystem, geht oder von dort kommt. Das *More Fragments*-Bit gibt an, dass weitere Fragmente folgen. Das Bit *Retry* gibt eine erneute Übertragung eines Rahmens an, der bereits früher gesendet wurde. Das Bit *Power Management* gibt an, dass der Sender in den Energiesparmodus übergeht. Das *More Data* gibt an, dass der Sender für den Empfänger weitere Rahmen hat. Das *Protected Frame*-Bit gibt an, dass der Rahmenhauptteil aus Sicherheitsgründen verschlüsselt wurde. Den Bereich Sicherheit werden wir im nächsten Abschnitt behandeln. Schließlich gibt das *Order*-Bit dem Empfänger an, dass die höhere Schicht die Rahmenfolge genau in dieser Reihenfolge erwartet.

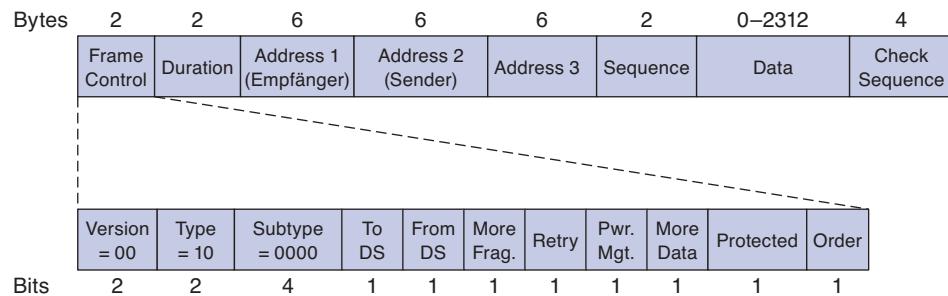


Abbildung 4.29: Format des IEEE-802.11-Datenrahmens.

Das zweite Feld des Datenrahmens ist *Duration*, es gibt an, wie lange der Rahmen und die dazugehörige Bestätigung den Kanal belegen werden, gemessen in Mikrosekunden. Es ist in allen Rahmenarten, einschließlich Steuerrahmen, vorhanden und gibt an, was Stationen benutzen, um den NAV-Mechanismus zu verwalten.

Als Nächstes kommen Adressen. Datenrahmen, die zu oder von einem Zugangspunkt gesendet werden, haben drei Adressen, die alle im IEEE-802-Standardformat sind. Die erste Adresse ist der Empfänger, die zweite der Sender. Diese beiden sind offensichtlich erforderlich, aber wofür ist die dritte Adresse? Vergessen Sie nicht, dass der Zugangspunkt einfach eine Schaltstelle für Rahmen ist, die sich zwischen einem Client und einem anderen Punkt des Netzes bewegen, vielleicht ein entfernter Client oder ein Internetportal. Die dritte Adresse benennt diesen entfernten Endpunkt.

Das *Sequence*-Feld nummeriert die Rahmen, sodass Duplikate entdeckt werden können. Von den 16 verfügbaren Bits identifizieren 4 das Fragment und 12 tragen eine Nummer, die mit jeder neuen Übertragung erhöht wird. Das Feld *Data* enthält die Nutzdaten bis zu 2 312 Byte. Die ersten Bytes dieser Nutzdaten sind in einem Format, das als **LLC** (*Logical Link Control*) bekannt ist. Diese Schicht identifiziert die Protokolle

höherer Schichten (z.B. IP), zu dem die Nutzdaten weitergeleitet werden sollen. Zuletzt kommt das Feld *Frame Check Sequence*, welches denselben 32-Bit-CRC-Wert enthält, den wir schon in Abschnitt 3.2.2 und anderswo getroffen haben.

Die Verwaltungsrahmen haben dasselbe Format wie Datenrahmen, plus einem Format für den Datenanteil, der mit dem Subtyp variiert (z.B. Parameter in Beacon-Rahmen). Steuerrahmen sind kurz. Wie alle Rahmen haben sie die Felder *Frame Control*, *Duration* und *Frame Check Sequence*. Es kann jedoch vorkommen, dass sie nur eine Adresse und keinen Datenteil haben. Der Großteil der Schlüsselinformation befindet sich im *Subtype*-Feld (z.B. ACK RTS und CTS).

4.4.5 Dienste

Der IEEE-802.11-Standard definiert die Dienste, sodass die Clients, die Zugangspunkte und das Netzwerk, das diese verbindet, ein konformes drahtlose LAN bilden müssen. Diese Dienste lassen sich zu mehreren Gruppen zusammenfassen.

Mit dem Dienst **Association** bauen Mobilstationen eine Verbindung zum Zugangspunkt auf. In der Regel wird dieser Dienst verwendet, wenn eine Station in die Reichweite eines Zugangspunktes eintritt. Beim Eintritt erfährt die Station die Identität und die Leistungsmerkmale des Zugangspunkts, entweder durch Beacon-Rahmen oder indem der Zugangspunkt direkt befragt wird. Zu den Leistungsmerkmalen gehören die unterstützten Datenübertragungsraten, Sicherheitseinrichtungen, Energiesparpotenzial, Unterstützung der Dienstgüte und vieles mehr. Die Station sendet eine Anfrage zur Verbindung mit dem Zugangspunkt. Dieser kann die Anfrage annehmen oder zurückweisen.

Mithilfe des Dienstes **Reassociation** kann eine Station ihren bevorzugten Zugangspunkt wechseln. Diese Einrichtung ist nützlich für Mobilstationen, die von einem Zugangspunkt zum nächsten im selben erweiterten IEEE-802.11-LAN wandern, wie bei einem Handover im zellulären Netz. Bei korrekter Handhabung gehen bei der Weiterreichung keine Daten verloren. (Aber IEEE 802.11 ist wie Ethernet nur ein Best-Effort-Dienst.) Entweder die Station oder der Zugangspunkt kann eine Trennung (**Dissociation**) herbeiführen, also die Beziehung lösen. Eine Station sollte diesen Dienst verwenden, bevor sie sich abschaltet oder das Netz verlässt. Der Zugangspunkt kann den Dienst verwenden, bevor er aus Wartungsgründen herunterfährt.

Stationen müssen sich außerdem **authentisieren**, bevor sie Rahmen über den Zugangspunkt senden können, aber Authentisierung wird auf unterschiedliche Arten behandelt, je nach Wahl des Sicherheitsschemas. Falls das IEEE-802.11-Netz „offen“ ist, dann darf es jeder benutzen. Andernfalls sind Berechtigungsnachweise zum Authentisieren erforderlich. Das empfohlene Verfahren, **WPA2** (*WiFi Protected Access 2*), implementiert Sicherheitsaspekte so, wie es im IEEE-802.11i-Standard definiert ist. (WPA selbst war eine vorläufige Version, die eine Teilmenge von IEEE 802.11i implementierte. Wir überspringen diese und gehen direkt zum vollständigen Schema.) Mit WPA2 kann der Zugangspunkt mit einem Authentisierungsserver sprechen, der einen

Benutzernamen und eine Passwortdatenbank hat, um festzustellen, ob es der Station erlaubt ist, Zugang zum Netz zu bekommen. Alternativ könnte ein sogenannter Pre-Shared Key (vorher vereinbarter Schlüssel) – was ein hochtrabender Name für ein Netzwerkpasswort ist – konfiguriert werden. Mehrere Rahmen werden zwischen der Station und dem Zugangspunkt ausgetauscht (Challenge-Response-Verfahren), womit die Station beweisen kann, dass sie die richtigen Berechtigungsnachweise hat. Dieser Austausch findet nach der Verknüpfung statt.

Das Verfahren, das vor WPA benutzt wurde, heißt **WEP** (*Wired Equivalent Privacy*). Bei diesem Verfahren findet die Authentifizierung mit einem vorher vereinbarten Schlüssel vor der Verknüpfung statt. Aufgrund der Schwachstellen im Entwurf, durch die WEP leicht zu kompromittieren war, wird jedoch von der Verwendung abgeraten. Die erste praktische Demonstration, dass WEP gebrochen werden kann, gelang Adam Stubblefield, ein Praktikant bei AT&T (Stubblefield et al., 2002). Er war in der Lage, einen Angriff in einer Woche zu programmieren und zu testen – wobei er viel Zeit damit zubrachte, die Erlaubnis von der Verwaltung einzuholen, die WiFi-Karten zu kaufen, die er für die Experimente benötigte. Software zum Knacken von WEP-Passwörtern ist heutzutage frei verfügbar.

Sobald Rahmen den Zugangspunkt erreichen, legt der Dienst **Distribution** fest, wie diese weitergeleitet werden. Ist das Ziel der lokale Zugangspunkt, dann können die Rahmen direkt auf dem Luftweg übertragen werden. Andernfalls müssen sie über das kabelgebundene Netz weitergeleitet werden. Der Dienst **Integration** verwaltet jede Übersetzung, die benötigt wird, um einen Rahmen außerhalb des IEEE-802.11-LAN zu senden bzw. von außerhalb zu empfangen. Der Regelfall ist hier die Verbindung des drahtlosen LAN mit dem Internet.

Letztendlich geht es um die Datenübertragung, sodass IEEE 802.11 natürlich auch einen Dienst **Datenzustellung** (*data delivery*) bereitstellt. Dieser Dienst ermöglicht es, dass Stationen Daten mithilfe der Protokolle, die wir in diesem Kapitel eingeführt haben, senden und empfangen. Da IEEE 802.11 nach dem Ethernet modelliert wurde und die Übertragungen im Ethernet nicht 100 %ig zuverlässig sind, gilt dies auch für die Übertragungen über IEEE 802.11. Die höheren Schichten müssen sich mit dem Entdecken und Korrigieren von Fehlern befassen.

Jedes drahtlose Signal ist ein Broadcast-Signal. Um Informationen, die über ein drahtloses LAN gesendet werden, vertraulich zu halten, müssen diese verschlüsselt werden. Dieses Ziel wird mithilfe eines Privacy-Dienstes erreicht, der die Einzelheiten der Ver- und Entschlüsselung verwaltet. Der Verschlüsselungsalgorithmus für WAP2 basiert auf **AES** (*Advanced Encryption Standard*), einem Standard der US-Regierung, der 2002 angenommen wurde. Die verwendeten Schlüssel werden während der Authentifizierungsprozedur festgelegt.

Um Datenverkehr mit unterschiedlichen Prioritäten zu handhaben, gibt es einen **QoS-Traffic-Scheduling**-Dienst. Dieser benutzt die beschriebenen Protokolle, um Sprach- und Videodatenverkehr vor Best-Effort- und Hintergrundverkehr zu bevorzugen. Ein begleitender Dienst stellt außerdem Timersynchronisation der höheren Schichten zur

Verfügung. Dadurch können die Stationen ihre Aktionen koordinieren, was bei Mediaverarbeitung hilfreich sein kann.

Schließlich gibt es zwei Dienste, welche die Stationen dabei unterstützen, ihre Verwendung des Spektrums zu verwalten. Der **Transmit-Power-Control**-Dienst gibt Stationen die nötigen Informationen, um regulatorische Einschränkungen der Sendeleistung einzuhalten, die von Region zu Region variieren. Durch den **Dynamic-Frequency-Selection**-Dienst bekommen Stationen die Information, die sie benötigen, um das Senden auf Frequenzen im 5-GHz-Band zu verhindern, welche für Radar in der Nähe benutzt werden.

Mit diesen Diensten stellt IEEE 802.11 ein reichhaltiges Angebot an Funktionalität zur Verfügung, um nahe gelegene mobile Clients mit dem Internet zu verbinden. IEEE 802.11 war ein riesiger Erfolg und der Standard wurde wiederholt abgeändert, um mehr Funktionalität hinzuzufügen. Ein Überblick, wie sich der Standard entwickelte und wohin der Weg geht, findet sich in Hiertz et al. (2010).

4.5 Drahtloses Breitband

Wir waren viel zu lange innerhalb von Gebäuden. Gehen wir nun nach draußen, wo sich bei den Netzen einiges Interessantes auf der sogenannten „letzten Meile“ tut. Durch die Deregulierung der Telefonsysteme in vielen Ländern können nun konkurrierende Unternehmen lokale Sprachdienste und den Hochgeschwindigkeitsinternetzugang bereitstellen. Hierfür gibt es natürlich einen großen Bedarf. Das Problem ist, dass die Verlegung von Glasfaser- oder Koaxialkabeln zu Millionen von Häusern und Unternehmen extrem teuer ist. Was soll ein konkurrierender Netzbetreiber hier tun?

Die Antwort lautet: drahtlose Breitbandnetze (*broadband wireless*) einsetzen. Wenn man eine große Antenne auf einem Hügel außerhalb einer Stadt aufbaut, ist dies viel einfacher und billiger, als quer durch die Stadt viele Gräben auszuheben und darin Kabel zu verlegen. Also haben Gesellschaften damit begonnen, drahtlose Kommunikationsdienste mit mehreren Megabit für Sprache, Internet, Movies on Demand etc. anzubieten.

Um den Markt anzuregen, hat IEEE eine Gruppe zur Standardisierung eines drahtlosen Breitband-MAN gebildet. Die nächste freie Nummer im Rahmen der 802-Numerierung war **IEEE 802.16**, sodass der Standard diesen Namen erhielt. Informell wird die Technologie **WiMAX** (*Worldwide Interoperability for Microwave Access*) genannt. Wir werden die Begriffe IEEE 802.16 und WiMAX synonym verwenden.

Der erste IEEE-802.16-Standard wurde im Dezember 2001 herausgegeben. Frühe Versionen boten eine drahtlose Teilnehmeranschlussleitung zwischen festen Punkten mit einer Sichtverbindung zueinander. Dieser Entwurf wurde bald verändert, damit WiMAX beim Internetzugang besser mit Kabel und DSL konkurriren konnte. Bis Januar 2003 wurde IEEE 802.16 überarbeitet, um Nicht-Sichtverbindungen zu unterstützen, welche OFDM-Technologie mit Frequenzen zwischen 2 GHz und 10 GHz verwenden. Diese Änderung vereinfachte die Installation wesentlich, obwohl Stationen immer noch feste Standorte waren. Das Aufkommen von zellulären 3G-Netzen stellte

eine Bedrohung dar, da diese hohe Datenraten *und* Mobilität versprachen. Als Reaktion darauf wurde IEEE 802.16 bis Dezember 2005 erneut erweitert, um Mobilität mit Fahrzeuggeschwindigkeiten zu ermöglichen. Mobiler Breitbandinternetzugang ist das Ziel des aktuellen Standards, IEEE 802.16-2009.

Wie die anderen IEEE-802-Standards wurde IEEE 802.16 sehr stark vom OSI-Modell beeinflusst, einschließlich der (Teil-)Schichten, der Terminologie, der Dienstprimitiven und anderem mehr. Unglücklicherweise ist es genau wie OSI ziemlich kompliziert. Das [WiMAX Forum](#) wurde ins Leben gerufen, um voll kompatible Teilmengen des Standards für kommerzielle Angebote festzulegen. In den folgenden Abschnitten beschreiben wir kurz die wichtigsten Funktionen der üblichen Formen von IEEE-802.16-Luftschnittstellen; diese sind aber bei Weitem nicht vollständig und lassen viele Details weg. Weitere Informationen zu WiMAX und zum drahtlosen Breitband allgemein finden Sie bei Andrews et al. (2007).

4.5.1 Vergleich von IEEE 802.16 mit IEEE 802.11 und 3G

An dieser Stelle werden Sie vielleicht einwenden: Warum muss ein neuer Standard definiert werden? Warum verwendet man nicht einfach IEEE 802.11 oder 3G? Tatsächlich kombiniert WiMAX Aspekte von beiden, sowohl von IEEE 802.11 als auch von 3G, was es eher zu einer 4G-Technologie macht.

Wie IEEE 802.11 geht es bei WiMAX um die drahtlose Verbindung von Geräten mit dem Internet zu Geschwindigkeiten im Bereich MBit/s anstelle der Verwendung von Kabel oder DSL. Die Geräte sind möglicherweise mobil oder zumindest tragbar. WiMAX hat gar nicht erst versucht, niedrige Dateraten auf Seite der sprachähnlichen zellulären Netzen hinzuzufügen – IEEE 802.16 war entworfen worden, um IP-Pakete über die Luft zu transportieren und mit einem IP-basierten verdrahteten Netz mit einem Minimum an Aufwand zu verbinden. Die Pakete konnten Peer-to-Peer-Datenverkehr, VoIP-Anrufe oder Media-Streaming übertragen, um eine große Bandbreite von Anwendungen zu unterstützen. Ebenso wie IEEE 802.11 basiert WiMAX einerseits auf OFDM-Technologie, um gute Performanz trotz drahtloser Signalverschlechterung wie Mehrwegeempfang zuzusichern, und andererseits auf MIMO-Technologie, um ein hohes Durchsatzniveau zu erreichen.

WiMAX gleicht jedoch in mehreren Schlüsselaspekten eher 3G (und somit nicht IEEE 802.11). Das technische Hauptproblem ist, hohe Kapazität durch effiziente Spektrumsnutzung zu erreichen, sodass eine große Anzahl von Teilnehmern in einem Abdeckungsbereich hohen Durchsatz bekommen können. Die typischen Entfernung sind mindestens zehnmal größer als bei einem IEEE-802.11-Netz. Folglich sind WiMAX-Basisstationen leistungsstärker als IEEE-802.11-Zugangspunkte. Um auch schwächere Signale über größere Entfernung zu übermitteln, benutzt die Basisstation mehr Strom und bessere Antennen und investiert mehr Aufwand in die Fehlerbehandlung. Um den Durchsatz zu maximieren, werden Übertragungen von der Basisstation für jeden einzelnen Teilnehmer sorgfältig zeitlich geplant; die Spektrumsnutzung wird

nicht dem Zufall überlassen, indem mit CSMA/CA verwendet und dabei möglicherweise Kapazität mit Kollisionen verschwendet wird.

Für WiMAX ist lizenziertes Spektrum vorgesehen, in den USA typischerweise um 2,5 GHz. Das gesamte System ist wesentlich besser optimiert als IEEE 802.11. In Anbetracht der großen Geldmenge, die beim lizenzierten Spektrum im Spiel ist, lohnt sich diese Komplexität. Anders als IEEE 802.11 ist das Ergebnis ein verwalteter und zuverlässiger Dienst mit guter Dienstgüteunterstützung.

Mit all diesen Merkmalen kommt IEEE 802.16 den zellulären 4G-Netzen am nächsten, die nun unter dem Namen **LTE** (*Long Term Evolution*) standardisiert werden. Während zelluläre 3G-Netze auf CDMA basieren und sowohl Sprache als auch Daten unterstützen, werden zelluläre 4G-Netze auf OFDM mit MIMO basieren und auf Daten abziehen, wobei Sprache nur eine Anwendung von vielen ist. Es sieht so aus, als befänden sich WiMAX und 4G auf einem Kollisionskurs bezüglich Technologie und Anwendungen. Vielleicht überrascht diese Annäherung nicht angesichts der Tatsache, dass das Internet die Killeranwendung ist und OFDM und MIMO die bekanntesten Technologien für die effiziente Spektrumsnutzung sind.

4.5.2 IEEE-802.16-Architektur und -Protokollstapel

Die IEEE-802.16-Architektur ist in ►Abbildung 4.30 dargestellt. Basisstationen sind direkt an das Backbone-Netz des Providers angeschlossen, welches wiederum mit dem Internet verbunden ist. Sie kommunizieren mit Stationen über die drahtlose Luftschnittstelle. Es gibt zwei Arten von Stationen: Teilnehmerstationen verbleiben an einem festen Standort, zum Beispiel der Breitbandinternetzugang für Privathaushalte. Mobile Stationen können Dienste empfangen, während sie sich fortbewegen, zum Beispiel ein mit WiMAX ausgestattetes Auto.

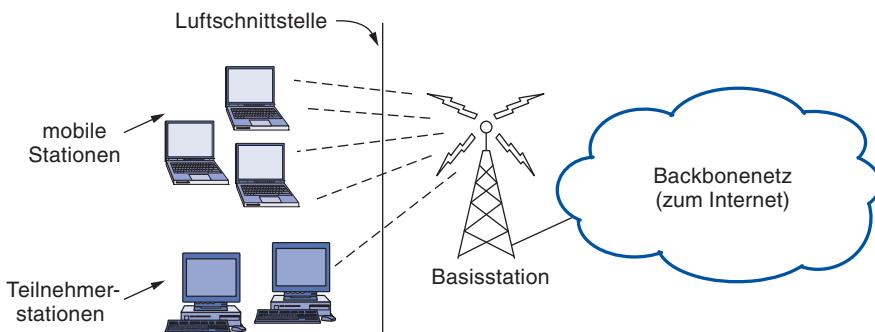


Abbildung 4.30: Die IEEE-802.16-Architektur.

Der IEEE-802.16-Protokollstapel, der über der Luftschnittstelle benutzt wird, ist in ►Abbildung 4.31 dargestellt. Die allgemeine Struktur entspricht der anderer IEEE-802-Netze, hat jedoch mehr Teilschichten. Die unterste Schicht beschäftigt sich mit der Übertragung – hier haben wir nur die beliebten Angebote von IEEE 802.16, festes und mobiles WiMAX, abgebildet. Es gibt für jedes Angebot eine andere Bitübertragungs-

schicht. Beide Schichten operieren im lizenzierten Spektrum unterhalb von 11 GHz und benutzen OFDM, jedoch auf unterschiedliche Weise.

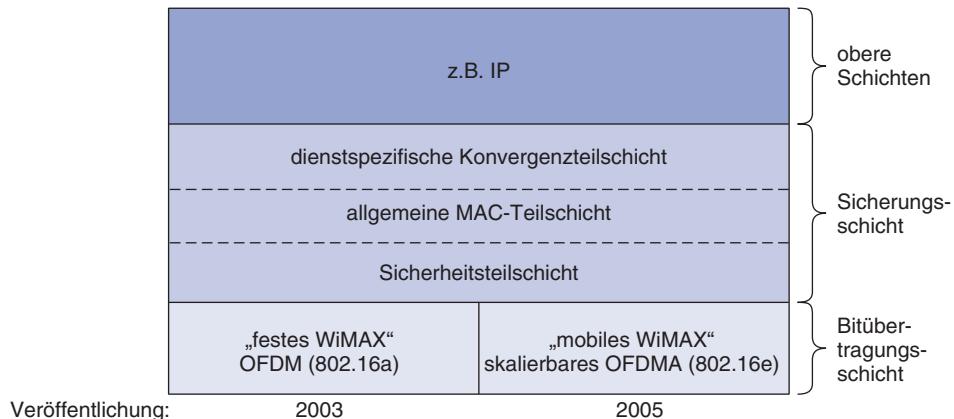


Abbildung 4.31: Der IEEE-802.16-Protokollstapel.

Die nächste Schicht ist der allgemeine Teil der MAC-Teilschicht. Hier befinden sich die Hauptprotokolle wie die Kanalverwaltung. Man geht hier davon aus, dass die Basisstation das System vollständig steuert. Es kann die Downlink-Kanäle (also Basis zum Teilnehmer) sehr effizient verwalten und spielt auch eine wichtige Rolle bei der Verwaltung des Uplink-Kanals (also Teilnehmer zur Basis). Eine ungewöhnliche Funktion dieser MAC-Teilschicht ist, dass sie vollständig verbindungsorientiert ist, um die Dienstgüte für Telefonie und Multimedia sicherzustellen.

Die dienstbezogene Konvergenzteilschicht übernimmt hier die Funktion der Teilschicht der logischen Verbindungssteuerung in den anderen IEEE-802-Protokollen. Die Funktion besteht darin, eine Schnittstelle zur Vermittlungsschicht anzubieten. Verschiedene Konvergenzschichten sind so definiert, dass sie nahtlos in unterschiedliche obere Schichten integriert werden können. Deshalb ist die Wahl von IP hier wichtig, obwohl der Standard auch Protokolle wie Ethernet und ATM zulässt. Da IP verbindungslos, die IEEE-802.16-MAC-Teilschicht aber verbindungsorientiert ist, muss diese Schicht Adressen und Verbindungen aufeinander abbilden.

4.5.3 IEEE-802.16-Bitübertragungsschicht

Die meisten WiMAX-Installationen benutzen lizenziertes Spektrum entweder um 3,5 GHz oder um 2,5 GHz. Ebenso wie bei 3G ist das Auffinden verfügbaren Spektrums das Hauptproblem. Um hier Abhilfe zu schaffen, wurde der IEEE-802.16-Standard auf Flexibilität hin entworfen. Er erlaubt Betrieb von 2 GHz bis zu 11 GHz. Kanäle unterschiedlicher Größe werden unterstützt, zum Beispiel 3,5 MHz für festes WiMAX und von 1,25 bis 20 MHz mit mobilem WiMAX.

Übertragungen werden über diese Kanäle mit OFDM gesendet – die Technik, die wir in Abschnitt 2.5.3 beschrieben haben. Verglichen mit IEEE 802.11 ist der IEEE-802.16-Entwurf optimiert, um das lizenzierte Spektrum und Langstreckenübertragungen voll

auszuschöpfen. Der Kanal ist in mehrere Unterträger mit einer längeren Symboldauer aufgeteilt, um größere drahtlose Signalverschlechterungen zu tolerieren: WiMAX-Parameter sind rund 20-mal größer als vergleichbare IEEE-802.11-Parameter. Beispielsweise gibt es beim mobilen WiMAX 512 Unterträger für einen 5-GHz-Kanal und die Zeit zum Senden eines Symbols auf jedem Unterträger beträgt grob 100µs.

Symbole auf den Unterträgern werden mit QPSK, QAM-16 oder QAM-64 gesendet, den Modulationsschemata, die wir in Abschnitt 2.5.3 beschrieben haben. Wenn die mobile Station oder die Teilnehmerstation nahe an der Basisstation ist und das empfangene Signal ein höheres Signal-Rausch-Verhältnis (SNR, *Signal-to-Noise-Ratio*) hat, dann kann QAM-64 benutzt werden, um 6 Bits pro Symbol zu senden. Um entfernte Stationen mit einem niedrigen SNR zu erreichen, kann QPSK benutzt werden, um 2 Bits pro Symbol zu liefern. Die Daten werden zuerst zur Fehlerkorrektur mit dem Faltungscode (oder besseren *Verfahren*) codiert, die wir in Abschnitt 3.2.1 besprochen haben. Diese Codierung ist auf verrauschten Kanälen üblich, um einige Bitfehler zu tolerieren, ohne dass Neuübertragungen erforderlich sind. Inzwischen sollten die Modulations- und Codierungsmethoden eigentlich vertraut klingen, da sie für viele der von uns untersuchten Netze verwendet werden, einschließlich IEEE-802.11-Kabel und DSL. Das Ergebnis ist, dass eine Basisstation bis zu 12,6 Mbit/s Downlink-Verkehr und 6,2 Mbit/s Uplink-Verkehr pro 5-MHz-Kanal und Antennenpaar unterstützen kann.

Eine Sache, die die Entwickler von IEEE 802.16 nicht mochten, war ein bestimmter Aspekt an der Art, wie GSM und DAMPS arbeiten. Beide Systeme verwenden gleich breite Frequenzbänder für den Upstream- und den Downstream-Verkehr. Das heißt, es wird implizit angenommen, dass es genauso viel Upstream- wie Downstream-Verkehr gibt. Bei Sprachdaten ist der Verkehr größtenteils symmetrisch, aber beim Internetzugang (und sicherlich beim Surfen im Web) gibt es sehr oft viel mehr Downstream- als Upstream-Datenverkehr. Das Verhältnis beträgt häufig 2:1, 3:1 oder mehr.

Daher wählten die Entwickler ein flexibles Schema, um den Kanal zwischen den Stationen aufzuteilen, genannt **OFDMA** (*Orthogonal Frequency Division Multiple Access*). Bei OFDMA können verschiedene Mengen von Unterträgern unterschiedlichen Stationen zugeordnet werden, sodass mehr als eine Station gleichzeitig senden oder empfangen kann. Auf IEEE 802.11 übertragen hieße dies, dass alle Unterträger von einer Station benutzt werden, um zu jedem beliebigen Zeitpunkt zu senden. Die zusätzliche Flexibilität in der Art der Bandbreitenzuweisung kann die Performanz verbessern, weil ein gegebener Unterträger bei einem Empfänger aufgrund der Auswirkungen des Mehrwegeempfangs zwar abgeschwächt ankommen könnte, aber bei einem anderen Empfänger klar sein kann. Unterträger können den Stationen zugewiesen werden, die sie am besten nutzen können.

Neben dem asymmetrischen Datenverkehr können Stationen in der Regel zwischen Senden und Empfangen wechseln. Diese Methode heißt **Zeitduplexverfahren (TDD, Time Division Duplex)**. Die alternative Methode, bei welcher eine Station gleichzeitig sendet und empfängt (auf verschiedenen Unterträgerfrequenzen) heißt **Frequenzduplexverfahren (FDD, Frequency Division Duplex)**. WiMAX lässt beide Methoden zu, aber TDD wird bevorzugt, weil es einfacher zu implementieren und flexibler ist.

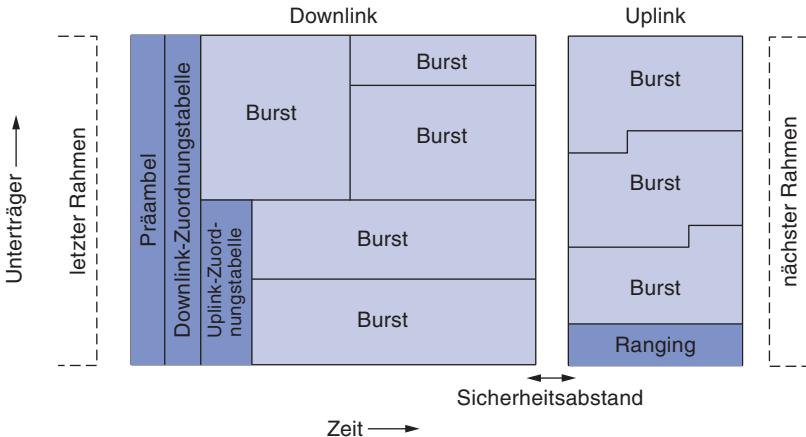


Abbildung 4.32: Rahmenstruktur für OFDMA mit Zeitduplexing.

► Abbildung 4.32 zeigt ein Beispiel der Rahmenstruktur, die im Laufe der Zeit wiederholt wird. Der Rahmen beginnt mit einer Präambel, um alle Stationen zu synchronisieren, gefolgt von den Downlink-Übertragungen der Basisstation. Zuerst sendet die Basisstation die Zuordnungstabelle, die allen Stationen mitteilt, wie die Downlink- und Uplink-Unterträger über dem Rahmen zugewiesen werden. Die Basisstation steuert die Zuordnungen und kann so den Stationen verschiedene Bandbreiten von Rahmen zu Rahmen zuteilen, abhängig von den Anforderungen jeder Station.

Als Nächstes sendet die Basisstation Daten-Bursts an verschiedene Teilnehmer und mobile Stationen auf den Unterträgern zu den Zeiten, die in der Zuordnungstabelle angegeben sind. Die Downlink-Übertragung endet mit einem Sicherheitsabstand, den Stationen zum Wechseln vom Modus „Empfangen“ zum Modus „Übertragen“ nutzen können. Schließlich senden der Teilnehmer und die mobilen Stationen ihre Daten-Bursts zur Basisstation aus den Uplink-Positionen heraus, die für sie in der Zuordnungstabelle reserviert waren. Eines dieser Uplink-Bursts ist für das sogenannte **Ranging** reserviert. Dies ist der Prozess, wenn sich eine neue Station zeitlich abstimmt und anfängliche Bandbreite anfordert, um sich mit der Basisstation zu verbinden. Da zu diesem Zeitpunkt keine Verbindung aufgebaut ist, beginnen neue Stationen einfach mit der Übertragung und hoffen, dass es keine Kollision gibt.

4.5.4 IEEE-802.16-MAC-Teilschichtprotokoll

Die Sicherungsschicht ist, wie in Abbildung 4.31 dargestellt, in drei Teilschichten unterteilt. Da wir uns mit der Verschlüsselung erst in *Kapitel 8* beschäftigen werden, ist die Arbeitsweise der Sicherheitsteilschichten hier schwer zu erklären. Es reicht aber einstweilen aus zu wissen, dass hier die Verschlüsselung eingesetzt wird, um die übertragenen Daten geheim zu halten. Es werden nur die Rahmennutzdaten verschlüsselt, nicht aber die Header. Dies besagt, dass ein Lauscher zwar erkennen kann, wer mit wem spricht, nicht aber, worüber beide sprechen.

Wenn Sie sich bereits etwas mit Verschlüsselungsverfahren auskennen, so finden Sie im Folgenden eine kurze Erläuterung der Sicherheitsteilschicht. Wenn Sie sich mit Verschlüsselung nicht auskennen, werden Sie mit dem folgenden Absatz höchstwahrscheinlich nicht viel anfangen können (aber Sie können ihn nach der Lektüre von *Kapitel 8* noch einmal durchlesen).

Wenn sich ein Teilnehmer bei der Basisstation anmeldet, wird eine gegenseitige Authentifizierung unter Verwendung der RSA-Verschlüsselung mit öffentlichen Schlüsseln mit X.509-Zertifikaten durchgeführt. Die Nutzdaten werden mit einem symmetrischen Schlüsselsystem verschlüsselt, entweder AES (Rijndael) oder DES mit der Verkettung von Verschlüsselungsblocks. Die Integritätsprüfung verwendet SHA-1. Nun, das war gar nicht so schlimm, oder?

Betrachten wir jetzt den allgemeinen Teil der MAC-Teilschicht. Die MAC-Teilschicht ist verbindungsorientiert und verwendet Punkt-zu-Multipunkt-Verbindungen, was bedeutet, dass eine Basisstation mit mehreren Teilnehmerstationen kommuniziert. Vieles aus diesem Entwurf ist von Kabelmodems übernommen, bei denen eine Kabelkopfstelle die Übertragung von mehreren Kabelmodems beim Kunden steuert.

Der Downlink-Kanal ist ganz einfach strukturiert. Die Basisstation steuert die Bursts der Bitübertragungsschicht, die benutzt werden, um Informationen zu den unterschiedlichen Teilnehmern zu senden. Die MAC-Teilschicht packt einfach ihre Rahmen in diese Struktur. Um den Overhead zu verhindern, gibt es mehrere Möglichkeiten. Zum Beispiel können MAC-Rahmen einzeln gesendet werden oder aufeinanderfolgend zu einer Gruppe verpackt werden.

Der Uplink-Kanal ist komplizierter, da hier konkurrierende Teilnehmer vorhanden sind, die darauf zugreifen möchten. Die Zuordnung ist eng mit dem Dienstgüthema verbunden. Es sind die folgenden vier Dienstklassen definiert:

- 1.** Dienst mit konstanten Bitübertragungsraten
- 2.** Echtzeitdienst mit variablen Bitübertragungsraten
- 3.** Nichtechtzeitdienst mit variablen Bitübertragungsraten
- 4.** Best-Effort-Dienst

Der gesamte Dienst in IEEE 802.16 ist verbindungsorientiert. Jede Verbindung erhält eine dieser Dienstklassen. Dies wird beim Aufbau der Verbindung festgelegt. Dieses Design unterscheidet sich von dem von IEEE 802.11 oder Ethernet, die in der MAC-Teilschicht verbindungslos sind.

Der Dienst mit konstanten Bitübertragungsraten dient vorzugsweise zur Übertragung nicht komprimierter Sprache. Dieser Dienst muss eine vorher festgelegte Datenmenge in vorab festgelegten Zeitintervallen senden. Hierzu werden jeder Verbindung dieses Typs spezielle Bursts zugewiesen. Wurde die Bandbreite einmal zugewiesen, dann sind die Bursts automatisch verfügbar, es muss also keines extra angefordert werden.

Der Echtzeitdienst mit variablen Bitübertragungsraten ist für komprimierte Multimediadaten und andere Echtzeitanwendungen gedacht, bei denen die erforderliche Bandbreite zu jedem Zeitpunkt variieren kann. Dies wird durch das Polling der Basisstation in festen Intervallen erreicht, in denen gefragt wird, wie viel Bandbreite dieses Mal benötigt wird.

Der Nichtechtzeitdienst mit variablen Bitübertragungsraten ist für datenintensive Übertragungen, die nicht in Echtzeit ablaufen müssen, wie die Übertragung großer Dateien. Bei diesem Dienst fragt die Basisstation die Teilnehmer häufig ab, jedoch nicht in streng festgelegten Zeitabständen. Verbindungen können zur Anforderung von Bandbreite außerdem den Best-Effort-Dienst benutzen, der als Nächstes beschrieben wird.

Der Best-Effort-Dienst steht für alles andere zur Verfügung. Es wird kein Polling gesendet und der Teilnehmer muss sich mit anderen Teilnehmern dieses Diensttypes um die Bandbreite bewerben. Die Bandbreitenanforderungen werden in Bursts gesendet, die in einer Uplink-Zuordnung als für den Wettbewerb verfügbar markiert sind. Ist eine Anforderung erfolgreich, so wird der Erfolg in der nächsten Downlink-Zuordnung vermerkt. Andernfalls muss der erfolglose Teilnehmer es später noch einmal versuchen. Um Kollisionen zu vermeiden, wird der binäre exponentielle Backoff-Algorithmus aus dem Ethernet verwendet.

4.5.5 IEEE-802.16-Rahmenstruktur

Alle MAC-Rahmen beginnen mit einem allgemeinen Header. Auf den Header folgen optionale Nutzdaten und eine optionale Prüfsumme (CRC), wie in Abbildung 4.33 dargestellt. Das Nutzdatenfeld wird beispielsweise in Steuerrahmen, die Zeitscheiben im Kanal anfordern, nicht benötigt. Die Prüfsumme ist (überraschenderweise) ebenfalls optional – dank der Fehlerkorrektur in der Bitübertragungsschicht und der Tatsache, dass nie versucht wird, die Echtzeitrahmen erneut zu überfragen. Wenn keine erneuten Übertragungen versucht werden, warum sollte man sich dann mit einer Prüfsumme belasten? Falls es jedoch eine Prüfsumme gibt, dann ist es das Standard-IEEE-802-CRC. Bestätigungen und Neuübertragungen werden dann aus Gründen der Zuverlässigkeit durchgeführt.

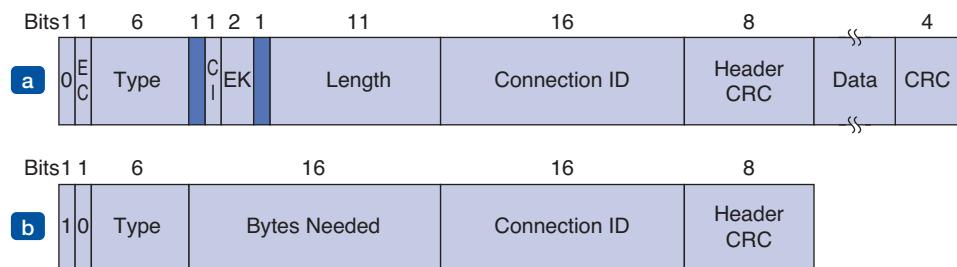


Abbildung 4.33: (a) Allgemeiner Rahmen. (b) Rahmen, der Bandbreite anfordert.

Die Header-Felder aus ►Abbildung 4.33a sehen kurz zusammengefasst wie folgt aus: Das *EC*-Bit gibt an, ob die Nutzdaten verschlüsselt sind. Das Feld *Type* gibt den Rahmentyp an, wobei hier meist angegeben wird, ob Fragmentierung verwendet wird. Das Feld *CI* gibt das Vorhandensein oder Fehlen einer finalen Prüfsumme am Ende des Rahmens an. Das Feld *EK* gibt an, welcher der Schlüssel verwendet wird (falls einer verwendet wird). Das Feld *Length* gibt die vollständige Länge des Rahmens einschließlich des Headers an. Das Feld *Connection ID* gibt an, zu welcher Verbindung dieser Rahmen gehört. Zuletzt enthält das Feld *Header CRC* eine Prüfsumme nur für den Header. Hierbei wird das Polynom x^8+x^2+x+1 verwendet.

Das IEEE-802.16-Protokoll enthält viele Rahmenarten. Ein Beispiel für eine andere Rahmenart, die zur Anforderung von Bandbreite benutzt wird, ist in ►Abbildung 4.33b gezeigt. Dieser Rahmen beginnt mit einem 1-Bit anstatt eines 0-Bits und ähnelt ansonsten dem allgemeinen Header mit der Ausnahme, dass das zweite und das dritte Byte eine 16-Bit-Zahl bilden (*Bytes Needed*), die angibt, wie viel Bandbreite zur Übertragung der angegebenen Byte-Anzahl erforderlich ist. Rahmen zur Anforderung von Bandbreiten enthalten keine Nutzdaten und keine Prüfsumme für den gesamten Rahmen.

Man könnte noch viel mehr über IEEE 802.16 erzählen, aber dies würde den Rahmen des Buches sprengen. Weitere Informationen finden Sie im veröffentlichten Standard IEEE 802.16-2009.

4.6 Bluetooth

1994 begann sich das norwegische Unternehmen L.M. Ericsson für die Verbindung von Mobiltelefonen mit anderen Geräten wie Laptops ohne Kabel zu interessieren. Zusammen mit vier anderen Unternehmen bildeten sie 1998 eine SIG (*Special Interest Group*, also ein Konsortium), um einen drahtlosen Standard zur Verbindung von Rechnern und Kommunikationsgeräten unter Verwendung von kostengünstigen drahtlosen Funkgeräten mit geringem Strombedarf zu erstellen. Das Projekt wurde **Bluetooth** genannt, nach Harald Blaatand (Bluetooth) II (940–981), einem Wikingerkönig, der Dänemark und Norwegen auch ganz ohne Kabel vereinte (also eroberte).

Bluetooth 1.0 wurde im Juli 1999 herausgegeben, seitdem hat die SIG niemals zurückgeblickt. Alle möglichen Gerätarten der Unterhaltungselektronik benutzen heute Bluetooth, von Handys und Laptops bis zu Headsets, Druckern, Tastaturen, Mäuse, Spielkonsolen, Uhren, Musikplayer, Navigationgeräte und mehr. Die Bluetooth-Protokolle ermöglichen es diesen Geräten, einander zu finden und sich zu verbinden – ein Vorgang, den man **Pairing** nennt – und Daten sicher zu übertragen.

Die Protokolle haben sich während des letzten Jahrzehnts ebenfalls weiterentwickelt. Nachdem die ersten Protokolle stabil waren, wurden im Jahr 2004 höhere Datenraten zu Bluetooth 2.0 hinzugefügt. Seit dem 3.0-Release 2009 kann Bluetooth zum Pairing von Geräten in Kombination mit IEEE 802.11 für Datenübertragung mit höherem Durchsatz verwendet werden. Das 4.0-Release im Dezember 2009 spezifiziert Niedrigenergiebetrieb. Dies ist für all diejenigen praktisch, die keine Lust haben, regelmäßig

Batterien in ihren Geräten überall im Haus zu wechseln. Wir werden nun die Hauptaspekte von Bluetooth betrachten.

4.6.1 Architektur von Bluetooth

Beginnen wir unsere Untersuchung des Bluetooth-Systems mit einer kurzen Übersicht über die Inhalte und Ziele. Die Grundeinheit eines Bluetooth-Systems ist ein **Piconetz**, das aus einem Master-Knoten und bis zu sieben Slave-Knoten in einem maximalen Umkreis von 10 m besteht. Mehrere Piconetze können in einem (großen) Raum vorhanden sein und sogar über einen Bridge-Knoten verbunden sein, der Teil von mehreren Piconetzen wie in ►Abbildung 4.34 ist. Eine miteinander verbundene Gruppe von Piconetzen wird als **Scatternetz** bezeichnet.

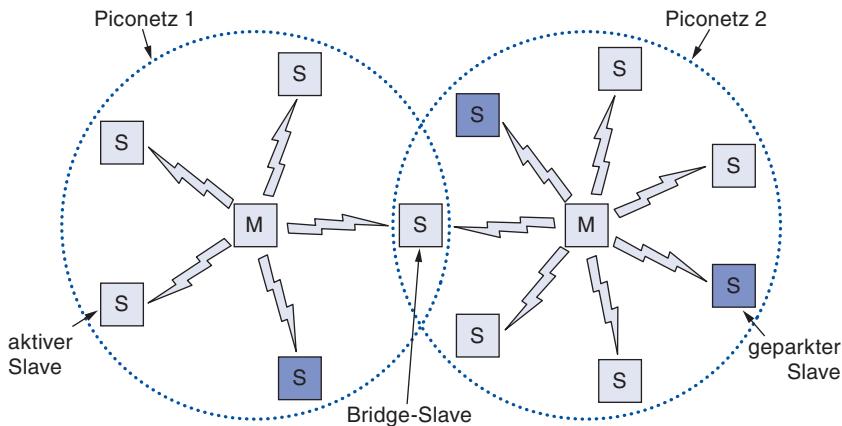


Abbildung 4.34: Zwei Piconetze können verbunden werden und ein Scatternetz bilden.

Zusätzlich zu den sieben aktiven Slave-Knoten können in einem Netz bis zu 255 geparkte Knoten vorhanden sein. Dies sind die Geräte, die der Master in einen Ruhezustand mit geringem Stromverbrauch gesetzt hat, um den Batterieverbrauch zu senken. In dem Parkzustand kann ein Gerät nur auf eine Aktivierung oder ein Beacon-Signal vom Master reagieren. Es gibt auch noch zwei dazwischenliegende Stromsparmodi, Hold (Halten) und Sniff (Abhören), die uns aber hier nicht weiter beschäftigen.

Der Grund für die Entwicklung eines Master-Slave-Designs ist, dass die Entwickler die Implementierung eines vollständigen Bluetooth-Chips für weniger als 5 US-Dollar im Auge hatten. Daher sind die Slaves ziemlich dumm und machen im Grunde nur das, was ihnen das Master-Gerät sagt. Im Kern ist ein Piconetz ein zentralisiertes TDM-System, wobei der Master den Takt steuert und entscheidet, wer in welcher Zeitscheibe übertragen darf. Die Kommunikation läuft immer zwischen Master und Slave ab. Eine direkte Kommunikation von Slave zu Slave ist nicht möglich.

4.6.2 Bluetooth-Anwendungen

Die meisten Netzprotokolle stellen nur Kanäle zwischen den kommunizierenden Objekten zur Verfügung und geben den Anwendungsentwicklern völlige Freiheit bei deren Nutzung. So gibt beispielsweise IEEE 802.11 nicht an, ob die Benutzer ihre Notebooks zum Lesen von E-Mail, zum Surfen im Web oder zu etwas anderem verwenden sollen. Im Unterschied dazu spezifiziert die Bluetooth SIG bestimmte Anwendungen, die unterstützt werden sollen, und stellt für jede einen anderen Protokollstapel bereit. Zum jetzigen Zeitpunkt gibt es 25 Anwendungen, die als **Profile** bezeichnet werden. Leider führt dies zu einer hohen Komplexität. Wir wollen diese Komplexität hier übergehen und stattdessen einen kurzen Blick auf die Profile werfen, um die Ziele der Bluetooth SIG besser zu verstehen.

Sechs der Profile sind für andere Nutzungen wie Audio und Video vorgesehen. Zum Beispiel ermöglicht das Interkommunikationsprofil (Intercom) die Verbindung zweier Telefone als Walkie-Talkies. Die Headset- und Freisprechprofile definieren beide die Sprachkommunikation zwischen einem Headset und der Basisstation, das zum Telefonieren beim Autofahren benutzt werden kann. Andere Profile sind für das Streamen von Audio und Video und Stereoqualität, beispielsweise von einem tragbaren Musikplayer auf Kopfhörer oder von einer digitalen Kamera auf ein Fernsehgerät.

Das HID-Profil (*Human Interface Device*) dient dem Verbinden von Tastaturen und Mäusen mit Computern. Andere Profile ermöglichen es einem Mobiltelefon oder anderen Rechnern, Bilder von einer Kamera zu empfangen oder Bilder an einen Drucker zu senden. Vielleicht von größerem Interesse ist ein Profil, um ein Handy als Fernbedienung für einen (Bluetooth-fähigen) Fernseher zu benutzen.

Noch andere Profile ermöglichen Netzbetrieb. Das PAN-Profil erlaubt es Bluetooth-Geräten, ein Ad-hoc-Netz zu bilden oder auf andere Netze (wie ein IEEE-802.11-LAN) über einen Zugangspunkt entfernt zuzugreifen. Das DUN-Profil (*Dial-Up Networking*) war tatsächlich die ursprüngliche Motivation für das gesamte Projekt. Es erlaubt einem Notebook, sich ohne Kabel mit einem Handy zu verbinden, das ein eingebautes Modem enthält.

Es wurden ebenfalls Profile für Informationsaustausch auf höheren Schichten definiert. Das Synchronisationsprofil ist dazu gedacht, beim Verlassen des Hauses Daten in ein Mobiltelefon zu laden und bei der Rückkehr Daten auszulesen.

Wir werden die übrigen Profile hier überspringen, wollen aber noch erwähnen, dass einige Profile als Bausteine dienen, aus denen die obigen Profile zusammengesetzt sind. Die generischen Zugriffsprofile, aus denen all die anderen Profile aufgebaut sind, stellen eine Möglichkeit zur Verfügung, sichere Verbindungen (Kanäle) zwischen Master und Slaves einzurichten und aufrechtzuerhalten. Die anderen generischen Profile definieren die Basis von Objektaustausch sowie Audio- und Videoübertragung. Hilfsprofile werden weitverbreitet für Funktionen wie die Emulation einer seriellen Leitung eingesetzt, was besonders nützlich für viele veraltete Anwendungen ist.

War es wirklich notwendig, alle diese Anwendungen im Detail darzulegen und für jeden einen eigenen Protokollstapel zu definieren? Wahrscheinlich nicht, aber es gab mehrere Arbeitsgruppen, die sich mit unterschiedlichen Teilen des Standards beschäftigten, wobei sich jede auf ihr spezielles Problem konzentrierte und ein eigenes Profil erarbeitete. Stellen Sie sich dies als das angewandte Gesetz von Conway vor. (In der Ausgabe vom April 1968 des *Datamation*-Magazins hat Melvin Conway beobachtet, dass wenn man n Personen den Auftrag erteilt, einen Compiler zu schreiben, man einen Compiler mit n Arbeitsgängen erhält. Allgemeiner ausgedrückt: Die Software spiegelt immer die Struktur der Gruppe wider, die diese erstellt hat.) Man wäre auch mit zwei Protokollstapeln anstelle der 25 ausgekommen, einen für die Dateiübertragung und einen für die unterbrechungsfreie Echtzeitübertragung.

4.6.3 Bluetooth-Protokollstapel

Der Bluetooth-Standard hat viele Protokolle, die lose in die in ►Abbildung 4.35 gezeigten Schichten eingeteilt werden. Die erste Beobachtung ist, dass die Strukturierung der Schichten weder dem OSI-Modell noch dem TCP/IP-Modell oder irgendinem anderen Modell folgt.

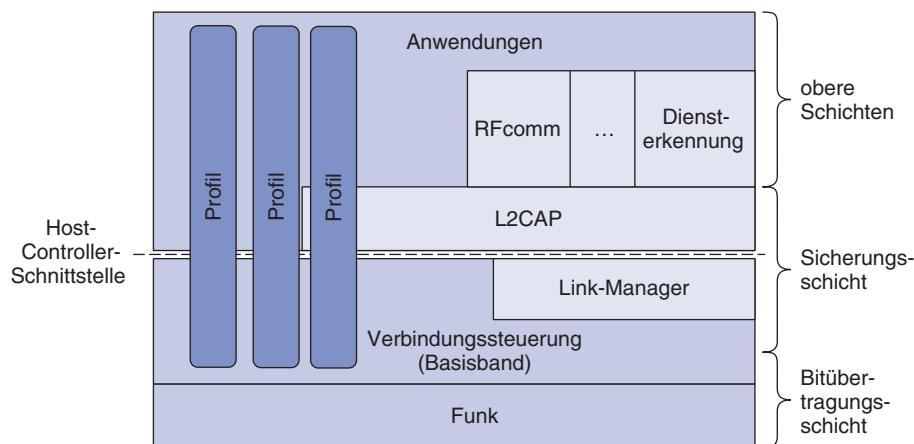


Abbildung 4.35: Die Bluetooth-Protokollarchitektur.

Die untere Schicht ist die physikalische Funkschicht, die im Großen und Ganzen der Bitübertragungsschicht des OSI- und des IEEE-802-Modells entspricht. Sie behandelt die Funkübertragung und die Modulation. Viele Entscheidungen beruhen hier auf der Forderung, das System kostengünstiger zu machen, um es für einen Massenmarkt produzieren zu können.

Die Schicht zur Verbindungssteuerung (*link control layer*) oder Basisbandschicht entspricht in etwa der MAC-Teilschicht, enthält aber auch Elemente der Bitübertragungsschicht. Hier wird behandelt, wie der Master die Zeitscheiben steuert und wie diese Zeitscheiben in Rahmen zusammengefasst werden.

Als Nächstes kommen zwei Protokolle, die das Protokoll zur Verbindungssteuerung benutzen. Der Link-Manager verwaltet die Errichtung logischer Kanäle zwischen den Geräten, einschließlich Energieverwaltung, Pairing und Verschlüsselung sowie Dienstgüte. Er befindet sich unterhalb der Host-Controller-Schnittstelle. Diese Schnittstelle ist eine Erleichterung für die Implementierung: In der Regel werden die Protokolle unterhalb dieser Linie auf einem Bluetooth-Chip implementiert, die Protokolle oberhalb der Linie werden in dem Bluetooth-Gerät implementiert, in dem der Chip untergebracht ist.

Das Verbindungsprotokoll oberhalb der Linie ist **L2CAP** (*Logical Link Control Adaptation Protocol*, Protokoll zur Anpassung an die logische Verbindungssteuerung). Es erlaubt Rahmenbildung von Nachrichten mit variabler Länge und stellt bei Bedarf Zuverlässigkeit zur Verfügung. Viele Protokolle, wie die zwei dargestellten Hilfsprotokolle, verwenden L2CAP. Das Diensterkennungsprotokoll (*service discovery protocol*) dient dem Auffinden von Diensten im Netz. Das RFcomm-Protokoll (*Radio Frequency Communication*) emuliert den seriellen Standardport an PCs zum Anschluss von Tastatur, Maus, Modem und anderen Geräten.

Auf der obersten Schicht befinden sich die Anwendungen. Die Profile werden durch vertikale Rechtecke dargestellt, weil diese jeweils eine Scheibe des Protokollstapels für einen bestimmten Zweck definieren. Spezielle Profile wie das Headsetprofil enthalten in der Regel nur die Protokolle, die von dieser Anwendung benötigt werden, und keine weiteren. Beispielsweise kann L2CAP zu einem Profil gehören, wenn Pakete zu versenden sind. Falls es dagegen nur einen stetigen Strom von Audiodaten gibt, kann L2CAP ausgelassen werden.

In den folgenden Abschnitten untersuchen wir die Bluetooth-Funkschicht und verschiedene Verbindungsprotokolle, da diese in etwa der Bitübertragungsschicht und den MAC-Teilschichten in den bisher besprochenen Protokollstapels entsprechen.

4.6.4 Bluetooth-Funkschicht

Die Funkschicht überträgt die Daten vom Master zum Slave oder umgekehrt. Es ist ein Niedrigenergiesystem mit einer Reichweite von 10 m, das im gleichen 2,4-GHz-ISM-Band wie IEEE 802.11 läuft. Das Band ist in 79 Kanäle von je 1 MHz aufgeteilt. Um neben anderen Netzen, die das ISM-Band benutzen, zu bestehen, wird eine Frequenzsprungtechnik mit Spektrumsspreizung verwendet. Es können bis zu 1 600 Umschaltungen/Sekunde über Zeitscheiben mit einer Verweilzeit von 625 µs vorkommen. In einem Piconetz springen alle Knoten gleichzeitig und folgen der Zeitscheibenvorgabe und der pseudozufälligen Sprungfolge, die der Master vorgibt.

Unglücklicherweise hat sich herausgestellt, dass sich frühe Versionen von Bluetooth und IEEE 802.11 gegenseitig so sehr beeinträchtigen, dass sie Übertragungen des jeweils anderen zerstören. Einige Unternehmen haben darauf reagiert, indem sie Bluetooth ganz verbannt haben, doch schließlich ist noch eine technische Lösung gefunden worden. Diese Lösung sieht für Bluetooth vor, seine Sprungfolge anzupassen, um Kanäle, auf denen es andere RF-Signale gibt, auszulassen. Dieses Vorgehen verringert

die schädigenden Interferenzen. Es wird **adaptives Frequenzsprungverfahren** (*adaptive frequency hopping*) genannt.

Drei Modulationsformen werden benutzt, um Bits über einen Kanal zu senden. Das Grundschema ist, Frequenzumtastung zu verwenden, um ein 1-Bit-Symbol jede Mikrosekunde zu senden, wodurch eine Bruttodatenrate von 1 Mbit/s erreicht wird. Verbesserte Raten wurden mit der 2.0-Version von Bluetooth eingeführt. Diese Raten benutzen Phasenumtastung, um 2 oder 3 Bits pro Symbol zu senden, was zu Bruttodatenraten von 2 oder 3 Mbit/s führt. Die verbesserten Raten werden nur in den Datenanteilen der Rahmen benutzt.

4.6.5 Bluetooth-Verbindungsschicht

Die Schicht zur Verbindungssteuerung (oder Basisbandschicht) von Bluetooth entspricht in etwa der MAC-Teilschicht. Sie verwandelt den reinen Bitstrom in Rahmen und definiert einige Schlüsselformate. In der einfachsten Form definiert der Master im Piconetz eine Reihe von Zeitscheiben mit 625 µs. Die Übertragungen des Masters beginnen in den geraden Zeitscheiben und die der Slaves in den ungeraden. Dieses Schema ist das normale Zeitmultiplexverfahren, bei dem der Master die eine Hälfte der Zeitscheiben und die Slaves die andere Hälfte erhalten. Rahmen können 1, 3 oder 5 Zeitscheiben lang sein. Jeder Rahmen hat einen Overhead von 126 Bit für einen Zugangscode und Header, plus eine Übergangsdauer von 250–260 µs pro Sprung, damit die kostengünstigen Funkleitungen stabil werden. Die Nutzdaten des Rahmens können zur Wahrung der Vertraulichkeit mit einem Schlüssel geschützt werden, der gewählt wird, wenn Master und Slave sich verbinden. Sprünge finden nur zwischen zwei Rahmen statt, nicht innerhalb eines Rahmens. Daher ist ein Rahmen von fünf Zeitscheiben viel effizienter als ein Rahmen mit einer Zeitscheibe, da bei konstantem Overhead mehr Daten gesendet werden können.

Das Protokoll zur Verbindungsverwaltung richtet logische Kanäle, sogenannte **Links** (Verbindung), zur Übertragung von Rahmen zwischen dem Master- und einem Slave-Gerät ein, die sich gegenseitig entdeckt haben. Es folgt eine Pairing-Prozedur um sicherzustellen, dass die beiden Geräte die Erlaubnis zur Kommunikation haben, bevor der Link benutzt wird. Bei der alten Pairing-Methode müssen beide Geräte mit derselben vierstelligen PIN (*Personal Identification Number*) konfiguriert werden. Die übereinstimmende PIN sorgt dafür, dass jedes Gerät weiß, ob es mit dem richtigen entfernten Gerät verbunden ist. Da fiktive Benutzer und Geräte gerne PINs wie „0000“ und „1234“ als Standardeinstellung wählen, bietet diese Methode in der Praxis sehr wenig Sicherheit.

Die neue **sichere einfache Pairing**-Methode ermöglicht es den Benutzern zum einen zu bestätigen, dass beide Geräte denselben Hauptschlüssel aufweisen, oder zum anderen den Hauptschlüssel auf dem einen Gerät zu beobachten und ihn in das zweite Gerät einzugeben. Diese Methode ist sicherer, weil Benutzer keine PIN auswählen oder einstellen müssen. Sie müssen lediglich einen längeren, vom Gerät erzeugten Hauptschlüssel bestätigen. Natürlich kann diese Methode bei einigen Geräten mit begrenzter

Ein-/Ausgabemöglichkeit nicht angewandt werden, wie beispielsweise Freisprecheinrichtungen.

Nachdem das Pairing abgeschlossen ist, richtet das Protokoll zur Verbindungsverwaltung die Links ein. Es gibt zwei grundsätzliche Arten von Links, um Benutzerdaten zu übertragen. Die erste ist der **SCO-Link** (*Synchronous Connection Oriented*) für Echtzeitdaten wie Telefonverbindungen. Bei diesem Linktyp ist jeder Richtung eine feste Zeitscheibe zugeordnet. Ein Slave kann bis zu drei SCO-Links zum Master besitzen. Ein SCO-Link kann einen PCM-Audiokanal mit 64 000 Bit/s übertragen. Aufgrund der zeikritischen Natur der SCO-Links werden Rahmen, die darüber versendet werden, nie erneut übertragen. Stattdessen kann mit der Vorwärtsfehlerkorrektur die Zuverlässigkeit erhöht werden.

Die andere Möglichkeit ist der **ACL-Link** (*Asynchronous ConnectionLess*). Dieser Linktyp wird für paketvermittelte Daten verwendet, die in unregelmäßigen Abständen verfügbar sind. ACL-Datenverkehr läuft auf der Best-Effort-Basis ab. Eine Garantie besteht nicht. Die Rahmen können verloren gehen und müssen dann erneut übertragen werden. Ein Slave hat möglicherweise nur einen ACL-Link zum Master.

Die Daten, die über ACL-Links gesendet werden, stammen aus der L2CAP-Schicht. Diese Schicht hat vier Hauptfunktionen. Als Erstes akzeptiert sie Pakete bis 64 KB Länge von den oberen Schichten und zerlegt sie zur Übertragung in Rahmen. Am anderen Ende werden die Rahmen wieder zu Paketen zusammengestellt. Zweitens verwaltet sie das Multiplexing und Demultiplexing mehrerer Paketquellen. Wird ein Paket erneut zusammengebaut, dann bestimmt die L2CAP-Schicht, an welches Protokoll der oberen Schicht es übergeben wird, beispielsweise RFcomm oder Diensterkennung. Drittens verwaltet L2CAP die Fehlerkontrolle und Neuübertragungen. Fehler werden erkannt und Pakete erneut gesendet, die nicht bestätigt wurden. Und schließlich behandelt L2CAP die Anforderungen für die Dienstgüte zwischen mehreren Links.

4.6.6 Bluetooth-Rahmenstruktur

Bluetooth definiert verschiedene Rahmenformate, zwei der wichtigsten sind in ►Abbildung 4.36 dargestellt. Ein Rahmen beginnt mit einem Zugriffscode, der in der Regel den Master bestimmt, sodass die Slaves in der Reichweite zweier Master unterscheiden können, welcher Datenverkehr für sie bestimmt ist. Als Nächstes kommt ein 54-Bit-Header mit den typischen Feldern der MAC-Teilschicht. Falls der Rahmen mit Basisdatenrate gesendet wird, dann kommt als Nächstes das Datenfeld. Es hat bis zu 2 744 Bit bei einer Übertragung über fünf Zeitscheiben. Bei einer einzigen Zeitscheibe ist das Format gleich, aber das Datenfeld hat nur 240 Bit.

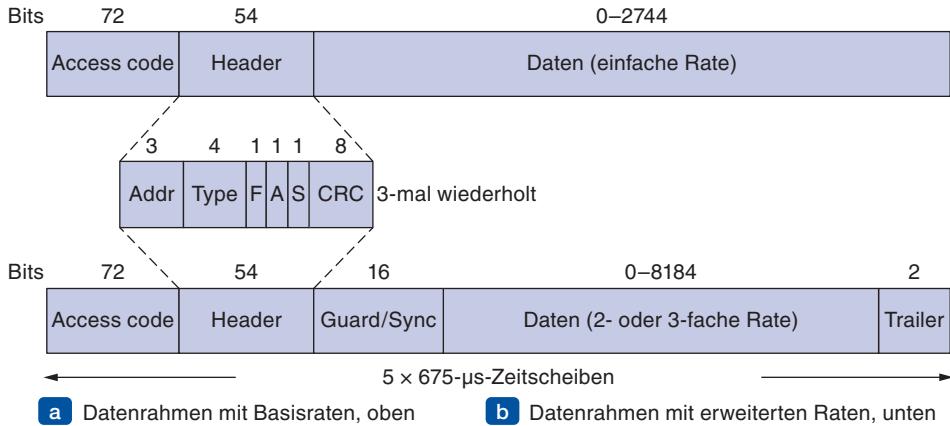


Abbildung 4.36: Typische Bluetooth-Datenrahmen (a) mit Basisdatenraten, (b) mit erweiterten Datenraten.

Falls der Rahmen mit erweiterter Rate gesendet wird, kann der Datenanteil bis zu zwei- oder dreimal so viele Bits betragen, weil jedes Symbol 2 oder 3 Bits statt 1 Bit überträgt. Diesen Daten ist ein Sicherheitsfeld und ein Synchronisationsmuster vorangestellt. Letzteres wird benutzt, um auf die schnellere Datenrate umzuschalten. Das heißt, Zugangscode und Header werden mit Basisrate übertragen und nur der Datenanteil mit der schnelleren Rate. Rahmen mit dieser erweiterten Datenrate enden mit einem kurzen Trailer.

Sehen wir uns den allgemeinen Header einmal kurz an. Das Feld *Address* gibt an, für welches der acht aktiven Geräte der Rahmen bestimmt ist. Das Feld *Type* gibt den Rahmentyp an (ACL, SCO, Polling oder Null), die Art der im Datenfeld verwendeten Fehlerkorrektur und wie viele Zeitscheiben der Rahmen umfasst. Das *Flow*-Bit wird von einem Slave gesetzt, wenn sein Puffer voll ist und er keine Daten mehr empfangen kann. Dieses Bit ermöglicht eine einfache Form der Flusskontrolle. Das *Acknowledgement*-Bit wird für den Huckepacktransport einer Bestätigung (ACK) auf einem Rahmen verwendet. Das *Sequence*-Bit dient der Nummerierung der Rahmen, um erneute Übertragungen zu erkennen. Als Protokoll wird Stop-and-Wait verwendet, sodass 1 Bit ausreichend ist. Dann folgt die 8-Bit-Prüfsumme (*Checksum*) des Headers. Der insgesamt 18 Bit lange Header wird dreimal wiederholt, um wie in Abbildung 4.36 einen 54 Bit langen Header zu bilden. Der Empfänger untersucht eine einfache Schaltung die drei Kopien jedes Bits. Sind alle drei gleich, wird das Bit akzeptiert. Wenn nicht, gewinnt die Meinung der Mehrheit. Es werden also 54 Bit Übertragungskapazität zum Senden der 10 Bit im Header verwendet; denn um Daten zuverlässig in einer rauschenden Umgebung mit billigen Geräten mit wenig Leistung (2,5 mW) und Rechenkapazität zu senden, ist ein hohes Maß an Redundanz erforderlich.

Für das Datenfeld von ACL- und SCO-Rahmen stehen verschiedene Formate zur Auswahl. Die SCO-Rahmen mit Basisrate sind ein einfaches Untersuchungsbeispiel: Das Datenfeld ist immer 240 Bit lang. Drei Varianten sind definiert, sodass an Nutzdaten 80, 160 oder 240 Bit möglich sind, und der Rest wird zur Fehlerkorrektur verwendet.

In der zuverlässigsten Version (80 Bit Nutzdaten) werden die Inhalte wie im Header dreimal wiederholt.

Wir können die Kapazität mit diesen Rahmen wie folgt berechnen. Da Slaves nur die ungeraden Zeitscheiben verwenden dürfen, erhalten sie wie der Master 800 Zeitscheiben/s. Bei 80 Bit Nutzdaten beträgt die Kanalkapazität des Slaves wie die des Masters 64 000 Bit/s. Diese Kapazität reicht für einen PCM-Duplexsprachkanal genau aus (daher wurde die Sprungrate von 1 600 Sprünge/s gewählt). Das heißt, trotz der rohen Bandbreite von 1 Mbit/s kann ein einzelner, unkomprimierter Duplexsprachkanal mit 64 000 Bit/s das Piconetz voll auslastet. Die Effizienz von 13 % resultiert daraus, dass 41 % der Kapazität für die Übergangsduer, 20 % für die Header und 26 % für die Codierung der Wiederholungen aufgewendet werden. Dieses Manövo verdeutlicht den Wert der erweiterten Raten und Rahmen auf mehr als einer einzelnen Zeitscheibe.

Man könnte noch viel mehr über Bluetooth sagen, leider fehlt hier der Platz. Für Wissbegierige: die Bluetooth-4.0-Spezifikation enthält alle Details.

4.7 RFID

Wir haben uns nun MAC-Entwürfe von PANs über LANs bis zu MANs angesehen. Als letztes Beispiel wollen wir eine Kategorie von billigen drahtlosen Geräten untersuchen, die häufig nicht als netzbildend erkannt werden: die **RFID-Tags** (*Radio Frequency IDentification*) und -Lesegeräte, die wir in Abschnitt 1.5.4 beschrieben haben.

Die RFID-Technologie hat viele Formen, sie wird in Smartcards, Tierimplantaten, Reisepässen, Bibliotheksbüchern und vielen weiteren Objekten benutzt. Die Form, die wir uns hier ansehen wollen, wurde im Zuge der Suche nach einem **elektronischer Produktcode (EPC, Electronic Product Code)** entwickelt, die mit der Gründung des Auto-ID Center am MIT im Jahr 1999 begann. Ein EPC ersetzt den Barcode, kann eine größere Informationsmenge tragen und ist über Entfernung bis zu 10 m elektronisch lesbar, selbst wenn es außer Sicht ist. Diese Technologie unterscheidet sich von anderen RFID-Technologien, beispielsweise von derjenigen, die in Reisepässen eingesetzt wird. Dort muss das RFID-Tag recht nah an ein Lesegerät geführt werden, um eine Transaktion durchzuführen. Aufgrund der Fähigkeit, über eine gewisse Entfernung zu kommunizieren, sind EPCs für unsere Untersuchungen relevanter.

Im Jahr 2003 wurde EPCglobal gegründet, um die RFID-Technologie zu kommerzialisieren, die vom Auto-ID Center entwickelt worden war. Die Bemühungen erhielten 2005 einen Auftrieb, als Walmart von seinen Top-100-Lieferanten forderte, alle Transporte mit RFID-Tags zu versehen. Eine breite Einführung wurde verhindert, da RFID nur schlecht mit billigen, gedruckten Barcodes konkurrieren kann, doch zurzeit entstehen neue Einsatzgebiete, beispielsweise in Führerscheinen. Wir werden die zweite Generation dieser Technologie beschreiben, die informell **EPC Gen 2** genannt wird (EPCglobal, 2008).

4.7.1 Architektur von EPC Gen 2

Die Architektur eines EPC-Gen-2-RFID-Netzes ist in ►Abbildung 4.37 dargestellt. Es hat zwei Hauptkomponenten: Tags und Lesegeräte. RFID-Tags sind kleine, billige Geräte, die einen eindeutigen 96-Bit-EPC-Identifikator und eine kleine Menge an Speicher haben, der vom RFID-Lesegerät ausgelesen und beschrieben werden kann. Der Speicher kann benutzt werden, um die Ortungshistorie einer Ware aufzuzeichnen, beispielsweise wenn diese die Lieferkette durchläuft.

Häufig sehen Tags wie Aufkleber aus, die zum Beispiel auf einer Jeanshose in den Regalen eines Geschäfts angebracht werden können. Der größte Teil des Aufklebers nimmt die Antenne ein, die darauf gedruckt ist. Ein winzig kleiner Punkt in der Mitte ist der integrierte Schaltkreis des RFID. Alternativ können RFID-Tags in ein Objekt eingebettet sein, wie beispielsweise beim Führerschein. In beiden Fällen haben die Tags keine Batterie und müssen Strom aus den Funkübertragungen eines in der Nähe befindlichen Lesegeräts gewinnen, um betriebsfähig zu sein. Diese Art Tags wird „Klasse 1“-Tags genannt, um sie von leistungsfähigeren Tags zu unterscheiden, die Batterien besitzen.

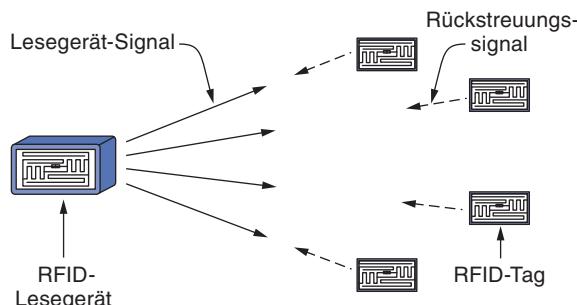


Abbildung 4.37: RFID-Architektur.

Die Lesegeräte stellen die Intelligenz im System dar, analog zu den Basisstationen und Zugangspunkten in zellulären und WiFi-Netzen. Lesegeräte sind sehr viel leistungsfähiger als Tags. Sie haben ihre eigenen Energiequellen, besitzen häufig mehrere Antennen und haben die Kontrolle, wenn Tags Nachrichten senden und empfangen. Da sich üblicherweise mehrere Tags innerhalb des Leseradius befinden, müssen die Lesegeräte das Problem des Mehrfachzugriffs lösen. Es kann auch vorkommen, dass sich mehrere Lesegeräte, die untereinander konkurrieren, im gleichen Bereich befinden.

Die Hauptaufgabe des Lesegeräts ist es, die Tags in der Nachbarschaft zu inventarisieren, das heißt, die Identifikatoren von in der Nähe befindlichen Tags zu entdecken. Diese Inventur wird mithilfe des Bitübertragungsprotokolls und des Protokolls zur Tag-Identifizierung durchgeführt, die in den nächsten Abschnitten kurz skizziert werden.

4.7.2 Bitübertragungsschicht von EPC Gen 2

Die Bitübertragungsschicht legt fest, wie Bits zwischen den RFID-Lesegeräten und den Tags gesendet werden. Dabei wird zu einem großen Teil auf Methoden zurückgegriffen, die zum Senden von drahtlosen Signalen verwendet werden und die wir bereits angesehen haben. In den USA werden Übertragungen im unlizenzierten 902–928-MHz-ISM-Band gesendet. Dieses Band fällt in den Dezimeterbereich oder UHF-Bereich, daher werden die Tags als UHF-RFID-Tags bezeichnet. Das Lesegerät führt mindestens alle 400 ms Frequenzsprünge durch, um das Signal über den Kanal zu streuen, Interferenzen zu begrenzen und regulatorische Anforderungen zu erfüllen. Das Lesegerät und die Tags benutzen zur Codierung der Bits Formen der ASK-Modulation (*Amplitude Shift Keying*), die wir in Abschnitt 2.5.2 besprochen haben. Sie wechseln sich mit dem Senden von Bits ab, daher ist die Verbindung halbduplex.

Es gibt zwei wesentliche Unterschiede zu den bisher untersuchten Bitübertragungsschichten. Der erste Unterschied ist, dass das Lesegerät immer ein Signal überträgt, unabhängig davon, ob das Lesegerät oder das Tag sendet. Natürlich übermittelt das Lesegerät ein Signal, um Bits zu den Tags zu senden. Damit die Tags Bits zum Lesegerät senden können, überträgt das Lesegerät ein festes Trägersignal, das keine Bits enthält. Die Tags sammeln dieses Signal ein, um die für ihren Betrieb nötige Energie zu erhalten; andernfalls wäre ein Tag überhaupt nicht in der Lage zu übermitteln. Um Daten zu senden, schaltet das Tag von dem Modus, in dem es das Signal vom Lesegerät reflektiert (wie ein Radarsignal, das von einem Ziel abprallt), in den Modus um, in dem es das Signal absorbiert.

Diese Methode wird **Rückstreuung** (*backscatter*) genannt. Sie unterscheidet sich dahingehend von all den anderen drahtlosen Situationen, die wir bisher betrachtet haben, dass Sender und Empfänger niemals gleichzeitig übertragen. Rückstreuung ist eine energiearme Möglichkeit, sodass die Tags ein schwaches Signal, das beim Lesegerät ankommt, selbst erzeugen können. Damit das Lesegerät das ankommende Signal decodieren kann, muss es das ausgehende Signal herausfiltern, welches das Lesegerät selbst übermittelt. Da das Tag-Signal schwach ist, können Tags ihre Bits nur mit einer niedrigen Rate zum Lesegerät senden und die Übertragungen von anderen Tags nicht empfangen bzw. noch nicht einmal mithören.

Der zweiter Unterschied ist, dass sehr einfache Formen der Modulation benutzt werden, damit diese auf einem Tag implementiert werden können, das mit sehr wenig Energie läuft und nur ein paar Cents in der Herstellung kostet. Um Daten zu den Tags zu senden, verwendet das Lesegerät zwei Amplitudenhöhen. Je nachdem, wie lange das Lesegerät vor einer Niedrigenergieperiode wartet, ist den Bits entweder eine 0 oder eine 1 zugeordnet. Das Tag misst die Zeit zwischen Niedrigenergieperioden und vergleicht diese Zeit mit einem Referenzwert, der während einer Präambel gemessen wurde. Wie in ► Abbildung 4.38 gezeigt, sind Einsen länger als Nullen.

Bei den Tag-Antworten wechselt der Rückstreuungszustand des Tags in festgelegten Intervallen, um eine Reihe von Impulsen in dem Signal zu erzeugen. Es können beliebig zwischen einem und acht Impulsperioden verwendet werden, um jeweils 0 oder 1

zu codieren, abhängig davon, wie zuverlässig die Übertragung sein muss. Einen haben weniger Übergänge als Nullen, wie Sie am Beispiel einer Codierung mit zwei Impulsperioden in Abbildung 4.38 sehen können.

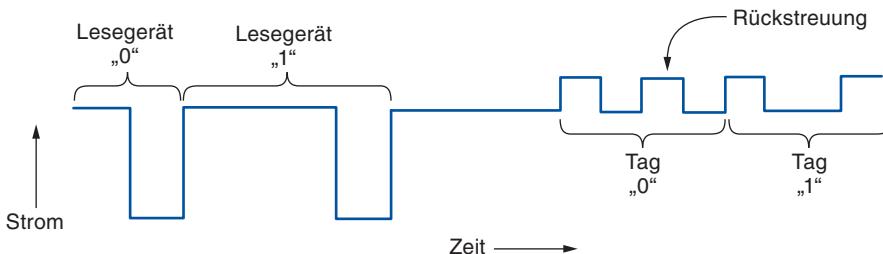


Abbildung 4.38: Rückstreuungssignale von Lesegerät und Tag.

4.7.3 Tag-Identifizierungsschicht von EPC Gen 2

Um die in der Nähe befindlichen Tags zu inventarisieren, müssen die Lesegeräte von jedem Tag eine Nachricht erhalten, die den Identifikator des Tags angibt. Diese Situation ist ein Mehrfachzugriffsproblem, bei dem im allgemeinen Fall die Anzahl der Tags unbekannt ist. Das Lesegerät könnte eine Anfrage via Broadcast aussenden, um alle Tags zu bitten, ihre Identifikatoren zu übermitteln. Doch dann würden die Antworten aller Tags, die direkt reagieren, genauso wie Stationen beim klassischen Ethernet kollidieren.

Wir haben in diesem Kapitel bereits viele Möglichkeiten gesehen, das Mehrfachzugriffsproblem zu lösen. Das Protokoll, das der aktuellen Situation, in der die Tags ihre Übertragungen gegenseitig nicht hören können, am nächsten kommt, ist S-ALOHA, eines der frühesten Protokolle, die wir untersucht haben. Dieses Protokoll wurde für den Einsatz in Gen 2 RFID angepasst.

In ►Abbildung 4.39 ist die Folge von Nachrichten dargestellt, die zur Identifikation eines Tags benutzt wird. In der ersten Zeitscheibe (*Slot 0*) sendet das Lesegerät eine *Query-Nachricht*, um den Prozess zu starten. Jede *QRepeat-Nachricht* rückt zur nächsten Zeitscheibe vor. Das Lesegerät teilt den Tags außerdem den Zeitscheibenbereich mit, über dem Übertragungen zufällig angeordnet sind. Die Verwendung von Bereichen ist notwendig, weil das Lesegerät am Anfang des Prozesses Tags synchronisiert; anders als bei Stationen im Ethernet wachen Tags nicht mit einer Nachricht zu einem Zeitpunkt ihrer Wahl auf.

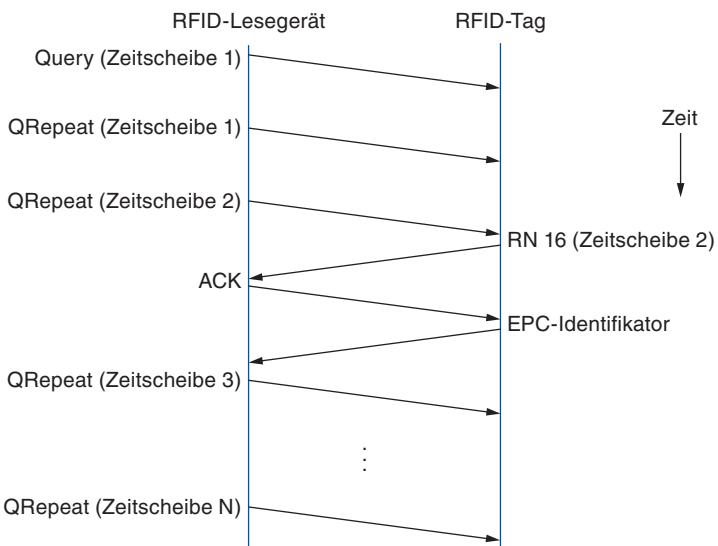


Abbildung 4.39: Beispiel eines Nachrichtenaustauschs zum Identifizieren eines Tags.

Tags wählen zufällig eine beliebige Zeitscheibe aus, in der sie antworten wollen. In Abbildung 4.39 antwortet das Tag in Zeitscheibe 2. Tags senden jedoch ihre Identifikatoren nicht mit ihrer ersten Antwort. Stattdessen sendet das Tag eine kurze 16-Bit-Zufallszahl in einer *RN16*-Nachricht. Wenn es keine Kollision gibt, empfängt das Lesegerät diese Nachricht und sendet seinerseits eine *ACK*-Nachricht. In dieser Phase hat das Tag die Zeitscheibe erhalten und sendet seinen EPC-Identifikator.

Der Grund für diese Art des Austauschs ist, dass EPC-Identifikatoren lang sind, sodass Kollisionen während des Sendens dieser Nachrichten teuer wären. Stattdessen wird zum Testen, ob das Tag die Zeitscheibe sicher verwenden kann, um seinen Identifikator zu senden, ein kurzer Nachrichtenaustausch vorgenommen. Sobald der Identifikator eines Tags erfolgreich übermittelt wurde, setzt dieses Tag vorläufig damit aus, auf neue *Query*-Nachrichten zu reagieren, sodass die verbleibenden Tags identifiziert werden können.

Ein Hauptproblem für das Lesegerät ist es, die Anzahl der Zeitscheiben so festzulegen, dass einerseits Kollisionen vermieden werden, aber andererseits nicht so viele Zeitscheiben verwendet werden, dass die Performanz leidet. Diese Einstellung ist analog dem binären exponentiellen Backoff bei Ethernet. Falls das Lesegerät feststellt, dass es zu viele Zeitscheiben ohne Antwort oder zu viele Zeitscheiben mit Kollisionen gibt, kann es eine *QAdjust*-Nachricht senden, um den Zeitscheibenbereich, in dem die Tags antworten, zu verkleinern bzw. zu vergrößern.

Das RFID-Lesegerät kann weitere Operationen auf den Tags durchführen. Zum Beispiel kann es eine Teilmenge der Tags auswählen, bevor es eine Inventur durchführt. Dadurch könnte es beispielsweise die Antworten von mit Tags versehenen Jeans, nicht aber die von Hemden erfassen. Das Lesegerät kann außerdem Daten auf die Tags schreiben, nachdem sie identifiziert sind. Diese Eigenschaft könnte benutzt werden, um den Verkaufsplatz oder andere relevanten Informationen aufzuzeichnen.

4.7.4 Nachrichtenformate der Tag-Identifizierung

Das Format der *Query*-Nachricht ist in ►Abbildung 4.40 als Beispiel einer Nachricht vom Lesegerät zum Tag gezeigt. Die Nachricht ist kompakt, weil die Downlink-Raten von 27 kbit/s bis zu 128 kbit/s begrenzt sind. Das *Command*-Feld trägt den Code 1 000, um die Nachricht als eine *Query* zu identifizieren.

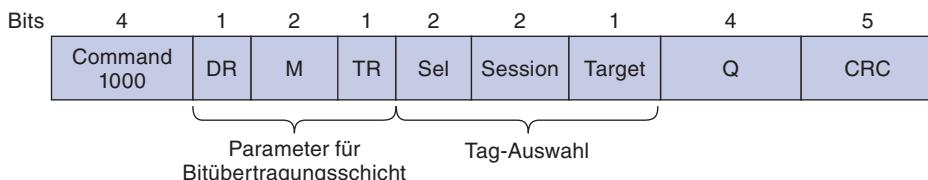


Abbildung 4.40: Format der *Query*-Nachricht.

Die nächsten Flags, *DR*, *M* und *TR*, legen die Parameter der Bitübertragungsschicht für die Übertragungen vom Lesegerät und die Tag-Antworten fest. Die Antwortrate kann beispielsweise zwischen 5 kbit/s und 640 kbit/s festgesetzt sein. Wir werden die Einzelheiten dieser Flags hier überspringen.

Dann kommen drei Felder, *Sel*, *Session* und *Target*, mit denen die Tags ausgewählt werden, die antworten dürfen. Ebenso wie die Lesegeräte in der Lage sind, eine Teilmenge an Identifikatoren auszuwählen, können die Tags bis zu vier gleichzeitig ablaufende Sitzungen nachvollziehen und festhalten, ob sie während dieser Sitzungen identifiziert wurden. Auf diese Art können mehrere Lesegeräte in überlappenden Abdeckungsbereichen betrieben werden, indem sie unterschiedliche Sitzungen verwenden.

Als Nächstes kommt der wichtigste Parameter für dieses Kommando, *Q*. Dieses Feld definiert den Bereich der Zeitscheiben, in denen Tags antworten werden, von 0 bis $2^Q - 1$. Zu guter Letzt gibt es zum Schutz der Nachrichtenfelder einen CRC-Wert. Mit 5 Bits ist dieser kürzer als die meisten CRCs, die wir bisher gesehen haben, aber die *Query*-Nachricht ist ebenfalls viel kürzer als die meisten Pakete.

Nachrichten vom Tag zum Lesegerät sind einfacher. Da das Lesegerät die Kontrolle hat, weiß es, mit welcher Nachricht als Antwort auf seine Übertragung zu rechnen ist. Die Tag-Antworten übertragen einfach Daten, wie den EPC-Identifikator.

Ursprünglich dienten die Tags nur der Identifikation. Sie vergrößerten sich jedoch im Laufe der Zeit, sodass sie heute sehr kleinen Computern ähneln. Einige Tags im Forschungsstadium haben Sensoren und sind in der Lage, kleine Programme auszuführen, die Daten sammeln und verarbeiten (Sample et al., 2008). Eine Vision für diese Technologie ist das „Internet der Dinge“, das Objekte in der physischen Welt mit dem Internet verbindet (Welbourne et al., 2009; Gershenfeld et al., 2004).

4.8 Switches der Sicherungsschicht

Viele Unternehmen haben mehrere LANs und möchten diese verbinden. Wäre es nicht praktisch, wenn wir die LANs einfach zusammenführen könnten, um ein größeres LAN zu bilden? Dies ist in der Tat möglich, wenn die Verbindungen mithilfe von sogenannten **Bridges** hergestellt werden. Die Ethernet-Switches, die wir in Abschnitt 4.3.4 beschrieben haben, sind ein moderner Name für Bridges; sie bieten Funktionalität, die über klassisches Ethernet und Ethernet-Hubs hinausgeht, um das Zusammenführen mehrerer LANs zu einem größeren und schnelleren Netzwerk zu vereinfachen. Wir werden die Begriffe „Bridge“ und „Switch“ synonym verwenden.

Bridges arbeiten auf der Sicherungsschicht, daher prüfen sie die Sicherungsschichtadressen zum Weiterleiten der Rahmen. Da sie nicht das Nutzdatenfeld des weitergeleiteten Rahmens untersuchen, können Sie IP-Pakete ebenso wie andere Paketarten wie AppleTalk-Pakete bearbeiten. Im Unterschied dazu untersuchen *Router* die Adressen in den Paketen und leiten sie dementsprechend weiter, sie können also nur mit den Protokollen betrieben werden, für die sie entworfen wurden.

In diesem Abschnitt werfen wir einen Blick darauf, wie Bridges funktionieren und benutzt werden, um mehrere physische LANs zu einem einzigen logischen LAN zu vereinen. Wir wollen uns außerdem den umgekehrten Fall ansehen, wie ein physisches LAN als viele logische LANs, sogenannte **virtuelle LANs (VLAN)**, behandelt werden kann. Beide Technologien stellen nützliche Flexibilität zur Netzverwaltung bereit. Eine umfassende Beschreibung von Bridges, Switches und damit in Verbindung stehenden Themen finden Sie bei Seifert und Edwards (2008) und Perlman (2000).

4.8.1 Verwendung von Bridges

Bevor wir in die Bridge-Techniken einsteigen, wollen wir einen Blick auf die üblichen Situationen werfen, in denen Bridges verwendet werden. Es werden drei Gründe aufgeführt, warum in einem einzelnen Unternehmen mehrere LANs vorhanden sein können.

Erstens haben viele Universitäts- und Firmenabteilungen ihr eigenes LAN, um damit ihre PCs, Server und Geräte wie Drucker zu verbinden. Da sich die Ziele der einzelnen Abteilungen unterscheiden, werden auch verschiedene LANs eingerichtet, ohne die Entscheidung anderer Abteilungen zu berücksichtigen. Früher oder später müssen sie jedoch zusammenarbeiten, also wird eine Art Brücke (*bridge*) benötigt. In diesem Beispiel entstanden die verschiedenen LANs aufgrund der Autonomie ihrer Besitzer.

Zweitens könnte das Unternehmen auf verschiedene, geografisch weit voneinander entfernte Gebäude aufgeteilt sein. Hier ist es kostengünstiger, in jedem Gebäude ein LAN einzurichten und diese LANs dann mit Bridges und ein paar Langstreckenglasfaserleitungen zu verbinden, als alle Kabel durch einen einzigen zentralen Switch zu verlegen. Selbst wenn das Verlegen der Kabel einfach durchzuführen ist, so gibt es doch Grenzen bezüglich ihrer Länge (z.B. 200 m für Twisted-Pair-Gigabit-Ethernet). Das Netzwerk würde für längere Kabel aufgrund der starken Signaldämpfung oder der Umlaufverzögerung nicht funktionieren. Die einzige Lösung ist, das LAN zu partitio-

nieren und Bridges zu installieren, um die Stücke zusammenzufügen. Damit wird die gesamte räumliche Entfernung vergrößert, die abgedeckt werden kann.

Drittens könnte es nötig sein, ein logisches Einzel-LAN in mehrere LANs (verbunden durch Bridges) aufzuteilen, um die Belastung zu verteilen. An vielen großen Universitäten stehen z.B. für Studenten und Mitarbeiter Tausende von Rechnern zur Verfügung. Unternehmen können ebenfalls Tausende von Angestellten haben. Bei Systemen dieses Umfangs können nicht alle Rechner an ein einzelnes LAN angeschlossen werden – es gibt mehr Computer als Ports an jedem Ethernet-Hub und mehr Stationen, als an einem einzigen klassischen Ethernet erlaubt ist.

Selbst wenn es möglich wäre, alle Workstations miteinander zu verdrahten – mehr Stationen an einen Ethernet-Hub oder klassisches Ethernet anzuschließen, würde nicht mehr Kapazität einbringen. All diese Stationen teilen die gleiche festgelegte Menge an Bandbreite. Je mehr Stationen es gibt, desto weniger durchschnittliche Bandbreite steht pro Station zur Verfügung.

Zwei separate LANs haben jedoch die doppelte Kapazität von einem einzelnen LAN. Bridges ermöglichen es, die LANs zusammenzuführen und gleichzeitig diese Kapazität beizubehalten. Der springende Punkt dabei ist, dass kein Datenverkehr auf Ports gesendet wird, wenn es nicht nötig ist, sodass jedes LAN mit voller Geschwindigkeit laufen kann. Dieses Verhalten erhöht außerdem die Zuverlässigkeit, da bei einem einzelnen LAN ein defekter Knoten, der unaufhörlich einen Nonsense-Datenstrom ausgibt, das gesamte LAN verstopfen kann. Bridges entscheiden, was weitergeleitet wird und was nicht, und verhalten sich damit wie Feuertüren in einem Gebäude: Sie verhindern, dass ein einzelner Knoten, der Amok läuft, das gesamte System zum Absturz bringt.

Um diese Vorteile leicht verfügbar zu machen, sollten Bridges im Idealfall völlig transparent sein. Es sollte möglich sein, loszugehen und Bridges zu kaufen, die LAN-Kabel in die Bridges stecken und alles läuft perfekt – sofort. Es sollten keine Hardwareänderungen erforderlich sein, keine Softwareänderungen, kein Einstellen von Adressen, kein Downloaden von Routing-Tabellen oder Parametern, nichts von all dem. Einfach die Kabel einstecken und fertig. Außerdem sollte der Betrieb der vorhandenen LANs in keiner Weise von den Bridges betroffen sein. Soweit es die Stationen betrifft, sollte es keinen beobachtbaren Unterschied geben, ob sie Teil eines per Bridges gekoppelten LAN sind oder nicht. Stationen innerhalb des Bridge-LAN zu verschieben sollte ebenso einfach sein, wie sie innerhalb eines Einzel-LAN zu bewegen.

Erstaunlicherweise ist es tatsächlich möglich, Bridges zu erzeugen, die transparent sind. Zwei Algorithmen werden verwendet: ein Backward-Learning-Algorithmus, um Datenverkehr zu stoppen, wo er nicht nötig ist; und ein Spannbaum-Algorithmus, um Schleifen aufzubrechen, die sich bilden können, wenn Switches notgedrungen miteinander verkabelt sind. Lassen Sie uns nun nacheinander einen Blick auf diese Algorithmen werfen und lernen, wie diese Zauberei bewerkstelligt wird.

4.8.2 Learning-Bridges

Die Topologie von zwei LANs, die per Bridge gekoppelt sind, ist in ▶ Abbildung 4.41 für zwei Fälle gezeigt. Auf der linken Seite werden zwei LANs (Kollisionsdomänen, zum Beispiel klassisches Ethernet) mithilfe einer speziellen Station – der Bridge – zusammengeführt, die in beiden LANs sitzt. Auf der rechten Seite sind LANs mit Punkt-zu-Punkt-Kabeln einschließlich eines Hubs miteinander verbunden. Die Stationen und der Hub sind an die Bridges angeschlossen. Falls die LAN-Technologie Ethernet ist, sind die Bridges besser bekannt als Ethernet-Switches.

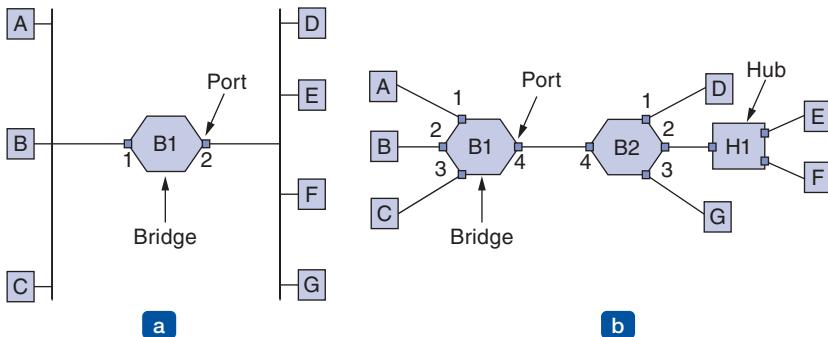


Abbildung 4.41: (a) Bridge, die zwei LANs verbindet. (b) Bridges (und ein Hub), die sieben Punkt-zu-Punkt-Stationen verbinden.

Bridges wurden entwickelt, als klassisches Ethernet in Gebrauch war, daher werden sie häufig in Topologien mit Multidrop-Kabeln wie in ▶ Abbildung 4.41a gezeigt. Die heutigen Topologien bestehen jedoch aus Punkt-zu-Punkt-Kabeln und -Switches. Die Bridges arbeiten in beiden Situationen auf die gleiche Weise. Alle Stationen, die am selben Port einer Bridge angeschlossen sind, gehören zur selben Kollisionsdomäne, die sich von den Kollisionsdomänen anderer Ports unterscheidet. Falls es wie beim klassischen Ethernet mehr als eine Station, einen Hub oder eine Halbduplexverbindung gibt, dann wird das CSMA/CD-Protokoll zum Senden der Rahmen verwendet.

Es gibt jedoch einen Unterschied in der Art, wie die Bridge-LANs aufgebaut werden. Um LANs per Bridge zu koppeln, wird eine Bridge als neue Station in jeder Kollisionsdomäne hinzugefügt (siehe Abbildung 4.41a). Um Punkt-zu-Punkt-LANs mithilfe von Bridges aufzubauen, sind die Hubs entweder mit einer Bridge verbunden, oder werden – vorzugsweise – zur Steigerung der Performanz durch eine Bridge ersetzt. In ▶ Abbildung 4.41b haben Bridges alle Hubs bis auf einen ersetzt.

Verschiedene Kabelarten können ebenfalls an eine Bridge angeschlossen werden. Zum Beispiel ist das Kabel, das in Abbildung 4.41b Bridge B_1 mit Bridge B_2 verbindet, vielleicht eine Glasfaserfernverbindung, während das Kabel, das die Bridges mit den Stationen verbindet, eine Kurzstrecken-Twisted-Pair-Leitung sein könnte. Dieses Arrangement ist sinnvoll, wenn LANs in verschiedenen Gebäuden per Bridge gekoppelt werden sollen.

Nun wollen wir einen Blick darauf werfen, was innerhalb der Bridges passiert. Jede Bridge operiert im Gemischtmodus (*promiscuous mode*), das heißt, sie akzeptiert jeden Rahmen, der von einer Station übermittelt wird, die an einen ihrer Ports angeschlossen sind. Die Bridge muss bei jedem Rahmen entscheiden, ob dieser weitergeleitet oder verworfen werden soll, und im ersten Fall außerdem, auf welchen Port der Rahmen ausgegeben wird. Diese Entscheidung wird mithilfe der Zieladresse getroffen. Betrachten Sie als Beispiel die Topologie von Abbildung 4.41a. Wenn Station A einen Rahmen an Station B sendet, dann wird Bridge B1 den Rahmen auf Port 1 erhalten. Dieser Rahmen kann direkt verworfen werden, weil er bereits auf dem richtigen Port ist. Nehmen wir jedoch an, in der Topologie von Abbildung 4.41b sendet A einen Rahmen an D. Bridge B1 wird diesen Rahmen auf Port 1 empfangen und auf Port 4 ausgeben. Bridge B2 wird dann den Rahmen auf ihrem Port 4 erhalten und auf ihrem Port 1 ausgeben.

Eine einfache Möglichkeit, dieses Schema zu implementieren, ist innerhalb der Bridge eine große (Hash-)Tabelle vorzuhalten. Die Tabelle kann jedes mögliche Ziel auflisten und zu welchem Ausgabeport dieses gehört. Zum Beispiel steht in Abbildung 4.41b in der Tabelle von B1, dass D zu Port 4 gehört, denn B1 muss lediglich wissen, auf welchen Port die Rahmen gelegt werden müssen, wenn D erreicht werden soll. Für B1 ist es nicht von Interesse, dass der Rahmen tatsächlich später, wenn er bei B2 eintrifft, noch weitergeleitet wird.

Bei der erstmaligen Installation der Bridges sind die Hash-Tabellen leer. Keine der Bridges weiß, wo welches Ziel liegt, sodass ein Flooding-Algorithmus (Fluten) verwendet wird: Jeder eintreffende Rahmen für ein unbekanntes Ziel wird auf jeden angeschlossenen Port ausgegeben, außer auf den, auf dem er angekommen ist. Mit der Zeit lernt die Bridge, wo die verschiedenen Ziele liegen. Irgendwann sind ihr alle Ziele bekannt und alle Rahmen werden an den jeweils richtigen Port abgegeben. Damit ist auch kein weiteres Fluten (Weiterleiten an alle LANs) mehr nötig.

Der Algorithmus, der von den Bridges benutzt wird, trägt die Bezeichnung **Backward Learning** (rückwärtsgerichtetes Lernen). Wie bereits erwähnt, arbeiten die Bridges im Gemischtmodus, sodass sie jeden gesendeten Rahmen auf jedem ihrer Ports sehen. Durch einen Blick auf die Quelladressen sehen sie, welche Rechner auf welchen Ports erreichbar sind. Wenn beispielsweise die Bridge B1 in Abbildung 4.41b erkennt, dass ein Rahmen auf Port 3 von C eingeht, weiß sie, dass C über Port 3 erreichbar ist. Folglich macht sie einen entsprechenden Eintrag in ihrer Hash-Tabelle. Jeder nachfolgende an C adressierte Rahmen, der an B1 über irgendeinen anderen Port eintrifft, wird an Port 3 weitergegeben.

Die Topologie kann sich ändern, wenn Rechner und Bridges ein- und ausgeschaltet oder verlegt werden. Für die dynamische Topologie wird die Ankunftszeit des Rahmens in den Hash-Tabelleneintrag mitaufgenommen. Jedes Mal, wenn ein Rahmen ankommt, dessen Quelle bereits eingetragen ist, wird die Zeit des Eintrags aktualisiert. Damit stellt die Zeitangabe im Eintrag den letzten Zeitpunkt dar, an dem ein Rahmen von diesem Rechner gesehen wurde.

In periodischen Abständen überprüft nun ein Prozess die Hash-Tabelle der Bridge und löscht alle Einträge, die älter als ein paar Minuten sind. Wird ein Computer von seinem LAN getrennt, an eine andere Stelle im Gebäude verlegt und dort wieder angeschaltet, ist er innerhalb weniger Minuten wieder lauffähig, ohne dass manuelle Eingaben erforderlich sind. Dieser Algorithmus bedeutet auch, dass der Verkehr zu einem mehrere Minuten lang untätigten Rechner per Fluten weitergeleitet werden muss, bis er selbst wieder einen Rahmen versendet.

Die Routing-Prozedur für einen eingehenden Rahmen hängt vom Port ab, auf dem der Rahmen ankommt (dem Quellport), und der Adresse, für die er gerichtet ist (der Zieladresse) ab. Das Verfahren ist wie folgt:

1. Ist der Port für die Zieladresse identisch mit dem Quellport, dann wird der Rahmen verworfen.
2. Sind der Port für die Zieladresse und der Quellport verschieden, dann wird der Rahmen an den Zielport weitergegeben.
3. Ist der Zielport unbekannt, dann wird Fluten verwendet und der Rahmen auf alle Ports außer dem Quellport geschickt.

Vielleicht fragen Sie sich, ob der erste Fall bei Punkt-zu-Punkt-Verbindungen überhaupt auftreten kann. Die Antwort ist: ja, falls Hubs verwendet werden, um mehrere Rechner mit einer Bridge zu verbinden. Ein Beispiel ist in Abbildung 4.41b zu sehen, wo die Stationen *E* und *F* mit Hub *H1* verbunden sind, der wiederum an Bridge *B2* angeschlossen ist. Falls *E* einen Rahmen zu *F* sendet, wird der Hub ihn zu *B2* sowie zu *F* weiterleiten. Das genau ist die Aufgabe von Hubs – sie verkabeln alle Ports miteinander, sodass die Eingabe für einen Port einfach die Ausgabe für alle anderen Ports ist. Der Rahmen kommt bei *B2* auf Port 4 an, was schon der richtige Ausgabeport ist, um das Ziel zu erreichen. Bridge *B2* muss nur den Rahmen verwerfen.

Dieser Algorithmus muss für jeden eingehenden Rahmen angewendet werden, sodass er gewöhnlich mit speziellen VLSI-Chips implementiert wird. Die Chips überprüfen die Tabelleneinträge in ein paar Mikrosekunden und bringen diese auf den neuesten Stand. Da sich die Bridges nur die MAC-Adressen ansehen, um zu entscheiden, wie ein Rahmen weitergeleitet werden soll, ist es möglich, mit der Weiterleitung anzufangen, sobald das Ziel-Header-Feld hereingekommen ist, also noch bevor der Rest des Rahmens angekommen ist (vorausgesetzt natürlich, dass die Ausgabeleitung verfügbar ist). Dieser Entwurf reduziert zum einen die Wartezeit, die Bridge zu durchlaufen, und ebenso die Anzahl der Rahmen, die die Bridge zwischenspeichern muss. Das Verfahren wird als **Cut-through-Switching** oder **Wormhole-Routing** bezeichnet und in der Regel von der Hardware behandelt.

Wir können die Arbeitsweise einer Bridge aus dem Blickwinkel von Protokollstapeln betrachten, um einen Eindruck davon zu bekommen, wie ein Gerät der Sicherungsschicht funktioniert. Sehen wir uns dazu einen Rahmen in der Konfiguration von Abbildung 4.41a an, in der die LANs Ethernet-LANs sind. Der Rahmen wird von Station *A* zu Station *D* gesendet und muss eine Bridge passieren. Die Protokollstapelsicht der Verarbeitung wird in ► Abbildung 4.42 gezeigt.

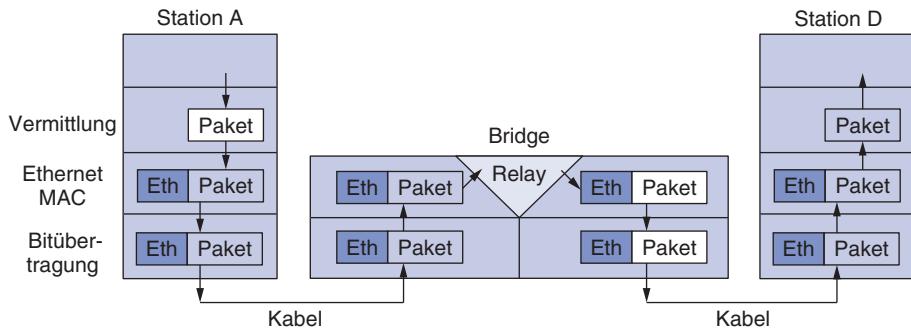


Abbildung 4.42: Protokollverarbeitung in einer Bridge.

Das Paket kommt von einer höheren Schicht und taucht in die Ethernet-MAC-Schicht hinab. Es bekommt einen Ethernet-Header (und auch einen Trailer, dieser ist nicht abgebildet). Diese Einheit wird an die Bitübertragungsschicht weitergereicht, über das Kabel hinausgeleitet und von der Bridge aufgenommen.

In der Bridge wird der Rahmen von der Bitübertragungsschicht an die Ethernet-MAC-Schicht weitergeleitet. Verglichen mit der Ethernet-MAC-Schicht der Station bietet diese Schicht erweiterte Verarbeitungsmöglichkeiten. Der Rahmen wird an ein Relais weitergegeben, immer noch innerhalb der MAC-Schicht. Die Bridge-Relaisfunktion verwendet nur den Ethernet-MAC-Header um festzustellen, wie der Rahmen zu behandeln ist. In diesem Fall gibt die Bridge den Rahmen zur Ethernet-MAC-Schicht des Ports weiter, über den Station D erreicht werden kann, und der Rahmen setzt seinen Weg fort.

Im allgemeinen Fall können Relais in einer bestimmten Schicht die Header für diese Schicht neu schreiben. VLANs werden uns dafür in Kürze ein Beispiel liefern. Auf keinen Fall sollte die Bridge in den Rahmen hineinsehen und herausfinden, dass dieser ein IP-Paket beinhaltet; das ist irrelevant für die Verarbeitung in der Bridge und stellt eine Verletzung der Protokollsichten dar. Beachten Sie außerdem, dass eine Bridge mit k Ports k Instanzen von MAC- und Bitübertragungsschichten haben wird. Der Wert von k ist in unserem einfachen Beispiel 2.

4.8.3 Spannbäume und Bridges

Um die Zuverlässigkeit zu erhöhen, können redundante Verbindungen zwischen Bridges eingesetzt werden. Im Beispiel von ►Abbildung 4.43 gibt es zwei parallele Verbindungen zwischen einem Paar von Bridges. Dieser Entwurf stellt sicher, dass im Falle einer Verbindungstrennung das Netzwerk nicht in zwei Mengen von Rechnern partitioniert wird, die nicht miteinander sprechen können.

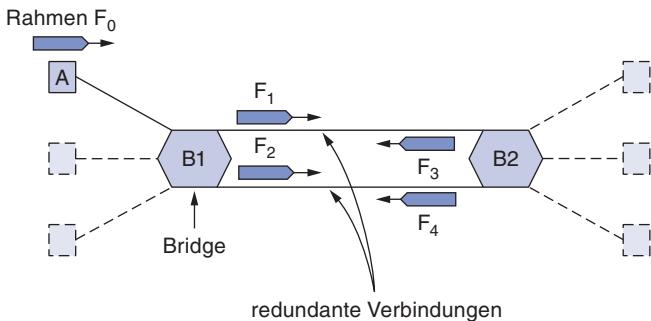


Abbildung 4.43: Bridges mit zwei parallelen Verbindungen.

Diese Redundanz verursacht jedoch zusätzliche Probleme, weil dadurch in der Topologie Schleifen erzeugt werden. Ein Beispiel dieser Problematik können wir in Abbildung 4.43 bei der Behandlung eines Rahmens sehen, der von A an ein bislang unbeobachtetes Ziel gesendet wurde. Jede Bridge folgt der normalen Regel für die Behandlung unbekannter Ziele, das heißt, Rahmen mit unbekannter Adresse werden geflutet. Nennen wir den Rahmen von A, der Bridge B1 erreicht, Rahmen F_0 . Die Bridge sendet Kopien dieses Rahmens auf jedem ihren Ports aus. Wir betrachten lediglich die Bridge-Ports, die B1 mit B2 verbinden (obwohl der Rahmen auch auf die anderen Ports gesendet wird). Da es zwei Verbindungen von B1 nach B2 gibt, kommen zwei Kopien des Rahmens bei B2 an. Diese sind in Abbildung 4.43 als F_1 und F_2 dargestellt.

Kurz danach erhält Bridge B2 die Rahmen. B2 weiß nicht (und kann auch nicht wissen), dass dies Kopien desselben Rahmens sind, und nicht zwei unterschiedliche Rahmen, die nacheinander gesendet wurden. Deshalb nimmt Bridge B2 den Rahmen F_1 an und sendet Kopien davon auf alle anderen Ports. Und B2 nimmt auch F_2 an und sendet Kopien davon auf alle anderen Ports. Dadurch werden die Rahmen F_3 und F_4 erzeugt, die über die zwei Verbindungen zurück zu B1 gesendet werden. Bridge B1 erkennt zwei neue Rahmen mit unbekanntem Ziel und kopiert diese erneut. Dieser Kreislauf setzt sich unendlich fort.

Die Lösung für dieses Problem sieht die Kommunikation der Bridges untereinander und die Überlagerung der aktuellen Topologie mit einem Spannbaum (*spanning tree*) vor, der jede Bridge erreicht. Um eine fiktive schleifenlose Topologie aufzubauen (eine Teilmenge der aktuellen Topologie), werden in der Tat einige potenzielle Verbindungen zwischen den Bridges ignoriert.

In ► Abbildung 4.44 sehen wir beispielsweise fünf Bridges, die zusammengeschlossen sind, außerdem ist jede mit Stationen verbunden. Jede Station ist nur mit einer Bridge verbunden. Es gibt ein paar redundante Verbindungen zwischen den Bridges, sodass Rahmen in Schleifen weitergeleitet werden, falls alle Verbindungen benutzt werden. Diese Topologie kann man sich als einen Graphen vorstellen, in dem die Bridges die Knoten und Punkt-zu-Punkt-Verbindungen die Kanten sind. Der Graph kann auf einen Spannbaum reduziert werden, der definitionsgemäß keine Zyklen enthält, indem man die in Abbildung 4.44 als gestrichelte Linien dargestellten Verbindungen weglässt. Bei

diesem Spannbaum gibt es genau einen Pfad von jeder Station zu jeder anderen. Haben die Bridges einmal Übereinkunft über den Spannbaum getroffen, erfolgen alle Übertragungen zwischen den Stationen über den Spannbaum. Da von jeder Quelle zu jedem Ziel ein eindeutiger Pfad führt, sind Schleifen unmöglich.

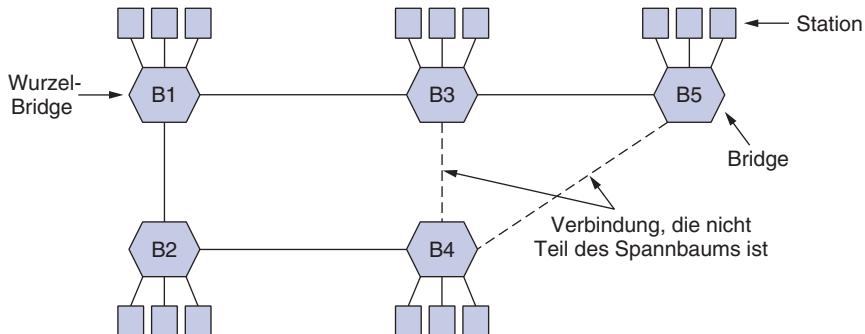


Abbildung 4.44: Ein Spannbaum verbindet fünf Bridges. Die gestrichelten Linien sind nicht Teil des Spannbaums.

Zur Errichtung des Spannbaums führen die Bridges einen verteilten Algorithmus aus. Jede Bridge sendet periodisch eine Konfigurationsnachricht auf allen seinen Ports zu seinen Nachbarn und verarbeitet die Nachrichten, die sie von anderen Bridges empfängt, wie im Folgenden beschrieben wird. Diese Nachrichten werden nicht weitergeleitet, da sie nur dazu dienen, den Baum aufzubauen, der dann zur Weiterleitung benutzt werden kann.

Die Bridges müssen zuerst eine Bridge als Wurzel des Spannbaums auswählen. Dazu fügt jede Bridge einen Identifikator, der auf ihrer MAC-Adresse basiert, in die Konfigurationsnachricht ein und außerdem den Identifikator der Bridge, den sie für die Wurzel halten. MAC-Adressen werden vom Hersteller installiert und sind weltweit garantiert eindeutig, was diese Adresse praktisch und einzigartig macht. Die Bridge mit dem niedrigsten Identifikator wird als Wurzel ausgewählt. Nachdem genügend Nachrichten ausgetauscht wurde, um die Neuigkeit zu verbreiten, wissen alle Bridges, welche von ihnen die Wurzel ist. In Abbildung 4.44 hat B1 den niedrigsten Identifikator und wird die Wurzel.

Als Nächstes wird ein Baum mit kürzestmöglichen Pfaden von der Wurzel zu jeder Bridge aufgebaut. In Abbildung 4.44 können die Bridges B2 und B3 jeweils von Bridge B1 aus direkt mit einem Sprung erreicht werden, dies ist der kürzeste Pfad. Bridge B4 kann mit zwei Sprüngen erreicht werden, entweder über B2 oder über B3. Um diese Verbindung abzubrechen, wird der Pfad mit dem kleinsten Identifikator ausgewählt, also wird B4 über B2 erreicht. Bridge B5 kann mit zwei Sprüngen über B3 erreicht werden.

Um diese kürzesten Pfade zu finden, fügen die Bridges die Entfernung von der Wurzel in ihre Konfigurationsnachrichten ein. Jede Bridge merkt sich den kürzesten Pfad, den sie zur Wurzel findet. Die Bridges schalten dann die Ports ab, die nicht Teil des kürzesten Pfads sind.

Obwohl der Baum alle Bridges umfasst, müssen nicht unbedingt alle Verbindungen (noch nicht einmal alle Bridges) im Baum enthalten sein. Dies liegt daran, dass durch das Abschalten der Ports einige Verbindungen vom Netzwerk getrennt werden, um Schleifen zu vermeiden. Sogar nach der Erstellung des Spannbaums läuft der Algorithmus im Normalbetrieb weiter, um Änderungen in der Topologie automatisch zu erfassen und den Baum entsprechend anzupassen.

Der zum Aufbau des Spannbaums verwendete Algorithmus wurde von Radia Perlman entwickelt. Sie hatte die Aufgabe, das Problem zu lösen, wie LANs ohne Schleifen zusammengeführt werden können. Ihr wurde eine Woche Zeit dazu gegeben, doch ihr kam die Idee für den Spannbaum-Algorithmus in einem Tag. Zum Glück, denn so blieb ihr noch genug Zeit, ein Gedicht darüber zu schreiben (Perlman, 1985):

*I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree which must be sure to span.
So packets can reach every LAN.
First the Root must be selected
By ID it is elected.
Least cost paths from Root are traced
In the tree these paths are placed.
A mesh is made by folks like me
Then bridges find a spanning tree.*

Der Spannbaum-Algorithmus wurde dann in IEEE 802.1D standardisiert und über viele Jahre verwendet. 2001 wurde er überarbeitet, um einen neuen Spannbaum nach einer Änderung in der Topologie noch schneller zu finden. Eine ausführliche Behandlung von Bridges finden Sie in (Perlman, 2000).

4.8.4 Repeater, Hubs, Bridges, Switches, Router und Gateways

Wir haben uns in diesem Buch bislang mit verschiedenen Möglichkeiten befasst, Rahmen und Pakete von einem Rechner zu einem anderen zu transportieren. Wir haben Repeater, Hubs, Bridges, Switches, Router und Gateways erwähnt. Diese Geräte werden alle häufig eingesetzt und unterscheiden sich auf subtile und auch auf weniger subtile Art und Weise. Da es so viele davon gibt, ist es wahrscheinlich eine gute Idee, dass wir sie uns zusammen ansehen, um die Ähnlichkeiten und Unterschiede zu entdecken.

Der Schlüssel zum Verständnis dieser Geräte ist sich bewusst zu machen, dass sie auf verschiedenen Schichten arbeiten (►Abbildung 4.45a). Die Schicht spielt eine Rolle, weil die verschiedenen Geräte verschiedene Informationsstücke verwenden, um die Art der Vermittlung zu bestimmen. In einem typischen Fall erzeugt ein Benutzer einige Daten, die an einen entfernten Rechner gesendet werden sollen. Diese Daten werden an die Transportschicht übergeben, die dann einen Header (beispielsweise

einen TCP-Header) hinzufügt und das Ergebnis an die Vermittlungsschicht übergibt. Die Vermittlungsschicht fügt einen eigenen Header hinzu, um ein Paket der Vermittlungsschicht (z.B. ein IP-Paket) zu bilden. In ►Abbildung 4.45b sehen wir ein dunkelblaues IP-Paket. Das Paket geht dann zur Sicherungsschicht, die einen eigenen Header und die Prüfsumme (CRC) hinzufügt und den entstandenen Rahmen an die Bitübertragungsschicht zur Übertragung, beispielsweise über ein LAN, weitergibt.

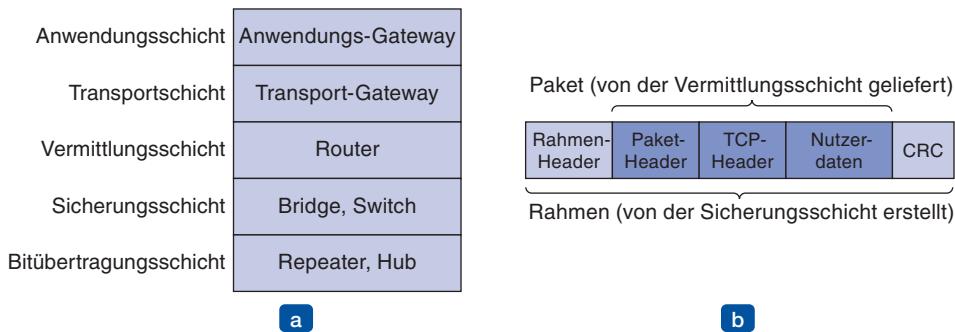


Abbildung 4.45: (a) Zuordnung Gerät zu Schicht. (b) Rahmen, Pakete und Header.

Betrachten wir nun die Vermittlungsgeräte und deren Beziehung zu Paketen und Rahmen. Auf der untersten Ebene in der Bitübertragungsschicht finden wir Repeater. Dabei handelt es sich um analoge Geräte, die mit Signalen auf den Kabeln, an die sie angegeschlossen sind, arbeiten. Ein Signal, das auf einem Kabel auftritt, wird bereinigt, verstärkt und auf einem anderen Kabel ausgegeben. Repeater verstehen keine Rahmen, Pakete oder Header. Sie verstehen die Symbole, die Bits als Voltangaben codieren. Das klassische Ethernet wurde beispielsweise zur Unterstützung von vier Repeatern konzipiert, die das Signal verstärken, um die maximale Kabellänge von 500 m auf 2 500 m zu erweitern.

Als Nächstes kommen wir zu den Hubs. Ein Hub verfügt über mehrere Eingangsleitungen, die er elektronisch verbindet. Rahmen, die auf einer Leitung eingehen, werden auf allen anderen versendet. Wenn zwei Rahmen gleichzeitig eintreffen, stoßen sie wie bei einem Koaxialkabel zusammen. Alle Leitungen, die zum Hub führen, müssen mit der gleichen Geschwindigkeit arbeiten. Hubs unterscheiden sich von Repeatern darin, dass sie (in der Regel) die eingehenden Signale nicht verstärken und für mehrere Eingänge ausgelegt sind, aber die Unterschiede sind hier gering. Wie Repeaters sind Hubs Geräte der Bitübertragungsschicht, die die Sicherungsschichtadressen nicht untersuchen und auch nicht verwenden.

Nun wollen wir zur Sicherungsschicht weitergehen, wo wir Bridges und Switches finden. Wir haben uns ausführlich mit Bridges beschäftigt. Eine Bridge verbindet zwei oder mehrere LANs. Wie ein Hub verfügt eine moderne Bridge über mehrere Ports, in der Regel genug für 4 bis 48 Eingangsleitungen eines bestimmten Typs. Anders als beim Hub ist jeder Port isoliert und besitzt eine eigene Kollisionsdomäne; hat der Port eine Duplex-Punkt-zu-Punkt-Leitung, so wird der CSMA/CD-Algorithmus nicht benötigt. Kommt ein Rahmen an, dann extrahiert die Bridge die Zieladresse aus dem Rahmen-Header und schlägt sie in einer Tabelle nach um herauszufinden, wohin der

Rahmen gesendet werden soll. Bei Ethernet ist diese Adresse die 48-Bit-Zieladresse aus Abbildung 4.14. Die Bridge gibt den Rahmen nur dann auf den Port, wenn er benötigt wird, und kann mehrere Rahmen gleichzeitig weiterleiten.

Bridges bieten deutlich bessere Performanz als Hubs und die Isolation zwischen den Bridge-Ports bedeutet auch, dass Eingabeleitungen mit unterschiedlichen Geschwindigkeiten laufen können, möglicherweise sogar mit unterschiedlichen Netztypen. Ein häufiges Beispiel ist eine Bridge mit Ports, die 10-, 100-, und 1 000-Mbit/s-Ethernet verbinden. Innerhalb der Bridge wird ein Zwischenspeicher benötigt, um einen Rahmen auf einem Port akzeptieren und diesen auf einem anderen Port übermitteln zu können. Falls Rahmen schneller ankommen als sie weitergeleitet werden können, kann der Bridge der Zwischenspeicher ausgehen und sie muss anfangen, Rahmen zu verwerten. Falls beispielsweise ein Gigabit-Ethernet Bits mit höchster Geschwindigkeit in ein 10-Mbit/s-Ethernet strömen lässt, dann wird die Bridge diese puffern müssen – in der Hoffnung, dass ihr der Speicher nicht ausgeht. Dieses Problem existiert selbst dann noch, wenn alle Ports mit derselben Geschwindigkeit laufen, weil eventuell mehr als ein Port Rahmen zu einem bestimmten Zielport sendet.

Bridges waren ursprünglich dazu gedacht, verschiedene LAN-Arten miteinander zu verknüpfen, zum Beispiel ein Ethernet- und ein Token-Ring-LAN. Aufgrund der Unterschiede zwischen den LANs hat dies jedoch niemals gut funktioniert. Verschiedene Rahmenformate erfordern Kopieren und Neuformatierungen, was CPU-Zeit in Anspruch nimmt, eine neue Prüfsummenberechnung nötig macht und möglicherweise unentdeckte Fehler aufgrund von defekten Bits im Speicher der Bridge einführt. Unterschiedliche Rahmenlängen sind ebenfalls ein ernstes Problem, für das es noch keine gute Lösung gibt. Im Grunde müssen Rahmen, die zu groß sind, um weitergeleitet werden, verworfen werden. So viel zur Transparenz.

Zwei andere Bereiche, in denen sich LANs unterscheiden können, sind Sicherheit und Dienstgüte. Einige LANs haben Sicherheitsschichtverschlüsselung, zum Beispiel IEEE 802.11, und einige nicht, zum Beispiel Ethernet. Einige LANs bieten Dienstgütfunktionen wie Prioritäten, zum Beispiel IEEE 802.11, und einige nicht, zum Beispiel Ethernet. Wenn sich also ein Rahmen zwischen diesen LANs bewegen muss, so können die Sicherheit oder die Dienstgüte, die vom Sender erwartet werden, möglicherweise nicht gewährleistet werden. Aus all diesen Gründen funktionieren moderne Bridges in der Regel für einen Netztyp, und wenn es darum geht, Netze unterschiedlicher Typen zusammenzuschließen, werden stattdessen Router verwendet, zu denen wir bald kommen.

Switches sind moderne Bridges mit anderem Namen. Die Unterschiede haben eher mit Marketing- als mit technischen Gründen zu tun, doch es gibt ein paar Dinge, die man darüber wissen sollte. Bridges wurden entwickelt, als klassisches Ethernet eingesetzt wurde. Sie mussten tendenziell relativ wenige LANs verbinden und haben daher relativ wenig Ports. Der Begriff „Switch“ ist heutzutage gebräuchlicher. Außerdem verwenden alle modernen Einrichtungen Punkt-zu-Punkt-Verbindungen, wie Twisted-Pair-Kabel, sodass einzelne Rechner direkt an einen Switch angeschlossen werden, und somit wird der Switch tendenziell viele Ports haben. Schließlich wird „Switch“

auch als allgemeiner Ausdruck verwendet. Bei einer Bridge ist die Funktionalität klar. Ein Switch andererseits könnte einen Ethernet-Switch bezeichnen oder eine völlig andere Geräteart, welche mit Entscheidungen zur Weiterleitung befasst ist, beispielsweise eine Telefonvermittlungsstelle (*telephone switch*).

Nun kennen wir also Repeaters und Hubs, die in der Tat recht ähnlich sind, wie auch Bridges und Switches, die sich sogar noch mehr ähneln. Betrachten wir nun die Router, die sich von den anderen unterscheiden. Kommt ein Paket zu einem Router, werden der Header und Trailer des Rahmens entfernt und das Paket, das sich im Nutzdatenfeld des Rahmens befindet (in Abbildung 4.45 dunkelblau), wird an die Routing-Software übergeben. Diese Software bestimmt anhand der Paket-Header eine Ausgabeleitung. Bei einem IP-Paket enthält der Paket-Header eine 32-Bit- (IPv4) oder 128-Bit-Adresse (IPv6), aber keine IEEE-802-Adresse mit 48 Bit. Die Routing-Software sieht die Rahmenadresse nicht. Sie weiß nicht einmal, ob ein Paket aus einem LAN oder einer Punkt-zu-Punkt-Leitung eingegangen ist. Wir beschäftigen uns mit Routern in *Kapitel 5*.

Eine Schicht weiter oben finden wir die Transport-Gateways. Sie verbinden zwei Rechner, die verschiedene verbindungsorientierte Transportprotokolle verwenden. Angenommen, ein Computer mit dem verbindungsorientierten TCP/IP-Protokoll muss mit einem Computer kommunizieren, der ein anderes verbindungsorientiertes Transportprotokoll namens SCTP verwendet. Das Transport-Gateway kann die Pakete von einer Verbindung auf eine andere kopieren und sie bei Bedarf umformatieren.

Darüber hinaus verstehen Anwendungs-Gateways das Format und die Inhalte der Daten und können Nachrichten von einem Format in ein anderes übersetzen. Ein E-Mail-Gateway könnte beispielsweise die Internetnachrichten in SMS-Nachrichten für Mobiltelefone konvertieren. Ebenso wie „Switch“ ist „Gateway“ ein eher allgemeiner Begriff, der sich auf den Weiterleitungsprozess in einer höheren Schicht bezieht.

4.8.5 Virtuelle LANs

In den frühen Tagen der lokalen Netze wandten sich dicke gelbe Kabel durch die Kabelführungen in vielen Bürogebäuden. Sie waren mit jedem Computer, an dem sie vorbeiliefen, verbunden. Man kümmerte sich nicht darum, welcher Computer zu welchem LAN gehörte. Alle Mitarbeiter in benachbarten Büros wurden im gleichen LAN zusammengefasst, ob sie nun zusammengehörten oder nicht. Die Geografie siegte über Firmen-Organigramme.

Mit der Einführung von Twisted-Pair-Kabeln und Hubs in den 1990er Jahren änderte sich dies. Die Gebäude wurden neu verkabelt (mit erheblichen Kosten), um die gelben Gartenschläuche zu entfernen und stattdessen Twisted-Pair-Kabel von jedem Büro in einen Kabelschrank am Ende des Flurs oder in ein zentrales Rechenzentrum zu legen (►Abbildung 4.46). Wenn der für die Verkabelung zuständige Abteilungsleiter ein Visionär war, wurden Twisted-Pair-Kabel der Kategorie 5 installiert. Wenn er ein Erbsenzähler war, wurde die vorhandene Telefonleitung (Kategorie 3) verwendet – nur um sie ein paar Jahre später mit dem Aufkommen von Fast Ethernet zu ersetzen.

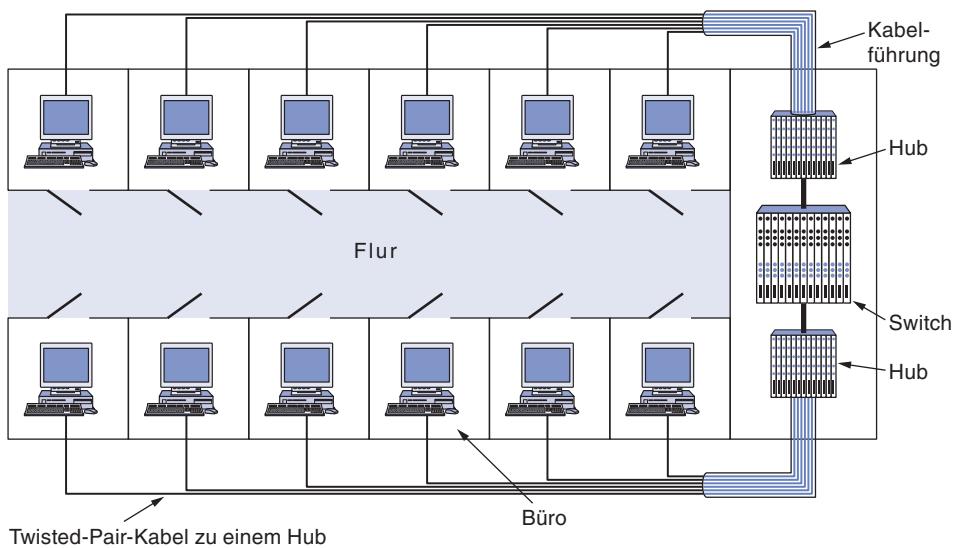


Abbildung 4.46: Gebäude mit zentraler Verkabelung mit Hubs und einem Switch.

Heute haben sich die Kabel verändert und Hubs sind zu Switches geworden, doch das Verkabelungsmuster ist immer noch dasselbe. Dieses Muster macht es möglich, LANs logisch anstatt physikalisch zu konfigurieren. Wenn beispielsweise ein Unternehmen k LANs benötigte, konnte es k Hubs kaufen. Indem die Stecker für das einzelne Switch sorgfältig zugeordnet werden, können die Rechner in einem LAN auf eine Weise gewählt werden, die auch organisatorisch sinnvoll ist – ohne Rücksicht auf die Geografie.

Spielt es eine Rolle, wer in welchem LAN ist? Schließlich sind ja in beinahe allen Unternehmen die LANs miteinander verbunden. Kurz gesagt: ja, es spielt eine Rolle. Netzadministratoren gruppieren Benutzer in LANs aus verschiedenen Gründen lieber nach organisatorischen Gesichtspunkten. Ein Aspekt ist hier die Sicherheit. Ein LAN könnte Webserver und andere Computer beherbergen, die zur öffentlichen Nutzung bestimmt sind. Ein anderes LAN könnte Host für Rechner sein, die Datensätze der Personalabteilung enthalten, die diese Abteilung nicht verlassen dürfen. In einer solchen Situation ist es sinnvoll, alle Computer in einem einzelnen LAN zusammenzufassen und keinen Zugriff auf die Server von außerhalb des LAN zuzulassen. Die Geschäftsleitung reagiert eher mit Stirnrunzeln, wenn sie hört, dass dies unmöglich ist.

Ein zweiter Aspekt ist die Last. Da einige LANs viel intensiver genutzt werden als andere, kann es vorteilhaft sein, diese zu trennen. Wenn beispielsweise die Forschungsabteilung allerlei knifflige Experimente durchführt, die das LAN völlig auslasten, dann sind die Leute in der Geschäftsführung möglicherweise nicht begeistert, etwas von ihrer Kapazität zu spenden, die sie gerade für eine Videokonferenz benötigen. Andererseits könnte dies der Geschäftsführung nachdrücklich vermitteln, dass ein schnelleres Netzwerk installiert werden muss.

Ein dritter Aspekt ist Broadcast-Verkehr. Bridges senden Datenverkehr via Broadcast, wenn der Standort des Ziels unbekannt ist, und auch Protokolle der oberen Schichten nutzen Broadcasting. Wenn beispielsweise ein Benutzer ein Paket an die IP-Adresse x senden möchte, woher weiß er dann, welche MAC-Adresse in diesen Rahmen aufgenommen werden soll? Dieses Thema wird in *Kapitel 5* behandelt. Doch kurz zusammengefasst lautet die Antwort, dass ein Rahmen mit der Frage rundgesendet wird: Wem gehört die IP-Adresse von x? Dann wird auf die Antwort gewartet. Mit dem Anwachsen der Rechneranzahl in einem LAN erhöht sich auch die Anzahl der Broadcasts. Jede Broadcast-Übertragung verbraucht mehr von der LAN-Kapazität als ein normaler Rahmen, weil es an jeden Computer im LAN gesendet wird. Die Auswirkung des Broadcast-Verkehrs wird verringert, indem LANs nicht größer als nötig gehalten werden.

Bei Broadcasting besteht das Problem, dass eine Netzwerkkarte einmal versagen kann oder falsch konfiguriert ist und anfängt, einen unendlichen Strom an Broadcast-Rahmen zu senden. Und wenn das Netz richtig Pech hat, wird einer dieser Rahmen Reaktionen auslösen, die zu noch mehr Datenverkehr führen. Das Ergebnis dieses **Broadcast-Sturms** ist, dass erstens die gesamte Kapazität des LAN belegt wird und zweitens alle Rechner in den miteinander verbundenen LANs damit beschäftigt sind, diese Rahmen zu verarbeiten und zu verwerfen.

Auf den ersten Blick erscheint es einfach, das Ausmaß von Broadcast-Stürmen durch das Aufteilen von LANs mit Bridges oder Switches zu begrenzen. Aber wenn Transparenz das Ziel ist (d.h., ein Rechner kann über die Bridge an ein anderes LAN angeschlossen werden, ohne dass dies jemand bemerkt), dann müssen Bridges die Broadcast-Rahmen weitergeben.

Da wir nun wissen, warum Unternehmen mehrere LANs mit begrenztem Wirkungskreis haben möchten, kehren wir zu dem Problem zurück, die logische Topologie von der physischen Topologie zu lösen. Eine physische Topologie aufzubauen, welche die organisatorische Struktur widerspiegelt, kann Arbeit und Kosten verursachen, selbst bei zentralisierter Verkabelung und mit Switches. Falls zum Beispiel zwei Leute aus der gleichen Abteilung in verschiedenen Gebäuden arbeiten, könnte es einfacher sein, diese an unterschiedliche Switches anzuschließen, die Teil von unterschiedlichen LANs sind. Selbst wenn dies nicht der Fall ist, könnte ein Benutzer in eine andere Abteilung versetzt werden, ohne das Büro zu wechseln, oder er könnte das Büro wechseln, nicht aber die Abteilung. Das Ergebnis wäre, dass der Benutzer im falschen LAN ist, bis ein Administrator den Anschlussstecker des Benutzers von einer Switch zu einer anderen wechselt. Außerdem könnte die Anzahl der Rechner, die zu verschiedenen Abteilungen gehören, nicht gut mit der Anzahl der Ports in Switches zusammenpassen; einige Abteilungen sind vielleicht zu klein, andere wiederum zu groß, sodass sie mehrere Switches benötigen. Dies führt dazu, dass ungenutzte Switch-Ports verschwendet werden.

In vielen Unternehmen gibt es häufig organisatorische Veränderungen, sodass die Systemadministratoren sehr viel Zeit damit verbringen, die Stecker richtig umzustecken. Manchmal geht das auch nicht, weil das Twisted-Pair-Kabel vom Rechner des Benut-

zers zu weit vom richtigen Switch entfernt ist (z.B. im falschen Gebäude ist) oder sich die verfügbaren Switch-Ports im falschen LAN befinden.

Da die Kunden mehr Flexibilität forderten, begannen die Netzwerkhersteller mit der Erarbeitung einer Möglichkeit, ganze Gebäude nur über Software neu zu verkabeln. Dieses Konzept wird als **virtuelles LAN (VLAN)**, *Virtual LAN* bezeichnet. Es wurde vom IEEE-802-Komitee standardisiert und wird derzeit in vielen Unternehmen eingesetzt. Sehen wir es uns einmal genauer an. Weitere Informationen über VLANs finden Sie in Seifert und Edwards (2008).

VLANs basieren auf Switches, die VLANs erkennen können. Um ein VLAN-basiertes Netz einzurichten, muss der Netzadministrator festlegen, wie viele VLANs es geben wird, welcher Computer zu dem VLAN gehören soll und wie die VLANs heißen sollen. Oftmals werden die VLANs (informell) nach Farben benannt, da man sie dann in Übersichten in Farbe ausgeben kann, auf denen die Mitglieder des roten LAN rot, die des grünen LAN grün usw. eingezeichnet sind. Auf diese Weise sind physikalisches und logisches Layout in einer Ansicht vorhanden.

Als Beispiel betrachten wir die vier Bridge-LANs aus ► Abbildung 4.47, wo neun der Rechner zu dem (grauen) VLAN G und fünf zum (weißen) VLAN W gehören. Die Maschinen des grauen VLAN verteilen sich auf die zwei Switches, wobei zwei Rechner mit einem Switch über einen Hub verbunden sind.

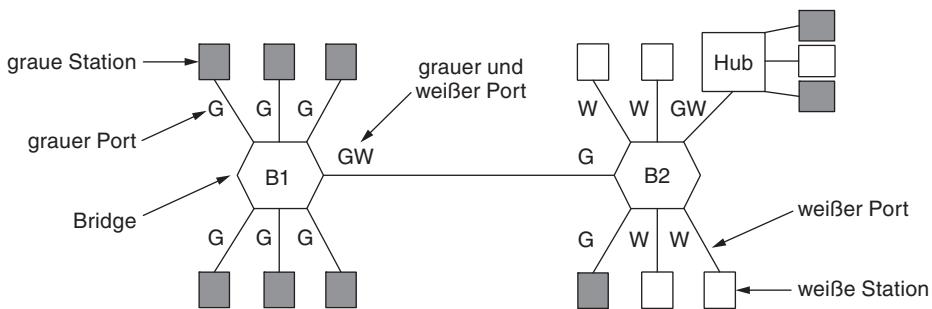


Abbildung 4.47: Zwei VLANS (grau und weiß) in einem Bridge-LAN.

Damit die VLANs korrekt funktionieren, müssen Konfigurationstabellen auf den Bridges eingerichtet werden. Diese Tabellen geben an, auf welche VLANs über welche Ports zugegriffen werden kann. Wenn ein Rahmen beispielsweise vom grauen VLAN eingeht, muss dieser an alle Ports, die mit G markiert sind, weitergeleitet werden. Dies gilt für normalen (Unicast-)Datenverkehr, bei dem die Bridges die Position des Ziels nicht lernen müssen, wie auch für Multicast- und Broadcast-Datenverkehr. Beachten Sie, dass ein Port mit mehreren VLAN-Farben gekennzeichnet sein kann.

Nehmen wir beispielsweise an, dass eine der grauen Stationen an Bridge *B1* in Abbildung 4.47 einen Rahmen an ein Ziel sendet, das vorab nicht beobachtet wurde. Bridge *B1* wird den Rahmen empfangen und sehen, dass er vom Rechner im grauen VLAN stammt, daher wird *B1* den Rahmen durch Fluten auf alle Ports mit der Bezeichnung G (mit Ausnahme des Eingangsports) weiterleiten. Der Rahmen wird zu den fünf anderen

grauen Stationen, die an B_1 angeschlossen sind, gesendet und ebenso über die Verbindung von B_1 zu Bridge B_2 . Bei Bridge B_2 wird der Rahmen ebenso auf alle anderen Ports weitergeleitet, die mit G bezeichnet sind. In diesem Fall wird der Rahmen also an eine weitere Station und den Hub gesendet (welcher den Rahmen zu allen seinen Stationen übermittelt). Der Hub ist sowohl mit G als auch mit W markiert, weil er mit Maschinen von beiden VLANs verbunden ist. Der Rahmen wird nicht an Ports gesandt, die nicht mit G markiert sind, weil die Bridge weiß, dass es keine weiteren Maschinen im grauen VLAN gibt, die über diese Ports erreicht werden können.

In unserem Beispiel wird der Rahmen nur deshalb von Bridge B_1 zu Bridge B_2 gesendet, weil es Maschinen im grauen VLAN gibt, die mit B_2 verbunden sind. Wenn wir das weiße VLAN betrachten, können wir sehen, dass der Bridge-Port von B_2 , der mit Bridge B_1 verbunden ist, *nicht* mit W bezeichnet ist. Das bedeutet, dass ein Rahmen im weißen VLAN nicht von Bridge B_2 zu Bridge B_1 weitergeleitet wird. Dieses Verhalten ist korrekt, weil keine Stationen des weißen VLAN mit B_1 verbunden sind.

Der IEEE-802.1Q-Standard

Um dieses Schema zu implementieren, müssen Bridges wissen, zu welchem VLAN der ankommende Rahmen gehört. Ohne diese Information kann beispielsweise Bridge B_2 in Abbildung 4.47 nicht wissen, wenn ein Rahmen von Bridge B_1 ankommt, ob dieser Rahmen an das graue oder das weiße VLAN weitergeleitet werden soll. Wenn man eine neue Art von LAN entwirft, könnte man ganz einfach ein VLAN-Feld im Header aufnehmen. Doch wie sieht es beim vorherrschenden LAN, dem Ethernet, aus? Hier liegen schließlich keine freie Felder mehr für eine VLAN-Kennung herum.

Das IEEE-802-Komitee bekam dieses Problem 1995 aufgetischt. Nach langen Diskussionen geschah das Undenkbare und der Ethernet-Header wurde geändert. Das neue Format wurde 1998 als IEEE-Standard **802.1Q** veröffentlicht. Das neue Format enthält ein VLAN-Tag, den wir nun kurz untersuchen wollen. Natürlich ist eine Änderung bei einer so etablierten Sache wie dem Ethernet-Header nicht ganz leicht zu bewerkstelligen. Hier stellt man sich sofort ein paar Fragen:

- 1.** Müssen hundert Millionen vorhandener Ethernet-Karten weggeworfen werden?
- 2.** Wenn nicht, wer erzeugt die neuen Felder?
- 3.** Was passiert mit Rahmen, die bereits die maximale Größe haben?

Natürlich waren dem IEEE-802-Komitee diese Probleme (nur zu schmerhaft) bewusst, ebenso wie die Tatsache, dass man dafür Lösungen anbieten musste – was man auch tat.

Der Schlüssel zur Lösung liegt in der Erkenntnis, dass die VLAN-Felder derzeit nur von Bridges und Switches, *nicht* aber von den Rechnern der Anwender genutzt werden. Daher ist es in Abbildung 4.47 nicht wirklich entscheidend, dass sie auf den Leitungen, die zu den Endstationen hinausgehen, vorhanden sind, solange sie sich auf der Leitung zwischen den Bridges befinden. Außerdem müssen die Bridges VLANs erkennen, um VLANs zu verwenden. Dadurch wird der Entwurf praktikabel.

Bei der Frage, ob man alle vorhandenen Ethernet-Karten wegwerfen muss, lautet die Antwort: nein. Wie Sie wissen, konnte das IEEE-802.3-Komitee die Leute nicht einmal überzeugen, das *Type*-Feld in ein *Length*-Feld zu ändern. Sie können sich den Aufschrei vorstellen, wenn alle vorhandenen Ethernet-Karten weggeworfen werden müssten. Neue Ethernet-Karten sind jedoch kompatibel mit IEEE 802.1Q und können die VLAN-Felder richtig füllen.

Da es Rechner (und Switches) geben kann, die nicht VLAN-fähig sind, fügt die erste VLAN-fähige Bridge, die den Rahmen bekommt, VLAN-Felder hinzu und die letzte Bridge in der Reihe entfernt diese wieder. Ein Beispiel einer gemischten Topologie ist in ►Abbildung 4.48 zu sehen. In dieser Abbildung erzeugen VLAN-fähige Rechner direkt Rahmen mit Tags (also IEEE-802.1Q-Rahmen), die bei der weiteren Vermittlung zum Einsatz kommen. Die dunkelblauen Symbole sind VLAN-fähig und die hellblauen Symbole sind es nicht.

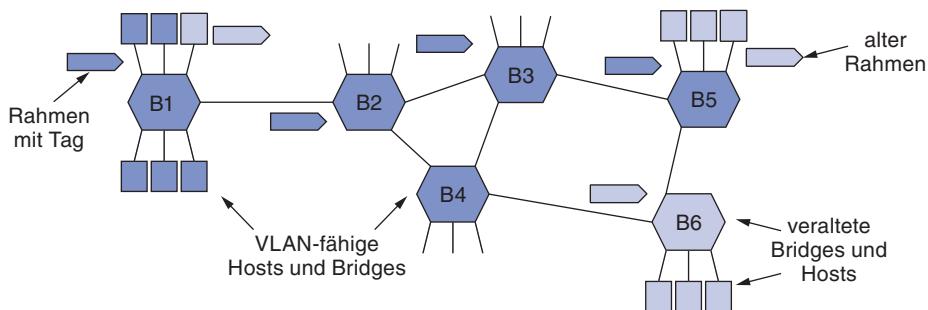


Abbildung 4.48: LAN mit Bridges, das nur teilweise VLAN-fähig ist. Die dunkelblauen Symbole sind VLAN-fähig, die hellblauen nicht.

Bei IEEE 802.1Q werden Rahmen farblich markiert, abhängig vom Port, auf dem sie empfangen werden. Damit diese Methode funktioniert, müssen alle Maschinen an einem Port zum selben VLAN gehören, was die Flexibilität verringert. In Abbildung 4.48 beispielsweise trifft diese Eigenschaft für alle Ports zu, bei denen ein einzelner Computer mit einer Bridge verbunden ist, aber nicht für die Ports, bei denen der Hub mit Bridge B2 verbunden ist.

Zusätzlich kann die Bridge zur Auswahl der Farbe das Protokoll höherer Schichten verwenden. Dann kann es sein, dass die am Port ankommende Rahmen zu verschiedenen VLANs gehören, je nachdem, ob sie IP-Pakete oder PPP-Rahmen übertragen.

Es sind auch noch andere Methoden möglich, doch diese werden nicht von IEEE 802.1Q unterstützt. Beispielsweise kann die MAC-Adresse zur Auswahl der VLAN-Farbe benutzt werden. Das könnte für Rahmen sinnvoll sein, die von einem in der Nähe befindlichen IEEE-802.11-LAN kommen, bei dem Laptops Rahmen über verschiedene Ports senden, wenn sie sich bewegen. Eine MAC-Adresse würde dann einem festen LAN zugeordnet, unabhängig davon, auf welchem Port es das LAN betritt.

Zum Problem der Rahmen, die länger als 1 518 Byte sind: IEEE 802.1Q hat gerade die Obergrenze auf 1 522 Byte erhöht. Zum Glück müssen nur VLAN-fähige Rechner und Switches diese längeren Rahmen unterstützen.

Sehen wir uns das IEEE-802.1Q-Rahmenformat einmal genauer an. Es ist in ▶ Abbildung 4.49 dargestellt. Die einzige Änderung ist das neu hinzugekommene Paar von 2-Byte-Feldern. Das erste ist die VLAN-Protokollkennung (*VLAN Protocol ID*). Es hat immer den Wert 0x8 100. Da die Zahl über 1 500 liegt, interpretieren sie alle Ethernet-Karten als Typ und nicht als Länge. Was eine alte Karte mit einem solchen Rahmen macht, ist egal, da solche Rahmen nicht an alte Karten gesendet werden.

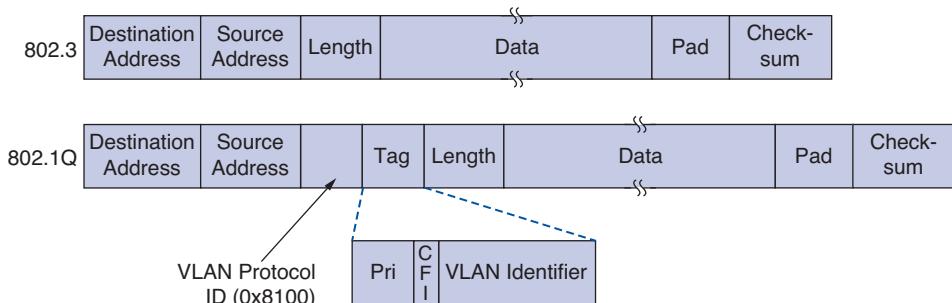


Abbildung 4.49: Die IEEE-802.3- (alt) und IEEE-802.1Q-Ethernet-Rahmenformate.

Das zweite 2-Byte-Feld enthält drei Unterfelder. Das wichtigste Feld ist die VLAN-Kennung (*VLAN Identifier*), die über die 12 hinteren Bits geht. Das ist genau das, worum es geht – die Farbe des VLANs, zu dem der Rahmen gehört. Das 3-Bit-Feld *Priority* hat nichts mit VLANs zu tun, doch da die Änderung des Ethernet-Headers ein Ereignis ist, das nur einmal im Jahrzehnt stattfindet, drei Jahre dauert und hundert Leute beschäftigt – warum dann nicht gleich ein paar weitere gute Dinge einbauen, wenn man einmal dabei ist? Dieses Feld ermöglicht es, zwischen „hartem“ Echtzeit-Datenverkehr, „weichem“ Echtzeit-Datenverkehr und nicht zeitkritischem Datenverkehr zu unterscheiden, um im Ethernet eine bessere Dienstgüte bereitzustellen. Es wird für Sprachübertragung über das Ethernet verwendet (obwohl der Fairness halber gesagt werden sollte, dass IP ein ähnliches Feld seit einem Vierteljahrhundert hat und niemand es jemals benutzt).

Das letzte Feld, *CFI* (*Canonical Format Indicator*, Kennzeichner für das Standardformat), sollte eigentlich *CEI* (*Corporate Ego Indicator*, Kennzeichner für das Unternehmens-Ego) heißen. Es war ursprünglich dazu bestimmt, die Reihenfolge der Bits in den MAC-Adressen anzugeben (also Little-Endian- von Big-Endian-MAC-Adressen zu unterscheiden), doch diese Verwendung ging im Laufe der Diskussionen unter. Seine Gegenwart gibt an, dass die Nutzdaten einen gefriergetrockneten IEEE-802.5-Rahmen enthalten, der hofft, am anderen Ende ebenfalls ein IEEE-802.5-LAN vorzufinden, zwischenzeitlich aber über Ethernet transportiert wird. Diese ganze Anordnung hat natürlich so rein gar nichts mit VLANs zu tun. Aber manchmal ähnelt die Politik des Standardisierungsgremiums der echten Politik: Wenn du für mein Bit stimmst, stimme ich auch für deins.

Wie bereits erwähnt, wenn ein Rahmen mit Tags an einem VLAN-fähigen Switch ankommt, verwendet der Switch die VLAN-Kennung als Index in einer Tabelle, um herauszufinden, an welchen Port das Ganze gehen soll. Woher kommt aber die Tabelle? Wenn sie von Hand erstellt wird, sind wir wieder am Anfang: bei der manuellen Konfiguration von Bridges. Das Wundervolle an der transparenten Bridge ist jedoch, dass sie Plug-and-Play-fähig ist und eben keine manuelle Konfiguration benötigt. Glücklicherweise können sich VLAN-fähige Bridges auch anhand der vorkommenden Tags selbst konfigurieren. Kommt ein als VLAN 4 gekennzeichneter Rahmen bei Port 3 an, dann ist anscheinend ein Rechner auf Port 3 auf VLAN 4. Der IEEE-802.1Q-Standard erklärt, wie die Tabellen dynamisch erstellt werden, hauptsächlich indem auf die entsprechenden Stellen im IEEE-802.1D-Standard verwiesen wird.

Bevor wir VLAN-Routing verlassen, hier noch eine letzte Beobachtung. Viele Leute in der Internet- und Ethernet-Welt sind fanatische Anhänger von verbindungslosen Netzen und wehren sich wie wild gegen alles, was nach Verbindungen auf der Sicherungs- oder Vermittlungsschicht aussieht. Aber VLANs führen etwas erstaunlich Ähnliches wie eine Verbindung ein. Um VLANs richtig zu verwenden, trägt jeder Rahmen eine neue spezielle Kennung, mit der eine Tabelle im Switch indiziert wird, um das Ziel des Rahmens nachzuschlagen. Das Gleiche geschieht auch in verbindungsorientierten Netzen. Bei verbindungslosen Netzen wird die Zieladresse zum Weiterleiten verwendet, nicht eine Verbindungskennung. Wir befassen uns hiermit ausführlicher in *Kapitel 5*.

Zusammenfassung

In einigen Netzen steht für die gesamte Kommunikation ein einziger Kanal zur Verfügung. Das Hauptproblem in diesen Netzen ist die Zuteilung des Kanals unter allen angeschlossenen bzw. sendewilligen Stationen. FDM und TDM sind einfache, effiziente **Zuordnungsmethoden**, wenn die Anzahl der Stationen klein und festgelegt und der Datenverkehr konstant ist. Beide Methoden werden unter entsprechenden Bedingungen angewendet, z. B. zur Aufteilung der Bandbreite von Satellitenverbindungen für Telefongespräche. Ist jedoch die Anzahl der Stationen groß und variabel oder ist der Datenverkehr ziemlich unregelmäßig – der Normalfall in Rechnernetzen –, dann ist man mit FDM und TDM schlecht beraten.

Es wurden zahlreiche dynamische **Kanalzuordnungsalgorithmen** entwickelt. Das ALOHA-Protokoll mit oder ohne Unterteilung in Zeitscheiben wird in vielen Varianten in realen Systemen eingesetzt, zum Beispiel bei Kabelmodems und bei RFID. Falls die Möglichkeit besteht, den Zustand des Kanals abzufragen, kann das Verfahren verbessert werden, indem während einer Übertragung durch eine Station die anderen Stationen davon absehen, ebenfalls eine Übertragung zu beginnen. Diese Technik – Trägerprüfung – hat zu einer Vielzahl von CSMA-Protokollen für LANs und MANs geführt. Sie ist die Grundlage für klassisches Ethernet und IEEE-802.11-Netze.

Es gibt eine Klasse von **Protokollen**, die Konkurrenz eliminiert oder sie zumindest erheblich reduziert. Beim Bitmusterprotokoll, bei bestimmten Topologien wie Ringen, und beim binären Countdown-Protokoll gibt es überhaupt keine Konkurrenz. Beim Tree-Walk-Protokoll wird die Konkurrenz dadurch reduziert, dass die Stationen dynamisch in zwei getrennte Gruppen unterschiedlicher Größe unterteilt werden und Konkurrenz jeweils nur innerhalb einer Gruppe zugelassen wird. Idealerweise wird die Gruppe so ausgewählt, dass nur eine einzige Station sendebereit ist, wenn die Erlaubnis dazu erteilt wird.

Bei **drahtlosen LANs** gibt es das zusätzliche Problem, dass es schwierig ist, die Kollision von Übertragungen zu erkennen, und dass die Abdeckungsbereiche der Stationen unterschiedlich sein können. Im vorherrschenden drahtlosen LAN, IEEE 802.11, benutzen Stationen CSM/CA, um das erste Problem zu entschärfen, indem kleine Lücken gelassen werden, um Kollisionen zu vermeiden. Die Stationen können auch das RTS/CTS-Protokoll verwenden, um das Hidden-Terminal-Phänomen zu bekämpfen, welches aufgrund des zweiten Problems auftaucht. IEEE 802.11 wird häufig benutzt, um Laptops und andere Geräte mit drahtlosen Zugangspunkten zu verbinden, aber es kann auch zwischen Geräten eingesetzt werden. Es können verschiedene Bitübertragungsschichten verwendet werden, einschließlich Mehrfachkanal-FDM mit und ohne mehrere Antennen, und Frequenzspezzierung.

RFID-Lesegeräte und -Tags benutzen wie IEEE 802.11 ein zufallsgesteuertes Protokoll, um Identifikatoren zu übermitteln. Andere drahtlose PANs und MANs haben unterschiedliche Designs. Das **Bluetooth-System** verbindet Headsets und viele Arten von Peripheriegeräten mit einem Computer ohne Kabel. IEEE 802.16 stellt eine große Bandbreite von drahtlosen Internetdatendiensten für stationäre und mobile Rechner zur Verfügung. Beide Netzwerke benutzen einen zentralisierten, verbindungsorientierten Entwurf, in dem der Bluetooth-Master und die WiMAX-Basisstation entscheiden, wann eine Station Daten senden oder empfangen kann. Für IEEE 802.16 unterstützt dieser Entwurf unterschiedliche Dienstgüte für Echtzeitdatenverkehr wie Telefongespräche und interaktiven Datenverkehr wie Webbrowsern. Das Verlegen der Komplexität bei Bluetooth in den Master führt zu billigen Slave-Geräten.

Ethernet ist bei verkabelten Netzen vorherrschend. Klassisches Ethernet verwendet CSMA/CD für die Kanalzuordnung auf einem gelben Kabel der Größe eines Gartenschlauchs, das sich von Rechner zu Rechner schlängelte. Die Architektur hat sich gewandelt und die Geschwindigkeiten sind von 10 Mbit/s auf 10 Gbit/s gestiegen und steigen weiterhin. Heute werden Punkt-zu-Punkt-Verbindungen wie Twisted-Pair-Kabel an Hubs und Switches angeschlossen. Mit modernen Switches und Voll duplexverbindungen gibt es keine Konkurrenz auf den Leitungen und die Switches können Rahmen zwischen verschiedenen Ports parallel weiterleiten.

Bei Gebäuden voll mit LANs muss es eine Möglichkeit geben, sie alle zu verbinden. Zu diesem Zweck werden **Plug-and-Play-Bridges** verwendet. Die Bridges werden mit einem Backward-Learning-Algorithmus und einem Spannbaum-Algorithmus aufgebaut. Da diese Funktionalität auch in moderne Switches eingebaut ist, werden die Begriffe „Bridge“ und „Switch“ synonym verwendet. Um die Verwaltung von LANs, die per Bridge gekoppelt sind, zu unterstützen, kann man mithilfe von VLANs die physische Topologie in unterschiedliche logische Topologien einteilen. Der VLAN-Standard, IEEE 802.1Q, führt ein neues Format für Ethernet-Rahmen ein.



Übungsaufgaben

- 1** Verwenden Sie für diese Übung eine Formel aus diesem Kapitel. Geben Sie aber zuerst die Formel an. Die Rahmen kommen bei einem Kanal mit 100 Mbit/s zufällig zur Übertragung an. Wenn der Kanal belegt ist und ein Rahmen ankommt, wartet dieser in einer Warteschlange. Die Länge des Rahmens ist exponentiell verteilt, wobei der Durchschnitt bei 10 000 Bit/Rahmen liegt. Geben Sie für jede der folgenden Rahmenankommensraten die für einen durchschnittlichen Rahmen anfallende Verzögerung einschließlich der Zeit in der Warteschlange und der Übertragungszeit an.
 - a. 90 Rahmen/Sekunde
 - b. 900 Rahmen/Sekunde
 - c. 9 000 Rahmen/Sekunde
- 2** Eine Gruppe von N Stationen teilt sich einen reinen ALOHA-Kanal mit 56 kbit/s. Jede Station gibt im Durchschnitt alle 100 Sekunden einen 1 000-Bit-Rahmen aus, auch wenn der vorherige noch nicht vollständig übertragen wurde (z.B. puffern die Stationen ausgehende Rahmen). Wie lautet der maximale Wert für N ?
- 3** Stellen Sie die Verzögerung von reinem ALOHA der von S-ALOHA bei niedriger Auslastung vergleichend gegenüber. Welche ist geringer? Erklären Sie Ihre Antwort.
- 4** Eine große Anzahl an ALOHA-Benutzern generiert 50 Anfragen/Sekunde, sowohl Originale als auch erneute Übertragungen. Die Zeitscheiben sind 40 ms lang.
 - a. Wie hoch ist die Erfolgschance beim ersten Versuch?
 - b. Wie hoch ist die Wahrscheinlichkeit, dass genau k Kollisionen auftreten und die Übertragung dann erfolgreich ist?
 - c. Was ist die erwartete Anzahl an erforderlichen Übertragungsversuchen?

- 5** In einem S-ALOHA-System mit einer unendlichen Menge von Benutzern liegt die mittlere Anzahl von Zeitscheiben, die eine Station zwischen einer Kollision und einer Übertragungswiederholung wartet, bei 4. Stellen Sie die Kurve für die Verzögerung in Abhängigkeit vom Durchsatz für dieses System dar.
- 6** Wie lang ist eine Konkurrenzscheibe bei CSMA/CD für
 - ein 2 km langes Paralleldrahtkabel (Signalausbreitungsgeschwindigkeit beträgt 82 % der Geschwindigkeit im Vakuum) und
 - ein 40 km langes Multimode-Glasfaserkabel (Signalausbreitungsgeschwindigkeit beträgt 65 % der Geschwindigkeit im Vakuum)?
- 7** Wie lange muss eine Station s im schlechtesten Fall warten, bis sie die Übertragung eines Rahmens über ein LAN beginnen kann, welches das Bitmusterprotokoll verwendet?
- 8** Erklären Sie, wie beim binären Countdown-Protokoll eine Station mit niedriger Nummer vom Senden eines Pakets abgehalten wird („verhungert“).
- 9** 16 Stationen (1 bis 16) konkurrieren um die Benutzung eines gemeinsamen Kanals unter Verwendung des Adaptive-Tree-Walk-Protokolls. Wie viele Zeitscheiben werden zur Auflösung der Konkurrenz benötigt, wenn alle Stationen, deren Adressen Primzahlen sind, auf einmal sendebereit sind?
- 10** Nehmen wir an, es gibt fünf drahtlose Stationen, A , B , C , D und E . Station A kann mit allen anderen Stationen kommunizieren. B kann mit A , C und E kommunizieren. C kann mit A , B und D kommunizieren. D kann mit A , C , und kommunizieren. E kann mit A , D und B kommunizieren.
 - Wenn A an B sendet, welche anderen Kommunikationen sind möglich?
 - Wenn B an A sendet, welche anderen Kommunikationen sind möglich?
 - Wenn B an C sendet, welche anderen Kommunikationen sind möglich?
- 11** Die sechs Stationen A bis F kommunizieren über das MACA-Protokoll. Können hier zwei Übertragungen gleichzeitig stattfinden? Erläutern Sie Ihre Antwort.
- 12** Ein siebenstöckiges Bürogebäude hat pro Stockwerk 15 aneinandergrenzende Büros. Jedes Büro ist in der vorderen Wand mit einem Wandstecker für ein Terminal ausgerüstet, sodass die Stecker in der Vertikalen ein rechteckiges Gitter mit einem horizontalen und vertikalen Abstand von je 4 m zwischen den Steckern bilden. Vorausgesetzt, dass jede direkte horizontale, vertikale oder diagonale Kabelverbindung zwischen zwei beliebigen Steckern machbar ist, wie viele Meter Kabel sind dann zur Verbindung der Stecker notwendig, wenn
 - eine Sternkonfiguration mit einem Router in der Mitte verwendet wird?
 - ein klassisches IEEE-802.3-LAN verwendet wird?
- 13** Wie hoch ist die Baudate des klassischen Ethernets mit 10 Mbit/s?
- 14** Skizzieren Sie die Manchester-Codierung in einem klassischen Ethernet für den Bitstrom 0001110101.

- 15** Die Ausbreitungsgeschwindigkeit in einem 1 km langen CSMA/CD-LAN (nicht IEEE 802.3) mit 10 Mbit/s beträgt 200 m/ μ s. Repeater sind nicht erlaubt. Die Datenrahmen sind 256 Bit lang, einschließlich 32 Bit für Header, Prüfsumme und anderen Overhead. Die erste Zeitscheibe nach jeder erfolgreichen Übertragung ist für den Empfänger reserviert, der dann den Kanal belegt und einen 32-Bit-Bestätigungsrahmen sendet. Wie groß ist die effektive Datenübertragungsrate ohne den Overhead, unter der Annahme, dass keine Kollisionen auftreten?
- 16** Zwei CSMA/CD-Stationen versuchen jeweils, lange Dateien (Mehrfrachrahmen) zu übertragen. Nach der Übertragung eines Rahmens konkurrieren sie über den binären exponentiellen Backoff-Algorithmus um die Benutzung des Kanals. Wie hoch ist die Wahrscheinlichkeit, dass die Konkurrenz in Runde k endet, und wie hoch ist die mittlere Anzahl der Runden in jeder Konkurrenzperiode?
- 17** Ein IP-Paket, das über Ethernet übertragen werden soll, ist 60 Byte lang, einschließlich aller Header. Wird LLC nicht verwendet, muss der Ethernet-Rahmen aufgefüllt werden? Wenn ja, mit vielen Byte?
- 18** Ethernet-Rahmen müssen mindestens 64 Byte lang sein, um sicherzustellen, dass der Sender im Fall einer Kollision am entfernten Ende des Kabels noch läuft. Fast Ethernet hat die gleiche Mindestrahmengröße, kann aber die Bits zehnmal schneller befördern. Wie ist es möglich, die gleiche minimale Rahmengröße einzuhalten?
- 19** Einige Bücher beziffern die maximale Größe eines Ethernet-Rahmens mit 1 522 Byte anstatt 1 500 Byte. Ist dies falsch? Erklären Sie Ihre Antwort.
- 20** Wie viele Rahmen pro Sekunde kann Gigabit-Ethernet verwalten? Denken Sie sorgfältig darüber nach und berücksichtigen Sie alle relevanten Fälle. *Tipp:* Wichtig ist, dass es sich um ein *Gigabit*-Ethernet handelt.
- 21** Nennen Sie die beiden Netze, in denen Rahmen direkt hintereinander gesendet werden können. Warum ist diese Funktion nützlich?
- 22** In Abbildung 4.27 werden die vier Stationen A , B , C und D gezeigt. Welche der letzten beiden Stationen liegt am nächsten bei A und warum?
- 23** Geben Sie ein Beispiel an, um zu zeigen, dass das RTS/CTS im IEEE-802.11-Protokoll ein wenig anders als im MACA-Protokoll ist.
- 24** Ein drahtloses LAN mit einem Zugangspunkt hat zehn Client-Stationen. Vier Stationen haben Datenraten von 6 Mbit/s, vier Stationen haben Datenraten von 18 Mbit/s und die letzten beiden Stationen haben Datenraten von 54 Mbit/s. Wie hoch ist die Datenrate, die jede Station erfährt, wenn alle zehn Stationen zusammen Daten senden und
- TXOP nicht benutzt wird?
 - TXOP benutzt wird?
- 25** Angenommen, ein IEEE-802.11b-LAN mit 11 Mbit/s überträgt 64-Byte-Rahmen direkt hintereinander über einen Funkkanal mit einer Bitfehlerrate von 10^{-7} . Wie viele Rahmen pro Sekunde werden im Durchschnitt beschädigt?

- 26** Ein IEEE-802.16-Netz hat eine Kanalbreite von 20 MHz. Wie viele Bit/s können an eine Teilnehmerstation gesendet werden?
- 27** Geben Sie zwei Gründe an, wann Netze einen Fehlerkorrekturcode anstatt der Fehlererkennung und erneuten Übertragung verwenden sollten.
- 28** Geben Sie zwei Punkte an, in denen sich WiMAX und IEEE 802.11 ähneln, und zwei Punkte, in denen sich WiMAX von IEEE 802.11 unterscheidet.
- 29** In Abbildung 4.34 sehen wir, dass ein Bluetooth-Gerät gleichzeitig in zwei Piconetzen aktiv sein kann. Gibt es einen Grund dafür, dass ein Gerät nicht gleichzeitig der Master von beiden sein kann?
- 30** Welches ist die maximale Größe eines Datenfelds für einen 3 Zeitscheiben langen Bluetooth-Rahmen mit Basisdatenraten. Erläutern Sie Ihre Antwort.
- 31** Abbildung 4.24 listet verschiedene Protokolle der Bitübertragungsschicht auf. Welches entspricht dem Bitübertragungsschichtprotokoll von Bluetooth am besten? Was ist der hauptsächliche Unterschied zwischen beiden?
- 32** In Abschnitt 4.6.6 wird erwähnt, dass die Effizienz eines 1 Zeitscheiben langen Rahmens mit Codierung der Wiederholungen ungefähr 13 % der Basisdatenrate entspricht. Wie groß wird die Effizienz sein, wenn stattdessen ein 5 Zeitscheiben langer Rahmen mit Codierung der Wiederholungen zur Basisdatenrate verwendet wird?
- 33** Beacon-Rahmen enthalten in der FHSS-Variante von IEEE 802.11 eine Verweilzeit. Glauben Sie, dass die analogen Beacon-Rahmen in Bluetooth auch eine Verweilzeit enthalten? Erläutern Sie Ihre Antwort.
- 34** Nehmen Sie an, es befinden sich 10 RFID-Tags in der Nähe eines RFID-Lesegeräts. Welches ist der beste Wert für Q? Wie wahrscheinlich ist es, dass ein Tag kollisionsfrei in einer bestimmten Zeitscheibe antwortet?
- 35** Zählen Sie einige Sicherheitsbedenken eines RFID-Systems auf.
- 36** Ein Switch für Fast Ethernet hat eine Platine, die 10 Gbit/s verarbeiten kann. Wie viele Rahmen/Sekunde kann sie im schlechtesten Fall verwalten?
- 37** Beschreiben Sie kurz den Unterschied zwischen Store-and-forward- und Cut-through-Switches.
- 38** Betrachten Sie das in Abbildung 4.41b dargestellte LAN, das mithilfe der Bridges $B1$ und $B2$ verbunden ist. Angenommen, die Hash-Tabellen der beiden Bridges sind anfangs leer. Geben Sie alle Ports an, zu denen ein Paket für die folgende Datenübertragungsfolge geleitet wird:
- A sendet ein Paket an C .
 - E sendet ein Paket an F .
 - F sendet ein Paket an E .
 - G sendet ein Paket an E .
 - D sendet ein Paket an A .
 - B sendet ein Paket an F .

- 39** Store-and-forward-Switches haben gegenüber Cut-through-Switches einen Vorteil bei beschädigten Rahmen. Erklären Sie dies.
- 40** In Abschnitt 4.8.3 wird erwähnt, dass einige Bridges eventuell im Spannbaum gar nicht vorkommen. Skizzieren Sie ein Szenario, in dem eine Bridge nicht im Spannbaum erscheint.
- 41** Damit VLANs korrekt arbeiten, sind in den Bridges Konfigurationstabellen erforderlich. Was passiert, wenn die VLANs in Abbildung 4.47 Hubs anstatt Switches verwenden? Benötigen Hubs auch Konfigurationstabellen? Warum bzw. warum nicht?
- 42** In Abbildung 4.48 ist der Switch am Altsystem-Ende der Domäne rechts ein Switch, der VLANs erkennt. Könnte man hier auch einen alten konventionellen Switch verwenden? Falls ja, wie funktioniert dies? Wenn nicht, warum nicht?
- 43** Schreiben Sie ein Programm, um das Verhalten des CSMA/CD-Protokolls über Ethernet zu simulieren, wenn N Stationen sendebereit sind, während ein Rahmen übertragen wird. Ihr Programm soll die Zeitpunkte ausgeben, zu denen jede Station erfolgreich beginnt, ihren Rahmen zu senden. Gehen Sie davon aus, dass ein Taktgeber pro Zeitscheibe ($51,2 \mu\text{s}$) einmal tickt und die Kollisionsentdeckung und das Senden eines Rausch-Bursts eine Zeitscheibe dauert. Alle Rahmen haben die maximal zulässige Länge.

5

ÜBERBLICK

Die Vermittlungsschicht

| | |
|---|-----|
| 5.1 Entwurfsaspekte der Vermittlungsschicht | 413 |
| 5.2 Routing-Algorithmen | 420 |
| 5.3 Algorithmen zur Überlastungsüberwachung | 452 |
| 5.4 Dienstgüte | 465 |
| 5.5 Internetworking | 487 |
| 5.6 Vermittlungsschicht im Internet | 500 |

» Die Vermittlungsschicht hat die Aufgabe, Pakete von der Quelle bis zum Ziel zu übertragen. Dazu kann auch das Durchqueren vieler Teilstrecken zwischen den auf dem Weg liegenden Routern gehören. Diese Aufgabe unterscheidet sich stark von der Funktion der Sicherungsschicht, die lediglich Rahmen von einem Ende der Leitung zum anderen Ende befördern muss. Die Vermittlungsschicht ist also die unterste Schicht, die mit der Übertragung von Endpunkt zu Endpunkt zu tun hat.

Um ihre Aufgabe zu bewältigen, muss die Vermittlungsschicht über die Topologie des Netzes (d.h. alle Router und Verbindungen) Bescheid wissen und auch in großen Netzen geeignete Pfade wählen. Sie muss auch die Routen sorgfältig wählen, um zu vermeiden, dass einige Übertragungsleitungen und Router überlastet sind, während sich andere im Leerlaufzustand befinden. Wenn sich dann die Quelle und das Ziel auch noch in verschiedenen Netzen befinden, treten neue Probleme auf. Damit muss sich dann die Vermittlungsschicht befassen. In diesem Kapitel untersuchen wir die oben aufgeführten Aspekte und veranschaulichen sie anhand des Internets und dessen Schicht-3-Protokoll IP (*Internet Protocol*). «

5.1 Entwurfsaspekte der Vermittlungsschicht

Die folgenden Abschnitte geben eine Einführung in einige der Aufgaben, mit denen sich die Entwickler der Vermittlungsschicht befassen müssen. Zu diesen Aufgaben zählen die Bereitstellung von Diensten für die Transportschicht und der interne Entwurf des Netzes.

5.1.1 Paketvermittlung unter Verwendung des Store-and-forward-Verfahrens

Bevor wir uns mit den Details der Vermittlungsschicht befassen, betrachten wir noch einmal den Kontext, in dem die Schicht-3-Protokolle arbeiten. Dieser Kontext ist in Abbildung 5.1 dargestellt. Die Hauptkomponenten des Netzes sind die Geräte des ISP (Router, die an Übertragungsleitungen angeschlossen sind), die sich in dem blau schattierten Oval befinden, sowie die Ausrüstung des Kunden, die sich außerhalb des Ovals befindet. Host H_1 ist direkt mit dem Router A des ISP verbunden, z.B. ein Heimanwender, der seinen Rechner an ein DSL-Modem angeschlossen hat. Dagegen befindet sich H_2 in einem LAN (z.B. ein Ethernet in einem Büro) mit einem Router F , der dem Kunden gehört und vom ihm betrieben wird. Der Router verfügt über eine Standleitung zu den ISP-Geräten. Wir haben den Router F außerhalb des Ovals gesetzt, weil er nicht zum ISP gehört. Für die Behandlung in diesem Kapitel werden wir jedoch die Router der Kunden als zum ISP-Netz gehörend betrachten, da sie die gleichen Algorithmen wie die ISP-Router ausführen (und unser Hauptaugenmerk liegt hier auf den Algorithmen).

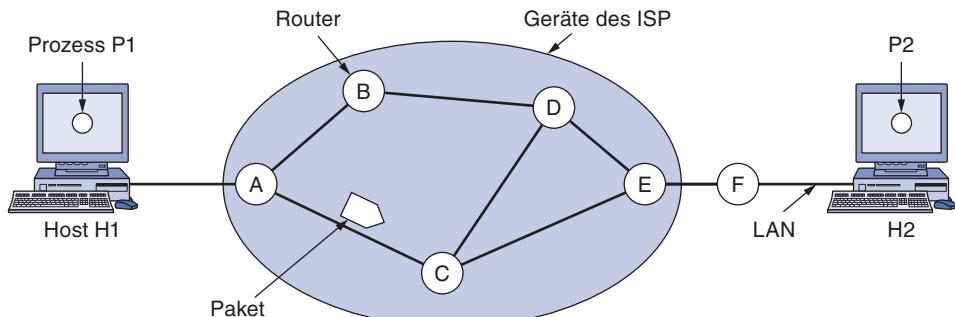


Abbildung 5.1: Die Umgebung des Schicht-3-Protokolls.

Diese Geräte werden wie folgt eingesetzt. Ein Host mit einem zu sendenden Paket überträgt dieses zum nächsten Router, entweder auf seinem eigenen LAN oder über eine Punkt-zu-Punkt-Verbindung zum ISP. Das Paket wird hier gespeichert, bis es vollständig angekommen ist, damit die Prüfsumme kontrolliert werden kann. Es wird dann auf dem Pfad zum nächsten Router weitergeleitet, bis es beim Ziel-Host ankommt und hier zugestellt wird. Dieser Mechanismus ist die Store-and-forward-Paketvermittlung, die wir bereits in den vorhergehenden Kapiteln kennengelernt haben.

5.1.2 Dienste für die Transportschicht

Die Vermittlungsschicht stellt Dienste für die Transportschicht an der Schnittstelle zwischen den beiden Schichten bereit. Eine wichtige Frage ist, welche Arten von Diensten die Vermittlungsschicht für die Transportschicht genau zur Verfügung stellt. Die Dienste müssen sorgfältig hinsichtlich der folgenden Ziele konzipiert werden:

1. Die Dienste sollten unabhängig von den Technologien sein, die bei den Routern verwendet werden.
2. Die Transportschicht muss vor der Anzahl, der Art und der Topologie der Router abgeschirmt werden.
3. Die für die Transportschicht zur Verfügung gestellten Netzadressen müssen ein einheitliches Nummerierungsschema darstellen, selbst zwischen LANs und WANs.

Mit diesen Vorgaben haben die Entwickler der Vermittlungsschicht sehr viel Freiraum bei der Erstellung genauer Dienstspezifikationen für die Transportschicht. Dieser Freiraum artet meist in eine wilde Schlacht zwischen zwei gegnerischen Parteien aus. Der wesentliche Streitpunkt besteht in der Frage, ob die Vermittlungsschicht verbindungsorientierte oder verbindungslose Dienste bereitstellen soll.

Ein Lager (vertreten durch die Internetgemeinde) argumentiert, dass der Router lediglich Pakete befördern und sonst nichts übernehmen soll. Aus ihrer Sicht (auf der Grundlage von 40 Jahren Erfahrung mit einem echten Rechnernetz) ist das Netz inhärent unzuverlässig, gleichgültig wie es entworfen wird. Deshalb sollten die Hosts diese Tatsache akzeptieren und Fehlerüberwachung (d.h. Fehlererkennung und -korrektur) sowie Flusskontrolle selbst übernehmen.

Dieser Standpunkt führt zu dem Schluss, dass der Dienst der Vermittlungsschicht verbindungslos sein sollte, mit nicht viel mehr als den Primitiven SEND PACKET und RECEIVE PACKET. Insbesondere sollten keine Fehlerüberwachung und keine Flusskontrolle stattfinden, da dies ohnehin von den Hosts vorgenommen wird. In der Regel wird nur wenig gewonnen, wenn man dies zweimal durchführt. Diese Schlussfolgerung ist ein Beispiel der **Ende-zu-Ende-Argumentation**, ein Entwurfsprinzip, das sehr einflussreich bei der Gestaltung des Internets war (Saltzer et al., 1984). Außerdem muss jedes Paket die volle Zieladresse enthalten, da es gänzlich unabhängig von seinen Vorgängern (falls vorhanden) übertragen wird.

Das andere Lager (verkörpert durch die Telefongesellschaften) argumentiert, dass das Netz einen zuverlässigen, verbindungsorientierten Dienst bieten soll. Sie weisen darauf hin, dass eine hundertjährige erfolgreiche Erfahrung mit dem weltweiten Telefonnetz ein hervorragender Wegweiser ist. Bei diesem Standpunkt ist die Dienstgüte der entscheidende Faktor. Ohne Verbindungsorientierung im Netz ist die Dienstgüte sehr schwer zu verwirklichen, insbesondere für Echtzeitdatenverkehr wie Sprache und Video.

Selbst nach mehreren Jahrzehnten ist dieser Streit immer noch im Gange. Frühe, weitverbreitete Datennetze, wie X.25 in den 1970er Jahren und sein Nachfolger Frame Relay in den 1980er Jahren, waren verbindungsorientiert. Jedoch sind seit den Tagen

von ARPANET und dem frühen Internet verbindungslose Vermittlungsschichten enorm in der Beliebtheit gestiegen. Das IP-Protokoll ist heute ein stets präsentes Symbol des Erfolgs. Dieser Erfolg ließ sich auch nicht durch eine verbindungslosen Technologie namens ATM aufhalten, die in den 1980ern Jahren entwickelt wurde, um IP vom Thron zu stoßen – im Gegenteil: heute findet man ATM nur in Nischenanwendungen und IP ist dabei, die Telefonnetze zu erobern. Mit steigender Bedeutung der Dienstgüte jedoch entwickelt das Internet unter der Oberfläche verbindungsorientierte Eigenschaften. Zwei Beispiele für verbindungsorientierter Technologien sind MPLS (*MultiProtocol Label Switching*), das wir später in diesem Kapitel beschreiben, und VLANs, die wir in *Kapitel 4* untersucht haben. Beide Technologien werden häufig eingesetzt.

5.1.3 Implementierung eines verbindungslosen Dienstes

Nachdem wir die beiden Dienstklassen behandelt haben, die die Vermittlungsschicht den Benutzern bereitstellt, befassen wir uns nun damit, wie diese Schicht intern arbeitet. Je nach angebotenem Diensttyp sind zwei verschiedene Strukturen möglich. Wird ein verbindungsloser Dienst angeboten, so werden Pakete einzeln in das Netz eingespeist und unabhängig voneinander weitergeleitet. Vorab ist keine Einrichtung erforderlich. In diesem Kontext werden die Pakete häufig als **Datagramme** (*datagram*; in Analogie zu Telegramm) bezeichnet und das Netz als **Datagrammnetz** (*datagram network*). Wird ein verbindungsorientierter Dienst eingesetzt, dann muss der gesamte Pfad vom Quell-Router zum Ziel-Router aufgebaut werden, bevor Datenpakete gesendet werden können. Diese Verbindung wird als **virtuelle Verbindung** (**VC**, *Virtual Circuit*) bezeichnet, analog zu den physikalischen Verbindungen (Leitungen), die das Telefon-System aufbaut. Das Netz wird dann als **VC-Netz** (*virtual circuit network*) bezeichnet. In diesem Abschnitt behandeln wir die Datagrammnetze, im nächsten Abschnitt gehen wir auf die VC-Netze ein.

Untersuchen wir nun einmal, wie ein Datagrammnetz arbeitet. Angenommen, der Prozess *P1* in ► Abbildung 5.2 hat eine lange Nachricht für *P2*. Er übergibt die Nachricht an die Transportschicht mit der Anweisung, sie Prozess *P2* auf Host *H2* zuzustellen. Der Transportschichtcode läuft auf *H1*, in der Regel im Betriebssystem. Er hängt an das vordere Ende der Nachricht einen Transport-Header an und übergibt das Ergebnis an die Vermittlungsschicht, die wahrscheinlich nur eine weitere Prozedur im Betriebssystem ist.

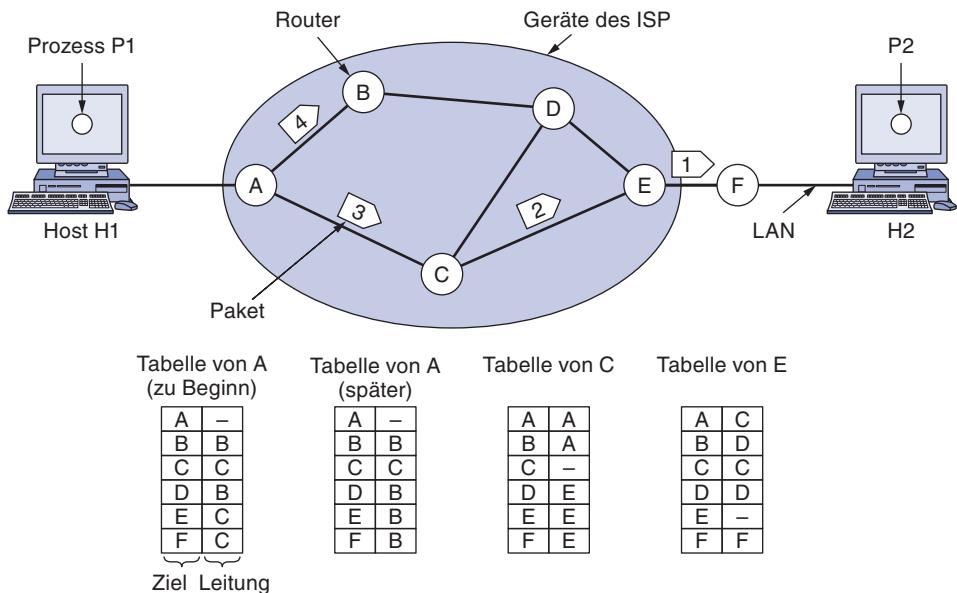


Abbildung 5.2: Routing in einem Datagrammnetz.

Wir wollen für dieses Beispiel annehmen, dass die Nachricht viermal länger als die maximale Paketgröße ist, sodass die Vermittlungsschicht sie in die vier Pakete 1, 2, 3 und 4 aufteilen muss und jedes Paket an den Router **A** sendet, wobei ein Punkt-zu-Punkt-Protokoll wie PPP verwendet wird. An diesem Punkt übernimmt der ISP die Weiterleitung. Jeder Router hat eine interne Tabelle, aus der er jedes mögliche Ziel zum Versenden der Pakete entnimmt. Jeder Tabelleneintrag ist ein Paar bestehend aus einem Ziel und der Ausgangsleitung, die für dieses Ziel verwendet werden soll. Es können nur direkte Verbindungsleitungen verwendet werden. So hat beispielsweise **A** in Abbildung 5.2 nur zwei Ausgangsleitungen – zu **B** und zu **C** –, sodass jedes eingehende Paket zu einem dieser Router gesendet werden muss, selbst wenn das endgültige Ziel sich auf einem anderen Router befindet. Die anfängliche Routing-Tabelle von **A** wird in der Abbildung unter der Überschrift „zu Beginn“ dargestellt.

Bei **A** werden die Pakete 1, 2 und 3 kurz gespeichert, nachdem sie auf der Eingangsleitung angekommen sind und ihre Prüfsummen verifiziert wurden. Dann werden die Pakete gemäß den Informationen in der Tabelle von **A** weitergeleitet, und zwar auf der Ausgangsleitung zu **C** innerhalb eines neuen Rahmens. Das Paket 1 wird dann an **E** weitergeleitet und anschließend an **F**. Bei **F** angekommen wird es in einem Rahmen über das LAN an **H2** gesendet. Die Pakete 2 und 3 nehmen den gleichen Weg.

Mit Paket 4 geschieht aber etwas anderes. Wenn es bei **A** ankommt, wird es an den Router **B** weitergeleitet, obwohl es eigentlich ebenfalls für **F** bestimmt war. Aus irgendeinem Grund hat **A** entschieden, das Paket 4 über einen anderen Weg als die ersten drei Pakete gehen soll. Vielleicht hat **A** von einem Datenstau auf dem Pfad **ACE** erfahren und die Routing-Tabelle aktualisiert. Dies wird unter der Überschrift „später“ aufgeführt. Der Algorithmus zur Verwaltung der Tabellen und der Routing-Entschei-

dungen wird als **Routing-Algorithmus** (Weiterleitungsalgorithmus) bezeichnet. Routing-Algorithmen gehören zu den Schwerpunkten in diesem Kapitel. Wie wir sehen werden, gibt es verschiedene Arten dieser Algorithmen.

5.1.4 Implementierung eines verbindungsorientierten Dienstes

Für einen verbindungsorientierten Dienst benötigen wir ein VC-Netz. Betrachten wir einmal, wie dies funktioniert. Bei virtuellen Verbindungen muss nicht wie in Abbildung 5.2 für jedes Paket oder jede Zelle ein neuer Weg gewählt werden. Vielmehr wird beim Aufbau einer Verbindung ein Weg zwischen der Quelle und dem Ziel als Teil der Verbindungseinrichtung gewählt, die dann in den Router-Tabellen gespeichert wird. Dieser Pfad wird für den gesamten Verkehr verwendet, der über die Verbindung fließt, wie beim Telefonnetz. Wird die Verbindung freigegeben, dann endet auch die virtuelle Verbindung. Bei einem verbindungsorientierten Dienst enthält jedes Paket einen Identifikator, der angibt, zu welcher virtuellen Verbindung es gehört.

Betrachten wir dazu als Beispiel die Situation in ►Abbildung 5.3. Hier errichtet der Host H_1 die Verbindung 1 zu Host H_2 . Diese Verbindung wird im ersten Eintrag der entsprechenden Routing-Tabellen notiert. Die erste Zeile in der Tabelle von A gibt an, dass ein von H_1 ankommendes Paket mit dem Verbindungsidentifikator 1 an den Router C gesendet wird und den Verbindungsidentifikator 1 enthält. Genauso leitet der erste Eintrag in C das Paket an E weiter, ebenfalls mit dem Verbindungsidentifikator 1.

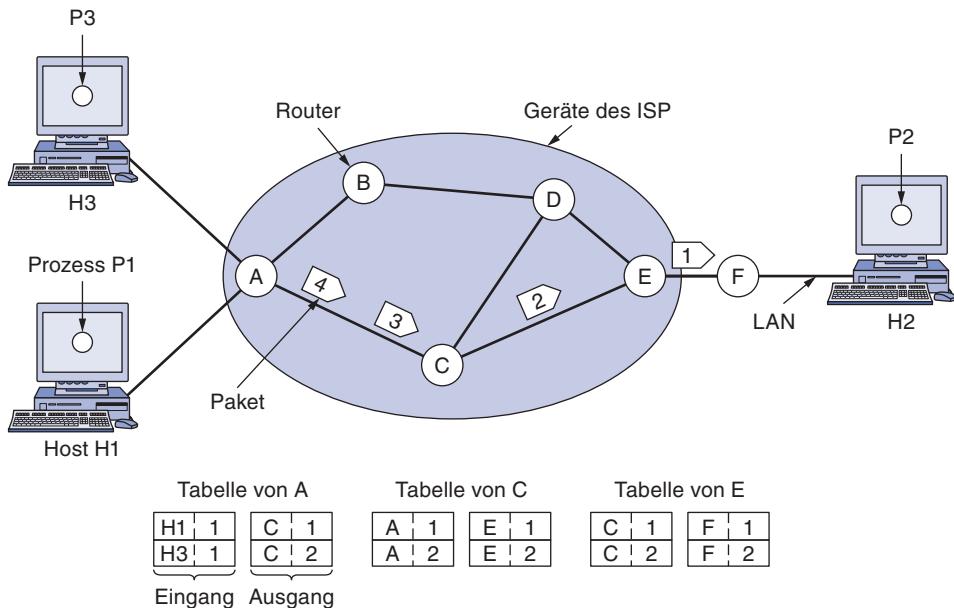


Abbildung 5.3: Routing in einem VC-Netz.

Nun wollen wir uns ansehen, was passiert, wenn H_3 eine Verbindung zu H_2 aufbauen möchte. Der Host wählt den Verbindungsidentifikator 1 (da er die Verbindung initiiert hat und dies die einzige Verbindung ist) und weist das Netz an, eine virtuelle Verbin-

dung einzurichten. Dies führt zu der zweiten Zeile in den Tabellen. Beachten Sie, dass hier ein Konflikt vorliegt: Obwohl A sehr einfach die Verbindung-1-Pakete von H1 von denen von H3 unterscheiden kann, ist C nicht dazu in der Lage. Daher weist A für den ausgehenden Datenverkehr der zweiten Verbindung einen anderen Verbindungsidentifikator zu. Um derartige Konflikte zu vermeiden, benötigen Router die Fähigkeit, die Verbindungsidentifikatoren ausgehender Pakete zu ersetzen.

In bestimmten Kontexten wird dies als **Label Switching** (Austausch von Beschriftungen) bezeichnet. Ein Beispiel für einen verbindungsorientierten Netzdienst ist **MPLS** (*Multi-Protocol Label Switching*). MPLS wird innerhalb von ISP-Netzwerken im Internet eingesetzt, indem IP-Pakete in einen MPLS-Header eingewickelt werden, der einen 20-Bit-Verbindungsidentifikator oder ein Label hat. MPLS wird oft vor den Kunden verborgen, wenn der ISP dauerhafte Verbindungen für die Übertragung großer Datenmengen einrichtet. Doch zunehmend wird MPLS auch dann verwendet, wenn Dienstgüte wichtig ist oder die Verwaltung des ISP-Verkehrs unterstützt werden soll. Wir werden später in diesem Kapitel noch auf MPLS zurückkommen.

5.1.5 Vergleich von VC-Netzen und Datagrammnetzen

Sowohl virtuelle Verbindungen als auch Datagramme haben ihre Befürworter und Gegner. Wir wollen hier den Versuch unternehmen, einen Überblick über die Argumente beider Seiten zu geben. Die wichtigsten Kriterien sind in ► Abbildung 5.4 aufgeführt – Puristen werden vermutlich für alles ein Gegenargument finden.

| Kriterium | Datagrammnetz | VC-Netz |
|----------------------------|---|--|
| Verbindungsaufbau | Nicht erforderlich | Erforderlich |
| Adressierung | Jedes Paket enthält die volle Ziel- und meist auch die Quelladresse | Jedes Paket enthält eine kurze VC-Nummer |
| Zustandsinformation | Router führen keine Zustandsinformationen über die Verbindung | Für jede virtuelle Verbindung ist ein Tabelleneintrag erforderlich |
| Routing | Jedes Paket wird unabhängig befördert | Die Route wird beim Aufbau der virtuellen Verbindung gewählt; alle Pakete folgen dieser Route |
| Wirkung von Router-Fehlern | Keine, außer dass Pakete verloren gehen | Alle virtuellen Verbindungen über den ausgefallenen Router werden beendet |
| Dienstgüte | Schwierig | Einfach, wenn im Voraus für jede virtuelle Verbindung ausreichend Ressourcen bereitgestellt werden |
| Überlastungskontrolle | Schwierig | Einfach, wenn im Voraus für jede virtuelle Verbindung ausreichend Ressourcen bereitgestellt werden |

Abbildung 5.4: Vergleich von VC-Netzen mit Datagrammnetzen.

Innerhalb des Netzes werden in Bezug auf virtuelle Verbindungen und Datagramme verschiedene Kompromisse getroffen. Ein Kompromiss muss zwischen der Einrichtzeit gegenüber der Zeit, die zum Parsen der Adresse verwendet wird, gefunden werden. Bei virtuellen Verbindungen ist eine Einrichtphase erforderlich, die Zeit und Ressourcen kostet. Ist dieser Preis jedoch einmal bezahlt, dann ist es in einem VC-Netz einfach herauszufinden, wie mit einem Datenpaket weiter verfahren werden soll: Der Router benutzt lediglich die Verbindungsnummer, um das Ziel eines Pakets zu ermitteln. Bei einem Datagrammnetz ist eine kompliziertere Nachschlageprozedur erforderlich, um den Zieleintrag zu ermitteln.

Ein ähnliches Problem ist, dass die in Datagrammnetzen verwendeten Zieladressen länger als Verbindungsnummern sind, die in VC-Netzen benutzt werden, weil sie eine globale Bedeutung haben. Bei relativ kurzen Paketen bedeutet eine volle Zieladresse in jedem Paket einen beträchtlichen Overhead, d.h. verschwendete Bandbreite.

Ein anderer Aspekt ist der Tabellenplatz, der im Router-Speicher belegt wird. Ein Datagrammnetz muss für jedes mögliche Ziel einen Eintrag besitzen, ein VC-Netz muss dagegen nur für jede virtuelle Verbindung einen Eintrag aufweisen. Dieser Vorteil ist aber eine gewisse Illusion, da die Pakete zur Errichtung einer Verbindung ebenfalls weitgeleitet werden müssen, und sie verwenden genau wie Datagramme Zieladressen.

Virtuelle Verbindungen haben gewisse Vorteile bezüglich der Sicherstellung der Dienstgüte und der Vermeidung von Überlastungen im Netz, weil Ressourcen (wie Puffer, Bandbreite und CPU-Zyklen) im Voraus – beim Aufbau der Verbindung – reserviert werden. Kommen Pakete an, dann sind Bandbreite und Router-Kapazität dem Bedarf entsprechend vorhanden. Bei einem Datagrammnetz ist die Vermeidung von Überlastung schwieriger.

Bei Transaktionsverarbeitungssystemen (z.B. die Anrufe von Geschäften bei einer Kreditkartengesellschaft, wenn Kunden mit Kreditkarte bezahlen) kann der durch den Auf- und Abbau einer virtuellen Verbindung entstehende Overhead die Nutzung der Verbindung leicht in den Schatten stellen. In Umgebungen, in denen vorwiegend mit dieser Verkehrsart zu rechnen ist, ist die Nutzung von virtuellen Verbindungen im Netz kaum sinnvoll. Dafür eignen sich permanente virtuelle Verbindungen für lang anhaltende Nutzung wie bei VPN-Verkehr zwischen zwei verbundenen Büros (die manuell aufgebaut und Monate oder Jahre aufrechterhalten werden).

Virtuelle Verbindungen sind außerdem recht empfindlich. Bricht ein Router zusammen und verliert seinen Hauptspeicherinhalt, dann werden alle vorhandenen virtuellen Verbindungen abgebrochen, auch wenn der Ausfall nur Sekunden dauert. Demgegenüber sind in einem Datagrammnetz bei einem Router-Ausfall – wenn überhaupt – nur die Benutzer betroffen, deren Pakete im Moment im Router in der Warteschlange anstanden (und wahrscheinlich nicht einmal dann, da der Sender diese vermutlich kurz darauf neu überträgt). Der Ausfall einer Übertragungsleitung ist für virtuelle Verbindungen fatal, kann aber leicht ausgeglichen werden, wenn Datagramme verwendet werden. Durch Datagramme können die Router auch den Verkehrs durchsatz im Netz ausgleichen, da bei einer langen Folge von Paketübertragungen die Versandwege mittendrin gewechselt werden können.

5.2 Routing-Algorithmen

Die wichtigste Funktion der Vermittlungsschicht besteht darin, die Pakete von der Quelle zum Ziel weiterzuleiten. In den meisten Netzwerken durchlaufen die Pakete bei dieser Reise mehrere Teilstrecken (Hops). Die einzige nennenswerte Ausnahme bilden Broadcast-Netze, aber auch hier ist das Routing ein Thema, wenn sich Quelle und Ziel nicht im gleichen Netzsegment befinden. Die Algorithmen zum Auswählen der Routen und der verwendeten Datenstrukturen sind ein zentrales Thema bei der Entwicklung der Vermittlungsschicht.

Der **Routing-Algorithmus** ist jener Teil der Software der Vermittlungsschicht, der über die Ausgabeleitung für eingehende Pakete entscheidet. Nutzt das Netz intern Datagramme, dann muss diese Entscheidung bei jedem Datenpaket neu getroffen werden, weil sich der beste Pfad seit der letzten Übertragung geändert haben kann. Verwendet das Netz intern virtuelle Verbindungen, werden Routing-Entscheidungen nur gefällt, wenn eine neue virtuelle Verbindung aufgebaut wird. Danach folgen alle Datenpakete diesem bereits bestehenden Weg. Dieser Fall wird auch **Session-Routing** genannt, weil der Weg während einer gesamten Sitzung bestehen bleibt (z.B. während der Anmeldung über ein VPN).

Manchmal ist eine Unterscheidung zwischen Routing – die Entscheidung, welche Pfade verwendet werden sollen – und der Weiterleitung – die Verarbeitung, die bei der Ankunft eines Pakets abläuft – von Vorteil. Man kann sich einen Router so vorstellen, dass er über zwei Prozesse verfügt. Einer verwaltet jedes ankommende Paket und sucht die zu verwendende Ausgangsleitung in den Routing-Tabellen. Diesen Prozess nennt man **Weiterleitung (forwarding)**. Der andere Prozess füllt die Routing-Tabellen aus und aktualisiert sie. Hier kommt dann der Routing-Algorithmus ins Spiel.

Unabhängig davon, ob die Routen für jedes Paket einzeln oder nur einmal beim Aufbau einer neuen Verbindung gewählt werden, sind folgende Eigenschaften eines Routing-Algorithmus wünschenswert: korrekt, einfach, robust, stabil, fair und effizient. Korrektheit und Einfachheit bedürfen keiner weiteren Erklärung. Der Anspruch auf Robustheit mag zunächst nicht einleuchten. Wird ein großes Netz eingerichtet, geht man davon aus, dass es über Jahre hinweg fehlerfrei läuft. In dieser Zeitspanne treten aber Hardware- und Softwarefehler aller Art auf. Hosts, Router und Leitungen werden immer wieder ausfallen und die Topologie wird immer wieder geändert werden. Der Routing-Algorithmus sollte Änderungen der Topologie und des Datenverkehrs bewältigen können, ohne dass nach dem Absturz eines Routers alle Jobs in allen Hosts abgebrochen werden muss. Stellen Sie sich das Chaos vor, wenn das Netz jedes Mal hochgefahren werden müsste, sobald irgendwo ein Router ausfällt!

Auch Stabilität ist ein wichtiges Ziel des Routing-Algorithmus. Es gibt Routing-Algorithmen, die nie eine feste Pfadmenge erreichen, ganz gleich, wie lange sie laufen. Ein stabiler Algorithmus konvergiert in einen stabilen Zustand und bleibt darin. Dies sollte außerdem schnell gehen, da die Kommunikation getrennt werden könnte, bevor der Routing-Algorithmus den stabilen Zustand erreicht.

Fairness und Effizienz mögen offensichtlich scheinen – sicher würde kein vernünftiger Mensch etwas dagegen einwenden –, sind aber oft widersprüchlich. Als einfaches Beispiel dafür betrachte man ►Abbildung 5.5. Angenommen, zwischen A und A' , zwischen B und B' sowie zwischen C und C' fließt ausreichend Datenverkehr, um die horizontalen Verbindungen auszulasten. Um den Gesamtfluss zu maximieren, müsste der Verkehr zwischen X und X' gänzlich abgeschaltet werden. Leider sehen das X und X' nicht unbedingt so. Folglich muss zwischen der globalen Effizienz und der Fairness gegenüber einzelnen Verbindungen ein Kompromiss getroffen werden.

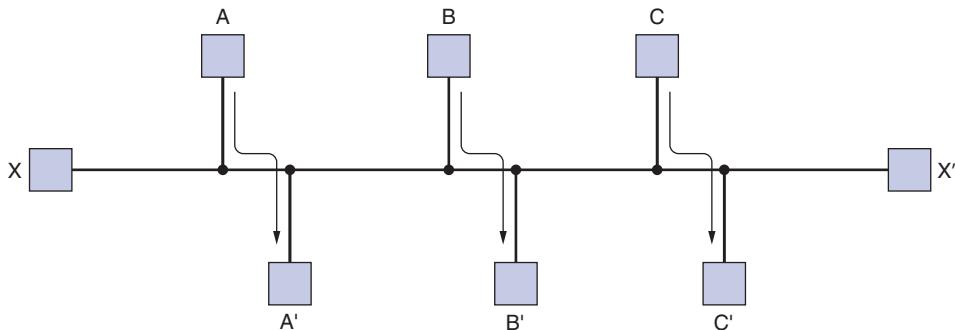


Abbildung 5.5: Netz mit einem Konflikt zwischen Fairness und Effizienz.

Bevor man überhaupt beginnt, sich mit Kompromissen zwischen Fairness und Effizienz zu befassen, muss ermittelt werden, was es zu optimieren gilt. Die Minimierung der durchschnittlichen Paketübertragungszeit, um Datenverkehr effektiv durch das Netz zu senden, ist eine Möglichkeit, interessant wäre aber auch die Maximierung des gesamten Netzdurchsatzes. Außerdem stellen diese zwei Ziele einen Konflikt dar, weil der Betrieb eines Warteschlangensystems nahe der Kapazitätsgrenze bedeutet, dass die Warteschlangenverzögerung lang ist. Bei vielen Netzen wird deshalb versucht, die Entfernung zu minimieren, die ein Paket zurücklegen muss, oder einfach die Anzahl der zu durchlaufenden Teilstrecken zu verringern. Beide Maßnahmen verbessern die Übertragungszeit und reduzieren außerdem den Bandbreitenverbrauch pro Paket, was sich wiederum positiv auf den gesamten Durchsatz des Netzes auswirkt.

Routing-Algorithmen können in zwei Hauptgruppen gegliedert werden: adaptive und nicht adaptive. **Nicht adaptive Algorithmen** gründen ihre Routing-Entscheidungen nicht auf Messungen oder Schätzungen der aktuellen Topologie und des Verkehrs. Stattdessen wird die Entscheidung, welcher Pfad von I zu J (für alle I und J) verwendet wird, vorab offline berechnet und beim Hochfahren des Netzes auf die Router geladen. Dieses Verfahren nennt man auch **statisches Routing** (*static routing*). Da statisches Routing nicht auf Ausfälle reagiert, ist es für solche Situationen am sinnvollsten, in denen die Routing-Entscheidung klar ist. Zum Beispiel sollte Router F in Abbildung 5.3 alle Pakete für das Netz unabhängig vom endgültigen Ziel immer zu Router E senden.

Adaptive Algorithmen ändern demgegenüber ihre Routing-Entscheidungen entsprechend den an der Topologie durchgeföhrten Änderungen und manchmal auch entsprechend den Änderungen beim Verkehr. Diese **dynamischen Routing-Algorithmen**

unterscheiden sich darin, woher sie ihre Informationen holen (z.B. lokal von benachbarten Routern oder von allen Routern), wann sie die Pfade ändern (z.B. alle ΔT Sekunden oder wenn sich die Last oder die Topologie ändert) und welche Metrik sie zur Optimierung verwenden (z.B. Entfernung, Anzahl der Teilstrecken oder geschätzte Übertragungszeit).

In den folgenden Abschnitten werden wir eine Vielzahl von Routing-Algorithmen beschreiben. Neben dem Senden eines Pakets von einer Quelle zu einem Ziel behandeln die Algorithmen auch Zustellungsmodelle. Manchmal soll das Paket zu mehreren Zielen, zu allen Zielen oder zu einem von vielen Zielen geschickt werden. Alle hier beschriebenen Routing-Algorithmen basieren ihre Entscheidungen auf der Topologie; in Abschnitt 5.3 werden wir die Möglichkeit besprechen, Entscheidungen aufgrund des Verkehrs zu treffen.

5.2.1 Das Optimalitätsprinzip

Vor der Beschreibung der einzelnen Algorithmen sollte man noch hervorheben, dass man eine allgemeine Aussage über optimale Pfade ohne Berücksichtigung der Netztopologie und des Verkehrs treffen kann. Diese Aussage wird als **Optimalitätsprinzip** (Bellman, 1957) bezeichnet. Es sagt Folgendes aus: Wenn Router J auf dem optimalen Pfad von Router I zu Router K liegt, dann gehört der optimale Weg von J nach K zum gleichen Pfad. Um das zu zeigen, nennen wir den Teil des Pfads von I nach J r_1 und den restlichen Pfad r_2 . Gibt es zwischen J und K einen besseren Weg als r_2 , kann dieser mit r_1 verkettet und der Weg von I nach K damit verbessert werden, was der Aussage widerspricht, dass r_1r_2 optimal ist.

Als unmittelbare Folge des Optimalitätsprinzips sehen wir, dass die optimalen Routen von allen Quellen zu einem bestimmten Ziel einen Baum bilden, der am Ziel wurzelt. Einen solchen Baum nennt man **Quelle-Senke-Baum** (*sink tree*). Ein Beispiel hierfür ist in ▶ Abbildung 5.6b dargestellt, wo die Distanz über die Anzahl der Teilstrecken gemessen wird. Das Ziel aller Routing-Algorithmen ist, die Quelle-Senke-Bäume für alle Router zu ermitteln und zu verwenden.

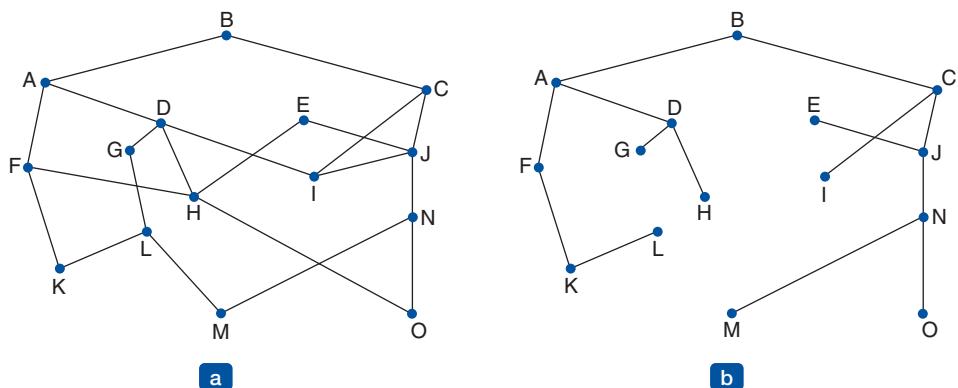


Abbildung 5.6: (a) Ein Netzwerk. (b) Ein Quelle-Senke-Baum für Router B .

Beachten Sie, dass ein Quelle-Senke-Baum nicht unbedingt eindeutig sein muss; es können auch weitere Bäume mit der gleichen Weglänge existieren. Wenn man es zulässt, dass alle möglichen Pfade gewählt werden können, so wird aus dem Baum eine allgemeinere Struktur: ein **gerichteter azyklischer Graph (DAG, Directed Acyclic Graph)**. DAGs haben keine Schleifen. Wir werden Quelle-Senke-Bäume als ein praktisches Kürzel für beide Fälle verwenden. Für beide trifft auch die (technische) Annahme zu, dass Pfade sich nicht gegenseitig beeinträchtigen, sodass beispielsweise durch einen Verkehrsstau auf dem einen Pfad kein anderer Pfad umgeleitet werden muss.

Da ein Quelle-Senke-Baum tatsächlich ein Baum ist, enthält er keine Schleifen, das heißt, Pakete werden auf einer endlichen und begrenzten Anzahl von Teilstrecken übertragen. In der Praxis ist das nicht ganz so einfach. Leitungen und Router können ausfallen und während des laufenden Betriebs wieder hochfahren, sodass verschiedene Router unterschiedliche Kenntnisse der aktuellen Topologie haben können. Auch die Frage, ob jeder Router die Informationen, auf die er seine Berechnung des Quelle-Senke-Baums gründet, selber finden muss oder ob diese Information auf andere Weise erfasst wird, wurde hier unter den Tisch gekehrt. Dieses Thema werden wir später noch einmal aufgreifen. Dennoch stellen das Optimalitätsprinzip und das Konzept des Quelle-Senke-Baums gute Bezugspunkte zur Beurteilung von Routing-Algorithmen dar.

5.2.2 Routing nach dem kürzesten Pfad

Wir beginnen unsere Untersuchung der Routing-Algorithmen mit einer einfachen Technik zur Berechnung von optimalen Pfaden, unter der Annahme, dass ein vollständiges Bild des Netzes gegeben ist. Dies sind genau die Pfade, von denen wir möchten, dass ein verteilter Routing-Algorithmus sie findet, selbst wenn nicht alle Router alle Details des Netzes kennen.

Die Idee hierbei ist, einen Graphen des Netzes zu erstellen, wobei jeder Knoten im Graphen einen Router und jede Kante eine Übertragungsleitung oder eine Verbindung darstellt. Um einen Weg zwischen einem bestimmten Router-Paar auszuwählen, berechnet der Algorithmus anhand des Graphen den kürzesten Weg.

Das Konzept des **kürzesten Pfades** (*shortest path*) verdient einige Erklärungen. Eine Möglichkeit, die Pfadlänge zu messen, ist die Anzahl der Teilstrecken zu berechnen. Bei dieser Bewertung sind die Pfade *ABC* und *ABE* in ▶ Abbildung 5.7 gleich lang. Ein anderes Maß ist die geografische Entfernung in Kilometern. In diesem Fall ist *ABC* deutlich länger als *ABE* (unter der Voraussetzung, dass die Abbildung maßstabgerecht ist).

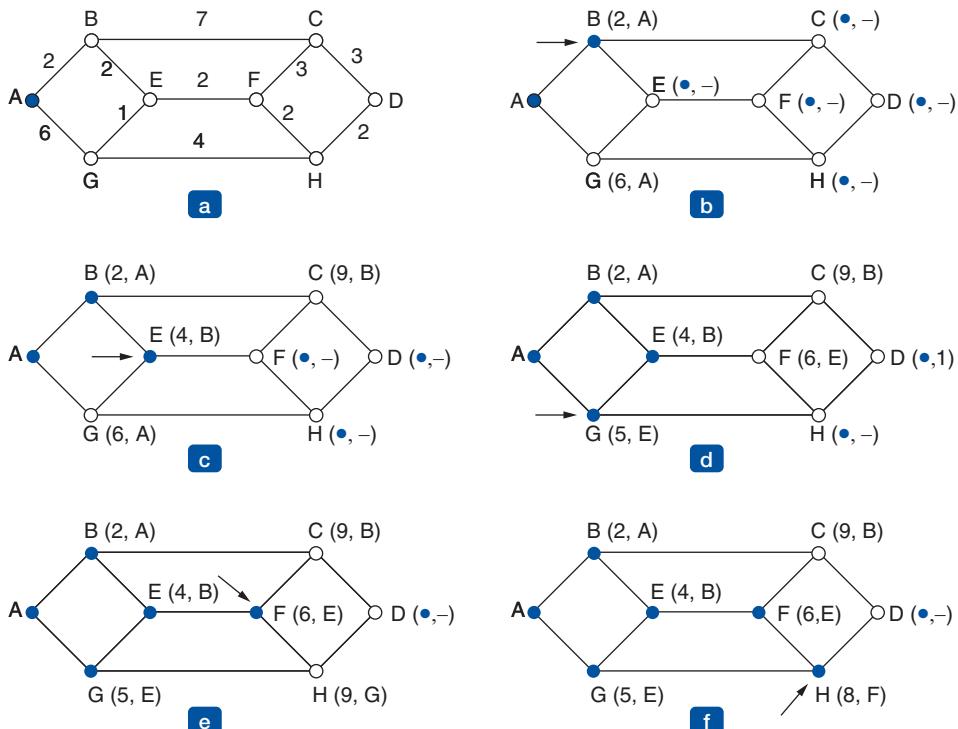


Abbildung 5.7: Die ersten sechs Schritte für die Berechnung des kürzesten Pfades von A nach D . Der Pfeil kennzeichnet den Arbeitsknoten.

Abgesehen von Teilstrecken und physikalischen Entfernungen sind noch andere Metriken möglich. Jede Kante kann beispielsweise mit der mittleren Übertragungszeit eines Standardtestpakets markiert werden, die stündlich ermittelt wird. Bei dieser Graphenbeschriftung ist der kürzeste Pfad der schnellste Pfad, und nicht derjenige mit den wenigsten Kanten oder Kilometern.

Im Allgemeinen können die Beschriftungen der Kanten als Funktion von Entfernung, Bandbreite, Durchschnittsverkehr, Übertragungskosten, gemessener Übertragungszeit und anderen Faktoren berechnet werden. Durch Ändern der Gewichtungsfunktion wird mit dem Algorithmus dann der „kürzeste“ Pfad nach einer Reihe von Kriterien oder einer Kombination derselben gemessen.

Zur Berechnung des kürzesten Pfads zwischen zwei Knoten eines Graphen sind verschiedene Algorithmen bekannt. Der hier vorgestellte Algorithmus geht auf Dijkstra (1959) zurück und findet den kürzesten Pfad zwischen einer Quelle und allen Zielen im Netz. Jeder Knoten wird (in Klammern) mit seiner Entfernung vom Quellknoten auf dem besten bekannten Pfad beschriftet. Die Entfernen dürfen nicht negativ sein, wie es der Fall wäre, wenn sie auf realen Quantitäten wie Bandbreite und Übertragungszeit basierten. Anfangs ist kein Pfad bekannt, sodass alle Knoten die Bezeichnung „unendlich“ tragen. Je weiter der Algorithmus fortschreitet und Pfade gefunden werden, desto mehr kann sich die Beschriftung ändern und jeweils bessere Pfade

anzeigen. Eine Beschriftung (Label) kann provisorisch oder permanent sein. Zunächst sind alle Beschriftungen provisorisch. Wird festgestellt, dass eine Beschriftung den kürzestmöglichen Pfad von der Quelle zu einem Knoten angibt, wird sie permanent gesetzt und danach nie mehr geändert.

Um zu verdeutlichen, wie der Algorithmus zur Vergabe von Beschriftungen arbeitet, betrachten wir den gewichteten ungerichteten Graphen in ►Abbildung 5.7a, in dem die Gewichtungen, wie beispielsweise die Entfernung, eingetragen sind. Wir suchen den kürzesten Weg von A nach D . Wir beginnen, indem wir Knoten A als permanent markieren – dargestellt durch einen gefüllten Kreis. Nun untersuchen wir von A (dem Arbeitsknoten) aus jeden benachbarten Knoten und beschriften jeden mit seiner Entfernung zu A . In jeder solchen Beschriftung zu einem Knoten wird auch der Knoten angegeben, von dem aus gemessen wurde, sodass der endgültige Pfad später rekonstruiert werden kann. Wenn das Netz mehr als einen kürzesten Pfad von A zu D hat und wir alle kürzesten Pfade finden wollen, dann müssten wir uns alle untersuchten Knoten merken, die ein Knoten mit derselben Entfernung erreichen kann.

Nach der Untersuchung jedes Nachbarknotens von A untersuchen wir alle vorläufig beschrifteten Knoten im gesamten Graphen und setzen den mit der kleinsten Beschriftung permanent, wie in ►Abbildung 5.7b. Dieser wird der neue Arbeitsknoten.

Nun beginnen wir bei B und untersuchen alle Nachbarn. Sollte die Summe der Beschriftung auf B und der Entfernung von B zu dem gerade untersuchten Knoten kleiner sein als die bisherige Beschriftung dieses Knotens, liegt ein kürzerer Pfad vor, also wird die Beschriftung geändert.

Nachdem alle Nachbarn des Arbeitsknotens untersucht und die vorläufigen Beschriftungen – soweit möglich – geändert wurden, wird der gesamte Graph nach dem vorläufig beschrifteten Knoten mit dem kleinsten Wert abgesucht. Dieser Knoten wird permanent gesetzt und in der nächsten Runde als Arbeitsknoten deklariert. Abbildung 5.7 zeigt die ersten sechs Schritte des Algorithmus.

Um zu verstehen, warum der Algorithmus funktioniert, betrachten wir ►Abbildung 5.7c. Hier haben wir gerade E permanent gesetzt. Nehmen wir an, dass es einen kürzeren Pfad als ABE gibt, sagen wir $AXYZE$ (für beliebige X und Y). Es gibt zwei Möglichkeiten: Entweder wurde der Knoten Z bereits permanent gemacht oder nicht. Wenn ja, wurde E bereits untersucht (in der Runde nach derjenigen, in der Z permanent gesetzt wurde). Der Pfad $AXYZE$ ist also unserer Aufmerksamkeit nicht entgangen und kann daher kein kürzerer Pfad sein.

Betrachten wir nun den Fall, dass Z immer noch provisorisch beschriftet ist. Falls die Beschriftung von Z größer oder gleich der von E ist, dann kann $AXYZE$ nicht kürzer als ABE sein. Falls die Beschriftung kleiner als die von E ist, so wird nicht E , sondern zuerst Z als permanent markiert, wodurch E von Z aus untersucht werden kann.

Dieser Algorithmus wird in ►Abbildung 5.8 dargestellt. Die globalen Variablen n und $dist$ beschreiben den Graphen und werden initialisiert, bevor $shortest_path$ aufgerufen wird. Der einzige Unterschied zwischen dem Programm und dem oben beschrie-

benen Algorithmus ist der, dass wir in Abbildung 5.8 den kürzesten Pfad vom Endknoten t und nicht vom Quellknoten s berechnen.

```
#define MAX_NODES 1024           /* maximale Anzahl der Knoten */
#define INFINITY 1000000000       /* Zahl, die größer ist als jeder maximale
                                Pfad */
int n, dist[MAX_NODES][MAX_NODES]; /* dist[i][j] ist die Entfernung
                                von i nach j */

void shortest_path(int s, int t, int path[])
{ struct state {               /* Arbeitspfad */
    int predecessor;          /* vorheriger Knoten */
    int length;                /* Länge von der Quelle zu diesem Knoten */
    enum {permanent, tentative} label; /* Beschriftungsart */
} state[MAX_NODES];

int i, k, min;
struct state * p;

for (p = &state[0]; p < &state[n]; p++) { /* Zustand initialisieren */
    p->predecessor = -1;
    p->length = INFINITY;
    p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t;                      /* k ist der anfängliche Arbeitsknoten */
do {
    for (i = 0; i < n; i++)      /* dieser Graph hat n Knoten */
        if (dist[k][i] != 0&&state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
    }
} /* Provisorisch beschrifteten Knoten mit der kleinsten Beschriftung
   ermitteln */

k = 0; min = INFINITY;
for (i = 0; i < n; i++)
    if (state[i].label == tentative &&state [i].length < min) {
        min = state[i].length;
        k = i;
    }
state[k].label = permanent;
} while (k != s);

/* Pfad in das Ausgangsarray kopieren */
i = 0; k = s;
do {path[i++] = k; k = state[k].predecessor;} while (k >= 0);
}
```

Abbildung 5.8: Der Algorithmus von Dijkstra zur Berechnung des kürzesten Pfads in einem Graphen.

Da der kürzeste Pfad von t nach s in einem ungerichteten Graphen und die kürzesten Pfade von s nach t gleich sind, spielt es keine Rolle, an welchem Ende man beginnt.

Der Grund für die Rückwärtssuche ist, dass jeder Knoten mit seinem Vorgänger statt mit seinem Nachfolger beschriftet ist. Wenn der endgültige Pfad in die Ausgabevariable *path* kopiert wurde, wird er umgedreht. Die beiden Effekte heben sich auf und die Antwort wird in der richtigen Reihenfolge erzeugt.

5.2.3 Fluten

Wenn ein Routing-Algorithmus implementiert wird, muss jeder Router Entscheidungen treffen, die auf lokalem Wissen basieren und nicht auf dem vollständigen Bild vom Netz. Eine einfache lokale Technik ist **Fluten** (*flooding*). Hier wird jedes ankommende Paket über jede Ausgangsleitung gesendet, außer über diejenige, auf der es angekommen ist.

Beim Fluten wird natürlich eine enorme Anzahl an Paketduplicaten erstellt. Diese Zahl kann in der Tat unendlich sein, wenn nicht Maßnahmen ergripen werden, um dies einzudämmen. Eine mögliche Maßnahme ist ein Teilstreckenzähler, der im Header der Pakete enthalten ist und bei jeder Teilstrecke um 1 reduziert wird. Erreicht der Zähler null, so wird das Paket verworfen. Im Idealfall sollte der Teilstreckenzähler auf die Länge des Pfads von der Quelle zum Ziel initialisiert sein. Ist dem Sender die Pfadlänge nicht bekannt, kann er den Zähler auf den schlechtesten Fall initialisieren, d.h. auf den vollen Durchmesser des Netzes.

Fluten mit einem Teilstreckenzähler kann eine exponentielle Zahl an Paketduplicaten hervorbringen, wenn der Streckenzähler wächst und Router Pakete duplizieren, die vorher bereits einmal bei diesem Router waren. Eine bessere Technik zum Eindämmen des Flutens ist, dass sich die Router merken, welche Pakete bereits via Fluten weitergeleitet wurden, sodass sie nicht ein zweites Mal gesendet werden. Das lässt sich z.B. dadurch erreichen, dass der Quell-Router in jedes Paket, das er von seinen Hosts empfängt, eine Sequenznummer einfügt. Jeder Router muss dann eine Liste pro Quell-Router führen, in der die Sequenznummern erfasst werden. Steht ein ankommendes Paket auf der Liste, wird es nicht per Fluten gesendet.

Um zu verhindern, dass sich die Liste endlos verlängert, sollte sie um einen Zähler k erweitert werden, der angibt, dass alle Sequenznummern bis k bereits gesehen wurden. Kommt ein Paket an, lässt sich leicht ermitteln, ob das Paket bereits per Fluten gesendet wurde (indem seine Sequenznummer mit k verglichen wird). In diesem Fall wird es verworfen. Der Teil der Liste vor k wird nicht mehr benötigt, da er sozusagen durch k zusammengefasst wird.

Für das Senden der meisten Pakete ist Fluten nicht praktisch, aber es hat einige wichtige Anwendungen. Zunächst einmal stellt Fluten sicher, dass ein Paket an jeden Knoten im Netz zugestellt wird. Falls nur ein einziges Ziel dieses Paket benötigt, ist das zwar verschwenderisch, aber für den Broadcast von Informationen ist Fluten sehr effektiv. In drahtlosen Netzen können alle Nachrichten, die von einer Station übertragen werden, von allen anderen Stationen im gleichen Funkbereich empfangen werden. Dies ist de facto Fluten. Einige Algorithmen nutzen diese Eigenschaft.

Zweitens ist Fluten unglaublich robust. Selbst wenn viele Router mit einem Schlag in die Luft fliegen (z.B. in einem militärischen Netz innerhalb einer Kriegszone), wird mithilfe von Fluten ein Pfad gefunden (falls einer existiert), um ein Paket an sein Ziel zu bringen. Außerdem stellt Fluten keine hohen Anforderungen an die Einrichtung. Die Router müssen nur ihre Nachbarn kennen. Dies bedeutet, Fluten kann als Baustein in anderen Routing-Algorithmen benutzt werden, die effizienter, aber aufwendiger einzurichten sind. Fluten kann auch als Maß für den Vergleich mit anderen Routing-Algorithmen herangezogen werden. Fluten wählt immer den kürzesten Pfad, weil es alle möglichen Pfade gleichzeitig wählt. Folglich kann kein anderer Algorithmus eine kürzere Übertragungszeit erzeugen (wenn man den von diesem Verfahren selbst erzeugten Overhead außer Betracht lässt).

5.2.4 Distanzvektoralgorithmus

Rechnernetze verwenden im Allgemeinen dynamische Routing-Algorithmen, die komplexer als Fluten, aber effizienter sind, weil sie die kürzesten Pfade für die aktuelle Topologie finden. Die am häufigsten angewandten dynamischen Routing-Algorithmen sind der Distanzvektoralgorithmus und das Link-State-Routing (Routing nach dem Verbindungszustand). In diesem Abschnitt behandeln wir den ersten Algorithmus, im folgenden Abschnitt gehen wir auf den zweiten ein.

Beim **Distanzvektoralgorithmus** (*distance vector routing*) verwaltet jeder Router eine Tabelle (d.h. einen Vektor), die für jedes Ziel die bestmögliche bekannte Entfernung und die zu verwendende Leitung enthält. Diese Tabellen werden durch Austausch von Informationen mit den benachbarten Routern aktualisiert. Zum Schluss kennt jeder Router die beste Verbindung, um ein Ziel zu erreichen.

Der Distanzvektoralgorithmus ist auch unter anderen Bezeichnungen bekannt, am häufigsten als verteilter **Bellman-Ford-Algorithmus**, benannt nach den Wissenschaftlern, die diese Algorithmen entwickelt haben (Bellman, 1957; Ford und Fulkerson, 1962). Dies war der ursprüngliche Routing-Algorithmus im ARPANET und wurde auch im Internet unter der Bezeichnung RIP (*Routing Information Protocol*) eingesetzt.

Beim Distanzvektoralgorithmus führt jeder Router eine Routing-Tabelle, die für jeden im Netz vorhandenen Router einen Eintrag enthält und nach diesen Einträgen indiziert wird. Dieser Eintrag hat zwei Teile: die bevorzugte Ausgangsleitung zu diesem Ziel und die geschätzte Zeit oder Entfernung zu diesem Ziel. Die Entfernung kann als Anzahl der Teilstrecken angegeben werden oder es werden andere Metriken benutzt, die wir bei der Berechnung der kürzesten Pfade besprochen haben.

Es wird vorausgesetzt, dass jeder Router die „Entfernung“ zu seinen Nachbarn kennt. Werden die Teilstrecken als Maß verwendet, dann ist die Entfernung einfach eine Teilstrecke. Wird die Übertragungszeit als Maß herangezogen, so kann der Router diese direkt mit speziellen ECHO-Paketen messen, in die der Empfänger einen Zeitstempel einfügt und die er so schnell wie möglich zurücksendet.

Als Beispiel nehmen wir an, dass die Übertragungszeit als Maß verwendet wird und der Router die Übertragungszeit zu all seinen Nachbarn kennt. Einmal alle T ms sendet jeder Router an jeden Nachbarn eine Liste mit den geschätzten Übertragungszeiten zu jedem Ziel. Er empfängt eine ähnliche Liste von seinen Nachbarn. Nehmen wir an, dass eine dieser Tabellen gerade vom Nachbarn X eingetroffen ist, wobei X_i die Schätzung von X ist, wie lange der Weg zu Router i dauert. Wenn der Router weiß, dass die Übertragung zu X m ms dauert, so weiß er auch, dass er jeden Router i über X in $X_i + m$ ms erreichen kann. Berechnet man dies für jeden Nachbarknoten, dann kann ein Router herausfinden, welche Schätzung am besten zu sein scheint, und diese mit der zugehörigen Verbindung dann in seiner neuen Routing-Tabelle verwenden. Die alte Routing-Tabelle wird nicht zur Berechnung herangezogen.

Dieser Aktualisierungsprozess ist in ▶ Abbildung 5.9 dargestellt. Teil (a) zeigt ein Netz. Die ersten vier Spalten von Teil (b) zeigen die von den Nachbarn von Router J eingegebenen Übertragungsvektoren. A schätzt die Übertragungszeit zu B auf 12 ms, 25 ms zu C , 40 ms zu D usw. Wir nehmen an, dass J die Übertragungszeit zu seinen Nachbarn A, I, H und K mit 8, 10, 12 bzw. 6 ms gemessen oder geschätzt hat.

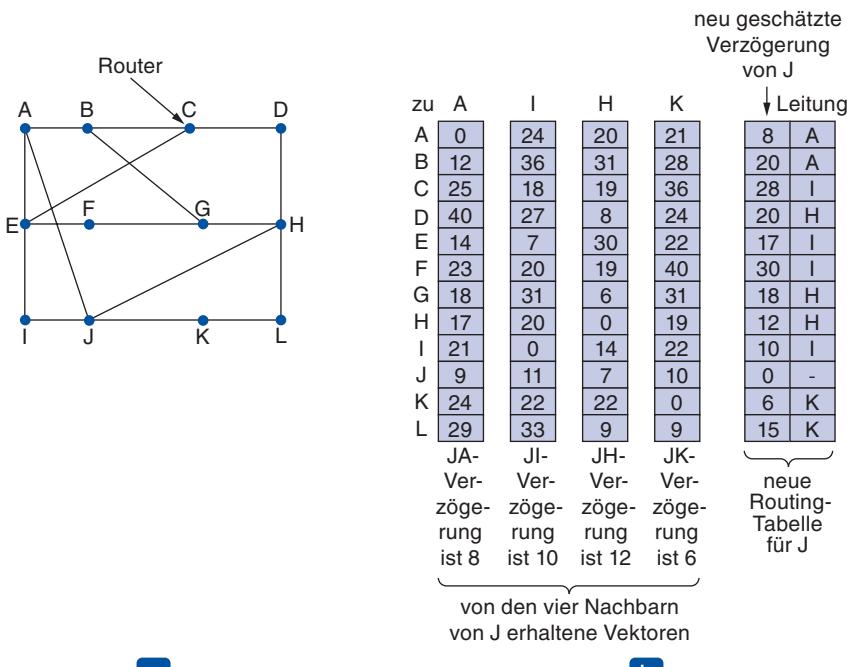


Abbildung 5.9: (a) Ein Netz. (b) Eingaben von A, I, H, K und die neue Routing-Tabelle für J .

Nun betrachten wir, wie J seinen neuen Weg zu Router G berechnet. J weiß, dass er A in 8 ms erreichen kann. A behauptet weiterhin, G in 18 ms erreichen zu können. Daraus schließt J , dass er mit einer Übertragungszeit von 26 ms zu G rechnen muss, wenn er Pakete für G nach A überträgt. Ebenso berechnet J die Übertragungszeit zu G über I, H und K mit 41 (31+10), 18 (6+12) und 37 (31+6) ms. Der beste dieser Werte ist 18.

Deshalb trägt J in seine Routing-Tabelle ein, dass die Übertragungszeit zu G 18 ms beträgt und der zu verwendende Weg über H führt. Die gleiche Berechnung wird für alle anderen Ziele durchgeführt. Die neue Routing-Tabelle ist in der letzten Spalte der Abbildung dargestellt.

Das Count-to-Infinity-Problem

Das endgültige Festlegen von Routen zu besten Pfaden durch das Netz wird **Konvergenz** genannt. Distanzvektoralgorithmen sind als eine einfache Technik sinnvoll, mit der Router gemeinsam kürzeste Pfade berechnen können, doch sie haben in der Praxis einen großen Nachteil: Der Algorithmus führt zwar zur richtigen Lösung, schafft das aber nur sehr langsam. Insbesondere reagiert er zwar schnell auf gute Nachrichten, aber sehr träge auf schlechte. Nehmen wir an, der beste Weg eines Routers zu Ziel X ist lang. Meldet Nachbar A beim nächsten Informationsaustausch plötzlich eine kürzere Übertragungszeit zu X , dann schaltet der Router einfach um, indem er Leitung A verwendet, um Datenverkehr an X zu senden. Die gute Nachricht wird mit einem einzigen Austausch im Vektor verarbeitet.

Um zu sehen, wie schnell sich gute Nachrichten verbreiten, betrachten wir das (lineare) Netz mit fünf Knoten aus ►Abbildung 5.10, bei dem das Maß für die Übertragung die Anzahl der Teilstrecken ist. Wir nehmen an, dass A anfänglich nicht in Betrieb ist und dass dies alle anderen Router wissen. Mit anderen Worten, sie haben alle die Übertragungszeit nach A als unendlich registriert.

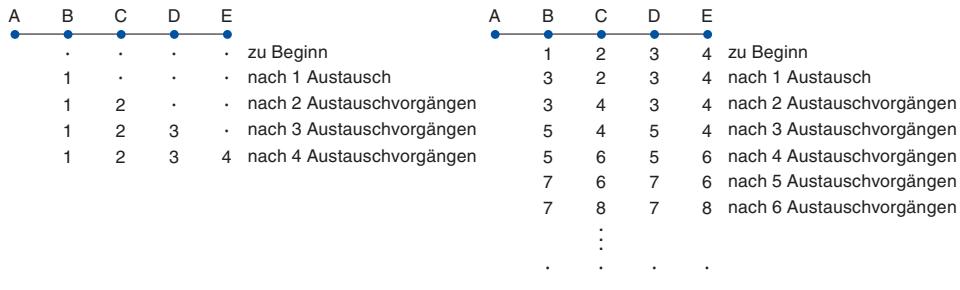


Abbildung 5.10: Das Count-to-Infinity-Problem.

Schaltet sich A wieder zu, erfahren das die anderen Router über den Vektaustaush. Der Einfachheit halber nehmen wir an, dass es irgendwo einen gigantischen Gong gibt, der in regelmäßigen Abständen ertönt, um einen Vektaustaush bei allen Routern gleichzeitig einzuleiten. Zum Zeitpunkt des ersten Austauschs erfährt B , dass sein linker Nachbar eine Übertragungszeit von null zu A hat. B trägt in seine Routing-Tabelle ein, dass A eine Teilstrecke nach links entfernt liegt. Alle anderen Router glauben noch, dass A ausgeschaltet ist. An diesem Punkt sehen die Einträge in der Routing-Tabelle für A so aus wie in der zweiten Zeile in ►Abbildung 5.10a. Beim nächsten Austausch erfährt C , dass B einen Pfad der Länge 1 zu A hat. Folglich aktualisiert C seine Routing-Tabelle auf einen Pfad der Länge 2, was D und E aber erst später erfahren. Die

guten Nachrichten verbreiten sich in einer Geschwindigkeit von einer Teilstrecke pro Austausch. In einem Netz, dessen längster Pfad eine Länge von N Teilstrecken hat, tauschen alle innerhalb von N die Nachricht über neu zugeschaltete Verbindungen und Router aus.

In der in ► Abbildung 5.10b dargestellten Situation sind alle Verbindungen und Router anfangs betriebsbereit. Die Router B , C , D und E haben eine Entfernung zu A von 1, 2, 3 bzw. 4 Teilstrecken. Plötzlich fällt entweder A aus oder die Leitung zwischen A und B wird unterbrochen (was aus der Sicht von B das Gleiche ist).

Beim ersten Paketaustausch hört B überhaupt nichts von A . Glücklicherweise sagt C : „Keine Panik, mein Pfad zu A hat die Länge 2.“ B weiß nicht, dass der Pfad von C über ihn – B – führt. Nach allem, was B weiß, könnte C zehn Verbindungen mit jeweils verschiedenen Pfaden der Länge 2 zu A haben. Deshalb denkt B , dass er A über C mit einer Pfadlänge von 3 erreichen kann. D und E aktualisieren ihre Einträge für A beim ersten Austausch nicht.

Beim zweiten Austausch bemerkt C , dass alle seine Nachbarn behaupten, einen Pfad von Länge 3 nach A zu haben. Er sucht sich einen davon aus und setzt seine neue Entfernung zu A auf 4. Dies entspricht der dritten Zeile in Abbildung 5.10b. Darauffolgende Übertragungen erzeugen den weiteren in Abbildung 5.10b dargestellten Verlauf.

Aus dieser Abbildung kann man ersehen, warum sich die schlechte Nachricht so langsam verbreitet: Kein Router hat je einen um mehr als 1 höheren Wert als der Minimalwert aller seiner Nachbarn. Allmählich arbeiten sich alle Router ihren Weg in Richtung Unendlichkeit, aber die Zahl der dafür erforderlichen Übertragungen hängt vom numerischen Wert, der für die Unendlichkeit verwendet wird, ab. Aus diesem Grund sollte die Unendlichkeit auf den längsten Pfad plus 1 gesetzt werden.

Die Bezeichnung für dieses Problem kommt nicht völlig überraschend: **Count-to-Infinity-Problem**. Es gibt viele Versuche, dieses Problem zu lösen, z.B. indem Router mit der „Split Horizon with Poisoned Reverse“-Regel davon abgehalten werden, ihre besten Pfade den Nachbarn mitzuteilen, von denen sie diese Pfade erhalten haben (siehe RFC 1058). Keine dieser Heuristiken funktioniert jedoch – trotz der klangvollen Namen – gut in der Praxis. Das Hauptproblem ist, dass wenn X Router Y mitteilt, dass er irgendwo einen Pfad hat, Y keine Möglichkeit hat festzustellen, ob er selbst Teil dieses Pfads ist.

5.2.5 Link-State-Routing

Distanzvektoralgorithmen wurden im ARPANET bis 1979 eingesetzt, dann wurden sie durch Link-State-Routing ersetzt. Der Hauptgrund für die Ablösung war, dass der Algorithmus nach einer Änderung der Netztopologie häufig zu langsam war (aufgrund des Count-to-Infinity-Problems). Daher wurde auf einen völlig neuen Algorithmus umgestellt, der **Link-State-Routing** (Routing nach dem Verbindungszustand) heißt. Varianten des Link-State-Routings sind IS-IS und OSPF – die Routing-Protokolle, die heute am häufigsten innerhalb großer Netzwerke und dem Internet eingesetzt werden.

Das Konzept von Link-State-Routing ist ziemlich einfach und besteht aus fünf Teilen. Jeder Router muss die folgenden Schritte ausführen, damit das Routing funktioniert:

1. Die Nachbarn und deren Netzadressen ermitteln
2. Die Entferungs- oder die Kostenmetrik zu jedem seiner Nachbarn festlegen
3. Ein Paket zusammenstellen, in dem alles steht, was er bisher gelernt hat
4. Dieses Paket an alle anderen Router senden und von allen Routern Pakete empfangen
5. Den kürzesten Pfad zu allen anderen Routern berechnen

Die vollständige Topologie wird an jeden Router verteilt. Dann kann der Algorithmus von Dijkstra angewendet werden, um den kürzesten Pfad zu allen anderen Routern zu ermitteln. Diese fünf Schritte werden im Folgenden ausführlicher dargestellt.

Ermitteln der benachbarten Router

Nach dem Start muss ein Router als Erstes ermitteln, wer seine Nachbarn sind. Hierzu sendet er ein spezielles HELLO-Paket auf jeder Punkt-zu-Punkt-Leitung. Der Router am anderen Ende muss eine Antwort zurücksenden, mit der er seinen Namen bekannt gibt. Diese Namen müssen global eindeutig sein, denn wenn ein entfernter Router später erfährt, dass drei Router mit *F* verbunden sind, muss er erkennen können, ob alle drei dasselbe *F* meinen.

Sind zwei oder mehr Router über eine Broadcast-Verbindung aneinander angeschlossen (z.B. Switch, Ring oder klassisches Ethernet), dann ist die Situation etwas komplizierter. ►Abbildung 5.11a zeigt ein Broadcast-LAN, mit dem die drei Router *A*, *C* und *F* direkt verbunden sind. Diese drei Router sind wiederum mit anderen Routern verbunden.

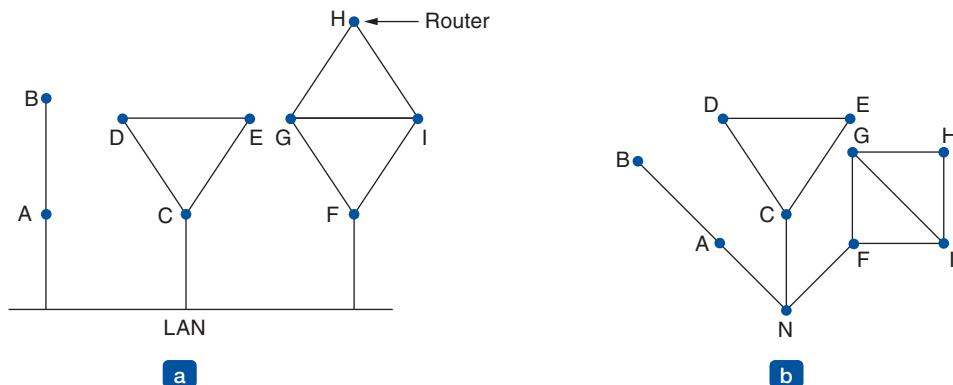


Abbildung 5.11: (a) Neun Router und ein Broadcast-LAN. (b) Der Graph für (a).

Das Broadcast-LAN verbindet je zwei angeschlossene Router. Die Modellierung der LANs als Punkt-zu-Punkt-Verbindungen würde jedoch die Größe der Topologie erhöhen und zu unnötigem Nachrichtenaustausch führen. Eine bessere Möglichkeit, das

LAN zu modellieren, ist das LAN selbst als Knoten zu betrachten (siehe ►Abbildung 5.11b). Hier haben wir einen neuen künstlichen Knoten N eingefügt, an den A , C und F angeschlossen sind. Es wird ein **designierter Router** im LAN ausgewählt, der die Rolle von N im Routing-Protokoll spielt. Die Tatsache, dass man im LAN von A nach C gelangen kann, ist hier durch den Pfad ANC dargestellt.

Festlegen der Verbindungskosten

Das Link-State-Routing setzt voraus, dass jede Verbindung eine Entferungs- oder Kostenmetrik zum Auffinden der kürzesten Pfade besitzt. Die Kosten zum Erreichen der Nachbarn können automatisch festgesetzt oder vom Netzbetreiber konfiguriert werden. Häufig werden die Kosten umgekehrt proportional zur Bandbreite der Verbindung festgelegt. Beispielsweise könnte 1-Gbit/s-Ethernet Kosten von 1 und 100-Mbit/s-Ethernet von 10 haben. Dadurch werden Pfade mit höherer Kapazität bevorzugt.

Wenn sich das Netz geografisch ausbreitet, könnte die Übertragungszeit der Verbindungen in die Kosten einbezogen werden, sodass Pfade über kürzere Verbindungen die bessere Wahl sind. Der direkteste Weg, diese Übertragungszeit zu ermitteln, ist das Aussenden eines speziellen ECHO-Pakets. Die andere Seite muss es sofort wieder zurücksenden. Durch Messen der Hin- und Rückreisezeit, geteilt durch zwei, kann der sendende Router die Übertragungszeit vernünftig abschätzen.

Link-State-Pakete erstellen

Nachdem die erforderlichen Informationen für den Austausch erfasst wurden, muss jeder Router im nächsten Schritt ein Paket mit allen Daten zusammenstellen. Das Paket beginnt mit der Identität des Senders, gefolgt von einer Sequenznummer und dem Alter (wird später beschrieben) sowie einer Liste der Nachbarn. Die Kosten zu jedem Nachbarn werden ebenfalls erfasst. ►Abbildung 5.12a zeigt ein Beispiel eines Netzes, wobei die Kosten an den Verbindungen vermerkt sind. Die entsprechenden Link-State-Pakete aller sechs Router sind in ►Abbildung 5.12b dargestellt.

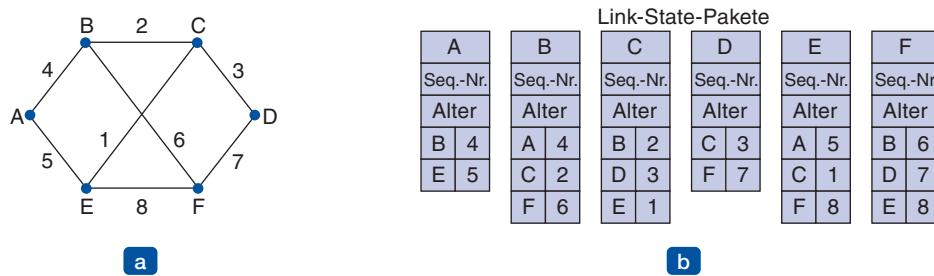


Abbildung 5.12: (a) Ein Netzwerk. (b) Die Link-State-Pakete für dieses Netz.

Das Erstellen von Link-State-Paketen ist einfach. Der schwierige Teil ist die Frage, wann sie erstellt werden sollen. Eine Möglichkeit ist, sie periodisch aufzubauen, d.h. in regelmäßigen Abständen. Eine Alternative ist die Erstellung, wenn ein bestimmtes wichtiges Ereignis eintritt, z.B. der Ausfall oder die Aufnahme einer Leitung oder eines benachbarten Routers oder eine erhebliche Änderung der Eigenschaften.

Link-State-Pakete verteilen

Der kniffligste Teil des Algorithmus ist die zuverlässige Verteilung der Link-State-Pakete. Alle Router müssen alle Link-State-Paket schnell und zuverlässig erhalten. Falls verschiedene Router unterschiedliche Versionen der Topologie verwenden, können die jeweils berechneten Wege Inkonsistenzen wie Schleifen, unerreichbare Rechner und andere Probleme enthalten.

Wir werden nun zuerst den grundlegenden Verteilungsalgorithmus beschreiben. Danach wird diese Darstellung etwas verfeinert. Das Basiskonzept ist die Anwendung von Fluten zur Verteilung der Link-State-Pakete an alle Router. Um das Ausmaß des Flutens unter Kontrolle zu halten, enthält jedes Paket eine Sequenznummer, die bei jedem neu ausgesendeten Paket erhöht wird. Die Router vermerken alle Paare (Quell-Router, Sequenznummer), die sie sehen. Kommt ein neues Link-State-Paket an, so wird es mit der bereits bekannten Paketliste verglichen. Ist es neu, dann wird es an alle Leitungen außer derjenigen, auf der es eingegangen ist, ausgegeben. Ist es ein Duplikat, wird es verworfen. Wenn die Sequenznummer eines Pakets niedriger als das höchste bisher erfasste ist, wird das Paket als veraltet abgelehnt.

Dieser Algorithmus wirft ein paar Probleme auf, die aber gelöst werden können. Erstens würde die Wiederverwendung von Sequenznummern zu Verwirrung führen. Dies kann durch die Verwendung von 32-Bit-Sequenznummern gelöst werden. Bei einem Link-State-Paket pro Sekunde würde es 137 Jahre dauern, bis die gleichen Nummern wieder auftauchen, sodass dieses Risiko ignoriert werden kann.

Zweitens verliert ein Router bei einem Absturz die Übersicht über seine Sequenznummern. Wenn er mit der Zählung wieder bei 0 beginnt, wird das nächste Paket, das dieser Router sendet, als Duplikat verworfen.

Drittens kann durch einen kleinen Fehler bei der Übertragung der Sequenznummer z.B. aus 4 eine 65 540 werden (ein 1-Bit-Fehler), und dann würden die Pakete 5 bis 65 540 als veraltet zurückgewiesen, da 65 540 als aktuelle Sequenznummer angenommen wird.

Die Lösung all dieser Probleme besteht darin, das Alter eines Pakets nach der Sequenznummer aufzunehmen und dieses pro Sekunde um 1 zu reduzieren. Ist das Alter gleich null, dann werden die Informationen von diesem Router verworfen. Kommt ein neues Paket z.B. alle zehn Sekunden an, „veraltet“ ein Paket nur, wenn ein Router ausfällt (oder sechs aufeinanderfolgende Pakete verloren gehen, was eher unwahrscheinlich ist). Das Feld *Age* wird von jedem Router auch während des anfänglichen Flutens um 1 reduziert, um sicherzustellen, dass kein Paket verloren geht und ewig weiterlebt (ein Paket mit dem Alter null wird verworfen).

Dieser Algorithmus kann durch einige Verfeinerungen robuster gemacht werden. Kommt ein Link-State-Paket zum Fluten in einem Router an, wird es nicht zur sofortigen Übertragung in die Warteschlange gestellt. Es wird vielmehr in einem Wartebereich abgestellt, in dem es eine Weile warten muss, falls weitere Verbindungen aufgebaut oder unterbrochen werden. Geht vor dessen Übertragung ein weiteres Link-State-Paket

von der gleichen Quelle ein, so werden die Sequenznummern der zwei Pakete verglichen. Stimmen sie überein, dann wird das Duplikat verworfen. Sind sie unterschiedlich, wird das ältere verworfen. Als Schutz vor Fehlern auf Verbindungen werden alle Link-State-Pakete bestätigt.

Die Datenstruktur von Router *B* des in Abbildung 5.12a dargestellten Netzes ist in ►Abbildung 5.13 gezeigt. Jede Zeile entspricht einem kürzlich angekommenen, aber noch nicht vollständig verarbeiteten Link-State-Paket. In der Tabelle werden der Ursprung des Pakets, die Sequenznummer, das Alter und die Daten erfasst. Zusätzlich gibt es Sende- und Bestätigungs-Flags für die drei Leitungen von *B* (zu *A*, *C* und *F*). Die Sende-Flags bedeuten, dass das Paket auf der angegebenen Leitung übertragen werden muss. Die Bestätigungs-Flags geben an, dass es dort bestätigt werden muss.

| Quelle | Seq.-Nr. | Alter | Sende-Flags | | | Bestätigungs-Flags | | | Daten |
|--------|----------|-------|-------------|---|---|--------------------|---|---|-------|
| | | | A | C | F | A | C | F | |
| A | 21 | 60 | 0 | 1 | 1 | 1 | 0 | 0 | |
| F | 21 | 60 | 1 | 1 | 0 | 0 | 0 | 1 | |
| E | 21 | 59 | 0 | 1 | 0 | 1 | 0 | 1 | |
| C | 20 | 60 | 1 | 0 | 1 | 0 | 1 | 0 | |
| D | 21 | 59 | 1 | 0 | 0 | 0 | 1 | 1 | |

Abbildung 5.13: Der Paketpuffer für Router *B* von Abbildung 5.12a.

In Abbildung 5.13 ist das Link-State-Paket von *A* direkt angekommen, deshalb muss es an *C* und *F* gesendet und für *A* bestätigt werden (wie in den Flag-Bits angegeben). Das Paket von *F* muss an *A* und *C* weitergeleitet und für *F* bestätigt werden.

Beim dritten Paket, das von *E* kommt, ist die Situation jedoch anders. Dieses Paket kommt zweimal an – einmal über *EAB* und einmal über *EFB*. Folglich muss es nur an *C* gesendet, aber sowohl für *A* als auch *F* bestätigt werden.

Geht ein Duplikat ein, während sich das Original noch im Puffer befindet, müssen einige Bits geändert werden. Kommt beispielsweise eine Kopie des Zustands von *C* über *F* an, bevor der vierte Eintrag in der Tabelle weitergeleitet wurde, werden die sechs Bits auf 100011 geändert, um anzudeuten, dass das Paket für *F* bestätigt, aber nicht dorthin übertragen werden muss.

Berechnung neuer Wege

Nachdem ein Router eine vollständige Menge an Link-State-Paketen angesammelt hat, kann er den Graphen für das gesamte Netz aufbauen, weil alle Verbindungen vertreten sind. Die Verbindungen werden zweimal dargestellt, je einmal für jede Richtung. Die verschiedenen Richtungen können sogar unterschiedliche Kosten haben. Die Berechnungen der kürzesten Pfade können dann dazu führen, dass es andere Pfade von Router *A* zu *B* als von *B* zu *A* gibt.

Jetzt kann der Algorithmus von Dijkstra lokal ausgeführt werden, um den kürzesten Pfad zu allen möglichen Zielen zu ermitteln. Die Ergebnisse dieses Algorithmus sagen dem Router, welche Verbindung zu verwenden ist, um das jeweilige Ziel zu erreichen. Diese Information wird in die Routing-Tabellen eingefügt und der Normalbetrieb wird wieder aufgenommen.

Verglichen mit Distanzvektoralgorithmus erfordert Link-State-Routing mehr Speicherplatz und Rechenzeit. Bei einem Netzwerk mit n Routern, die je k Nachbarn haben, ist der zum Ablegen der Eingangsdaten erforderliche Speicher proportional zu kn , was mindestens so groß ist wie eine Routing-Tabelle, die alle Ziele auflistet. Außerdem wächst die Berechnungszeit schneller als kn , selbst mit den effizientesten Datenstrukturen, was in großen Netzen ein Problem ist. Dennoch funktioniert Link-State-Routing in den meisten praktischen Anwendungen gut, weil es nicht am Problem der langsam Konvergenz leidet.

Link-State-Routing wird in der Praxis recht häufig verwendet. Deshalb wollen wir hier ein paar Worte zu Beispielprotokollen verlieren. Viele ISPs benutzen das **IS-IS**-Link-State-Protokoll (*Intermediate System-Intermediate System*; Oran, 1990). Es wurde für ein frühes Netzwerk, DECnet, entwickelt, später von der ISO zur Nutzung mit den OSI-Protokollen übernommen und dann geändert, damit es auch andere Protokolle – insbesondere IP – unterstützt. Das **OSPF**-Protokoll (*Open Shortest Path First*) ist das andere wichtige Link-State-Protokoll. Es wurde von IETF mehrere Jahre nach IS-IS entwickelt und übernahm viele der Innovationen von IS-IS. Zu diesen Innovationen zählen eine sich selbst stabilisierende Methode des Flutens von Link-State-Aktualisierungen, das Konzept eines designierten Routers in einem LAN sowie die Methode der Berechnung und Unterstützung von Parametern für die Pfadaufteilung, außerdem mehrere Metriken. Folglich sind sich IS-IS und OSPF sehr ähnlich. Der wichtigste Unterschied ist der, dass IS-IS Informationen über mehrere Protokolle der Vermittlungsschicht gleichzeitig verwalten kann (z.B. IP, IPX und AppleTalk). OSPF bietet diese Funktion nicht, die in großen Netzmumgebungen mit mehreren Protokollen ein Vorteil ist. Wir werden auf OSPF in Abschnitt 5.6.6 eingehen.

Ein allgemeiner Kommentar zu Routing-Algorithmen soll hier ebenfalls nicht fehlen. Link-State-, Distanzvektor- und andere Algorithmen beruhen darauf, dass die Berechnung der Wege an allen Routern stattfindet. Probleme mit der Hard- oder Software selbst bei einer kleinen Anzahl von Routern können sich bereits verheerend auf das Netzwerk auswirken. Behauptet z.B. ein Router, eine Verbindung zu besitzen, die er in Wirklichkeit nicht hat, oder vergisst er eine Verbindung, dann ist der Netzgraph falsch. Wenn ein Router beim Weiterleiten von Paketen versagt oder er die Pakete bei der Übertragung verstümmelt, dann funktioniert die Route nicht wie erwartet. Hat der Router schließlich keinen Speicherplatz mehr oder berechnet das Routing falsch, passieren unangenehme Dinge. Mit zunehmender Netzgröße, z.B. mit Hunderttausenden von Knoten, kann die Wahrscheinlichkeit, dass ein Router gelegentlich ausfällt, nicht mehr vernachlässigt werden. Hier muss man versuchen, den Schaden zu begrenzen. In Perlman (1988) werden diese Probleme und die möglichen Lösungen ausführlich diskutiert.

5.2.6 Hierarchisches Routing

Mit zunehmender Größe eines Netzes wachsen auch die Routing-Tabellen der Router proportional an. Dabei wird nicht nur im Router mehr Speicherplatz durch fortlaufend anwachsende Tabellen verbraucht, sondern auch mehr CPU-Zeit zu ihrer Durchsuchung und mehr Bandbreite zur Übertragung der zugehörigen Statusberichte. Irgendwann ist das Netz so groß, dass es nicht mehr sinnvoll ist, jeden Router einen Eintrag für alle anderen Router führen zu lassen. Folglich muss das Routing wie beim Telefonnetz hierarchisch durchgeführt werden.

Beim hierarchischen Routing werden die Router in **Regionen** unterteilt. Jeder Router weiß, wie er Pakete an Ziele innerhalb seiner Region leiten muss, die innere Struktur anderer Regionen kennt er aber nicht. Wenn verschiedene Netze verbunden werden, ist es ganz natürlich, die einzelnen Netze als eigenständige Regionen zu betrachten, um die eigenen Router nicht noch mit Informationen über fremde Topologien zu belasten.

Bei riesigen Netzen ist eine zweistufige Hierarchie meist ungenügend. In diesem Fall können die Regionen in Cluster, die Cluster in Zonen, die Zonen in Gruppen usw. unterteilt werden, bis irgendwann die Bezeichnungen ausgehen. Betrachten wir als Beispiel, wie ein Paket vom kalifornischen Berkeley nach Malindi in Kenia geleitet wird. Der Berkeley-Router kennt die Topologie innerhalb von Kalifornien, versendet aber alle überregionalen Pakete an den Router in Los Angeles. Der Router in Los Angeles kann Datenverkehr an andere inländische Router weiterleiten, sendet allen ausländischen Verkehr aber nach New York. Der Router in New York ist so programmiert, dass er alle an das Bestimmungsland gerichteten Pakete an den Router schickt, der für Auslandsdaten zuständig ist, in diesem Fall wahrscheinlich Nairobi. Schließlich arbeitet sich das Paket in Kenia wieder durch den Baum nach unten, bis es in Malindi angelangt ist.

► Abbildung 5.14 zeigt ein quantitatives Beispiel für das Routing in einer zweistufigen Hierarchie mit fünf Regionen. Die vollständige Routing-Tabelle für Router 1A weist 17 Einträge auf, wie in ► Abbildung 5.14b dargestellt. Beim hierarchischen Routing wie in ► Abbildung 5.14c gibt es wie vorhin Einträge für die lokalen Router, alle anderen Regionen wurden aber in einem einzigen Router zusammengefasst. Aller Datenverkehr für Region 2 fließt also über die Leitung 1B–2A. Der Rest des Fernverkehrs läuft über die Leitung 1C–3B. Das hierarchische Routing hat die Tabelle von 17 auf 7 Einträge reduziert. Wenn das Verhältnis der Anzahl von Regionen gegenüber der Anzahl von Routern in einer Region zunimmt, nimmt der Umfang der Tabelle im gleichen Verhältnis ab.

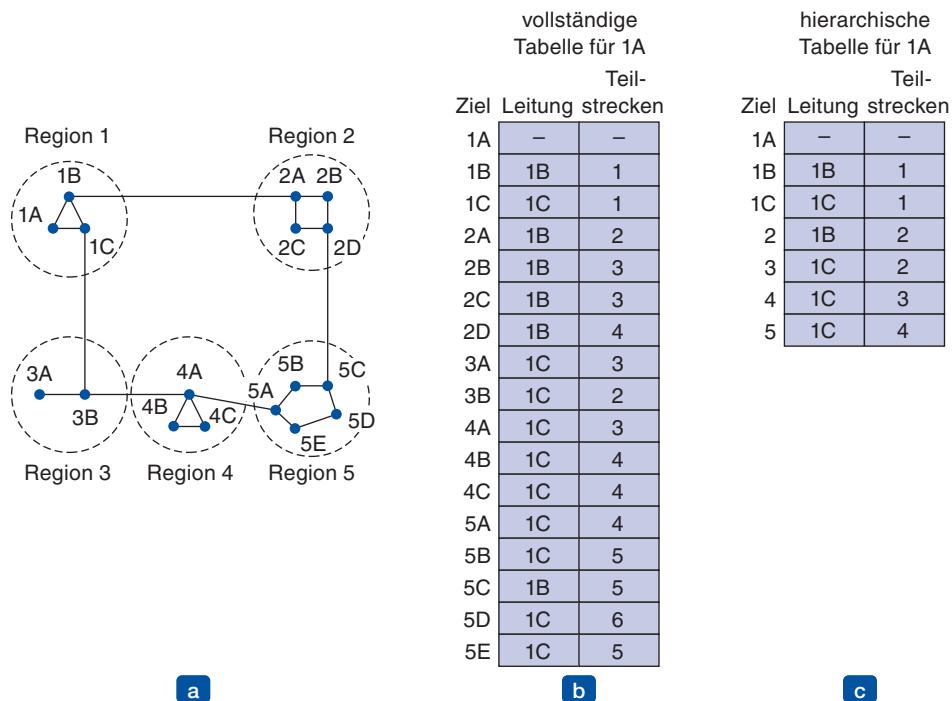


Abbildung 5.14: Hierarchisches Routing.

Leider fordert diese Platzersparnis aber ihren Preis, nämlich eine ansteigende Pfadlänge. Der beste Weg von 1A zu 5C verläuft über Region 2. Beim hierarchischen Routing fließen aber die gesamten Daten für Region 5 über Region 3, weil das für die meisten anderen Bestimmungsorte günstiger ist.

Wird ein einzelnes Netz zu groß, muss die Frage gestellt werden: Wie viele Ebenen muss die Hierarchie aufweisen? Man stelle sich beispielsweise ein Netz mit 720 Routern vor. Gäbe es keine Hierarchie, dann würde jeder Router 720 Einträge in seiner Routing-Tabelle benötigen. Wird das Netz in 24 Regionen mit je 30 Routern unterteilt, so benötigt jeder Router 30 lokale und 23 entfernte Einträge, also insgesamt 53 Einträge. Bei einer dreistufigen Hierarchie mit 8 Clustern zu je 9 Regionen mit 10 Routern benötigt jeder Router 10 Einträge für die anderen Router der eigenen Region, 8 Einträge für die Wege in die anderen Regionen im eigenen Cluster und 7 Einträge für die übrigen Cluster, also insgesamt 25 Einträge. Kamoun und Kleinrock (1979) haben entdeckt, dass die optimale Anzahl an Ebenen für ein Netz mit N Routern bei $\ln N$ liegt, wobei insgesamt $e \ln N$ Einträge pro Router nötig sind. Außerdem haben sie gezeigt, dass die durch ein hierarchisches Routing bedingte Steigerung der durchschnittlichen Pfadlänge klein genug ist, um sie vernachlässigen zu können.

5.2.7 Broadcast-Routing

Bei einigen Anwendungen müssen Hosts Nachrichten an viele oder sogar alle anderen Hosts versenden. Zum Beispiel funktionieren ein Dienst zum Verteilen von Wetterberichten, die Aktualisierung der Börsenkurse oder die Übertragung von Radiosendungen in Echtzeit vermutlich am besten, wenn die Information an alle Maschinen gesendet wird und diejenigen, die interessiert sind, lesen die Daten. Das gleichzeitige Senden eines Pakets an alle Ziele heißt **Broadcasting**. Hierfür wurden verschiedene Methoden vorgeschlagen.

Eine Broadcasting-Methode, die an das Netzwerk keine besonderen Anforderungen stellt, besteht darin, dass die Quelle ein bestimmtes Paket einfach an jedes Ziel sendet. Diese Methode ist nicht nur langsam und verschwendet Bandbreite, sondern die Quelle muss auch noch eine Liste aller Ziele führen. Diese Methode ist in der Praxis eigentlich nicht wünschenswert, auch wenn sie häufig eingesetzt wird.

Eine Verbesserung ist das **Multidestination-Routing** (Routing an mehrere Ziele), wobei jedes Paket eine Liste von Zielen oder ein Bitmuster enthält, das die gewünschten Ziele bezeichnet. Der empfangende Router überprüft alle Ziele, um die benötigten Leitungen zu bestimmen. (Eine Ausgabeleitung wird dann verwendet, wenn sie den besten Pfad zu mindestens einem der Ziele darstellt.) Der Router erzeugt für jede zu verwendende Ausgabeleitung eine neue Paketkopie und fügt dem Paket nur jene Ziele hinzu, die von dieser Leitung aus zu erreichen sind. Damit werden die Ziele auf die Leitungen aufgeteilt. Nach einer ausreichend großen Anzahl von Teilstrecken enthält jedes Paket nur noch ein Ziel und wird wie ein ganz normales Paket behandelt. Das Multidestination-Routing ähnelt einer Methode mit einzeln adressierten Paketen, abgesehen davon, dass bei mehreren Paketen auf dem gleichen Pfad eines davon den vollen Fahrpreis zu zahlen hat, während die übrigen umsonst mitreisen. Die Netzbänderbreite wird daher effizienter ausgenutzt. Dieses Schema verlangt jedoch immer noch, dass die Quelle alle Ziele kennt. Außerdem ist es für einen Router genauso viel Aufwand festzulegen, wohin ein Multidestination-Paket zu senden ist, wie für viele verschiedene Pakete.

Wir haben bereits eine bessere Technik für Broadcast-Routing kennengelernt: Fluten. Wird Fluten mit einer Sequenznummer pro Quelle implementiert, dann werden Verbindungen effektiv genutzt, wobei Router eine relativ einfache Entscheidungsregel anwenden. Obwohl Fluten für die Punkt-zu-Punkt-Kommunikation schlecht geeignet ist, erweist es sich beim Broadcasting als gute Lösung. Es stellt sich jedoch heraus, dass wir das Verfahren noch weiter verbessern können, sobald der kürzeste Pfad für reguläre Pakete berechnet ist.

Das Konzept für **Reverse Path Forwarding** (Weiterleitung auf dem umgekehrten Pfad) ist elegant und bemerkenswert einfach, nachdem erst einmal jemand auf die Idee gekommen ist (Dalal und Metcalfe, 1978). Erreicht ein Broadcast-Paket einen Router, überprüft dieser, ob das Paket auf der Verbindung angekommen ist, die normalerweise für die Übertragung von Paketen zum Absender der Rundsendung verwendet wird. Trifft das zu, dann ist es sehr wahrscheinlich, dass das Broadcast-Paket auf dem besten Pfad vom Absender angekommen ist und daher die erste ankommende Kopie darstellt. Ist

dies der Fall, kopiert der Router das Paket auf alle Verbindungen mit Ausnahme der Verbindung, auf der es ankam. Wenn das Paket aber über eine Verbindung einging, die nicht die für die betreffende Quelle bevorzugte Verbindung ist, wird das Paket als vermutliches Duplikat verworfen.

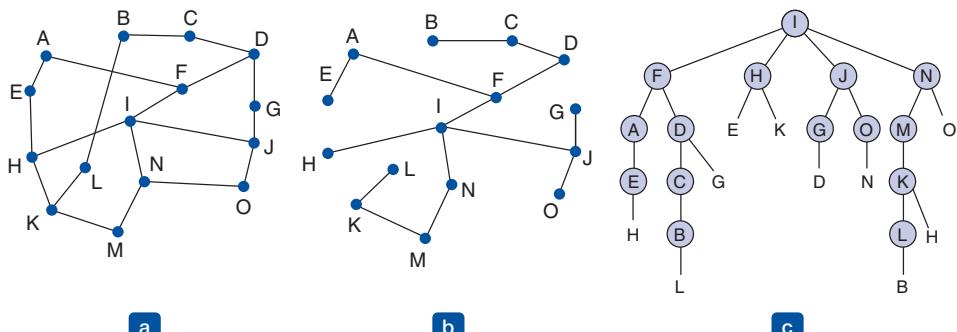


Abbildung 5.15: Reverse Path Forwarding: (a) Netzwerk, (b) Quelle-Senke-Baum, (c) der durch Reverse Path Forwarding erstellte Baum.

Ein Beispiel für Reverse Path Forwarding wird in ▶ Abbildung 5.15 dargestellt. Teil (a) ist ein Netzwerk, (b) zeigt einen Quelle-Senke-Baum für Router *I* dieses Netzes und (c) die Funktionsweise des Reverse-Path-Algorithmus. Auf der ersten Teilstrecke sendet *I* Pakete an *F, H, J* und *N* (zweite Reihe des Baums). Diese Pakete kommen alle auf dem bevorzugten Pfad zu *I* an (unter der Annahme, dass der bevorzugte Pfad im Quelle-Senke-Baum liegt), dies wird in der Abbildung durch einen Kreis um die Buchstaben gekennzeichnet. Auf der zweiten Teilstrecke werden acht Pakete erzeugt, je zwei von den Routern, die ein Paket auf der ersten Teilstrecke empfangen haben. Alle acht Pakete kommen bei bisher noch nicht besuchten Routern an und fünf gehen auf der bevorzugten Leitung ein. Von den sechs auf der dritten Teilstrecke erzeugten Paketen kommen nur drei auf dem bevorzugten Pfad an (*C, E* und *K*), die anderen sind Duplikate. Nach fünf Teilstrecken und 24 Paketen endet das Broadcasting im Gegensatz zu vier Teilstrecken und 14 Paketen, wenn der Quelle-Senke-Baum exakt durchlaufen worden wäre.

Der wesentliche Vorteil beim Reverse Path Forwarding liegt darin, dass die Methode effizient und gleichzeitig leicht zu implementieren ist. Es werden genau wie beim Fluten Broadcast-Pakete über jede Verbindung nur einmal in jede Richtung ausgesandt, doch müssen hier die Router lediglich wissen, wie alle Ziele erreicht werden, und sich keine Sequenznummern merken (oder andere Mechanismen anwenden, um die Flut zu stoppen) oder eine Liste aller Ziele im Paket mitführen.

Unser letzter Broadcasting-Algorithmus verbessert das Verhalten von Reverse Path Forwarding. Dieser Algorithmus verwendet explizit den Quelle-Senke-Baum – oder einen anderen geeigneten Spannbaum – für den Router, der die Broadcast-Übertragung einleitet. Ein **Spannbaum** ist eine Teilmenge des Netzes, der alle Router enthält, aber keine Schleifen aufweist. Quelle-Senke-Bäume sind Spannbäume. Wenn jeder Router weiß, welche seiner Leitungen zum Spannbaum gehören, kann er ein ankommendes Broadcast-Paket auf alle Spannbaumleitungen kopieren, außer auf diejenige,

auf der es angekommen ist. Diese Methode nutzt die Bandbreite hervorragend, da nur die absolute Mindestzahl an Paketen erzeugt wird. Wenn zum Beispiel in Abbildung 5.15 der Quelle-Senke-Baum aus Teil (b) als Spannbaum verwendet wird, dann wird das Broadcast-Paket mit mindestens 14 Paketen gesendet. Das einzige Problem ist hier, dass jeder Router einen geeigneten Spannbaum kennen muss, damit die Methode anwendbar ist. Manchmal ist diese Information verfügbar (z.B. beim Link-State-Routing kennen alle Router die vollständige Topologie, sodass sie einen Spannbaum berechnen können), aber manchmal auch nicht (z.B. beim Distanzvektoralgorithmus).

5.2.8 Multicast-Routing

Bei manchen Anwendungen wie einem Multiplayer-Spiel oder der Live-Übertragung eines Sportereignisses, das an vielen Standorten betrachtet wird, werden Pakete zu mehreren Empfängern gesendet. Falls die Gruppe nicht sehr klein ist, ist das Senden eines einzelnen Pakets zu jedem Empfänger teuer. Andererseits ist das Broadcasting eines Paktes Verschwendungen, falls die Gruppe beispielsweise auf 1 000 Rechnern in einem Netz mit Millionen von Knoten besteht, sodass die meisten Empfänger nicht an der Nachricht interessiert sind (oder noch schlimmer, weil sie zwar brennend daran interessiert wären, aber die Nachricht nicht sehen sollen). Das heißt, wir benötigen eine Methode, um Nachrichten gezielt an bestimmte Gruppen zu versenden, die zwar viele Mitglieder haben können, aber im Vergleich zum Gesamtnetz klein sind.

Das Übertragen einer Nachricht an eine solche Gruppe heißt **Multicasting** und der zugrunde liegende Routing-Algorithmus ist das **Multicast-Routing**. Alle Multicasting-Methoden benötigen ein Verfahren, um Gruppen anzulegen und zu löschen sowie um festzustellen, welche Router Mitglieder einer Gruppe sind. Wie diese Aufgaben durchgeführt werden, kümmert den Routing-Algorithmus nicht. Für den Moment nehmen wir an, dass jede Gruppe über eine Multicast-Adresse identifizierbar ist und dass jeder Router weiß, zu welcher Gruppe er gehört. Wir werden auf das Thema der Gruppenmitgliedschaft zurückkommen, wenn wir die Vermittlungsschicht des Internets in Abschnitt 5.6 beschreiben.

Methoden zum Multicast-Routing bauen auf den Broadcast-Routing-Verfahren auf, die wir bereits untersucht haben. Sie senden Pakete entlang der Spannbäume, um diese allen Mitgliedern der Gruppe zuzustellen, wobei die Bandbreite effizient ausgenutzt wird. Welcher Spannbaum jedoch der beste ist, hängt davon ab, ob die Gruppe „dicht“ (*dense*) ist, das heißt, die Empfänger sind fast über das ganze Netz verteilt, oder „spärlich“ (*sparse*), wenn ein großer Teil des Netzes nicht zur Gruppe gehört. In diesem Abschnitt werden wir uns beide Fälle ansehen.

Falls die Dichte der Gruppe hoch ist, dann ist Broadcasting ein guter Anfang, weil dadurch die Pakete effizient in alle Teile des Netzes kommen. Doch die Broadcast-Übertragung wird auch einige Router erreichen, die keine Mitglieder der Gruppe sind, was Verschwendungen ist. Deering und Cheriton (1990) haben sich daher folgende Lösung erdacht: der Broadcast-Spannbaum wird gestutzt (*prune*), indem alle Verbindungen gelöscht werden, die nicht zu Mitgliedern führen. Das Ergebnis ist ein effizienter Multicast-Spannbaum.

Betrachten wir als Beispiel zwei Gruppen, 1 und 2, im Netzwerk von ▶ Abbildung 5.16a. Einige Router sind an Hosts angeschlossen, die zu einem oder zu beiden dieser Gruppen gehören, wie in der Abbildung angegeben. Ein Spannbaum für den Router ganz links ist in ▶ Abbildung 5.16b gezeigt. Dieser Baum kann für Broadcasting verwendet werden, aber ist für Multicasting überdimensioniert, wie man an den zwei gestützten Versionen sehen kann, die in Teil (c) und (d) abgebildet sind. In ▶ Abbildung 5.16c wurden alle Verbindungen entfernt, die nicht zu Hosts führen, die Mitglieder von Gruppe 1 sind. Das Ergebnis ist der Multicast-Spannbaum für den Router links außen, wenn an Gruppe 1 gesendet wird. Pakete werden nur entlang dieses Spannbaums weitergeleitet, welcher effizienter ist als der Broadcast-Baum, da es hier 7 anstatt 10 Verbindungen gibt. ▶ Abbildung 5.16d zeigt den Multicast-Spannbaum nach dem Stutzen für Gruppe 2. Auch dieser ist effizient, er hat jetzt nur 5 Verbindungen. Hieraus wird außerdem ersichtlich, dass unterschiedliche Multicast-Gruppen auch unterschiedliche Spannbäume haben.

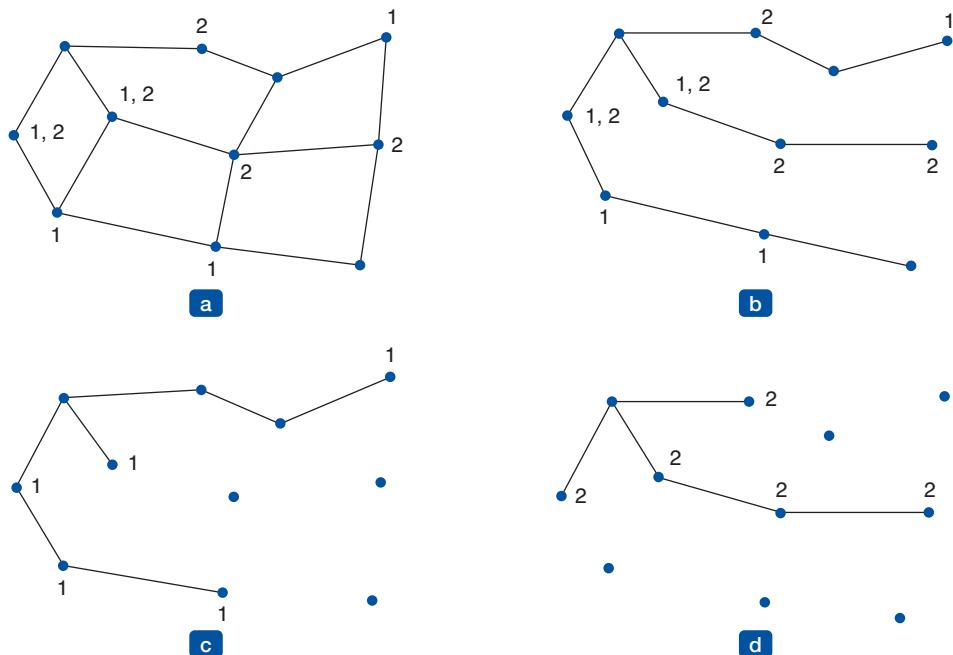


Abbildung 5.16: (a) Netzwerk. (b) Spannbaum für den Router ganz links. (c) Multicast-Baum für Gruppe 1. (d) Multicast-Baum für Gruppe 2.

Es gibt verschiedene Möglichkeiten, einen Spannbaum zu stutzen. Die einfachste kann beim Link-State-Routing eingesetzt werden, wenn jeder Router sowohl die vollständige Topologie kennt als auch weiß, welcher Host zu welcher Gruppe gehört. Jeder Router kann dann für jeden Sender der betreffenden Gruppe seinen eigenen gestützten Spannbaum aufbauen, indem wie üblich ein Quelle-Senke-Baum für den Sender konstruiert wird und anschließend alle Verbindungen entfernt werden, die keine Gruppenmitglieder mit dem Senken-Knoten verbinden. **MOSPF** (Multicast OSPF) ist ein Beispiel für solch ein Link-State-Protokoll (Moy, 1994).

Beim Distanzvektoralgorithmus kann eine andere Strategie zum Stutzen der Bäume verfolgt werden. Im Prinzip handelt es sich um das Reverse Path Forwarding. Ist ein Router ohne Hosts an einer bestimmten Gruppe interessiert und bestehen keine Verbindungen zu anderen Routern, so antwortet er mit einer PRUNE-Nachricht. Damit weist er den Nachbarknoten an, der diese Nachricht gesendet hat, ihm von diesem Sender für die betreffende Gruppe keine Multicast-Pakete mehr zu schicken. Hat ein Router ohne Gruppenmitglieder unter seinen eigenen Hosts solche Nachrichten auf all seinen Leitungen erhalten, auf denen er die Multicast-Pakete gesendet hat, kann er ebenfalls mit einer PRUNE-Nachricht antworten. Auf diese Weise wird der Spannbaum rekursiv gestutzt. **DVMRP** (*Distance Vector Multicast Routing Protocol*) ist ein Beispiel eines Multicast-Routing-Protokolls, das auf diese Weise arbeitet (Waitzman et al., 1988).

Das Stutzen führt zu effizienten Spannbäumen, die nur die Verbindungen benutzen, die tatsächlich benötigt werden, um die Gruppenmitglieder zu erreichen. Ein potenzieller Nachteil ist, dass dieses Verfahren für den Router jede Menge Arbeit mit sich bringt, insbesondere in großen Netzen. Angenommen, ein Netz hat n Gruppen mit je durchschnittlich m Knoten. Für jeden Router und jede Gruppe müssen m gestutzte Spannbäume gespeichert werden. Das sind insgesamt mn Bäume. Abbildung 5.16c zeigt als Beispiel den Spannbaum für den Router ganz links, der an Gruppe 1 sendet. Der Spannbaum für den ganz rechts stehenden Router zum Senden an Gruppe 1 (nicht abgebildet) sähe ganz anders aus, da die Pakete direkt an die Gruppenmitglieder geleitet würden, anstatt den Weg über die linke Seite des Graphen zu nehmen. Dies wiederum bedeutet, dass Router die Pakete, welche für Gruppe 1 bestimmt sind, in unterschiedliche Richtungen weiterleiten müssen, je nachdem, welcher Knoten an die Gruppe sendet. In Umgebungen mit vielen großen Gruppen mit vielen Sendern bedeutet das ein beträchtliches Speichervolumen.

Ein alternativer Entwurf basiert auf sogenannten **Core-Based Trees** (wurzelbasierter Baum), der einen einzigen Spannbaum für die Gruppe berechnet (Ballardie et al., 1993). Alle Router einigen sich auf eine Wurzel (genannt **Core** oder **Rendezvous-Punkt**) und bauen den Baum auf, indem jedes Mitglied ein Paket zur Wurzel sendet. Der Baum ist die Vereinigung der Pfade, auf denen diese Pakete gereist sind. ►Abbildung 5.17a zeigt einen Core-Based Tree für Gruppe 1. Möchte ein Sender an diese Gruppe senden, dann schickt er ein Paket an den Core. Wenn das Paket dort ankommt, wird es den Baum hinab weitergeleitet. Dies ist in ►Abbildung 5.17b für den Sender auf der rechten Seite des Netzes zu sehen. Zur Optimierung der Performanz müssen Pakete, die für die Gruppe bestimmt sind, den Core nicht erreichen, bevor sie per Multicast weitergeleitet werden. Sobald ein Paket den Baum erreicht, kann es sowohl nach oben zur Wurzel ebenso wie hinunter zu allen Zweigen weitergeleitet werden. Dies ist für den Sender oben in Abbildung 5.17b dargestellt.

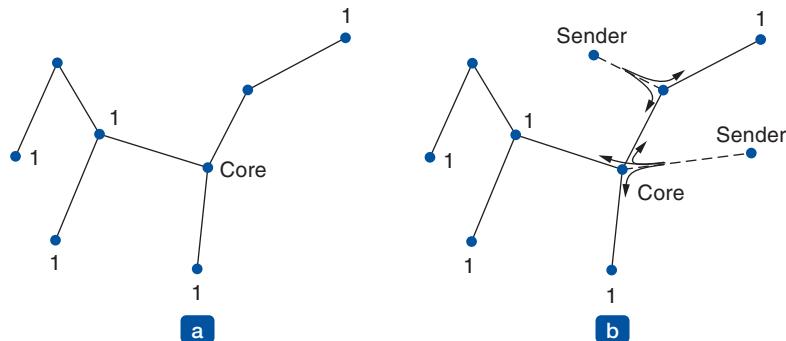


Abbildung 5.17: (a) Core-Based Tree für Gruppe 1. (b) Senden zu Gruppe 1.

Ein gemeinsam genutzter Baum ist nicht für alle Quellen optimal. In Abbildung 5.17b zum Beispiel erreicht das Paket vom rechten Sender das Gruppenmitglied ganz oben rechts nicht direkt, sondern über den Core in drei Teilstrecken. Wie ineffizient dieses Vorgehen ist, hängt davon ab, wo sich Core und Sender befinden. Meistens ist diese Ineffizienz jedoch vertretbar, wenn der Core inmitten der Sender liegt. Gibt es nur einen einzigen Sender, wie bei einem Videostrom, der an eine Gruppe übertragen wird, dann setzt man am besten den Sender als Core ein.

Hervorzuheben ist auch, dass gemeinsam benutzte Bäume wesentliche Einsparungen bei den Speicher Kosten, den gesendeten Nachrichten und der Rechenzeit bringen können. Jeder Router muss nur einen Baum pro Gruppe anstatt m Bäume vorhalten. Außerdem tragen Router, die nicht zum Baum gehören, nichts zur Unterstützung der Gruppe bei. Daher werden Ansätze mit gemeinsam genutzten Bäumen wie Core-Based Trees innerhalb von bekannten Protokollen wie **PIM** (*Protocol Independant Multicast*; Fenner et al., 2006) eingesetzt, wenn via Multicasting an spärlich besetzte Gruppen im Internet gesendet werden soll.

5.2.9 Anycast-Routing

Bislang haben wir Zustellungsmodelle betrachtet, bei denen eine Quelle zu einem einzelnen Ziel sendet (**Unicast**), zu allen Zielen (**Broadcast**) und zu einer Gruppe von Zielen (**Multicast**). Ein weiteres Modell ist manchmal ebenfalls nützlich: **Anycast**. Hier wird ein Paket an das am nächsten gelegene Gruppenmitglied gesendet (Partridge et al., 1993). Verfahren, die diese Pfade finden, werden **Anycast-Routing** genannt.

Warum wollen wir Anycasting überhaupt? Manchmal stellen Knoten einen Dienst zur Verfügung, wie die Tageszeit oder die Verteilung von Inhalten, bei dem es nur darauf ankommt, dass man die richtige Information bekommt, nicht aber, welcher Knoten dafür kontaktiert wird – jeder Knoten passt. Im Internet wird Anycasting beispielsweise als Teil von DNS verwendet, wie wir in *Kapitel 7* noch sehen werden.

Glücklicherweise müssen wir keine neuen Routing-Verfahren für Anycasting erfinden, weil normale Distanzvektoralgorithmen und Link-State-Routing auch Anycast-Routen erzeugen können. Nehmen wir an, wir wollen Anycasting auf die Mitglieder von

Gruppe 1 anwenden. Anstatt allen unterschiedliche Adressen zu geben, bekommen alle die Adresse „1“. Der Distanzvektoralgorithmus verteilt Vektoren wie üblich und die Knoten wählen den kürzesten Pfad zu Ziel 1. Dies führt dazu, dass ein Knoten zur nächstgelegenen Instanz von Ziel 1 sendet. Die Routen sind in ► Abbildung 5.18a dargestellt. Diese Prozedur funktioniert, weil das Routing-Protokoll nicht weiß, dass die es mehrere Instanzen von Ziel 1 gibt. Tatsächlich hält es alle Instanzen von Knoten 1 für ein und denselben Knoten, wie in der Topologie in ► Abbildung 5.18b zu sehen ist.

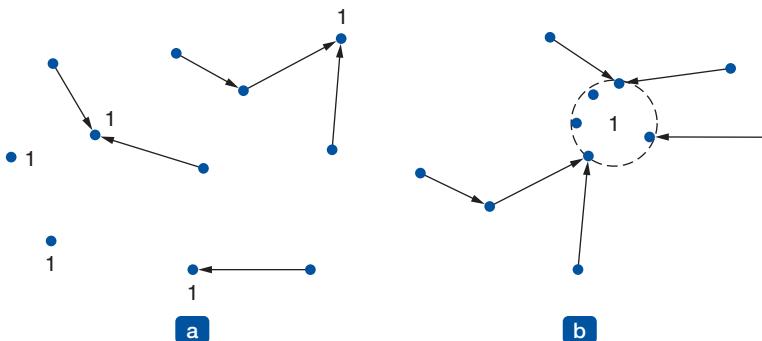


Abbildung 5.18: (a) Anycast-Routen zu Gruppe 1. (b) Topologie aus Sicht des Routing-Protokolls.

Das Verfahren funktioniert auch beim Link-State-Routing, obwohl hier außerdem bedacht werden muss, dass das Routing-Protokoll keine scheinbar kurzen Pfade finden muss, die durch Knoten 1 hindurchführen. Dies würde Sprünge durch den Hyperraum gleichkommen, da die Instanzen von Knoten 1 in Wirklichkeit Knoten sind, die sich in verschiedenen Teilen des Netzes befinden. Tatsächlich machen Link-State-Protokolle bereits diese Unterscheidung zwischen Routern und Hosts. Wir haben diese Tatsache jedoch früher unter den Teppich gekehrt, weil es für unsere Untersuchung nicht notwendig war.

5.2.10 Routing für mobile Hosts

Millionen von Leuten benutzen Computer unterwegs, von wirklich mobilen Situationen mit kabellosen Geräten in sich bewegenden Autos bis hin zu nomadenhaften Situationen, bei denen Laptops an vielen unterschiedlichen Orten benutzt werden. Wir werden den Begriff **mobiler Host** verwenden, wenn wir eine der beiden Kategorien meinen, im Unterschied zum stationären Host, der sich niemals bewegt. Die Menschen wollen zunehmend verbunden bleiben, wo auch immer in der Welt sie sich gerade befinden, und zwar genauso einfach, als wären sie zu Hause. Diese mobilen Hosts führen zu einer neuen Komplikation: Um ein Paket an einen mobilen Host weiterzuleiten, muss das Netz ihn zuerst einmal finden.

Das Modell der Welt, das wir betrachten werden, ist eines, bei dem wir davon ausgehen, dass alle Hosts einen dauerhaften **Heimatstandort** (*home location*) haben, der sich nie ändert. Jeder Host besitzt auch eine ständige Heimatadresse, die zur Ermittlung seines Heimatstandorts herangezogen wird, ähnlich wie beim Telefonnetz 0049-89-12356572

bedeutet, dass 0049 für Deutschland (Ländercode) und 89 für München steht. Das Ziel des Routings bei Systemen mit mobilen Hosts ist, die Möglichkeit zu schaffen, dass Pakete an mobile Hosts unter Verwendung ihrer festen Heimatadresse übertragen werden, gleichgültig, wo sie sich gerade befinden. Zuerst muss man sie natürlich ausfindig machen.

Bevor wir weitergehen, wollen wir noch ein paar Betrachtungen zu diesem Modell anstellen. Ein anderes Modell müsste Routen neu berechnen, wenn sich der mobile Host bewegt und sich damit die Topologie ändert. Wir könnten dann einfach die Routing-Verfahren verwenden, die wir vorne in diesem Abschnitt beschrieben haben. Dies würde allerdings bei wachsender Zahl von mobilen Hosts bald dazu führen, dass das gesamte Netzwerk ununterbrochen damit beschäftigt ist, neue Routen zu berechnen. Die Verwendung von Heimatadressen verringert diese Belastung erheblich.

Eine andere Alternative wäre, Mobilität oberhalb der Vermittlungsschicht zur Verfügung zu stellen, was heute typischerweise bei Laptops geschieht. Wird ein Laptop zu einem neuen Internetstandort bewegt, dann bekommt es eine neue Netzadresse. Es gibt keine Beziehung zwischen der alten und der neuen Adresse; das Netz weiß nicht, dass sie zum gleichen Laptop gehören. Bei diesem Vorgehen kann das Laptop zwar zum Webrowsing benutzt werden, aber andere Hosts können ihm keine Pakete schicken (zum Beispiel bei einem ankommenden Anruf), ohne einen standortbezogenen Dienst auf höheren Schichten aufzubauen, zum Beispiel nach dem Standortwechsel die *erneute* Anmeldung bei Skype. Zudem können Verbindungen nicht gehalten werden, während sich der Host bewegt; stattdessen müssen neue Verbindungen aufgebaut werden. Durch die Behandlung der Mobilität auf der Vermittlungsschicht können diese Probleme behoben werden.

Die Grundidee beim mobilen Routing im Internet und in zellulären Netzen ist, dass der mobile Host einem Host am Heimatstandort mitteilt, wo er sich gerade befindet. Dieser Host, der auf Anweisung des mobilen Hosts agiert, heißt **Heimatagent** (*home agent*). Sobald dieser weiß, wo sich der mobile Host aktuell aufhält, kann er Pakete so weiterleiten, dass sie den mobilen Host erreichen.

► Abbildung 5.19 zeigt mobiles Routing im Einsatz. Ein Sender in Seattle im Nordwesten der USA möchte ein Paket an einen Host senden, der sich normalerweise in New York befindet. Uns interessiert nun der Fall, wenn der mobile Host nicht zu Hause, sondern zum Beispiel zeitweilig in San Diego ist.

Der mobile Host in San Diego muss eine lokale Netzadresse erlangen, bevor er das Netz nutzen kann. Dies geschieht auf die übliche Art und Weise, in der Hosts Netzwerkadressen bekommen; wir werden dies für das Internet noch später in diesem Kapitel behandeln. Die lokale Adresse heißt **Care-of-Adresse**. Sobald der mobile Host diese Adresse hat, kann er seinem Heimatagent mitteilen, wo er sich gerade aufhält. Dazu sendet er eine Registrierungsnachricht mit der Care-of-Adresse zum Heimatagenten (Schritt 1). Diese Nachricht wird in Abbildung 5.19 durch eine gestrichelte Linie dargestellt. Damit soll angedeutet werden, dass dies eine Steuerungsnachricht, keine Datennachricht ist.

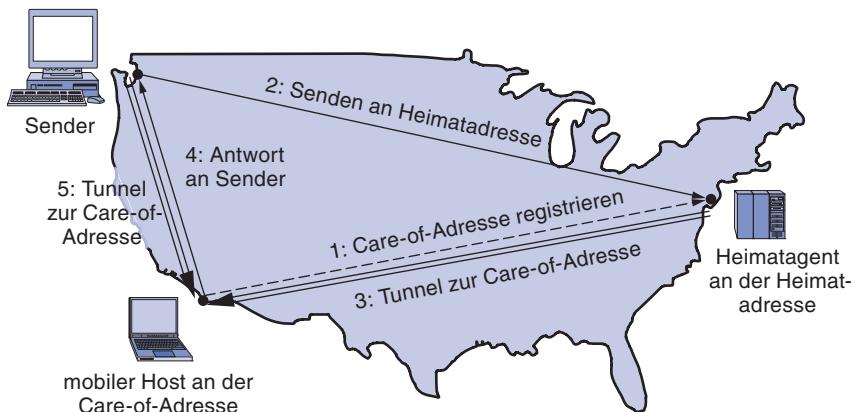


Abbildung 5.19: Paket-Routing für mobile Hosts.

Als Nächstes übermittelt der Sender ein Datenpaket zum mobilen Host, indem er dessen permanente Adresse benutzt (Schritt 2). Das Paket wird also zunächst durch das Netz zum Heimatstandort des Hosts geleitet. In New York fängt der Heimatagent das Paket ab, weil der mobile Host nicht zu Hause ist. Dann umhüllt oder **verkapselt** er das Paket mit einem neuen Header und sendet dieses Bündel zur Care-of-Adresse (Schritt 3). Dieser Mechanismus wird **Tunneling** genannt. Tunneling spielt im Internet eine wichtige Rolle, deshalb werden wir es uns später noch im Detail ansehen.

Kommt das gekapselte Paket bei der Care-of-Adresse an, dann entpackt es der mobile Host und erhält so das Paket vom Sender. Der mobile Host sendet dann sein Antwortpaket direkt zum Sender (Schritt 4). Die Gesamtroute wird **Triangle-Routing** genannt, da sie einen Umweg darstellt, falls der entfernte Standort weit vom Heimatstandort entfernt ist. Als Teil von Schritt 4 erfährt der Sender die aktuelle Care-of-Adresse. Nachfolgende Pakete können nun direkt zum mobilen Host geleitet werden, indem sie per Tunneling zur Care-of-Adresse geschickt werden (Schritt 5), der Heimatstandort wird also völlig umgangen. Falls aus irgendeinem Grund die Verbindung abbricht, wenn sich das mobile Gerät bewegt, kann immer die Heimatadresse verwendet werden, um das mobile Gerät zu erreichen.

Ein wichtiger Aspekt, den wir in dieser Beschreibung ausgelassen haben, ist die Sicherheit. Wenn ein Host oder Router eine Nachricht der Form „Von jetzt an bitte alle E-Mails für Stefanie an mich senden“ bekommt, hat er möglicherweise eine Reihe von Fragen dazu, wer da überhaupt mit ihm spricht und ob dies wirklich eine gute Idee ist. Daher sind Sicherheitsinformationen in den Nachrichten enthalten, sodass deren Gültigkeit mithilfe kryptografischer Protokolle (siehe Kapitel 8) überprüft werden kann.

Es gibt viele Variationen des mobilen Routings. Das obige Verfahren ist auf IPv6-Mobilität abgestimmt, also auf die Art der Mobilität, die im Internet (Johnson et al., 2004) und als Teil von IP-basierten zellulären Netzen wie UMTS verwendet wird. Wir haben den Sender aus Gründen der Einfachheit als stationären Knoten dargestellt, aber der Entwurf lässt es zu, dass beide Knoten mobile Hosts sind. Alternativ kann der Host Teil eines mobilen Netzes sein, zum Beispiel ein Computer in einem Flugzeug. Es gibt

Erweiterungen des Grundschemas zur Unterstützung mobiler Netze, bei denen die Host nichts mehr zu tun haben (Devarapalli et al., 2005).

Einige Verfahren benutzen analog zum Heimatagenten einen sogenannten Fremdagente (*foreign agent*) am entfernten Standort bzw. analog zum VLR (*Visitor Location Register*) in zellulären Netzen. In aktuelleren Verfahren jedoch wird der Fremdagente nicht benötigt; mobile Hosts agieren als ihre eigenen Fremdagente. In beiden Fällen hat nur eine kleine Anzahl Hosts (z.B. der mobile Host, der Heimatagent und die Sender) Kenntnis vom momentanen Standort des mobilen Hosts, sodass viele Router in einem großen Netz ihre Routen nicht neu berechnen müssen.

Für weitere Informationen zum mobilen Routing verweisen wir auf Perkins (1998, 2002) sowie Snoeren und Balakrishnan (2000).

5.2.11 Routing in Ad-hoc-Netzen

Bislang haben wir gesehen, wie Routing funktioniert, wenn die Hosts mobil, aber die Router stationär sind. Ein noch extremerer Fall ist es, wenn selbst die Router mobil sind. Dazu gehören zum Beispiel Situationen bei Rettungskräften in einem Erdbebengebiet, einer Schiffsflotte auf hoher See oder einer Versammlung von Personen mit Laptops in einem Gebiet ohne IEEE 802.11.

In all diesen und noch anderen Fällen kommuniziert jeder Knoten drahtlos und agiert sowohl als Host als auch als Router. Netze aus Knoten, die sich einfach zufällig in der Nähe voneinander befinden können, werden als **Ad-hoc-Netze** oder **MANETs** (*Mobile Ad hoc NETwork*) bezeichnet. Untersuchen wir diese nun kurz. Weitere Informationen finden Sie in Perkins (2001).

Ad-hoc-Netze unterscheiden sich von Kabelnetzen darin, dass die Topologie plötzlich über Bord geworfen wird. Knoten können kommen und gehen oder mit einem Mal an neuen Orten auftauchen. Verfügt ein Router in einem Kabelnetz über einen gültigen Pfad zu einem Ziel, bleibt dieser Pfad gültig – vorbehaltlich eines Systemausfalls, der hoffentlich selten auftritt. Bei einem Ad-hoc-Netzwerk kann sich die Topologie ständig verändern, sodass sich der angestrebte Pfad sowie die Gültigkeit von Pfaden spontan ohne Vorwarnung ändern können. Es versteht sich, dass durch diese Umstände das Routing in Ad-hoc-Netzen anspruchsvoller als das Routing in stationären Netzen wird.

Für Ad-hoc-Netze wurden viele, viele Routing-Algorithmen vorgeschlagen. Da Ad-hoc-Netze in der Praxis im Vergleich zu mobilen Netzwerken bisher wenig angewendet wurden, ist unklar, welche dieser Protokolle am nützlichsten sind. Als Beispiel werden wir uns einen der bekanntesten Routing-Algorithmen ansehen: **AODV** (*Ad hoc On-demand Distance Vector*, Ad-hoc-Distanzvektor nach Bedarf), der in Perkins und Royer (1999) beschrieben wird. Es ist ein Verwandter des Distanzvektoralgorithmus, der auf den Einsatz in einer mobilen Umgebung abgestimmt wurde, in dem Knoten häufig begrenzte Bandbreite und Batterielebensdauer haben. Betrachten wir einmal, wie bei diesem Algorithmus Routen entdeckt und erhalten werden.

Ermittlung eines Weges

Bei AODV werden Routen zu einem Ziel nach Bedarf ermittelt, das heißt, nur wenn jemand ein Paket an dieses Ziel senden möchte. Dies spart viel Arbeit, die anderenfalls verschwendet wäre, wenn sich die Topologie ändert, bevor die Route benutzt wird. Die Topologie eines Ad-hoc-Netzes kann zu jedem Zeitpunkt als ein Graph mit verbundenen Knoten beschrieben werden. Zwei Knoten sind miteinander verbunden (es verläuft also eine Kante zwischen den beiden Knoten im Graphen), wenn sie mit ihren Funksendern direkt kommunizieren können. Ein grundlegendes, aber adäquates Modell, das für unsere Zwecke ausreichend ist, ist, dass jeder Knoten mit allen anderen Knoten kommunizieren kann, die innerhalb seines Abdeckungskreises liegen. Reale Netze sind komplizierter, mit Gebäuden, Bergen oder anderen Hindernissen, die die Kommunikation behindern, und Knoten, für die A mit B verbunden ist, aber B nicht mit A , weil A einen leistungsstärkeren Sender als B besitzt. Der Einfachheit halber nehmen wir aber an, dass alle Verbindungen symmetrisch sind.

Betrachten wir zur Beschreibung des Algorithmus das neu gebildete Ad-hoc-Netz in ►Abbildung 5.20. Wir nehmen an, dass ein Prozess in Knoten A ein Paket an Knoten I senden möchte. Der AODV-Algorithmus unterhält eine Distanzvektortabelle auf jedem Knoten, die nach Ziel indiziert ist und Informationen zum Ziel enthält, sowie den Nachbarn, an den die Pakete gesendet werden sollen, um das Ziel zu erreichen. Zuerst schlägt A in seiner Tabelle nach und findet für I keinen Eintrag. Es muss nun ein Weg nach I ermittelt werden. Der Algorithmus wird wegen der Eigenschaft, Wege nur dann zu ermitteln, wenn sie benötigt werden, als bedarfsgesteuert (*on demand*) bezeichnet.

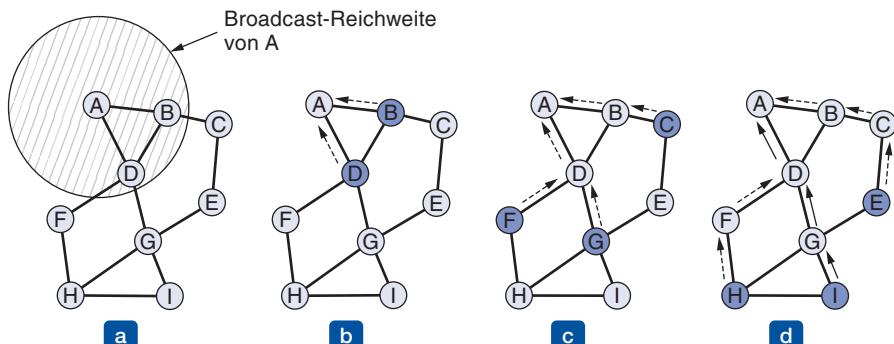


Abbildung 5.20: (a) Reichweite eines Broadcasts von A . (b) Nachdem B und D diesen erhalten haben. (c) Nachdem C , F und D diesen erhalten haben. (d) Nachdem E , H und I diesen erhalten haben. Die dunkelblauen Knoten sind neue Empfänger. Die gestrichelten Linien geben die möglichen umgekehrten Routen an. Die durchgezogenen Linien zeigen die ermittelte Route.

Um I zu finden, erstellt A ein ROUTE REQUEST-Paket und sendet es via Broadcast mittels Fluten, wie in Abschnitt 5.2.3 beschrieben. Die Übertragung von A erreicht B und D , wie aus ►Abbildung 5.20a ersichtlich ist. Jeder Knoten sendet die Anfrage via Broadcast weiter, wodurch schließlich die Knoten F , G und C in ►Abbildung 5.20c und die Knoten H , E und I in ►Abbildung 5.20d erreicht werden. Eine Sequenznummer, die an der Quelle festgesetzt wird, wird benutzt, um Duplikate während des

Flutens auszumerzen. Zum Beispiel verwirft D die Übertragung von B in Abbildung 5.20c, weil D diese Anfrage bereits weitergeleitet hat.

Schließlich erreicht die Anfrage den Knoten I , welcher ein ROUTE REPLY-Paket erzeugt. Dieses Paket wird per Unicast an den Sender in umgekehrter Reihenfolge auf dem Pfad zurückgeschickt, auf dem es angekommen ist. Dazu muss sich jeder Zwischenknoten den Knoten merken, von dem er die Anfrage erhalten hat. Die Pfeile in ►Abbildung 5.20b–d zeigen die gespeicherte Information zur umgekehrten Route. Jeder Zwischenknoten inkrementiert außerdem einen Teilstreckenzähler, wenn er die Antwort weiterleitet. Dadurch werden die Knoten darüber informiert, wie weit sie vom Ziel entfernt sind. Aus den Antworten erfährt jeder Zwischenknoten, welchen Nachbarn er nehmen muss, um das Ziel zu erreichen: den Knoten, der die Antwort gesendet hat. Die Zwischenknoten G und D tragen die beste Route in ihre Routing-Tabellen ein, wenn sie die Antwort bearbeiten. Kommt die Antwort bei A an, dann wurde eine neue Route – $ADGI$ – erzeugt.

In einem großen Netzwerk erzeugt dieser Algorithmus viele Broadcast-Sendungen, selbst bei nahe gelegenen Zielen. Um den Overhead zu reduzieren, wird die Reichweite des Broadcast eingeschränkt. Dazu wird das Feld *Time to Live* der IP-Pakete benutzt. Dieses Feld wird vom Sender initialisiert und mit jeder durchlaufenen Teilstrecke reduziert. Wird sie 0, dann wird das Paket verworfen und nicht rundgesendet. Der Prozess zur Ermittlung der Route wird dann wie folgt geändert: Um ein Ziel zu ermitteln, sendet der Sender ein ROUTE REQUEST-Paket via Broadcast, in dem *Time to Live* auf 1 gesetzt ist. Kommt innerhalb einer vernünftigen Zeitspanne kein Feedback zurück, wird ein weiteres gesendet, wobei diesmal *Time to Live* auf 2 gesetzt wird. Nachfolgende Versuche verwenden dann 3, 4, 5 usw. Auf diese Weise wird eine Ringsuche realisiert, die zuerst lokal vorgenommen und dann immer weiter ausgedehnt wird.

Verwaltung von Wegen

Da die Knoten ihren Standort wechseln oder ausgeschaltet werden können, kann sich die Topologie spontan ändern. Wird beispielsweise in Abbildung 5.20 G ausgeschaltet, dann erkennt A nicht, dass der Weg zu I ($ADGI$) nicht mehr gültig ist. Der Algorithmus muss dies berücksichtigen. In periodischen Abständen sendet jeder Knoten eine *Hello*-Nachricht via Broadcast. Wenn diese mehrfach ausbleiben, weiß der Knoten, dass sich dieser Nachbar aus seinem Funkbereich entfernt hat oder ausgestorben ist und er nicht mehr mit ihm verbunden ist. Auch wenn ein Knoten versucht, ein Paket an einen Nachbarn zu senden, der nicht reagiert, erfährt er daraus, dass der Nachbar nicht mehr verfügbar ist.

Anhand dieser Informationen werden Routen gelöscht, die nicht mehr gültig sind. Für jedes mögliche Ziel merkt sich jeder Knoten N diejenigen seiner aktiven Nachbarn, von denen er während der letzten ΔT Sekunden ein Paket für das Ziel erhalten hat. Ist einer der Nachbarn von N nicht mehr erreichbar, dann überprüft N seine Routing-Tabelle um festzustellen, für welche Ziele der nun nicht mehr verfügbare Nachbar eingetragen ist. Für jede Route, die davon betroffen ist, werden die aktiven Nachbarn informiert, dass ihr Weg über N nun ungültig ist und aus ihren Routing-Tabellen ent-

fernt werden muss. In unserem Beispiel löscht D seinen Eintrag für G und I aus seiner Routing-Tabelle und informiert A , welches seinen Eintrag für I löscht. Im allgemeinen Fall informieren die aktiven Nachbarn dann ihre aktiven Nachbarn und immer so weiter, bis alle Routen, die über den nicht mehr verfügbaren Knoten führen, aus allen Routing-Tabellen entfernt sind.

Zu diesem Zeitpunkt wurden alle ungültigen Wege aus dem Netz entfernt und die Sender können neue gültige Routen finden, indem sie den Mechanismus zu Wegermittlung, den wir beschrieben haben, verwenden. Es gibt jedoch noch eine Schwierigkeit. Erinnern Sie sich daran, dass Distanzvektorprotokolle an langsamer Konvergenz oder am Count-to-Infinity-Problem leiden können, nachdem eine Topologie sich geändert hat, indem sie alte ungültige Routen mit neuen gültigen Wegen verwechseln.

Um schnelle Konvergenz zu erreichen, wird Routen eine Sequenznummer zugewiesen, die vom Ziel kontrolliert wird. Diese Zielsequenznummer funktioniert wie eine logische Uhr. Das Ziel inkrementiert die Nummer jedes Mal, wenn es eine neue ROUTE REPLY-Nachricht sendet. Sender fordern eine neue Route an, indem sie in die ROUTE REQUEST-Nachricht die Zielsequenznummer der letzten verwendeten Route einfügen. Dies ist entweder die Sequenznummer der Route, die soeben gelöscht wurde, oder 0 (als Anfangswert). Die Anfrage wird so lange per Broadcast gesendet, bis eine Route mit einer höheren Sequenznummer gefunden wird. Zwischenknoten speichern die Wege, die höhere Sequenznummern oder die geringste Anzahl an Teilstrecken für die aktuelle Sequenznummer haben.

Ganz im Sinne eines bedarfsgesteuerten Protokolls speichern Zwischenknoten nur die Wege, die in Benutzung sind. Andere Routeninformationen, die während eines Broadcasts empfangen werden, werden nach kurzer Zeit gelöscht. Das Ermitteln und Speichern nur der verwendeten Routen hilft, Bandbreite und Batterielebensdauer zu sparen – im Vergleich zu einem standardmäßigen Distanzvektorprotokoll, das in regelmäßigen Abständen Aktualisierungen per Broadcast überträgt.

Bislang haben wir nur eine einzelne Route betrachtet, in unserem Beispiel von A nach I . Zur weiteren Einsparung von Ressourcen können Routen, die sich überlagern, kooperativ ermittelt und verwaltet werden. Möchte beispielsweise B ebenfalls Pakete nach I senden, so wird B eine Routenermittlung durchführen. In diesem Fall erreicht die Anfrage jedoch zuerst D , welches bereits einen Weg zu I hat. Knoten D kann dann eine Antwort erzeugen, um B diese Route mitzuteilen, sodass das Fluten des ROUTE REQUEST in diese Richtung abgebrochen werden kann.

Es gibt noch viele andere Verfahren zum Ad-hoc-Routing. Ein weiteres, gut bekanntes bedarfsgesteuertes Verfahren ist DSR (*Dynamic Source Routing*; Johnson et al., 2001). Eine auf der Geografie beruhende Strategie wird von GPSR (*Greedy Perimeter Stateless Routing*) verwendet (Karp und Kung, 2000). Kennen alle Knoten ihre geografische Position, so kann ein Paket ohne Routenberechnung zu seinem Ziel weitergeleitet werden. Dazu wird es einfach in die richtige Richtung losgeschickt und aus Sackgassen durch den Einsatz der „Rechte-Hand-Regel“ entweichen kann. Welche Protokolle sich durchsetzen werden, wird von der Art der Ad-hoc-Netze abhängen, die sich in der Praxis als erfolgreich erweisen.

5.3 Algorithmen zur Überlastungsüberwachung

Durch zu viele Pakete in einem Netz (oder einem Teil davon) steigt die Übertragungszeit der Pakete und die Leistung fällt ab. In dieser Situation spricht man von **Überlastung** (*congestion*). Die Vermittlungs- und Transportschichten teilen sich die Verantwortung für die Behandlung von Überlastung. Da Überlastung innerhalb des Netzes auftritt, erfährt die Vermittlungsschicht diese direkt und muss letztendlich festlegen, was mit den überschüssigen Paketen geschehen soll. Der effektivste Weg der Überlastungsüberwachung ist jedoch, die Last zu reduzieren, die die Transportschicht dem Netzwerk auferlegt. Dazu müssen die Vermittlungs- und die Transportschicht zusammenarbeiten. In diesem Kapitel wollen wir Überlastung aus der Perspektive der Vermittlungsschicht betrachten. In *Kapitel 6* werden wir das Thema vervollständigen, indem wir Überlastung aus Sicht der Transportsschicht behandeln.

► Abbildung 5.21 zeigt den Beginn von Überlastung. Entspricht die Anzahl der Pakete, die von den Hosts in das Netz gesendet werden, der Übertragungskapazität des Netzes, dann ist die Anzahl der zugestellten Pakete proportional zur Anzahl der gesendeten Pakete. Falls doppelt so viele Pakete gesendet werden, werden doppelt so viele zugestellt. Erreicht die angebotene Last jedoch die Übertragungskapazität, dann kommt es hin und wieder vor, dass Daten-Bursts die Puffer innerhalb der Router füllen und einige Pakete verloren gehen. Diese verlorenen Pakete verbrauchen einen Teil der Kapazität, daher fällt die Anzahl der zugestellten Pakete unter die ideale Kurve. Das Netz ist nun überlastet.

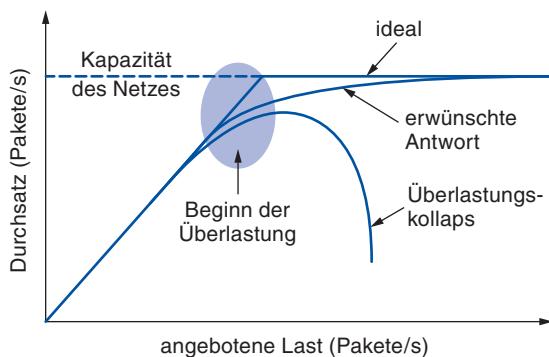


Abbildung 5.21: Bei übermäßiger Verkehrsbelastung sinkt die Performanz rapide ab.

Sofern das Netzwerk nicht sehr gut entworfen wurde, kann es nun einen **Überlastungskollaps** (*congestion collapse*) erleben, bei dem die Performanz stark absinkt, während die angebotene Last über die Kapazität hinaus ansteigt. Dies liegt daran, dass Pakete auch so lange innerhalb des Netzes verzögert werden können, dass sie bereits nutzlos sind, wenn sie das Netz verlassen. In den Anfangsjahren des Internets konnte beispielsweise die Zeit, die ein Paket auf einen Rückstau von Paketen warten musste, die vor diesem über eine langsame 56-kbit/s-Verbindung gesendet werden sollten, die maximale Zeit erreichen, die das Paket im Netz bleiben durfte. Dann musste es verworfen werden. Eine andere Art des Ausfalls tritt auf, wenn Sender Pakete, die stark verzögert

sind, erneut übertragen in der Annahme, dass diese Pakete verloren gingen. In diesem Fall stellt das Netz Kopien desselben Pakets zu, was wiederum Kapazität verschwendet. Daher wird auf der y-Achse von Abbildung 5.21 der Datendurchsatz (*goodput*) dargestellt, also die Rate, mit der *nützliche* Pakete durch das Netz zugestellt werden.

Wir würden gerne Netze entwerfen, die Überlastung nach Möglichkeit vermeiden bzw. keinen Überlastungskollaps erleiden, sollte dies nicht gelingen. Leider kann Überlastung nicht völlig vermieden werden. Kommen plötzlich Paketströme auf drei oder vier Eingangsleitungen an, die alle an die gleiche Ausgangsleitung gesendet werden müssen, so entsteht eine Warteschlange. Wenn nicht ausreichend Speicherplatz vorhanden ist, um alle Pakete vorübergehend abzulegen, dann gehen Pakete verloren. Die Erweiterung des Speichers kann in gewissem Umfang Abhilfe schaffen. Andererseits hat Nagle (1987) entdeckt, dass sich die Überlastung noch verschlimmert, wenn Router eine unendliche Speicherkapazität haben, weil die Timer bereits (wiederholt) abgelaufen sind, bis Pakete die Warteschlange durchlaufen haben und Duplikate gesendet wurden. Dies macht die Sache schlimmer, nicht besser – es führt zum Überlastungskollaps.

Andererseits können auch Verbindungen oder Router mit niedriger Bandbreite, die Pakete langsamer bearbeiten als die Leitungsrate, zu Überlastung führen. In diesem Fall kann die Situation verbessert werden, indem ein Teil des Verkehrs weg vom Engpass zu anderen Teilen des Netzes geleitet wird. Schließlich jedoch werden alle Bereiche des Netzes überlastet sein. In dieser Situation gibt es keine Alternative als die Last zu verringern oder ein schnelleres Netz zu bauen.

Wir wollen hier noch auf den Unterschied zwischen Überlastungsüberwachung und Flusskontrolle hinweisen, da der Zusammenhang zwischen diesen beiden sehr subtil ist. Überlastungsüberwachung betrifft die Sicherstellung, dass das Netz in der Lage ist, den angebotenen Verkehr zu bewältigen. Das ist eine globale Frage, die auch das Verhalten aller Hosts und Router betrifft. Flusskontrolle bezieht sich demgegenüber auf den Datenverkehr zwischen einem bestimmten Sender und einem bestimmten Empfänger. Mit Flusskontrolle wird sichergestellt, dass ein schneller Sender einen langsamen Empfänger nicht mit Daten überschwemmt, die dieser nicht verarbeiten kann.

Um den Unterschied zwischen diesen zwei Konzepten zu verdeutlichen, betrachten wir ein Glasfasernetz mit einer Kapazität von 1 000 Gbit/s, über das ein Supercomputer versucht, einem PC, der lediglich 1 Gbit/s behandeln kann, eine große Datei aufzudrücken. Obwohl im Netz keine Überlastung vorliegt (das Netz selbst hat kein Problem), ist Flusskontrolle erforderlich, um den Supercomputer zu zwingen, während der Übertragung mehrmals anzuhalten, damit der PC Luft holen kann.

Als anderes Extrem betrachten wir ein Netz mit 1-Mbit/s-Leitungen und 1 000 leistungsstarken Rechnern, von denen die Hälfte Dateien mit 100 kbit/s an die andere Hälfte übertragen will. Hier liegt das Problem nicht an schnellen Sendern, die langsame Empfänger überschütten, sondern daran, dass das Netz den insgesamt aufkommenden Verkehr nicht bewältigen kann.

Es gibt einen Grund dafür, dass Überlastungsüberwachung und Flusskontrolle oft verwechselt werden: bei beiden Problemen besteht der beste Lösungsweg darin, den Host

zu verlangsamen. Ein Host kann also eine Nachricht zur Verlangsamung erhalten, weil der Empfänger mit der Flut nicht fertig wird oder weil das Netz überfordert ist. Auf diesen Punkt kommen wir in *Kapitel 6* noch einmal zurück.

Wir beginnen unsere Studie der Überlastungsüberwachung mit der Vorstellung mehrerer Ansätze, die zu jeweils unterschiedlichen Zeitpunkten benutzt werden können. Anschließend kommen wir zu Verfahren, die bereits das Auftreten von Überlastung verhindern, gefolgt von Ansätzen zur Bewältigung von aktuell eingetreteren Überlastungen.

5.3.1 Prinzipien der Überlastungsüberwachung

Das Vorhandensein einer Überlastung bedeutet, dass die Last (vorübergehend) größer ist, als die Ressourcen (in Teilen des Netzes) bewältigen können. Hier bieten sich zwei Lösungen an: Erhöhung der Ressourcen oder Reduzierung der Last. Wie in ►Abbildung 5.22 gezeigt, werden diese Lösungen in der Regel auf verschiedenen Zeitskalen angewandt, um entweder Überlastung zu verhindern oder darauf zu reagieren, wenn Überlastung aufgetreten ist.

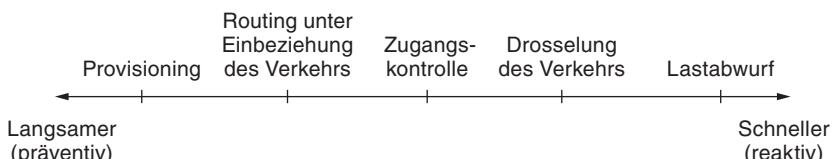


Abbildung 5.22: Zeitskalen von Ansätzen zur Überlastungsüberwachung.

Der einfachste Weg zur Vermeidung von Überlastung ist, bei der Konstruktion des Netzes darauf zu achten, dass es gut auf den zu übertragenden Verkehr abgestimmt ist. Wenn es eine Verbindung mit niedriger Bandbreite auf dem Pfad gibt, der den größten Teil des Verkehrs trägt, dann ist das Auftreten von Überlastung wahrscheinlich. Manchmal können Ressourcen dynamisch hinzugefügt werden, wenn es zu extremer Überlastung kommt, zum Beispiel können Ersatzrouten hinzugeschaltet oder Leitungen aktiviert werden, die normalerweise nur zur Sicherung verwendet werden (um das System fehlertoleranter zu machen), oder es kann Bandbreite auf dem freien Markt erworben werden. Überwiegend werden Verbindungen und Router, die regelmäßig stark benutzt werden, bei der ersten Gelegenheit aufgerüstet. Diesen Vorgang nennt man **Provisioning**, er findet über Monate hinweg statt und wird von langfristigen Entwicklungen im Datenverkehr angetrieben.

Um die vorhandene Netzkapazität bestmöglich auszunutzen, können Routen auf Verkehrsmuster zugeschnitten werden, die sich während des Tages verändern, um sich den unterschiedlichen Wach- und Schlafzeiten der Netzbenutzer in verschiedenen Zeitzonen anzupassen. Zum Beispiel können Routen geändert werden, um stark genutzte Pfade zu entlasten, indem die Gewichtung der kürzesten Pfade verändert wird. Einige lokale Radiostationen haben Helikopter, die um ihre Städte herum fliegen, um über Verkehrsstaus zu berichten, sodass ihre mobilen Zuhörer ihre Pakete (Autos) um Hotspots herumleiten können. Dies nennt man **verkehrsgewahres Routing** (*traffic-aware routing*). Verkehr auf mehrere Pfade aufzuteilen ist ebenfalls hilfreich.

In manchen Fällen jedoch lässt sich die Kapazität nicht steigern. Die einzige Lösung zur Beseitigung einer Überlastung besteht dann darin, die Last zu senken. In einem VC-Netz können neue Verbindungen abgelehnt werden, wenn sie dazu führen würden, dass das Netz überlastet wäre. Dies nennt man **Zugangssteuerung** (*admission control*).

Detaillierter betrachtet, wenn Überlastung unmittelbar bevorsteht, kann das Netz den Quellen eine Rückmeldung geben, die mit ihrem Verkehr für das Problem verantwortlich sind. Das Netz kann entweder diese Quellen auffordern, ihren Verkehr zu drosseln, oder den Verkehr selbst verlangsamen.

Bei diesem Ansatz tauchen zwei Schwierigkeiten auf: Wie kann der Beginn der Überlastung festgestellt werden und wie wird die Quelle, die verlangsamt werden muss, darüber informiert? Das erste Problem kann gelöst werden, indem Router die durchschnittliche Last, die Verzögerung durch Warteschlangenbildung oder den Paketverlust überwachen. In allen Fällen sind steigende Zahlen ein Anzeichen für wachsende Überlastung.

Um das zweite Problem anzugehen, müssen Router an einer Feedbackschleife mit den Quellen teilnehmen. Damit ein Verfahren korrekt arbeitet, muss die Zeitskala sorgfältig angepasst werden. Brüllt ein Router jedes Mal STOPP, wenn zwei Pakete hintereinander ankommen, und jedes Mal LOS, wenn er 20 µs nichts zu tun hatte, dann entstehen im System wilde Schwingungen. Zum Schluss geht nichts mehr. Wenn der Router andererseits 30 Minuten wartet, um absolut sicherzugehen, bevor er irgend etwas kundtut, reagiert der Mechanismus zur Überlastungsüberwachung so schwerfällig, dass er keine Hilfe ist. Die rechtzeitige Bereitstellung von Feedback ist keine banale Angelegenheit. Router, die immer weiter Nachrichten austauschen, wenn das Netz bereits überlastet ist, machen die Sache nicht leichter.

Wenn schlussendlich alles andere fehlgeschlagen ist, bleibt dem Netz nichts übrig als Pakete zu verwerfen, die es nicht zustellen kann. Die allgemeine Bezeichnung dafür ist **Lastabwurf** (*load shedding*). Steht eine gute Strategie zur Verfügung, um auszuwählen, welche Pakete zu verwerfen sind, so kann eventuell ein Überlastungskollaps verhindert werden.

5.3.2 Routing unter Berücksichtigung des Verkehrs

Der erste Ansatz, den wir untersuchen wollen, ist Routing unter Verkehrsberücksichtigung. Die Routing-Verfahren, die wir in Abschnitt 5.2 gesehen haben, benutzten festgelegte Gewichtungen für die Verbindungen. Diese Verfahren sind an Änderungen in der Topologie, nicht aber an Änderungen in der Last angepasst. Wenn man die Last in die Berechnung von Routen einbezieht, ist es das Ziel, den Verkehr von Hotspots wegzuverlagern, da dies die ersten Stellen im Netz sind, wo Überlastung auftritt.

Der direkteste Weg dazu ist die Festlegung der Gewichtung von Verbindungen als Funktion von (fester) Verbindungsbandbreite und Übertragungsverzögerung plus der (variablen) gemessenen Last oder durchschnittlichen Warteschlangenverzögerung. Pfade mit niedrigen Gewichten werden dann Pfade vorgezogen, die weniger belastet sind, wenn alles andere gleich bleibt.

Routing unter Berücksichtigung des Verkehrs wurde in der Anfangszeit des Internets entsprechend diesem Modell eingesetzt (Khanna und Zinky, 1989). Es gibt jedoch eine Gefahr dabei. Betrachten Sie das Netz in ►Abbildung 5.23, das in zwei Teile aufgeteilt ist, Ost und West, die durch zwei Verbindungen, *CF* und *EI*, aneinander angeschlossen sind. Angenommen, der Großteil des Verkehrs zwischen Ost und West benutzt Verbindung *CF*. Als Folge davon ist diese Verbindung stark belastet und benötigt lange für Übertragungen. Wird die Warteschlangenverzögerung in die Gewichtung zur Berechnung des kürzesten Pfads einbezogen, dann wird *EI* attraktiver. Nachdem die neuen Routing-Tabellen installiert wurden, wird der größte Teil des Ost-West-Verkehrs nun über *EI* gehen, wodurch diese Verbindung jetzt belastet wird. Folglich scheint bei der nächsten Aktualisierung *CF* der kürzeste Pfad zu sein. Dies führt dazu, dass die Routing-Tabellen wild pendeln, wodurch das Routing unregelmäßig wird und viele Probleme auftreten können.

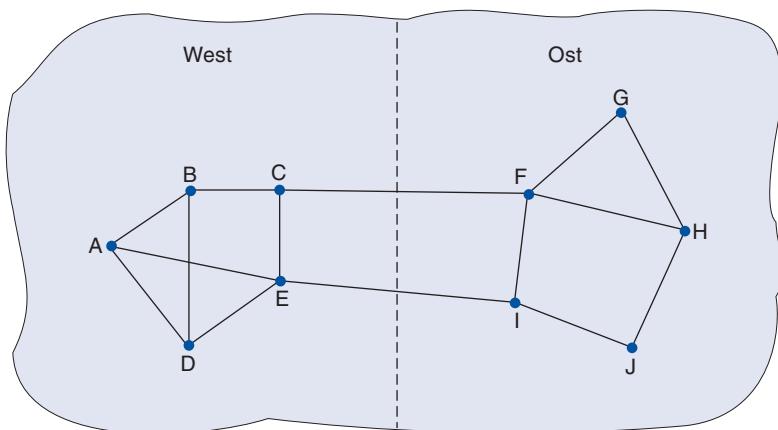


Abbildung 5.23: Ein Netz, in dem der Ost- und der Westteil durch zwei Verbindungen verbunden sind.

Wenn die Last ignoriert wird und stattdessen Bandbreite und Übertragungsverzögerung berücksichtigt werden, tritt dieses Problem nicht auf. Versucht man die Last zwar einzubeziehen, aber die Gewichtungen innerhalb eines engen Bereichs zu ändern, verlangsamt dies lediglich das Oszillieren des Routings. Zwei Techniken sind hier erfolgsversprechend: Die erste ist Multipath-Routing, bei der es mehrere Pfade von einer Quelle zu einem Ziel geben kann. In unserem Beispiel bedeutet dies, dass sich der Verkehr sowohl auf die Ost- als auch auf die Westverbindungen aufteilen kann. Bei der zweiten Technik wird der Verkehr zwischen den Routen so langsam verlagert, dass das Routing-Verfahren konvergiert. Ein Beispiel dafür ist der Algorithmus von Gallagher (1977).

Aufgrund dieser Schwierigkeiten berichtigen die Internet-Routing-Protokolle ihre Routen nicht generell in Abhängigkeit von der Last. Stattdessen werden Anpassungen außerhalb der Routing-Protokolle vorgenommen, indem die Eingaben langsam geändert werden. Dies nennt man **Traffic-Engineering**.

5.3.3 Zugangssteuerung

Eine häufig in VC-Netzen verwendete Technik, um Überlastung einzudämmen, ist die **Zugangssteuerung** (*admission control*). Das Konzept ist ganz einfach: Baue keine neue virtuelle Verbindung auf, falls das Netz den zusätzlichen Verkehr nicht verkraften kann, ohne überlastet zu werden. Das heißt, alle Versuche, eine virtuelle Verbindung aufzubauen, schlagen fehl. Dies ist immerhin besser als die Alternative, denn noch mehr Personen Zugang zu gewähren, wenn das Netz schon ausgelastet ist, verschlimmert das Ganze noch. Im Telefonnetz wird diese Art der Zugangssteuerung ebenfalls praktiziert: Ist eine Vermittlungseinrichtung überlastet, erhält man einfach kein Freizeichen.

Das Kunststück bei diesem Ansatz ist es herauszufinden, wann eine neue virtuelle Schaltung zur Überlastung führen wird. Im Telefonnetz ist diese Aufgabe wegen der festen Bandbreite von Anrufen einfach (64 kbit/s für unkomprimierte Sprachdaten). Virtuelle Verbindungen in Rechnernetzen treten jedoch in unterschiedlichster Gestalt auf. Wenn wir also Zugangssteuerung betreiben sollen, ist es notwendig, auch Informationen über die Art des jeweiligen Verkehrs einer Verbindung zu bekommen.

Datenverkehr wird häufig hinsichtlich seiner Rate und Form beschrieben. Es ist nicht ganz einfach, Verkehr auf eine einfache und doch aussagekräftige Weise zu beschreiben, da er typischerweise in Bursts auftritt – die durchschnittliche Rate erzählt nur die halbe Wahrheit. Beispielsweise ist Datenverkehr, der beim Webbrowsing entsteht, sehr variabel und damit schwieriger zu behandeln als der Verkehr beim Streaming eines Films, bei dem der Durchsatz lange Zeit konstant bleibt – die Bursts des Webverkehrs überlasten die Router im Netz viel wahrscheinlicher. Um diesen Umstand zu erfassen, wird häufig eine Beschreibungsmethode namens **Token Bucket** oder sein Spezialfall namens **Leaky Bucket** verwendet. Ein Token Bucket besitzt zwei Parameter, die die durchschnittliche Rate und die momentane Burst-Größe des Datenverkehrs eingrenzen. Da Token Buckets sehr gerne für die Bereitstellung von Dienstgüte eingesetzt werden, behandeln wir sie detailliert in Abschnitt 5.4.

Ausgerüstet mit den Beschreibungen des jeweiligen Verkehrs kann das Netzwerk nun entscheiden, ob es die neue virtuelle Verbindung zulässt. Dazu könnte das Netz ausreichend Kapazität entlang der Pfade jedes seiner virtuellen Verbindungen reservieren, sodass Überlastung nicht auftreten wird. In diesem Fall entspricht die Verkehrsbeschreibung einer Dienstleistungsvereinbarung darüber, was das Netz seinen Nutzern garantieren kann. Wir haben Überlastung verhindert, sind aber etwas zu früh in das verwandte Thema der Dienstgüte abgedriftet; wir werden im nächsten Abschnitt darauf zurückkommen.

Selbst ohne Garantien zu geben, kann das Netz Verkehrsbeschreibungen zur Zugangssteuerung benutzen. Dann besteht die Aufgabe darin zu schätzen, wie viele virtuelle Verbindungen die Übertragungskapazität des Netzes ohne Überlastung aufnehmen kann. Nehmen wir an, dass virtuelle Verbindungen, die Daten mit Raten bis zu 10 Mbit/s schicken können, alle dieselbe physische 100-Mbit/s-Verbindung passieren. Wie viele virtuelle Verbindungen sollten zugelassen werden? Sicherlich können 10 Verbindungen erlaubt werden, ohne Überlastung zu riskieren, aber im Normalfall ist dies verschwendungsreich, da selten alle 10 Verbindungen gleichzeitig auf Hochtouren übertragen. In echten

Netzen kann bisheriges Verhalten gemessen und somit die Übertragungen statistisch erfasst werden. Diese Messungen können verwendet werden, um die Anzahl der zulässenden Verbindungen abzuschätzen. Damit wird eine bessere Performanz gegen ein akzeptables Risiko eingetauscht.

Zugangssteuerung kann außerdem mit Routing unter Verkehrsberücksichtigung kombiniert werden. In diesem Fall werden Routen rund um Hotspots bereits während des Einrichtens berücksichtigt. Betrachten Sie zum Beispiel das Netzwerk aus ►Abbildung 5.24a, in dem zwei Router wie gezeigt überlastet sind.

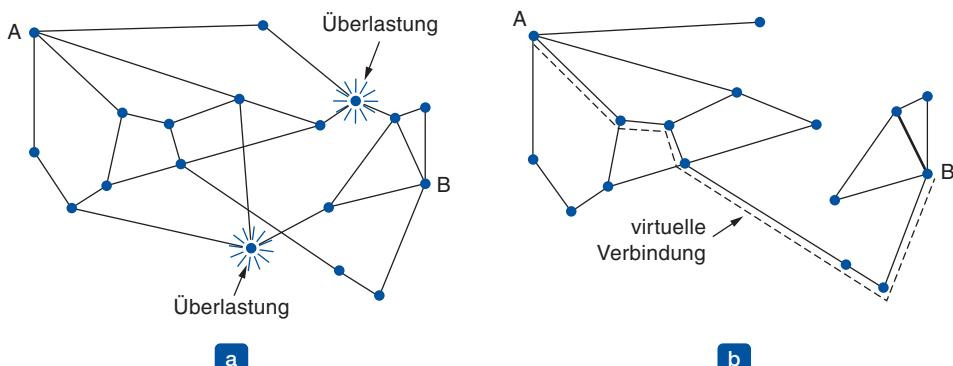


Abbildung 5.24: (a) Überlastetes Netzwerk. (b) Der Teil des Netzes, der nicht überlastet ist. Außerdem ist eine virtuelle Verbindung von A nach B gezeigt.

Angenommen, ein mit Router A verbundener Host möchte eine Verbindung zu einem mit Router B verbundenen Host aufbauen. In der Regel würde diese Verbindung über einen der überlasteten Router aufgebaut werden. Um diese Situation zu vermeiden, kann man das Netz neu auslegen, etwa wie in ►Abbildung 5.24b, wo die überlasteten Router und alle ihre Leitungen umgangen werden. Die gestrichelte Linie ist eine mögliche Route für die virtuelle Verbindung, die die überlasteten Router umgeht. Shaikh et al. (1999) stellen ein Modell für diese Art von lastabhängigem Routing vor.

5.3.4 Drosseln des Verkehrs

Im Internet und vielen anderen Rechnernetzen passen Sender ihre Übertragungen an, um so viel Verkehr zu senden, wie das Netz bequem zustellen kann. In diesem Zusammenhang ist das Netz darauf ausgerichtet, immer kurz vor Einsetzen der Überlastung zu arbeiten. Steht Überlastung unmittelbar bevor, dann muss das Netz den Sendern mitteilen, ihre Übertragungen zu drosseln und zu verlangsamen. Diese Rückmeldung ist eher der Normalzustand als eine Ausnahmesituation. Der Begriff **Überlastungsvermeidung** (*congestion avoidance*) wird manchmal benutzt, um diesen Moment im Betriebsablauf von der Situation abzugrenzen, in der das Netz bereits (allzu) überlastet ist.

Wir wollen uns nun einige Ansätze zur Drosselung des Verkehrs ansehen, die sowohl in Datagrammnetzen als auch in VC-Netzen benutzt werden können. Jeder Ansatz muss zwei Probleme beheben. Erstens müssen Router feststellen, wann Überlastung

auftritt, idealerweise bevor sie eintritt. Dazu kann jeder Router fortlaufend die von ihm verwendeten Ressourcen überwachen. Es gibt hier drei potenzielle Ressourcen: die Ausnutzung der Ausgangsverbindungen, das Puffern der Pakete in der Warteschlange des Routers und die Anzahl der Pakete, die aufgrund ungenügender Pufferung verloren gehen. Die zweite dieser Möglichkeiten ist am sinnvollsten. Durchschnittswerte der Ausnutzung erklären nicht direkt den Grad des Burst-Anteils des größten Teils des Verkehrs – eine Ausnutzung von 50 % kann für gleichmäßigen Verkehr gering sein und zu hoch für extrem schwankenden Verkehr. Zählungen der Paketverluste kommen zu spät. Überlastung hat zu dem Zeitpunkt, an dem die Pakete verloren werden, bereits eingesetzt.

Die Warteschlangenverzögerung innerhalb der Router fängt direkt jegliche Überlastung auf, der Pakete ausgesetzt sind. Dieser Wert sollte im Allgemeinen niedrig sein, aber er steigt sprunghaft an, sobald es einen Daten-Burst gibt, der einen Rückstau erzeugt. Um eine gute Schätzung der Warteschlangenverzögerung d sicherzustellen, kann periodisch eine Abtastung der aktuellen Warteschlangenlänge s vorgenommen und d entsprechend der folgenden Formel aktualisiert werden:

$$d_{\text{neu}} = \alpha d_{\text{alt}} + (1 - \alpha)s$$

wobei die Konstante α bestimmt, wie schnell der Router die neuere Historie vergisst. Dies nennt man einen **exponentiell gewichteten gleitenden Durchschnitt** (EWMA, *Exponentially Weighted Moving Average*). Er glättet Fluktuationen und ist äquivalent zu einem Tiefpassfilter. Jedes Mal, wenn d über einen Schwellenwert hinausgeht, bemerkt der Router den Beginn von Überlastung.

Das zweite Problem ist, dass Router den Sendern, die die Überlastung verursachen, rechtzeitig eine Rückmeldung darüber geben müssen. Überlastung wird zwar im Netz wahrgenommen, doch um sie abzubauen sind Aktionen seitens der Sender nötig, die das Netz benutzen. Um diese Rückmeldung zustellen zu können, muss der Router die entsprechenden Sender identifizieren. Dies muss vorsichtig geschehen, damit nicht noch sehr viele weitere Pakete in das bereits überlastete Netz geschickt werden. Es gibt verschiedene Verfahren, die jeweils andere Feedbackmechanismen benutzen und die wir nun beschreiben werden.

Choke-Pakete

Der direkteste Weg, einen Sender von Überlastung zu unterrichten, ist es ihm direkt mitzuteilen. Bei diesem Ansatz wählt der Router ein überlastetes Paket und sendet ein **Choke-Paket** (Drosselpaket) an den Quell-Host zurück (die Adresse wird im Paket vorgefunden). Das Originalpaket kann gekennzeichnet werden (ein Header-Bit wird gesetzt), sodass keine weiteren Choke-Pakete auf dem Pfad erzeugt werden. Dann wird es auf die übliche Weise weitergeleitet. Um steigende Last auf dem Netz während einer Überlastungszeit zu vermeiden, kann der Router nur ein Choke-Paket mit einer niedrigen Rate senden.

Erhält der Quell-Host das Choke-Paket, dann muss dieser den an das betreffende Ziel gesendeten Datenverkehr um beispielsweise 50 % reduzieren. In einem Datagramm-

netz führt das zufällige Auswählen von Paketen beim Auftreten von Überlastung wahrscheinlich dazu, dass Choke-Pakete zu schnellen Sendern geschickt werden, weil diese die meisten Pakete in der Warteschlange haben. Das diesem Protokoll eigene Feedback trägt zur Vermeidung von Überlastungen bei, drosselt aber nicht jeden Sender, solange er keine Probleme verursacht. Aus dem gleichen Grund ist es wahrscheinlich, dass mehrere Choke-Pakete zu einem bestimmten Host und Ziel gesendet werden. Der Host sollte diese zusätzlichen Choke-Pakete für eine festgelegte Zeit ignorieren, bis seine Reduktion beim Verkehr Wirkung zeigt. Nach dieser Zeitspanne zeigen weitere Choke-Pakete an, dass das Netz immer noch überlastet ist.

Ein Beispiel eines Choke-Pakets, das in den Anfängen des Internets verwendet wurde, ist die SOURCE QUENCH-Nachricht (Postel, 1981). Diese hat sich allerdings niemals durchgesetzt, teilweise weil die Umstände, in denen sie erzeugt wurde, und der Effekt, den sie hatte, nicht klar spezifiziert waren. Das moderne Internet benutzt ein alternatives Benachrichtigungsverfahren, das wir als Nächstes beschreiben wollen.

ECN – Explicit Congestion Notification

Anstatt zusätzliche Pakete zu erzeugen, um vor Überlastung zu warnen, kann ein Router irgendein Paket, das er weiterleitet, mit einem Kennzeichen versehen (indem ein Bit im Header des Pakets gesetzt wird), um zu signalisieren, dass er überlastet ist. Wenn das Netz das Paket ausliefert, registriert das Ziel, dass es Überlastung gibt, und informiert den Sender beim Senden eines Antwortpakets. Der Sender kann dann wie vorher seine Übertragungen drosseln.

Dieser Entwurf heißt **ECN** (*Explicit Congestion Notification*) und wird im Internet verwendet (Ramakrishnan et al., 2001). Es verfeinert frühere Protokolle zur Überlastungswarnung, namentlich das binäre Feedbackverfahren von Ramakrishnan und Jain (1988), das in der DECnet-Architektur eingesetzt wurde. Zwei Bits in dem IP-Paket-Header werden verwendet um aufzuzeichnen, ob das Paket Überlastung ausgesetzt war. Ein Paket ohne Markierung abgesendet ist in ▶ Abbildung 5.25 dargestellt. Falls einer der Router, den das Paket durchläuft, überlastet ist, so markiert dieser Router das Paket entsprechend, bevor es weitergeleitet wird. Das Ziel wird dann in seinem nächsten Antwortpaket alle Markierungen als Rückmeldung an den Sender als explizites Überlastungszeichen weitergeben. In der Abbildung wird dies mit einer gestrichelten Linie dargestellt, um anzudeuten, dass sich dieser Vorgang oberhalb der IP-Ebene abspielt (z.B. in TCP). Der Sender muss dann seine Übertragungen drosseln, wie im Fall der Choke-Pakete.

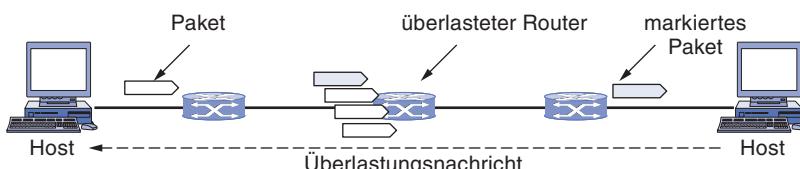


Abbildung 5.25: ECN.

Hop-by-Hop-Rückstau

Bei hohen Geschwindigkeiten oder über große Entfernungen können aufgrund der Übertragungsverzögerung noch viele neue Pakete übertragen werden, nachdem die Überlastung gemeldet wurde, bevor das Signal Wirkung zeigt. Man betrachte z.B. einen Host in San Francisco (Router A in ► Abbildung 5.26), der mit der OC-3-Geschwindigkeit von 155 Mbit/s Daten an einen Host in New York (Router D in Abbildung 5.26) sendet. Gehen beim New Yorker Host die Puffer zur Neige, so dauert es etwa 40 ms, bis ein Choke-Paket nach San Francisco zurückkommt und dort die Nachricht zur Verlangsamung überbringt. Ein ECN-Hinweis bräuchte sogar noch länger, weil es über das Ziel zugestellt wird. Die Weiterleitung des Choke-Pakets ist im zweiten, dritten und vierten Schritt in ► Abbildung 5.26a dargestellt. In diesen 40 ms können 6,2 Mbit gesendet werden. Auch wenn der Host in San Francisco sofort vollständig abschaltet, kommen die 6,2 Mbit weiter in der Leitung ein und müssen irgendwie verarbeitet werden. Erst im siebten Diagramm in Abbildung 5.26a bemerkt der Router in New York einen langsameren Datenfluss.

Bei einem alternativen Ansatz wird das Choke-Paket auf jeder durchlaufenden Teilstrecke (Hop) wirksam (siehe ► Abbildung 5.26b). Sobald ein Choke-Paket F erreicht, muss F den Datenfluss nach D reduzieren. Hierdurch muss F mehr Puffer für die Verbindung bereitstellen, weil die Quelle immer noch mit voller Kraft weitersendet. Jedoch wird D sofort entlastet – wie bei der Fernsehwerbung für Kopfschmerztabletten. Im nächsten Schritt erreicht das Choke-Paket E , womit E angewiesen wird, den Fluss nach F zu reduzieren. Diese Maßnahme stellt eine größere Anforderung an die Puffer von E , entlastet aber F sofort. Schließlich erreicht das Choke-Paket A , und der Datenfluss verlangsamt sich stetig.

Die Nettowirkung bei diesem Hop-by-Hop-Schema ist die schnelle Entlastung der überlasteten Stelle auf Kosten der stärkeren Belastung der Upstream-Puffer. Auf diese Weise kann die Überlastung im Keim ersticken werden, ohne dass Pakete verloren gehen. Das Konzept wird ausführlich in Mishra et al. (1996) behandelt.

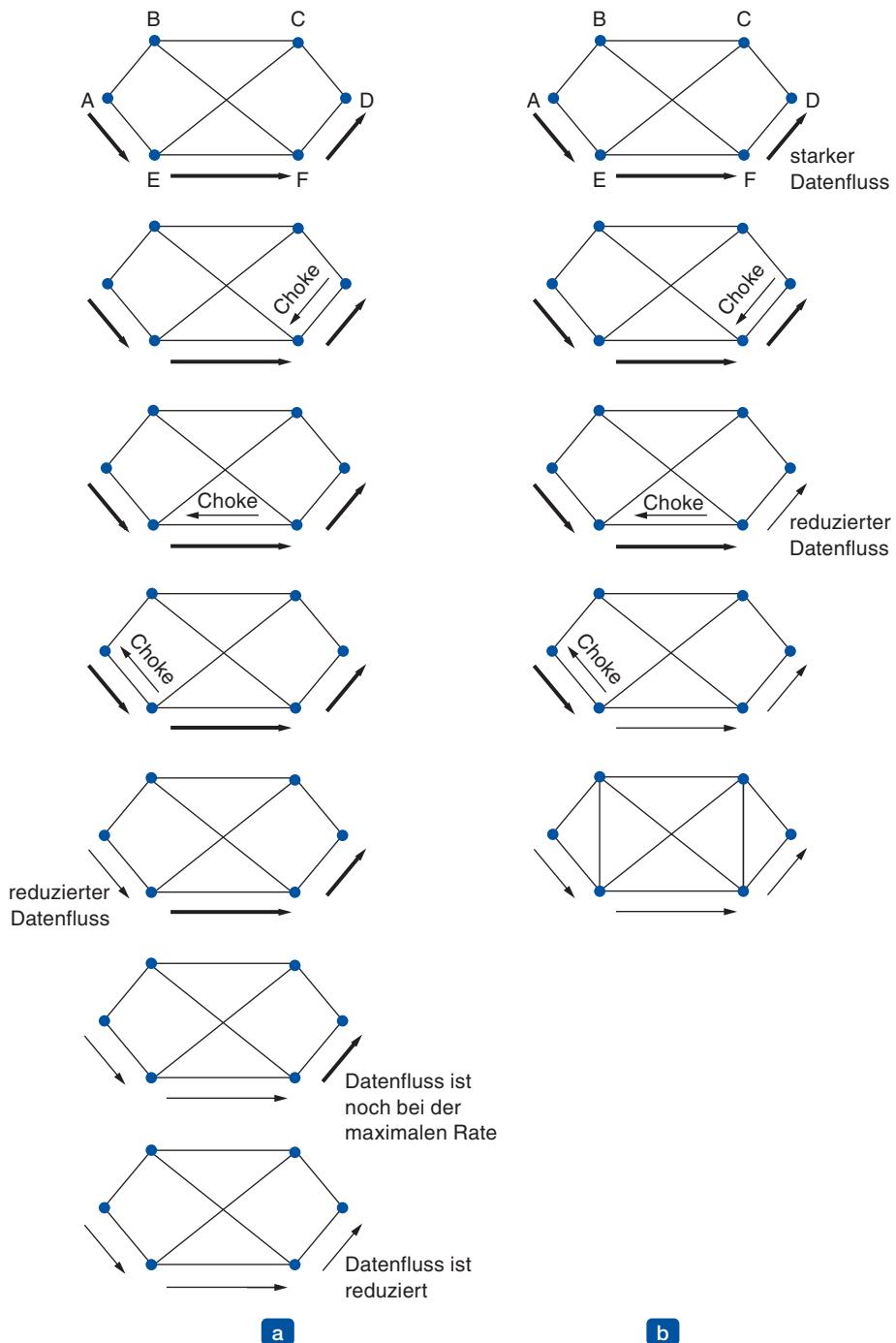


Abbildung 5.26: (a) Ein Choke-Paket, das sich nur auf die Quelle auswirkt.
 (b) Ein Choke-Paket, das sich auf jede durchlaufene Teilstrecke auswirkt.

5.3.5 Lastabwurf

Kann eine Überlastung durch keine der genannten Methoden eingedämmt werden, dann können Router mit schwererem Geschütz auffahren: dem **Lastabwurf** (*load shedding*). Das ist eine originelle Methode, bei der Router Pakete, die sie nicht bewältigen können, einfach verwerfen können. Der Begriff „Lastabwurf“ stammt aus der Welt der Stromerzeugung, wo er die absichtliche Abschaltung bestimmter Bereiche bezeichnet, um das gesamte Stromnetz vor dem Zusammenbruch zu bewahren, wenn der Stromverbrauch das Angebot um ein Vielfaches übersteigt.

Ein Router, der gerade in Paketen ertrinkt, steht nun vor der Entscheidungsfrage, welche der Pakete er fallen lassen soll. Dies könnte von der Art der Anwendungen abhängen, die das Netz benutzen. Bei einer Dateiübertragung ist ein altes Paket mehr wert als ein neues. Dies liegt daran, dass das Verwerfen von Paket 6 und das Behalten der Pakete 7 bis 10 beispielsweise den Empfänger nur zwingt, sich noch mehr anzustrengen, Daten zu puffern, die er noch nicht benutzen kann. Demgegenüber ist bei Echtzeit-Multimedia ein neues Paket wertvoller als ein altes. Dies liegt daran, dass Pakete nutzlos werden, wenn sie verzögert sind und somit der Zeitpunkt verstreicht, an dem sie beim Benutzer abgespielt werden sollten.

Die Regel „lieber alt als neu“ wird oft **Wein** und die Regel „lieber neu als alt“ wird **Milch** genannt, weil die meisten Menschen eher neue Milch und alten Wein trinken würden als umgekehrt.

Intelligentere Verfahren zum Lastabwurf setzen die Kooperation der Sender voraus. Ein Beispiel dafür sind Pakete, die Routing-Informationen übertragen. Diese Pakete sind wichtiger als normale Datenpakete, weil sie Routen aufbauen; wenn diese Pakete verloren gehen, kann das Netz Konnektivität verlieren. Ein anderes Beispiel ist, dass Algorithmen zur Kompression von Video (wie MPEG) periodisch einen ganzen Rahmen übertragen, dann senden sie weitere Rahmen mit Unterschieden zum letzten vollständigen Rahmen. In diesem Fall ist das Verwerfen eines Pakets, das Teil eines Differenzrahmens ist, weniger schmerzlich als das Verwerfen eines Pakets, das einen vollständigen Rahmen enthält, weil zukünftige Pakete von dem vollständigen Rahmen abhängen.

Soll eine intelligente Methode zum Verwerfen implementiert werden, dann müssen Anwendungen ihre Pakete kennzeichnen, um dem Netz deren jeweilige Bedeutung anzuzeigen. In diesem Fall können Router, wenn sie Pakete verwerfen müssen, zuerst die Pakete der am wenigsten wichtigen Klasse verwerfen, dann der zweitunwichtigsten und so fort.

Wenn natürlich kein erheblicher Anreiz damit verbunden wäre, zu vermeiden, dass Pakete mit SEHR WICHTIG – NIEMALS VERWERFEN gekennzeichnet werden, so würde dies niemand tun. Häufig werden finanzielle Anreize benutzt, um Sender vom leichtenfertigen Markieren abzuschrecken. Beispielsweise könnte das Netz einem Sender erlauben, schneller zu senden als es der bezahlte Dienst eigentlich zulässt, falls er überzählige Pakete als von niedriger Priorität kennzeichnet. Eine solche Strategie ist nicht schlecht, weil sie zu einer effizienteren Nutzung der Ressourcen im Leerlaufzu-

stand führt. Denn Hosts können sie verwenden, solange niemand daran interessiert ist, erhalten aber nicht das ständige Recht darauf, wenn die Zeiten härter werden.

Früherkennung nach dem Zufallsprinzip

Es ist effizienter, die Überlastung zu behandeln, wenn sie anfängt, als wenn man wartet, bis alles verstopft ist, und dies dann zu beheben versucht. Diese Beobachtung führt zu einer interessanten Form des Lastabwurfs, nämlich Pakete zu verwerfen, bevor der Puffer vollständig belegt ist.

Die Motivation für dieses Konzept ist, dass die meisten Internethosts noch keine Überlastungsmitteilungen von Routern in der Form von ECN bekommen. Stattdessen ist der Paketverlust das einzige zuverlässige Anzeichen einer Überlastung, die Hosts vom Netz erhalten. Schließlich ist es nicht ganz einfach, einen Router zu konstruieren, der keine Pakete fallen lässt, wenn er überlastet ist. In Transportprotokollen wie TCP ist somit fest vorgegeben, auf Verlust als Anzeichen von Überlastung zu reagieren und die Quelle zu verlangsamen. Dies ist darauf zurückzuführen, dass TCP für kabelgebundene Netze konzipiert wurde, die sehr zuverlässig sind. Daher sind verlorene Pakete in der Regel auf Pufferüberläufe und nicht auf Übertragungsfehler zurückzuführen. Drahtlose Verbindungen müssen Übertragungsfehler in der Sicherungsschicht berichtigen (sie sind auf der Vermittlungsschicht also nicht zu sehen), um gut mit TCP zusammenzuarbeiten.

Diese Situation kann zur Verringerung der Überlastung genutzt werden. Wenn Router Pakete früh verwerfen, bevor die Situation aussichtslos wird, hat die Quelle Zeit, etwas zu unternehmen, bevor es zu spät ist. Ein weitverbreiteter Algorithmus hierfür ist der sogenannte **RED**-Algorithmus (*Random Early Detection*, Früherkennung nach dem Zufallsprinzip), der in Floyd und Jacobson (1993) beschrieben wird. Die Router ermitteln laufend die durchschnittliche Länge der Warteschlange, um festzustellen, ab wann die Pakete verworfen werden sollen. Wenn die durchschnittliche Länge der Warteschlange einen bestimmten Grenzwert übersteigt, geht man davon aus, dass die Verbindung überlastet ist, und ein kleiner Teil der Pakete wird zufällig fallen gelassen. Diese zufällige Auswahl der Pakete in der Warteschlange erhöht die Wahrscheinlichkeit, dass darunter ein Paket der schnellsten Sender ist; dies ist die beste Option, da der Router nicht feststellen kann, welche Quelle die meisten Probleme in einem Datagrammnetz verursacht. Der betroffene Sender wird den Verlust bemerken, wenn es keine Bestätigung gibt, und dann das Transportprotokoll verlangsamen. Das verlorene Paket liefert somit dieselbe Nachricht wie ein Choke-Paket, aber indirekt, ohne dass der Router ein explizites Signal sendet.

RED-Router verbessern die Performanz im Vergleich zu Routern, die Pakete nur fallen lassen, wenn ihre Puffer voll sind, obwohl es bei RED eventuell notwendig ist, die Router zu justieren, damit das Verfahren gut funktioniert. Beispielsweise hängt die ideale Anzahl der Pakete, die fallen gelassen werden sollen, davon ab, wie viele Sender von der Überlastung benachrichtigt werden. ECN sollte jedoch, sofern es verfügbar ist, dennoch bevorzugt werden. ECN funktioniert auf genau dieselbe Art und Weise, sendet aber ein explizites Überlastungssignal statt einer indirekten Benachrichtigung über Paketverluste. RED wird immer dann benutzt, wenn Hosts keine expliziten Signale empfangen können.

5.4 Dienstgüte

Die in den vorherigen Abschnitten beschriebenen Verfahren dienen der Verringerung der Belastung und der Verbesserung der Netzleistung. Es gibt jedoch Anwendungen (und Kunden), die stärkere Performanzgarantien vom Netz verlangen als „das Beste, das unter den Umständen getan werden konnte“. Besonders Multimedia-Anwendungen benötigen häufig einen minimalen Durchsatz und maximale Latenz zum Arbeiten. In diesem Abschnitt werden wir uns weiterhin mit der Untersuchung der Netzleistung beschäftigen, wobei nun der Schwerpunkt auf den Möglichkeiten liegt, die von den Anwendungen benötigte Dienstgüte bereitzustellen. Dies ist ein Bereich, in dem das Internet gerade eine nachhaltige Aufrüstung erfährt.

Eine einfache Lösung, um gute Dienstgüte zur Verfügung zu stellen, ist ein Netz so zu bauen, dass es immer ausreichend Kapazität hat, egal mit welchem Verkehr es konfrontiert wird. Diese Lösung, Ressourcen im Überfluss vorzuhalten, ist unter dem Namen **Overprovisioning** bekannt. Das resultierende Netz wird Anwendungsdatenverkehr ohne bedeutenden Verlust übertragen und – ein ordentliches Routing-Schema vorausgesetzt – Pakete mit geringer Latenz zustellen. Bessere Performanz wird es nicht geben. In gewissem Maß sind im Telefonsystem Ressourcen im Überfluss vorhanden, weil es ganz selten passiert, dass man einen Telefonhörer aufnimmt und nicht sofort einen Amtston erhält. Es ist ganz einfach so viel Kapazität verfügbar, dass der Bedarf immer erfüllt werden kann.

Diese Methode hat den Nachteil, dass sie teuer ist. Im Grunde wird ein Problem durch den Einsatz von Geld gelöst. Mithilfe von Dienstgütemechanismen kann ein Netz mit weniger Kapazität ebenso gut die Anforderungen der Anwendungen erfüllen, aber zu geringeren Kosten. Des Weiteren basiert Overprovisioning auf erwartetem Verkehr. Alles ist möglich, wenn sich die Verkehrsmuster stark verändern. Mit Dienstgütemechanismen kann das Netz die versprochenen Performanzgarantien einlösen, selbst wenn es zu Spitzen im Datenverkehr kommt. Dann werden lediglich einige Anfragen abgewiesen.

Um Dienstgüte zuzusichern, müssen vier Fragen beantwortet werden:

1. Was benötigen Anwendungen vom Netz?
2. Wie wird der Verkehr reguliert, der das Netz betritt?
3. Wie werden Ressourcen beim Router reserviert, um Performanz zu garantieren?
4. Kann das Netz gefahrlos mehr Verkehr annehmen?

Es gibt kein einzelnes Verfahren, das effizient mit all diesen Fragestellungen umgeht. Stattdessen wurden verschiedene Verfahren zum Einsatz in der Vermittlungsschicht (und der Transportschicht) entwickelt. Praxislösungen für Dienstgüte kombinieren mehrere Verfahren. Zu diesem Zweck werden wir zwei Dienstgüteversionen für das Internet beschreiben, die integrierten Dienste (*Integrated Services*) und die differenzierten Dienste (*Differentiated Services*).

5.4.1 Anforderungen der Anwendungen

Ein Strom von Paketen von einer Quelle zum Ziel wird als **Datenfluss** (*flow*; Clark, 1988) bezeichnet. In einem verbindungsorientierten Netz können alle Pakete einer Verbindung zu einem Datenfluss gehören, in einem verbindungslosen Netz alle Pakete, die von einem Prozess zu einem anderen gesendet werden. Die Anforderungen eines Datenflusses können mit vier primären Parametern angegeben werden: Bandbreite, Übertragungsverzögerung, Jitter und Verlustrate. Miteinander legen diese die **Dienstgüte** (*QoS, Quality of Service*) fest, die ein Datenfluss benötigt.

Einige weitverbreitete Anwendungen und die Stringenz ihrer Netzanforderungen werden in ►Abbildung 5.27 aufgelistet. Beachten Sie, dass Netzanforderungen in solchen Fällen weniger anspruchsvoll sind als Anwendungsanforderungen, in denen die Anwendung den Dienst verbessern kann, der vom Netz zur Verfügung gestellt wird. Insbesondere müssen Netze für zuverlässige Dateiübertragung nicht verlustfrei sein und sie müssen keine Pakete mit identischen Übertragungsverzögerungen für Audio- und Videowiedergabe zustellen. Eine gewisse Verlustmenge kann mit Neuübertragungen behoben werden und eine weitere kann durch Jitter geglättet werden, indem Pakete beim Empfänger zwischengespeichert werden. Anwendungen können jedoch nichts dagegen unternehmen, wenn das Netz zu geringe Bandbreite anbietet oder zu groÙe Verzögerung aufweist.

| Anwendung | Bandbreite | Übertragungsverzögerung | Jitter | Verlustrate |
|---------------------|------------|-------------------------|---------|-------------|
| E-Mail | Niedrig | Niedrig | Niedrig | Mittel |
| Dateiaustausch | Hoch | Niedrig | Niedrig | Mittel |
| Webzugriff | Mittel | Mittel | Niedrig | Mittel |
| Entfernte Anmeldung | Niedrig | Mittel | Mittel | Mittel |
| Audio on Demand | Niedrig | Niedrig | Hoch | Niedrig |
| Video on Demand | Hoch | Niedrig | Hoch | Niedrig |
| Telefonie | Niedrig | Hoch | Hoch | Niedrig |
| Videokonferenzen | Hoch | Hoch | Hoch | Niedrig |

Abbildung 5.27: Stringenz der Anforderungen für die Dienstgüte verschiedener Anwendungen.

Die Anwendungen unterscheiden sich hinsichtlich ihrer Anforderungen an die Bandbreite. Hier benötigen E-Mail, Audio in allen Formen und Fernanmeldung nicht viel, aber Dateiaustausch und Video in allen Facetten benötigen sehr viel.

Interessanter sind die Anforderungen an die Verzögerung. Bei Anwendungen zur Übertragung von Dateien, inklusive E-Mail oder Video, spielt eine Verzögerung der Übertragung keine Rolle. Werden alle Pakete einheitlich um ein paar Sekunden verzögert übertragen, dann schadet dies nichts. Bei interaktiven Anwendungen wie Surfen

im Web und Fernanmeldungen ist die Übertragungsverzögerung kritischer. Echtzeitanwendungen, wie Telefonie und Videokonferenzen, haben strenge Anforderungen bezüglich der Übertragungsverzögerung. Wenn alle Wörter bei einem Telefongespräch zu lange verzögert werden, wird dies für die Gesprächsteilnehmer nicht mehr akzeptabel sein. Wenn andererseits Audio- oder Videodateien von einem Server abgespielt werden, ist eine geringe Übertragungsverzögerung nicht notwendig.

Die Schwankung (also die Standardabweichung) bei der Übertragungsverzögerung oder den Paketankunftszeiten wird als **Jitter** bezeichnet. Bei den ersten drei Anwendungen von Abbildung 5.27 spielt es keine Rolle, wenn die Pakete in unregelmäßigen Zeitabständen ankommen. Die Fernanmeldung ist hier etwas kritischer, da Aktualisierungen auf dem Bildschirm in kleinen Bursts angezeigt werden, wenn zu viel Jitter in der Verbindung besteht. Bei Video und vor allem Audio ist Jitter extrem kritisch. Wenn ein Benutzer ein Video über das Netz ansieht und alle Rahmen um genau 2 000 Sekunden verzögert ankommen, ist dies nicht weiter schlimm. Wenn aber die Übertragungszeit nach dem Zufallsprinzip um ein oder zwei Sekunden variiert, ist das Ergebnis schrecklich. Bei Audio ist selbst ein Jitter von wenigen Millisekunden deutlich hörbar.

Die ersten vier Anwendungen haben höhere Anforderungen bezüglich der Verlustrate als Audio und Video, weil alle Bits korrekt geliefert werden müssen. Dieses Ziel wird in der Regel mittels Neuübertragungen durch die Transportschicht von Paketen, die im Netz verloren gehen, erreicht. Dies ist verschwendete Arbeit; es wäre besser, wenn das Netz von vornherein Pakete ablehnen würde, die wahrscheinlich verloren werden. Audio- und Videoanwendungen können einige Pakete ohne Neuübertragungen tolerieren, weil niemand kurz Pausen oder gelegentlich ausgelassene Rahmen wahrnimmt.

Um für eine breite Auswahl von Anwendungen geeignet zu sein, können Netze unterschiedliche Kategorien von Dienstgüte unterstützen. Ein einflussreiches Beispiel kommt von ATM-Netzen, die einst Teil einer großen Netzwerktechnik-Vision waren, seitdem aber zu einer Nischentechnologie wurden. Sie unterstützen:

- 1.** Konstante Bitübertragungsrate (wie Telefonie)
- 2.** Variable Echtzeit-Bitübertragungsrate (wie komprimierte Übertragung von Videokonferenzen)
- 3.** Variable Nichthechtzeit-Bitübertragungsrate (wie einen Film „on demand“ ansehen)
- 4.** Verfügbare Bitübertragungsrate (wie Dateiübertragung)

Diese Kategorien sind auch für andere Zwecke und Netze nützlich. Eine konstante Bitübertragungsrate ist der Versuch, ein Kabel zu simulieren, indem einheitliche Bandbreite und Übertragungsverzögerung zur Verfügung gestellt werden. Eine variable Bitübertragungsrate tritt auf, wobei einige Rahmen stärker komprimiert werden als andere. Bei einem Rahmen mit vielen Einzelinformationen ist die Übertragung vieler Bits erforderlich, wohingegen sich ein Foto einer weißen Wand normalerweise sehr gut komprimieren lässt. Filme auf Bestellung (*movie on demand*) sind keine richtige Echtzeitanwendung, da ein paar Sekunden der Videodaten leicht beim Empfänger

gepuffert werden können, bevor die Abspielung beginnt, daher wird durch Jitter im Netz die Menge der gespeicherten, aber noch nicht abgespielten Videos kaum verändert. Die verfügbare Bitübertragungsrate bezieht sich auf Anwendungen wie E-Mail, bei denen die Übertragungsverzögerung oder Jitter keine Rolle spielen und die so viel Bandbreite beanspruchen, wie sie bekommen können.

5.4.2 Traffic-Shaping

Bevor das Netz Dienstgütegarantien geben kann, muss es wissen, welche Art von Datenverkehr garantiert wird. Im Telefonnetz ist diese Charakterisierung einfach. Zum Beispiel benötigt ein Sprachanruf (in unkomprimiertem Format) 64 kbit/s und besteht aus einer 8-Bit-Abtastung alle 125 µs. Verkehr in Datennetzen tritt jedoch in **Bursts** auf. Er kommt typischerweise mit ungleichmäßigen Raten an, da die Verkehrsrate variiert (z.B. Videokonferenzen mit Komprimierung), Benutzer mit Anwendungen interagieren (z.B. Browsen einer neuen Webseite) und Computer zwischen Aufgaben hin- und herwechseln. Bursts sind schwieriger zu behandeln als Verkehr mit einer konstanten Rate, weil sie die Puffer füllen und somit Pakete verloren gehen.

Traffic-Shaping ist eine Technik, die durchschnittliche Übertragungsrate und das Burst-Aufkommen eines Datenflusses zu regulieren, der das Netz betritt. Ziel ist, dass Anwendungen zahlreiche Arten von jeweils auf ihre Bedürfnisse angepassten Verkehr übermitteln können, einschließlich einiger Bursts, und gleichzeitig die möglichen Verkehrsmuster des Netzes auf einfache und zweckmäßige Art und Weise beschreiben zu können. Wird ein Datenfluss eingerichtet, so vereinbaren der Benutzer und das Netz (d.h. der Kunde und der Provider) ein bestimmtes Verkehrsmuster für diesen Datenfluss. Im Prinzip sagt der Kunde zum Provider: „Mein Übertragungsmuster sieht folgendermaßen aus. Kannst du dies verarbeiten?“

Diese Absprache wird manchmal **Dienstgütevereinbarung** (*Service Level Agreement, SLA*) genannt, besonders wenn sie sich auf aggregierte Datenflüsse bezieht und über einen langen Zeitraum abgeschlossen wird (z.B. der gesamte Datenverkehr eines bestimmten Kunden). Solange der Kunde seinen Teil der Vereinbarung erfüllt und Pakete nur entsprechend der Vereinbarung sendet, gewährleistet der Provider deren zeitgerechte Zustellung.

Traffic-Shaping reduziert Überlastungen und unterstützt damit das Netz, sein Versprechen zu erfüllen. Damit das Verfahren jedoch funktioniert, muss außerdem die Frage geklärt werden, wie der Provider feststellen kann, ob der Kunde die Vereinbarung einhält, und was zu tun ist, wenn dies nicht der Fall ist. Pakete, die über das vereinbarte Muster hinausgehen, können zum Beispiel vom Netz fallen gelassen werden oder eine Kennzeichnung erhalten, dass sie eine geringere Priorität haben. Die Überwachung eines Verkehrsflusses nennt man **Traffic-Policing** (Regelüberwachung für den Datenverkehr).

Traffic-Shaping und Traffic-Policing sind nicht so entscheidend für Peer-to-Peer- und andere Übertragungen, die sämtliche verfügbare Bandbreite verbrauchen werden, haben aber eine große Bedeutung für Echtzeitdaten wie Audio- und Videoverbindungen, die hohe Dienstgüteanforderungen aufweisen.

Leaky-Bucket- und Token-Bucket-Algorithmus

Wir haben bereits eine Möglichkeit gesehen, die von einer Anwendung gesendete Datenmenge zu begrenzen: Schiebefenster mit einem Parameter, durch welchen die Datenmenge begrenzt wird, die zu einem beliebigen Zeitpunkt unterwegs ist, wodurch indirekt die Datenrate eingeschränkt wird. Nun wollen wir uns mit dem Leaky-Bucket- und dem Token-Bucket-Algorithmus einen allgemeineren Weg zur Charakterisierung des Verkehrs ansehen. Die Formulierungen sind leicht anders, aber das Ergebnis ist ähnlich.

Stellen Sie sich einen Eimer (Bucket) mit einem kleinen Loch im Boden vor, wie in ▶ Abbildung 5.28b dargestellt. Gleichgültig, in welcher Geschwindigkeit Wasser in den Eimer eingefüllt wird, der Ausfluss hat immer die gleiche Rate R , wenn Wasser vorhanden ist, und null, wenn der Eimer leer ist. Ist der Eimer mit der Kapazität B voll, so läuft überschüssiges Wasser über den Eimerrand und geht verloren.

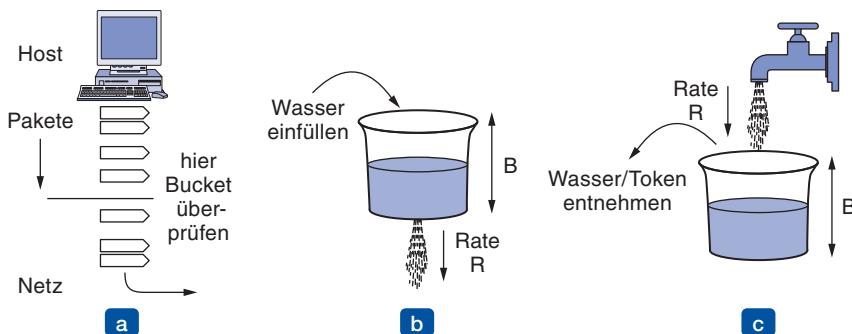


Abbildung 5.28: (a) Traffic-Shaping von Paketen. (b) Leaky Bucket. (c) Token Bucket.

Der Eimer kann verwendet werden, um Traffic-Shaping oder Traffic-Policing auf Pakete anzuwenden, die das Netz betreten (siehe ▶ Abbildung 5.28a). Vom Konzept her ist jeder Host mit dem Netz über eine Schnittstelle verbunden, die einen undichten Eimer (*leaky bucket*) enthält (Abbildung 5.28b). Um ein Paket in das Netz zu senden, muss es möglich sein, den Eimer mit mehr Wasser zu füllen. Kommt ein Paket an, wenn der Eimer voll ist, dann muss das Paket entweder in die Warteschlange gestellt werden, bis genügend Wasser herausgeflossen ist, um das Paket aufzunehmen, oder das Paket wird verworfen. Der erste Fall kann bei einem Host auftreten, der seinen Verkehr auf Betriebssystemebene mittels Traffic-Shaping für das Netz aufbereitet. Der zweite Fall könnte auf der Hardwareebene bei einer Provider-Netzwerk-Schnittstelle auftreten, wenn Traffic-Policing beim Eintritt ins Netz verwendet wird. Diese Technik wurde von Turner (1986) vorgeschlagen und heißt **Leaky-Bucket-Algorithmus**.

Eine andere Formulierung ist, sich die Netzschmittstelle als einen Eimer vorzustellen, der wie in ▶ Abbildung 5.28c dargestellt gefüllt wird. Wie vorher läuft der Wasserhahn mit Rate R und der Eimer hat die Kapazität B . Nun müssen wir, um ein Paket zu senden, in der Lage sein, Wasser – oder Token, wie die Inhalte gewöhnlich genannt werden – aus dem Eimer zu entnehmen (statt Wasser in den Eimer zu füllen). Im Eimer

kann sich höchstens eine feste Anzahl von Token, B , ansammeln, und falls der Eimer leer ist, müssen wir warten, bis mehr Token ankommen, bevor wir ein weiteres Paket senden können. Dieser Algorithmus heißt **Token-Bucket-Algorithmus**.

Der Token Bucket begrenzt die langfristige Rate eines Datenflusses, erlaubt es aber, dass kurzfristige Bursts bis zu einer maximalen Länge unverändert durchkommen, ohne künstlich verzögert zu werden. Große Bursts werden von einem Token-Bucket-Traffic-Shaper geglättet, um die Überlastung im Netz zu verringern. Der Leaky Bucket regelt im Gegensatz dazu die maximale Rate eines Datenflusses. Stellen Sie sich beispielsweise vor, dass ein Computer Daten mit bis zu 1 000 Mbit/s (125 Millionen Byte/s) erzeugen kann und dass die erste Verbindungs des Netzes ebenfalls mit dieser Geschwindigkeit läuft. Das Verkehrsmuster, das der Host erzeugt, ist in ▶ Abbildung 5.29 gezeigt. Bei diesem Muster tritt der Verkehr in Bursts auf. Die durchschnittliche Rate über einer Sekunde ist 200 Mbit/s, selbst wenn der Host einen Burst von 16 000 KB mit der Spitzengeschwindigkeit von 1 000 Mbit/s sendet (für 1/8 der Sekunde).

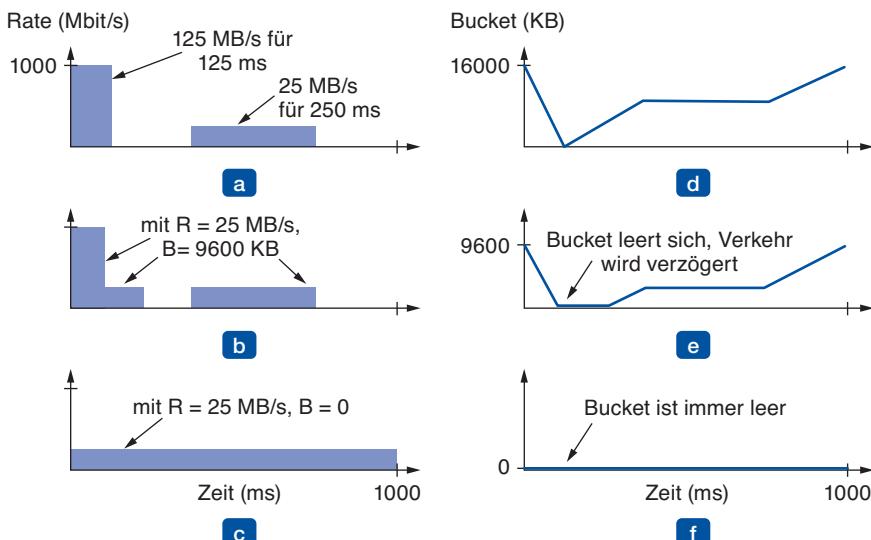


Abbildung 5.29: (a) Verkehr von einem Host. Die Ausgabe ist durch einen Token Bucket geglättet auf $R = 200$ Mbit/s und (b) $B = 9\,600$ KB und (c) $B = 0$ KB. Füllstand des Token Bucket für Traffic-Shaping mit $R = 200$ Mbit/s und (d) $B = 16\,000$ KB, (e) $B = 9\,600$ KB und (f) $B = 0$ KB.

Stellen wir uns nun vor, dass die Router nur für kurze Intervalle Daten mit der Spitzengeschwindigkeit annehmen können, bis ihre Puffer aufgefüllt sind. Die Puffergröße beträgt 9 600 KB, ist also kleiner als der Datenburst. Bei langen Intervallen arbeiten die Router am besten mit Raten, die 200 Mbit/s nicht überschreiten (zum Beispiel, weil dies die gesamte Bandbreite ist, die dem Kunden zur Verfügung steht). Als Folge davon wird Verkehr, der mit diesem Muster gesendet wird, einen Teil der Pakete im Netz verlieren, weil sie nicht in die Puffer der Router passen.

Um diesen Paketverlust zu vermeiden, können wir beim Host Traffic-Shaping mit dem Token-Bucket-Algorithmus anwenden. Wenn wir eine Rate R von 200 Mbit/s und eine

Kapazität B von 9 600 KB benutzen, dann liegt der Datenverkehr innerhalb der Grenzen, in denen das Netz die Daten verarbeiten kann. Die Ausgabe dieses Token Bucket ist in ► Abbildung 5.29b dargestellt. Der Host kann eine kurze Zeit Vollgas geben und mit 1 000 Mbit/s senden, bis das Bucket leer ist. Dann muss er auf 200 Mbit/s zurück-schrauben, bis der Burst gesendet wurde. Das Senden des Bursts wird also über die Zeit verteilt, weil der Burst nicht auf einmal verarbeitet werden kann. Der Füllstand des Token Bucket ist in ► Abbildung 5.29e gezeigt. Anfangs ist das Bucket voll, dann wird es vom ersten Burst aufgebraucht. Wenn der Füllstand null erreicht, können neue Pakete mit der Rate gesendet werden, mit der der Puffer sich füllt. Es kann jetzt keine weiteren Bursts geben, bis das Bucket sich erholt hat. Das Bucket füllt sich, wenn kein Verkehr gesendet wird, und der Füllstand bleibt gleich, wenn Datenverkehr mit der Füllrate gesendet wird.

Traffic-Shaping können wir außerdem anwenden, um zu verhindern, dass der Verkehr in Bursts auftritt. Abbildung 1.29c zeigt die Ausgabe eines Token Bucket mit $R=200$ Mbit/s und einer Kapazität von 0. Dies ist der extreme Fall, bei dem sich Token und Leaky Bucket in ihrer Funktion nicht unterscheiden und der Verkehr vollständig geglättet wurde. Es sind keine Bursts erlaubt und der Verkehr betritt das Netz mit einer gleichbleibenden Rate. Der entsprechende Bucket-Füllstand (► Abbildung 5.29f) ist immer leer. Der Verkehr wird beim Host in einer Warteschlange bis zur Freigabe gespeichert und es ist immer ein Paket vorhanden, das darauf wartet, gesendet zu werden, sobald die Freigabe erteilt wird.

Zu guter Letzt zeigt ► Abbildung 5.29d den Bucket-Füllstand für ein Token Bucket mit $R=200$ Mbit/s und einer Kapazität von $B=16\,000$ KB. Dies ist das kleinste Token Bucket, das der Verkehr unverändert passieren kann. Dieses Bucket könnte von einem Router im Netz genutzt werden, der Traffic-Policing auf den vom Host gesendeten Verkehr anwendet. Falls der Host Verkehr sendet, der mit dem Token Bucket übereinstimmt, auf das sich der Host mit dem Netz geeinigt hat, so wird der Verkehr durch eben dieses Token Bucket passen, welches vom Router an den Randbereichen des Netzes ausgeführt wird. Falls der Host mit einer schnelleren Rate oder in Bursts sendet, wird dem Token Bucket das „Wasser“ ausgehen. In diesem Fall weiß ein Traffic-Policier, dass der Verkehr nicht wie vorher beschrieben aussieht. Je nachdem, auf welchem Entwurf das Netz basiert, werden dann entweder die überschüssigen Pakete verworfen oder deren Priorität verringert. In unserem Beispiel leert sich das Bucket nur vorübergehend zum Ende des anfänglichen Bursts, dann erholt es sich ausreichend für den nächsten Burst.

Leaky Buckets und Token Buckets sind einfach zu implementieren. Wir werden nun die Arbeitsweise eines Token Bucket beschreiben. Bisher haben wir den Zu- und Ablauf des Bucket als kontinuierlich dargestellt, reale Implementierungen müssen jedoch mit diskreten Quantitäten arbeiten. Ein Token Bucket wird mit einem Zähler für den Bucket-Füllstand implementiert. Der Zähler wird bei jedem Taktimpuls von ΔT s um $R/\Delta T$ Einheiten hochgesetzt. In unserem obigen Beispiel wären das 200 kbit jede 1 ms. Jedes Mal, wenn eine Einheit des Datenverkehrs in das Netz gesendet wird, wird der Zähler dekrementiert, und es kann so lange gesendet werden, bis der Zähler null erreicht.

Wenn die Pakete alle dieselbe Größe haben, kann der Bucket-Füllstand einfach in Paketen gezählt werden (z.B. 200 Mbit entsprechen 20 Paketen von 1 250 Byte). Häufig werden jedoch Pakete mit variabler Größe verwendet. In diesem Fall wird der Bucket-Füllstand in Byte gezählt. Falls die verbliebene Bytezahl zu niedrig ist, um ein großes Paket zu senden, muss das Paket bis zum nächsten Taktimpuls warten (oder sogar länger, falls die Füllrate klein ist).

Die Berechnung der Länge des maximalen Bursts (bis das Bucket leer ist) ist nicht ganz einfach. Sie ist länger als einfach nur 9 600 KB geteilt durch 125 MB/s, weil während der Ausgabe des Bursts weitere Token ankommen. Wenn wir die Burst-Länge mit S Sekunden bezeichnen, die maximale Ausgaberate M Byte/s, die Token-Bucket-Kapazität mit B Byte und die Token-Ankunftsrate mit R Byte/s, dann zeigt sich, dass ein Ausgabe-Burst maximal $B+RS$ Byte enthält. Wir wissen auch, dass die Anzahl von Bytes in einem Höchstgeschwindigkeits-Burst der Länge S Sekunden MS ist. Somit erhält man:

$$B + RS = MS$$

Diese Gleichung kann man auflösen, um $S=B/(M-R)$ zu erhalten. Mit unseren Parametern $B=9\,600$ KB, $M=125$ MB/s und $R=25$ MB/s erhalten wir eine Burst-Zeit von ca. 94 ms.

Ein mögliches Problem des Token-Bucket-Algorithmus ist, dass er große Bursts bis hin zur langfristigen Rate R reduziert. Häufig ist es wünschenswert, die Spitzenrate zu reduzieren, aber nicht wieder bis zur langfristigen Rate zurückzugehen (und auch ohne die langfristige Rate zu erhöhen, um mehr Verkehr im Netz zuzulassen). Eine Möglichkeit, den Datenverkehr auszugleichen, besteht im Einfügen eines zweiten Token Bucket bzw. eines Leaky Bucket nach dem ersten. Die Rate des zweiten Bucket sollte viel höher als die des ersten sein. Das erste Bucket charakterisiert den Verkehr und setzt seine durchschnittliche Rate fest, erlaubt aber einige Bursts. Das zweite Bucket reduziert die Spitzenrate, mit der Bursts in das Netz gesendet werden. Wenn zum Beispiel die Rate des zweiten Token Bucket auf 500 Mbit/s und die Kapazität auf 0 eingestellt ist, dann wird der erste Burst das Netzwerk mit einer Spitzenrate von 500 Mbit/s betreten, was niedriger ist als die 1 000-Mbit/s-Rate, die wir vorher hatten.

Das Benutzen all dieser Buckets ist ein wenig knifflig. Wenn Token Buckets für Traffic-Shaping beim Host verwendet werden, dann werden Pakete in Warteschlangen gespeichert und verzögert, bis die Buckets es erlauben, dass sie gesendet werden. Wenn Token Buckets für Traffic-Policing bei den Routern im Netz benutzt werden, so wird der Algorithmus simuliert, um sicherzustellen, dass nicht mehr Pakete als zulässig gesendet werden. Dennoch stellen diese Werkzeuge verschiedene Möglichkeiten zur Verfügung, um den Netzverkehr in eine besser verwaltbare Form zu bringen, damit die Anforderungen hinsichtlich der Dienstgüte erfüllt werden können.

5.4.3 Scheduling der Pakete

Die Regulierung der Form des angebotenen Datenverkehrs ist ein guter Ausgangspunkt. Um jedoch eine Performanzgarantie zu bieten, müssen wir ausreichend Ressourcen entlang der Route reservieren, auf der die Pakete durch das Netz wandern. Dazu nehmen wir an, dass die Pakete eines Datenflusses den gleichen Weg nehmen müssen. Wenn man sie zufällig und wild über die Router verteilt, kann man schwerlich etwas garantieren. Daher ist eine ähnliche Einrichtung wie eine virtuelle Verbindung von der Quelle zum Ziel erforderlich, und alle Pakete eines Datenflusses müssen diesem Weg folgen.

Algorithmen, die Router-Ressourcen zwischen den Paketen eines Datenflusses und zwischen konkurrierenden Datenflüssen zuteilen, werden **Paket-Scheduling-Algorithmen** genannt. Drei unterschiedliche Arten von Ressourcen können für verschiedene Datenflüsse reserviert werden:

1. Bandbreite
2. Pufferspeicher
3. CPU-Zyklen

Die erste Ressource, Bandbreite, ist die offensichtlichste. Wenn ein Datenfluss 1 Mbit/s benötigt und die Ausgangsleitung eine Kapazität von 2 Mbit/s unterstützt, kann man über diese Leitung nicht drei Datenflüsse leiten. Daher bedeutet die Reservierung von Bandbreite, eine Ausgangsleitung nicht überzubelegen.

Ein zweite Ressource, die oftmals knapp wird, ist der Pufferspeicher. Kommt ein Paket an, dann wird es innerhalb des Routers gepuffert, bis es auf der gewählten Ausgangsleitung übertragen werden kann. Der Zweck eines Puffers besteht darin, kleinere Daten-Bursts abzufangen, wenn die Datenflüsse miteinander konkurrieren. Ist kein Puffer verfügbar, dann muss das Paket verworfen werden, da kein Platz für dessen Aufbewahrung vorhanden ist. Für eine hohe Dienstgüte können einige Puffer für bestimmte Datenflüsse reserviert werden, sodass ein bestimmter Datenfluss nicht mit anderen Datenflüssen um diesen Puffer konkurriert muss. So ist bis zu einem gewissen Maximalwert immer ein Puffer vorhanden, wenn der Datenfluss einen benötigt.

Schließlich können auch die CPU-Zyklen eine rare Ressource sein. Zur Verarbeitung eines Paketes wird CPU-Zeit vom Router benötigt, sodass ein Router nur eine bestimmte Anzahl an Paketen pro Sekunde verarbeiten kann. Auch wenn moderne Router in der Lage sind, die meisten Pakete schnell zu verarbeiten, so erfordern doch einige Pakete aufwendigere CPU-Verarbeitung, wie die ICMP-Pakete, die wir in Abschnitt 5.6 beschreiben. Man muss sicherstellen, dass die CPU nicht überlastet ist, um die zeitgerechte Verarbeitung der Pakete zu gewährleisten.

Paket-Scheduling-Algorithmen allozieren Bandbreite und andere Router-Ressourcen, indem sie festlegen, welches der gepufferten Pakete als Nächstes auf die Ausgabeleitung gesendet wird. Wir haben bereits den einfachsten Scheduling-Mechanismus beschrieben, als wir erklärt haben, wie Router arbeiten. Jeder Router puffert für jede

Ausgabeleitung Pakete in einer Warteschlange, bis diese gesendet werden können. Die Pakete werden in derselben Reihenfolge gesendet, in der sie angekommen sind. Dieser Algorithmus ist als **FIFO** (**First-In, First-Out**) oder auch **FCFS** (**First-Come, First-Serve**) bekannt.

FIFO-Router verwerfen in der Regel neu ankommende Pakete, wenn die Warteschlange voll ist. Da das neu angekommende Paket am Ende der Schlange platziert worden wäre, wird dieses Verhalten **Tailldrop** genannt. Dies ist intuitiv und Sie fragen sich vielleicht, welche Alternativen es gibt. In der Tat entscheidet sich der RED-Algorithmus (siehe Abschnitt 5.3.5) dafür, ein zufällig neu ankommendes Paket zu verwerfen, wenn die durchschnittliche Warteschlangenlänge anwächst. Die anderen Scheduling-Algorithmen, die wir beschreiben werden, erzeugen auch andere Möglichkeiten, um zu entscheiden, welche Pakete fallen gelassen werden, wenn die Puffer voll sind.

FIFO-Scheduling ist einfach zu implementieren, aber es ist nicht geeignet, wenn Dienstgüte wichtig ist. Dies liegt daran, dass bei mehreren Datenflüssen ein Fluss leicht die Performanz der anderen Flüsse beeinträchtigen kann. Falls der erste Datenfluss aggressiv ist und große Bursts von Paketen sendet, werden diese sich in der Warteschlange einnisten. Die Verarbeitung von Paketen in der Reihenfolge ihrer Ankunft bedeutet, dass der aggressive Sender den größten Teil der Kapazität aller Router, durch die seine Pakete laufen, in Beschlag nimmt und die anderen Datenflüsse verhungern, was deren Dienstgüte reduziert. Um dem Ganzen noch die Krone aufzusetzen, werden die Pakete der anderen Datenflüsse, die durchkommen, wahrscheinlich verzögert sein, weil sie in der Warteschlange hinter vielen Paketen des aggressiven Senders sitzen.

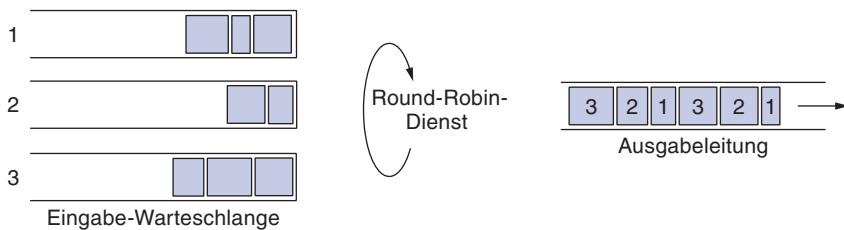


Abbildung 5.30: Round-Robin-Fair-Queueing.

Viele Paket-Scheduling-Algorithmen wurden erfunden, um die Datenflüsse stärker voneinander zu isolieren und um allen Störungssversuchen entgegenzuwirken (Bhatti und Crowcroft, 2000). Einer der ersten war der Algorithmus der fairen Warteschlangenstrategie, **Fair Queueing**, der von Nagle (1987) entwickelt wurde. Im Wesentlichen sieht dieser Algorithmus vor, dass Router mehrere Warteschlangen haben, je eine für jeden Datenfluss pro Ausgangsleitung. Ist eine Leitung im Leerlaufzustand, so durchsucht der Router die Warteschlangen im Round-Robin-Verfahren, wie in ► Abbildung 5.30 gezeigt. Dann nimmt der Router das erste Paket der nächsten Warteschlange heraus. Auf diese Weise erhält jeder Host Gelegenheit, eines von n Paketen zu senden, wenn n Hosts um die Ausgangsleitung konkurrieren. Dieses Verfahren ist in der Hinsicht fair, dass alle Datenflüsse ihre Pakete mit derselben Rate senden können. Durch Versenden mehrerer Pakete wird diese Rate nicht verbessert.

Der Algorithmus ist zwar ein guter Ausgangspunkt, hat aber eine Schwachstelle: Hosts, die große Pakete übertragen, bekommen mehr Bandbreite als Hosts, die kleine Pakete senden. Demers et al. (1990) haben eine Verbesserung vorgeschlagen, bei der das Round-Robin-Verfahren so erfolgt, dass anstelle eines paketweisen ein byteweises Round-Robin simuliert wird. Der Trick ist, eine virtuelle Zeit zu berechnen, die die Nummer der Runde angibt, in der das Senden eines Pakets beendet sein wird. In jeder Runde lassen alle Warteschlangen, die Daten zu senden haben, ein Byte abfließen. Die Pakete werden dann nach ihren Beendigungszeiten sortiert und in dieser Reihenfolge übertragen.

Dieser Algorithmus sowie ein Beispiel von Beendigungszeiten für Pakete, die in drei Datenflüssen ankommen, sind in ►Abbildung 5.31 dargestellt. Falls ein Paket die Länge L hat, wird es einfach L Runden nach seiner Startzeit beendet sein. Die Startzeit ist entweder die Beendigungszeit des vorherigen Pakets oder die Ankunftszeit des Pakets, falls die Warteschlange leer ist, wenn es ankommt.

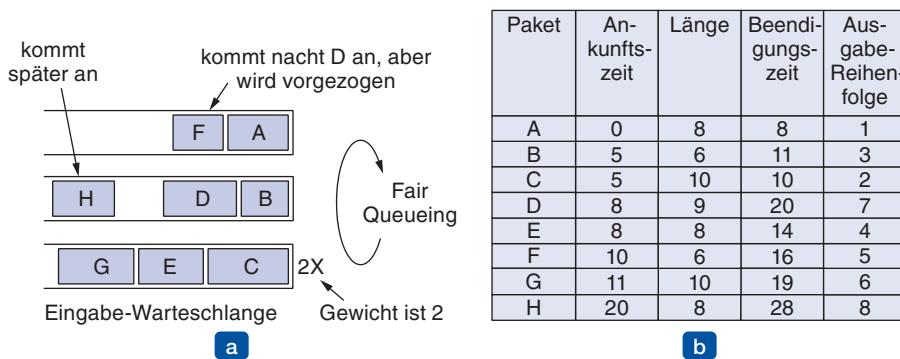


Abbildung 5.31: (a) Gewichtetes Fair Queueing. (b) Beendigungszeiten der Pakete.

Betrachtet man nur die jeweils ersten zwei Pakete in den beiden oberen Warteschlangen, so kann man der Tabelle in ►Abbildung 5.31b entnehmen, dass die Pakete in der Reihenfolge A, B, D und F ankommen. Paket A kommt in Runde 0 an und ist 8 Byte lang, also ist die Beendigungszeit von A Runde 8. Ähnlich ist die Beendigungszeit für Paket B 11. Paket D kommt an, während B gesendet wird. Die Beendigungszeit von D ist 9 Byte-Runden nach der Startzeit, wenn B endet, oder 20. Ebenso berechnet man, dass die Beendigungszeit für F 16 ist. Falls keine neuen Pakete ankommen, ist die Sendereihenfolge A, B, F, D, obwohl F später als D angekommen ist. Möglicherweise kommt noch ein anderes kleines Paket auf dem oberen Datenfluss an, welches dann eine Beendigungszeit vor D erhält. Es wird nur dann vor D springen, falls die Übertragung von diesem Paket noch nicht begonnen hat. Fair Queueing unterbricht keine Pakete, die gerade übermittelt werden. Da Pakete als Ganzes gesendet werden, ist Fair Queueing nur eine Annäherung an das ideale Byte-für-Byte-Schema. Doch es ist eine sehr gute Annäherung, die innerhalb einer Paketübertragung die ganze Zeit beibehalten wird.

Ein Manko dieses Algorithmus ist, dass alle Hosts die gleiche Priorität haben. In vielen Situationen ist es wünschenswert, beispielsweise Videoservern mehr Bandbreite als normalen Dateiservern zu geben. Dies kann einfach realisiert werden, indem dem

Videoserver zwei oder mehr Bytes pro Runde zugestanden werden. Dieser geänderte Algorithmus heißt **Weighted Fair Queueing (WFQ)**, gewichtete faire Warteschlangenstrategie). Wenn wir das Gewicht eines Datenflusses W als Anzahl der Bytes pro Runde festlegen, so können wir nun die Formel zur Berechnung der Beendigungszeit angeben:

$$F_i = \max(A_i, F_{i-1}) + L_i / W$$

wobei A_i die Ankunftszeit ist, F_i die Beendigungszeit und L_i die Länge von Paket i . Die untere Warteschlange von ► Abbildung 5.31a hat ein Gewicht von 2, daher werden diese Pakete schneller gesendet, wie Sie an den Beendigungszeiten in Abbildung 5.31b sehen können.

Eine weitere praktische Überlegung ist die Komplexität der Implementierung. WFQ verlangt, dass Pakete nach ihrer Beendigungszeit in eine sortierte Warteschlange eingefügt werden. Mit N Datenflüssen sind dies bestenfalls $O(\log N)$ Operationen pro Paket, was für viele Datenflüsse in Hochgeschwindigkeitsroutern schwierig zu erreichen ist. Shreedhar und Varghese (1995) beschreiben eine Annäherung, die **Deficit-Round-Robin** genannt wird und die sehr effizient mit nur $O(1)$ Operationen pro Paket implementiert werden kann. WFQ wird häufig in dieser Annäherung eingesetzt.

Es gibt auch noch andere Arten von Scheduling-Algorithmen. Ein einfaches Beispiel ist Prioritätsscheduling, bei dem jedem Paket eine Priorität zugewiesen wird. Pakete mit hoher Priorität werden immer vor den zwischengespeicherten Paketen mit niedriger Priorität gesendet. Innerhalb einer Prioritätsklasse werden Pakete immer in FIFO-Reihenfolge gesendet. Prioritätsscheduling hat jedoch den Nachteil, dass ein Burst von Paketen mit hoher Priorität die Pakete mit niedriger Priorität verhungern lassen kann, weil diese unter Umständen unendlich lange warten müssen. WFQ bietet häufig eine bessere Alternative. Wenn der Warteschlange mit hoher Priorität ein großes Gewicht, zum Beispiel 3, zugeteilt wird, dann werden Pakete mit hoher Priorität häufig nur kurz warten müssen (da nur relativ wenige Pakete eine hohe Priorität haben sollten), sodass trotzdem ein Teil der Pakete mit niedriger Priorität weiterhin gesendet werden kann. Ein System mit hoher und niedriger Priorität ist im Grunde ein Zwei-Warteschlangen-WFQ-System, bei dem die hohe Priorität unendliches Gewicht hat.

Kommen wir nun zum letzten Beispiel einer Scheduling-Strategie. Hierbei tragen Pakete einen Zeitstempel und werden in der Reihenfolge ihrer Zeitstempel gesendet. Clark et al. (1992) beschreiben einen Entwurf, bei dem der Zeitstempel dazu dient festzuhalten, wie lange das Paket im Rückstand ist bzw. vor seinem Zeitplan liegt, während es durch eine Reihe von Routern auf dem Pfad gesendet wird. Pakete, die hinter anderen Paketen in der Warteschlange eingereiht werden, sind tendenziell eher im Rückstand und die Pakete, die zuerst bedient werden, sind ihrem Zeitplan eher voraus. Werden Pakete in der Reihenfolge ihrer Zeitstempel gesendet, so hat dies den günstigen Effekt, dass langsame Pakete beschleunigt werden, während gleichzeitig schnelle Pakete verlangsamt werden. Daher können alle Pakete vom Netz mit einer gleichmäßigeren Übertragungszeit zugestellt werden.

5.4.4 Zugangssteuerung

Wir haben jetzt all die notwendigen Elemente für Dienstgüte einzeln betrachtet, nun ist es Zeit, diese für den praktischen Gebrauch zusammenzusetzen. Dienstgütegarantien werden durch den Prozess der Zugangssteuerung gebildet. Bisher hatten wir Zugangssteuerung als Mittel angesehen, um Überlastung zu steuern, was eine Performanzgarantie darstellt, wenngleich eine schwache. Die Garantien, die wir uns nun ansehen wollen, sind stärker, das Modell ist jedoch dasselbe. Der Benutzer übergibt dem Netz einen Datenfluss zusammen mit einer Dienstgüteanforderung. Das Netz entscheidet dann, ob der Datenfluss angenommen oder abgewiesen wird auf Grundlage seiner Kapazität und den Verpflichtungen, die das Netz anderen Datenflüssen gegenüber hat. Falls der Fluss akzeptiert wird, reserviert das Netz im Voraus Kapazitäten bei Routern, um Dienstgüte zu garantieren, wenn Verkehr auf dem neuen Datenfluss gesendet wird.

Die Reservierungen müssen bei allen Routern entlang der Route, auf der die Pakete durch das Netz reisen, vorgenommen werden. Gibt es für einen Router auf dem Pfad keine Reservierungen, so könnte dieser überlastet werden – und ein einziger überlasteter Router kann die Dienstgütegarantien ruinieren. Viele Routing-Algorithmen finden den einzigen besten Pfad zwischen jeder Quelle und jedem Ziel und senden den gesamten Verkehr über diesen Pfad. Dies kann dazu führen, dass einige Datenflüsse abgelehnt werden, wenn es nicht genügend freie Kapazität entlang des besten Pfads gibt. Für einen neuen Datenfluss können Dienstgütegarantien noch angepasst werden, indem für den Datenfluss eine andere Route gewählt wird, der überschüssige Kapazität hat. Dies wird **QoS-Routing** genannt. Chen und Nahrstedt (1998) geben einen Überblick über diese Techniken. Es ist auch möglich, den Verkehr für jedes Ziel auf mehrere Pfade aufzuteilen, um leichter überschüssige Kapazität zu finden. Dies kann einfach realisiert werden, indem die Router Pfade mit gleichen Kosten auswählen und den Verkehr gleichmäßig oder im Verhältnis zur Kapazität der ausgehenden Verbindungen auf diese verteilen. Es gibt jedoch auch noch ausgefeilte Algorithmen dafür (Nelakuditi und Zhang, 2002).

Für einen gegebenen Pfad genügt es dann bei der Entscheidung, einen Datenfluss anzunehmen oder zurückzuweisen, nicht, einfach die vom Datenfluss angeforderten Ressourcen (Bandbreite, Puffer, CPU-Zyklen) mit den freien Kapazitäten des Routers in diesen drei Bereichen zu vergleichen. Es ist leider etwas komplizierter. Auch wenn einige Anwendungen ihre Bandbreitenanforderungen kennen mögen, so kennen doch nur wenige die Anforderungen bezüglich der Puffer oder CPU-Zyklen, sodass mindestens noch eine andere Kategorie erforderlich ist, um einen Datenfluss zu beschreiben und um diese Beschreibung auf Router-Ressourcen zu übersetzen. Wir werden in Kürze darauf eingehen.

Darüber hinaus sind einige Anwendungen gegenüber einer gelegentlich nicht eingehaltenen Deadline sehr viel toleranter als andere. Die Anwendungen müssen aus den Arten von Garantien wählen, die das Netz geben kann, ob harte Garantien oder Verhalten, das meistens gelten soll. Unter ansonsten gleichen Bedingungen würde jeder lieber harte Garantien bekommen, doch diese sind teuer, da sie Worst-Case-Verhalten erzwingen. Garantien, die für den Großteil der Pakete gelten, sind für Anwendungen

häufig ausreichend und es können mehr Datenflüsse mit dieser Garantie für eine feste Kapazität unterstützt werden.

Schließlich können einige Anwendungen bereit sein, um die Flussparameter zu feilschen, wohingegen andere dies nicht sind. So kann beispielsweise ein Programm zum Abspielen von Filmen mit einer Wiedergaberate von 30 Rahmen/s bereit sein, auf 25 Rahmen/s zurückzugehen, wenn nicht ausreichend Bandbreite zur Unterstützung der 30 Rahmen/s verfügbar ist. Ebenso können die Anzahl der Pixel pro Rahmen, die Audiobandbreite und andere Eigenschaften anpassbar sein.

Da bei der Aushandlung des Datenflusses viele Parteien beteiligt sein können (der Sender, der Empfänger und alle dazwischenliegenden Router), müssen die Datenflüsse mit den betreffenden zu verhandelnden Parametern genau beschrieben werden. Eine Menge solcher Parameter nennt man **Flussspezifikation** (*flow specification*). Ein Sender (z.B. ein Videoserver) erzeugt in der Regel eine Flussspezifikation, die die zu verwendenden Parameter vorschlägt. Wenn die Spezifikation den Weg durchläuft, wird sie von jedem Router untersucht und die Parameter werden nach Bedarf angepasst. Die Änderungen können den Datenfluss nur reduzieren, nicht aber erhöhen (d.h. eine niedrigere Datenübertragungsrate, keine höhere). Wenn die Spezifikation am anderen Ende ankommt, können die Parameter festgelegt werden.

Ein Beispiel für den Inhalt einer Flussspezifikation ist in ▶ Abbildung 5.32 dargestellt, das auf den RFCs 2210 und 2211 für integrierte Dienste basiert, ein Dienstgüteentwurf, den wir im nächsten Abschnitt behandeln. Die Spezifikation verfügt über fünf Parameter. Die ersten beiden Parameter, die *Token-Bucket-Rate* und die *Token-Bucket-Größe*, verwenden ein Token Bucket, um die maximale Übertragungsrate, die der Sender im Mittel über ein langes Zeitintervall übertragen darf, und den größten Burst, den er über ein kurzfristiges Intervall senden kann, zu geben.

| Parameter | Einheit |
|---------------------|---------|
| Token-Bucket-Rate | Byte/s |
| Token-Bucket-Größe | Byte |
| Spitzendatenrate | Byte/s |
| Minimale Paketgröße | Byte |
| Maximale Paketgröße | Byte |

Abbildung 5.32: Beispiel einer Flussspezifikation.

Der dritte Parameter, *Spitzendatenrate*, ist die selbst bei kurzen Zeitintervallen maximal tolerierte Übertragungsrate. Der Sender darf diese Übertragungsrate nie überschreiten, auch nicht für kurze Bursts.

Die letzten beiden Parameter geben die minimale und die maximale Paketgröße an, einschließlich der Header der Transport- und Vermittlungsschicht (d.h. TCP und IP).

Die Mindestgröße ist nützlich, weil die Verarbeitung eines Pakets eine bestimmte festgelegte Zeitspanne dauert, selbst wenn diese noch so kurz ist. Ein Router kann unter Umständen 10 000 Pakete/s mit je 1 KB verarbeiten, nicht aber 100 000 Pakete/s mit je 50 Byte, selbst wenn dies eine niedrigere Datenübertragungsrate darstellt. Die maximale Paketgröße ist aufgrund der internen Netzbeschränkungen wichtig, die nicht überschritten werden dürfen. Wenn beispielsweise ein Teil des Pfads über ein Ethernet verläuft, ist die maximale Paketgröße auf maximal 1 500 Byte beschränkt, unabhängig davon, was das restliche Netz verarbeiten kann.

Eine interessante Frage ist, wie ein Router eine Flussspezifikation in eine Menge bestimmter Ressourcenreservierungen umwandelt. Auf den ersten Blick sieht es so aus, dass falls ein Router eine Verbindung hat, die mit beispielsweise 1 Gbit/s läuft, und ein durchschnittliches Paket 1 000 Bit groß ist, er dann 1 Million Pakete/s verarbeiten könnte. Dies trifft aber nicht zu, weil es aufgrund statistischer Fluktuationen in der Last immer Leerlaufzeiten auf der Verbindung geben wird. Wenn die Verbindung zur Erledigung ihrer Aufgaben jedes bisschen Kapazität benötigt, so bedeutet selbst ein Leerlauf von nur einigen wenigen Bits einen Rückstand, der nie wieder aufgeholt werden kann.

Selbst bei einer Last, die etwas unter der theoretischen Kapazität liegt, können Warteschlangen und Übertragungsverzögerungen entstehen. Betrachten wir eine Situation, in der Pakete zufällig mit einer mittleren Ankunftsrate von λ Pakete/s ankommen. Die Pakete haben zufällige Längen und können auf der Verbindung mit einer mittleren Rate von μ Pakete/s gesendet werden. Unter der Annahme, dass die Ankunftszeiten und Verarbeitungszeiten einer Poisson-Verteilung folgen (was ein M/M/1-Warteschlangensystem genannt wird, wobei „M“ für Markov, d.h. Poisson, steht), kann man mithilfe der Warteschlangentheorie beweisen, dass die mittlere Übertragungsverzögerung pro Paket T wie folgt lautet:

$$T = \frac{1}{\mu} \times \frac{1}{1 - \lambda / \mu} = \frac{1}{\mu} \times \frac{1}{1 - \rho}$$

wobei $\rho = \lambda / \mu$ die CPU-Auslastung ist. Der erste Faktor $1/\mu$ gibt an, wie die Verarbeitungszeit aussieht, wenn kein Wettbewerb vorliegt. Der zweite Faktor ist die Verlangsamung aufgrund des Wettbewerbs mit anderen Datenflüssen. Ist beispielsweise $\lambda = 950\,000$ Pakete/s und $\mu = 1\,000\,000$ Pakete/s, dann ist $\rho = 0,95$ und die mittlere Übertragungsverzögerung pro Paket beträgt 20 μ s anstelle von 1 μ s. Diese Zeit beinhaltet sowohl die Zeit in der Warteschlange als auch die Dienstzeit. Dies ist gut erkennbar, wenn die Last sehr gering ist ($\lambda / \mu \approx 0$). Wenn der Datenfluss beispielsweise 30 Router durchläuft, beträgt alleine die Übertragungsverzögerung durch die Warteschlangen 600 μ s.

Eine Methode, wie Flussspezifikationen mit Router-Ressourcen bezüglich Garantien für Bandbreite- und Übertragungszeit verbunden werden können, wird von Parekh und Gallagher (1993, 1994) beschrieben. Das Verfahren basiert auf Verkehrsquellen, die von (R,B) -Token-Buckets geglättet werden und WFQ beim Router einsetzen. Jedem Datenfluss wird ein WFQ-Gewicht W zugeteilt, das groß genug ist, um seine Token-Bucket-Rate R abfließen zu lassen (►Abbildung 5.33). Wenn z.B. der Datenfluss eine Rate von 1 Mbit/s hat und der Router und die Ausgabeleitung eine Kapazität von

1 Gbit/s, dann muss das Gewicht für den Datenfluss größer als 1/1 000stel des Gesamtgewichts aller Datenflüsse beim Router für die Ausgabeleitung sein. Dies garantiert dem Datenfluss eine minimale Bandbreite. Ist es nicht möglich, dem Datenfluss eine ausreichend große Rate zuzuteilen, so kann der Datenfluss nicht zugelassen werden.

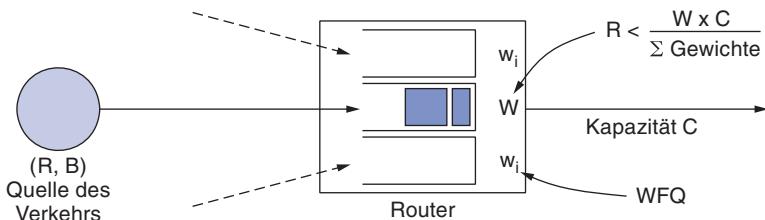


Abbildung 5.33: Bandbreiten- und Übertragungszeitgarantien mit Token Bucket und WFQ.

Die größte Warteschlangenverzögerung, die der Datenfluss erleben wird, ist eine Funktion der Burst-Größe des Token Bucket. Betrachten wir zwei extreme Fälle. Falls der Verkehr völlig ausgeglichen ist, ohne jegliche Bursts, dann werden Pakete den Router genauso schnell verlassen, wie sie ankommen. Es gibt keine Warteschlangenverzögerung (wenn man von den Auswirkungen der Paketisierung einmal absieht). Wenn andererseits der Verkehr für das Senden in Bursts angesammelt wird, dann kann ein Burst der maximalen Größe B mit einem Mal am Router ankommen. In diesem Fall ist die maximale Warteschlangenverzögerung D die Zeit, die benötigt wird, um diesen Burst mit der garantierten Bandbreite oder B/R abließen zu lassen (wiederum werden die Effekte der Paketisierung nicht beachtet). Falls diese Verzögerung zu lang ist, muss der Datenfluss mehr Bandbreite vom Netz anfordern.

Diese Garantien sind hart. Die Token Buckets begrenzen den Burst-Anteil des Verkehrs von der Quelle und Fair Queueing isoliert die Bandbreite, die den verschiedenen Datenflüssen zugeteilt wird. Dies bedeutet, der Datenfluss wird seine Bandbreiten- und Übertragungszeitgarantien erfüllen, unabhängig davon, wie sich die anderen konkurrierenden Datenflüsse beim Router verhalten. Diese anderen Datenflüsse können die Garantien nicht brechen, selbst wenn sie allen Verkehr ansammeln und auf einmal schicken.

Dies gilt außerdem auch für einen Pfad durch mehrere Router in einer beliebigen Netztopologie. Jeder Datenfluss bekommt eine minimale Bandbreite, da diese Bandbreite bei jedem Router garantiert wird. Der Grund, warum jeder Datenfluss eine maximale Verzögerung bekommt, ist subtler. Im schlechtesten Fall, wenn ein Daten-Burst beim ersten Router eintrifft und dort mit dem Verkehr der anderen Datenflüsse konkurriert muss, wird er bis zum Maximalwert D verzögert. Diese Verzögerung wird jedoch ebenfalls den Burst glätten. Dies bedeutet wiederum, dass der Burst keine weiteren Warteschlangenverzögerungen bei späteren Routern erfährt. Die gesamte Warteschlangenverzögerung wird somit höchstens D sein.

5.4.5 Integrierte Dienste

Zwischen 1995 und 1997 hat das IETF erhebliche Anstrengungen in die Erarbeitung einer Architektur für Multimedia-Streaming investiert. Das Ergebnis waren über zwei Dutzend RFCs, wobei die RFCs 2205–2210 den Anfang machten. Diese Arbeiten werden im Allgemeinen als **integrierte Dienste** (*integrated services*) bezeichnet. Sie sind sowohl für Unicast- als auch für Multicast-Anwendungen ausgelegt. Ein Beispiel für die Unicast-Anwendung ist ein Benutzer, der einen Videoclip von einer News-Site herunterlädt. Ein Beispiel für Multicasting sind mehrere digitale Fernsehstationen, die ihre Programme als Ströme von IP-Paketen an viele Empfänger an verschiedenen Orten übertragen. Im Folgenden konzentrieren wir uns auf Multicasting, weil Unicasting ein Sonderfall von Multicasting ist.

In vielen Multicast-Anwendungen können Gruppen die Menge ihrer Mitglieder dynamisch ändern. So können z.B. Leute einer Videokonferenz beitreten und nach einer Weile auf eine Seifenoper oder auf den Krocket-Kanal umschalten, wenn sie sich langweilen. Unter diesen Bedingungen funktioniert der Ansatz, die Sender im Voraus Bandbreite reservieren zu lassen, nicht gut. Das würde bedeuten, dass jeder Sender alle Ein- und Austritte des Publikums laufend mitverfolgen muss. Bei einem System zur Übertragung von Kabelfernsehsendungen mit Millionen von Teilnehmern ist das überhaupt undenkbar.

RSVP

Der Hauptteil des IETF-Protokolls für die Architektur der integrierten Dienste, die sichtbar für die Benutzer des Netzes ist, ist **RSVP** (*Resource reSerVation Protocol*, Ressourcen-Reservierungsprotokoll). Es wird in RFC 2205–2210 beschrieben. Mit diesem Protokoll werden Reservierungen vorgenommen. Andere Protokolle dienen dem Versenden von Daten. RSVP ermöglicht mehreren Sendern, an mehrere Empfängergruppen zu übertragen, und einzelnen Empfängern, nach Wunsch Kanäle zu wechseln. Außerdem optimiert es die Nutzung der Bandbreite bei gleichzeitiger Vermeidung von Überlastungen.

In seiner einfachsten Form nutzt das Protokoll Multicast-Routing mit Spannbäumen. Jede Gruppe erhält eine Gruppenadresse. Um an eine Gruppe senden zu können, setzt ein Sender die Adresse der betreffenden Gruppe in seine Pakete. Der Multicast-Routing-Standardalgorithmus baut dann einen Spannbaum auf, der alle Gruppenmitglieder abdeckt. Der Routing-Algorithmus ist nicht Teil des RSVP-Protokolls. Der einzige Unterschied zwischen dieser Methode und dem normalen Multicasting sind einige wenige weitere Informationen, die periodisch an die Gruppe gesendet werden, damit die Router im Baum in ihren Hauptspeichern gewisse Datenstrukturen führen können.

Als Beispiel betrachten wir das Netz in ►Abbildung 5.34a. Host 1 und 2 sind Multicast-Sender, Host 3, 4 und 5 Multicast-Empfänger. Die Sender und Empfänger sind in diesem Beispiel getrennt, aber im Allgemeinen überlappen sich die zwei Mengen. Die Multicast-Bäume für Host 1 und 2 sind in ►Abbildung 5.34b bzw. ►Abbildung 5.34c dargestellt.

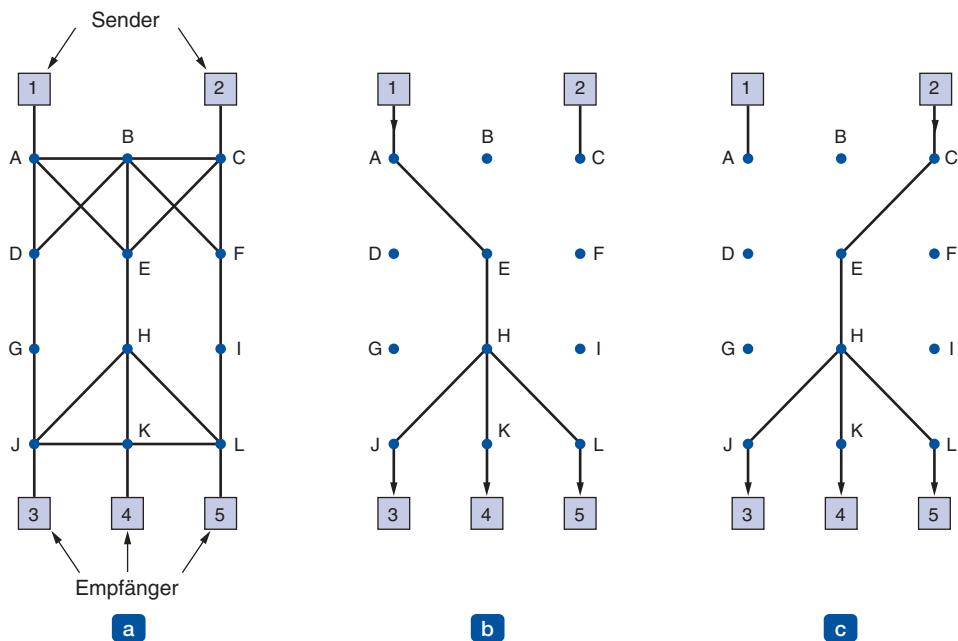


Abbildung 5.34: (a) Ein Netz. (b) Multicast-Spannbaum für Host 1.(c) Multicast-Spannbaum für Host 2.

Um einen besseren Empfang zu erhalten und Überlastung zu vermeiden, kann jeder beliebige Empfänger einer Gruppe eine Reservierungsnachricht den Baum hinauf zum Sender schicken. Die Nachricht wird über den an früherer Stelle beschriebenen Reverse-Path-Forwarding-Algorithmus weitergeleitet. Auf jeder Teilstrecke nimmt der Router die Reservierung zur Kenntnis und reserviert die erforderliche Bandbreite. Wir haben im vorigen Abschnitt gesehen, wie ein gewichteter Fair-Queueing-Scheduler eingesetzt werden kann, um diese Reservierung vorzunehmen. Ist nicht genügend Bandbreite verfügbar, meldet er einen Fehler. Bis die Nachricht wieder bei der Quelle ankommt, wurde die Bandbreite über die gesamte Strecke vom Sender bis zu dem Empfänger reserviert, der die Reservierungsanfrage im Spannbaum gemacht hatte.

► Abbildung 5.35a zeigt ein Beispiel einer solchen Reservierung. Hier hat Host 3 einen Kanal zu Host 1 angefordert. Nach dem Aufbau können Pakete ohne Überlastung von 1 nach 3 fließen. Nun betrachten wir, was passiert, wenn Host 3 als Nächstes einen Kanal zu Host 2 reserviert, damit der Benutzer zwei Fernsehprogramme gleichzeitig sehen kann. Ein zweiter Pfad wird reserviert, wie in ► Abbildung 5.35b gezeigt. Von Host 3 zu Router E sind zwei getrennte Kanäle erforderlich, weil zwei unabhängige Datenströme übertragen werden.

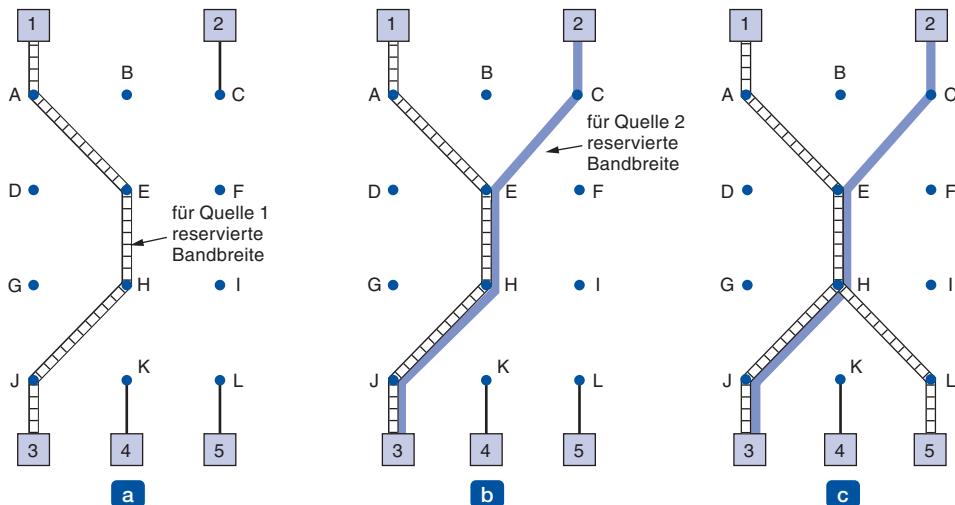


Abbildung 5.35: (a) Host 3 fordert einen Kanal zu Host 1 an. (b) Host 3 fordert dann einen zweiten Kanal zu Host 2 an. (c) Host 5 fordert einen Kanal zu Host 1 an.

In ►Abbildung 5.35c entschließt sich Host 5, ein Programm zu sehen, das von Host 1 übertragen wird. Also nimmt er eine Reservierung vor. Zuerst wird dedizierte Bandbreite bis zu Router H reserviert. Dieser Router erkennt aber, dass er bereits von Host 1 Daten erhält und eine zusätzliche Reservierung nicht nötig ist, falls bereits genügend Bandbreite reserviert wurde. Host 3 und 5 könnten aber auch nach einer unterschiedlichen Bandbreitenmenge gefragt haben (z.B. falls Host 3 auf einem kleinen Bildschirm spielt und nur die geringe Auflösung wünscht). Aus diesem Grund muss die reservierte Kapazität groß genug sein, um den gierigsten Empfänger zufriedenzustellen.

Bei einer Reservierung kann ein Empfänger (optional) eine oder mehrere Quellen angeben, von der bzw. denen er empfangen möchte. Er kann auch angeben, ob diese Auswahl für die Dauer der Reservierung fest ist oder ob er es sich offen halten will, eventuell später zu einer anderen Quelle zu wechseln. Die Router verwenden diese Informationen, um die Bandbreitenplanung zu optimieren. Vor allem werden zwei Empfänger nur für einen gemeinsamen Pfad eingeplant, wenn beide zusagen, die jeweilige Quelle nicht zu wechseln.

Der Grund, warum diese Strategie für den vollkommen dynamischen Fall verwendet wird, ist, dass die reservierte Bandbreite von der Wahl der Quelle abgekoppelt ist. Hat ein Empfänger Bandbreite reserviert, kann er zu einer anderen Quelle wechseln und den Teil des vorhandenen Pfads behalten, der für die neue Quelle gültig ist. Überträgt Host 2 mehrere Videoströme in Echtzeit, z.B. ein TV-Sender mit mehreren Kanälen, so kann Host 3 zwischen ihnen wechseln, ohne seine Reservierung ändern zu müssen. Die Router kümmert es nicht, welches Programm der Empfänger sieht.

5.4.6 Differenzierte Dienste

Flussbasierte Algorithmen können für einen oder mehrere Datenflüsse eine bessere Dienstgüte bereitstellen, da sie die benötigten Ressourcen entlang des Weges reservieren. Dennoch haben sie einen Nachteil. Sie erfordern eine ausgefeilte Einrichtung zur Herstellung eines Datenflusses. Diese Methode skaliert nicht gut, wenn Tausende oder Millionen von Datenflüssen vorliegen. Darüber hinaus unterhalten diese Algorithmen auch einen internen Zustand für jeden Datenfluss in den Routern, sodass sie gegen Routerabstürze anfällig sind. Schließlich sind auch erhebliche Änderungen am Routercode erforderlich, die komplexe Austauschvorgänge zwischen den Routern zum Einrichten des Datenflusses beinhalten. Die Bemühungen, integrierte Dienste weiterzuentwickeln, werden zwar fortgesetzt, dennoch gibt es derzeit nur wenige Implementierungen davon oder etwas Ähnlichem.

Aus diesen Gründen hat die IETF auch einen einfacheren Ansatz für die Dienstgüte entwickelt, der weitgehend lokal an jedem Router ohne vorherige Einrichtung und ohne Einbeziehung des gesamten Pfades implementiert werden kann. Dieser Ansatz wird als **klassenbasierte** (*class-based*) Dienstgüte im Unterschied zu der flussbasierten Dienstgüte bezeichnet. IETF hat eine Architektur dafür standardisiert, die sogenannten **differenzierten Dienste** (*differentiated service*), die in den RFCs 2474, 2475 und anderen beschrieben werden. Wir betrachten diese im Folgenden etwas näher.

Differenzierte Dienste können in Form einer Gruppe von Routern angeboten werden, die eine administrative Domäne bilden (wie ein ISP oder eine Telefongesellschaft). Die Administration definiert eine Menge von Dienstklassen mit den dazugehörigen Weiterleitungsregeln. Wenn sich ein Kunde für differenzierte Dienste anmeldet, werden die Kundenpakete, die in die Domäne eintreten, mit der Klasse gekennzeichnet, zu der sie gehören. Diese Information wird in das Feld *Differentiated Services* von IPv4- oder IPv6-Paketen eingetragen (beschrieben in Abschnitt 5.6). Die Klassen werden über ihr Weiterleitungsverhalten (**Per-Hop Behavior**) definiert, da sie der Behandlung entsprechen, die die Pakete an jedem Router erhalten, und keine Garantie für das gesamte Netz darstellen. Pakete mit bestimmtem Per-Hop Behavior (z.B. Premiumdienste) erhalten besseren Service als andere Pakete (z.B. Normaldienst). Der Datenverkehr innerhalb einer Klasse muss eventuell einer bestimmten Form entsprechen, wie einem Leaky Bucket mit einer angegebenen Fließrate. Ein geschäftstüchtiger Betreiber wird unter Umständen eine Extragebühr für jedes übertragene Premiumpaket berechnen oder nur N Premiumpakete pro Monat für eine feste Zusatzgebühr zulassen. Beachten Sie, dass bei diesem Schema im Unterschied zu den integrierten Schemata keine Vorabeinrichtung, keine Ressourcenreservierung und keine zeitaufwendige Verhandlung zwischen den Endpunkten eines Datenflusses erforderlich ist. Hierdurch sind differenzierte Dienste leicht zu implementieren.

Klassenbasierte Dienste finden sich auch in anderen Branchen. So bieten Paketzusteller oftmals die Zustellung über Nacht, in zwei oder in drei Tagen an. Fluggesellschaften bieten die erste Klasse, Business-Klasse und Economy-Klasse an. Auch in Fernzügen gibt es in der Regel mehrere Klassen. Selbst die U-Bahn in Paris verfügt über zwei Klassen. Bei

Paketen können sich die Klassen hinsichtlich der Übertragungsverzögerung, Jitter und der Wahrscheinlichkeit, bei einer Überlastung verworfen zu werden, neben anderen Möglichkeiten (aber z.B. nicht durch geräumigere Ethernet-Rahmen) unterscheiden.

Um die Unterscheidung zwischen flussbasierter Dienstgüte und klassenbasierter Dienstgüte deutlicher zu machen, hier ein Beispiel: Telefonieren über das Internet. Bei einem flussbasierten Schema erhält jeder Telefonanruf seine eigenen Ressourcen und Garantien. Bei einem klassenbasierten Schema erhalten alle Telefonanrufe zusammen die für die Klasse Telefonie reservierten Ressourcen. Diese Ressourcen können nicht von Paketen der Klasse Webbrowsen oder anderen Klassen weggenommen werden, aber kein einzelner Telefonanruf bekommt nur für sich private Ressourcen reserviert.

Expedited Forwarding

Der Betreiber legt die Dienstklassen fest. Da aber die Pakete häufig über Netze weitergegeben werden, die von verschiedenen Anbietern betrieben werden, hat IETF einige netzunabhängige Dienstklassen definiert. Die einfachste Klasse ist **Expedited Forwarding** (Express-Weiterleitung), mit der wir beginnen. Sie wird in RFC 3246 beschrieben.

Das Konzept von Expedited Forwarding ist ganz einfach. Es sind zwei Dienstklassen verfügbar: Normal (*regular*) und Express (*expedited*). Der Großteil des Datenverkehrs wird normal, ein begrenzter Teil aber wird beschleunigt übertragen. Die Expresspakete sollten das Netz so durchqueren können, als ob keine anderen Pakete vorhanden wären. Auf diese Art werden sie weniger Verlust, geringe Übertragungsverzögerung und niedrigen Jitter-Service bekommen – genau was für VoIP benötigt wird. Eine symbolische Darstellung dieses „Zwei-Röhren“-Systems wird in ▶ Abbildung 5.36 dargestellt. Beachten Sie, dass es hier nur eine physikalische Leitung gibt. Die beiden logischen Röhren in der Abbildung stellen eine Möglichkeit der Reservierung von Bandbreite für verschiedene Dienstklassen dar, aber keine zweite Leitung.

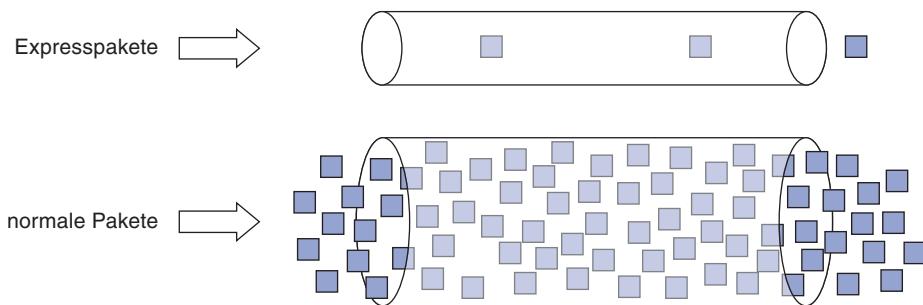


Abbildung 5.36: Expresspakete erleben ein Netz ohne Datenverkehr.

Eine Möglichkeit zur Implementierung dieser Strategie sieht wie folgt aus. Pakete werden als „Express“ oder „Normal“ klassifiziert und entsprechend markiert. Dieser Schritt könnte beim sendenden Host oder am Ingress-Router (dem ersten Router) durchgeführt werden. Der Vorteil, diese Klassifizierung beim sendenden Host vorzunehmen, besteht darin, dass dann mehr Informationen darüber vorliegen, welche

Pakete zu welchem Datenfluss gehören. Diese Aufgabe kann von der Netzsoftware oder sogar vom Betriebssystem durchgeführt werden, um zu vermeiden, existierende Anwendungen zu ändern. Zum Beispiel ist es üblich geworden, VoIP-Pakete für den Expressdienst durch die Hosts markieren zu lassen. Falls die Pakete ein Unternehmensnetz oder ISP passieren, wo der Expressdienst unterstützt wird, dann werden sie bevorzugt behandelt. Unterstützt das Netz keinen Expressdienst, schadet das nicht.

Wird die Markierung vom Host vorgenommen, dann wird natürlich der Ingress-Router den Verkehr überwachen, um sicherzustellen, dass Kunden nur so viel Expressverkehr senden, wie sie bezahlt haben. Innerhalb des Netzes könnten die Router zwei Warteschlangen für jede Ausgangsleitung haben, eine für Expresspaket und eine für normale Pakete. Kommt ein Paket an, so wird es in der entsprechenden Warteschlange eingereiht. Die Expresswarteschlange bekommt eine höhere Priorität als die normale Warteschlange, zum Beispiel indem ein Prioritätsscheduler verwendet wird. Auf diese Weise erleben Expresspaket ein unbelastetes Netz, selbst wenn es in Wirklichkeit eine hohe Belastung mit normalem Datenverkehr gibt.

Assured Forwarding

Ein etwas ausgefeilteres Schema zur Verwaltung von Dienstklassen wird als **Assured Forwarding** (gesicherte Weiterleitung) bezeichnet. Sie wird in RFC 2597 beschrieben. Beim Assured Forwarding werden vier Prioritätsklassen angegeben, wobei jede Klasse ihre eigenen Ressourcen hat. Die drei Topklassen könnten Gold, Silber und Bronze genannt werden. Darüber hinaus werden drei Klassen zum Verwerfen von Paketen definiert, die sich in einer Überlastung befinden: Niedrig, Mittel und Hoch. Diese beiden Faktoren definieren zusammen zwölf Dienstklassen.

► Abbildung 5.37 zeigt eine Möglichkeit der Verarbeitung von Paketen mit Assured Forwarding. Im ersten Schritt werden Pakete einer von vier Prioritätsklassen zugeordnet. Wie vorher kann dies auf dem sendenden Host ausgeführt werden (wie in der Abbildung gezeigt) oder auf dem Ingress-Router. Außerdem kann die Rate von Paketen höherer Priorität durch den Operator als Teil des Dienstangebots begrenzt werden.

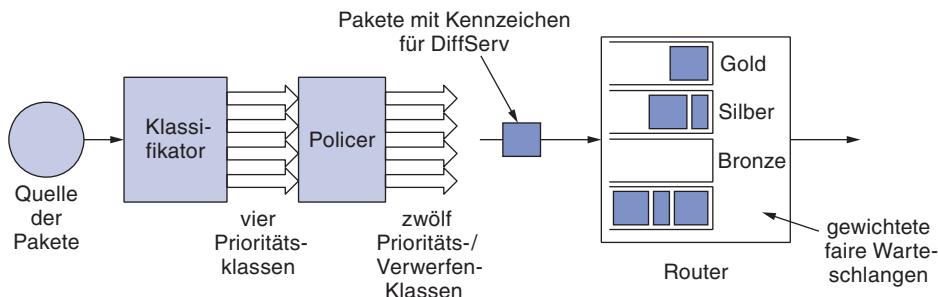


Abbildung 5.37: Eine mögliche Implementierung von Assured Forwarding.

Der nächste Schritt besteht darin, für jedes Paket die Verwerfen-Klasse festzulegen. Dazu werden die Pakete jeder Prioritätsklasse durch einen Traffic-Policier wie ein Token Bucket geleitet. Der Traffic-Policier lässt den gesamten Verkehr passieren, aber

kennzeichnet die Pakete hinsichtlich ihrer Verwerfen-Klasse wie folgt: Pakete, die in kleine Bursts hineinpassen, als „Niedrig“; Pakete, größer als kleine Bursts sind, als „Mittel“; und Pakete, die über große Bursts hinausgehen, als „Hoch“. Die Kombination von Priorität- und Verwerfen-Klassen wird dann in jedem Paket codiert.

Schließlich werden die Pakete von den Netz-Routern mit einem Paket-Scheduler verarbeitet, der die verschiedenen Klassen unterscheidet. Gewöhnlich wird gewichtetes Fair Queueing für die vier Prioritätsklassen verwendet, wobei höhere Klassen höhere Gewichte bekommen. Auf diese Art wird den höheren Klassen der Großteil der Bandbreite zugeteilt, gleichzeitig verhungern aber die niedrigen Klassen nicht völlig. Falls sich zum Beispiel die Gewichte von einer Klasse zur nächsthöheren Klasse verdoppeln, dann werden die höheren Klassen das Doppelte an Bandbreite bekommen. Innerhalb einer Prioritätsklasse können Pakete mit einer höheren Verwerfen-Klasse bevorzugt fallen gelassen werden, dazu kann ein Algorithmus wie RED (*Random Early Detection*) ausgeführt werden, den wir in Abschnitt 5.3.5 kennengelernt haben. RED beginnt mit dem Verwerfen von Paketen, sobald sich Überlastung abzeichnet, aber noch bevor dem Router der Pufferspeicher ausgegangen ist. Zu diesem Zeitpunkt gibt es noch Pufferspeicher, der Pakete mit niedriger Verwerfen-Klasse annimmt, während Pakete mit hoher Verwerfen-Klasse fallen gelassen werden.

5.5 Internetworking

Bislang haben wir implizit angenommen, dass wir es mit einem einzelnen homogenen Netz zu tun haben, das auf jeder Schicht das gleiche Protokoll verwendet. Leider ist diese Annahme viel zu optimistisch. Es gibt viele verschiedene Netze, darunter PANs, LANs, MANs und WANs. Wir haben Ethernet, Internet über Kabel, das Festnetz und das mobile Telefonnetz, IEEE 802.11, IEEE 802.16 und weitere. Auf jeder Schicht werden innerhalb dieser Netze zahlreiche verschiedene Protokolle verwendet. In den folgenden Abschnitten werden die Fragen behandelt, die entstehen, wenn zwei oder mehr Netze zu einem **Internetwork** zusammengeschlossen werden.

Das Vereinen von Netzen wäre viel einfacher, wenn alle eine einzige Netzwerktechnologie verwenden würden, und es ist häufig der Fall, dass es eine vorherrschende Netzart gibt, beispielsweise Ethernet. Einige Experten spekulieren darauf, dass die Fülle an Technologien verschwinden wird, sobald alle gemerkt haben, wie wundervoll [tragen Sie hier Ihr bevorzugtes Netz ein] ist. Rechnen Sie nicht damit. Die Geschichte zeigt, dass dies Wunschdenken ist. Unterschiedliche Arten von Netzen schlagen sich mit unterschiedlichen Problemen herum, sodass sich beispielsweise Ethernet und Satellitennetze wahrscheinlich immer voneinander unterscheiden werden. Die Wiederverwendung vorhandener Systeme, wie Datennetze auf Kabel, dem Telefonnetz oder Stromleitungen auszuführen, fügt Beschränkungen hinzu, die dazu führen, dass die Merkmale der Netze sich in verschiedene Richtungen entwickeln. Heterogenität wird uns erhalten bleiben.

Wenn es immer unterschiedliche Netze geben wird, wäre es einfacher, wir müssten diese nicht miteinander verbinden. Doch das ist ebenfalls unwahrscheinlich. Bob Metcalfe postulierte, dass der Nutzen eines Netzes mit N Knoten die Anzahl von Verbindungen ist, die zwischen Knoten hergestellt werden, oder N^2 (Gilder, 1993). Dies bedeutet, dass große Netze nützlicher sind als kleine Netze, weil sie viel mehr Verbindungen erlauben, daher wird es immer einen Anreiz geben, kleinere Netze miteinander zu kombinieren.

Das Internet ist das Paradebeispiel für diese Vernetzung. Das Ziel der Verbindung dieser Netze ist, dass die Benutzer der verschiedenen Netze miteinander kommunizieren und gegenseitig auf Daten zugreifen können. Wenn Sie einen ISP für Internetdienst bezahlen, werden Sie möglicherweise in Abhängigkeit von der Bandbreite Ihrer Leitung bezahlen, doch wofür Sie eigentlich bezahlen ist die Möglichkeit, Pakete mit jedem anderen Host auszutauschen, der ebenfalls mit dem Internet verbunden ist. Schließlich wäre das Internet nicht sehr erfolgreich, falls Sie nur Pakete zu anderen Hosts in der gleichen Stadt senden könnten.

Da sich Netze aber häufig in wesentlichen Eigenschaften unterscheiden, ist die Übertragung eines Pakets von einem Netz in ein anderes nicht so einfach. Wir müssen Probleme der Heterogenität angehen und ebenfalls Skalierungsprobleme, da das resultierende Internetwork sehr groß wird. Wir werden zunächst untersuchen, worin sich Netze unterscheiden können, um zu verstehen, womit wir es zu tun haben. Dann wollen wir uns den Ansatz ansehen, der so erfolgreich von IP (*Internet Protocol*) verwendet wird, dem Protokoll der Vermittlungsschicht des Internets, einschließlich Techniken zum Tunneling durch Netze, Routing in Internetworks und Paketfragmentierung.

5.5.1 Unterscheidungsmerkmale von Netzen

Netze können sich auf viele Arten unterscheiden. Einige der Unterschiede, wie verschiedene Modulierungstechniken oder Rahmenformate, sind innerhalb der Bitübertragungsschicht oder der Sicherungsschicht anzutreffen. Damit beschäftigen wir uns hier aber nicht. In ► Abbildung 5.38 werden einige Unterschiede aufgeführt, denen die Vermittlungsschicht ausgesetzt sein könnte. Durch diese Unterschiede ist der Betrieb eines Internetworks komplizierter als der eines einzelnen Netzes.

Wenn die von einer Quelle im Netz gesendeten Pakete eines oder mehrere fremde Netze durchlaufen müssen, um ihr Zielnetz zu erreichen, können an den Schnittstellen zwischen den Netzen viele Probleme auftreten. Zunächst muss die Quelle in der Lage sein, das Ziel zu adressieren. Was machen wir, wenn die Quelle in einem Ethernet-Netz und das Ziel in einem WiMAX-Netz liegt? Angenommen, wir könnten sogar ein WiMAX-Ziel von einem Ethernet-Netz aus spezifizieren, dann müssten Pakete von einem verbindungslosen Netz zu einem verbindungsorientierten wechseln. Dies könnte es erforderlich machen, kurzfristig eine neue Verbindung aufzubauen, wodurch eine Verzögerung eingefügt wird – und viel Overhead, wenn diese Verbindung nicht für viele weitere Pakete verwendet wird.

| Merkmal | Mögliche Varianten |
|--------------------|--|
| Angebotener Dienst | Verbindungslos oder verbindungsorientiert |
| Adressierung | Unterschiedliche Größen, flach oder hierarchisch |
| Broadcasting | Vorhanden oder nicht (auch Multicasting) |
| Paketgröße | Jedes Netz hat ein eigenes Maximum |
| Ordnung | Geordnete oder ungeordnete Zustellung |
| Dienstgüte | Vorhanden oder nicht; viele verschiedene Arten |
| Zuverlässigkeit | Verschiedene Ebenen der Verlustraten |
| Sicherheit | Datenschutzregeln, Verschlüsselung usw. |
| Parameter | Verschiedene Timeouts, Flussspezifikationen usw. |
| Abrechnung | Nach Verbindungszeit, Paket, Byte oder keine |

Abbildung 5.38: Eigenschaften, in denen sich Netze unterscheiden.

Unter Umständen müssen auch noch viele spezielle Unterschiede ausgeglichen werden. Wie senden wir ein Paket via Multicasting zu einer Gruppe, von der sich einige Mitglieder in einem Netz befinden, das keinen Multicast unterstützt? Die verschiedenen maximalen Paketgrößen, die von unterschiedlichen Netzen verwendet werden, können ebenfalls ein großes Ärgernis darstellen. Wie überträgt man ein 8 000 Byte großes Paket in einem Netz, dessen maximale Paketgröße 1 500 Byte beträgt? Wenn Pakete eines verbindungsorientierten Netzes ein verbindungsloses Netz durchlaufen, kann es passieren, dass sie in einer anderen Reihenfolge ankommen als in der, in der sie gesendet wurden. Damit rechnet der Sender wahrscheinlich nicht – und es könnte auch für den Empfänger eine (unangenehme) Überraschung sein.

All diese Unterschiede können mit einem gewissen Aufwand kaschiert werden. Zum Beispiel könnte ein Gateway, das zwei Netze vereint, für jedes Ziel separate Pakete erzeugen anstelle einer besseren Netzunterstützung für Multicasting. Ein großes Paket könnte aufgespalten, in Einzelteilen gesendet und dann wieder zusammengesetzt werden. Empfänger könnten Pakete zwischenspeichern und diese der Reihe nach zustellen.

Netze können sich außerdem zu weiten Teilen in Bereichen unterscheiden, die schwieriger miteinander in Einklang zu bringen sind. Das offensichtlichste Beispiel dafür ist die Dienstgüte. Wenn das eine Netz strenge Dienstgüte und das andere Best-Effort-Dienst anbietet, wird es unmöglich sein, Garantien bezüglich Bandbreite und Übertragungszeit für Ende-zu-Ende-Echtzeitverkehr zu geben. In der Tat können Garantien vermutlich nur gemacht werden, während das Best-Effort-Netz mit einer niedrigen Auslastung betrieben oder kaum benutzt wird, was wahrscheinlich nicht dem Ziel der meisten ISPs entspricht. Sicherheitsmechanismen sind problematisch, doch zumindest eine Verschlüsselung für Vertraulichkeit und Datenintegrität kann oberhalb von solchen Netzen angesiedelt werden, die dies noch nicht bieten. Schließ-

lich können Unterschiede in der Abrechnung zu unwillkommenen Rechnungen führen, wenn normale Benutzung plötzlich teuer wird – wie Roaming-Handy-Benutzer mit Datentarifen herausgefunden haben.

5.5.2 Verbindung von Netzen

Es gibt zwei grundlegende Optionen für das Verbinden von unterschiedlichen Netzen: wir können Geräte bauen, die Pakete der einen Netzart in Pakete für jedes andere Netz übersetzen oder konvertieren, oder wir können – wie gute Informatiker – versuchen, das Problem zu lösen, indem wir eine Art Umleitungsschicht hinzufügen und eine gemeinsame Schicht oberhalb der unterschiedlichen Netze bauen. In beiden Fällen werden die Geräte an den Schnittstellen zwischen den Netzen platziert.

Schon früh haben sich Cerf und Kahn (1974) für eine gemeinsame Schicht ausgesprochen und dafür, die Unterschiede der bestehenden Netze zu verborgen. Dieser Ansatz war ungemein erfolgreich und die vorgeschlagene Schicht wurde schließlich in TCP- und IP-Protokolle geteilt. Fast vier Jahrzehnte später ist IP die Grundlage des modernen Internets. Für diese Leistung wurden Cerf und Kahn im Jahr 2004 mit dem Turing Award ausgezeichnet, der allgemein als Nobelpreis der Informatik bezeichnet wird. IP stellt ein universelles Paketformat zur Verfügung, das alle Router erkennen und das fast jedes Netz passieren kann. IP hat seine Reichweite von Rechnernetzen erweitert, um das Telefonnetz zu übernehmen. Es läuft außerdem auf Sensornetzen und anderen winzigen Geräten, von denen einst angenommen wurde, dass sie zuressourcenbeschränkt seien, um IP zu unterstützen.

Wir haben mehrere unterschiedliche Geräte vorgestellt, die Netze verbinden, einschließlich Repeaters, Hubs, Bridges, Router und Gateways. Repeaters und Hubs verschieben einfach Bits von einem Kabel zum anderen. Sie sind größtenteils analoge Geräte und verstehen nichts von Protokollen der höheren Schichten. Bridges und Switches operieren auf der Sicherungsschicht. Sie können zum Aufbau von Netzen verwendet werden, aber nur mit kleineren Protokollübersetzungen, zum Beispiel zwischen 10-, 100- und 1 000-Mbit/s-Ethernet-Switches. Unser Fokus in diesem Abschnitt liegt auf Vernetzungsgeräten, die auf der Vermittlungsschicht operieren, also den Routern. Wir werden die Behandlung von Gateways, die Geräte zum Verbinden auf höheren Ebenen, auf später verschieben.

Zunächst wollen wir auf einer höheren Ebene erforschen, wie Vernetzung mit einer üblichen Vermittlungsschicht benutzt werden kann, um ungleiche Netze zu verbinden. Ein Internetwork, das sich aus IEEE-802.11-, MPLS- und Ethernet-Netzen zusammensetzt, ist in ► Abbildung 5.39a dargestellt. Angenommen, der Quellrechner im IEEE-802.11-Netz möchte ein Paket zu dem Zielrechner im Ethernet-Netz senden. Da diese Technologien unterschiedlich sind und sie außerdem durch eine weitere Netzart (MPLS) getrennt sind, wird zusätzliche Verarbeitung an den Schnittstellen zwischen den Netzen nötig sein.

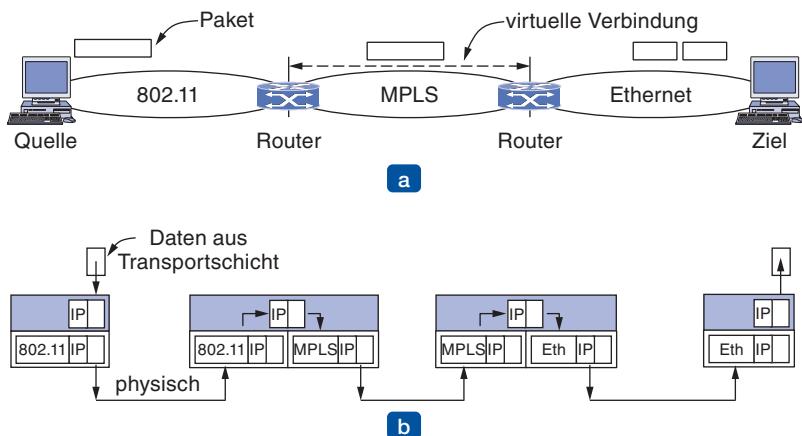


Abbildung 5.39: (a) Ein Paket, das unterschiedliche Netze durchquert. (b) Abarbeitung der Vermittlungsschicht- und Sicherheitsschichtprotokolle.

Da unterschiedliche Netze im Allgemeinen unterschiedliche Adressierungsformen haben können, trägt das Paket eine Vermittlungsschichtadresse, die jeden Host über die drei Netze identifizieren kann. Die erste Grenze, die das Paket erreicht, ist, wenn es von einem IEEE-802.11-Netz in ein MPLS-Netz übergeht. IEEE 802.11 stellt einen verbindungslosen Dienst zur Verfügung, aber MPLS einen verbindungsorientierten Dienst. Das bedeutet, dass eine virtuelle Verbindung aufgebaut werden muss, um dieses Netz zu durchqueren. Reist das Paket erst einmal entlang der virtuellen Verbindung, wird es das Ethernet-Netz erreichen. An dieser Grenze könnte das Paket zu groß sein, um übertragen zu werden, da IEEE 802.11 mit größeren Rahmen als Ethernet arbeiten kann. Um dieses Problem zu behandeln, wird das Paket in Fragmente aufgeteilt, und jedes Fragment wird einzeln gesendet. Wenn die Fragmente das Ziel erreichen, werden sie wieder zusammengesetzt. Dann hat das Paket seine Reise beendet.

Die Abarbeitung der Protokolle für diese Reise ist in ►Abbildung 5.39b dargestellt. Die Quelle akzeptiert Daten von der Transportsschicht und erzeugt ein Paket mit dem gemeinsamen Vermittlungsschicht-Header, welches in diesem Beispiel IP ist. Der Header enthält die endgültige Zieladresse, welche für die Feststellung benutzt wird, dass das Paket über den ersten Router gesendet werden sollte. Das Paket ist also in einem IEEE-802.11-Rahmen verpackt, dessen Ziel der erste Router ist, und wird übermittelt. Beim Router wird das Paket aus dem Datenfeld des Rahmens entfernt und der IEEE-802.11-Rahmen-Header wird verworfen. Der Router untersucht nun die IP-Adresse im Paket und schlägt diese Adresse in seiner Routing-Tabelle nach. Basierend auf dieser Adresse entscheidet er, das Paket als Nächstes zum zweiten Router zu senden. Für diesen Teil des Pfads muss eine virtuelle MPLS-Verbindung zum zweiten Router eingerichtet werden und das Paket muss mit MPLS-Headern verpackt werden, die diese Verbindung bereisen. Am entfernten Ende wird der MPLS-Header verworfen und die Netzwerkadresse wird wiederum konsultiert, um die nächste Teilstrecke der Vermittlungsschicht zu finden. Es ist das Ziel selbst. Da das Paket zu lang ist, um über Ethernet gesendet zu werden, wird es in zwei Teile aufgespalten. Jedes dieser Teile wird in das

Datenfeld eines Ethernet-Rahmens gepackt und zur Ethernet-Adresse des Ziels gesendet. Am Ziel wird der Ethernet-Header aus jedem der Rahmen herausgezogen und die Inhalte werden wieder zusammengesetzt. Das Paket hat zu guter Letzt sein Ziel erreicht.

Beachten Sie, dass es einen entscheidenden Unterschied gibt zwischen der Situation mit Routern und der mit Switches (oder Bridges). Bei einem Router wird das Paket aus dem Rahmen extrahiert und die Netzwerkadresse in dem Paket wird dazu benutzt, zu entscheiden, wohin es zu senden ist. Bei einem Switch (oder einer Bridge) wird der gesamte Rahmen auf Basis seiner MAC-Adresse transportiert. Switches müssen das Protokoll der Vermittlungsschicht nicht verstehen, das zum Weiterleiten der Pakete verwendet wird. Router ja.

Leider ist Internetworking nicht ganz so leicht, wie es sich hier vielleicht anhört. Als Bridges eingeführt wurden, war es tatsächlich beabsichtigt, damit unterschiedliche Netzwerktypen zu vereinen – oder zumindest unterschiedliche LAN-Typen. Dazu sollten die Bridges die Rahmen von einem LAN in Rahmen eines anderen LAN übersetzen. Dies funktionierte jedoch aus dem gleichen Grund nicht gut, aus dem Internetworking schwierig ist: Die Unterschiede in den Merkmalen von LANs, wie verschiedene maximale Paketgrößen und LANs mit und ohne Prioritätsklassen, sind schwer zu verbergen. Heute werden Bridges vorwiegend verwendet, um dieselbe Art von Netz auf der Sicherungsschicht zu verbinden, und Router verbinden unterschiedliche Netze auf der Vermittlungsschicht.

Internetworking war sehr erfolgreich beim Aufbau großer Netze, aber es funktioniert nur, wenn es eine gemeinsame Vermittlungsschicht gibt. In der Tat gab es im Laufe der Zeit viele Netzwerkprotokolle. Es ist schwierig, alle Beteiligten davon zu überzeugen, sich auf ein einziges Format zu einigen, wenn Unternehmen es als ihren wirtschaftlichen Vorteil ansehen, ein proprietäres Format zu besitzen, das sie kontrollieren können. Außer IP, dem heute fast universellen Netzprotokoll, waren da noch IPX, SNA und AppleTalk. Keines dieser Protokolle wird noch häufig eingesetzt, doch es wird immer andere Protokolle geben. Das relevanteste Beispiel sind heute wahrscheinlich IPv4 und IPv6. Obwohl beides Versionen von IP sind, sind sie nicht kompatibel (sonst wäre es nicht nötig gewesen, IPv6 zu entwerfen).

Ein Router, der mehrere Netzprotokolle verarbeiten kann, wird als **Multiprotokoll-Router** bezeichnet. Er muss entweder die Protokolle übersetzen oder die Verbindung einem Protokoll der höheren Schichten überlassen. Keiner der beiden Ansätze ist völlig befriedigend. Für eine Verbindung auf einer höheren Schicht, z.B. indem TCP verwendet wird, ist es erforderlich, dass alle Netze TCP implementieren (was möglicherweise nicht der Fall ist). Damit wird die Nutzung über die Netze hinweg auf Anwendungen begrenzt, die TCP verwenden (was viele Echtzeitanwendungen nicht einschließt).

Die Alternative ist, Pakete zwischen den Netzen zu übersetzen. Falls jedoch die Paketformate nicht nahe Verwandte mit denselben Informationsfeldern sind, werden solche Umformungen immer unvollständig und damit häufig zum Scheitern verurteilt sein. Beispielsweise sind IPv6-Adressen 128 Bit lang. Sie werden nicht in ein 32-Bit-IPv4-Adressfeld passen, egal, wie sehr der Router sich anstrengt. Die gleichzeitige Ausfüh-

rung von IPv4 und IPv6 in einem Netz hat sich als eines der größten Hindernisse bei Verbreitung von IPv6 erwiesen. (Um fair zu bleiben: ein ebenso großes Hindernis war es, Kunden begreiflich zu machen, warum sie überhaupt IPv6 wollen sollten.) Größere Probleme können erwartet werden, wenn zwischen grundlegend verschiedenen Protokollen übersetzt wird, wie verbindungslosen und verbindungsorientierten Netzprotokollen. Angesichts dieser Schwierigkeiten wird nur selten der Versuch einer Konvertierung unternommen. Zugegeben, selbst IP hat nur deshalb so gut funktioniert, weil es eine Art kleinster gemeinsamer Nenner war. IP stellt keine hohen Anforderungen an die Netze, auf denen es ausgeführt wird, bietet aber als Ergebnis auch nur Best-Effort-Dienst.

5.5.3 Tunneling

Die Darstellung des allgemeinen Falls, dass zwei verschiedene Netze miteinander zur Zusammenarbeit verbunden werden, ist extrem schwierig. Andererseits gibt es aber einen bekannten Sonderfall, der selbst für unterschiedliche Netzprotokolle handhabbar ist. Das ist der Fall, wenn die Quell- und Ziel-Hosts am gleichen Netztyp hängen, dazwischen aber ein anderer Netztyp liegt. Als Beispiel kann man sich eine internationale Bank mit einem IPv6-Netz in Paris, einem IPv6-Netz in London und Konnektivität zwischen den Büros über das IPv4-Internet vorstellen. Diese Situation ist in ► Abbildung 5.40 zu sehen.

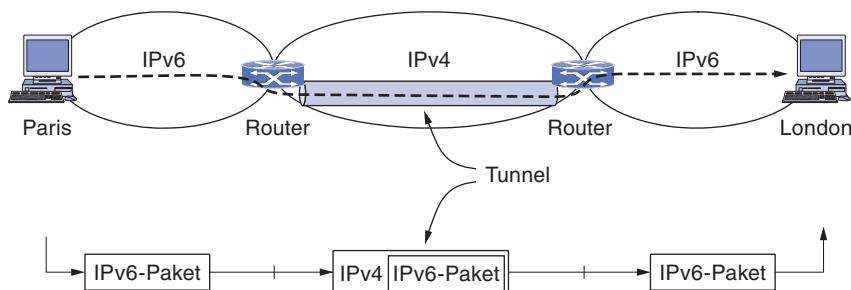


Abbildung 5.40: Tunneling eines Pakets von Paris nach London.

Eine Lösung dieses Problems ist eine Technik namens **Tunneling**. Um ein IP-Paket an einen Host im Londoner Büro zu senden, stellt ein Host im Pariser Büro das Paket mit der IPv6-Adresse in London zusammen und sendet es zu dem Multiprotokoll-Router, der das Pariser IPv6-Netz mit dem IPv4-Internet verbindet. Wenn dieser Router das IPv6-Paket bekommt, verkapselt er das Paket mit einem IPv4-Header, der die IPv4-Seite des Multiprotokoll-Routers adressiert, der mit dem Londoner IPv6-Netz verbunden ist. Das heißt, der Router verpackt ein (IPv6-)Paket in ein (IPv4-)Paket. Wenn dieses eingeschlossene Paket ankommt, entfernt der Londoner Router das ursprüngliche IPv6-Paket und sendet es weiter zum Ziel-Host.

Den Pfad durch das IPv4-Internet kann man sich als riesigen Tunnel vorstellen, der sich von einem Multiprotokoll-Router zum anderen erstreckt. Das IPv6-Paket reist, angenehm in einer Box verpackt, von einem Ende des Tunnels zum anderen. Es muss

sich überhaupt nicht um IPv4 kümmern, ebenso wenig wie die Hosts in Paris oder London. Nur die Multiprotokoll-Router müssen sowohl die IPv4- als auch die IPv6-Pakete verstehen. Die gesamte Reise von einem Multiprotokoll-Router zum anderen ist praktisch wie eine Teilstrecke über eine einzelne Verbindung.

Eine Analogie soll das Konzept des Tunnelings verdeutlichen. Jemand reist mit seinem Wagen von Paris nach London. In Frankreich fährt der Wagen selbst, aber bei der Ankunft am Ärmelkanal wird der Wagen auf einen Hochgeschwindigkeitszug verfrachtet und durch den Tunnel nach England befördert (Autos dürfen nicht selbst im Tunnel fahren). Wie in ►Abbildung 5.41 ersichtlich ist, wird der Wagen als Frachtgut befördert. Am anderen Ende wird der Wagen auf Englands Straßen losgelassen und fährt wieder selbst. Das Tunneling von Paketen durch ein fremdes Netz funktioniert auf ähnliche Weise.

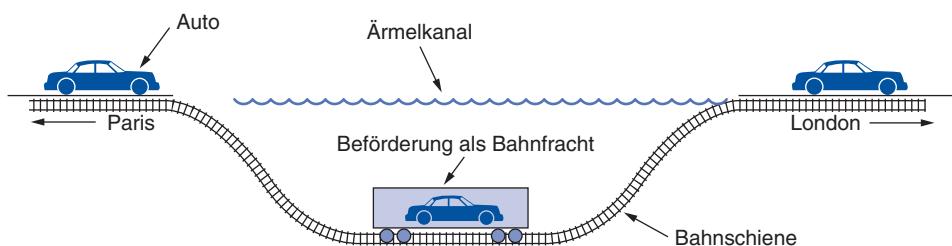


Abbildung 5.41: Beförderung eines Autos von Frankreich nach England.

Tunneling wird oft verwendet, um isolierte Hosts und Netze mithilfe anderer Netze zu verbinden. Das resultierende Netz wird ein **Overlay** genannt, da es praktisch auf dem Basisnetz aufsetzt. Ein häufiges Motiv dafür ist die Installation eines Netzprotokolls mit einer neuen Funktion, wie unser „IPv6 über IPv4“-Beispiel zeigt. Der Nachteil von Tunneling ist, dass keiner der Hosts in dem Netz, durch das der Tunnel hindurchläuft, erreicht werden kann, da die Pakete nicht irgendwo in der Mitte aus dem Tunnel austreten können. Diese Begrenzung des Tunneling wird jedoch in einen Vorteil mit **VPNs (virtuelles privates Netz)** gewandelt. Ein VPN ist einfach ein Overlay, das verwendet wird, um ein Maß an Sicherheit zur Verfügung zu stellen. Wir werden VPNs untersuchen, wenn wir zu *Kapitel 8* kommen.

5.5.4 Internetwork-Routing

Das Routing durch ein Internetwork wirft dasselbe Grundproblem auf wie das Routing innerhalb eines einzelnen Netzes, enthält aber ein paar zusätzliche Komplikationen. Zunächst können die Netze intern unterschiedliche Routing-Algorithmen verwenden. Zum Beispiel könnte das eine Netz Link-State-Routing benutzen und ein anderes Netz den Distanzvektoralgorithmus. Da für das Link-State-Routing die Topologie des Netzes bekannt sein muss, für den Distanzvektoralgorithmus aber nicht, wäre allein durch diesen Unterschied unklar, wie die kürzesten Pfade durch das Internetwork zu finden sind.

Werden Netze von verschiedenen Betreibern verwaltet, dann bringt das noch größere Probleme mit sich. Zunächst einmal haben die Betreiber vermutlich unterschiedliche Vorstellungen davon, was ein guter Pfad durch das Netz ist. Der eine Betreiber möchte vielleicht die Route mit der geringsten Verzögerung, während ein anderer die billigste Route bevorzugt. Dies wird dazu führen, dass die Betreiber unterschiedliche Quantitäten verwenden, um die Kosten des kürzesten Pfads festzulegen (z.B. Millisekunden Verzögerung oder finanzielle Kosten). Die Gewichte sind dann über das Netzwerk hinweg nicht vergleichbar, sodass in dem Internetwork nicht gut definiert ist, was kürzeste Pfade sind.

Schlimmer noch, der eine Betreiber möchte eventuell nicht, dass ein anderer Betreiber überhaupt die Einzelheiten des Pfads in seinem Netz kennt, vielleicht weil die Gewichte und Pfade sensible Informationen widerspiegeln könnten (wie die finanziellen Kosten), die einen Wettbewerbsvorteil darstellen.

Schließlich könnte das Internetwork viel größer sein als jedes der Netze, aus denen es sich zusammensetzt. Dann werden eventuell Routing-Algorithmen erforderlich sein, die mithilfe einer Hierarchie skaliert werden, selbst wenn keines der einzelnen Netze eine Hierarchie verwenden muss.

All diese Überlegungen führen zu einem zweistufigen Routing-Algorithmus. Innerhalb jedes Netzes wird ein **Intradomain-Protokoll** oder **internes Gateway-Protokoll** verwendet. („Gateway“ ist eine ältere Bezeichnung für „Router“.) Es könnte ein Link-State-Protokoll von der Art sein, die wir schon beschrieben haben. Zwischen den Netzen, die das Internetwork bilden, wird ein **Interdomain-Protokoll** oder **externes Gateway-Protokoll** benutzt. Die einzelnen Netze verwenden möglicherweise unterschiedliche Intradomain-Protokolle, aber sie müssen dasselbe Interdomain-Protokoll benutzen. Im Internet wird das Interdomain-Routing-Protokoll **BGP** (*Border Gateway Protocol*) genannt. Wir werden es im nächsten Abschnitt beschreiben.

Es gibt hier noch einen wichtigen Begriff einzuführen. Da jedes Netz von allen anderen Netzen unabhängig ist, wird es häufig als **autonomes System (AS, Autonomous System)** bezeichnet. Ein gutes gedankliches Modell für ein autonomes System ist ein ISP-Netz. In der Tat kann sich ein ISP-Netz aus mehr als einem autonomen System zusammensetzen, wenn es verwaltet wird oder als mehrere Netze erworben wurde. Doch der Unterschied ist in der Regel nicht signifikant.

Die beiden Ebenen sind gewöhnlich nicht streng hierarchisch, da hochgradig suboptimale Pfade entstehen können, wenn ein großes internationales Netz und ein kleines regionales Netz zu einem einzigen Netz zusammengefasst werden. Es werden jedoch relativ wenige Informationen über Routen innerhalb der Netze preisgegeben, um Wege durch das Internetwork zu finden. Mithilfe dieses Konzepts können all die angesprochenen Komplikationen in Angriff genommen werden. Die Skalierung wird verbessert und Betreiber können Routen innerhalb ihrer Netze frei auswählen, indem sie die Protokolle ihrer Wahl verwenden. Es müssen auch keine netzübergreifenden Gewichte verglichen werden und sensible Informationen müssen nicht nach außen preisgegeben werden.

Wir haben bisher jedoch wenig dazu gesagt, wie die Routen durch die Netze des Internetworks festgelegt werden. Im Internet ist ein entscheidender Faktor die Geschäftsvereinbarung zwischen ISPs. Jeder ISP kann den anderen ISPs das Übertragen des Verkehrs in Rechnung stellen. Falls beim Internetwork-Routing internationale Grenzen überquert werden müssen, taucht ein weiterer Faktor auf: eventuell kommen plötzlich verschiedene Gesetze ins Spiel, wie die strengen Datenschutzgesetze in Schweden über den Export persönlicher Daten von schwedischen Bürgern aus Schweden heraus. All diese nicht technischen Faktoren sind im Konzept einer Routing-Strategie enthalten, die überwacht, wie autonome Netze die von ihnen verwendeten Routen auswählen. Wir werden auf Routing-Strategien zurückkommen, wenn wir BGP beschreiben.

5.5.5 Fragmentierung der Pakete

Jedes Netz bzw. jede Verbindung gibt eine bestimmte Höchstgröße für Pakete vor. Diese Grenzen haben verschiedene Ursachen, wie:

1. Hardware (z.B. die Größe eines Ethernet-Rahmens)
2. Betriebssystem (z.B. sind alle Puffer 512 Byte groß)
3. Protokolle (z.B. die Anzahl an Bits im Längenfeld des Pakets)
4. Übereinstimmung mit nationalen und/oder internationalen Standards
5. Das Bestreben, durch Fehler verursachte Neuübertragungen auf ein Mindestmaß zu reduzieren
6. Das Bestreben, ein Paket daran zu hindern, den Kanal zu lange zu belegen

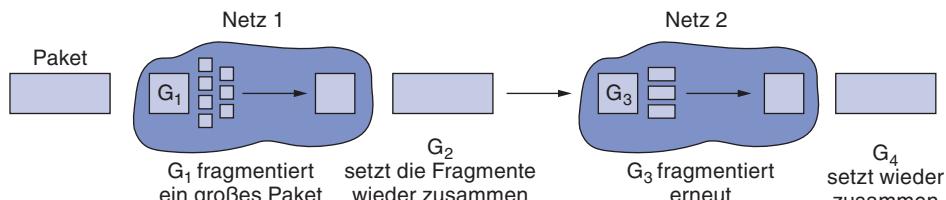
Aus diesen Faktoren ergibt sich, dass die Netzentwickler die alte maximale Paketgröße nicht frei wählen können. Die maximalen Nutzdaten für einige häufige Technologien betragen 1 500 Byte für Ethernet und 2 272 Byte für IEEE 802.11. IP ist etwas großzügiger und lässt Pakete zu, die 65 515 Byte groß sind.

Hosts ziehen es in der Regel vor, große Pakete zu übermitteln, weil dies die Paket-Overheads wie verschwendete Bandbreite an Header-Bytes reduziert. Ein offensichtliches Problem bei Internetworks entsteht, wenn ein großes Paket durch ein Netz reisen will, dessen maximale Paketgröße zu klein ist. Diese Beeinträchtigung war ein hartnäckiges Problem und die Lösungen dafür haben sich zusammen mit der vielen Erfahrung herausgebildet, die im Internet gesammelt wurde.

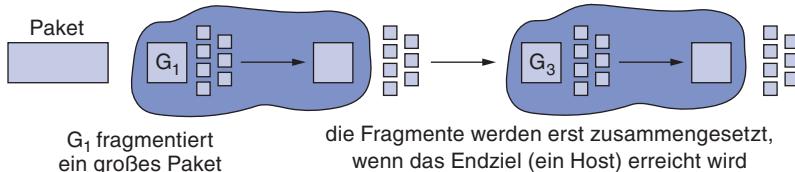
Eine Lösung ist, von vornherein sicherzustellen, dass das Problem erst gar nicht auftritt. Das ist jedoch leichter gesagt als getan. Eine Quelle kennt in der Regel nicht den Pfad, den ein Paket durch das Netz zu einem Ziel nimmt, daher weiß sie sicher auch nicht, wie klein die Pakete sein müssen, um dort anzukommen. Diese Paketgröße wird **Path MTU** (*Path Maximum Transmission Unit*) genannt. Selbst wenn die Quelle die Path MTU nicht kennt, werden Pakete unabhängig in einem verbindungslosen Netz wie dem Internet weitergeleitet. Dieses Routing bedeutet, dass Pfade sich plötzlich ändern können, wodurch sich unerwarteterweise die Path MTU ändern kann.

Die alternative Lösung dieses Problems ist, den Routern zu erlauben, die Pakete in **Fragmente** aufzuteilen und jedes Fragment als getrenntes Vermittlungsschichtpaket zu übertragen. Wie alle Eltern von kleinen Kindern jedoch wissen, ist es aber viel einfacher, ein großes Objekt in kleine Teile zu zerlegen, als umgekehrt. (Physiker haben dafür sogar eine Bezeichnung: das zweite Gesetz der Thermodynamik.) Auch paketvermittelte Netze haben Schwierigkeiten, ein in Fragmente aufgeteiltes Paket wieder zusammenzusetzen.

Für das Zusammensetzen der Fragmente in die Originalpakete gibt es zwei entgegengesetzte Strategien. Bei der ersten Strategie wird die Fragmentierung, die von einem Netz mit ausschließlich kleinen Paketen veranlasst wurde, für alle anderen Netze, die das Paket bis zum Endziel durchläuft, transparent gemacht. Diese Option ist in ►Abbildung 5.42a dargestellt. Kommt bei diesem Ansatz ein über großes Paket bei G_1 an, so teilt der Router das Paket in Fragmente auf. Jedes Fragment wird an den gleichen Ausgangsrouter G_2 adressiert, wo die Einzelteile wieder zusammengesetzt werden. Auf diese Weise ist die Reise durch das Netz mit kleinen Paketen transparent. Die anderen durchlaufenden Netze sind sich nicht einmal dessen bewusst, dass eine Fragmentierung stattgefunden hat.



a



b

Abbildung 5.42: (a) Transparente Fragmentierung. (b) Nicht transparente Fragmentierung.

Die transparente Fragmentierung ist einfach, aber nicht ganz unproblematisch. Zunächst einmal muss der Ausgangsrouter wissen, wann alle Stücke bei ihm angekommen sind, sodass entweder ein Zählerfeld oder ein „Paketende“-Bit in jedes Paket eingefügt werden muss. Da außerdem alle Pakete das Netz über den gleichen Router verlassen müssen, damit sie wieder zusammengesetzt werden können, ist die Auswahl der Routen eingeschränkt. Können nicht alle Fragmente eines Pakets über den gleichen Weg an das Ziel gelangen, kann Leistung verloren gehen. Bedeutsamer ist jedoch der Aufwand, den der Router möglicherweise bewältigen muss. Eventuell muss der Router die Fragmente bei ihrer Ankunft zwischenspeichern und entschei-

den, wann sie zu verwerfen sind, falls nicht alle Fragmente ankommen. Ein Teil dieser Arbeit stellt sich vielleicht auch als Verschwendungen heraus, wenn das Paket beispielsweise eine Reihe von Netzen mit kleiner Paketgröße passieren und wiederholt fragmentiert und zusammengesetzt werden muss.

Bei der nicht transparenten Fragmentierungsstrategie werden die Fragmente auf dazwischenliegenden Routern nicht zusammengesetzt. Die Teile eines einmal fragmentierten Pakets werden so behandelt, als wären sie ein Originalpaket. Die Router geben die Fragmente weiter (siehe ► Abbildung 5.42b), das Zusammensetzen der Teile wird erst am Ziel-Host durchgeführt.

Der Hauptvorteil der nicht transparenten Fragmentierung ist, dass damit den Routern weniger Arbeit abverlangt wird. IP funktioniert auf diese Weise. Ein vollständiger Entwurf erfordert die Durchnummerierung der Fragmente auf solche Weise, dass der ursprüngliche Datenstrom rekonstruiert werden kann. Bei dem Entwurf, den IP verwendet, wird jedes Fragment mit einer Paketnummer (die auf allen Paketen transportiert wird), einem absoluten Byte-Offset innerhalb des Pakets und einem Flag versehen, das angibt, ob das Fragment das Paketende darstellt. Ein Beispiel ist in ► Abbildung 5.43 zu sehen. Obwohl dieser Entwurf einfach ist, hat er doch einige attraktive Eigenschaften. Fragmente können in einem Puffer am Ziel auf dem richtigen Platz für die Wiederzusammensetzung angeordnet werden, selbst wenn sie nicht in der richtigen Reihenfolge ankommen. Fragmente können außerdem erneut fragmentiert werden, falls sie ein Netz mit noch kleinerer MTU durchqueren. Dies ist in ► Abbildung 5.43c dargestellt. Bei einer Neuübertragung des Pakets (falls keines der Fragmente ankommt) kann das Paket in andere Teile zerlegt werden. Schließlich können Fragmente von beliebiger Größe sein, bis hinunter zu einem einzelnen Byte plus dem Paket-Header. In allen Fällen verwendet das Ziel einfach die Paketnummer und den Fragmente-Offset, um die Daten in die richtige Position zu bringen, und das Ende-des-Pakets-Flag, um festzustellen, wann es das ganze Paket erhalten hat.

Leider gibt es noch Probleme mit diesem Entwurf. Der Overhead kann höher sein als bei der transparenten Fragmentierung, weil die Fragment-Header nun auch über Verbindungen übertragen werden, wo sie möglicherweise nicht benötigt werden. Doch das wahre Problem ist die Existenz von Fragmenten überhaupt. Kent und Mogul (1987) argumentieren, dass Fragmentierung nachteilig für die Performanz ist, weil – ebenso wie die Header-Overheads – ein ganzes Paket verloren ist, falls eines seiner Fragmente verloren geht, und weil Fragmentierung eine größere Last für Hosts ist, als man ursprünglich angenommen hatte.

Nummer des ersten elementaren Fragments in diesem Paket

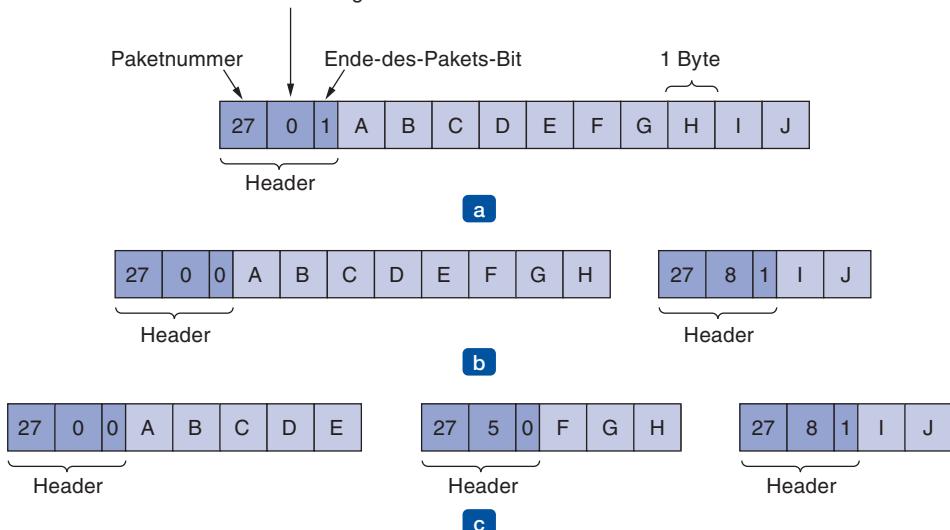


Abbildung 5.43: Fragmentierung bei einer elementaren Datengröße von 1 Byte: (a) Originalpaket mit 10 Datenbytes. (b) Fragmente nach dem Durchqueren eines Netzes mit der maximalen Paketgröße von 8 Bytes Nutzdaten plus Header. c) Fragmente nach dem Durchqueren eines Gateways der Größe 5.

Dies bringt uns zurück zu der ursprünglichen Lösung, Fragmentierung im Netz zu beiseitigen – die Strategie, die im modernen Internet verwendet wird. Der Prozess heißt **Path MTU Discovery** (Mogul und Deering, 1990) und funktioniert folgendermaßen. Jedes IP-Paket wird mit seinen Header-Bits gesendet, um zu markieren, dass keine Fragmentierung erlaubt ist. Falls ein Router ein Paket empfängt, das zu groß ist, erzeugt er ein Fehlerpaket, schickt dieses zurück zur Quelle und verwirft das Paket. Dies ist in ▶ Abbildung 5.44 gezeigt. Wenn die Quelle das Fehlerpaket bekommt, verwendet sie die enthaltene Information, um das Paket in Stücke zu zerlegen, die klein genug sind, dass der Router sie behandeln kann. Falls ein Router im weiteren Verlauf des Pfads eine noch kleinere MTU hat, wird der Prozess wiederholt.

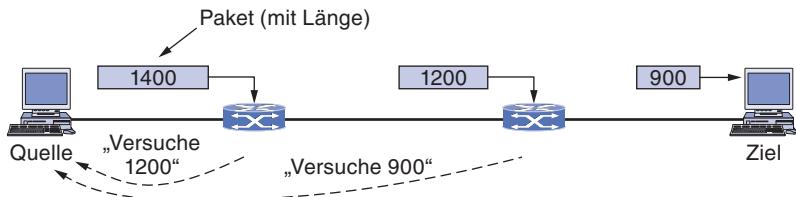


Abbildung 5.44: Path MTU Discovery.

Der Vorteil von Path MTU Discovery ist, dass die Quelle nun weiß, welche Paketlänge gesendet werden kann. Falls sich die Routen und die Path MTU ändern, dann werden neue Fehlerpakete ausgelöst und die Quelle passt sich an den neuen Pfad an. Solange die höheren Schichten jedoch die Path MTU nicht kennen und die richtige Datenmenge an IP übergeben, wird weiterhin Fragmentierung zwischen der Quelle und dem Ziel

benötigt. TCP und IP werden typischerweise zusammen implementiert (als „TCP/IP“), damit diese Art von Informationen weitergegeben werden kann. Obwohl dies für andere Protokolle nicht gilt, hat sich die Fragmentierung doch aus den Netzen zu den Hosts hin verschoben.

Der Nachteil von Path MTU Discovery ist die zusätzliche Anfangsverzögerung beim einfachen Senden eines Pakets. Es kann sein, dass mehr als eine Paketumlaufzeit benötigt werden, um den Pfad zu erforschen und die MTU ausfindig zu machen, bevor Daten zum Ziel geliefert werden. Dies wirft die Frage auf, ob es nicht bessere Entwürfe gibt. Die Antwort ist wahrscheinlich „Ja“. Stellen Sie sich zum Beispiel ein Konzept vor, bei dem jeder Router einfach Pakete, die seine MTU überschreiten, abschneidet. Damit wäre sichergestellt, dass das Ziel die MTU auf dem schnellsten Weg lernt (durch die zugestellte Datenmenge) und einen Teil der Daten erhält.

5.6 Vermittlungsschicht im Internet

Nun ist es an der Zeit, die Vermittlungsschicht im Internet genauer zu untersuchen. Doch bevor wir ins Detail gehen, lohnt sich ein Blick auf die Prinzipien, die in der Vergangenheit für den Entwurf verantwortlich waren und heutzutage seinen Erfolg ausmachen. Es scheint zu oft, dass diese heute in Vergessenheit geraten sind. Diese Prinzipien werden in RFC 1958, der sehr leserwert ist (und zur Pflichtlektüre aller Protokollentwickler gehören sollte – mit entsprechender Abschlussprüfung) aufgeführt und erörtert. Dieser RFC stützt sich stark auf Konzepte, die in Clark (1988) und Saltzer et al. (1984) vorgestellt werden. Das Folgende ist eine Übersicht der nach unserem Dafürhalten wichtigsten zehn Prinzipien (vom wichtigsten zum am wenigsten wichtigen):

- 1. Stellen Sie sicher, dass es funktioniert.** Schließen Sie das Design oder den Standard nicht ab, bevor nicht mehrere Prototypen erfolgreich miteinander kommuniziert haben. Zu oft erstellen die Entwickler zuerst einen 100 Seiten langen Standard, lassen diesen genehmigen und entdecken dann, dass so viele Fehler enthalten sind, dass er nicht funktioniert. Dann erstellen Sie die Version 1.1 des Standards. Dies ist nicht der richtige Weg.
- 2. Halten Sie den Entwurf einfach.** Wählen Sie in Zweifelsfällen die einfachste Lösung. William von Occam formulierte dieses Prinzip im 14. Jahrhundert (das Ökonomieprinzip von Occam). In modernen Begriffen formuliert: Vermeiden Sie zu viele Funktionen. Ist eine Funktion nicht absolut notwendig, lassen Sie sie weg, Insbesondere dann, wenn die gleiche Wirkung durch die Kombination anderer Funktionen erzielt werden kann.
- 3. Treffen Sie eine klare Auswahl.** Wenn verschiedene Möglichkeiten bestehen, das Gleiche zu erledigen, entscheiden Sie sich für eine. Wenn Sie die gleiche Sache auf zwei oder mehr Arten ausführen, sieht dies nach Problemen aus. Standards verfügen oftmals über mehrere Optionen, Modi oder Parameter, da verschiedene einflussreiche Parteien darauf bestehen, dass ihr Weg der beste ist. Entwickler sollten sich dieser Tendenz mit ganzer Kraft widersetzen. Sagen Sie einfach nein.

4. **Nutzen Sie die Modularität.** Dieses Prinzip führt direkt zum Konzept von Protokollstapeln, bei denen jede Schicht unabhängig von allen anderen ist. Wenn es erforderlich wird, dass ein Modul oder eine Schicht geändert wird, so sind auf diese Weise keine anderen betroffen.
5. **Erwarten Sie Heterogenität.** In jedem großen Netz treten verschiedene Typen von Hardware, Übertragungseinrichtungen und Anwendungen auf. Um diese handhaben zu können, muss der Netzentwurf einfach, allgemein und flexibel sein.
6. **Vermeiden Sie statische Optionen und Parameter.** Können Parameter nicht vermieden werden (wie z.B. die maximale Paketgröße), so ist es am besten, wenn der Sender und der Empfänger einen Wert aushandeln, anstatt feste Optionen vorzugeben.
7. **Streben Sie nach einem guten Entwurf, er muss nicht perfekt sein.** Oftmals haben Entwickler einen guten Entwurf, können aber einen merkwürdigen Sonderfall nicht behandeln. Entwickler sollten dann, anstatt sich den Entwurf zu verderben, den guten Entwurf nehmen und den Entwicklungsaufwand für eine Lösung des Sonderfalls den Leuten mit den ausgefallenen Anforderungen überlassen.
8. **Handhaben Sie das Senden streng, das Empfangen aber tolerant.** Anders ausgedrückt, senden Sie nur Pakete, die den Standards hundertprozentig entsprechen. Akzeptieren Sie auch ankommende Pakete, die die Standards nicht vollständig erfüllen, und versuchen Sie, diese zu verarbeiten.
9. **Berücksichtigen Sie die Skalierbarkeit.** Wenn das System Millionen von Hosts und Milliarden von Benutzern leistungsstark verwalten soll, sind zentrale Datenbanken absolut nicht möglich, und die Last muss gleichmäßig über die verfügbaren Ressourcen verteilt werden.
10. **Berücksichtigen Sie Leistung und Kosten.** Wenn ein Netz eine schlechte Leistung oder enorme Kosten aufweist, wird es niemand verwenden.

Wenden wir uns nun von den allgemeinen Prinzipien hin zu den Einzelheiten der Vermittlungsschicht im Internet. Auf der Vermittlungsschicht kann das Internet als Sammlung von Teilnetzen oder **autonomen Systemen (AS)** betrachtet werden, die miteinander verbunden sind. Es gibt keine echte Struktur, sondern mehrere größere Backbones. Diese Backbones werden aus Leitungen mit hoher Bandbreite und schnellen Routern gebildet. Die größten dieser Backbones, mit denen sich alle anderen verbinden, um den Rest des Internets zu erreichen, werden **Tier-1-Netze** genannt. An die Backbones sind ISPs angeschlossen, die einen Internetzugang für private und geschäftliche Anwendungen, für Datenzentren und Housing-Einrichtungen mit vielen Servern sowie für regionale Netze (mittlere Ebene) zur Verfügung stellen. Die Datenzentren verarbeiten einen Großteil des Inhalts, der über das Internet gesendet wird. Mit den regionalen Netzen sind weitere ISPs, die LANs vieler Universitäten und Unternehmen sowie andere Randnetze verbunden. Diese quasi hierarchische Organisation ist in ► Abbildung 5.45 dargestellt.

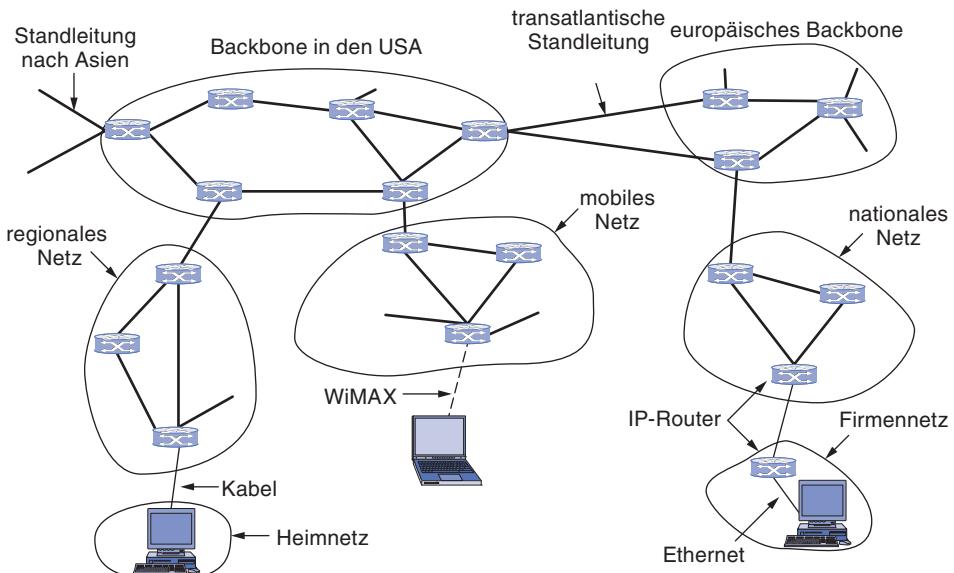


Abbildung 5.45: Das Internet besteht aus vielen miteinander verbundenen Netzen.

Der Klebstoff, der das alles zusammenhält, ist das Protokoll der Vermittlungsschicht – **IP (Internet Protocol)**. Im Gegensatz zu den meisten älteren Protokollen der Vermittlungsschicht wurde IP von Anfang für das Internetworking ausgelegt. Stellen Sie sich die Vermittlungsschicht auf die folgende Weise vor: die Aufgabe der Vermittlungsschicht ist die Bereitstellung einer Möglichkeit, nach bestem Bestreben (d.h. nicht garantiert) Pakete von der Quelle zum Ziel zu befördern, unabhängig davon, ob sich diese Rechner im gleichen Netz befinden oder ob andere Netze dazwischenliegen.

Die Kommunikation im Internet funktioniert so, dass die Transportschicht Datenströme entgegennimmt und so aufgeteilt, dass sie als IP-Pakete gesendet werden können. Theoretisch können Pakete jeweils bis zu 64 KB groß sein, doch in der Praxis liegt die Größe bei nicht mehr als 1 500 Byte (so passen Sie in einen Ethernet-Rahmen). IP-Router leiten jedes Paket durch das Internet entlang eines Pfads von einem Router zum nächsten weiter, bis das Ziel erreicht ist. Am Ziel reicht die Vermittlungsschicht die Daten an die Transportschicht weiter, welche diese dem empfangenen Prozess übergibt. Wenn alle Teile schließlich das Ziel erreicht haben, werden sie von der Vermittlungsschicht wieder zum ursprünglichen Datagramm zusammengesetzt. Dieses Datagramm wird dann an die Transportschicht übergeben.

Im Beispiel von Abbildung 5.45 muss ein Paket, das seinen Ursprung bei einem Host im Heimnetz hat, vier Netze und eine große Anzahl von IP-Routern passieren, bevor es überhaupt das Firmennetz erreicht, in dem sich der Ziel-Host befindet. Dies ist in der Praxis nicht unüblich und es gibt viele noch längere Pfade. Außerdem gibt es viel redundante Konnektivität im Internet, da sich Backbones und ISPs an mehreren Stellen miteinander verbinden. Dies bedeutet, dass es viele mögliche Pfade zwischen zwei Hosts gibt. Es ist die Aufgabe des IP-Routing-Protokolls zu entscheiden, welche Pfade benutzt werden sollen.

5.6.1 IPv4

Das Format der IP-Datagramme ist ein guter Ausgangspunkt für unsere Erforschung der Vermittlungsschicht des Internets. Ein IPv4-Datagramm besteht aus einem Header- und einem Body- oder Nutzdatenteil. Der Header besteht aus einem festen, 20 Byte großen Abschnitt und einem optionalen Abschnitt mit variabler Länge. Das Header-Format ist in ▶ Abbildung 5.46 dargestellt. Die Bits werden von links nach rechts und von oben nach unten übertragen, beginnend mit dem höchstwertigen Bit des *Version*-Feldes. (Dies ist eine Bytereihenfolge aus „Big-Endian“-Netzen. Auf Little-Endian-Rechnern, wie den Intel-x86-Computern, ist eine Softwarekonvertierung bei der Übertragung und beim Empfang erforderlich.) Rückblickend wäre Little-Endian die bessere Wahl gewesen, aber zu der Zeit, als IP entworfen wurde, wusste niemand, dass es einmal die Computerwelt beherrschen würde.

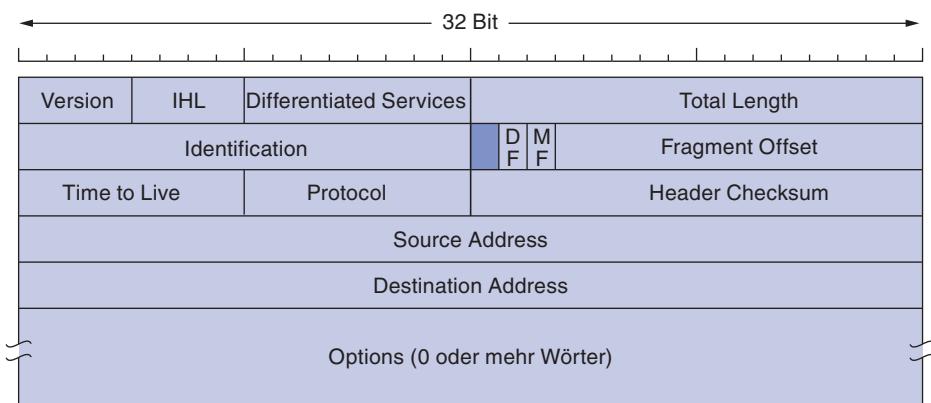


Abbildung 5.46: Der IPv4-Header.

Das *Version*-Feld verfolgt die Version des Protokolls, zu dem das Datagramm gehört. Version 4 dominiert heute das Internet, und damit haben wir unsere Betrachtung begonnen. Durch Einbindung der Version am Anfang jedes Datagramms ist es möglich geworden, dass sich ein Übergang zwischen Versionen über eine lange Zeitspanne hinzieht. In der Tat wurde IPv6, die nächste IP-Version, vor mehr als einem Jahrzehnt definiert und dennoch beginnt die Installation von IPv6 gerade erst. Wir werden später in diesem Abschnitt darauf zurückkommen. Die Verwendung von IPv6 wird letztendlich erzwungen werden, wenn jeder der fast 2^{31} Einwohner Chinas einen Desktop-PC, einen Laptop und ein IP-Telefon hat. Nebenbei zur fortlaufenden Nummerierung: IPv5 war ein experimentelles Echtzeitprotokoll, das nie große Verbreitung fand.

Da die Header-Länge nicht konstant ist, wird das Header-Feld *IHL* bereitgestellt. In diesem Feld wird die Header-Länge in 32-Bit-Wörtern angegeben. Der Mindestwert ist 5, was zutrifft, wenn keine Optionen vorhanden sind. Der Höchstwert dieses 4-Bit-Feldes ist 15, was den Header auf 60 Byte und damit das *Options*-Feld auf 40 Byte begrenzt. Für manche Optionen, z.B. eine, die den Weg eines Pakets aufzeichnet, ist das viel zu klein, sodass die Option unbrauchbar ist.

Das Feld *Differentiated Services* ist eines der wenigen Felder, die ihre Bedeutung im Laufe der Jahre (leicht) verändert haben. Ursprünglich hieß das Feld *Type of Service*. Es wurde und wird zur Unterscheidung diverser Dienstklassen verwendet. Hier sind verschiedene Kombinationen aus Zuverlässigkeit und Geschwindigkeit möglich. Für digitalisierte Sprache ist Schnelligkeit wichtiger als Genauigkeit. Für Dateiübertragung ist die fehlerfreie Übertragung wichtiger als die Schnelligkeit. Das *Type of Service*-Feld stellt 3 Bits für die Anzeige der Priorität und 3 Bits, um mitzuteilen, ob einem Host eher Verzögerung, Durchsatz oder Zuverlässigkeit wichtig ist. Da jedoch niemand wusste, was Router mit diesen Bits anfangen sollten, blieben sie viele Jahre ungenutzt. Als differenzierte Dienste eingeführt wurden, warf die IETF das Handtuch und belegte dieses Feld neu. Nun werden die ersten 6 Bits benutzt, um das Paket mit seiner Dienstklasse zu markieren; wir haben die Dienstklassen Expedited Forwarding und Assured Forwarding bereits früher in diesem Kapitel beschrieben. Die letzten 2 Bits werden zur Übertragung von ECN-Informationen verwendet, zum Beispiel, ob das Paket Überlastung erfahren hat; wir haben ECN als Teil der Überlastungsüberwachung weiter vorne in diesem Kapitel beschrieben.

Das Feld *Total Length* beinhaltet das komplette Datagramm – Header und Daten. Die Höchstlänge ist 65 535 Byte. Derzeit ist diese Obergrenze akzeptabel, aber in künftigen Netzen können größere Datagramme nötig sein.

Das Feld *Identification* ist erforderlich, damit der Ziel-Host feststellen kann, zu welchem Paket ein neu angekommenes Fragment gehört. Alle Fragmente eines Pakets enthalten den gleichen Wert in *Identification*.

Danach folgt ein nicht verwendetes Bit, was erstaunlich ist, da verfügbare Fläche im IP-Header extrem rar gesät ist. Bellovin (2003) hat – als Aprilscherz – vorgeschlagen, dieses Bit zu benutzen, um bösartigen Verkehr aufzudecken. Dies würde die Sicherheit sehr vereinfachen, da man wüsste, dass Pakete, bei denen das „Böse“-Bit gesetzt ist, von Angreifern gesendet wurden und somit einfach verworfen werden könnten. Leider ist Netzwerksicherheit doch nicht ganz so einfach.

Als Nächstes kommen zwei 1-Bit-Felder, die sich auf die Fragmentierung beziehen. *DF* steht für „Don't Fragment“ (nicht fragmentieren). Das ist ein Befehl an die Router, dass ein Paket nicht fragmentiert werden darf. Ursprünglich war es dazu gedacht, Hosts zu unterstützen, die nicht in der Lage sind, die einzelnen Stücke wieder zusammenzusetzen. Jetzt wird es innerhalb des Prozesses verwendet, mit dem die Path MTU herausgefunden wird – die Größe, die angibt, wie groß ein Paket sein darf, das ohne Fragmentierung einen Pfad entlang reisen kann. Durch Kennzeichnung des Datagramms mit dem *DF*-Bit weiß der Sender, dass das Paket entweder in einem Stück ankommt oder dass eine Fehlermeldung an den Sender zurückgeschickt wird.

MF bedeutet „More Fragments“ (mehr Fragmente). Dieses Bit ist bei allen Fragmenten außer dem letzten gesetzt. Es wird benötigt, um feststellen zu können, wann alle Fragmente eines Datagramms angekommen sind.

Fragment Offset gibt an, an welche Stelle im aktuellen Paket ein Fragment gehört. Alle Fragmente außer dem letzten müssen in einem Datagramm ein Vielfaches von 8 Byte

sein. Das ist die elementare Fragmenteinheit. Da 13 Bit verfügbar sind, sind maximal 8 192 Fragmente pro Datagramm möglich, wodurch eine maximale Paketlänge bis zur Grenze des Felds *Total Length* unterstützt wird. Die Felder *Identification*, *MF* und *Fragment Offset* werden zusammen verwendet, um Fragmentierung wie in Abschnitt 5.5.5 beschrieben zu implementieren.

Das Feld *TTL* (*Time to Live*) ist ein Zähler, mit dem die Lebensdauer von Paketen begrenzt werden kann. Ursprünglich sollte das Feld die Zeit in Sekunden erfassen. Zulässig ist eine maximale Lebensdauer von 255 Sekunden. Der Zähler muss bei jeder Teilstrecke um 1 reduziert werden und bei einer längeren Zwischenspeicherung in einem Router mehrmals. In der Praxis werden nur die Teilstrecken gezählt. Erreicht der Zähler null, wird das Paket verworfen und ein Warnpaket an den Quell-Host gesendet. Durch dieses Merkmal werden Pakete daran gehindert, ewig herumzuschwirren, was manchmal passiert, wenn die Routing-Tabellen beschädigt wurden.

Hat die Vermittlungsschicht ein komplettes Paket zusammengestellt, muss sie wissen, was sie damit anfangen soll. Durch das Feld *Protocol* kann sie erkennen, an welchen Transportprozess das Paket weiterzugeben ist. TCP ist eine Möglichkeit, UDP und andere sind weitere. Die Nummerierung der Protokolle ist im ganzen Internet einheitlich. Protokolle und andere zugewiesene Nummern wurden früher im RFC 1700 aufgeführt, sind aber heutzutage in einer Onlinedatenbank unter www.iana.org zu finden.

Da der Header wesentliche Informationen wie Adressen beinhaltet, legt er seine eigene Prüfsumme zum Schutz an, das Feld *Header Checksum*. Der Algorithmus addiert alle 16-Bit-Halbwörter des Headers bei ihrer Ankunft in der Einerkomplement-Arithmetik und nimmt dann das Einerkomplement des Ergebnisses. Zum Zweck dieses Algorithmus wird angenommen, dass *Header Checksum* initial null ist. Eine solche Prüfsumme ist nützlich zum Erkennen von Fehlern, während das Paket durch das Netz reist. Beachten Sie, dass es bei jeder Teilstrecke neu berechnet werden muss, weil sich mindestens ein Feld immer ändert (das *TTL*-Feld). Es sind aber verschiedene Tricks möglich, um die Berechnung zu beschleunigen.

Die Felder *Source Address* und *Destination Address* bezeichnen die IP-Adresse der Netzschnittstellen von Quelle und Ziel. Internetadressen werden im nächsten Abschnitt behandelt.

Das Feld *Options* wurde vorgesehen, um es späteren Versionen des Protokolls zu ermöglichen, Informationen aufzunehmen, die im ursprünglichen Entwurf nicht vorhanden sind. Damit können Experimentierfreudige neue Ideen ausprobieren und es wird gleichzeitig vermieden, dass Header-Bits für Informationen genutzt werden, die selten benötigt werden. Das Feld hat eine variable Länge. Jede Option beginnt mit einem 1-Byte-Code, der die Option identifiziert. Bei einigen Optionen folgt darauf ein 1-Byte-Feld mit der Optionslänge und dann eines oder mehrere Datenbytes. Das Feld *Options* wird auf ein Vielfaches von 4 Byte aufgefüllt. Ursprünglich waren fünf Optionen definiert, sie sind in ►Abbildung 5.47 aufgeführt.

| Option | Beschreibung |
|-----------------------|--|
| Security | Bezeichnet, wie geheim das Datagramm ist |
| Strict Source Routing | Gibt den vollständigen Pfad an |
| Loose Source Routing | Gibt eine Liste von Routern an, die nicht verfehlt werden dürfen |
| Record Route | Veranlasst jeden Router, seine IP-Adresse anzuhängen |
| Timestamp | Veranlasst jeden Router, seine IP-Adresse und einen Zeitstempel anzuhängen |

Abbildung 5.47: Einige ausgewählte IP-Optionen.

Die Option *Security* (Sicherheit) bestimmt, wie geheim die Informationen sind. Theoretisch kann ein Router in einer militärischen Anwendung diese Option benutzen, um festzulegen, dass die Übertragung der Pakete nicht durch bestimmte Länder fließen soll. In der Praxis wird es von allen Routern ignoriert; deshalb liegt seine einzige praktische Verwendung darin, dass Spione schneller bestimmte Leckerbissen finden können.

Die Option *Strict Source Routing* (exaktes Source-Routing) gibt den vollständigen Pfad von der Quelle zum Ziel als Folge von IP-Adressen an. Das Datagramm muss diese Route genau einhalten. Sie ist besonders für Systemmanager nützlich, die Notpakete aussenden müssen, wenn Routing-Tabellen beschädigt wurden, oder für bestimmte Zeitmessungen.

Die Option *Loose Source Routing* (lockeres Source-Routing) verlangt, dass das Paket die spezifizierten Router in der angegebenen Reihenfolge durchläuft, es kann aber auf dem Weg auch andere Router durchqueren. Normalerweise stellt diese Option nur ein paar Router zur Verfügung, um einen bestimmten Pfad zu erzwingen. Wenn beispielsweise durchgesetzt werden soll, dass ein Paket von London nach Sydney in westlicher und nicht in östlicher Richtung übertragen werden soll, können mit dieser Option Router in New York, Los Angeles und Honolulu angegeben werden. Diese Option ist am nützlichsten, wenn Datenströme aus politischen oder wirtschaftlichen Gründen bestimmte Länder durchlaufen oder aber meiden sollen.

Die Option *Record Route* (Wegaufzeichnung) weist jeden Router auf dem durchquerten Weg an, seine IP-Adresse an das Feld *Options* anzuhängen. Dadurch können Systemmanager Fehler in den Routing-Algorithmen verfolgen („Warum besuchen Pakete von Houston nach Dallas zuerst Tokio?“). Als das ARPANET erstmals eingerichtet wurde, überquerte nie ein Paket mehr als neun Router, sodass die 40-Byte-Option reichlich bemessen war. Heute ist das zu wenig.

Die Option *Time Stamp* (Zeitstempel) schließlich ist mit der Option *Record Route* vergleichbar, außer dass hier jeder Router zusätzlich zur 32-Bit-IP-Adresse auch einen 32-Bit-Zeitstempel erfasst. Auch diese Option wird vorwiegend für Messungen verwendet.

Heutzutage sind IP-Optionen in Ungnade gefallen. Viele Router ignorieren sie oder verarbeiten sie nicht effizient, schieben sie als unüblichen Fall beiseite. Das heißt, Optionen werden nur teilweise unterstützt und selten benutzt.

5.6.2 IP-Adressen

Ein definierendes Merkmal von IPv4 sind dessen 32-Bit-Adressen. Jeder Host und jeder Router im Internet hat eine IP-Adresse, die in den Feldern *Source Address* und *Destination Address* von IP-Paketen verwendet werden kann. Es ist wichtig, sich vor Augen zu halten, dass sich eine IP-Adresse eigentlich nicht auf einen Host, sondern auf eine Netzwerkschnittstelle bezieht. Wenn ein Host in zwei Netzen verfügbar ist, muss er zwei IP-Adressen haben. In der Praxis sind aber die meisten Hosts in einem Netz und haben daher eine IP-Adresse. Im Gegensatz dazu haben Router mehrere Schnittstellen und somit mehrere IP-Adressen.

Präfixe

Anders als Ethernet-Adressen sind IP-Adressen hierarchisch aufgebaut. Jede 32-Bit-Adresse setzt sich zusammen aus einem Netzwerkteil von variabler Länge in den oberen Bits und einem Hostteil in den unteren Bits. Der Netzwerkanteil hat für alle Hosts auf einem Netz, beispielsweise einem Ethernet-LAN, denselben Wert. Dies bedeutet, dass ein Netz einem zusammenhängenden Block von IP-Adressräumen entspricht. Dieser Block wird **Präfix** genannt.

IP-Adressen können in **Dezimalpunktschreibweise** (*dotted decimal notation*) geschrieben werden. In diesem Format wird jedes der 4 Bytes dezimal geschrieben, von 0 bis 255. Zum Beispiel wird die 32-Bit-Hexadezimalzahl 80D00297 als 128.208.2.151 geschrieben. Präfixe werden geschrieben, indem die niedrigste IP-Adresse im Block und die Blockgröße angegeben werden. Die Größe wird durch die Anzahl der Bits im Netzwerkteil bestimmt; die verbleibenden Bits im Hostteil können variieren. Dies bedeutet, dass die Größe eine Zweierpotenz sein muss. Konventionalgemäß wird die Größe nach der Präfix-IP-Adresse als Schrägstrich gefolgt von der Länge des Netzwerkteils in Bits geschrieben. Wenn in unserem Beispiel das Präfix 2⁸ Adressen enthält und somit 24 Bits für den Netzwerkteil bleiben, so wird dies als 128.208.0.0/24 geschrieben.

Da die Präfixlänge nicht allein aus der IP-Adresse abgeleitet werden kann, müssen Routing-Protokolle die Präfixe zu den Routern bringen. Manchmal werden Präfixe einfach durch ihre Länge beschrieben, wie in „/16“ (was „Slash 16“ gesprochen wird). Die Länge des Präfixes entspricht einer binären Maske von Einsen im Netzwerkteil. Wird das Präfix auf diese Art ausgeschrieben, dann wird es eine **Netzmaske** (*subnet mask*) genannt. Es kann über eine UND-Operation mit der IP-Adresse verknüpft werden, um nur den Netzwerkteil zu extrahieren. Für unser Beispiel ist die Netzmaske 255.255.255.0. ►Abbildung 5.48 zeigt ein Präfix und eine Netzmaske.

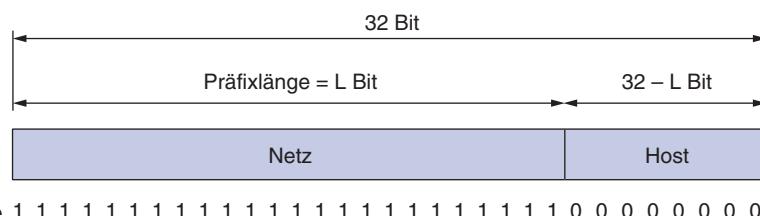


Abbildung 5.48: IP-Präfix und eine Netzmaske.

Hierarchisch aufgebaute Adressen haben entscheidende Vor- und Nachteile. Der Hauptvorteil von Präfixen ist, dass Router die Weiterleitung von Paketen lediglich aufgrund des Netzwerkteils der Adresse durchführen können, solange jedes Netz einen eindeutigen Addressblock besitzt. Der Hostteil interessiert die Router nicht, weil alle Pakete an Hosts in demselben Netzwerk in dieselbe Richtung gesendet werden. Der Hostteil wird erst benötigt, wenn die Pakete das Netz erreichen, für das sie vorgesehen sind, um sie zu dem richtigen Host weiterzuleiten. Dadurch sind die Routing-Tabellen viel kleiner. Bedenken Sie, dass die Anzahl von Hosts im Internet eine Milliarde erreicht. Das wäre eine sehr große Tabelle, die jeder Router vorhalten müsste. Indem jedoch eine Hierarchie verwendet wird, müssen Router nur die Wege für rund 300 000 Präfixe behalten.

Die Verwendung einer Hierarchie ermöglicht zwar das Skalieren von Internet-Routing, hat aber zwei Nachteile. Erstens ist die IP-Adresse eines Hosts davon abhängig, wo sich dieser innerhalb des Netzes befindet. Eine Ethernet-Adresse kann überall in der Welt verwendet werden, doch eine IP-Adresse gehört zu einem konkreten Netz und ein Router kann nur Pakete ausliefern, die für diese Adresse in diesem Netz bestimmt sind. Es werden Entwürfe wie mobiles IP benötigt, um Hosts zu unterstützen, die sich zwischen Netzen hin- und herbewegen, aber dieselbe IP-Adresse behalten wollen.

Der zweite Nachteil ist, dass die Hierarchie verschwenderisch mit Adressen umgeht, wenn keine sorgfältige Verwaltung betrieben wird. Falls Adressen an Netze in (zu) großen Blöcken zugewiesen werden, dann gibt es (viele) Adressen, die zwar zugeteilt sind, aber nicht verwendet werden. Diese Zuweisung wäre nicht weiter von Belang, wenn es reichlich Adressen zum Verteilen gäbe. Man weiß jedoch seit mehr als zwei Jahrzehnten, dass das ungeheure Wachstum des Internets rasant den freien Adressraum aufbraucht. IPv6 ist die Lösung für diesen Adressmangel, aber bis IPv6 weitgehend installiert ist, bleibt der Druck weiterhin groß, IP-Adressen so zuzuteilen, dass sie sehr effizient benutzt werden.

Subnetze

Die Netznummern werden von einer gemeinnützigen Organisation namens **ICANN** (*Internet Corporation for Assigned Names and Numbers*) verwaltet, um Konflikte zu vermeiden. ICANN wiederum hat verschiedene Teile des Adressraums an verschiedene regionale Behörden übergeben, die IP-Adressen an ISPs und andere Unternehmen vergeben. Auf diese Weise wird einem Unternehmen ein Block von IP-Adressen zugewiesen.

Dieser Vorgang ist jedoch erst der Anfang der Geschichte, denn die IP-Adresszuweisung geht weiter, wenn Unternehmen wachsen. Wir hatten oben gesagt, dass Routing mithilfe von Präfixen erfordert, dass alle Hosts in einem Netz dieselbe Netzwerknummer haben. Diese Eigenschaft kann Probleme verursachen, wenn Netze größer werden. Stellen Sie sich zum Beispiel vor, dass eine Universität unseren /16-Präfix aus dem obigen Beispiel benutzt, die Rechner der Informatikfakultät auf ihrem Ethernet mit Adressen zu versorgen. Ein Jahr später möchte die Elektrotechnikfakultät ins Internet gehen. Die Kunsthochschule zieht bald darauf nach. Welche IP-Adressen sollten diese Fakultäten verwenden? Sollen weitere Blöcke angefordert werden, dann muss der Universitätsrahmen

verlassen werden, und dies kann teuer oder umständlich sein. Zudem liefern die bereits zugewiesenen /16 schon genug Adressen für über 60 000 Hosts. Dieser Schritt könnte geplant werden, wenn bedeutendes Wachstum vorgesehen ist, aber bevor dies nicht passiert, wäre es verschwenderisch, derselben Universität weitere Blöcke von IP-Adressen zuzuweisen. Eine andere Organisationsmethode wird hier benötigt.

Die Lösung ist, den Adressblock in mehrere Teile aufzuspalten, die intern wie mehrere Netze benutzt werden, während sie der Außenwelt gegenüber noch wie ein einzelnes Netz auftreten. Dieser Vorgang wird **Subnetting** genannt und die Netze (wie Ethernet-LANs), die aus dem Aufteilen eines größeren Netzes resultieren, werden **Subnetze** genannt. Wie wir in *Kapitel 1* erwähnt haben, sollten Sie sich darüber bewusst sein, dass dieser neue Gebrauch der Begriffe mit der älteren Nutzung von „Subnetz“ in Konflikt steht, welcher die Menge aller Router und Kommunikationsleitungen in einem Netz bezeichnet.

► Abbildung 5.49 zeigt, wie Subnetze uns bei unserem Beispiel weiterhelfen können. Das einzelne /16 wurde in Teile gespalten. Diese Aufteilung muss keine gleich großen Teile ergeben, aber alle Teile müssen aufeinander abgestimmt sein, sodass alle Bits im unteren Hostteil verwendet werden können. In diesem Fall wird die Hälfte des Blocks (a /17) der Informatikfakultät zugewiesen, ein Viertel (a /18) der Elektrotechnikfakultät und ein Achtel (a /19) der Kunstfakultät. Das verbleibende Achtel wird nicht zugewiesen. Man kann sich die Aufteilung des Blocks auch verständlich machen, wenn man einen Blick auf die resultierenden Präfixe in Binärschreibweise wirft:

| | | | | |
|-----------------|----------|----------|--------------|----------|
| Informatik: | 10000000 | 11010000 | 1 xxxxxxxx | xxxxxxxx |
| Elektrotechnik: | 10000000 | 11010000 | 00 xxxxxx | xxxxxxxx |
| Kunst: | 10000000 | 11010000 | 011 xxxx | xxxxxxxx |

Hier gibt der senkrechte Balken (!) die Grenze zwischen der Subnetznummer und dem Hostteil an.

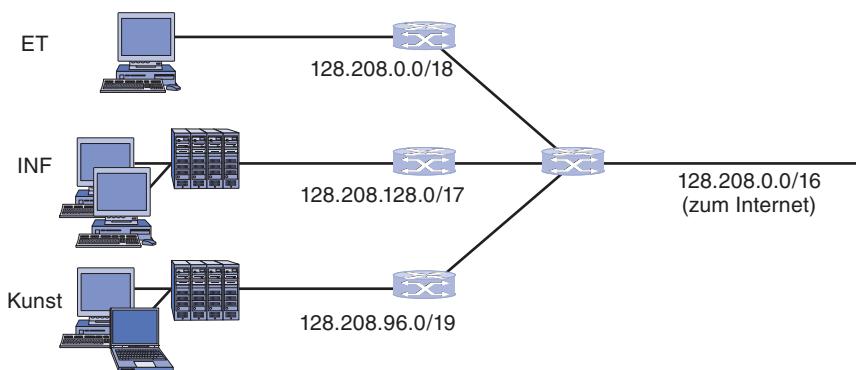


Abbildung 5.49: Aufspalten eines IP-Präfixes in unterschiedliche Netze mithilfe von Subnetting.

Kommt ein Paket am Hauptrouter an, wie weiß der Router dann, an welches Subnetz dieses Paket übergeben werden soll? Das ist der Punkt, an dem die Einzelheiten unserer Präfixe ins Spiel kommen. Eine Möglichkeit wäre, dass jeder Router eine Tabelle mit

65 536 Einträgen vorhält, die angibt, welche Ausgangsleitung für jeden Host im Campus benutzt werden soll. Aber dies würde den Hauptskalierungsvorteil unterminieren, den wir uns von der Verwendung einer Hierarchie versprechen. Stattdessen müssen die Router einfach die Netzmasken für die Netzwerke auf dem Campus kennen.

Wenn ein Paket ankommt, sieht sich der Router die Zieladresse des Pakets an und überprüft, zu welchem Subnetz das Paket gehört. Der Router kann dies tun, indem er die Zieladresse mit der Maske für jedes Subnetz UND-verknüpft und dann prüft, ob das Ergebnis mit dem entsprechenden Präfix übereinstimmt. Betrachten wir zum Beispiel ein Paket, das für IP-Adresse 128.208.2.151 bestimmt ist. Um festzustellen, ob es an die Informatikfakultät gerichtet ist, wenden wir eine UND-Verknüpfung mit 255.255.128.0 an und vergleichen die ersten 17 Bit (also 128.208.0.0) mit der Präfix-Adresse (128.208.128.0). Sie stimmen nicht überein. Nun prüfen wir die ersten 18 Bit für die Elektrotechnikfakultät. Hier erhalten wir 128.208.128.0, wenn wir eine UND-Verknüpfung mit der Netzmaske durchführen. Hier haben wir eine Übereinstimmung mit der Präfix-Adresse, daher wird das Paket zur Schnittstelle des Elektrotechniknetzes weitergeleitet.

Die Subnetzaufteilung kann falls nötig später geändert werden, indem alle Netzmasken an den Routern innerhalb der Universität aktualisiert werden. Außerhalb des Netzes sind die Subnetze nicht sichtbar, daher muss ICANN bei der Zuweisung von neuen Subnetzen nicht kontaktiert werden und es müssen keine externen Datenbanken geändert werden.

CIDR – Classless InterDomain Routing

Selbst wenn Blöcke von IP-Adressen reserviert werden, sodass die Adressen effizient genutzt werden, gibt es noch ein verbleibendes Problem: explodierende Routing-Tabellen.

Router von Organisationen am Rand des Netzes, wie eine Universität, müssen für jedes ihrer Subnetze einen Eintrag besitzen, der dem Router mitteilt, welche Leitung benutzt werden muss, um dieses Netz zu erreichen. Für Routen zu Zielen außerhalb der Organisation kann die einfache Default-Regel verwendet werden, die Pakete auf der Leitung zu dem ISP zu senden, der die Organisation mit dem Rest des Internets verbindet. Die anderen Zieladressen müssen sich alle irgendwo da draußen befinden.

Router von ISPs und Backbones aus dem Kernbereich des Internets haben diesen Luxus nicht. Sie müssen genau wissen, auf welchem Weg jedes einzelne Netz erreicht werden kann, und hierfür gibt es keine einfache Default-Regel. Diese Kernrouten befinden sich damit in der sogenannten **Default-freien Zone** des Internets. Niemand weiß mit Sicherheit, wie viele Netze darüber hinaus noch mit dem Internet verbunden sind, doch die Zahl dürfte groß sein, wahrscheinlich mindestens eine Million. Dies kann zu einer sehr großen Tabelle führen. In Computermaßstäben mag sich dies nicht groß anhören, aber bedenken Sie, dass ein Router für jedes Paket eine Suche in dieser Tabelle durchführen muss, und Router in großen ISPs müssen Millionen von Paketen pro Sekunde weiterleiten. Spezialisierte Hardware und schneller Speicher sind nötig, um Pakete mit diesen Raten zu verarbeiten, ein Allzweckrechner kann dies nicht leisten.

Außerdem setzen Routing-Algorithmen voraus, dass jeder Router mit anderen Routern Informationen über die Adressen austauscht, die der Router erreichen kann. Je größer die Tabellen, desto mehr Information muss kommuniziert und verarbeitet werden. Die Verarbeitung wächst mindestens linear mit der Tabellengröße. Größerer Kommunikationsaufwand erhöht die Wahrscheinlichkeit, dass ein Teil davon unterwegs verloren geht, zumindest zeitweilig, was möglicherweise zu einem instabilen Routing führt.

Die Probleme mit den Routing-Tabellen können gelöst werden, wenn man zu einer tieferen Hierarchie übergeht, wie dem Telefonnetz. Wenn jede IP-Adresse je ein Feld für Land, Staat, Stadt, Netz und Host enthält, kann dies funktionieren. In diesem Fall muss jeder Router nur wissen, wie er die einzelnen Länder, Bundesstaaten, Städte und Netze in seiner Stadt erreicht. Leider würde diese Lösung wesentlich mehr als 32 Bit für die IP-Adresse voraussetzen und außerdem Adressen ineffizient nutzen (und Liechtenstein hätte beispielsweise genauso viele Bits in seinen Adressen wie die USA).

Zum Glück gibt es etwas, was wir tun können, um die Größen der Routing-Tabellen zu reduzieren. Wir können dieselbe Erkenntnis wie beim Subnetting anwenden: Router an unterschiedlichen Orten können wissen, dass eine gegebene IP-Adresse zu Präfixen unterschiedlicher Größen gehört. Anstatt jedoch einen Adressblock in Subnetze aufzuspalten, kombinieren wir hier mehrere kleine Präfixe zu einem größeren Präfix. Dieser Vorgang wird als **Routen-Aggregation** bezeichnet. Das resultierende größere Präfix wird manchmal **Supernetz** genannt, um es von Subnetzen als Aufteilung von Adressblöcken abzugrenzen.

Bei der Aggregation kommen IP-Adressen in Präfixen unterschiedlicher Größe vor. Dieselben IP-Adressen, die der eine Router als Teil eines /22 (ein Block, der 2^{10} Adressen enthält) betrachtet, können von einem anderen Router als Teil eines größeren /20 angesehen werden (welches 2^{12} Adressen enthält). Es liegt an jedem einzelnen Router, ob er die entsprechende Präfixinformation hat. Dieser Entwurf wird zusammen mit Subnetting verwendet und **CIDR** (*Classless Inter-Domain Routing*) genannt, was wie das Getränk „Cider“ ausgesprochen wird. Die aktuelle Version wird in RFC 4632 spezifiziert (Fuller und Li, 2006). Der Name unterstreicht den Gegensatz zu Adressen, bei denen die Hierarchie mithilfe von Klassen codiert wird, welche wir in Kürze beschreiben werden.

Um CIDR verständlicher zu machen, sehen wir uns ein Beispiel an, in dem ein Block von 8 192 IP-Adressen ab der Adresse 194.24.0.0 verfügbar ist. Nehmen wir an, die Cambridge-Universität braucht 2 048 Adressen und erhält die Adressen 194.24.0.0 bis 194.24.7.255 mit der Maske 255.255.248.0. Dies ist ein /21-Präfix. Als Nächstes fordert die Oxford-Universität 4 096 Adressen an. Da ein Block von 4 096 Adressen auf einer Grenze von 4 096 Byte liegen muss, können Oxford keine Adressen gegeben werden, die mit 194.24.8.0 beginnen. Oxford erhält stattdessen 194.24.16.0 bis 194.24.31.255 mit der Netzmaske 255.255.240.0. Dann fordert die Universität von Edinburgh 1 024 Adressen an. Sie erhält die Adressen 194.24.8.0 bis 194.24.11.255 und die Maske 255.255.252.0. Diese Zuweisungen sind in ► Abbildung 5.50 zusammengefasst.

| Universität | Erste Adresse | Letzte Adresse | Wie viele | Präfix |
|-------------|---------------|----------------|-----------|----------------|
| Cambridge | 194.24.0.0 | 194.24.7.255 | 2048 | 194.24.0.0/21 |
| Edinburgh | 194.24.8.0 | 194.24.11.255 | 1024 | 194.24.8.0/22 |
| (verfügbar) | 194.24.12.0 | 194.24.15.255 | 1024 | 194.24.12.0/22 |
| Oxford | 194.24.16.0 | 194.24.31.255 | 4096 | 194.24.16.0/20 |

Abbildung 5.50: Eine Menge von IP-Adresszuweisungen.

Alle Router in der Default-freien Zone werden nun über die IP-Adressen in den drei Netzen informiert. Router, die sich nahe der Universitäten befinden, müssen eventuell für jedes der Präfixe auf einer anderen Ausgangsleitung senden, sie benötigen also einen Eintrag für jedes Prifix in ihren Routing-Tabellen. Ein Beispiel ist der Router in London in ► Abbildung 5.51.

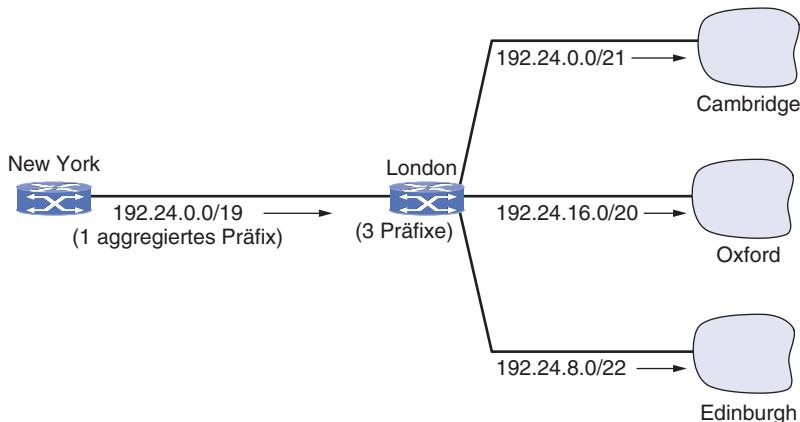


Abbildung 5.51: Aggregation von IP-Präfixen.

Betrachten wir nun diese drei Universitäten vom Standpunkt eines entfernten Routers in New York aus. Alle IP-Adressen in den drei Präfixen sollten von New York (oder der USA im Allgemeinen) nach London gesendet werden. Der Routing-Prozess in London bemerkt dies und kombiniert die drei Präfixe zu einem aggregierten Eintrag für das Prifix 194.24.0.0/19, das er dem New Yorker Router weitergibt. Dieses Prifix enthält 8 192 Adressen und deckt die drei Universitäten sowie die ansonsten unreservierten 1 024 Adressen ab. Durch Aggregation sind drei Präfixe auf einer reduziert worden. Somit haben sich die Präfixe, über die der New Yorker Router informiert werden muss, sowie die Einträge in den Routing-Tabellen des New Yorker Routers verringert.

Sobald Aggregation zugeschaltet wird, läuft der Prozess automatisch ab. Es kommt darauf an, welche Präfixe sich wo im Internet befinden, und hängt nicht von den Aktionen eines Administrators ab, der Adressen den Netzen zuordnet. Aggregation wird im Internet stark genutzt, die Größe von Router-Tabellen kann damit um rund 200 000 Präfixe reduziert werden.

Es gibt noch eine weitere Verwicklung: Präfixe können sich überlappen. Die Regel ist, dass Pakete in Richtung der speziellsten Route oder des **Longest-Prefix-Match** (*longest matching prefix*) gesendet werden, das die wenigsten IP-Adressen hat. Longest-Prefix-Match-Routing bietet einen sinnvollen Grad an Flexibilität, wie man an dem Verhalten des Routers in New York in ▶ Abbildung 5.52 sehen kann. Dieser Router verwendet noch ein einzelnes aggregiertes Präfix, um Verkehr an die drei Universitäten nach London zu senden. Der zuvor verfügbare Adressblock innerhalb dieses Präfixes wurde jedoch nun einem Netz in San Francisco zugeteilt. Der New Yorker Router hat nun die Möglichkeit, vier Präfixe zu behalten und die Pakete für drei davon nach London zu senden, während die Pakete für das vierte Präfix nach San Francisco gehen. Oder das Longest-Prefix-Match-Routing erledigt diese Weiterleitung mit den zwei gezeigten Präfixen. Ein Gesamtpräfix wird verwendet, um Verkehr für den gesamten Block nach London zu dirigieren. Ein spezelleres Präfix wird außerdem benutzt, um einen Teil des längeren Präfixes nach San Francisco zu leiten. Mit der Longest-Prefix-Match-Regel werden IP-Adressen innerhalb des Netzes in San Francisco auf die Ausgangsleitung nach San Francisco gesendet und alle anderen IP-Adressen im größeren Präfix werden nach London gesendet.



Abbildung 5.52: Longest-Prefix-Match-Routing am New Yorker Router.

Im Prinzip arbeitet CIDR wie folgt. Wenn ein Paket ankommt, wird die Routing-Tabelle durchsucht um festzustellen, ob das Ziel innerhalb des Präfixes liegt. Es ist möglich, dass mehrere Einträge mit unterschiedlichen Präfixlängen passen, in diesem Fall wird der Eintrag mit dem längsten Präfix benutzt. Falls es also eine Übereinstimmung für eine /20-Maske und für eine /24-Maske gibt, wird die /24-Maske verwendet, um die Ausgangsleitung für das Paket nachzuschlagen. Dieser Prozess wäre jedoch sehr mühsam, wenn die Tabelle tatsächlich Eintrag für Eintrag durchsucht würde. Stattdessen wurden komplexe Algorithmen entwickelt, um den Prozess der Adresszuordnung zu beschleunigen (Ruiz-Sanchez et al., 2001). Handelsübliche Router verwenden angepasste VLSI-Chips, bei denen diese Algorithmen in der Hardware eingebettet sind.

Klassenbasierte und spezielle Adressierung

Damit Sie besser schätzen können, warum CIDR so nützlich ist, wollen wir kurz den Entwurf ansehen, der ihm zeitlich voranging. Vor 1993 waren IP-Adressen über mehrere Jahrzehnte in fünf Kategorien aufgeteilt, die in ▶ Abbildung 5.53 aufgeführt sind. Diese Zuordnungen werden als **klassenbasierte IP-Adressierung** (*classful addressing*) bezeichnet.

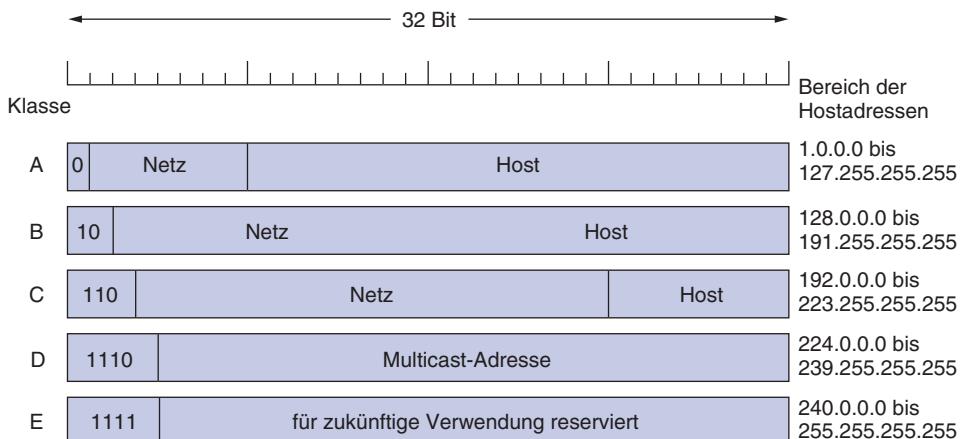


Abbildung 5.53: IP-Adressformate.

Die Formate der Klassen A, B und C ermöglichen die Konfiguration von bis zu 128 Netzen mit je 16 Millionen Hosts, 16 384 Netzen mit bis zu 65 536 Hosts, 2 Millionen Netzen (z.B. LANs) mit bis zu je 256 Hosts (obwohl einige davon für spezielle Zwecke reserviert sind). Ebenso wird Multicasting unterstützt (das Format der Klasse D), bei dem ein Datagramm an mehrere Hosts gerichtet werden kann. Adressen, die mit 1111 beginnen, sind für zukünftige Verwendungen reserviert. In Anbetracht des schwindenden IPv4-Adressraums könnten sie nun sinnvoll eingesetzt werden. Leider akzeptieren viele Hosts diese Adressen nicht als gültig, da sie so lange Zeit tabu waren, und jetzt ist es schwer, einem alten Host neue Tricks beizubringen.

Dies ist ein hierarchischer Entwurf, doch anders als bei CIDR sind die Größen der Adressblöcke fest. Es gibt über zwei Milliarden Adressen, aber durch die Organisation des Adressraums nach Klassen werden Millionen davon verschwendet. Der eigentliche Übeltäter ist die Klasse B. Für die meisten Unternehmen ist ein Klasse-A-Netz mit 16 Millionen Adressen zu groß und ein Klasse-C-Netz mit 256 Adressen zu klein. Meist ist ein Klasse-B-Netz mit 65 536 Adressen gerade richtig. In der Internetwelt nennt man diese Situation das **Problem der drei Bären** (wie im Märchen *Goldlöckchen und die drei Bären*; Southeby, 1848).

In Wirklichkeit jedoch ist eine Klasse-B-Adresse für die meisten Unternehmen viel zu groß. Untersuchungen haben gezeigt, dass mehr als die Hälfte der Klasse-B-Netze weniger als 50 Hosts haben. Für sie wäre ein Netz der Klasse C ausreichend. Zweifellos hatten alle Organisationen, die nach der Klasse B verlangten, das eventuell irgendwann zu knappe 8-Bit-Hostfeld vor Augen. Rückblickend wäre es besser gewesen, man hätte die C-Klasse mit einem 10-Bit-Feld anstelle von 8 Bit für die Hostnummer ausgestattet, was 1 022 Hosts pro Netz ermöglicht hätte. Wäre das der Fall gewesen, hätten sich die meisten Organisationen sicherlich für ein Netz der Klasse C entschieden (von denen es eine Million gibt, im Gegensatz zu nur 16 384 B-Netzen).

Es ist schwierig, die Internetentwickler dafür zu tadeln, dass sie nicht mehrere (und kleinere) Klasse-B-Adressen bereitgestellt haben. Als die Entscheidung getroffen

wurde, drei Klassen zu verwenden, war das Internet ein Forschungsnetz, das die großen Forschungsuniversitäten der USA miteinander verband (plus einige wenige Unternehmen und militärische Standorte, die Netzforschung betrieben). Niemand konnte damals ahnen, dass das Internet ein Kommunikationssystem für den Massenmarkt werden sollte, das eine Konkurrenz zum Telefonsystem darstellt. Zu dieser Zeit hat zweifellos irgendjemand gesagt: „Die Vereinigten Staaten verfügen über 2 000 Colleges und Universitäten. Selbst wenn man alle mit dem Internet verbindet und viele Universitäten aus anderen Ländern hinzukommen, werden wir 16 000 nie erreichen, da es weltweit nicht so viele Universitäten gibt. Darüber hinaus beschleunigt es die Paketverarbeitung, wenn die Hostnummer ein ganzes Vielfaches eines Bytes ist“ (was zum damaligen Zeitpunkt vollständig softwareseitig ausgeführt wurde). Vielleicht werden eines Tages die Menschen zurückblicken und diejenigen, die das Telefonnummernschema entwickelt haben, mit den Worten kritisieren: „Was für Idioten. Warum haben sie nicht die Planetennummern in die Telefonnummern aufgenommen?“ Doch zu jener Zeit schien es nicht nötig zu sein.

Zur Lösung dieser Probleme wurden Subnetze eingeführt, mit deren Hilfe man innerhalb einer Organisation flexibel Adressblöcke zuweisen kann. Später kam CIDR hinzu, um die Größe der globalen Routing-Tabelle zu reduzieren. Heute werden die Bits, die anzeigen, ob eine IP-Adresse zu einem Netz der Klasse A, B oder C gehört, nicht länger verwendet, obwohl Verweise auf diese Klassen in der Literatur immer noch üblich sind.

Um zu verstehen, warum durch den Verzicht auf Klassen die Weiterleitung verkompliziert wird, wollen wir uns ansehen, wie einfach es im alten klassenbasierten System war. Wenn ein Paket bei einem Router ankam, wurde eine Kopie der IP-Adresse um 28 Bit nach rechts verschoben, um eine 4-Bit-Klassennummer zu ergeben. Eine 16-fache Verzweigung sortierte dann die Pakete in A, B, C (und D und E), wobei acht der Fälle für Klasse A, vier für Klasse B und zwei für Klasse C reserviert waren. Für den Code jeder Klasse wurde dann durch Maskierung die 8-, 16-, oder 24-Bit-Netznummer erstellt und das Ergebnis nach rechts in einem 32-Bit-Wort ausgerichtet. Die Netznummer wurde dann in der Tabelle A, B oder C nachgeschlagen, in der Regel für A- und B-Netze durch Indizierung sowie für C-Netze durch Hashing. Wurde der Eintrag gefunden, konnte die Ausgangsleitung nachgeschlagen und das Paket weitergeleitet werden. Dies ist viel einfacher als die Longest-Prefix-Match-Operation, da hier ein einfacher Tabellennachschlag nicht mehr ausreicht, denn eine IP-Adresse könnte jede beliebige Präfixlänge haben.

Klasse-D-Adressen werden weiterhin im Internet für Multicasting verwendet. In der Tat wäre es zutreffender zu sagen, dass gerade erst damit begonnen wird, sie für Multicasting einzusetzen, da Internet-Multicasting in der Vergangenheit nicht sehr verbreitet war.

Es gibt außerdem mehrere andere Adressen, die spezielle Bedeutung haben, wie in ▶ Abbildung 5.54 gezeigt. Die IP-Adresse 0.0.0.0, die niedrigste Adresse, wird von Hosts benutzt, wenn sie hochgefährten werden. Es bedeutet „dieses Netz“ oder „dieser Host“. IP-Adressen mit 0 als Netznummer beziehen sich auf das aktuelle Netz. Anhand dieser Adressen können sich Rechner auf ihr Netz beziehen, ohne dessen Nummer zu kennen (sie müssen aber seine Netzmaske kennen, um zu wissen, wie

viele Nullen einzubeziehen sind). Die nur aus Einsen bestehende Adresse bzw. 255.255.255.255 – die höchste Adresse – wird benutzt, um alle Hosts im bezeichneten Netz zu meinen. Es ermöglicht Broadcasting im lokalen Netz, das normalerweise ein LAN ist. Die Adressen mit einer richtigen Netznummer und nur Einsen im Hostfeld erlauben Rechnern, Broadcast-Pakete an entfernte LANs irgendwo im Internet zu senden. Viele Netzwerkadministratoren deaktivieren diese Funktion jedoch, da sie meist ein Sicherheitsrisiko ist. Schließlich sind alle Adressen in der Form 127.xx.yy.zz für Loopback-Tests reserviert. Pakete, die an diese Adresse gesendet werden, werden nicht auf einer Leitung ausgegeben. Sie werden lokal verarbeitet und als Eingangs-pakete behandelt. Dadurch können Pakete an den Host gesendet werden, ohne dass der Sender die Nummer wissen muss, was nützlich für das Testen ist.

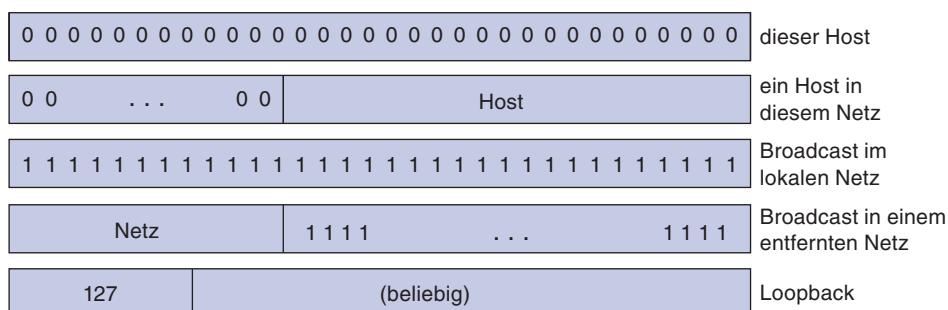


Abbildung 5.54: Spezielle IP-Adressen.

NAT – Network Address Translation

IP-Adressen sind rar. Ein ISP hat beispielsweise eine /16-Adresse, durch die er 65 534 benutzbare Hostnummern erhält. Wenn er mehr Kunden hat, ist dies ein Problem.

Diese Verknappung hat Techniken Zum sparsamen Umgang mit IP-Adressen hervorgebracht. Ein Ansatz ist, eine IP-Adresse dynamisch an einen Computer zuzuweisen, wenn dieser angeschaltet ist und das Netzwerk benutzt, und diese IP-Adresse wieder zurückzunehmen, wenn der Host inaktiv wird. Die IP-Adresse kann dann einem anderen Computer zugewiesen werden, der aktiv wird. Auf diese Weise kann man mit einer /16-Adresse bis zu 65 534 aktive Benutzer unterstützen.

Diese Strategie funktioniert in einigen Fällen gut, zum Beispiel für DFÜ-Netzverbindungen oder für mobile und andere Rechner, die zeitweise abwesend oder ausgeschaltet sind. Für Geschäftskunden ist es jedoch nicht geeignet. Von vielen geschäftlich genutzten PCs wird erwartet, dass sie ununterbrochen angeschaltet sind. Bei einigen dieser Rechner werden nachts Datensicherungen durchgeführt und andere sind Server, die eventuell von jetzt auf gleich auf Fernanfragen reagieren müssen. Diese Unternehmen haben eine Zugangsleitung, die immer eine Verbindung mit dem Rest des Internets herstellt.

In steigendem Maße trifft diese Situation auch auf Privatanwender zu, die ADSL oder Internet über Kabel abonnieren, da es keine Verbindungsgebühren gibt (nur eine monatliche Flatrate). Viele dieser Nutzer haben zwei oder mehr Rechner zu Hause, oftmals einen pro Familienmitglied, die alle immer online sein möchten. Die Lösung

ist, alle Rechner in einem Heimnetz über ein LAN zu verbinden und einen (drahtlosen) Router einzusetzen. Der Router verbindet dann mit dem ISP. Aus der Sicht des ISP ist die Familie nun wie ein kleines Unternehmen mit ein paar Rechnern. Willkommen bei der Schmidt AG. Mit den Techniken, die wir bisher gesehen haben, muss jeder Computer den ganzen Tag eine eigene IP-Adresse haben. Für einen ISP mit vielen Tausenden von Kunden, besonders Geschäftskunden und Familien, die wie kleine Unternehmen sind, kann die Nachfrage nach IP-Adressen den verfügbaren Block schnell überschreiten.

Das Problem, dass die IP-Adressen ausgehen, ist kein theoretisches, das irgendwann in der fernen Zukunft einmal eintreten könnte. Dies geschieht hier und jetzt. Die langfristige Lösung ist, dass das gesamte Internet auf IPv6 migriert, wo 128-Bit-Adressen zur Verfügung stehen. Der Übergang findet langsam statt, aber es wird noch Jahre dauern, bis der Prozess abgeschlossen ist. Um diese Zeit zu überbrücken, musste eine schnelle Lösung geschaffen werden. Diese schnelle Lösung, die heute weitverbreitet genutzt wird, kam in Gestalt von **NAT** (*Network Address Translation*, Netzwerkadressübersetzung), die in RFC 3022 beschrieben ist und die wir im Folgenden kurz zusammenfassen. Weitere Informationen hierzu finden Sie in Dutcher (2001).

Die Grundidee hinter NAT für den ISP ist, dass jeder Privathaushalt oder Unternehmen für den Internetdatenverkehr eine einzige IP-Adresse erhält (oder maximal eine kleine Anzahl). Innerhalb des Kundennetzes erhält jeder Rechner eine eindeutige IP-Adresse, die zum Weiterleiten des unternehmensinternen Datenverkehrs eingesetzt wird. Kurz bevor aber ein Paket das Kundennetz verlässt und zum ISP geht, findet eine Adressübersetzung von der eindeutigen IP-Adresse zu der gemeinsamen öffentlichen IP-Adresse statt. Diese Übersetzung verwendet drei Bereiche von IP-Adressen, die als privat deklariert wurden. Netzwerke können diese beliebig intern einsetzen. Sie müssen sich nur an die einzige Regel halten, die besagt, dass keine Pakete mit diesen Adressen im Internet auftauchen dürfen. Die drei reservierten Bereiche sind:

| | | | |
|-------------|---|--------------------|------------------|
| 10.0.0.0 | - | 10.255.255.255/8 | (16777216 Hosts) |
| 172.16.0.0 | - | 172.31.255.255/12 | (1048576 Hosts) |
| 192.168.0.0 | - | 192.168.255.255/16 | (65536 Hosts) |

Der erste Bereich umfasst 16 777 216 Adressen (außer für alle Nullen und alle Einsen, wie üblich) und ist die bevorzugte Wahl, selbst wenn das Netz nicht groß ist.

Die Funktionsweise von NAT ist in ► Abbildung 5.55 dargestellt. Beim Kunden hat jeder Rechner eine eindeutige Adresse in der Form 10.x.y.z. Bevor jedoch ein Paket die Kundengrenzen verlässt, durchquert es eine **NAT-Box**, die die interne IP-Quelladresse, in der Abbildung 10.0.0.1, in die echte IP-Adresse des Kunden, in unserem Beispiel 198.60.42.12, konvertiert. Die NAT-Box wird oft in einem Gerät mit einer Firewall kombiniert, die Sicherheit bietet, indem genau kontrolliert wird, welche Daten in das Kundennetz hineinkommen und welche herausgehen. Dieses Thema wird ausführlich in *Kapitel 8* behandelt. Die NAT-Box kann auch in einen Router oder ein ADSL-Modem integriert werden.

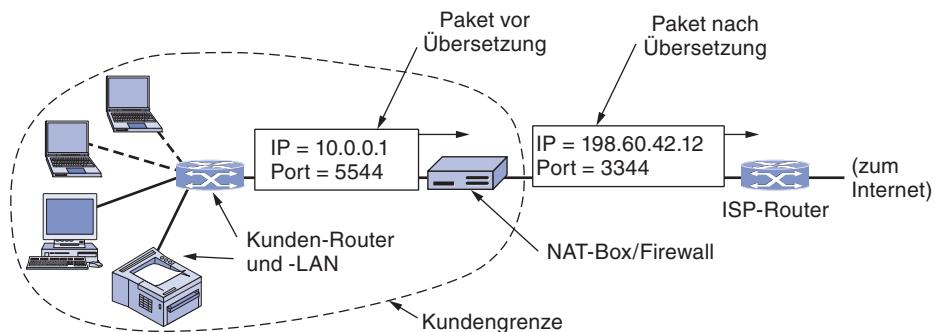


Abbildung 5.55: Positionierung und Betrieb einer NAT-Box.

Bis jetzt haben wir ein kleines, aber wichtiges Detail übersehen: Wenn die Antwort zurückkommt, (z.B. von einem Webserver), ist diese natürlich an 198.60.42.12 adressiert. Wie weiß die NAT-Box dann aber, durch welche interne Adresse diese ersetzt werden soll? Dies ist das Problem bei NAT. Wenn im IP-Header ein freies Feld vorhanden wäre, könnte man damit verfolgen, wer der echte Sender ist. Es ist aber nur ein Bit nicht genutzt. Prinzipiell könnte man eine neue Option erstellen, die die echte Quelladresse enthält, aber dies würde erfordern, dass der IP-Code auf allen Rechnern im gesamten Internet geändert werden müsste, um diese neue Option zu unterstützen. Dies ist keine vielversprechende Alternative für eine schnelle Lösung.

In Wirklichkeit passiert Folgendes: Die NAT-Entwickler haben beobachtet, dass die meisten IP-Pakete entweder TCP- oder UDP-Nutzdaten enthalten. Wir beschäftigen uns mit TCP und UDP in *Kapitel 6*. Hier werden wir sehen, dass beide Header haben, die einen Quellport und einen Zielport angeben. Im Folgenden werden wir nur TCP-Ports behandeln, aber das Gleiche gilt auch für UDP-Ports. Die Ports sind 16-Bit-Integerverte, die angeben, wo die TCP-Verbindung beginnt und wo sie endet. Diese Ports enthalten das Feld, mit dem NAT arbeitet.

Möchte ein Prozess eine TCP-Verbindung zu einem entfernten Prozess aufbauen, weist er sich einem unbenutzten TCP-Port auf dem eigenen Rechner zu. Dieser wird als **Quellport** bezeichnet. Er gibt dem TCP-Code an, wohin die eingehenden Pakete gesendet werden sollen, die zu dieser Verbindung gehören. Der Prozess gibt auch den **Zielport** an, damit bekannt ist, wer die Pakete am entfernten Standort erhält. Die Ports 0–1 023 sind für bestimmte Dienste reserviert. So wird beispielsweise Port 80 von Webservern verwendet, damit entfernte Clients sie ausfindig machen können. Jede ausgehende TCP-Nachricht enthält einen Quell- und einen Zielport. Diese Ports legen zusammen die Prozesse fest, die die Verbindung an beiden Endpunkten nutzen.

Eine Analogie soll das Konzept der Ports verdeutlichen. Stellen Sie sich ein Unternehmen mit einer einzigen Haupttelefonnummer vor. Wenn Anrufer die Hauptnummer wählen, kommen sie zur Vermittlung und werden gefragt, welche Nebenstelle sie möchten, worauf sie durchgestellt werden. Die Hauptnummer ist analog zur IP-Adresse des Kunden und die Nebenstellen an beiden Endpunkten entsprechen den Ports. Ports stellen im Endeffekt eine zusätzliche 16-Bit-Adressierung dar, die angibt, welcher Prozess das eingehende Paket erhält.

Mit dem Feld *Source Port* können wir auch unser Zuordnungsproblem lösen. Immer wenn ein ausgehendes Paket in der NAT-Box ankommt, wird die Quelladresse 10.x.y.z durch die echte IP-Adresse des Kunden ersetzt. Darüber hinaus wird das TCP-Feld *Source Port* durch einen Index in die Übersetzungstabelle mit 65 536 Einträgen der NAT-Box ersetzt. Dieser Tabelleneintrag enthält die ursprüngliche IP-Adresse und den ursprünglichen Quellport. Schließlich werden die Prüfsummen der IP- und TCP-Header erneut berechnet und in das Paket eingefügt. *Source Port* muss ersetzt werden, weil Verbindungen von den Rechnern 10.0.0.1 und 10.0.0.2 eventuell beide beispielsweise Port 5 000 nutzen, sodass der *Source Port* alleine nicht ausreicht, um den sendenden Prozess zu identifizieren.

Wenn ein Paket vom ISP in der NAT-Box eintrifft, wird *Source Port* im TCP-Header extrahiert und als Index in die Zuordnungstabelle der NAT-Box verwendet. Von dem ermittelten Eintrag werden die interne IP-Adresse und der ursprüngliche TCP-*Source Port* ermittelt und in das Paket eingefügt. Schließlich werden die Prüfsummen der IP- und TCP-Header erneut berechnet und in das Paket eingefügt. Das Paket wird dann an den Router des Kunden übergeben, um normal mit einer 10.x.y.z-Adresse ausgeliefert zu werden.

Obwohl dieses Schema das Problem in etwa löst, haben Netzwerk-Puristen in der IP-Gemeinde die Tendenz, es als eines der verabscheuungswürdigsten Dinge auf Erden zu betrachten. Hier sind kurz zusammengefasst einige der Einwände: Als Erstes verletzt NAT das architektonische Modell von IP, das besagt, dass jede IP-Adresse einen Rechner weltweit eindeutig bestimmt. Die gesamte Softwarestruktur des Internets beruht auf dieser Tatsache. Bei NAT können Tausende von Rechnern die Adresse 10.0.0.1 verwenden (und tun dies auch).

Zweitens durchbricht NAT das Modell der Ende-zu-Ende-Verbindung des Internets, welches besagt, dass jeder Host jederzeit an jedem anderen Host ein Paket senden kann. Da die Zuordnung in der NAT-Box durch die ausgehenden Pakete bestimmt wird, können keine ankommenden Pakete akzeptiert werden, ehe nicht Pakete hinausgeschickt wurden. In der Praxis bedeutet dies, dass ein Privatanwender mit NAT zwar TCP/IP-Verbindungen zu einem entfernten Webserver aufbauen kann, aber ein entfernter Benutzer kann keine Verbindung zu einem Spieleserver im Heimnetz herstellen. Spezielle Konfigurationen oder Techniken zur sogenannten **NAT-Durchdringung** (*NAT traversal*) werden benötigt, um mit diesen Situationen umzugehen.

Drittens ändert NAT das Internet von einem verbindungslosen Netz in eine eigentümliche Art von verbindungsorientiertem Netz. Das Problem dabei ist, dass die NAT-Box für jede durchgehende Verbindung die Informationen (d.h. die Zuweisungen) verwalten muss. Wenn das Netz den Verbindungszustand verwaltet, so ist dies eine Eigenschaft eines verbindungsorientierten Netzes und nicht eines verbindungslosen. Fällt die NAT-Box aus und die Zuordnungstabelle geht verloren, werden alle TCP-Verbindungen zerstört. In Abwesenheit von NAT kann ein Router ausfallen und neu starten ohne langfristige Auswirkungen auf TCP-Verbindungen. Der sendende Prozess erfährt nur innerhalb weniger Sekunden einen Zeitüberlauf und überträgt alle unbestätigten Pakete erneut. Bei NAT wird das Internet verletzlich wie ein leitungsvermitteltes Netz.

Viertens verletzt NAT die grundlegendste Regel der Protokollsichtbildung: Schicht k darf keine Annahmen darüber treffen, was Schicht $k+1$ in das Nutzdatenfeld abgelegt hat. Das Grundprinzip ist, die Schichten unabhängig zu halten. Wenn TCP in Zukunft einmal auf TCP-2 aktualisiert wird, so scheitert NAT an einem anderen Header-Layout (wie 32-Bit-Ports). Das Konzept der Protokollsichten dient dazu sicherzustellen, dass Änderungen in einer Schicht keine Änderungen in anderen Schichten nach sich ziehen. NAT zerstört diese Unabhängigkeit.

Fünftens müssen Prozesse im Internet nicht unbedingt TCP oder UDP verwenden. Wenn ein Benutzer auf Rechner A sich entscheidet, ein neues Transportprotokoll einzusetzen, um mit einem Benutzer auf Rechner B zu kommunizieren (beispielsweise bei einer Multimedia-Anwendung), führt eine dazwischenliegende NAT-Box zum Ausfall der Anwendung, weil die NAT-Box nicht in der Lage ist, den *Source Port* korrekt zu bestimmen.

Ein sechstes und verwandtes Problem ist, dass einige Anwendungen mehrere TCP/IP-Verbindungen oder UDP-Ports auf vorgeschriebene Weise benutzen. Zum Beispiel fügt **FTP (File Transfer Protocol)**, das Standardprotokoll für Dateiübertragungen, IP-Adressen in den Hauptteil des Pakets ein, sodass der Empfänger diese extrahieren und verwenden kann. Da NAT nichts über dieses Arrangement weiß, kann NAT die IP-Adressen auch nicht umschreiben oder diese anderweitig berücksichtigen. Diese Wissenslücke führt dazu, dass FTP und andere Anwendungen (wie das H.323-Protokoll für Telefon Gespräche über das Internet, welches wir in *Kapitel 7* untersuchen werden) bei einem NAT scheitern werden, falls nicht besondere Vorsichtsmaßnahmen getroffen werden. Häufig kann man in diesen Fällen NAT daraufhin ergänzen, aber wenn man den Code in der NAT-Box jedes Mal korrigieren muss, sobald eine neue Anwendung auftaucht, so ist dies auch nicht optimal.

Schließlich ist das *Source Port*-TCP-Feld 16 Bit lang, sodass einer IP-Adresse maximal 65 536 Rechner zugewiesen werden können. Tatsächlich ist die Anzahl etwas geringer, weil die ersten 4 096 Ports für spezielle Einsatzbereiche reserviert sind. Wenn aber mehrere IP-Adressen verfügbar sind, kann jede bis zu 61 440 Rechner verwalten.

Ein Überblick über diese und andere Probleme ist in RFC 2993 zu finden. Trotz dieser Probleme wird NAT in der Praxis häufig eingesetzt, speziell für Privatanwender und kleine Firmennetze, da es die einzige brauchbare Technik ist, um mit der Verknappung der IP-Adressen umzugehen. NAT wurde um Firewalls und Datenschutz erweitert, weil es unerwünscht ankommende Pakete standardmäßig blockiert. Aus diesem Grund ist es unwahrscheinlich, dass NAT vollständig verschwindet, wenn IPv6 weithin installiert wurde.

5.6.3 IP Version 6

IP wird seit Jahrzehnten vielfach eingesetzt. Es hat sich bewährt, wie das exponentielle Wachstum des Internets belegt. Leider wurde IP ein Opfer seiner eigenen Beliebtheit: ihm gehen langsam die Adressen aus. Selbst mit der sparsameren Verwendung von Adressen durch CIDR und NAT wird erwartet, dass die letzten IPv4-Adres-

sen vor dem Ende des Jahres 2012 von ICANN zugewiesen werden. Diese im Hintergrund lauernde Katastrophe wurde vor zwei Jahrzehnten entdeckt und hat vehement Diskussionen und Kontroversen in der Internetgemeinde ausgelöst.

In diesem Abschnitt werden das Problem und mehrere vorgeschlagene Lösungen beschrieben. Die einzige langfristige Lösung ist die Migration zu größeren Adressen. **IPv6 (IP Version 6)** ist ein Entwurf zur Ersetzung, der genau dies umsetzt. IPv6 benutzt 128-Bit-Adressen; eine Verknappung dieser Adressen in absehbarer Zukunft ist nicht sehr wahrscheinlich. Es hat sich jedoch erwiesen, dass IPv6 sehr schwierig zu installieren ist. IPv6 ist ein Vermittlungsschichtprotokoll, das sich von IPv4 unterscheidet und eigentlich nicht mit diesem zusammenarbeitet, trotz vieler Gemeinsamkeiten. Außerdem sind Firmen und Benutzer nicht wirklich sicher, warum sie IPv6 auf jeden Fall wollen sollten. Das Ergebnis ist, dass IPv6 sich nicht verbreitet und nur in einem kleinen Teil des Internets (schätzungsweise 1 %) benutzt wird, obwohl es seit 1998 ein Internetstandard ist. Die nächsten Jahre werden eine interessante Zeit sein, wenn die paar verbleibenden IPv4-Adressen zugewiesen werden. Wird man dann anfangen, seine IPv4-Adressen bei eBay zu versteigern? Wird ein Schwarzmarkt für Adressen aufblühen? Wer weiß.

Abgesehen von diesen Adressproblemen lauern noch andere Probleme im Hintergrund. In seinen Anfangsjahren wurde das Internet größtenteils von Universitäten, High-Tech-Industrien und der US-Regierung (insbesondere dem US-Verteidigungsministerium) genutzt. Mit der explosionsartigen Ausdehnung des Internets ab Mitte der 1990er Jahre wird es von ganz anderen Personen eingesetzt, die häufig andere Anforderungen haben. Zum einen möchten unzählige Leuten mit Smartphones das Internet nutzen, um mit ihrem Heimatstandort in Verbindung zu bleiben. Zum anderen wird es angesichts der bevorstehenden Verschmelzung der Computer-, Kommunikations- und Unterhaltungsbranchen nicht mehr lange dauern, bis jedes Telefon und jeder Fernseher der Welt ein Internetknoten ist. Das bedeutet, dass dann Milliarden von Geräten für Audio- und Video-on-Demand genutzt werden. Unter diesen Umständen ist ganz klar, dass sich IP weiterentwickeln und flexibler werden muss.

Angesichts dieser am Horizont auftauchenden Probleme begann die IETF 1990 die Arbeit an einer neuen Version von IP. Angestrebt wurde eine Version, die nie die Adressen ausgeben, die eine Vielzahl anderer Probleme löst und die auch flexibler und effizienter ist. Die wesentlichen Ziele waren:

1. Unterstützung von Milliarden von Hosts, auch bei ineffizienter Zuweisung des Adressraums
2. Reduzierung des Umfangs der Routing-Tabellen
3. Vereinfachung des Protokolls, damit Router Pakete schneller verarbeiten können
4. Höhere Sicherheit (Authentifizierung und Datenschutz)
5. Stärkere Betonung des Diensttypes, insbesondere für Echtzeitdaten
6. Unterstützung von Multicasting durch die Möglichkeit, dessen Umfang anzugeben

7. Möglichkeit für Hosts, ohne Adressänderung auf Reisen zu gehen
8. Möglichkeit für das Protokoll, sich künftig weiterzuentwickeln
9. Unterstützung der alten und neuen Protokolle in Koexistenz für Jahre

Der Entwurf von IPv6 stellt eine großartige Möglichkeit dar, all die Eigenschaften von IPv4 zu verbessern, die hinter den heutigen Wünschen zurückbleiben. Um ein Protokoll zu finden, das diese Anforderungen erfüllt, veröffentlichte die IETF eine Ausschreibung zur Einreichung von Angeboten und zur Diskussion in RFC 1550. Anfänglich sind 22 Vorschläge eingegangen. Im Dezember 1992 lagen sieben brauchbare Protokolle auf dem Tisch. Die Lösungen reichten von der Durchführung geringfügiger Änderungen in IP bis zur vollständigen Ablösung durch ein ganz anderes Protokoll.

Eine Lösung sah vor, TCP auf CLNP aufzusetzen, dem Vermittlungsschichtprotokoll, das für OSI entworfen wurde. Mit seinen 160-Bit-Adressen würde CLNP ausreichend Adressraum für ewige Zeiten geboten haben, da es jedem Wassermolekül in den Ozeanen genügend Adressen (grob 2^5) geben könnte, um ein kleines Netz einzurichten. Diese Wahl hätte außerdem zwei wichtige Protokolle der Vermittlungsschicht vereint. Viele Leute waren aber der Meinung, dass das einem Zugeständnis gleichgekommen wäre, dass etwas in der OSI-Welt tatsächlich richtig gemacht wurde – eine Aussage, die in Internetkreisen als „politisch inkorrekt“ erachtet wurde. CLNP wurde eng an IP angelehnt, sodass sich die zwei Protokolle kaum unterscheiden. Im Grunde unterscheidet sich das letztlich angenommene neue Protokoll mehr vom alten IP als CLNP. Ein weiterer Punkt gegen CLNP war seine schlechte Unterstützung von Diensttypen, die aber besonders im Hinblick auf die effiziente Übertragung von Multimediadaten als wichtig erachtet wurden.

Drei der besseren Vorschläge wurden in *IEEE Network* veröffentlicht (Deering, 1993; Francis, 1993; Katz und Ford, 1993). Nach heftigen Diskussionen und zahlreichen Revisionen wurde eine geänderte kombinierte Version der Vorschläge von Deering und Francis unter dem damaligen Namen **SIPP** (*Simple Internet Protocol Plus*) gewählt und schließlich als **IPv6** bezeichnet.

IPv6 erfüllt die Ziele von IETF ziemlich gut. Es hat die guten Eigenschaften von IP, wurde von den schlechten befreit und umfasst neue, die erforderlich waren. Im Allgemeinen ist IPv6 nicht mit IPv4 kompatibel, jedoch mit den anderen Internetprotokollen, darunter TCP, UDP, ICMP, IGMP, OSPF, BGP und DNS, nach geringen Modifikationen, die vorwiegend mit den längeren Adressen zu tun haben. Die wichtigsten Merkmale von IPv6 werden im Folgenden beschrieben. Ausführliche Informationen sind in RFC 2460 und RFC 2466 enthalten.

Als wichtigstes Merkmal hat IPv6 gegenüber IPv4 längere Adressen. Sie sind 128 Bits lang, was das Problem löst, weswegen IPv6 überhaupt entwickelt wurde: Bereitstellen einer wirklich unbegrenzten Menge an Internetadressen. Das Thema Adressen wird gleich noch einmal aufgegriffen.

Die zweite Verbesserung von IPv6 ist die Vereinfachung des Headers. Er enthält nur sieben Felder (gegenüber 13 in IPv4). Diese Änderung ermöglicht Routern, Pakete schneller zu verarbeiten. Damit verbessern sich der Durchsatz und die Übertragungsverzögerung. Dieses Thema wird später ebenfalls erneut aufgegriffen.

Die dritte wichtige Verbesserung ist eine bessere Unterstützung von Optionen. Diese Änderung war für den neuen Header erforderlich, weil vormals zwingende Felder nun optional sind (da sie nicht so häufig verwendet werden). Darüber hinaus unterscheidet sich auch die Art, wie Optionen dargestellt werden. Für Router ist es damit einfacher, Optionen zu überspringen, die sie nichts angehen. Diese Eigenschaft beschleunigt die Verarbeitungszeit von Paketen.

Ein vierter Bereich, in dem IPv6 gegenüber dem Vorgänger deutliche Vorteile bringt, ist Sicherheit. Die IETF hatte genug von Zeitungsgeschichten über fröhreife Zwöljfährige, die ihre PCs dazu einsetzen, in Banken und Militärbasen im ganzen Internet einzubrechen. Allseits war man zu der Überzeugung gekommen, dass etwas Durchschlagendes unternommen werden musste, um die Sicherheit zu verbessern. Die wichtigsten Merkmale des neuen IP sind somit Authentifizierung und Datenschutz. Diese wurden später jedoch auch in IPv4 übernommen, sodass die Unterschiede im Bereich Sicherheit nicht mehr so groß sind.

Schließlich wurde auf die Dienstgüte mehr Gewicht gelegt. In der Vergangenheit waren verschiedene halbherzige Anstrengungen zur Verbesserung der Dienstgüte unternommen worden, aber derzeit ist durch die Verbreitung von Multimedainhalten im Internet die Dringlichkeit größer.

Der Hauptheader von IPv6

► Abbildung 5.56 zeigt den IPv6-Header. Das Feld *Version* ist für IPv6 immer 6 (und 4 für IPv4). Während der Übergangszeit von IPv4 auf IPv6, die bereits mehr als ein Jahrzehnt andauert, können Router dieses Feld prüfen, um die Version festzustellen. Nebenbei bemerkt: Da dieser Test ein paar Anweisungen auf dem kritischen Pfad vergründet – vorausgesetzt der Sicherheitsschicht-Header kennzeichnet in der Regel das Netzprotokoll für Demultiplexing –, könnten einige Router die Überprüfung auslassen. Zum Beispiel hat das *Type*-Feld von Ethernet verschiedene Werte, um IPv4- oder IPv6-Nutzdaten anzudeuten. Die Diskussionen zwischen den beiden Lagern „mach es richtig“ und „mach es schnell“ werden zweifellos langwierig und vehement sein.

Das Feld *Differentiated Services* (ursprünglich *Traffic Class* genannt) dient zur Unterscheidung der Dienstklassen von Pakten mit unterschiedlichen Anforderungen bezüglich der Anforderungen für Echtzeitzustellung. Es wird mit der Architektur differenzierter Dienste für Dienstgüte auf die gleiche Weise verwendet wie das Feld desselben Namens im IPv4-Paket. Auch die beiden niederwertigen Bits werden verwendet, um explizite Überlastungsanzeigen zu signalisieren, ebenfalls auf die gleiche Art wie bei IPv4.

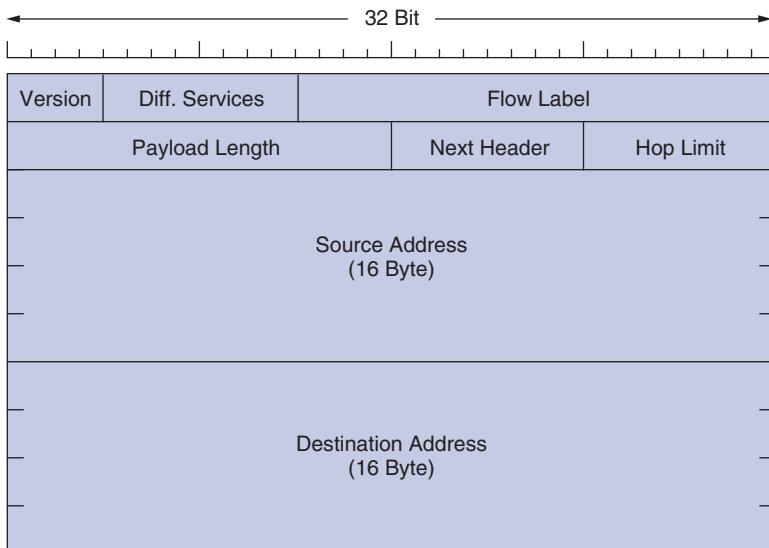


Abbildung 5.56: Der feste Header von IPv6 (erforderlich).

Das Feld *Flow Label* ermöglicht es einer Quelle und einem Ziel, Gruppen von Paketen zu markieren, die dieselben Anforderungen haben und auf dieselbe Art und Weise vom Netzwerk behandelt werden sollten, indem eine Pseudoverbindung aufgebaut wird. Ein Paketstrom von einem Prozess eines bestimmten Quell-Hosts zu einem Prozess auf einem bestimmten Ziel-Host kann straffe Übertragungszeitanforderungen stellen und damit reservierte Bandbreite benötigen. Der Datenfluss kann im Voraus aufgebaut werden und erhält einen Identifikator. Kommt ein Paket mit einem anderen Eintrag als null im Feld *Flow Label* an, können alle Router ihre internen Tabellen durchsehen, um festzustellen, welche spezielle Behandlung gefordert wird. Datenflüsse sind also ein Versuch, beides zu erlangen: die Flexibilität eines Datagrammnetzes und die Garantien eines VC-Netzes.

Jeder Fluss für Dienstgütezwecke wird durch die Quelladresse, die Zieladresse und die Flussnummer bezeichnet. Dieser Entwurf bedeutet, dass bis zu 2^{20} Datenflüsse zwischen einem Paar von IP-Adressen gleichzeitig aktiv sein können. Es bedeutet auch, dass selbst wenn zwei Datenflüsse von verschiedenen Hosts kommen, aber mit der gleichen Flussnummer durch den gleichen Router strömen, ist der Router dadurch in der Lage, sie anhand der Quell- und Zieladressen zu unterscheiden. Es wird erwartet, dass Flussnummern zufallsgesteuert ausgewählt werden, also nicht sequenziell ab 1 zugewiesen werden, deshalb wird von Routern der Aufbau von Hash-Tabellen erwartet.

Das Feld *Payload Length* gibt an, wie viele Bytes dem 40 Byte langen Header folgen (siehe Abbildung 5.56). Der Name wurde geändert (das Feld heißt in IPv4 *Total Length*), weil sich auch die Bedeutung leicht geändert hat: Die 40-Header-Bytes werden nicht mehr (wie vorher) als Teil der Länge gezählt. Diese Veränderung bedeutet, dass die Nutzlast nun 65 535 Byte anstelle von nur 65 515 Byte betragen kann.

Das Feld *Next Header* lässt die Katze aus dem Sack. Der Grund, warum der Header vereinfacht werden konnte, liegt darin, dass es zusätzliche (optionale) Erweiterungs-Header gibt. Dieses Feld gibt an, welcher der (derzeit) sechs Erweiterungs-Header diesem folgt, falls überhaupt. Ist das der letzte IP-Header, gibt das Feld *Next Header* an, an welchen Transportprotokoll-Handler (z.B. TCP, UDP) das Paket übergeben wird.

Das Feld *Hop Limit* wird verwendet, um Pakete am ewigen Leben zu hindern. Das entspricht dem Feld *Time to Live* in IPv4, d.h., es ist ein Feld, das nach jeder Teilstrecke um 1 reduziert wird. Theoretisch war das in IPv4 eine Zeit in Sekunden, aber kein Router hat das Feld so verwendet; deshalb wurde der Name geändert, um die eigentliche Nutzung widerzuspiegeln.

Die nächsten Felder sind *Source Address* und *Destination Address*. Deerings ursprünglicher Vorschlag, SIP, verwendet 8-Byte-Adressen. Im Laufe der Revisionsdurchläufe kam man aber überein, dass IPv6 dadurch innerhalb von wenigen Jahrzehnten keine Adressen mehr haben würde. 16-Byte-Adressen würden demgegenüber niemals ausgenügen. Andere waren der Meinung, dass 16 Byte überbemessen seien, während wieder andere 20-Byte-Adressen befürworteten, um mit dem OSI-Datagrammprotokoll kompatibel zu sein. Eine weitere Gruppe wollte Adressen mit variabler Länge. Nach heftigen Diskussionen und einigen Bemerkungen, die für ein akademisches Lehrbuch nicht druckreif sind, einigte man sich auf 16-Byte-Adressen mit fester Länge.

Eine neue Notation wurde zum Schreiben von 16-Byte-Adressen entwickelt. Sie werden in acht Gruppen von je vier hexadezimalen Ziffern geschrieben, die durch Doppelpunkte getrennt werden, wie beispielsweise

8000:0000:0000:0000:0123:4567:89AB:CDEF

Da viele Adressen viele Nullen aufweisen werden, wurden drei Optimierungen beschlossen. Erstens können führende Nullen in einer Gruppe weggelassen werden, sodass 0123 als 123 geschrieben werden kann. Zweitens können eine oder mehrere Gruppen mit 16 Nullbits durch zwei Doppelpunkte ersetzt werden. Die obige Adresse wäre dann z.B.

8000::123:4567:89AB:CDEF

Drittens können IPv4-Adressen als zwei Doppelpunkte und eine alte Dezimalzahldarstellung mit Punkt geschrieben werden, z.B.

::192.31.20.46

Die neue Form der Adressierung wird hier so ausführlich beschrieben, weil es viele 16-Byte-Adressen gibt. Ganz genau gibt es 2^{128} davon, was ungefähr 3×10^{38} entspricht. Wäre die ganze Erde, einschließlich der Meere, mit Computern bedeckt, würde IPv6 7×10^{23} IP-Adressen pro Quadratmeter bieten. Chemiestudenten werden feststellen, dass diese Zahl größer ist als die von Avogadro. Obwohl es nicht beabsichtigt war, jedem Molekül auf der Erdoberfläche eine eigene IP-Adresse zu geben, sind wir nicht mehr weit davon entfernt.

In der Praxis wird dieser Adressraum nicht effizient genutzt, genauso wie beim Telefonensystem (die Vorwahl für Manhattan – 212 – ist fast ausgeschöpft, aber 307 für Wyoming ist fast leer). In RFC 3194 haben Durand und Huitema berechnet, dass man unter Verwendung der Zuweisung der Telefonnummern als Richtlinie auch in einem extrem pessimistischen Szenario immer noch weit über 1 000 IP-Adressen pro Quadratmeter der Erdoberfläche (Land und Wasser) erhält. In jedem realistischen derartigen Szenario gibt es davon Trillionen pro Quadratmeter. Kurz: Es scheint sehr unwahrscheinlich, dass uns in vorhersehbarer Zukunft die IP-Adressen ausgehen.

Ein Vergleich des IPv4-Headers (Abbildung 5.46) mit dem IPv6-Header (Abbildung 5.56) zeigt auf, was in IPv6 weggelassen wurde. Das *IHL*-Feld ist verschwunden, weil der IPv6-Header eine feste Länge hat. Das *Protocol*-Feld wurde herausgenommen, weil das Feld *Next Header* angibt, was nach dem letzten IP-Header folgt (z.B. ein UDP- oder TCP-Segment).

Alle Felder in Bezug auf die Fragmentierung wurden weggelassen, weil die Fragmentierung in IPv6 anders gehandhabt wird. Erstens wird von allen IPv6-kompatiblen Hosts erwartet, dass sie die Größe der zu verwendenden Pakete dynamisch ermitteln. Dies wird mithilfe der Path-MTU-Discovery-Prozedur durchgeführt, die wir in Abschnitt 5.5.5 beschrieben haben. Kurz, sendet ein Host ein zu großes IPv6-Paket, dann wird es nicht fragmentiert, vielmehr sendet der Router, der es nicht weitergeben kann, eine Nachricht an den sendenden Host zurück. In dieser Nachricht wird der Host angewiesen, alle künftigen Pakete zu diesem Ziel aufzuteilen. Dadurch, dass der Host von vornherein Pakete in der richtigen Größe senden muss, wird mehr Effizienz erreicht, als wenn Pakete unterwegs fragmentiert werden. Auch wurde die minimale Paketgröße von 576 auf 1 280 erhöht, um 1 024 Byte Daten und viele Header zu unterstützen.

Das Feld *Checksum* ist verschwunden, weil die Berechnung eine nachteilige Wirkung auf die Leistung hat. Bei den neuen zuverlässigen Netzen und aufgrund der Tatsache, dass die Sicherungs- und Transportschicht normalerweise ihre eigenen Prüfsummen haben, hielt man es nicht für notwendig, zum Preis einer gewissen Leistungseinbuße noch ein Prüffeld bereitzustellen. Die Entfernung dieser Merkmale hat zu einem schlanken Protokoll für die Vermittlungsschicht geführt. Mit diesem Entwurf erfüllt IPv6 seine Ziele: Es ist ein schnelles, flexibles Protokoll mit reichlich Adressraum.

Erweiterungs-Header

In manchen Situationen sind aber einige der fehlenden Felder trotzdem notwendig, sodass IPv6 das Konzept eines (optionalen) **Erweiterungs-Headers** (*extension header*) beinhaltet. Diese Header können verwendet werden, um zusätzliche Informationen bereitzustellen, sie werden aber auf effiziente Weise codiert. Derzeit sind sechs Erweiterungs-Header definiert (►Abbildung 5.57). Alle sind optional; wenn sie benutzt werden, müssen sie direkt nach dem festen Header und vorzugsweise in der unten aufgeführten Reihenfolge erscheinen.

| Erweiterungs-Header | Beschreibung |
|--|--|
| Optionen für Teilstrecken (<i>Hop-by-Hop Options</i>) | Verschiedene Informationen für Router |
| Optionen für Ziele (<i>Destination Options</i>) | Zusätzliche Informationen für das betreffende Ziel |
| Routing (<i>Routing</i>) | Definition eines vollen oder teilweisen Wegs |
| Fragmentierung (<i>Fragmentation</i>) | Verwaltung von Datagrammfragmenten |
| Authentifizierung (<i>Authentification</i>) | Überprüfung der Identität des Senders |
| Verschlüsselte Sicherheitsdaten (<i>Encrypted Security Payload</i>) | Informationen über den verschlüsselten Inhalt |

Abbildung 5.57: IPv6-Erweiterungs-Header.

Einige Header haben ein festes Format, andere enthalten eine variable Anzahl von Feldern mit Optionen variabler Länge. Bei diesen Feldern wird jedes Element als Tupel (*Type, Length, Value*) codiert. *Type* ist ein 1-Byte-Feld, das die Option bezeichnet. Die Werte für *Type* wurden so gewählt, dass die ersten zwei Bits jenen Routern, die mit der Option nichts anfangen können, sagen, was sie tun sollen. Die Möglichkeiten lauten wie folgt: Option überspringen; Paket verwerfen; Paket verwerfen und ein ICMP-Paket zurücksenden; Paket verwerfen, aber keine ICMP-Pakete für Multicast-Adressen senden (um zu verhindern, dass ein fehlerhaftes Multicast-Paket Millionen von ICMP-Berichten erzeugt).

Length ist ebenfalls ein 1-Byte-Feld. Es gibt an, wie lang der Wert ist (0 bis 255 Byte). *Value* kann jede beliebige Information mit einer Länge von bis zu 255 Byte sein.

Der Header für Teilstreckenoptionen (Hop-by-Hop) wird für Informationen verwendet, die alle Router auf der jeweiligen Strecke prüfen müssen. Bisher wurde eine Option definiert: Unterstützung von Datagrammen, die über 64 KB hinausgehen. ►Abbildung 5.58 zeigt das Format dieses Headers. Wenn er verwendet wird, wird das Feld Nutzdatenlänge im festen Header auf 0 gesetzt.

| | | | |
|----------------------|---|-----|---|
| Next Header | 0 | 194 | 4 |
| Jumbo Payload Length | | | |

Abbildung 5.58: Der Erweiterungs-Teilstrecken-Header für große Datagramme (Jumbogramme).

Wie alle Erweiterungs-Header beginnt auch dieser mit einem Byte, das angibt, welcher Headertyp als Nächstes kommt. Diesem Byte folgt eines, das bezeichnet, wie lang der Teilstrecken-Header in Byte ist, einschließlich der ersten 8 Byte, die zwingend sind. Alle Erweiterungen beginnen auf diese Weise.

Die nächsten 2 Bytes geben an, dass diese Option die Datagrammgröße als 4-Byte-Zahl definiert (Code 194). Die letzten 4 Bytes geben die Größe des Datagramms an. Größen von weniger als 65 536 Byte sind nicht zulässig. Sie führen dazu, dass das Paket vom

ersten Router verworfen und eine ICMP-Fehlermeldung ausgegeben wird. Datagramme mit dieser Header-Erweiterung nennt man **Jumbogramme**. Die Verwendung von Jumbogrammen ist für Supercomputeranwendungen wichtig, die mehrere Gigabyte Daten effizient im Internet übertragen müssen.

Der Erweiterungs-Header für Zieloptionen kann für Felder verwendet werden, die nur vom Ziel-Host interpretiert werden müssen. In der ersten Version von IPv6 sind nur Nulloptionen zum Auffüllen dieses Headers auf ein Vielfaches von 8 Byte definiert. Der Header wird vorläufig nicht verwendet. Er wurde aber berücksichtigt, um für künftige Optionen neuer Routing- und Host-Software vorzusorgen.

Der Routing-Header führt bestimmte Router auf, die Pakete auf ihrem Weg zum Ziel durchlaufen müssen. Er ist dem freien Routing (*Loose Source Routing*) in IPv4 sehr ähnlich, da alle aufgeführten Adressen der Reihe nach besucht werden müssen, aber andere Router, die nicht aufgeführt sind, dürfen dazwischen besucht werden. ▶ Abbildung 5.59 zeigt das Format des Routing-Headers.

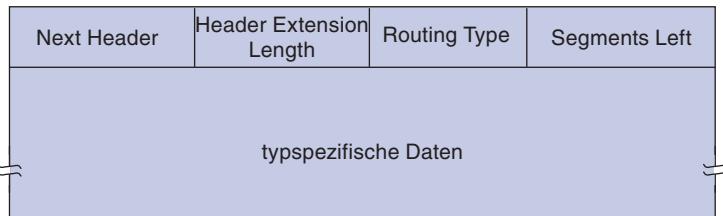


Abbildung 5.59: Der Erweiterungs-Header für Routing.

Die ersten 4 Bytes des Erweiterungs-Headers enthalten vier 1-Byte-Integerzahlen: Die Felder *Next Header* und *Header Extension Length* wurden oben beschrieben. Das Feld *Routing Type* gibt die vollständige Länge des Rahmens einschließlich des Headers an. Typ 0 gibt an, dass ein reserviertes 32-Bit-Wort auf das erste Wort folgt, wiederum gefolgt von einer Reihe von IPv6-Adressen. Andere Typen können bei Bedarf entwickelt werden. Schließlich verfolgt das Feld *Segments Left*, wie viele Adressen in der Liste noch nicht besucht wurden. Es wird jedes Mal um 1 reduziert, wenn eine Adresse besucht wurde. Wird es 0, steht das Paket alleine da und weiß nicht mehr, welchen Weg es wählen soll. An diesem Punkt ist es in der Regel so nahe am Ziel, dass der beste Weg offensichtlich ist.

Der Fragment-Header behandelt die Fragmentierung in einer ähnlichen Weise wie IPv4. Er enthält den Datagrammidentifier, die Fragmentnummer und ein Bit, das bezeichnet, ob weitere Fragmente folgen. Im Gegensatz zu IPv4 kann in IPv6 nur der Quell-Host ein Paket fragmentieren. Router entlang der Strecke sind dazu nicht berechtigt. Diese Änderung ist ein größerer philosophischer Bruch mit dem originalen IP, aber in Einklang mit aktueller Praxis für IPv4. Zusätzlich vereinfacht es aber die Arbeit der Router und beschleunigt das Routing insgesamt. Wird ein Router mit einem zu großen Paket konfrontiert, verwirft er es und sendet stattdessen ein ICMP-Fehlerpaket an die Quelle zurück. Dadurch wird der Quell-Host angewiesen, das Paket zu fragmentieren und es in kleineren Stücken erneut zu übertragen.

Der Authentifizierungs-Header bietet einen Mechanismus, durch den der Empfänger eines Pakets sicher sein kann, wer es gesendet hat. Die verschlüsselten Sicherheitsdaten ermöglichen die Verschlüsselung des Inhalts eines Pakets, sodass es nur der beabsichtigte Empfänger lesen kann. Diese Header wenden Verschlüsselungstechniken an, die wir in *Kapitel 8* beschreiben werden, um ihre Aufgaben zu bewältigen.

Kontroversen

Angesichts des offenen Entwicklungsprozesses und der vehement verteidigten Meinungen vieler Beteiligter muss es nicht überraschen, dass viele für IPv6 getroffene Entscheidungen höchst kontrovers sind. Einige davon werden in diesem Abschnitt kurz zusammengefasst. Die schmutzigen Details können sie in den RFCs nachlesen.

Die Streitigkeiten über die Adresslänge wurden bereits erwähnt. Das Ergebnis war ein Kompromiss: 16-Byte-Adressen in fester Länge.

Die Länge des Feldes *Hop Limit* ist ein weiteres strittiges Thema. Ein Lager behauptete, dass die Begrenzung der Höchstzahl von Teilstrecken auf 255 (implizit durch Verwendung eines 8-Bit-Feldes) ein großer Fehler sei. Schließlich seien heute Pfade mit 32 Teilstrecken üblich, und in zehn Jahren dürften es noch viel mehr sein. Diese Leute argumentierten, dass eine riesige Adressgröße weitsichtig, eine winzige Anzahl von Teilstrecken aber zu kurzsichtig sei. Ihrer Ansicht nach ist die Bereitstellung von zu wenig Bits an irgendeiner Stelle die größte Sünde, die ein Informatiker begehen kann.

Als Gegenargument wurde angeführt, dass jedes Feld vergrößert werden kann, dass das aber zu einem aufgeblähten Header führen würde. Außerdem sei die Funktion des Feldes *Hop Limit* die, Pakete von einer zu ausgedehnten Wanderung abzuhalten, und 65 535 Teilstrecken seien viel, viel zu lang. Schließlich werden mit zunehmendem Umfang des Internets mehr und mehr Fernverbindungen eingerichtet, sodass man über maximal ein halbes Dutzend Teilstrecken von einem Land zu einem anderen gelangen kann. Sind mehr als 125 Teilstrecken erforderlich, um von der Quelle oder dem Ziel zum jeweiligen internationalen Gateway zu gelangen, stimmt mit den nationalen Backbones etwas nicht. Die 8-Bit-Leute haben letztendlich den Kampf für sich entschieden.

Ein weiteres heißes Eisen war die maximale Paketgröße. Die Supercomputergemeinde wollte Pakete über 64 KB. Beginnt ein Supercomputer mit einer Übertragung, ist das keine Spielerei und er will sich nicht alle 64 KB unterbrechen lassen. Das Argument gegen große Pakete lautete, dass ein 1-Mbyte-Paket, das auf eine 1,5-Mbit/s-T1-Leitung trifft, die Leitung über fünf Sekunden lang in Beschlag nimmt, was für interaktive Benutzer, die gleichzeitig die Leitung nutzen, größere Übertragungszeiten bedeutet. Hier wurde folgender Kompromiss getroffen: Normale Pakete werden auf 64 KB begrenzt, aber der Erweiterungs-Header für Teilstrecken kann verwendet werden, um Jumbogramme zu senden.

Ein drittes heißes Eisen war die Entfernung der IPv4-Prüfsumme. Einige Leute verglichen das mit dem Ausbau der Bremsen in einem Auto. Das Auto werde zwar leichter und fahre auch schneller, habe aber spätestens dann Probleme, wenn Hindernisse auftauchen.

Als Gegenargument wurde angeführt, dass Prüfsummen etwas seien, das von Anwendungen übernommen werden könne, sofern für die Anwendungen die Datenintegrität wirklich erforderlich ist. Prüfsummen seien auf der Transportschicht ohnehin vorhanden, sodass keine weitere in IP erforderlich sei. Des Weiteren hätten die Erfahrungen gezeigt, dass die Berechnung der IP-Prüfsumme in IPv4 teuer zu stehen komme. In diesem Streit hat das Lager der Prüfsummengegner gewonnen. IPv6 hat also keine Prüfsumme.

Mobile Hosts waren ein weiterer Streitpunkt. Fliegt ein mobiler Host um die halbe Welt, kann er am Ziel mit der gleichen IPv6-Adresse weiterarbeiten oder soll er ein Schema mit Heimatagenten benutzen? Einige Leute wollten die explizite Unterstützung mobiler Hosts in IPv6 einbringen. Diese Bemühung ist misslungen, weil keine Einigung auf einen spezifischen Vorschlag erzielt werden konnte.

Die größte Schlacht ist beim Thema Sicherheit entbrannt. Jeder stimmte zu, dass Sicherheit entscheidend ist. Gestritten wurde nur über das Wo und das Wie. Zuerst zum Wo. Das Argument dafür, die Sicherheit auf die Vermittlungsschicht zu legen, ist, dass sie dann ein Standarddienst wird, den alle Anwendungen ohne Vorausplanung nutzen können. Die Gegenstimmen behaupteten, dass wirklich sichere Anwendungen im Allgemeinen nicht ohne eine Verschlüsselung von Ende zu Ende auskommen, wobei die Quellanwendung die Verschlüsselung übernimmt und die Zielanwendung alles wieder entschlüsselt. Bei jeder anderen Lösung ist der Benutzer wohl oder übel potenziell fehlerhaften Implementierungen der Vermittlungsschicht ausgeliefert, worüber er keinerlei Kontrolle hat. Die Antwort auf dieses Argument ist, dass diese Anwendungen schlichtweg darauf verzichten können, die IP-Sicherheitsfunktionen zu benutzen, und die Arbeit selbst übernehmen. Die Erwiderung darauf ist, dass Leute, die dem Netz nicht vertrauen, dass es dies richtig macht, nicht den Preis für langsame, überfrachtete IP-Implementierungen zahlen wollen, die diese Eigenschaft aufweisen, selbst wenn sie deaktiviert ist.

Ein weiterer Aspekt im Zusammenhang mit Sicherheit ist die Tatsache, dass viele (aber nicht alle) Länder strenge Ausfuhrbestimmungen in Bezug auf Verschlüsselung haben. Einige, darunter Frankreich und der Irak, schränken auch die Anwendung von Verschlüsselungsalgorithmen im Inland ein, damit die Leute vor der Regierung nicht zu viele Geheimnisse haben können. Das bedeutet, dass eine IP-Implementierung mit einem brauchbaren Verschlüsselungssystem von den USA (und vielen anderen Ländern) nicht an internationale Kunden exportiert werden darf. Das Unterhalten von zwei Softwareprodukten, einem für das Inland und einem für den Export, wird von den meisten Computeranbietern vehement abgelehnt.

Zu einem Punkt gab es keinerlei Meinungsverschiedenheiten: Keiner erwartet, dass das IPv4-Internet an einem Sonntagabend ausgeschaltet und am Montagmorgen als IPv6-Internet wieder angeschaltet wird. Vielmehr werden isolierte „Inseln“ auf IPv6 umgestellt, die anfangs über Tunnel kommunizieren, wie wir es in Abschnitt 5.5.3 gezeigt haben. Mit zunehmender Größe der IPv6-Inseln verschmelzen sie zu wieder größeren Inseln. Irgendwann werden alle Inseln verschmelzen, sodass das Internet dann vollständig auf IPv6 umgestellt ist.

So war zumindest der Plan. Die Installation hat sich jedoch als Achillesferse von IPv6 herausgestellt. IPv6 wird weiterhin wenig eingesetzt, obwohl es von allen großen Betriebssystemen vollständig unterstützt wird. Am häufigsten wird IPv6 in neuen Situationen installiert, wenn ein Netzbetreiber – zum Beispiel ein Mobilfunkbetreiber – eine große Anzahl von IP-Adressen benötigt. Es wurden viele Strategien definiert, die dabei helfen sollen, den Übergang zu erleichtern. Darunter waren Verfahren der automatischen Konfiguration der Tunnel, die IPv6 über das IPv4-Internet tragen, und Methoden, damit Hosts automatisch die Endpunkte der Tunnel finden. Dual-Stack-Hosts haben eine IPv4- und eine IPv6-Implementierung, sodass diese abhängig vom Ziel des Pakets auswählen können, welches Protokoll sie einsetzen. Diese Strategien werden die umfangreiche Installation optimieren, die unvermeidbar scheint, sobald die IPv4-Adressen aufgebraucht sind. Weitere Informationen über IPv6 finden Sie in Davies (2008).

5.6.4 Internetsteuerprotokolle

Zusätzlich zu IP, das für die Datenübertragung verwendet wird, werden im Internet verschiedene begleitende Steuerprotokolle auf der Vermittlungsschicht verwendet. Darunter sind ICMP, ARP und DHCP. In diesem Abschnitt wollen wir uns jedes dieser Protokolle der Reihe nach ansehen und die Versionen beschreiben, die IPv4 entsprechen, da dies die Protokolle sind, die üblicherweise benutzt werden. ICMP und DHCP haben ähnliche Versionen für IPv6; das Äquivalent von ARP für IPv6 heißt NDP (*Neighbor Discovery Protocol*).

ICMP

Der Betrieb des Internets wird von Routern überwacht. Passiert etwas Unerwartetes, während das Paket am Router verarbeitet wird, dann wird dem Sender das Ereignis vom **ICMP** (*Internet Control Message Protocol*) berichtet. ICMP wird auch für Tests im Internet verwendet. Etwa ein Dutzend Arten von ICMP-Nachrichten sind definiert. Jeder ICMP-Nachrichtentyp wird in einem IP-Paket gekapselt. Die wichtigsten sind in ▶ Abbildung 5.60 aufgeführt.

| Nachrichttyp | Beschreibung |
|--|---|
| Destination unreachable (Ziel nicht erreichbar) | Paket konnte nicht zugestellt werden |
| Time exceeded (Zeitablauf) | Das Feld <i>Time to Live</i> hat den Wert 0 |
| Parameter Problem (Parameterproblem) | Ungültiges Header-Feld |
| Source Quench (Quelldrosselung) | Choke-Paket |
| Redirect (Umleitung) | Bringt dem Router etwas über Geografie bei |
| Echo und Echo Reply (Echo und Echo-Antwort) | Prüft, ob ein Rechner noch am Leben ist |
| Timestamp Request/Reply (Zeitstempel-Anforderung/Antwort) | Wie Echo, aber mit Zeitstempel |
| Router Advertisement/Solicitation | Findet einen nahe gelegenen Router |

Abbildung 5.60: Die wichtigsten ICMP-Nachrichtentypen.

Die Nachricht DESTINATION UNREACHABLE wird verwendet, wenn der Router das Ziel nicht finden oder ein Paket mit dem DF-Bit nicht zugestellt werden kann, weil ein Netz mit kleiner Paketgröße im Weg steht.

Die Nachricht TIME EXCEEDED wird gesendet, wenn ein Paket weggeworfen wird, weil sein *TTL*-Zähler (*Time to Live*) null erreicht hat. Dieses Ereignis ist ein Anzeichen dafür, dass Pakete in Schleifen kreisen oder dass die Zählerwerte zu niedrig eingestellt wurden.

Diese Fehlernachricht wird geschickt im **Traceroute**-Hilfsprogramm eingesetzt, das 1987 von van Jacobson entwickelt wurde. Traceroute findet die Router entlang des Pfads vom Host zu einer Ziel-IP-Adresse, und zwar ohne irgendeine Art bevorzugter Netzunterstützung. Die Methode besteht darin, einfach eine Folge von Paketen zu einem Ziel zu senden, zuerst mit einem TTL von 1, dann einem TTL von 2, 3 und so weiter. Die Zähler in den Paketen werden an aufeinanderfolgenden Routern entlang des Pfads null erreichen. Jeder dieser Router wird gehorsam eine TIME EXCEEDED-Nachricht zurück an den Host senden. Anhand dieser Nachrichten kann der Host die IP-Adresse der Router entlang des Pfads feststellen und gleichzeitig statistische und zeitliche Informationen über Teile des Pfads erhalten. Dafür war die TIME EXCEEDED-Nachricht zwar eigentlich nicht gedacht, aber Traceroute ist vielleicht das nützlichste Debugging-Werkzeug aller Zeiten.

Die Nachricht PARAMETER PROBLEM zeigt an, dass in einem Header-Feld ein unzulässiger Wert festgestellt wurde. Dieses Problem deutet auf einen Fehler in der IP-Software des sendenden Hosts oder möglicherweise in der Software eines durchquerten Routers hin.

Die Nachricht SOURCE QUENCH wurde vor Langem verwendet, um Hosts zu drosseln, die zu viele Pakete schickten. Wenn ein Host diese Nachricht erhielt, wurde von ihm erwartet, sofort seine Aktivitäten herunterzuschauben. Sie wird heute kaum mehr verwendet, weil diese Pakete im Fall von Überlastungen nur noch mehr Öl ins Feuer gießen und es unklar ist, wie auf sie zu reagieren ist. Zur Überlastungsüberwachung werden Maßnahmen heute im Internet vorwiegend auf der Transportschicht ergriffen, indem Paketverluste als Überlastungssignal verwendet werden. Wir werden uns dies in *Kapitel 6* genauer ansehen.

Die Nachricht REDIRECT wird verwendet, wenn ein Router festgestellt hat, dass ein Paket falsch weitergeleitet wurde. Der Router verwendet die Nachricht, um den sendenden Host auf eine bessere Route aufmerksam zu machen.

Die Nachrichten ECHO REQUEST und ECHO REPLY werden von Hosts gesendet, um festzustellen, ob ein bestimmtes Ziel erreichbar und momentan noch am Leben ist. Bei Erhalt der ECHO-Nachricht wird vom Ziel erwartet, dass es eine Nachricht ECHO REPLY schickt. Diese Nachrichten werden im **ping**-Hilfsprogramm verwendet, das prüft, ob ein Host aktiv und im Internet ist.

Die Nachrichten TIMESTAMP REQUEST und TIMESTAMP REPLY sind ähnlich, außer dass die Ankunftszeit der Nachricht und die Sendezeit der Antwort in der Antwort erfasst werden. Diese Einrichtung kann zum Messen der Netzeistung verwendet werden.

Die Nachrichten ROUTER ADVERTISEMENT und ROUTER SOLICITATION werden benutzt, damit Hosts Router in der Nähe finden können. Ein Host muss die IP-Adresse von mindestens einem Router kennen, um Pakete aus dem lokalen Netz heraus zu senden.

Darüber hinaus wurden auch noch weitere Nachrichten definiert. Die Onlineliste befindet sich nun unter www.iana.org/assignments/icmp-parameters.

ARP

Obwohl jeder Rechner im Internet eine oder mehrere IP-Adressen hat, reichen diese Adressen zum Versenden von Paketen nicht aus. Netzwerkkarten der Sicherungsschicht wie Ethernet-Karten verstehen die Internetadressen nicht. Im Fall von Ethernet wird jede Netzwerkkarte werkseitig mit einer eindeutigen Ethernet-Adresse von 48 Bit ausgeliefert. Die Hersteller von Ethernet-Schnittstellenkarten fordern von der IEEE einen Ethernet-Addressblock an, um sicherzustellen, dass jede Netzwerkkarte eine eindeutige Adresse hat (damit nicht zufällig zwei Netzwerkkarten mit derselben Adresse im gleichen LAN auftauchen). Die Netzwerkkarten senden und empfangen Rahmen auf der Grundlage von 48 Bit langen Ethernet-Adressen. Sie wissen absolut nichts über 32-Bit-IP-Adressen.

Nun stellt sich die Frage, wie IP-Adressen in Adressen der Sicherungsschicht, z.B. Ethernet, konvertiert werden können. Um die Frage zu beantworten, betrachten wir das Beispiel von ►Abbildung 5.61: eine kleine Universität mit zwei /24-Netzen. Ein Netz (INF) ist ein Switched Ethernet in der Fakultät für Informatik. Es hat das Präfix 192.32.65.0/24. Das andere LAN (ET), ebenfalls ein Switched Ethernet, gehört zur Fakultät für Elektrotechnik und hat das Präfix 192.32.63.0/24. Die beiden LANs sind über einen IP-Router verbunden. Jeder Rechner in einem Ethernet und jede Schnittstelle am Router hat eine eindeutige Ethernet-Adresse von E1 bis E6 und eine eindeutige IP-Adresse im INF- oder ET-Netz.

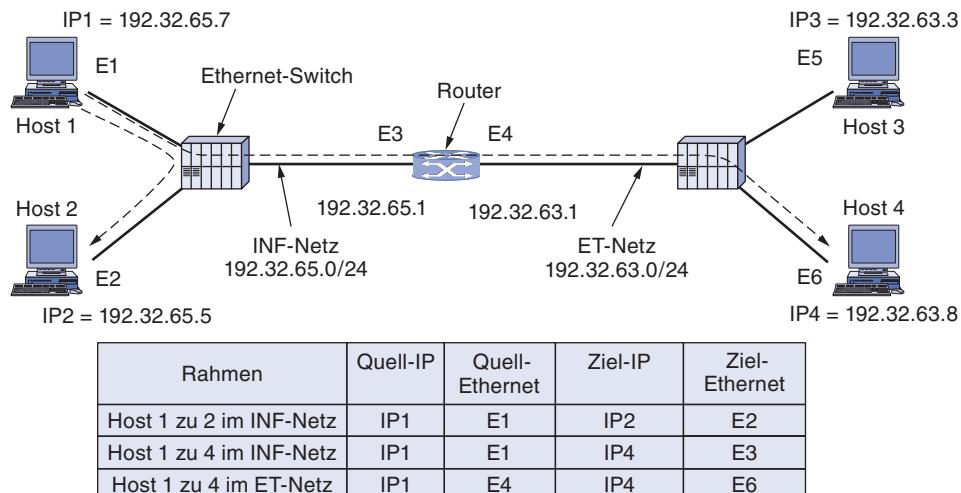


Abbildung 5.61: Zwei miteinander verbundene Ethernets, verbunden durch einen Router.

Wir beginnen mit der Untersuchung, wie ein Benutzer an Host 1 ein Paket an einen Benutzer an Host 2 im INF-Netz sendet. Nehmen wir an, der Sender kennt den Namen eines beabsichtigten Empfängers, eventuell etwas wie *eagle.cs.uni.edu*. Im ersten Schritt wird die IP-Adresse für Host 2 herausgefunden. Diese Suche wird vom DNS durchgeführt, das in *Kapitel 7* beschrieben wird. Vorläufig gehen wir einfach davon aus, dass DNS die IP-Adresse für Host 2 ausgibt (192.32.65.5).

Die Software der oberen Schicht auf Host 1 stellt jetzt ein Paket mit 192.32.65.5 im Feld *Destination Address* zusammen und gibt es zur Übertragung an die IP-Software ab. Die IP-Software kann die Adresse ansehen und erkennen, dass das Ziel im INF-Netz (also in ihrem eigenen Netz) liegt. Nun muss noch irgendwie die Ethernet-Adresse des Ziels herausgefunden werden, um den Rahmen zu senden. Das ist unter anderem über eine Konfigurationsdatei möglich, die irgendwo im System IP-Adressen in Ethernet-Adressen zuordnet. Diese Lösung ist sicherlich möglich, aber für Unternehmen oder Institute mit Tausenden von Rechnern eine mühsame und sehr fehleranfällige Angelegenheit.

Als bessere Lösung kann Host 1 ein Broadcast-Paket im Ethernet ausgeben und fragen, wem die IP-Adresse 192.32.65.5 gehört. Das Paket kommt auf jedem Rechner im INF-Ethernet an und jeder Rechner prüft daraufhin seine IP-Adresse. Host 2 wird als Einziger antworten, weil das seine IP-Adresse (*E2*) ist. Auf diese Weise erfährt Host 1, dass die IP-Adresse 192.32.65.5 zu dem Host mit der Ethernet-Adresse *E2* gehört. Das Protokoll, das für diese Frage und die Antwort verwendet wird, heißt **ARP** (*Address Resolution Protocol*, Adressauflösungsprotokoll). Fast jeder Rechner im Internet führt dieses Protokoll aus. ARP ist in RFC 826 definiert.

ARP hat im Vergleich zur Verwendung von Konfigurationsdateien den Vorteil, dass es einfacher ist. Der Systemmanager hat nicht viel Arbeit damit, abgesehen davon, dass er jedem Rechner eine IP-Adresse zuweisen und eine Entscheidung über die Netzmasken treffen muss. Der Rest wird von ARP übernommen.

An diesem Punkt stellt die IP-Software auf Host 1 einen Ethernet-Rahmen zusammen, der an *E2* adressiert ist, stellt das IP-Paket (adressiert an 192.32.65.5) in das Nutzdatenfeld und schickt es auf das Ethernet. Die IP- und Ethernet-Adressen dieses Pakets sind in Abbildung 5.61 angegeben. Die Ethernet-Schnittstellenkarte von Host 2 empfängt diesen Rahmen und löst einen Interrupt aus. Der Ethernet-Treiber zieht das IP-Paket aus den Nutzdaten heraus und leitet es an die IP-Software weiter, die es verarbeitet.

Um ARP effizienter auszulegen, sind verschiedene Optimierungen möglich. Ein Rechner, auf dem ARP läuft, speichert das Ergebnis für den Fall, dass er den gleichen Rechner bald wieder kontaktieren muss. Beim nächsten Mal findet er die konvertierte Adresse im Cache, sodass kein weiteres Broadcast-Paket erforderlich ist. In vielen Fällen muss Host 2 eine Antwort zurücksenden, sodass auch er ARP ausführen muss, um die Ethernet-Adresse des Senders festzustellen. Dieses ARP-Broadcasting kann vermieden werden, wenn Host 1 seine IP-zu-Ethernet-Zuordnung in das ARP-Paket einfügt. Kommt das ARP-Broadcasting bei Host 2 an, so wird das Paar (192.32.65.7, *E1*) in den ARP-Cache von Host 2 gestellt. Alle Rechner im Ethernet können sich solche Adresspaare in ihren ARP-Cache stellen.

Um Änderungen von Zuordnungen zulassen zu können, wenn beispielsweise ein Host konfiguriert wird, um eine neue IP-Adresse zu benutzen (aber seine alte Ethernet-Adresse behält), sollten Einträge im ARP-Cache nach ein paar Minuten zeitlich ablaufen. Eine intelligente Methode, die hilft, zwischengespeicherte Informationen aktuell zu halten und Performanz zu optimieren, lässt jede Maschine ihre Zuordnungstabelle via Broadcast versenden, sobald sie konfiguriert wurde. Dieses Broadcast wird in der Regel durchgeführt, indem mithilfe von ARP die eigene IP-Adresse gesucht wird. Es sollte keine Antwort zurückkommen, aber als Nebeneffekt des Broadcast wurde in allen ARP-Caches ein Eintrag hinzugefügt oder aktualisiert. Dieses Vorgehen wird als **Gratuitous ARP** (unaufgefordertes ARP) bezeichnet. Falls (unerwarteterweise) eine Antwort ankommt, dann wurden zwei Rechnern dieselbe IP-Adresse zugewiesen. Der Fehler muss vom Netzmanager behoben werden, bevor beide Maschinen das Netz benutzen können.

Werfen wir noch einmal einen Blick auf Abbildung 5.61, doch dieses Mal nehmen wir an, dass Host 1 ein Paket an Host 4 (192.32.63.8) im ET-Netz senden möchte. Host 1 wird feststellen, dass die Ziel-IP-Adresse nicht im INF-Netz liegt. Der Host weiß, dass sämtlicher Verkehr, der das Netz verlässt, zum Router zu senden ist, welcher auch als **Default-Gateway** bezeichnet wird. Konventionsgemäß ist das Default-Gateway die niedrigste Adresse im Netz (198.31.65.1). Um einen Rahmen zum Router zu senden, muss Host 1 noch die Ethernet-Adresse der Routerschnittstelle im INF-Netz kennen. Er findet diese heraus, indem ein ARP-Broadcast für 198.31.65.1 gesendet wird, wodurch er *E3* erfährt. Host 1 sendet dann den Rahmen. Derselbe Suchmechanismus wird verwendet, um ein Paket von einem zum nächsten Router über eine Folge von Routern in einem Internetpfad zu senden.

Wenn die Ethernet-Netzwerkkarte des Routers diesen Rahmen bekommt, übergibt sie das Paket an die IP-Software. Der Router weiß anhand der Netzmasks, dass das Paket zum ET-Netz gesendet werden sollte, wo es Host 4 erreichen wird. Falls der Router die Ethernet-Adresse für Host 4 nicht kennt, dann wird er noch einmal ARP einsetzen. Die Tabelle in Abbildung 5.60 listet die Quell- und Zieladressen von Ethernet und IP auf, die im Rahmen enthalten sind. Beachten Sie, dass sich die Ethernet-Adressen mit dem Rahmen auf jedem Netz ändern, während die IP-Adressen konstant bleiben (da sie die Endpunkte in allen verbundenen Netzen angeben).

Es ist auch möglich, ein Paket von Host 1 zu Host 4 zu senden, ohne dass Host 1 weiß, dass sich Host 4 in einem anderen Netz befindet. Dies geschieht, indem der Router auf ARPs im INF-Netz für Host 4 antwortet und seine Ethernet-Adresse *E3* als Antwort sendet. Host 4 kann jedoch nicht direkt antworten, da er die ARP-Anfrage nicht sieht (denn Router leiten keine Broadcasts auf Ethernet-Ebene weiter). Der Router erhält somit Rahmen, die an 192.32.63.8 gesendet wurden, und leitet diese an das ET-Netz weiter. Diese Lösung heißt **Proxy ARP**. Sie wird in speziellen Fällen eingesetzt, in denen ein Host in einem Netz als anwesend erscheinen will, obwohl er sich in Wirklichkeit in einem anderen Netz befindet. Eine übliche Situation ist zum Beispiel ein mobiler Computer, der möchte, dass ein anderer Knoten Pakete für ihn annimmt, wenn er nicht in seinem Heimnetz ist.

DHCP

ARP (ebenso wie andere Internetprotokolle) geht von der Annahme aus, dass Hosts mit einer Art Grundinformation konfiguriert sind, wie beispielsweise ihrer eigenen IP-Adresse. Wie kommen Hosts an diese Information? Es ist möglich, jeden Rechner manuell zu konfigurieren, aber das ist mühsam und fehleranfällig. Es gibt einen besseren Weg – **DHCP** (*Dynamic Host Configuration Protocol*).

Bei DHCP muss jedes Netz einen DHCP-Server haben, der für die Konfiguration verantwortlich ist. Wenn ein Rechner gestartet wird, dann ist die eigene Ethernet-Adresse oder andere Sicherheitsschichtadressen in der Netzwerkkarte integriert, aber keine IP-Adresse. Genau wie bei ARP sendet der Rechner via Broadcast eine Anfrage für eine IP-Adresse in seinem Netz. Dazu setzt er ein DHCP DISCOVER-Paket ein. Dieses Paket muss den DHCP-Server erreichen. Für den Fall, dass dieser Server nicht direkt an das Netz angeschlossen ist, ist der Router konfiguriert, um DHCP-Broadcast zu empfangen und an den DHCP-Server weiterzuleiten, wo immer sich dieser befindet.

Erhält der Server die Anfrage, so weist er eine freie IP-Adresse zu und sendet diese dem Host in einem DHCP OFFER-Paket (welches wiederum über den Router weitergegeben werden kann). Um diese Aufgabe zu erledigen, selbst wenn Hosts keine IP-Adressen haben, identifiziert der Server einen Host über dessen Ethernet-Adresse (die in dem DHCP DISCOVER-Paket übertragen wird).

Eine Frage, die sich bei der automatischen Zuweisung von IP-Adressen aus einem Pool stellt, ist, wie lange eine IP-Adresse zugewiesen bleiben soll. Verlässt ein Host das Netz und gibt seine IP-Adresse nicht an den DHCP-Server zurück, dann ist diese Adresse für immer verloren. Nach einer gewissen Zeit können viele Adressen verloren sein. Um das zu verhindern, kann die Zuweisung der IP-Adresse für eine bestimmte Zeitspanne fest sein. Diese Technik wird als **Leasing** bezeichnet. Kurz bevor das Leasing abläuft, muss der Host DHCP um eine Erneuerung bitten. Wenn er dies nicht tut oder die Anforderung abgewiesen wird, darf der Host die erhaltene IP-Adresse nicht mehr länger verwenden.

DHCP wird in RFC 2131 und RFC 2132 beschrieben. Zusätzlich zu der Bereitstellung von IP-Adressen für Hosts wird es im Internet häufig eingesetzt, um alle Arten von Parametern zu konfigurieren. Sowohl in Geschäfts- als auch Privatnetzen wird DHCP von ISPs verwendet, um die Geräteparameter über die Internetzugangsverbindung einzurichten, sodass Kunden ihre ISPs nicht anrufen müssen, um diese Information zu bekommen. Bekannte Beispiele für Informationen, die so konfiguriert werden, sind die Netzmase, die IP-Adresse des Default-Gateways und die IP-Adresse von DNS und Zeitservern. DHCP hat weitgehend frühere Protokolle (namens RARP und BOOTP) mit begrenzter Funktionalität ersetzt.

5.6.5 Label Switching und MPLS

Bisher haben wir uns auf unserer Reise durch die Vermittlungsschicht des Internets ausschließlich auf Pakete als Datagramme konzentriert, die von IP-Routern weitergeleitet werden. Es gibt aber auch noch eine andere Technologieart, die gerade immer

beliebter wird, besonders bei ISPs zur Übertragung von Internetverkehr durch ihre Netze. Diese Technologie heißt **MPLS** (*MultiProtocol Label Switching*) und reicht gefährlich nahe an Leitungsvermittlung heran. Trotz der Tatsache, dass viele Leute aus der Internetgemeinde eine intensive Abneigung gegen verbindungsorientierte Netze haben, scheint das Konzept eine Renaissance zu erleben. Oder wie Yogi Berra es ausgedrückt hat: Es ist schon wieder wie ein Déjà-vu. Es bestehen aber grundlegende Unterschiede zwischen der Art, wie das Internet und wie verbindungsorientierte Netze den Wegaufbau durchführen, sodass die Technik sicherlich keine herkömmliche Leitungsvermittlung ist.

MPLS fügt einen Kennzeichner (ein Label) an den Anfang eines jeden Pakets ein und die Weiterleitung basiert auf dem Label und nicht auf der Zieladresse. Da das Label als Index in eine interne Tabelle fungiert, wird die Ermittlung der korrekten Ausgangsleitung ein einfacher Nachschlagevorgang in einer Tabelle. Mit diesem Verfahren kann ein Paket sehr schnell weitergeleitet werden. Dies war auch die ursprüngliche Motivation für MLPS, welches als proprietäre Technologie begann und unter verschiedenen Namen bekannt war, einschließlich **Tag Switching**. IETF hat irgendwann mit der Standardisierung der Idee begonnen. MLPS wird in RFC 3031 und vielen anderen RFCs beschrieben. Als Hauptvorteile haben sich im Laufe der Zeit das flexible Routing herausgestellt sowie die Weiterleitung, die sowohl schnell ist als auch dazu geeignet, Dienstgüte zur Verfügung zu stellen.

Die erste Frage ist nun, an welcher Stelle das Label stehen soll. Da IP-Pakete nicht für virtuelle Verbindungen entworfen wurden, gibt es im IP-Header kein Feld für virtuelle Verbindungsnummern. Aus diesem Grund musste ein neuer MPLS-Header vor dem IP-Header hinzugefügt werden. Bei einer Router-zu-Router-Leitung mit PPP als Rahmenbildungsprotokoll entspricht das Rahmenformat, einschließlich der PPP-, MPLS-, IP- und TCP-Header der Darstellung in ►Abbildung 5.62.

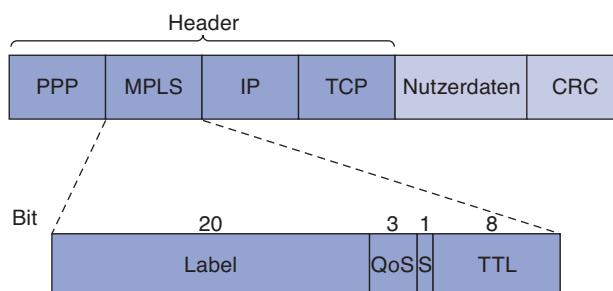


Abbildung 5.62: Übertragung eines TCP-Segments mit IP, MPLS und PPP.

Der allgemeine MPLS-Header ist 4 Byte lang und verfügt über vier Felder. Das wichtigste ist das Feld *Label*, das den Index enthält. Das Feld *QoS* (Quality of Service) gibt die Dienstgüte an. Das Feld *S* bezieht sich auf die Stapel mehrerer Label (wie weiter unten beschrieben wird). Das *TTL*-Feld zeigt an, wie viele Male das Paket noch weitergeleitet werden kann. Es wird bei jedem Router dekrementiert und wenn es 0 erreicht, wird das Paket verworfen. Diese Funktion verhindert unendliche Schleifen im Falle einer instabilen Weiterleitung.

MPLS fällt zwischen das IP-Vermittlungsschichtprotokoll und das PPP-Sicherheitschichtprotokoll. Es ist kein richtiges Schicht-3-Protokoll, da MPLS von IP- oder anderen Vermittlungsschichtadressen abhängt, um die Label-Pfade aufzubauen. Es ist auch kein richtiges Schicht-2-Protokoll, weil Pakete über mehrere Teilstrecken, nicht über eine einzelne Verbindung weitergeleitet werden. Aus diesem Grund wird MPLS manchmal als ein Schicht-2,5-Protokoll beschrieben. Es ist ein Beispiel dafür, dass echte Protokolle nicht immer nahtlos in unser ideales Schichtenprotokollmodell passen.

Da auf der positiven Seite die MPLS-Header nicht zum Vermittlungsschichtpaket oder dem Rahmen der Sicherungsschicht gehören, ist MPLS größtenteils von beiden Schichten unabhängig. Diese Eigenschaft bedeutet unter anderem, dass MPLS-Switches gebaut werden können, die je nach ankommenden Daten sowohl IP-Pakete als auch Nicht-IP-Pakete weiterleiten können. Diese Funktion ist der Grund für die Bezeichnung „Multiprotocol“ in MPLS. MLPS kann außerdem IP-Pakete über Nicht-IP-Netze übertragen.

Wenn ein um MPLS erweitertes Paket an einem **LSR** (*Label Switched Router*) ankommt, wird das Label als Index in eine Tabelle verwendet, um die zu verwendende Ausgangsleitung und das neue Label zu bestimmen. Dieser Austausch von Labels wird in allen VC-Netzen verwendet. Label besitzen nur lokale Bedeutung und zwei verschiedene Router können nicht miteinander in Verbindung stehende Pakete mit dem gleichen Label an einen anderen Router zur Übertragung auf der gleichen Ausgangsleitung schicken. Damit dies am anderen Ende unterscheidbar ist, müssen die Label in jeder Teilstrecke erneut zugewiesen werden. In Abbildung 5.3 hatten wir diesen Mechanismus in Aktion dargestellt. MPLS verwendet das gleiche Verfahren.

Nebenbei bemerkt, einige Leute unterscheiden zwischen *Weiterleitung* und *Switching*. Weiterleitung ist der Prozess des Findens der besten Übereinstimmung für eine Zieladresse in einer Tabelle, um zu entscheiden, wohin Pakete gesendet werden sollen. Ein Beispiel dafür ist der Longest-Prefix-Match-Algorithmus, der für IP-Weiterleitung verwendet wird. Demgegenüber wird beim Switching ein Label aus dem Paket als Index in eine Weiterleitungstabelle benutzt. Das ist einfacher und schneller. Diese Definitionen sind jedoch weit entfernt davon, allgemeingültig zu sein.

Da die meisten Hosts und Router MPLS nicht verstehen, sollten wir auch danach fragen, wann und wie die Label an die Pakete gehängt werden. Dies passiert, wenn ein IP-Paket das Ende eines MPLS-Netzes erreicht. Der **LER** (*Label Edge Router*) untersucht die Ziel-IP-Adresse und andere Felder, um herauszufinden, welchem MPLS-Pfad das Paket folgen sollte, und stellt das richtige Label vor das Paket. Innerhalb des MPLS-Netzes wird dieses Label zur Weiterleitung des Pakets verwendet. Am anderen Ende des MPLS-Netzes hat das Label seinen Zweck erfüllt und wird entfernt, wodurch das IP-Paket für das nächste Netz wieder zum Vorschein kommt. Dieser Prozess ist in ►Abbildung 5.63 gezeigt. Ein Unterschied zu den herkömmlichen virtuellen Verbindungen ist die Ebene der Aggregation. Es ist sicherlich möglich, dass jeder Datenfluss eine eigene Labelmenge in dem MPLS-Netz hat. Es ist aber bei Routern üblicher, dass sie mehrere Datenflüsse gruppieren, die an einem bestimmten Router oder LAN enden und dafür einen Kennzeichner verwenden. Datenflüsse, die unter einem Label grup-

piert werden, werden als zur gleichen **FEC** (*Forwarding Equivalence Class*, Äquivalenzklasse für die Weiterleitung) gehörig bezeichnet. Diese Klasse enthält nicht nur, an welches Ziel die Pakete gehen, sondern auch deren Dienstklasse (im Sinne der differenzierten Dienste), da alle ihre Pakete im Hinblick auf die Weiterleitung gleich behandelt werden.

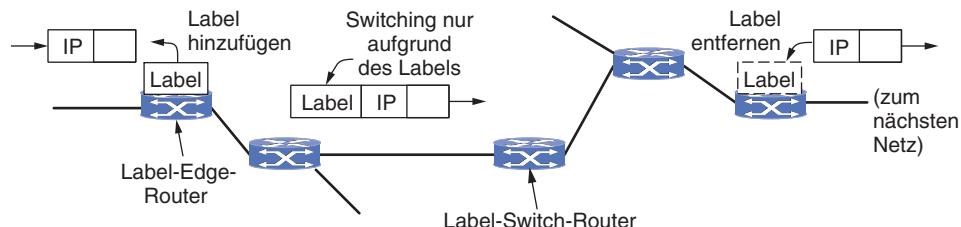


Abbildung 5.63: Weiterleitung eines IP-Pakets durch ein MPLS-Netz.

Bei herkömmlichem Routing über virtuelle Verbindungen ist es nicht möglich, mehrere einzelne Pfade mit verschiedenen Endpunkten durch Verwendung des gleichen Labels für virtuelle Verbindungen zu gruppieren, weil man diese am Ziel nicht auseinanderhalten kann. Bei MPLS enthalten die Pakete zusätzlich zu den Labeln immer noch das Endziel. Am Ende des gekennzeichneten Wegs kann der Label-Header entfernt werden und die Weiterleitung kann auf die übliche Weise mit den Zieladressen der Vermittlungsschicht vorgenommen werden.

Tatsächlich geht MPLS sogar noch weiter. Es kann auf mehreren Ebenen gleichzeitig arbeiten, indem dem Paket mehr als ein Label vorangestellt wird. Nehmen wir zum Beispiel an, dass es viele Pakete mit bereits unterschiedlichen Labeln gibt (weil wir die Pakete im Netz unterschiedlich behandeln möchten), die einem gemeinsamen Pfad zu einem Ziel folgen sollen. Anstatt viele Label-Switching-Pfade aufzubauen – einen für jedes der unterschiedlichen Label –, brauchen wir nur einen einzigen Pfad einzurichten. Wenn die bereits mit Labeln versehenen Pakete den Anfang dieses Pfads erreichen, wird vorne ein weiteres Label hinzugefügt. Dies wird ein Label-Stack genannt. Das äußere Label führt die Pakete den Pfad entlang. Es wird am Ende des Pfads entfernt und die zum Vorschein kommenden Label – falls es welche gibt – werden verwendet, um das Paket weiterzuleiten. Das S-Bit in Abbildung 5.62 ermöglicht es einem Router, ein Label zu entfernen, um herauszufinden, ob es noch zusätzliche Label gibt. Es ist auf 1 gesetzt für das untere Label und auf 0 für alle anderen Labels.

Die letzte Frage, die wir uns stellen wollen, ist, wie die Weiterleitungstabellen bei MPLS angelegt werden. In diesem Bereich gibt es den größten Unterschied zwischen MLPS und konventionellen VC-Entwürfen. Möchte ein Benutzer in herkömmlichen VC-Netzen eine Verbindung aufbauen, wird ein Einrichtungspaket im Netz gestartet, um den Pfad anzulegen und die Einträge für die Weiterleitungstabelle vorzunehmen. MPLS bezieht die Nutzer nicht in die Aufbauphase ein. Zu verlangen, dass Benutzer mehr tun als Datagramme zu senden, würde zu viel der vorhandenen Internetsoftware unbrauchbar machen.

Stattdessen wird die Information zur Weiterleitung von Protokollen eingerichtet, die eine Kombination aus Routing-Protokollen und Verbindungsauflaufprotokollen sind. Diese Steuerprotokolle sind sauber von der Label-Weiterleitung getrennt, wodurch es möglich wird, mehrere unterschiedliche Steuerprotokolle zu verwenden. Eine dieser Varianten arbeitet wie folgt. Wenn ein Router hochgefahren wird, überprüft er, für welche Routen er das Endziel darstellt (z.B. welche Präfixe zu seinen Schnittstellen gehören). Dann erzeugt er einen oder mehrere FECs für sie, ordnet jedem ein Label zu und reicht diese Label an seine Nachbarn weiter. Diese wiederum tragen die Label in ihre Weiterleitungstabellen ein und senden neue Label an ihre Nachbarn, bis alle Router die Pfadinformationen erhalten haben. Ressourcen können ebenfalls während der Konstruktion des Pfads reserviert werden, um eine angemessene Dienstgüte zu garantieren. Andere Varianten bauen eventuell andere Pfade auf – zum Beispiel beziehen Traffic-Engineering-Pfade die ungenutzte Kapazität ein – oder erzeugen Pfade auf Anfrage, um Serviceangebote wie Dienstgüte zu unterstützen.

Obwohl die Konzepte hinter MPLS einfach sind, sind die Details, die mit vielen Variationen und Anwendungsfällen aktiv weiterentwickelt werden, kompliziert. Weiterführende Informationen sind in Davie und Farrel (2008) und Davie und Rekhter (2000) enthalten.

5.6.6 OSPF-Protokoll

Wir haben nun unsere Untersuchung darüber, wie Pakete im Internet weitergeleitet werden, abgeschlossen. Es ist an der Zeit, zum nächsten Thema überzugehen: Routing im Internet. Wie bereits erwähnt, setzt sich das Internet aus einer großen Zahl von unabhängigen Netzen oder **autonomen Systemen (AS)** zusammen, die von verschiedenen Organisationen betrieben werden, in der Regel einer Firma, einer Universität oder einem ISP. Innerhalb ihres eigenen Netzes kann eine Organisation ihren eigenen Algorithmus für internes Routing (oder **Intradomain-Routing**, wie es üblicherweise genannt wird) verwenden. Nichtsdestotrotz gibt es nur eine Handvoll von bekannten Standardprotokollen. In diesem Abschnitt werden wir das Problem des Intradomain-Routings untersuchen und einen Blick auf das OSPF-Protokoll werfen, welches in der Praxis vielfach eingesetzt wird. Ein Intradomain-Routing-Protokoll wird auch **internes Gateway-Protokoll** genannt. Im nächsten Abschnitt werden wir das Problem des Routings zwischen unabhängig arbeitenden Netzen – oder **Interdomain-Routing** – betrachten. Für diesen Fall müssen alle Netze dasselbe Interdomain-Routing-Protokoll bzw. **externe Gateway-Protokoll** verwenden. Das Protokoll, das im Internet eingesetzt wird, ist BGP (*Border Gateway Protocol*).

Frühe Intradomain-Routing-Protokolle verwendeten einen Distanzvektor-Entwurf auf der Grundlage des verteilten Bellman-Ford-Algorithmus aus dem ARPANET. RIP (*Routing Information Protocol*) ist das wichtigste Beispiel hierfür, das bis zum heutigen Tag verwendet wird. Es funktioniert in kleinen Systemen recht gut, aber nicht so gut in größeren Netzen. Außerdem leidet es an dem Count-to-Infinity-Problem und einer allgemein langsamen Konvergenz. Das ARPANET wechselte im Mai 1979 zu einem Link-State-Protokoll für Intradomain-Routing. Dieses Protokoll heißt **OSPF**.

(*Open Shortest Path First*) und wurde 1990 zum Standard erklärt. Es stützt sich auf ein Protokoll namens **IS-IS** (*Intermediate-System to Intermediate-System*), welches ein ISO-Standard wurde. Aufgrund ihres gemeinsamen Erbes haben die beiden Protokolle viel mehr Ähnlichkeiten als Unterschiede. Die vollständige Geschichte finden Sie in RFC 2328. Es sind die vorherrschenden Intradomain-Routing-Protokolle und die meisten Router-Anbieter unterstützen beide. OSPF wird eher in Unternehmensnetzen verwendet und IS-IS eher in ISP-Netzen. Nachfolgend wird die Funktionsweise von OSPF kurz beschrieben.

Angesichts der langjährigen Erfahrungen mit anderen Routing-Protokollen lag der Gruppe, die OSPF entwickelte, eine lange Liste mit Anforderungen vor, die alle erfüllt werden sollten. Der Algorithmus sollte in der offenen Literatur veröffentlicht werden, was das „O“ in der Bezeichnung OSPF erklärt. Es sollte keine proprietäre Lösung von einem Unternehmen sein. Zweitens sollte das neue Protokoll eine Vielzahl von Entfernungsmetriken unterstützen, darunter die physikalische Entfernung, die Übertragungsverzögerung usw. Drittens sollte es ein dynamischer Algorithmus sein, d.h. einer, der sich schnell und automatisch an Änderungen in der Topologie anpasst.

Viertens war bei OSPF neu, dass es Routing auf der Grundlage des Diensttypes unterstützen musste. Das neue Protokoll musste in der Lage sein, Echtzeitverkehr in eine und Datenverkehr in eine andere Richtung zu lenken. Zu der Zeit besaß IP das Feld *Type of Service*, das aber von keinem verfügbaren Routing-Protokoll verwendet wurde. Das Feld wurde auch in OSPF aufgenommen, aber da es immer noch niemand verwendete, wurde es irgendwann entfernt. Vielleicht war diese Anforderung ihrer Zeit voraus, da sie eingefügt wurde, bevor die Arbeit der IETF zu differenzierten Diensten begann, mit der Dienstklassen erneuert wurden.

Fünftens sollte OSPF einen gewissen Lastenausgleich bieten, d.h. die Last auf mehrere Leitungen verteilen. Die meisten früheren Protokolle schickten alle Pakete über eine einzelne beste Route, selbst wenn es zwei Routen gab, die gleich gut waren. Die andere Route wurde überhaupt nicht verwendet. In vielen Fällen führt die Aufteilung der Last auf mehrere Routen zu einer besseren Leistung.

Sechstens war die Unterstützung hierarchischer Systeme gefordert. Im Jahr 1988 waren einige Netze schon derartig angewachsen, dass von keinem Router erwartet werden konnte, die gesamte Topologie zu kennen. OSPF sollte so ausgelegt werden, dass dies von vornherein von keinem Router erwartet werden muss.

Siebtens sollten gewisse Sicherheitsmaßnahmen geboten werden, um Studenten mit Sinn für Humor daran zu hindern, Router durch Übersenden falscher Routing-Daten ganz aus dem Häuschen zu bringen. Schließlich waren Vorkehrungen nötig, um Router zu handhaben, die über einen Tunnel an das Internet angeschlossen sind. Die früheren Protokolle hatten diese Aufgabe nicht gut bewältigt.

OSPF unterstützt sowohl Punkt-zu-Punkt-Verbindungen (z.B. SONET) als auch Broadcast-Netze (z.B. die meisten LANs). Tatsächlich ist OSPF in der Lage, Netze mit mehreren Routern zu unterstützen, von denen jeder direkt mit allen anderen kommunizieren

kann (**Mehrfachzugriffsnetze** genannt), selbst wenn sie keine Broadcast-Fähigkeit haben. Frühere Protokolle konnten mit diesem Fall nicht gut umgehen.

Ein Beispiel eines Netzwerks aus autonomen Systemen ist in ▶ Abbildung 5.64a zu sehen. Bei der Darstellung wurden Hosts weggelassen, da sie in der Regel bei OSPF keine Rolle spielen, im Gegensatz zu Routern und Netzen (welche Hosts enthalten können). Die meisten der Router in Abbildung 5.64a sind mit anderen Routern über Punkt-zu-Punkt-Verbindungen verbunden, und zu Netzen, um die Hosts in diesen Netzen zu erreichen. Die Router R3, R4 und R5 sind jedoch über ein Broadcast-LAN wie Switched Ethernet miteinander verbunden.

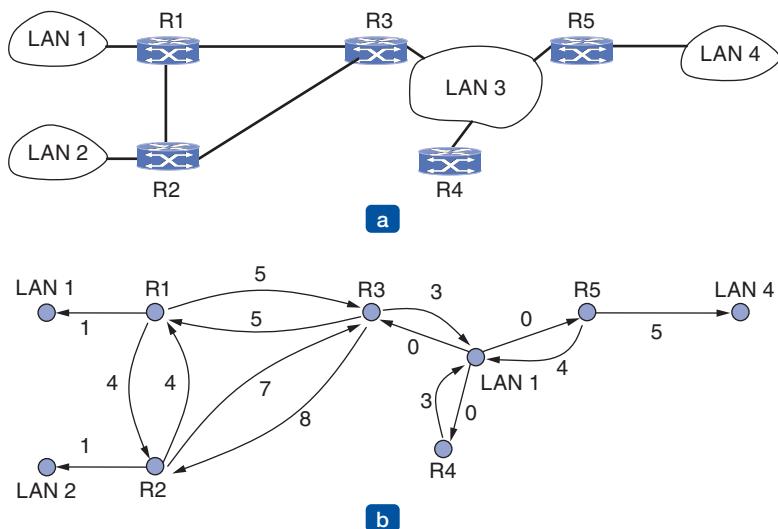


Abbildung 5.64: (a) Autonomes System. (b) Darstellung von (a) als Graph.

OSPF abstrahiert die Anzahl der tatsächlichen Netze, Router und Verbindungen in einen gerichteten Graphen, in dem jeder Kante ein Gewicht (Entfernung, Übertragungsverzögerung usw.) zugewiesen wird. Eine Punkt-zu-Punkt-Verbindung zwischen zwei Routern wird durch ein Kantenpaar darstellt, je eine Kante in eine Richtung. Ihre Gewichtungen können unterschiedlich sein. Ein Broadcast-Netz wird durch einen Knoten für das Netz selbst und einen Knoten für jeden Router dargestellt. Die Kanten von diesem Netzknoten zu den Routern haben das Gewicht 0. Sie sind dennoch wichtig, denn ohne sie gibt es keinen Pfad durch das Netz. Andere Netze, die nur Hosts haben, haben nur eine einfache Kante zum Netz hin und keine ausgehende Kante. Diese Anordnung bietet Routen zu Hosts hin, aber nicht durch sie hindurch.

▶ Abbildung 5.64b stellt den Graphen des Netzes von Abbildung 5.64a dar. OSPF stellt im Grunde das eigentliche Netz als Graphen wie diesen dar und verwendet dann die Link-State-Methode, mit deren Hilfe jeder Router den kürzesten Pfad von ihm zu allen anderen Knoten berechnet. Es kann sein, dass mehrere gleich kurze Pfade gefunden werden. In diesem Fall speichert OSPF die Menge der kürzesten Pfade und während der Paketweiterleitung wird der Verkehr zwischen diesen Pforden aufgeteilt. Diese Strategie, **ECMP** (*Equal Cost MultiPath*) genannt, unterstützt den Lastausgleich.

Viele autonome Systeme im Internet sind selbst groß und nicht so einfach in der Verwaltung. Für den Betrieb in diesem Maßstab kann ein autonomes System mithilfe von OSPF in nummerierte **Bereiche** (area) aufgeteilt werden, wobei ein Bereich ein Netz oder eine Menge zusammenhängender Netze ist. Bereiche überlappen sich nicht, müssen aber nicht erschöpfend sein, was bedeutet, dass eventuell einige Router zu keinem Bereich gehören. Router, die vollständig innerhalb eines Bereichs liegen, werden **interne Router** genannt. Ein Bereich ist eine Generalisierung eines individuellen Netzes. Außerhalb eines Bereichs sind dessen Ziele sichtbar, nicht aber die Topologie. Diese Eigenschaft hilft bei der Skalierung des Routings.

Jedes autonome System hat einen **Backbone-Bereich**, der als Bereich 0 bezeichnet wird. Die Router in diesem Bereich heißen **Backbone-Router**. Alle Bereiche sind mit dem Backbone verbunden, möglicherweise über Tunnel, sodass man über das Backbone von einem beliebigen Bereich eines autonomen Systems zu einem beliebigen anderen Bereich des autonomen Systems gelangen kann. Ein Tunnel wird im Graph lediglich als weitere Kante mit Kosten dargestellt. Wie bei anderen Bereichen ist die Topologie des Backbones außerhalb desselben nicht sichtbar.

Jeder Router, der mit zwei oder mehr Bereichen verbunden ist, wird **Grenzrouter** (*area border router*) genannt. Er muss außerdem Teil des Backbones sein. Die Aufgabe eines Grenzrouters ist es, die Ziele in einem Bereich zusammenzufassen und diese Zusammenfassung an andere Bereiche, mit denen er verbunden ist, weiterzugeben. Die Zusammenfassung enthält Kosteninformationen, aber nicht alle Einzelheiten der Topologie innerhalb eines Bereichs. Das Weiterreichen von Kosteninformation erlaubt es Hosts, in anderen Bereichen die besten Grenzrouter zum Betreten des Bereichs zu finden. Indem keine Informationen über die Topologie weitergereicht werden, wird der Verkehr reduziert und die Berechnungen des kürzesten Pfads von Routern in anderen Bereichen vereinfachen sich. Falls es jedoch nur einen Grenzrouter aus einem Bereich heraus gibt, muss nicht einmal die Zusammenfassung weitergegeben werden. Routen zu Zielen außerhalb eines Bereichs beginnen immer mit der Anweisung „Gehe zum Grenzrouter.“ Diese Art von Bereich heißt **Stub-Bereich**.

Die letzte Art von Router ist der **AS-Grenzrouter** (*AS boundary router*). Er versorgt den Bereich mit Routen zu externen Zielen in anderen autonomen Systemen. Die externen Routen erscheinen dann als Ziele, die über den AS-Grenzrouter mit bestimmten Kosten erreicht werden können. Eine externe Route kann in einen oder mehrere AS-Grenzrouter eingegeben werden. Die Beziehung zwischen autonomen Systemen, Bereichen und verschiedenen Routerarten ist in ► Abbildung 5.65 dargestellt. Ein Router kann mehrere Rollen spielen, zum Beispiel ist ein Grenzrouter gleichzeitig ein Backbone-Router.

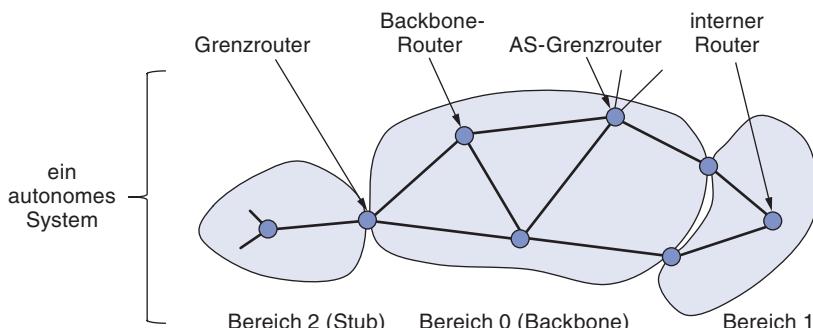


Abbildung 5.65: Die Beziehung zwischen autonomen Systemen, Backbones und Bereichen bei OSPF.

Im normalen Betrieb hat jeder Router innerhalb eines Bereichs die gleiche Link-State-Datenbank und führt den gleichen Algorithmus (des kürzesten Pfads) aus. Seine Hauptaufgabe ist die Berechnung des kürzesten Pfades von sich zu jedem anderen Router und jedem Netz im gesamten autonomen System. Ein Grenzrouter benötigt die Datenbanken von allen Bereichen, mit denen er verbunden ist, und muss den Algorithmus des kürzesten Pfades für jeden Bereich getrennt ausführen.

Befinden sich Quelle und Ziel im selben Bereich, so wird die beste bereichsinterne Route (die vollständig im Bereich liegt) gewählt. Falls Quelle und Ziel in verschiedenen Bereichen liegen, muss die bereichsinterne Route von der Quelle zum Backbone gehen, über das Backbone zum Zielbereich und dann zum Ziel. Dieser Algorithmus zwingt dem OSPF eine Sternkonfiguration auf, in der das Backbone der Hub und die anderen Bereiche die Speichen sind. Da die Route mit den niedrigsten Kosten gewählt wird, kann es sein, dass Router in unterschiedlichen Teilen des Netzes verschiedene Grenzrouter verwenden, um das Backbone und den Zielbereich zu betreten. Pakete werden von der Quelle zum Ziel befördert, „wie sie sind“. Sie werden weder gekapselt noch durch Tunnel geschleust (es sei denn, sie gehen an einen Bereich, dessen einzige Verbindung zum Backbone ein Tunnel ist). Außerdem können Routen zu externen Zielen die externen Kosten von dem AS-Grenzrouter über den externen Pfad enthalten, falls gewünscht, oder nur die internen Kosten des autonomen Systems.

Beim Starten sendet ein Router HELLO-Nachrichten an alle seine Punkt-zu-Punkt-Leitungen und verteilt sie per Multicasting über LANs an die Gruppe, die von allen anderen Routern gebildet wird. Aus den Antworten erfährt jeder Router Einzelheiten über seine Nachbarn. Router im gleichen LAN sind alle Nachbarn.

OSPF tauscht Informationen zwischen aneinander angrenzenden Routern aus. Das ist nicht das Gleiche wie zwischen benachbarten Routern. Insbesondere ist es ineffizient, wenn jeder Router mit jedem anderen Router innerhalb eines LAN kommuniziert. Um diese Situation zu vermeiden, wird ein Router als **designierter Router** (*designated router*) ausgewählt. Dieser Router gilt als **angrenzend** (*adjacent*) zu allen anderen und tauscht mit ihnen Informationen aus. Tatsächlich agiert er als der einzelne Knoten, der das LAN repräsentiert. Benachbarte Router, die nicht aneinander angrenzen, tauschen keine Informationen aus. Ein Backup für den designierten Router wird laufend aktualisiert.

lisiert, um bei Ausfall des primären designierten Routers sofort auf eine Ausweichmöglichkeit umsteigen zu können.

Im Normalbetrieb flutet jeder Router gelegentlich Nachrichten vom Typ LINK STATE UPDATE an alle angrenzenden Router. Diese Nachrichten geben den Status des sendenden Routers und die in der Topologiedatenbank verwendeten Kosten an. Die Nachrichten werden bestätigt, sind also zuverlässig. Jede Nachricht hat eine laufende Nummer, sodass ein Router erkennen kann, ob eine ankommende LINK STATE UPDATE-Nachricht älter oder jünger als die momentan vorliegende ist. Router senden diese Nachrichten auch aus, wenn eine Verbindung zu- oder abgeschaltet wird und wenn sich ihre Kosten ändern.

Nachrichten vom Typ DATABASE DESCRIPTION geben die Sequenznummern aller Link-State-Einträge aus, die momentan beim Sender vorhanden sind. Durch Vergleichen seiner eigenen Werte mit denen des Senders kann der Empfänger feststellen, wer neuere Werte hat. Diese Nachrichten werden ausgesendet, wenn eine Verbindung zugeschaltet wird.

Beide Partner können voneinander mittels Nachrichten vom Typ LINK STATE REQUEST Link-State-Informationen anfordern. Das Ergebnis dieses Algorithmus ist, dass jedes Paar angrenzender Router prüft, wer die aktuellsten Daten hat. Auf diese Weise werden neue Informationen im ganzen Bereich verbreitet. Alle Nachrichten werden direkt als IP-Pakete übertragen. Die fünf Nachrichtenarten sind in ► Abbildung 5.66 zusammengefasst.

| Nachrichtentyp | Beschreibung |
|----------------------|---|
| Hello | Dient zur Feststellung, wer die Nachbarn sind |
| Link State Update | Gibt die Kosten des Senders an seine Nachbarn aus |
| Link State Ack | Bestätigt eine Link-State-Aktualisierung |
| Database Description | Gibt die neuesten Daten des Senders bekannt |
| Link State Request | Fordert Informationen vom Partner an |

Abbildung 5.66: Die fünf Nachrichtentypen von OSPF.

An diesem Punkt angelangt, kann man alle Stücke zusammensetzen. Mithilfe von Fluten informiert jeder Router alle anderen in seinem Bereich über seine Verbindungen zu anderen Routern und Netzen sowie über die Kosten dieser Verbindungen. Anhand dieser Informationen kann jeder Router den Graphen für seinen Bereich bzw. seine Bereiche aufbauen und den kürzesten Pfad berechnen. Das macht auch der Backbone-Bereich. Zusätzlich nehmen die Backbone-Router Informationen von Grenzroutern zwischen Bereichen entgegen, um die beste Route von jedem Backbone-Router zu jedem anderen Router zu berechnen. Diese Information wird an alle Grenzrouten zurückgeschickt, die sie wiederum innerhalb ihrer Bereiche verteilen. Anhand dieser Information können interne Router die beste Route zu einem Ziel außerhalb ihres Bereichs auswählen, einschließlich des besten Ausgangsrouters zum Backbone.

5.6.7 BGP

Innerhalb eines autonomen Systems sind OSPF und IS-IS die Protokolle, die üblicherweise eingesetzt werden. Zwischen mehreren autonomen Systemen wird ein anderes Protokoll verwendet: **BGP** (*Border Gateway Protocol*). Hier wird ein anderes Protokoll benötigt, weil Intradomain- und Interdomain-Protokolle nicht die gleichen Ziele verfolgen. Ein Intradomain-Protokoll muss lediglich Pakete so effizient wie möglich von der Quelle zum Ziel befördern. Es muss sich um keinerlei Regeln kümmern.

Demgegenüber haben Interdomain-Protokolle sehr wohl Regeln, die beachtet werden müssen (Metz, 2001). Ein autonomes System eines Unternehmens muss beispielsweise die Fähigkeit aufweisen, Pakete an jeden beliebigen Internetstandort zu senden und von jedem Internetstandort zu empfangen. Eventuell ist es aber nicht willens, Pakete auf dem Transit von einem fremden autonomen System zu einem anderen fremden autonomen System zu befördern, obwohl es auf dem kürzesten Weg zwischen den beiden fremden Systemen liegt („Das ist deren Problem, nicht meins“). Andererseits ist es vielleicht bereit, Transitverkehr für seine Nachbarn oder bestimmte andere autonome Systeme zu übernehmen, falls sie für diesen Dienst zahlen. Telefongesellschaften stellen ihre Übertragungsdienste gerne ihren Kunden zur Verfügung, nicht aber fremden Parteien. Externe Gateway-Protokolle im Allgemeinen und BGP im Besonderen wurden ausgelegt, um viele Arten von Routing-Regeln, denen der Verkehr zwischen autonomen Systemen unterliegt, zu unterstützen.

Zu den typischen Regeln gehören politische, sicherheitstechnische und wirtschaftliche Überlegungen. Beispiele für mögliche Routing-Einschränkungen sind:

1. Übertrage keinen kommerziellen Datenverkehr auf einem Bildungsnetz.
2. Sende niemals Datenverkehr vom Pentagon auf einer Route durch den Irak.
3. Benutze TeliaSonera anstelle von Verizon, weil es billiger ist.
4. Verwende AT&T nicht in Australien, weil die Leistung schlecht ist.
5. Datenverkehr, der bei Apple beginnt und endet, darf nicht über Google führen.

Wenn Sie diese Aufzählung lesen, können Sie sich sicher vorstellen, dass Routing-Regeln sehr individuell sein können. Häufig sind sie proprietär, weil sie sensible Geschäftsinformationen enthalten. Man kann jedoch zwei Muster ausmachen, die eine Grundlage für die obigen Überlegungen sind und die oft als Ausgangspunkt verwendet werden.

Eine Routing-Regel wird implementiert, indem man entscheidet, welcher Datenverkehr über welche der Verbindungen zwischen autonomen Systemen fließen kann. Eine übliche Regel ist, dass ein Kunden-ISP einen anderen Provider-ISP bezahlt, um Pakete an beliebige andere Ziele im Internet zuzustellen und um Pakete zu erhalten, die von einem beliebigen anderen Ziel gesendet wurden. Man sagt, der Kunden-ISP kauft **Transitdienst** vom ISP des Providers. Dies ist ganz so, wenn ein Kunde zu Hause von einem ISP Internetzugangsdiest kauft. Damit dies funktioniert, stellt der Provider seinen Kunden die

Routen zu allen Zielen im Internet über die Verbindung zur Verfügung, die ihn mit dem Kunden verbindet. Auf diese Weise hat der Kunde eine Route, die er verwenden kann, wenn er Pakete an ein beliebiges Ziel senden möchte. Umgekehrt gibt der Kunde nur Routen zu den Zielen auf seinem eigenen Netz an den Provider weiter. Dadurch wird sichergestellt, dass der Provider dem Kunden nur Verkehr für dessen Adressen sendet; der Kunde möchte keinen Verkehr bearbeiten, der für andere Zielen bestimmt ist.

Wir können ein Beispiel eines Transitdienstes in ►Abbildung 5.67 sehen. Hier sind vier autonome Systeme miteinander verbunden. Die Verbindung wird häufig über **Internetknoten (IXP, Internet eXchange Point)** durchgeführt. Dies sind Einrichtungen, an denen viele ISPs angeschlossen sind, um sich mit anderen ISPs zu verbinden. AS2, AS3 und AS4 sind Kunden von AS1, sie kaufen Transitdienst von AS1. Wenn somit Quelle A an das Ziel C sendet, reisen die Pakete von AS2 zu AS1 und schließlich zu AS4. Die Verteilung der Routen erfolgt in entgegengesetzter Richtung. AS4 teilt seinem Transit-Provider AS1 C als ein Ziel mit, sodass Quellen C über AS1 erreichen können. Später gibt AS1 eine Route zu C an seine anderen Kunden, einschließlich AS2, weiter, damit diese wissen, dass sie Verkehr zu C über AS1 senden können.

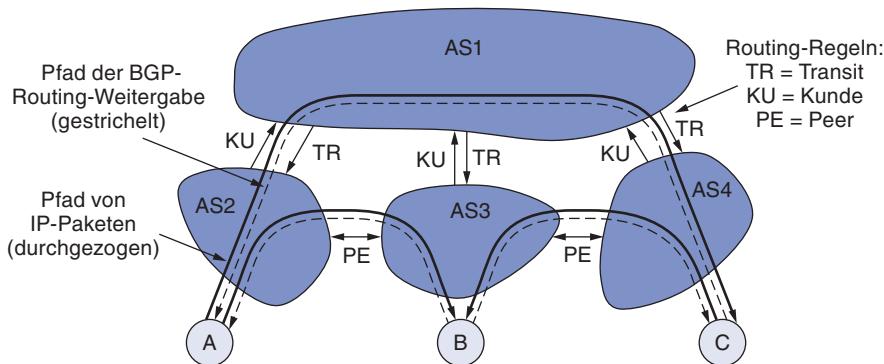


Abbildung 5.67: Routing-Regeln zwischen vier autonomen Systemen.

In Abbildung 5.67 kaufen alle der anderen autonomen Systeme Transitdienst von AS1. Dadurch erhalten sie die Konnektivität, um mit jedem Host im Internet interagieren zu können. Sie müssen für dieses Privileg jedoch bezahlen. Nehmen wir an, dass AS2 und AS3 viel Verkehr austauschen. Vorausgesetzt, dass ihre Netze bereits verbunden sind, dann können AS2 und AS3 eine abweichende Routing-Regel verwenden, falls sie möchten – sie können Verkehr kostenlos direkt miteinander austauschen. Dies verringert die Menge an Verkehr, den AS1 in ihrem Namen zustellt und wird hoffentlich ihre Rechnungen senken. Diese Methode wird **Peering** genannt.

Zur Implementierung von Peering teilen zwei autonome Systeme ihre Routing-Informationen für die Adressen in ihren Netzen miteinander. Daher kann AS2 Pakete für AS3 von A nach B und umgekehrt senden. Beachten Sie jedoch, dass Peering nicht transitiv ist. In Abbildung 5.67 gehen AS3 und AS4 eine Peering-Verbindung miteinander ein. Dieses Peering ermöglicht es, den Datenverkehr von C direkt nach B zu senden. Was passiert, wenn C ein Paket zu A sendet? AS3 gibt nur seine Route zu B an

AS4 weiter, nicht jedoch seine Route zu A. Daher wird kein Verkehr von AS4 über AS3 zu AS2 weitergeleitet, selbst wenn ein physischer Pfad existiert. Genau diese Einschränkung ist von AS3 gewünscht: es unterhält eine Peering-Verbindung zu AS4 zum Austausch von Datenverkehr, möchte aber keinen Verkehr von AS4 zu anderen Teilen des Internets übertragen, da AS3 dafür nicht bezahlt wird. Stattdessen erhält AS4 Transitdienst von AS1. Somit wird das Paket von C nach A von AS1 übermittelt.

Nun, da wir etwas über Transit und Peering wissen, können wir auch erkennen, dass A, B und C Transitvereinbarungen haben. Zum Beispiel muss A Internetzugang von AS2 kaufen. A könnte ein einzelner privater Rechner oder ein Firmennetzwerk mit vielen LANs sein. A muss jedoch nicht BGP ausführen, weil es ein **Stub-Netz** ist, das an den Rest des Internets nur über eine Verbindung angeschlossen ist. A muss daher alle Pakete, die an ein Ziel außerhalb des Netzes gerichtet sind, über die Verbindung zu AS2 senden kann. Es gibt keine andere Möglichkeit. Dieser Pfad kann leicht aufgebaut werden, indem eine Standardroute eingerichtet wird. Aus diesem Grund haben wir A, B und C nicht als autonome Systeme dargestellt, die am Interdomain-Routing teilnehmen.

Andererseits sind einige Firmennetze mit mehreren ISPs verbunden. Diese Technik wird verwendet, um die Zuverlässigkeit zu verbessern: Falls der Pfad des einen ISP ausfällt, kann das Unternehmen den Pfad über einen anderen ISP benutzen. Diese Technik heißt **Multihoming**. Hierbei führt das Firmennetz vermutlich ein Interdomain-Routing-Protokoll (z.B. BGP) aus, um anderen autonomen Systemen mitzuteilen, welche Adressen über welche ISP-Verbindung erreicht werden sollen.

Viele Variationen dieser Transit- und Peering-Regeln sind möglich, aber die bisher gezeigten lassen bereits erkennen, inwiefern Geschäftsbeziehungen und die Kontrolle darüber, an wen Routen weitergegeben werden, unterschiedliche Arten von Regeln nach sich ziehen können. Nun werden wir uns detaillierter ansehen, wie Router, die BGP ausführen, untereinander Routen anzeigen und Pfade auswählen, über die Pakete weitergeleitet werden sollen.

BGP ist eine Form von Distanzvektorprotokoll, unterscheidet sich aber beträchtlich von Intradomain-Distanzvektorprotokollen wie RIP. Wir haben bereits festgestellt, dass zur Auswahl der zu verwendenden Route Regeln anstelle der minimalen Entfernung benutzt werden. Ein weiterer bedeutender Unterschied ist, dass anstatt einfach nur die Kosten der Route für jedes Ziel zu verwalten, jeder BGP-Router auch den verwendeten Pfad verfolgt. Dieser Ansatz heißt **Pfad-Vektor-Protokoll**. Der Pfad besteht aus dem nächsten Teilstrecken-Router (der nicht unbedingt angrenzend sein muss, sondern sich auf der anderen Seite des ISP befinden kann) und der Folge von autonomen Systemen oder **AS-Pfad**, der die Route gefolgt ist (in umgekehrter Reihenfolge). Schließlich kommunizieren Paare von BGP-Routern durch Aufbau von TCP-Verbindungen miteinander. Diese Betriebsweise bietet eine zuverlässige Kommunikation und verbirgt alle Einzelheiten des durchquerten Netzes.

Ein Beispiel dafür, wie BGP-Routen verbreitet werden, ist in ► Abbildung 5.68 gezeigt. Hier haben wir drei autonome Systeme, das mittlere stellt Transit für den linken und den rechten ISP zur Verfügung. Eine Weitergabe von Routen zu Präfix C beginnt in

AS3. Läuft diese Weitergabe über die Verbindung zu $R2c$ oben in der Abbildung, so besteht der AS-Pfad einfach aus AS3 und der nächste Teilstrecke ist $R2a$. Die Weitergabe über die untere Verbindung liefert denselben AS-Pfad, aber eine unterschiedliche nächste Teilstrecke. Bei Fortführung der Weitergabe wird die Grenze zu AS1 überschritten. Am Router $R1a$ oben in der Abbildung lautet der AS-Pfad AS2, AS3 und die nächste Teilstrecke ist $R2a$.

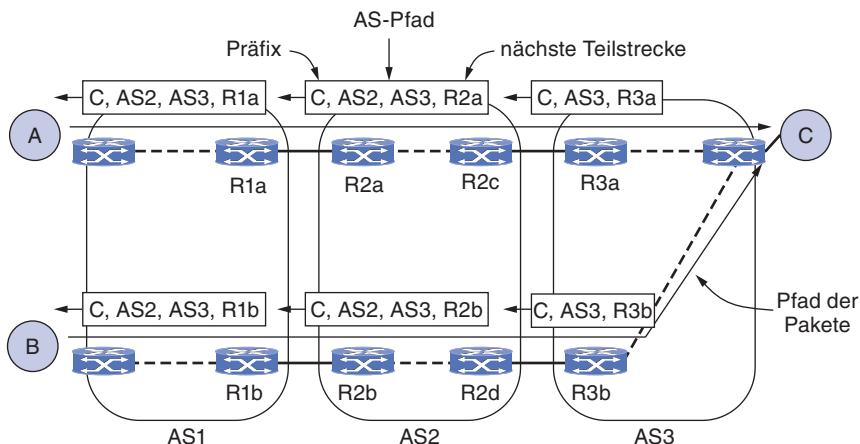


Abbildung 5.68: Weiterleitung von BGP-Routen.

Das Übertragen des vollständigen Pfads mit der Route vereinfacht es dem empfangenen Router, Schleifen im Routing zu erkennen und zu durchbrechen. Die Regel ist, dass jeder Router, der eine Route außerhalb des autonomen Systems sendet, seine eigene AS-Nummer der Route voranstellt. (Deshalb ist die Liste in umgekehrter Reihenfolge.) Wenn ein Router eine Route empfängt, überprüft er, ob seine eigene AS-Nummer schon in dem AS-Pfad enthalten ist. Ist dies der Fall, dann wurde eine Schleife entdeckt und diese Route wird nicht weitergegeben. Jedoch – und irgendwie ironisch – hat man in den späten 1990er Jahren herausgefunden, dass trotz dieser Vorsichtsmaßnahme BGP an einer Form des Count-to-Infinity-Problems leidet (Labovitz et al., 2001). Es gibt keine langlebigen Schleifen, aber Routen konvergieren manchmal langsam und haben vorübergehende Schleifen.

Eine Liste von autonomen Systemen anzugeben, ist eine sehr primitive Art und Weise, einen Pfad zu spezifizieren. Ein autonomes System kann eine kleine Firma oder ein internationales Backbonenetz sein. Es besteht keine Möglichkeit, dies aus der Route abzulesen. BGP versucht dies nicht einmal, da unterschiedliche autonome Systeme unterschiedliche Intradomain-Protokolle verwenden können, deren Kosten man nicht vergleichen kann. Und selbst wenn sie vergleichbar wären, kann es sein, dass ein autonomes System seine internen Metriken nicht preisgeben möchte. Dies ist ein Beispiel dafür, wie sich Interdomain-Routing-Protokolle von Intradomain-Protokollen unterscheiden.

Bisher haben wir gesehen, wie eine Routen-Weitergabe über Verbindungen zwischen zwei ISPs gesendet wird. Wir benötigen noch Möglichkeiten, BGP-Routen von einer Seite des ISP zur anderen zu leiten, sodass sie zum nächsten ISP gesendet werden können. Diese Aufgabe könnte von Interdomain-Protokollen erledigt werden, doch weil BGP sehr gut zum Skalieren auf große Netze geeignet ist, wird häufig eine Variante von BGP verwendet. Diese heißt **internes BGP (iBGP)**, um es vom regulären Einsatz von BGP als **externes BGP (eBGP)** zu unterscheiden.

Die Regel für das Weiterleiten von Routen innerhalb eines ISP besagt, dass jeder Router an der Grenze des ISP aus Gründen der Konsistenz von allen Routen erfährt, die für alle anderen AS-Grenzrouten sichtbar sind. Falls ein AS-Grenzrouter des ISP von einem Präfix zu IP 128.208.0.0/16 Kenntnis erhält, so werden alle anderen Router von diesem Präfix erfahren. Das Präfix wird dann von allen Teilen des ISP aus erreichbar sein, unabhängig davon, wie Pakete den ISP von anderen autonomen Systemen betreten.

Wir haben diese Weiterleitung in Abbildung 5.68 nicht gezeigt, um ein Durcheinander zu vermeiden, aber beispielsweise Router *R2b* wird wissen, dass er *C* entweder über Router *R2c* (oben) oder über Router *R2d* (unten) erreichen kann. Die nächste Teilstrecke wird aktualisiert, wenn sich die Route innerhalb des ISP überschneidet, sodass Router auf der entfernten Seite des ISP wissen, welche Router sie verwenden müssen, um den ISP auf der anderen Seite zu verlassen. Dies kann man an den Wegen ganz links in der Abbildung sehen: die nächste Teilstrecke verweist auf einen Router im selben ISP und nicht auf einen Router im nächsten ISP.

Wir können nun zu dem entscheidenden, noch fehlenden Puzzlestück kommen: wie BGP-Router entscheiden, welche Route für jedes Ziel verwendet wird. Jeder BGP-Router kann von einem Weg für ein bestimmtes Ziel von dem Router erfahren, mit dem er im nächsten ISP verbunden ist, sowie von allen anderen AS-Grenzroutern (die wiederum von verschiedenen Wegen von den Routern gehört haben, mit denen sie in anderen ISPs verbunden sind). Jeder Router muss entscheiden, welcher Weg aus dieser Routenmenge der beste ist. Letztendlich sieht die Antwort so aus, dass es Aufgabe des ISP ist, einige Regeln zur Auswahl der bevorzugten Route aufzustellen. Diese Erklärung ist jedoch sehr allgemein und überhaupt nicht befriedigend. Deshalb wollen wir zumindest einige allgemeine Strategien beschreiben.

Die erste Strategie ist, dass Routen über Netze, die mittels Peering miteinander verbunden sind, vor Routen über Transit-Provider bevorzugt werden. Die ersten sind frei – die zweiten kosten Geld. Eine ähnliche Strategie ist, Kundenrouten die höchste Präferenz zu geben. Es ist nur gutes Geschäftsgefahren, Verkehr direkt an die zahlenden Kunden zu senden.

Eine andere Art Strategie ist die Standardregel, dass kürzere AS-Pfade besser sind. Diese Regel ist jedoch fragwürdig, wenn man bedenkt, dass ein autonomes System ein Netz beliebiger Größe sein kann, ein Pfad durch drei kleine autonome Systeme also tatsächlich kürzer sein könnte als ein Pfad durch ein großes autonomes System. Im Durchschnitt scheint „kürzer“ aber „besser“ zu sein, und diese Regel ist ein übliches Entscheidungskriterium.

Die letzte Strategie ist die Bevorzugung von Routen, die innerhalb des ISP die geringsten Kosten haben. Diese Strategie ist in Abbildung 5.68 realisiert. Pakete, die von A zu C gesendet werden, verlassen $AS1$ am oberen Router $R1a$. Pakete, die von B aus gesendet werden, nehmen den Ausgang über den unteren Router $R1b$. Der Grund dafür ist, dass sowohl A als auch B den Pfad mit den niedrigsten Kosten bzw. die schnellste Route aus $AS1$ heraus wählen. Da A und B sich in unterschiedlichen Teilen des ISP befinden, ist der schnellste Ausgang jeweils ein anderer. Dasselbe passiert, wenn die Pakete $AS2$ passieren. Auf der letzten Etappe muss $AS3$ das Paket von B durch sein eigenes Netz transportieren.

Diese Strategie ist als **Early-Exit-Routing** oder **Hot-Potato-Routing** bekannt. Es gibt hierbei den seltsamen Nebeneffekt, dass Routen tendenziell asymmetrisch werden. Betrachten Sie zum Beispiel den Pfad, der gewählt wird, wenn C ein Paket zurück zu B sendet. Das Paket verlässt $AS3$ schnell, und zwar am oberen Router, damit möglichst wenige Ressourcen verschwendet werden. Aus dem gleichen Grund wird das Paket auf der oberen Route bleiben, wenn $AS2$ es so schnell wie möglich zu $AS1$ weiterreicht. Danach hat das Paket eine längere Reise in $AS1$ vor sich. Dies ist ein Spiegelbild des Pfads, der von B zu C genommen wurde.

Die obige Diskussion sollte verdeutlichen, dass jeder BGP-Router seine eigene beste Route aus den bekannten Möglichkeiten auswählt. Es ist nicht der Fall – wie man naiv annehmen könnte –, dass BGP den zu folgendem Pfad auf AS-Ebene auswählt und OSPF die Pfade innerhalb des jeweiligen autonomen Systems. BGP und das interne Gateway-Protokoll sind weit tiefer integriert. Dies bedeutet, dass zum Beispiel BGP den besten Austrittspunkt von einem ISP zum nächsten finden kann und dass dieser Punkt innerhalb eines ISP variieren kann, wie im Fall der Hot-Potato-Regel. Es bedeutet außerdem, dass BGP-Router in unterschiedlichen Bereichen eines autonomen Systems unterschiedliche AS-Pfade wählen können, um dasselbe Ziel zu erreichen. Der ISP muss bei der Konfiguration aller BGP-Router große Sorgfalt an den Tag legen, um angesichts all dieser Freiheiten eine passende Auswahl zu treffen, doch dies kann in der Praxis durchgeführt werden.

Erstaunlicherweise haben wir nur die Oberfläche von BGP angekratzt. Für weitere Informationen verweisen wir auf die Spezifikation von BGP Version 4 in RFC 4271 und verwandte RFCs. Halten Sie sich jedoch vor Augen, dass ein Großteil der Komplexität von den Regeln abhängt, die in der Spezifikation des BGP-Protokolls nicht beschrieben sind.

5.6.8 Internet-Multicasting

Die übliche IP-Kommunikation findet zwischen einem Sender und einem Empfänger statt. Bei manchen Anwendungen ist es aber für einen Prozess nützlich, an viele Empfänger gleichzeitig zu senden. Beispiele hierfür sind das Live-Streaming von Sportveranstaltungen an viele Benutzer, das Senden von Programmaktualisierungen an einen Pool von replizierten Servern sowie die Verwaltung digitaler Telefonkonferenzen (d.h. mit mehreren Teilnehmern).

IP unterstützt One-to-Many-Kommunikation oder Multicasting unter Verwendung von IP-Adressen der Klasse D. Jede Klasse-D-Adresse gibt eine Gruppe von Hosts an. Zur Bestimmung von Gruppen sind 28 Bit verfügbar, sodass über 250 Millionen Gruppen gleichzeitig existieren können. Sendet ein Prozess ein Paket an eine Adresse der Klasse D, wird es allen Mitgliedern der adressierten Gruppe nach so gut es geht (Best Effort), aber ohne Garantien zugestellt. Eventuell erhalten einige Mitglieder das Paket nicht.

Der IP-Adressen-Bereich 224.0.0.0/24 ist für Multicasting auf dem lokalen Netz reserviert. In diesem Fall wird kein Routing-Protokoll benötigt. Die Pakete werden an alle Mitglieder geschickt, indem sie einfach via Broadcasting auf dem LAN mit einer Multicast-Adresse gesendet werden. Alle Hosts auf dem LAN empfangen den Broadcast und die Hosts, die Mitglieder der Gruppe sind, verarbeiten das Paket. Router leiten das Paket nicht aus dem LAN hinaus. Beispiele für lokale Multicast-Adressen sind:

- 224.0.0.1 Alle Systeme in einem LAN
- 224.0.0.2 Alle Router in einem LAN
- 224.0.0.5 Alle OSPF-Router in einem LAN
- 224.0.0.251 Alle DNS-Server in einem LAN

Andere Multicast-Adressen können Mitglieder in verschiedenen Netzen haben. In diesem Fall wird ein Routing-Protokoll benötigt. Aber zunächst müssen die Multicast-Router wissen, welche Hosts Mitglieder einer Gruppe sind. Ein Prozess bittet seinen Host, ihn in eine bestimmte Gruppe aufzunehmen oder diese wieder zu verlassen. Jeder Host verwaltet die Prozesse, die momentan an einer Gruppe teilnehmen. Verlässt der letzte Prozess eines Hosts eine Gruppe, ist der Host nicht länger Mitglied dieser Gruppe. Etwa einmal pro Minute sendet jeder Multicast-Router ein Anfragepaket an alle Hosts in seinem LAN (natürlich unter Verwendung der lokalen Multicast-Adresse 224.0.0.1) und fordert sie auf, die Gruppen zu melden, zu denen sie momentan gehören. Die Multicast-Router können mit Standardroutern am gleichen Ort existieren, aber sie müssen nicht. Jeder Host sendet seine Antworten für alle Adressen der Klasse D zurück, an denen er interessiert ist. Diese Anfrage-Antwort-Pakete benutzen ein Protokoll namens **IGMP** (*Internet Group Management Protocol*, Protokoll zur Verwaltung von Gruppen im Internet). IGMP wird in RFC 1112 beschrieben.

Jedes der vielen Multicast-Routing-Protokolle kann benutzt werden, um Multicast-Spannbäume aufzubauen, die Pfade von Sendern zu allen Mitgliedern der Gruppe liefern. Die verwendeten Algorithmen sind dieselben, die wir in Abschnitt 5.2.8 beschrieben haben. Innerhalb eines autonomen Systems wird **PIM** (*Protocol Independent Multicast*) als Hauptprotokoll eingesetzt. PIM gibt es in mehreren Ausführungen. Bei PIM Dense Mode wird ein gestützter Reverse-Path-Forwarding-Baum erzeugt. Dieser Modus ist auf Situationen zugeschnitten, bei denen sich die Mitglieder überall im Netz befinden, beispielsweise verteilte Dateien auf vielen Servern innerhalb eines Netzes für Datenzentren. Bei PIM Sparse Mode ähneln die aufgebauten Spannbäume den Core-Based Trees. Dieser Modus ist in Situationen angemessen, in denen beispielsweise ein Content-Provider TV-Inhalte per Multicasting zu vielen Teilnehmern in seinem IP-Netz sendet. Eine Variante dieses Entwurfs, Source-Specific Multicast PIM, ist

für den Fall optimiert, dass es nur einen Sender für die Gruppe gibt. Schließlich müssen Multicast-Erweiterungen für BGP oder Tunneling verwendet werden, um Multicast-Routen zu erzeugen, wenn sich die Gruppenmitglieder in mehr als einem autonomen System befinden.

5.6.9 Mobiles IP

Viele Internetbenutzer haben mobile Computer und möchten auch dann Zugang zum Internet haben, wenn sie nicht zu Hause sind. Leider ist der Zugang zum Internet über das IP-Adressierungssystem unterwegs leichter gesagt als getan, was wir im Folgenden kurz beschreiben wollen. Als die Nachfrage trotzdem anstieg, stellte die IETF eine Arbeitsgruppe zusammen, um eine Lösung auszuarbeiten. Diese Arbeitsgruppe formulierte in kurzer Zeit eine Reihe von Zielen. Die wichtigsten sind:

- 1.** Jeder mobile Host muss in der Lage sein, seine IP-Heimatadresse an jedem beliebigen Ort zu benutzen.
- 2.** Softwareänderungen in festen Hosts sind nicht zulässig.
- 3.** Änderungen der Router-Software und -Tabellen sind nicht zulässig.
- 4.** Die meisten Pakete für mobile Hosts sollen bei der Übertragung keine Umwege machen.
- 5.** Während der mobile Host an seinem Heimatort steht, soll kein Overhead entstehen.

Letztlich wurde die in Abschnitt 5.2.10 beschriebene Lösung gewählt. Um es kurz zusammenzufassen: Jeder Standort, der seinen Benutzern mobilen Anschluss gewähren will, muss dabei einen Helper zur Verfügung stellen: den **Heimatagenten**. Wenn ein mobiler Host an einem fremden Standort auftaucht, erhält er dort eine neue IP-Adresse (die sogenannte Care-of-Adresse). Das mobile Gerät teilt dann dem Heimatagenten mit, wo er nun ist, indem er ihm die Care-of-Adresse gibt. Wenn ein Paket für das mobile Gerät am Heimatstandort ankommt und das mobile Gerät anderswo ist, dann nimmt sich der Heimatagent das Paket sendet es via Tunneling zum mobilen Gerät an die aktuelle Care-of-Adresse. Das mobile Gerät kann Antwortpakete direkt zu demjenigen senden, mit dem es kommuniziert, aber trotzdem seine Heimatadresse als Quelladresse verwenden. Diese Lösung erfüllt alle oben aufgeführten Anforderungen, außer dass Pakete für mobile Hosts Umwege machen.

Nun da wir die Vermittlungsschicht des Internets behandelt haben, können wir die Lösung detaillierter betrachten. Dass Mobilitätsunterstützung überhaupt notwendig ist, liegt am IP-Adressierungsschema selbst. Jede IP-Adresse enthält eine Netznummer und eine Hostnummer. Betrachten wir als Beispiel den Rechner mit der IP-Adresse 160.80.40.20/16. Die Zahl 160.80 gibt die Netznummer an; die Zahl 40.20 ist die Hostnummer. Router in der ganzen Welt haben Routing-Tabellen, aus denen hervorgeht, welche Leitung zu benutzen ist, um Netz 160.80 zu erreichen. Kommt ein Paket für ein Ziel mit der IP-Adresse in der Form 160.80.xxx.yyy an, wird es auf dieser Leitung übertragen. Wird der Rechner mit dieser Adresse plötzlich an einen entfernten Ort

verschleppt, werden die an ihn adressierten Pakete trotzdem nach wie vor an sein Heimat-LAN (oder seinen Router) übertragen.

Jetzt gibt es zwei Optionen – beide sind unattraktiv. Die erste besteht darin, dass wir eine Route zu einem spezifischeren Präfix erzeugen könnten. Das heißt, falls der entfernte Standort eine Route zu 160.80.40.20/32 verbreitet, dann kommen Pakete, die zu dem Ziel gesendet werden, wieder am richtigen Ort an. Diese Option hängt von dem Longest-Prefix-Match-Algorithmus ab, der von den Routern eingesetzt wird. Wir haben jedoch eine Route zu einem IP-Präfix hinzugefügt, die nur eine einzige IP-Adresse enthält. Alle ISPs auf der Welt werden von diesem Präfix erfahren. Falls jeder, der mit seinem Rechner unterwegs ist, auf diese Weise globale IP-Routen ändert, dann würde jeder Router Millionen von Einträgen in seiner Tabelle verwahren, was im Internet astronomische Kosten bedeutet. Diese Option ist nicht praktikabel.

Die zweite Möglichkeit besteht darin, die IP-Adresse des mobilen Geräts zu verändern. Dann werden allerdings Pakete, die zur Heimat-IP-Adresse gesendet werden, so lange nicht ausgeliefert, bis alle betroffenen Menschen, Programme und Datenbanken über die Änderung informiert sind. Aber das mobile Gerät kann trotzdem am neuen Standort das Internet benutzen, um im Web zu browsen und andere Anwendungen auszuführen. Diese Option behandelt Mobilität auf einer höheren Schicht. Dies ist die typische Situation, wenn ein Benutzer mit einem Laptop in einem Café sitzt und dort das Internet über das lokale drahtlose Netz benutzt. Der Nachteil ist, dass einige Anwendungen abgebrochen werden und die Verbindung nicht aufrechterhalten wird, wenn sich das mobile Gerät bewegt.

Nebenbei bemerkt, Mobilität kann auch auf einer niedrigeren Schicht behandelt werden, der Sicherungsschicht. Dies ist der Fall, wenn ein Laptop auf einem einzelnen drahtlosen IEEE- 802.11-Netz verwendet wird. Die IP-Adresse des mobilen Geräts ändert sich nicht und der Netzpfad bleibt derselbe. Die Mobilität wird durch die drahtlose Verbindung zur Verfügung gestellt. Der Grad an Mobilität ist jedoch begrenzt. Falls der Laptop sich zu weit weg bewegt, muss es sich mit dem Internet über ein anderes Netz mit einer anderen IP-Adresse verbinden.

Die Lösung für mobiles IP für IPv4 ist in RFC 3344 festgehalten. Diese Lösung arbeitet mit dem vorhandenen Internet-Routing zusammen und ermöglicht es Hosts, mit ihren eigenen IP-Adressen verbunden zu bleiben, wenn sie sich bewegen. Dazu muss das mobile Gerät feststellen können, wenn es sich bewegt hat. Dies wird mithilfe von ICMP-Router-Advertisement und Anfragenachrichten durchgeführt. Mobile Geräte rufen regelmäßige Router-Advertisements ab oder führen eine Router-Solicitation aus, um den nächstgelegenen Router zu entdecken. Falls dieser Router nicht der üblichen Router-Adresse entspricht, wenn das mobile Gerät zu Hause ist, dann muss sich das mobile Gerät in einem fremden Netz befinden. Falls dieser Router sich seit dem letzten Mal geändert hat, hat sich das mobile Gerät zu einem anderen fremden Netz bewegt. Mithilfe desselben Mechanismus können mobile Hosts ihre Heimatagenten finden.

Um eine Care-of-IP-Adresse im fremden Netzwerk zu bekommen, kann ein mobiles Gerät einfach DHCP verwenden. Alternativ, wenn IPv4-Adressen knapp sind, kann

das mobile Gerät Pakete über einen Fremdagagenten senden und empfangen, der schon eine IP-Adresse in dem Netz hat. Der mobile Host findet einen Fremdagagenten, der denselben ICMP-Mechanismus verwendet, mit dem er üblicherweise den Heimatagenten findet. Nachdem das mobile Gerät eine IP-Adresse erhalten hat oder einen Fremdagagenten gefunden hat, ist es in der Lage, das Netz zu benutzen, um eine Nachricht zu seinem Heimatagenten zu senden. Mit dieser Nachricht informiert er den Heimatagenten über seinen aktuellen Standort.

Der Heimatagent benötigt eine Möglichkeit um Pakete abzufangen, die zum mobilen Gerät gesendet werden, solange das mobile Gerät nicht zu Hause ist. ARP bietet hierfür einen bequemen Mechanismus. Um ein Paket über ein Ethernet zu einem IP-Host zu senden, muss der Router die Ethernet-Adresse des Hosts kennen. Das übliche Vorgehen ist, dass der Router eine ARP-Anfrage sendet, beispielsweise um zu fragen, wie die Ethernet-Adresse von 160.80.40.20 lautet. Ist das mobile Gerät zu Hause, dann beantwortet es solche ARP-Anfragen nach seiner IP-Adresse mit seiner eigenen Ethernet-Adresse. Wenn das mobile Gerät unterwegs ist, antwortet der Heimatagent auf diese Anfrage durch Mitteilung seiner Ethernet-Adresse. Der Router sendet dann Pakete für 160.80.40.20 zum Heimatagenten. Wir erinnern daran, dass diese Lösung Proxy ARP genannt wird.

Zur schnellen Aktualisierung von ARP-Zuordnungen, wenn das mobile Gerät das Haus verlässt oder zum Haus zurückkommt, kann eine weitere ARP-Technik namens **Gratuitous ARP** (unaufgefordertes ARP) eingesetzt werden. Grundsätzlich sendet das mobile Gerät oder der Heimatagent selbst eine ARP-Anfrage für die mobile IP-Adresse, die die richtige Antwort liefert, welche der Router zur Kenntnis nimmt und seine Zuordnung aktualisiert.

Tunneling zum Senden eines Paket zwischen dem Heimatagenten und dem mobilen Host an die Care-of-Adresse wird durchgeführt, indem das Paket innerhalb eines anderen IP-Headers gekapselt wird, der an die Care-of-Adresse gerichtet ist. Wenn das gekapselte Paket an der Care-of-Adresse ankommt, wird der äußere IP-Header entfernt, um das Paket zum Vorschein zu bringen.

Wie bei vielen Internetprotokollen steckt der Teufel im Detail – und zwar meistens in den Details der Kompatibilität mit anderen installierten Protokollen. Es gibt zwei Komplikationen. Die erste Schwierigkeit ist, dass NAT-Boxen einen Blick hinter den IP-Header werfen, um sich den TCP- oder UDP-Header anzuschauen. Die ursprüngliche Form des Tunneling mit mobilem IP benutzte diese Header nicht, deshalb konnte dieses Verfahren nicht mit NAT-Boxen angewendet werden. Die Lösung war, die Verkapselung so zu ändern, dass ein UDP-Header enthalten ist.

Die zweite Komplikation ist, dass einige ISPs die Quell-IP-Adresse von Paketen überprüfen, um festzustellen, ob sie mit dem tatsächlichen Standort übereinstimmt, den das Routing-Protokoll als Quelle vermutet. Diese Technik heißt **Ingress-Filter** und ist eine Sicherheitsmaßnahme, die dazu dient, Verkehr zu verwerfen, der von einer anscheinend falschen Adresse kommt und bösartig sein könnte. Pakete jedoch, die von einem mobilen Gerät von einem fremden Netz aus zu anderen Internethosts

gesendet werden, werden eine Quell-IP-Adresse haben, die nicht ihrem Standort entspricht, deshalb werden sie verworfen. Um dieses Problem zu umgehen, kann das mobile Gerät die Care-of-Adresse als Quelle verwenden, um die Pakete per Tunneling zurück zum Heimatagenten zu senden. Von hier aus werden sie in das Internet verschickt, sodass der richtige Standort als Quelle erscheint. Der Preis dafür ist, dass die Route jetzt umständlicher ist.

Ein weiteres Thema, das wir nicht besprochen haben, ist Sicherheit. Wenn ein Heimatagent eine Nachricht mit der Anfrage bekommt, ob jetzt bitte alle Pakete von Robert zu einer anderen IP-Adresse weiterzuleiten, dann sollte diese Anweisung besser nicht befolgt werden, wenn man nicht sicher weiß, dass Robert die Quelle dieser Anforderung ist und nicht irgendjemand anderes, der versucht, sich als Robert auszugeben. Aus diesem Grund werden Protokolle zur kryptografischen Authentifizierung eingesetzt. Wir werden solche Protokolle in *Kapitel 8* untersuchen.

Mobilitätsprotokolle für IPv6 bauen auf der IPv4-Grundlage auf. Das oben angegebene Verfahren leidet am Triangel-Routing-Problem: Pakete, die zum mobilen Gerät gesendet werden, nehmen einen Umweg über einen entfernten Heimatagenten. Bei IPv6 werden Routen optimiert, um einem direkten Pfad zwischen dem mobilen Gerät und anderen IP-Adressen zu folgen, nachdem die ersten Pakete über die lange Route gereist sind. Mobiles IPv6 wird in RFC 3775 definiert.

Es gibt noch eine weitere Art von Mobilität, die ebenfalls für das Internet definiert wurde. Einige Flugzeuge haben eingebautes drahtloses Networking, das Passagiere benutzen können, um ihre Laptops mit dem Internet zu verbinden. Das Flugzeug hat einen Router, der an den Rest des Internets über eine drahtlose Verbindung angelassen ist. (Haben Sie eine verdrahtete Verbindung erwartet?) Also, jetzt haben wir einen fliegenden Router, was bedeutet, dass das gesamte Netz mobil ist. Entwürfe zur Netzmobilität unterstützen diesen Fall, ohne dass die Laptops mitbekommen, dass das Flugzeug mobil ist. Soweit es die Laptops betrifft, befinden sie sich lediglich in einem weiteren Netz. Natürlich könnten einige der Laptops mobiles IP verwenden, um ihre Heimatadressen zu behalten, während sie im Flugzeug sind. Somit haben wir zwei Ebenen an Mobilität. Netzmobilität für IPv6 ist in RFC 3963 definiert.

Zusammenfassung

Die **Vermittlungsschicht** bietet der Transportschicht Dienste an. Diese Dienste können auf Datagrammen oder virtuellen Verbindungen basieren. In beiden Fällen ist ihre Hauptaufgabe das Routing von Paketen von der Quelle zum Ziel. Bei Datagrammnetzen wird eine Routing-Entscheidung bei jedem Paket getroffen. In VC-Netzen wird diese Entscheidung beim Aufbau einer virtuellen Verbindung getroffen.

In Rechnernetzen werden viele verschiedene **Routing-Algorithmen** verwendet. Fluten ist ein einfacher Algorithmus, um ein Paket über alle Pfade zu senden. Die meisten Algorithmen finden den kürzesten Pfad und passen diesen an Änderungen in der Netztopologie an. Die wichtigsten Algorithmen sind Distanzvektoralgorithmus und Link-State-Routing. Die meisten neuen Netze benutzen einen dieser Algorithmen. Andere wichtige Routing-Techniken sind die Verwendung einer Hierarchie in großen Netzen, Routing für mobile Hosts sowie Broadcast-, Multicast-Routing und Anycast-Routing.

Netze können leicht überlastet werden, was die Übertragungszeit und die Paketverluste erhöht. Netzentwickler versuchen **Überlastung** zu vermeiden, indem sie das Netz mit ausreichend Kapazität entwerfen und es außerdem befähigen, unüberlastete Routen zu wählen, die Annahme von weiterem Verkehr abzulehnen, an die Quellen Nachrichten zur Verlangsamung zu senden und Last abzuwerfen.

Der nächste Schritt, der über die Behandlung einer Überlastung hinausgeht, ist die versprochene **Dienstgüte** wirklich zu erzielen. Einige Anwendungen bemühen sich mehr um Durchsatz, wohingegen andere eher Übertragungsverzögerung und Jitter wichtig nehmen. Die verwendeten Methoden, um unterschiedliche Grade an Dienstgüte zur Verfügung zu stellen, beinhalten eine Kombination von Traffic-Shaping, das Reservieren von Ressourcen bei Routern und die Zugangssteuerung. Zu den Ansätzen, die zur Sicherstellung der Dienstgüte entwickelt wurden, gehören integrierte Dienste (einschließlich RSVP) und differenzierte Dienste der IETF.

Netze unterscheiden sich anhand verschiedener Faktoren, deshalb können Probleme auftreten, wenn mehrere Netze miteinander verbunden werden. Haben verschiedene Netze unterschiedliche Höchstgrenzen für Paketgrößen, dann könnte eine Fragmentierung der Pakete nötig werden. Unterschiedliche Netze können intern unterschiedliche Routing-Protokolle ausführen, müssen aber extern ein gemeinsames Protokoll haben. Teilweise können diese Probleme durch das Tunneling-Verfahren gelöst werden. Dabei werden Pakete durch ein fremdes Netz geschleust. Wenn aber das Quell- und das Zielnetz verschieden sind, scheitert dieser Ansatz.

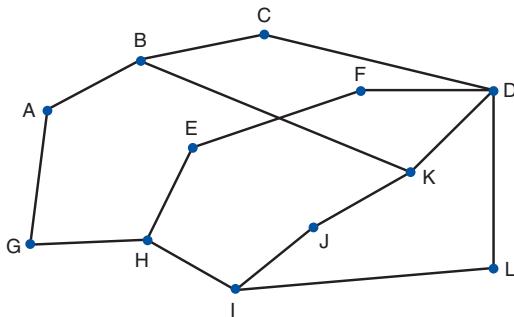
Das Internet verfügt über eine Vielzahl von **Protokollen für die Vermittlungsschicht**. Hierzu gehören beispielsweise das Datagrammprotokoll IP und verbundene Steuerungsprotokolle wie ICMP, ARP und DHCP. Ein verbindungsorientiertes Protokoll namens MPLS trägt IP-Pakete über einige Netze. Eines der wichtigsten Routing-Protokolle, das innerhalb von Netzen eingesetzt wird, ist OSPF, und das Routing-Protokoll, das zwischen Netzen verwendet wird, ist BGP. Dem Internet gehen inzwischen die IP-Adressen aus, sodass eine neue IP-Version, IPv6, entwickelt wurde und nun langsam installiert wird.



Übungsaufgaben

- 1** Geben Sie zwei Beispiele für Computeranwendungen, bei denen ein verbindungsorientierter Dienst angemessen ist. Geben Sie ebenso zwei Beispiele, bei denen ein verbindungsloser Dienst am besten geeignet ist.
- 2** Datagrammnetze leiten jedes Paket als separate Einheit unabhängig von allen anderen weiter. VC-Netze brauchen das nicht, da jedes Paket einem vorab festgelegten Weg folgt. Bedeutet dies, dass VC-Netze nicht die Fähigkeit haben müssen, isolierte Pakete von einer bestimmten Quelle an ein bestimmtes Ziel zu leiten? Erklären Sie Ihre Antwort.
- 3** Nennen Sie drei Beispiele von Protokollparametern, die beim Aufbau einer Verbindung vereinbart werden können.
- 4** Angenommen, alle Router und Hosts funktionieren störungsfrei und die jeweilige Software enthält keine Fehler. Besteht dann eine Möglichkeit, egal wie gering, dass ein Paket zu einem falschen Ziel übertragen wird?
- 5** Geben Sie einen einfachen Algorithmus an, der durch ein Netz zwei Pfade von einer bestimmten Quelle zu einem bestimmten Ziel findet, mit denen der Ausfall irgendeiner Übertragungsleitung überstanden wird (unter der Annahme, dass zwei solche Pfade existieren). Die Router gelten als zuverlässig, sodass man sich über mögliche Routerabstürze keine Gedanken machen muss.
- 6** Betrachten Sie das Netz von Abbildung 5.12a. Hier wird der Distanzvektoralgorithmus angewandt, und folgende Vektoren sind eben in Router *C* eingegangen: von *B* (5, 0, 8, 12, 6, 2), von *D* (16, 12, 6, 0, 9, 10) und von *E* (7, 6, 3, 9, 0, 4). Die Kosten der Verbindungen von *C* zu *B*, *D* und *E* sind 6, 3 bzw. 5. Wie sieht die neue Routing-Tabelle von *C* aus? Führen Sie sowohl die zu benutzende Ausgangsleitung als auch die erwartete Übertragungsverzögerung auf.
- 7** Wie viel Bandbreite pro (Vollduplex-)Leitung wird vom verteilten Routing-Algorithmus verschluckt, wenn Kosten in einem Netz mit 50 Routern als 8-Bit-Zahlen erfasst und Distanzvektoren zweimal pro Sekunde ausgetauscht werden? Gehen Sie davon aus, dass jeder Router drei Leitungen zu anderen Routern hat.
- 8** In Abbildung 5.13 ist das boolesche ODER der beiden Mengen von ACF-Bits in jeder Reihe jeweils 111. Ist dies nur Zufall oder gilt dies für alle Netze unter allen Umständen?
- 9** Welche Regions- und Cluster-Größen sollten bei einem hierarchischen Routing mit 4 800 Routern gewählt werden, um die Größe der Routing-Tabelle bei einer dreistufigen Hierarchie zu minimieren? Die Hypothese, dass eine Lösung mit k Clustern von k Regionen von k Routern fast optimal ist, bildet einen guten Ausgangspunkt. Dies bedeutet, dass k in etwa die Quadratwurzel von 4 800 (um die 16) ist. Finden Sie über Versuch und Irrtum Kombinationen heraus, bei denen alle drei Parameter in der Nähe von 16 liegen.
- 10** Im Text wurde Folgendes behauptet: Wenn sich ein mobiler Host nicht im Heimatbereich befindet, werden Pakete, die an sein Heimat-LAN gesendet werden, von seinem Heimatagenten in diesem LAN abgefangen. Wie führt der Heimatagent dies bei einem IP-Netz auf einem IEEE-802.3-LAN durch?

- 11** Wenn man das Netzwerk von Abbildung 5.6 betrachtet, wie viele Pakete werden bei einem Broadcast von B erzeugt, unter Verwendung von
- Reverse Path Forwarding?
 - einem Quelle-Senke-Baum?
- 12** Betrachten Sie das Netzwerk von Abbildung 5.15a. Stellen Sie sich vor, dass eine neue Linie zwischen F und G eingefügt wird, aber der Quelle-Senke-Baum von Abbildung 5.15b unverändert bleibt. Welche Änderungen ergeben sich bei Abbildung 5.15c?
- 13** Berechnen Sie einen Multicast-Spannbaum für Router C in dem folgenden Netz für eine Gruppe mit Mitgliedern an den Routern A, B, C, D, E, F, J und K .



- 14** Angenommen, Knoten B in ►Abbildung 5.20 wurde gerade neu gebootet und hat in seinen Tabellen noch keine Routing-Informationen vorliegen. Er benötigt plötzlich einen Weg zu H . Er sendet Broadcasts mit einem TTL-Wert von 1, 2, 3 etc. Wie viele Runden benötigt er, um einen Weg zu finden?
- 15** Als ein möglicher Algorithmus zur Überlastungsüberwachung in einem VC-Netz könnte ein Router sich das Bestätigen eines empfangenen Pakets so lange aufsparen, bis er (1) weiß, dass seine letzte Übertragung auf der virtuellen Verbindung erfolgreich empfangen wurde und er (2) Puffer frei hat. Der Einfachheit halber nehmen wir an, dass die Router ein Stop-and-Wait-Protokoll benutzen und dass jede virtuelle Verbindung für jede Verkehrsrichtung einen dedizierten Puffer hat. Mit welcher Rate werden Pakete dem Ziel-Host zugestellt, wenn es T Sekunden dauert, um ein Paket (Daten oder Bestätigung) zu übertragen, und n Router auf dem Pfad vorhanden sind? Gehen Sie davon aus, dass Übertragungsfehler selten sind und die Host-Router-Verbindung unendlich schnell ist.
- 16** In einem Datagrammnetz können Router Pakete wann immer nötig verwerfen. Die Wahrscheinlichkeit, dass ein Router ein Paket verwirft, ist ρ . Betrachten Sie den Fall eines Quell-Hosts, der mit dem Quell-Router verbunden ist, der wiederum mit dem Ziel-Router verbunden ist, der seinerseits mit dem Ziel-Host verbunden ist. Wirft einer der Router ein Paket weg, überträgt der Quell-Host irgendwann nach dem Timeout erneut. Wenn sowohl die Host-Router- als auch die Router-Router-Leitungen als Teilstrecken betrachtet werden, wie lautet der Mittelwert von
- Teilstrecken, die ein Paket pro Übertragung durchläuft?
 - Übertragungen, die ein Paket macht?
 - Teilstrecken, die pro empfangenem Paket erforderlich sind?

- 17** Beschreiben Sie zwei große Unterschiede zwischen der ECN-Methode und der RED-Methode zur Überlastungsvermeidung.
- 18** Ein Token-Bucket-Algorithmus wird zur Verkehrsglättung benutzt. In den Eimer fließt alle 5 µs ein neues Token ein. Jedes Token ist für ein kurzes Paket, das 48 Byte an Daten enthält. Wie hoch ist die maximale durchhaltbare Übertragungsrate?
- 19** Ein Computer in einem 6-Mbit/s-Netz wird über einen Token Bucket reguliert. Der Token Bucket wird mit einer Rate von 1 Mbit/s gefüllt. Er ist anfangs mit 8 Mbit bis zur Kapazitätsgrenze voll. Wie lange kann der Rechner mit vollen 6 Mbit/s senden?
- 20** Das Netz in Abbildung 5.34 verwendet RSVP mit Multicast-Bäumen für Host 1 und Host 2, wie gezeigt. Angenommen, Host 3 hat einen Kanal mit einer Bandbreite von 2 MB/s für einen Datenfluss von Host 1 und einen anderen Kanal mit einer Bandbreite von 1 MB/s für einen Datenfluss von Host 2 angefordert. Gleichzeitig fordert Host 4 einen Kanal mit einer Bandbreite von 2 MB/s für einen Datenfluss von Host 1, und Host 5 fordert einen Kanal mit einer Bandbreite von 1 MB/s für einen Datenfluss von Host 2 an. Wie viel Bandbreite wird insgesamt für die Anforderungen an den Routern *A*, *B*, *C*, *E*, *H*, *J*, *K* und *L* reserviert?
- 21** Ein Router kann 2 Millionen Pakete/s verarbeiten. Die ihm angebotene Last beträgt durchschnittlich 1,5 Millionen Pakete. Falls eine Route von der Quelle zum Ziel 10 Router enthält, wie viel Zeit verbringen Pakete in der Warteschlange des Routers und während der Verarbeitung durch den Router?
- 22** Betrachten Sie einen Benutzer differenzierter Dienste mit Expressweiterleitung. Gibt es eine Garantie, dass Expresspakete eine kürzere Übertragungsverzögerung erfahren als normale Pakete? Warum oder warum nicht?
- 23** Angenommen, Host *A* ist mit einem Router *R1* verbunden, *R1* ist mit einem anderen Router *R2* verbunden und *R2* ist mit Host *B* verbunden. Angenommen, eine TCP-Nachricht mit 900 Byte Daten und 20 Byte TCP-Header wird an den IP-Code auf Host *A* zur Übertragung an *B* übergeben. Stellen Sie die IP-Header-Felder *Total Length*, *Identification*, *DF*, *MF* und *Fragment Offset* in jedem Paket dar, das über die drei Verbindungen übertragen wird. Nehmen Sie an, dass die Verbindung *A-R1* eine maximale Rahmengröße von 1 024 Byte einschließlich eines 14-Byte-Rahmen-Headers unterstützen kann, die Verbindung *R1-R2* kann eine maximale Rahmengröße von 512 Byte mit einem 8-Byte-Rahmen-Header unterstützen, und *R2-B* kann eine maximale Rahmengröße von 512 Byte mit einem 12-Byte-Rahmen-Header unterstützen.
- 24** Ein Router sendet mit voller Kraft IP-Pakete, deren Gesamtlänge (Daten plus Header) 1 024 Byte betragen. Angenommen, dass die Pakete 10 s leben; wie hoch ist dann die maximale Leitungsgeschwindigkeit, mit der der Router arbeiten kann, ohne dass die Gefahr besteht, dass der Sequenznummernraum der IP-Datenpakete wiederholt durchlaufen wird?
- 25** Ein IP-Datagramm mit der Option *Strict Source Routing* muss fragmentiert werden. Glauben Sie, dass die Option in jedes Fragment kopiert werden muss, oder genügt es, sie einfach in das erste Fragment einzufügen? Erklären Sie Ihre Antwort.
- 26** Angenommen, es werden anstelle von 16 Bit für den Netzteil in einer Adresse der Klasse B 20 Bit verwendet. Wie viele Klasse-B-Netze würde es dann geben?

- 27** Konvertieren Sie die IP-Adresse mit der hexadezimalen Darstellung C22F1582 in eine Dezimalpunktdarstellung.
- 28** Ein Netz im Internet hat die Netzmaske 255.255.240.0. Wie viele Hosts können hier maximal unterstützt werden?
- 29** IP-Adressen sind an spezifische Netzen gebunden, Ethernet-Adressen dagegen nicht. Können Sie sich einen guten Grund dafür vorstellen, warum dies so ist?
- 30** Es ist eine große Anzahl aufeinanderfolgender IP-Adressen verfügbar, die bei 198.16.0.0 beginnen. Angenommen, vier Organisationen, A, B, C und D, fordern 4 000, 2 000, 4 000 und 8 000 Adressen in dieser Reihenfolge an. Geben Sie für jede die erste und die letzte zugewiesene IP-Adresse sowie die Maske in der Notation w.x.y.z/s an.
- 31** Ein Router hat gerade die folgenden neuen IP-Adressen erhalten: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 und 57.6.120.0/21. Wenn alle die gleiche Ausgangsleitung verwenden, können sie zusammengefasst werden? Wenn ja, wozu? Wenn nicht, warum nicht?
- 32** Die Gruppe der IP-Adressen von 29.18.0.0 bis 29.18.128.255 wurde zu 29.18.0.0/17 zusammengefasst. Es besteht aber eine Lücke von 1 024 nicht zugewiesenen Adressen von 29.18.60.0 bis 29.18.63.255, die nun plötzlich an einen Host an einer anderen Ausgangsleitung zugewiesen werden. Muss nun die aggregierte Adresse in einzelne Blöcke zerlegt, der neue Block in die Tabelle aufgenommen und dann geprüft werden, ob eine erneute Zusammenfassung möglich ist? Wenn nicht, was kann man stattdessen tun?
- 33** Ein Router hat in seiner Routing-Tabelle die folgenden (CIDR-)Einträge stehen:

| Adresse/Maske | Nächste Teilstrecke |
|----------------|---------------------|
| 135.46.56.0/22 | Schnittstelle 0 |
| 135.46.60.0/22 | Schnittstelle 1 |
| 192.53.40.0/23 | Router 1 |
| Standard | Router 2 |

Was macht der Router, wenn ein Paket mit einer der untenstehenden Adressen ankommt?
Beschreiben Sie das Verhalten des Routers für jede der folgenden Adressen:

- 135.46.63.10
- 135.46.57.14
- 135.46.52.2
- 192.53.40.7
- 192.53.56.7

- 34** Viele Unternehmen haben zwei (oder mehr) Router für den Zugang zum Internet im Einsatz, sodass sie eine gewisse Sicherheit haben, falls einer davon ausfällt. Ist dies mit NAT immer noch möglich? Erklären Sie Ihre Antwort.
- 35** Sie haben das ARP-Protokoll gerade einem Freund erklärt. Als Sie fertig sind, sagt er: „Habe verstanden, ARP stellt für die Vermittlungsschicht einen Dienst zur Verfügung und ist deshalb Teil der Sicherungsschicht.“ Was sagen Sie ihm?

- 36** Beschreiben Sie eine Möglichkeit, IP-Fragmente am Ziel wieder zusammenzusetzen.
- 37** Die meisten Algorithmen zum Zusammensetzen von IP-Datagrammen haben einen Timer, um zu vermeiden, dass sich verlorene Fragmente für ewig in Puffern ansammeln. Nehmen Sie an, dass ein Datagramm in vier Fragmente aufgeteilt wird. Die ersten drei Fragmente kommen an, das letzte verzögert sich. Schließlich läuft der Timer ab, und die drei Fragmente im Speicher des Empfängers werden verworfen. Etwas später kommt das letzte Fragment an. Was soll mit ihm geschehen?
- 38** Bei IP deckt die Prüfsumme nur den Header, nicht aber die Daten ab. Warum wurde Ihrer Meinung nach dieser Entwurf gewählt?
- 39** Ein Frau, die in Boston lebt, reist nach Minneapolis und nimmt ihr Notebook mit. Zu ihrer Überraschung ist das LAN an ihrem Zielort in Minneapolis ein drahtloses IP-LAN, sodass sie keine Stecker einstecken muss. Müssen hier dann immer noch Heimatagenten und Fremdagenten eingesetzt werden, damit E-Mail und anderer Datenverkehr korrekt ankommt?
- 40** IPv6 verwendet 16-Byte-Adressen. Wenn ein Block mit 1 Million Adressen jede Pikosekunde zugewiesen wird, wie lange reichen die Adressen?
- 41** Das Feld *Protocol* im IPv4-Header ist im festen IPv6-Header nicht vorhanden. Warum nicht?
- 42** Muss das ARP-Protokoll geändert werden, wenn man das IPv6-Protokoll einführt? Wenn dies der Fall ist, sind die Änderungen dann konzeptioneller oder technischer Art?
- 43** Schreiben Sie ein Programm, um Routing mittels Fluten zu simulieren. Jedes Paket sollte einen Zähler enthalten, der auf jeder Teilstrecke um 1 reduziert wird. Erreicht der Zähler null, wird das Paket verworfen. Die Zeit ist diskret, wobei jede Leitung ein Paket pro Zeitintervall abwickelt. Erstellen Sie von dem Programm drei Versionen: eine mit Fluten aller Leitungen, eine mit Fluten aller Leitungen außer der Eingangsleitung und eine mit Fluten der (statisch gewählten) besten k Leitungen. Vergleichen Sie Fluten mit dem deterministischen Routing ($k=1$) in Bezug auf die Übertragungsverzögerung und die verwendete Bandbreite.
- 44** Schreiben Sie ein Programm zur Simulation eines Rechnernetzes mit diskreter Zeit. Das erste Paket in jeder Router-Warteschlange legt eine Teilstrecke pro Zeitintervall zurück. Jeder Router hat nur eine endliche Zahl von Puffern. Kommt ein Paket an und ist kein Platz dafür vorhanden, wird es verworfen und nicht erneut übertragen. Stattdessen gibt es ein Ende-zu-Ende-Protokoll, komplett mit Timeout- und Bestätigungspaketen, das letztlich das Paket vom Quell-Router regeneriert. Ermitteln Sie den Durchsatz des Netzes als Funktion des Timeout-Intervalls von Ende zu Ende mit der Fehlerrate als Parameter.
- 45** Schreiben Sie eine Funktion, um in einem IP-Router Pakete weiterzuleiten. Die Prozedur hat einen Parameter, eine IP-Adresse. Sie hat auch Zugriff auf eine globale Tabelle, die aus einem Array von Tripeln besteht. Jedes Tripel enthält drei Integerwerte: eine IP-Adresse, eine Netzmaske und die zu verwendende Ausgangsleitung. Die Funktion schlägt die IP-Adresse in der Tabelle unter Verwendung von CIDR nach und die Leitung aus, die als dessen Wert verwendet werden soll.

- 46** Verwenden Sie das Programm *traceroute* (Unix-) bzw. *tracert* (Windows), um den Weg von Ihrem Computer zu verschiedenen Universitäten auf anderen Kontinenten zu verfolgen. Erstellen Sie eine Liste von Überseelinks, die Sie entdeckt haben. Sites, die Sie kontaktieren können, sind beispielsweise

www.berkeley.edu (Kalifornien)

www.mit.edu (Massachusetts)

www.vu.nl (Amsterdam)

www.ucl.ac.uk (London)

www.usyd.edu.au (Sydney)

www.u-tokyo.ac.jp (Tokio)

www.uct.ac.za (Kapstadt)

Die Transportschicht

| | |
|---|-----|
| 6.1 Dienste der Transportschicht | 567 |
| 6.2 Elemente von Transportprotokollen | 580 |
| 6.3 Überlastungsüberwachung | 604 |
| 6.4 Internettransportprotokolle: UDP | 616 |
| 6.5 Internettransportprotokolle: TCP | 628 |
| 6.6 Leistungsaspekte | 660 |
| 6.7 Verzögerungstolerante Netze | 679 |

» Zusammen mit der Vermittlungsschicht bildet die Transportschicht den Kern der Protokollhierarchie. Die Vermittlungsschicht bietet Ende-zu-Ende-Paketübertragung mittels Datagrammen und virtuellen Verbindungen. Die Transportschicht verlässt sich darauf, dass die Vermittlungsschicht die Daten zuverlässig und unabhängig von den physikalischen Netzen von einem Prozess auf einem Quellrechner zu einem Prozess auf einem Zielrechner überträgt. Sie bietet die Abstraktionen, die Anwendungen benötigen, um das Netz zu nutzen. Ohne die Transportschicht wäre das gesamte Konzept der Schichtprotokolle wenig sinnvoll. In diesem Kapitel wird die Transportschicht ausführlich behandelt, einschließlich ihrer Dienste und der Wahl des API-Designs, um auf Punkte wie Zuverlässigkeit, Verbindungen und Überlastungsüberwachung, Protokolle wie TCP und UDP und Leistungsmerkmale einzugehen.



6.1 Dienste der Transportschicht

In den folgenden Abschnitten werden Sie in die Dienste der Transportschicht eingeführt. Wir betrachten, welcher Dienst der Anwendungsschicht zur Verfügung gestellt wird. Hierzu untersuchen wir zwei Gruppen von Transportschichtprimitiven. Als Erstes behandeln wir eine einfache (aber hypothetische) Gruppe, um die Grundideen zu veranschaulichen. Dann wenden wir uns den Schnittstellen zu, die in der Regel im Internet verwendet werden.

6.1.1 Dienste für die oberen Schichten

Die Aufgabe der Transportschicht ist es, ihren Benutzern einen gut funktionierenden, zuverlässigen und kosteneffizienten Datenübertragungsdienst bereitzustellen. Diese Benutzer sind normalerweise Prozesse der Anwendungsschicht. Um dieses Ziel zu erreichen, macht die Transportschicht Gebrauch von den Diensten der Vermittlungsschicht. Die Hardware und/oder Software in der Transportschicht, die diese Aufgabe übernimmt, heißt **Transportinstanz** (*transport entity*). Die Transportinstanz kann sich im Betriebssystemkern, in einem eigenen Benutzerprozess, in einem von Netzanwendungen eingebundenen Bibliothekspaket oder auch auf einer Netzwerkkarte befinden. Die ersten beiden Optionen sind im Internet am geläufigsten. Die (logische) Beziehung der Vermittlungs-, Transport- und Anwendungsschicht wird in ▶ Abbildung 6.1 aufgezeigt.

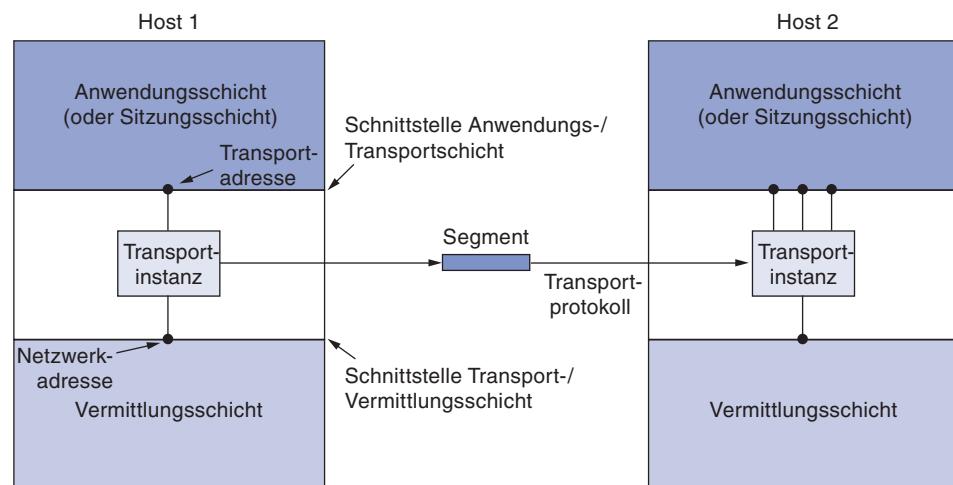


Abbildung 6.1: Beziehung der Vermittlungs-, der Transport- und der Anwendungsschicht.

Genau so wie es zwei Arten von Vermittlungsschichtdiensten gibt, so gibt es auch zwei Arten von Transportschichtdiensten: den verbindungsorientierten und den verbindungslosen Dienst. Der verbindungsorientierte Dienst der Transportschicht ist dem verbindungsorientierten Dienst der Vermittlungsschicht in vielen Punkten ähnlich. Bei beiden Diensten bestehen die Verbindungen aus drei Phasen: Verbindungsaufbau, Datenübertragung und Verbindungsfreigabe. Adressierung und Flusskontrolle beider

Schichten sind einander ebenfalls ähnlich. Darüber hinaus weist auch der verbindungslose Dienst zwischen Transportschicht und Vermittlungsschicht eine große Ähnlichkeit auf. Es ist jedoch zu beachten, dass es schwer sein kann, aufgesetzt auf einen verbindungsorientierten Vermittlungsdienst einen verbindungslosen Transportdienst anzubieten, da es nicht sehr effizient ist, eine Verbindung aufzubauen, um ein einzelnes Paket zu senden, und sie danach direkt wieder zu schließen.

Die naheliegende Frage ist nun: Wenn der Transportschichtdienst dem Vermittlungsschichtdienst so ähnlich ist, warum gibt es dann zwei verschiedene Schichten? Warum reicht nicht eine Schicht aus? Die Antwort ist nicht ganz einfach, aber wichtig. Der Transportcode wird vollständig auf dem Rechner des Benutzers ausgeführt, aber die Vermittlungsschicht läuft in der Regel auf Routern, die von einem Betreiber bereitgestellt werden (zumindest für Fernnetze). Was geschieht, wenn die Vermittlungsschicht keinen adäquaten Dienst zur Verfügung stellt? Was passiert, wenn häufig Pakete verloren gehen? Was passiert, wenn ab und zu Router ausfallen?

Ganz einfach: Es kommt zu Problemen. Die Benutzer haben keinerlei Kontrolle über die Vermittlungsschicht; sie können also das Problem eines schlechten Dienstes nicht durch Verwendung besserer Router oder Einbindung einer besseren Fehlerbehandlung in der Sicherungsschicht lösen, da sie die Router nicht besitzen. Die einzige Möglichkeit besteht darin, auf die Vermittlungsschicht eine weitere Schicht aufzusetzen, um die Qualität des Dienstes zu verbessern. Wenn in einem verbindungslosen Netz Pakete verloren gehen oder verstümmelt werden, kann die Transportinstanz das Problem nach dem Entdecken durch erneutes Übertragen ausgleichen. Wenn in einem verbindungsorientierten Teilnetz eine Transportinstanz mitten in einer langen Übertragung informiert wird, dass die Verbindung abrupt abgebrochen wurde, ohne Hinweis auf den Verbleib der gerade gesendeten Daten, kann sie eine neue Netzverbindung zur entfernten Transportinstanz aufbauen. Über diese Netzverbindung kann sie bei der empfangenden Einheit anfragen, welche Daten angekommen sind und welche nicht, um dort weiterzumachen, wo die Übertragung unterbrochen wurde.

Im Grunde sorgt die Existenz der Transportschicht dafür, dass der Dienst der Transportschicht zuverlässiger ist als das zugrunde liegende Netzwerk. Darüber hinaus können die Transportschichtprimitiven als Aufrufe von Bibliotheksprozeduren implementiert werden, um sie von den Vermittlungsschichtprimitiven unabhängig zu machen. Die Aufrufe der Vermittlungsschichtdienste können von Netz zu Netz sehr unterschiedlich sein, z.B. können Aufrufe auf der Basis eines verbindungslosen Ethernet völlig anders sein als Aufrufe auf einem verbindungsorientierten WiMAX-Netz. Indem der Dienst der Vermittlungsschicht hinter einer Gruppe von Transportschichtdienstprimitiven verborgen wird, muss bei einer Änderung des Netzes nur eine Gruppe von Bibliotheksprozeduren durch eine andere ersetzt werden, die das Gleiche macht mit einem anderen zugrunde liegenden Dienst.

Der Transportschicht ist es auch zu verdanken, dass Anwendungsprogrammierer unter Verwendung einer Standardmenge von Dienstprimitiven Code schreiben und diese Programme in vielen verschiedenen Netzen laufen lassen können, ohne sich dabei Gedanken über die verschiedenen Netzschichtstellen und Zuverlässigkeit-

stufen machen zu müssen. Wenn alle existierenden Netze fehlerlos funktionieren würden, die gleichen Dienstprimitive hätten und sich garantiert niemals änderten, wäre die Transportschicht vielleicht überflüssig. Im wirklichen Leben hat sie aber die Schlüsselfunktion, die oberen Schichten von den technischen Details, dem Aufbau und den Unzulänglichkeiten des zugrunde liegenden Netzes abzuschirmen.

Aus diesem Grund unterscheiden viele Fachleute die Schichten 1 bis 4 einerseits und die Schicht(en) darüber andererseits. Die unteren vier Schichten werden als **Transportschichtdienst-Provider** betrachtet, während die oberen Schichten als **Transportschicht-dienst-Benutzer** bezeichnet werden. Diese Unterscheidung zwischen Provider und Benutzer hat beträchtliche Auswirkungen auf den Aufbau der Schichten. Die Transportschicht nimmt eine Schlüsselposition ein, weil sie eine wichtige Grenze zwischen Provider und Benutzer des zuverlässigen Datenübertragungsdienstes darstellt. Es handelt sich dabei um die Schicht, die von den Anwendungen gesehen wird.

6.1.2 Dienstprimitive der Transportschicht

Um den Benutzern den Zugriff auf die Dienste der Transportschicht zu gewähren, muss die Transportschicht den Anwendungen einige Operationen zur Verfügung stellen, und zwar über Transportschichtdienst-Schnittstellen. Jeder Dienst der Transportschicht hat seine eigene Schnittstelle. In diesem Abschnitt werden zuerst ein einfacher (hypothetischer) Dienst der Transportschicht und seine Schnittstelle beschrieben, um die elementaren Grundlagen kennenzulernen, und im darauffolgenden Abschnitt betrachten wir ein Beispiel aus der Praxis.

Trotz der Ähnlichkeit zwischen dem Transport- und dem Vermittlungsschichtdienst gibt es einige wichtige Unterschiede. Der größte Unterschied ist der, dass der Dienst der Vermittlungsschicht dazu dient, den von echten Netzen gebotenen Dienst zu modellieren, und zwar mit allen Fehlern und Mängeln. Echte Netze können Pakete verlieren; deshalb ist der Dienst der Vermittlungsschicht im Allgemeinen unzuverlässig.

Demgegenüber ist der verbindungsorientierte Dienst der Transportschicht zuverlässig. Selbstverständlich sind echte Netze nicht fehlerfrei, aber das ist genau der Punkt, an dem die Transportschicht greift – Bereitstellung eines zuverlässigen Dienstes auf einem unzuverlässigen Netz.

Als Beispiel betrachten wir zwei Prozesse auf einem Rechner, die durch eine Unix-Pipe (oder irgendeinen anderen Interprozesskommunikationsweg) verbunden sind. Die Prozesse gehen davon aus, dass die Verbindung zwischen ihnen perfekt ist. Sie wollen nichts von Bestätigungen, verlorenen Paketen, Überlastung und ähnlichen Dingen wissen. Was sie wollen, ist eine 100-prozentig zuverlässige Verbindung. Prozess A speist Daten an einem Ende der Pipe ein und Prozess B entnimmt sie am anderen Ende. Das ist das Grundprinzip eines verbindungsorientierten Transportschichtdienstes: Die Mängel des Vermittlungsschichtdienstes bleiben verborgen, sodass die Benutzerprozesse von einem fehlerfreien Bitstrom ausgehen können, selbst wenn sie auf verschiedenen Rechnern liegen.

Nebenbei bemerkt, kann die Transportschicht auch einen unzuverlässigen Datagrammdienst zur Verfügung stellen. Hierzu gibt es allerdings nur wenig zu sagen, außer dass es sich um „Datagramme“ handelt, sodass wir uns in diesem Kapitel vor allem auf den verbindungsorientierten Dienst der Transportschicht konzentrieren. Dennoch gibt es einige Anwendungen, wie Client-Server-Computing und das Streamen von Multimediadaten, die von einem verbindungslosen Transport profitieren. Hierzu später mehr.

Ein weiterer Unterschied zwischen dem Vermittlungs- und dem Transportschichtdienst ist, für wen die Dienste bereitgestellt werden. Der Dienst der Vermittlungsschicht wird nur von den Transportinstanzen benutzt. Und da nur wenige Benutzer ihre eigenen Transportinstanzen schreiben, sehen auch nur wenige Benutzer oder Programme je den reinen Dienst der Vermittlungsschicht. Im Gegensatz dazu gibt es viele Programme (und damit Programmierer), die die Transportprimitiven sehen. Folglich muss der Dienst der Transportschicht benutzerfreundlich und einfach sein.

Um eine Vorstellung davon zu erhalten, wie ein Dienst der Transportschicht aussehen könnte, betrachten Sie die fünf Primitiven, die in ►Abbildung 6.2 aufgeführt sind. Diese Transportschnittstelle ist wirklich sehr einfach, enthält aber die grundlegenden Funktionen einer verbindungsorientierten Transportschnittstelle. Anwendungsprogramme können Verbindungen aufbauen, verwenden und freigeben. Dies ist für viele Anwendungen völlig ausreichend.

| Primitive | Gesendetes Paket | Bedeutung |
|------------|-----------------------|---|
| LISTEN | (Keines) | Blockiert, bis ein Prozess eine Verbindung aufbauen möchte. |
| CONNECT | CONNCECTION REQUEST | Versucht aktiv, eine Verbindung aufzubauen. |
| SEND | DATA | Sendet Informationen. |
| RECEIVE | (Keines) | Blockiert, bis ein DATA-Paket ankommt. |
| DISCONNECT | DISCONNECTION REQUEST | Fordert die Freigabe der Verbindung. |

Abbildung 6.2: Die Primitiven für einen einfachen Dienst der Transportschicht.

Um zu sehen, wie diese Primitiven benutzt werden, betrachten wir eine Anwendung mit einem Server und mehreren entfernten Clients. Der Server führt eine LISTEN-Primitive normalerweise durch Aufruf einer Bibliotheksprozedur aus, durch die ein Systemaufruf erstellt wird, der den Server so lange blockiert, bis ein Client auftaucht. Möchte ein Client mit dem Server kommunizieren, führt er eine CONNECT-Primitive aus. Zur Ausführung dieser Primitive blockiert die Transportinstanz den Aufrufer und sendet ein Paket an den Server. In den Nutzdaten dieses Pakets ist eine Nachricht der Transportschicht für die Transportinstanz des Servers gekapselt.

Hier eine kurze Anmerkung zur Terminologie: Mangels besserer Begriffe bezeichnen wir Nachrichten, die von Transportinstanz zu Transportinstanz gesendet werden, als **Segment**. Dieser Begriff wird von TCP, UDP und anderen Internetprotokollen verwendet.

Einige ältere Protokolle verwenden die eher unglückliche Abkürzung **TPDU** (*Transport Protocol Data Unit*, Dateneinheit des Transportprotokolls). Diese Bezeichnung wird heutzutage nicht mehr häufig verwendet, außer in älteren Aufsätzen und Büchern.

Um genau zu sein, sind Segmente (von der Transportschicht ausgetauscht) in Paketen (von der Vermittlungsschicht ausgetauscht) enthalten und Pakete wiederum in Rahmen (von der Sicherungsschicht ausgetauscht). Kommt ein Rahmen an, verarbeitet die Sicherungsschicht den Rahmen-Header und gibt den Inhalt des Rahmennutzdatenfeldes an die Vermittlungsinstanz weiter, wenn die Zieladresse für die lokale Zustellung stimmt. Die Vermittlungsinstanz verarbeitet den Paket-Header und gibt den Inhalt des Paketnutzdatenfeldes an die Transportinstanz weiter. Diese Verschachtelung ist in ▶ Abbildung 6.3 dargestellt.

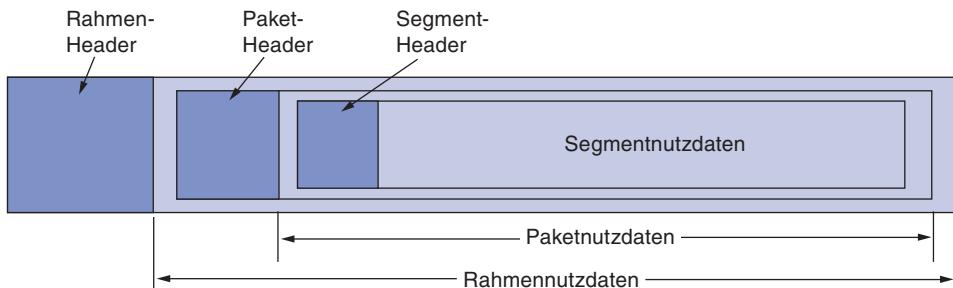


Abbildung 6.3: Verschachtelung von Segmenten, Paketen und Rahmen.

Zurück zu unserem Client-Server-Beispiel. Der CONNECT-Aufruf des Clients veranlasst das Senden eines CONNECTION REQUEST-Segments an den Server. Wenn das Segment ankommt, wird von der Transportinstanz geprüft, ob der Server durch Aufruf einer LISTEN-Primitive blockiert ist (d.h., der Server ist bereit, eingehende Anforderungen zu verarbeiten). In diesem Fall beendet sie das Blockieren des Servers und sendet ein CONNECTION ACCEPTED-Segment an den Client zurück. Kommt dieses Segment an, wird das Blockieren des Clients aufgehoben und die Verbindung wird aufgebaut.

Mit den Primitiven SEND und RECEIVE können jetzt Daten ausgetauscht werden. In der einfachsten Form kann eine Partei ein (blockierendes) RECEIVE ausführen und warten, bis von der anderen Partei SEND kommt. Kommt das Segment an, wird das Blockieren des Empfängers aufgehoben. Er kann das Segment verarbeiten und eine Antwort senden. Solange beide Seiten überblicken, wer mit dem Senden an der Reihe ist, funktioniert dieser Ablauf problemlos.

Man beachte, dass in der Transportschicht sogar ein einfacher unidirektonaler Datenaustausch komplizierter ist als in der Vermittlungsschicht. Jedes gesendete Datenpaket wird (irgendwann) auch bestätigt. Pakete, die Steuersegmente enthalten, werden implizit oder explizit ebenfalls bestätigt. Diese Bestätigungen werden von den Transportinstanzen über das Protokoll der Vermittlungsschicht verwaltet und sind für die Benutzer auf der Transportschicht nicht erkennbar. Außerdem müssen sich Transportinstanzen um Timer und erneute Übertragungen kümmern. Die Benutzer der Transportschicht bekommen diese Mechanismen nicht mit. Für sie ist eine Verbindung eine zuverlässige Bitpipe: Ein

Benutzer stopft Bits hinein und am anderen Ende kommen sie wie von Zauberhand wieder in gleicher Reihenfolge heraus. Diese Fähigkeit, die Komplexität zu verbergen, ist der Grund, warum Schichtprotokolle ein derart leistungsstarkes Werkzeug sind.

Wird eine Verbindung nicht mehr gebraucht, muss sie freigegeben werden, um in beiden Transportinstanzen Tabellenplatz freizumachen. Für die Verbindungs freigabe gibt es zwei Varianten: asymmetrisch und symmetrisch. Bei der asymmetrischen Variante kann jeder Benutzer der Transportschicht eine DISCONNECT-Primitive aufrufen, was dazu führt, dass ein DISCONNECT-Segment an die entfernte Transportinstanz gesendet wird. Bei deren Ankunft wird die Verbindung freigegeben.

Bei der symmetrischen Variante wird jede Richtung unabhängig von der anderen abgebaut. Führt eine Seite ein DISCONNECT aus, bedeutet das, dass keine weiteren Daten mehr zu übertragen sind, dass aber Daten vom Partner noch angenommen werden. Bei diesem Modell wird eine Verbindung erst getrennt, wenn beide Seiten ein DISCONNECT ausführen.

Ein Zustandsdiagramm für den Verbindungs aufbau und die Verbindungs freigabe für diese einfachen Primitiven ist in ► Abbildung 6.4 dargestellt. Jeder Übergang wird durch ein Ereignis ausgelöst, entweder eine vom lokalen Benutzer ausgeführte Primitive oder einem eingehenden Paket. Der Einfachheit halber nehmen wir hier an, dass jedes Segment separat bestätigt wird. Wir nehmen auch an, dass ein symmetrisches Freigabe modell verwendet wird und der Client zuerst abbauen will. Dieses Modell ist sehr einfach. Später werden wir im Zusammenhang mit TCP realistischere Modelle betrachten.

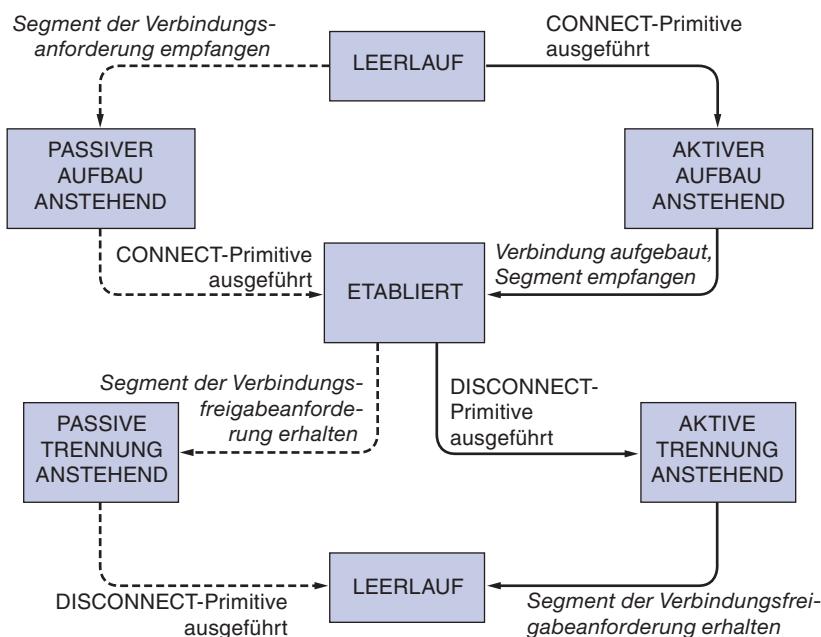


Abbildung 6.4: Zustandsdiagramm einer einfachen Verbindungsverwaltung. Die kursiv gedruckten Übergänge werden von ankommenden Paketen ausgelöst. Die durchgezogenen Linien zeigen die Zustandsfolge des Clients. Die gestrichelten Linien zeigen die Zustandsfolge des Servers.

6.1.3 Berkeley-Sockets

In diesem Abschnitt wird kurz eine andere Gruppe von Transportprimitiven betrachtet, die Socket-Primitiven, wie sie für TCP verwendet werden. Sockets wurden das erste Mal 1983 als Teil der Berkeley-Unix-4.2BSD-Software eingeführt. Sie fanden schnell große Verbreitung. Die Primitiven werden heute häufig in der Internetprogrammierung auf vielen Betriebssystemen verwendet, besonders auf Unix-basierten Systemen; außerdem gibt es eine Socket-API für Windows namens „Winsock“.

Die Primitiven sind in ► Abbildung 6.5 aufgeführt. Ganz allgemein gesprochen folgen sie dem Modell unseres ersten Beispiels, bieten aber mehr Funktionen und eine höhere Flexibilität. Wir betrachten hier nicht die entsprechenden Segmente. Diese Diskussion folgt später.

| Primitive | Bedeutung |
|-----------|--|
| SOCKET | Erzeugt einen neuen Kommunikationsendpunkt. |
| BIND | Verknüpft eine lokale Adresse mit dem Socket. |
| LISTEN | Kündigt die Bereitschaft zur Annahme von Verbindungen an; gibt die Größe der Warteschlange an. |
| ACCEPT | Baut passiv eine eingehende Verbindung auf. |
| CONNECT | Versucht aktiv, eine Verbindung aufzubauen. |
| SEND | Sendet Daten über die Verbindung. |
| RECEIVE | Empfängt Daten über die Verbindung. |
| CLOSE | Gibt die Verbindung frei. |

Abbildung 6.5: Die Socket-Primitiven für TCP.

Die ersten vier Primitiven in der Liste werden von Servern in der aufgeführten Reihenfolge ausgeführt. Die SOCKET-Primitive erzeugt einen neuen Endpunkt und weist in der Transportinstanz dafür entsprechend Tabellenplatz zu. Die Aufrufparameter geben das zu benutzende Adressierformat, die gewünschte Dienststart (z.B. zuverlässigen Bytestrom) und das Protokoll an. Ein erfolgreicher SOCKET-Aufruf liefert einen normalen Dateideskriptor für nachfolgende Aufrufe zurück, genau wie ein OPEN-Aufruf für eine Datei.

Neu erzeugte Sockets haben keine Netzwerkadressen. Diese werden mit BIND zugewiesen. Hat ein Server eine Adresse mit dem Socket „verbunden“, können sich entfernte Clients hier anmelden. Der Grund, warum der SOCKET-Aufruf nicht direkt Adressen erzeugen soll, ist, dass manche Prozesse sich für ihre Adressen interessieren (z.B. wenn sie über Jahre die gleiche Adresse benutzen und alle diese Adresse kennen) und andere Prozesse nicht.

Der LISTEN-Aufruf weist eingehenden Verbindungsanfragen Platz in Warteschlangen zu für den Fall, dass mehrere Clients gleichzeitig eine Verbindung anfordern. Im Gegensatz zu LISTEN in unserem ersten Beispiel ist LISTEN im Socket-Modell kein blockierender Aufruf.

Um sich beim Warten auf eine eingehende Verbindung zu blockieren, führt der Server die ACCEPT-Primitive aus. Kommt ein Segment an und fragt nach einer Verbindung, erzeugt die Transportinstanz einen neuen Socket mit den gleichen Eigenschaften wie der ursprüngliche und gibt dafür einen Dateideskriptor zurück. Der Server kann dann in einem neu erzeugten Prozess oder Thread die Verbindung auf dem neuen Socket verarbeiten, und auf dem ursprünglichen Socket auf die nächste Verbindung warten. ACCEPT gibt einen normalen Dateideskriptor zurück, der wie bei Dateien für normale Lese- und Schreibvorgänge verwendet werden kann.

Betrachten wir nun die Clientseite. Clientseitig muss ebenfalls zuerst mit SOCKET ein Socket erzeugt werden. BIND ist nicht erforderlich, da die benutzte Adresse für den Server keine Rolle spielt. Die CONNECT-Primitive blockiert den Aufrufer und startet den Verbindungsprozess. Ist dies abgeschlossen (d.h. geht das entsprechende Segment vom Server ein), wird die Sperre beim Client-Prozess gelöst und die Verbindung ist aufgebaut. Beide Seiten können nun über die Duplexverbindung Daten mit SEND und RECEIVE senden und empfangen. Wenn keine speziellen SEND- und RECEIVE-Optionen erforderlich sind, können auch die Unix-Standardsystemaufrufe READ und WRITE verwendet werden.

Die Verbindungsfreigabe verläuft symmetrisch. Haben beide Seiten CLOSE ausgeführt, wird die Verbindung freigegeben.

Sockets erfreuen sich inzwischen großer Beliebtheit und sind der De-facto-Standard, um Anwendungen Transportschichtdienste auf höherer Ebene abstrahiert zur Verfügung zu stellen. Die Socket-API wird oft zusammen mit dem TCP-Protokoll verwendet, um einen verbindungsorientierten Dienst in Form eines **zuverlässigen Bytestroms** zu bieten. Hierbei handelt es sich einfach um die bereits beschriebene zuverlässige Bitpipe. Es können jedoch auch andere Protokolle verwendet werden, um diesen Dienst mit der gleichen API zu implementieren. Die Benutzer der Transportschichtdienste sollten keinen Unterschied bemerken.

Eine Stärke der Socket-API ist, dass sie von einer Anwendung auch für andere Transportschichtdienste verwendet werden kann. Zum Beispiel können Sockets mit einem verbindungslosen Transportschichtdienst verwendet werden. In diesem Fall setzt CONNECT die Adresse des entfernten Transport-Peers und SEND und RECEIVE senden bzw. empfangen Datagramme vom und zum entfernten Peer. (Es ist durchaus üblich, einen erweiterten Satz von Aufrufen zu verwenden, z.B. SENDTO und RECEIVEFROM, die die Nachrichten hervorheben und eine Anwendung nicht auf einen einzigen Transport-Peer beschränken.) Sockets können außerdem mit Transportprotokollen verwendet werden, die über einen Nachrichtenstrom anstelle eines Bytestroms kommunizieren und über eine Überlastungsüberwachung verfügen oder nicht. Beispielsweise ist **DCCP** (*Datagram Congestion Controlled Protocol*) eine Version eines UDP-Protokolls mit Überlas-

tungsüberwachung (Kohler et al., 2006). Es ist Aufgabe der Benutzer der Transportschicht, sich über den Dienst zu informieren, den sie erhalten.

Mit Sockets ist jedoch wahrscheinlich noch nicht das letzte Wort zu den Transportschnittstellen gesprochen. Zum Beispiel arbeiten Anwendungen oft mit einer Gruppe verwandter Ströme – wie ein Webbrower, der mehrere Objekte vom gleichen Server anfordert. Im Falle von Sockets verwenden Anwendungsprogramme idealerweise einen Strom pro Objekt. Bei einer solchen Struktur erfolgt die Überlastungsüberwachung für jeden Strom separat und nicht für eine ganze Gruppe, was suboptimal ist. In diesem Fall muss die Anwendung die Aufgabe übernehmen, diese zu verwalten. Neuere Protokolle und Schnittstellen sind entwickelt worden, die Gruppen verwandter Ströme effektiver und ausschließlich für die Anwendung unterstützen. Zwei Beispiele sind **SCTP** (*Stream Control Transmission Protocol*), das in RFC 4960 definiert ist, und **SST** (*Structured Stream Transport*; Ford, 2007). Diese Protokolle müssen die Socket-API etwas verändern, um Gruppen verwandter Ströme optimal zu nutzen. Außerdem unterstützen sie neben mehreren Netzpfaden auch Funktionen wie die Kombination von verbindungsorientiertem und verbindungslosem Verkehr. Die Zeit wird zeigen, ob sie erfolgreich sind.

6.1.4 Beispiel für Socket-Programmierung: Ein Internetdateiserver

Als Beispiel für die wesentlichen Schritte eines erfolgreichen Socket-Aufrufs betrachten wir den Client- und Server-Code in ►Abbildung 6.6. Es ist ein ganz einfacher Internetdateiserver mit einem Beispiel-Client, der darauf zugreift. Der Code weist viele Einschränkungen auf, die nachfolgend diskutiert werden. Der Server-Code kann aber kompiliert und auf einem beliebigen Unix-System mit Internetzugang ausgeführt werden. Der Client-Code kann kompiliert und auf irgendeinem anderen Unix-Rechner im Internet an einem beliebigen Standort in der Welt ausgeführt werden. Der Client-Code kann mit den entsprechenden Parametern ausgeführt werden, um jede Datei zu holen, auf die der Server auf seinem Rechner zugreifen kann. Diese Datei wird in der Standardausgabe ausgegeben, die natürlich in eine Datei oder Pipe umgeleitet werden kann.

```
/* Bei diesem Code handelt es sich um ein Client-Programm, das eine Datei von
 * dem Server-Programm, das in dem nächsten Listing aufgeführt ist, anfordern
 * kann. Der Server antwortet, indem er die ganze Datei sendet.
 */
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345          /* beliebig, aber Client & Server müssen
                                sich auf einen einigen */
#define BUF_SIZE 4096             /* Größe des übertragenen Blocks */

int main(int argc, char **argv)
{
    int c, s, bytes;
```

```

char buf[BUF_SIZE];           /* Puffer für eingehende Datei */
struct hostent *h;            /* Info über den Server */
struct sockaddr_in channel;   /* enthält IP-Adresse */

if (argc != 3) fatal("Verwendung: client Servername Dateiname ");
h = gethostbyname(argv[1]);    /* IP-Adresse des Hosts nachschlagen */
if (!h) fatal("gethostbyname failed");

s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (s < 0) fatal("Socket");
memset(&channel, 0, sizeof(channel));
channel.sin_family= AF_INET;
memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
channel.sin_port= htons(SERVER_PORT);

c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
if (c < 0) fatal("Verbindung fehlgeschlagen");

/* Verbindung nun aufgebaut. Sende Dateiname mit 0 Byte am Ende. */
write(s, argv[2], strlen(argv[2])+1);

/* Datei holen und auf die Standardausgabe schreiben. */
while (1) {
    bytes = read(s, buf, BUF_SIZE);    /* aus dem Socket lesen */
    if (bytes <= 0) exit(0);          /* auf Dateiende prüfen */
    write(1, buf, bytes);            /* auf Standardausgabe schreiben */
}
fatal(char *string)
{
    printf("%s\n", string);
    exit(1);
}

```

a

```

#include <sys/types.h>           /* Dies ist der Server-Code. */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345          /* beliebig, aber Client & Server
                                    müssen sich auf einen einigen */
#define BUF_SIZE 4096             /* Größe des übertragenen Blocks */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
    int s, b, l, fd, sa, bytes, on = 1;
    char buf[BUF_SIZE];           /* Puffer für ausgehende Datei */
    struct sockaddr_in channel;   /* enthält IP-Adresse */

    /* Erstelle die Adressstruktur, die an den Socket gebunden wird. */
    memset(&channel, 0, sizeof(channel)); /* Nullkanal */

```

```

channel.sin_family = AF_INET;
channel.sin_addr.s_addr = htonl(INADDR_ANY);
channel.sin_port = htons(SERVER_PORT);

/* Passiv offen. Warte auf Verbindung.*/
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); /* Socket erstellen */
if (s < 0) fatal("Socket fehlgeschlagen");
setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
if (b < 0) fatal("bind fehlgeschlagen");

l = listen(s, QUEUE_SIZE); /* Warteschlangengröße angeben */
if (l < 0) fatal("listen fehlgeschlagen");

/* Socket ist nun eingerichtet und mit Adresse verbunden.*/
/* Auf Verbindung warten und sie verarbeiten.*/
while (1) {
    sa = accept(s, 0, 0); /* blockiere für die
                           Verbindungsanforderung */
    if (sa < 0) fatal("accept fehlgeschlagen ");

    read(sa, buf, BUF_SIZE); /* Lesen des Dateinamens aus Socket */

    /* Hole Datei und gib sie aus.*/
    fd = open(buf, O_RDONLY); /* zurückzusendende Datei öffnen */
    if (fd < 0) fatal("open fehlgeschlagen ");

    while (1) {
        bytes = read(fd, buf, BUF_SIZE); /* aus Datei lesen */
        if (bytes <= 0) break; /* auf Dateiende prüfen */
        write(sa, buf, bytes); /* Bytes in Socket schreiben */
    }
    close(fd); /* Datei schließen */
    close(sa); /* Verbindung schließen */
}
}

```

b**Abbildung 6.6:** (a) Client-Code mit Sockets. (b)Server-Code.

Betrachten wir als Erstes den Server-Code. Dieser lädt zunächst einige Standard-Header-Dateien, wobei die drei letzten die wichtigsten internetbezogenen Definitionen und Datenstrukturen enthalten. Als Nächstes wird *SERVER_PORT* als 12 345 festgelegt. Die Zahl wurde frei gewählt. Jede Zahl zwischen 1 024 und 65 535 funktioniert, solange sie nicht von einem anderen Prozess verwendet wird. Portnummern unter 1 023 sind für privilegierte Benutzer reserviert.

Die nächsten zwei Zeilen des Servers definieren die beiden erforderlichen Konstanten. Die erste legt die für die Dateiübertragung verwendete Blockgröße in Byte fest. Die zweite legt fest, wie viele Verbindungen gleichzeitig gehalten werden können, bevor alle weiteren eingehenden verworfen werden.

Nach der Deklaration der lokalen Variablen beginnt der Server-Code. Als Erstes wird eine Datenstruktur initialisiert, die die IP-Adresse des Servers enthält. Diese Datenstruktur wird bald an den Socket des Servers gebunden. Der Aufruf von *memset* setzt die komplette Datenstruktur auf 0. Die drei nächsten Zuweisungen füllen drei Felder davon. Die letzte enthält den Port des Servers. Die Funktionen *htonl* und *htons* konvertieren Werte in ein Standardformat, sodass der Code auf Big-Endian-Rechnern (wie SPARC) und Little-Endian-Rechnern (wie Intel x86) gleichermaßen läuft. Ihre exakte Semantik spielt hier keine Rolle.

Als Nächstes erstellt der Server einen Socket und führt eine Prüfung auf Fehler durch (angegeben durch *s<0*). In der Produktionsversion des Codes könnte die Fehlermeldung ruhig etwas ausführlicher ausfallen. Der Aufruf von *setsockopt* ist erforderlich, um den Port wiederzuverwenden, damit der Server unbegrenzt Anforderungen abarbeiten kann. Anschließend wird die IP-Adresse an den Socket gebunden und es wird geprüft, ob der Aufruf von *bind* erfolgreich war. Der letzte Schritt bei der Initialisierung ist der Aufruf von *listen*, um anzukündigen, dass der Server nun eingehende Aufrufe behandelt, und um dem System mitzuteilen, dass es bis zu *QUEUE_SIZE* Aufrufe puffern soll, falls neue Anforderungen eingehen, während der Server immer noch die aktuelle verarbeitet. Ist die Warteschlange voll und es kommen weitere Anforderungen herein, werden sie ohne weitere Benachrichtigung verworfen.

An diesem Punkt steigt der Server in die Hauptschleife ein, die er nie wieder verlässt. Sie können ihn nur beenden, indem Sie ihn von außen „abschießen“. Der Aufruf von *accept* blockiert den Server, bis ein Client versucht, eine Verbindung aufzubauen. Ist der *accept*-Aufruf erfolgreich, liefert er einen Socket-Descriptor zurück, der zum Lesen und Schreiben verwendet werden kann, so wie Dateideskriptoren zum Lesen und Schreiben von Pipes eingesetzt werden können. Aber im Unterschied zu Pipes, die unidirektional sind, sind Sockets bidirektional, sodass mit *sa* (Socket-Adresse) von der Verbindung gelesen und in sie geschrieben werden kann. Ein Dateideskriptor für Pipes kann entweder zum Lesen oder zum Schreiben eingesetzt werden, aber nicht für beides.

Nachdem die Verbindung aufgebaut ist, liest der Server darüber den Dateinamen. Ist der Name noch nicht verfügbar, blockiert der Server und wartet darauf. Nach Erhalt des Dateinamens öffnet der Server die Datei und tritt in eine Schleife ein, in der er abwechselnd Blöcke aus der Datei liest und sie in den Socket schreibt, bis die gesamte Datei kopiert ist. Dann schließt der Server die Datei und die Verbindung und wartet auf die nächste eingehende Verbindung. Diese Schleife wird unendlich wiederholt.

Nun betrachten wir den Client-Code. Um ihn zu verstehen, müssen wir wissen, wie er aufgerufen wird. Angenommen der Name lautet *client*, so sieht ein typischer Aufruf wie folgt aus:

```
client flits.cs.vu.nl /usr/tom/filename >f
```

Dieser Aufruf funktioniert nur, wenn der Server bereits auf *flits.cs.vu.nl* läuft, die Datei */usr/tom/filename* existiert und der Server Lesezugriff darauf hat. Ist der Aufruf erfolgreich, wird die Datei über das Internet übertragen und nach *f* geschrieben. Danach wird das Client-Programm beendet. Da der Server nach einer Übertragung

weiterarbeitet, kann der Client immer wieder erneut gestartet werden und weitere Dateien holen.

Der Client-Code beginnt mit einigen *include*-Direktiven und Deklarationen. Die Ausführung beginnt mit einer Prüfung, ob er mit der richtigen Anzahl an Argumenten aufgerufen wurde (*argc=3* steht für den Programmnamen plus zwei Argumente). Beachten Sie, dass *argv[1]* den Server-Namen enthält (wie *flits.cs.vu.nl*) und mit *gethostbyname* in eine IP-Adresse konvertiert wird. Diese Funktion schlägt unter Verwendung von DNS den Namen nach. DNS wird in *Kapitel 7* behandelt.

Als Nächstes wird ein Socket erstellt und initialisiert. Danach versucht der Client, mit *connect* eine TCP-Verbindung zum Server aufzubauen. Wenn der Server auf dem angegebenen Rechner läuft und mit *SERVER_PORT* verbunden wurde, und sich entweder im Leerlauf befindet oder in seiner *listen*-Warteschlange Platz hat, wird die Verbindung (endlich) aufgebaut. Der Client sendet über die Verbindung den Dateinamen, indem er ihn in den Socket schreibt. Es wird 1 Byte mehr gesendet als der eigentliche Name, da das 0-Byte, das den Namen abschließt, auch an den Server übertragen werden muss, damit dieser weiß, wo der Name endet.

Nun steigt der Client in eine Schleife ein, indem er die Datei blockweise aus dem Socket liest und sie auf die Standardausgabe kopiert. Ist dies fertig, beendet sich der Client.

Die Prozedur *fatal* gibt eine Fehlermeldung aus und beendet das Programm. Der Server benötigt die gleiche Prozedur, diese wurde hier nicht noch einmal aufgeführt. Da Client und Server getrennt kompiliert werden und in der Regel auf verschiedenen Rechnern ausgeführt werden, können sie den Code von *fatal* nicht gemeinsam nutzen.

Diese beiden Programme (wie auch die anderen Materialien zu diesem Buch) können Sie über die Companion Website des Buches herunterladen:

<http://www.pearson-studium.de/4137>



indem Sie auf den Website-Link neben dem Foto des Buchumschlags klicken. Sie können den Code herunterladen und auf einem beliebigen Unix-System (wie Solaris, BSD, Linux) wie folgt kompilieren:

```
cc -o client client.c -lsocket -lnsl
cc -o server server.c -lsocket -lnsl
```

Sie starten den Server durch Eingabe von

server

Der Client benötigt wie oben erwähnt zwei Argumente. Auf der Website ist auch eine Windows-Version verfügbar.

Nur nochmals zu Erinnerung: Dies ist nicht der ultimative Server. Seine Fehlerprüfung ist dürfsig und die Fehlerberichte sind mittelmäßig. Da er alle Anforderungen streng der Reihe nach abarbeitet (er besitzt nur einen Thread), ist die Leistung ungenügend. Er hat eindeutig noch nie etwas über Sicherheit gehört, und die Verwendung reiner Unix-Systemaufrufe garantiert noch keine Plattformunabhängigkeit. Weiterhin geht er von

einigen Annahmen aus, die technisch nicht zulässig sind, wie beispielsweise dass der Dateiname in den Puffer passt und „atomar“ übertragen wird. Abgesehen von diesen Mängeln ist er jedoch ein vollständiger funktionierender Internetdateiserver. In den Übungen ist der Leser herzlich eingeladen, ihn zu verbessern. Weitere Informationen über die Programmierung mit Sockets finden Sie in Donahoo und Calvert (2008, 2009).

6.2 Elemente von Transportprotokollen

Der Dienst der Transportschicht wird durch ein **Transportprotokoll** (*transport protocol*) implementiert, das zwischen den zwei Transportinstanzen benutzt wird. In gewisser Weise ähneln Transportprotokolle den in *Kapitel 3* behandelten Protokollen der Sicherungsschicht. Beide befassen sich u.a. mit Fehlerüberwachung, Steuerung der Reihenfolge und Flusskontrolle.

Es gibt aber auch wichtige Unterschiede zwischen den beiden Protokollarten. Diese Unterschiede sind auf die gänzlich anderen Umgebungen zurückzuführen, in denen diese Protokolle arbeiten (►Abbildung 6.7). Auf der Sicherungsschicht kommunizieren zwei Router direkt über einen physikalischen Kanal, mit Kabel oder drahtlos. Auf der Transportschicht hingegen wird dieser Kanal durch ein ganzes Netz ersetzt. Dieser Unterschied hat viele wichtige Auswirkungen auf die Protokolle.

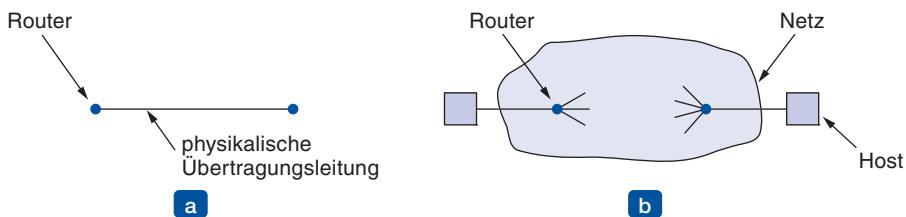


Abbildung 6.7: (a) Umgebung der Sicherungsschicht. (b) Umgebung der Transportschicht.

Zum einen muss ein Router bei Punkt-zu-Punkt-Verbindungen mittels Glasfaser oder andere Kabel normalerweise nicht angeben, mit welchem Router er kommunizieren möchte – jede Ausgangsleitung führt genau zu einem bestimmten Router. Auf der Transportschicht muss das Ziel ausdrücklich adressiert werden.

Zum anderen ist es einfach, eine Verbindung über ein Kabel wie in ►Abbildung 6.7a einzurichten, denn das andere Ende ist immer bereit (außer, wenn es abgestürzt ist). Es gibt nicht viel zu beachten. Sogar bei drahtlosen Verbindungen gibt es keinen großen Unterschied. Durch Senden einer Nachricht werden automatisch alle Ziele erreicht. Wird die Nachricht aufgrund eines Fehlers nicht bestätigt, kann sie erneut gesendet werden. Bei der Transportschicht ist der erste Verbindungsaufbau kompliziert, wie Sie noch sehen werden.

Ein weiterer (zunehmend lästiger) Unterschied zwischen der Sicherungsschicht und der Transportschicht ist die potenzielle Speicherkapazität im Netz. Wenn ein Router ein Paket über eine Verbindung schickt, kommt dieses entweder an oder geht verloren,

aber es kann nicht wild umherreisen, sich im hintersten Winkel der Welt verstecken und plötzlich wieder auftauchen, nachdem alle anderen Paketen, einschließlich derer, die viel später abgeschickt wurden, bereits angekommen sind. Wenn das Netz Datagramme verwendet, die intern unabhängig voneinander weitergeleitet werden, besteht durchaus die Wahrscheinlichkeit, dass Pakete Umwege machen und zu spät und nicht in der erwarteten Reihenfolge ankommen. Es kann sogar passieren, dass Duplikate der Pakete ankommen. Die Fähigkeit des Netzes, Pakete zu verzögern und zu duplizieren, kann manchmal verhängnisvolle Folgen haben und den Einsatz spezieller Protokolle erfordern, um die Informationen korrekt zu übertragen.

Ein weiterer Unterschied zwischen der Sicherungs- und der Transportschicht ist eher eine Frage des Umfangs als eine Frage des Typs. Auf beiden Schichten sind Zwischen-speicherung und Flusskontrolle nötig. Die große und schwankende Anzahl an Verbindungen auf der Transportschicht mit einer Bandbreite, die in Abhängigkeit der konkurrierenden Verbindungen fluktuiert, erfordert unter Umständen eine andere Methode als in der Sicherungsschicht. In *Kapitel 3* weisen manche Protokolle jeder Leitung eine feste Pufferzahl zu, sodass bei Ankunft eines Rahmens immer Pufferspeicher verfügbar ist. Aufgrund der Vielzahl der zu verwaltenden Verbindungen auf der Transportschicht und die schwankende Bandbreite, die jede Verbindung empfangen kann, ist das Konzept, jeder Verbindung viele Puffer zuzuordnen, nicht so reizvoll. Diese und andere wichtige Fragen werden in den nächsten Abschnitten behandelt.

6.2.1 Adressierung

Möchte ein Anwendungsprozess (z.B. ein Benutzer) eine Verbindung zu einem entfernten Anwendungsprozess aufbauen, muss er den gewünschten Partner genau angeben. (Beim verbindungslosen Transport gibt es das gleiche Problem: An wen soll die Nachricht gesendet werden?) Normalerweise werden Transportadressen definiert, auf denen Prozesse die eingehenden Verbindungsanforderungen abhören können. Im Internet werden diese Endpunkte als **Ports** bezeichnet. Wir verwenden hier den neutralen Begriff **TSAP** (*Transport Service Access Point*, Zugangspunkt des Transportschichtdienstes). Die analogen Endpunkte auf der Vermittlungsschicht (d.h. Adressen der Vermittlungsschicht) heißen **NSAPs** (*Network Service Access Point*, Zugangspunkt des Vermittlungsschichtdienstes). IP-Adressen beispielsweise sind NSAPs.

In ▶ Abbildung 6.8 wird die Beziehung zwischen NSAP, TSAP und der Transportverbindung dargestellt. Anwendungsprozesse, sowohl Clients als auch Server, können sich mit einem lokalen TSAP verbinden, um eine Verbindung zu einem entfernten TSAP aufzubauen. Diese Verbindungen laufen auf jedem Host, wie gezeigt, über NSAPs. Da in einigen Netzen jeder Rechner einen einzigen NSAP besitzt, ist eine Methode erforderlich, um mehrere Transportendpunkte zu unterscheiden, die diesen NSAP gemeinsam nutzen. Dies ist der Zweck von TSAPs.

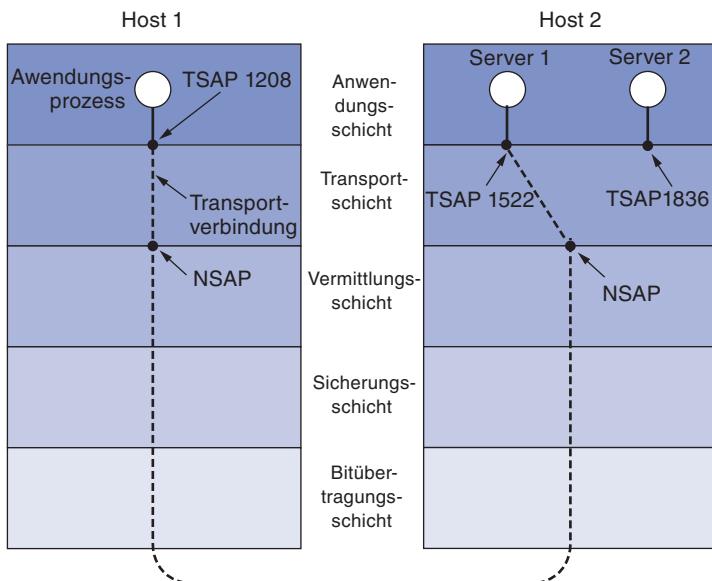


Abbildung 6.8: TSAPs, NSAPs und Transportverbindungen.

Für eine Transportverbindung kann ein mögliches Szenario wie folgt aussehen:

1. Ein Mailserver-Prozess verbündet sich mit TSAP 1522 auf Host 2, um auf einen ankommenden Anruf zu warten. Wie ein Prozess sich mit einem TSAP verbinden kann, liegt außerhalb des Netzmodells und hängt gänzlich vom lokalen Betriebssystem ab. Beispielsweise kann ein Aufruf wie unser LISTEN verwendet werden.
2. Ein Anwendungsprozess auf Host 1 möchte eine E-Mail-Nachricht senden. Also verbindet er sich mit TSAP 1208 und gibt eine CONNECT-Anforderung aus. Die Anforderung gibt TSAP 1208 auf Host 1 als Quelle und TSAP 1522 als Ziel an. Hierdurch wird eine Transportverbindung zwischen dem Anwendungsprozess und dem Server aufgebaut.
3. Der Anwendungsprozess sendet die E-Mail-Nachricht.
4. Der Mailserver antwortet, dass er die Nachricht ausliefern wird.
5. Die Transportverbindung wird danach freigegeben.

Es können auch andere Server auf Host 2 vorhanden sein, die mit anderen TSAPs verbunden sind und auf eingehende Verbindungen warten, die über den gleichen NSAP eingehen.

Abbildung 6.8 ist ganz nett, abgesehen davon, dass ein kleines Problem unter den Teppich gekehrt wurde: Woher weiß der Benutzerprozess auf Host 1, dass der Mailserver mit TSAP 1522 verbunden ist? Eine Möglichkeit ist, dass der Mailserver bereits seit Jahren mit TSAP 1522 verbunden ist und alle Netzbürger darüber inzwischen informiert sind. In diesem Modell haben Dienste feste TSAP-Adressen, die in Dateien an

bekannten Orten gespeichert sind. Auf Unix-Systemen ist dies die Datei `/etc/services`, die auflistet, welche Server dauerhaft mit welchen Ports verbunden sind, einschließlich der Information, dass der Mailserver auf TCP Port 25 zu finden ist.

Nun sind solch feste TSAP-Adressen vielleicht bei einer kleinen Zahl von wichtigen Diensten (wie Webservern), die sich nie ändern, angebracht. Meist aber möchten Benutzerprozesse mit anderen Benutzerprozessen kommunizieren, die nur vorübergehend existieren und keine im Voraus bekannte TSAP-Adresse haben.

Für diese Situation kann ein anderes System verwendet werden. Bei diesem Modell gibt es einen speziellen Prozess, den sogenannten **Portmapper**. Um die TSAP-Adresse für einen bestimmten Dienst, z.B. „BitTorrent“, herauszufinden, baut der Benutzer eine Verbindung zum Portmapper auf (der auf einem bekannten TSAP auf eingehende Verbindungen wartet). Dann sendet der Benutzer eine Nachricht, in der er den Namen des Dienstes angibt. Der Portmapper sendet ihm die TSAP-Adresse zurück. Dann baut der Benutzer die Verbindung zum Portmapper ab und eine neue zu dem gewünschten Dienst auf.

Bei diesem Modell muss sich ein neu erzeugter Dienst unter Angabe seines Namens (normalerweise eine ASCII-Zeichenkette) und seiner TSAP-Adresse beim Portmapper anmelden. Der Portmapper speichert diese Informationen in seiner internen Datenbank, damit er bei späteren Anfragen die Antwort kennt.

Die Funktion des Portmappers entspricht der einer Telefonauskunft. Er gibt zu einem Namen die betreffende Adresse aus. Genau wie bei der Telefonauskunft ist es sehr wichtig, dass die TSAP-Adresse des Portmappers möglichst allen bekannt ist. Wenn Sie die Nummer der Auskunft nicht kennen, können Sie dort auch nicht anrufen, um eine Telefonnummer zu erfragen. Sollten Sie glauben, die Nummer der Auskunft sei einfach, dann versuchen Sie es doch einmal im Ausland.

Viele der Server-Prozesse, die auf einem Rechner laufen können, werden nur selten benötigt. Es ist reine Verschwendug, wenn jeder von ihnen den ganzen Tag lang aktiv ist und auf eine feste TSAP-Adresse hört. Ein alternatives System ist in ► Abbildung 6.9 vereinfacht dargestellt. Es wird als **Verbindungsaufbauprotokoll** (*initial connection protocol*) bezeichnet. Anstatt dass jeder Server an einem bekannten TSAP auf eingehende Verbindungen lauscht, hat jeder Rechner, der Dienste für entfernte Benutzer zur Verfügung stellen möchte, einen speziellen **Prozess-Server**, der als Proxy für seltener verwendete Server fungiert. Dieser Server wird auf Unix-Systemen als `inetd` bezeichnet. Er hört gleichzeitig eine Reihe von Ports ab und wartet auf eine Verbindungsanforderung. Mögliche Benutzer eines Dienstes beginnen mit einer CONNECT-Anforderung, die die TSAP-Adresse des gewünschten Dienstes enthält. Wartet kein Server auf sie, erhalten sie eine Verbindung zum Prozess-Server (► Abbildung 6.9a).

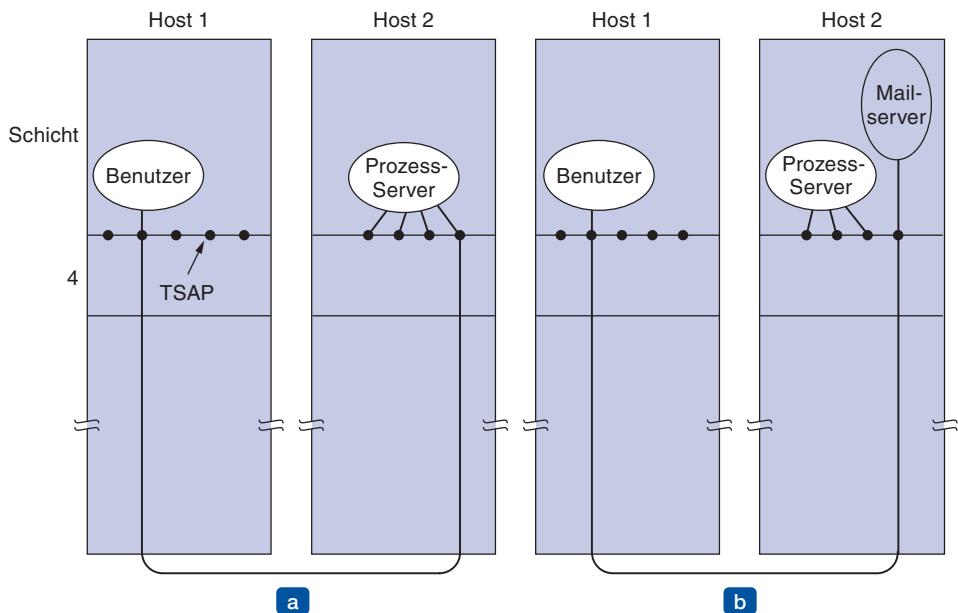


Abbildung 6.9: Wie ein Benutzerprozess auf Host 1 über einen Prozess-Server eine Verbindung zu dem Mailserver auf Host 2 aufbaut.

Nach Erhalt der Anforderung startet der Prozess-Server den angeforderten Server und übergibt ihm die bereits vorhandene Verbindung zu dem Benutzer. Der neue Server führt die angeforderte Aufgabe aus, während der Prozess-Server wieder in seine Ausgangsposition zurückgeht und auf neue Anforderungen prüft (►Abbildung 6.9b). Diese Methode ist nur anwendbar, wenn Server bei Bedarf erzeugt werden können.

6.2.2 Verbindungsauflaufbau

Verbindungsauflaufbau hört sich einfach an, ist aber erstaunlich kompliziert. Auf den ersten Blick mag es ausreichend erscheinen, dass eine Transportinstanz einfach ein CONNECTION REQUEST-Segment an das Ziel sendet und auf die Antwort CONNECTION ACCEPTED wartet. Probleme entstehen, wenn das Netz Pakete verliert, verzögert, beschädigt und dupliziert. Dies führt zu ernsthaften Komplikationen.

Man stelle sich ein Netz vor, das so überlastet ist, dass Bestätigungen nur selten rechtzeitig ankommen und jedes Paket das Zeitlimit überschreitet und zweimal- bis dreimal übertragen wird. Angenommen, das Netz verwendet intern Datagramme und die Pakete wandern auf verschiedenen Wegen. Einige Pakete bleiben in einem Datenstau stecken und benötigen sehr viel Zeit, bis sie ankommen. Das heißt, sie verzögern sich im Netz und tauchen erst viel später wieder auf, wenn der Sender sie schon als verloren aufgegeben hat.

Im schlimmsten Fall kann beispielsweise Folgendes passieren: Ein Benutzer baut eine Verbindung zu einer Bank auf und weist die Bank an, einen hohen Geldbetrag auf das

Konto einer nicht ganz vertrauenswürdigen Person zu überweisen. Unglücklicherweise schlagen die Pakete Umwege ein und treiben sich in den hintersten Winkeln des Netzes herum, bevor sie ihr Ziel ansteuern. In der Zwischenzeit bricht der Sender wegen Zeitüberschreitung die Übertragung ab und sendet die Pakete erneut. Dieses Mal nehmen die Pakete den kürzesten Weg und werden direkt zugestellt, woraufhin der Sender die Verbindung abbaut.

Leider tauchen danach die zuerst gesendeten Pakete wieder auf und kommen in der richtigen Reihenfolge am Ziel an. Die Pakete bitten die Bank, eine neue Verbindung aufzubauen und das Geld (noch einmal) zu überweisen. Die Bank kann nicht beurteilen, ob es sich hier um Duplikate handelt. Sie geht davon aus, dass es sich um eine zweite unabhängige Transaktion handelt, und überweist den Betrag erneut.

Dieses Szenario mag an den Haaren herbeigezogen scheinen, ja sogar unglaublich klingen, aber der Punkt ist, dass Protokolle für alle Fälle korrekt entworfen sein müssen. Während die geläufigsten Fälle effizient implementiert werden müssen, um eine gute Netzleistung zu erzielen, steht bei den seltenen Fällen im Mittelpunkt, dass das Protokoll nicht versagt. Denn sonst hätten wir nur ein Schönwetternetz, das bei extremen Bedingungen ohne Warnung versagen kann.

Im restlichen Teil dieses Abschnitts wenden wir uns den Problemen mit verzögerten Duplikaten zu, mit besonderer Betonung auf die Algorithmen, die für einen zuverlässigen Verbindungsauflauf sorgen. Diese können Horrorszenarien, wie sie oben beschrieben wurden, verhindern. Der Kern des Problems ist, dass die verzögerten Duplikate als neue Pakete betrachtet werden. Wir können nicht verhindern, dass Pakete dupliziert und verzögert werden. Doch wenn dies passiert, müssen die Pakete als Duplikate erkannt und verworfen werden, anstatt sie als neue Pakete zu verarbeiten.

Für dieses Problem gibt es verschiedene Lösungen, von denen keine wirklich zufriedenstellend ist. Eine Möglichkeit wäre die Verwendung von Wegwerfadressen. Bei diesem Ansatz wird immer, wenn eine Transportadresse benötigt wird, eine neue erzeugt. Nach der Freigabe der Verbindung wird die Adresse „weggeworfen“ und nie wieder verwendet. Verzögerte Paketduplikate finden dann nie wieder ihren Weg zu einem Transportprozess und können keinen Schaden mehr anrichten. Diese Strategie erschwert allerdings, überhaupt eine Verbindung mit einem Prozess herzustellen.

Eine weitere Möglichkeit wäre, jeder Verbindung eine eindeutige Verbindungskenntnung zu geben (d.h. eine Sequenznummer, die mit jeder aufgebauten Verbindung um 1 erhöht wird). Diese Kennung wählt der anfordernde Partner aus und überträgt sie in jedem Segment, einschließlich dem, mit dem die Verbindung angefordert wird. Bei jeder Verbindungs freigabe könnten beide Transportinstanzen eine Tabelle aktualisieren, in der die bereits abgebauten Verbindungen als Paare der Form „gleichgeordnete Transportinstanz – Verbindungs kennung“ eingetragen sind. Wenn nun eine Verbindungsanforderung eingeht, kann sie mit der Tabelle verglichen werden, um festzustellen, ob es sich dabei um eine bereits früher freigegebene Verbindung handelt.

Leider hat dieses Methode einen grundlegenden Haken: Jede Transportinstanz müsste eine bestimmte Menge von Archivdaten für unbegrenzte Zeit aufbewahren. Diese

Archivdaten müssen sowohl auf dem Ausgangs- als auch auf dem Zielrechner gespeichert sein. Andernfalls hat ein Rechner, der abstürzt und seinen Speicherinhalt verliert, keine Daten mehr darüber, welche Verbindungskennungen bereits von seinen gleichgeordneten Transportinstanzen verwendet wurden.

Das Problem muss also anders angepackt werden. Anstatt Paketen die Möglichkeit einzuräumen, ewig in einem Netz zu überleben, entwickeln wir einen Mechanismus, der alte, noch umherwandernde Pakete beseitigt. Damit ist das Problem besser in den Griff zu bekommen.

Die Lebensdauer eines Pakets kann durch eine (oder mehrere) der folgenden Methoden auf ein definiertes Maximum beschränkt werden:

- 1.** Entwurf beschränkter Netze.
- 2.** Jedes Paket enthält einen Teilstreckenzähler.
- 3.** Jedes Paket wird mit einem Zeitstempel versehen.

Der erste Punkt umfasst alle Methoden, die verhindern, dass Pakete sich auf Umwege begeben, kombiniert mit einer Möglichkeit, Verzögerung einzugrenzen, einschließlich Stau auf dem (jetzt bekannten) längstmöglichen Pfad. Dies ist schwierig angesichts der Tatsache, dass der Umfang eines Netzes von kleinräumig (z.B. stadtbezogen) bis international reicht. Bei der zweiten Methode wird ein Teilstreckenzähler mit einem passenden Wert initialisiert und jedes Mal reduziert, wenn das Paket weitergereicht wird. Das Netzprotokoll verwirft einfach alle Pakete, deren Teilstreckenzähler null wird. Bei der dritten Methode erhält jedes Paket bei seiner Erzeugung einen Zeitstempel und die Router sind gehalten, alle Pakete zu entfernen, die eine zuvor festgelegte Zeitspanne überschritten haben. Diese Methode setzt voraus, dass die Uhren der Router synchronisiert werden, was allein schon eine nicht ganz einfache Aufgabe ist. In der Praxis ist ein Teilstreckenzähler eine hinreichend genaue Annäherung an die Zeit.

In der Praxis muss garantiert werden, dass nicht nur das Paket erledigt ist, sondern auch alle Bestätigungen dieses Pakets. Daher führen wir die Variable T ein, die ein kleines Vielfaches der tatsächlichen maximalen Paketlebensdauer darstellt. Die maximale Paketlebensdauer für ein Netz ist eine konservative Konstante; für das Internet wurden dafür mehr oder weniger willkürlich 120 Sekunden festgelegt. Das Vielfache davon ist vom Protokoll abhängig und hat einfach nur den Zweck, T zu verlängern. Wenn wir nach Verschicken eines Paket T Sekunden abwarten, können wir sicher sein, dass alle Spuren dieses Pakets verschwunden sind. Weder das Paket selbst noch Bestätigungen desselben können plötzlich auftauchen und Verwirrung stiften.

Durch die Begrenzung der Paketlebensdauer ist es möglich, eine narrensichere Methode zum Verwerfen von verzögerten, duplizierten Segmenten zu entwickeln. Die im Folgenden beschriebene Methode stammt von Tomlinson (1975), mit Verbesserungen von Sunshine und Dalal (1978). In der Praxis werden häufig Varianten davon eingesetzt, auch in TCP.

Das Grundkonzept ist dabei, dass die Quelle alle Segmente mit Sequenznummern versieht, die innerhalb der T Sekunden nicht wiederverwendet werden. Der Zeitraum T und die Anzahl der Pakete pro Sekunde legen die Größe der Sequenznummern fest. Auf diese Weise steht jederzeit immer nur ein Paket mit einer gegebenen Sequenznummer aus. Es können immer noch Duplikate von diesem Paket auftreten, die dann vom Ziel verworfen werden müssen. Es kann jedoch nicht mehr passieren, dass ein verzögertes Duplikat eines alten Pakets ein neues Paket mit der gleichen Sequenznummer überholt und an seiner statt vom Ziel akzeptiert wird.

Um das Problem zu umgehen, dass ein Rechner nach einem Absturz nicht mehr weiß, wo er war, haben Sie unter anderem die Möglichkeit, nach der Wiederherstellung die Transportinstanzen für eine Zeit von T Sekunden lahmzulegen. In dieser inaktiven Zeit werden alle alten Segmente aufgelöst, sodass der Sender mit einer beliebigen Sequenznummer neu starten kann. In einem komplexen Internetwork kann T allerdings sehr groß ausfallen, sodass diese Strategie relativ unattraktiv ist.

Tomlinson schlug stattdessen vor, jeden Host mit einer Tageszeituhr auszustatten. Die Uhren der einzelnen Hosts brauchen nicht synchronisiert zu werden. Jede dieser Uhren soll ein binärer Zähler sein und sich in gleichmäßigen Abständen erhöhen. Außerdem muss die Bitzahl im Zähler größer oder gleich der Bitzahl der Sequenznummern sein. Schließlich noch die wichtigste Voraussetzung: Die Uhr muss auch dann weiterlaufen, wenn der Host abstürzt.

Wenn eine Verbindung aufgebaut wird, werden die niederwertigen k Bit der Uhr als k -Bit-Anfangssequenznummer verwendet. Auf diese Weise nummeriert jede Verbindung ihre Segmente mit anderen Sequenznummern (im Gegensatz zu den in *Kapitel 3* beschriebenen Protokollen). Der Folgeabstand sollte so groß sein, dass zu dem Zeitpunkt, an dem die Sequenznummern wieder von vorn beginnen, alte Segmente mit bereits existierenden Sequenznummern längst wieder aufgelöst sind. Diese lineare Beziehung zwischen Zeit und Anfangsnummern ist in ►Abbildung 6.10a dargestellt. Die verbotene Zone zeigt die Zeiten, in denen die Segmentsequenznummern nicht gültig sind, woraus sich deren Nutzung ergibt. Wenn irgendein Segment mit einer Sequenznummer in diesem Bereich gesendet wird, könnte es sich verzögern und als anderes Paket mit der gleichen Sequenznummer auftreten, das nur kurz darauf ausgegeben wurde. Wenn beispielsweise der Host abstürzt und nach 70 Sekunden neu startet, wird er Anfangssequenznummern verwenden, die auf der Uhr basieren, um dort fortzufahren, wo aufgehört wurde. Der Host startet nicht mit einer niederen Sequenznummer aus der verbotenen Zone.

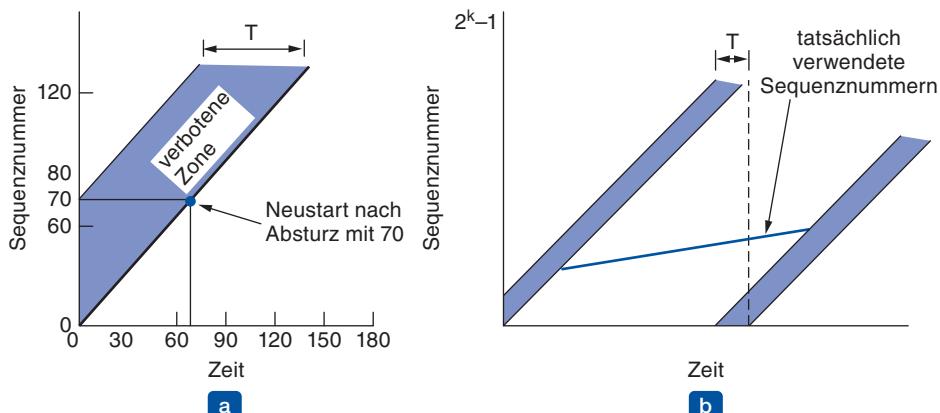


Abbildung 6.10: (a) Segmente dürfen sich nicht in der verbotenen Zone bewegen.
 (b) Das Problem der erneuten Synchronisation.

Haben sich die beiden Transportinstanzen auf die Anfangssequenznummer geeinigt, kann für die Datenflusskontrolle jedes beliebige Schiebefensterprotokoll verwendet werden. Dieses Fensterprotokoll sucht und entsorgt vorschriftsmäßig Paketduplicata, nachdem sie bereits akzeptiert worden sind. In der Realität ist die Kurve der Anfangssequenznummern (die fette Linie) nicht linear, sondern treppenförmig. Das liegt daran, dass die Uhr in diskreten Schritten erhöht wird, was wir hier der Einfachheit halber ignorieren.

Um die Sequenznummern der Pakete aus der verbotenen Zone herauszuhalten, müssen wir zwei Dinge berücksichtigen. Wir können in zweierlei Hinsicht in Schwierigkeiten geraten. Wenn ein Host zu viele Daten zu schnell über eine neu geöffnete Verbindung sendet, kann die Kurve der tatsächlichen Sequenznummern in Abhängigkeit von der Zeit steiler ansteigen als die Kurve der Anfangssequenznummern, sodass die Sequenznummern in die verbotene Zone rutschen. Um dies zu verhindern, beträgt die maximale Datenübertragungsrate über eine Verbindung nur ein Segment pro Takteinheit. Dies bedeutet auch, dass die Transportinstanz warten muss, bis die Uhr tickt, bevor sie nach dem Hochfahren nach einem Absturz eine neue Verbindung öffnet, damit die gleiche Sequenznummer nicht zweimal vergeben wird. Diese beiden Tatsachen sprechen für eine kurze Takteinheit (wenige μ s oder noch kleiner). Aber die Uhr kann bezüglich der Sequenznummer nicht zu schnell ticken. Bei einer Taktrate C und einem Sequenznummernabstand S muss $S/C > T$ sein, damit die Sequenznummern nicht zu schnell von vorne beginnen.

Das Betreten der verbotenen Zone durch zu schnelles Senden ist leider nicht das einzige Problem, das zu Schwierigkeiten führen kann. In ►Abbildung 6.10b ist ersichtlich, dass bei jeder Übertragungsrate, die niedriger ist als die Taktrate, die Kurve der tatsächlich verwendeten Sequenznummern in Abhängigkeit von der Zeit irgendwann von links in die verbotene Zone eindringen wird. Je steiler die Kurve der tatsächlichen Sequenznummern ansteigt, desto länger dauert es, bis die verbotene Zone erreicht ist. Das Ver-

meiden dieser Situation hat Einfluss darauf, wie langsam Sequenznummern auf einer Verbindung voranschreiten können (oder wie lang die Verbindungen dauern können).

Die taktgesteuerte Methode löst das Problem, die verzögerten, duplizierten Segmente nicht von den neuen Segmenten unterscheiden zu können. Sie wirft jedoch ein praktisches Problem auf, wenn sie zum Aufbau von Verbindungen verwendet wird. Da wir uns normalerweise am Ziel die Sequenznummern über Verbindungen hinweg nicht merken, wissen wir immer noch nicht, ob ein CONNECTION REQUEST-Segment, das eine Anfangssequenznummer enthält, ein Duplikat einer neueren Verbindung ist. Dieses Problem taucht während einer Verbindung nicht auf, da das Schiebefensterprotokoll die aktuelle Sequenznummer festhält.

Um dieses Problem zu lösen, führte Tomlinson (1975) eine Methode namens **Dreiwege-Handshake** (*three-way handshake*) ein. Dieses Verbindungsauflaufbauprotokoll setzt voraus, dass ein Partner beim anderen prüft, ob die Verbindungsanforderung tatsächlich aktuell ist. Die normale Aufbauprozessur mit Host 1 als Initiator ist in ►Abbildung 6.11a dargestellt. Host 1 wählt eine Sequenznummer x und sendet ein CONNECTION REQUEST-Segment mit dieser Nummer an Host 2. Host 2 antwortet mit einem ACK-Segment, das x bestätigt und seine eigene Anfangssequenznummer y angibt. Schließlich bestätigt Host 1 die Auswahl der Anfangssequenznummer von Host 2 im ersten gesendeten Datensegment.

Wir betrachten nun, wie das Dreiwege-Handshake bei verzögerten Duplikaten von Steuersegmenten funktioniert. In ►Abbildung 6.11b ist das erste Segment ein verzögertes CONNECTION REQUEST-Duplikat aus einer alten Verbindung. Dieses Segment kommt bei Host 2 ohne Wissen von Host 1 an. Host 2 reagiert auf dieses Segment, indem er Host 1 ein ACK-Segment sendet, mit der er fragt, ob Host 1 tatsächlich versucht hat, eine neue Verbindung aufzubauen. Weist Host 1 das Anliegen von Host 2 ab, erkennt Host 2, dass er auf ein verzögertes Duplikat hereingefallen ist, und gibt den Verbindungsauflaufbau auf. Auf diese Weise können verzögerte Duplikate keine Schäden anrichten.

Der schlimmste Fall tritt ein, wenn sowohl ein verzögertes CONNECTION REQUEST als auch eine ACK im Teilnetz umhergeistern. Dies ist in ►Abbildung 6.11c der Fall. Wie im vorherigen Beispiel erhält Host 2 ein verzögertes CONNECTION REQUEST und antwortet darauf. An dieser Stelle ist entscheidend, dass Host 2 die Verwendung von y als Anfangssequenznummer für den Datenverkehr von Host 2 zu Host 1 vorgeschlagen hat, da er weiß, dass es kein Segment mit der Sequenznummer y oder Bestätigungen für y mehr gibt. Kommt das zweite verzögerte Segment bei Host 2 an, erkennt Host 2 seinerseits an der Tat-sache, dass z und nicht y bestätigt wurde, dass es sich hier ebenfalls um ein altes Duplikat handelt. Wichtig ist hier, dass es keine Kombination von alten Segmenten gibt, durch die das Protokoll einen Fehler machen und aus Versehen eine Verbindung aufbauen könnte.

TCP verwendet dieses Dreiwege-Handshake zum Aufbau von Verbindungen. Für eine Verbindung wird ein Zeitstempel verwendet, der die 32-Bit-Sequenznummer erweitert, damit sie sich innerhalb der maximalen Paketlebensdauer nicht wiederholt – nicht einmal bei Gbit/s-Verbindungen. Dieser dringend erforderliche Mechanismus behebt eine Schwäche des TCP, das zunehmend bei immer schnelleren Verbindungen eingesetzt wird.

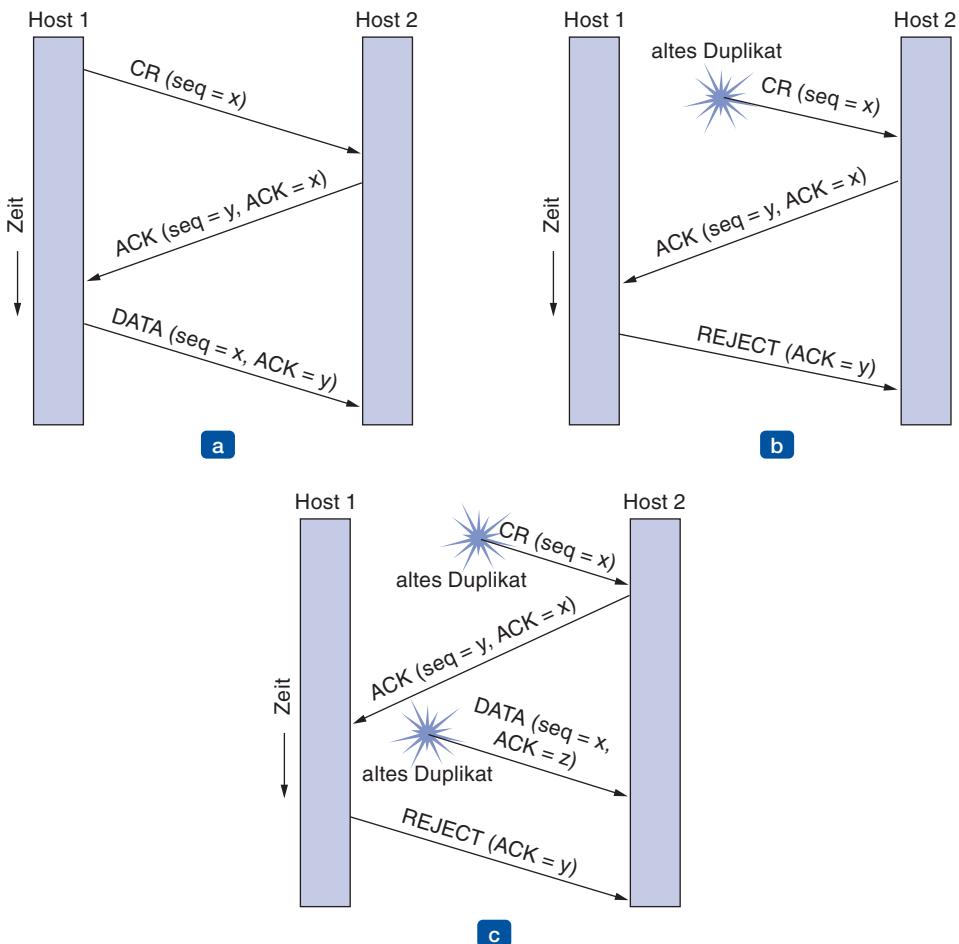


Abbildung 6.11: Drei Protokollszenarien für den Aufbau einer Verbindung mittels Dreiwege-Handshake: CR steht für CONNECTION REQUEST. (a) Normalbetrieb (b) Altes CONNECTION REQUEST-Duplikat, das plötzlich auftaucht. (c) Duplike von CONNECTION REQUEST und ACK.

Er wird in RFC 1323 beschrieben und mit **PAWS** (*Protection Against Wrapped Sequence Numbers*, Schutz gegen wiederholte Sequenznummern) abgekürzt. Bevor PAWS ins Spiel kam, verwendete TCP in seinen Verbindungen für die Anfangssequenznummern die weiter oben beschriebene taktgesteuerte Methode. Es zeigte sich jedoch, dass diese Methode mit einem Sicherheitsrisiko verbunden war. Die Taktrate erleichterte es Angreifern, die nächste Anfangssequenznummer vorherzusagen und Pakete zu senden, die den Dreiwege-Handshake austricksen und eine gefälschte Verbindung aufzubauen. Um diese Sicherheitslücke zu schließen, werden in der Praxis Pseudozufallszahlen für die Anfangssequenznummern verwendet. Es ist jedoch auch wichtig, dass sich die Anfangssequenznummern für einen Zeitraum nicht wiederholen, auch wenn sie auf den ersten Blick zufällig wirken. Andernfalls können verzögerte Duplikate Schaden anrichten.

6.2.3 Freigabe von Verbindungen

Es ist viel einfacher, eine Verbindung freizugeben als aufzubauen. Trotzdem gibt es dabei mehr Fallen als erwartet. Wie bereits erwähnt, gibt es zwei Arten, eine Verbindung freizugeben: asymmetrisch und symmetrisch. Mit einer asymmetrischen Verbindungs-freigabe haben wir es zum Beispiel beim Telefonieren zu tun: Wenn ein Teilnehmer auflegt, wird die Verbindung abgebrochen. Bei der symmetrischen Freigabe wird eine Verbindung als zwei getrennte unidirektionale Verbindungen betrachtet, die jeweils getrennt freigegeben werden müssen.

Die asymmetrische Freigabe ist abrupt und kann zu Datenverlust führen. Betrachten Sie das Szenario aus ► Abbildung 6.12. Nach Aufbau der Verbindung sendet Host 1 ein Segment, das korrekt bei Host 2 ankommt. Dann sendet Host 1 ein weiteres Segment. Leider setzt Host 2 vor Ankunft des zweiten Segments ein DISCONNECT ab. Somit wird die Verbindung abgebaut, und die Daten gehen verloren.

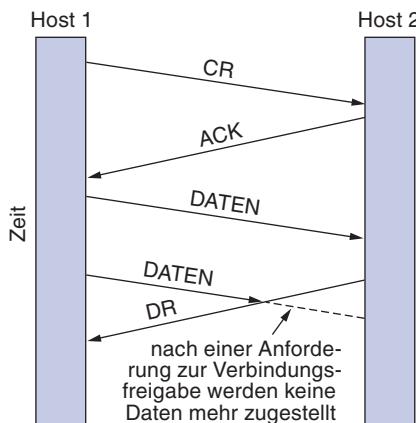


Abbildung 6.12: Abrupter Abbruch mit Datenverlust.

Um Datenverlust zu vermeiden, ist also ein anspruchsvolleres Freigabeprotokoll erforderlich. Eine Möglichkeit ist die Verwendung der symmetrischen Freigabe, bei der jede Richtung unabhängig von der anderen getrennt wird. In diesem Fall kann ein Host weiter Daten empfangen, auch wenn er bereits ein DISCONNECT-Segment ausgegeben hat.

Die symmetrische Verbindungs-freigabe eignet sich gut, wenn jeder Prozess eine feste Datenmenge zu senden hat und genau weiß, wann das Versenden abgeschlossen ist. In anderen Situationen ist das eventuell nicht so klar. Man könnte sich ein Protokoll vorstellen, in dem Host 1 sagt: „Ich bin fertig. Bist Du auch fertig?“ Wenn Host 2 antwortet: „Ich bin auch fertig“, kann die Verbindung sicher abgebaut werden.

Leider funktioniert dieses Protokoll nicht immer. Es gibt ein berühmtes Problem, das dies veranschaulicht. Man nennt es das **Zwei-Armeen-Problem**. Stellen Sie sich vor, dass die Armee der Weißröcke in einem Tal lagert (► Abbildung 6.13). Auf beiden Anhöhen steht die Armee der Blauröcke. Die weiße Armee ist zwar zahlenmäßig jeder einzelnen

blauen Truppe überlegen, aber zusammen haben die Blauröcke mehr Soldaten als die Weißrösche. Greift eine der blauen Truppen allein an, wird sie zurückgeschlagen. Wenn aber beide blauen Truppen gleichzeitig angreifen, unterliegt die weiße Armee.

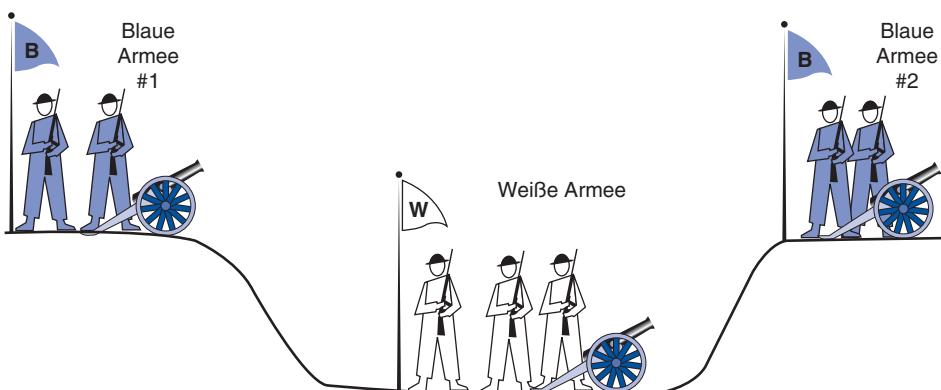


Abbildung 6.13: Das Zwei-Armeen-Problem.

Die Blauröcke möchten ihren Angriff synchronisieren. Das einzige Kommunikationsmittel sind Boten, die zu Fuß durch das Tal rennen müssen und dort unter Umständen in Gefangenschaft geraten. Eine Nachricht kann also verloren gehen. Das heißt, die Blauröcke haben einen unzuverlässigen Kommunikationskanal. Die Frage ist nun: Gibt es ein Protokoll, durch das die blaue Armee gewinnen könnte?

Angenommen, der Kommandant der blauen Truppe 1 sendet eine Nachricht mit dem Inhalt: „Ich schlage vor, im Morgengrauen des 29. März anzugreifen. Wie wär's?“ Nehmen wir weiter an, dass die Nachricht ankommt, der Kommandant der blauen Truppe 2 einverstanden ist und dessen Antwort sicher bei der blauen Truppe 1 ankommt. Findet der Angriff statt? Wahrscheinlich nicht, da der Kommandant der zweiten Truppe nicht weiß, ob sein Bote mit der Bestätigung durchgekommen ist. Ist sie nicht angekommen, greift die blaue Truppe 1 nicht mit an und er wäre dumm, sich allein in den Kampf zu stürzen.

Also verbessern wir diesen Plan, indem wir das Dreiwege-Handshake anwenden. Der Initiator des ursprünglichen Vorschlags muss die Bestätigung bestätigen. Wenn wir nun davon ausgehen, dass keine Nachricht verloren geht, erhält Truppe 2 eine Bestätigung. Jetzt beginnt jedoch der Kommandant der ersten Truppe zu zweifeln, denn er weiß nicht, ob diese Bestätigung bei Truppe 2 angekommen ist. Falls der Bote nicht durchgekommen ist, wird Truppe 2 nicht angreifen. Man könnte das Protokoll als Vierwege-Handshake auslegen, aber dies hilft hier auch nicht weiter.

Man kann also leicht nachweisen, dass es kein funktionierendes Protokoll gibt. Angenommen, es gäbe ein solches Protokoll. Entweder ist die letzte Nachricht des Protokolls wichtig oder nicht. Wenn sie nicht wichtig ist, sollte sie entfernt werden (wie auch alle anderen unnötigen Nachrichten). Zum Schluss haben wir ein Protokoll, in dem jede Nachricht wichtig ist. Was passiert nun, wenn die letzte Nachricht nicht ankommt? Wir haben ja gerade festgestellt, dass diese Nachricht wichtig ist. Wenn sie

also verloren geht, findet der Angriff nicht statt. Da der Sender der letzten Nachricht aber nie sicher sein kann, ob sie angekommen ist, wird auch er nicht angreifen. Außerdem ist sich die andere blaue Truppe darüber im Klaren; deshalb greift sie ihrerseits nicht an.

Um nun das Zwei-Armeen-Problem vom militärischen Bereich auf die Freigabe von Verbindungen zu übertragen, ersetzen wir den Begriff „angreifen“ durch „freigeben“. Wenn keiner der beiden Partner abbauen will, bevor er nicht überzeugt ist, dass der Partner ebenfalls freigeben möchte, kommt nie eine Trennung zustande.

In der Praxis können wir dieses Dilemma lösen, indem wir auf die Absprache verzichten und das Problem auf den Transportbenutzer abschieben, wobei jede Seite unabhängig von der anderen entscheidet, wann es soweit ist. Dieses Problem lässt sich leichter lösen. In ►Abbildung 6.14 werden vier Szenarien aufgezeigt, die das Dreiecks-Handshake verwenden. Dieses Protokoll ist zwar nicht unfehlbar, aber im Normalfall ausreichend.

►Abbildung 6.14a stellt den Normalfall dar. Einer der Benutzer sendet ein DISCONNECTION REQUEST-Segment (DR), um eine Verbindungs freigabe einzuleiten. Kommt sie an, sendet der Empfänger ein DR-Segment zurück und startet einen Timer für den Fall, dass sein DR-Segment verloren geht. Kommt dieses DR-Segment an, schickt der ursprüngliche Sender ein ACK-Segment zurück und löst die Verbindung. Kommt das ACK-Segment an, löst auch der Empfänger die Verbindung. Freigeben einer Verbindung bedeutet in diesem Falle, dass die Transportinstanz die Informationen über die Verbindung aus ihrer Tabelle der offenen Verbindungen entfernt und dem Verbindungseigentümer (dem Transportbenutzer) dies irgendwie signalisiert. Diese Vorgehensweise unterscheidet sich von einem Transportbenutzer, der eine DISCONNECT-Primitive absetzt.

Geht das letzte ACK-Segment verloren, wie in ►Abbildung 6.14b, rettet der Timer die Lage. Läuft er ab, wird die Verbindung auf jeden Fall freigegeben.

Betrachten wir nun den Fall, dass das zweite DR-Segment verloren geht. Das heißt, der Benutzer, der die Verbindungs freigabe eingeleitet hat, erhält nicht die erwartete Antwort, sodass der Timer abläuft und alles von vorn beginnt. In ►Abbildung 6.14c sieht man diesen Ablauf, wobei davon ausgegangen wird, dass beim zweiten Mal nichts verloren geht und alle Segmente richtig und pünktlich zugestellt werden.

Das letzte Szenario in ►Abbildung 6.14d entspricht dem von Abbildung 6.14c mit folgender Ausnahme: Jetzt wird davon ausgegangen, dass alle Wiederholungsversuche, das DR erneut zu übertragen, scheitern, weil die Segmente immer wieder verloren gehen. Nach N Versuchen gibt der Sender auf und baut die Verbindung ab. Zwischenzeitlich läuft der Timer beim Empfänger ab, sodass auch er die Verbindung freigibt.

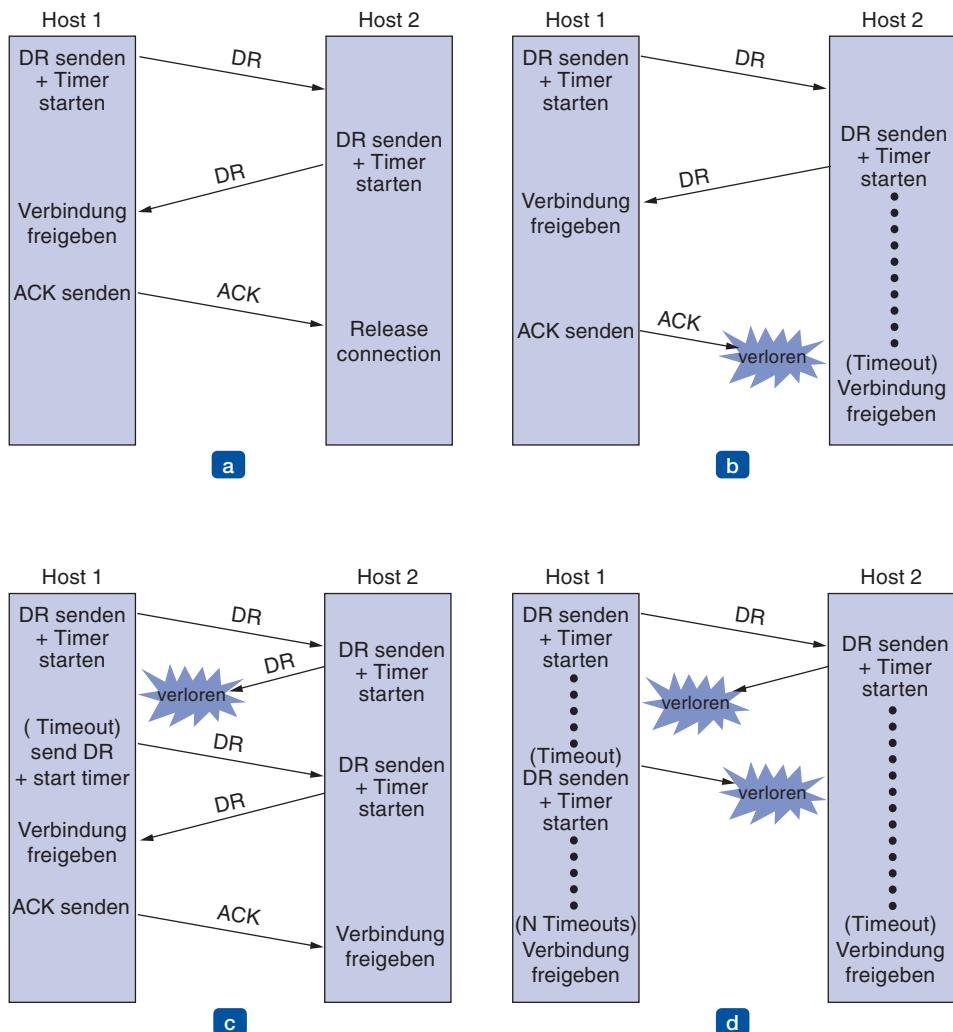


Abbildung 6.14: Verbindungsfreigabe in vier Protokollszenarien: (a) normaler Fall eines Dreiecks-Handshakes, (b) letzte Bestätigung (ACK) ging verloren, (c) Antwort ging verloren, (d) Antwort und erneut gesendete DRs gingen verloren.

Dieses Protokoll genügt normalerweise, kann theoretisch aber versagen, wenn das erste DR und N erneute Übertragungen verloren gehen. Der Sender gibt auf und baut die Verbindung ab, während die andere Seite nichts von den wiederholten Freigabevorwissen weiß und immer noch aktiv ist. Diese Situation führt zu einer halb offenen Verbindung.

Dieses Problem kann umgangen werden, wenn der Sender nach N Versuchen nicht aufgeben darf, sondern weiter senden muss, bis er eine Antwort erhält. Wenn die andere Seite die Möglichkeit eines Timeouts hat, würde der Sender bis in alle Ewigkeit weitermachen, da er nie eine Antwort erhielt. Verbietet man dem Empfänger, ein Timeout zu aktivieren, scheitert das Protokoll an der Situation von Abbildung 6.14d.

Eine Möglichkeit, halboffene Verbindungen zu beenden, ist die Aufnahme der folgenden Regel: Falls eine bestimmte Anzahl von Sekunden keine Segmente angekommen sind, wird die Verbindung automatisch abgebaut. So kann eine Seite feststellen, dass die andere Seite abgebaut hat und nicht mehr aktiv ist, und daraufhin selbst ebenfalls abbauen. Diese Regel berücksichtigt auch den Fall, dass die Verbindung unterbrochen wird (weil das Netz nicht länger Pakete zwischen den Hosts hin- und herschicken kann), ohne das irgendeine der Seiten zuerst die Verbindung abbaut. Diese Regel setzt natürlich voraus, dass jede Transportinstanz mit einem Timer ausgestattet ist, der gestoppt und immer wieder neu gesetzt wird, wenn ein Segment gesendet wird. Wenn dieser Timer abläuft, wird ein Pseudosegment übertragen, nur um die andere Seite davon abzuhalten, abzubauen. Wird andererseits die Regel für den automatischen Abbruch verwendet und gehen zu viele Pseudosegmente nacheinander über eine ansonsten inaktive Verbindung verloren, baut zuerst die eine und dann die andere Seite automatisch ab.

Wir wollen dieses Thema an dieser Stelle nicht weiter vertiefen. Es sollte aber klar sein, dass eine Verbindungs freigabe ohne Datenverlust nicht ganz so einfach ist, wie es anfangs aussieht. Die Lektion, die wir hieraus ziehen, ist, dass der Transportbenutzer an der Entscheidung, wann der Verbindungsabbruch erfolgt, beteiligt werden muss – das Problem kann von den Transportinstanzen allein nicht sauber gelöst werden. Wie wichtig diese Anwendung ist, zeigt sich, wenn Sie bedenken, dass im Gegensatz zu TCP, das eine Verbindung normalerweise symmetrisch beendet (indem jede Seite nach dem Senden der Daten seine Hälfte der Verbindung mit einem FIN-Paket abschließt), viele Webserver dem Client ein RST-Paket senden und so ein abruptes Schließen der Verbindung verursachen, was eher einem asymmetrischen Abbau entspricht. Dies funktioniert nur, weil der Webserver das Muster des Datenaustauschs kennt. Zuerst empfängt er eine Anforderung vom Client mitsamt allen Client-Daten und dann sendet er eine Antwort an den Client. Wenn der Webserver seine Antwort beendet hat, wurden die Daten in beide Richtungen gesendet. Der Server kann dem Client eine Warnung schicken und abrupt die Verbindung abbauen. Erhält der Client keine Warnung, wird er irgendwann merken, dass der Server nicht mehr mit ihm kommuniziert, und die Verbindung freigeben. In beiden Fällen wurden die Daten erfolgreich übertragen.

6.2.4 Fehlerüberwachung und Flusskontrolle

Nachdem wir den Verbindungs aufbau und die Verbindungs freigabe ausführlich besprochen haben, befassen wir uns damit, wie Verbindungen während der Benutzung verwaltet werden. Die zentralen Themen sind Fehlerüberwachung und Flusskontrolle. Fehlerüberwachung besteht darin, sicherzustellen, dass die Daten mit dem gewünschten Grad an Zuverlässigkeit übertragen werden, was in der Regel bedeutet, dass die Daten ohne Fehler übertragen werden. Bei der Flusskontrolle wird darauf geachtet, dass ein schneller Sender einen langsamen Empfänger nicht überflutet.

Beide Themen wurden bereits in Zusammenhang mit der Sicherungsschicht angeprochen. Die Lösungen, die auf der Transportschicht verwendet werden, sind die gleichen Mechanismen wie die, die wir in *Kapitel 3* beschrieben haben. Lassen Sie uns dazu kurz rekapitulieren:

1. Ein Rahmen wird mit einem Fehlererkennungscode versehen (z.B. einem CRC-Wert oder einer Prüfsumme), mit dem geprüft wird, ob die Informationen korrekt empfangen wurden.
2. Ein Rahmen wird zur Identifikation mit einer Sequenznummer versehen und vom Sender noch einmal übertragen, bis er vom Empfänger eine Bestätigung über den erfolgreichen Empfang erhält. Dies wird als **ARQ** bezeichnet (*Automatic Repeat Request*, automatische Wiederholungsanfrage).
3. Es gibt eine maximale Anzahl an Rahmen, die der Sender zu jeder Zeit als Dateneinheit losschicken kann und die warten müssen, wenn der Empfänger die Rahmen nicht schnell genug bestätigt. Ist das Maximum ein Paket, wird das Protokoll als **Stop-and-Wait** bezeichnet. Größere Fenster erlauben Pipelining und verbessern die Leistung auf langen, schnellen Verbindungen.
4. Das **Schiebefensterprotokoll** kombiniert diese Funktionen und wird auch zur Unterstützung von bidirektonaler Datenübertragung verwendet.

In Anbetracht der Tatsache, dass diese Mechanismen für Rahmen auf der Sicherungsschicht gedacht sind, fragt man sich natürlich, warum sie auch für Segmente auf der Transportschicht verwendet werden. In der Praxis weisen Sicherungs- und Transportschicht kaum Überschneidungen auf. Auch wenn die gleichen Mechanismen verwendet werden, gibt es Unterschiede in Funktion und Umfang.

Um Unterschiede in der Funktion zu veranschaulichen, betrachten wir die Fehlererkennung. Die Prüfsumme der Sicherungsschicht schützt einen Rahmen nur während der Übertragung auf einer einzelnen Verbindung. Die Prüfsumme auf der Transportschicht hingegen schützt ein Segment, während es einen ganzen Netzpfad, d.h. mehrere Verbindungen durchläuft. Es ist eine Ende-zu-Ende-Prüfung, was nicht das Gleiche ist wie eine Prüfung auf jeder Verbindung. Saltzer et al. (1984) beschreiben eine Situation, in der Pakete in einem Router beschädigt wurden. Die Prüfsummen der Sicherungsschicht schützen die Pakete nur während ihrer Reise über die Verbindung und nicht im Router. Deshalb wurden die Pakete falsch zugestellt, obwohl sie laut Prüfungen auf jeder Verbindung korrekt waren.

Dieses und andere Beispiele haben Saltzer et al. zu der Formulierung des **Ende-zu-Ende-Arguments** veranlasst. Laut diesem Argument ist die Ende-zu-Ende-Prüfung auf der Transportschicht unerlässlich für die Korrektheit; die Prüfungen auf der Sicherungsschicht hingegen sind zwar nicht unerlässlich, aber für die Leistungsverbesserung ganz nützlich (da ansonsten ein beschädigtes Paket unnötig auf den ganzen Pfad gesendet werden kann).

Als Beispiel für den unterschiedlichen Umfang betrachten wir die Neuübertragung und das Schiebefensterprotokoll. Auf den meisten drahtlosen Verbindungen, abgesehen von den Satellitenverbindungen, kann nur ein Rahmen zurzeit vom Sender auf den Weg geschickt werden. Das heißt, das Bandbreite-Verzögerung-Produkt für die Verbindung ist klein genug, dass nicht einmal ein ganzer Rahmen in der Verbindung gespeichert werden kann. In diesem Fall ist eine kleine Fenstergröße für gute Leistung ausreichend. Zum Beispiel verwendet IEEE 802.11 ein Stop-and-Wait-Protokoll, das jeden Rahmen überträgt bzw. erneut überträgt und darauf wartet, dass es bestätigt wird, bevor es den nächsten Rahmen sendet. Ist die Fenstergröße größer als ein Rahmen, wäre alles nur komplizierter, ohne dass die Leistung sich verbessert. Für Kabel- und Glasfaserverbindungen, wie (Switched) Ethernet oder ISP-Backbones, ist die Fehlerrate niedrig genug, dass auf die erneute Übertragung auf der Sicherungsschicht verzichtet werden kann, weil die Ende-zu-Ende-Neuübertragungen den übrig bleibenden Rahmenverlust ausgleichen.

Viele TCP-Verbindungen hingegen haben ein Bandbreite-Verzögerung-Produkt, das viel größer ist als ein einzelnes Segment. Betrachten wir beispielsweise eine Verbindung, die Daten mit 1 Mbit/s und einer Paketumlaufzeit von 100 ms quer durch die Vereinigten Staaten schickt. Sogar bei einer so langsamen Verbindung werden beim Empfänger für die Dauer des Sendens eines Segments und des Empfangs einer Bestätigung 200 KB Daten gespeichert. In solchen Situationen muss ein großes Schiebefenster verwendet werden. Stop-and-Wait würde die Leistung beeinträchtigen. In unserem Beispiel würde es die Leistung auf ein Segment alle 200 ms (oder 5 Segmente/s) begrenzen, egal wie schnell das Netz wirklich ist.

In Anbetracht der Tatsache, dass Transportprotokolle normalerweise größere Schiebefenster verwenden, werden wir das Thema Datenpufferung eingehender behandeln. Da ein Host mehrere Verbindungen haben kann, die jeweils getrennt behandelt werden, benötigt er für die Schiebefenster eventuell erhebliche Pufferkapazitäten. Die Puffer werden sowohl beim Sender als auch beim Empfänger benötigt. Der Puffer beim Sender dient dazu, alle übertragenen, aber noch nicht bestätigten Segmente zu speichern. Dies ist nötig, da diese Segmente verloren gehen können und erneut übertragen werden müssen.

Da der Sender die Pufferung übernimmt, kann der Empfänger je nach Bedarf entscheiden, ob spezielle Verbindungen Puffer erhalten oder nicht. Der Empfänger kann beispielsweise einen einzigen Pufferpool einrichten, der von allen Verbindungen genutzt wird. Wenn ein Segment eintrifft, wird versucht, dynamisch einen neuen Puffer anzufordern. Ist einer verfügbar, wird das Segment akzeptiert; andernfalls wird es verworfen. Da der Sender darauf vorbereitet ist, Segmente, die im Netz verloren gegangen sind, erneut zu übertragen, verursacht es keinen dauerhaften Schaden, wenn der Empfänger Segmente fallen lässt, auch wenn einige Ressourcen dabei verschwendet werden. Der Sender versucht es einfach weiter, bis er eine Bestätigung erhält.

Der optimale Kompromiss zwischen der Pufferung bei Quelle und Ziel hängt von der Art des Verkehrs ab, den die Verbindungen übertragen. Bei schubweisem Verkehr (Bursts) mit geringer Bandbreite, wie er von einem interaktiven Terminal produziert

wird, ist es besser, keine Puffer zuzuteilen, sondern an beiden Endpunkten Pufferplatz dynamisch anzufordern und sich auf die Pufferung beim Sender zu verlassen, falls Segmente gelegentlich verworfen werden müssen. Andererseits ist es für die Dateiübertragung und anderen Datenverkehr mit hoher Bandbreite besser, wenn der Empfänger ein ganzes Fenster mit Puffern zuweist, damit die Daten mit maximaler Geschwindigkeit fließen können. Diese Strategie wird vom TCP-Protokoll verfolgt.

Es bleibt noch die Frage offen, wie der Pufferpool organisiert werden sollte. Haben die meisten Segmente fast die gleiche Länge, bietet es sich an, den Pufferbereich als Pool mit gleich großen Puffern anzulegen, aus dem jedes Segment einen Puffer erhält (►Abbildung 6.15a). Wenn die Größe der Segmente stark schwankt (von kurzen Anfragen von Webseiten bis zu großen Paketen bei Dateiübertragungen zwischen gleichberechtigten Partnern), ist ein Pufferpool mit Puffern fester Größe problematisch. Wird die Puffergröße gleich dem größtmöglichen Segment gewählt, verschwendet ein kurzes Segment zu viel Speicherplatz. Wenn die Puffergröße kleiner als die größte Segmentlänge gewählt wird, werden für lange Segmente mehrere Puffer benötigt, was zu einer entsprechenden Komplexität führt.

Ein anderer Ansatz zur Behandlung des Puffergrößenproblems ist die Verwendung von Puffern variabler Größe (►Abbildung 6.15b). Diese Methode hat den Vorteil, dass der Speicher besser genutzt wird – auf Kosten einer wesentlich komplizierteren Pufferverwaltung. Als dritte Möglichkeit kann jeder Verbindung ein großer Ringpuffer zugewiesen werden (►Abbildung 6.15c). Dieses System ist einfach, elegant, hängt nicht von den Segmentgrößen ab und nutzt den Speicher nur, wenn die Verbindungen stark belastet werden.

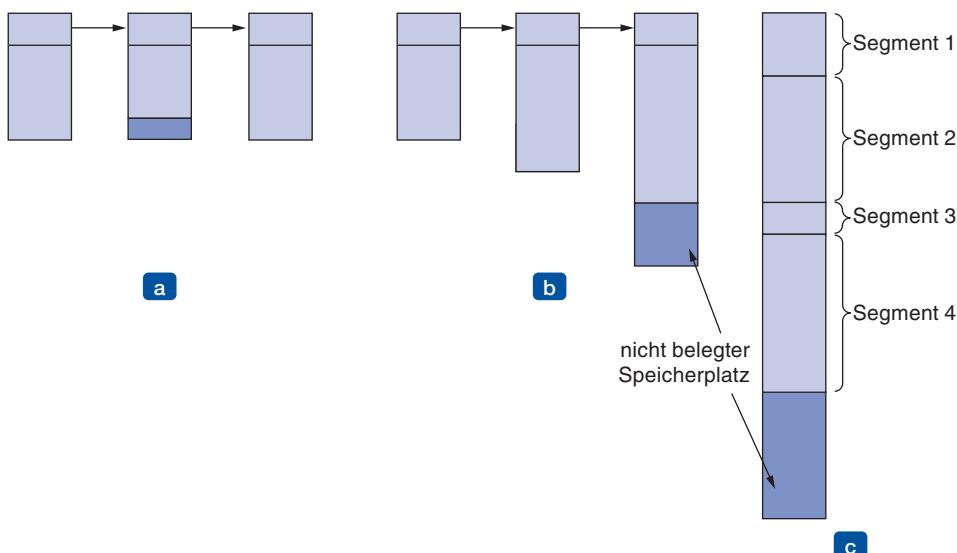


Abbildung 6.15: (a) Verkettete Puffer fester Größe. (b) Verkettete Puffer variabler Größe.
(c) Ein großer Ringpuffer pro Verbindung.

Wenn Verbindungen geöffnet oder geschlossen werden oder sich die Struktur des Datenverkehrs ändert, müssen Sender und Empfänger ihre Pufferzuordnungen dynamisch anpassen. Ein Transportprotokoll sollte also dem sendenden Host erlauben, Pufferplatz am anderen Endpunkt anzufordern. Puffer können pro Verbindung oder gemeinsam für alle Verbindungen zwischen zwei Hosts zugewiesen werden. Andererseits könnte der Empfänger, der zwar seine eigene Puffersituation kennt, aber nichts über den bevorstehenden Datenverkehr weiß, dem Sender mitteilen: „Ich habe X Puffer für dich reserviert.“ Wenn die Anzahl der geöffneten Verbindungen steigt, muss die Zuordnung möglicherweise eingeschränkt werden. Das Protokoll sollte diese Möglichkeit zulassen.

Eine vernünftige allgemeine Vorgehensweise bei der dynamischen Pufferzuweisung ist, Puffer und Bestätigungen voneinander zu trennen, womit sie sich von den Schiebefensterprotokollen in *Kapitel 3* unterscheidet. Die dynamische Pufferverwaltung funktioniert eigentlich wie ein Fenster mit veränderlicher Größe. Zuerst fordert der Sender für sein voraussichtliches Datenaufkommen eine bestimmte Anzahl von Puffern an. Daraufhin stellt der Empfänger so viele Puffer zur Verfügung, wie er beschaffen kann. Jedes Mal, wenn der Sender ein Segment sendet, muss er seine Zuordnung reduzieren, bis diese den Stand null erreicht hat. Dann schickt der Empfänger sowohl Bestätigungen als auch Pufferzuordnungen getrennt im Huckepackverfahren auf den Rückweg. TCP verwendet dieses Verfahren und überträgt die Pufferzuordnungen in einem Header-Feld namens *Window Size*.

► Abbildung 6.16 zeigt eine Möglichkeit, wie die dynamische Fensterverwaltung in einem Datagrammnetz mit 4-Bit-Sequenznummern funktionieren könnte. In diesem Beispiel fließen die Daten in Segmenten von Host A nach Host B und die Bestätigungen und Pufferzuordnungen in Segmenten in die umgekehrte Richtung. Anfangs fordert A acht Puffer an, B stellt allerdings nur vier zur Verfügung. Danach sendet A drei Segmente, von denen das letzte verloren geht. Segment 6 bestätigt den Erhalt aller Segmente bis einschließlich Sequenznummer 1. Damit kann A diese Puffer freigeben. Außerdem wird A durch Segment 6 darüber informiert, dass er nun drei weitere Segmente mit Sequenznummern größer 1 (also Segment 2, 3 und 4) senden kann. A weiß, dass er Nummer 2 bereits verschickt hat, und meint daher, er dürfe nun Segment 3 und 4 senden, was er auch tut. Hier wird A blockiert und muss auf eine weitere Pufferzuordnung warten. Erneute Übertragungen aufgrund eines Timeouts (Zeile 9) können vorkommen, während A blockiert ist, da diese bereits zugewiesene Puffer verwenden. In Zeile 10 bestätigt B, dass alle Segmente bis einschließlich 4 eingetroffen sind, lässt A jedoch nicht weitersenden. Diese Situation wäre bei den Protokollen mit festen Fenstern von *Kapitel 3* unmöglich. Das nächste Segment von B nach A weist einen neuen Puffer zu. Damit kann A weitersenden. Dies passiert, wenn B Pufferkapazität hat – wohl weil der Transportbenutzer mehr Segmentdaten akzeptiert hat.

| <u>A</u> | <u>Nachricht</u> | <u>B</u> | <u>Kommentare</u> |
|----------|----------------------|----------|--|
| 1 → | < request 8 buffers> | → | A möchte 8 Puffer. |
| 2 ← | <ack = 15, buf = 4> | ← | B lässt nur Nachrichten 0-3 zu. |
| 3 → | <seq = 0, data = m0> | → | A hat jetzt noch 3 Puffer übrig. |
| 4 → | <seq = 1, data = m1> | → | A hat jetzt noch 2 Puffer übrig. |
| 5 → | <seq = 2, data = m2> | ... | Nachricht verloren, aber A denkt, er habe noch 1 übrig. |
| 6 ← | <ack = 1, buf = 3> | ← | B bestätigt 0 und 1, und erlaubt 2-4. |
| 7 → | <seq = 3, data = m3> | → | A hat noch 1 Puffer übrig. |
| 8 → | <seq = 4, data = m4> | → | A hat 0 Puffer übrig und muss aufhören. |
| 9 → | <seq = 2, data = m2> | → | A hat die Zeit überschritten und wiederholt die Übertragung. |
| 10 ← | <ack = 4, buf = 0> | ← | Alles bestätigt, aber A ist noch blockiert. |
| 11 ← | <ack = 4, buf = 1> | ← | A kann jetzt 5 senden. |
| 12 ← | <ack = 4, buf = 2> | ← | B hat irgendwo einen neuen Puffer gefunden. |
| 13 → | <seq = 5, data = m5> | → | A hat 1 Puffer übrig. |
| 14 → | <seq = 6, data = m6> | → | A ist wieder blockiert. |
| 15 ← | <ack = 6, buf = 0> | ← | A ist immer noch blockiert. |
| 16 ... | <ack = 6, buf = 4> | ← | Potenzieller Deadlock. |

Abbildung 6.16: Dynamische Pufferzuweisung: Die Pfeile geben die Übertragungsrichtung an; (...) steht für ein verlorenes Segment.

Probleme mit Pufferzuordnungen dieser Art können in Datagrammnetzen auftreten, wenn dort Steuersegmente verloren gehen – was häufig der Fall ist. Betrachten wir Zeile 16. *B* hat für *A* weitere Puffer zugeordnet, aber das Zuordnungssegment ging verloren. Da es bei Zuordnungssegmenten weder Flusskontrolle noch Timeout gibt, befindet sich *A* nun in einem Deadlock. Um diese Situation zu verhindern, sollte jeder Host in regelmäßigen Abständen Steuersegmente aussenden, die den Bestätigungs- und Pufferstatus jeder Verbindung angeben. So wird der Deadlock früher oder später beseitigt.

Bisher sind wir stillschweigend davon ausgegangen, dass die Datenübertragungsrate des Senders nur durch die Puffergröße beim Empfänger begrenzt ist. Dies ist jedoch nicht oft der Fall. Speicherkapazität war früher einmal sehr teuer, doch inzwischen sind die Preise dafür drastisch gesunken, sodass es durchaus im Rahmen des Möglichen ist, Hosts mit so viel Speicher auszustatten, dass Puffermangel kein bzw. ein seltenes Thema ist, nicht einmal für WAN-Verbindungen. Dies setzt natürlich voraus, dass die gesetzte Puffergröße groß genug ist, was bei TCP nicht immer der Fall war (Zhang et al., 2002).

Wenn die Pufferkapazität den maximalen Datenfluss nicht mehr einschränkt, taucht ein anderer Engpass auf: die Übertragungsleistung des Netzes. Wenn benachbarte Router maximal x Pakete pro Sekunde austauschen können und k disjunkte Pfade zwischen einem Host-Paar bestehen, haben die beiden Hosts keine Möglichkeit, mehr als kx Segmente pro Sekunde auszutauschen, gleichgültig, wie viel Pufferkapazität ihnen an beiden Endpunkten zur Verfügung steht. Wenn der Sender zu sehr drängt (also mehr als kx Segmente pro Sekunde sendet), ist das Netz überlastet und kann die Segmente nicht so schnell abliefern, wie sie hereinkommen.

Sinnvoll ist hier also ein Mechanismus, der die Übertragungen des Senders auf der Grundlage der Übertragungsleistung des Netzes, nicht auf der Pufferleistung des Emp-

fängers basiert. Belsnes (1975) schlug vor, ein Schiebefensterprotokoll mit Flusskontrolle zu verwenden, bei dem der Sender die Fenstergröße dynamisch an die Übertragungsleistung des Netzes anpasst. Das bedeutet, dass ein dynamisches Schiebefenster sowohl Flusskontrolle als auch Überlastungsüberwachung implementieren kann. Wenn das Netz c Segmente pro Sekunde bewältigen kann und die Paketumlaufzeit (einschließlich Übertragung, Weiterleitung, Stauen, Verarbeitung beim Empfänger und Bestätigung) r beträgt, sollte das Fenster des Senders cr sein. Hat das Fenster diese Größe, arbeitet der Sender unter voller Ausnutzung der Leitung. Eine kleine Verminde rung der Netzeistung hätte schon zur Folge, dass der Sender blockiert. Da die Leistung des Netzes vom Verkehrsaufkommen abhängt, sollte die Fenstergröße häufig angepasst werden, damit eventuelle Änderungen der Übertragungskapazität berücksichtigt werden. Wie später noch aufgezeigt wird, benutzt TCP ein ähnliches Verfahren.

6.2.5 Multiplexing

In einigen Schichten der Netzarchitektur ist es nötig, mehrere Kommunikationsvorgänge auf konkreten oder virtuellen Verbindungen und physikalischen Leitungen zu multiplexen. Auf der Transportschicht kann Multiplexing aus verschiedenen Gründen notwendig werden. Ein Beispiel: Ist auf dem Host nur eine Netzadresse verfügbar, muss sie von allen Transportverbindungen auf diesem Rechner verwendet werden. Kommt ein Segment an, muss es eine Methode geben, um anzugeben, an welchen Prozess sie übergeben werden soll. Diese Situation nennt man **Multiplexing** oder **Upward-Multiplexing** (► Abbildung 6.17a). In dieser Abbildung verwenden vier verschiedene Transportverbindungen die gleiche Netzverbindung (z.B. IP-Adresse) zum entfernten Host.

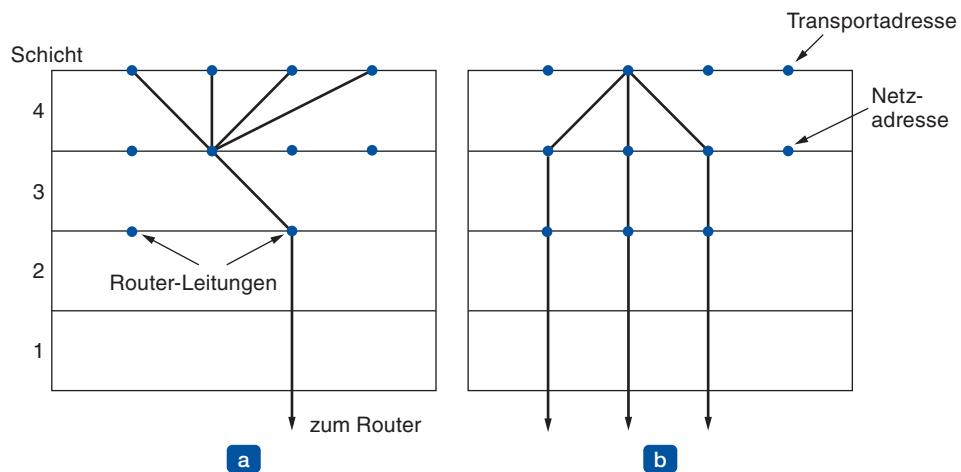


Abbildung 6.17: (a) Multiplexing. (b) Inverses Multiplexing.

Multiplexen kann auch noch aus einem anderen Grund auf der Transportschicht nützlich sein. Angenommen, ein Host hat mehrere Netzpfade, die er verwenden kann. Wenn nun ein Benutzer mehr Bandbreite benötigt oder eine höhere Zuverlässigkeit,

als einer der Netzpfade bieten kann, besteht ein Ausweg darin, eine Verbindung aufzubauen, die den Verkehr im Rotationsverfahren auf mehrere Netzpfade verteilt (►Abbildung 6.17b). Dieses Vorgehen wird **inverses Multiplexing** oder **Downward-Multiplexing** genannt. Bei k offenen Netzverbindungen erhöht sich die effektive Bandbreite um den Faktor k . Ein Beispiel für inverses Multiplexing ist das **SCTP** (*Stream Control Transmission Protocol*), das eine Verbindung aufbauen kann, die mehrere Netzchnittstellen verwendet. Im Gegensatz dazu verwendet das TCP nur einen Netzendpunkt. Inverses Multiplexing ist auch auf der Sicherungsschicht zu finden, auf der mehrere parallele Verbindungen geringer Übertragungsrate zu einer Verbindung hoher Übertragungsrate zusammengefasst werden.

6.2.6 Systemwiederherstellung

Da Hosts und Router absturzgefährdet oder die Verbindungen langlebig sind (z.B. große Software- oder Media-Downloads), ist das Thema Wiederherstellung nach Systemabstürzen wichtig. Befindet sich die Transportinstanz gänzlich in den Hosts, ist die Wiederherstellung von Netz- und Routerabstürzen nicht problematisch. Die Transportinstanzen rechnen immer mit verlorenen Segmenten und begegnen diesem Problem mit Neuübertragungen.

Etwas mühsamer ist die Wiederherstellung nach Host-Abstürzen. Insbesondere möchten die Clients weiterarbeiten können, wenn Server abstürzen und neu hochfahren. Um das zu veranschaulichen, nehmen wir an, dass ein Host (Client) eine große Datei über ein einfaches Stop-and-Wait-Protokoll an einen anderen Host (Dateiserver) sendet. Die Transportschicht im Server leitet die ankommenden Segmente nacheinander an den Transportbenutzer weiter. Mitten in der Übertragung bricht der Server zusammen. Er fährt zwar sofort wieder hoch, aber seine Tabellen wurden dabei neu initialisiert, sodass er nicht mehr genau weiß, an welcher Stelle er unterbrochen wurde.

Bei dem Versuch, seinen vorherigen Zustand wiederherzustellen, kann der Server ein Broadcast-Segment an alle anderen Hosts senden. Er teilt ihnen mit, dass er abgestürzt ist, und fordert die Clients auf, ihn über den Status aller offenen Verbindungen zu informieren. Jeder Client kann sich in einem von zwei Zuständen befinden: Ein Segment steht aus ($S1$) oder kein Segment steht aus ($S0$). Allein anhand dieser Zustandsinformationen muss der Client entscheiden, ob das letzte Segment erneut übertragen werden muss.

Auf den ersten Blick scheint es klar, dass der Client nur dann nochmals senden sollte, wenn ihm zu dem Zeitpunkt, an dem er von dem Absturz erfährt, eine Bestätigung für ein Segment fehlt (also Zustand $S1$). Bei näherer Betrachtung zeigt dieser etwas naive Vorschlag jedoch Schwächen. Stellen Sie sich z.B. die Situation vor, in der die Transportinstanz des Servers zuerst eine Bestätigung abschickt und dann – nachdem die Bestätigung versendet wurde – in den Anwendungsprozess schreibt. Ein Segment in den Ausgangstrom zu schreiben und eine Bestätigung zu senden, sind zwei getrennte Vorgänge, die nicht gleichzeitig erfolgen können. Ereignet sich der Absturz, nachdem die Bestätigung geschickt, aber bevor der Schreibvorgang beendet wurde, erhält der Client die Bestätigung und ist dann in Zustand $S0$, wenn die Mitteilung über die

Absturzwiederherstellung eintrifft. Der Client überträgt das Segment nicht noch einmal, weil er irrtümlich annimmt, dass das Segment angekommen ist. Diese Entscheidung führt zu einem fehlenden Segment.

An dieser Stelle denken Sie vielleicht: „Das Problem ist doch ganz einfach zu beheben. Wir programmieren einfach die Transportinstanz um, damit sie zuerst den Schreibvorgang vornimmt und dann die Bestätigung losschickt.“ Spielen wir das einmal durch: Der Schreibvorgang ist erfolgt, aber der Server stürzt ab, bevor die Bestätigung gesendet werden konnte. Der Client befindet sich nun in Zustand S_1 und sendet daher erneut, was ein unentdecktes Segmentdoppelat im Ausgangsstrom zum Server-Anwendungsprozess zur Folge hat.

Unabhängig davon, wie Sender und Empfänger programmiert sind, gibt es immer Situationen, in denen das Protokoll nicht richtig wiederhergestellt werden kann. Der Server kann auf zwei Arten programmiert sein: zuerst bestätigen oder zuerst schreiben. Der Client kann auf vier Arten programmiert sein: Er überträgt immer das letzte Segment neu, er überträgt nie das letzte Segment neu, er überträgt nur im Zustand S_0 neu oder er überträgt nur im Zustand S_1 neu. Das ergibt acht Kombinationen. Wir werden gleich sehen, dass es bei jeder Kombination Abläufe gibt, die zum Scheitern des Protokolls führen.

Beim Server sind drei Ereignisse möglich: eine Bestätigung senden (A), in den Ausgangsprozess schreiben (W) und abstürzen (C). Die drei Ereignisse können in sechs verschiedenen Abfolgen auftreten: $AC(W)$, AWC , $C(AW)$, $C(WA)$, WAC und $WC(A)$. Die Klammern zeigen an, dass weder A noch W nach C kommen kann (nach einem Absturz geht nichts mehr). ► Abbildung 6.18 zeigt alle acht Kombinationen der Client- und Server-Strategien und die gültigen Ereignisfolgen. Für jede Strategie gibt es eine Folge von Ereignissen, die das Protokoll zum Scheitern verurteilt. Wenn der Client z.B. immer erneut überträgt, erzeugt das AWC -Ereignis ein unentdecktes Duplikat, auch wenn die anderen zwei Ereignisse korrekt ablaufen.

| | | Strategie des empfangenden Hosts | | | | | | |
|----------------------------------|--|--------------------------------------|-----|---------|--------------------------------------|------|-------|--|
| | | zuerst Bestätigen, dann Schreiben | | | zuerst Schreiben, dann Bestätigen | | | |
| Strategie des sendenden Hosts | | AC(W) | AWC | C(AW) | C(WA) | W AC | WC(A) | |
| | | OK | DUP | OK | OK | DUP | DUP | |
| Immer erneut übertragen | | Verlust | OK | Verlust | Verlust | OK | OK | |
| Nie erneut übertragen | | OK | DUP | Verlust | Verlust | DUP | OK | |
| Bei S_0 erneut übertragen | | Verlust | OK | OK | OK | OK | DUP | |
| Bei S_1 erneut übertragen | | | | | | | | |

OK = Protokoll arbeitet korrekt
 DUP = Protokoll erzeugt Duplikatnachricht
 Verlust = Protokoll verliert eine Nachricht

Abbildung 6.18: Verschiedene Kombinationen der Client- und Server-Strategien.

Auch ein weiteres Feilen am Protokoll bringt hier keine Abhilfe. Sogar wenn Client und Server mehrere Segmente austauschen, bevor der Server schreibt, sodass der Client genau weiß, was passiert, hat der Client keine Möglichkeit, festzustellen, ob ein Absturz vor oder nach dem Schreibvorgang passiert ist. Wir schließen daraus: Nach unserer Grundregel, dass Ereignisse nicht gleichzeitig stattfinden (das heißt, getrennte Ereignisse passieren hintereinander), sind der Absturz und die Wiederherstellung eines Hosts für die höheren Schichten nicht transparent.

Allgemeiner ausgedrückt, kann diese Schlussfolgerung auch so formuliert werden: Eine Wiederherstellung nach einem Absturz in Schicht N kann nur in Schicht $N+1$ erfolgen und dann nur, falls die höhere Schicht ausreichend Statusinformationen hat, um den Zustand so wiederherzustellen, wie er vor dem Problem war. Dies stimmt mit dem oben erwähnten Fall überein, dass die Transportschicht den Zustand nach Ausfällen in der Vermittlungsschicht wiederherstellen kann, sofern beide Endpunkte einer Verbindung sich merken, wo sie gerade sind.

Diese Problematik führt uns zu folgendem Thema: Was bedeutet eigentlich eine sogenannte Ende-zu-Ende-Bestätigung? Im Prinzip ist das Transportprotokoll ein Ende-zu-Ende-Protokoll, das nicht wie die Protokolle der unteren Schichten verkettet ist. Nun nehmen wir den Fall an, dass ein Benutzer Transaktionen bei einer entfernten Datenbank anfordert. Die entfernte Transportinstanz ist so programmiert, dass sie die Segmente zuerst zur nächsthöheren Schicht weitergibt und dann bestätigt. Sogar in dieser Situation bedeutet der Empfang der Bestätigung beim Rechner des Benutzers nicht unbedingt, dass der entfernte Host so lange aktiv war, dass er die Datenbank auch wirklich aktualisieren konnte. Eine echte Ende-zu-Ende-Bestätigung, die besagt, dass die angeforderte Arbeit auch tatsächlich ausgeführt wurde, und deren Abwesenheit bedeutet, dass die Arbeit nicht ausgeführt wurde, ist vielleicht etwas Unerreichbares. Dieses Thema wird bei Saltzer et al. (1984) ausführlich behandelt.

6.3 Überlastungsüberwachung

Wenn die Transportinstanzen auf vielen Rechnern zu viele Pakete zu schnell ins Netz stellen, kommt es zu einer Überlastung des Netzes, die mit einer Verschlechterung der Leistung einhergeht, weil Pakete sich verzögern oder verloren gehen. Zur Vermeidung dieses Problems muss die Überlastung überwacht werden, wofür die Vermittlungs- und die Transportschicht zuständig sind. Die Überlastung tritt in den Routern auf, sodass sie auf der Vermittlungsschicht erkannt wird. Verursacht wird sie jedoch letztlich durch den Verkehr, der im Netz auf der Transportschicht erzeugt wird. Der einzige effektive Weg, die Überlastung zu überwachen, besteht darin, die Pakete mithilfe der Transportprotokolle langsamer in das Netz zu stellen.

In Kapitel 5 haben wir Mechanismen zur Überlastungsüberwachung auf der Vermittlungsschicht untersucht. In diesem Abschnitt wenden wir uns der anderen Hälfte des Problems zu: der Überlastungsüberwachung auf der Transportschicht. Nachdem wir die Ziele der Überlastungsüberwachung vorgestellt haben, beschreiben wir, wie Hosts

die Senderate für ihre Pakete im Netz regulieren. Das Internet verlässt sich stark darauf, dass die Transportschicht die Überlastungsüberwachung übernimmt. Zu diesem Zweck sind bestimmte Algorithmen in TCP und anderen Protokollen integriert.

6.3.1 Gewünschte Bandbreitenzuordnung

Bevor wir beschreiben, wie Sie den Verkehr regulieren, müssen Sie verstehen, was mit der Ausführung eines Überlastungsüberwachungsalgorithmus erreicht werden soll. Das heißt, wir müssen den Zustand angeben, in dem ein guter Algorithmus zur Überwachung der Überlastung im Netz zum Einsatz kommt. Ziel ist es jedenfalls nicht, einfach nur die Überlastung zu vermeiden. Es wird angestrebt, die Bandbreite optimal auf die das Netz verwendenden Transportinstanzen zu verteilen. Eine gute Zuordnung bedeutet auch gleichzeitig eine gute Leistung, da sie die gesamte zur Verfügung stehende Bandbreite nutzt, während sie gleichzeitig Überlastung vermeidet; sie behandelt alle konkurrierenden Transportinstanzen gleich und erkennt schnell Änderungen im Verkehrsaufkommen. Wir werden nach und nach auf diese Kriterien genauer eingehen.

Effizienz und Leistungsvermögen

Bei einer effizienten Zuordnung der Bandbreite auf die Transportinstanzen wird die verfügbare Netzkapazität vollständig genutzt. Sie sollten dabei jedoch nicht automatisch annehmen, dass auf einer 100 Mbit/s-Verbindung mit fünf Transportinstanzen jede davon jeweils 20 Mbit/s erhält. Für eine gute Leistung steht einer Transportinstanz in der Regel weniger als 20 Mbit/s zur Verfügung. Grund ist, dass der Verkehr oft schubweise erfolgt. Erinnern wir uns, dass in Abschnitt 5.3 der **Datendurchsatz** (oder Rate der Nutzpakete, die beim Empfänger ankommen) als eine Funktion der gegebenen Last beschrieben wurde. Diese Kurve und eine zugehörige Kurve für die Verzögerung als Funktion der gegebenen Last finden Sie in ▶ Abbildung 6.19.

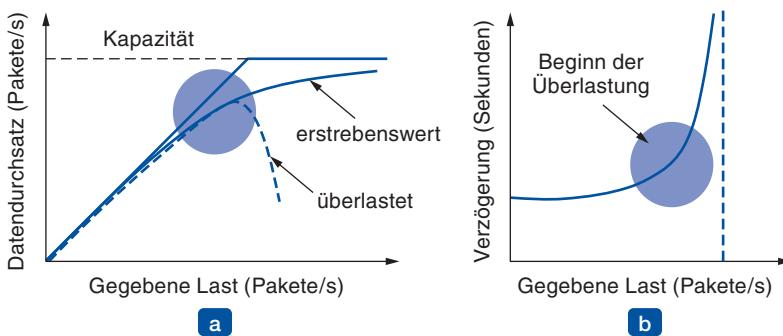


Abbildung 6.19: (a) Datendurchsatz und (b) Verzögerung als Funktion der gegebenen Last.

Der Datendurchsatz erhöht sich am Anfang mit gleicher Rate wie die Last (▶ Abbildung 6.19a). Doch wenn sich die Last der Kapazitätsgrenze nähert, steigt der Datendurchsatz nur noch allmählich an. Die Abflachung der Kurve ist darauf zurückzuführen, dass Verkehrsschübe sich gelegentlich akkumulieren und dadurch einige Pufferverluste im

Netz verursachen. Wenn das Transportprotokoll mit Mängeln behaftet ist und Pakete neu überträgt, die nur verzögert, aber nicht verloren sind, kann das Netz vor Überlastung zusammenbrechen. In diesem Zustand werden vom Sender wie wild Pakete gesendet, während gleichzeitig immer weniger nützliche Arbeit erledigt wird.

Die entsprechende Verzögerung im Netz können Sie ▶ Abbildung 6.19b entnehmen. Am Anfang ist die Verzögerung fest und entspricht der Ausbreitungsverzögerung im Netz. Wenn die Last sich der Kapazitätsgrenze nähert, steigt die Verzögerung an – zuerst ganz langsam und dann immer stärker. Auch dies liegt daran, dass Verkehrs schübe unter hoher Last zu langen Warteschlangen führen. Die Verzögerung kann dabei nie wirklich unendlich groß werden, außer in einem Modell mit unendlichem Puffer für die Router. Stattdessen gehen bei maximaler Pufferverzögerung Pakete verloren.

Sobald die Überlastung einsetzt, verschlechtert sich der Datendurchsatz und die Verzö gerung nimmt zu. Gesunder Menschenverstand sagt uns, dass wir die beste Netzteistung erhalten, wenn wir die Bandbreite so lange zuordnen, bis die Verzögerung beginnt, stark anzusteigen. Dieser Punkt liegt unter der Kapazitätsgrenze. Um ihn zu ermitteln, schlug Kleinrock (1979) folgende Metrik des **Leistungsvermögens** (*power*) vor:

$$\text{Leistungsvermögen} = \frac{\text{Last}}{\text{Verzögerung}}$$

Solange die Verzögerung klein und relativ konstant ist, steigt das Leistungsvermögen am Anfang mit der gegebenen Last, sobald die Verzögerung jedoch stark ansteigt, erreicht sie ein Maximum und fällt. Die Last mit dem höchsten Leistungsvermögen entspricht der effizienten Last, die die Transportinstanz ins Netz stellen sollte.

Max-Min-Fairness

In den obigen Abschnitten ist noch eine Frage offen geblieben: Wie wird die Band breite auf verschiedene Transportsender verteilt? Auf den ersten Blick scheint die Antwort auf diese Frage einfach: Jeder Sender erhält den gleichen Anteil an der Band breite. Hierbei gibt es jedoch Verschiedenes zu bedenken.

Als Erstes sollte man sich vielleicht fragen, was dieses Problem mit Überlastungsüber wachung zu tun hat. Denn schließlich sollte der Sender, wenn das Netz ihm einen Anteil an der Bandbreite zuteilt, auch nur diesen Teil nutzen. Oft jedoch nehmen Netze keine strengen Bandbreitenreservierungen für einzelne Datenflüsse oder Verbindungen vor. Davon ausgenommen können Datenflüsse sein, wenn die Dienstgüte unterstützt wird; aber viele Verbindungen streben danach, jede zur Verfügung stehende Bandbreite zu nutzen, oder werden vom Netz unter einer gemeinsamen Zuordnung zusammenge fasst. Die von der IETF vorgeschlagenen differenzierten Dienste (*differentiated service*) zum Beispiel trennen den Verkehr in zwei Klassen, wobei die Verbindungen innerhalb jeder Klasse um Bandbreite konkurrieren. Bei IP-Routern konkurrieren oft alle Verbin dungen um dieselbe Bandbreite. In dieser Situation ist es der Mechanismus zur Über lastungsüberwachung, der den konkurrierenden Verbindungen Bandbreite zuweist.

Zweitens sollte man überlegen, was ein fairer Anteil hinsichtlich der Datenflüsse in einem Netz bedeutet. Für den Fall, dass N Datenflüsse nur eine Verbindung verwenden, liegt es auf der Hand, dass für alle jeweils $1/N$ der Bandbreite zur Verfügung steht (obwohl Effizienz vorschreibt, dass sie etwas weniger verwenden, wenn der Verkehr schubweise erfolgt). Doch was passiert, wenn die Datenflüsse verschiedene, aber überlappende Netzpfade aufweisen? Zum Beispiel kann ein Datenfluss über drei Verbindungen laufen und die anderen Datenflüsse über eine Verbindung. Der Datenfluss über drei Verbindungen verbraucht mehr Netzressourcen. In gewisser Hinsicht wäre es fairer, ihm weniger Bandbreite zuzuteilen als den Datenflüssen über eine Verbindung. Durch Reduzieren der Bandbreite für den Datenfluss über drei Verbindungen sollte es allerdings möglich sein, mehrere Datenflüsse über eine Verbindung zu unterstützen. Dies zeigt, dass es eine inhärente Konkurrenz zwischen Fairness und Effizienz gibt.

Wir werden jedoch von einem Fairnesskonzept ausgehen, das nicht von der Länge des Netzpfades abhängt. Sogar bei diesem einfachen Modell ist es relativ kompliziert, Verbindungen einen gleichen Anteil an der Bandbreite zu geben, da verschiedene Verbindungen verschiedene Pfade durch das Netz nehmen und diese Pfade wiederum verschiedene Kapazitäten haben. In diesem Fall kann es passieren, dass ein Datenfluss auf einer Downstream-Verbindung in einen Engpass gerät und einen kleineren Anteil einer Upstream-Verbindung nutzt als andere Datenflüsse; die Reduzierung der Bandbreite der anderen Datenflüsse würde diese nur verlangsamen, wäre jedoch keine Hilfe für den Datenfluss im Engpass.

Die Art der Fairness, die oft für die Netznutzung gewünscht wird, ist die **Max-Min-Fairness**. Eine Zuordnung wird als max-min-fair bezeichnet, wenn die Erhöhung der Bandbreite für einen Datenfluss die Verringerung der Bandbreite für einen anderen Datenfluss mit gleicher oder geringerer Zuordnung zur Folge hat. Das bedeutet, wenn Sie die Bandbreite eines Datenflusses erhöhen, verschlechtern Sie die Situation für die weniger privilegierten Datenflüsse.

Lassen Sie uns das anhand eines Beispiels veranschaulichen. ►Abbildung 6.20 zeigt eine max-min-faire Zuordnung für ein Netz mit vier Datenflüssen A , B , C und D . Alle Verbindungen zwischen den Routern haben der Einfachheit halber die gleiche Kapazität (der Einheit 1), auch wenn sie im Allgemeinen unterschiedliche Kapazitäten aufweisen. Drei Datenflüsse konkurrieren um die Verbindung unten links zwischen den Routern $R4$ und $R5$. Jeder dieser drei Datenflüsse erhält $1/3$ von der Verbindung. Der übrig gebliebene Datenfluss A konkurriert mit B um die Verbindung von $R2$ zu $R3$. Da B $1/3$ der Bandbreite zugeordnet wurde, erhält A die restlichen $2/3$ der Verbindung. Beachten Sie, dass alle anderen Verbindungen Kapazitätsreserven haben. Diese Kapazität kann jedoch nicht an einen der Datenflüsse zugewiesen werden, ohne die Kapazität eines anderen, geringeren Datenflusses zu verringern. Wenn zum Beispiel dem Datenfluss B mehr Bandbreite auf der Verbindung zwischen $R2$ und $R3$ gegeben wird, erhält A automatisch weniger. Dies ist vertretbar, da dem Datenfluss A bereits mehr Bandbreite zur Verfügung steht. Es muss jedoch die Kapazität des Datenflusses C oder D (oder beide) verringert werden, um B mehr Bandbreite zu geben, und diese Datenflüsse haben weniger Bandbreite als B . Das heißt, die Zuordnung ist max-min-fair.

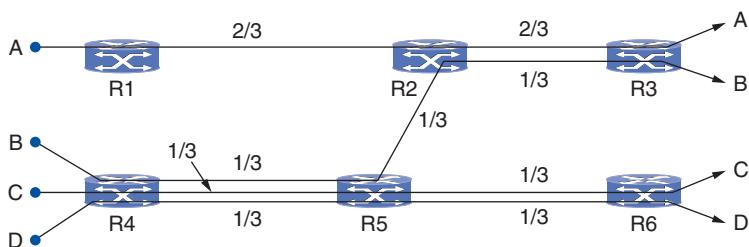


Abbildung 6.20: Max-Min-Zuordnung von Bandbreite für vier Datenflüsse.

Die Berechnung der Max-Min-Zuordnungen setzt eine globale Kenntnis des Netzes voraus. Intuitiv könnte man es sich so vorstellen, dass die Rate für alle Datenflüsse bei null beginnt und langsam ansteigt. Wenn die Rate für irgendeinen Datenfluss einen Engpass erreicht, nimmt dieser Datenfluss nicht länger zu. Die anderen Datenflüsse wachsen weiter und teilen sich die verfügbare Kapazität gerecht, bis sie ebenfalls ihre jeweiligen Engpässe erreichen.

Eine dritte Überlegung ist die Ebene, auf der wir Fairness betrachten: Ein Netz kann auf der Ebene der Verbindungen, der Verbindungen zwischen zwei Hosts oder allen Verbindungen eines Hosts fair sein. Wir haben uns mit diesem Problem in Abschnitt 5.4 im Zusammenhang mit der gewichteten fairen Warteschlange (WFQ, *Weighted Fair Queueing*) beschäftigt und festgestellt, dass jede dieser Definitionen ihre Nachteile hat. Fairness pro Host zu definieren, bedeutet beispielsweise, dass ein ausgelasteter Server nicht besser abschneidet als ein Handy, während die Definition der Fairness pro Verbindung die Hosts ermutigt, weitere Verbindungen zu öffnen. In Anbetracht der Tatsache, dass es keine eindeutige Antwort gibt, wird Fairness oft pro Verbindung betrachtet. Doch ist es in der Regel nicht die absolute Fairness, die uns interessiert. In der Praxis ist es oft wichtiger, dass keine Verbindung „verhungert“, weil ihr jegliche Bandbreite entzogen wird, als dass alle Verbindungen exakt den gleichen Anteil an der Bandbreite zugewiesen bekommen. Mithilfe von TCP ist es möglich, mehrere Verbindungen zu öffnen und aggressiver um Bandbreite zu konkurrieren. Diese Taktik wird von bandbreitenintensiven Anwendungen wie BitTorrent für den Peer-to-Peer-Datenaustausch (*file sharing*) verfolgt.

Konvergenz

Ein letztes Kriterium ist, dass der Algorithmus zur Überlastungsüberwachung möglichst schnell gegen eine faire und effiziente Zuordnung der Bandbreite konvergiert. Die obige Diskussion des gewünschten Arbeitspunktes geht von einer statischen Netzumgebung aus. In der Realität werden in einem Netz jedoch ständig Verbindungen auf- und abgebaut und außerdem schwankt die Bandbreite, die von einer gegebenen Verbindung benötigt wird, über die Zeit, zum Beispiel wenn ein Benutzer Webseiten ansteuert und gelegentlich große Videos herunterlädt.

Aufgrund der wechselhaften Nachfrage schwankt der ideale Arbeitspunkt für das Netz über die Zeit betrachtet. Ein guter Algorithmus zur Überlastungsüberwachung sollte schnell gegen den idealen Arbeitspunkt konvergieren und verfolgen, wie sich dieser

Punkt über die Zeit verändert. Ist die Konvergenz zu langsam, wird der Algorithmus nie in die Nähe des sich ändernden Arbeitspunktes kommen. Wenn der Algorithmus nicht stabil ist, wird er in manchen Fällen nicht gegen den richtigen Arbeitspunkt konvergieren oder nicht einmal um den richtigen Punkt pendeln.

Ein Beispiel für eine Bandbreitenzuordnung, die sich über die Zeit ändert und schnell konvergiert, finden Sie in ►Abbildung 6.21. Am Anfang gehört dem Datenfluss 1 die gesamte Bandbreite. Eine Sekunde später beginnt Datenfluss 2. Auch er muss berücksichtigt werden. Die Zuordnung stellt sich darauf ein und teilt jedem Datenfluss jeweils die Hälfte der Bandbreite zu. Nach 4 Sekunden beginnt ein dritter Datenfluss. Er erhält jedoch nur 20 % der Bandbreite, was weniger als die ihm zustehende Kapazität (ein Drittel) ist. Die Datenflüsse 1 und 2 stellen sich schnell darauf ein und teilen die übrig gebliebene Bandbreite so auf, dass jeder Datenfluss 40 % davon erhält. Nach 9 Sekunden hört der zweite Datenfluss auf. Für den dritten Datenfluss ändert sich nichts, während der erste Datenfluss schnell 80 % der Bandbreite einnimmt. Zu jeder Zeit ist die gesamte zugewiesene Bandbreite 100 %, sodass das Netz voll ausgenutzt wird. Dabei werden konkurrierende Datenflüsse gleich behandelt (müssen aber nicht mehr Bandbreite belegen als sie benötigen).

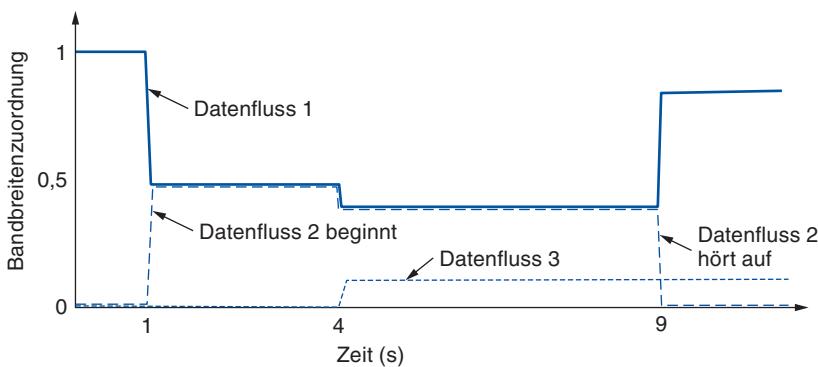


Abbildung 6.21: Sich über die Zeit ändernde Zuordnung der Bandbreite.

6.3.2 Regulierung der Senderate

Inzwischen ist es Zeit, sich dem Hauptproblem zuzuwenden: Wie regulieren wir die Senderaten, um die gewünschte Bandbreitenzuordnung zu erhalten? Die Sendegeschwindigkeit kann von zwei Faktoren beeinflusst werden. Der erste ist Flusskontrolle, falls beim Empfänger nicht genug Puffer zur Verfügung steht. Der zweite ist Überlastungsüberwachung, falls die Kapazität des Netzes nicht ausreicht. ►Abbildung 6.22 veranschaulicht dieses Problem an einem hydraulischen Beispiel. In ►Abbildung 6.22a sehen wir ein dickes Rohr, das zu einem kleinen Empfänger führt. Dies ist die Situation, die durch Flusskontrolle entschärft werden kann, d.h., solange der Sender nicht mehr Wasser sendet, als der Eimer aufnehmen kann, geht kein Wasser verloren. In ►Abbildung 6.22b ist der beschränkende Faktor nicht die Eimerkapazität, sondern die interne Übertragungskapazität des Netzes. Wenn zu viel Wasser zu schnell ankommt, staut es sich an und kann zum Teil verloren gehen (in diesem Fall weil der Trichter überläuft).

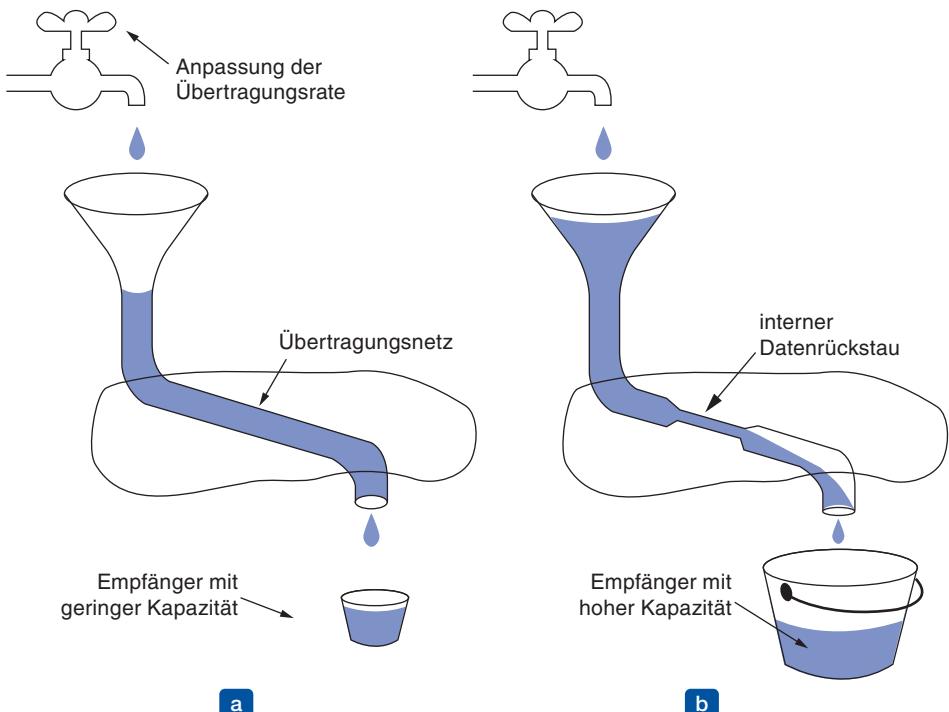


Abbildung 6.22: (a) Ein schnelles Netz, das an einen Empfänger mit geringer Kapazität weiterleitet.
 (b) Ein langsames Netz, das an einen Empfänger mit hoher Kapazität weiterleitet.

Diese Fälle mögen für den Sender ähnlich erscheinen, da zu schnelles Übertragen den Verlust von Paketen verursachen kann. Die Gründe sind jedoch unterschiedlich und verlangen nach individuellen Lösungen. Über Flusskontrolle als Lösung mit einem Fenster variabler Größe haben wir bereits gesprochen. Jetzt betrachten wir Überlastungsüberwachung als Lösung. Da beide Probleme auftreten können, muss das Transportprotokoll im Allgemeinen beide Lösungen berücksichtigen und die Übertragung verlangsamen, wenn eines der Probleme auftritt.

Wie ein Transportprotokoll die Senderate regulieren sollte, hängt von der Art des vom Netz zurückgelieferten Feedbacks ab. Je nach Vermittlungsschicht kann das Feedback anders aussehen. Es kann explizit oder implizit erfolgen, präzise oder unpräzise sein.

Sie haben es zum Beispiel mit einem expliziten, präzisen Ansatz zu tun, wenn die Router den Quellen mitteilen, mit welcher Rate sie senden sollen. Ansätze und Entwürfe in der Literatur wie XCP (*eXplicit Congestion Protocol*) verfolgen diese Strategie (Katabi et al. 2002). Ein expliziter, unpräziser Ansatz ist der Einsatz von ECN (*Explicit Congestion Notification*) bei der Verwendung des TCP-Protokolls. Bei diesem Ansatz setzen Router bei Pakete, die im Stau stehen, spezielle Bits, um die Sender anzuhalten, die Geschwindigkeit zu reduzieren, aber sie geben nicht vor, um wie viel die Sender die Rate reduzieren sollen.

In anderen Ansätzen gibt es kein explizites Signal. FAST TCP beispielsweise misst die Übertragungsverzögerung und verwendet diese Metrik als Signal zur Vermeidung von Staus (Wei et al., 2006). Und schließlich gibt es TCP mit Drop-tail- oder RED-Routern. Diese Form der Überlastungsüberwachung ist heutzutage im Internet am weitesten verbreitet und erkennt Paketverluste, die Signal für ein überlastetes Netz sind. Es gibt von dieser Art des TCP eine Vielzahl von Varianten, einschließlich CUBIC TCP, das unter Linux verwendet wird (Ha et al., 2008). Kombinationen sind auch denkbar. Windows beispielsweise umfasst Compound TCP, das sowohl Paketverlust als auch Verzögerung als Feedbacksignale verwendet (Tan et al., 2006). Diese Ansätze sind in ▶ Abbildung 6.23 zusammengefasst.

| Protokoll | Signal | Explizit? | Präzise? |
|--------------|---|-----------|----------|
| XCP | Zu verwendende Rate | Ja | Ja |
| TCP mit ECN | Überlastungswarnung | Ja | Nein |
| FAST TCP | Ende-zu-Ende-Verzögerung | Nein | Ja |
| Compound TCP | Paketverlust und Ende-zu-Ende-Verzögerung | Nein | Ja |
| CUBIC TCP | Paketverlust | Nein | Nein |
| TCP | Paketverlust | Nein | Nein |

Abbildung 6.23: Signale einiger Protokolle zur Überlastungsüberwachung.

Erhält die Transportinstanz ein explizites und präzises Signal, kann sie dieses Signal dazu verwenden, ihre Rate an den neuen Arbeitspunkt anzupassen. Wenn beispielsweise XCP den Sendern die zu verwendende Rate mitteilt, können die Sender diese Rate einfach übernehmen. In den anderen Fällen basieren ihre Entscheidungen auf Vermutungen. Liegt kein Überlastungssignal vor, sollten die Sender ihre Raten erhöhen. Wird ein Überlastungssignal gegeben, sollten die Sender ihre Raten verringern. Wie diese Raten erhöht oder erniedrigt werden, wird von einem **Steuerungsgesetz** vorgegeben. Diese Gesetze haben eine bedeutende Auswirkung auf die Leistung.

Chiu und Jain (1989) haben den Fall des binären Überlastungsfeedbacks untersucht und sind zu dem Schluss gekommen, dass **AIMD** (*Additive Increase Multiplicative Decrease*) das angemessene Steuerungsgesetz ist, um den effizienten und gerechten Arbeitspunkt zu ermitteln. Zur Begründung haben sie den einfachen Fall, dass zwei Verbindungen um die Bandbreite einer Leitung konkurrieren, grafisch veranschaulicht. Das Schaubild in ▶ Abbildung 6.24 zeigt die Bandbreite, die dem Benutzer 1 auf der x-Achse und dem Benutzer 2 auf der y-Achse zugewiesen wird. Wenn die Zuordnung gerecht ist, erhalten beide Benutzer den gleichen Anteil an der Bandbreite. Dies wird durch die gestrichelte Fairnesslinie angezeigt. Wenn die Zuordnungen sich zu 100 %, der Kapazität der Verbindung, aufaddieren, wird die Zuordnung als effizient bezeichnet. Dies wird durch die gestrichelte Effizienzlinie angezeigt. Das Netz sendet an beide Benutzer ein Überlastungssignal, wenn die Summe ihrer Zuordnungen diese Linie

kreuzt. Der Schnittpunkt dieser Linien ist der gewünschte Arbeitspunkt, an dem beide Benutzer die gleiche Bandbreite haben, und die gesamte Netzbänderbreite genutzt wird.

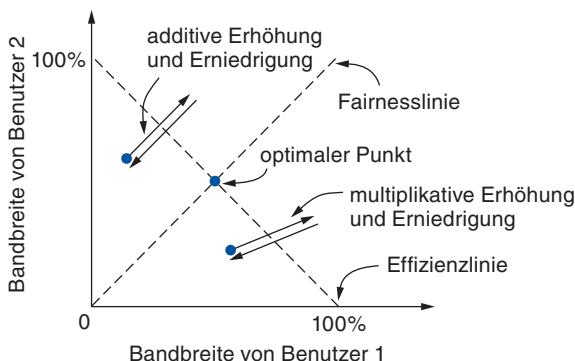


Abbildung 6.24: Additive und multiplikative Bandbreitenanpassungen.

Betrachten wir, was bei einer Ausgangszuordnung passiert, wenn Benutzer 1 und Benutzer 2 ihre jeweiligen Bandbreiten über die Zeit additiv erhöhen. Angenommen, die Benutzer erhöhen ihre Senderaten jede Sekunde um 1 Mbit/s. Zum Schluss kreuzt der Arbeitspunkt die Effizienzlinie und beide Benutzer erhalten ein Überlastungssignal vom Netz. Auf dieser Stufe müssen sie ihre Zuordnungen verringern. Eine additive Erniedrigung würde jedoch nur dazu führen, dass die Zuordnungen um eine additive Linie schwanken. Diese Situation sehen Sie in Abbildung 6.24. Bei diesem Verhalten ist der Arbeitspunkt immer relativ effizient, aber nicht notwendigerweise fair.

Oder betrachten wir den Fall, dass beide Benutzer ihre Bandbreite über die Zeit multiplikativ erhöhen, bis sie ein Überlastungssignal erhalten. Die Benutzer könnten beispielsweise ihre Senderaten jede Sekunde um 10 % erhöhen. Wenn sie dann ihre Senderaten multiplikativ reduzieren, schwankt der Arbeitspunkt der Benutzer einfach entlang einer multiplikativen Linie. Auch dieses Verhalten ist der Abbildung 6.24 zu entnehmen. Die multiplikative Linie hat eine andere Steigung als die additive Linie. (Sie zeigt auf den Ursprung, während die additive Linie einen Winkel von 45 Grad hat.) Ansonsten ist sie jedoch nicht besser. In keinem Fall werden die Benutzer gegen die optimalen Senderaten konvergieren, die sowohl fair als auch effizient sind.

Kommen wir jetzt zu dem Fall, dass die Benutzer ihre Bandbreitenzuordnungen additiv erhöhen und multiplikativ reduzieren, wenn eine Überlastung signalisiert wird. Dieses Verhalten entspricht dem AIMD-Steuerungsgesetz und wird in ►Abbildung 6.25 gezeigt. Es ist zu erkennen, dass der von diesem Verhalten erzeugte Pfad gegen den optimalen Punkt konvergiert. Dieser Punkt ist sowohl fair als auch effizient. Diese Konvergenz wird unabhängig vom Startpunkt erreicht, sodass AIMD vielseitig einsetzbar ist. Dieser Argumentation folgend würde die einzige andere Kombination (multiplikative Erhöhung und additive Erniedrigung) vom optimalen Punkt wegführen.

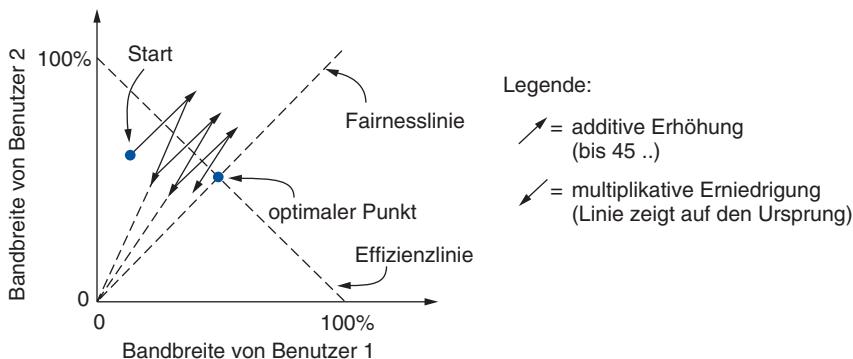


Abbildung 6.25: Das AIMD-Steuerungsgesetz.

AIMD ist das von TCP verwendete Steuerungsgesetz – basierend auf diesem Argument sowie einem weiteren Stabilitätsargument (demzufolge es einfach ist, ein Netz zu überlasten, aber schwer, die Überlastung zu beseitigen, sodass die Erhöhung langsam und die Erniedrigung aggressiv erfolgen sollten). Es ist nicht absolut fair, da TCP-Verbindungen ihre Fenstergröße bei jeder Paketumlaufzeit um einen gegebenen Betrag anpassen. Unterschiedliche Verbindungen haben unterschiedliche Paketumlaufzeiten. Dies führt dazu, dass unter sonst gleichen Bedingungen die Verbindungen zu näheren Hosts mehr Bandbreite erhalten als Verbindungen zu entfernten Hosts.

In Abschnitt 6.4 beschreiben wir ausführlich, wie TCP ein AIMD-Steuerungsgesetz implementiert, um die Senderate anzupassen und die Überlastung zu überwachen. Diese Aufgabe ist schwieriger als sie klingt, da die Raten über einen Intervall gemessen werden und der Verkehr schubweise erfolgt. Anstatt die Rate direkt anzupassen, wird in der Praxis oft die Strategie verfolgt, die Größe eines Schiebefensters anzupassen. TCP verwendet diese Strategie. Wenn die Fenstergröße W ist und die Paketumlaufzeit RTT (*Round Trip Time*), beträgt die entsprechende Rate W/RTT . Diese Strategie lässt sich leicht mit der Flusskontrolle kombinieren, die bereits ein Fenster verwendet, und hat den Vorteil, dass der Sender Pakete nach Erhalt von Bestätigungen verschickt und damit in einem RTT langsamer wird, wenn er keine Berichte mehr erhält, dass Pakete das Netzwerk verlassen.

Zum Schluss soll noch angesprochen werden, dass viele verschiedene Transportprotokolle den Verkehr im Netz verursachen können. Was passiert, wenn die verschiedenen Protokolle zur Vermeidung von Überlastung verschiedenen Steuerungsgesetzen gehorchen, die dann miteinander im Wettstreit stehen? Ungleiche Bandbreitenzuordnung, natürlich! Seit TCP die vorherrschende Form der Überlastungsüberwachung im Internet ist, ist der Ruf nach dem Entwurf neuer Transportprotokolle in der Gemeinde lauter geworden, sodass sie gerecht miteinander konkurrieren. Die frühen Streaming-Media-Protokolle verursachten Probleme, da sie den TCP-Durchsatz aufgrund des unfairen Wettbewerbs extrem reduzierten. Dies führte zu dem Begriff der **TCP-freundlichen** Überlastungsüberwachung, in der TCP- und Nicht-TCP-Transportprotokolle ohne gravierende Nebenwirkungen frei kombiniert werden können (Floyd et al., 2000).

6.3.3 Probleme mit der drahtlosen Übertragung

Transportprotokolle wie TCP, die Überlastungsüberwachung implementieren, sollten von dem zugrunde liegenden Netz und den Technologien der Sicherungsschicht unabhängig sein. In der Theorie klingt das ganz gut, aber in der Praxis gibt es Probleme mit den drahtlosen Netzen. Das Hauptproblem ist, dass oft Paketverlust als Überlastungssignals verwendet wird – nicht zuletzt von TCP, wie wir bereits besprochen haben. Drahtlose Netze verlieren aufgrund von Übertragungsfehlern die ganze Zeit über Pakete.

Beim AIMD-Steuerungsgesetz erfordern hohe Durchsätze sehr geringe Paketverluste. Untersuchungen von Padhye et al. (1998) zeigen, dass der Durchsatz als Umkehrfunktion der Quadratwurzel der Paketverlustrate steigt. Das bedeutet in der Praxis, dass die Verlustrate für schnelle TCP-Verbindungen sehr gering ist. Eine gemäßigte Verlustrate liegt bei 1 %; hat sie 10 % erreicht, hört die Verbindung praktisch auf zu funktionieren. Bei drahtlosen Netzen wie IEEE-802.11-LANs sind Rahmenverlustraten von mindestens 10 % jedoch häufig der Fall. Dieser Unterschied bedeutet, dass angesichts fehlender Schutzmaßnahmen die Überlastungsüberwachungsmechanismen, die mit Paketverlust als Signal arbeiten, drahtlose Verbindungen unnötig auf sehr niedrige Raten drosseln werden.

Um gut zu funktionieren, sollte der Algorithmus zur Überlastungsüberwachung nur Paketverluste registrieren, die auf unzureichende Bandbreite zurückzuführen sind und nicht auf Übertragungsfehler. Eine Lösung hierfür besteht darin, die Verluste auf drahtlosen Verbindungen durch Neuübertragungen zu „maskieren“. Zum Beispiel verwendet IEEE 802.11 ein Stop-and-Wait-Protokoll für die Zustellung aller Rahmen, wobei im Bedarfsfall Rahmen mehrmals übertragen werden, bevor ein Paketverlust der höheren Ebene mitgeteilt wird. Im Normalfall wird jedes Paket zugestellt – trotz vorübergehender Übertragungsfehler, die für die höheren Ebenen nicht sichtbar sind.

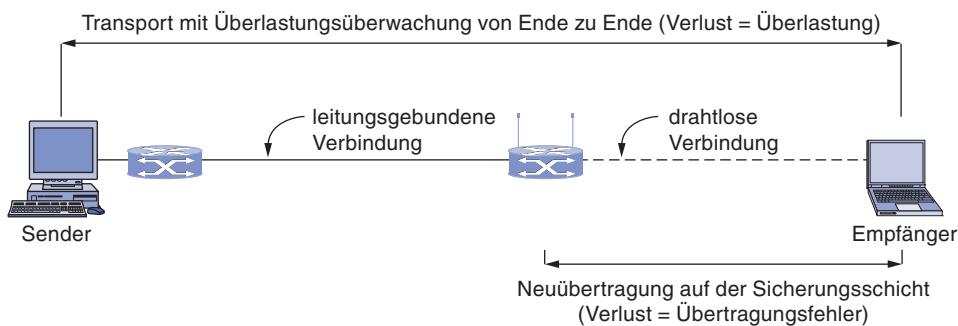


Abbildung 6.26: Überlastungsüberwachung über einen Pfad mit einer drahtlosen Verbindung.

► Abbildung 6.26 zeigt einen Pfad mit einer leitungsgebundenen und einer drahtlosen Verbindung, für den die Maskierungsstrategie verwendet wird. Dabei gibt es zwei Aspekte zu beachten. Zum einen weiß der Sender nicht unbedingt, dass der Pfad eine drahtlose Verbindung umfasst, da er nur die leitungsgebundene Verbindung sieht, mit der er verbunden ist. Internetpfade sind heterogen und es gibt für den Sender keinen allgemeinen Weg festzustellen, welche Art von Verbindungen zum Pfad gehört. Dies

kompliziert das Problem der Überlastungsüberwachung, da es keine leichte Möglichkeit gibt, ein Protokoll für leitungsungebundene Verbindungen und ein weiteres für leitungsgebundene Verbindungen zu verwenden.

Der zweite Aspekt wirft eine Frage auf. Die Abbildung zeigt zwei Mechanismen, die von Verlust gesteuert werden: die Neuübertragung der Rahmen über die Sicherungsschicht und die Überlastungsüberwachung auf der Transportschicht. Die Frage ist, wie diese beiden Mechanismen nebeneinander existieren können, ohne durcheinander zu geraten. Schließlich sollte ein Verlust nur einen Mechanismus auslösen, da es sich entweder um einen Übertragungsfehler oder um ein Überlastungssignal handelt. Beides gleichzeitig ist nicht möglich. Wenn beiden Mechanismen Maßnahmen einleiten (durch Neuübertragung des Rahmens und Verlangsamung der Senderate), stehen wir wieder vor dem ursprünglichen Problem der Transporte, die viel zu langsam über drahtlose Verbindungen laufen. Beschäftigen Sie sich ein wenig mit dieser Frage und überlegen Sie, ob Sie es lösen können.

Die Lösung ist, dass die beiden Mechanismen auf verschiedenen Zeitskalen ausgeführt werden. Neuübertragungen auf der Sicherungsschicht finden bei drahtlosen Verbindungen wie IEEE 802.11 im Mikro- bis Millisekundenbereich statt. Verlust-Timer in den Transportprotokollen feuern im Bereich von Millisekunden bis Sekunden. Der Unterschied beträgt drei Größenordnungen. So können drahtlose Verbindungen Rahmenverluste entdecken und neu übertragen, um Übertragungsfehler zu beheben, lange bevor die Transportinstanz auf Paketverlust schließt.

Die Maskierungsstrategie reicht, damit der Großteil der Transportprotokolle auf den meisten drahtlosen Verbindungen problemlos läuft. Es ist jedoch nicht immer eine passende Lösung. Manche drahtlosen Verbindungen, wie beim Satellitenfunk, haben lange Paketumlaufzeiten. Für diese Verbindungen müssen andere Methoden angewendet werden, um den Verlust zu maskieren, z.B. Vorwärtsfehlerkorrektur (*FEC, Forward Error Correction*), oder das Transportprotokoll muss ein Nicht-Verlust-Signal zur Überlastungsüberwachung verwenden.

Ein zweites Problem mit der Überlastungsüberwachung bei drahtlosen Verbindungen ist die variable Kapazität. Das heißt, die Kapazität einer drahtlosen Verbindung ändert sich über die Zeit, manchmal sogar sehr abrupt, da sich die Position von Knoten ändern kann und das Signal-Rausch-Verhältnis mit den sich ändernden Kanalbedingungen variiert. Dies ist bei leitungsgebundenen Verbindungen anders, da deren Kapazität fest vorgegeben ist. Das Transportprotokoll muss sich an die wechselnde Kapazität von drahtlosen Verbindungen anpassen, da es sonst das Netz überlastet oder die verfügbare Kapazität nicht nutzen kann.

Eine mögliche Lösung besteht darin, sich hierüber keine Gedanken zu machen. Diese Strategie ist vertretbar, da Algorithmen zu Überlastungsüberwachung bereits den Fall bewältigen müssen, dass neue Netzbenutzer hinzukommen oder bestehende Benutzer ihre Senderaten ändern. Auch wenn die Kapazität von leitungsgebundenen Verbindungen fest ist, spiegelt sich das wechselnde Verhalten der anderen Netzbenutzer in einer Schwankung der Bandbreite wider, die für einen gegebenen Benutzer verfügbar

ist. Also ist es durchaus möglich, TCP über einen Pfad in einem IEEE-802.11-Funknetz auszuführen und eine vernünftige Leistung zu erhalten.

Wenn es jedoch große Schwankungen auf einer drahtlosen Verbindung gibt, können Transportprotokolle, die für leitungsgebundene Verbindungen entwickelt wurden, irgendwann nicht mehr mithalten und schwächeln. Die Lösung in diesem Fall ist ein Transportprotokoll, das speziell für drahtlose Verbindungen entworfen wurde. Eine besondere Herausforderung ist ein Funknetz, in dem die Übertragung über viele, sich störende drahtlose Verbindungen läuft, Routen sich aufgrund von Mobilität ändern und häufig mit Verlust zu rechnen ist. In diesem Bereich wird zurzeit ausgiebig geforscht. Ein Beispiel für den Entwurf drahtloser Transportprotokolle finden Sie bei Li et al. (2009).

6.4 Internettransportprotokolle: UDP

Das Internet hat auf der Transportschicht zwei Protokolltypen, ein verbindungsorientiertes und ein verbindungsloses. Die Protokolle ergänzen einander. Das verbindungslose Protokoll ist UDP. Es macht im Prinzip nichts anderes, als Pakete zwischen Anwendungen zu senden, die bei Bedarf darauf ihre eigenen Protokolle aufsetzen können. Das verbindungsorientierte Protokoll ist TCP. Es macht so ziemlich alles. Es stellt Verbindungen her und erhöht die Zuverlässigkeit mittels Neuübertragungen, zusammen mit Flusskontrolle und Überlastungsüberwachung – und alles im Interesse der Anwendungen, die dieses Protokoll verwenden.

In den folgenden Abschnitten werden wir uns den beiden Protokollen UDP und TCP zuwenden. Wir beginnen mit UDP, da es am einfachsten ist. Wir betrachten unter anderem zwei Einsätze von UDP. Da UDP ein Protokoll auf der Transportschicht ist, das normalerweise im Betriebssystem läuft, und Protokolle, die UDP verwenden, normalerweise im Benutzerbereich laufen, können diese Einsätze auch als Anwendungen betrachtet werden. Allerdings sind die eingesetzten Techniken für viele Anwendungen nützlich und sollten besser als Teil eines Transportdiensts betrachtet werden. Deshalb werden wir sie hier behandeln.

6.4.1 Einführung in UDP

Die Internetprotokollsuite unterstützt mit **UDP** (*User Datagram Protocol*, Benutzer-Datagramm-Protokoll) ein verbindungsloses Transportprotokoll. UDP ermöglicht es den Anwendungen, gekapselte IP-Datagramme zu übertragen, ohne eine Verbindung aufzubauen. UDP ist in RFC 768 definiert.

UDP überträgt **Segmente**, die aus einem 8-Byte-Header sowie den darauf folgenden Nutzdaten bestehen. Dieser Header ist in ▶ Abbildung 6.27 dargestellt. Die beiden **Ports** identifizieren die Endpunkte im Quell- und Zielrechner. Kommt ein UDP-Paket an, werden die Nutzdaten an den Prozess übergeben, der dem Zielport zugeordnet ist. Diese Zuordnung erfolgt über eine BIND-Primitive oder etwas Ähnlichem, wie in Abbildung 6.6 für TCP dargestellt (der Bindeprozess ist bei UDP gleich). Stellen Sie

sich Ports als Briefkästen vor, die von Anwendungen für den Empfang von Paketen gemietet werden können. Wir werden näher darauf eingehen, wenn wir TCP beschreiben, das ebenfalls Ports verwendet. Der Hauptvorteil von UDP gegenüber dem reinen IP sind die Quell- und Zielports. Ohne die Portfelder wüsste die Transportschicht nicht, was sie mit den eingehenden Paketen machen soll. Mit ihnen stellt sie die Segmente jedoch der korrekten Anwendung zu.

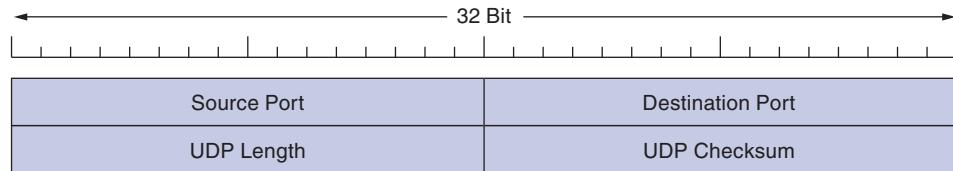


Abbildung 6.27: Der UDP-Header.

Der Quellport wird vor allem dann benötigt, wenn eine Antwort an die Quelle zurückgesendet werden muss. Durch Kopieren des Feldes *Source Port* aus dem eingehenden Segment in das Feld *Destination Port* des ausgehenden Segments kann der Prozess, der die Antwort sendet, genau angeben, welcher Prozess auf dem sendenden Rechner das Segment erhalten soll.

Das Feld *UDP Length* beinhaltet den 8-Byte-Header und die Daten. Die minimale Länge ist 8 Byte, um den Header abzudecken. Die Maximallänge beträgt 65 515 Byte, was aufgrund der Größenbeschränkung für IP-Pakete noch unter der Maximalzahl der Bytes liegt, die in 16 Bit darstellbar ist.

Für erhöhte Zuverlässigkeit wird eine optionales Feld *Checksum* angeboten. In diese Prüfsumme fließen der Header, die Daten und ein konzeptioneller IP-Pseudoheader ein. Bei der Berechnung wird das Feld *Checksum* auf 0 gesetzt und das Datenfeld wird mit einem zusätzlichen Nullbyte aufgefüllt, wenn seine Länge eine ungerade Zahl ist. Der Prüfsummenalgorithmus besteht lediglich darin, alle 16-Bit-Wörter im Einerkomplement zu addieren und wiederum das Einerkomplement der Summe als Prüfsumme zu nehmen. Wenn der Empfänger die Berechnung auf dem gesamten Segment einschließlich des Feldes *Checksum* ausführt, sollte das Ergebnis 0 lauten. Wird die Prüfsumme nicht berechnet, wird sie als 0 gespeichert, da durch einen glücklichen Zufall in der Einerkomplement-Arithmetik eine echte 0-berechnete Prüfsumme als eine Folge von Einsen gespeichert wird. Sie zu deaktivieren, wäre jedoch unsinnig, es sei denn, die Qualität der Daten spielt keine Rolle (z.B. digitalisierte Sprache).

Der Pseudoheader für IPv4 finden Sie in ▶ Abbildung 6.28. Er enthält die 32-Bit-IPv4-Adressen der Quell- und Zielrechner, die Protokollnummer für UDP (17) und die Bytezahl für das UDP-Segment (einschließlich des Headers). Für IPv6 ist es anders aber analog. Die Berücksichtigung des Pseudoheaders in der Berechnung der UDP-Prüfsumme trägt dazu bei, falsch zugestellte Pakete zu entdecken, doch es verletzt auch die Protokollhierarchie, da die darin enthaltenen IP-Adressen zu der IP-Schicht und nicht zu der UDP-Schicht gehören. TCP verwendet für seine Prüfsumme den gleichen Pseudoheader.

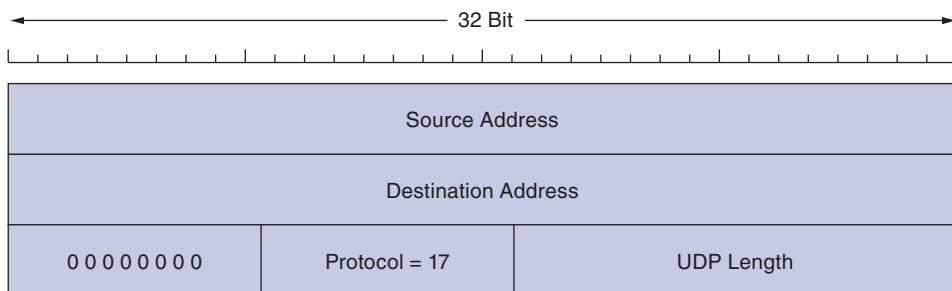


Abbildung 6.28: Der IPv4-Pseudoheader als Teil der UDP-Prüfsumme.

Man sollte auch einige Dinge explizit erwähnen, die UDP *nicht* unterstützt. Hierzu gehören die Flusskontrolle, die Überlastungsüberwachung oder die erneute Übertragung eines beschädigten Segments. Hierfür sind die Benutzerprozesse zuständig. Was UDP jedoch leistet, ist, eine Schnittstelle zum IP-Protokoll zur Verfügung zu stellen, einschließlich der zusätzlichen Funktion, mehrere Prozesse zu demultiplexen, und zwar unter Verwendung der Ports und einer optionalen Ende-zu-Ende-Fehlerermittlung. Mehr kann UDP nicht.

Anwendungen, die eine exakte Kontrolle über den Paketfluss, die Fehlerkontrolle oder das zeitliche Verhalten benötigen, erhalten bei UDP genau das, was sie benötigen. Es gibt einen Bereich, in dem UDP besonders nützlich ist – den Client-Server-Bereich. Oftmals sendet ein Client eine kurze Anfrage an den Server und erwartet eine kurze Antwort. Geht entweder die Anfrage oder die Antwort verloren, kann der Client bis zum Timeout warten und es dann erneut versuchen. Nicht nur der Code ist einfach, auch sind weniger Nachrichten erforderlich (eine pro Richtung) als bei einem Protokoll wie TCP, das zuvor einen Verbindungsaufbau erfordert.

Eine Anwendung, die UDP auf diese Weise nutzt, ist beispielsweise das DNS (*Domain Name System*, Domänennamensystem). Wir werden darauf in Kapitel 7 näher eingehen. Kurz, ein Programm, das die IP-Adresse eines Host-Namens (z.B. `www.cs.berkeley.edu`) nachschlagen muss, kann ein UDP-Paket mit dem Host-Namen an einen DNS-Server senden. Der Server antwortet mit einem UDP-Paket, das die IP-Adresse des Hosts enthält. Es ist vorab kein Verbindungsauflauf erforderlich und folglich nachher auch keinerlei Abbau. Es gehen nur zwei Nachrichten über das Netzwerk.

6.4.2 Entfernte Prozeduraufrufe

In gewissem Sinne sind das Senden einer Nachricht an einen entfernten Host und der Erhalt einer Rückantwort mit einem Funktionsaufruf in einer Programmiersprache vergleichbar. In beiden Fällen beginnt man mit einem oder mehreren Parametern und erhält ein Ergebnis zurück. Diese Beobachtung hat dazu geführt, dass man versucht hat, Anfrage-Antwort-Interaktionen in Netzen in Form von Prozeduraufrufen auszuführen. Hierdurch können Netzanwendungen viel einfacher programmiert und verwaltet werden. Stellen Sie sich beispielsweise eine Prozedur namens `get_IP_address(host_name)` vor, die ein UDP-Paket an einen DNS-Server sendet und auf eine Antwort

wartet, einen Zeitüberlauf erfährt und es erneut versucht, wenn die Antwort nicht schnell genug kommt. Auf diese Weise können alle Einzelheiten des Netzbetriebs vor dem Programmierer verborgen werden.

Die Hauptarbeit in diesem Bereich leisteten Birrell und Nelson (1984). Kurz, Birrell und Nelson schlugen vor, dass Programme Prozeduren auf entfernten Hosts aufrufen dürfen. Wenn ein Prozess auf Rechner 1 eine Prozedur auf Rechner 2 aufruft, wird der aufrufende Prozess auf 1 zeitweilig unterbrochen und die aufgerufene Prozedur auf Rechner 2 ausgeführt. Der Aufrufende kann Informationen in Form von Parametern an den Aufgerufenen übergeben und erhält als Rückgabewert das Prozedurergebnis zurück. Der Programmierer sieht nichts von dem Nachrichtenaustausch. Diese Technik wird als **entfernter Prozederaufruf** oder **RPC** (*Remote Procedure Call*) bezeichnet und bildet die Grundlage für viele Netzanwendungen. Die aufrufende Prozedur wird in der Regel als Client und die aufgerufene Prozedur als Server bezeichnet. Wir werden uns im Folgenden an diese Bezeichnungen halten.

Die Idee hinter den entfernten Prozederaufrufen ist, dass ein entfernter Prozederaufruf weitestgehend wie ein lokaler Aufruf aussehen soll. Für einen entfernten Prozederaufruf in seiner einfachsten Form muss das Client-Programm mit einer kleinen Bibliotheksprozedur, dem sogenannten **Client-Stub**, gebunden werden, die die Server-Prozedur im Adressraum des Clients repräsentiert. Ebenso wird der Server mit einer Prozedur namens **Server-Stub** gebunden. Diese Prozeduren verbergen die Tatsache, dass der Prozederaufruf vom Client zum Server nicht lokal ist.

Die einzelnen Schritte eines entfernten Prozederaufrufs werden in ►Abbildung 6.29 dargestellt. In Schritt 1 ruft der Client den Client-Stub auf. Dieser Aufruf ist ein lokaler Prozederaufruf, bei dem die Parameter wie gewohnt auf dem Stapel abgelegt werden. In Schritt 2 packt der Client-Stub die Parameter in eine Nachricht und benutzt einen Systemaufruf, um die Nachricht zu senden. Das Verpacken der Parameter wird als **Marshaling** (Anordnen, Bereitstellen) bezeichnet. In Schritt 3 sendet das Betriebssystem die Nachricht vom Client zum Server. In Schritt 4 übergibt das Betriebssystem das eingehende Paket an den Server-Stub. Schließlich, in Schritt 5, ruft der Server-Stub die Server-Prozedur mit den inzwischen entpackten Parametern auf. Die Antwort nimmt den gleichen Pfad, allerdings in die andere Richtung.

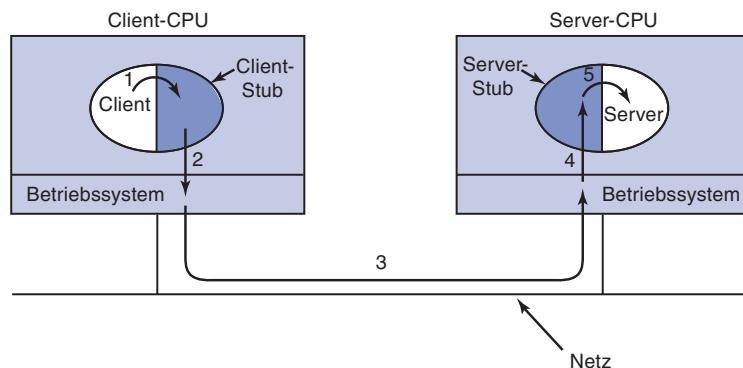


Abbildung 6.29: Schritte in einem entfernten Prozederaufruf. Die Stubs sind dunkelblau schattiert.

Der zentrale Punkt hierbei ist, dass die Client-Prozedur, die vom Benutzer erstellt wurde, an den Client-Stub einen normalen (also lokalen) Prozeduraufruf schickt, der den gleichen Namen trägt wie die Server-Prozedur. Da die Client-Prozedur und der Client-Stub sich im gleichen Adressraum befinden, werden die Parameter normal übergeben. Entsprechend wird die Server-Prozedur von einer Prozedur in ihrem Adressraum mit den erwarteten Parametern aufgerufen. Für die Server-Prozedur verläuft nichts ungewöhnlich. Auf diese Weise erfolgt die E/A nicht über Sockets, sondern die Kommunikation findet über das Netz durch Vortäuschen eines normalen Prozeduraufrufs statt.

Auch wenn entfernte Prozeduraufrufe konzeptionell sehr elegant sind, haben sie doch einige Haken. Ein großer Nachteil ist die Verwendung von Zeigerparametern. Normalerweise ist die Übergabe eines Zeigers an eine Prozedur kein Problem. Die aufgerufene Prozedur kann den Zeiger genauso verwenden wie die aufrufende Prozedur, da beide im gleichen virtuellen Adressraum existieren. Bei entfernten Prozeduraufrufen hingegen sind Zeiger als Parameter nicht möglich, weil sich Client und Server in verschiedenen Adressräumen befinden.

In manchen Fällen lassen sich Zeiger mit einem Griff in die Trickkiste doch noch übergeben. Angenommen, der erste Parameter ist ein Zeiger auf einen Integerwert k . Der Client-Stub kann dann k verpacken und zum Server senden. Der Server-Stub erzeugt daraufhin einen Zeiger auf k und übergibt ihn wie erwartet an die Server-Prozedur. Wenn die Server-Prozedur anschließend die Kontrolle wieder an den Server-Stub zurückgibt, sendet dieser k zurück an den Client, wobei das alte k durch das neue k ersetzt wird, für den Fall, dass der Server es verändert hat. Genau genommen wurde also der standardmäßige Aufruf durch Verweis (*call-by-reference*) durch einen Aufruf durch Kopieren und Wiederherstellen (*call-by-copy restore*) ersetzt. Leider funktioniert dies nicht immer, beispielsweise wenn der Zeiger auf einen Graphen oder eine andere komplexe Datenstruktur zeigt. Aus diesem Grund müssen den Parametern an entfernt aufgerufene Prozeduren gewisse Beschränkungen auferlegt werden.

Ein zweites Problem ist, dass bei schwach typisierten Programmiersprachen wie C es vollkommen legal ist, eine Prozedur zu schreiben, die das innere Produkt zweier Vektoren (Arrays) berechnet, ohne dass angegeben werden muss, wie groß diese sind. Jeder könnte durch einen besonderen Wert terminiert werden, der nur der aufrufenden und der aufgerufenen Prozedur bekannt ist. Unter diesen Umständen ist es für den Client-Stub mehr oder weniger unmöglich, die Parameter zu verpacken: Er kann nicht feststellen, wie groß diese sind.

Ein drittes Problem besteht darin, dass es nicht immer möglich ist, die Parametertypen abzuleiten, selbst nicht aus einer formalen Spezifikation oder dem Code selbst. Ein Beispiel ist *printf*, das eine beliebige Anzahl an Parametern aufweisen kann (mindestens einen). Diese Parameter können eine beliebige Mischung aus Integer-, Short-, Long-Datentypen, Zeichen, Zeichenfolgen, Gleitkommazahlen verschiedener Länge und anderen Datentypen sein. Der Aufruf von *printf* als entfernte Prozedur ist praktisch unmöglich, weil C nicht restriktiv genug ist. Allerdings würde sich eine Regel, die besagt, dass man einen entfernten Prozeduraufruf verwenden kann, solange er nicht in C oder C++ programmiert ist, keiner großen Beliebtheit erfreuen.

Ein vierter Problem bezieht sich auf die Verwendung globaler Variablen. In der Regel können die aufrufende und die aufgerufene Prozedur nicht nur über Parameter, sondern auch über globale Variablen kommunizieren. Wird aber die aufgerufene Prozedur auf einen entfernten Rechner verschoben, kann der Code nicht ausgeführt werden, weil die globalen Variablen nicht mehr länger gemeinsam verwendet werden.

Diese Probleme sollen nicht implizieren, dass der entfernte Prozeduraufruf ein hoffnungsloser Fall ist. Eigentlich wird er sogar häufig verwendet. In der Praxis sind aber einige Einschränkungen erforderlich.

Bezüglich der Transportschichtprotokolle ist UDP eine gute Basis zur Implementierung von entfernten Prozeduraufrufen. Sowohl Anfragen als auch Antworten können im einfachsten Fall als einzelne UDP-Pakete gesendet werden – eine Operation, die sehr schnell sein kann. Eine Implementierung muss jedoch noch weitere Vorkehrungen treffen. Da die Anfrage oder Antwort verloren gehen kann, muss der Client einen Timer vorsehen, um die Anfrage erneut zu übertragen. Beachten Sie, dass eine Antwort auf eine Anfrage implizit als Bestätigung verstanden wird, sodass die Anfrage nicht separat bestätigt werden muss. Manchmal sind die Parameter oder Ergebnisse größer als die maximale UDP-Paketgröße. In diesem Fall wird ein Protokoll benötigt, das große Nachrichten übertragen kann. Wenn mehrere Anfragen und Antworten sich überlappen (wie im Fall der nebenläufigen Programmierung), wird ein Identifikator benötigt, über den die Anfrage einer Antwort zugeordnet werden kann.

Auf höherer Ebene gibt es noch das Problem, dass die Operation unter Umständen nicht idempotent (d.h. sicher zu wiederholen) ist. Den einfachen Fall stellen idempotente Operationen wie DNS-Anfragen und -Antworten dar. Der Client kann diese Anfragen immer wieder sicher übertragen, wenn keine Antworten eingehen. Es spielt keine Rolle, ob der Server die Anfrage nie empfangen hat oder dass es die Antwort war, die verloren gegangen ist. Die Antwort wird, wenn sie denn endgültig ankommt, die gleiche sein (vorausgesetzt die DNS-Datenbank wird in der Zwischenzeit nicht aktualisiert). Nicht alle Operationen sind indes idempotent, was zum Beispiel daran liegt, dass sie wichtige Nebeneffekte haben, wie die Inkrementierung eines Zählers. Entfernte Prozeduraufrufe für diese Operationen erfordern eine striktere Semantik, sodass eine Prozedur, die von einem Programmierer aufgerufen wird, nicht mehrfach ausgeführt wird. In diesem Falle mag es erforderlich sein, eine TCP-Verbindung einzurichten und die Anfrage darüber zu senden anstatt über UDP.

6.4.3 Echtzeittransportprotokolle

Neben dem Einsatz bei Client-Server-RPCs wird UDP insbesondere auch für Echtzeit-Multimedia-Anwendungen genutzt. Angesichts der immer stärkeren Verbreitung von Internetradio, Internettelefonie, Music-on-Demand, Videokonferenzen, Video-on-Demand und anderen Multimedia-Anwendungen stellte man bald fest, dass jede Anwendung das gleiche Echtzeittransportprotokoll mehr oder weniger immer wieder neu erfand. Nach und nach zeigte sich, dass ein generisches Echtzeittransportprotokoll für mehrere Anwendungen keine schlechte Idee wäre.

Dies führte zu der Entwicklung des **Echtzeittransportprotokolls (RTP, Real-time Transport Protocol)**. Es ist in RFC 3550 definiert und inzwischen aus Multimedia-Anwendungen kaum noch wegzudenken. Wir werden zwei Aspekte des Echtzeittransports beschreiben. Der erste ist das RTP-Protokoll für den Transport von Audio- und Videodaten in Paketen. Der zweite ist die Verarbeitung, die hauptsächlich beim Empfänger stattfindet, um die Audio- und Videodaten zur richtigen Zeit abzuspielen. Diese Funktionen passen in den Protokollstapel, wie er in ► Abbildung 6.30 zu sehen ist.

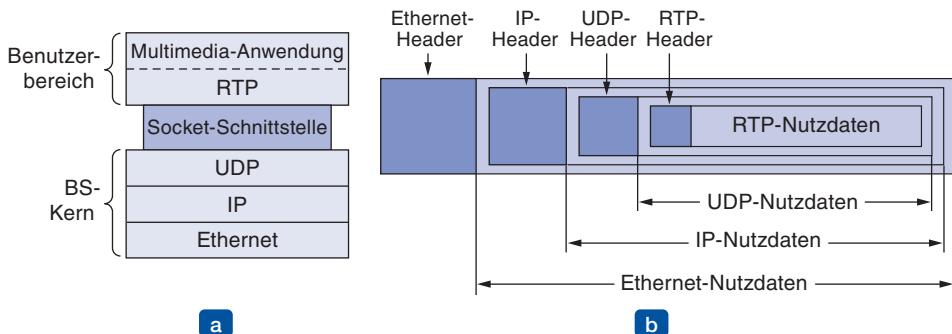


Abbildung 6.30: (a) Die Position von RTP im Protokollstapel. (b) Paketverschachtelung.

RTP wird normalerweise im Benutzerbereich über UDP (im Betriebssystem) ausgeführt. Es arbeitet wie folgt: Die Multimedia-Anwendung besteht aus mehreren Audio-, Video-, Text- und eventuell noch anderen Strömen. Diese werden in die RTP-Bibliothek eingefügt, die sich zusammen mit der Anwendung im Adressraum des Benutzers befindet. Diese Bibliothek multiplext dann die Ströme, codiert sie in RTP-Pakete und legt sie auf einem Socket ab. Am anderen Ende des Sockets (im Betriebssystemkern) werden UDP-Pakete erzeugt, in die die RTP-Pakete eingebettet werden, und an IP übergeben, damit sie beispielsweise über Ethernet übertragen werden können. Auf Empfängerseite findet der umgekehrte Prozess statt. Zum Schluss empfängt die Multimedia-Anwendung die Multimediadaten aus der RTP-Bibliothek. Sie ist für das Abspielen der Mediendaten verantwortlich. Der Protokollstapel für diese Situation wird in ► Abbildung 6.30a dargestellt, die Paketverschachtelung in ► Abbildung 6.30b.

Bei diesem Design kann man nicht so einfach sagen, in welcher Schicht sich RTP befindet. Da es im Benutzerbereich läuft und mit der Anwendung verknüpft ist, sieht es auf den ersten Blick wie ein Anwendungsprotokoll aus. Andererseits ist es ein generisches, anwendungsunabhängiges Protokoll, das nur die Transporteinrichtungen zur Verfügung stellt, sodass es auch als Transportprotokoll betrachtet werden kann. Die wahrscheinlich beste Beschreibung ist, dass es ein Transportprotokoll ist, das nur zufällig in der Anwendungsschicht implementiert ist, weshalb wir es auch in diesem Kapitel behandeln wollen.

RTP – Das Real-time Transport Protocol

Die Hauptfunktion von RTP ist das Multiplexen verschiedener Echtzeitdatenströme in einen einzigen Strom von UDP-Paketen. Der UDP-Strom kann an ein einziges Ziel

(Unicasting) oder an mehrere Ziele (Multicasting) gesendet werden. Da RTP nur normales UDP verwendet, werden die Pakete von den Routern nicht besonders behandelt, außer es sind einige IP-Funktionen zur Unterstützung der Dienstgüte aktiviert. Insbesondere gibt es keine besonderen Garantien bezüglich der Zustellung, d.h. Pakete können verloren gehen, verspätet zugestellt werden, beschädigt sein, usw.

Das RTP-Format verfügt über mehrere Funktionen, die Empfängern die Arbeit mit Multimediadaten erleichtern. Jedes in einem Strom von RTP gesendete Paket erhält eine Nummer, die um eins höher ist als die des Vorgängers. Anhand dieser Nummerierung kann das Ziel feststellen, ob Pakete fehlen. Fehlt ein Paket, obliegt es der Anwendung zu entscheiden, welche Maßnahme das Ziel ergreifen soll. Im Fall von Videodaten kann zum Beispiel ein Video-Frame übersprungen oder im Fall von Audiodaten der fehlende Wert durch Interpolation näherungsweise bestimmt werden. Die erneute Übertragung bietet sich nicht an, da dieses Paket höchstwahrscheinlich viel zu spät ankommt, um noch nützlich zu sein. Aus diesem Grund gibt es bei RTP keine Bestätigungen und auch keinen Mechanismus, um erneute Übertragungen anzufordern.

Alle Nutzdaten in RTP können mehrere Samples enthalten, die so codiert sein können, wie dies die Anwendung benötigt. Für ein reibungsloses Zusammenspiel definiert RTP verschiedene Profile (beispielsweise einen einfachen Audiostrom) und für jedes Profil können mehrere Codierungsformate erlaubt sein. So kann beispielsweise ein einzelner Audiostrom als 8-Bit-PCM-Samples mit 8 kHz, einer Deltacodierung, einer prädiktiven Codierung, GSM, MP3 usw. codiert werden. RTP enthält ein Header-Feld, in dem die Quelle die Codierung angeben kann, aber nicht daran beteiligt ist, wie die Codierung erfolgt.

Eine weitere Funktion, die viele Echtzeitanwendungen benötigen, sind Zeitstempel. Die Quelle soll die Möglichkeit haben, dem ersten Sample in jedem Paket einen Zeitstempel mitzugeben. Die Zeitstempel sind relativ zum Beginn eines Datenstroms, sodass nur die Unterschiede zwischen den Zeitstempeln relevant sind. Die Absolutwerte haben keine Bedeutung. Wie wir gleich beschreiben werden, erlaubt dieser Mechanismus dem Ziel, einen kleinen Puffer einzurichten und jedes Sample die korrekte Zahl Millisekunden nach Start des Datenstroms abzuspielen, unabhängig von der Ankunftszeit des Samples.

Das Zeitstempelverfahren reduziert nicht nur die Auswirkungen der schwankenden Netzverzögerung, sondern erlaubt auch, mehrere Ströme miteinander zu synchronisieren. So kann beispielsweise ein digitales Fernsehprogramm aus einem Videodatenstrom und zwei Audiodatenströmen bestehen. Die beiden Audiodatenströme können zur Stereoübertragung oder zur Bereitstellung des Films in Originalsprache und einer lokalen, vom Benutzer auszuwählenden Sprache dienen. Jeder Datenstrom stammt von einem anderen physikalischen Gerät. Wenn die Datenströme aber von einem einzelnen Zähler mit einem Zeitstempel versehen werden, können sie synchron wiedergegeben werden, selbst wenn die Datenströme etwas unregelmäßig übertragen beziehungsweise empfangen wurden.

► Abbildung 6.31 zeigt den RTP-Header. Er besteht aus drei 32-Bit-Wörtern und einigen potenziellen Erweiterungen. Das erste Wort enthält das Feld *Version*, das bereits auf 2 steht. Hoffen wir einmal, dass diese Version der endgültigen Version sehr nahe kommt, da nur noch ein Codewert übrig ist (obwohl 3 so definiert werden könnte, dass die echte Version in einem Erweiterungswort steht).

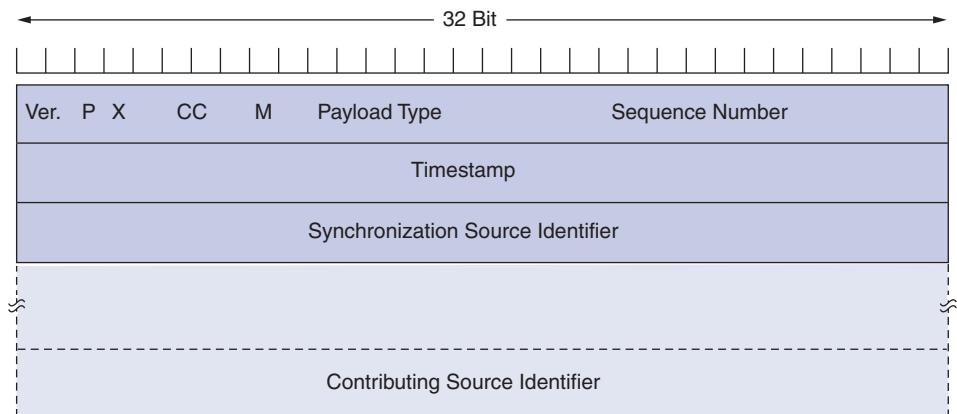


Abbildung 6.31: Der RTP-Header.

Das Bit *P* gibt an, dass das Paket auf ein Vielfaches von 4 Byte aufgefüllt wurde. Das letzte Füllbyte gibt an, wie viele Byte hinzugefügt wurden. Das Bit *X* gibt an, dass ein Erweiterungs-Header vorhanden ist. Das Format und die Bedeutung des Erweiterungs-Headers sind nicht definiert. Es ist nur definiert, dass das erste Wort in der Erweiterung die Länge angibt. Dies ist ein Notausgang für unvorhergesehene Erfordernisse.

Das Feld *CC* gibt an, wie viele Quellen Daten liefern (von 0 bis 15, siehe unten). Das Bit *M* ist ein anwendungsspezifisches Markierungsbit. Es kann zur Markierung des Beginns eines Videorahmens, eines Wortes in einem Audiokanal oder etwas anderem verwendet werden, das die Anwendung versteht. Das Feld *Payload Type* gibt an, welcher Codieralgorithmus verwendet wurde (wie nicht komprimiertes 8-Bit-Audio, MP3 usw.). Da jedes Paket dieses Feld überträgt, kann sich die Codierung während der Übertragung ändern. Die *Sequence Number* ist nur ein Zähler, der bei jedem gesendeten RTP-Paket inkrementiert wird. Anhand der Sequenznummern kann festgestellt werden, ob Pakete verloren gegangen sind.

Der *Timestamp* wird von der Quelle des Datenstroms erzeugt, um anzugeben, wann das erste Sample im Paket erstellt wurde. Mit diesem Wert kann der Jitter beim Empfänger reduziert werden, indem die Wiedergabe von der Paketankunftszeit losgelöst wird. Das Feld *Synchronization Source Identifier* gibt an, zu welchem Datenstrom das Paket gehört. Mit diesem Verfahren werden mehrere Datenströme in einen Strom von UDP-Paketen gemultiplext und gedemultiplext. Und zum Schluss werden die *Contributing Source Identifier* gesetzt, wenn im Studio Mischpulte verwendet werden. In diesem Fall ist das Mischpult die synchronisierende Quelle, und die gemischten Ströme werden hier aufgeführt.

RTCP – Das Real-time Transport Control Protocol

RTP hat ein kleines Schwesternprotokoll, das **Echtzeittransportsteuerungsprotokoll** namens **RTCP** (*Real-time Transport Control Protocol*). Es ist zusammen mit RTP in RFC 3550 definiert und verwaltet das Feedback, die Synchronisation und die Benutzerschnittstelle. Es überträgt aber keine Daten.

Mit der ersten Funktion kann den Quellen Feedback über Übertragungsverzögerung, Jitter, Bandbreite, Überlastung und andere Netzeigenschaften zur Verfügung gestellt werden. Anhand dieser Informationen kann ein Codierprozess die Datenübertragungsrate erhöhen (und eine bessere Qualität bereitstellen), wenn das Netz gut funktioniert, beziehungsweise andernfalls die Datenübertragungsraten reduzieren. Durch die kontinuierlichen Feedbacks kann der Codieralgorithmus laufend angepasst werden, um die unter den aktuellen Umständen bestmögliche Qualität zu bieten. Wenn beispielsweise die Bandbreite während einer Übertragung zu- oder abnimmt, kann die Codierung je nach Bedarf von MP3 auf 8-Bit-PCM auf Deltacodierung wechseln. Das Feld *Payload Type* gibt dem Ziel an, welcher Codieralgorithmus für das aktuelle Paket verwendet wird, sodass er bei Bedarf geändert werden kann.

Ein Problem des Feedbacks ist, dass die RTCP-Berichte an alle Beteiligte gesendet werden. Bei einer Multicast-Anwendung mit einer großen Gruppe würde die von RTCP verwendete Bandbreite schnell sehr groß werden. Um dies zu verhindern, reduzieren RTCP-Sender den Anteil ihrer Berichte, um zusammen nicht mehr als sagen wir 5 % der Medienbandbreite zu belegen. Hierzu muss jeder Beteiligte die Medienbandbreite kennen, die er vom Sender erfährt, und die Anzahl der Beteiligten, die er durch Lauschen auf andere RTCP-Berichte schätzt.

RTCP verwaltet auch die Synchronisation zwischen Datenströmen. Das Problem ist hier, dass verschiedene Datenströme verschiedene Uhren mit unterschiedlicher Auflösung (Granularität) und verschiedenen Abweichungsraten verwenden. Mit dem Echtzeit-Transportsteuerungsprotokoll können diese synchron gehalten werden.

Schließlich ermöglicht RTCP, die verschiedenen Quellen zu benennen (z.B. in ASCII). Diese Information kann dann am Bildschirm des Empfängers anzeigen, wer gerade „spricht“.

In Perkins (2003) findet der Leser weitere Informationen zu RTP.

Abspielen mit Zwischenspeicherung und Jitter-Kontrolle

Sobald die Mediendaten den Empfänger erreichen, müssen sie zum richtigen Zeitpunkt abgespielt werden. Im Allgemeinen ist dies nicht der Moment, an dem das RTP-Paket beim Empfänger ankommt, da Pakete leicht unterschiedliche Zeiten benötigen, um das Netz zu durchqueren. Sogar wenn die Pakete mit genau den richtigen Intervallen ins Netz geschoben werden, weisen sie beim Empfänger verschiedene relative Zeiten auf. Diese Varianz der Laufzeit wird **Jitter** genannt. Bereits geringfügiges Paket-Jitter reicht aus, um verwirrende Medienartefakte zu schaffen, wie abgehackte Bildübertragung und unverständliche Audiodaten, wenn die Medien direkt nach dem Eintreffen abgespielt werden.

Die Lösung zu diesem Jitter-Problem besteht darin, die Pakete in einem **Puffer** zwischenzuspeichern, bevor sie abgespielt werden. Als Beispiel sehen wir in ► Abbildung 6.32 einen Strom von Paketen, die mit beträchtlichem Jitter zugestellt werden. Paket 1 wird vom Server bei $t=0$ Sekunden verschickt und kommt beim Client bei $t=1$ Sekunde an. Paket 2 verzögert sich etwas und benötigt 2 Sekunden für die Strecke. Sobald die Pakete eintreffen, werden sie auf dem Client-Rechner gepuffert.

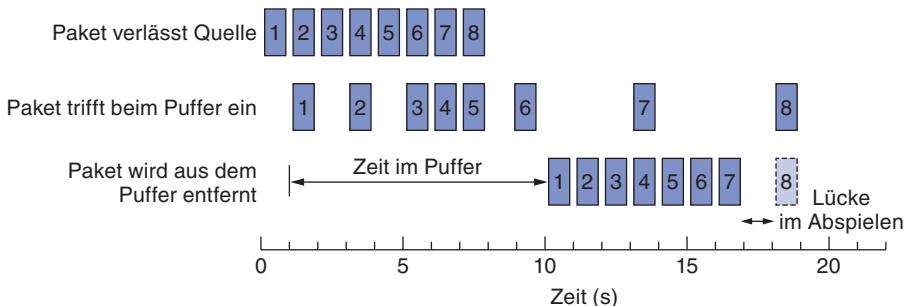


Abbildung 6.32: Den Ausgabestrom durch Puffern von Paketen glätten.

Bei $t=10$ Sekunden wird mit dem Abspielen begonnen. Bis dahin wurden die Pakete 1–6 gespeichert, die jetzt mit gleich großen Intervallen aus dem Puffer entfernt werden können, um reibungslose Übergänge zu gewährleisten. Im Allgemeinen ist es nicht notwendig, gleichmäßige Intervalle zu verwenden, da die RTP-Zeitstempel genau vorgeben, wann die Medienpakete abgespielt werden sollen.

Leider müssen wir feststellen, dass Paket 8 sich so stark verspätet hat, dass es noch nicht eingetroffen ist, wenn es mit dem Abspielen an der Reihe ist. In diesem Fall gibt es zwei Optionen. Der Player kann Paket 8 überspringen und mit den nachfolgenden Paketen fortfahren. Oder der Abspielprozess kann unterbrochen werden, bis Paket 8 ankommt, was zu ärgerlichen Lücken in der Musik oder dem Film führt. Bei einer Live-Anwendung wie einem Telefonat über das Internet wird das Paket normalerweise übersprungen, denn für Live-Anwendungen ist es nicht so gut, wenn sie in einer Warteschleife gesetzt werden. In einer Streaming-Anwendung jedoch würde der Player wahrscheinlich anhalten. Dieses Problem ließe sich etwas weiter reduzieren, indem man die Startzeit durch Verwendung eines größeren Puffers noch etwas weiter nach hinten hinausschiebt. Für Player zum Streamen von Audio- und Videodaten werden oft Puffer von ungefähr 10 Sekunden verwendet, um sicherzustellen, dass der Empfänger alle Pakete (die nicht im Netz verloren gegangen sind) rechtzeitig erhält. Für Live-Anwendungen wie beispielsweise Videokonferenzen sind hingegen nur kleine Puffer nötig, die der Reaktionsbereitschaft dienen.

Eine der wichtigsten Überlegungen beim reibungslosen Abspielen gilt dem **Abspielstart** beziehungsweise wie lange der Empfänger auf die Medien warten muss, bevor mit dem Abspielen begonnen wird. Diese Entscheidung hängt vom Jitter ab. Der Unterschied zwischen einer Verbindung mit niedrigem und einer mit hohem Jitter wird in ► Abbildung 6.33 deutlich. Auch wenn die durchschnittlichen Verzögerungen der beiden nur wenig voneinander abweichen, muss bei einer Verbindung mit hohem

Jitter der Abspielstart viel weiter nach hinten gesetzt werden, um 99 % der Pakete zu umfassen, als dies bei einer Verbindung mit niedrigem Jitter nötig ist.

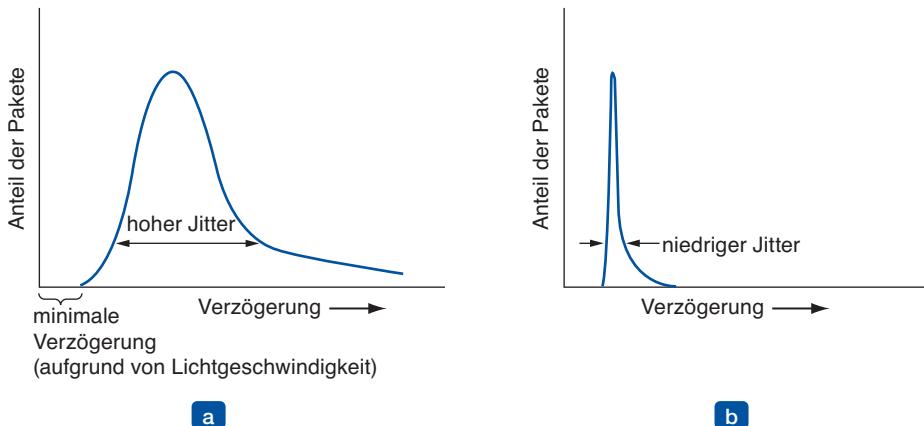


Abbildung 6.33: (a) Hoher Jitter. (b) Niedriger Jitter.

Für die Wahl eines günstigen Abspielstartpunktes kann die Anwendung den Jitter messen, indem sie den Unterschied zwischen den RTP-Zeitstempeln und die Ankunftszeit ermittelt. Jeder Unterschied liefert ein Beispiel der Verzögerung (einschließlich eines beliebigen, festen Offsets). Allerdings kann sich die Verzögerung über die Zeit aufgrund starken Verkehrsaufkommens und wechselnder Routen ändern. Um diesen Änderungen Rechnung zu tragen, können Anwendungen ihren Abspielstartpunkt sogar während der Ausführung verschieben. Wenn das Verschieben dieses Zeitpunktes jedoch nicht professionell erfolgt, kann es bei der Wiedergabe zu einem kleinen Aussetzer kommen. Bei Audiodaten können Sie dieses Problem vermeiden, indem Sie die Anpassung des Abspielstartpunktes zwischen **Talkspurts**, d.h. in den Lücken einer Konversation, vornehmen. Keiner wird den Unterschied zwischen einer kurzen und einer etwas längeren Redepause bemerken. RTP erlaubt es deshalb den Anwendungen, mithilfe des Markierungs-Bits *M* den Anfang eines neuen Talkspurts zu kennzeichnen.

Wenn die absolute Verzögerung bis zum Abspielen der Mediendaten zu lang ist, leiden Live-Anwendungen darunter. Wird bereits ein direkter Pfad verwendet, gibt es keine Möglichkeit mehr, die Ausbreitungsverzögerung zu reduzieren. Wenn es nichts ausmacht, dass ein Teil der Pakete zu spät zum Abspielen eintrifft, kann der Abspielstartpunkt vorgezogen werden. Ansonsten besteht die einzige Möglichkeit zur Verkürzung des Abspielstartpunktes darin, den Jitter mittels einer bessere Dienstgüte zu reduzieren, z.B. den *Expedited Forwarding Differentiated Service* (kurz: es wird ein besseres Netz benötigt).

6.5 Internettransportprotokolle: TCP

UDP ist ein einfaches Protokoll, das einige sehr wichtige Einsatzbereiche hat, wie Client-Server-Interaktionen und Multimedia. Bei den meisten Internetanwendungen ist jedoch eine zuverlässige Zustellung erforderlich, die die Reihenfolge der Pakete einhält. Da UDP dies nicht leisten kann, ist ein anderes Protokoll vonnöten. Und genau hier kommt TCP ins Spiel. Es ist das Standardprotokoll im Internet. Befassen wir uns damit nun genauer.

6.5.1 Einführung in TCP

Das **Übertragungssteuerungsprotokoll TCP** (*Transmission Control Protocol*) wurde speziell zur Bereitstellung eines zuverlässigen Bytestroms von Endpunkt zu Endpunkt in einem unzuverlässigen Internetwork entwickelt. Ein Internetwork unterscheidet sich von einem Einzelnets dahingehend, dass verschiedene Teile völlig unterschiedliche Topologien, Bandbreiten, Übertragungsverzögerungen, Paketgrößen und andere Parameter haben können. TCP wurde so entwickelt, dass es sich an die verschiedenen Eigenschaften eines Internetworks dynamisch anpasst und robust auf alle möglichen Arten von Fehlern reagiert.

Im September 1981 wurde TCP in RFC 793 formal definiert. Im Laufe der Zeit wurden viele Verbesserungen vorgenommen und etliche Fehler und Inkonsistenzen beseitigt. Um Ihnen eine Vorstellung von dem Umfang von TCP zu geben, sei erwähnt, dass es neben RFC 793 inzwischen noch eine ganze Reihe anderer RFCs gibt, die erforderlich geworden sind: RFC 1122 (Verbesserungen und einige Fehlerbehebungen), RFC 1323 (Erweiterungen zur Leistungsverbesserung), RFC 2018 (selektive Bestätigungen), RFC 2581 (Überlastungsüberwachung), RFC 2873 (Umnutzung der Header-Felder für Dienstgüte), RFC 2988 (verbesserte Timer für die erneute Übertragung) und RFC 3168 (explizite Überlastungsbenachrichtigung). Das sind aber noch längst nicht alle, sodass es einen Leitfaden zu den vielen RFCs gibt, der natürlich ebenfalls als RFC-Dokument (RFC 4614) veröffentlicht wurde.

Jeder Rechner, der TCP unterstützt, verfügt über eine TCP-Transportinstanz, entweder in Form einer Bibliotheksprozedur, eines Benutzerprozesses oder, was am häufigsten vorkommt, als Teil des Betriebssystemkerns. In jedem Fall verwaltet sie TCP-Ströme und Schnittstellen zur IP-Schicht. Eine TCP-Instanz akzeptiert Benutzerdatenströme aus lokalen Prozessen, zerlegt diese in Blöcke bis maximal 64 KB (in der Praxis oftmals 1 460 Datenbyte, damit sie zusammen mit den IP- und TCP-Headern in einen einzelnen Ethernet-Rahmen passen) und sendet jeden Block als eigenes IP-Datagramm. Kommen IP-Datagramme mit TCP-Daten bei einem Rechner an, werden sie an die TCP-Instanz weitergereicht, die die ursprünglichen Byteströme wieder zusammensetzt. Der Einfachheit halber verwenden wir im weiteren Verlauf „TCP“ manchmal anstelle von „TCP-Transportinstanz“ (Software) oder anstelle von „TCP-Protokoll“. Aus dem Zusammenhang geht jedoch hervor, was jeweils gemeint ist. Dem Satz „Der Benutzer gibt Daten an TCP weiter“ kann z.B. direkt entnommen werden, dass die TCP-Transportinstanz gemeint ist.

Die IP-Schicht gibt keinerlei Gewähr, dass Datagramme richtig zugestellt werden, noch verrät sie, wie schnell die Datagramme gesendet werden können. Deshalb obliegt es TCP, die Datagramme schnell genug zu senden, um die Kapazität voll auszunutzen, ohne das Netz zu überlasten, und bei Zeitüberschreitung die Datagramme bei Bedarf erneut zu übertragen. Andererseits ist es auch möglich, dass Datagramme zwar ankommen, aber in der falschen Reihenfolge. In diesem Fall werden sie von TCP wieder in die richtige Reihenfolge gebracht. Kurz gesagt, TCP muss die Zuverlässigkeit bieten, die von den meisten Benutzern erwartet wird und die IP nicht liefern kann.

6.5.2 TCP-Dienstmodell

Für einen funktionierenden TCP-Dienst müssen sowohl der Sender als auch der Empfänger Endpunkte, die sogenannten **Sockets** (siehe Abschnitt 6.1.3), einrichten. Jeder Socket trägt eine Socket-Nummer (Adresse), die aus einer IP-Adresse des Hosts und einer lokalen 16-Bit-Nummer des Hosts, als **Port** bezeichnet, besteht. Ein Port ist der TCP-Name für einen TSAP. Für den Zugriff auf einen TCP-Dienst muss explizit eine Verbindung zwischen dem Socket des sendenden Rechners und dem Socket des empfangenden Rechners aufgebaut werden. Die Socketaufrufe sind in Abbildung 6.5 aufgeführt.

Ein Socket kann gleichzeitig für mehrere Verbindungen verwendet werden. Mit anderen Worten, zwei oder mehr Verbindungen enden auf dem gleichen Socket. Verbindungen werden anhand der Socket-Bezeichner an beiden Enden identifiziert, d.h. (*socket1*, *socket2*). Virtuelle Verbindungsnummern oder andere Bezeichner werden nicht benutzt.

Portnummern unter 1 024 sind für Standarddienste reserviert, die normalerweise nur von privilegierten Benutzern (z.B. *root* in Unix-Systemen) gestartet werden können. Diese Ports nennt man **reservierte Ports** (*well-known port*). Beispielsweise kann jeder Prozess, der aus der Ferne die Mail von einem Host abrufen will, eine Verbindung zu dem Port 143 des Ziel-Hosts aufbauen, um dessen IMAP-Dämon zu kontaktieren. Eine Liste der reservierten Ports findet man unter www.iana.org. Es wurden bereits über 700 Ports Nummern zugewiesen. Einige der bekannteren werden in ►Abbildung 6.34 aufgelistet.

Weitere Ports von 1 024 bis 49 151 können bei IANA zur Benutzung durch nicht privilegierte Benutzer registriert werden. Doch Anwendungen können durchaus ihre eigenen Ports wählen und machen dies auch. Zum Beispiel verwendet die Peer-to-Peer-Filesharing-Anwendung BitTorrent (inoffiziell) die Ports 6 881-6 887, kann jedoch auch auf anderen Ports ausgeführt werden.

Natürlich wäre es auch möglich, dafür zu sorgen, dass sich der FTP-Dämon beim Booten mit Port 21 verbindet oder der SSH-Dämon mit Port 22 usw. Dies würde aber den Hauptspeicher mit Dämonen zumüllen, die sich die meiste Zeit im Leerlauf befinden. Stattdessen wird ein einziger Dämon, in Unix der sogenannte **inetd** (**Internetdämon**), verwendet, der sich selbst mit mehreren Ports verbindet und auf die erste eingehende Verbindung wartet.

| Port | Protokoll | Verwendung |
|--------|-----------|----------------------------------|
| 20, 21 | FTP | Dateiübertragung |
| 22 | SSH | Fern-Login, Ersatz für Telnet |
| 25 | SMTP | E-Mail |
| 80 | HTTP | World Wide Web |
| 110 | POP-3 | Fernzugriff auf E-Mail |
| 143 | IMAP | Fernzugriff auf E-Mail |
| 443 | HTTPS | Sicheres Web (HTTP über SSL/TLS) |
| 543 | RTSP | Mediaplayer-Steuerung |
| 631 | IPP | Gemeinsame Druckernutzung |

Abbildung 6.34: Einige reservierte Ports.

Sobald eine eingeht, startet *inetd* einen neuen Prozess und führt darin den entsprechenden Dämon aus, der diese Anfrage behandelt. Auf diese Weise sind neben *inetd* nur dann andere Dämonen aktiv, wenn für sie Arbeit da ist. Aus einer Konfigurationsdatei erfährt *inetd*, welche Ports verwendet werden sollen. Daher kann der Systemadministrator ein System so einrichten, dass permanente Dämonen an den meistbeschäftigen Ports (wie Port 80) Anfragen entgegennehmen und *inetd* auf dem Rest läuft.

Alle TCP-Verbindungen sind Duplex- und Punkt-zu-Punkt-Verbindungen. Duplex bedeutet, dass der Verkehr gleichzeitig in beide Richtungen fließen kann. Punkt-zu-Punkt bedeutet, dass jede Verbindung genau zwei Endpunkte hat. TCP unterstützt weder Multicasting noch Broadcasting.

Eine TCP-Verbindung ist ein Bytestrom, kein Nachrichtenstrom. Die Grenzen von Nachrichten werden nicht von Endpunkt zu Endpunkt beibehalten. Schickt z.B. ein sendender Prozess vier Blöcke mit je 512 Byte als TCP-Strom, können diese Daten am anderen Ende als vier 512-Byte-Blöcke, zwei 1 024-Byte-Blöcke oder ein 2 048-Byte-Block empfangen werden (► Abbildung 6.35). Es gibt für den Empfänger keine Möglichkeit zu erkennen, in welchen Einheiten die Daten geschrieben wurden, so sehr er sich auch bemüht.

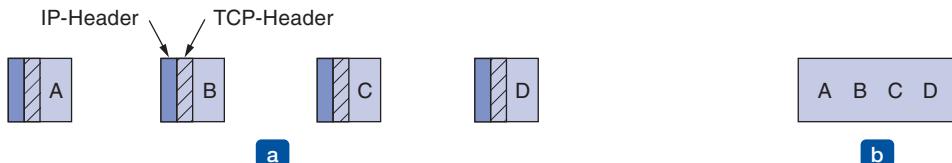


Abbildung 6.35: (a) Vier 512-Byte-Segmente als einzelne IP-Datagramme. (b) Die 2 048-Byte-Daten werden der Anwendung in einem einzigen READ-Aufruf zugestellt.

Dateien in Unix haben ebenfalls diese Eigenschaft. Der Leser einer Datei kann nicht feststellen, ob die Datei blockweise, byteweise oder in einem Zuge geschrieben wurde. Wie bei einer Unix-Datei hat die TCP-Software keine Ahnung, was die Bytes bedeuten, und auch kein Interesse daran, das herauszufinden. Ein Byte ist einfach ein Byte.

Übergibt eine Anwendung Daten an TCP, kann TCP sie nach freiem Ermessen sofort weitergeben oder zwischenspeichern (um eine größere Menge anzusammeln und dann gemeinsam zu versenden). Manche Anwendungen bedingen, dass die Daten sofort weitergegeben werden. Nehmen wir z.B. an, ein Teilnehmer an einem interaktiven Spiel möchte einen Strom an Aktualisierungen senden. Es ist wichtig, dass die Aktualisierungen direkt gesendet werden und nicht in einem Puffer zwischengespeichert werden, um mehrere Werte gleichzeitig übertragen zu können. Um das Absenden der Daten zu erzwingen, verfügt TCP über ein sogenanntes PUSH-Flag, das bei den Paketen gesetzt wird. Ursprünglich war die Absicht, den Anwendungen die Möglichkeit einzuräumen, TCP-Implementierungen über das PUSH-Flag mitzuteilen, die Übertragung nicht zu verzögern. Allerdings können Anwendungen beim Senden der Daten nicht wirklich das PUSH-Flag setzen. Deshalb haben die verschiedenen Betriebssysteme verschiedene Optionen entwickelt, um die Übertragung zu beschleunigen (z.B. TCP_NODELAY unter Windows und Linux).

Für Internetarchäologen wollen wir hier ein weiteres Merkmal des TCP-Dienstes erwähnen, das es noch im Protokoll gibt, aber selten genutzt wird: **Vorrangdaten** (*urgent data*). Wenn eine Anwendung sehr wichtige Daten hat, die direkt verarbeitet werden müssen, zum Beispiel wenn ein Benutzer in einer interaktiven Anwendung die Tasten STRG-C drückt, um eine bereits eingeleitete entfernte Berechnung abzubrechen, kann die sendende Anwendung entsprechende Steuerinformationen im Datenstrom schicken und sie mit dem URGENT-Flag an TCP weiterreichen. Durch dieses Ereignis stoppt TCP das Sammeln von Daten und überträgt sofort alles, was für die betreffende Verbindung anliegt.

Gehen mit URGENT gekennzeichnete Daten am Ziel ein, wird die empfangende Anwendung unterbrochen (im Unix-Jargon ausgedrückt, erhält sie ein Signal). Sie unterbricht alles, was sie gerade bearbeitet, und liest den Datenstrom mit den Vorrangdaten. Das Ende der Vorrangdaten ist gekennzeichnet, sodass die Anwendung direkt weiß, wann Schluss ist. Der Beginn der Vorrangdaten ist hingegen nicht gekennzeichnet. Den Anfang zu erkennen, bleibt der Anwendung überlassen.

Diese Vorgehensweise entspricht im Grunde einem groben Signalisierungsmechanismus und überlässt alles andere der Anwendung. Zwar trifft es zu, dass Vorrangdaten durchaus nützlich sein können, doch zeigte sich schon früh, dass sie nicht zwingend erforderlich waren, sodass sie mehr und mehr in Vergessenheit gerieten. Von ihrer Verwendung wird inzwischen aufgrund von Implementierungsunterschieden abgeraten, sodass Anwendungen für ihre eigene Signalisierung verantwortlich sind. Vielleicht werden zukünftige Transportprotokolle bessere Signalisierungsunterstützung bieten.

6.5.3 TCP-Protokoll

Dieser Abschnitt enthält einen allgemeinen Überblick über das TCP-Protokoll. Im nächsten Abschnitt wird der Protokoll-Header Feld für Feld beschrieben.

Ein zentrales Charakteristikum von TCP ist, dass jedes Byte in einer TCP-Verbindung eine eigene 32-Bit-Sequenznummer besitzt, was nicht zuletzt das gesamte Protokoll-Design prägt. In den Anfangszeiten des Internets waren die Leitungen zwischen den Routern in der Regel 56-kbit/s-Standleitungen. Wenn ein Host in voller Geschwindigkeit Daten über die Leitung übertrug, dauerte es über eine Woche, alle Sequenznummern zu durchlaufen. Bei den aktuellen Netzgeschwindigkeiten können die Sequenznummern in alarmierender Geschwindigkeit verbraucht werden, wie wir später noch sehen werden. Eigene 32-Bit-Sequenznummern werden Paketen für die Schiebefensterposition in die eine Richtung und für Bestätigungen in die entgegengesetzte Richtung mitgegeben (siehe unten).

Die sendenden und empfangenden TCP-Instanzen tauschen Daten in Form von Segmenten aus. Ein **TCP-Segment** besteht aus einem festen 20-Byte-Header (sowie einem optionalen Teil), gefolgt von null oder mehr Datenbytes. Die TCP-Software entscheidet über die Größe der Segmente. Sie kann Daten von mehreren Schreiboperationen in einem Segment zusammenfassen oder Daten aus einem Schreibvorgang auf mehrere Segmente aufteilen. Die Segmentgröße wird durch zwei Faktoren begrenzt. Erstens muss jedes Segment, einschließlich des TCP-Headers, in das IP-Nutzdatenfeld von 65 515 Byte passen. Zweitens hat jedes Netz eine **maximale Übertragungseinheit (MTU, Maximum Transfer Unit)**, in die jedes Segment passen muss. Ein Segment darf nicht größer sein als die maximale Übertragungseinheit beim Sender und Empfänger, sodass es unzerlegt in einem Paket gesendet und empfangen werden kann. In der Praxis ist die maximale Übertragungseinheit 1 500 Byte groß (die Größe der Ethernet-Nutzlast) – ein Wert, der die Obergrenze der Segmentgröße definiert.

Es ist jedoch immer noch möglich, IP-Pakete, die TCP-Segmente übertragen, zu zerlegen, wenn sie einen Netzpfad durchlaufen, der eine Verbindung mit einer kleinen maximalen Übertragungseinheit aufweist. Wenn dies geschieht, sinkt die Leistung und andere Probleme tauchen auf (Kent und Mogul, 1987). Moderne TCP-Implementierungen führen stattdessen **Path MTU Discovery** durch und verwenden dazu die in RFC 1191 definierte Methode, die wir in Abschnitt 5.5.5 beschrieben haben. Diese Methode verwendet ICMP-Fehlermeldungen, um nach der kleinsten Übertragungseinheit für jede Verbindung auf dem Pfad zu suchen. TCP passt dann die Segmentgröße nach unten an, um eine Fragmentierung zu vermeiden.

Das grundlegende Protokoll, das von TCP-Instanzen verwendet wird, ist das Schiebefensterprotokoll mit dynamischer Fenstergröße. Überträgt ein Sender ein Segment, startet er gleichzeitig einen Timer. Kommt das Segment am Ziel an, sendet die empfangende TCP-Instanz ein Segment (mit Daten, falls vorhanden, andernfalls ohne) mit einer Bestätigungsnummer zurück, die der nächsten erwarteten Sequenznummer entspricht, sowie die verbleibende Fenstergröße. Läuft der Timer des Senders ab, bevor die Bestätigung eingeht, überträgt der Sender das Segment erneut.

Das Protokoll klingt einfach, hat aber einige Besonderheiten, auf die wir im Folgenden noch eingehen werden: Manche Segmente kommen eventuell nicht in der gewünschten Reihenfolge an, z.B. kann es vorkommen, dass die Bytes 3 072–4 095 ankommen, aber nicht bestätigt werden können, weil die Bytes 2 048–3 071 noch fehlen. Oder Segmente werden auf dem Transit so lange verzögert, dass der Timer des Senders inzwischen abläuft und die Segmente erneut übertragen werden. Die erneute Übertragung kann andere Bytebereiche enthalten als die ursprüngliche Übertragung, sodass hier eine sorgfältige Administration erforderlich ist, um zu verfolgen, welche Bytes bislang korrekt empfangen wurden. Dies ist möglich, da jedes Byte im Datenstrom einen eigenen, eindeutigen Offset hat.

TCP muss diese Probleme behandeln und effizient lösen können. Ein beträchtlicher Aufwand ist in die Optimierung der Leistung von TCP-Strömen geflossen, sogar angesichts von Netzproblemen. In den nächsten Abschnitten werden einige der Algorithmen beschrieben, die in vielen TCP-Implementierungen benutzt werden.

6.5.4 TCP-Header

► Abbildung 6.36 zeigt das Layout eines TCP-Segments. Jedes Segment beginnt mit einem 20-Byte-Header, der ein standardisiertes Format aufweist. Auf diesen Header können Header-Optionen folgen. Den Optionen, falls vorhanden, können bis zu $65\ 535 - 20 - 20 = 65\ 495$ Datenbyte folgen, wobei sich die ersten 20 auf den IP-Header und die zweiten 20 auf den TCP-Header beziehen. Segmente ohne Daten sind zulässig und werden normalerweise für Bestätigungen und Steuernachrichten verwendet.

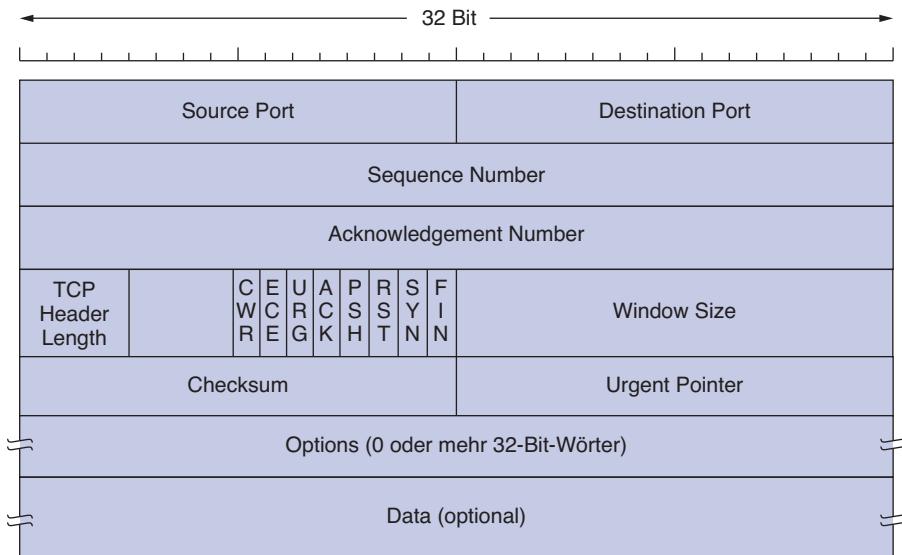


Abbildung 6.36: Der TCP-Header.

Lassen Sie uns im Folgenden den TCP-Header Feld für Feld durchgehen. Die Felder *Source Port* und *Destination Port* geben die lokalen Endpunkte der Verbindung an. Ein

TCP-Port plus die IP-Adresse seines Hosts bilden zusammen einen eindeutigen 48-Bit-Endpunkt. Die Endpunkte der Quelle und des Ziels identifizieren zusammen die Verbindung. Dieser Verbindungsidentifikator wird **5-Tupel** genannt, weil er aus fünf Informationen besteht: das Protokoll (TCP), Quell-IP und Quellport sowie Ziel-IP und Zielport.

Die Felder *Sequence Number* und *Acknowledgement Number* führen die üblichen Funktionen aus. Beachten Sie, dass mit der Bestätigungsnummer das nächste erwartete Byte angegeben wird und nicht das zuletzt korrekt empfangene Byte. Es ist eine **kumulative Bestätigung**, weil sie die empfangenen Daten in einer einzigen Zahl zusammenfasst. Sie geht nicht über verlorene Daten hinaus. Beide Felder sind 32 Bit lang, weil in einem TCP-Strom jedes Datenbyte nummeriert ist.

Das Feld *TCP Header Length* gibt an, wie viele 32-Bit-Wörter im TCP-Header enthalten sind. Diese Information ist erforderlich, weil das *Options*-Feld eine variable Länge hat und somit auch der Header. Dieses Feld gibt technisch den Start der Daten im Segment an, gemessen in 32-Bit-Wörtern. Diese Zahl gibt nur die Länge des Headers in Wörtern an, sodass das Ergebnis quasi das gleiche ist.

Als Nächstes folgt ein 4-Bit-Feld, das nicht genutzt wird. Die Tatsache, dass diese Bits seit über 30 Jahren nicht genutzt werden (nur 2 der ursprünglich 6 reservierten Bits wurden zurückfordert), belegt, wie gut durchdacht das TCP-Design ist. Mittelmäßige Protokolle hätten diese Bits benötigt, um Fehler im ursprünglichen Design auszubügeln.

Danach folgen acht 1-Bit-Flags. *CWR* und *ECE* werden verwendet, um wie in RFC 3168 definiert bei der Verwendung von *ECN (Explicit Congestion Notification)* Überlastung zu signalisieren. *ECE* wird gesetzt, um über einem *ECN-Echo* einem TCP-Sender mitzuteilen, langsamer zu werden, wenn der TCP-Empfänger eine Überlastungsanzeige vom Netz erhält. *CWR* wird gesetzt, um ein verkleinertes Überlastungsfenster (*Congestion Window Reduced*) vom TCP-Sender an den TCP-Empfänger zu signalisieren, sodass er weiß, dass der Sender langsamer geworden ist und er aufhören kann, das *ECN-Echo* zu senden. Wir besprechen die Rolle von *ECN* im Zusammenhang mit der TCP-Überlastungsüberwachung in Abschnitt 6.5.10.

URG wird auf 1 gesetzt, wenn ein Zeiger auf Vorrangdaten (*Urgent Pointer*) verwendet wird. Der Zeiger auf Vorrangdaten bezeichnet einen Byte-Offset ab der aktuellen Sequenznummer, an dem Vorrangdaten stehen. Diese Funktion ersetzt Interruptnachrichten. Wie bereits erwähnt ist dies die einfache Art, einem Sender die Möglichkeit zu geben, ein Signal an einen Empfänger zu übertragen, ohne TCP in den Grund der Unterbrechung miteinzubeziehen. Sie wird jedoch selten verwendet.

Das *ACK*-Bit (*Acknowledgement*) wird auf 1 gesetzt, um anzugeben, dass die *Acknowledgement Number* gültig ist. Dies ist für fast alle Pakete der Fall. Ist *ACK* auf 0 gesetzt, enthält das Segment keine Bestätigung. In diesem Fall wird das Feld *Acknowledgement Number* ignoriert.

Das *PSH*-Bit bezeichnet direkt zugestellte Daten (*PUSHed*). Der Empfänger wird damit aufgefordert, die Daten der Anwendung bei Ankunft sofort weiterzuleiten und sie

nicht so lange zwischenzuspeichern, bis der Puffer voll ist (was sonst aus Gründen der Effizienz geschehen könnte).

Mit dem *RST*-Bit (*Reset*) wird eine Verbindung zurückgesetzt, wenn ein Host abstürzt oder eine ähnliche Störung aufgetreten ist. Es wird auch verwendet, um ein ungültiges Segment oder einen Versuch, eine Verbindung zu öffnen, abzuweisen. Im Allgemeinen liegt ein Problem vor, wenn ein Segment mit dem gesetzten *RST*-Bit empfangen wird.

Das *SYN*-Bit wird verwendet, um Verbindungen aufzubauen. Die Verbindungsanfrage verwendet *SYN*=1 und *ACK*=0, um anzugeben, dass das Feld für Huckepackbestätigungen nicht verwendet wird. Die Verbindungsantwort enthält indes eine Bestätigung, sodass *SYN*=1 und *ACK*=1 ist. Das *SYN*-Bit dient zur Angabe von CONNECTION REQUEST und CONNECTION ACCEPTED, wobei mit dem *ACK*-Bit zwischen den beiden Möglichkeiten unterschieden wird.

Das *FIN*-Bit dient zur Freigabe einer Verbindung. Es gibt an, dass der Sender keine weiteren Daten mehr zu übertragen hat. Nach dem Schließen einer Verbindung kann der abschließende Prozess endlos weiter Daten empfangen. Die *SYN*- und *FIN*-Segmente haben Sequenznummern und werden dadurch garantiert in der richtigen Reihenfolge verarbeitet.

Die Flusskontrolle erfolgt in TCP anhand eines Schiebefensters mit variabler Größe. Das Feld *Window Size* gibt an, wie viele Byte ab dem bestätigten Byte gesendet werden können. Ein *Window Size*-Feld von 0 ist zulässig und besagt, dass die Bytes bis einschließlich *Acknowledgement Number*–1 empfangen wurden, dass der Empfänger aber momentan unbedingt eine Verschnaufpause braucht und vorläufig keine weiteren Daten mehr will. Der Empfänger kann dann später dem weiteren Datenempfang zustimmen, indem er ein Segment mit der gleichen *Acknowledgement Number* und einem *Window Size*-Feld, das nicht null ist, überträgt.

In den Protokollen in Kapitel 3 waren die Bestätigungen für die empfangenen Rahmen und die Erlaubnis zum Senden neuer Rahmen miteinander verknüpft. Dies war auf die feste Fenstergröße dieser Protokolle zurückzuführen. Bei TCP sind Bestätigungen und die Senderlaubnis völlig voneinander getrennt. Ein Empfänger kann also sagen: „Ich habe alle Bytes bis k empfangen, möchte jetzt aber keine weiteren“. Die Trennung der beiden Aktionen (in Form eines Fensters mit variabler Größe) gibt zusätzliche Flexibilität. Dieses Thema wird weiter unten ausführlich behandelt.

Das Feld *Checksum* dient der Erhöhung der Zuverlässigkeit. Es enthält eine Prüfsumme für den Header, die Daten und den Pseudoheader, so wie es auch bei UDP der Fall ist, außer dass der Pseudoheader hier die Protokollnummer für TCP (6) aufweist und die Prüfsumme obligatorisch ist. Näheres hierzu können Sie in Abschnitt 6.4.1 nachlesen.

In dem *Options*-Feld können zusätzliche Funktionen aufgenommen werden, die im normalen Header nicht verfügbar sind. Es wurden bereits viele Optionen definiert und einige davon werden sogar häufig verwendet. Die Optionen weisen unterschiedliche Längen auf und füllen ein Mehrfaches von 32 Bit, wobei im Bedarfsfall mit Nullen auf-

gefüllt wird. Sie können bis auf 40 Byte ausgedehnt werden, um den längsten anzugebenden TCP-Header aufzunehmen. Einige Optionen werden beim Verbindungsaufbau übertragen, um die andere Seite über die Möglichkeiten zu informieren oder darüber zu verhandeln. Andere Optionen werden den Paketen während der bestehenden Verbindung mitgegeben. Jede Option hat eine Typ-Länge-Wert-Codierung.

Weitverbreitet ist die Option, die jedem Host erlaubt, die von ihm akzeptierte **maximale Segmentgröße (MSS, Maximum Segment Size)** anzugeben. Große Segmente sind effizienter als kleinere, da der 20-Byte-Header sich über mehr Daten amortisiert. Allerdings sind kleinere Hosts nicht immer in der Lage, große Segmente entgegenzunehmen. Beim Verbindungsaufbau kann jede Seite ihr Maximum angeben und das des Partners einsehen. Wenn ein Host diese Option nicht nutzt, werden 536 Byte als die Standardgröße für Nutzdaten verwendet. Alle Internethosts müssen TCP-Segmente von $536+20=556$ Byte annehmen. Die maximale Segmentgröße muss nicht in beiden Richtungen die gleiche sein.

Bei Leitungen mit hoher Bandbreite, hoher Verzögerung oder beidem ist ein 64-KB-Fenster, das einem 16-Bit-Feld entspricht, oft problematisch. Bei einer OC-12-Leitung (von ungefähr 600 Mbit/s) dauert es weniger als 1 ms, um ein ganzes 64-KB-Fenster auszugeben. Beträgt die Round-Trip-Übertragungszeit 50 ms (typisch für ein transkontinentales Kabel), wartet der Sender 98 % der Zeit auf Bestätigungen. Ein größeres Fenster würde es dem Sender ermöglichen, weiter Daten herauszupumpen. Die Option **Fensterskalierung (window scale)** erlaubt Sender und Empfänger, beim Aufbau der Verbindung einen Skalierfaktor für das Fenster zu vereinbaren. Beide Seiten verwenden den Skalierfaktor, um das Feld *Window Size* um 14 Bit nach links zu erweitern, sodass Fenster mit bis zu 2^{30} Byte möglich sind. Die meisten TCP-Implementierungen unterstützen heute diese Option.

Mit der Option **Zeitstempel (timestamp)** kann der Sender einen Zeitstempel schicken und der Empfänger diesen Wert als Echo zurücksenden. Wurde beim Verbindungsaufbau die Verwendung dieser Option festgelegt, ist sie Teil eines jeden Pakets und berechnet normalerweise die Paketumlaufzeiten, mit deren Hilfe abgeschätzt werden kann, ob ein Paket verloren gegangen ist. Sie wird auch als logische Erweiterung der 32-Bit-Sequenznummer verwendet. Auf einer schnellen Verbindung wiederholt sich unter Umständen auch die Sequenznummer schnell, was Verwirrung hinsichtlich der alten und neuen Daten stiften kann. Mit **PAWS (Protection Against Wrapped Sequence Numbers, Schutz gegen wiederholte Sequenznummern)** wird jedes mit einem alten Zeitstempel ankommende Segment verworfen, um dieses Problem zu beheben.

Und zum Schluss soll die Option **SACK (Selective ACKnowledgement, selektive Bestätigung)** erwähnt werden, die es einem Empfänger erlaubt, dem Sender den bereits empfangenen Sequenznummernbereich mitzuteilen. Diese Option ergänzt die *Acknowledgement Number* und wird verwendet, wenn ein Paket verloren gegangen ist, aber bereits Folge- oder Ersatzdaten angekommen sind. Die neuen Daten werden nicht durch das Feld *Acknowledgement Number* im Header ausgedrückt, da dieses Feld nur das nächste zu erwartende Byte angibt. Mit SACK weiß der Sender genau, welche Daten der Empfänger bereits hat, und kann daraufhin entscheiden, welche Daten neu

übertragen werden müssen. SACK wird in RFC 2108 und RFC 2883 definiert und wird immer häufiger verwendet. Wir kommen auf SACK noch in Abschnitt 6.5.10 in Zusammenhang mit der Überlastungsüberwachung zu sprechen.

6.5.5 Verbindungsaufbau in TCP

In TCP werden Verbindungen über einen Dreiwege-Handshake aufgebaut, der in Abschnitt 6.2.2 erörtert wurde. Um eine Verbindung aufzubauen, muss eine Seite, z.B. der Server, passiv auf eine ankommende Verbindung warten, wobei er die Primitiven LISTEN und ACCEPT in genau dieser Reihenfolge ausführt und entweder eine bestimmte Quelle oder auch nichts angibt.

Die andere Seite, z.B. der Client, führt eine CONNECT-Primitive aus und gibt die IP-Adresse und den Port an, zu denen er eine Verbindung wünscht, sowie die maximale Größe des TCP-Segments, die er akzeptieren kann, und optional einige Benutzerdaten (z.B. ein Kennwort). Die CONNECT-Primitive sendet ein TCP-Segment mit gesetztem SYN-Bit und ausgeschaltetem ACK-Bit und wartet auf eine Antwort.

Kommt dieses Segment am Ziel an, prüft die TCP-Instanz, ob es einen Prozess gibt, der auf dem im Feld *Destination Port* angegebenen Port ein LISTEN ausgeführt hat. Ist dies nicht der Fall, sendet die Instanz eine Antwort mit gesetztem RST-Bit, um die Verbindung abzuweisen.

Hört ein Prozess den Port auf eingehende Verbindungen ab, erhält er das ankommende TCP-Segment. Er kann die Verbindung annehmen oder abweisen. Nimmt er an, wird ein Bestätigungssegment zurückgeschickt. ► Abbildung 6.37a zeigt die Folge von TCP-Segmenten, die im Normalfall gesendet wird. Ein SYN-Segment verbraucht 1 Byte des Folgeraums, deshalb kann es unmissverständlich bestätigt werden.

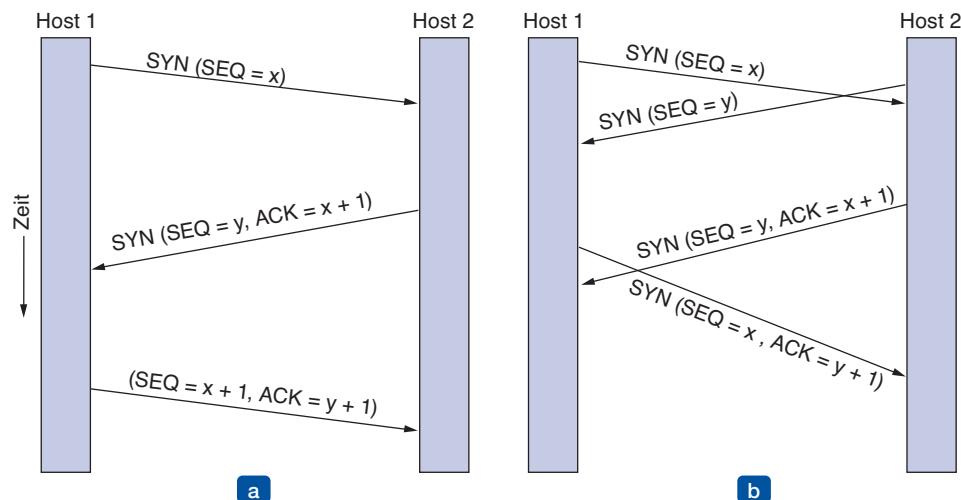


Abbildung 6.37: (a) Normaler Aufbau einer TCP-Verbindung. (b) Gleichzeitiger Verbindungsauftbau von beiden Seiten.

Versuchen zwei Hosts gleichzeitig, eine Verbindung zwischen denselben beiden Sockets aufzubauen, tritt die in ▶ Abbildung 6.37b dargestellte Situation ein. Das Ergebnis ist, dass anstatt zwei Verbindungen nur eine Verbindung aufgebaut wird, weil die Verbindungen durch ihre Endpunkte bestimmt sind. Führt die erste Verbindungseinrichtung zu einer mit (x, y) dargestellten Verbindung und die zweite ebenfalls, erfolgt nur ein Tabelleneintrag für (x, y) .

Denken Sie daran, dass die Anfangssequenznummer, die von jedem Host gewählt wird, langsam verändert werden sollte, und vor allem keine Konstante wie 0 sein sollte. Diese Regel soll vor verzögerten Paketduplicaten schützen, wie wir es bereits in Abschnitt 6.2.2 besprochen haben. Ursprünglich wurde dies mit einer taktgesteuerten Methode erreicht, die eine Taktrate von $4 \mu\text{s}$ verwendet.

Eine Schwäche bei der Implementierung des Dreiwege-Handshakes ist, dass der lauschende Prozess sich seine Sequenznummer merken muss, da er schon bald mit seinem eigenen SYN-Segment antwortet. Das bedeutet, dass ein bösartiger Sender die Ressourcen auf einem Host binden kann, indem er einen Strom von SYN-Segmenten sendet, ohne letztendlich die Verbindung herzustellen. Dieser Angriff wird **SYN-Flood** (SYN-Flut) genannt und hat seit den 1990ern viele Webserver lahmgelegt.

Eine Möglichkeit, sich vor einem solchen Angriff zu schützen, ist die Verwendung von **SYN-Cookies**. Anstatt sich die Sequenznummer zu merken, wählt ein Host eine kryptografisch erzeugte Sequenznummer, verbindet sie mit dem ausgehenden Segment und vergisst sie. Bei Abschluss des Dreiwege-Handshakes wird diese Sequenznummer (plus 1) an den Host zurückgeliefert. Er kann dann die korrekte Sequenznummer neu erzeugen, indem er die gleiche kryptografische Funktion ausführt. Voraussetzung ist, dass die Eingaben der Funktion bekannt sind, zum Beispiel IP-Adresse und Port des anderen Hosts, sowie ein lokales Geheimnis. Dieses Verfahren erlaubt es dem Host, zu prüfen, ob eine bestätigte Sequenznummer korrekt ist, ohne sich die Sequenznummer separat merken zu müssen. Kritisiert wird hieran oft, dass es nicht möglich ist, TCP-Optionen zu verarbeiten, sodass SYN-Cookies nur verwendet werden sollten, wenn der Host einer SYN-Flut ausgesetzt ist. Sie sind jedoch eine interessante Alternative beim Verbindungsauflaufbau. Weitere Informationen hierzu finden Sie in RFC 4987 und Lemon (2002).

6.5.6 Verbindungsfreigabe in TCP

TCP-Verbindungen sind Vollduplexverbindungen. Um besser zu verstehen, wie Verbindungen freigegeben werden, stellt man sie sich am besten als ein Paar von Simplexverbindungen vor. Jede Simplexverbindung wird unabhängig von der anderen abgebaut. Für den Verbindungsabbau kann eine Partei ein TCP-Segment mit gesetztem *FIN*-Bit senden. Dies bedeutet, dass die Partei keine weiteren Daten mehr zu übertragen hat. Wird *FIN* bestätigt, wird die Verbindung in dieser Richtung für neue Daten geschlossen. In die andere Richtung können Daten jedoch weiterhin beliebig lang fließen. Erst wenn die Verbindungen für beide Richtungen beendet wurden, wird die Verbindung freigegeben. Normalerweise sind vier TCP-Segmente erforderlich, um eine Verbindung abzubauen, d.h. ein *FIN* und ein *ACK* für jede Richtung. Es ist allerdings

möglich, dass sich das erste *ACK* und das zweite *FIN* im gleichen Segment befinden, was die Gesamtzahl auf drei reduzieren würde.

Wie bei einem Telefongespräch, bei dem sich die beiden Teilnehmer verabschieden und mehr oder weniger gleichzeitig auflegen, können die beiden Enden einer TCP-Verbindung gleichzeitig *FIN*-Segmente senden. Diese werden auf die übliche Weise bestätigt, und die Verbindung wird freigegeben. Ob zwei Hosts nacheinander oder gleichzeitig eine Verbindung freigeben, macht im Grunde keinen Unterschied.

Um das an früherer Stelle geschilderte Zwei-Armeen-Problem zu verhindern (siehe Abschnitt 6.2.3), werden Timer eingesetzt. Folgt auf ein *FIN* innerhalb von zwei maximalen Paketlebenszeiten keine Antwort, baut der *FIN*-Sender die Verbindung ab. Die andere Seite stellt irgendwann fest, dass ihr offensichtlich niemand mehr zuhört, und beendet ihrerseits auch. Diese Lösung ist zwar nicht perfekt, aber akzeptabel, da eine bessere Lösung theoretisch nicht möglich ist. In der Praxis kommt es selten zu Problemen.

6.5.7 Modellierung der Verwaltung von TCP-Verbindungen

Die zum Auf- und Abbau von Verbindungen erforderlichen Schritte können anhand eines endlichen Automaten und den elf in ► Abbildung 6.38 aufgeführten Zuständen dargestellt werden. In jedem Zustand sind bestimmte Ereignisse zulässig. Tritt ein zulässiges Ereignis ein, kann eine bestimmte Aktion erfolgen. Tritt ein nicht zulässiges Ereignis ein, wird ein Fehler gemeldet.

| Zustand | Beschreibung |
|-------------|---|
| CLOSED | Keine Verbindung aktiv oder anstehend. |
| LISTEN | Der Server wartet auf eine ankommende Verbindung. |
| SYN RCV | Eine Verbindungsanfrage ist angekommen. Warten auf Bestätigung. |
| SYN SENT | Die Anwendung hat begonnen, eine Verbindung zu öffnen. |
| ESTABLISHED | Zustand der normalen Datenübertragung. |
| FIN WAIT 1 | Die Anwendung möchte die Übertragung beenden. |
| FIN WAIT 2 | Die andere Seite ist einverstanden, die Verbindung abzubauen. |
| TIME WAIT | Warten, bis keine Pakete mehr kommen. |
| CLOSING | Beide Seiten haben gleichzeitig versucht, zu beenden. |
| CLOSE WAIT | Die Gegenseite hat die Freigabe eingeleitet. |
| LAST ACK | Warten, bis keine Pakete mehr kommen. |

Abbildung 6.38: Zustände des endlichen Automaten für die Verwaltung von TCP-Verbindungen.

Jede Verbindung beginnt im *CLOSED*-Zustand. Sie verlässt diesen Zustand, wenn sie entweder passiv (LISTEN) oder aktiv (CONNECT) geöffnet wird. Wenn die andere Seite das entsprechende Gegenstück ausführt, wird eine Verbindung aufgebaut und es erfolgt der Übergang in den Zustand *ESTABLISHED*. Die Verbindungs freigabe kann von beiden Seiten eingeleitet werden. Nach deren Beendigung wird wieder der Zustand *CLOSED* eingenommen.

Dieser endliche Automat ist in ► Abbildung 6.39 dargestellt. Der normale Fall, bei dem sich ein Client aktiv bei einem passiven Server anmeldet, wird durch fette Linien (Client) und gestrichelte Linien (Server) dargestellt. Die feinen Linien bezeichnen ungewöhnliche Ereignisfolgen. Jede Linie in Abbildung 6.39 ist mit einem Ereignis/ Aktion-Paar gekennzeichnet. Das Ereignis kann ein vom Benutzer eingeleiteter Systemaufruf (CONNECT, LISTEN, SEND oder CLOSE), eine Segmentankunft (SYN, FIN, ACK oder RST) oder – in einem Fall – ein Timeout nach zweimal der maximalen Paketlebenszeit sein. Als Aktion wird entweder ein Steuersegment gesendet (SYN, FIN oder RST) oder nichts gemacht (was in der Abbildung durch einen Strich – gekennzeichnet ist). Kommentare stehen in Klammern.

Das Diagramm ist am verständlichsten, wenn man zuerst dem Pfad eines Clients (fette durchgezogene Linie) und dann dem Pfad eines Servers (fette gestrichelte Linie) folgt. Schickt eine Client-Anwendung eine CONNECT-Anfrage ab, erzeugt die lokale TCP-Instanz einen Verbindungsdatensatz, kennzeichnet ihn als im Zustand *SYN SENT* befindlich und sendet ein *SYN*-Segment. Beachten Sie, dass im Falle von mehreren geöffneten Anwendungen viele Verbindungen gleichzeitig offen sein (oder geöffnet werden) können, sodass der Zustand für genau eine Verbindung gilt und im Verbindungsdatensatz aufgezeichnet wird. Kommt *SYN+ACK* an, sendet TCP das letzte *ACK* des Dreiwege-Handshakes und wechselt in den *ESTABLISHED*-Zustand. Nun können Daten gesendet und übertragen werden.

Ist eine Anwendung fertig, führt sie eine CLOSE-Primitive aus. Hierdurch sendet die lokale TCP-Instanz ein *FIN*-Segment und wartet auf das entsprechende *ACK* (gestrichelter Rahmen mit der Beschriftung „Aktives Schließen“). Kommt *ACK* an, erfolgt ein Übergang in den Zustand *FIN WAIT 2*, und eine Richtung der Verbindung wird geschlossen. Wenn auch die andere Seite schließt, sendet sie ein *FIN*, das bestätigt wird. Nun sind beide Seiten geschlossen; TCP wartet jedoch so lange, bis zweimal die maximale Paketlebenszeit verstrichen ist, um sicherzugehen, dass es keine weiteren Pakete mehr gibt, für den Fall, dass die Bestätigung verloren gegangen ist. Läuft der Timer ab, löscht TCP den Datensatz der betreffenden Verbindung.

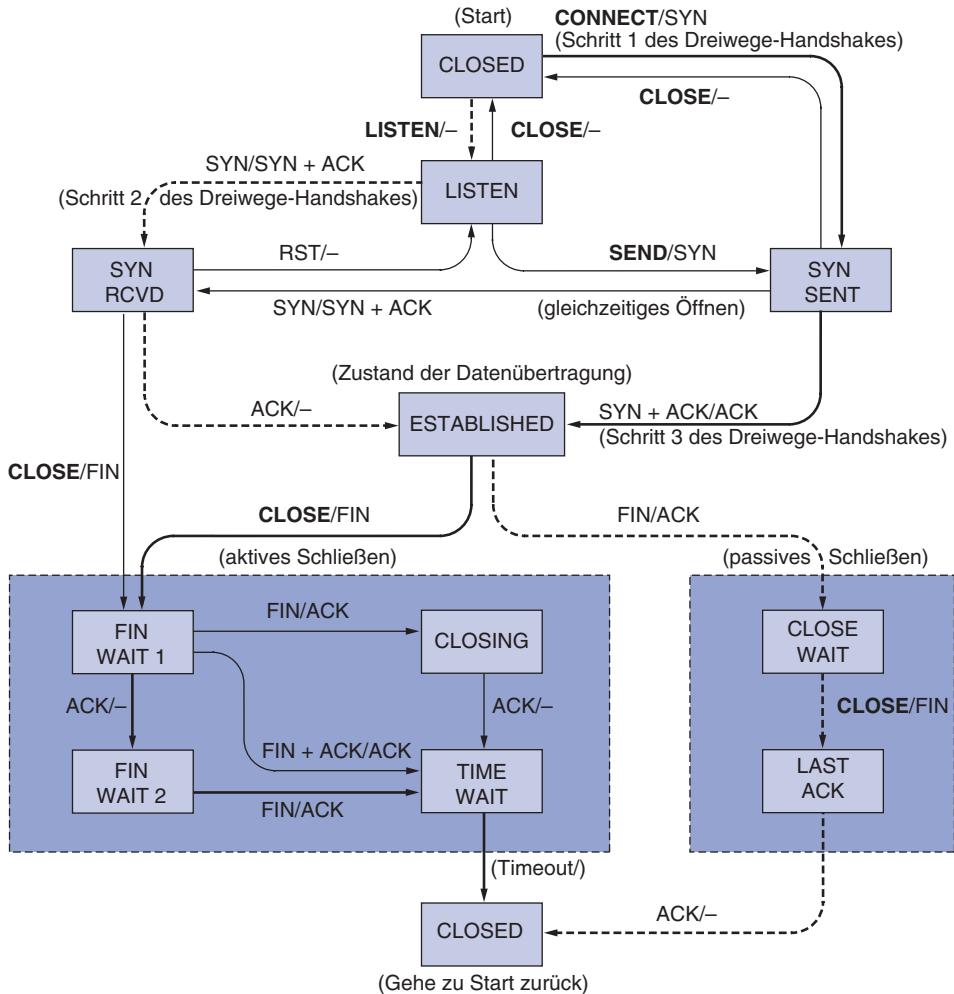


Abbildung 6.39: TCP-Verbindungsmanagement als Zustandsautomat (fette Linie = normaler Pfad des Clients; fette gestrichelte Linie = normaler Pfad des Servers; feine Linien = ungewöhnliche Ereignisse). Jeder Übergang wird durch das Ereignis bezeichnet, das ihn verursacht, sowie durch die daraus resultierenden Aktionen (getrennt durch einen Schrägstrich).

Betrachten wir nun das Verbindungsmanagement aus der Sicht des Servers. Der Server führt ein LISTEN aus und wartet, was passiert. Kommt SYN an, wird es bestätigt, und der Server wechselt in den Zustand **SYN RCV**. Nachdem das SYN des Servers bestätigt wurde, ist das Dreiecks-Handshake abgeschlossen, und der Server tritt in den **ESTABLISHED**-Zustand. Die Datenübertragung kann nun stattfinden.

Ist der Client mit der Übertragung fertig, führt er CLOSE aus, woraufhin ein FIN beim Server ankommt (gestrichelter Kasten mit der Beschriftung „Passives Schließen“). Dies ist das Signal für den Server. Der Server führt seinerseits CLOSE aus, und FIN wird zum Client gesendet. Wenn die Bestätigung des Clients eintrifft, gibt der Server die Verbindung frei und löscht den Verbindungsdatensatz.

6.5.8 TCP-Schiebefenster

Wie bereits erwähnt, wird beim Fenstermanagement in TCP die Bestätigung des korrekten Segmentempfangs von der Zuordnung des Empfängerbuffers getrennt. Ein Beispiel: Angenommen ein Empfänger hat einen Puffer von 4 096 Byte, wie in ►Abbildung 6.40 dargestellt. Wenn der Sender ein 2 048-Byte-Segment überträgt, das korrekt empfangen wird, bestätigt der Empfänger dieses Segment. Da in seinem Puffer jetzt nur noch 2 048 Byte frei sind (bis die Anwendung einige Daten aus dem Puffer entfernt), kündigt er, beginnend mit dem nächsten erwarteten Byte, ein 2 048 Byte großes Fenster an.

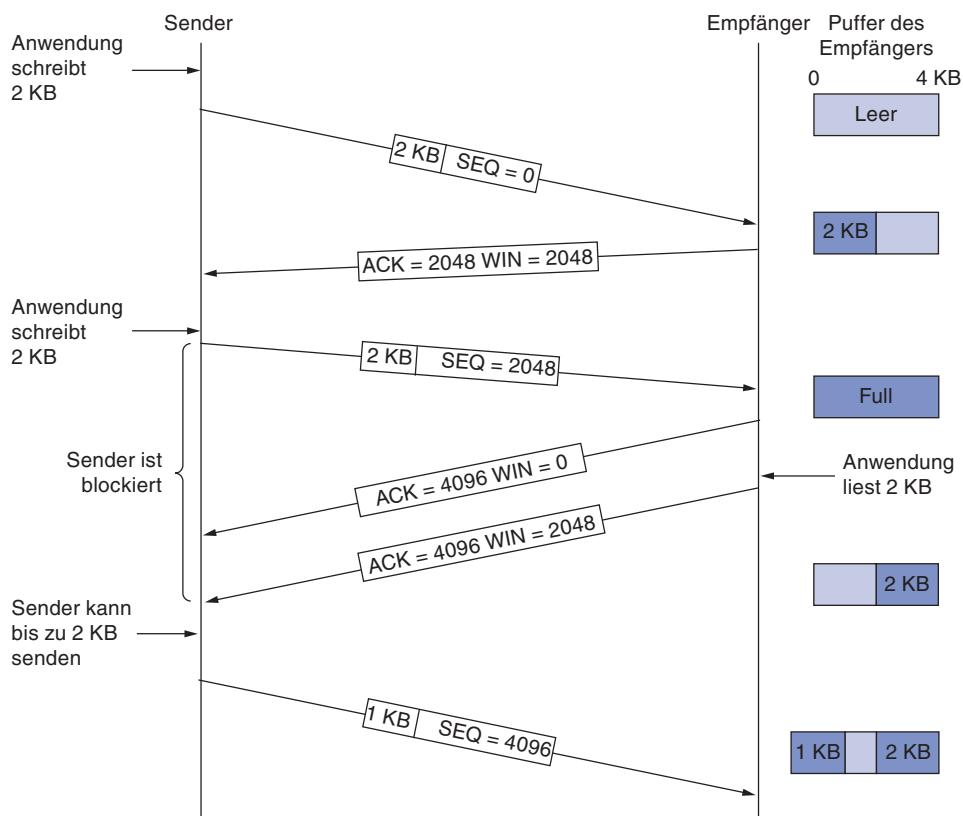


Abbildung 6.40: Fensterverwaltung in TCP.

Anschließend überträgt der Sender weitere 2 048 Byte, die bestätigt werden. Das angekündigte Fenster ist nun 0. Der Sender muss anhalten, bis der Anwendungsprozess auf dem empfangenden Host einige Daten aus dem Puffer entfernt hat, bevor TCP ein größeres Fenster ankündigen kann und weitere Daten gesendet werden können.

Ist das Fenster 0, kann der Sender normalerweise keine Segmente senden. Es gibt jedoch zwei Ausnahmen: Erstens können Vorrangdaten immer gesendet werden, z.B. um es dem Benutzer zu ermöglichen, den auf dem entfernten Rechner laufenden Pro-

zess abzubrechen. Zweitens kann der Sender ein 1-Byte-Segment senden, um den Empfänger zu zwingen, das nächste erwartete Byte und die Fenstergröße erneut anzugeben. Dieses Paket wird **Fenstersondieranfrage** (*window probe*) genannt. Der TCP-Standard bietet diese Option ausdrücklich, um Deadlocks zu vermeiden, falls eine Fensteraktualisierung einmal verloren gehen sollte.

Sender müssen die Daten, die von der Anwendung eintreffen, nicht unbedingt sofort übertragen. Ebenso wenig müssen Empfänger schnellstmöglich Bestätigungen senden. Wenn beispielsweise in Abbildung 6.40 die ersten 2 KB Daten ankommen und TCP weiß, dass ein 4-KB-Fenster verfügbar ist, wäre es absolut korrekt, die Daten zwischenzuspeichern, bis weitere 2 KB ankommen, damit ein Segment mit 4 KB Nutzdaten übertragen werden kann. Diese Entscheidungsfreiheit kann zur Leistungsverbesserung genutzt werden.

Betrachten Sie eine Verbindung zu einem entfernten Terminal (beispielsweise unter Verwendung von SSH oder Telnet), die auf jeden Tastenanschlag reagiert. Im schlimmsten Fall erzeugt TCP für jedes Zeichen, das bei der sendenden TCP-Instanz ankommt, ein aus 21 Byte bestehendes TCP-Segment, das an IP zur Weiterleitung als 41-Byte-IP-Datagramm übergeben wird. Am empfangenden Ende sendet TCP unmittelbar darauf eine Bestätigung von 40 Byte (20 Byte für den TCP-Header und 20 Byte für den IP-Header). Später, wenn das entfernte Terminal das Byte gelesen hat, sendet TCP eine Fensteraktualisierung, wodurch das Fenster um 1 Byte nach rechts verschoben wird. Dieses Paket ist auch 40 Byte groß. Hat das entfernte Terminal schließlich das Zeichen verarbeitet, gibt es dieses zur lokalen Anzeige als 41-Byte-Paket weiter. Pro jedes eingegebene Zeichen werden dabei eine Bandbreite von 162 Byte belegt und vier Segmente gesendet. Wenn Bandbreite knapp ist, ist diese Methode nicht unbedingt zu empfehlen.

Ein Ansatz, mit dem viele TCP-Implementierungen diese Situation optimieren, sind die sogenannten **verzögerten Bestätigungen** (*delayed acknowledgement*). Bestätigungen und Fensteraktualisierungen werden bis zu 500 ms verzögert in der Hoffnung, dass einige Daten eintreffen, die eine Mitreisemöglichkeit bieten. Angenommen, das Terminal antwortet innerhalb von 500 ms, dann muss von der entfernten Seite nur ein 41-Byte-Paket zurückgesendet werden, was die Anzahl der Pakete und die genutzte Bandbreite halbiert.

Doch auch wenn diese verzögerten Bestätigungen die empfängerseitig erzeugte Last im Netz reduzieren, arbeitet ein Sender, der mehrere kurze Pakete schickt (z.B. 41-Byte-Pakete mit je einem Datenbyte), nach wie vor ineffizient. Der **Nagle-Algorithmus** (1984) bietet hier Abhilfe. Nagles Vorschlag ist einfach: Wenn Daten beim Sender byteweise ankommen, wird nur das erste Byte gesendet, und der Rest wird so lange im Puffer zwischengespeichert, bis das erste Byte bestätigt wird. Dann werden alle zwischengespeicherten Zeichen in einem TCP-Segment gesendet, und die Zwischenspeicherung beginnt erneut, bis das nächste Segment bestätigt wird. Das bedeutet, zu jedem Zeitpunkt kann immer nur ein kleines Paket ausstehen. Wenn viele Daten von der Anwendung in einer Paketumlaufzeit gesendet werden, werden diese vom Nagle-Algorithmus in einem Segment untergebracht, was die verwendete Bandbreite beträchtlich reduziert. Der Algorithmus besagt darüber hinaus, dass ein neues Seg-

ment geschickt werden soll, wenn genug Daten eingegangen sind, um ein Segment von maximaler Größe zu bilden.

Nagles Algorithmus wird heute sehr häufig von TCP-Implementierungen verwendet, sollte aber unter bestimmten Bedingungen besser deaktiviert werden. Besonders bei interaktiven Spielen über das Internet wollen Spieler normalerweise lieber einen schnellen Strom von kleinen Aktualisierungspaketen. Das Sammeln der Aktualisierungen, um sie dann in Schüben zu schicken, lässt das Spiel erratisch reagieren, was viele Benutzer verärgert. Ein selteneres Problem ist, dass der Nagle-Algorithmus manchmal mit verzögerten Bestätigungen interagieren kann und so ein temporäres Deadlock auslöst: Der Empfänger wartet auf Daten, die eine Bestätigung Huckepack mitnehmen können, und der Sender wartet auf die Bestätigung, weitere Daten zu senden. Diese Interaktion kann das Herunterladen von Webseiten verzögern. Zur Vermeidung dieser Probleme kann der Nagle-Algorithmus abgeschaltet werden (wird `TCP_NODELAY`-Option genannt). Mogul und Minshall (2001) diskutieren diese und andere Lösungen.

Ein weiteres Problem, das sich negativ auf die TCP-Leistung auswirkt, ist das sogenannte **Silly Window Syndrome** (Dummes-Fenster-Syndrom; Clark, 1982). Dieses Problem tritt auf, wenn Daten in großen Blöcken an die sendende TCP-Instanz übergeben werden, eine interaktive Anwendung am empfangenden Ende die Daten aber nur Byte für Byte einliest. Das Problem ist in ►Abbildung 6.41 dargestellt. Anfangs ist der TCP-Puffer auf der empfangenden Seite voll (d.h., er hat eine Fenstergröße von 0) und der Sender weiß dies. Dann liest die interaktive Anwendung ein Zeichen aus dem TCP-Strom ein. Diese Aktion wird vom empfangenden TCP mit Freude aufgenommen, sodass eine Fensteraktualisierung mit dem Hinweis an den Sender geht, dass ein weiteres Byte gesendet werden darf. Der Sender kommt diesem Wunsch geflissenstlich nach und sendet ein Byte. Der Puffer ist jetzt voll, sodass der Empfänger das 1-Byte-Segment bestätigt, das Fenster aber auf 0 setzt. Dies kann endlos fortgesetzt werden.

Clarks Lösung bestand darin zu verhindern, dass der Empfänger eine Fensteraktualisierung für nur 1 Byte sendet. Stattdessen wird er gezwungen zu warten, bis ausreichend Speicherplatz frei ist, und dies dann anzusehen. Der Empfänger sollte erst eine Fensteraktualisierung senden, wenn er die maximale Segmentgröße, die er beim Aufbau der Verbindung angekündigt hat, verarbeiten kann oder sein Puffer halb leer ist, je nachdem, was kleiner ist. Aber auch der Sender kann hilfreich mitwirken, indem er beispielsweise keine winzigen Segmente sendet. Er sollte vielmehr warten, bis genügend Platz im Fenster vorhanden ist, und dann ein volles Segment senden oder zumindest eines, das die Hälfte der Größe des Empfängerpuffers belegt.

Nagles Algorithmus und Clarks Lösung zu dem Silly Window Syndrome ergänzen sich. Nagle versuchte, das Problem zu lösen, das entsteht, wenn eine sendende Anwendung die Daten byteweise an TCP übergibt. Clark versuchte, das Problem zu lösen, das entsteht, wenn die empfangende Anwendung die Daten byteweise von TCP entgegennimmt. Beide Lösungen sind sinnvoll und können zusammenarbeiten. Das gemeinsame Ziel ist es, den Sender am Senden kleiner Segmente und den Empfänger an deren Anforderung zu hindern.

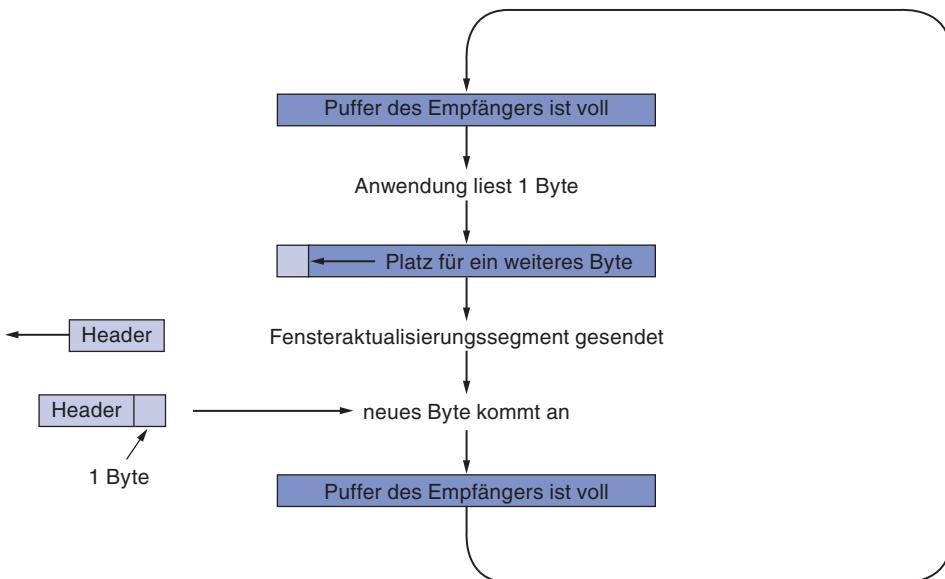


Abbildung 6.41: Silly Window Syndrome.

TCP kann empfängerseitig die Leistung noch weiter verbessern, als sich auf Fensteraktualisierungen in großen Einheiten zu beschränken. Wie beim TCP auf der Senderseite können auch hier Daten zwischengespeichert werden, d.h., TCP kann eine READ-Anforderung der Anwendung so lange blockieren, bis eine größere Datenmenge bereitsteht. Das reduziert die Anzahl der TCP-Aufrufe (und damit den Overhead). Es erhöht allerdings auch die Antwortzeit, wobei Effizienz für nicht interaktive Anwendungen wie Dateiübertragung oft wichtiger ist als die Reaktionszeit auf einzelne Anforderungen.

Der Empfänger muss des Weiteren das Problem meistern, dass die Segmente nicht in der richtigen Reihenfolge ankommen. Dazu puffert er die Daten, bis sie in der korrekten Reihenfolge an die Anwendung übertragen werden können. Er könnte zu früh eintreffende Segmente, die die Reihenfolge durcheinanderbringen, auch verwerfen. Passieren würde wohl nichts, da diese Segmente am Ende vom Sender neu übertragen werden. Aber es wäre eine ziemliche Verschwendungen.

Bestätigungen können selbstverständlich nur gesendet werden, wenn alle Daten bis zum bestätigten Byte eingegangen sind. Dies wird **kumulative Bestätigung** genannt. Wenn der Empfänger die Segmente 0, 1, 2, 4, 5, 6 und 7 erhält, kann er alles bis einschließlich des letzten Bytes in Segment 2 bestätigen. Erfährt der Sender einen Zeitüberlauf, überträgt er Segment 3 neu. Da der Empfänger die Segmente 4 bis 7 im Puffer abgelegt hat, können bei Eingang von Segment 3 alle Bytes bis zum Ende von Segment 7 bestätigt werden.

6.5.9 Verwaltung von Timern in TCP

TCP benutzt mehrere Timer (zumindest konzeptionell). Der wichtigste ist der **Retransmission-Timer** für erneute Übertragungen (*Retransmission TimeOut, RTO*). Sobald ein Segment gesendet wird, startet ein Retransmission-Timer. Wird das Segment bestätigt, bevor der Timer abläuft, wird der Timer gestoppt. Läuft andererseits der Timer vor Ankunft der Bestätigung ab, wird das Segment erneut übertragen (und der Timer startet wieder von vorn). Die Frage ist nun: Wie lang soll die Zeitspanne bis zum Timeout sein?

Dieses Problem ist auf der Transportschicht viel schwieriger als in den Sicherungsschichtprotokollen wie IEEE 802.11. Im letzteren Fall wird die erwartete Übertragungsverzögerung in Mikrosekunden gemessen und ist gut vorhersagbar (d.h., sie unterliegt nur geringen Schwankungen). Daher kann der Timer so gesetzt werden, dass er abläuft, kurz nachdem eine Bestätigung erwartet wird (► Abbildung 6.42a). Da sich Bestätigungen auf der Sicherungsschicht selten verzögern (hier tritt keine Überlastung auf), bedeutet das Fehlen einer Bestätigung zum erwarteten Zeitpunkt, dass der Rahmen oder die Bestätigung verloren gegangen ist.

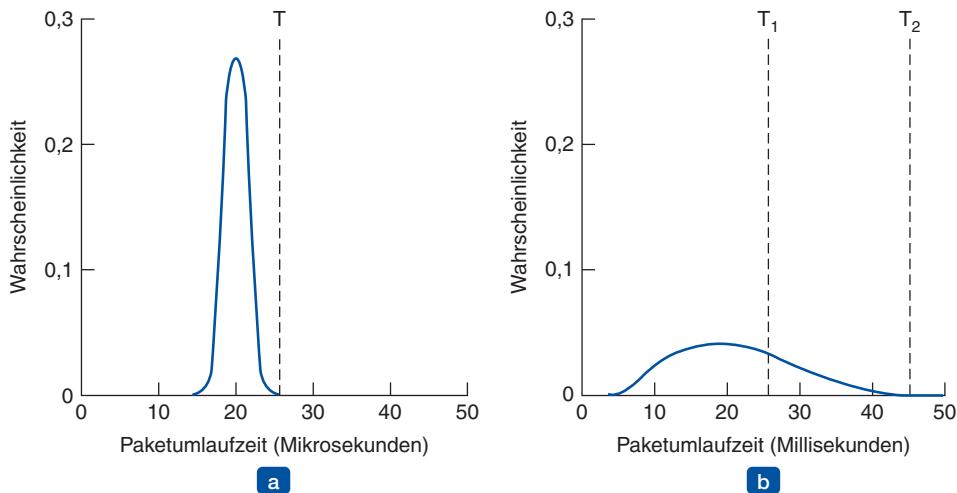


Abbildung 6.42: (a) Wahrscheinlichkeitsdichte der Ankunftszeiten von Bestätigungen auf der Sicherungsschicht. (b) Wahrscheinlichkeitsdichte der Ankunftszeiten von Bestätigungen für TCP.

TCP hingegen hat es mit einer gänzlich anderen Umgebung zu tun. Die Wahrscheinlichkeitsdichtefunktion der Zeitdauer, bis eine TCP-Bestätigung zurückkommt, ähnelt eher ► Abbildung 6.42b als Abbildung 6.42a. Sie ist größer und variabler. Das Bestimmen der Paketumlaufzeit (RTT) zum Ziel ist nicht einfach. Doch auch wenn sie bekannt ist, ist die Festlegung eines Zeitintervalls schwierig. Ist der Timeout zu kurz, z.B. T_1 in Abbildung 6.42b, erfolgen unnötige Neuübertragungen und das Internet wird mit nutzlosen Paketen überfrachtet. Ist die Zeitspanne zu lang (T_2), entstehen Leistungseinbußen durch lange Übertragungsverzögerungen für Neuübertragungen, wenn ein Paket verloren gegangen ist. Des Weiteren können sich der Mittelwert und die Verteilungsvarianz der Ankunftszeiten von Bestätigungen schnell innerhalb von ein paar Sekunden ändern, wenn eine Überlastung entsteht oder aufgelöst wird.

Eine Lösung ist ein dynamischer Algorithmus, der das Timeout-Intervall auf der Grundlage von Messungen der Netzleistung fortlaufend anpasst. Der bei TCP allgemein verwendete Algorithmus ist auf Jacobson (1988) zurückzuführen und funktioniert wie folgt: Für jede Verbindung verwaltet TCP die Variable *SRTT* (*Smoothed Round-Trip Time*), das heißt, die aktuell beste Schätzung der Paketumlaufzeit zum fraglichen Ziel bis zum Erhalt der Bestätigung. Wird ein Segment gesendet, startet ein Timer. Er überwacht einerseits, wie lange die Bestätigung braucht, und löst andererseits eine Neuübertragung aus, wenn die Bestätigung zu lange dauert. Kommt die Bestätigung zurück, bevor der Timer abläuft, misst TCP, wie lange die Bestätigung unterwegs war, beispielsweise *R*. Dann wird *SRTT* unter Verwendung der Formel

$$SRTT = \alpha SRTT + (1 - \alpha)R$$

berechnet, wobei α ein Glättungsfaktor ist, der festlegt, wie schnell die alten Werte vergessen werden. In der Regel ist $\alpha=7/8$. Diese Art Formel ist ein **exponentiell gewichteter gleitender Durchschnitt** (*Exponentially Weighted Moving Average, EWMA*) oder ein Tiefpassfilter, der Rauschen in den Samples entfernt.

Auch bei einem guten *SRTT*-Wert ist die Auswahl eines geeigneten Timeouts für Neuübertragungen keine leichte Aufgabe. Erste Implementierungen von TCP verwendeten $2 \times SRTT$, aber Erfahrungen haben gezeigt, dass ein konstanter Wert inflexibel ist, weil er nicht reagieren kann, wenn die Varianz steigt. Vor allem Warteschlangenmodelle des Poisson-verteilten Verkehrs sagen vorher, dass, wenn die Last die Kapazitätsgrenze erreicht, die Verzögerung groß wird und extrem schwankt. Dies kann dazu führen, dass der Retransmission-Timer ausgelöst und eine Kopie des Pakets erneut übertragen wird, obwohl das ursprüngliche Paket immer noch im Netz herumgeistert. Dieses Verhalten ist am häufigsten bei hoher Last zu beobachten, eigentlich die schlechteste Zeit, um zusätzliche Pakete in das Netz zu stellen.

Um dieses Problem zu beheben, schlug Jacobson vor, den Timeout-Wert abhängig von der Varianz der Paketumlaufzeiten sowie von der geglätteten Paketumlaufzeit zu machen. Diese Änderung erfordert die Überwachung einer weiteren geglätteten Variablen, *RTTVar* (*Round-Trip Time Variation*), die mit folgender Formel aktualisiert wird:

$$RTTVar = \beta RTTVar + (1 - \beta)|SRTT - R|$$

Dies ist wie oben ein exponentiell gewichteter gleitender Durchschnitt (EWMA), und β liegt normalerweise bei $3/4$. Der Timeout-Wert für die Neuübertragung (*RTO*) wird gesetzt auf

$$RTO = SRTT + 4 \times RTTVar$$

Der Faktor 4 wurde eigentlich eher willkürlich gewählt, hat aber zwei Vorteile. Erstens können Multiplikationen mit 4 durch eine einzige Bitverschiebung ausgeführt werden. Zweitens kommen weniger als 1 % aller Pakete um mehr als vier Standardabweichungen zu spät an. Beachten Sie, dass die Variable *RTTVar* nicht genau das Gleiche ist wie die Standardabweichung, denn sie ist eigentlich die mittlere Abweichung (*mean deviation*), aber für die Praxis ist sie gut genug. Der Aufsatz von Jacobson ist voll von schlauen Tricks, die zeigen, wie dieTimeouts allein durch Addieren, Subtrahieren und

Verschieben von ganzen Zahlen berechnet werden. Diese Wirtschaftlichkeit wird bei modernen Hosts nicht benötigt, ist aber Teil der Kultur geworden, die es TCP erlaubt, auf allen möglichen Geräten ausgeführt zu werden, von Supercomputern bis zu winzigen Geräten. Bis jetzt hat noch niemand TCP auf einem RFID-Chip angewendet, doch wer weiß? Das kann sich eines Tages ändern.

Weitere Einzelheiten zur Berechnung dieses Timeouts, einschließlich der Anfangseinstellungen der Variablen, werden in RFC 2988 definiert. Der Retransmission-Timer wird unabhängig von den Schätzungen auf ein Minimum von 1 Sekunde gesetzt. Dieses ist ein konservativer Wert, der gewählt wurde, um basierend auf Messungen störende Neuübertragungen zu verhindern (Allman und Paxson, 1999).

Beim Zusammenstellen der Samples R der Paketumlaufzeit stellt sich die Frage, was zu tun ist, wenn ein Segment abläuft und erneut gesendet wird. Wenn die Bestätigung ankommt, ist nicht klar, ob sie sich auf die erste oder eine spätere Übertragung bezieht. Eine Fehleinschätzung kann das Timeout der Neuübertragung ernsthaft beeinträchtigen. Phil Karn entdeckte dieses Problem auf seine Art. Er ist begeisterter Amateurfunker und daran interessiert, TCP/IP-Pakete über „Amateurfunk“ – ein notorisch unzuverlässiges Medium – zu übertragen. Er machte einen einfachen Vorschlag: Aktualisieren Sie keine Schätzungen zu irgendwelchen Segmenten, die erneut übertragen wurden. Stattdessen wird das Timeout bei jeder nachfolgenden Neuübertragung so lange verdoppelt, bis die Segmente beim ersten Mal durchkommen. Das nennt man heute den **Karn-Algorithmus** (Karn und Partridge, 1987). Er wird in den meisten TCP-Implementierungen verwendet.

In TCP wird nicht nur der Retransmission-Timer verwendet. Ein weiterer Timer ist der **Persistence-Timer**. Er ist dazu da, um folgenden Deadlock zu verhindern: Der Empfänger sendet eine Bestätigung mit einer Fenstergröße von 0 und weist damit den Sender zum Warten an. Später aktualisiert der Empfänger das Fenster, aber das Paket mit der Aktualisierung geht verloren. Nun warten beide, der Sender und der Empfänger, bis einer etwas unternimmt. Läuft der Persistence-Timer ab, überträgt der Sender eine Anfrage zum Empfänger. Die Antwort auf diese Anfrage ergibt die Fenstergröße. Ist sie immer noch null, wird der Persistence-Timer wieder gesetzt, und der Zyklus beginnt von neuem. Ist sie nicht null, können Daten übertragen werden.

Bei manchen Implementierungen wird noch ein dritter Timer, der **Keepalive-Timer**, benutzt. Wenn eine Verbindung über längere Zeit inaktiv ist, läuft der Keepalive-Timer ab, wodurch eine Seite veranlasst wird, zu prüfen, ob die andere Seite noch da ist. Kommt keine Antwort, wird die Verbindung beendet. Diese Funktion ist umstritten, weil sie mit Overhead verbunden ist und eine ansonsten funktionstüchtige Verbindung aufgrund einer vorübergehenden Netzaufspaltung beenden kann.

Der letzte Timer, der in jeder TCP-Verbindung verwendet wird, ist der, der im Zustand **TIME WAIT** während des Schließens benutzt wird. Er läuft das Doppelte der maximalen Paketlebensdauer, um sicherzustellen, dass alle Pakete einer Verbindung übertragen wurden, bevor diese beendet wird.

6.5.10 TCP-Überlastungsüberwachung

Eine der Hauptfunktionen von TCP haben wir uns für den Schluss aufgehoben: die Überlastungsüberwachung. Wenn die Last in einem Netz größer wird als das Netz bewältigen kann, kommt es zu Verstopfungen. Das Internet bildet hier keine Ausnahme. Die Vermittlungsschicht erkennt diese Verstopfungen an den immer länger werdenden Warteschlangen bei den Routern und versucht Abhilfe zu leisten – und sei es durch Verlieren von Paketen. Aufgabe der Transportschicht ist es, das Überlastungsfeedback von der Vermittlungsschicht entgegenzunehmen und daraufhin ihr Verkehrsaufkommen im Netz zu reduzieren. Im Internet spielt TCP bei der Überwachung der Überlastung sowie beim zuverlässigen Transport die Hauptrolle. Aus diesem Grunde ist es ein so besonderes Protokoll.

Die allgemeine Situation der Überlastungsüberwachung haben wir bereits in Abschnitt 6.3 besprochen. Eine unserer Haupterkenntnisse war, dass ein Transportprotokoll, das ein AIMD-Steuerungsgesetz (*Additive Increase Multiplicative Decrease*) als Antwort auf binäre Überlastungssignale vom Netz verwendet, gegen eine faire und effiziente Bandbreitenzuordnung konvergiert. TCP-Überlastungsüberwachung basiert auf der Implementierung dieses Ansatzes unter Verwendung eines Fensters und mit Paketverlust als binäres Signal. Zu diesem Zwecke verwaltet TCP ein **Überlastungsfenster** (*congestion window*), dessen Größe gleich der Anzahl von Bytes ist, die der Sender zu jeder Zeit im Netz haben kann. Die entsprechende Rate ist die Fenstergröße geteilt durch die Paketumlaufzeit der Verbindung. TCP passt die Größe des Fensters entsprechend der AIMD-Regel an.

Erinnern wir uns, dass neben dem Überlastungsfenster *zusätzlich* ein Flusskontrollfenster verwaltet wird, das die Anzahl der Bytes angibt, die der Empfänger puffern kann. Beide Fenster werden parallel verfolgt; die Anzahl der Bytes, die gesendet werden dürfen, entspricht dem kleineren der beiden Fenster. Demzufolge ist das effektive Fenster das kleinere von dem, was der Sender beziehungsweise der Empfänger für angemessen hält. Voraussetzung ist jedoch, dass es immer zwei gibt. TCP hört auf, Daten zu senden, wenn entweder das Überlastungs- oder das Flusskontrollfenster vorübergehend voll ist. Wenn der Empfänger sagt: „Sende 64 KB“, aber der Sender weiß, dass Datenschübe von mehr als 32 KB das Netz verstopfen, schickt der Sender 32 KB. Wenn andererseits der Empfänger sagt „Sende 64 KB“, und der Sender weiß, dass Datenschübe bis 128 KB problemlos übertragen werden können, sendet er die ganzen angeforderten 64 KB. Das Flusskontrollfenster wurde bereits vorher beschrieben, sodass wir uns im Folgenden nur auf das Überlastungsfenster beschränken wollen.

Die moderne Überlastungsüberwachung in TCP ist größtenteils auf die Bemühungen von Van Jacobson (1988) zurückzuführen. Die Geschichte dazu ist faszinierend. Im Zuge der seit 1986 zunehmenden Popularität des frühen Internets kam es zu einem Vorfall, der später unter der Bezeichnung **Überlastungskollaps** in die Geschichte eingehen sollte – ein längerer Zeitraum, in dem der Datendurchsatz aufgrund einer Verstopfung im Netz jäh (d.h. um mehr als den Faktor 100) fiel. Jacobson (so wie viele andere) versuchte zu verstehen, was passiert war, und suchte nach einer Lösung.

Die von Jacobson gewählte Implementierung bestand in der Annäherung an ein AIMD-Überlastungsfenster. Der interessante Teil, der außerdem stark zur Komplexität der TCP-Überlastungsüberwachung beitrug, ist dabei, wie es ihm gelang, eine bestehende Implementierung zu ergänzen, ohne eines der Nachrichtenformate zu ändern, sodass es direkt einsetzbar war. Zum einen beobachtete er, dass Paketverlust sich gut als Überlastungssignal eignete. Dieses Signal kommt zwar ein wenig spät (da das Netz bereits verstopft ist), ist aber ziemlich zuverlässig. Schließlich ist es schwer, einen Router zu entwickeln, der keine Pakete verliert, wenn er überlastet ist. Das wird sich auch nicht ändern. Denn selbst wenn wir über Terabyte-Speicher verfügen, die eine riesige Anzahl von Paketen puffern können, haben wir es wahrscheinlich mit einem Terabit/s-Netz zu tun, um diesen Speicher zu füllen.

Die Verwendung von Paketverlust als Überlastungssignal setzt voraus, dass Übertragungsfehler relativ selten sind. Dies ist für drahtlose Verbindungen wie IEEE 802.11 normalerweise nicht der Fall, weshalb sie auch ihre eigenen Mechanismen zur erneuten Übertragung auf der Sicherungsschicht mit sich bringen. Aufgrund der drahtlosen Neuübertragungen wird der durch Übertragungsfehler auf der Sicherungsschicht verursachte Paketverlust normalerweise in drahtlosen Netzen maskiert. Er ist auch auf anderen Verbindungen selten, da Leitungen und Glasfaserkabel normalerweise niedrige Bitfehlerraten haben.

Alle TCP-Algorithmen im Internet gehen davon aus, dass verlorene Pakete auf Überlastung zurückzuführen sind, und überwachen Timeouts und suchen nach Anzeichen von Problemen wie früher Bergleute ihre Kanarienvögel beobachteten. Es wird ein guter Retransmission-Timer benötigt, um die Signale des Paketverlusts genau und rechtzeitig zu entdecken. Wir haben bereits diskutiert, wie der TCP-Retransmission-Timer Schätzungen von Mittelwert und Varianz der Übertragungszeiten berücksichtigt. Die Verbesserung dieses Timers durch die Ergänzung eines Varianzfaktors war ein wichtiger Schritt in Jacobsons Arbeit. Mithilfe eines guten Timeouts für Neuübertragungen kann der TCP-Sender die ausstehende Anzahl Bytes verfolgen, die das Netz verstopfen. Er schaut einfach auf die Differenz zwischen den übertragenen und bestätigten Sequenznummern.

Damit scheint unsere Aufgabe jetzt sehr einfach. Alles, was wir machen müssen, ist das Überlastungsfenster anhand der Folge- und Bestätigungsnummern zu überwachen und dann mittels einer AIMD-Regel anzupassen. Doch wie zu erwarten war, ist es natürlich viel komplizierter. Eine erste Überlegung ist, dass die Art, wie Pakete, auch über kurze Zeiträume, ins Netz gestellt werden, auf den Netzpfad abgestimmt werden muss. Andernfalls kommt es zu einer Verkehrsüberlastung. Betrachten wir beispielsweise einen Host mit einem Überlastungsfenster von 64 KB, der mit einem 1-Gbit/s-Switched-Ethernet verbunden ist. Wenn der Host das ganze Fenster in einem Schub (Burst) sendet, kann diese riesige Datenmenge über eine langsame 1-Mbit/s-ADSL-Leitung weiter den Pfad entlang wandern. Der Schub, der auf einer 1-Gbit/s-Leitung nur eine halbe Millisekunde benötigt, wird die 1-Mbit/s-Leitung für eine halbe Sekunde blockieren und damit Protokolle wie Voice-over-IP völlig lahmlegen. Dieses Verhalten mag eine gute Idee für ein Protokoll sein, das eine Überlastung auslösen soll, aber nicht für ein Protokoll, um die Überlastung zu überwachen.

Es zeigt sich jedoch, dass wir kleine Paketschübe zu unserem Vorteil nutzen können. ▶ Abbildung 6.43 zeigt, was passiert, wenn ein Sender auf einem schnellen Netz (der 1-Gbit/s-Verbindung) eine kleine Schubeinheit von vier Paketen an einen Empfänger auf einem langsamen Netz (der 1-Mbit/s-Verbindung) schickt, das den Engpass oder den langsamsten Teil des Pfades darstellt. Am Anfang reisen die vier Pakete über die Verbindung so schnell, wie sie vom Sender geschickt werden können. Beim Router laufen sie auf, weil es länger dauert, ein Paket über die langsame Verbindung zu schicken, als das nächste Paket über die schnelle Verbindung zu empfangen. Aber die Warteschlange ist nicht lang, da jeweils nur eine kleine Anzahl von Paketen gleichzeitig gesendet werden. Beachten Sie die zunehmende Länge der Pakete auf der langsamen Verbindung. Das gleiche Paket, von sagen wir 1 KB, ist jetzt länger, da es länger dauert, es auf einer langsamen Verbindung zu senden als auf einer schnellen.

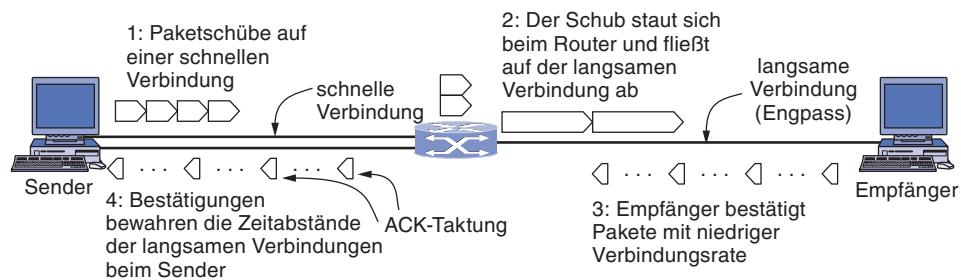


Abbildung 6.43: Ein Schub von Paketen vom Sender und die zurückgelieferte ACK-Taktung.

Schließlich treffen die Pakete beim Empfänger ein, wo sie bestätigt werden. Die Zeitangaben für die Bestätigungen geben an, wann die Pakete nach Passieren der langsamen Verbindung beim Empfänger eingetroffen sind. Verglichen mit den ursprünglichen Paketen auf der schnellen Verbindung liegen sie weiter auseinander. Bei der Rückreise der Bestätigungen über das Netz zurück zum Sender ändern sich diese Zeitabstände nicht.

Die wichtigste Beobachtung hierbei ist: Das Zurücksender der Bestätigungen an den Sender erfolgt mit der gleichen Übertragungsrate wie das Zusenden der Pakete über die langsamste Verbindung im Pfad. Dies ist genau die Übertragungsrate, die der Sender verwenden möchte. Wenn der Sender anschließend neue Pakete mit dieser Rate in das Netz stellt, werden sie genau so schnell übertragen, wie es die langsame Verbindung erlaubt, ohne dass es zu Staus und Verstopfungen bei irgendwelchen Routern auf der Strecke kommt. Diese Anpassung wird als **ACK-Taktung** (*ack clock*) bezeichnet. Er ist ein wesentlicher Bestandteil von TCP. Durch die Verwendung einer ACK-Taktung glättet TCP den Verkehr und vermeidet unnötige Warteschlangen vor den Routern.

Die zweite Überlegung ist, dass die AIMD-Regel sehr lange braucht, um auf schnellen Netzen einen guten Arbeitspunkt zu erreichen, wenn das Überlastungsfenster von einer kleinen Größe ausgeht. Betrachten wir dazu einen einfachen Netzpfad, der 10 Mbit/s mit einer Paketumlaufzeit von 100 ms unterstützen kann. Das dazu passende Überlastungsfenster entspricht dem Bandbreite-Verzögerung-Produkt (1 MB oder 100 Pakete von jeweils 1 250 Byte). Wenn das Überlastungsfenster mit einem Paket beginnt und sich bei jedem Paketumlauf um 1 Paket erhöht, wird es 100 Umlaufzeiten oder

10 Sekunden dauern, bis die Verbindung in etwa die richtige Rate aufweist. Das ist eine lange Wartezeit, nur um die richtige Übertragungsgeschwindigkeit zu finden. Wir könnten diese Anlaufzeit reduzieren, indem wir von einem größeren Anfangsfenster von sagen wir 50 Paketen ausgehen. Aber dieses Fenster wäre für langsame oder kurze Verbindungen viel zu groß. Wie oben beschrieben, käme es zu einem Stau, wenn es komplett auf einmal verwendet würde.

Die Lösung, die Jacobson wählte, um beiden Überlegungen gerecht zu werden, besteht aus einer Mischung von linearer und multiplikativer Zunahme. Beim Aufbau einer Verbindung initialisiert der Sender das Überlastungsfenster mit einem kleinen Anfangswert von maximal vier Segmenten. Die Einzelheiten sind in RFC 3390 beschrieben; die Verwendung von vier Segmenten basiert auf Erfahrungen und stellt eine Erhöhung des früheren Anfangswertes von einem Segment dar. Anschließend sendet der Sender das Anfangsfenster. Die Pakete benötigen einen Paketumlaufzyklus, um bestätigt zu werden. Für jedes Segment, das bestätigt wird, bevor der Retransmission-Timer abläuft, werden dem Überlastungsfenster vom Sender die Bytes im Umfang eines neuen Segments hinzugefügt. Und da das Segment nun bestätigt wurde, gibt es ein Segment weniger im Netz. Fazit ist, dass für jedes bestätigte Segment zwei weitere Segmente gesendet werden können. Das Überlastungsfenster verdoppelt sich sozusagen mit jedem Paketumlaufzyklus.

Dieser Algorithmus heißt **Slow-Start**, wiewohl er aufgrund seines exponentiellen Wachstums überhaupt nicht langsam ist – außer im Vergleich zum vorherigen Algorithmus, bei dem ein ganzes Flusskontrollfenster auf einmal gesendet wurde. Einen Slow-Start sehen Sie in ▶ Abbildung 6.44. Beim ersten Paketumlauf schickt der Sender ein Paket über das Netz (und der Empfänger nimmt ein Paket entgegen). Beim nächsten Paketumlaufzyklus sind es schon zwei Pakete, beim dritten Paketumlauf vier usw.

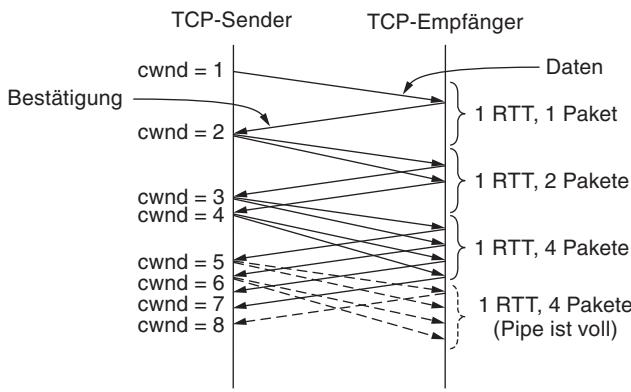


Abbildung 6.44: Slow-Start ausgehend von einem anfänglichen Überlastungsfenster ($cwnd$) von einem Segment.

Slow-Start eignet sich sehr gut für eine Reihe von Verbindungsgeschwindigkeiten und Paketumlaufzeiten und verwendet eine ACK-Taktung, um die Übertragungsrate des Senders an den Netzpfad anzupassen. Betrachten wir einmal genau, wie die Bestätigungen in Abbildung 6.44 vom Sender zum Empfänger zurückkehren. Wenn der Sender eine Bestätigung erhält, erhöht er das Überlastungsfenster um eins und schickt

sofort zwei Pakete ins Netz. (Ein Paket ist die Erhöhung um eins, das andere der Ersatz für das bestätigte Paket, das das Netz verlassen hat. Stets wird die Anzahl der nicht bestätigten Pakete vom Überlastungsfenster vorgegeben.) Der Abstand dieser beiden Pakete muss jedoch bei der Ankunft beim Empfänger nicht notwendigerweise genauso eng sein wie beim Losschicken. Angenommen, der Sender ist an ein Ethernet mit 100 Mbit/s angeschlossen. Jedes 1 250-Byte-Paket benötigt zum Senden 100 µs, sodass die Verzögerung zwischen den Paketen minimal bis zu 100 µs betragen kann. Die Situation ändert sich, wenn diese Pakete irgendwann auf ihrem Weg eine ADSL-Verbindung von 1 Mbit/s passieren. Das gleiche Paket benötigt jetzt für die Strecke 10 ms. Das bedeutet, dass der minimale Abstand zwischen den beiden Paketen um den Faktor 100 ansteigt. Dieser Abstand wird groß bleiben, es sei denn die Pakete stehen irgendwann später in einer Warteschlange vor einer nachfolgenden Verbindung.

In Abbildung 6.44 wird dieser Effekt deutlich, indem ein Minimalabstand zwischen den Datenpaketen erzwungen wird, die beim Empfänger ankommen. Dieser Abstand wird beim Versenden der Bestätigungen beibehalten und somit auch beim Empfang der Bestätigungen. Wenn der Netzpfad langsam ist, kommen die Bestätigungen langsam an (nach einer Paketumlaufzeit). Wenn der Netzpfad schnell ist, kommen die Bestätigungen schnell an (ebenfalls nach dem Paketumlauf). Alles, was der Sender machen muss, ist, sich beim Verschicken neuer Pakete an die Zeitabstände der ACK-Taktung zu halten. Und genau das ist es, was der Slow-Start macht.

Da der Slow-Start mit exponentiellem Wachstum verbunden ist, werden irgendwann (und eher früher als später) zu viele Pakete ins Netz gestellt. Wenn dies passiert, kommt es im Netz zur Bildung von Warteschlangen. Sind die Warteschlangen voll, gehen ein oder mehrere Pakete verloren. Als Folge einer nicht rechtzeitig ankommenden Bestätigung sendet der TCP-Sender ein Timeout-Signal. Abbildung 6.44 zeigt, wie ein Slow-Start zu schnell anwachsen kann. Nach drei Paketumlaufzeiten befinden sich vier Pakete im Netz. Diese vier Pakete benötigen eine ganze Paketumlaufzeit, um beim Empfänger anzukommen. Das heißt, ein Überlastungsfenster von vier Paketen ist die richtige Größe für diese Verbindung. Nachdem diese Pakete jedoch bestätigt wurden, erhöht Slow-Start das Überlastungsfenster weiter, und zwar auf acht Pakete in einer weiteren Paketumlaufzeit. Nur vier von diesen Paketen können den Empfänger innerhalb einer Paketumlaufzeit erreichen, egal wie viele gesendet werden. Das heißt, die Netzzippe ist voll. Zusätzliche Pakete, die vom Sender in das Netz gestellt werden, stauen sich in Router-Warteschlangen an, da sie nicht schnell genug beim Empfänger zugestellt werden können. Es wird schon bald zu einer Überlastung und Paketverlust kommen.

Um den Slow-Start unter Kontrolle zu bekommen, gibt der Sender für die Verbindung einen Schwellenwert vor, den sogenannten **Slow-Start-Schwellenwert**. Am Anfang wird dieser Wert beliebig hoch gesetzt, und zwar auf die Größe des Flusskontrollfensters, sodass er die Verbindung nicht beschränkt. TCP erhöht das Überlastungsfenster beim Slow-Start so lange, bis ein Timeout gemeldet wird oder das Überlastungsfenster den Schwellenwert überschreitet (oder das Fenster des Empfängers gefüllt ist).

Immer wenn der Verlust eines Pakets festgestellt wird, beispielsweise durch ein Timeout, wird der Slow-Start-Schwellenwert auf die halbe Größe des Überlastungsfensters

gesetzt und der ganze Prozess beginnt von vorn. Der Gedanke dahinter ist, dass das aktuelle Fenster zu groß ist, weil es die vorherige Überlastung verursachte, die erst jetzt durch das Timeout entdeckt wurde. Die Hälfte, die zuvor erfolgreich verwendet wurde, ist wahrscheinlich eine bessere Schätzung für das Überlastungsfenster, denn es liegt nah an der Pfadkapazität, verursacht aber keinen Verlust. In unserem Beispiel aus Abbildung 6.44 kann das Anwachsen des Überlastungsfensters auf acht Pakete zu Verlusten führen, während das Überlastungsfenster von vier Paketen im vorherigen Paketumlauf der richtige Wert war. Das Überlastungsfenster wird dann auf seinen kleinen Anfangswert zurückgesetzt und der Slow-Start geht weiter.

Immer wenn der Slow-Start-Schwellenwert überschritten wird, wechselt TCP vom Slow-Start in den Modus der additiven Erhöhung. In diesem Modus wird das Überlastungsfenster bei jedem Paketumlauf um ein Segment erhöht. Wie beim Slow-Start sieht die Implementierung normalerweise eine Erhöhung für jedes bestätigte Segment vor anstatt einer Erhöhung pro Paketumlaufzeit. Wir bezeichnen das Überlastungsfenster mit $cwnd$ und die maximale Segmentgröße mit MSS . Eine häufige Annäherung ist es, $cwnd$ um $(MSS \times MSS)/cwnd$ für jedes der $cwnd/MSS$ Pakete zu erhöhen, die bestätigt werden können. Diese Erhöhung muss nicht schnell erfolgen. Die Idee dahinter ist, dass das Überlastungsfenster in der TCP-Verbindung möglichst lange einen optimalen Wert einnimmt – nicht zu klein, sodass der Durchsatz sehr niedrig ist, und nicht zu groß, sodass es zu einer Überlastung kommt.

Die additive Erhöhung finden Sie in ►Abbildung 6.45, und zwar für die gleiche Situation wie beim Slow-Start. Nach jedem Paketumlauf ist das Überlastungsfenster des Senders wieder soweit angewachsen, dass er ein weiteres Paket in das Netz stellen kann. Im Vergleich zum Slow-Start ist die lineare Wachstumsrate viel langsamer. Für kleine Überlastungsfenster wie dieses hier ist der Unterschied nicht so gravierend, aber hinsichtlich der Zeit, die es dauert, ein Überlastungsfenster auf, sagen wir, 100 Segmente anwachsen zu lassen, ist der Unterschied schon gravierend.

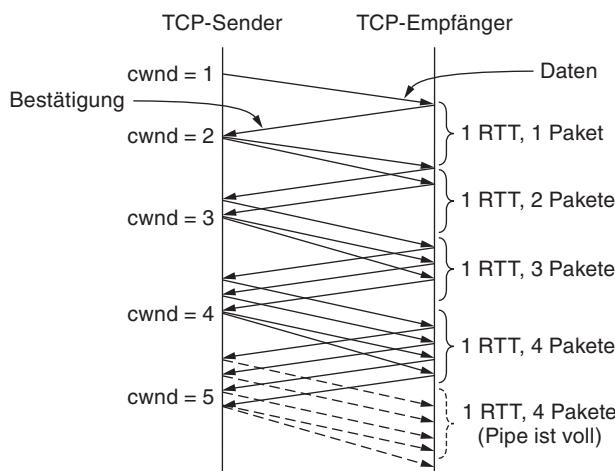


Abbildung 6.45: Additive Erhöhung ausgehend von einem anfänglichen Überlastungsfenster ($cwnd$) der Größe eines Segments.

Es gibt noch etwas anderes, was wir machen können, um die Leistung zu verbessern. Der bisherige Plan hat ein Manko: das Warten auf Timeouts. Timeouts sind relativ lang, da sie konservativ sein müssen. Nachdem ein Paket verloren gegangen ist, kann der Empfänger nachfolgende Pakete nicht bestätigen, sodass die Bestätigungsnummer unverändert bleibt und der Sender keine neuen Pakete ins Netz stellen kann, weil sein Überlastungsfenster voll ist. Dieser Zustand kann relativ lange anhalten, und zwar bis der Timer sich meldet und das verlorene Paket erneut übertragen wird. An diesem Punkt beginnt der Slow-Start bei TCP erneut.

Es gibt eine schnelle Möglichkeit für den Sender zu erkennen, dass eines seiner Pakete verloren gegangen ist. Wenn nach dem verlorenen Paket weitere Pakete beim Empfänger ankommen, lösen sie Bestätigungen aus, die an den Sender zurückkehren. Diese Bestätigungen tragen alle die gleiche Bestätigungsnummer. Sie heißen **Dup-Acks** oder **Bestätigungsduplikate** (*duplicate acknowledgement*). Jedes Mal wenn der Sender ein Dup-Ack empfängt, ist es wahrscheinlich, dass ein weiteres Paket beim Empfänger angekommen ist und das verlorene Paket immer noch nicht aufgetaucht ist.

Da Pakete verschiedene Pfade durch das Netz nehmen können, kann es passieren, dass sie nicht in der richtigen Reihenfolge ankommen. Dies löst Dup-Acks aus, auch wenn keine Pakete verloren gegangen sind. Dies kommt jedoch im Internet nicht sehr häufig vor. Selbst wenn die Pakete bei der Übertragung über verschiedene Pfade laufen sollten, ist die Reihenfolge der empfangenen Pakete normalerweise relativ korrekt. Deshalb geht TCP etwas willkürlich davon aus, dass drei Dup-Acks den Verlust eines Pakets implizieren. Die Identität des verlorenen Pakets kann aus der Bestätigungsnummer abgeleitet werden. Es ist das nächste Paket in der Folge. Dieses Paket kann dann direkt neu übertragen werden, bevor der Retransmission-Timer sich meldet.

Diese Heuristik wird **schnelle Neuübertragung** (*fast retransmission*) genannt. Wird sie ausgelöst, ist der Slow-Start-Schwellenwert immer noch auf die Hälfte des aktuellen Überlastungsfensters gesetzt – genau wie beim Timeout. Slow-Start kann neu gestartet werden, indem das Überlastungsfenster auf ein Paket gesetzt wird. Bei dieser Fenstergröße wird ein neues Paket gesendet, sobald die Bestätigungen für das neu übertragene Paket und alle Daten eingegangen sind, die vor dem Entdecken des Datenverlustes gesendet wurden.

Einen Eindruck von dem Überlastungsalgorithmus, wie wir ihn bisher entwickelt haben, erhalten Sie in ►Abbildung 6.46. Diese TCP-Version wird TCP Tahoe genannt, und zwar nach dem 4.2BSD-Tahoe-Release 1988, von dem sie ein Teil war. Die maximale Segmentgröße ist 1 KB. Am Anfang war das Übertragungsfenster 64 KB groß, aber ein Timeout erfolgte, sodass der Schwellenwert auf 32 KB festgesetzt wird und das Überlastungsfenster auf 1 KB für die Übertragung 0. Das Übertragungsfenster wächst exponentiell, bis es den Schwellenwert (32 KB) erreicht. Das Fenster wird nicht ständig, sondern nur jedes Mal, wenn eine neue Bestätigung eingeht, erhöht, was zu dem diskreten Treppenmuster führt. Nachdem der Schwellenwert passiert wurde, wächst das Fenster linear. Es wird um ein Segment pro Paketumlaufzeit erhöht.

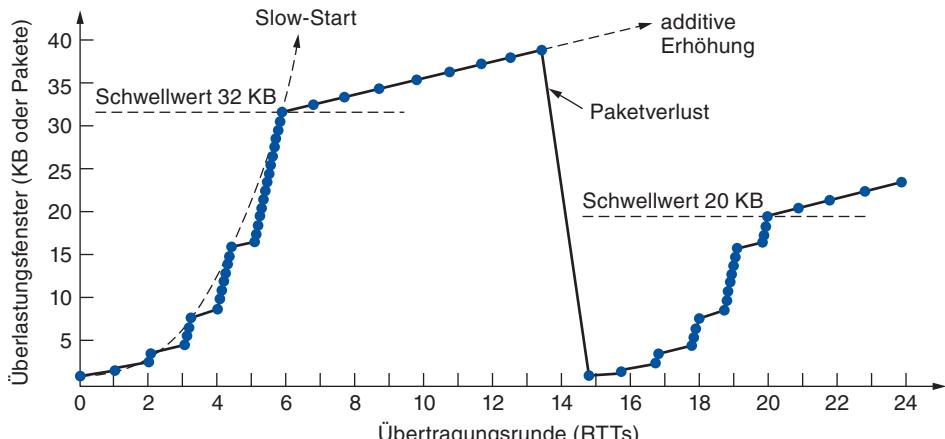


Abbildung 6.46: Slow-Start gefolgt von einer additiven Erhöhung in TCP Tahoe.

Die Übertragungen in Runde 13 haben Pech (das war irgendwann zu erwarten), denn ein Paket ist im Netz verloren gegangen. Dies kommt heraus, wenn drei Dup-Acks eingehen. Daraufhin wird das verlorene Paket erneut übertragen, der Schwellenwert auf die Hälfte des aktuellen Fensters gesetzt (inzwischen 40 KB, d.h., die Hälfte ist 20 KB) und der Slow-Start von neuem initiiert. Das Neustarten mit einem Überlastungsfenster von einem Paket benötigt einen Paketumlaufzyklus, damit all die zuvor übertragenen Daten das Netz verlassen und bestätigt werden, einschließlich des neu übertragenen Pakets. Das Überlastungsfenster wächst bei dem Slow-Start wie zuvor, bis es den neuen Schwellenwert von 20 KB erreicht. Ab diesem Zeitpunkt verläuft das Wachstum wieder linear. Damit wird fortgefahrene, bis ein weiterer Paketverlust anhand von Dup-Acks oder einem Timeout festgestellt wird (oder das Fenster des Empfängers zum Grenzwert wird).

TCP Tahoe (zu dem gute Retransmission-Timer gehören) bot einen funktionierenden Algorithmus zur Überlastungsüberwachung, der das Problem des Überlastungskollapses löste. Jacobson erkannte, dass es sogar noch besser ging. Zurzeit der schnellen Neuübertragung weist die Verbindung ein zu großes Überlastungsfenster auf, aber nichtsdestotrotz eine funktionierende ACK-Taktung. Jedes Mal, wenn ein weiteres Dup-Ack ankommt, hat ein anderes Paket das Netz verlassen. Die Verwendung von Dup-Acks zum Zählen der Pakete im Netz ermöglicht es, einige Pakete aus dem Netz zu entlassen und weiterhin für jedes zusätzliche Dup-Ack ein neues Paket zu schicken.

Die Heuristik, die dieses Verhalten implementiert, ist die **schnelle Wiederherstellung** (*fast recovery*). Sie ist ein temporärer Modus, der darauf abzielt, die ACK-Taktung nicht zu unterbrechen, mit einem Überlastungsfenster, das so groß ist wie der neue Schwellenwert oder halb so groß wie das Überlastungsfenster zurzeit der schnellen Neuübertragung. Hierzu werden die Dup-Acks gezählt (einschließlich der drei, die die schnelle Neuübertragung ausgelöst haben), bis die Anzahl der Pakete im Netz auf den neuen Schwellenwert gefallen ist. Dies dauert ungefähr eine halbe Paketumlaufzeit. Ab da kann für jedes Dup-Ack, das empfangen wurde, ein neues Paket gesendet werden. Einen Paketumlaufzyklus nach der schnellen Neuübertragung wird das verlorene

Paket bestätigt sein. Zu diesem Zeitpunkt hört der Strom der Dup-Acks auf und der Modus der schnellen Wiederherstellung wird verlassen. Das Überlastungsfenster wird auf den neuen Slow-Start-Schwellenwert gesetzt und wächst danach linear an.

Diese Heuristik soll bei TCP Slow-Start möglichst vermeiden, es sei denn, die Verbindung wird das erste Mal gestartet und es tritt ein Timeout auf. Letzteres kann immer noch passieren, wenn mehr als ein Paket verloren geht und die schnelle Neuübertragung nicht zu einer angemessenen Wiederherstellung führt. Anstelle von wiederholtem Slow-Start folgt das Überlastungsfenster einer aktiven Verbindung einem **Sägezahn-muster** von additiver Erhöhung (ein Segment pro Paketumlaufzeit) und multiplikativer Erniedrigung (um die Hälfte in einer Paketumlaufzeit). Das ist genau die AIMD-Regel, die wir implementieren wollen.

Das Sägezahnverhalten sehen Sie in ► Abbildung 6.47. Es wird von der Version TCP Reno erzeugt, benannt nach dem 4.3BSD-Reno-Release 1990, von dem sie ein Teil war. TCP Reno entspricht im Wesentlichen TCP Tahoe plus der schnellen Wiederherstellung. Nach einem anfänglichen Slow-Start steigt das Überlastungsfenster linear an, bis ein Paketverlust durch Dup-Acks festgestellt wird. Das verlorene Paket wird erneut übertragen und die schnelle Wiederherstellung sorgt dafür, dass die ACK-Taktung weiterläuft, bis die erneute Übertragung bestätigt wurde. Dann fährt das Überlastungsfenster mit dem neuen Slow-Start-Schwellenwert fort anstatt mit 1. Dieses Verhalten wird unendlich fortgesetzt, wobei das Überlastungsfenster in der Verbindung die meiste Zeit nahe dem Optimalwert des Bandbreite-Verzögerung-Produkts ist.

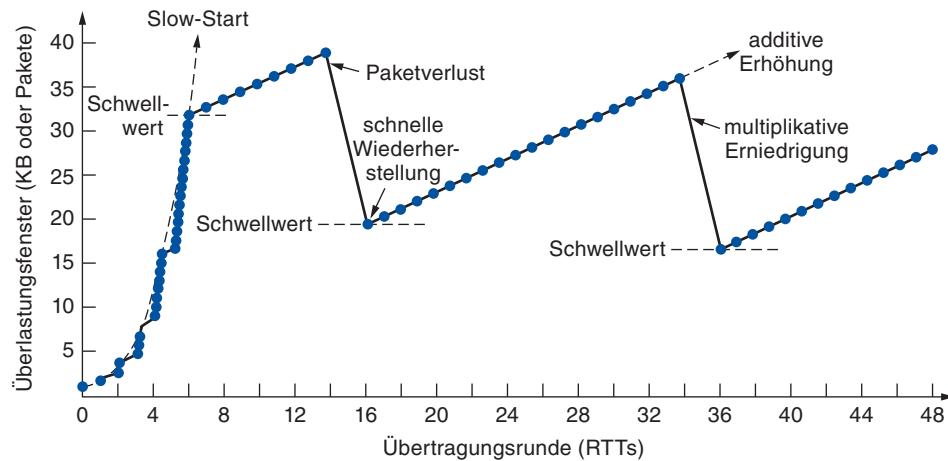


Abbildung 6.47: Schnelle Wiederherstellung und das Sägezahn-Muster von TCP Reno.

TCP Reno mit seinen Mechanismen zur Anpassung des Überlastungsfensters bildet seit zwei Jahrzehnten die Basis für die TCP-Überlastungsüberwachung. Die meisten Änderungen in dieser Zeit waren kleinere Korrekturen an diesen Mechanismen. So wurden zum Beispiel die Auswahlmöglichkeiten des Anfangsfensters geändert und diverse Doppeldeutigkeiten beseitigt. Auch gab es einige Verbesserungen zur Wiederherstellung nach zwei oder mehr Verlusten in einem Fenster von Paketen. Zum Bei-

spiel verwendet die Version TCP NewReno ein teilweises Weiterzählen der Bestätigungsnummer nach einer Neuübertragung, um einen weiteren Verlust zu entdecken und zu beheben (Hoe, 1996); dies wird in RFC 3782 beschrieben. Seit Mitte der 1990er Jahre gibt es mehrere Varianten, die den von uns beschriebenen Prinzipien folgen, aber etwas andere Steuerungsgesetze zugrunde legen. So verwendet Linux beispielsweise eine Variante namens CUBIC TCP (Ha et al., 2008) und Windows eine namens Compound TCP (Tan et al., 2006).

Zwei größere Änderungen hatten ebenfalls Einfluss auf die TCP-Implementierungen. Zum einen ist ein Großteil der Komplexität von TCP darauf zurückzuführen, dass einem Strom von Dup-Acks entnommen werden kann, welche Pakete angekommen und welche verloren gegangen sind. Die kumulative Bestätigungsnummer kann keine Auskunft darüber geben. Eine einfache Lösung ist die Verwendung von **selektiven Bestätigungen (SACK, Selective ACKnowledgments)**, die bis zu drei Bytebereiche auflisten, die empfangen wurden. Mit dieser Information kann der Sender schneller entscheiden, welche Pakete neu zu übertragen sind, und die Pakete, die unterwegs sind, verfolgen, um das Überlastungsfenster zu implementieren.

Wenn Sender und Empfänger eine Verbindung aufbauen, senden beide die TCP-Option *SACK Permitted*, um anzudeuten, dass sie selektive Bestätigungen verstehen. Wurde diese Option für eine Verbindung aktiviert, sieht die Funktionsweise wie in ►Abbildung 6.48 aus. Ein Empfänger verwendet das TCP-Feld *Acknowledgement Number* auf normale Art und Weise, als eine kumulative Bestätigung des höchsten Bytes, das in der korrekten Reihenfolge empfangen wurde. Wenn ihm Paket 3 außerhalb der Reihenfolge zugestellt wird (weil Paket 2 verloren gegangen ist), sendet er eine *SACK Option* für die empfangenen Daten zusammen mit der kumulativen Bestätigung (Dup-Ack) für Paket 1. Die *SACK Option* gibt die Bytebereiche an, die über die Zahl, die durch die kumulative Bestätigung angegeben wird, hinaus empfangen wurden. Der erste Bereich ist das Paket, das das Dup-Ack ausgelöst hat. Die nächsten Bereiche, falls vorhanden, sind ältere Blöcke. Normalerweise werden bis zu drei Bereiche verwendet. Wenn Paket 6 empfangen wurde, werden zwei SACK-Bytebereiche verwendet, um anzudeuten, dass Paket 6 und Pakete 3 bis 4 empfangen wurden, zusätzlich zu allen Paketen bis Paket 1. Anhand der Informationen, die der Sender durch jede SACK-Option empfängt, kann er entscheiden, welche Pakete neu zu übertragen sind. In diesem Fall wäre die erneute Übertragung der Pakete 2 und 5 eine gute Idee.

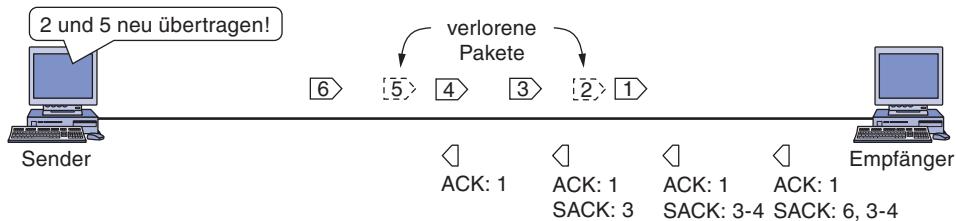


Abbildung 6.48: Selektive Bestätigungen.

SACK hat lediglich informativen Charakter. Die eigentliche Ermittlung eines Verlustes erfolgt wie zuvor mithilfe von Dup-Acks und Anpassungen des Überwachungsfensters. Doch mit SACK kann TCP in Situationen, in denen mehrere Pakete mehr oder weniger gleichzeitig verloren gehen, die Wiederherstellung viel schneller realisieren, da der TCP-Sender weiß, welche Pakete nicht empfangen wurden. SACK hat inzwischen eine weite Verbreitung gefunden. Es wird in RFC 2883 beschrieben und die TCP-Überlastungsüberwachung mit SACK in RFC 3517.

Die zweite Änderung ist die Verwendung von ECN (*Explicit Congestion Notification*) zusätzlich zu dem Paketverlust als Überlastungssignal. ECN ist ein Mechanismus auf der IP-Schicht, der Hosts über die Überlastung informiert. Wir sind hierauf schon in *Kapitel 5.3.4* eingegangen. Damit kann der TCP-Empfänger Überlastungssignale von IP empfangen.

Die Verwendung von ECN für eine TCP-Verbindung wird aktiviert, wenn sowohl Sender als Empfänger anzeigen, dass sie ECN verwenden können, indem sie während des Verbindungsbaus die Bits *ECE* und *CWR* setzen. Wenn ECN verwendet wird, ist bei jedem Paket, das ein TCP-Segment überträgt, ein Flag im IP-Header gesetzt, um anzudeuten, dass es ein ECN-Signal tragen kann. Router, die ECN unterstützen, werden bei Paketen, die ECN-Flags tragen können, ein Überlastungssignal setzen, wenn eine Überlastung droht, anstatt diese Pakete fallen zu lassen, nachdem eine Überlastung aufgetreten ist.

Der TCP-Empfänger wird darüber informiert, ob irgendeines der Pakete, die ankommen, ein ECN-Überlastungssignal trägt. Der Empfänger verwendet dann das *ECE*-Flag (ECN-Echo), um dem TCP-Sender zu signalisieren, dass die angekommenen Pakete im Stau gestanden haben. Der Sender teilt dem Empfänger mit, dass er das Signal gehört hat, indem er seinerseits das *CWR*-Flag verwendet (*Congestion Window Reduced*).

Der TCP-Sender reagiert auf diese Überlastungsbemerkungen auf genau die gleiche Weise wie bei einem Paketverlust, der mittels Dup-Acks festgestellt wird. Die Situation ist jedoch eindeutig besser. Die Überlastung wurde entdeckt und kein Paket hat in irgendeiner Form Schaden erlitten. ECN wird in RFC 3168 beschrieben. Es muss sowohl vom Host und vom Router unterstützt werden und ist noch nicht sehr weit im Internet verbreitet.

Weitere Informationen zu dem vollständigen Satz der in TCP implementierten Überlastungsüberwachungsmethoden finden Sie in RFC 5681.

6.5.11 Die Zukunft von TCP

Als Arbeitspferd des Internets wird TCP für viele Anwendungen eingesetzt und hat mit der Zeit etliche Erweiterungen erfahren, um in einer Vielzahl von Netzen eine gute Leistung zu zeigen. Es gibt viele Versionen mit jeweils leicht abgewandelten Implementierungen der von uns beschriebenen klassischen Algorithmen, vor allem zur Verbesserung von Überlastungsüberwachung und Robustheit gegen Angriffe. Es ist zu erwarten, dass sich TCP auch in Zukunft mit dem Internet weiterentwickelt. Auf zwei Aspekte in diesem Zusammenhang wollen wir besonders eingehen.

Der erste Aspekt ist, dass die Transportsemantik von TCP nicht allen Anwendungen gleich gerecht wird. Einige Anwendungen wollen beispielsweise Nachrichten oder Datensätze schicken, deren Umfang bewahrt werden muss. Andere Anwendungen nutzen eine Gruppe verwandter Konversationsprogramme wie einen Webbrowsert, der mehrere Objekte vom gleichen Server überträgt. Oder noch andere Anwendungen wollen bessere Kontrolle über die von ihnen verwendeten Netzpfade. TCP mit seiner standardmäßigen Socket-Schnittstelle kann all diese Anforderungen nicht mehr voll erfüllen. Kurz und gut, die Anwendung muss sich um all die Probleme kümmern, die TCP nicht lösen kann. Dies führte zu Vorschlägen für neue Protokolle, die eine etwas andere Schnittstelle boten. Zwei Beispiele dafür sind SCTP (*Stream Control Transmission Protocol*), definiert in RFC 4960, und SST (*Structured Stream Transport*; Ford, 2007). Jedoch immer, wenn irgendjemand vorschlägt, etwas zu ändern, dass so lange gut funktioniert hat, gibt es einen Grabenkampf zwischen denen, die meinen, „dass die Benutzern mehr Funktionen verlangen“, und den Anhängern des Spruchs, „dass man nichts reparieren sollte, was nicht kaputt ist“.

Der zweite Aspekt ist die Überlastungsüberwachung. Vielleicht haben Sie erwartet, dass nach all unseren Überlegungen und den mit der Zeit entwickelten Mechanismen dieses Problem gelöst wäre. Weit gefehlt. Die Art der TCP-Überlastungsüberwachung, die wir beschrieben haben und die weitverbreitet ist, basiert auf Paketverlusten als Signal der Überlastung. Als Padhye et al. (1998) den TCP-Durchsatz auf der Basis des Sägezahn-musters modellierten, stellten sie fest, dass die Paketverlustrate mit zunehmender Geschwindigkeit rapide zurückgehen muss. Um einen Durchsatz von 1 Gbit/s mit einer Paketumlaufzeit von 100 ms und 1 500 Bytepakete zu erreichen, kann ungefähr alle 10 Minuten ein Paket verloren gehen. Das ist eine Paketverlustrate von 2×10^{-8} , was unglaublich niedrig ist. Der Paketverlust ist somit zu sporadisch, um als gutes Überlastungssignal zu dienen, und jede andere Verlustquelle (z.B. Paketübertragungsfehlerraten von 10^{-7}) kann ihn leicht dominieren und so den Durchsatz beschränken.

Diese Beziehung war in der Vergangenheit kein Problem, doch werden die Netze zunehmend schneller, was viele veranlasst, sich erneut mit der Überlastungsüberwachung zu befassen. Eine Möglichkeit ist, eine alternative Überlastungsüberwachung zu verwenden, in der das Signal nicht der Paketverlust ist. Wir haben mehrere Beispiele in Abschnitt 6.2 angeführt. Beispielsweise könnte wie bei FAST TCP (Wie et al., 2006) die Paketumlaufzeit als Signal dienen, die ansteigt, wenn das Netz verstopft. Es sind aber auch andere Ansätze denkbar, von denen erst die Zeit zeigen wird, welcher am besten geeignet ist.

6.6 Leistungsaspekte

Leistungsaspekte sind in Rechnernetzen sehr wichtig. Wenn Hunderte oder Tausende von Rechnern miteinander verbunden sind, ergeben sich oft komplexe Interaktionen und unvorhergesehene Folgen. Häufig führt diese Komplexität zu schlechter Leistung, und keiner weiß den Grund. In den folgenden Abschnitten werden viele Aspekte im Zusammenhang mit Netzeistung und mögliche Problemlösungen beschrieben.

Leider ist das Verständnis der Netzleistung mehr Kunst als Wissenschaft. Es gibt kaum eine zugrunde liegende Theorie, die in der Praxis Anwendung findet. Das Beste, was man in diesem Umfeld tun kann, ist die Weitergabe von Faustregeln, die aus praktischen Erfahrungen gesammelt wurden, und die Präsentation von Beispielen der realen Welt. Wir haben absichtlich damit bis zur Behandlung der Transportschicht in TCP gewartet, da die Leistung, die bei den Anwendungen ankommt, von der gemeinsamen Leistung der Transport-, Vermittlungs- und Sicherungsschicht abhängt und wir TCP als Beispiel in verschiedenen Bereichen verwenden können.

Die Transportschicht ist nicht die einzige Stelle, an der Leistungsaspekte berücksichtigt werden müssen. Wir haben einige davon schon im Zusammenhang mit der Vermittlungsschicht im vorherigen Kapitel angeschnitten. Dennoch neigt die Vermittlungsschicht dazu, sich größtenteils mit Routing und Überlastungsüberwachung zu befassen. Die umfassenderen, systemorientierten Aspekte haben eher mit der Transportschicht zu tun; deshalb wird dieses Thema in diesem Kapitel beschrieben.

In den nächsten Abschnitten werden wir sechs Aspekte im Zusammenhang mit Netzleistung behandeln:

- 1. Leistungsprobleme**
- 2. Messen der Netzleistung**
- 3. Hostdesign für schnellere Netzwerke**
- 4. Schnelle Segmentverarbeitung**
- 5. Header-Komprimierung**
- 6. Protokolle für „Long Fat Networks“**

Diese Aspekte betrachten die Netzleistung sowohl beim Host als auch über das Netz und für den Fall, dass Netze in Größe und Geschwindigkeit zunehmen.

6.6.1 Leistungsprobleme in Rechnernetzen

Einige Leistungsprobleme, z.B. Verstopfung, werden durch die vorübergehende Überlastung von Ressourcen verursacht. Kommt plötzlich mehr Verkehr an, als ein Router verarbeiten kann, baut sich ein Datenstau auf, und die Leistung wird beeinträchtigt. In diesem und dem vorherigen Kapitel wurde das Thema Überlastung ausführlich behandelt.

Die Leistung kann auch abfallen, wenn ein Ungleichgewicht der strukturellen Ressourcen vorliegt. Wird z.B. eine Gigabit-Übertragungsleitung an einen bescheidenen PC angeschlossen, ist der arme Host nicht in der Lage, die ankommenden Pakete schnell genug zu verarbeiten. Zwangsläufig gehen einige verloren. Diese Pakete werden irgendwann erneut übertragen, was die Übertragungsverzögerung erhöht, Bandbreite vergeudet und die Leistung allgemein reduziert.

Überlastungen können auch synchron ausgelöst werden. Wenn beispielsweise ein Segment einen falschen Parameter enthält (z.B. den Port, für den es bestimmt ist), schickt der Empfänger in vielen Fällen netterweise eine Fehlermeldung zurück. Betrachten wir nun, was passieren kann, wenn ein fehlerhaftes Segment an 1 000 Rechner gesendet wird. Jedes könnte eine Fehlermeldung zurücksenden. Der daraus resultierende **Broadcast-Sturm** kann das Netz in die Knie zwingen. UDP litt an diesem Problem, bis das ICMP-Protokoll dergestalt geändert wurde, dass Hosts nicht mehr auf Fehler in UDP-Segmenten antworteten, die an Broadcast-Adressen gesendet wurden. Vor allem drahtlose Netze müssen sorgfältig darauf achten, ungeprüfte Broadcast-Antworten zu vermeiden, da Broadcasting natürlich erfolgt und die drahtlose Bandbreite begrenzt ist.

Ein zweites Beispiel synchroner Überlastung ist die Situation nach einem Stromausfall. Kommt der Strom wieder zurück, starten alle Rechner gleichzeitig. Eine typische Reboot-Folge kann so aussehen, dass zuerst irgendein (DHCP-)Server abgefragt wird, um die korrekte Identität zu erfahren, dann wird irgendein Dateiserver abgefragt, um eine Kopie des Betriebssystems einzuholen. Machen das Hunderte von Rechnern in einem Datenzentrum gleichzeitig, bricht der Server wahrscheinlich unter der Last zusammen.

Sogar wenn keine synchronen Überlastungen vorliegen und genügend Ressourcen verfügbar sind, kann sich die Leistung aufgrund mangelhafter Systemabstimmung verschlechtern. Hat ein Rechner z.B. eine sehr hohe CPU-Leistung und viel Speicher, von dem aber nicht genügend Speicher als Puffer zugewiesen wurde, wird die Flusskontrolle den Empfang der Segmente verlangsamen und die Leistung herunterfahren. Dies stellte bei vielen TCP-Verbindungen ein Problem dar, da das Internet schneller geworden war, aber die Standardgröße des Flusskontrollfensters von 64 KB sich nicht geändert hatte.

Ein weiteres Einstellungsproblem ist das Setzen der Timer. Wenn ein Segment gesendet wird, wird normalerweise ein Timer gesetzt, um sich gegen den Verlust des Segments zu schützen. Ist der Timeout zu kurz, erfolgen unnötig viele Neuübertragungen, die die Leitung verstopfen. Ist der Timeout zu lang, treten nach dem Verlust eines Segments unnötige Übertragungsverzögerungen auf. Zu den weiteren einstellbaren Parametern zählen die Wartezeit für die Übertragung von Daten im Huckepackverfahren, bevor eine getrennte Bestätigung gesendet wird, und die Anzahl der Neuübertragungen vor einem Abbruch.

Ein weiteres Leistungsproblem ist Jitter, der vor allem bei in Echtzeitanwendungen wie Audio und Video zutage tritt. Im Schnitt über genügend Bandbreite zu verfügen, reicht nicht für eine gute Leistung. Kurze Übermittlungsverzögerungen sind ebenfalls erforderlich. Auf Dauer kurze Verzögerungen zu bieten, setzt eine sorgfältige Planung der Netzlast, Dienstgüteunterstützung auf den Vermittlungs- und Sicherungsschichten oder beides voraus.

6.6.2 Messung der Netzwerkleistung

Bei einer schlechten Netzleistung beschweren sich die Benutzer meist beim Betreiber des Netzes und fordern Abhilfe. Doch um die Leistung zu verbessern, müssen sich die Betreiber zuerst einen genauen Überblick über den Gesamtzustand verschaffen. Hierzu müssen sie Messungen durchführen. In diesem Abschnitt werden wir etwas

näher auf die Probleme bei Netzleistungsmessungen eingehen. Unsere Ausführungen basieren auf der bahnbrechenden Arbeit von Mogul (1993).

Messungen können auf vielerlei Arten und an vielen Stellen (im Protokollstapel und physikalisch) durchgeführt werden. Die einfachste Messung besteht darin, bei Beginn einer Aktivität einen Timer zu starten und zu messen, wie lange diese Aktivität dauert. Weiß man z.B., wie lange es dauert, bis ein Segment bestätigt wird, hat man einen wichtigen Messwert gewonnen. Andere Messungen werden mit Zählern durchgeführt, mit denen die Häufigkeit bestimmter Ereignisse erfasst wird (z.B. die Anzahl verlorener Segmente). Schließlich ist es auch von Interesse, die Menge von etwas zu erfassen, z.B. die Anzahl der Bytes, die innerhalb eines bestimmten Zeitintervalls verarbeitet wurden.

Beim Messen der Netzleistung und der verschiedenen Parameter lauern viele Fallstricke. Einige davon werden im Folgenden beschrieben. Jeder systematische Versuch, die Netzleistung zu messen, sollte diese Fallstricke sorgfältig umgehen.

Sicherstellen, dass die Samplegröße groß genug ist

Man misst nicht nur einmal die Zeit, die es dauert, ein Segment zu übertragen, sondern wiederholt die Messung etwa eine Million Mal, um einen Durchschnittswert zu erhalten. Einschalteffekte wie sie auftreten, wenn IEEE-802.16-Netzwerkkarten oder Kabelmodems nach einer Leerlaufzeit eine Bandbreitenreservierung vornehmen, können das erste Segment verlangsamen; außerdem sind Warteschlangen mit Übertragungsschwankungen verbunden. Ein großes Sample reduziert die Ungenauigkeit des gemessenen Mittelwerts und der Standardabweichung. Diese Ungenauigkeit kann mithilfe von statistischen Standardformeln berechnet werden.

Sicherstellen, dass die Samples repräsentativ sind

Im Idealfall wird die gesamte Folge von einer Million Messungen zu unterschiedlichen Tages- und Wochenzeiten wiederholt, um die Wirkung verschiedener Netzbedingungen auf die gemessene Menge zu ermitteln. Überlastungsmessungen sind z.B. kaum nützlich, wenn sie zu einem Zeitpunkt durchgeführt werden, wo keine Überlastung vorliegt. Zuweilen sind die Ergebnisse am Anfang nicht einleuchtend, z.B. wenn um 11 Uhr und 13 Uhr die Überlastung groß ist, aber um 12 Uhr mittags keine Überlastungen zu verzeichnen sind (weil viele Benutzer zum Mittagessen gehen).

Bei drahtlosen Netzen spielt der Ort der Messung aufgrund der Signalausbreitung eine wichtige Rolle. Ein Messknoten, der nahe einem drahtlosen Client platziert wurde, muss nicht unbedingt die gleichen Pakete beobachten wie der Client, da sich die Antennen unterscheiden. Am besten führen Sie die Messungen direkt beim drahtlosen Client aus, um zu sehen, was er sieht. Ersatzweise können Sie mit bestimmten Techniken die drahtlosen Messungen, die an verschiedenen Punkten durchgeführt werden, kombinieren, um ein vollständigeres Bild von den Übertragungsabläufen zu erhalten (Mahajan et al., 2006).

Zwischenspeicherung kann sich verheerend auf Messergebnisse auswirken

Sie erhalten bei einer häufig wiederholten Messung eine erstaunlich schnelle Antwort, wenn das Protokoll Mechanismen zur Zwischenspeicherung (*caching*) aktiviert hat. Das Abrufen einer Webseite oder das Nachschlagen eines DNS-Namens (um die IP-Adresse zu erhalten) ist nur beim ersten Mal mit einem Datenaustausch über das Netz verbunden. Anschließend kann die Antwort aus einem lokalen Zwischenspeicher (Cache) zurückgeliefert werden, ohne dass irgendwelche Pakete über das Netz gesendet werden müssen. Die Ergebnisse solcher Messungen sind mehr oder weniger wertlos (es sei denn, Sie wollen die Leistung des Zwischenspeichers messen).

Einen ähnlichen Effekt kann auch das Puffern von Daten haben. TCP/IP-Leistungsmessungen sind bekannt dafür, dass sie UDP gelegentlich eine Leistung bescheinigen, die wesentlich höher ist, als das Netz zulässt. Wie ist das möglich? Ein Aufruf von UDP gibt normalerweise die Steuerung zurück, sobald die Nachricht vom Betriebssystemkern angenommen und in die Übertragungswarteschlange gestellt wurde. Ist genügend Pufferplatz vorhanden, bedeuten 1 000 UDP-Aufrufe nicht, dass alle Daten gesendet wurden. Die meisten sind immer noch im Betriebssystemkern, aber das Leistungstestprogramm glaubt, dass alle übertragen wurden.

Seien Sie also vorsichtig und achten Sie darauf, dass Sie genau verstehen, wie die Daten als Teil einer Netzoperation im Cache zwischengespeichert oder im Puffer abgelegt werden.

Sicherstellen, dass nichts Unerwartetes während der Tests passiert

Wenn Sie Ihre Messungen ausführen, während ein Benutzer über das Netz eine Videokonferenz abhält, erhalten Sie oft ganz andere Ergebnisse, als wenn es keine Videokonferenz gegeben hätte. Am besten führen Sie die Tests auf einem Netz im Leerlauf durch und erzeugen die ganze Arbeitslast selbst. Doch sogar dieser Ansatz weist Haken auf. Wenn man glaubt, dass morgens um 3 Uhr niemand das Netz nutzt, fängt vielleicht genau zu dieser Zeit ein automatisches Backup-Programm an, alle Platten auf Band zu kopieren. Oder es kommt zu starkem Verkehr, weil Ihre wunderbaren Webseiten aus anderen Zeitzonen abgerufen werden.

Drahtlose Netze stellen in dieser Hinsicht eine besondere Herausforderung dar, da es oft nicht möglich ist, sie von allen Störquellen fern zu halten. Sogar wenn es keine anderen drahtlosen Netze gibt, die in der Nähe Verkehr verursachen, kann es sein, dass jemand Popcorn in der Mikrowelle macht und aus Versehen eine Störung verursacht, die die Leistung des IEEE 802.11 reduziert. Aus diesen Gründen ist es zu empfehlen, die gesamte Netzwerktätigkeit zu überwachen, sodass Sie zumindest erkennen können, wenn etwas Unerwartetes passiert.

Vorsicht bei Verwendung einer unpräzisen Uhr

Computeruhren basieren darauf, dass ein Zähler in regelmäßigen Intervallen erhöht wird. Ein Millisekunden-Timer erhöht den Zähler z.B. jede Millisekunde um 1. Mit einem solchen Timer ein Ereignis zu messen, das weniger als eine Millisekunde dau-

ert, ist möglich, erfordert aber ein sorgfältiges Vorgehen. Einige Rechner verfügen natürlich über genauere Uhren, aber dann gibt es auch immer wieder kürzere Ereignisse zu messen. Beachten Sie, dass die Uhren nicht immer so genau sind, wie die Präzision, mit der die Zeit nach dem Ablesen der Uhr zurückgeliefert wird.

Um beispielsweise zu messen, wie lange der Aufbau einer TCP-Verbindung dauert, sollte die Uhr (z.B. in Millisekunden) abgelesen werden, wenn in den Code der Transportschicht eingetreten wird, und erneut, wenn er verlassen wird. Beträgt die wahre Zeit für den Verbindungsaufbau 300 μ s, dann ist der Unterschied zwischen den zwei Ablesungen entweder 0 oder 1. Beide Ergebnisse sind falsch. Wird die Messung dann eine Million Mal wiederholt, ist nach Addition aller Messwerte und Division durch eine Million, die gemittelte Durchschnittszeit bis auf 1 μ s genau.

Vorsicht beim Extrapolieren der Ergebnisse

Nehmen wir an, Sie führen Messungen mit simulierten Netzlasten von 0 (Leerlaufzustand) bis 0,4 (40 % der Kapazität) durch. Zum Beispiel könnte die Antwortzeit, um ein VoIP-Paket über ein IEEE-802.11-Netz zu schicken, durch die Datenpunkte und die sie verbindende durchgezogene Linie in ▶ Abbildung 6.49 ausgedrückt werden. Man mag versucht sein, linear zu extrapolieren, wie durch die gepunktete Linie angezeigt. Viele Warteschlangenergebnisse beinhalten aber einen Faktor von $1/(1-\rho)$ wobei ρ die Last ist, sodass die tatsächlichen Werte bei zunehmender Last eher der gestrichelten Linie ähneln, die viel steiler ansteigt als die lineare. Das heißt, seien Sie vorsichtig bei Konkurrenzeffekten, die sich bei großer Last viel stärker auswirken.

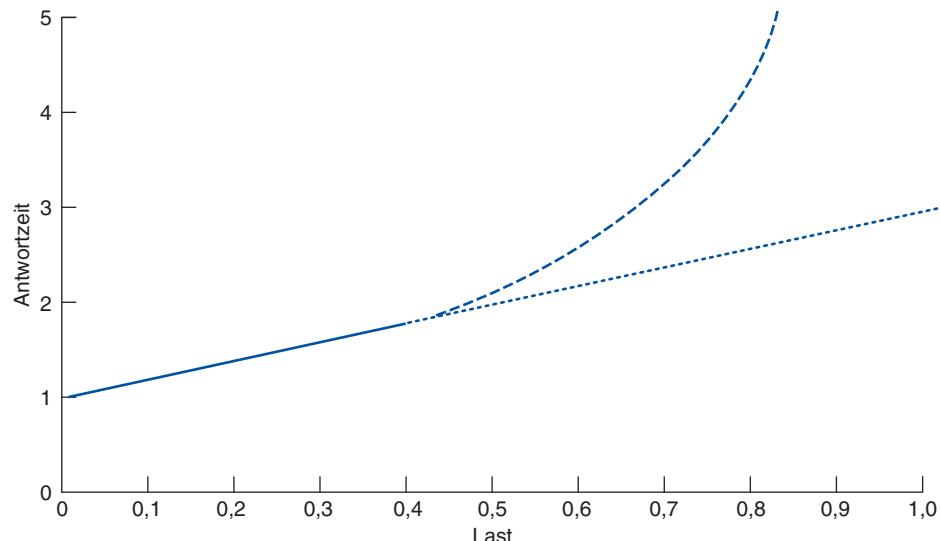


Abbildung 6.49: Antwortzeit als Funktion der Last.

6.6.3 Hostdesign für eine Optimierung der Leistung

Messungen und entsprechende Nachbesserungen können die Leistung meist beträchtlich erhöhen, sind aber kein Ersatz für guten Entwurf. Ein schlecht ausgelegtes Netz kann nur bis zu einem gewissen Punkt verbessert werden. Darüber hinaus ist ein ganz neuer Entwurf gefordert.

In diesem Abschnitt werden einige Faustregeln zur Implementierung von Netzprotokollen auf Hosts vorgestellt. Erstaunlicherweise haben sich diese aus zwei Gründen oft als Leistungsengpässe auf ansonsten schnellen Netzwerken entpuppt. Zum einen sind Netzwerkarten und Router bereits so eingerichtet (mit Hardwareunterstützung), dass sie mit „Kabelgeschwindigkeit“ laufen. Das bedeutet, dass sie Pakete so schnell verarbeiten können, wie diese über die Verbindung zugesendet werden. Zweitens, die wichtige Leistung ist die von den Anwendungen erzielte. Es ist nicht die Verbindungs Kapazität, sondern der Durchsatz und die Verzögerung nach der Netz- und Transportverarbeitung.

Die Reduzierung des Software-Overheads verbessert die Leistung durch Erhöhen des Durchsatzes und Verringern der Verzögerung. Sie können damit auch den Energieaufwand für den Netzwerkbetrieb reduzieren, was eine wichtige Überlegung für mobile Computer ist. Die meisten der hier vorgestellten Konzepte gehören seit Jahren zum Standardwissen von Netzentwicklern. Sie wurden erstmals explizit von Mogul (1993) beschrieben. Auf dieser Arbeit basieren die folgenden Ausführungen. Eine weitere gute Quelle ist Metcalfe (1993).

Hostgeschwindigkeit ist wichtiger als Netzgeschwindigkeit

Langjährige Erfahrungen haben gezeigt, dass Betriebssystem- und Protokoll-Overhead bei fast allen Netzen die tatsächliche Übertragungszeit bestimmen. Theoretisch beträgt die Mindestzeit für den entfernten Prozedurauftrag auf einem 1-Gbit/s-Ethernet 1 µs, was einer Minimumanfrage (512 Byte), gefolgt von einer Minimumantwort (512 Byte) entspricht. In der Praxis ist es ein enormer Fortschritt, wenn der Software-Overhead so weit eingeschränkt werden kann, dass die Zeit für entfernte Prozeduraufrufe dem obigen Wert auch nur nahe kommt. Leider ist dies nur sehr selten der Fall.

Das größte Problem im Betrieb bei 1 Gbit/s ist, die Bits schnell genug vom Puffer des Benutzers in das Netz zu bekommen und vom empfangenden Host sofort verarbeiten zu lassen. Kurz: Wird die Hostgeschwindigkeit (CPU und Speicher) verdoppelt, nähert man sich oft einer Verdoppelung des Durchsatzes. Wird die Netzkapazität verdoppelt, erreicht man meist keine Wirkung, weil der Engpass vorwiegend bei den Hosts zu suchen ist.

Overhead durch geringere Paketzahl reduzieren

Jedes Segment hat einen gewissen Overhead (z.B. der Header) sowie eine bestimmte Menge Daten (z.B. die Nutzlast). Beide Komponenten benötigen und belegen Bandbreite. Außerdem müssen beide Komponenten verarbeitet werden (z.B. Header-Verarbeitung und Berechnung der Prüfsumme). Wenn 1 Million Byte gesendet werden, bleiben die Datenkosten immer die gleichen, egal wie groß die Segmentgröße ist. Die

Verwendung von 128-Byte-Segmenten anstelle von 4-KB-Segmenten bedeutet jedoch 32-mal mehr Overhead pro Segment. Der Overhead für Bandbreite und Verarbeitung summiert sich schnell, um den Durchsatz zu reduzieren.

Der Overhead pro Paket in den unteren Schichten verstärkt diesen Effekt. Jedes ankommende Paket verursacht einen neuen Interrupt, wenn der Host weitermacht. Bei einem modernen Pipeline-Prozessor unterbricht jeder Interrupt die CPU-Pipeline, stört den Cache, erfordert eine Änderung bei der Speicherverwaltung, leert die Sprungvorhersagetabelle und erzwingt das Sichern einer beträchtlichen Anzahl von CPU-Registern. Eine n -fache Reduzierung der gesendeten Segmente reduziert somit den Interrupt- und den Paket-Overhead um den Faktor n .

Man könnte sagen, dass weder der Mensch noch der Computer besonders multitasking-fähig wären. Diese Beobachtung entspringt dem Wunsch, MTU-Pakete zu senden, die so groß sind, wie maximal über den Netzpfad ohne weitere Fragmentierung übertragen werden können. Nagles Algorithmus und Clarks Lösung sind ebenfalls Versuche, das Senden kleiner Pakete zu vermeiden.

Kopieren der Daten minimieren

Der einfachste Weg, einen aus mehreren Schichten bestehenden Protokollstapel zu implementieren, besteht darin, für jede Schicht ein Modul anzulegen. Leider erfordert dies das Kopieren von oder zumindest das Zugreifen auf Daten, während sie die Schichten durchqueren, da jede Schicht ihre eigene Arbeit zu erledigen hat. Zum Beispiel wird ein Paket, nachdem es von der Netzwerkkarte empfangen wurde, normalerweise in einen Puffer im Betriebssystemkern kopiert. Von dort wird es in einen Puffer der Vermittlungsschicht (zur Verarbeitung in der Vermittlungsschicht), dann in einen Puffer der Transportschicht (zur Verarbeitung in der Transportschicht) und schließlich noch in den empfangenden Anwendungsprozess kopiert. Es ist nicht unüblich, dass ein ankommendes Paket drei- oder viermal kopiert wird, bevor das darin enthaltene Segment zugestellt wird.

Diese ganzen Kopiervorgänge können die Leistung erheblich beeinträchtigen, da Speicheroperationen um Größenordnungen langsamer sind als Register-Register-Anweisungen. Wenn 20 % der Anweisungen tatsächlich auf den Hauptspeicher zugreifen (d.h., die benötigten Daten liegen im Zwischenspeicher), was bei der Behandlung eingehender Pakete wahrscheinlich ist, wird die durchschnittliche Ausführungszeit einer Anweisung um den Faktor 2,8 verlangsamt ($0,8 \times 1 + 0,2 \times 10$). Hardwareseitig ist hier keine Hilfe zu erwarten. Das Problem liegt ausschließlich darin, dass zu viel vom Betriebssystem kopiert wird.

Ein cleveres Betriebssystem wird die Kopiervorgänge minimieren, indem es die Verarbeitung auf mehreren Schichten kombiniert. So werden TCP und IP normalerweise zusammen (als „TCP/IP“) implementiert, sodass es nicht nötig ist, die Nutzlast des Pakets zu kopieren, wenn die Verarbeitung von der Vermittlungs- zur Transportschicht wechselt. Ein anderer bekannter Trick ist, in einer Schicht mehrere Operationen gleichzeitig auf die Daten auszuführen. So werden beispielsweise Prüfsummen

oft berechnet, während die Daten kopiert werden (sofern sie kopiert werden müssen), und die neu berechnete Prüfsumme wird an das Ende angehängt.

Kontextwechsel minimieren

Eine verwandte Regel besagt, dass Kontextwechsel (z.B. vom Kern- in den Benutzermodus) tödlich sind. Sie haben die gleichen schlechten Eigenschaften wie Interrupts und Kopievorgänge zusammengenommen. Diese Kosten sind der Grund, warum Transportprotokolle oft im Betriebssystemkern implementiert werden. Wie beim Reduzieren der Paketzahl können Kontextwechsel dadurch reduziert werden, dass die Bibliotheksprozedur, die Daten sendet, diese intern puffert, bis sie eine ausreichende Menge gesammelt hat. Ebenso sollten auf der Empfangsseite kleine Segmente gesammelt und nicht einzeln, sondern in einem Schwung an den Benutzer übergeben werden, um übermäßige Kontextwechsel zu vermeiden.

Im besten Fall verursacht ein ankommendes Paket einen Kontextwechsel vom aktuellen Benutzer zum Kern und dann einen Wechsel zum Empfangsprozess, um die neu angekommenen Daten abzugeben. Leider spielen sich bei vielen Betriebssystemen zusätzliche Kontextwechsel ab. Wenn z.B. der Netzmanager als eigener Prozess im Benutzerraum läuft, könnte die Ankunft eines Pakets einen Kontextwechsel vom aktuellen Benutzer zum Kern, einen Kontextwechsel vom Kern zum Netzmanager, einen weiteren zum Kern zurück und schließlich einen vom Kern zum Empfangsprozess verursachen. Dieser Vorgang ist in ▶ Abbildung 6.50 dargestellt. Diese Kontextwechsel bei jedem Paket sind eine große Vergeudung von CPU-Zeit und haben auf die Netzeistung eine verheerende Wirkung.

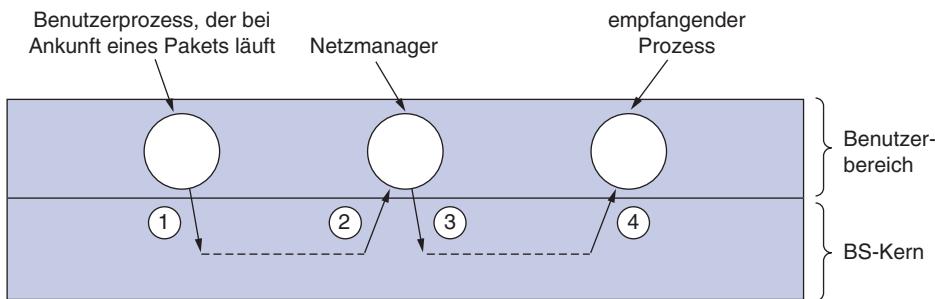


Abbildung 6.50: Vier Kontextwechsel, um ein Paket mit einem Netzmanager im Benutzerraum zu verarbeiten.

Überlastung vermeiden ist besser als beheben

Die alte Regel, dass Vorbeugen besser ist als Heilen, trifft auch auf Netzüberlastungen zu. Ist ein Netz überlastet, gehen Pakete verloren, Bandbreite wird verschwendet, sinnlose Übertragungsverzögerungen entstehen usw. All diese Kosten sind unnötig und die Wiederherstellung nach Überlastung erfordert Zeit und Geduld. Viel besser ist, wenn sie erst gar nicht auftreten. Das Vermeiden von Überlastungen ist wie eine Impfung: Es sticht ein bisschen, verhindert aber größere Schmerzen in der Zukunft.

Timeouts vermeiden

Timer sind in Netzen notwendig, sollten aber sparsam eingesetzt werden. Auch sollten Timeouts minimiert werden. Läuft ein Timer ab, wird normalerweise eine bestimmte Aktion wiederholt. Sofern die Wiederholung einer Aktion dringend erforderlich ist, muss es eben so sein, aber unnötige Wiederholungen belasten nur die Ressourcen.

Das lässt sich vermeiden, indem man die Timer etwas konservativer setzt. Ein Timer, der zu lange läuft, ist – im (unwahrscheinlichen) Falle eines verlorenen Segments – für eine Verbindung mit einer geringfügigen zusätzlichen Übertragungsverzögerung verbunden. Ein Timer hingegen, der zu früh abläuft, verbraucht wertvolle CPU-Zeit, verschwendet Bandbreite und bedeutet eine unnötige zusätzliche Last auf vielleicht Dutzenden von Routern.

6.6.4 Schnelle Segmentverarbeitung

Nachdem wir nun die allgemeinen Regeln abgehandelt haben, werden wir einige spezielle Methoden betrachten, die die Segmentverarbeitung beschleunigen. Weitere Informationen hierzu finden Sie in Clark et al. (1989) sowie Chase et al. (2001).

Der durch die Segmentverarbeitung entstehende Overhead setzt sich aus zwei Teilen zusammen: Overhead pro Segment und Overhead pro Byte. Beide müssen behandelt werden. Der Schlüssel zu einer schnellen Segmentverarbeitung besteht darin, den Normalfall (unidirektionale Datenübertragung) zu isolieren und speziell zu behandeln. Viele Protokolle heben vor allem das hervor, was sie im Problemfall machen (z.B. wenn ein Paket verloren geht). Aber um die Protokolle schneller zu machen, sollte der Entwickler darauf abzielen, die Verarbeitungszeit für den Normalfall zu minimieren. Die Verarbeitungszeit im Falle eines Fehlers zu verringern ist von zweitrangiger Bedeutung.

Obwohl eine Folge spezieller Segmenten erforderlich ist, um in den *ESTABLISHED*-Zustand zu gelangen, ist die Segmentverarbeitung dort ziemlich einfach, bis eine Seite beginnt, die Verbindung zu beenden. Betrachten wir zuerst die sendende Seite im *ESTABLISHED*-Zustand für den Fall, dass Daten übertragen werden müssen. Der Klarheit halber nehmen wir hier an, dass die Transportinstanz im Betriebssystemkern sitzt. Das gleiche Prinzip trifft aber auch zu, wenn es sich um einen Prozess im Benutzerraum oder eine Bibliothek im sendenden Prozess handelt. In ▶ Abbildung 6.51 geht der sendende Prozess in den Kern und führt SEND aus. Die Transportinstanz führt als Erstes einen Test durch, um zu sehen, ob es sich um den Normalfall handelt: Der Zustand ist *ESTABLISHED*, keine Seite versucht, die Verbindung abzubauen, ein volles Segment wird normal (d.h. nicht außerhalb der Reihenfolge) übertragen, und beim Empfänger steht ausreichend Fensterraum zur Verfügung. Wenn alle Bedingungen erfüllt sind, sind keine weiteren Tests mehr nötig und es kann der schnelle Pfad (*fast path*) durch die sendende Transportinstanz genommen werden. In der Regel wird dieser Pfad die meiste Zeit verwendet.

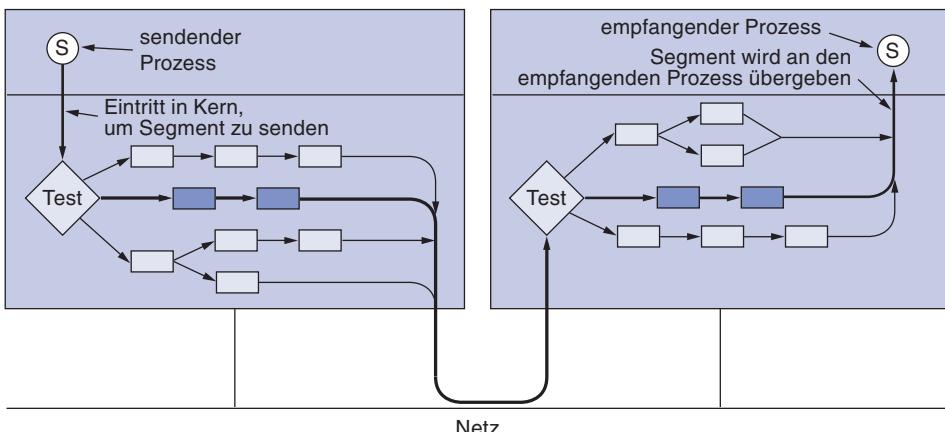


Abbildung 6.51: Der schnelle Pfad vom Sender zum Empfänger wird mit einer dicken Linie dargestellt. Die Verarbeitungsschritte entlang dieses Pfades sind dunkelblau dargestellt.

Im Normalfall sind die Header der aufeinanderfolgenden Datensegmente fast gleich. Um sich diese Tatsache zunutze zu machen, wird ein Prototyp-Header in der Transportinstanz gespeichert. Am Anfang des schnellen Pfades wird er so schnell wie möglich in einen Hilfspuffer kopiert, und zwar Wort für Wort. Die Felder, die sich von Segment zu Segment ändern, werden im Puffer überschrieben. Häufig können diese Felder leicht von Zustandsvariablen abgeleitet werden, z.B. aus der nächsten Sequenznummer. Anschließend werden ein Zeiger auf den vollen Segment-Header sowie einer auf die Benutzerdaten an die Vermittlungsschicht übergeben. Hier kann die gleiche Strategie verwendet werden (in Abbildung 6.51 nicht dargestellt). Schließlich gibt die Vermittlungsschicht das resultierende Paket zur Übertragung an die Sicherungsschicht weiter.

Als Beispiel, wie dieses Prinzip in der Praxis funktioniert, betrachten wir TCP/IP.
 ► Abbildung 6.52a zeigt den TCP-Header. Die Felder, die in den aufeinanderfolgenden Segmenten in einer Richtung gleich sind, sind dunkelblau dargestellt. Die sendende Transportinstanz braucht nur die fünf Wörter aus dem Prototyp-Header in den Ausgabepuffer zu kopieren, die nächste Sequenznummer einzutragen (durch Kopieren eines Wortes im Speicher), die Prüfsumme zu berechnen und die Sequenznummer im Speicher zu erhöhen. Dann können Header und Daten an eine spezielle IP-Prozedur zur Übertragung als normales maximales Segment abgegeben werden. IP kopiert seien aus fünf Wörtern bestehenden Prototyp-Header (siehe ► Abbildung 6.52b) in den Puffer, füllt das *Identification*-Feld aus und berechnet die Prüfsumme. Dann ist das Paket zur Übertragung bereit.

Betrachten wir nun die Verarbeitung des schnellen Pfades auf der Empfängerseite in Abbildung 6.51. Im ersten Schritt wird der Verbindungsdatensatz für das eingehende Segment ermittelt. Bei TCP kann der Verbindungsdatensatz in einer Hash-Tabelle gespeichert sein, für die eine einfache Funktion aus den beiden IP-Adressen und den beiden Ports den Schlüssel bildet. Wurde der Verbindungsdatensatz gefunden, müssen beide Adressen und beide Ports verglichen werden, um sicherzustellen, dass der herausgesuchte Datensatz der richtige ist.

| Source Port | | Destination Port | | VER. | IHL | Diff. Serv. | Total Length | | | |
|------------------------|--------|------------------|--|----------------|---------------------|-----------------|-----------------|--|--|--|
| Sequence Number | | | | Identification | | | Fragment Offset | | | |
| Acknowledgement Number | | | | TTL | Protocol | Header checksum | | | | |
| Len | Unused | | | Window Size | | | Source Address | | | |
| Checksum | | Urgent Pointer | | | Destination Address | | | | | |

Abbildung 6.52: (a) TCP-Header. (b) IP-Header. In beiden Fällen werden sie unverändert dem Prototyp entnommen.

Zur Optimierung können Sie einen Zeiger verwenden, der auf den zuletzt verwendeten Datensatz zeigt und diesen als ersten prüft. Dies beschleunigt oft die Suche nach Verbindungsdatensätzen. Clark et al. (1989) haben dies getestet und eine Trefferrate von mehr als 90 % erzielt.

Dann wird das Segment geprüft, um festzustellen, ob es normal ist: Der Zustand ist *ESTABLISHED*, keine Seite versucht, die Verbindung abzubauen, das Segment ist vollständig, es sind keine speziellen Flags gesetzt, und die Sequenznummer stimmt auch. Diese Tests erfordern nur ein paar Anweisungen. Sind alle Bedingungen erfüllt, wird eine spezielle TCP-Prozedur für den schnellen Pfad aufgerufen.

Der schnelle Pfad aktualisiert den Verbindungsdatensatz und kopiert die Daten zum Benutzer. Während des Kopiervorgangs wird auch die Prüfsumme berechnet, sodass ein Durchgang eingespart wird. Ist die Prüfsumme richtig, wird der Verbindungsdatensatz aktualisiert und eine Bestätigung zurückgeschickt. Die allgemeine Vorgehensweise, zuerst eine schnelle Prüfung durchzuführen, ob der Header den Erwartungen entspricht, und dann eine spezielle Prozedur für diesen Fall aufzurufen, nennt man **Header-Prediction** (Header-Prädiktion). Viele TCP-Implementierungen wenden diese Methode an. Mit dieser und allen anderen in diesem Kapitel beschriebenen Optimierungen kann TCP mit 90 % der Geschwindigkeit einer lokalen Speicher-zu-Speicher-Kopie laufen – unter der Voraussetzung, dass das Netz selbst schnell genug ist.

Puffer- und Timer-Management sind zwei weitere Bereiche, in denen sich hohe Leistungsverbesserungen erzielen lassen. Beim Puffermanagement liegt die Aufgabe darin, unnötiges Kopieren zu vermeiden. Timer-Management ist wichtig, weil fast alle gesetzten Timer nicht ablaufen. Sie werden gesetzt, um sich vor Segmentverlust zu schützen. Die meisten Segmente und ihre Bestätigungen kommen aber korrekt an. Folglich ist es wichtig, das Timer-Management für die Situation zu optimieren, dass der Timer nur selten abläuft.

Eine übliche Methode ist die Verwendung einer verketteten Liste von Timer-Ereignissen, die nach Ablaufzeit sortiert ist. Der Kopfeintrag enthält einen Zähler, der die Takte bis zum Timerablauf erfasst. Jeder nachfolgende Eintrag enthält einen Zähler, der die Zählimpulse nach dem vorherigen Eintrag erfasst. Laufen Timer nach 3, 10 bzw. 12 Impulse ab, dann sind die drei Zähler 3, 7 bzw. 2.

Bei jedem Impuls wird der Zähler im Kopfeintrag um 1 reduziert. Wird er null, so wird sein Ereignis verarbeitet, und das nächste Element auf der Liste ist dann der Kopfeintrag. Sein Zähler muss nicht geändert werden. Bei dieser Methode ist das Einfügen und Löschen von Timern teuer, mit Ausführungszeiten, die proportional zur Länge der Liste sind.

Ein effizienterer Ansatz kann verwendet werden, wenn das maximale Timerintervall begrenzt und im Voraus bekannt ist. Hier kann ein Array namens **Timing-Wheel** (Zeitrad) verwendet werden (►Abbildung 6.53). Jeder Slot entspricht einem Takt. Die aktuelle Zeit ist $T=4$. Es sind Timer gesetzt, die ab dem aktuellen Zeitpunkt nach 3, 10 und 12 Zählimpulsen ablaufen. Wenn ein neuer Timer plötzlich auf den Ablauf nach sieben Impulsen gesetzt wird, so wird in Slot 11 ein Eintrag vorgenommen. Ebenso muss, wenn der Timer für $T+10$ gestrichen werden soll, die Liste, auf die Slot 14 zeigt, durchsucht und der betreffende Eintrag entfernt werden. Man beachte, dass das Array in Abbildung 6.53 keine Timer über $T+15$ unterstützen kann.

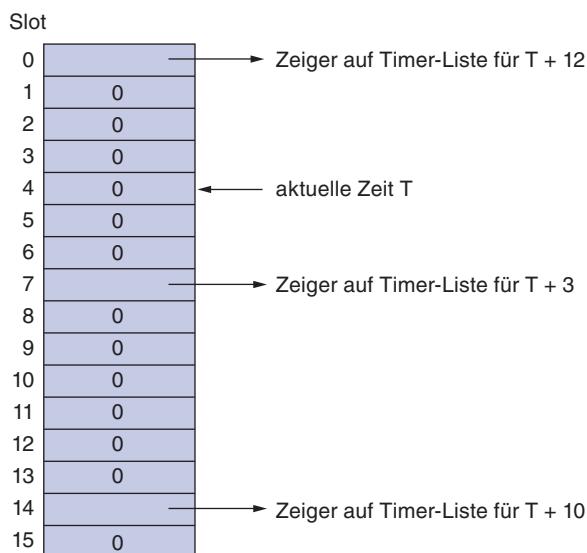


Abbildung 6.53: Ein Timing-Wheel.

Tickt die Uhr, so bewegt sich der aktuelle Zeitzeiger um ein Feld (im Kreis) vorwärts. Ist der Eintrag, auf den der Zeiger jetzt verweist, nicht null, werden alle dazugehörigen Timer abgearbeitet. Viele Varianten dieses Grundkonzepts werden von Varghese und Lauck (1987) beschrieben.

6.6.5 Header-Komprimierung

Eigentlich haben wir uns viel zu lange mit den schnellen Netzen beschäftigt. Es gibt auch noch andere Themen. Wenden wir uns also der Leistung auf drahtlosen und anderen Netzen zu, deren Bandbreite beschränkt ist. Die Reduzierung des Software-Overheads kann Ihre mobilen Rechner effizienter machen, trägt aber nicht zur Verbesserung der Leistung bei, wenn die Netzverbindungen der Engpass sind.

Um die Bandbreite gut auszunutzen, sollten Protokoll-Header und Nutzdaten mit einem Minimum an Bits übertragen werden. Für Nutzdaten bedeutet dies, dass sie eine komprimierte Codierung aufweisen sollten; Bilder könnten beispielsweise als JPEG und nicht als Bitmap formatiert sein, oder Dokumente haben ein Format wie PDF, das bereits komprimiert ist. Es bedeutet aber auch, auf Anwendungsebene Mechanismen zur Zwischenspeicherung zu aktivieren, wie Web-Caches, die die Übertragungen von vornherein reduzieren.

Und wie steht es mit den Protokoll-Headern? Auf der Sicherungsschicht sind die Header für drahtlose Netze normalerweise kompakt, da sie in Erwartung von Bandbreitenknappheit entwickelt wurden. Die Header in IEEE 802.16 haben zum Beispiel kurze Verbindungsidentifikatoren anstelle von längeren Adressen. Die Protokolle der höheren Schichten, wie IP, TCP und UDP, haben jedoch nur eine Version für alle Sicherungsschichten, und zwar ohne kompakten Header. Genau genommen führt eine verschlankte Verarbeitung, die der Verringerung des Software-Overheads dient, oft zu Headern, die nicht so kompakt sind, wie sie eigentlich sein könnten (z.B. hat IPv6 einen umfangreicheren Header als IPv4).

Die Header der höheren Schichten können beträchtlich zum Leistungsverlust beitragen. Betrachten wir beispielsweise VoIP-Daten, die mit einer Kombination von IP, UDP und RTP übertragen werden. Diese Protokolle belegen 40 Byte für die Header (20 für IPv4, 8 für UDP und 12 für RTP). Bei IPv6 ist die Situation sogar noch schlimmer: 60 Byte, einschließlich des 40-Byte-Headers für IPv6. Am Ende machen die Header den Großteil der übertragenen Daten aus und belegen mehr als die Hälfte der Bandbreite.

Mit **Header-Komprimierung** lässt sich die Bandbreite reduzieren, die von den Headern der höheren Schichten auf den einzelnen Verbindungen belegt wird. Anstelle von allgemeinen Methoden werden speziell entwickelte Verfahren verwendet. Dies liegt daran, dass Header kurz sind und sich einzeln nur schlecht komprimieren lassen. Außerdem setzt die Dekomprimierung voraus, dass alle vorherigen Daten empfangen wurden. Dies ist nicht der Fall, wenn ein Paket verloren gegangen ist.

Sie können mit der Header-Komprimierung große Einsparungen erzielen, wenn Sie das Protokollformat kennen. Eine der ersten Methoden wurde von Van Jacobson (1990) entwickelt, um TCP/IP-Header auf langsamem seriellen Verbindungen zu komprimieren. Sie können damit einen normalen TCP/IP-Header von 40 Byte auf durchschnittlich 3 Byte komprimieren. Der Trick dieser Methode ist Abbildung 6.52 zu entnehmen. Viele der Header-Felder bleiben von Paket zu Paket unverändert. Es ist zum Beispiel nicht nötig, in jedem Paket den gleichen IP-TTL (*Time to Live*) oder die gleiche TCP-Portnummer zu senden. Sie können diese Daten sendeseitig weglassen und empfangsseitig wieder ergänzen.

Andere Felder wiederum ändern sich in vorhersagbarer Art und Weise. Sieht man von dem Fall eines Verlustes ab, erhöht sich beispielsweise die TCP-Sequenznummer mit den Daten. In diesen Fällen kann der Empfänger den wahrscheinlichen Wert vorhersagen. Die eigentliche Nummer muss nur übertragen werden, wenn sie von der erwarteten Nummer abweicht. Doch auch dann kann sie als kleine Änderung vom vorherigen

Wert übertragen werden. Eigentlich genau wie bei der Bestätigungsnummer, die sich erhöht, wenn neue Daten in der umgekehrten Richtung empfangen werden.

Dank der Header-Komprimierung ist es möglich, in Protokollen der höheren Schichten einfache Header zu verwenden und auf Verbindungen mit geringer Bandbreite kompakte Codierungen einzusetzen. Die **robuste Header-Komprimierung** (*ROHC, RObust Header Compression*) ist eine moderne Version der Header-Komprimierung, die in RFC 5795 als Rahmenplan definiert ist. Sie wurde für drahtlose Verbindungen entwickelt, die sich durch häufige Verluste auszeichnen. Es gibt für jeden zu komprimierenden Protokollsatz (beispielsweise IP/UDP/RTP) ein Profil. Komprimierte Header werden übertragen, indem ein Bezug zu einem Kontext hergestellt wird, der im Wesentlichen eine Verbindung ist; Header-Felder können für Pakete der gleichen Verbindung leicht vorhergesagt werden, aber nicht für Pakete verschiedener Verbindungen. Normalerweise lässt sich mit der robusten Header-Komprimierung der Umfang von IP/UDP/RTP-Headern von 40 Byte auf 1–3 Byte reduzieren.

Auch wenn Header-Komprimierung primär darauf abzielt, den Bedarf an Bandbreite zu verringern, kann sie auch für die Reduzierung der Verzögerung ganz nützlich sein. Die gesamte Verzögerung setzt sich zusammen aus der Ausbreitungsverzögerung (*propagation delay*), die bei einem gegebenen Netzpfad fest ist, und der Übermittlungsverzögerung (*transmission delay*), die von der Bandbreite und der Menge der zu sendenden Daten abhängt. So sendet beispielsweise eine 1-Mbit/s-Verbindung 1 Bit in 1 µs. Im Falle von Mediendaten über drahtlose Netze ist das Netz relativ langsam, sodass die Übermittlungsverzögerung ein wichtiger Faktor in der Gesamtverzögerung sein kann. Außerdem ist eine ständig niedrige Verzögerung wichtig für die Dienstgüte.

Header-Komprimierung kann Abhilfe schaffen, indem sie die Menge der gesendeten Daten reduziert und damit die Übermittlungsverzögerung verringert. Der gleiche Effekt kann durch Senden kleinerer Pakete erzielt werden. In diesem Fall finden Sie sich mit einer erhöhten Software-Overhead ab zugunsten einer verringerten Übermittlungsverzögerung. Beachten Sie, dass es eine weitere potenzielle Verzögerungsquelle gibt, die sogenannte Warteschlangenverzögerung beim Zugriff auf die drahtlose Verbindung. Diese kann ebenfalls sehr wichtig sein, da drahtlose Verbindungen als begrenzte Ressource in einem Netz oft stark benutzt werden. In diesem Fall muss die drahtlose Verbindung Dienstgütemechanismen aufweisen, die dafür sorgen, dass die Echtzeitpakete mit einer niedrigen Verzögerung übertragen werden. Header-Komprimierung allein reicht nicht aus.

6.6.6 Protokolle für Long Fat Networks

Seit den 1990er Jahren gibt es Gigabit-Netze, die Daten über lange Strecken übertragen. Aufgrund der Kombination von einem schnellen Netz (oder „fat pipe“) und einer langen Verzögerung werden diese Netze **Long Fat Networks** genannt. Als diese Netze aufkamen, bestand die erste Reaktion der Benutzer darin, die bestehenden Protokolle einzusetzen, aber es tauchten schnell diverse Probleme auf. In diesem Abschnitt werden einige der Probleme beim Erhöhen der Geschwindigkeit und der Übertragungszeit des Netzprotokolls beschrieben.

Ein Problem ist, dass viele Protokolle 32-Bit-Sequenznummern verwenden. In den Anfangszeiten des Internets waren die Leitungen zwischen den Routern in der Regel 56-kbit/s-Standleitungen. Wenn ein Host in voller Geschwindigkeit Daten über die Leitung übertrug, dauerte es über eine Woche, die Sequenznummern zu durchlaufen. Für die Entwickler von TCP war 2^{32} eine ziemlich gute Annäherung an die Unendlichkeit, da wenig Gefahr bestand, dass alte Pakete eine Woche nach ihrer Übertragung noch irgendwo unterwegs waren. Beim 10-Mbit/s-Ethernet betrug die Durchlaufzeit durch alle Sequenznummern 57 Minuten, also viel kürzer, aber immer noch praktikabel. Wenn bei einem 1-Gbit/s-Ethernet Daten in das Internet eingespeist werden, beträgt die Durchlaufzeit etwa 34 Sekunden. Diese liegt also erheblich unter der maximalen Lebensdauer eines Pakets im Internet von 120 Sekunden. So ist 2^{32} dann plötzlich keine so gute Annäherung an die Unendlichkeit mehr, da ein schneller Sender alle Sequenznummern zyklisch durchlaufen kann, während immer noch alte Pakete existieren.

Die Schwierigkeit liegt darin, dass viele Protokollentwickler davon ausgegangen sind, dass die Zeitspanne zum Durchlaufen der gesamten Sequenznummern die maximale Paketlebensdauer weit überschreitet. Folglich bestand keine Notwendigkeit, sich über das Problem Gedanken zu machen, dass es noch alte Duplikate gibt, wenn die Sequenznummern durchlaufen wurden und von vorn beginnen. Bei Gigabit-Geschwindigkeiten ist diese Annahme nicht mehr gültig. Zum Glück war es möglich, die effektive Sequenznummer zu erweitern, indem der Zeitstempel, der optional im TCP-Header eines jeden Pakets angegeben werden kann, als höchstwertige Bits behandelt wird. Dieser Mechanismus wird Schutz gegen wiederholte Sequenznummern (*Protection Against Wrapped Sequence Numbers*, PAWS) genannt und ist in RFC 1323 beschrieben.

Ein zweites Problem ist, dass das Flusskontrollfenster stark vergrößert werden muss. Stellen Sie sich vor, Sie senden 64 KB an Daten von San Diego nach Boston, um dort den 64-KB-Puffer des Empfängers zu füllen. Weiterhin angenommen, die Verbindung ist 1 Gbit/s schnell und weist in eine Richtung eine Ausbreitungsverzögerung von 20 ms (im Glasfaserkabel) auf. Zu Beginn, bei $t = 0$, ist die Pipe leer (► Abbildung 6.54a). Nur 500 μ s später befinden sich alle Segmente auf der Reise (► Abbildung 6.54b). Das erste Segment dürfte sich irgendwo in der Nähe von Brawley, immer noch im Staat Kalifornien, befinden. Der Sender muss jedoch unterbrechen, bis er eine Fensteraktualisierung erhält.

Nach 20 ms kommt das erste Segment in Boston an (► Abbildung 6.54c) und wird bestätigt. Schließlich, 40 ms nach Übertragungsbeginn, erreicht die erste Bestätigung wieder den Sender und der zweite Datenschub kann übertragen werden (► Abbildung 6.54d). Da die Übertragungsleitung nur für 1,25 ms von 100 ms genutzt wurde, beträgt die Effizienz ungefähr 1,25 %. Diese Situation ist typisch für ältere Protokolle, die auf Gigabit-Verbindungen ausgeführt werden.

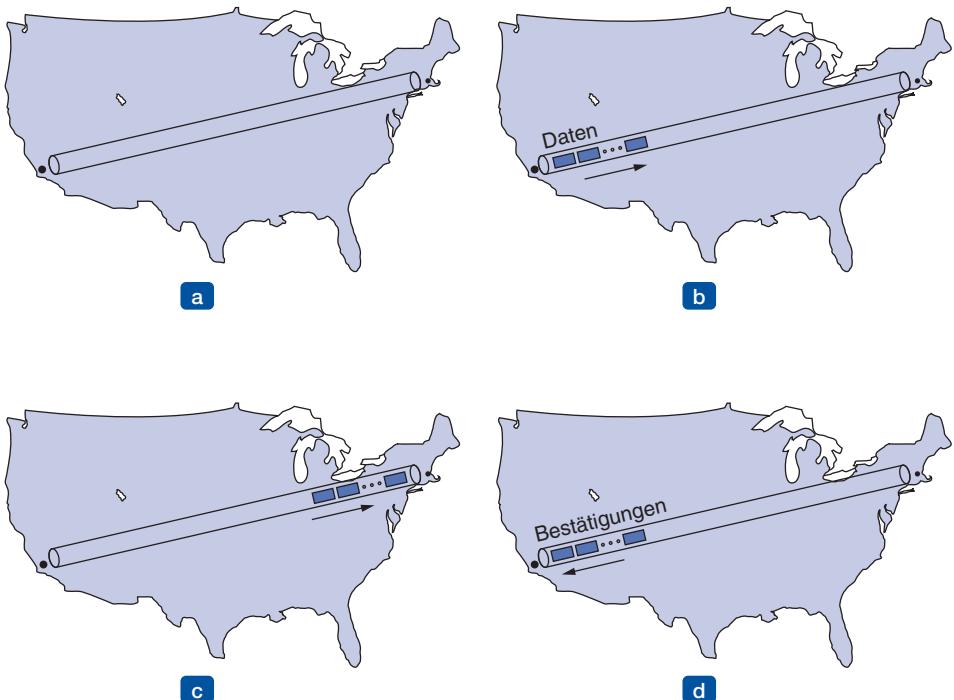


Abbildung 6.54: Der Zustand beim Übertragen von 1 MB von San Diego nach Boston. (a) Bei $t=0$. (b) Nach 500 μ s. (c) Nach 20 ms. (d) Nach 40 ms.

Ein weiterer wichtiger Faktor, den Sie bei der Analyse der Netzeleistung berücksichtigen sollten, ist das **Bandbreite-Verzögerung-Produkt**. Es berechnet sich durch die Multiplikation der Bandbreite (in Bit/s) mit der Paketumlaufzeit (in s). Das Produkt drückt die Kapazität (in Bit) der Leitung vom Sender zum Empfänger und wieder zurück aus.

Daraus kann man folgern, dass für eine gute Leistung das Fenster des Empfängers mindestens so groß sein muss wie das Bandbreite-Verzögerung-Produkt. Vorzugsweise sollte es etwas größer sein, da der Empfänger eventuell nicht sofort reagiert. Bei einer transkontinentalen Gigabit-Leitung sind mindestens 5 MB erforderlich.

Ein drittes und verwandtes Problem ist, dass einfache Neuübertragungsstrategien wie Go-back-N-Protokoll auf Leitungen mit einem großen Bandbreite-Verzögerung-Produkt schlechte Leistungen zeigen. Betrachten Sie z.B. eine transkontinentale 1-Gbit/s-Verbindung mit einer Übertragungszeit von 40 ms. In dieser Zeit kann ein Sender 5 MB übertragen. Wird ein Fehler erkannt, dauert es 40 ms, bis der Sender darüber informiert wird. Bei Verwendung von Go-back-N muss der Sender nicht nur das fehlerhafte Paket erneut übertragen, sondern auch die gesamten 5 MB an Paketen, die danach kamen. Das ist eine massive Verschwendug von Ressourcen. Es besteht also eindeutig ein Bedarf an komplexeren Protokollen, die beispielsweise eine selektive Wiederholung unterstützen.

Ein vierter Problem ist, dass sich Gigabit- von Megabit-Leitungen grundsätzlich dahingehend unterscheiden, dass die langen Gigabit-Leitungen durch die Übertragungsverzögerung und nicht durch die Bandbreite begrenzt werden. ►Abbildung 6.55 ist zu entnehmen, wie lange es dauert, eine 1-MB-Datei 4 000 km weit mit verschiedenen Übertragungsgeschwindigkeiten zu befördern. Bei Geschwindigkeiten von bis zu 1 Mbit/s wird die Übertragungszeit von der Übertragungsrate beherrscht, in der die Bits gesendet werden können. Bei 1 Gbit/s überwiegt die Übertragungsverzögerung von 40 ms die 1 ms, die nötig ist, um die Bits in die Leitung einzuspeisen. Weitere Erhöhungen der Bandbreite haben keine Wirkung.

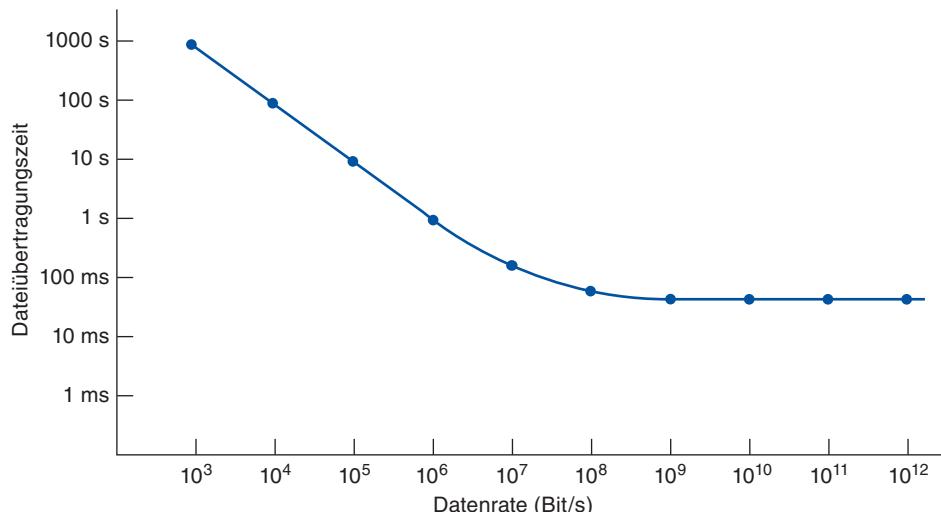


Abbildung 6.55: Benötigte Zeit zur Übertragung und Bestätigung einer 1-MB-Datei über eine 4 000 km lange Leitung.

Abbildung 6.55 hat unerfreuliche Folgen für Netzprotokolle, denn sie zeigt, dass der Leistung von Stop-and-Wait-Protokollen (z.B. entfernte Prozeduraufrufe) inhärent eine obere Grenze gesetzt ist. Diese Grenze wird von der Lichtgeschwindigkeit diktiert. Unabhängig von den technischen Fortschritten in der Optik wird sich hieran nichts ändern (obwohl neue physikalische Gesetze durchaus hilfreich wären). Solange Sie keine andere Verwendung für eine Gigabit-Leitung finden, während ein Host auf eine Antwort wartet, ist die Gigabit-Leitung nicht besser als eine Megabit-Leitung, nur wesentlich teurer.

Ein fünftes Problem ist, dass sich die Kommunikationsgeschwindigkeiten wesentlich stärker erhöht haben als die Rechengeschwindigkeiten. (Auf, Ihr Computerentwickler! Ihr werden Euch doch den Kommunikationsingenieuren nicht geschlagen geben! Wir rechnen mit Euch!) In den 1970er Jahren lief das ARPANET mit 56 kbit/s, wobei Rechner eingesetzt wurden, die eine Million Instruktionen pro Sekunde (1 MIPS) ausführten. Vergleichen Sie diese Zahlen mit den 1 000-MIPS-Computern, die Pakete über eine 1 Gbit/s-Leitung austauschen. Die Zahl der Anweisungen pro Byte hat um mehr als den Faktor 10 abgenommen. Die genauen Zahlen sind je nach Datum und Szenario diskutabel, lassen aber folgende Schlussfolgerung zu: Es steht weniger Zeit für die Protokollverarbeitung zur Verfügung als früher, sodass die Protokolle einfacher werden müssen.

Doch kommen wir nun von den Problemen zu den möglichen Lösungen. Das Grundprinzip, das alle Entwickler von Hochgeschwindigkeitsnetzen auswendig lernen sollten, lautet:

Richten Sie die Entwicklung auf die Geschwindigkeit aus und nicht auf die Optimierung der Bandbreite.

Alte Protokolle legten den Fokus oft auf die Minimierung der Anzahl von Bits in der Leitung. Meist wurde das durch Verwendung kleiner Felder und Verpacken derselben in Bytes und Wörtern realisiert. Diese Probleme bestehen für drahtlose Netze immer noch, aber nicht für Gigabit-Netze. Hier ist die Protokollverarbeitung das Problem; deshalb sollten Protokolle auf kürzere Verarbeitung ausgelegt werden. Die Entwickler von IPv6 haben dies klar verstanden.

Eine verführerische Möglichkeit, Schnelligkeit zu erreichen, ist die Entwicklung schneller Netzwerkkarten (Hardware). Abgesehen von extrem einfachen Protokollen ist die Schwierigkeit bei dieser Strategie, dass Hardware lediglich eine Einstekkarte mit einer zweiten CPU und einem eigenen Programm bedeutet. Um zu vermeiden, dass der Netz-Koprozessor so teuer kommt wie die eigentliche CPU, muss man sich meist mit einem langsameren Chip zufriedengeben. Dieser Entwurf hat zur Folge, dass viel Zeit der eigentlichen (schnellen) CPU mit Warten auf die zweite (langsame) CPU verstreicht. Die Vorstellung, dass die erste CPU indessen andere Arbeiten auszuführen hat, ist eine Illusion. Außerdem können, wenn zwei CPUs miteinander kommunizieren, Wettkampfbedingungen (*race condition*) auftreten. Deshalb sind ausfeilte Protokolle zwischen den beiden Prozessoren erforderlich, um sie richtig zu synchronisieren und Wettkämpfe zu vermeiden. Die beste Lösung ist meist ein einfaches Protokoll und die Ausführung der Hauptarbeit durch die eigentliche CPU.

In Gigabit-Netzen spielt das Paket-Layout eine wichtige Rolle. Der Header sollte möglichst wenige Felder enthalten, um die Verarbeitungszeit zu reduzieren. Außerdem sollten die Felder groß genug sein, um ihre Arbeit tun zu können, und auf Wortgrenzen beginnen, um leicht verarbeitet werden zu können. In diesem Zusammenhang bedeutet „groß genug“, dass Probleme wie das Wiederverwenden von Sequenznummern, während noch alte Pakete vorhanden sind, oder dass der Empfänger keinen ausreichend großen Fensterbereich ankündigen kann, weil das Fensterfeld zu klein ist, nicht auftreten.

Die maximale Datengröße sollte groß sein, um den Software-Overhead zu reduzieren und einen effizienten Betrieb zu erlauben. 1 500 Byte ist zu wenig für Hochgeschwindigkeitsnetze, was der Grund ist, warum Gigabit-Ethernet Jumborahmen von bis zu 9 KB und IPv6-Jumbopakete über 64 KB unterstützen.

Betrachten wir nun das Feedback bei Hochgeschwindigkeitsprotokollen. Aufgrund der (relativ) langen Verzögerungsschleife sollten Feedbacks vermieden werden. Es dauert einfach zu lange, bis der Empfänger den Sender benachrichtigt. Ein Beispiel eines Feedbacks ist die Steuerung der Übertragungsrate mithilfe eines Schiebefensterprotokolls. Zukünftige Protokolle werden vielleicht eher auf der Übertragungsrate basieren, um die (langen) Verzögerungen zu vermeiden, wenn der Empfänger Fensteraktualisierungen an den Sender schickt. Mit einem solchen Protokoll kann der Sender

alles übertragen, was er will, solange er nicht schneller sendet, als es zwischen den beiden im Voraus vereinbart wurde.

Ein zweites Beispiel eines Feedbacks ist der Slow-Start-Algorithmus von Jacobson. Dieser Algorithmus nimmt mehrere Messungen vor, um zu prüfen, wie viel das Netz bewältigen kann. Bei Hochgeschwindigkeitsnetzen wird durch ein halbes Dutzend kleiner Messläufe viel Bandbreite verschwendet. Effizienter ist, wenn sich Sender, Empfänger und Netz beim Verbindungsaufbau auf die Reservierung der nötigen Ressourcen einigen. Die Reservierung von Ressourcen im Voraus hat auch den Vorteil, dass Jitter einfacher reduziert werden kann. Kurz: Je höher die Geschwindigkeiten, desto mehr muss sich der Entwurf in Richtung verbindungsorientierter Betrieb oder etwas sehr Ähnlichem verlagern.

Ein weiteres wichtiges Merkmal ist die Fähigkeit, eine normale Datenmenge mit der Verbindungsanforderung zu senden. So wird bereits eine Hin- und Rückübertragung eingespart.

6.7 Verzögerungstolerante Netze

Wir beenden dieses Kapitel mit der Beschreibung einer neuen Art von Transport, die eines Tages eine wichtige Komponente des Internets werden könnte. TCP und die meisten anderen Transportprotokolle basieren auf der Annahme, dass Sender und Empfänger ständig durch irgendeinen Arbeitspfad verbunden sein müssen oder das Protokoll funktioniert nicht und die Daten können nicht zugestellt werden. In einigen Netzen gibt es oft keinen Ende-zu-Ende-Pfad. Ein Beispiel hierfür ist das Weltraumnetz, da LEO-Satelliten (*Low-Earth Orbit*) den Bereich der Bodenstationen immer wieder verlassen und betreten. Ein gegebener Satellit kann unter Umständen nur zu bestimmten Zeiten mit einer Bodenstation kommunizieren und zwei Satelliten kommunizieren eventuell nie miteinander, nicht einmal über eine Bodenstation, weil einer der Satelliten sich vielleicht immer außerhalb des Kommunikationsbereichs befindet. Andere Beispielnetze betreffen U-Boote, Busse, mobile Telefone und andere computergestützte Geräte, die aufgrund von Mobilität oder extremen Bedingungen eine unregelmäßige Konnektivität aufweisen.

In diesen gelegentlich verbundenen Netzen können Daten trotzdem übertragen werden, und zwar indem sie in Knoten gespeichert werden, bis irgendwann eine funktionierende Verbindung zur Verfügung steht und sie weitergeleitet werden können. Diese Technik wird **Nachrichtenvermittlung** (*message switching*) genannt. Am Ende werden die Daten ihr Ziel erreichen. Ein Netz, dessen Architektur auf diesem Ansatz basiert, heißt **verzögerungstolerantes Netz (DTN)**, *disruption-tolerant network* oder *delay-tolerant network*).

Die Arbeit an den verzögerungstoleranten Netzen begann 2002, als die IETF eine Forschungsgruppe zu diesem Thema einrichtete. Die Idee zu den DTNs kam aus einer ungewöhnlichen Ecke: den Bestrebungen, Pakete im Weltraum zu senden. Weltraumnetze müssen mit einer unregelmäßigen Kommunikation und sehr langen Verzögerungen fertig werden. Kevin Fall stellte fest, dass die Ideen für diese interplanetaren Internets sich

auch auf Netze auf der Erde übertragen ließen, bei denen eine unregelmäßige Konnektivität die Norm war (Fall, 2003). Dieses Modell ist eine nützliche Verallgemeinerung des Internets, in dem es während der Kommunikation zu Speicherungen und Verzögerungen kommen kann. Die Datenzustellung lässt sich eher mit der Zustellung von Post oder mit elektronischer Mail vergleichen als mit der Paketvermittlung über Router.

Seit 2002 wird die DTN-Architektur ständig verbessert und die Anwendungen des DTN-Modells werden immer größer. Betrachten wir als Standardanwendung große Datensätze von vielen Terabyte, die von wissenschaftlichen Experimenten, Medienereignissen oder webbasierten Diensten erzeugt werden und in Datenzentren an verschiedenen Orten weltweit kopiert werden müssen. Die Betreiber würden diese Riesendatenmenge am liebsten zu verkehrsschwachen Zeiten senden, um Bandbreite zu nutzen, die bereits bezahlt wurde, aber nicht verwendet wird, und sind deshalb bereit, einige Verzögerungen in Kauf zu nehmen. Es ist, als würde man Datensicherungen nachts laufen lassen, wenn die anderen Anwendungen das Netz nicht stark belasten. Das Problem bei globalen Diensten ist jedoch, dass verkehrsschwache Zeiten ortsbabhängig sind. Zum Beispiel dürfte es für Datenzentren in Boston und Perth kaum einen gemeinsamen Zeitraum für die Nutzung einer verkehrsarmen Netzbänderbreite geben, da es in einer Stadt Nacht ist, wenn in der anderen die Sonne scheint.

DTN-Modelle bieten jedoch den Vorteil, dass die Daten während der Übertragung gespeichert und verzögert werden können. Mit diesem Modell ist es möglich, die Datenmenge über eine verkehrsarme Bandbreite von Boston nach Amsterdam zu schicken, da die Zeitzonen dieser Städte nur 6 Stunden auseinanderliegen. Die Datenmenge wird dann in Amsterdam zwischengespeichert, bis es eine verkehrsarme Bandbreite zur günstigen Übertragung von Amsterdam nach Perth gibt. In diesem Zeitraum werden die Daten dann nach Perth geschickt, um die Übertragung zu beenden. Laoutaris et al. (2009) haben dieses Modell studiert und festgestellt, dass es erstaunliche Kapazität zu geringen Kosten bieten kann und dass seine Verwendung im Vergleich zu einem traditionellen Ende-zu-Ende-Modell die Kapazität oft verdoppelt.

Im Folgenden werden wir die DTN-Architektur und -Protokolle, so wie sie von der IETF entwickelt wurden, beschreiben.

6.7.1 DTN-Architektur

Bisher wurde im Internet primär davon ausgegangen, dass es für die ganze Dauer einer Kommunikationssitzung einen Ende-zu-Ende-Pfad zwischen einer Quelle und einem Ziel gibt. Diese Hauptannahme wollen die DTNs etwas lockern. Fehlt ein solcher Pfad, funktionieren die normalen Internetprotokolle nicht. DTNs umgehen die fehlende Ende-zu-Ende-Konnektivität mit einer Architektur, die auf Nachrichtenvermittlung basiert (► Abbildung 6.56). Es sollen dabei auch Verbindungen akzeptiert werden, die nur relativ unzuverlässig sind und große Verzögerungen aufweisen. Die Architektur wird in RFC 4838 angegeben.

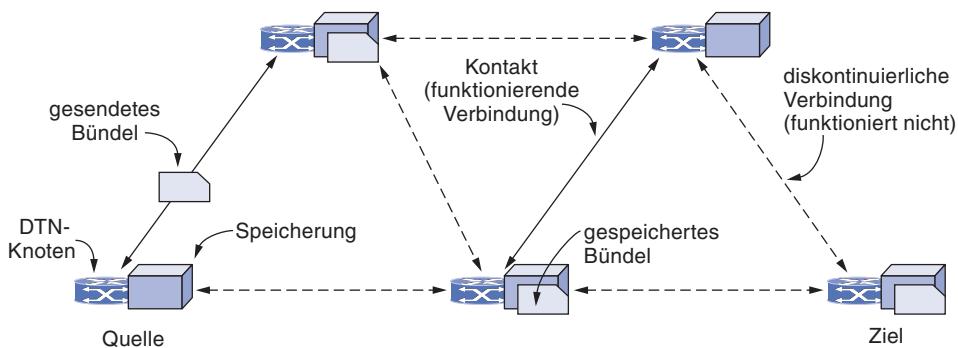


Abbildung 6.56: Architektur der verzögerungstoleranten Netze.

In der DTN-Terminologie wird eine Nachricht **Bündel** (*bundle*) genannt. DTN-Knoten verfügen über eine Speichermöglichkeit, normalerweise in Form von permanenten Speichern wie Platten- oder Flash-Speicher. Sie speichern die Bündel, bis Verbindungen zur Verfügung stehen, und leiten dann die Bündel weiter. Die Verbindungen arbeiten diskontinuierlich. Abbildung 6.56 zeigt fünf diskontinuierliche Verbindungen, die zurzeit nicht arbeiten, und zwei Verbindungen, die genutzt werden. Eine funktionierende Verbindung wird als **Kontakt** bezeichnet. Abbildung 6.56 zeigt außerdem Bündel, die bei zwei DTN-Knoten gespeichert sind, die auf Kontakte warten, um die Bündel weiterzuleiten. Auf diese Weise werden die Bündel über Kontakte von der Quelle zum Ziel weitergeleitet.

Das Speichern und Weiterleiten der Bündel bei DTN-Knoten klingt ähnlich wie das Anstauen und Weiterleiten von Paketen bei Routern, weist aber qualitative Unterschiede auf. In den Internetroutern dauert das Anstauen Millisekunden oder maximal Sekunden. Bei den DTN-Knoten können die Bündel stundenlang gespeichert sein – bis ein Bus in der Stadt ankommt, während ein Flugzeug seinen Flug beendet, bis ein Sensorknoten ausreichend Solarenergie getankt hat, bis ein schlafender Computer aufwacht und so weiter. Diese Beispiele veranschaulichen einen zweiten Unterschied, nämlich dass Knoten ihren Standort verändern können (mit Bus oder Flugzeug), während sie die gespeicherten Daten halten. Diese Bewegung kann sogar bei der Datenzustellung eingeplant sein. Router hingegen dürfen sich im Internet nicht bewegen. Der ganze Prozess der Übertragung von Bündeln ist vielleicht besser als „Store-Carry-Forward“-Strategie bekannt.

Als Beispiel betrachten wir das Szenario in ►Abbildung 6.57, das die erste Verwendung von DTN-Protokollen im Weltraum veranschaulicht (Wood et al., 2008). Die Quelle der Bündel ist ein LEO-Satellit, der einem Verbund von Erdbeobachtungssatelliten (*Disaster Monitoring Constellation, DMC*) angehört und Bilder von der Erde aufzeichnet. Die Bilder müssen an den Sammelpunkt geschickt werden. Der Satellit hat jedoch nur unregelmäßig Kontakt mit den drei Bodenstationen, da er die Erde umkreist. Er nimmt nacheinander Kontakt mit den einzelnen Bodenstationen auf. Jeder Satellit, jede Bodenstation und der Sammelpunkt agieren als DTN-Knoten. Bei jedem Kontakt wird ein Bündel (oder ein Teil eines Bündels) an eine Bodenstation

geschickt. Anschließend werden die Bündel über ein terrestrisches Backhaul-Netz an den Sammelpunkt gesendet, um die Übertragung abzuschließen.

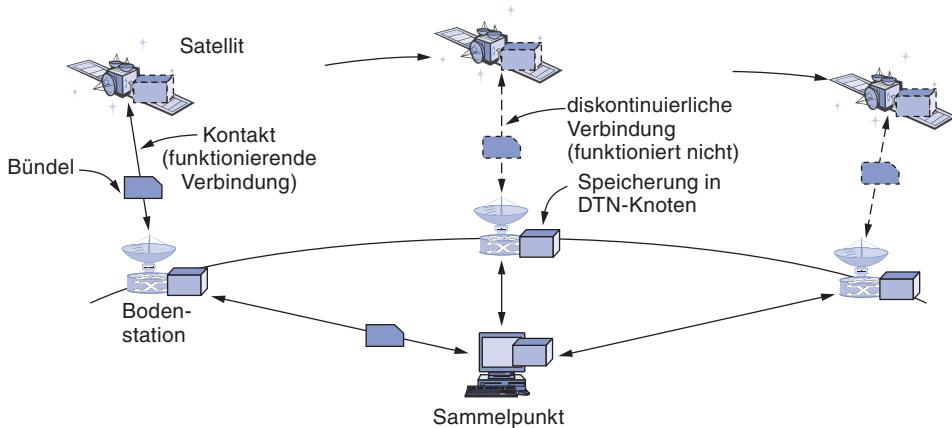


Abbildung 6.57: Verwendung eines DTN im Weltraum.

Der Hauptvorteil der DTN-Architektur in diesem Beispiel ist, dass sie sich der Situation des Satelliten perfekt anpasst, der die Bilder speichern muss, weil es zu der Zeit der Bildaufnahme keine Konnektivität gibt. Es gibt zwei weitere Vorteile. Zum einen kann es sein, dass kein Kontakt lang genug dauert, um die Bilder zu senden. Sie können jedoch über die Kontakte auf die drei Bodenstationen verteilt werden. Zum anderen wird die Verwendung der Verbindung zwischen dem Satelliten und der Bodenstation von der Verbindung über das Backhaul-Netz abgekoppelt. Das bedeutet, dass der Satelliten-Download nicht durch eine langsame terrestrische Verbindung beschränkt wird, sondern mit voller Geschwindigkeit erfolgen kann. Dabei wird das Bündel bei der Bodenstation gespeichert, bis es an den Sammelpunkt weitergeleitet werden kann.

Ein wichtiges Thema, das von der Architektur nicht abgedeckt wird, ist die Suche nach guten Routen über die DTN-Knoten, d.h. nach einer zu verwendenden Route im gegebenen Pfad. Gute Routen hängen davon ab, wann Daten gesendet werden sollen und außerdem, welche Kontakte verwendet werden können. Einige Kontakte sind im Voraus bekannt. Ein gutes Beispiel ist die Bewegung von Himmelskörpern im Welt Raum. Bei diesem Weltraumexperiment war vorher bekannt, wann es zu Kontaktintervalle kommen würde, dass die Kontaktintervalle von 5 bis 14 Minuten pro Bodenstation reichten und dass die Kapazität der Abwärtsverbindung 8 134 Mbit/s betrug. Mit diesem Wissen kann der Transport eines Bündels von Bildern im Voraus geplant werden.

In anderen Fällen können die Kontakte vorhergesagt werden, aber mit geringerer Gewissheit. Als Beispiele seien Busse genannt, die sich nach einem mehr oder weniger festen Fahrplan regelmäßig begegnen (was jedoch auch schwanken kann), sowie Zeiten und Umfang der verkehrsarmen Bandbreite in ISP-Netzen, die aus vorherigen Daten abgeleitet werden. Im anderen Extrem gibt es Kontakte, die zeitweilig oder zufällig sind. Ein Beispiel ist das Übertragen von Daten über Handy von Benutzer zu Benutzer, in Abhängigkeit davon, welche Benutzer während des Tages einen Kontakt zueinander

herstellen. Wenn die Kontakte unberechenbar sind, besteht eine Routing-Strategie darin, Kopien des Bündels auf verschiedenen Pfaden zu schicken, in der Hoffnung, dass eine der Kopien am Ziel ankommt, bevor die Lebensdauer erreicht wird.

6.7.2 Bündel-Protokoll

Um die Funktionsweise von DTNs genauer zu beleuchten, wenden wir uns nun den IETF-Protokollen zu. DTNs sind Netze, die sich noch in der Entwicklungsphase befinden, was zur Folge hat, dass für experimentelle DTNs die verschiedensten Protokolle verwendet wurden, da nicht unbedingt IETF-Protokolle verwendet werden mussten. Dennoch bieten sie sich als guter Ausgangspunkt an, um viele der wichtigsten Themen anzusprechen.

In ► Abbildung 6.58 finden Sie den DTN-Protokollstapel. Das wichtigste Protokoll ist das Bündel-Protokoll, das in RFC 5050 beschrieben ist. Es ist dafür zuständig, Nachrichten von der Anwendung anzunehmen und als ein oder mehrere Bündel mittels Store-Carry-Forward an den DTN-Zielknoten zu senden. An der Abbildung ist auch zu erkennen, dass das Bündel-Protokoll über der TCP/IP-Ebene angesiedelt ist. In anderen Worten, TCP/IP kann auf jedem Kontakt verwendet werden, um Bündel zwischen DTN-Knoten zu verschieben. Diese Position wirft die Frage auf, ob das Bündel-Protokoll ein Protokoll der Transportschicht oder der Anwendungsschicht ist. Wie bei RTP sind wir der Meinung, dass das Bündel-Protokoll, auch wenn es über einem Transportprotokoll ausgeführt wird, vielen Anwendungen einen Transportdienst bietet. Aus diesem Grund behandeln wir DTNs in diesem Kapitel.

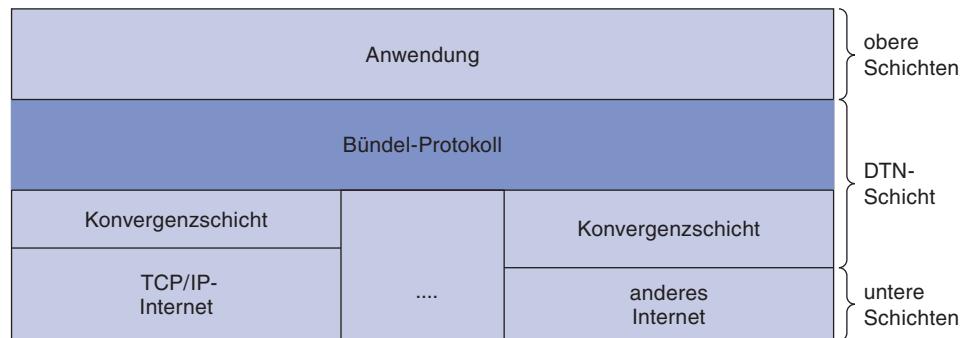


Abbildung 6.58: Protokollstapel eines verzögerungstoleranten Netzes.

In Abbildung 6.58 sehen wir, dass das Bündel-Protokoll auch über andere Arten von Protokollen wie UDP oder sogar andere Arten von Internet ausgeführt werden kann. In einem Weltraumnetz können die Verbindungen beispielsweise sehr lange Paketumlaufzeiten aufweisen. Die Übertragungszeit zwischen Erde und Mars kann, je nach der relativen Position der Planeten, leicht 20 Minuten betragen. Nun stellen Sie sich vor, wie gut TCP-Bestätigungen und -Neuübertragungen über eine solche Verbindung funktionieren würden, vor allem für relativ kurze Nachrichten. Gar nicht gut. Als Alternative könnte ein Protokoll verwendet werden, das Fehlerkorrekturcodes ein-

setzt. In sehr ressourcenarmen Sensornetzen könnte sogar ein etwas leichteres Protokoll als TCP verwendet werden.

Da das Bündel-Protokoll fest ist, aber auf einer Vielzahl von Transporten ausgeführt werden soll, muss es eine Funktionalitätslücke zwischen den Protokollen geben. Diese Lücke ist der Grund für die Konvergenzschicht in Abbildung 6.58. Diese Konvergenzschicht ist nur eine Zwischenschicht, die die Schnittstellen der Protokolle, die sie verbindet, aufeinander abstimmt. Per Definition gibt es für jeden Transport auf der unteren Schicht eine eigene Konvergenzschicht. Konvergenzschichten finden sich gewöhnlich in Standards, um neue und bestehende Protokolle zu verbinden.

Das Format der Bündel-Protokollnachrichten finden Sie in ► Abbildung 6.59. Die verschiedenen Felder in diesen Nachrichten verraten uns einige der Hauptaufgaben, die von dem Bündel-Protokoll zu bewältigen sind.

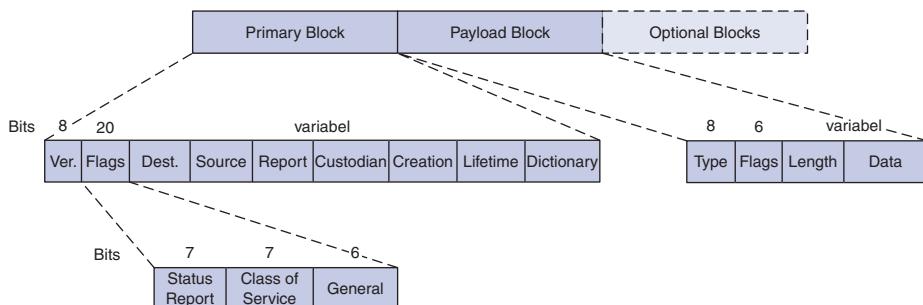


Abbildung 6.59: Das Format von Bündel-Protokollnachrichten.

Jede Nachricht besteht aus einem primären Block, den man sich als Header vorstellen kann, einem Nutzdatenblock für die Daten und optional anderen Blöcken, zum Beispiel zum Übertragen von Sicherheitsparametern. Der Primärblock beginnt mit dem *Version*-Feld (zurzeit 6), gefolgt von einem *Flags*-Feld. Unter anderem codieren die Flags eine Dienstklasse, die es einer Quelle erlaubt, ihre Bündel zu priorisieren, sowie andere Verarbeitungsanfragen, z.B. ob das Ziel das Bündel bestätigen soll.

Anschließend folgen Adressen, die drei interessante Teile des Entwurfs hervorheben. Neben den Identifikatorfeldern *Destination* und *Source* gibt es einen Identifikator *Custodian*. Dieser Wächter ist dafür zuständig, dass das Bündel zugestellt wird. Im Internet ist dies normalerweise der Quellknoten, da er die Daten erneut überträgt, wenn sie nicht am Ziel ankommen. In einem DTN hingegen besteht nicht immer eine Verbindung zum Quellknoten, sodass dieser nicht wissen kann, ob die Daten das Ziel erreicht haben. Diesem Problem wird in DTNs mit dem sogenannten **Wächtertransfer** (*custody transfer*) begegnet, bei dem ein anderer Knoten, der näher beim Ziel liegt, die Verantwortung dafür übernehmen kann, dass die Daten sicher ankommen. Wenn ein Bündel beispielsweise in einem Flugzeug gespeichert wird, um später von einem anderen Ort aus weitergeleitet zu werden, dann kann das Flugzeug zum Wächter des Bündels werden.

Der zweite interessante Aspekt ist, dass diese Identifikatoren *keine* IP-Adressen sind. Da das Bündel-Protokoll über einer Vielzahl von Transporten und Internets funktionieren soll, definiert es seine eigenen Identifikatoren. Diese Identifikatoren sind eigentlich eher konkrete Namen wie Webseiten-URLs als einfache IP-Adressen. Sie vermitteln den Eindruck, als ob bei DTNs das Routing auf Anwendungsebene erfolgt, wie E-Mail-Zustellung oder Verbreitung von Software-Updates.

Der dritte interessante Aspekt ist die Art, wie die Identifikatoren codiert sind. Es gibt sogar einen Identifikator namens *Report* für Diagnosenachrichten. Alle Identifikatoren sind als Referenzen zu dem *Dictionary*-Feld variabler Länge codiert. Dies erlaubt Komprimierung, wenn die Wächter- oder Report-Knoten die gleichen sind wie die Quelle oder das Ziel. Genau genommen standen bei einem Großteil des Nachrichtenformats die Erweiterbarkeit und die Effizienz im Vordergrund, und zwar durch die Verwendung einer kompakten Darstellung von Feldern variabler Länge. Die kompakte Darstellung ist wichtig für drahtlose Verbindungen und ressourcenarmen Knoten, wie sie in Sensornetzen zu finden sind.

Als Nächstes folgt das Feld *Creation*, das die Zeit angibt, zu der das Bündel erzeugt wurde, zusammen mit einer Sequenznummer von der Quelle zur Herstellung der Reihenfolge, plus dem Feld *Lifetime*, das mitteilt, wann die Bündeldaten nicht mehr nützlich sind. Diese Felder gibt es, weil die Daten unter Umständen recht lange in DTN-Knoten gespeichert werden und es eine Möglichkeit geben muss, abgelaufene Daten aus dem Netz zu entfernen. Im Gegensatz zum Internet setzen sie voraus, dass die DTN-Knoten lose synchronisierte Uhren haben.

Der Primärblock wird mit dem Feld *Dictionary* abgeschlossen. Anschließend folgt der Block mit den Nutzdaten. Dieser Block beginnt mit einem kurzen *Type*-Feld, das den Block als Nutzdatenblock identifiziert, gefolgt von einem kleinen Satz von Flags, die die Verarbeitungsoptionen beschreiben. Dann kommt das *Data*-Feld, dem das Feld *Length* vorausgeht. Und zum Schluss kann es noch weitere optionale Blöcke geben, wie einen Block, der die Sicherheitsparameter überträgt.

Viele Aspekte der DTNs sind zurzeit Gegenstand intensiver Forschung. Gute Routing-Strategien hängen von der Art der Kontakte ab, wie wir bereits oben erwähnt haben. Das Speichern der Daten im Netz wirft neue Probleme auf. Jetzt muss die Überlastungsüberwachung den Speicher der Knoten als eine weitere Art von Ressource betrachten, die geleert werden kann. Das Fehlen von Ende-zu-Ende-Kommunikation verschärft außerdem die Sicherheitsprobleme. Bevor ein DTN-Knoten als Wächter für ein Bündel fungiert, könnte es wissen wollen, dass der Sender berechtigt ist, das Netz zu verwenden, und dass das Bündel wahrscheinlich von dem Ziel gewollt ist. Lösungen zu diesen Problemen hängen von der Art des DTN ab, da Weltraumnetze sich von Sensornetzen unterscheiden.

Zusammenfassung

Die **Transportschicht** ist der Schlüssel für das Verständnis der **Schichtprotokolle**. Sie stellt verschiedene Dienste zur Verfügung. Der wichtigste ist ein zuverlässiger verbindungsorientierter Ende-zu-Ende-Bitstrom vom Sender zum Empfänger. Der Zugriff auf die Transportschicht erfolgt über die Dienstprimitive, die den Aufbau, die Nutzung und die Freigabe von Verbindungen ermöglichen. Eine weitverbreitete Transportschichtschnittstelle bieten die Berkeley-Sockets.

Transportprotokolle müssen in der Lage sein, Verbindungen in unzuverlässigen Netzen zu verwalten. Der Aufbau von Verbindungen wird durch das Vorhandensein verzögerter Duplikatpakete, die in unpassenden Momenten wieder auftauchen können, erschwert. Wegen der dadurch entstehenden Probleme ist für den Verbindungsaufbau das Dreiwege-Handshake erforderlich. Die Freigabe einer Verbindung ist einfacher als deren Aufbau, aber bei Weitem nicht so einfach wie man glaubt (siehe das Zwei-Armeen-Beispiel).

Selbst wenn die **Vermittlungsschicht** absolut zuverlässig ist, bleibt für die Transportschicht noch viel Arbeit. Sie muss alle Dienstprimitive verarbeiten, die Verbindungen und Timer verwalten, Bandbreite mit Überlastungsüberwachung zuweisen und für die Flusskontrolle ein Schiebefenster variabler Größe ausführen.

Die **Überlastungsüberwachung** sollte die die ganze zur Verfügung stehende Bandbreite gerecht auf die konkurrierenden Datenströme aufteilen und Änderungen bei der Netznutzung verfolgen. Das AIMD-Steuerungsgesetz konvergiert gegen eine faire und effiziente Zuweisung.

Das Internet verfügt über **zwei Haupttransportprotokolle**: UDP und TCP. UDP ist ein verbindungsloses Protokoll, das einfach eine Hülle für IP-Pakete bereitstellt und die zusätzliche Funktion bietet, mehrere Prozesse unter Verwendung einer einzigen IP-Adresse zu multiplexen und zu demultiplexen. UDP kann für Client-Server-Interaktionen verwendet werden, beispielsweise bei der Verwendung eines entfernten Prozeduraufrufs. Es kann auch zur Errichtung von Echtzeitprotokollen wie RTP verwendet werden.

Das wichtigste Transportprotokoll im Internet ist **TCP**. Es zeichnet sich durch einen zuverlässigen, bidirektionalen, überlastungsüberwachten Bytestrom aus mit einem 20-Byte-Header für alle Segmente. Im Internet können Segmente von Routern fragmentiert werden. Umfassende Arbeiten ist in die Optimierung der TCP-Leistung mit Algorithmen von Nagle, Clark, Jacobson, Karn und anderen geflossen.

Die **Netzleistung** wird normalerweise vom **Overhead** der Protokoll- und Segmentverarbeitung bestimmt – eine Situation, die sich mit steigender Geschwindigkeit noch verschlechtert. Protokolle sollten deshalb auf Minimierung der Anzahl von Segmenten und der Arbeit für große Bandbreite verzögerungspfade ausgelegt werden. Bei Gigabit-Netzen sind einfache Protokolle und eine schlanke Verarbeitung gefragt.

Verzögerungstolerante Netze bieten einen **Zustelldienst** über Verbindungen, die eine gelegentliche Konnektivität oder große Verzögerungen aufweisen. Zwischenknoten speichern, tragen und leiten Informationsbündel weiter, sodass sie am Ende am Ziel ankommen, auch wenn es nicht jederzeit einen funktionierenden Pfad vom Sender zum Empfänger gibt.



Übungsaufgaben

- 1** In unserem Beispiel mit den Transportprimitiven in Abbildung 6.2 ist LISTEN ein blockierender Aufruf. Ist er unbedingt erforderlich? Falls nicht, erklären Sie, wie eine nicht blockierende Primitive benutzt werden könnte. Welchen Vorteil hätte das gegenüber der im Text beschriebenen Methode?
- 2** Die Primitiven des Transportdienstes gehen beim Verbindungsaufbau von einer Asymmetrie zwischen den beiden Endpunkten aus, ein Ende (Server) führt ein LISTEN aus, während das andere Ende (Client) ein CONNECT ausführt. In Peer-to-Peer-Anwendungen zum Dateiaustausch wie BitTorrent hingegen sind alle Endpunkte Peers. Es gibt keine Server- oder Clientfunktionalität. Wie können die Transportdienstprimitiven verwendet werden, um solche Peer-to-Peer-Anwendungen zu erstellen?
- 3** In dem Modell von Abbildung 6.4 wird davon ausgegangen, dass Pakete auf der Vermittlungsschicht verloren gehen können und deshalb einzeln bestätigt werden müssen. Nehmen wir an, dass die Vermittlungsschicht zu 100 % zuverlässig ist und nie Pakete verloren gehen. Muss das Modell in Abbildung 6.4 geändert werden und wenn ja, wie?
- 4** In beiden Teilen von Abbildung 6.6 gibt es jeweils einen Kommentar, dass der Wert von SERVER_PORT bei Client und Server gleich sein muss. Warum ist dies so wichtig?
- 5** Kann in dem Beispiel mit dem Internetdateiserver (Abbildung 6.6) der Systemaufruf CONNECT() auf dem Client auch aus einem anderen Grund fehlschlagen, als dass die LISTEN-Warteschlange auf dem Server voll ist? Gehen Sie bei Ihrer Antwort von einem perfekten Netz aus.
- 6** Ein Kriterium für die Entscheidung, ob ein Server die ganze Zeit aktiv sein sollte oder nur, wie ein Prozessserver, auf Anfrage, hängt davon ab, wie oft der angebotene Dienst verwendet wird. Fällt Ihnen irgendein anderes Entscheidungskriterium ein?
- 7** Nehmen Sie an, dass eine taktgesteuerte Methode zum Erzeugen anfänglicher Sequenznummern mit einem 15 Bit großen Taktzähler benutzt wird. Jede 100 ms erfolgt ein Taktimpuls, und die maximale Paketlebensdauer ist 60 s. Wie oft muss eine Synchronisation stattfinden:
 - a. im schlechtesten Fall?
 - b. wenn die Daten 240 Sequenznummern/Minute verbrauchen?
- 8** Warum muss die maximale Paketlebensdauer T lang genug sein, um sicherzustellen, dass nicht nur das Paket, sondern auch seine Bestätigung verschwunden sind?
- 9** Stellen Sie sich vor, dass zum Aufbau von Verbindungen ein Zweiwege- anstelle eines Dreiwege-Handshakes benutzt wird. Mit anderen Worten, die dritte Nachricht war nicht erforderlich. Sind jetzt Deadlocks möglich? Führen Sie ein Beispiel auf oder zeigen Sie, dass es keine gibt.
- 10** Stellen Sie sich ein n -Armeen-Problem vor, bei dem die Übereinkunft von zwei beliebigen blauen Armeen zum Sieg ausreicht. Gibt es ein Protokoll, durch das die blaue Armee gewinnen könnte?

- 11** Betrachten Sie das Problem der Wiederherstellung nach einem Host-Absturz (Abbildung 6.18). Welche zwei Sender-Empfänger-Strategien sind die besten zur Minimierung des Risikos eines Protokollfehlers, wenn das Intervall zwischen dem Schreiben und dem Senden einer Bestätigung, oder umgekehrt, relativ kurz gehalten werden kann?
- 12** Angenommen in Abbildung 6.20 wird ein neuer Datenfluss E hinzugefügt, der einen Pfad von $R1$ nach $R2$ nach $R6$ nimmt. Wie ändert sich die Max-Min-Zuordnung der Bandbreite für die fünf Datenflüsse?
- 13** Weitere Strategien zur Verbesserung der Fairness bei der Überlastungsüberwachung sind AIAD (*Additive Increase Additive Decrease*), MIAD (*Multiplicative Increase Additive Decrease*) und MIMD (*Multiplicative Increase Multiplicative Decrease*). Diskutieren Sie diese drei Strategien hinsichtlich der Konvergenz und der Stabilität.
- 14** Wozu gibt es UDP? Hätte es nicht genügt, die Benutzerprozesse einfach nackte IP-Pakete senden zu lassen?
- 15** Betrachten Sie ein einfaches Protokoll auf Anwendungsebene, das auf UDP aufgebaut wurde und dem Client ermöglicht, eine Datei von einem entfernten Server abzurufen, der über eine bekannte Adresse verfügt. Der Client sendet zuerst eine Anforderung mit dem Dateinamen und der Server antwortet mit einer Folge von Datenpaketen, die verschiedene Teile der angeforderten Datei enthalten. Um die Zuverlässigkeit und die Zustellung in der korrekten Reihenfolge sicherzustellen, verwenden Client und Server ein Stop-and-Wait-Protokoll. Treten bei diesem Protokoll Probleme auf, abgesehen von den offensichtlichen Leistungsproblemen? Denken Sie sorgfältig über die Möglichkeit nach, dass Prozesse abstürzen können.
- 16** Ein Client sendet eine 128-Byte-Anfrage über ein 1-GB-Glasfaserkabel an einen Server, der 100 km entfernt ist. Wie effizient ist die Leitung während des entfernten Prozeduraufrufs?
- 17** Greifen Sie die Situation in der obigen Frage erneut auf. Berechnen Sie die geringstmögliche Antwortzeit für die vorhandene 1-Gbit/s-Leitung sowie für eine 1-Mbit/s-Leitung. Welche Schlüsse können Sie ziehen?
- 18** Sowohl UDP als auch TCP verwenden Portnummern, um das Ziel bei der Zustellung einer Nachricht anzugeben. Geben Sie zwei Gründe an, warum diese Protokolle eine neue abstrakte Kennung (Portnummer) einführen, anstatt die Prozesskennungen zu verwenden, die bereits bei Entwurf der Protokolle vorhanden waren.
- 19** Mehrere RPC-Implementierungen bieten dem Client die Option, RPC implementiert für UDP oder RPC implementiert für TPC zu verwenden. Unter welchen Bedingungen wird ein Client RPC auf UDP bzw. RPC auf TCP vorziehen?
- 20** Betrachten Sie zwei Netze $N1$ und $N2$ mit der gleichen durchschnittlichen Verzögerung zwischen einer Quelle A und einem Ziel D haben. In $N1$ ist die Verzögerung der verschiedenen Pakete gleichmäßig verteilt, mit einer maximalen Verzögerung von 10 Sekunden. Im Gegensatz dazu haben in $N2$ 99 % der Pakete eine Verzögerung von weniger als eine Sekunde, aber die maximale Verzögerung ist unbegrenzt hoch. Diskutieren Sie, wie in diesen beiden Fällen ein RTP (*Real-time Transport Protocol*) verwendet werden kann, um Audio-/Video-daten live zu übertragen.

- 21** Wie groß ist die kleinste maximale TCP-Übertragungseinheit, einschließlich TCP- und IP-Overhead, aber ohne den Overhead der Sicherungsschicht?
- 22** Die Fragmentierung von Datagrammen und das erneute Zusammensetzen werden von IP gehandhabt und sind für TCP unsichtbar. Heißt das, dass sich TCP über die Ankunft von Datagrammen in der falschen Reihenfolge keine Gedanken machen muss?
- 23** Über RTP werden Audiodaten in CD-Qualität übertragen, d. h. als ein Paar von 16-Bit-Samples mit einer Rate von 44 100 Samples/s, je eines für jeden Stereokanal. Wie viele Pakete muss RTP in der Sekunde übertragen?
- 24** Könnte man den RTP-Code in den Betriebssystemkern einbauen, und zwar zusammen mit dem UDP-Code? Erklären Sie Ihre Antwort.
- 25** Ein Prozess auf Host 1 wurde Port p und einer auf Host 2 wurde Port q zugewiesen. Kann es zwischen diesen beiden Ports gleichzeitig zwei oder mehr TCP-Verbindungen geben?
- 26** In Abbildung 6.36 finden Sie zusätzlich zu dem 32-Bit-Feld *Acknowledgement* ein *ACK*-Bit im vierten Wort. Wird hier wirklich etwas hinzugefügt? Warum oder warum nicht?
- 27** Die maximale Nutzlast eines TCP-Segments sind 65 495 Byte. Warum wurde eine derart seltsame Zahl gewählt?
- 28** Beschreiben Sie zwei Möglichkeiten, um in den Zustand *SYN RCVD* in Abbildung 6.39 überzugehen.
- 29** Betrachten Sie die Wirkung der Slow-Start-Methode auf eine Leitung mit einer Paketumlaufzeit von 10 ms und ohne Überlastung. Das Empfangsfenster ist 24 KB und die maximale Segmentgröße beträgt 2 KB. Wie lange dauert es, bis das erste volle Fenster gesendet werden kann?
- 30** Nehmen Sie an, dass das TCP-Überlastungsfenster auf 18 KB gesetzt ist und ein Timeout stattfindet. Wie groß ist das Fenster, wenn die nächsten vier Übertragungen erfolgreich sind? Gehen Sie von einer maximalen Segmentgröße von 1 KB aus.
- 31** Angenommen die TCP-Paketumlaufzeit beträgt aktuell 30 ms und die folgenden Bestätigungen kommen nach 26, 32 bzw. 24 ms an. Wie lautet die neue Schätzung der Übertragungszeit unter Verwendung des Algorithmus von Jacobson? Verwenden Sie $\alpha=0,9$.
- 32** Ein TCP-Rechner sendet vollständige Fenster mit 65 535 Byte über einen 1-Gbit/s-Kanal, der eine Übertragungsverzögerung von 10 ms in eine Richtung hat. Welcher maximale Durchsatz kann erreicht werden? Welche Leistungseffizienz ergibt sich?
- 33** Wie lautet die höchste Leitungsgeschwindigkeit, mit der ein Host TCP-Nutzdaten von 1 500 Byte und einer maximalen Paketlebensdauer von 120 s übertragen kann, ohne dass Sequenznummern mehrmals verwendet werden? Berücksichtigen Sie den Overhead von TCP, IP und Ethernet. Gehen Sie davon aus, dass Ethernet-Rahmen fortlaufend gesendet werden.
- 34** Um den Beschränkungen von IPv4 zu begegnen, musste die IETF große Anstrengungen unternehmen, die letztlich zu der Entwicklung von IPv6 führten. Selbst heute noch wird diese neue Version nur zögerlich angenommen. Die Beschränkungen von TCP bedürfen nicht so großer Anstrengungen. Erklären Sie, warum.

- 35** Wie hoch ist die maximale Datenrate pro Verbindung in einem Netz, das eine maximale Segmentgröße von 128 Byte, eine maximale Segmentlebensdauer von 30 s und eine 8-Bit-Sequenznummer hat?
- 36** Angenommen, Sie messen die für den Empfang eines Segments benötigte Zeit. Findet ein Interrupt statt, lesen Sie die Systemuhr in Millisekunden ab. Nachdem das Segment vollständig verarbeitet wurde, lesen Sie die Uhr wieder ab. Sie haben 270 000-mal 0 ms und 730 000-mal 1 ms gemessen. Wie lange dauert es, um ein Segment zu empfangen?
- 37** Eine CPU führt Anweisungen mit 1 000 MIPS aus. Daten können jeweils in 64-Bit-Gruppen kopiert werden, wobei das Kopieren jedes Worts zehn Anweisungen kostet. Kann das System eine 1-Gbit/s-Leitung unterstützen, wenn ein ankommendes Paket viermal kopiert werden muss? Gehen Sie der Einfachheit halber davon aus, dass alle Anweisungen (auch die zum Lesen oder Schreiben im Speicher) mit der vollen 1 000-MIPS-Rate laufen.
- 38** Um das Problem der mehrmaligen Verwendung von Sequenznummern zu vermeiden, während noch alte Pakete vorhanden sind, kann man 64-Bit-Sequenznummern verwenden. Theoretisch läuft ein Lichtwellenleiter mit 75 Tbit/s. Welche maximale Paketlebensdauer ist erforderlich, um sicherzustellen, dass künftige 75-Tbit/s-Netze mit 64-Bit-Sequenznummern keine Probleme haben? Gehen Sie davon aus, dass jedes Byte wie bei TCP eine eigene Sequenznummer hat.
- 39** In Abschnitt 6.6.5 wurde berechnet, dass eine Gigabit-Leitung 80 000 Pakete/s in den Host einspeisen kann, wobei zur Verarbeitung nur 6 250 Anweisungen erlaubt sind und die Hälfte der CPU-Zeit für Anwendungen übrig bleibt. Diese Berechnung basiert auf 1 500-Byte-Paketen. Führen Sie die Berechnung erneut mit Paketen der ARPANET-Größe (128 Byte) durch. Gehen Sie in beiden Fällen davon aus, dass die angegebenen Paketgrößen den gesamten Overhead enthalten.
- 40** Bei einem 1-Gbit/s-Netz mit 4 000 km Länge ist die Übertragungsverzögerung und nicht die Bandbreite der einschränkende Faktor. Betrachten Sie ein MAN, bei dem die Quelle und das Ziel im Durchschnitt 20 km voneinander entfernt sind. Bei welcher Datenrate entspricht die Paketumlaufzeit aufgrund der Lichtgeschwindigkeit der Übermittlungsverzögerung für ein 1-KB-Paket?
- 41** Berechnen Sie das Bandbreite-Verzögerung-Produkt für die folgenden Netze: (1) T1 (1,5 Mbit/s), (2) Ethernet (10 Mbit/s), (3) T3 (45 Mbit/s) und (4) STS-3 (155 Mbit/s). Gehen Sie von einer Paketumlaufzeit von 100 ms aus. Denken Sie daran, dass im TCP-Header 16 Bit für die Fenstergröße reserviert sind. Was sind die Folgerungen angesichts Ihrer Berechnungen?
- 42** Wie sieht das Produkt aus Bandbreite und Übertragungsverzögerung für einen 50-Mbit/s-Kanal für einen geostationären Satelliten aus? Wie groß sollte das Fenster (in Anzahl von Paketen) sein, wenn die Pakete alle 1 500 Byte (einschließlich Overhead) groß sind?
- 43** Der Dateiserver in Abbildung 6.6 ist noch lange nicht perfekt und könnte ein paar Verbesserungen vertragen. Nehmen Sie die folgenden Änderungen vor:
- Geben Sie dem Client ein drittes Argument, das einen Bytebereich angibt.
 - Fügen Sie ein Client-Flag hinzu, -w, mit dem die Datei auf den Server geschrieben werden kann.

- 44** Eine gemeinsame Funktion, die alle Netzprotokolle benötigen, dient der Behandlung von Nachrichten. Erinnern wir uns, dass Protokolle Nachrichten bearbeiten, indem sie Header hinzufügen/entfernen. Einige Protokolle können eine Nachricht in mehrere Teile zerlegen, die sie später wieder zu einer Nachricht zusammenfügen. Entwickeln und implementieren Sie hierzu eine Bibliothek zur Nachrichtenverwaltung, die Unterstützung für folgende Funktionen bieten soll: eine neue Nachricht erzeugen, eine Nachricht mit einem Header versehen, einen Header aus einer Nachricht entfernen, eine Nachricht in zwei Nachrichten zerlegen, zwei Nachrichten zu einer zusammenfassen und eine Kopie einer Nachricht speichern. Ihre Implementierung muss beim Kopieren von einem Puffer in einen anderen die Daten so weit wie möglich minimieren. Es ist extrem wichtig, dass die Operationen zur Behandlung der Nachrichten nicht die Daten in den Nachrichten berühren, sondern nur Zeiger einsetzen.
- 45** Entwerfen und implementieren Sie ein Chat-System, in dem sich mehrere Benutzergruppen miteinander unterhalten können. Ein Chat-Koordinator befindet sich an einer bekannten Netzadresse, verwendet zur Kommunikation mit den Chat-Clients UDP, richtet für jede Chat-Sitzung Chat-Server ein und pflegt ein Chat-Sitzungsverzeichnis. Pro Chat-Sitzung gibt es einen Chat-Server. Ein Chat-Server verwendet TCP für die Kommunikation mit den Clients. Ein Chat-Client ermöglicht Benutzern, eine Chat-Sitzung zu beginnen, einer Sitzung beizutreten und eine Sitzung zu verlassen. Entwickeln und implementieren Sie den Koordinator-, Server- und Client-Code.

Die Anwendungsschicht

| | |
|--|-----|
| 7.1 DNS – Domain Name System | 695 |
| 7.2 E-Mail | 708 |
| 7.3 World Wide Web | 734 |
| 7.4 Streaming Audio und Video | 792 |
| 7.5 Content Delivery | 832 |

» Nachdem wir mit den vorbereitenden Ausführungen fertig sind, kommen wir nun zu der Schicht, in der die Anwendungen zu finden sind. Die Schichten unterhalb der Anwendungsschicht stellen die grundlegenden Transportdienste bereit, führen aber noch keine Aufgaben aus, wie sie Benutzern vorschweben. Dies übernehmen die eigentlichen Netzanwendungen, von denen wir in diesem Kapitel einige eingehender untersuchen werden.

Selbst auf der Anwendungsschicht bedarf es allerdings unterstützender Protokolle, damit Anwendungen funktionieren können. Wie z.B. dem Domain Name System, kurz DNS, welches die Namensgebung im Internet regelt und von so großer Bedeutung ist, dass wir es vorab betrachten müssen, bevor wir uns anschließend drei Arten von Netzanwendungen zuwenden können: E-Mail, World Wide Web und Multimedia. Zum Abschluss des Kapitels beschäftigen wir uns mit der Verteilung von Inhalten, auch über Peer-to-Peer-Netzwerke.



7.1 DNS – Domain Name System

Theoretisch könnten Programme auf Hosts, Mailboxen und andere Ressourcen über die Netzadressen (also IP-Adressen) der Computer zugreifen, auf denen diese Ressourcen abgelegt sind; nur können sich Menschen diese Adressen nicht gut merken. Hinzu kommt, dass ein Unternehmen, dessen Kunden es gewohnt sind, die Webseiten des Unternehmens unter `128.111.24.41` zu finden, im Falle eines Umzugs des Webservers auf einen anderen Rechner mit neuer IP-Adresse alle Kunden über die geänderte IP-Adresse informieren müsste. Aufgrund dieser Überlegungen wurde ein neues System mit besser lesbaren Namen eingeführt, das die Rechnernamen von den Rechneradressen entkoppelte. In diesem System könnte der Webserver unseres Unternehmens nun beispielsweise `www.cs.washington.edu` lauten – ungeachtet der aktuellen IP-Adresse. Bleibt der Fakt, dass das Netz selbst nur numerische Adressen versteht. Es wird daher ein Verfahren benötigt, wie die Namen in Netzadressen umgewandelt werden können. Wie diese Umwandlung im Internet funktioniert, ist Thema der nächsten Abschnitte.

In den guten alten Tagen des ARPANET gab es einfach eine Datei `hosts.txt`, in der alle Rechnernamen und ihre IP-Adressen aufgelistet waren. Jede Nacht holten sich die Hosts die Adressendatei von dem Standort, an dem sie gepflegt wurde. Für ein Netz mit ein paar Hundert großen Timesharing-Rechnern funktionierte dieser Ansatz relativ gut.

Doch den Verantwortlichen wurde bald klar – und zwar lange bevor Millionen von PCs an das Internet angeschlossen wurden –, dass dieser Ansatz auf Dauer nicht funktionsfähig sein würde. Nicht nur, dass die Datei irgendwann zu groß werden würde, man musste auch noch mit ständigen Konflikten aufgrund doppelter Host-Namen rechnen, sofern die Namen nicht zentral verwaltet würden – was sich jedoch bei einem riesigen internationalen Netz aufgrund der damit verbundenen Last und Latenzzeit von selbst verbat. Um diese Probleme zu lösen, wurde 1983 das **Domain Name System**, kurz **DNS** entwickelt. Seitdem ist das DNS ein essenzieller Bestandteil des Internets.

Im Kern besteht das DNS aus dem Entwurf eines hierarchischen, auf Domänen basierten Benennungsschemas und eines verteilten Datenbanksystems zur Implementierung dieses Benennungsschemas. Es wird vorwiegend zur Abbildung von Host-Namen auf IP-Adressen benutzt, dient aber auch noch anderen Zwecken. DNS ist in den RFCs 1034, 1035, 2181 definiert und in vielen anderen RFCs weiter ausgearbeitet.

Kurz gefasst, funktioniert das DNS folgendermaßen: Um einen Namen in eine IP-Adresse umzuwandeln, ruft ein Anwendungsprogramm eine als **Resolver** bezeichnete Bibliotheksprozedur auf und übergibt ihr den Namen als Parameter. In Abbildung 6.6 haben wir bereits ein Beispiel für einen solchen Resolver, `gethostbyname`, gesehen. Der Resolver sendet eine Anfrage mit dem gesuchten Namen an einen lokalen DNS-Server, der den Namen nachschlägt. Der DNS-Server schickt dem Resolver eine Antwort mit der zugehörigen IP-Adresse, die vom Resolver schließlich an den Aufrufer zurückgeliefert wird. (Anfrage und Antwort werden als UDP-Pakete übertragen.) Ausgestattet mit der IP-Adresse, kann das Programm dann eine TCP-Verbindung zum Ziel aufbauen oder UDP-Pakete an dieses Ziel senden.

7.1.1 DNS-Namensraum

Die Verwaltung zahlreicher Namen, die sich ständig ändern, ist nicht einfach. Beim Postsystem erfolgt die Namensverwaltung dadurch, dass Briefe (implizit oder explizit) Land, Stadt und Straße sowie Hausnummer des Adressaten aufweisen müssen. Dank dieser hierarchischen Adressierung gibt es keine Verwechslung zwischen dem Martin Albers in der Hauptstraße in Dortmund und dem Martin Albers in der Bleibtreustraße in Berlin. DNS funktioniert auf die gleiche Weise.

Im Falle des Internets wird die oberste Ebene der Namenshierarchie von der **ICANN**-Organisation (*Internet Corporation for Assigned Names and Numbers*) verwaltet. ICANN wurde speziell zu diesem Zweck im Jahre 1998 gegründet, als Teil des Reifungsprozesses, der das Internet in ein weltumspannendes Wirtschaftsunternehmen transformierte. Konzeptionell ist das Internet in über 200 **Top-Level-Domänen** aufgeteilt, die jede zahlreiche Hosts umfasst. Jede dieser Domänen ist in Subdomänen unterteilt; diese sind weiter unterteilt usw. All diese Domänen können durch einen Baum dargestellt werden (►Abbildung 7.1). Die Blätter des Baumes stellen Domänen dar, die keine Subdomänen (selbstverständlich aber Rechner) enthalten. Eine Blatt-Domäne in diesem Baum kann für einen einzelnen Host stehen oder ein Unternehmen mit Tausenden von Hosts darstellen.

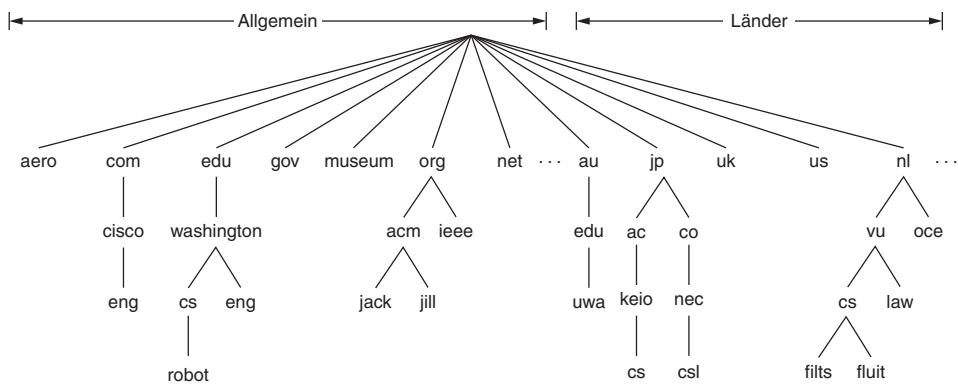


Abbildung 7.1: Teil des Namensraums der Internetdomänen.

Die Domänen der obersten Ebene gliedern sich in zwei Bereiche: Allgemein (Generic) und Länder (Countries). Die allgemeinen Domänen umfassen zum einem die Originaldomänen aus den 1980ern und zum anderen Domänen, die beim ICANN beantragt wurden. Weitere allgemeine Top-Level-Domänen werden in der Zukunft hinzukommen.

Die Liste der Länderdomänen enthält einen Eintrag für jedes Land, wie in ISO 3166 festgelegt. Internationalisierte Länderdomänennamen mit Zeichen, die nicht aus dem lateinischen Alphabet stammen, wurden 2010 eingeführt. Dank dieser Domänen können Hosts nun in Arabisch, Kyrillisch, Chinesisch oder beliebigen anderen Sprachen angegeben werden.

Eine Domäne der zweiten Ebene, wie z.B. *Name-des-Unternehmens.com*, für sich registrieren zu lassen, ist nicht schwierig. Die Top-Level-Domänen werden von **Registrierungsstellen** verwaltet, die vom ICANN eingesetzt wurden. Um nun einen Namen zu registrieren, wendet man sich an die Registrierungsstelle für die gewünschte Top-Level-Domäne (*.com* in diesem Fall) um abzuklären, ob der gewünschte Name verfügbar und nicht bereits das Warenzeichen eines anderen Unternehmens ist. Wenn es keine Probleme gibt, zahlt der Interessent eine kleine Jahresgebühr und erhält den Namen.

Die zunehmende Kommerzialisierung und Internationalisierung des Internets führt immer wieder zu Streitpunkten, gerade auch im Bereich der Namensvergabe. Nicht einmal das ICANN ist vor diesbezüglichen Kontroversen gefeit. Beispielsweise erforderte es mehrere Jahre und etliche Gerichtsverhandlungen, bevor über die Einführung einer *xxx*-Domäne und die Frage entschieden wurde, ob es eine gute oder eine schlechte Sache sei, wenn pornografische Inhalte auf freiwilliger Basis in eine eigene Domäne gestellt würden. (Einige Leute waren der Ansicht, dass pornografische Inhalte überhaupt nicht ins Internet gehören, während andere eine Konzentrierung dieser Inhalte in einer einzigen Domäne forderten, damit Filter sie leicht erkennen und vor Kindern verbergen können.) Während sich einige Domänen selbst organisieren, gibt es für andere Restriktionen, die festlegen, wer in dieser Domäne einen Namen erwerben kann (►Abbildung 7.2). Doch welche Restriktionen sind sinnvoll? Nehmen wir z.B. die *pro*-Domäne, die für qualifizierte Fachkräfte gedacht ist. Wer aber ist eine Fachkraft? Ärzte und Anwälte sind natürlich Fachkräfte. Wie aber verhält es sich mit selbstständigen Fotografen, Klavierlehrern, Zauberern, Installateuren, Friseuren, Kammerjägern, Tattoo-Künstlern, Söldnern oder Prostituierten? Ist jemand, der einem solchen Beruf nachgeht, eine qualifizierte Fachkraft? Und wer entscheidet darüber?

| Domäne | Gedacht für | Einführung | Restriktiv |
|---------------|-----------------------------------|-------------------|-------------------|
| com | Kommerzielle Unternehmen | 1985 | Nein |
| edu | Bildungseinrichtungen | 1985 | Ja |
| gov | Regierung | 1985 | Ja |
| int | Internationale Organisationen | 1988 | Ja |
| mil | Militär | 1985 | Ja |
| net | Netzverwaltung | 1985 | Nein |
| org | nicht kommerzielle Organisationen | 1985 | Nein |
| aero | Luftfahrt | 2001 | Ja |
| biz | Unternehmen | 2001 | Nein |
| coop | Gesellschaften | 2001 | Ja |
| info | Information | 2002 | Nein |

Abbildung 7.2: Allgemeine Top-Level-Domänen.

| Domäne | Gedacht für | Einführung | Restriktiv |
|--------|----------------------|------------|------------|
| museum | Museen | 2002 | Ja |
| name | Leute | 2002 | Nein |
| pro | Fachkräfte | 2002 | Ja |
| cat | Katalanische Kultur | 2005 | Ja |
| jobs | Beruf | 2005 | Ja |
| mobi | Mobilgeräte | 2005 | Ja |
| tel | Kontaktinformationen | 2005 | Ja |
| travel | Reise-Industrie | 2005 | Ja |
| xxx | Sex-Industrie | 2010 | Nein |

Abbildung 7.2: Allgemeine Top-Level-Domänen. (*Forts.*)

Mit Domänennamen lässt sich auch Geld machen. Tuvalu (der Inselstaat), dessen Landcode sich als Werbeplattform für Fernsehsender geradezu anbietet, hat z.B. seine *tv*-Domäne für 50 Millionen US-Dollar verpachtet. Es gibt kaum ein Wort der englischen Sprache, das noch nicht in eine *com*-Domäne verwandelt wurde. Probieren Sie es einmal aus: Haushaltsartikel, Tiere, Pflanzen, Körperteile etc. – es ist fast alles belegt. Mittlerweile gibt es sogar einen eigenen Namen für das Registrieren von Domänen zu dem alleinigen Zweck diese später zu einem viel höheren Preis an tatsächlich Interessierte weiterzuverkaufen: **Cybersquatting** (zu deutsch „Cyberbesetzung“). Viele Unternehmen, die zu Beginn der Internetära nicht schnell genug reagierten, mussten später, als sie versuchten, sich die passende Domäne zu sichern, feststellen, dass die gewünschten Domänennamen bereits weg waren. Bei der Vergabe der Namen gilt eben grundsätzlich: wer zuerst kommt, bekommt den Namen – sofern nicht gerade eingetragene Warenzeichen verletzt werden oder ein Betrugsvorwurf vorliegt. Gleichwohl werden die Richtlinien für die Schlichtung von Namensstreitigkeiten ständig weiterentwickelt und verbessert.

Der Name einer Domäne ist der Pfad von ihr aufwärts bis zur (nicht benannten) Wurzel. Die einzelnen Komponenten werden durch Punkte („dot“) getrennt. Die technische Abteilung von Cisco Systems würde also gemäß dieses Schemas z.B. *eng.cisco.com*. lauten – und nicht etwa nach Unix-Art */com/cisco/eng*. Diese hierarchische Benennung bedeutet, dass *eng.cisco.com*. nicht mit einer möglichen Verwendung von *eng* in *eng.washington.edu*., das die Fakultät für Anglistik an der Universität von Washington eventuell benutzt, in Konflikt steht.

Domänennamen können absolut oder relativ sein. Ein absoluter Domänenname endet immer mit einem Punkt (z.B. *eng.cisco.com*.), ein relativer Domänenname nicht. Relative Namen müssen in einem Kontext interpretiert werden, aus dem ihre tatsächliche Bedeutung eindeutig erschlossen werden kann. In beiden Fällen bezieht sich eine benannte Domäne auf einen spezifischen Knoten im Baum und alle Knoten darunter.

Bei Domänennamen wird die Groß- und Kleinschreibung nicht beachtet, d.h., *edu*, *Edu* und *EDU* bedeuten allesamt das Gleiche. Der Name einer einzelnen Komponente kann bis zu 63 Zeichen enthalten, der vollständige Pfadname darf allerdings nicht länger als 255 Zeichen sein.

Grundsätzlich können Domänen sowohl unter allgemeinen als auch unter Länderdomänen in den Baum eingefügt werden. Beispielsweise könnte *cs.washington.edu* auch ebenso gut unter der Landesdomäne *us* als *cs.washington.wa.us* aufgelistet werden. In der Praxis stehen aber fast alle Organisationen der USA unter einer allgemeinen Domäne, während fast alle Organisationen außerhalb der USA unter der Domäne ihres jeweiligen Landes stehen. Die Registrierung unter mehreren Top-Level-Domänen ist nicht verboten, und wird von großen Unternehmen häufig genutzt (z.B. *sony.com*, *sony.net* und *sony.de*).

Jede Domäne bestimmt, wie die unter ihr liegenden Domänen zugewiesen werden. Japan verwendet z.B. die Domänen *ac.jp* und *co.jp* als Pendants zu *edu* und *com*. Die Niederlande machen diese Unterscheidung nicht und platzieren alle Organisationen direkt unter *nl*. Die folgenden Domänennamen stehen daher allesamt für die Informatikfakultäten ihrer Universitäten (CS steht für Computer Science):

- *cs.washington.edu* (University of Washington, USA)
- *cs.vu.nl* (Vrije Universiteit, Niederlande)
- *cs.keio.ac.jp* (Keio Universität, Japan)

Um eine neue Domäne anzulegen, ist die Genehmigung der übergeordneten Domäne erforderlich. Wird z.B. eine VLSI-Gruppe an der Universität von Washington ins Leben gerufen und möchte die Gruppe unter *vlsi.cs.washington.edu* firmieren, braucht sie die Genehmigung von demjenigen, der *cs.washington.edu* verwaltet. Wird eine neue Universität gegründet, sagen wir die Universität von Northern South Dakota, muss sie den Verwalter der *edu*-Domäne um die Zuweisung der Adresse *unsd.edu* bitten (sofern diese noch frei ist). Auf diese Weise werden Namenskonflikte vermieden und jede Domäne behält die Kontrolle über die unter ihr registrierten Unterdomänen. Nachdem eine neue Domäne angelegt und registriert wurde, kann sie ihrerseits Unterdomänen anlegen, z.B. *cs.unsd.edu*. Eine Genehmigung der übergeordneten Domänen ist dafür nicht erforderlich.

Die Benennung basiert auf organisatorischen Grenzen, nicht auf physikalischen Netzen. Befinden sich z.B. die Fakultäten für Informatik und Elektrotechnik im gleichen Gebäude und verwenden beide das gleiche LAN, dann können sie trotzdem getrennte Domänen haben. Umgekehrt können die Hosts des Fachbereichs Informatik physikalisch auf die Gebäude Babbage Hall und Turing Hall verteilt sein und trotzdem der gleichen Domäne angehören.

7.1.2 Ressourcendatensätze

Mit jeder Domäne, sei es ein einzelner Host oder eine Top-Level-Domäne, können mehrere **Ressourcendatensätze** (*resource record*) verbunden sein. Der übliche Ressour-

cendatensatz für einen einzelnen Host ist seine IP-Adresse. Daneben gibt es aber noch viele andere Arten von Ressourcendatensätzen. Gibt ein Resolver einen Domänenamen an das DNS weiter, erhält er die Ressourcendatensätze zurück, die mit diesem Namen verbunden sind. Die primäre Funktion des DNS ist also die Abbildung von Domänennamen auf Ressourcendatensätzen.

Ein Ressourcendatensatz ist ein Fünfertupel. Obwohl Ressourcendatensätze der besseren Effizienz wegen binär codiert sind, werden sie in Publikationen üblicherweise als ASCII-Text dargestellt, und zwar je ein Ressourcendatensatz pro Zeile. Das Format, das wir verwenden, ist wie folgt aufgebaut:

```
Domain_name    Time_to_live    Class    Type    Value
```

Der *Domain_name* gibt die Domäne an, für die der Datensatz gilt. Normalerweise liegen für eine Domäne viele Datensätze vor und jede Kopie der Datenbank enthält Informationen über mehrere Domänen. Dieses Feld ist somit der primäre Suchschlüssel für Datenbankabfragen. Die Reihenfolge der Datensätze in der Datenbank hat keine Bedeutung.

Das Feld *Time_to_live* gibt einen Hinweis darauf, wie dauerhaft die Informationen eines Datensatz sind. Sehr beständige Informationen erhalten einen hohen Wert, z.B. 86 400 (Anzahl der Sekunden eines Tages), während flüchtigen Informationen ein kleiner Wert wie z.B. 60 (eine Minute) zugewiesen wird. Wir kommen auf diesen Punkt später im Zusammenhang mit Caching noch einmal zurück.

Das dritte Feld jedes Ressourcendatensatzes ist die *Class*. Für Internetinformationen ist dies immer *IN*. Bei Nichtinternet-Informationen können andere Codes verwendet werden. In der Praxis ist dies aber selten.

Das Feld *Type* gibt an, um welche Art von Datensatz es sich handelt. Die wichtigsten Typen sind in ►Abbildung 7.3 aufgeführt.

| Typ | Bedeutung | Wert |
|-------|--------------------------|---|
| SOA | Start of Authority | Parameter für die betreffende Zone |
| A | IPv4-Adresse eines Hosts | 32-Bit-Ganzzahl |
| AAAA | IPv6-Adresse eines Hosts | 128-Bit-Ganzzahl |
| MX | Mail Exchange | Mit Prioritäten versehener Mailserver, der bereit ist, E-Mails anzunehmen |
| NS | Nameserver | Name eines Servers der betreffenden Domäne |
| CNAME | Kanonischer Name | Alias-Eintrag |
| PTR | Zeiger (Pointer) | Alias für eine IP-Adresse |
| SPF | Sender Policy Framework | Textcodierung der Mail-Sende-Richtlinie |
| SRV | Service | Host, der den Dienst anbietet |
| TXT | Text | Nicht interpretierter ASCII-Text |

Abbildung 7.3: Die wichtigsten DNS-Ressourcendatensatztypen.

Ein *SOA*-Satz gibt den Namen der primären Informationsquelle über die Zone des Nameservers (wird unten beschrieben), die E-Mail-Adresse des Administrators, eine eindeutige Seriennummer und verschiedene Flags und Timeouts an.

Der wichtigste Datensatztyp ist *A* (Adresse). Er enthält eine 32 Bit lange IP-Adresse einer Schnittstelle für einen bestimmten Host. Das Pendant dazu, der Typ *AAAA* oder „quad A“ enthält eine 128 Bit lange IPv6-Adresse. Jeder Internethost muss mindestens eine IP-Adresse haben, damit andere Rechner mit ihm kommunizieren können. Einige Hosts haben zwei oder mehr Netzschmittstellen. In diesem Fall gibt es für jede Netzschmittstelle einen eigenen Ressourcendatensatz vom Typ *A* oder *AAAA*. Entsprechend kann das DNS mehrere Adressen für einen einzigen Namen zurückliefern.

Ein häufiger vorkommender Datensatztyp ist *MX*. Er gibt den Namen des Hosts an, der bereit ist, E-Mail für die betreffende Domäne anzunehmen. Er wird verwendet, weil nicht jeder Rechner dafür eingerichtet ist, E-Mail anzunehmen. Möchte beispielsweise jemand eine E-Mail an *bill@microsoft.com* senden, dann muss der absendende Host unter *microsoft.com* einen Mailserver finden, der E-Mails entgegennimmt. Der *MX*-Datensatz kann ihm diese Informationen liefern.

Von besonderer Bedeutung ist auch der *NS*-Datensatztyp. Der *NS*-Datensatz gibt einen Nameserver für die Domäne oder Subdomäne an. Dies ist ein Host, der über eine Kopie der DNS-Datenbank (für eine Domäne) verfügt. Nameserver sind wichtig für das Nachschlagen von Namen – ein Prozess, den wir uns in Kürze näher ansehen werden.

CNAME-Datensätze ermöglichen das Erstellen von Alias-Namen. Jemand, der mit der Namensvergabe im Internet vertraut ist und dem Benutzer *paul* von der Informatikfakultät des MIT eine E-Mail senden möchte, könnte sich z.B. ableiten, dass die Adresse des Benutzers *paul* am MIT *paul@cs.mit.edu* lauten muss. Tatsächlich wird seine E-Mail aber niemals ankommen, denn die Domäne der Informatikfakultät des MIT ist *csail.mit.edu*. Um Uneingeweihten Ärger zu ersparen, könnte das MIT daher als zusätzlichen Service einen *CNAME*-Eintrag erstellen, der Menschen und Programme in die richtige Richtung weist – beispielsweise:

```
cs.mit.edu    86400   IN   CNAME   csail.mit.edu
```

Wie *CNAME* verweist *PTR* auf einen anderen Namen. Im Unterschied zu *CNAME*, bei dem es sich im Grunde nur um eine Makrodefinition handelt (d.h. einen Mechanismus, der einen Text durch einen anderen Text ersetzt), ist *PTR* ein regulärer DNS-Datentyp, dessen Interpretation vom Kontext abhängt. In der Praxis wird er fast immer verwendet, um einen Namen mit einer IP-Adresse zu verbinden – und so die Suche nach IP-Adressen zu unterstützen und die Namen der entsprechenden Rechner zurückliefern zu können. Man spricht in diesem Fall auch von **Reverse Lookups** (umgekehrte Namensauflösung, Inverssuche).

SRV ist ein relativ neuer Datensatztyp, mit dessen Hilfe ein Host als Anbieter eines speziellen Dienstes innerhalb einer Domäne identifiziert werden kann. So könnte der Webserver für *cs.washington.edu* beispielsweise als *cockatoo.cs.washington.edu* identifiziert werden. Der *SRV*-Datensatz ist eine Verallgemeinerung des auf Mailserver spezialisierten *MX*-Datensatzes.

Ebenfalls relativ neu ist der *SPF*-Datensatz, in dem eine Domäne in codierter Form die Information ablegen kann, welcher Rechner in der Domäne für das Senden von E-Mails ins restliche Internet zuständig ist. Die Empfängerrechner können dann anhand dieser Information die Gültigkeit der E-Mail überprüfen. Wurde eine E-Mail von einem Rechner empfangen, der sich selbst *zweifelhaft* nennt, der Domänen-Datensatz aber besagt, dass E-Mails aus der betreffenden Domäne immer von einem Rechner namens *smtp* kommen, kann man wohl davon ausgehen, dass es sich um eine Junkmail handelt.

Der letzte Datensatztyp in der Tabelle ist *TXT*. *TXT*-Datensätze wurden ursprünglich vorgesehen, um Domänen die Möglichkeit zu geben, sich selbst auf beliebige Weise zu identifizieren. Heute nutzt man dazu üblicherweise maschinenlesbare Informationen, meist in Form eines *SPF*-Datensatzes.

Das letzte Feld im Ressourcendatensatz ist das Feld *Value*. Dieses Feld kann eine Zahl, ein Domänenname oder eine ASCII-Zeichenfolge sein. Die Semantik hängt vom Datensatztyp ab. Eine kurze Beschreibung der *Value*-Felder der wichtigsten Datensatztypen ist in Abbildung 7.3 aufgeführt.

In ►Abbildung 7.4 können Sie sehen, welche Art von Informationen man typischerweise in der DNS-Datenbank einer Domäne findet. Die Abbildung zeigt einen Teil einer (halbhypothetischen) Datenbank für die Domäne *cs.vu.nl* (siehe Abbildung 7.1) mit sieben verschiedenen Ressourcedatensatztypen.

```
; Authoritative data for cs.vu.nl
cs.vu.nl.    86400   IN  SOA    star boss (9527,7200,7200,241920,86400)
cs.vu.nl.    86400   IN  MX     1 zephyr
cs.vu.nl.    86400   IN  MX     2 top
cs.vu.nl.    86400   IN  NS     star

star        86400   IN  A      130.37.56.205
zephyr      86400   IN  A      130.37.20.10
top         86400   IN  A      130.37.20.11
www         86400   IN  CNAME  star.cs.vu.nl
ftp          86400   IN  CNAME  zephyr.cs.vu.nl

flits        86400   IN  A      130.37.16.112
flits        86400   IN  A      192.31.231.165
flits        86400   IN  MX     1 flits
flits        86400   IN  MX     2 zephyr
flits        86400   IN  MX     3 top

rowboat      IN  A      130.37.56.201
              IN  MX     1 rowboat
              IN  MX     2 zephyr

little-sister  IN  A      130.37.62.23
laserjet     IN  A      192.31.231.216
```

Abbildung 7.4: Ausschnitt aus einer fiktiven DNS-Datenbank für *cs.vu.nl*.

Die erste Nicht-Kommentar-Zeile in Abbildung 7.4 liefert ein paar grundlegende Informationen über die Domäne, mit denen wir uns nicht weiter beschäftigen. Dann folgen zwei Einträge, die angeben, wohin man sich wenden kann, um eine an *person@cs.vu.nl* adressierte E-Mail zu übergeben. Als Erstes sollte man es stets mit dem Rechner *zephyr* versuchen. Falls das nicht klappt, kann man auf *top* ausweichen. Die nächste Zeile gibt den Nameserver für die Domäne an: *star*.

Nach der Leerzeile, die zur besseren Übersichtlichkeit eingefügt wurde, folgen Zeilen, die die IP-Adressen zu *star*, *zephyr* und *top* angeben. Danach kommt ein Alias, *www.cs.vu.nl*, sodass diese Adresse auch ohne Angabe eines bestimmten Rechners verwendet werden kann. Dank dieses Alias kann die Domäne *cs.vu.nl* ihren WWW-Server ändern, ohne dass dadurch die Adresse ungültig wird, unter der sie in der Öffentlichkeit bekannt ist. Gleiches gilt für das Alias *ftp.cs.vu.nl*.

Der Abschnitt für den Rechner *flits* enthält zwei IP-Adressen und drei Alternativen für die Behandlung von E-Mails, die an *flits.cs.vu.nl* gesendet wurden. Die erste Wahl ist natürlich *flits* selbst. Sollte *flits* heruntergefahren sein, stehen als zweite und dritte Option *zephyr* und *top* zur Verfügung.

Die nächsten drei Zeilen sind ein typischer Eintrag für einen Rechner, in diesem Fall *rowboat.cs.vu.nl*. Angegeben sind die IP-Adresse sowie die primäre und sekundäre Mail-Annahmestelle. Dann folgt ein Eintrag für einen Rechner, der selbst keine E-Mail empfangen kann, gefolgt von einem Eintrag, der typisch für Drucker ist, die mit dem Internet verbunden sind.

7.1.3 Nameserver

Rein theoretisch könnte ein einzelner Nameserver die komplette DNS-Datenbank beherbergen und alle an die Datenbank gerichteten Anfragen beantworten. In der Realität wäre ein solcher Server jedoch derart überladen, dass er praktisch nutzlos würde. Schlimmer noch: würde er jemals abstürzen, dann bräche das ganze Internet zusammen.

Um solche Probleme zu vermeiden, wird der DNS-Namensraum in nicht überlappende **Zonen** aufgeteilt. Eine Möglichkeit, wie der Namensraum aus Abbildung 7.1 aufgeteilt werden könnte, ist in ▶ Abbildung 7.5 dargestellt. Jede der eingekreisten Zonen enthält dabei einen bestimmten Teil des Baums.

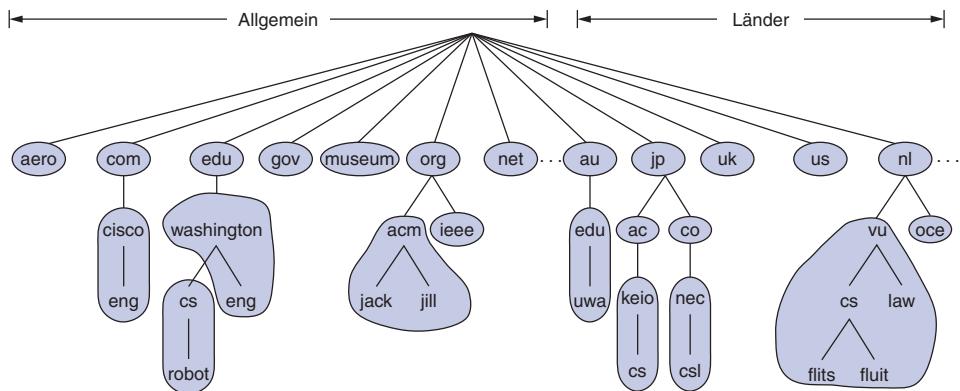


Abbildung 7.5: Teil des DNS-Namensraums, aufgeteilt in Zonen (in der Abbildung eingekreist).

Die Entscheidung, wo die Grenzen innerhalb einer Zone gezogen werden, liegt beim Zonenadministrator und wird größtenteils auf der Grundlage getroffen, wie viele Nameserver gewünscht sind und wo diese liegen sollen. In Abbildung 7.5 z.B. verfügt die University of Washington über eine Zone `washington.edu`, die `eng.washington.edu`, nicht aber `cs.washington.edu` mit verwaltet. Letzteres ist eine separate Zone mit eigenen Nameservern. Der Grund für eine solche Aufteilung kann z.B. sein, dass eine Fakultät, hier die Anglistik, keinen eigenen Nameserver führen möchte, während die Informatik sehr wohl darauf Wert legt.

Jede Zone ist mit einem oder mehreren Nameservern verbunden. Dies sind Hosts, auf denen die Datenbank für die Zone gespeichert ist. Normalerweise hat eine Zone einen primären Nameserver, der die Informationen einer auf seiner Festplatte befindlichen Datei entnimmt, und einen oder mehrere sekundäre Nameserver, die ihre Informationen vom primären Nameserver erhalten. Um die Zuverlässigkeit zu verbessern, können einige der Nameserver außerhalb der Zone installiert werden.

Das Nachschlagen eines Namens und die Suche nach der zugehörigen Adresse wird als **Namensauflösung** (*name resolution*) bezeichnet. Liegt einem Resolver eine Anfrage zu einem Domänenamen vor, leitet er die Anfrage an einen lokalen Nameserver weiter. Fällt die gesuchte Domäne unter die Zuständigkeit des Nameservers, wie z.B. `top.cs.vu.nl` unter `cs.vu.nl` fällt, so gibt er die autoritativen Ressourcendatensätze aus. Ein **autoritativer Datensatz** (*authoritative record*) ist ein Datensatz, der von der Stelle (Autorität) stammt, die den Datensatz verwaltet, und ist somit immer korrekt. Autoritative Datensätze stehen im Gegensatz zu im Cache zwischengespeicherten Datensätzen (*cached record*), die eventuell überholt sein können.

Was aber passiert, wenn es sich um eine sogenannte Remote-Domäne handelt, wie es z.B. der Fall ist, wenn `flits.cs.vu.nl` die IP-Adresse von `robot.cs.washington.edu` aus der UW-Zone (Universität von Washington) sucht? In diesem Fall startet der Nameserver, sofern es lokal keine im Cache zwischengespeicherten Informationen gibt, eine Fernabfrage (*remote query*). Diese Anfrage folgt dem Schema, das in ► Abbildung 7.6 dargestellt ist. Schritt 1 zeigt die Anfrage, die an den lokalen Nameserver gesendet wird. Die Anfrage enthält den gesuchten Domänenamen, den Typ (A) und die Klasse (IN).

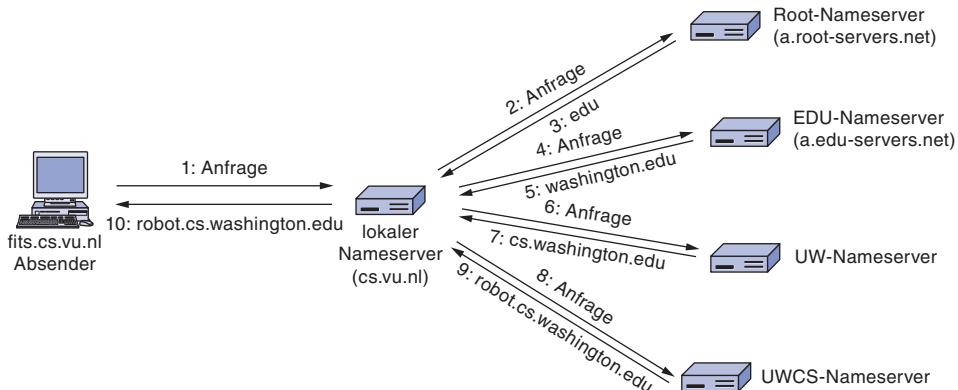


Abbildung 7.6: Suche nach einem entfernten Namen durch einen Resolver in zehn Schritten.

Weiter geht es oben in der Namenshierarchie (Schritt 2 in Abbildung 7.6). Dazu wird eine Anfrage an einen der **Root-Nameserver** gesendet – also einen Nameserver, der die Informationen über alle Top-Level-Domänen besitzt. Damit ein Nameserver Kontakt zu einem Root-Server aufnehmen kann, benötigt er entsprechende Informationen über einen oder mehrere Root-Nameserver. Diese Informationen liegen üblicherweise in Form einer Systemkonfigurationsdatei vor, die beim Start des DNS-Servers in den DNS-Cache geladen wird. Sie besteht einfach aus einer Liste von NS-Datensätzen für die Root-Server und die zugehörigen A-Datensätze.

Es gibt insgesamt 13 DNS-Root-Server, die wenig einfallsreich `a.root-servers.net`, `b.root-servers.net` bis `m.root-servers.net` heißen. Rein theoretisch könnte es sich bei einem Root-Server um einen einzelnen Rechner handeln. Da jedoch das gesamte Internet von den Root-Servern abhängt, handelt es sich um äußerst leistungsfähige und mehrfach replizierte Rechner. So sind die meisten dieser Server in vielen geografischen Regionen verfügbar und können per Anycast-Routing erreicht werden (d.h., es wird ein Paket an die am nächsten gelegene Instanz einer Zieladresse geschickt, siehe Kapitel 5). Stabilität und Leistung des Internets werden auf diese Weise verbessert.

Es ist unwahrscheinlich, dass der Root-Nameserver die Adresse eines Rechners aus der UW-Zone kennt. Vermutlich kennt er nicht einmal den Nameserver von UW. Er muss allerdings den Nameserver der `edu`-Domäne kennen, innerhalb der sich `cs.washington.edu` befindet. Dessen Namen und IP-Adresse liefert er als Teil seiner Antwort zurück (Schritt 3).

Der lokale Nameserver setzt nun seine Suche fort. Er schickt die gesamte Anfrage an den `edu`-Nameserver (`a.edu-servers.net`), der den Nameserver für UW zurückliefert (Schritte 4 und 5). Seinem Ziel näher gekommen, sendet der lokale Nameserver die Anfrage zum UW-Nameserver (Schritt 6). Würde der gesuchte Domänenname zur Anglistik gehören, wäre hier das Ende der Suche erreicht. Doch die Informatik hat sich entschlossen, ihren eigenen Nameserver zu unterhalten. Die Anfrage ergibt daher den Namen und die IP-Adresse des Nameservers der Informatik („Computer Science“) an der Universität von Washington (Schritt 7).

Zu guter Letzt richtet der lokale Nameserver seine Anfrage an den Nameserver der Informatik an der Universität von Washington (Schritt 8). Da dieser für die Domäne *cs.washington.edu* zuständig ist, weiß er die endgültige Antwort und sendet diese zurück (Schritt 9). Der lokale Nameserver leitet die Antwort an *flits.cs.vu.nl* weiter (Schritt 10). Die Namensauflösung ist vollzogen.

Mithilfe von Programmen wie *dig* können Sie diesen Vorgang selbst nachvollziehen (*dig* ist auf den meisten Unix-System standardmäßig installiert). Die Eingabe

```
dig@a.edu-servers.net robot.cs.washington.edu
```

führt z.B. dazu, dass dem Nameserver *a.edu-servers.net* eine Anfrage nach *robot.cs.washington.edu* gesendet und das Ergebnis ausgegeben wird. Mit anderen Worten: Sie erhalten die Informationen, die Schritt 4 aus dem obigen Beispiel entsprechen und denen Sie den Namen und die IP-Adresse des UW-Nameservers entnehmen können.

In dem oben dargelegten Szenario gibt es drei Punkte, drei technische Details, die besondere Beachtung verdienen. Zum einen kommen in Abbildung 7.6 zwei Arten von Anfragemechanismen zum Einsatz. Wenn der Host *flits.cs.vu.nl* seine Anfrage an den lokalen Nameserver sendet, verfolgt dieser im Auftrag von *flits* die Namensauflösung, bis er die gewünschte Antwort zurückliefern kann. Teilantworten liefert er *nicht* zurück. Teilantworten können hilfreich sein, aber sie sind nicht das, worauf die Anfrage abzielte. Diesen Mechanismus bezeichnet man als **rekursive Anfrage** (*recursive query*).

Demgegenüber setzt der Root-Nameserver (und alle nachgeschalteten Nameserver) die Anfrage für den lokalen Nameserver nicht rekursiv fort. Er liefert einfach die gefundene Teilantwort zurück und wendet sich der nächsten Anfrage zu. Für die Fortsetzung der Namensauflösung durch Abschicken weiterer Anfragen ist der lokale Nameserver zuständig. Diesen Mechanismus bezeichnet man als **iterative Anfrage** (*iterative query*).

Im Zuge einer Namensauflösung können beide Mechanismen zum Einsatz kommen (siehe obiges Beispiel). Rekursive Anfragen mögen grundsätzlich wünschenswert sein, doch viele Nameserver, insbesondere Root-Server, unterstützen sie nicht. Sie sind zu beschäftigt. Iterative Anfragen bürden die Hauptlast dem Urheber der Anfrage auf. Der Grund, warum lokale Nameserver rekursive Anfragen unterstützen, ist, dass sie als Dienstleister für die Hosts in ihrer Domäne fungieren. Dadurch müssen diese Host keinen eigenen Nameserver betreiben; es genügt ihnen, den lokalen Nameserver zu adressieren.

Der zweite Punkt betrifft das Caching. Alle Antworten, Teilantworten mit eingeschlossen, werden zwischengespeichert. Auf diese Weise kann der lokale Nameserver, wenn ihm ein zweiter *cs.vu.nl*-Host eine Anfrage nach *robot.cs.washington.edu* sendet, sofort mit einer Antwort dienen. Besser noch, wenn ein Host eine Anfrage nach einem anderen Host aus derselben Domäne schickt, sagen wir *galah.cs.washington.edu*, kann die Anfrage direkt an den autoritativen Nameserver weitergeleitet werden. Analog können Anfragen nach anderen Domänen in *washington.edu* direkt mit dem *washington.edu*-Nameserver beginnen. Auf diese Weise reduziert das Caching die Anzahl der zur Namensauflösung notwendigen Schritte und verbessert den Durchsatz. Bei dem oben skizzierten Beispiel handelt es sich also um das Worst-Case-Szenario, das eintritt, wenn keinerlei verwertbare Information zwischengespeichert ist.

Allerdings sind zwischengespeicherte Informationen nicht autoritativ, da eventuelle Änderungen in `cs.washington.edu` nicht automatisch an alle Caches der Welt, die von diesen Änderungen betroffen sind, weitergegeben werden. Aus diesem Grund sollten Cache-Einträge nicht zu lange leben und deshalb verfügt jeder Ressourcendatensatz über das Feld *Gültigkeit*. Es teilt entfernten Nameservern mit, wie lange Datensätze im Cache zwischenzuspeichern sind. Hat ein bestimmter Rechner jahrelang die gleiche IP-Adresse, kann man davon ausgehen, dass es sicher ist, die Informationen einen Tag lang im Cache zu behalten. Andere Informationen sollten dagegen nach ein paar Sekunden oder einer Minute entfernt werden.

Der dritte anzusprechende Punkt ist das Übertragungsprotokoll, das für die Anfragen und Antworten verwendet wird: UDP. DNS-Nachrichten werden als UDP-Pakete gesendet, mit einem einfachen, hier nicht weiter beschriebenen Format für Anfragen, Antworten und Nameservern, die für die Fortsetzung der Namensaflösung verwendet werden können. Trifft in einem vorgegeben, kurzen Zeitraum keine Antwort ein, wiederholt der DNS-Client die Anfrage bzw. versucht es nach mehreren vergeblichen Wiederholungen bei einem anderen Server für die Domäne. Auf diese Weise können Anfragen auch beantwortet werden, wenn der Server heruntergefahren wurde oder das Anfrage- bzw. Antwortpaket einmal verloren geht. Jede Anfrage wird mit einer 16-Bit-Kennung versehen, die in die Antwort kopiert wird, damit die Nameserver die eingehenden Antworten auch dann den Anfragen zuordnen können, wenn es mehrere gleichzeitig bearbeitete Anfragen gibt.

Das DNS dient einem einfachen und klar umrissenen Zweck. Doch lassen Sie sich davon nicht täuschen. Es handelt sich um ein ebenso großes wie komplexes verteiltes System, das aus Millionen von zusammenarbeitenden Nameservern besteht. Das DNS ist ein wichtiges Bindeglied, das die Verbindung zwischen für Menschen verständliche Domänennamen und den IP-Adressen der Rechner herstellt. Es nutzt Replikation und Caching zur Leistungsverbesserung, arbeitet äußerst zuverlässig und zeichnet sich durch hohe Stabilität aus.

Bisher wurde noch gar nichts über das Thema Sicherheit gesagt. Wie man sich aber leicht vorstellen kann, hätte es katastrophale Folgen, wenn jemand in böswilliger Absicht die Abbildung der Namen auf die IP-Adressen verändern würde. Aus diesem Grunde wurden für das DNS spezielle Sicherheitserweiterungen, die DNSSEC, entwickelt, denen wir uns in *Kapitel 8* zuwenden.

Für manche Anwendungsgebiete ist es interessant, die aufzulösenden Namen etwas weiter zu fassen, beispielsweise indem man als Namen einen gesuchten Inhalt angibt und als Antwort die IP-Adresse eines nahe gelegenen Hosts erhält, der den gewünschten Inhalt anbietet. Denken Sie z.B. an die Suche nach einem herunterladbaren Kinofilm. In diesem Fall geht es um den Kinofilm und als Antwort ist die IP-Adresse jedes nahe gelegenen Rechners geeignet, auf dem eine Kopie des Films verfügbar ist. Netze zur Verteilung von Inhalten (*Content Distribution Network, CDN*) sind eine Möglichkeit, diese Art von Abbildung zu unterstützen. Wie diese auf dem DNS aufsetzen, erfahren Sie in Abschnitt 7.5.

7.2 E-Mail

Elektronische Post, heute allseits **E-Mail** genannt, gibt es bereits seit etwa drei Jahrzehnten. Schneller und billiger als die traditionelle Post, gehörte die E-Mail quasi von Beginn an zu den populärsten Internetanwendungen. Vor 1990 wurde E-Mail vor allem in akademischen Kreisen verwendet. In den neunziger Jahren des 20. Jahrhunderts erfuhr sie weite Verbreitung in der Öffentlichkeit und das E-Mail-Aufkommen wuchs exponentiell an, sodass die Anzahl der täglich versendeten E-Mails mittlerweile sehr viel höher ist als die der auf dem normalen Postweg versendeten Briefe. Andere Formen der Netzwerkkommunikation, wie z.B. Instant Messaging oder Voice-over-IP konnten in den letzten zehn Jahren starke Zuwächse verzeichnen, doch das Arbeitstier der Internetkommunikation ist und bleibt die E-Mail. In der Wirtschaft wird E-Mail für die firmeninterne Kommunikation eingesetzt, beispielsweise um externe Mitarbeiter, die über die ganze Welt verteilt sind, so zu vernetzen, dass sie an einem gemeinsamen Projekt zusammenarbeiten können. Unglücklicherweise leidet die E-Mail unter einem Phänomen, das auch schon von der Briefpost bekannt ist: ein Großteil der eingehenden E-Mail-Nachrichten – ungefähr 9 von 10 Nachrichten – sind Junkmails oder **Spam** (McAfee, 2010).

Die E-Mail hat wie die meisten Kommunikationsformen ihre eigenen Konventionen und Stile entwickelt. E-Mail ist z.B. sehr informell und die Leute sind schneller bereit, eine E-Mail zu senden als einen Brief zu schreiben. Leute, denen es nie in den Sinn gekommen wäre, einen Prominenten anzurufen oder ihm gar einen Brief zu schreiben, zögern nicht eine Sekunde, eine schnell aufgesetzte E-Mail zu senden. Da bei der E-Mail-Kommunikation Hinweise auf Rang, Alter oder Geschlecht weitgehend unterbleiben, konzentrieren sich E-Mail-Debatten meist auf Inhalte. Als E-Mail verteilt, kann die brillante Idee eines Praktikanten mehr Aufmerksamkeit bekommen und mehr bewegen als eine nichtssagende Verlautbarung des stellvertretenden Generaldirektors.

E-Mail ist voller Jargon-Ausdrücke und Abkürzungen, wie z.B. MfG (Mit freundlichen Grüßen), ROFL (Rolling On the Floor Laughing) und IMHO (In My Humble Opinion). Viele Leute verwenden auch kleine ASCII-Symbole, sogenannte **Smileys** wie das allgemeinwährtige „;-)“. Sollte Ihnen dieses Symbol unbekannt sein, drehen Sie das Buch einfach um 90 Grad im Uhrzeigersinn. Dieses und andere **Emoticons** helfen dabei, die Tonlage einer E-Mail-Nachricht zu vermitteln, und werden mittlerweile auch in anderen mittelbaren Kommunikationsformen, wie z.B. dem Instant Messaging, genutzt.

Doch nicht nur die Gepflogenheiten haben sich über die vielen Jahre der E-Mail-Nutzung verändert, auch die E-Mail-Protokolle haben sich weiterentwickelt. Die ersten E-Mail-Systeme basierten lediglich auf Dateiübertragungsprotokollen mit der Konvention, dass die erste Zeile jeder Nachricht (d.h. der Datei) die Adresse des Empfängers enthält. Im Laufe der Zeit wichen man von diesem Ansatz immer weiter ab und es kamen immer mehr zusätzliche Optionen hinzu, wie z.B. die Möglichkeit, eine Nachricht bequem an eine Gruppe von Empfängern zu versenden, oder das Versenden von Bildern und anderen Multimediainhalten, das in den 1990ern immer wichtiger wurde. Auch die Programme zum Lesen von E-Mail wurden immer ausgefeilter, bekamen eine grafische Oberfläche

und erlaubten schließlich dem Benutzer, seine E-Mail-Nachrichten per Notebook von jedem Punkt der Erde aus abzufragen. Zu den jüngsten Entwicklungen gehört das Erkennen und Aussortieren unerwünschter E-Mail, das aufgrund des Überhandnehmens von Spam heute wichtiger Bestandteil der E-Mail-Programme und -Übertragungsprotokolle ist.

In unserer Beschreibung werden wir uns auf die Art und Weise konzentrieren, wie E-Mail-Nachrichten zwischen den Anwendern übertragen werden – und nicht auf den Umgang mit diversen E-Mail-Programmen. Nichtsdestotrotz werden wir nach einer kurzen Übersicht über die grundlegende Architektur mit der Benutzerschnittstelle des E-Mail-Systems beginnen, da dies der Teil des Systems ist, der den meisten Leser wohl am besten vertraut ist.

7.2.1 Architektur und Dienste

Dieser Abschnitt enthält eine Übersicht dessen, was E-Mail-Systeme können und wie sie organisiert sind. Die Architektur des E-Mail-Systems ist in ► Abbildung 7.7 dargestellt und besteht aus zwei Teilsystemen: den **Benutzeragenten** (*user agent*), mit denen die Benutzer Nachrichten lesen und senden können, und den **Nachrichtenübertragungsagenten** (*message transfer agent*), die Nachrichten von der Quelle zum Ziel befördern. Statt von Nachrichtenübertragungsagenten sprechen wir informell auch von **Mailservern**.

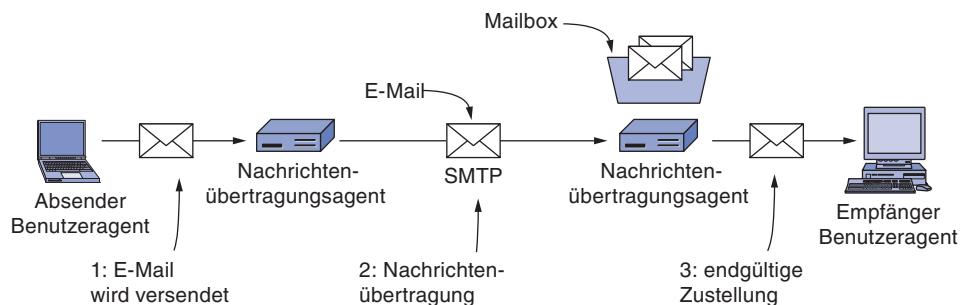


Abbildung 7.7: Architektur des E-Mail-Systems.

Die Benutzeragenten sind Programme mit grafischer – manchmal auch rein text- oder befehlsbasierter – Benutzerschnittstelle, über die der Benutzer mit dem E-Mail-System kommunizieren kann. Dazu gehört, dass der Benutzer neue Nachrichten erstellen, eingehende Nachrichten anzeigen und beantworten sowie empfangene Nachrichten archivieren, durchsuchen und löschen kann. Die Übergabe gesendeter Nachrichten an das E-Mail-System zur Übertragung an den Empfänger wird als **Mail Submission** bezeichnet.

Bestimmte Aufgaben übernehmen die Benutzeragenten automatisch, wobei sie versuchen, im Sinne des Benutzers zu handeln – beispielsweise wenn sie eingehende Mail filtern, um potenzielle Spamnachrichten auszusondern oder zumindest herabzustufen. Einige Benutzeragenten verfügen über fortgeschrittene Features, wie z.B. die Generierung automatischer Antworten („Ich bin im Urlaub, es ist wunderschön hier

und ich komme sobald nicht zurück.“). Der Benutzeragent läuft auf demselben Rechner, auf dem der Benutzer seine Mail liest. Es ist einfach ein weiteres Programm, das wahlweise auch nur nach Bedarf ausgeführt werden kann.

Die Nachrichtenübertragungsagenten sind üblicherweise Systemprozesse, die als Dienste im Hintergrund eines Mailserver-Rechners laufen, damit sie ständig verfügbar sind. Ihr Job besteht darin, E-Mail mittels **SMTP** (*Simple Mail Transfer Protocol*) automatisch vom Absender aus durch das System zum Empfänger zu leiten (siehe Schritt 2 in Abbildung 7.7).

SMTP wurde ursprünglich in RFC 821 spezifiziert, bevor es überarbeitet wurde und jetzt unter RFC 5321 läuft. SMTP sendet E-Mail-Nachrichten über bestehende Verbindungen und meldet den Übertragungsstatus sowie etwaige Fehler zurück. Für viele Anwendungen ist die Bestätigung der erfolgreichen Zustellung von großer Bedeutung, ja kann sogar von rechtlicher Relevanz sein. („Sehen Sie Herr Richter, mein E-Mail-System arbeitet nicht sehr zuverlässig. Ich vermute daher, dass die an mich gerichtete elektronische Vorladung irgendwo unterwegs verloren gegangen ist.“)

Nachrichtenübertragungsagenten erlauben auch die Erstellung von Mailinglisten, mit deren Hilfe identische Kopien einer Nachricht an alle in der Liste aufgeführten E-Mail-Adressen gesendet wird. Weitere fortgeschrittene Funktionen sind z.B. das Versenden von Kopien an zusätzliche Empfänger, Blindkopien, E-Mail mit hoher Priorität, geheime (verschlüsselte) E-Mail, alternative Empfänger (wenn der erste Empfänger nicht erreichbar ist) und die Möglichkeit für Sekretärinnen, die E-Mail ihres Chefs zu bearbeiten.

Das Konzept der Mailboxen beruht auf der Verknüpfung von Benutzer- und Nachrichtenübertragungsagenten und ist ein gängiges Format für die Verwaltung von E-Mail-Nachrichten. **Mailboxen** speichern die E-Mail-Nachrichten, die für einen bestimmten Benutzer empfangen wurden. Ihre Verwaltung obliegt den Mailservern. Die Benutzeragenten fungieren einfach nur als Sichtfenster auf den Inhalt der Mailboxen. Zu diesem Zweck senden die Benutzeragenten den Mailservern Befehle zur Manipulation der Mailboxen, zum Inspizieren ihres Inhalts, zum Löschen von Nachrichten und so weiter. Das Abrufen der Mails ist in Abbildung 7.7 der Abschluss der Zustellung (Schritt 3). Gemäß dieser Architektur kann ein Benutzer mehrere Benutzeragenten auf verschiedenen Rechnern für den Zugriff auf seine Mailbox verwenden.

Zwischen den Nachrichtenübertragungsagenten wird die E-Mail in einem Standardformat übertragen. Das ursprüngliche Format, RFC 822, wurde zu RFC 5322 überarbeitet und um die Unterstützung für Multimedia und internationalen Text erweitert. Diese Erweiterungen werden abgekürzt als **MIME** bezeichnet und weiter unten besprochen. Viele Leute beziehen sich auf Internet-E-Mail aber immer noch als RFC 822.

Ein wichtiges Konzept des E-Mail-Formats ist die Unterscheidung zwischen dem Umschlag (**Envelope**) und dem Inhalt. Im Umschlag ist die Nachricht gekapselt. Er enthält alle für die Beförderung der Nachricht nötigen Informationen, z.B. Zieladresse, Priorität und Sicherheit, die mit der eigentlichen Nachricht nichts zu tun haben. Der Umschlag hat für die Nachrichtenübertragungsagenten also die gleiche Bedeutung wie der Briefumschlag für die Post.

Die im Umschlag befindliche Nachricht umfasst zwei Teile: den **Header** und den Nachrichtenrumpf (**Body**). Der Header enthält Informationen für die Benutzeragenten. Der Nachrichtenteil ist gänzlich für den Empfänger bestimmt. Benutzeragenten interessieren sich kaum für ihn. Umschläge und Nachrichten sind in ▶ Abbildung 7.8 dargestellt.

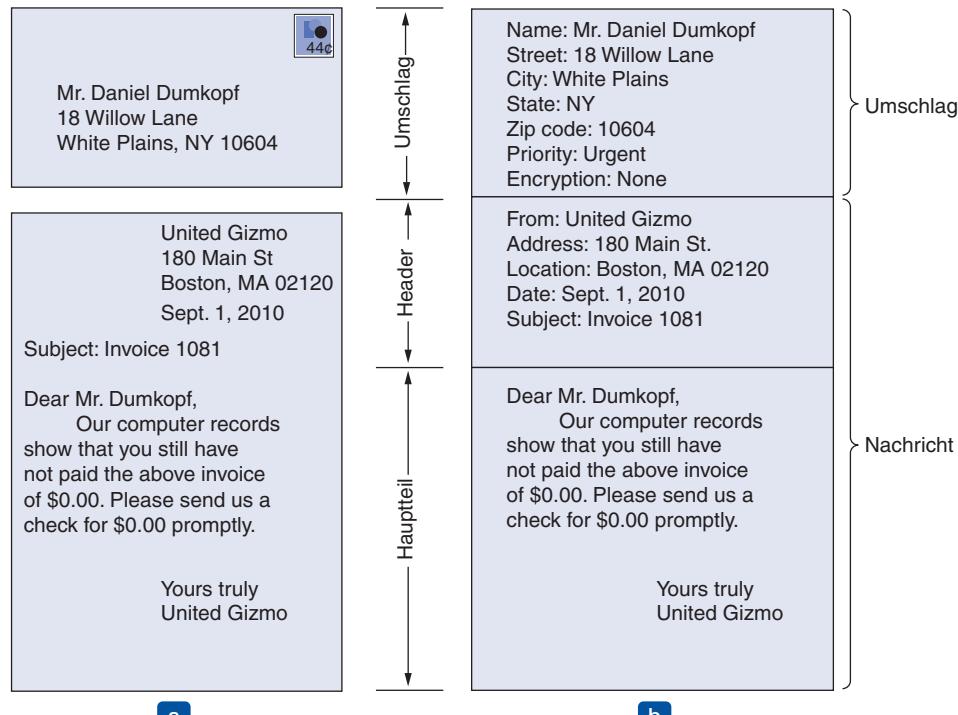


Abbildung 7.8: Umschläge und Nachrichten: (a) traditionelle Post, (b) E-Mail.

Im Folgenden werden wir den Weg einer E-Mail vom Absender zum Empfänger nachvollziehen und uns dabei die entsprechenden Teile des E-Mail-Systems näher ansehen. Unsere Reise beginnt mit dem Benutzeragenten.

7.2.2 Benutzeragenten

Ein Benutzeragent ist ein Programm (manchmal auch **Mailprogramm** genannt), das verschiedene Befehle zum Verfassen, Entgegennehmen und Beantworten von Nachrichten sowie zur Verwaltung von Mailboxen annimmt. Es gibt zahlreiche populäre Benutzeragenten, darunter beispielsweise Google Gmail, Microsoft Outlook, Mozilla Thunderbird oder Apple Mail, die ganz unterschiedliche Benutzeroberflächen besitzen. Die meisten Benutzeragenten haben anspruchsvolle menü- oder schaltflächengesteuerte grafische Benutzeroberflächen, die mit einer Maus oder – im Falle mobiler Kleingeräte – über eine Touchscreen bedient werden. Ältere Benutzeragenten, wie z.B. *Elm*, *mh* oder *Pine*, haben textgesteuerte Schnittstellen und verarbeiten Befehle, die aus einzelnen Zeichen bestehen und über die Tastatur eingegeben werden. Funktionell sind beide Systeme gleich, zumindest soweit es die Verarbeitung von Textnachrichten angeht.

► Abbildung 7.9 zeigt die typischen Elemente der Benutzeroberfläche eines Benutzeragenten. Vermutlich sieht die Oberfläche Ihres eigenen Benutzeragenten etwas ansprechender aus, aber die dargebotene Funktionalität dürfte ziemlich identisch sein.

Nach dem Aufruf zeigen die meisten Benutzeragenten eine Zusammenfassung der Nachrichten in der Mailbox des Benutzers an. Oft handelt es sich dabei um tabellarische Zusammenfassungen, die jede Nachricht in einer eigenen Zeile präsentieren, sortiert sind und Schlüsseleigenschaften der Nachricht, wie im Umschlag oder Header angegeben, hervorheben.

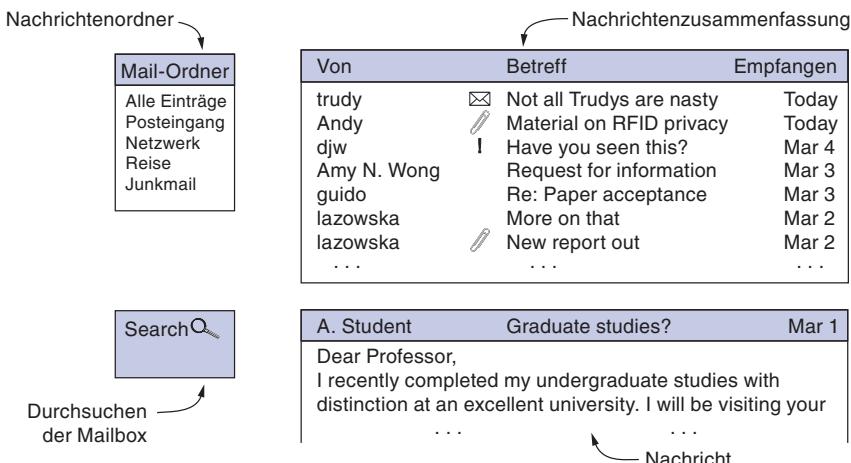


Abbildung 7.9: Typische Elemente der Benutzeroberfläche eines Benutzeragenten.

In Abbildung 7.9 sind sieben Zeilen einer solchen Zusammenfassung zu sehen. Die Zeilen bestehen aus den Felder *Von (From)*, *Betreff (Subject)* und *Datum (Date)*, in genau dieser Reihenfolge, damit der Benutzer schnell ablesen kann, wer die Nachricht gesendet hat, worum es in der Nachricht geht und wann die Nachricht empfangen wurde. Die Informationen werden so aufbereitet, dass sie leicht zu erfassen sind, d.h., sie basieren auf den Inhalten der Nachrichtenfelder, ohne aber deren buchstabengetreuen Inhalt anzugeben. So erklärt sich z.B. auch, dass Leute, die versäumen, das *Betreff*-Feld auszufüllen, oft feststellen müssen, dass ihre E-Mails spät oder gar nicht beantwortet werden.

Daneben gibt es noch zahlreiche weitere Felder und Indikatoren. Die in Abbildung 7.9 zu sehenden Symbole vor dem Betreff zeigen z.B. an, dass Mails noch nicht gelesen wurden (Umschlag), einen Anhang besitzen (Büroklammer) oder – nach Einschätzung des Absenders – besonders wichtig sind (Ausrufezeichen).

Mails können auf vielfache Weise sortiert werden. Die häufigste Sortierung ist die Anordnung nach dem Empfangsdatum, mit der zuletzt empfangenen Mail ganz oben und irgendeiner Form der Kennzeichnung, ob die Mail neu ist oder bereits gelesen wurde. Sowohl die Auswahl der in der Zusammenfassung verwendeten Felder als auch die Sortierung können vom Benutzer nach eigenen Vorlieben angepasst werden.

Benutzeragenten müssen auch in der Lage sein, einkommende Nachrichten so anzuzeigen, dass der Benutzer seine E-Mail lesen kann. Vorschauen auf den Nachrichten-

inhalt (siehe Abbildung 7.9) helfen dem Benutzer zu entscheiden, welche Nachrichten er lesen möchte. Vorschauen können kleine Symbole oder Bilder enthalten, um den Inhalt der Nachricht näher zu beschreiben. Weitere Maßnahmen, die Benutzeragenten zur benutzerfreundlichen Präsentation der Nachrichten ergreifen, sind z.B. die Einpassung der Nachrichten in die Anzeigemaske sowie die Übersetzung oder Konvertierung der Inhalte in andere Formate (z.B. digitalisierte Sprache in Text).

Nachdem der Benutzer eine Nachricht gelesen hat, kann er entscheiden, was weiter mit ihr geschehen soll. Man nennt dies **Message Disposition** (Nachrichtenverwertung). Mögliche Optionen sind: die Nachricht löschen, eine Antwort senden, die Nachricht an einen anderen Benutzer weiterleiten oder die Nachricht für später aufbewahren. Die meisten Benutzeragenten können mehrere Mailboxen für einkommende E-Mail gleichzeitig verwalten, inklusive mehrerer Unterordner zum Abspeichern der E-Mail. Auf diese Weise kann der Benutzer seine Nachrichten geordnet nach Absender, Thema oder anderen Kriterien abspeichern.

Benutzeragenten können einkommende E-Mail, noch bevor der Benutzer diese liest, filtern. Häufig wird dies dazu genutzt, den Benutzeragenten die Felder und Inhalte der Nachrichten inspizieren und ihn aufgrund der gesammelten Daten sowie der Bewertungen früherer Nachrichten durch den Benutzer entscheiden zu lassen, ob es sich um Spam handeln könnte. ISPs und Firmen verwenden spezielle Software, die E-Mails als wichtig oder als Spam einstufen, sodass die Benutzeragenten die Nachrichten in die zugehörigen Mailboxen bzw. deren Unterordner einordnen können. Zugute kommt ihnen dabei, dass sie auf umfangreiche Listen bekannter Spammer zurückgreifen und die E-Mail vieler Benutzer sichten können. Erhalten Hunderte von Benutzern mehr oder weniger gleichzeitig die gleichen E-Mails, dann ist dies ein starkes Indiz, dass es sich um Spam handelt. Durch Einstufung der eingehenden Mail als „unverdächtig“ oder „wahrscheinlich Spam“ kann der Benutzeragent dem Benutzer bei der Trennung von Spreu und Weizen viel Arbeit sparen.

Und was ist mit der am weitesten verbreitete Form von Spam? Diese Art von Spam wird von sogenannten **Botnets**, Zusammenschaltungen gekaperter Rechner, generiert. Ihr Inhalt richtet sich danach, wo Sie leben. In Asien werden aktuell vor allem gefälschte Diplome angeboten, während in den USA schwerpunktmaßig billige Medikamente und andere zweifelhafte Produkte beworben werden. Daneben wimmelt es immer noch von Bankkonten aus Nigeria, auf die niemand Anspruch erhebt, sowie den unvermeidlichen Pillen, mit deren Hilfe diverse Körperteile vergrößert werden können.

Darüber hinaus kann auch der Benutzer Filterregeln erstellen. Eine solche Filterregel besteht aus einer Bedingung und einer Aktion. So kann beispielsweise eine Regel besagen, dass Nachrichten vom Chef im Ordner für direkt zu lesende Nachrichten abgelegt werden sollen, während Nachrichten von Absendern aus einer bestimmten Mailingliste in den Ordner für weniger dringliche Nachrichten gehören. In Abbildung 7.9 sind einige Beispiel für solche Ordner dargestellt. Die wichtigsten Ordner sind der Posteingang für eingehende E-Mail, die nicht anderweitig einsortiert werden konnte, und Junkmail für Nachrichten, die als Spam eingestuft wurden.

Neben der Erstellung von Ordnern und anderen Objekten bieten moderne Benutzeragenten auch zahlreiche Möglichkeiten, die Mailbox zu durchsuchen (siehe Abbildung 7.9). Leistungsfähige Suchfunktionen erlauben dem Benutzer bestimmte Nachrichten schnell zu finden – wie z.B. die Nachricht über „Bezugsquellen für Vegemite-Sandwichs“, die jemand letzten Monat geschickt hat.

Seit den Zeiten, da E-Mail noch ein reiner Dateiübertragungsmechanismus war, hat sich viel geändert. Provider stellen für Mailboxen heutzutage routinemäßig bis zu 1 GByte Speicher bereit, ausreichend Platz, dass die gespeicherten E-Mails über einen langen Zeitraum hinweg einen detaillierten Überblick über die Aktivitäten eines Benutzers geben können. Eine solche Masse an E-Mail kann faktisch nur noch dank der Unterstützung durch die Benutzeragenten und ihrer leistungsfähigen Funktionen zum Durchsuchen oder zur automatischen Verarbeitung verwaltet werden. Für Benutzer, die jährlich Tausende von Nachrichten senden oder empfangen, sind diese Tools von unschätzbarem Wert.

Ebenfalls sehr nützlich ist die Möglichkeit, Nachrichten einer automatischen Verarbeitung zuzuführen. Dies kann z.B. so aussehen, dass eingehende E-Mail an eine andere Adresse weitergeleitet wird – beispielsweise an einen Computer, der von einem kommerziellen Paging-Dienst betrieben wird. Der Benutzer wird in diesem Fall über Radio oder Satellit benachrichtigt, wobei die *Betreff*-Zeile auf seinem Pager erscheint. Ein solcher **Autoresponder** sollte auf dem Mailserver ausgeführt werden, da Benutzeragenten nicht notwendigerweise ununterbrochen ausgeführt werden und möglicherweise auch nur gelegentlich nach neuer E-Mail Ausschau halten. Aus diesem Grund können Benutzeragenten keine echte Autoresponder-Funktion anbieten. Sie können aber sehr wohl die Benutzeroberfläche für die Einrichtung und Verwaltung von automatischen Antworten anbieten – und meist tun sie dies auch.

Eine andere Form der automatischen Antwort ist der **Vacation Agent**. Das ist ein Programm, das jede ankommende Nachricht prüft und dem Absender eine mehr oder weniger geistreiche Antwort sendet, z.B.: „Hallo, ich bin bis zum 24. August in Urlaub. Ich melde mich, wenn ich wieder zurück bin.“ Sie können in einer solchen Antwort auch angeben, wie dringende Angelegenheiten zu behandeln sind, welche anderen Ansprechpartner bei bestimmten Problemen kontaktiert werden können oder was auch immer Ihnen wichtig erscheint. Die meisten Autoresponder merken sich, an wen sie die Standardantwort senden, und achten darauf, die Antwort nicht zweimal der gleichen Person zu schicken. Trotzdem hat die Verwendung dieser Agenten auch ihre Tücken. Beispielsweise ist es nicht ratsam, eine Standardantwort an die Mitglieder einer größeren Mailingliste zu senden.

Betrachten wir nun den Fall, dass ein Internetnutzer einem anderen via E-Mail eine Nachricht sendet. Zu den grundlegenden Aufgaben eines Benutzeragenten, die wir noch nicht besprochen haben, gehört auch die Erstellung von E-Mail-Nachrichten. Dies schließt das Anlegen neuer Nachrichten, das Beantworten von Nachrichten und das Senden von Nachrichten, also die Übergabe der Nachricht an den restlichen Teil des E-Mail-Systems zur Weiterbeförderung, ein. Obwohl für das Aufsetzen einer Nachricht grundsätzlich jeder beliebige Texteditor verwendet werden kann, verfügen Benutzer-

agenten üblicherweise über integrierte Texteditoren, um zusätzliche Unterstützung für die Adressierung und die zahlreichen Header-Felder, die an jede Nachricht angehängt werden, anbieten zu können. Beispielsweise kann der Benutzeragent bei der Beantwortung einer Nachricht die Adresse des Absenders aus der eingegangenen Nachricht extrahieren und automatisch an der richtigen Stelle in die Antwort einfügen. Andere weitverbreitete Funktionen sind das Anhängen eines **Signaturblocks** am Ende der Nachricht, die Überprüfung der Rechtschreibung oder die Erstellung digitaler Signaturen, um die Korrektheit der Nachricht anzuzeigen.

Nachrichten, die zur Übertragung an das E-Mail-System übergeben werden, haben ein vorgegebenes Format, das aus den Daten erzeugt wird, die dem Benutzeragenten angegeben wurden. Der für die Übertragung wichtigste Teil der Nachricht ist der Umschlag und der wichtigste Teil des Umschlags ist die Zieladresse. Diese muss in einem Format angegeben werden, das dem Benutzeragenten bekannt ist.

Grundsätzlich erwarten Benutzeragenten, dass Adressen in der Form *user@dns-address* angegeben werden. Da wir das DNS und den Aufbau von Adressen bereits weiter oben eingehend erläutert haben, sollte eine Wiederholung an dieser Stelle nicht nötig sein. Zu beachten ist allerdings, dass es auch noch andere Formen der Adressierung gibt. Vor allem **X.400**-Adressen sehen völlig anders aus als DNS-Adressen.

X.400 ist ein ISO-Standard für Nachrichten verarbeitende Systeme, der zeitweilig in Wettbewerb zu SMTP stand. SMTP hat diesen Wettbewerb überzeugend für sich entschieden. Dennoch gibt es immer noch X.400-basierte Systeme, vor allem außerhalb der USA. X.400-Adressen setzen sich aus *attribut=wert*-Paaren zusammen, die durch Schrägstriche trennt werden, wie z.B.:

```
/C=US/ST=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/
```

Diese Adresse umfasst ein Land (*C*), einen Bundesstaat (*ST*), einen Ort (*L*), die persönliche Adresse (*PA*) und einen Namen (*CN*). Daneben sind viele andere Attribute möglich, sodass man eine E-Mail auch an jemanden senden kann, dessen exakte E-Mail-Adresse man nicht weiß, sofern man andere Attribute kennt (z. B. Firma und Berufsbezeichnung).

Obwohl die X.400-Bezeichnungen viel unhandlicher sind als die DNS-Bezeichnungen, fällt dies in der Praxis kaum ins Gewicht, da die meisten Benutzeragenten mit anwenderfreundlichen Aliasen arbeiten (manchmal als Spitzname bezeichnet), sodass die Benutzer einfach den Namen einer Person eingeben oder auswählen können und die korrekte E-Mail-Adresse erhalten. Folglich ist es nicht erforderlich, diese seltsamen Zeichenfolgen einzutippen.

Der letzte Punkt, den wir in Zusammenhang mit dem Senden von E-Mail ansprechen möchten, sind die Mailinglisten, die es dem Benutzer gestatten, die gleiche Nachricht mit einem Befehl an mehrere Empfänger zu senden. Es gibt zwei Möglichkeiten, wie Mailinglisten verwaltet werden können. Die erste Möglichkeit ist die lokale Verwaltung durch den Benutzeragenten. In diesem Fall kann der Benutzeragent einfach eine eigene Nachricht an jeden der aufgeführten Empfänger senden.

Alternativ kann die Liste auf einem entfernten Rechner durch einen Nachrichtenübertragungsagenten verwaltet werden. Die Nachrichten werden dann an das Nachrichtenübertragungssystem übergeben, wodurch es möglich ist, dass mehrere Benutzer Nachrichten an die Empfänger in der Liste entsenden. Hat z.B. eine Gruppe von Ornithologen eine Mailingliste namens *birders* auf dem Übertragungsagenten *meadowlark.arizona.edu* installiert, so wird jede an *birders@meadowlark.arizona.edu* adressierte Nachricht an die Universität von Arizona weitergeleitet und dort in die einzelnen Nachrichten für alle Mitglieder der Mailingliste aufgeteilt, gleichgültig, wo sie sich befinden. Die Benutzer dieser Mailingliste können nicht erkennen, dass es sich um eine Mailingliste handelt. Es könnte sich genauso gut um die persönliche Mailbox von Prof. Gabriel O. Birders handeln.

7.2.3 Nachrichtenformate

Wir wenden uns nun von der Benutzeroberfläche ab und dem Format der E-Mail-Nachrichten zu. Die Nachrichten, die der Benutzeragent versendet, müssen ein standardisiertes Format haben, um vom Nachrichtenübertragungsagenten verarbeitet werden zu können. Wir beginnen mit der grundlegenden ASCII-E-Mail gemäß RFC 5322, der letzten Überarbeitung des ursprünglichen Internetnachrichtenformats, wie es in RFC 822 beschrieben wurde. Danach betrachten wir die auf diesem Format aufbauenden Multimedia-Erweiterungen.

RFC 5322 – das Internetnachrichtenformat

Nachrichten bestehen aus einem einfachen Umschlag (in RFC 5321 im Kontext von SMTP beschrieben), einigen Header-Feldern, einer Leerzeile und dem Nachrichtentext. Jedes Header-Feld besteht (logisch) aus einer einzelne Zeile mit ASCII-Text, die den Feldnamen, ein Doppelpunkt und meist einen Wert enthält. Das ursprüngliche RFC 822 wurde vor Jahrzehnten entwickelt und trennte nicht klar zwischen Umschlag- und Header-Feldern. Obwohl dies in RFC 5322 korrigiert wurde, war aufgrund der weiten Verbreitung keine völlige Erneuerung möglich. Im Normalfall erstellt der Benutzeragent eine Nachricht und übergibt sie an den Nachrichtenübertragungsagenten, der dann mithilfe der Daten diverser Header-Felder den Umschlag zusammenstellt – wir haben es in der Praxis also mit einer etwas veralteten Art der Kombination von Nachricht und Umschlag zu tun.

Die wichtigsten Header-Felder in Bezug auf die Nachrichtenübertragung sind in ►Abbildung 7.10 aufgeführt. Das Feld *To:* gibt die DNS-Adresse des primären Empfängers an. Es sind auch mehrere Empfänger zulässig. Das Feld *Cc:* gibt die Adressen etwaiger sekundärer Empfänger an. Bei der Zustellung wird zwischen primären und sekundären Empfängern kein Unterschied gemacht. Es handelt sich um einen rein psychologischen Unterschied, der für die Benutzer, nicht aber für das System wichtig ist. Die Abkürzung *Cc:* (für Carbon Copy, zu Deutsch „Durchschlag“) ist etwas altmodisch und nicht ganz passend, da Computer ja kein Kohlepapier verwenden, um Durchschläge anzufertigen. Das Feld *Bcc:* (Blind Carbon Copy) hat die gleiche Bedeutung wie das Feld *Cc:*, nur dass diese Zeile in allen Kopien, die an die primären und sekundären Empfänger gesendet werden, gelöscht wird. Mit dieser Funktion kann der

Benutzer Kopien einer Nachricht an Dritte senden, ohne dass die primären und sekundären Empfänger davon wissen.

| Header | Bedeutung |
|--------------|--|
| To: | E-Mail-Adresse(n) des/der primären Empfänger(s) |
| Cc: | E-Mail-Adresse(n) des/der sekundären Empfänger(s) |
| Bcc: | E-Mail-Adresse(n) für blinde Kopien an Dritte |
| From: | Ersteller der Nachricht |
| Sender: | E-Mail-Adresse des tatsächlichen Absenders |
| Received: | Zeile, die von jedem Übertragungsagenten auf dem Weg eingefügt wird |
| Return-Path: | Kann verwendet werden, um einen Pfad zurück zum Absender zu bezeichnen |

Abbildung 7.10: Header-Felder nach RFC 5322 für die Nachrichtenübertragung.

Die nächsten beiden Felder *From:* und *Sender:* geben an, wer die Nachricht geschrieben bzw. gesendet hat. Das muss nicht unbedingt die gleiche Person sein. Eine Geschäftsfrau schreibt z.B. eine Nachricht, die sie dann von ihrem Sekretär versenden lässt. In diesem Fall wird die Geschäftsfrau im Feld *From:* und der Sekretär im Feld *Sender:* genannt. Das Feld *From:* muss ausgefüllt werden, das Feld *Sender:* kann leer bleiben, wenn es mit *From:* identisch ist. Die Informationen dieser Felder werden benötigt, wenn die Nachricht nicht zustellbar ist und an den Absender zurückgeschickt werden muss.

Auf dem Weg durch das E-Mail-System wird die Nachricht von jedem Nachrichtenübertragungsagenten, der die Nachricht entgegennimmt, um eine Zeile *Received:* erweitert. Diese Zeile enthält die Identität des Agenten, Datum und Uhrzeit des Empfangs der Nachricht und weitere Informationen, die Auskunft über Fehler im Routing-System geben können.

Das Feld *Return-Path:* wird vom letzten Nachrichtenübertragungsagenten eingefügt und soll angeben, wie die Nachricht an den Sender zurückgeschickt werden kann. Theoretisch kann diese Information aus den Angaben in den *Received:*-Headern zusammengestellt werden (es fehlt dann nur noch der Name der Mailbox des Absenders). Es wird aber selten vollständig ausgefüllt und enthält meist nur die Adresse des Absenders.

Zusätzlich zu den in Abbildung 7.10 aufgeführten Feldern können RFC-5322-Nachrichten noch eine Vielzahl von Header-Feldern enthalten, die von den Benutzeragenten oder den Benutzern verwendet werden. Die am weitesten verbreiteten Felder dieser Kategorie sind in ►Abbildung 7.11 aufgeführt. Die meisten sind selbsterklärend, sodass wir sie hier nicht im Detail behandeln.

| Header | Bedeutung |
|--------------|--|
| Date: | Datum und Uhrzeit, wann die Nachricht gesendet wurde |
| Reply-To: | E-Mail-Adresse, an die Antworten gesendet werden können |
| Message-Id: | Eindeutige Kennung der Nachricht für eine spätere Bezugnahme |
| In-Reply-To: | Kennung der Nachricht, der diese Antwort gilt |
| References: | Andere relevante Nachrichtenkennungen |
| Keywords: | Vom Benutzer gewählte Schlüsselwörter |
| Subject: | Kurzer einzeiliger Betreff der Nachricht |

Abbildung 7.11: Weitere Felder für den Nachrichten-Header nach RFC 5322.

Das Feld *Reply-To*: wird benutzt, wenn weder der Verfasser noch der Absender der Nachricht die Antwort erhalten wollen. Ein Marketingmanager schreibt beispielsweise eine E-Mail, mit der er seinen Kunden ein neues Produkt ankündigt. Die Nachricht wird von der Sekretärin gesendet, im Feld *Reply-To*: wird aber der Vertriebsleiter angegeben, der dann die Fragen und Bestellungen entgegennehmen kann. Das Feld ist auch nützlich, wenn der Sender zwei E-Mail-Adressen hat und möchte, dass die Antwort an die andere Adresse geht.

Das Feld *Message-Id*: ist eine automatisch erzeugte Kennnummer, mit deren Hilfe a) Nachrichten miteinander verknüpft werden können (beispielsweise indem die Nummer im *Reply-To*-Feld angegeben wird) und b) doppelte Zustellungen vermieden werden.

RFC 5322 statuiert explizit, dass Benutzer eigene Header für ihre persönliche Verwendung erzeugen können. Per Konvention beginnen dieser Header seit RFC 822 mit der Zeichenfolge *X-*, wobei gewährleistet ist, dass die offiziellen Header auch in Zukunft keine Namen haben werden, die mit *X-* beginnen, um Konflikte zwischen offiziellen und privaten Headern zu vermeiden. Manchmal kommen besonders erfunderische Studenten auf Felder wie *X-Frucht-des-Tages*: oder *X-Krankheit-der-Woche*: – diese sind zwar zulässig, nur leider wenig erhelltend.

Im Anschluss an die Header folgt der eigentliche Nachrichtentext. Die Benutzer können hier eingeben, was sie wollen. Einige Leute beenden ihre Nachrichten mit ausgefeilten Unterschriften, einfachen ASCII-Cartoons, Zitaten von mehr oder weniger großen Koryphäen, politischen Statements und anderen Anhängseln aller Art (z.B. „Die Firma XYZ ist nicht für die von mir geäußerten Meinungen verantwortlich; sie versteht sie ja nicht einmal.“)

MIME – Multipurpose Internet Mail Extensions

Als das ARPANET seinen Anfang nahm, bestand E-Mail ausschließlich aus in Englisch verfassten Textnachrichten im ASCII-Format. Für diesen Kontext war RFC 822 absolut ausreichend. Der Standard spezifizierte die Header, überließ aber den Inhalt

gänzlich den Benutzern. In den 1990ern führte die weltweite Nutzung des Internets und der Wunsch, reichere, multimediale Inhalte über das E-Mail-System zu übertragen, dazu, dass dieser Ansatz nicht mehr länger zeitgemäß war. Problematisch war unter anderem das Senden und Empfangen von:

- Nachrichten in Sprachen mit Akzenten (z.B. Französisch und Deutsch)
- Nachrichten in nicht lateinischen Sprachen (z.B. Hebräisch und Russisch)
- Nachrichten in Sprachen ohne Alphabet (z.B. Chinesisch und Japanisch)
- Nachrichten, die überhaupt keinen Text enthalten (z.B. Audio und Video)

Die Lösung brachte die Ausarbeitung der **MIME-Erweiterungen** (*Multipurpose Internet Mail Extensions*). MIME wird heute vielfach genutzt, nicht nur für E-Mail-Nachrichten, die über das Internet gesendet werden, sondern auch ganz allgemein zur Beschreibung von Inhalten, wie sie z.B. beim Surfen im Internet auftreten. MIME selbst ist in den RFCs 2045-2047, 4288, 4289 und 2049 beschrieben.

Das Grundkonzept von MIME besteht darin, das Format RFC 822 (der Vorgänger zu RFC 5322; RFC 5322 gab es noch nicht, als MIME vorgeschlagen wurde) weiterhin zu verwenden, jedoch den Nachrichtentext um zusätzliche Strukturen zu erweitern und die Codierregeln für die Übertragung von Nicht-ASCII-Nachrichten zu definieren. Die Treue zu RFC 822 gewährleistete, dass MIME-Nachrichten mit den vorhandenen E-Mail-Programmen und -Protokollen (damals basierend auf RFC 821, heute RFC 5321) übertragen werden konnten. Es mussten lediglich die sendenden und empfangenden Programme ausgetauscht werden, was die Benutzer selbst vornehmen konnten.

MIME definiert fünf neue Nachrichten-Header (► Abbildung 7.12). Der erste informiert den Benutzeragenten, der die Nachricht erhält, dass er es mit einer MIME-Nachricht zu tun hat und welche MIME-Version er benutzt. Bei Nachrichten ohne den Header **MIME-Version**: wird angenommen, dass es sich um englischen ASCII-Text handelt, der dann als solcher verarbeitet wird.

| Header | Bedeutung |
|----------------------------|--|
| MIME-Version: | Bezeichnet die MIME-Version |
| Content-Description: | Vom Benutzer lesbare Zeichenkette, die den Inhalt der Nachricht andeutet |
| Content-Id: | Eindeutiger Identifikator |
| Content-Transfer-Encoding: | Gibt an, wie der Nachrichteninhalt für die Übertragung verpackt wurde |
| Content-Type: | Bezeichnet den Typ und das Format der Nachricht |

Abbildung 7.12: Nachrichten-Header, die gemäß MIME hinzugefügt wurden.

Der Header **Content-Description:** (Inhaltsbeschreibung) ist eine ASCII-Zeichenkette, die auf den Inhalt der Nachricht hinweist. Dieser Header ist erforderlich, damit der Empfänger beurteilen kann, ob es sich lohnt, die Nachricht zu decodieren und zu lesen. Sagt die Zeichenkette: „Foto von Barbaras Hamster“ und der Empfänger ist kein

großer Hamster-Fan, dann ist es eher wahrscheinlich, dass die Nachricht verworfen wird, als dass sie in ein hochauflötes Farbfoto decodiert wird.

Der Header *Content-Id*: (Inhaltskennung) bezeichnet den Inhalt. Er benutzt das gleiche Format wie der Standard-Header *Message-Id*.

Der Header *Content-Transfer-Encoding*: (Inhaltsübertragungscodierung) gibt an, wie die Nachricht für die Übertragung über das Netz aufbereitet wurde. Zu der Zeit als MIME ausgearbeitet wurde, bestand eines der größten Probleme darin, dass die Nachrichtenübertragungsprotokolle (SMTP) erwarteten, dass die Nachrichten aus ASCII-Zeichen bestanden und keine Zeile mehr als 1 000 Zeichen enthält. ASCII-Zeichen verwenden von jedem 8-Bit-Byte nur 7 Bit. Binäre Daten wie z.B. ausführbare Programme oder Bilder verwenden dagegen, ebenso wie erweiterte Zeichensätze, alle 8 Bit eines Byte. Da es keine Garantie gab, dass diese Daten korrekt übertragen werden konnten, bedurfte es irgendeines Verfahrens, das dafür sorgte, dass die Daten wie eine reguläre ASCII-Nachricht aussahen. Seit der Entwicklung von MIME gibt es daher SMTP-Erweiterungen, die die Übertragung von binären 8-Bit-Daten gestatten – auch wenn es selbst heute noch keine Gewähr dafür gibt, dass uncodierte binäre Daten korrekt über das Mailsystem weitergeleitet werden.

MIME kennt fünf Codierschemata für die Übertragung, plus ein Hintertürchen für benutzerdefinierte Schemata – falls ein solches benötigt werden sollte. Das einfachste Codierschema ist die Übertragung reiner ASCII-Textnachrichten. ASCII-Zeichen verwenden 7 Bits und können direkt vom E-Mail-Protokoll übertragen werden, sofern keine Zeile mehr als 1 000 Zeichen umfasst.

Das nächste Schema ist nahezu identisch zu seinem Vorgänger, verwendet aber 8-Bit-Zeichen, d.h., es sind sämtliche Werte von 0 bis 255 einschließlich erlaubt. Nachrichten, die auf der 8-Bit-Codierung basieren, müssen sich ebenfalls an die Standardbegrenzung der Zeilenlänge halten.

Dann gibt es noch die Nachrichten, die eine echte Binärcodierung verwenden. Diese bestehen aus beliebigen binären Dateien, die nicht nur alle 8 Bit verwenden, sondern auch Zeilen beliebiger Länge haben (ohne Berücksichtigung des Zeilenlimits von 1 000 Zeichen). Ausführbare Programme fallen z.B. in diese Kategorie. Heutzutage können Mailserver selbstständig aushandeln, ob sie Daten binärcodiert (bzw. 8-Bit-codiert) senden oder ob sie auf ASCII zugreifen, wenn die Erweiterung von keinem der beiden Enden unterstützt wird.

Die ASCII-Codierung für binäre Daten wird als **Base64-Codierung** bezeichnet. Bei diesem Schema werden Gruppen von je 24 Bit in vier 6-Bit-Einheiten zerlegt, von denen jede Einheit als zulässiges ASCII-Zeichen übertragen wird. Die Codierung lautet „A“ für 0, „B“ für 1 usw., gefolgt von den 26 Kleinbuchstaben, den zehn Ziffern sowie + und / für 62 bzw. 63. Die Folgen == und = geben an, dass die letzte Gruppe nur 8 bzw. 16 Bit enthielt. Zeilenvorschub und Zeilenrücklauf werden ignoriert. Sie können also willkürlich eingefügt werden, um die Zeilen kurz zu halten. Mit diesem Schema kann beliebiger Binärtext einigermaßen zuverlässig, wenn auch nicht sonderlich effizient übertragen werden. Vor dem Aufkommen von Mailservern, die Binärdaten direkt verarbeiten, war diese Codierung sehr populär und sie ist immer noch weitverbreitet.

Für Nachrichten, die fast gänzlich aus ASCII bestehen, zusätzlich aber ein paar Nicht-ASCII-Zeichen enthalten, ist die Base64-Codierung etwas ineffizient. In solchen Fällen wird die sogenannte **Quoted-Printable-Codierung** verwendet. Das ist eigentlich 7-Bit-ASCII, aber alle Zeichen über 127 werden als Gleichheitszeichen, gefolgt vom Wert des Zeichens (als zwei hexadezimale Ziffern), codiert. Steuerzeichen, diverse Satzzeichen und mathematische Symbole sowie vorangehender oder nachgestellter Leerraum werden ebenfalls auf diese Weise codiert.

Gibt es stichhaltige Gründe, keines dieser Schemata anzuwenden, besteht zu guter Letzt die Möglichkeit, im Header *Content-Transfer-Encoding*: eine benutzerdefinierte Codierung anzugeben.

Der letzte Header aus Abbildung 7.12 ist eigentlich der interessanteste. Er gibt die Art des Nachrichteninhalts an und hat über das E-Mail-System hinaus an Einfluss gewonnen. Beispielsweise werden Inhalte, die aus dem Web heruntergeladen werden, mittels MIME-Typen klassifiziert, damit der Browser weiß, wie er die Daten anzeigen soll. Gleichermaßen gilt für Inhalte, die über Streaming Media oder Echtzeit-Transportsysteme wie Voice-over-IP gesendet werden.

Anfänglich wurden in RFC 1521 sieben MIME-Typen definiert. Zu jedem dieser Typen gibt es einen oder mehreren Untertypen, wobei die Typ- und Untertypangaben durch Schrägstriche getrennt werden, wie z.B. in „Content-Type: video/mpeg“. Im Laufe der Zeit kamen immer neue Arten von Inhalten auf und weitere Einträge wurden hinzugefügt. Die von der IANA-Organisation verwaltete Liste der Typen und Untertypen ist online unter www.iana.org/assignments/media-types verfügbar.

In ▶ Abbildung 7.13 sind die einzelnen Typen zusammen mit Beispielen für gängige Untertypen aufgelistet. Lassen Sie uns die Liste, beginnend mit *text*, kurz durchgehen. Die Kombination *text/plain* ist für gewöhnliche Nachrichten, die so angezeigt werden können, wie sie eingegangen sind, d.h. ohne Codierung und ohne weitere Verarbeitung. Diese Option ermöglicht die Übertragung gewöhnlicher Nachrichten in MIME mit nur wenigen zusätzlichen Headern. Mit der zunehmenden Verbreitung des Webs wurde ein neuer Untertyp namens *text/html* hinzugefügt (in RFC 2854), damit Webseiten in RFC-822-E-Mail versendet werden können. Für XML (eXtensible Markup Language) wurde in RFC 3023 der Untertyp *text/xml* definiert. XML-Dokumente haben sich im Zuge der Entwicklung des Webs stark ausgebreitet. HTML und XML werden in Abschnitt 7.3 besprochen.

Der nächste MIME-Typ ist *image* und wird zur Übertragung von Standbildern benutzt wird. Zum Speichern und Übertragen von Bildern werden heute viele verschiedene Formate benutzt, sowohl mit als auch ohne Komprimierung. Einige dieser Formate, wie z.B. GIF, JPEG, PNG und TIFF, sind in beinahe alle Browser integriert. Daneben gibt es noch viele weitere Formate – und passende Untertypen.

| Typ | Untertyp | Beschreibung |
|-------------|--------------------------------------|---|
| text | plain, html, xml, css | Text in verschiedenen Formaten |
| image | gif, jpeg, tiff, png | Bilder |
| audio | basic, mpeg, mp4 | Audiodateien |
| video | mpeg, mp4, quicktime | Filme |
| model | vrm | 3-D-Modelle |
| application | octet-stream, pdf, javascript, zip | Daten, die von Anwendungen erzeugt werden |
| message | http, rfc822 | Eingeschlossene Nachricht |
| multipart | mixed, alternative, parallel, digest | Kombination verschiedener Typen |

Abbildung 7.13: MIME-Typen und ausgewählte Untertypen.

Die Typen *audio* und *video* dienen der Unterstützung von Ton und beweglichen Bildern. Beachten Sie, dass *video* nur die visuellen Informationen, nicht aber den Begleitton enthält. Wird ein Film mit Ton übertragen, müssen beide Teile, d.h. Audio und Video, je nach dem verwendeten Codiersystem getrennt übertragen werden. Das erste definierte Videoformat war MPEG und wurde von einer Gruppe mit dem bescheidenen Namen „Moving Picture Experts Group“ ersonnen. Etliche weitere Formate sind im Laufe der Zeit hinzugekommen. Zusätzlich zu *audio/basic* wurde in RFC 3003 ein neuer Audiotyp, *audio/mpeg*, aufgenommen, um das Versenden von MP3-Audiodateien über E-Mail zu ermöglichen. Die Typen *video/mp4* und *audio/mp4* stehen für Video- und Audiodaten, die in dem moderneren MPEG 4-Format abgespeichert sind.

Der *model*-Typ ist erst nach den anderen Inhaltstypen hinzugekommen und soll 3-D-Modelldaten beschreiben. Allzu große Verbreitung hat er bisher allerdings nicht gefunden.

Der Typ *application* ist ein Tausendsassa für Formate, die externe Verarbeitung benötigen und von keinem der anderen Typen abgedeckt werden. Die beispielhaft aufgelisteten Untertypen *pdf*, *javascript* und *zip* stehen für PDF-Dokumente, JavaScript-Programme bzw. ZIP-Archive. Benutzeragenten, die solche Inhalte empfangen, verwenden Bibliotheken von Drittanbietern oder externe Programme, um den jeweiligen Inhalt anzuzeigen. Die Anzeige kann, muss aber nicht in die Benutzeroberfläche des Benutzeragenten integriert sein.

Dank der MIME-Typen können Benutzeragenten die eigene Funktionalität jederzeit so ausbauen, dass sie neu aufkommende Anwendungsdaten verarbeiten können. Dies ist ein entscheidender Vorteil. Die Kehrseite der Medaille ist, dass viele neue Inhaltstypen von Anwendungen ausgeführt oder interpretiert werden, die eine Gefahrenquelle darstellen. Es ist heute allgemein bekannt, dass die Ausführung eines unbekannten Programms, welches von einem „Freund“ via E-Mail geschickt wurde, ein großes Sicherheitsrisiko darstellt. Das Programm könnte den Teilen des Rechners, auf die es Zugriff hat, jede erdenkliche Art von Schaden zufügen, vor allem, wenn es

Dateien lesen und schreiben und das Netzwerk verwenden kann. Weniger bekannt ist, dass Dokumentformate das gleiche Gefahrenpotenzial darstellen. Dies liegt daran, dass Formate wie z.B. PDF im Grunde voll ausgereifte, maskierte Programmiersprachen sind. Sie werden zwar interpretiert und auf die nötigsten Zugriffsrechte beschränkt, doch Bugs in den Interpretern können bösartigen Dokumenten Schlupflöcher öffnen, durch die sie den Restriktionen entkommen können.

Neben den bisher aufgeführten Beispielen gibt es noch viele weitere *application*-Untertypen, was angesichts der großen Zahl an Anwendungen und Programmen nicht verwundert. Als letzte Zuflucht, für Fälle, in denen sich kein wirklich passender Untertyp findet, bezeichnet *octet-stream* eine reine Folge nicht interpretierter Bytes. Beim Empfang eines solchen Datenstroms schlagen viele Benutzeragenten dem Benutzer vor, den Inhalt in eine Datei zu kopieren. Die weitere Verarbeitung obliegt dann dem Benutzer.

Die beiden letzten Typen beziehen sich auf den Aufbau der Nachrichten. Der Typ *message* ermöglicht die volle Kapselung einer Nachricht in einer anderen. Dieses Modell ist z.B. nützlich zum Weiterleiten von E-Mail. Wird eine komplette Nachricht in RFC 822 in eine äußere Nachricht gekapselt, sollte der Untertyp *rfc822* verwendet werden. Auch HTML-Dokumente werden üblicherweise gekapselt. Der Untertyp *partial* ermöglicht, eine gekapselte Nachricht in verschiedene Teile zu unterteilen und diese getrennt zu versenden (z.B. wenn die gekapselte Nachricht sehr lang ist). Anhand von Parametern können alle Teile am Ziel wieder in der richtigen Reihenfolge zusammengesetzt werden.

Der Typ *multipart* schließlich ist für Nachrichten, die aus mehreren Teilen bestehen, wobei Anfang und Ende der einzelnen Teile deutlich abgegrenzt sind. Bei Verwendung des Untertyps *mixed* kann jeder Teil von einem anderen Typ sein, ohne dass dafür die Struktur näher angegeben werden muss. Viele E-Mail-Programme erlauben dem Benutzer, einen oder mehrere Anhänge an eine Textnachricht anzufügen. Diese Anhänge werden unter Verwendung des Typs *multipart* gesendet.

Im Gegensatz zu dem Untertyp *mixed* erlaubt der Untertyp *alternative* die gleiche Nachricht mehrfach einzubinden, allerdings in unterschiedlichen Medienarten. Auf diese Weise kann eine Nachricht z.B. in einfachem ASCII, in HTML und als PDF versendet werden. Ein gut konzipierter Benutzeragent würde diese Nachricht dann so anzeigen, wie es den Präferenzen des Benutzers entspricht. Die erste Wahl wäre dann vermutlich PDF, die zweite HTML. Funktioniert beides nicht, kann immer noch der einfache ASCII-Text angezeigt werden. Die Teile sollten vom Einfachsten zum Komplexesten angeordnet werden, sodass auch Benutzer, die noch mit Prä-MIME-Benutzeragenten arbeiten, die Nachricht zu sehen bekommen (sogar ein Prä-MIME-Benutzeragent kann reinen ASCII-Text lesen).

Der Untertyp *alternative* kann auch für mehrere Sprachen verwendet werden. In diesem Zusammenhang kann der Stein von Rosetta als eine frühe *multipart/alternative*-Nachricht gesehen werden.

Von den beiden verbliebenen Beispiel-Untertypen ist der Untertyp *parallel* für Fälle gedacht, wenn alle Teile gleichzeitig „angezeigt“ werden müssen. Das ist z.B. bei Fil-

men mit einem Ton- und einem Bildkanal der Fall. Filme sind besser abspielbar, wenn diese zwei Kanäle parallel anstatt nacheinander wiedergegeben werden. Der Untertyp *digest* wird verwendet, wenn viele Nachrichten zu einer zusammengesetzten verpackt werden. Einige Diskussionsgruppen im Internet sammeln z.B. Nachrichten von Teilnehmern und senden sie dann als einzelne Nachricht vom Typ *multipart/digest* aus.

Als ein Beispiel dafür, wie MIME-Typen für E-Mail-Nachrichten verwendet werden, zeigt ►Abbildung 7.14 eine multimediale Mail – in diesem Fall ein Geburtstagsglückwunsch, der in alternativen Varianten als Text und als Lied übertragen wird. Ist der Rechner des Empfängers für Audioausgaben konfiguriert, dann spielt der Benutzeragent die Audiodatei ab. Im Beispiel wird der Sound in Form einer Referenz als *message/external-body*-Untertyp übermittelt. Der Benutzeragent muss die Audiodatei *birthday.snd* also zuerst via FTP herunterladen. Gibt es aus dem Rechner des Empfängers keine Audiounterstützung, so wird der Text in Totenstille auf dem Bildschirm angezeigt. Die beiden Teile sind durch zwei Bindestriche getrennt, gefolgt von der (benutzerdefinierten) Zeichenkette, die im Parameter *boundary* spezifiziert ist.

```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b> Bob </b><br>
Happy birthday to you</p>

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
access-type="anon-ftp";
site="bicycle.cs.washington.edu";
directory="pub";
name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Abbildung 7.14: Eine mehrteilige Nachricht, die alternative HTML- und Audiodaten enthält.

Beachten Sie, dass der Header *Content-Type* in diesem Beispiel an drei Stellen vorkommt. Auf der obersten Ebene zeigt er an, dass die Nachricht aus mehreren Teilen besteht. Innerhalb jedes dieser Teile werden der Typ und der Untertyp des Teils angegeben. Im Code des zweiten Teils muss dem Benutzeragenten zudem mitgeteilt werden, welche externe Datei zu laden ist. Um Sie auf diese leicht abweichende Verwendung

aufmerksam zu machen, haben wir den zugehörigen Header vollständig in Kleinbuchstaben abgedruckt – rein technisch wird aber bei den Headern nicht zwischen Groß- und Kleinschreibung unterschieden. Der Header *Content-transfer-encoding* wird für jeden externen Inhalt benötigt, der nicht als 7-Bit-ASCII codiert ist.

7.2.4 Nachrichtenübertragung

Nachdem wir uns mit den Benutzeragenten und den E-Mail-Nachrichten bekannt gemacht haben, sind wir nun so weit, dass wir uns ansehen können, wie die Nachrichtenübertragungsagenten Nachrichten vom Absender zum Empfänger leiten. Das hierfür zuständige Protokoll ist SMTP.

Die einfachste Art, Nachrichten zu übertragen, ist, eine Transportverbindung vom Quellrechner zum Zielrechner aufzubauen und dann die Nachricht einfach über diese Verbindung zu übertragen. Das ist das Prinzip, nach dem SMTP ursprünglich arbeitete. Im Laufe der Jahre haben sich jedoch zwei unterschiedliche Verwendungen für SMTP ausgebildet. Die erste ist die **Mail Submission** (Schritt 1 in der in Abbildung 7.7 dargestellten Mailarchitektur). Auf diese Weise übergeben Benutzeragenten Nachrichten zur Übertragung und an das E-Mail-System. Die andere Art der Verwendung ist die Weiterleitung von Nachrichten zwischen Nachrichtenübertragungsagenten (Schritt 2 in Abbildung 7.7), wodurch die E-Mail in einem Sprung den ganzen Weg vom sendenden zum empfangenden Nachrichtenübertragungsagenten übertragen wird. Die endgültige Zustellung erfolgt unter Verwendung anderer Protokolle, zu denen wir am Ende des nächsten Abschnitts kommen.

In diesem Abschnitt werden wir die Grundlagen des SMTP-Protokolls und seiner Erweiterungsmechanismen vorstellen. Anschließend beleuchten wir die Unterschiede zwischen Mail Submission und Nachrichtenübertragung.

SMTP (Simple Mail Transfer Protocol) und Erweiterungen

Im Internet wird E-Mail zugestellt, indem die Quelle eine TCP-Verbindung zu Port 25 des Zielrechners aufbaut. Das Abhören dieses Ports übernimmt ein Mailserver, der **SMTP (Simple Mail Transfer Protocol)** spricht. Dieser Server akzeptiert ankommende Verbindungen, abhängig vom Ergebnis diverser Sicherheitschecks, und nimmt die zu zustellenden Nachrichten entgegen. Kann eine Nachricht nicht zugestellt werden, so wird ein Fehlerbericht mit dem ersten Teil der unzustellbaren Nachricht ausgegeben und an den Absender zurückgeschickt.

SMTP ist ein einfaches ASCII-Protokoll. Dies ist kein Nachteil, im Gegenteil, es ist ein bewusst gewähltes Merkmal, denn auf ASCII-Text basierende Protokolle sind leichter zu entwickeln, zu testen und zu debuggen. Man kann sie mittels eigenhändig abgeschickter Befehle testen und die Aufzeichnungen der Nachrichten sind leicht zu lesen. Auf der Anwendungsebene arbeiten mittlerweile fast alle Internetprotokolle nach diesem Prinzip (so z.B. auch HTTP).

Lassen Sie uns nun eine einfache Nachrichtenübertragung zwischen Mailservern nachverfolgen, die eine Nachricht zustellen. Nach dem Aufbau der TCP-Verbindung zu Port 25 wartet der sendende Rechner (Client), bis der empfangende Rechner (Server) zuerst mit der Kommunikation beginnt. Der Server beginnt durch Aussenden einer Textzeile, durch die er sich identifiziert und mitteilt, ob er E-Mail annehmen kann. Andernfalls gibt der Client die Verbindung frei und versucht es später noch einmal.

Ist der Server bereit, E-Mail entgegenzunehmen, kündigt der Client an, von wem die E-Mail kommt und an wen sie gerichtet ist. Existiert der Empfänger am Ziel, gibt der Server dem Client das Startzeichen zum Senden. Dann sendet der Client die Nachricht und der Server bestätigt sie. Im Allgemeinen sind keine Prüfsummen erforderlich, weil TCP zuverlässige Byteströme befördert. Sind mehrere Nachrichten zu versenden, werden sie nacheinander übertragen. Wurden alle Nachrichten in beiden Richtungen ausgetauscht, dann wird die Verbindung freigegeben. ► Abbildung 7.15 zeigt einen Beispieldialog zum Senden der Nachricht aus Abbildung 7.14, inklusive der von SMTP verwendeten numerischen Codes. Die vom Client gesendeten Zeilen sind durch ein vorangestelltes C: markiert, die vom Server gesendeten Zeilen beginnen mit S:.

Der erste Befehl des Clients lautet tatsächlich *HELO*. Unter den verschiedenen Möglichkeiten, *HELO* auf vier Zeichen zu kürzen, hat die Variante *HELO* gewisse Vorteile, die sie vor ihren größten Mitkonkurrenten auszeichnet. Warum überhaupt alle Befehle vier Zeichen lang sein müssen, ist heute nicht mehr nachzuvollziehen.

In Abbildung 7.15 wird die Nachricht nur an einen Empfänger gesendet; deshalb kommt nur ein *RCPT*-Befehl vor. Durch mehrere solche Befehle kann eine Nachricht an mehrere Empfänger gesendet werden. Jeder wird einzeln bestätigt oder abgewiesen. Auch wenn einige Empfänger die Nachricht abweisen (weil sie am Ziel nicht existieren), kann sie an die übrigen gesendet werden.

Während die Syntax für die vier Zeichen langen Client-Befehle genau festgelegt ist, wird die Syntax der Antworten etwas lockerer gehandhabt. Im Grunde zählt nur der numerische Code. Jede Implementierung kann beliebige Zeichenketten nach dem Code anfügen.

```
S: 220 ee.uwa.edu.au SMTP service ready
C: HELO abcd.com
                               S: 250 cs.washington.edu says hello to ee.uwa.edu.au
C: MAIL FROM: <alice@cs.washington.edu>
                               S: 250 sender ok
C: RCPT TO: <bob@ee.uwa.edu.au>
                               S: 250 recipient ok
C: DATA
                               S: 354 Send mail; end with "." on a line by itself
C: From: alice@cs.washington.edu
C: To: bob@ee.uwa.edu.au
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@ee.uwa.edu.au>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Earth orbits sun integral number of times
C:
```

```

C: This is the preamble. The user agent ignores it. Have a nice day.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/html
C:
C: <p>Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <b>Bob</b>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C: access-type="anon-ftp";
C: site="bicycle.cs.washington.edu";
C: directory="pub";
C: name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
S: 250 message accepted
C: QUIT
S: 221 ee.uwa.edu.au closing connection

```

Abbildung 7.15: Übertragung einer Nachricht von *alice@cs.washington.edu* an *bob@ee.uwa.edu.au*.

Obwohl SMTP gut funktioniert, gibt es diverse Beschränkungen. Beispielsweise gibt es keine Authentifizierung. Der *FROM*-Befehl in unserem Beispiel könnte also jede beliebige Senderadresse angeben. Eine Sicherheitslücke, von der vor allem die Sender von Spammnachrichten profitieren. Eine weitere Beschränkung ist, dass SMTP keine binären Daten, sondern ASCII-Nachrichten überträgt. Um dies auszugleichen, bedurfte es der MIME-Inhaltsübertragungscodierung Base64. E-Mail-Übertragungen mit dieser Codierung können die zur Verfügung stehende Bandbreite aber nicht effizient nutzen, was sich vor allem bei großen Nachrichten bemerkbar macht. Ein dritter Nachteil ist, dass SMTP Nachrichten als Klartext sendet, Verschlüsselungsmechanismen zum Schutz der Privatsphäre gegen neugierige Lauscher sind nicht vorgesehen.

Um diese und etliche weitere mit der Nachrichtenverarbeitung verbundene Probleme angehen zu können, wurde SMTP ein Erweiterungsmechanismus hinzugefügt. Dieser Mechanismus ist obligatorischer Bestandteil des RFC-5321-Standards. Die Verwendung von SMTP mit Erweiterungen wird als **ESMTP** (*Extended SMTP*) bezeichnet.

Clients, die die Erweiterungen nutzen möchten, senden anfangs anstatt der *HELO*-Nachricht eine *EHLO*-Nachricht. Wird diese abgewiesen, dann agiert der Server als gewöhnlicher SMTP-Server und der Client verfährt auf die übliche Weise. Wird *EHLO* akzeptiert, antwortet der Server mit den Erweiterungen, die er akzeptiert, und der Client kann entscheiden, welche dieser Erweiterungen er verwenden möchte. In ►Abbildung 7.16 sind einige Erweiterungen aufgeführt. Links ist das jeweilige Schlüsselwort angegeben, das von dem Erweiterungsmechanismus verwendet wird, danach folgt eine kurze Beschreibung der neuen Funktionalität, die uns hier genügen soll.

| Schlüsselwort | Bedeutung |
|---------------|--|
| AUTH | Client-Authentifizierung |
| BINARYMIME | Server akzeptiert binäre Nachrichten |
| CHUNKING | Server akzeptiert große, in Blöcke aufgeteilte Nachrichten |
| SIZE | Überprüft vor dem Senden die Nachrichtengröße |
| STARTTLS | Wechsel zu sicherer Übertragung (mehr zu TLS in <i>Kapitel 8</i>) |
| UTF8SMTP | Internationale Adressen |

Abbildung 7.16: Ausgewählte SMTP-Erweiterungen.

Um ein besseres Gefühl für die Arbeitsweise von SMTP und einigen der anderen in diesem Kapitel beschriebenen Protokolle zu bekommen, sollten Sie sie einmal ausprobieren. Setzen Sie sich dazu zunächst an einen Rechner mit Internetzugang. Auf einem Unix-System geben Sie in die Shell folgenden Befehl ein:

```
telnet mail.isp.com 25
```

wobei Sie *mail.isp.com* durch den DNS-Namen des Mailservers Ihres ISP ersetzen. Auf Windows-XP-Systemen klicken Sie auf „Start“, dann auf „Ausführen“ und geben den Befehl in das erscheinende Dialogfeld ein. Unter Windows Vista oder Windows 7 müssen Sie das Telnet-Programm (oder ein vergleichbares Programm) zuerst installieren und anschließend manuell starten. Der obige Befehl baut eine Telnet-Verbindung (d.h. TCP) zu Port 25 auf Ihrem Rechner auf. Port 25 ist der SMTP-Port. (Eine Liste der Ports der wichtigsten Protokolle finden Sie in *Abbildung 6.34*.) Die Antwort, die Sie daraufhin erhalten, dürfte ungefähr wie folgt aussehen:

```
Trying 192.30.200.66...
Connected to mail.isp.com
Escape character is '^]'.
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

Die ersten drei Zeilen kommen von Telnet und geben an, was passiert. Die letzte Zeile kommt vom SMTP-Server auf dem entfernten Rechner und zeigt dessen Bereitschaft an, mit Ihnen zu kommunizieren und eine E-Mail zu akzeptieren. Um herauszufinden, welche Befehle er akzeptiert, geben Sie ein:

```
HELP
```

Ab diesem Punkt ist eine Befehlsfolge wie die aus Abbildung 7.15 möglich, vorausgesetzt, der Server ist willig, E-Mails von Ihnen entgegenzunehmen.

Mail Submission

Ursprünglich wurde der Benutzeragent auf demselben Rechner ausgeführt wie der sendende Nachrichtenübertragungsagent. Bei dieser Konstellation muss der Benutzeragent, um eine Nachricht zu versenden, lediglich mit dem lokalen Mailserver kommunizieren. Den zugehörigen Dialog haben wir gerade beschrieben. In der Praxis hat man es aber zumeist mit anderen Konstellationen zu tun.

Oftmals werden Benutzeragenten auf Laptops, Heim-PCs oder mobilen Endgeräten ausgeführt. Selten sind sie permanent mit dem Internet verbunden. Nachrichtenübertragungsagenten dagegen laufen auf ISP- und Firmenrechnern und sind ständig mit dem Internet verbunden. Für einen Benutzeragenten, der sich in Boston befindet, bedeutet dies möglicherweise, dass er zum Versenden einer Nachricht zuerst seinen regulären Mailserver in Seattle kontaktieren muss, weil der Benutzer auf Reise ist.

Die Remote-Kommunikation selbst stellt dabei kein Problem dar, schließlich wurde das TCP/IP-Protokoll extra für die Unterstützung dieser Form der Kommunikation konzipiert. Das Problem liegt vielmehr bei den ISPs bzw. den Firmen, die kein Interesse daran haben, dass x-beliebige entfernte Benutzer Nachrichten an ihren Mailserver senden, damit dieser sie weiterleitet. ISPs oder Firmen führen ihre Server nicht als öffentlich zugängliche Dienste aus. Auch würde diese Art der **offenen Mail-Verteilung** (*open mail relay*) unweigerlich Spammer anziehen, die den Server als „Geldwaschanlage“ zum Vertuschen des ursprünglichen Absenders benutzen können, wodurch die Nachricht schwieriger als Spam zu erkennen ist.

Aus den oben genannten Gründen wird SMTP bei der Mail Submission zumeist in Verbindung mit der *AUTH*-Erweiterung verwendet. Die Erweiterung ermöglicht dem Server die Überprüfung der Identität des Clients (Benutzername und Passwort) und bestätigt auf diese Weise, dass der Server den E-Mail-Dienst zur Verfügung stellen soll.

Es gibt noch verschiedene weitere Eigentümlichkeiten, die die Verwendung von SMTP für die Mail Submission auszeichnen. Beispielsweise wird anstelle des Ports 25 meist der Port 587 verwendet und der SMTP-Server kann das Format der vom Benutzeragenten gesendeten Nachrichten überprüfen und korrigieren. Für mehr Informationen über die Besonderheiten beim der Verwendung von SMTP für die Mail Submission, siehe RFC 4409.

Nachrichtenübertragung

Sowie der sendende Nachrichtenübertragungsagent vom Benutzeragenten eine Nachricht empfängt, schickt er sie mittels SMTP an den empfangenden Nachrichtenübertragungsagenten. Dazu benutzt der Sender die angegebene Zieladresse. Nehmen Sie z.B. die Nachricht aus Abbildung 7.15, die an *bob@ee.uwa.edu.au* adressiert ist. Zu welchem Mailserver sollte diese Nachricht geschickt werden?

Um herauszufinden, welches der richtige zu kontaktierende Mailserver ist, wird das DNS-System zurate gezogen. Weiter oben wurden bereits die verschiedenen Arten von DNS-Datensätzen beschrieben, darunter auch der MX-Datensatz (*Mail eXchange*). Die Lösung besteht folglich darin, eine DNS-Anfrage nach dem MX-Datensatz der Domäne *ee.uwa.edu.au* abzuschicken. Als Antwort erhält man eine geordnete Liste der Namen und IP-Adressen der vorhandenen Mailserver.

Der sendende Nachrichtenübertragungsagent baut dann auf Port 25 eine TCP-Verbindung zu der IP-Adresse des Mailservers auf, um den empfangenden Nachrichtenübertragungsagent zu erreichen und diesem mittels SMTP die Nachricht weiterzuleiten. Der empfangende Nachrichtenübertragungsagent legt die Nachricht für den Benutzer *bob* in

dessen Mailbox ab, sodass Bob sie später bei Gelegenheit lesen kann. Je nach Größe der lokalen E-Mail-Infrastruktur kann es auch bei diesem Zustellungsschritt erforderlich sein, die Nachricht von einem Rechner auf einen anderen Rechner zu übertragen.

Bei dem beschriebenen Verfahren wandert die E-Mail direkt vom ersten zum letzten Nachrichtenübertragungsagenten, d.h., es gibt keine zwischengeschalteten Server in dem Nachrichtenübertragungsschritt. Wohl aber ist es möglich, dass dieser Übertragungsschritt mehrfach ausgeführt wird – beispielsweise, wenn der Nachrichtenübertragungsagent, wie bereits angesprochen, eine Mailingliste ausführt. In diesem Fall nimmt der Agent die Nachricht für die Liste entgegen, vervielfältigt die Nachricht für jeden in der Liste einge-tragenen Empfänger und sendet die Nachrichten an die Adressen der Empfänger.

Betrachten wir noch ein weiteres Beispiel für die Nachrichtenübertragung: Nehmen wir an, Bob hätte seinen Abschluss am MIT gemacht und wäre zusätzlich unter der Adresse *bob@alum.mit.edu* zu erreichen. Statt die E-Mail von beiden Benutzerkonten einzeln abzufragen, könnte Bob es so einrichten, dass E-Mail, die an seine MIT-Adresse gesendet wird, automatisch an *bob@ee.uwa.edu.au* weitergeleitet wird. E-Mail-Nachrichten, die an *bob@alum.mit.edu* gesendet werden, werden dann zweimal zugestellt: Zuerst werden sie an den Mailserver für *alum.mit.edu* gesendet, dann weiter zu dem Mailserver für *ee.uwa.edu.au*. Beide Abschnitte sind aus Sicht der Nachrichtenübertragungsagenten vollständige, separate Zustellungen.

Mittlerweile spielt bei der Nachrichtenübertragung noch ein weiterer Aspekt eine große Rolle: Spam. Neun von zehn Nachrichten, die heutzutage versendet werden, sind Spammnachrichten (McAfee, 2010). Die meisten Menschen ärgern sich über diese Flut von Spammnachrichten, doch ihr zu entgehen ist schwierig, denn Spammnachrich-tten tarnen sich als normale E-Mail. Bevor eine Nachricht entgegengenommen wird, können zusätzliche Tests vorgenommen werden, um die Versendung von Spam zu erschweren. Die Nachricht für Bob wurde beispielsweise von *alice@cs.washing-ton.edu* abgeschickt, d.h., der empfangende Nachrichtenübertragungsagent kann im DNS den sendenden Nachrichtenübertragungsagenten nachschlagen, um sicherzustel-len, dass die IP-Adresse am anderen Ende der TCP-Verbindung mit dem DNS-Namen übereinstimmt. Der empfangende Agent könnte auch ganz allgemein im DNS die sen-dende Domäne nachschlagen, um zu sehen, ob diese über eine Mail-Sende-Richtlinie verfügt. Die entsprechende Information findet sich meist in den Datensätzen *TXT* und *SPF* und liefert womöglich Hinweise für weitere Tests. Beispielsweise könnte in einer solchen Richtlinie festgelegt sein, dass E-Mail von *cs.washington.edu* stets von dem Host *june.cs.washington.edu* abgeschickt wird. Ist der sendende Nachrichtenüber-tragungsagent also nicht *june*, gibt es offensichtlich ein Problem.

Scheitert einer dieser Tests, bedeutet dies, dass die E-Mail vermutlich manipuliert und mit einem falschen Absender versehen wurde. In diesem Fall wird die E-Mail aussortiert und verworfen. Umgekehrt heißt dies aber nicht, dass eine E-Mail, die alle Tests bestanden hat, keine Spam wäre. Die Tests können lediglich sicherstellen, dass die E-Mail aller Voraussicht nach aus dem Bereich des Netzwerks stammt, aus dem sie vorgibt zu sein. Worum es demnach geht, ist, die Spammer dazu zu zwingen, beim

Senden von E-Mail den tatsächlichen Absender anzugeben. Auf diese Weise kann Spammmail leichter erkannt und unerwünschte Spam kann leichter gelöscht werden.

Endzustellung

Unsere E-Mail-Nachricht ist nun beinahe zugestellt. Sie ist in Bobs Mailbox angekommen und alles, was noch zu tun bleibt, ist, eine Kopie der Nachricht an Bobs Benutzeragenten zu schicken, damit dieser die Nachricht anzeigen kann (siehe Schritt 3 in der Architektur von Abbildung 7.7). Früher, als der Benutzeragent und der Nachrichtenübertragungsagent auf demselben Rechner liefen (als eigenständige Prozesse), war dies schnell erledigt. Der Nachrichtenübertragungsagent schrieb neue Nachrichten einfach an das Ende der Mailbox-Datei und der Benutzeragent kontrollierte einfach, ob in der Mailbox-Datei neue E-Mail hinzugekommen ist.

Heutzutage werden Benutzeragenten auf PCs, Laptops und mobilen Endgeräten installiert, d.h., sie laufen eher selten auf demselben Rechner wie der ISP oder der Mailserver einer Firma. Die Leute möchten Ihre E-Mail eben auch von außerhalb abfragen – wo immer sie sich auch gerade befinden – und über jeden zur Verfügung stehenden Rechner, sei dies nun ihr Arbeitsplatzrechner, ihr Heim-PC, der Laptop, den sie auf Geschäftsreise mitgenommen haben, oder ein Rechner in einem Cybercafe an ihrem „Urlaubsort“. Darüber hinaus möchten sie offline arbeiten können und sich nur bei Bedarf mit dem Internet verbinden, um einkommende E-Mail zu empfangen und ausgehende E-Mail zu versenden. Viele Benutzer verwenden mehrere Benutzeragenten, je nachdem welcher Rechner im Moment gerade zur Verfügung steht oder bequem zu nutzen ist. Und manchmal werden sogar mehrere Benutzeragenten gleichzeitig ausgeführt.

In diesem Szenario besteht die Aufgabe des Benutzeragenten darin, die Inhalte in der Mailbox zu präsentieren und Befehle anzubieten, mit denen sich die Mailbox aus der Ferne verwalten lässt. Dabei können verschiedene Protokolle zum Einsatz kommen, SMTP ist jedoch nicht darunter. SMTP ist ein Push-basiertes Protokoll: Es übernimmt eine Nachricht und verbindet sich mit einem entfernten Server, um die Nachricht zu übertragen. Die endgültige Zustellung kann nicht auf diese Weise erfolgen, da zum einen die Mailbox weiterhin auf dem Nachrichtenübertragungsagenten gespeichert bleiben muss und zum anderen nicht sichergestellt ist, dass der Benutzeragent zu dem Zeitpunkt, da SMTP versucht, die Nachrichten zu übertragen, mit dem Internet verbunden ist.

IMAP (Internet Message Access Protocol)

Eines der wichtigsten Protokolle für die endgültige Zustellung ist **IMAP** (*Internet Message Access Protocol*). Version 4 des Protokolls ist in RFC 3501 definiert. Um IMAP nutzen zu können, führt der Mailserver einen IMAP-Server aus, der auf Port 143 lauscht. Der Benutzeragent führt einen IMAP-Client aus. Der Client verbindet sich mit dem Server und beginnt, IMAP-Befehle abzuschicken (►Abbildung 7.17).

Als Erstes initiiert der Client eine sichere Datenübertragung – sofern eine solche gewünscht wird, um Nachrichten und Befehle vor allzu neugierigen Blicken zu verbergen. Dann meldet der Client sich an bzw. authentifiziert sich anderweitig beim Server.

| Befehl | Beschreibung |
|--------------|--|
| CAPABILITY | Listet die Fähigkeiten des Servers auf |
| STARTTLS | Beginnt eine sichere Datenübertragung (TLS, siehe <i>Kapitel 6</i>) |
| LOGIN | Anmeldung beim Server |
| AUTHENTICATE | Alternative Anmeldung |
| SELECT | Wählt einen Ordner aus |
| EXAMINE | Wählt einen Nur-Lesen-Ordner aus |
| CREATE | Erzeugt einen Ordner |
| DELETE | Löscht einen Ordner |
| RENAME | Benennt einen Ordner um |
| SUBSCRIBE | Fügt dem aktiven Satz einen Ordner hinzu |
| UNSUBSCRIBE | Löscht einen Ordner aus dem aktiven Satz |
| LIST | Listet die verfügbaren Ordner auf |
| LSUB | Listet die aktiven Ordner auf |
| STATUS | Liefert den Status eines Ordners zurück |
| APPEND | Hängt eine Nachricht an einen Ordner an |
| CHECK | Liefert einen Kontrollpunkt für einen Ordner zurück |
| FETCH | Holt Nachrichten aus einem Ordner |
| SEARCH | Sucht Nachrichten in einem Ordner |
| STORE | Ändert Nachrichtenflags |
| COPY | Legt von einer Nachricht aus einem Ordner eine Kopie an |
| EXPUNGE | Entfernt Nachrichten, die für das Löschen vorgemerkt sind |
| UID | Zum Abschicken von Befehlen über eindeutige IDs |
| NOOP | Tue nichts |
| CLOSE | Entfernt markierte Ordner und schließt den Ordner |
| LOGOUT | Abmeldung und Beendigung der Verbindung |

Abbildung 7.17: IMAP-Befehle (Version 4).

Einmal angemeldet, steht ihm eine große Auswahl an Befehlen zur Verfügung, mit denen Ordner und Nachrichten aufgelistet, Nachrichten oder Teile von Nachrichten abgeholt, Nachrichten mit Flags für das Löschen markiert und Nachrichten in Ordner verteilt werden können. Um Missverständnissen vorzubeugen, sei darauf hingewiesen,

dass wir den Begriff „Ordner“ hier im gleichen Sinne gebrauchen wie im Rest dieses Abschnitts: Ein Benutzer besitzt eine einzelne Mailbox, die aus mehreren Ordnern besteht. Die IMAP-Spezifikation dagegen spricht grundsätzlich von *mailboxes*, d.h., ein Benutzer verfügt über mehrere Mailboxen, die ihm vom Benutzeragenten typischerweise als Ordner präsentiert werden.

IMAP weist noch viele weitere Merkmale auf, z.B. die Fähigkeit, E-Mails nicht nach der Nachrichtennummer, sondern nach Attributen (z.B. „Gib mir die erste Nachricht von Alice“) zu adressieren. Auch können auf dem Server Suchoperationen durchgeführt werden, um Nachrichten nach bestimmten Kriterien zu filtern und nur die gefilterten Nachrichten vom Client abholen zu lassen.

IMAP stellt eine Verbesserung gegenüber dem älteren **POP3**-Protokoll dar (*Post Office Protocol, Version 3*), das ebenfalls der endgültigen Zustellung dient und in RFC 1939 definiert wurde. POP3 ist einfacher, hat aber auch einen kleineren Funktionsumfang und ist bei normaler Anwendung weniger sicher. Zudem geht POP3 davon aus, dass die E-Mail nicht auf dem Mailserver verbleibt, sondern auf den Rechner mit dem Benutzeragenten heruntergeladen wird. Dies entlastet den Server, ist aber nachteilig für den Benutzer, der seine E-Mail nicht mehr bequem von mehreren Computern aus lesen kann und möglicherweise seine gesamte E-Mail verliert, wenn der Rechner mit dem Benutzeragenten abstürzt. Trotzdem ist POP3 immer noch in Gebrauch.

Stammen Mailserver und Benutzeragent von demselben Unternehmen, dann ist auch der Einsatz eines proprietären Protokolls möglich. Microsoft Exchange ist z.B. ein Mailsystem mit einem proprietären Protokoll.

Webmail

Eine immer populärer werdende Alternative zu IMAP und SMTP als E-Mail-Dienst ist die Verwendung des Webs als Benutzeroberfläche zum Senden und Empfangen von Mail. Weitverbreitete Webmailsysteme sind unter anderem Google Gmail, Microsoft Hotmail und Yahoo! Mail. Webmail ist ein Beispiel für eine Software (in diesem Fall ein E-Mail-Benutzeragent), die als Dienst über das Web zur Verfügung gestellt wird.

Auch in dieser Architektur führt der Provider wie gehabt Mailserver aus, um Nachrichten für die Benutzer entgegenzunehmen, mit SMTP am Port 25. Änderungen gibt es beim Benutzeragenten. Dieser ist nun kein eigenständiges Programm mehr, sondern eine Benutzeroberfläche, die in Form von Webseiten angeboten wird. Für den Benutzer bedeutet dies, dass er zum Abfragen seiner E-Mail wie auch zum Senden von Nachrichten jeden beliebigen Browser verwenden kann.

Ungeachtet dessen, dass wir uns mit dem Web noch nicht näher beschäftigt haben, folgt nun eine kurze Beschreibung der Arbeit mit Webmail. Sollten Sie Verständnis-schwierigkeiten haben, lesen Sie sich die Beschreibung einfach noch einmal durch, nachdem Sie den Abschnitt zum World Wide Web durchgearbeitet haben. Wenn der Benutzer zur E-Mail-Webseite des Providers geht, wird ein Formular angezeigt, in dem er seinen Benutzernamen und ein Passwort eingeben muss. Benutzername und Passwort werden daraufhin zum Server gesendet, der diese dann prüft. Ist die Anmeldung

erfolgreich, ermittelt der Server die Mailbox des Benutzers und generiert ad hoc eine Webseite, die den Inhalt der Mailbox auflistet. Diese Webseite wird dann an den Browser zur Anzeige übertragen.

Solche Webseiten, die den Inhalt einer Mailbox präsentieren, besitzen üblicherweise viele Elemente, die anklickbar sind, sodass die Nachrichten gelesen, gelöscht und anderweitig bearbeitet werden können. Meist sind in die Webseite JavaScript-Programme eingebettet, um die Benutzeroberfläche interaktiv zu machen. Diese Programme werden als Antwort auf lokale Ereignisse (z.B. Mausklicks) auf dem Client ausgeführt und können unter anderem auch Nachrichten im Hintergrund hoch- oder herunterladen, die nächste Nachricht für die Anzeige vorbereiten oder neu aufgesetzte Nachrichten für die Übergabe an das E-Mail-System aufbereiten. Die Übergabe, die Mail Submission, erfolgt in diesem Modell unter Verwendung der normalen Webprotokolle, indem Daten an einen URL gesendet werden. Der Webserver übernimmt die Aufgabe, die Nachrichten in das oben beschriebene traditionelle E-Mail-Zustellungs- system einzuspeisen. Aus Sicherheitsgründen können auch die Standard-Webprotokolle verwendet werden, die der Verschlüsselung von Webseiten dienen. Ob es sich bei dem Inhalt einer Webseite um eine E-Mail-Nachricht handelt, ist dabei irrelevant.

7.3 World Wide Web

Das World Wide Web – im allgemeinen Sprachgebrauch meist nur „das Web“ genannt – ist ein architektonisches Rahmenwerk für den Zugriff auf verknüpfte Inhalte, die auf Millionen von Rechnern überall im Internet verteilt sind. Innerhalb von zehn Jahren entwickelte es sich von einer Einrichtung zur Koordinierung von Versuchsaufbauten auf dem Gebiet der Hochenergiephysik in der Schweiz zu der Anwendung, die Millionen von Leuten als das „Internet“ begreifen. Seine enorme Beliebtheit ist auf die Tatsache zurückzuführen, dass es selbst für Anfänger leicht zu bedienen ist und über eine reiche grafische Benutzeroberfläche Zugriff auf eine unglaubliche Fülle von Informationen über jedes erdenkliche Thema, von Aardvarks bis Zulus bietet.

Das Web wurde 1989 am CERN-Institut, dem europäischen Zentrum für Atomphysik, ins Leben gerufen. Die ursprüngliche Zielsetzung war, größere Forscherteams, deren Mitglieder über ein halbes Dutzend oder mehr Länder und Zeitzonen verstreut sind, bei der Durchführung gemeinsamer Projekte zu unterstützen und den Wissenschaftlern Zugriff auf die sich ständig verändernde Sammlung von Berichten, technischen Zeichnungen, Diagrammen, Fotos und anderen Dokumenten zu geben, die sich aus den auf dem Gebiet der Teilchenphysik durchgeführten Experimenten ergaben. Der erste Vorschlag für ein Netz aus verknüpften Dokumenten kam von dem CERN-Physiker Sir Tim Berners-Lee im März 1989. Der erste (textbasierte) Prototyp wurde 18 Monate später in Betrieb genommen. Eine öffentliche Demonstration auf der *Hypertext '91*-Konferenz zog die Aufmerksamkeit anderer Forscher auf sich und inspirierte Marc Andreessen von der Universität von Illinois zur Entwicklung des ersten grafischen Browsers. Im Februar 1993 wurde dieser Browser unter den Namen Mosaic vorgestellt.

Der Rest ist, wie es so schön heißt, Geschichte. Mosaic war so beliebt, dass Andreesen sich ein Jahr später aufmachte und eine eigene Firma gründete, Netscape Communications Corp, deren Ziel die Entwicklung von Websoftware war. Die nächsten drei Jahre tobte zwischen dem Netscape Navigator und dem Internet Explorer von Microsoft der sogenannte „Browserkrieg“, bei dem beide Unternehmen verzweifelt versuchten, ihre Marktanteile dadurch zu steigern, dass sie ihre Browser mit mehr Funktionen (und somit mehr Bugs) ausstatteten als der Konkurrent.

Während der 1990er und 2000er wuchs die Zahl der Websites und Webseiten – wie die im Web angebotenen Inhalte bezeichnet werden – exponentiell, bis es Millionen von Sites und Milliarden von Seiten gab. Einige wenige dieser Sites erfreuen sich enormer Popularität. Diese Sites und die dahinterstehenden Unternehmen bestimmen weitgehend, wie die Menschen heute das Web wahrnehmen. Zu ihnen gehören z.B.: ein Buchhändler (Amazon, gegründet 1994, Marktkapitalisierung 50 Milliarden US-Dollar), ein Flohmarkt (eBay, 1995, 30 Mrd. US-Dollar, eine Suchfunktion (Google, 1998, 150 Mrd. US-Dollar) und ein soziales Netzwerk (Facebook, 2004, ein Privatunternehmen, dessen Wert auf 15 Mrd. US-Dollar geschätzt wird). Für die Zeit um das Jahr 2000, als viele Webunternehmen über Nacht Hunderte Millionen von Dollar wert wurden, nur um kurz danach, als sich alles nur als ein Hype herausstellte, komplett einzubrechen, wurde sogar ein eigener Name geprägt: die **Dotcom-Ära**. Für innovative Ideen ist das Web immer noch ein günstiges Umfeld. Viele dieser Ideen kommen von Studenten. Mark Zuckerberg z.B. war Student an der Harvard-Universität, als er Facebook gründete, und Sergey Brin und Larry Page waren Studenten in Stanford, als sie Google starteten. Und wer weiß, vielleicht haben Sie ja die nächste geniale Idee.

Im Jahr 1994 unterzeichneten CERN und das MIT eine Vereinbarung über die Gründung des W3C (World Wide Web Consortium), einer Organisation zur Weiterentwicklung des Webs, zur Standardisierung von Protokollen und zur Förderung der Interoperabilität zwischen Websites. Berners-Lee wurde zum Direktor ernannt. Seither sind mehrere Hundert Universitäten und Unternehmen dem Konsortium beigetreten. Über das Web wurden unzählige Bücher geschrieben, doch die beste Stelle, wo man aktuelle Informationen finden kann, ist (natürlich) das Web selbst. Die Homepage des Konsortiums befindet sich unter <http://www.w3.org>. Interessierte Leser finden dort Links zu Seiten, die alle Aktivitäten und Dokumente des Konsortiums abdecken.

7.3.1 Übersicht über die Architektur

Aus Sicht des Benutzers besteht das Web aus einer riesigen weltweiten Sammlung von Dokumenten, die **Webseiten** oder einfach nur **Seiten (page)** genannt werden. Jede Seite kann Links (Verweise) zu anderen Seiten enthalten, die sich irgendwo auf der Welt befinden. Die Benutzer können einem Link folgen (z.B. durch Anklicken), um zu der betreffenden Seite zu gelangen. Dieser Prozess kann unendlich oft wiederholt werden. Das Konzept, dass eine Seite auf eine andere verweist, heutzutage als **Hypertext** bezeichnet, wurde von einem hellsichtigen Professor für Elektrotechnik am MIT, Vannevar Bush, im Jahre 1945 lange vor Erfindung des Internets entwickelt, ja sogar noch

bevor es kommerzielle Rechner gab (immerhin, einige Universitäten hatten bereits erste klobige Prototypen entwickelt, die ganze Säle füllten und weniger Rechenleistung aufwiesen als ein moderner Taschenrechner).

Webseiten werden üblicherweise in einem speziellen Programm namens Browser angesehen. Populäre Browser sind z.B. Firefox, Internet Explorer oder Chrome. Der Browser holt die angeforderte Seite, interpretiert den Inhalt und zeigt die Seite ordentlich formatiert auf dem Bildschirm an. Bei dem darzustellenden Inhalt kann es sich um einen Mix aus Text, Bildern und Formatierungsbefehlen, vergleichbar einem traditionellen Dokument, oder um alternative Formen handeln, wie z.B. Videofilm oder einem Programm, das eine grafische Benutzeroberfläche erzeugt, mit dem der Benutzer interagieren kann.

Auf der linken Seite von ► Abbildung 7.18 sehen Sie ein Beispiel für eine solche Webseite, in diesem Fall die Seite der Computer- und Ingenieurwissenschaften an der Universität von Washington. Wie Sie sehen, enthält die Seite sowohl Text als auch grafische Elemente (die meist zu klein zum Entziffern sind). Einige Teile der Seite sind mit Links zu anderen Webseiten verknüpft. Textpassagen, Symbole oder Bilder, die mit einer anderen Seite verknüpft sind, heißen **Hyperlinks**. Um einem Link zu folgen, setzt der Benutzer den Mauszeiger auf das verknüpfte Element (wodurch sich die Form des Mauszeigers verändert) und klickt darauf. Einem Link zu folgen ist nichts anderes als ein besonders bequemer Weg, um dem Browser mitzuteilen, dass er eine andere Seite laden soll. Früher wurden Hyperlinks meist als unterstrichener, farbiger Text dargestellt, um sie aus der Umgebung hervorzuheben. Heute stehen den Designern von Webseiten ganz andere Mittel zur Gestaltung von verknüpften Elementen zur Verfügung, sodass Links beispielsweise auch als Symbole erscheinen oder ihr Aussehen verändern können, wenn die Maus über sie bewegt wird. Wichtig ist, dass der Designer sich um die optische Hervorhebung der Links und damit um die benutzerfreundliche Gestaltung der Seitenoberfläche kümmert.

Die Studenten der Fakultät brauchen nur dem für sie vorgesehenen Link zu folgen, um zu einer Seite zu gelangen, auf der alle für sie wichtigen Informationen zusammengetragen sind. Um dem Link zu folgen, klickt man in den eingekreisten Bereich. Der Browser holt daraufhin die neue Seite und zeigt sie an (siehe den Bereich links unten in Abbildung 7.18). Daneben gibt es noch Dutzende weiterer Seiten, die via Links von der ersten Seite aus erreichbar sind. Jede dieser Seiten kann aus Inhalten zusammengesetzt sein, die sich auf denselben Rechner (oder Rechnern) wie die erste befinden oder auf irgendeinem anderen Rechner auf der Welt. Der Benutzer kann das nicht erkennen. Seiten werden vom Browser ohne Zutun des Benutzers abgerufen, der Wechsel zwischen den Rechnern erfolgt nahtlos, während der Benutzer die angebotenen Inhalte betrachtet.

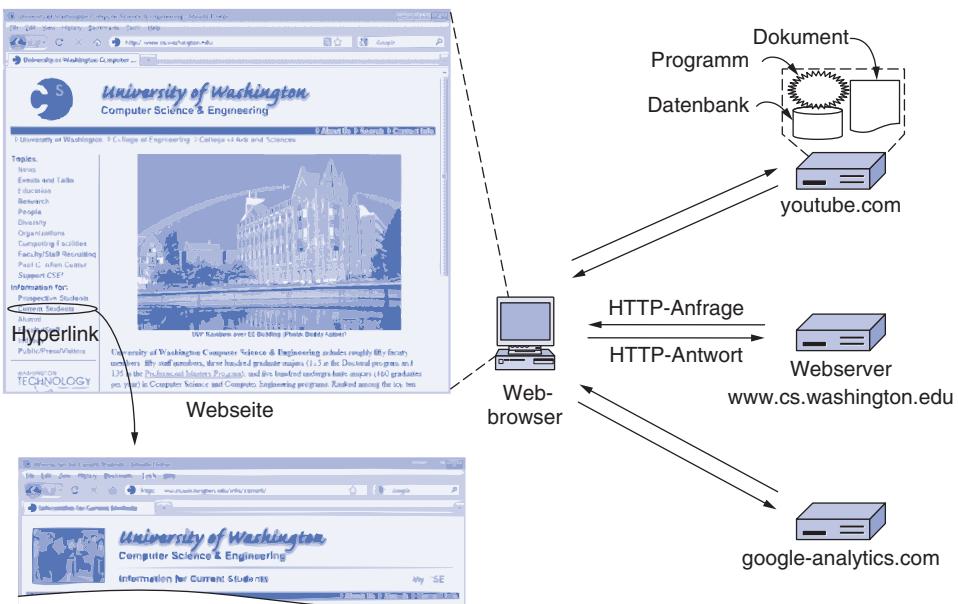


Abbildung 7.18: Architektur des Webs.

Abbildung 7.18 verdeutlicht auch das grundlegende Modell, das hinter der Darstellung der Seiten steht. Der Browser präsentiert auf dem Client-Rechner eine Webseite. Für jede anzuzeigende Seite sendet der Browser eine Anfrage (Request) an einen oder mehrere Server, die ihm als Antwort die zu der Seite gehörenden Inhalte schicken. Das Anfrage-Antwort-Protokoll zum Anfordern der Seiten ist ein einfaches textbasiertes Protokoll, das ebenso wie SMTP auf TCP aufsetzt. Die Rede ist von **HTTP (HyperText Transfer Protocol)**. Bei dem zurückgesendeten Inhalt kann es sich um ein Dokument handeln, das einfach von der Festplatte gelesen wurde, um das Ergebnis einer Datenbankabfrage oder auch das Produkt eines extra ausgeführten Programms. Besteht die Seite aus einem Dokument, das jedes Mal genau gleich angezeigt wird, spricht man von einer **statischen Seite**, wohingegen Seiten, die auf Anfrage von einem Programm erzeugt werden oder ein Programm enthalten, als **dynamische Seiten** bezeichnet werden.

Dynamische Seiten können jedes Mal, wenn sie angezeigt werden, anders aussehen. Beispielsweise könnte sich die Startseite eines Elektronikhändlers jedem Besucher anders präsentieren. Hat ein Kunde eines Online-Buchhändlers in der Vergangenheit mehrere Mystery-Thriller erstanden, ist es gut möglich, dass ihm bei seinem nächsten Besuch auf der Hauptseite des Händlers neue Thriller auffällig angepriesen werden, während ein Kunde, der eher an kulinarischen Genüssen interessiert ist, mit neuen Kochbüchern begrüßt wird. Wie eine Website sich merken kann, welcher Besucher welche Vorlieben hat, ist eine Frage, der wir uns in Kürze ausführlicher zuwenden werden. Soviel sei aber schon verraten: Die Antwort hat mit Cookies, zu Deutsch „Keksen“, zu tun (und dies gilt auch für Besucher, die keine Backwaren mögen).

In der Abbildung kontaktiert der Browser drei Server, um zwei Seiten anzufordern: *cs.washington.edu*, *youtube.com* und *google-analytics.com*. Die Inhalte von diesen verschiedenen Servern baut der Browser für die Anzeige zusammen. Das Anzeigen der Seiten umfasst verschiedene Prozesse, die von der Art des darzustellenden Inhalts abhängen. Neben der Darstellung von Texten und Bildern ist es unter Umständen erforderlich, ein Video abzuspielen oder ein Skript auszuführen, das seine eigene Benutzeroberfläche als Teil der Seite präsentiert. In diesem Fall, liefert der *cs.washington.edu*-Server die Hauptseite und der *youtube.com*-Server steuert ein eingebettetes Video bei. Der *google-analytics.com*-Server liefert keine für den Benutzer sichtbaren Inhalte, sondern zeichnet auf, wer die Site besucht. Über diese sogenannten Tracker zur Analyse des Besucherverhaltens wird später noch etwas mehr zu sagen sein.

Die Clientseite

Sehen wir uns nun die Clientseite aus Abbildung 7.18 etwas genauer an. Im Wesentlichen ist ein Browser ein Programm, das eine Webseite anzeigen und Mausklicks auf Elemente auf der angezeigten Seite abfangen und interpretieren kann. Wird ein Element ausgewählt, dann folgt der Browser dem Hyperlink und ruft die ausgewählte Seite ab.

Als das Web ins Leben gerufen wurde, zeichnete sich schnell ab, dass man spezielle Verfahren zum Benennen und Lokalisieren von Seiten benötigen würde, um von einer Webseite auf eine andere verweisen zu können. Es sind vor allem drei Fragen, die beantwortet werden müssen, bevor eine ausgewählte Seite angezeigt werden kann:

1. Wie heißt die Seite?
2. Wo ist die Seite zu finden?
3. Wie kann man auf die Seite zugreifen?

Wenn jede Seite irgendwo einen eindeutigen Namen zugewiesen bekäme, gäbe es keine Unklarheiten bei der Identifizierung der Seiten. Das Problem wäre damit allerdings noch immer nicht gelöst. Eine Parallele zwischen Menschen und Seiten soll dies verdeutlichen. In den Vereinigten Staaten besitzt nahezu jeder eine Sozialversicherungsnummer. Da keine zwei Personen dieselbe Nummer zugewiesen bekommen, können Sie eine Person über die Sozialversicherungsnummer eindeutig identifizieren. Doch es gibt keine Möglichkeit für Sie, aus der Sozialversicherungsnummer die Adresse ihres Eigentümers zu bestimmen oder zu entscheiden, ob Sie der Person auf Englisch, Deutsch oder Chinesisch schreiben sollen. Und vor praktisch den gleichen Problemen steht auch das Web.

Die Lösung, die letztlich umgesetzt wurde, identifiziert die Webseiten auf eine Weise, dass alle drei Probleme gleichzeitig gelöst werden. Jeder Seite wird eine **URL** (*Uniform Resource Locator*) zugewiesen, die als weltweit gültiger Name der Seite fungiert. URLs bestehen aus drei Teilen: dem Namen des Protokolls (auch als *scheme* bezeichnet), dem DNS-Namen des Rechners, auf dem sich die Seite befindet, und dem Pfad, der eindeutig zu der gewünschten Seite (d.h. der zu lesenden Datei oder dem ausführen-

den Programm auf dem Rechner) führt. Im Allgemeinen besteht der Pfad aus einem hierarchischen Namen wie man ihn auch aus Verzeichnisangaben kennt. Beachten Sie jedoch, dass die Interpretation des Pfads dem Server obliegt. Der Pfad kann also, muss aber nicht die tatsächliche Verzeichnisstruktur widerspiegeln.

Die URL zu der Seite aus Abbildung 7.18 lautet z.B.:

`http://www.cs.washington.edu/index.html`

Diese URL besteht aus drei Teilen: dem Protokoll (*http*), dem DNS-Namen des Hostrechners (*www.cs.washington.edu*) und dem Pfad (*index.html*).

Klickt ein Benutzer auf einen Hyperlink, führt der Browser verschiedene Schritte aus, um die referenzierte Seite abzurufen. Verfolgen wir einmal die Schritte, die ablaufen, wenn unser Beispielink ausgewählt wird:

1. Der Browser ermittelt die URL (indem er prüft, was ausgewählt wurde).
2. Der Browser fragt das DNS-System nach der IP-Adresse von *www.cs.washington.edu*.
3. DNS gibt die Antwort 128.208.3.88 zurück.
4. Der Browser baut zu Port 80 von 128.208.3.88 eine TCP-Verbindung auf.
5. Dann sendet er eine HTTP-Anfrage nach der Datei *index.html*.
6. Der Server von *www.cs.washington.edu* sendet als Antwort die angeforderte Seite, beispielsweise indem er die Datei */index.html* schickt.
7. Enthält die Seite URLs, die für die Anzeige benötigt werden, beschafft der Browser nach dem gleichen Verfahren auch die Inhalte dieser URLs. Im Falle unserer Beispelseite handelt es sich um mehrere eingebettete Bilder, die sich ebenfalls auf *www.cs.washington.edu* befinden, einem eingebetteten Video auf *youtube.com* und einem Skript von *google-analytics.com*.
8. Der Browser zeigt die Seite */index.html* wie in Abbildung 7.18 an.
9. Gibt es keine unmittelbaren weiteren Anfragen an den Server, wird die TCP-Verbindung getrennt.

Viele Browser zeigen den aktuell durchgeführten Schritt in einer Statuszeile am unteren Bildschirmrand an. Dauert das Laden einer Seite unverhältnismäßig lange, kann der Benutzer dann in der Statuszeile nachlesen, ob dies darauf zurückzuführen ist, dass das DNS nicht reagiert, der Server nicht reagiert oder die Übertragung einfach über ein langsames oder überlastetes Netz erfolgt.

Das URL-Schema ist in dem Sinn offen, als die Browser verschiedene gängige Protokolle einsetzen können, um auf verschiedene Arten von Ressourcen zuzugreifen. Entsprechend wurden URLs für viele weitere Protokolle definiert. Die wichtigsten sind in leicht vereinfachter Form in ▶ Abbildung 7.19 zusammengestellt.

| Name | Verwendet für | Beispiel |
|--------|--------------------|---|
| http | Hypertext (HTML) | http://www.ee.uwa.edu/~rob/ |
| https | Sicherer Hypertext | https://www.bank.com/accounts/ |
| ftp | FTP | ftp://ftp.cs.vu.nl/pub/minix/README |
| file | Lokale Datei | file:///usr/suzanne/prog.c |
| mailto | Senden von E-Mail | mailto:JohnUser@acm.org |
| rtsp | Streaming Media | rtsp://youtube.com/montypython.mpg |
| sip | Multimedia-Aufrufe | sip:eve@adversary.com |
| about | Browserinformation | about:plugins |

Abbildung 7.19: Einige bekannte URL-Schemata.

Gehen wir nun die Liste kurz durch. Das *http*-Protokoll ist die ureigene Sprache des Webs, die Webserver von Haus aus sprechen. **HTTP** steht für **HyperText Transfer Protocol**. Wir befassen uns damit später in diesem Kapitel ausführlicher.

Das *ftp*-Protokoll wird eingesetzt, um über FTP, das Dateiübertragungsprotokoll des Internets, auf Dateien zuzugreifen. FTP ist älter als das Web, es gibt es bereits seit mehr als drei Jahrzehnten. Vom Web aus ist es besonders einfach, Dateien von verschiedenen FTP-Servern rund um die Welt abzurufen, da es die traditionelle, befehlzeilenorientierte Benutzeroberfläche von FTP durch eine einfache, klickbare Benutzeroberfläche ersetzt. Dieser vereinfachte und verbesserte Zugriff auf Informationen ist einer der Gründe für das spektakuläre Wachstum des Webs.

Will man eine lokale Datei als Webseite anzeigen, benutzt man das *file*-Protokoll oder gibt einfach den Dateinamen ein. Für diese Art des Zugriffs wird kein Server benötigt; dafür funktioniert es aber auch nur bei lokalen Dateien, nicht bei entfernten.

Mit dem Protokoll *mailto* können zwar keine Webseiten besorgt werden, es ist aber auf andere Weise nützlich. Es erlaubt Benutzern, E-Mail über einen Webbrowser zu versenden. Die meisten Browser reagieren auf das Anklicken eines *mailto*-Links, indem sie den Mail-Agenten des Benutzers starten und ihn eine Nachricht mit vorgegebener Adresse anlegen lassen.

Die *rtsp*- und *sip*-Protokolle sind für Streaming-Media-Sitzungen und Audio- oder Videoanrufe.

Das *about*-Protokoll schließlich ist eine Konvention, die Informationen über den Browser liefert. So veranlasst z.B. der *about:plugins*-Link die meisten Browser dazu, eine Seite mit den MIME-Typen anzuzeigen, die sie mithilfe spezieller Browsererweiterungen, Plug-ins genannt, anzeigen können.

Mit anderen Worten, das System der URLs ist nicht nur dafür ausgelegt, dem Benutzer die Navigation im Web zu ermöglichen, sondern erlaubt auch die Arbeit mit älteren

Protokollen wie FTP und E-Mail, die Verwendung neuerer Protokolle für Audio und Video sowie den bequemen Zugriff auf lokale Dateien und Browserinformationen. Dieser Ansatz macht all die Programme überflüssig, die spezialisierte Benutzeroberflächen zu den oben aufgeführten Diensten anbieten, und integriert stattdessen nahezu alle Formen des Internetzugriffs in einem einzigen Programm: dem Webbrowser. Wäre es nicht allgemein bekannt, dass dieses Konzept von einem britischen Physiker entwickelt wurde, der in einem Forschungslabor in der Schweiz arbeitete, man könnte es leicht für den Plan der Marketingabteilung irgendeines Softwareunternehmens halten.

Trotz dieser positiven Eigenschaften hat die zunehmende Nutzung des Webs auch eine Schwäche des URL-Schemas aufgedeckt. Eine URL verweist immer auf einen speziellen Host. Manchmal wäre es aber hilfreich, auf eine Seite verweisen zu können, ohne gleichzeitig angeben zu müssen, wo die Seite zu finden ist. Für Seiten, auf die sehr häufig zugegriffen wird, wäre es beispielsweise wünschenswert, wenn mehrere Kopien der Seite an weit auseinanderliegenden Stellen zur Verfügung ständen, um den Netzverkehr zu reduzieren. Nur leider gibt es keine Möglichkeit zu sagen: „Ich möchte die Seite xyz, gleichgültig, von wo du sie holst.“

Um dieses Problem zu lösen, wurde als Verallgemeinerung der URLs das Konzept der **URIs** (*Universal Resource Identifier*) entwickelt. Einige URIs geben an, wo eine Ressource zu finden ist. Dies sind die URLs. Andere URIs geben den Namen einer Ressource an, ohne etwas über deren Speicherort zu verraten. Diese URIs werden **URNs** (*Universal Resource Name*) genannt. Die Regeln für das Aufsetzen von URIs sind in RFC 3986 definiert, während die verschiedenen im Gebrauch befindlichen URI-Schemata vom IANA verwaltet werden. Neben den in Abbildung 7.19 aufgeführten Schemata gibt es noch viele weitere Arten von URIs, die aber für die Art und Weise, wie das Web heutzutage genutzt wird, bei Weitem nicht die gleiche Rolle spielen.

MIME-Typen

Um eine neue Seite (oder jede beliebige Seite) anzeigen zu können, muss der Browser das Format verstehen. Damit alle Browser alle Webseiten verstehen können, werden die Webseiten in einer standardisierten Sprache namens HTML, der (derzeitigen) Lingua Franca des Webs, geschrieben. Wir werden uns weiter unten in diesem Kapitel noch ausführlicher mit HTML befassen.

Obwohl es sich bei einem Browser im Grunde um einen HTML-Interpreter handelt, verfügen die meisten Browser über zahlreiche Schaltflächen und Funktionen, die die Navigation im Web vereinfachen – wie z.B. eine Schaltfläche, um zur vorherigen Seite zurückzugehen, eine Schaltfläche, um zur nächsten Seite zu gehen (funktioniert nur, wenn der Benutzer zuvor von dieser Seite zurückgegangen ist), eine Schaltfläche um direkt zur bevorzugten Startseite des Benutzers zu gelangen. Die meisten Browser besitzen zudem eine Schaltfläche oder einen Menübefehl, um eine Seite mit einem Lesezeichen (Bookmarks) zu versehen, sowie als Gegenstück eine Schaltfläche bzw. einen Menübefehl, um die Liste der Lesezeichen anzuzeigen. Auf diese Weise kann der Benutzer die aufgeführten Seiten jederzeit mit nur wenigen Mausklicks erneut besuchen.

Wie unser Beispiel bereits demonstrierte, können HTML-Seiten neben einfachem Text und Hypertext auch zahlreiche andere Inhaltselemente enthalten. Ja, die Seiten müssen nicht einmal HTML enthalten, d.h., eine Seite kann auch einfach aus einem Video im MPEG-Format, aus einem Dokument im PDF-Format, einer Fotografie im JPEG-Format, einem Song im MP3-Format oder aus einem von hundert anderen Dateitypen bestehen. Da Standard-HTML-Seiten Links auf jeden dieser Inhaltstypen enthalten kann, steht der Browser vor einem Problem, wenn er auf eine Seite trifft, deren Inhalt er nicht interpretieren kann.

Anstatt die Browser immer weiter auszubauen, indem ihnen Code hinzugefügt wird, um die schnell anwachsende Anzahl der verschiedenen Dateitypen interpretieren zu können, wählen die meisten Browser eine allgemeinere Lösung. Wenn ein Server eine Seite sendet, liefert er auch diverse Zusatzinformationen über die Seite. Zu diesen Informationen gehört der MIME-Typ der Seite (siehe Abbildung 7.13). Seiten vom Typ *text/html* werden direkt angezeigt, ebenso wie die Seiten der anderen integrierten Typen. Ist der MIME-Typ nicht integriert, dann schlägt der Browser in seiner MIME-Typ-Tabelle nach, die angibt, wie die Seite angezeigt werden soll. Die Tabelle verknüpft einen MIME-Typ mit einem Wiedergabeprogramm (Viewer).

Es gibt zwei Möglichkeiten: Plug-ins und Hilfsprogramme. Ein **Plug-in** ist ein Code-modul eines Drittanbieters, das als Erweiterung des Browsers installiert ist (►Abbildung 7.20a). Typische Beispiele sind Plug-ins für PDF, Flash oder QuickTime zum Anzeigen von Dokumenten bzw. Abspielen von Audio und Video. Da Plug-ins im Browser ausgeführt werden, können sie auf die aktuelle Seite zugreifen und deren Darstellung verändern.

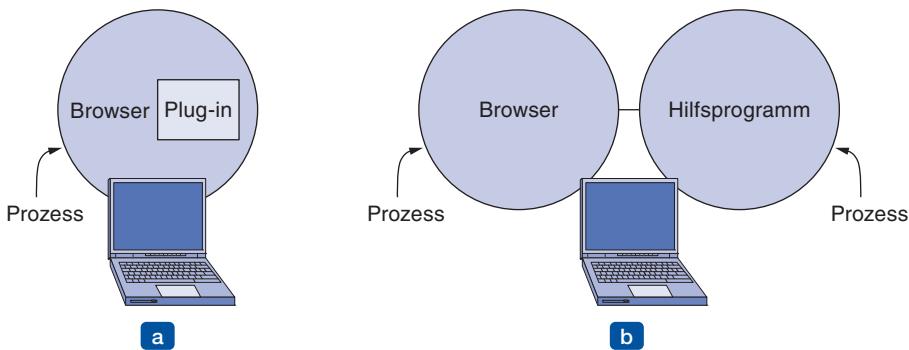


Abbildung 7.20: (a) Ein Browser-Plug-in. (b) Ein Hilfsprogramm.

Jeder Browser verfügt über einen Satz von Prozeduren, die jedes Plug-in implementieren muss, damit der Browser das Plug-in aufrufen kann. Beispielsweise gibt es fast immer eine Prozedur, die der Basiscode des Browsers aufruft, um dem Plug-in mitzuteilen, welche Daten es anzeigen soll. Dieser Satz von Prozeduren bildet die Plug-in-Schnittstelle und ist browserspezifisch.

Umgekehrt stellt auch der Browser dem Plug-in diverse Dienste in Form eines Satzes eigener Prozeduren zur Verfügung. Typische Prozeduren dieser Browserschnittstelle

dienen der Zuweisung und Freigabe von Speicher, der Anzeige einer Nachricht in der Statuszeile des Browsers und der Abfrage von Informationen über den Browser.

Bevor ein Plug-in eingesetzt werden kann, muss es installiert werden. Die übliche Vorgehensweise ist, dass der Benutzer zu der Website des Plug-ins geht und eine Installationsdatei herunterlädt. Bei Ausführung der Installationsdatei wird das Plug-in extrahiert und es werden diverse Aufrufe ausgeführt, um den MIME-Typ des Plug-ins mit dem Plug-in zu verknüpfen und beim Browser zu registrieren. Die gängigsten Plug-ins sind in der Regel in den Browser schon vorinstalliert.

Die andere Möglichkeit, einen Browser zu erweitern, ist die Verwendung eines **Hilfsprogramms** (*helper application*). Dies ist ein vollwertiges Programm, das als eigener Prozess läuft (siehe ► Abbildung 7.20b). Da das Hilfsprogramm eine eigenständige Anwendung ist, hält die Schnittstelle es auf Distanz zum Browser. Alles, was sie erlaubt, ist in der Regel die Übergabe des Namens einer Arbeitsdatei, in der die Inhaltsdatei gespeichert wurde. Diese wird dann von dem Hilfsprogramm geöffnet und angezeigt. Hilfsprogramme sind in der Regel große Programme, die unabhängig vom Browser existieren, wie z.B. Microsoft Word oder PowerPoint.

Viele Hilfsprogramme setzen den MIME-Typ *application* ein. Eine beträchtliche Anzahl an Untertypen wurde definiert, wie beispielsweise *application/vnd.ms-powerpoint* für PowerPoint-Dateien. Das Präfix *vnd* kennzeichnet das Format als spezifisch für einen Softwarehersteller („vendor-specific format“). Auf diese Weise kann eine URL direkt auf eine PowerPoint-Datei zeigen. Wenn der Benutzer darauf klickt, wird PowerPoint automatisch gestartet und bekommt den anzulegenden Inhalt übergeben. Hilfsprogramme müssen nicht den MIME-Typ *application* verwenden. Adobe Photoshop verwendet z.B. *image/x-photoshop*.

Folglich können Browser so konfiguriert werden, dass sie eine praktisch unbegrenzte Anzahl an Dokumenttypen ohne Änderungen am Browser verarbeiten können. Moderne Webserver werden oftmals mit Hunderten von Typ/Untertyp-Kombinationen konfiguriert und neue werden häufig direkt bei Installation des neuen Programms hinzugefügt.

Konflikte entstehen, wenn mehrere Plug-ins und Hilfsprogramme für einen MIME-Untertyp wie z.B. *video/mpg* verfügbar sind. Was passiert, ist, dass das letzte zu registrierende Programm die bestehenden Verknüpfung (MIME-Typ, Hilfsprogramm) überschreibt und den Typ für sich beansprucht. Daher kann die Installation eines neuen Programms die Art und Weise verändern, wie ein Browser vorhandene Typen handhabt.

Browser können auch lokale Dateien öffnen, ohne Inanspruchnahme eines Netzwerks, anstatt sie von einem entfernten Webserver abzurufen. Der Browser muss allerdings auf irgendeine Weise den MIME-Typ ermitteln können. Meist springt hier das Betriebssystem bei, welches die Dateierweiterungen mit den MIME-Typen verknüpft. Bei typischer Konfiguration wird *foo.pdf* beim Öffnen in den Browser geladen und mithilfe eines *application/pdf*-Plug-ins angezeigt, während *bar.doc* in Word als das *application/msword*-Hilfsprogramm geöffnet wird.

Auch hier kann es zu Konflikten kommen, da viele Programme bereit, ja sogar darauf erpicht sind, z.B. *mpg* zu verarbeiten. Bei der Installation zeigen Programme, die für professionelle Benutzer konzipiert sind, oft Kästchen für die Auswahl von MIME-Typen und Erweiterungen an, für die sie ausgelegt sind, damit die Benutzer die gewünschten auswählen können und vorhandene Verknüpfungen nicht zufällig überschreiben. Programme, die auf den Endverbrauchermarkt zielen, gehen davon aus, dass der Benutzer nicht weiß, was ein MIME-Typ ist, und schnappen sich alles, was sie können, ohne Rücksicht auf bereits installierte Programme.

Die Möglichkeit, den Browser mit vielen neuen Typen zu erweitern, ist bequem, kann aber zu Problemen führen. Wenn ein Browser auf einem Windows-PC eine Datei mit der Erweiterung *exe* abruft, erkennt er, dass es sich um eine ausführbare Datei handelt und es daher kein Hilfsprogramm gibt. Die offensichtliche Aktion ist, das Programm auszuführen. Dies stellt aber ein enormes Sicherheitsrisiko dar. Eine bösartige Website muss lediglich eine Webseite mit Bildern (beispielsweise von Film- oder Sportgrößen) erzeugen, die alle mit einem Virus verknüpft sind. Mit einem einfachen Klick auf eines dieser Bilder wird dann ein unbekanntes und möglicherweise gefährliches ausführbares Programm heruntergeladen und auf dem Benutzerrechner ausgeführt. Um sich vor solch unerwünschten Gästen zu schützen, sind Firefox und andere Browser werksmäßig so konfiguriert, dass sie mit der Ausführung unbekannter Programme vorsichtig sind, aber nicht jeder Benutzer versteht, welche Einstellungen sicher und welche bequem, aber unsicher sind.

Die Serverseite

So viel zur Clientseite. Nun betrachten wir die Serverseite: Wenn, wie oben erklärt, ein Benutzer eine URL eintippt oder auf eine Hypertextzeile klickt, parst der Browser die URL und interpretiert den Teil zwischen *http://* und dem nächsten Schrägstrich als den DNS-Namen, der nachzuschlagen ist. Ausgerüstet mit der IP-Adresse des Servers, baut der Browser eine TCP-Verbindung zu Port 80 auf diesem Server auf. Dann sendet er einen Befehl, welcher den Rest der URL enthält und den Namen der Datei auf diesem Server angibt. Der Server gibt dann die Datei zurück, die der Browser anzeigen kann.

Als erste Näherung kann man sagen, dass ein einfacher Webserver ähnlich wie der Server in Abbildung 6.6 aussieht. Dem Server wird der Name einer Datei übergeben, die er lokalisieren und über das Netz zurückliefern soll. In beiden Fällen sehen die Schritte, die der Server in einer Schleife wieder und wieder ausführt, wie folgt aus:

1. Eine TCP-Verbindung von einem Client (einem Browser) akzeptieren
2. Den Pfad zu der Seite entgegennehmen (dies ist der Name der angeforderten Datei)
3. Die Datei (von der Festplatte) abrufen
4. Die Inhalte der Datei an Client senden
5. Die TCP-Verbindung freigeben

Moderne Webserver verfügen über mehr Funktionen, aber dies sind im Prinzip die Schritte, die ein Webserver ausführt. Bei dynamischen Inhalten kann der dritte Schritt durch die Ausführung eines Programms ersetzt werden, das die gewünschten Inhalte zurückliefert. Die Information, welches Programm auszuführen ist, steckt in der Pfadangabe.

Allerdings sind Webserver etwas anders konzipiert, damit sie mehrere Anfragen pro Sekunde verarbeiten können. Ein grundsätzliches Problem des einfachen Designs ist, dass der Zugriff auf die Dateien oft zum Nadelöhr wird. Lesezugriffe auf eine Festplatte sind im Vergleich zur normalen Programmausführung sehr langsam. Hinzu kommt, dass immer nur eine Anfrage gleichzeitig bearbeitet wird. Ist eine angeforderte Datei sehr groß, dann werden andere Anfragen so lange blockiert, bis die Übertragung abgeschlossen ist.

Eine offensichtliche Verbesserung (die von allen Webservern verwendet wird) ist, im Speicher einen Cache anzulegen, in dem die n zuletzt gelesenen Dateien oder Inhaltsdaten bis zu einer bestimmten Anzahl von Gigabytes zwischengespeichert werden. Bevor der Server auf die Festplatte zugreift, um eine Datei zu holen, prüft er den Cache. Wenn die Datei vorhanden ist, kann sie direkt aus dem Speicher übernommen werden, sodass nicht auf die Festplatte zugegriffen werden muss. Obwohl effektives Caching sehr viel Hauptspeicher und zusätzliche Verarbeitungszeit zum Prüfen und Verwalten des Cache benötigt, sind die Zeiteinsparungen fast immer den Overhead und die Kosten wert.

Eine Strategie, wie man das Problem angehen kann, dass immer nur eine Anfrage gleichzeitig bedient wird, ist, den Server **Multithread-fähig** zu machen. Bei einem Design besteht der Server aus einem Front-End-Modul, das alle eingehenden Anfragen akzeptiert, und k Verarbeitungsmodulen, ▶ Abbildung 7.21. Die $k+1$ Threads gehören alle zu einem Prozess, sodass die verarbeitenden Module alle Zugriff auf den Cache im Adressraum des Prozesses haben. Kommt eine Anfrage herein, nimmt das Front-End sie an und erstellt einen kurzen Datensatz, der sie beschreibt. Dann übergibt es den Datensatz an eines der verarbeitenden Module.

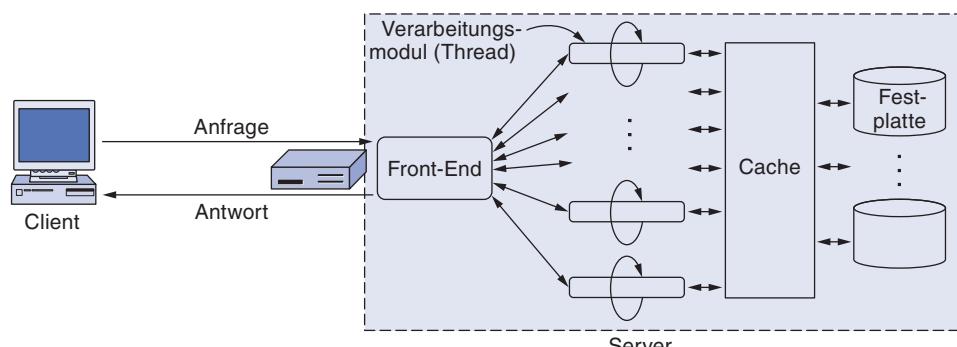


Abbildung 7.21: Ein Multithread-Webserver mit einem Front-End und verarbeitenden Modulen.

Das verarbeitende Modul prüft zuerst den Cache, um zu sehen, ob die benötigte Datei dort vorhanden ist. Ist dies der Fall, wird der Datensatz aktualisiert, d.h., es wird ein Zeiger auf die Datei hinzugefügt. Ist die Datei nicht vorhanden, startet das verarbeitende Modul einen Festplattenzugriff, um sie in den Cache einzulesen (und möglicherweise einige andere im Cache abgelegte Dateien zu verwerfen, um Platz zu machen). Die Datei von der Festplatte wird sowohl im Cache abgelegt als auch an den Client zurückgesendet.

Der Vorteil dieses Schemas ist, dass während ein oder mehrere verarbeitende Module blockiert sind und auf die Beendigung der Festplatten- oder Netzwerkoperation warten (und daher keine CPU-Zeit verbrauchen), andere Module aktiv an anderen Anfragen arbeiten können. Bei k verarbeitenden Modulen kann der Durchsatz k -mal höher sein als bei einem Single-Threaded-Server. Sollte allerdings der begrenzende Faktor die Festplatte oder das Netzwerk sein, lässt sich eine tatsächliche Verbesserung gegenüber dem Single-Threaded-Modell nur mithilfe mehrerer Festplatten oder eines schnelleren Netzwerks erzielen.

Moderne Webserver leisten weit mehr als einfach nur Dateinamen zu akzeptieren und Dateien zurückzugeben. Tatsächlich kann die Verarbeitung einer Anfrage ziemlich kompliziert werden. Aus diesem Grund führt bei vielen Servern jedes verarbeitende Modul eine Reihe von Schritten aus. Das Front-End übergibt jede eingehende Anfrage an das erste verfügbare Modul, das es dann unter Verwendung einer Teilmenge der folgenden Schritte ausführt. Die genaue Auswahl der Schritte hängt davon ab, welche Schritte für die gegebene Anfrage benötigt werden.

1. Namen der angeforderten Webseite auflösen
2. Zugriffskontrolle auf der Webseite durchführen
3. Cache prüfen
4. Angeforderte Seite von Festplatte abrufen oder ein Programm zur Erzeugung der Seite ausführen
5. Rest der Antwort festlegen (wie z.B. den zu sendenden MIME-Typ)
6. Antwort an Client übergeben
7. Eintrag im Server-Log vornehmen

Schritt 1 ist nötig, da die eingehende Anfrage nicht immer den wortwörtlichen Namen der Datei bzw. des auszuführenden Programms enthält. Manchmal enthält die Anfrage Kürzel, die zuerst übersetzt werden müssen. Betrachten wir beispielsweise die URL <http://www.cs.vu.nl>, bei der der Platz für den Dateinamen leer bleibt und die daher um einen Standarddateinamen, meist *index.html*, erweitert werden muss. Eine andere weitverbreitete Regel ist, *~user/* durch das Webverzeichnis des Benutzers *user* zu ersetzen. Alle diese Regel können auch zusammen angewendet werden. So kann z.B. die Homepage eines der Autoren dieses Buches (AST) über folgende URL erreicht werden,

<http://www.cs.vu.nl/~ast/>

obwohl der eigentliche Dateiname *index.html* lautet und in einem bestimmten Standardverzeichnis liegt.

Moderne Browser können auch Konfigurationsdaten wie z.B. die Browsersoftware oder die Standardsprache des Benutzers (beispielsweise Deutsch oder Englisch) angeben, sodass der Server für Mobilgeräte, sofern verfügbar, eine Webseite mit kleineren Symbolen und in der vom Benutzer bevorzugten Sprache auswählen kann. Im Allgemeinen ist die Erweiterung des Namens nicht so trivial, wie es zuerst scheinen mag, da bei der Abbildung der Pfade auf Verzeichnisse und Programme diverse Konventionen zu berücksichtigen sind.

Schritt 4 prüft, ob für die Seite Zugriffsbeschränkungen existieren. Nicht alle Seiten sind für die Öffentlichkeit zugänglich. Ob ein Client berechtigt ist, eine Seite anzufordern, kann beispielsweise von der Identität des Clients (ausgewiesen z.B. durch Angabe von Benutzernamen und Passwort) oder der Position des Clients im DNS- oder IP-Raum abhängig gemacht werden. So könnte eine Firma entscheiden, dass eine bestimmte Seite nur für Mitarbeiter der Firma zugänglich sein soll. Wie diese Zugriffsbeschränkung umgesetzt wird, hängt vom Design des Servers ab. Der allseits beliebte Apache Server verlangt z.B., dass dem Verzeichnis, in dem sich die besagte Seite befindet, eine Datei namens *.htaccess* hinzugefügt wird, in der alle Zugriffsbeschränkungen aufgeführt sind.

Im Zuge der Schritte 3 und 4 wird die Seite beschafft. Ob sie aus dem Cache entnommen werden kann, hängt von den Verarbeitungsregeln ab. Seiten, die durch Ausführung eines Programms erzeugt werden, können möglicherweise nicht zwischengespeichert werden, da bei jeder Programmausführung ein anderes Ergebnis erzeugt wird. Aber auch normale Seiten sollten gelegentlich auf Aktualität überprüft werden, um zu erkennen, wenn sich ihre Inhalte geändert haben und die alten Inhalte aus dem Cache entfernt werden können. Für Seiten, an deren Aufbau ein Programm beteiligt ist, kommt erschwerend hinzu, dass dem Programm Konfigurationsparameter oder Eingabedaten zu übergeben sind. Diese Daten werden der Pfadangabe oder anderen Teilen der Anfrage entnommen.

In Schritt 5 wird entschieden, wie die restlichen Teile der Antwort auszusehen haben, die die Seiteninhalte begleiten. Hierzu gehört z.B. der MIME-Typ, der aus der Dateinamenserweiterungen, den ersten Worten in der Datei oder der Programmausgabe, einer Konfigurationsdatei oder diversen anderen Quellen abgeleitet werden kann.

In Schritt 6 wird die Seite über das Netzwerk zurückgeliefert. Um die Leistung zu verbessern, können Client und Server eine einzelne TCP-Verbindung zur Übertragung mehrerer Seiten verwenden. Diese Art der Wiederverwendung setzt allerdings voraus, dass eine Logik vorhanden ist, die Anfragen auf eine gemeinsam genutzte Verbindung abbildet und die einzelnen Antworten so zurückliefert, dass diese den richtigen Anfragen zugeordnet werden können.

Schritt 7 dient administrativen Zwecken und trägt den Vorgang zusammen mit etwaigen weiteren statistischen Daten in das Systemlog ein. Diese Logs können später nach wertvollen Informationen durchsucht werden, beispielsweise um das Benutzerverhalten zu erforschen und herauszufinden, welche Personen auf die Webseite zugreifen.

Cookies

Das Surfen im Web, so wie wir es bisher beschrieben haben, besteht aus einer Folge von separaten Seitenanfragen, d.h., es gibt kein Anmelde- oder Sitzungskonzept. Der Browser sendet eine Anfrage an den Server und erhält eine Datei zurück. Danach vergisst der Server, dass er diesen Client jemals gesehen hat.

Zum Abrufen öffentlich zugänglicher Dokumente ist dieses Modell praktisch perfekt und in den Anfangszeiten des Webs funktionierte es auch hervorragend. Um allerdings unterschiedlichen Benutzern in Abhängigkeit von ihren bisherigen Aktionen auf dem Server unterschiedliche Seiten zurückzuliefern, ist dieses Modell nicht geeignet. Genau dieses Verhalten wird aber für viele Interaktionen mit Websites benötigt. Manche Websites (beispielsweise von Tageszeitungen) verlangen z.B. von den Clients, dass sie sich anmelden (und möglicherweise auch bezahlen), bevor sie die Website nutzen können. Was zu der Frage führt, wie die Server zwischen den Anfragen registrierter Benutzer und anderen unterscheiden können. Ein zweites Beispiel stammt aus dem E-Commerce-Bereich. Betritt ein Benutzer einen elektronischen Laden und legt ab und zu einige Produkte in seinen virtuellen Warenkorb, wie kann dann der Server mitverfolgen, welche Inhalte sich aktuell im Warenkorb befinden? Ein drittes Beispiel sind angepasste Webportale wie Yahoo, die den Benutzer erlauben, personalisierte Anfangsseiten einzurichten und mit den Informationen zu füllen, die sie interessieren (also z.B. die aktuellen Kurse ihrer Aktien oder die letzten Sportnachrichten zu ihrer Lieblingsmannschaft). Wie aber kann der Server die richtige Seite anzeigen, wenn er nicht weiß, wer der Benutzer ist?

Auf den ersten Blick könnte man denken, dass die Server die Benutzer durch Beobachten der IP-Adressen verfolgen. Leider funktioniert dieses Konzept nicht immer. Erstens arbeiten viele Benutzer an gemeinsam genutzten Rechnern, vor allem in Unternehmen, sodass die IP-Adresse nur den Rechner, nicht aber den Benutzer angibt. Zweitens, und dies ist sogar noch schlimmer, verwenden viele ISPs NAT, sodass die ausgehenden Pakete für alle Benutzern die gleiche IP-Adresse tragen. Der Server kann folglich nicht zwischen den einzelnen Rechnern hinter der NAT-Box unterscheiden. Viele ISPs verwenden zudem DHCP, um den Kunden IP-Adressen zuzuweisen. Da diese IP-Adressen im Laufe der Zeit wechseln, wäre nicht auszuschließen, dass ein Server, der seine Besucher anhand ihrer IP-Adressen identifiziert, Sie irgendwann für Ihren Nachbarn hält. Aus diesen Überlegungen folgt, dass Server ihre Besucher eben nicht anhand der IP-Adresse identifizieren können.

Die Lösung dieses Problems ist eine viel kritisierte Technik, die als Cookies bekannt ist. Der Name stammt aus einem alten Programmiererslang, bei dem ein Programm eine Prozedur aufruft und Informationen zurückhält, die es benötigt, um später bestimmte Aufgaben zu erledigen. In diesem Sinn kann ein Unix-Dateideskriptor oder ein Windows-Objekt-Handle als Cookie betrachtet werden. Cookies wurden erstmals 1994 im Netscape-Browser implementiert und später in RFC 2109 offiziell definiert.

Wenn ein Client eine Webseite anfordert, kann der Server zusammen mit der angeforderten Seite weitere Informationen in Form eines Cookies bereitstellen. Ein Cookie ist eine recht kleine, benannte Zeichenfolge (von höchstens 4 KB), die der Server mit

einem Browser assoziieren kann. Das ist noch immer keine direkte Assoziation mit dem Benutzer, aber immerhin schon wesentlich besser als die Verknüpfung mit einer IP-Adresse. Browser speichern die angebotenen Cookies für eine gewisse Zeit in einem eigenen Cookie-Verzeichnis auf der Festplatte des Clients, wo sie in der Regel über mehrere Browseraufrufe hinweg erhalten bleiben – es sei denn, der Benutzer hat die Verwendung von Cookies deaktiviert. Cookies sind einfache Zeichenfolgen, keine ausführbaren Programme. Theoretisch können Cookies zwar durchaus Viren enthalten, da sie aber als Daten behandelt werden, gibt es für den Virus keine offizielle Möglichkeit, ausgeführt zu werden und Schaden anzurichten. Dennoch ist Vorsicht geboten, da Hacker unter Umständen Bugs in den Browser dazu nutzen können, den Virus zu aktivieren.

| Domain | Path | Content | Expires | Secure |
|-----------------|------|------------------------------|----------------|--------|
| toms-casino.com | / | CustomerID=297793521 | 15-10-10 17:00 | Ja |
| jills-store.com | / | Cart=1-00501;1-07031;2-13721 | 11-1-11 14:22 | Nein |
| aportal.com | / | Prefs=Stk:CSCO+ORCL;Spt:Jets | 31-12-20 23:59 | Nein |
| sneaky.com | / | UserID=4627239101 | 31-12-19 23:59 | Nein |

Abbildung 7.22: Einige Beispiele für Cookies.

Ein Cookie kann bis zu fünf Felder enthalten (►Abbildung 7.22). Das Feld *Domain* gibt an, woher das Cookie stammt. Browser sollten stets sicherstellen, dass die Server bezüglich ihrer Domäne auch wirklich die Wahrheit sagen. Eine Domäne sollte nicht mehr als 20 Cookies pro Client speichern. Das Feld *Path* ist ein Pfad in der Verzeichnisstruktur des Servers, der angibt, welche Teile im Dateibaum des Servers das Cookie verwenden dürfen. Dies ist oftmals /, also der ganze Baum.

Das Feld *Content* hat die Form *Name = Wert*. Sowohl *Name* als auch *Wert* können alles sein, was der Server möchte. In diesem Feld werden die Inhalte der Cookies gespeichert.

Das Feld *Expires* gibt an, wann das Cookie abläuft. Fehlt dieses Feld, verwirft der Browser das Cookie, wenn er beendet wird. Ein derartiges Cookie nennt man **temporäres Cookie** (*nonpersistent cookie*). Wenn Uhrzeit und Datum angegeben werden, spricht man von einem **dauerhaften Cookie** (*persistent cookie*), das so lange bestehen bleibt, bis es abläuft. Die Ablaufzeiten werden in der mittleren Greenwich-Zeit angegeben. Um ein Cookie von der Festplatte eines Clients zu entfernen, sendet der Server es einfach erneut, diesmal aber mit einer Ablaufzeit, die in der Vergangenheit liegt.

Zuletzt kann noch das Feld *Secure* gesetzt werden, welches anzeigt, dass der Browser das Cookie nur an einen Server mit geschützter Datenübertragung senden sollte, namentlich SSL/TLS (siehe Kapitel 8). Diese Option wird für E-Commerce-, Bank- und anderen sicheren Anwendungen eingesetzt.

Wir wissen nun, wie man Cookies erhält; wie aber werden sie eingesetzt? Unmittelbar bevor ein Brower eine Seitenanfrage an eine Website schickt, prüft er, ob sich im Cookie-Verzeichnis Cookies befinden, die von der Domäne stammen, für die die Anfrage

bestimmt ist. Wenn ja, werden alle Cookies, die von dieser Domäne stammen (und nur diese), in die Anfrage aufgenommen. Der Server nimmt die Cookies mit der Anfrage entgegen und kann sie in seinem Sinne auswerten.

Sehen wir uns einige mögliche Einsatzbereiche für Cookies an. Das erste Cookie aus Abbildung 7.22 wurde von *toms-casino.com* gesetzt und dient der Identifikation des Kunden. Wenn der Kunde die Website in der nächsten Woche erneut besucht, um wieder etwas Geld zum Fenster hinauszuwerfen, sendet der Browser das Cookie, damit der Server weiß, um wen es sich handelt. Mit der Kundennummer gerüstet kann der Server den Kundendatensatz in einer Datenbank nachschlagen und mit diesen Informationen eine entsprechende Webseite anzeigen. Je nach den bekannten Spielgewohnheiten des Kunden kann diese Seite aus einer Pokerhand, einer Liste der aktuellen Pferderennen oder einem Spielautomaten bestehen.

Das zweite Cookie stammt von *joes-store.com*. Dieses Szenario sieht so aus, dass sich die Kundin im Laden nach interessanten Dingen umsieht, die sie kaufen möchte. Wenn sie ein Schnäppchen findet und darauf klickt, fügt der Server das Produkt in den Einkaufswagen (der auf dem Server verwaltet wird), erstellt ein Cookie, das den Produktcode enthält, und sendet dieses an den Client zurück. Während die Kundin sich weiter im Laden umsieht und neue Seiten anklickt, wird das Cookie bei jeder neuen Seitenanfrage zurück zum Server gesendet. Kommen weitere Einkäufe hinzu, fügt der Server sie dem Cookie hinzu. Klickt die Kunden schließlich auf ZUR KASSE, dann wird das Cookie, das nun die vollständige Einkaufliste enthält, zusammen mit der Anfrage zum Server gesendet. Auf diese Weise weiß der Server genau, welche Produkte der Kunde kaufen möchte.

Das dritte Cookie ist für ein Webportal. Klickt ein Kunde auf einen Link zu dem Portal, sendet der Browser das Cookie. Dieses weist das Portal an, eine Seite mit Aktienkursen für Cisco und Oracle sowie mit den Football-Ergebnissen der New York Jets anzuseigen. Da ein Cookie bis zu 4 KB groß sein kann, ist hier viel Platz für spezielle Wünsche wie Zeitungsüberschriften, lokales Wetter, Sonderangebote etc.

Weitaus kontroverser als die obigen Szenarien ist die Verwendung von Cookies zur Beobachtung des Onlineverhaltens der Benutzer. Die Betreiber einer Website können an den so gewonnenen Daten ablesen, wie Besucher sich durch ihre Site bewegen, Werbeagenturen können Profile der Werbelinks oder Sites erstellen, die eine bestimmte Person besucht hat. Kritisch wird dabei vor allem gesehen, dass die Nutzer meist gar nicht wissen, dass ihre Aktivitäten überwacht werden, dass umfangreiche, detaillierte Profile erstellt werden und dass die Überwachung über mehrere Websites erfolgt, die augenscheinlich gar nicht zusammengehören. Wie auch immer, **Webtracking** ist ein großes Geschäft. DoubleClick, eine Website zur Erstellung und Überwachung von Werbung, gehört laut Alexa, einem Web-Monitoring-Unternehmen, zu den 100 geschäftigsten Websites auf der Welt. Google Analytics, welches für Website-Betreiber die Nutzung ihrer Site überwacht, wird von mehr als der Hälfte der 100 000 geschäftigsten Sites im Web genutzt.

Mithilfe von Cookies können Server die Bewegungen ihrer Besucher ganz einfach verfolgen. Angenommen, ein Server möchte feststellen, wie viele einzelne Besucher

vorbeigeschaut haben und wie viele Seiten jeder dieser Besucher aufgerufen hat, bevor er die Site verlassen hat. Kommt die erste Anfrage herein, gibt es noch kein begleitendes Cookie, sodass der Server ein Cookie mit *Counter* = 1 zurücksendet. Wird die Site später wieder angeklickt, so wird das Cookie an den Server zurückgesendet. Jedes Mal wird der Zähler um eins erhöht und an den Client zurückgesendet. Durch Auswertung der Zählerstände kann der Server erkennen, wie viele Personen nach der ersten Seite, nach der zweiten Seite usw. aufgegeben.

Das Überwachen des Surferverhaltens über Site-Grenzen hinweg ist nur unwesentlich komplizierter und funktioniert wie folgt. Eine Werbeagentur, sagen wir mal Schnüffler-Werbung, kontaktiert größere Websites und platziert Bannerwerbungen für die Produkte seiner Kunden auf deren Seiten, wofür die Site-Besitzer eine gewisse Gebühr erhalten. Anstatt den Sites die Werbung in Form einer GIF-Datei zu übersenden, welche die Site-Betreiber auf den einzelnen Seiten platzieren können, schicken sie eine URL, die in jede Seite eingefügt wird. Jede ausgegebene URL enthält im *Path*-Feld eine eindeutige Nummer, z.B.:

<http://www.schnueffler.com/382674902342.gif>

Wenn ein Benutzer eine Seite *P*, die eine solche Anzeige enthält, das erste Mal besucht, ruft der Browser die HTML-Seite ab. Der Browser untersucht die HTML-Datei und findet einen Link auf die Bilddatei auf www.schnueffler.com. Folglich sendet er eine Anfrage für das Bild. Als Antwort erhält er eine GIF-Datei mit der Werbung und ein Cookie, das eine eindeutige Benutzerkennung enthält, wie z.B. 4627239101 (siehe Abbildung 7.22). Schnüffler-Werbung hält daraufhin fest, dass der Benutzer mit dieser Kennung die Seite *P* besucht hat. Dies ist möglich, weil auf die angeforderte Datei (*382674902342.gif*) nur von der Seite *P* verwiesen wird. Die eigentliche Werbeanzeige kann natürlich identisch auf Tausenden von Seiten erscheinen, hat dann aber jedes Mal einen anderen Dateinamen. Und Schnüffler-Werbung berechnet dem Produkthersteller vermutlich für jede ausgelieferte Anzeige einen halben Cent oder so.

Besucht der Benutzer später eine andere Webseite mit Anzeigen von Schnüffler-Werbung, holt der Browser zuerst die HTML-Datei vom Server, entdeckt auf der Seite den Link zu, sagen wir <http://www.schnueffler.com/193654919923.gif>, und fordert diese Datei an. Da er bereits ein Cookie von der Domäne [schnueffler.com](http://www.schnueffler.com) besitzt, nimmt der Browser das Cookie von Schnüffler, das die Benutzerkennung enthält, mit auf. Jetzt kennt Schnüffler-Werbung bereits eine zweite Seite, die der Benutzer besucht hat.

Nach einer gewissen Zeit kann Schnüffler-Werbung ein vollständiges Benutzerprofil der Surfgewohnheiten des Benutzers zeichnen, selbst wenn der Benutzer nie auf eine der Anzeigen geklickt hat. Natürlich kennt Schnüffler-Werbung noch nicht den Namen des Benutzers (obwohl sie die IP-Adresse haben, welche vielleicht schon genügt, um den Namen in einer Datenbank zu finden). Sobald aber der Benutzer seinen Namen jemals bei einer Site angibt, deren Eigentümer mit Schnüffler-Werbung kooperiert, ist ein vollständiges Profil samt Namen verfügbar, das an beliebige Interessenten verkauft werden kann. Der Verkauf dieser Informationen ist für Schnüffler-Werbung unter Umständen so profitabel, dass die Firma weitere Anzeigen auf anderen Webseiten platziert, um noch mehr Informationen zu erfassen.

Und wenn Schnüffler-Werbung gänzlich unbemerkt schnüffeln möchte, kann die Firma auch noch auf die eine klassische Bannerwerbung verzichten. Eine „Werbung“, die aus einem einzigen Pixel in der Hintergrundfarbe der Webseite besteht (und daher unsichtbar ist), hat genau die gleiche Wirkung wie eine Bannerwerbung: Der Browser holt sich die 1×1 -Pixel-Grafik und sendet alle Cookies, die zu der Domäne des Pixels gehören.

Aktivitäten wie die oben beschriebene haben dem Ruf der Cookies sehr geschadet und sie ins Zentrum der Debatte über den Schutz der Privatsphäre im Internet gerückt. Besonders bedenklich an den geschilderten Machenschaften ist, dass die meisten Benutzer gar nicht mitbekommen, dass Daten über sie gesammelt werden, ja sich sogar in Sicherheit wiegen, da sie keine Anzeigen angeklickt haben. Cookies, die Benutzer über Website-Grenzen hinweg beobachten, werden daher mittlerweile von vielen als **Spyware** betrachtet. Werfen Sie einmal einen Blick auf die Cookies, die von Ihrem Browser gespeichert wurden. Die meisten Browser zeigen diese Informationen gemeinsam mit den Einstellungen zum Schutz der Privatsphäre. Vielleicht werden Sie zu Ihrem Erstaunen Namen, E-Mail-Adressen, Passwörter und seltsame Bezeichner darin finden. Und auch wenn Sie (hoffentlich) keine Kreditkartennummern finden, dürfte klar sein, dass hier ein erhebliches Potenzial für Missbrauch besteht.

Um dennoch so etwas wie eine Privatsphäre aufrechtzuerhalten, konfigurieren manche Benutzer ihre Browser so, dass diese überhaupt keine Cookies akzeptieren – und handeln sich anderweitigen Ärger ein, da viele Websites auf Cookies angewiesen sind. Als mögliche Alternative erlauben die meisten Browser das Blockieren von **Cookies von Drittanbietern** (*third-party cookie*). Gemeint sind Cookies, die nicht von der gleichen Site wie die Hauptseite stammen – wie z.B. das *schnueffler.com*-Cookie aus unserem Beispiel, wenn es über die Seite *P* einer ganz anderen Website aktiviert wird. Durch das Blockieren dieser Cookies wird das Verfolgen von Benutzern über Website-Grenzen erschwert. Zusätzlich können Browsererweiterungen installiert werden, über die der Benutzer detailliert festlegen kann, wie Cookies verwendet werden (oder besser eben nicht verwendet werden). Die fortgesetzte Diskussion über Cookies und den Schutz der Privatsphäre im Internet hat dazu geführt, dass viele Unternehmen mittlerweile eigene Richtlinien aufstellen, wie Sie den Austausch von Daten zwischen den Unternehmen beschränken wollen, um Missbrauch zu verhindern. Meist geben diese Richtlinien lediglich grob an, wie die Firma mit den Daten umgeht, etwa: „Die über Sie gesammelten Daten werden ausschließlich zur Betreibung unseres Geschäfts verwendet“ – was möglicherweise bedeutet, dass die Daten verkauft werden.

7.3.2 Statische Webdokumente

Das Grundprinzip im Web ist die Übertragung von Webseiten vom Server auf den Client. In der einfachsten Form sind Webseiten statisch. Das heißt, es handelt sich einfach um Dateien, die auf einem Server liegen und jedes Mal, wenn sie abgerufen und angezeigt werden, gleich aussehen. Was allerdings nicht bedeutet, dass statische Seiten im Browser komplett inaktiv wären – schließlich kann auch eine Seite, die ein Video enthält, eine statische Webseite sein.

Wie bereits weiter oben erwähnt wurde, ist die Lingua Franca des Webs, in der die meisten Seiten geschrieben werden, HTML. Die Homepages von Lehrern sind meist statische HTML-Seiten, während es sich bei den Homepages von Unternehmen üblicherweise um dynamische Seiten handelt, die von professionellen Webdesignern erstellt wurden. In diesem Abschnitt werfen wir als Vorbereitung für später einen kurzen Blick auf die Erstellung statischer HTML-Seiten. Leser, die mit HTML bereits vertraut sind, können diesen Abschnitt überspringen und gleich mit dem nächsten Unterkapitel fortfahren, wo es um dynamische Inhalte und Webservices geht.

HTML – HyperText Markup Language

HTML (*HyperText Markup Language*) wurde zusammen mit dem Web eingeführt. Mithilfe von HTML kann man Webseiten erstellen, die Text, Grafik, Video und Verweise auf andere Webseiten enthalten. HTML ist eine Markup-Sprache zur Beschreibung, d.h. eine Sprache, die beschreibt, wie Dokumente formatiert werden sollen. Der Begriff „Markup“ stammt aus den alten Tagen, als Texter und Schriftsteller Dokumente manuell gekennzeichnet haben, um dem Drucker – in jenen Tagen ein menschliches Wesen – anzugeben, welche Schriftarten und Hervorhebungen zu verwenden sind. Markup-Sprachen enthalten somit explizite Formatierungsbefehle. Beispielsweise gibt in HTML die Zeichenfolge `` den Beginn der Fettformatierung und `` das Ende der Fettformatierung an. LaTeX und TeX sind weitere Beispiele für Markup-Sprachen, die vor allem in Akademikerkreisen bekannt sind.

Der besondere Vorteil einer Markup-Sprache gegenüber einer Sprache ohne explizites Markup ist die Trennung von Inhalt und Darstellung. Für die Implementierung eines Browsers bedeutet dies: Er muss die Markup-Befehle verstehen und braucht sie dann nur noch richtig auf den Inhalt anzuwenden. Durch die Einbettung der standardisierten Markup-Befehle in die HTML-Dateien ist jeder Webbrower in der Lage, jede beliebige Webseite zu lesen und neu formatiert anzuzeigen. Das erneute Formatieren von Webseiten nach deren Empfang ist wichtig, da es ja gut möglich ist, dass eine Seite z.B. auf einem High-End-Rechner mit einem $1\,600 \times 1\,200$ -Bildschirm mit 24-Bit-Farbtiefe erstellt wurde, aber auch auf einem Handy mit einem 640×320 angezeigt werden soll.

Grundsätzlich kann man HTML-Dokumente mit jedem beliebigen Standardeditor aufsetzen – was viele auch tun. Schneller geht es meist mit richtigen Textverarbeitungssystemen oder spezialisierten HTML-Editoren, die dem Autor einen Großteil der Arbeit abnehmen (ihm aber dementsprechend auch weniger Kontrolle über die Feinheiten des endgültigen Ergebnisses lassen).

Wie eine einfache, in HTML geschriebene Webseite aussieht und in einem Browser angezeigt wird, sehen Sie in ▶ Abbildung 7.23. Eine korrekt erstellte Webseite besteht aus einem Kopfteil (Head) und einem Hauptteil (Body). Beide Teile sollten zwischen den **Tags** (Formatierungsbefehlen) `<html>` und `</html>` eingefasst werden, obwohl sich die meisten Browser nicht beschweren, falls diese Tags fehlen. Wie Sie in ▶ Abbildung 7.23a sehen können, wird der Kopfteil in die Tags `<head>` und `</head>` und der Hauptteil in die Tags `<body>` und `</body>` eingefasst. Die Zeichenketten in den Tags nennt man **Directives** (Anweisungen). Dieses Format ist für die meisten HTML-Tags – wenn auch nicht alle – typisch: mit `<etwas>` wird der Beginn von etwas angezeigt und mit `</etwas>` das Ende.

```

<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
 <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
  <li> <a href="http://widget.com/products/big"> Big widgets </a> </li>
  <li> <a href="http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
  <li> By telephone: 1-800-WIDGETS </li>
  <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>

```

Abbildung 7.23: (a) HTML-Code für eine Beispielwebseite.

Tags können groß- oder kleingeschrieben werden. Daher bedeuten `<head>` und `<HEAD>` das Gleiche – für bestmögliche Kompatibilität empfiehlt sich allerdings die Kleinschreibung. Das Layout des HTML-Dokuments ist irrelevant. HTML-Parser ignorieren Leerzeichen und Zeilenumbrüche, da sie den Text sowieso umformatieren müssen, damit er in den aktuellen Anzeigebereich passt. Folglich können Autoren von HTML-Seiten nach Belieben Leerraum einfügen, um ihre HTML-Dokumente übersichtlicher zu gestalten – was leider viel zu oft vernachlässigt wird. Es bedeutet aber auch, dass Leerzeilen, da sie ja ignoriert werden, nicht zum Trennen von Absätzen benutzt werden können. Dafür ist ein explizites Tag erforderlich.

Manche Tags haben (benannte) Parameter, sogenannte Attribute. Nehmen Sie z.B. das ``-Tag aus Abbildung 7.23, das dazu benutzt wird, ein Bild in den Text einzufügen. Es besitzt zwei Attribute: `src` und `alt`. Das erste Attribut gibt die URL des Bildes an. Was die erlaubten Grafikformate angeht, so schweigt sich der HTML-Standard dazu aus. In der Praxis sieht es aber so aus, dass alle Browser GIF- und JPEG-Dateien unterstützen. Darüber hinaus steht es den Browsern frei, noch weitere Formate zu unterstützen – doch derlei Erweiterungen sind immer ein zweischneidiges Schwert. Ist ein Benutzer daran gewöhnt, dass sein Browser z.B. TIFF-Dateien unterstützt, fügt er diese vielleicht in seine Webseiten ein – und muss später zu seiner Überraschung feststellen, dass sein wunderbares Kunstwerk in einem anderen Browser einfach ignoriert wird.

Das zweite Attribut gibt einen alternativen Text an, der verwendet wird, wenn das Bild nicht angezeigt werden kann. Für jedes Tag definiert der HTML-Standard eine Liste der zulässigen Parameter und, sofern es Parameter gibt, was sie bedeuten. Da jeder Parameter benannt ist, spielt die Reihenfolge, in der die Parameter angegeben werden, keine Rolle.

Technisch werden HTML-Dokumente im Zeichensatz ISO 8859-1 Latin-1 geschrieben. Für Benutzer mit Tastaturen, die nur ASCII unterstützen, gibt es aber auch Escape-

Zeichenfolgen (Umschaltzeichen) für Sonderzeichen, wie z.B. è. Die Aufstellung der Sonderzeichen ist im Standard enthalten. Alle beginnen mit einem kaufmännischen Und (&) und enden mit einem Semikolon. So erzeugt beispielsweise ein Leerzeichen, è ein è und ´ ergibt ein é. Da <, > und & eine besondere Bedeutung haben, können sie nur über ihre Escape-Folgen <, > und & ausgedrückt werden.

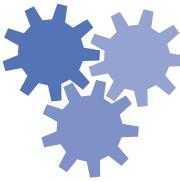
Das wichtigste Element im Kopfteil ist der Titel, der in die Tags <title> und </title> eingefasst wird. Darüber hinaus können noch bestimmte Metainformationen eingefügt werden (worauf wir in unserer Beispieleite allerdings verzichtet haben). Der Titel selbst wird nicht auf der Seite angezeigt. Einige Browser verwenden ihn, als Titel für das Fenster, in dem die Seite angezeigt wird.

In der Seite aus Abbildung 7.23 werden auch diverse Überschriften verwendet. Überschriften werden durch ein <hn>-Tag erzeugt, wobei n eine Ziffer von 1 bis 6 ist. <h1> ist die Überschrift der ersten Ebene, <h6> ist die der niedrigsten Ebene. Der Browser entscheidet aber letztendlich, wie die Überschriften am Bildschirm angezeigt werden. Meist werden die Überschriften mit den niedrigeren Ebenennummern in einer größeren und stärkeren Schriftart angezeigt. Eventuell wählt der Browser auch für jede Überschriftenebene eine eigene Farbe. Überschriften der Ebene <h1> werden üblicherweise in einer großen Schrift und fett dargestellt, mit mindestens einer Leerzeile ober- und unterhalb. Für Überschriften der Ebene <h2> wird dann eine kleinere Schrift mit weniger Leerraum darüber und darunter verwendet.

Die Tags und <i> schalten Fett- (Bold) bzw. Kursivschnitt (Italic) ein. Das Tag <hr> erzwingt einen Zeilenumbruch und zeichnet eine horizontale Linie quer über den Bildschirm.

Mit <p> wird ein neuer Absatz begonnen. Der Browser kann neue Absätze z.B. durch Einfügen einer Leerzeile und durch Einrücken darstellen. (Das </p>-Tag, welches das Ende eines Absatzes markiert, wird von faulen HTML-Autoren häufig weggelassen.)

Welcome to AWI's Home Page



We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope you will find all the information you need here.

Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by email.

Product Information

- [Big widgets](#)
- [Little widgets](#)

Contact information

- By telephone: 1-800-WIDGETS
- By email: info@amalgamated-widget.com

Abbildung 7.23: (b) Die formatierte Seite.

HTML bietet verschiedene Mechanismen zum Erstellen von Listen, verschachtelte Listen mit eingeschlossen. Ungeordnete Listen, wie die aus Abbildung 7.23, beginnen mit ``, die einzelnen Listenelemente beginnen mit ``. Daneben gibt es noch das Tag ``, mit dem eine geordnete Liste begonnen wird. Die Elemente ungeordneter Listen werden oft mit einem vorangestellten Aufzählungszeichen (•) dargestellt. Die Elemente geordneter Listen werden vom Browser durchnummeriert.

Und dann sind da noch die Hyperlinks, die mit den Tags `<a>` und `` (das „a“ steht für „anchor“, also Anker) erzeugt werden, siehe Abbildung 7.23. Zu dem Tag `<a>` gibt es verschiedene Parameter, unter denen `href`, der die verknüpfte URL angibt, der wichtigste ist. Der Text zwischen den Tags `<a>` und `` wird angezeigt. Wenn der Benutzer ihn anklickt, folgt er dem Hyperlink zu einer neuen Seite. Es können auch andere Elemente mit Hyperlinks verknüpft werden – wie z.B. Bilder. Dazu genügt es, mittels `` ein Bild zwischen den Tags `<a>` und `` einzufügen. Das Bild wird dann auf der Webseite angezeigt und wenn es angeklickt wird, wird der Hyperlink aktiv.

Es gibt noch zahlreiche weitere HTML-Tags und -Attribute, die wir in unserem einfachen Beispiel nicht gesehen haben. Beispielsweise kann für das Tag `<a>` der Parameter `name` gesetzt werden, wodurch ein Hyperlink-Ziel definiert wird. Auf diese Weise können Hyperlinks auf Positionen innerhalb der Seite verweisen. Einige Webseiten beginnen z.B. mit einem anklickbaren Inhaltsverzeichnis. Klickt der Benutzer auf einen Eintrag im Inhaltsverzeichnis, gelangt er direkt an die entsprechende Stelle in der Seite. Ein weiteres Beispiel ist das Tag `
`, welches einen Zeilenumbruch erzwingt.

Welchen Effekt Tags haben, prägt sich am besten ein, wenn man sie in Aktion sieht. Besuchen Sie dazu einfach eine beliebige Webseite und lassen Sie sich von Ihrem Browser den zugehörigen originalen HTML-Quelltext anzeigen, um zu sehen, wie die Seite aufgebaut wurde. Die meisten Browser bieten hierfür einen Menübefehl namens SEITENQUELLTEXT ANZEIGEN (oder ähnlich) an.

Im Zuge unseres Streifzugs haben wir Ihnen die Tags vorgestellt, die seit Beginn des Webs existieren. Doch HTML entwickelt sich ständig weiter. In ►Abbildung 7.24 haben wir einige der Elemente, die im Laufe der Zeit von Version zu Version hinzugekommen sind, zusammengetragen. HTML 1.0 bezieht sich auf jene HTML-Version, die bei Einführung des Webs verwendet wurde. Die HTML-Versionen 2.0, 3.0 und 4.0 folgten innerhalb weniger Jahre aufeinander, während das Web förmlich explodierte. Nach Einführung von HTML 4.0 vergingen fast zehn Jahre, bevor sich ein klarer Weg zur Standardisierung der nächsten Hauptversion, HTML 5.0, abzeichnete. Da es sich um ein wegweisendes Upgrade handelt, das festlegen soll, wie Browser die verschiedenen medialen Inhalte verarbeiten, dauern die Arbeiten an HTML 5.0 noch an und es wird frühestens im Jahr 2012 mit der offiziellen Verabschiedung des neuen Standards gerechnet. Die in HTML 5.0 eingeführte Funktionalität wird aber bereits heute von allen großen Browsern unterstützt.

Angetrieben wurde die Entwicklung der verschiedenen HTML-Versionen vor allem von der Einführung immer neuer Elemente und Möglichkeiten – Features, die Webautoren und -designer unbedingt haben wollten, die sie aber vor deren Aufnahme in

den Standard nur auf nicht standardisierten Wegen nutzen konnten (beispielsweise in Form von Plug-ins). So gab es z.B. in HTML 1.0 und HTML 2.0 keine Tabellen. Diese wurden erst in HTML 3.0 eingeführt. Eine HTML-Tabelle besteht aus einer oder mehreren Zeilen, mit jeweils einer oder mehreren Tabellenzellen, die ein breites Spektrum an Inhaltselementen aufnehmen können (wie z.B. Text, Bilder oder andere Tabellen). Vor HTML 3.0 mussten Webautoren, die Tabellen einbauen wollten, auf andere Wege sinnen – wie z.B. die Einbettung eines Bildes, das eine Tabelle zeigt.

In HTML 4.0 wurden weitere Funktionen hinzugefügt. Hierzu gehören barrierefreie Zugriffsmöglichkeiten für Benutzer mit Behinderungen, die Einbettung von Objekten (eine Verallgemeinerung des ``-Tags, damit auch andere Objekte in die Seiten eingebettet werden können), Unterstützung für Skriptsprachen (für dynamische Inhalte) und anderes mehr.

| Funktion | HTML 1.0 | HTML 2.0 | HTML 3.0 | HTML 4.0 | HTML 5.0 |
|-------------------------------|----------|----------|----------|----------|----------|
| Hyperlinks | x | x | x | x | x |
| Bilder | x | x | x | x | x |
| Listen | x | x | x | x | x |
| ImageMaps | | x | x | x | x |
| Formulare | | x | x | x | x |
| Gleichungen | | | x | x | x |
| Werkzeugleisten | | | x | x | x |
| Tabellen | | | x | x | x |
| Barrierefreiheit | | | | x | x |
| Einbettung von Objekten | | | | x | x |
| Stylesheets | | | | x | x |
| Skriptsprachen | | | | x | x |
| Video und Audio | | | | | x |
| Eingegebettete Vektorgrafiken | | | | | x |
| XML-Darstellung | | | | | x |
| Hintergrundthreads | | | | | x |
| Browserspeicher | | | | | x |
| Canvas (Zeichenflächen) | | | | | x |

Abbildung 7.24: Auswahl der wichtigsten Unterschiede zwischen den HTML-Versionen.

HTML 5.0 offeriert viele neue Optionen für die Verwendung der verschiedenen Medien, wie sie mittlerweile überall im Web zu finden sind. Video und Audio können in Seiten eingefügt und vom Browser abgespielt werden, ohne dass der Benutzer dazu ein Plug-in installieren muss. Als Alternative zu Rastergrafikformaten (wie JPEG oder GIF) können Zeichnungen im Browser als Vektorgrafiken aufgebaut werden. Noch mehr Unterstützung gibt es für Skripts, die im Browser ausgeführt werden, so z.B. Hintergrundthreads zur Durchführung aufwendiger Berechnungen und den Zugriff auf externen Speicher. Alle diese neuen Möglichkeiten dienen der Unterstützung von Webseiten, die eher wie traditionelle Anwendungen mit einer grafischen Benutzeroberfläche statt wie Dokumente aussehen. Sie sind der neue Trend im Web.

Eingaben und Formulare

Eine wichtige Option, auf die wir noch gar nicht zu sprechen gekommen sind, ist das Entgegennehmen von Benutzereingaben. HTML 1.0 war im Grunde eine Einbahnstraße. Benutzer konnten Seiten von Informationsanbietern abrufen, es gab aber kaum eine Möglichkeit, Informationen in die andere Richtung zu senden. Schnell wurde jedoch klar, dass ein bidirektonaler Datenaustausch benötigt wird, damit z.B. Produktbestellungen über Webseiten aufgegeben, Registrierkarten elektronisch ausgefüllt, Suchbegriffe eingegeben werden können und vieles mehr.

Um Eingaben vom Benutzer (über den Browser) zum Server senden zu können, müssen zwei Voraussetzungen erfüllt sein. Die erste Voraussetzung ist, dass HTTP in der Lage ist, Daten in diese Richtung zu übertragen. Wie dies bewerkstelligt wird, beschreiben wir in einem späteren Abschnitt; es sei aber schon verraten, dass dabei die *POST*-Methode verwendet wird. Die zweite Voraussetzung ist, dass es eine Möglichkeit gibt, Benutzeroberflächenelemente zu präsentieren, über die Eingaben eingesammelt und verpackt werden können. Diese Funktionalität bieten die in HTML 2.0 eingeführten **Formulare**.

Formulare enthalten Felder oder Schaltflächen, über die Benutzer Daten eingeben oder eine Auswahl treffen und dann an den Seiteneigentümer senden können. Formulare werden auf die gleiche Weise definiert wie andere HTML-Komponenten (►Abbildung 7.25). Zudem ist zu beachten, dass es sich noch immer um statische Inhalte handelt. Formulare zeigen immer das gleiche Verhalten, unabhängig davon, wer sie verwendet. Dynamische Inhalte, auf die wir später noch zu sprechen kommen, stellen einen ausgefilterten Mechanismus zum Einsammeln von Daten dar, indem sie ein Programm schicken, dessen Verhalten unter Umständen von der Browserumgebung abhängt.

Wie alle Formulare ist auch das Formular aus Abbildung 7.25 zwischen den Tags `<form>` und `</form>` eingefasst. Die Attribute dieses Tags geben an, wie mit den Daten, die Eingaben darstellen, zu verfahren ist – in unserem Fall werden die Daten vermittels der *POST*-Methode an die angegebene URL geschickt. Text, der nicht in einem Tag eingeschlossen ist, wird einfach angezeigt. Ansonsten sind die üblichen Tags (wie z.B. ``) alle auch in Formularen zugelassen, damit der Autor der Seite steuern kann, wie das Formular auf dem Bildschirm aussehen wird.

In dem Formular werden drei Arten von Eingabefeldern benutzt, die alle mit dem `<input>`-Tag beginnen. Das `<input>`-Tag besitzt diverse Parameter zum Festlegen der Größe, des Typus und der Art und Weise, wie der Benutzer mit dem angezeigten Feld interagiert. Die am häufigsten verwendeten Typen sind leere Felder zur Aufnahme von Text, Kästen, die markiert werden können, und Abschicken-Schaltflächen (*submit*), die die Übertragung der Daten an den Server veranlassen.

Die erste Art von Eingabefeld ist ein Textfeld (*text*), wie es z.B. auf den Text „Name“ folgt. Das Textfeld ist 46 Zeichen lang und wartet darauf, dass der Benutzer eine Textfolge eingibt, die dann in der Variablen *customer* gespeichert wird.

In der nächsten Zeile des Formulars wird der Benutzer zur Eingabe seiner Straße aufgefordert. Danach folgt eine Zeile zur Eingabe der Stadt, des Bundesstaates und des Landes. Da zwischen diesen Feldern kein `<p>`-Tag eingefügt wurde, zeigt der Browser diese Felder nicht in eigenen Absätzen, sondern zusammen in einer Zeile an – sofern sie in eine Zeile passen. Aus Sicht des Browsers enthält dieser Absatz einfach sechs Elemente: drei Zeichenketten, die sich mit drei Textfeldern abwechseln. Die nächste Zeile fordert zur Eingabe der Kreditkartennummer und des Ablaufdatums auf. Kreditkartennummern sollten nur über das Internet übertragen werden, wenn ausreichende Sicherheitsmaßnahmen getroffen wurden. Wir werden dieses Thema in *Kapitel 8* ausführlich behandeln.

Unter dem Eingabefeld für das Ablaufdatum der Kreditkarte stoßen wir auf eine neue Art von Eingabefeld: Optionsfelder (*radio button*). Optionsfelder werden bereitgestellt, wenn der Benutzer eine Auswahl aus mehreren Optionen treffen soll. Diese funktionieren nach dem gleichen Prinzip wie ein Autoradio mit einem halben Dutzend Tasten zur Auswahl der Stationen. Klickt der Benutzer auf eine der Tasten (Optionsfelder), werden alle anderen ausgeschaltet. Die grafische Darstellung ist Sache des Browsers. Für die Abfrage der Gerätgröße (*Widget size*) wurden ebenfalls zwei Optionsfelder verwendet. Auseinandergehalten werden die beiden Optionsfelder-Gruppen nicht etwa durch statische Gültigkeitsbereiche wie z.B. eine einfassende `<radiobutton> ... </radiobutton>`-Konstruktion, sondern durch ihr *name*-Feld.

Der Parameter *value* wird verwendet, um anzugeben, welches Optionsfeld ausgewählt wurde. Abhängig davon, welche Kreditkartenoption der Benutzer ausgewählt hat, wird die Variable *cc* auf die Zeichenkette „mastercard“ oder „visacard“ gesetzt.

```

<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/order.cgi" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size =4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
Expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">

```

```
Little <input name="product" type="radio value="cheap">
Ship by express courier <input name="express" type="checkbox" />
<p><input type="submit value="Submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

a

Widget Order Form

Name

Street address

City State Country

Credit card # Expires M/C Visa

Widget size Big Little Ship by express courier

Thank you for ordering an AWI widget, the best widget money can buy!

b

Abbildung 7.25: (a) HTML-Code für ein Bestellformular (b) Die formatierte Seite.

Auf die beiden Gruppen von Optionsfeldern folgt eine Lieferoption, die durch ein Kontrollkästchen (Eingabefeld vom Typ *checkbox*) repräsentiert wird. Kontrollkästchen sehen meist genauso aus wie Optionsfelder, unterscheiden sich aber durch eine wichtige Eigenschaft: Im Unterschied zu Optionsfeldern, bei denen genau eines ausgewählt werden muss, sind Kontrollkästchen unabhängig voneinander, sodass jedes aktiviert oder deaktiviert sein kann.

Damit wären wir bei der *submit*-Schaltfläche angelangt. Zeichenkette *value* ist der Titel der Schaltfläche, der vom Browser angezeigt wird. Klickt der Benutzer auf die *submit*-Schaltfläche, packt der Browser die gesammelten Eingabedaten in eine einzelne lange Zeile und sendet diese zurück an den Server, an die URL, die als Teil des *<form>*-Tags angegeben ist. Die Codierung dieser Zeile ist relativ simpel: Zum Trennen der Felder wird das Zeichen & benutzt; Leerzeichen werden durch das Pluszeichen + repräsentiert. Für unser Beispielformular könnte die Zeile z.B. wie in ► Abbildung 7.26 aussehen.

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&
state=NY&country=USA&cardno=1234567890&expires=6/14&cc=mastercard&
product=cheap&express=on
```

Abbildung 7.26: Eine mögliche Antwort, wie sie vom Browser an den Server gesendet wird, mit den vom Benutzer eingegebenen Daten.

Die Zeichenfolge wird in einer Zeile an den Server gesendet. (In Abbildung 7.26 wurde die Zeile zwangsweise in drei Zeilen umgebrochen, da die Seite nicht breit genug ist.) Die Interpretation der Zeichenkette wird dem Server überlassen. In den meisten Fällen wird er die gesendeten Daten zur weiteren Verarbeitung an ein Programm übergeben; mehr dazu im nächsten Abschnitt.

Es gibt noch weitere Arten von Eingabefeldern, die in unserem einfachen Beispiel nicht auftauchten – wie z.B. *password* und *textarea*. Ein Eingabefeld vom Typ *password* verhält sich wie ein normales Textfeld (der Standardtyp *text* muss nicht explizit angegeben werden), nur dass die Eingabe nicht am Bildschirm angezeigt wird. Eingabefelder vom Typ *textarea* sind Textfelder, die mehrere Zeilen umfassen können.

Für Fälle, in denen eine Auswahl aus einer langen Liste von Optionen zu treffen ist, gibt es die Tags `<select>` und `</select>`, die eine Gruppe von Alternativen umschließen und meist in Form eines Drop-down-Menüs angezeigt werden. Die Semantik ist die gleiche wie bei einer Gruppe von Optionsfeldern, vorausgesetzt der Parameter *multiple* wurde nicht gesetzt. In letzterem Fall entspricht das Verhalten dem einer Gruppe von Kontrollkästchen.

CSS – Cascading Style Sheets

Das ursprüngliche Ziel von HTML war es, die Struktur des Dokuments anzugeben, nicht seine Darstellung. Zum Beispiel weist

```
<h1> Deborah's Photos </h1>
```

den Browser an, die Überschrift hervorzuheben, sagt aber nichts über die Schriftart, Schriftgröße oder Farbe aus. Die Auswahl dieser Parameter bleibt dem Browser überlassen, der die Eigenschaften des Bildschirms (z.B. wie viele Pixel er hat) kennt. Da aber viele Webseiten-Designer die Darstellung ihrer Seiten bis ins Detail festlegen wollten, wurden neue Tags in den Standard aufgenommen, die das Erscheinungsbild steuern, wie z.B.:

```
<font face="helvetica" size="24" color="red"> Deborah's Photos </font>
```

Auch wurden Techniken hinzugefügt, mit deren Hilfe man die Positionierung auf dem Bildschirm genau vorgeben konnte. Der Nachteil dieser Techniken ist jedoch, dass sie zum einen recht mühsam ist und zum anderen aufgeblasenes HTML erzeugt, das nicht portabel ist. Das heißt, eine Seite, die in dem Browser, in dem sie entwickelt wurde, perfekt dargestellt wird, kann in einem anderen Browser, ja selbst in einer anderen Version desselben Browsers vollständig durcheinandergeraten.

Ein weitaus besserer Ansatz ist die Verwendung von Stylesheets. In Textverarbeitungssystemen erlauben Stylesheets (dort auch Formatvorlagen genannt) dem Autor, einzelnen Textpassagen logische statt physikalische Formatierungen (Stile) zuzuweisen – also beispielsweise „einleitender Absatz“ statt „kursiver Text“. Definiert werden die Stile einmalig an separater Stelle. Auf diese Weise muss der Autor, wenn er beschließt, alle einleitenden Absätze statt wie bis bisher in Blau und in kursiver 14-Punkt-Schrift doch lieber in einem schrillen Rosa und einer fetten 18-Punkt-Schrift

darstellen möchte, nichts weiter tun, als eine Definition zu ändern, und die Änderung wird automatisch für das gesamte Dokument übernommen.

CSS (*Cascading Style Sheets*) führte die Stylesheets bereits mit HTML 4.0 in das Web ein. Es sollte jedoch bis zum Jahr 2000 dauern, bevor CSS unter den Webdesignern populär und von den Browsern allgemein unterstützt wurde. CSS definiert eine einfache Sprache zum Formulieren von Regeln, die das Aussehen durch Tags markierter Inhalte beschreiben. Betrachten wir ein Beispiel. Angenommen, AWI wünscht todschicke Webseiten mit marineblauem Text in Arial-Schrift vor eierschalenfarbenem Hintergrund und Überschriftenebenen, die um 100 % bzw. 50 % größer sind als der normale Text. Die zugehörigen Regeln sind in der CSS-Definition aus ► Abbildung 7.27 zusammengefasst.

```
body {background-color:linen; color:navy; font-family:Arial;}
h1 {font-size:200%;}
h2 {font-size:150%;}
```

Abbildung 7.27: CSS-Beispiel.

Wie man sehen kann, können Stildefinitionen sehr kompakt sein. Jede Zeile gibt zuerst das Element an, auf die sie sich bezieht, und listet dann die Werte für ausgewählte Eigenschaften auf. Die Eigenschaften eines Elements werden als Vorgabewerte an alle HTML-Elemente zugewiesen, die in dem Element eingebettet sind. So gibt z.B. der Stil für body auch den Stil für alle Textabsätze im body-Abschnitt vor. Viele Farben können dankenswerter über vordefinierte Namen angegeben werden (wie z.B. red). Und für Stilparameter, die nicht definiert wurden, verwendet der Browser seine Standardwerte. Dieses Verhalten bedeutet, dass Stylesheet-Definitionen optional sind; eine vernünftige Darstellung ist auch da sichergestellt, wo Stylesheet-Definitionen fehlen.

Stylesheets können direkt in eine HTML-Datei geschrieben werden (mittels des Tags `<style>`). Gängiger ist es aber, die Stylesheets in eine separate Datei auszulagern und auf sie zu verweisen. ► Abbildung 7.28 demonstriert wie das `<head>`-Tag der AWI-Seite so modifiziert werden kann, dass es auf ein Stylesheet in der Datei `awistyle.css` verweist. An dem Code lässt sich auch ablesen, dass der MIME-Typ von CSS-Dateien `text/css` lautet.

```
<head>
<title> AMALGAMATED WIDGET, INC. </title>
<link rel="stylesheet" type="text/css" href="awistyle.css" />
</head>
```

Abbildung 7.28: Verwendung eines CSS-Stylesheet.

Diese Strategie bringt zwei Vorteile. Erstens: Ein einmal definierter Satz von Stilen kann auf mehrere Seiten einer Website angewendet werden. Dieses Konzept sorgt nicht nur dafür, dass die betreffenden Seiten ein einheitliches Erscheinungsbild aufweisen, selbst wenn sie zu unterschiedlichen Zeiten von unterschiedlichen Autoren aufgebaut wurden. Es bedeutet auch, dass das Aussehen der gesamten Site durch Bearbeitung einer einzigen CSS-Datei angepasst werden kann, ohne Eingriff in den HTML-Code. Man kann dies mit dem Konzept der `#include`-Datei von C-Programmen vergleichen:

Ändert man die Makrodefinition in einer solchen `#include`-Datei, ändert dies die Definition des Makros in allen Programmdateien, die diesen Header einbinden. Der zweite Vorteil ist, dass die herunterzuladenden HTML-Dateien klein bleiben. Der Browser muss ja für alle Seiten nur eine Kopie der CSS-Datei herunterladen (anstatt für alle Seiten, die auf die CSS-Datei verweisen, eine eigene Kopie zu besorgen).

7.3.3 Dynamische Webseiten und Webanwendungen

Das statische Seitenmodell, auf das wir uns bisher konzentriert haben, betrachtet Seiten als multimediale Dokumente, die auf bequeme Weise miteinander verknüpft sind. Für die Frühzeit des Webs, als gigantische Mengen von Informationen ins Netz gestellt wurden, war es ein äußerst passendes Modell. Doch die Zeiten haben sich gewandelt, heute sind mit die heißesten Themen die Nutzung des Webs für Anwendungen und Webservices – wie z.B. der Verkauf von Produkten über E-Commerce-Sites, das Recherchieren in Bibliothekskatalogen, das Studieren von Karten, das Lesen und Senden von E-Mails oder die Zusammenarbeit an einem Dokument.

Diese neuen Einsatzmöglichkeiten gleichen traditioneller Anwendungssoftware (z.B. E-Mail-Programmen oder Textverarbeitungsprogrammen), nur dass diese Anwendungen innerhalb des Browsers ausgeführt werden, mit Daten, die auf Servern in Internetdatenzentren gespeichert sind. Sie nutzen Webprotokolle, um über das Internet auf Informationen zugreifen zu können, und den Browser, um die Benutzeroberfläche anzuzeigen. Der Vorteil dieses Ansatzes ist, dass die Benutzer keine separaten Anwendungsprogramme installieren müssen und dass die Benutzerdaten von verschiedenen Rechnern aus zugänglich sind und vom Service-Operator mittels Backups gesichert werden können. Das Modell ist so erfolgreich, dass es für die herkömmliche Anwendungssoftware mittlerweile zu einer ernsten Konkurrenz geworden ist. Und der Umstand, dass diese neue Generation von Anwendungen von vielen Anbietern kostenlos zur Verfügung gestellt wird, hilft natürlich auch ein wenig. Dieses Modell ist die vorherrschende Form des **Cloud Computing**, bei dem die Rechenarbeit weg von den individuellen Desktop-Rechnern hin zu gemeinsam genutzten Server-Cluster im Internet verschoben wird.

Damit Webseiten als Anwendungen fungieren können, dürfen sie nicht länger statisch bleiben. Dynamische Inhalte sind nötig. Beispielsweise sollte die Seite eines Bibliothekskatalogs anzeigen, welche Bücher aktuell verfügbar sind und welche Bücher ausgeliehen wurden. Entsprechend würde eine gute Seite für Aktienkurse dem Benutzer interaktive Elemente anbieten, damit er sich die Entwicklung der Aktienkurse über verschiedene Zeiträume ansehen oder Profite und Verluste ausrechnen lassen kann. Wie sich an diesen Beispielen bereits erkennen lässt, können dynamische Inhalte sowohl von Programmen erzeugt werden, die auf dem Server ausgeführt werden, als auch von Programmen, die im Browser laufen (oder sie werden auf dem Server und im Browser generiert).

Beide Fälle werden wir im Folgenden näher untersuchen. Das allgemeine Modell sehen Sie in ► Abbildung 7.29. Stellen Sie sich z.B. einen Online-Atlas vor, der so konzipiert ist, dass die Benutzer eine Stadt und eine Straße eingeben und als Ergebnis

eine Karte der näheren Umgebung angezeigt bekommen. Empfängt der Webserver eine Anfrage zu einem bestimmten Ort, dann muss er ein Programm aufrufen, dass basierend auf den Daten aus einer Datenbank mit Straßennamen und anderen geografischen Informationen die Seite mit der gewünschten Karte erzeugt (siehe Schritte 1 bis 3 in der Abbildung 7.29). Die Anfrage (Schritt 1) startet ein Programm auf dem Server. Das Programm konsultiert eine Datenbank, mit deren Hilfe es die Seite erzeugt (Schritt 2). Letztere schickt es dann zurück an den Browser (Schritt 3).

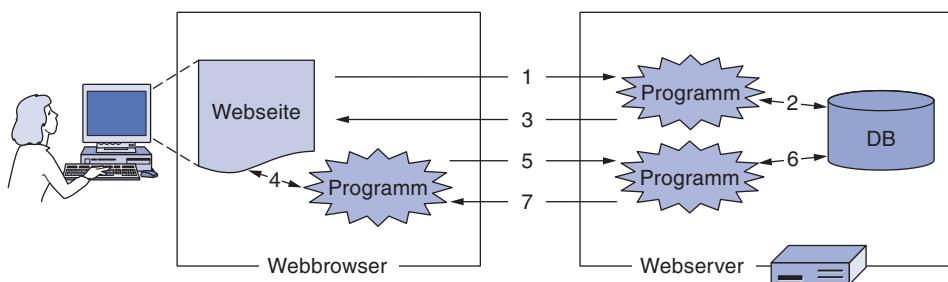


Abbildung 7.29: Dynamische Seiten.

Das ist aber noch nicht alles. Die Seite, die zurückgeliefert wird, kann selbst auch wieder Programme enthalten, die dann im Browser ausgeführt werden. So könnte die Seite unseres Online-Atlases z.B. ein Programm verwenden, dass es dem Benutzer erlaubt, Fahrt Routen zusammenzustellen oder die Umgebung zu erforschen (in unterschiedlich detaillierten Ansichten). Das Programm würde die Seite aktualisieren und dabei die Anzeige nach Anweisung des Benutzers vergrößern oder verkleinern (Schritt 4). Für einige Interaktionen benötigt das Programm womöglich weitere Daten vom Server. In diesem Fall sendet das Programm eine Anfrage an den Server (Schritt 5), die weitere Informationen aus der Datenbank entnimmt (Schritt 6) und als Antwort zurückschickt (Schritt 7). Anschließend fährt das Programm mit der Aktualisierung der Seite fort. Anfrage und Antwort laufen gänzlich im Hintergrund ab. Der Benutzer merkt davon in der Regel nichts, da die Seiten-URL und der Titel typischerweise nicht geändert werden. Durch die Einbettung solcher clientseitiger Programme können Seiten ihren Besuchern noch wesentlich interaktiver Oberflächen präsentieren als dies allein mit serverseitigen Programmen möglich wäre.

Serverseitige dynamische Erzeugung von Webseiten

Lassen Sie uns die serverseitige Erzeugung von Inhalten noch etwas genauer betrachten. Ein einfaches Szenario, in dem eine serverseitige Verarbeitung notwendig wird, ist der Einsatz von Formularen. Stellen Sie sich vor, ein Besucher füllt das AWI-Bestellformular aus Abbildung 7.25 aus und klickt auf die *Submit order*-Schaltfläche. Der Klick führt dazu, dass eine Anfrage an den Server gesendet wird, an die URL, die in dem HTML-Code des Formulars angegeben ist (in diesem Fall ein *POST* zu <http://widget.com/cgi-bin/order.cgi>), und zusammen mit den Inhalten des vom Benutzer ausgefüllten Formulars. Diese Daten müssen einem Programm, einem Skript oder einem Prozess übergeben werden. Das heißt, die URL identifiziert das auszuführende Programm; die Daten werden dem Programm als Eingabe übergeben. In diesem Fall gehört zur Verarbeitung, dass

die Bestellung in das interne System von AWI eingefüttert, die Kundendatensätze aktualisiert und die Kreditkarte belastet werden. Danach wird eine Antwortseite erstellt, die allerdings keine statische Seite ist, sondern davon abhängt, was sich während der Verarbeitung ergibt. Konnte die Bestellung erfolgreich bearbeitet werden, enthält die Antwortseite vielleicht das voraussichtliche Zustelldatum. Andernfalls könnte die Antwortseite dem Kunden mitteilen, dass eines der gewünschten Produkte nicht vorrätig ist oder dass die Kreditkarte aus irgendeinem Grund nicht gültig ist.

Wie der Server im Detail vorgeht, um ein Programm auszuführen statt eine einfache Datei zu beschaffen, hängt von der Konzeption des Webservers ab. Die Webprotokolle schweigen sich zu diesem Thema aus, da die betreffende Schnittstelle proprietär sein kann und der Browser die Details ja nicht kennen muss. Der Browser schickt einfach eine Anfrage und erhält eine Datei zurück.

Dessen ungeachtet wurden im Laufe der Zeit verschiedene Standard-APIs für den Aufruf von Programmen durch den Webserver entwickelt. Für Entwickler ist das Vorhandensein dieser APIs ein Segen, denn es erleichtert die Erweiterung der Server um Webanwendungen. Um Ihnen einen Eindruck davon zu geben, wie die Arbeit mit diesen APIs aussieht, werden wir uns zwei dieser APIs im Folgenden etwas näher ansehen.

Die erste API ist ein Verfahren zur Verarbeitung von Anfragen nach dynamischen Seiten, das bereits seit der Geburt des Webs besteht. Es nennt sich **CGI** (*Common Gateway Interface*) und ist in RFC 3875 definiert. CGI stellt eine standardisierte Schnittstelle zur Verfügung, mit der Webserver mit Back-End-Programmen und -Skripts kommunizieren können, die Eingaben (z.B. aus Formularen) annehmen und HTML-Seiten als Antwort erzeugen können. Diese Programme können in jeder beliebigen Sprache geschrieben werden, die dem Entwickler geeignet scheint. Meist wird dies eine Skriptsprache sein, da sich mit diesen Standardaufgaben leichter erledigen lassen. Wählen Sie z.B. Python, Ruby, Perl oder einfach Ihre Lieblingssprache.

Standardmäßig befinden sich Programme, die via CGI aufgerufen werden, in einem Verzeichnis namens *cgi-bin*, wie auch an der URL abzulesen ist. Der Server bildet an dieses Verzeichnis gerichtete Anfragen auf einen Programmnamen ab und führt das betreffende Programm als eigenständigen Prozess aus. Wurden zusammen mit der Anfrage auch Daten geschickt, übergibt der Server diese als Eingaben an das Programm. Die Ausgabe des Programms ist eine Webseite, die dem Browser zurückgesendet wird.

In unserem Beispiel wird das Programm *order.cgi* aufgerufen, mit den codierten Eingaben aus dem Formular (siehe Abbildung 7.26). Das Programm parst die Parameter und verarbeitet die Bestellung. Eine recht sinnvolle Konvention ist, dass das Programm den HTML-Code des Bestellformulars zurückliefert, wenn es ohne Formulareingaben aufgerufen wird. Auf diese Weise ist sichergestellt, dass das Programm die Repräsentation des Formulars kennt.

Die zweite API, die wir uns ansehen wollen, könnte kaum unterschiedlicher sein. Sie basiert auf dem Ansatz, kleine Skripts in die HTML-Seiten einzubetten und diese vom Server zur Erzeugung der Seiten ausführen zu lassen. Eine gängige Sprache zum Erstellen dieser Skripts ist **PHP** (*PHP: Hypertext Preprocessor*). Um sie verwenden zu

können, muss der Server PHP verstehen (so wie ein Browser CSS verstehen muss, um Webseiten mit Stylesheets interpretieren zu können). In der Regel erkennen Server Webseiten, die PHP-Code enthalten, daran, dass deren Dateinamenserweiterung *php* – statt *html* oder *htm* – lautet.

PHP ist einfacher zu verwenden als CGI. ►Abbildung 7.30a demonstriert die Verwendung von PHP zur Erstellung von Formularen. Der obere Teil der Abbildung zeigt eine normale HTML-Seite mit einem einfachen Formular. Diesmal gibt das `<form>`-Tag an, dass *action.php* für die Verarbeitung der Parameter aufgerufen werden soll, wenn der Benutzer das Formular abschickt. Die Seite zeigt zwei Textfelder an: eines zur Abfrage eines Namens und eines zur Abfrage des Alters. Nachdem die beiden Felder ausgefüllt und das Formular abgesendet wurde, parst der Server die zurückgesendete Zeichenfolge (die vom gleichen Aufbau wie die Zeichenfolge aus Abbildung 7.26 ist) und speichert den Namen in der Variablen *name* und das Alter in der Variablen *age*. Dann beginnt er mit der Verarbeitung der Datei *action.php*, die in ►Abbildung 7.30b dargestellt ist, um die Antwort zu erzeugen. Im Zuge der Verarbeitung der Datei werden die eingebetteten PHP-Befehle ausgeführt. Hat der Benutzer z.B. „Barbara“ und „24“ in das Formular eingegeben, so wird die HTML-Datei aus ►Abbildung 7.30c zurückgesendet. Die Verarbeitung von Formularen mittels PHP ist also extrem einfach.

```
<html>
<body>
<form action="action.php" method="post">
<p> Please enter your name: <input type="text" name="name"> </p>
<p> Please enter your age: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

a

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

b

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```

c

Abbildung 7.30: (a) Eine Webseite mit einem Formular. (b) Ein PHP-Skript zur Verarbeitung der Ausgaben des Formulars. (c) Ausgabe des PHP-Skripts für die Eingaben „Barbara“ und „24“.

Obwohl PHP leicht einzusetzen ist, ist es doch eine leistungsstarke Programmiersprache, die eine Schnittstelle zwischen dem Web und der Server-Datenbank bildet. Sie verfügt über Variablen, Zeichenketten und Arrays sowie die meisten Kontrollstrukturen, die auch in C bekannt sind, besitzt aber eine weitaus leistungsstärkere E/A-Unterstützung als nur *printf*. PHP ist Open Source, kostenlos erhältlich und weitverbreitet. Es wurde speziell für den Einsatz mit dem Apache-Server entwickelt, einer weiteren Open-Source-Software und dem wohl am häufigsten eingesetzten Webserver weltweit. Für weiterführende Informationen über PHP siehe Valade (2009).

Wir haben nun zwei verschiedene Möglichkeiten gesehen, dynamische HTML-Seiten zu erzeugen: CGI-Skripts und eingebettetes PHP. Darüber hinaus gibt es noch zahlreiche weitere Optionen. **JSP** (*JavaServer Pages*) gleicht PHP. Der einzige Unterschied ist, dass der dynamische Teil in der Programmiersprache Java und nicht in PHP geschrieben wird. Seiten mit dieser Technik tragen die Dateinamenserweiterung *jsp*. **ASP.NET** (*Active Server Pages .NET*) ist Microsofts Alternative zu PHP und JavaServer Pages und arbeitet mit Programmen, die zur Erzeugung der dynamischen Inhalte mit Microsofts proprietärem Framework .NET für die Erzeugung geschrieben wurden. Seiten mit dieser Technik tragen die Dateinamenserweiterung *aspx*. Die Entscheidung zwischen PHP, JSP oder ASP.NET hat in der Regel mehr politische Gründe (Open Source versus Microsoft) als technologische, da die drei Sprachen im Grunde vergleichbar sind.

Clientseitige dynamische Erzeugung von Webseiten

PHP- und CGI-Skripts lösen das Problem der Verarbeitung von Formularen und Interaktionen mithilfe von Datenbanken auf dem Server. Sie können alle aus Formularen stammenden Informationen entgegennehmen, Informationen in einer oder mehreren Datenbanken nachschlagen und HTML-Seiten mit den Ergebnissen erstellen. Was sie nicht können, ist, auf Mausbewegungen zu reagieren oder direkt mit den Benutzern zu interagieren. Hierzu sind Skripts erforderlich, die in HTML-Seiten eingebettet und auf dem Client-Rechner statt auf dem Server-Rechner ausgeführt werden. Seit HTML 4.0 werden Skripts durch das Tag `<script>` unterstützt. Die Technologien, mit deren Hilfe interaktive Webseiten erzeugt werden, werden gemeinhin unter der Bezeichnung **dynamisches HTML** zusammengefasst.

Die beliebteste Skriptsprache für die Clientseite ist **JavaScript**, das wir uns nun etwas näher ansehen werden. Trotz des ähnlichen Namens, hat JavaScript so gut wie nichts mit der Programmiersprache Java zu tun. Wie andere Skriptsprachen ist es eine höhere Programmiersprache. So kann man beispielsweise in einer einzigen JavaScript-Zeile ein Dialogfenster aufrufen, auf Texteingaben warten und die eingegebene Zeichenfolge in einer Variablen speichern. Höhere Funktionen wie diese machen JavaScript zur idealen Sprache für die Entwicklung interaktiver Webseiten. Andererseits muss man sehen, dass JavaScript schneller mutiert als eine Fruchtfliege, die in einem Röntgengerät gefangen ist, was das Schreiben von JavaScript-Programmen, die auf allen Plattformen laufen, extrem schwierig macht. Aber vielleicht wird sich JavaScript ja eines Tages stabilisieren.

Als Beispiel für ein JavaScript-Programm betrachten wir ►Abbildung 7.31. Wie in Abbildung 7.30 wird ein Formular angezeigt, das den Namen und das Alter abfragt und vorhersagt, wie alt die Person nächstes Jahr sein wird. Der Body-Teil ist fast der gleiche wie in unserem PHP-Beispiel. Der wichtigste Unterschied ist die Deklaration der *submit*-Schaltfläche und der darin enthaltenen Zuweisungsanweisung. Die Zuweisungsanweisung teilt dem Browser mit, dass beim Anklicken der Schaltfläche das Skript *response* aufzurufen und das Formular als Parameter zu übergeben ist.

Ganz neu ist hier die Deklaration der JavaScript-Funktion *response* im Kopfteil der HTML-Datei, ein Bereich, der normalerweise für Überschriften, Hintergrundfarben etc. reserviert ist. Diese Funktion extrahiert den Wert des Felds *name* aus dem Formular und speichert ihn als Zeichenfolge in der Variablen *person*. Auch der Wert des Feldes *age* wird extrahiert, mit der Funktion *eval* in einen ganzzahligen Wert konvertiert, 1 hinzugefügt und das Ergebnis in *years* gespeichert. Dann öffnet die Funktion ein Dokument für die Ausgabe, schreibt viermal mithilfe der Methode *writeln* in das Dokument und schließt es danach. Das erzeugte Dokument ist, wie an den HTML-Tags in den *writeln*-Ausgaben ersichtlich ist, eine HTML-Datei, die der Browser anschließend auf dem Bildschirm anzeigt.

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test#form) {
    var person = test#form.name.value;
    var years = eval(test#form.age.value) + 1;
    document.writeln("<html> <body>");
    document.writeln("Hello " + person + ".<br>");
    document.writeln("Prediction: next year you will be " + years + ".");
    document.writeln("</body> </html>");
    document.close();
}
</script>
</head>

<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Abbildung 7.31: Verarbeitung eines Formulars mit JavaScript.

Es ist wichtig zu verstehen, dass PHP und JavaScript trotz der offensichtlichen Ähnlichkeit, dass beide Code in HTML-Dateien einbetten, auf gänzlich unterschiedliche Weise verarbeitet werden. Nachdem der Benutzer in dem PHP-Beispiel aus Abbildung 7.30 die *submit*-Schaltfläche angeklickt hat, stellt der Browser eine lange Zeichenfolge

mit den Daten aus dem Formular zusammen und sendet diese an den Server, als Anfrage einer PHP-Seite. Der Server lädt die PHP-Datei und führt das darin eingebettete PHP-Skript aus. Das PHP-Skript erzeugt eine neue HTML-Seite, die an den Browser zur Anzeige zurückgesendet wird. Der Browser kann nicht einmal sicher erkennen, ob die Seite von einem Programm erzeugt wurde. Der ganze Vorgang ist noch einmal in ►Abbildung 7.32a dargestellt.

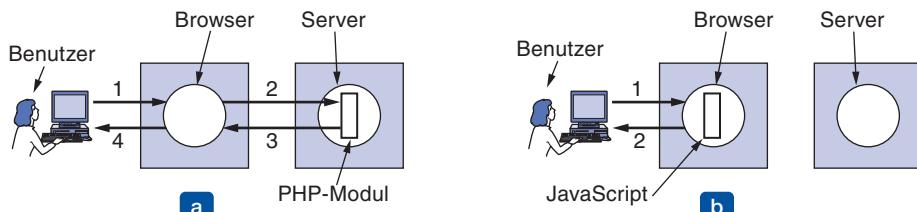


Abbildung 7.32: (a) Serverseitige Skriptausführung mit PHP. (b) Clientseitige Skriptausführung mit JavaScript.

Wenn der Benutzer in dem JavaScript-Beispiel aus Abbildung 7.31 die *submit*-Schaltfläche anklickt, führt der Interpreter des Browsers eine JavaScript-Funktion aus, die in der Seite enthalten ist. Die ganze Arbeit wird als lokal im Browser erledigt, es gibt keinen Kontakt zum Server (Schritte 1 und 2 in ►Abbildung 7.32b). Folglich werden die Ergebnisse praktisch sofort angezeigt, wohingegen bei PHP eine Verzögerung von ein paar Sekunden auftreten kann, bevor die resultierende HTML-Seite auf dem Client angezeigt wird.

Dieser Unterschied bedeutet nicht, dass JavaScript besser ist als PHP. Die Einsatzbereiche sind völlig verschieden. PHP (und hierin sind auch JSP und ASP impliziert) wird verwendet, wenn die Interaktion mit einer Datenbank auf dem Server erforderlich ist. JavaScript (und andere Sprachen für die Clientseite wie z.B. VBScript) wird verwendet, wenn eine Interaktion mit dem Benutzer auf dem Client-Rechner erforderlich ist. Kombinationen beider Techniken sind natürlich ebenfalls möglich, wie wir in Kürze sehen werden.

JavaScript ist nicht die einzige Möglichkeit, die Interaktivität einer Webseite zu verbessern. Auf der Windows-Plattform gibt es als Alternative **VBScript**, das auf Visual Basic basiert. Eine weitere beliebte Methode, die plattformübergreifend verwendet werden kann, sind **Applets**. Dabei handelt es sich um kleine Java-Programme, die in Anweisungen für einen virtuellen Rechner namens **JVM** (*Java Virtual Machine*) kompiliert wurden. Applets können in HTML-Seiten eingebettet (zwischen den Tags `<applet>` und `</applet>`) und mit JVM-fähigen Browsern interpretiert werden. Da Java-Applets interpretiert anstatt direkt ausgeführt werden, kann der Java-Interpreter sie davon abhalten, Schaden anzurichten. Soweit zumindest die Theorie. Tatsächlich haben Applet-Entwickler in den Java-E/A-Bibliotheken aber bereits eine nahezu endlose Reihe von Bugs gefunden, die als Schlupflöcher benutzt werden können.

Microsoft antwortete auf die Java-Applets von Sun, indem es Webseiten die Einbettung von **ActiveX-Steuerelementen** ermöglichte. Dies sind Programme, die in die x86er-Maschinensprache kompiliert und direkt von der Hardware ausgeführt werden. Das macht sie erheblich schneller und flexibler als interpretierte Java-Applets, da sie alles

tun können, was auch ein normales Programm tun kann. Erkennt der Internet Explorer ein ActiveX-Steuerelement in einer Webseite, lädt er es herunter, verifiziert seine Identität und führt es dann aus. Das Herunterladen und Ausführen fremder Programme birgt aber ein großes Sicherheitsrisiko und wirft Fragen auf, die wir in *Kapitel 8* näher betrachten werden.

Da nahezu alle Browser sowohl Java-Programme wie auch JavaScripts interpretieren können, stehen Entwicklern, die möglichst interaktive Webseite erstellen möchten, mindestens zwei Techniken zur Verfügung. Und wenn die Portierung auf andere Plattformen keine Rolle spielt, kommt als dritte Option noch ActiveX hinzu. Als Entscheidungshilfe kann man grundsätzlich feststellen, dass JavaScript-Programme einfacher zu erstellen sind, Java-Applets schneller ausgeführt werden und ActiveX-Steuerelemente am schnellsten laufen. Da alle Browser dieselbe JVM implementieren, aber keine zwei Browser die gleiche Version von JavaScript implementieren, sind Java-Applets besser portierbar als JavaScript-Programme. Weitere Informationen zu JavaScript finden Sie in unzähligen Büchern, viele mit mehr als 1 000 Seiten, siehe z.B. Flanagan (2010).

AJAX – Asynchronous JavaScript and XML

Attraktive Webseiten verlangen schnell reagierende Benutzeroberflächen und verzögerungsfreien Zugriff auf Daten, die auf entfernten Webservern liegen. Der Einsatz von Skripts auf dem Client (beispielsweise mit JavaScript) und auf dem Server (beispielsweise mit PHP) sind Basistechnologien, die bei der Erfüllung dieser Anfrage helfen können. In der Praxis werden sie häufig mit diversen anderen Schlüsseltechnologien zu einem Technologiemix kombiniert, der als **AJAX** (*Asynchronous JAvascript and Xml*) bezeichnet wird. Viele vollwertige Webanwendungen wie Google Mail, Google Maps and Google Text & Tabellen wurden mit AJAX implementiert.

AJAX ist – und dies ist das Verwirrende daran – keine Sprache, sondern ein Mix von Technologien, die zusammen dafür sorgen, dass Webanwendungen ebenso schnell reagieren und ebenso leistungsfähig sind wie konventionelle Desktop-Anwendungen. Zu diesem Technologiemix gehören:

- HTML und CSS für die Präsentation der Informationen auf den Seiten
- DOM (Document Object Model), um Teile der Seiten ad hoc (also während der Besucher sie betrachtet) verändern zu können
- XML (eXtensible Markup Language), damit Programme Anwendungsdaten mit dem Server austauschen können
- Ein asynchroner Kanal, über den Programme XML-Daten senden und empfangen können
- JavaScript als eine Sprache, die all diese Funktionalität miteinander verbindet

Eine ganz nette Sammlung also, die wir im Folgenden einzeln durchgehen werden, um besser zu verstehen, worin die Beiträge der verschiedenen Technologien liegen. HTML und CSS wurden bereits vorgestellt. Es sind Standards, die Inhalte und deren Darstellung beschreiben. Programme, die HTML- und CSS-Ausgaben erzeugen, können zur Anzeige ihrer Ausgaben einen Webbrowswer benutzen.

DOM (*Document Object Model*) ist eine spezielle Repräsentation einer HTML-Seite, die für Programme zugänglich ist. Die Repräsentation besitzt eine Baumstruktur und spiegelt die Hierarchie der HTML-Elemente wider. Den DOM-Baum zu dem HTML-Code aus Abbildung 7.30a sehen Sie z.B. in ► Abbildung 7.33. Die Wurzel bildet ein *html*-Element, das für den gesamten HTML-Block steht. Dieses Element ist der Elternknoten des *body*-Elements, das selbst wieder der Elternknoten eines *form*-Elements ist. Das Formular besitzt zwei Attribute, die rechts daneben abgebildet sind: eines für die Übertragungsmethode des Formulars (*POST*) und eines für die Aktion (die anzufordernde URL). Das *form*-Element besitzt drei Kinder, welche die beiden Absatz-Tags und das Eingabe-Tag repräsentieren, die in dem Formular eingebettet sind. Den unteren Abschluss des Baums bilden die Blätter, die entweder Elemente oder Literale (wie z.B. Zeichenketten) enthalten.

Die besondere Bedeutung des DOM-Modells besteht darin, dass Programme mit seiner Hilfe die Teile der Seite auf direktem Weg verändern können – ohne dass es dazu notwendig wäre, die ganze Seite umzuschreiben, sondern einfach indem sie den Knoten des Elements austauschen, das geändert werden soll. Nach dem Austausch aktualisiert der Browser die Anzeige und die Änderungen werden sichtbar. Wird z.B. über das DOM ein Bild auf der Seite ausgewechselt, dann aktualisiert der Browser nur das Bild und belässt die anderen Teile der Seite so, wie sie sind. Wir konnten das DOM sogar schon in Aktion erleben: Das JavaScript-Beispiel aus Abbildung 7.31 benutzte das DOM, als es dem *document*-Element Zeilen hinzufügte, die dann als neue Textzeilen am unteren Rand des Browserfensters angezeigt wurden.

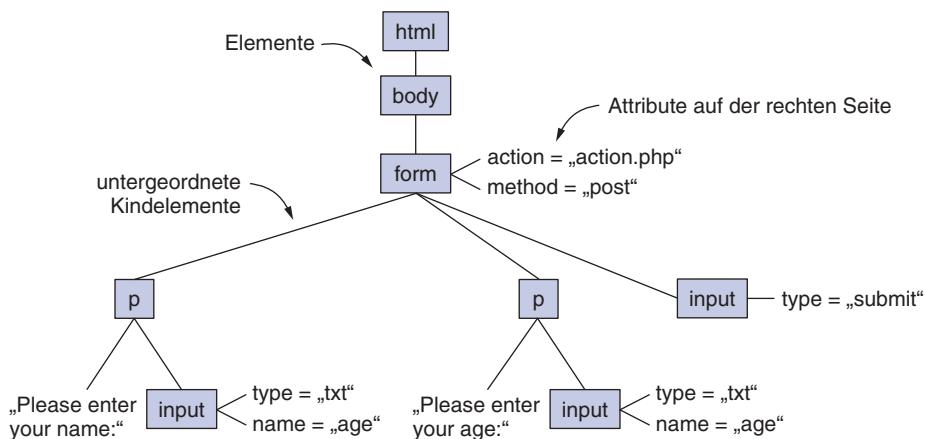


Abbildung 7.33: DOM-Hierarchie für den HTML-Code aus Abbildung 7.30a.

Die dritte Technologie, **XML** (*eXtensible Markup Language*), ist eine Sprache zur Beschreibung von strukturierten Inhalten. HTML vermischt Inhalt und Formatierung, da es für die Präsentation von Informationen konzipiert wurde. Durch die zunehmende Verbreitung von Webanwendungen gibt es aber immer mehr Bedarf an Techniken, wie man den Inhalt strukturieren und von der Formatierung trennen kann. Nehmen Sie z.B. ein Programm, das im Web den besten Preis eines Buches recherchieren soll. Um seine Aufgabe zu erfüllen, muss das Programm viele Webseiten nach Titel und Preis des gewünschten Objekts durchsuchen. Nur ... wurden die Webseiten in

HTML geschrieben, ist es für das Programm schwierig herauszufinden, wo auf der Seite der Titel und wo der Preis steht.

Aus diesem Grund hat das W3C den XML-Standard entwickelt (Bray et al., 2006), der festlegt, wie Webinhalte für die automatisierte Verarbeitung strukturiert werden können. Anders als in HTML gibt es in XML keine vordefinierten Tags. Jeder kann seine eigenen Tags definieren. Als ein einfaches Beispiel betrachten wir das XML-Dokument aus ►Abbildung 7.34. Es definiert eine Struktur namens book_list, also einer Liste von Büchern. Jedes Buch hat drei Felder: den Titel (title), den Autor (author) und das Erscheinungsjahr (year). XML-Strukturen sind extrem einfach organisiert. Strukturen dürfen sich wiederholende Felder besitzen (z.B. mehrere Autoren), optionale Felder (wie die URL des Hörbuchs) sowie alternative Felder (wie die URL eines Buchladens, wenn das Buch noch lieferbar ist, oder die URL einer Auktionswebsite, wenn es nicht mehr im Buchhandel erhältlich ist).

In unserem Beispiel ist jedes der drei Felder eine atomare Einheit. Es ist aber auch erlaubt, Felder weiter zu unterteilen. So könnte das Feld author beispielsweise wie folgt definiert sein, um noch mehr Details für Suchoperationen oder die Formatierung bereitzustellen:

```
<author>
    <first_name> George </first_name>
    <last_name> Zipf </last_name>
</author>
```

Jedes Feld kann in Unterfelder und Unterunterfelder aufgegliedert werden; die Anzahl der Unterebenen ist nicht begrenzt.

```
<?xml version="1.0" ?>

<book_list>

    <book>
        <title> Human Behavior and the Principle of Least Effort </title>
        <author> George Zipf </author>
        <year> 1949 </year>
    </book>

    <book>
        <title> The Mathematical Theory of Communication </title>
        <author> Claude E. Shannon </author>
        <author> Warren Weaver </author>
        <year> 1949 </year>
    </book>

    <book>
        <title> Nineteen Eighty-Four </title>
        <author> George Orwell </author>
        <year> 1949 </year>
    </book>

</book_list>
```

Abbildung 7.34: Ein einfaches XML-Dokument.

Alles, was die Datei aus Abbildung 7.34 tut, ist eine Bücherliste mit drei Büchern zu definieren. Dies ist ideal für die Übertragung von Informationen zwischen Programmen, die in Browsern und auf Servern ausgeführt werden, liefert aber keine Hinweise, wie die Webseite auf dem Bildschirm angezeigt werden soll. Ein Programm, dass die Informationen aus der XML-Datei verarbeitet, könnte entscheiden, dass das Jahr 1949 ein gutes Jahr für Bücher war, und HTML erzeugen, in dem die Titel durch Kursivschrift hervorgehoben sind. Alternativ kann mithilfe der Sprache **XSLT** (*eXtensible Stylesheet Language Transformations*) festgelegt werden, wie der XML-Code in HTML transformiert werden soll. XSLT gleicht CSS, ist aber wesentlich leistungsfähiger – und komplizierter; wir ersparen Ihnen daher die Details.

Daten in XML statt in HTML auszudrücken, hat noch einen weiteren Vorteil: Die Daten sind für Programme leichter zu analysieren. HTML-Code wurde ursprünglich (und wird immer noch) manuell eingetippt. Folglich gibt es eine Menge schlampig aufgesetzten HTML-Code. Manchmal fehlt das schließende Tag (wie z.B. `</p>`), andere Tags sind vielleicht falsch ineinander geschachtelt. Die Groß- und Kleinschreibung der Tag- und Attributnamen geht möglicherweise durcheinander und für manche Tags sieht der Standard nicht einmal schließende Tags vor (wie im Falle von `
`). Die meisten Browser geben ihr Bestes, um zu erraten, was der Autor der Webseite eigentlich mit seinem HTML-Code erreichen wollte. XML ist in diesem Punkt wesentlich strenger und klarer konzipiert. Die Namen von Tags und Attributen werden immer kleingeschrieben, Tags müssen immer in der umgekehrten Reihenfolge abgeschlossen werden, in der sie geöffnet wurden (oder klar anzeigen, wenn es sich um ein leerer Tag ohne abschließendes Tag handelt), und Attributwerte müssen in Anführungszeichen eingeschlossen werden. Dank dieser Strenge lässt sich XML-Code wesentlich leichter und ohne Zweideutigkeiten parsen.

Es gibt sogar eine XML-Definition für HTML: **XHTML** (*eXtended HyperText Markup Language*). Grundsätzlich ist XHTML eine besonders pingelige Variante von HTML. XHTML-Seiten müssen sich streng an die XML-Regeln halten, andernfalls werden sie von den Browsern nicht akzeptiert. Ziel ist es, ebenso wie bei XML, Seiten zu produzieren, die von Programmen (in diesem Fall Webanwendungen) besser verarbeitet werden können. Obwohl es XHTML bereits seit 1998 gibt, hat es bisher nur wenig Zuspruch gefunden. Autoren und Designer, die mit HTML arbeiten, sehen oft keinen Grund, warum sie auf XHTML umstellen sollten, zumal die Unterstützung seitens der Browser ziemlich auf sich warten ließ. Um den Übergang von HTML zu XHTML zu beschleunigen, wurde HTML 5.0 jetzt so definiert, dass Seiten sowohl in HTML als auch in XHTML repräsentiert werden können. Das endgültige Ziel ist es, HTML durch XHTML zu ersetzen; doch bis dahin dürfte es noch ein weiter Weg sein.

Auch als Sprache für die Kommunikation zwischen Programmen hat XML sehr an Popularität gewonnen. Läuft diese Kommunikation über das HTTP-Protokoll (siehe nächster Abschnitt), so spricht man von einem Webservice. Insbesondere **SOAP** (*Simple Object Access Protocol*) ist eine Möglichkeit, Webservices zu implementieren, die RPCs zwischen Programmen auf sprach- und systemunabhängige Weise durchführen. Der Client erstellt die Anfrage dabei einfach als XML-Nachricht und sendet sie

unter Verwendung des HTTP-Protokolls an den Server. Der Server sendet die Antwort als in XML formatierte Nachricht. Auf diese Weise können Anwendungen auf heterogenen Plattformen miteinander kommunizieren.

Zurück zu AJAX: Mit XML steht also ein geeignetes Format zum Austausch von Daten zwischen Programmen, die im Browser ausgeführt werden, und dem Server zur Verfügung. Um jedoch im Browser eine reaktionsfähige Benutzeroberfläche präsentieren zu können, während im Hintergrund Daten gesendet oder empfangen werden, muss es für die Skripts noch einen Weg geben, wie sie **asynchrone E/A-Operatoren** ausführen können – Operationen, die die Anzeige nicht gleich lahmlegen, während das Skript auf Antwort wartet. Denken Sie z.B. an eine Straßenkarte, die im Browser gescrollt werden kann. Wenn der im Browser sichtbare Ausschnitt der Karte sich bereits am Rand der heruntergeladenen Kartendaten befindet und das zuständige Skript auf der Kartenseite über eine weitere Scrolloperation informiert wird, muss es womöglich weitere Karten-daten vom Server anfordern. Während diese Daten heruntergeladen werden, soll die Benutzeroberfläche natürlich nicht einfrieren. Mit einer derart unfreundlichen Benutzeroberfläche würde man heute schließlich keinen Blumentopf mehr gewinnen. Statt dessen sollte das Scrollen möglichst kontinuierlich und ruckelfrei ablaufen. Wenn die Daten ankommen, wird das Skript benachrichtigt, sodass es die Daten verwenden kann. Läuft alles glatt, werden die Daten heruntergeladen, noch bevor sie benötigt werden. Moderne Browser unterstützen dieses Kommunikationsmodell.

Das letzte Puzzlestück ist die Skriptsprache, die alles zusammensetzt, indem sie den Zugriff auf die oben aufgeführten Technologien ermöglicht. Meist wird hierfür JavaScript benutzt, andere Sprachen wie z.B. VBScript sind aber ebenfalls möglich. Etwas weiter unten werden wir Ihnen noch ein einfaches JavaScript-Beispiel vorstellen. Lassen Sie sich von der Einfachheit des Codes aber nicht täuschen. JavaScript hat zwar seine Eigenarten, ist aber eine ausgereifte Programmiersprache, mit der gleichen Leistungsfähigkeit wie C oder Java. Es kennt Variablen, Zeichenfolgen, Arrays, Objekte, Funktionen sowie die üblichen Kontrollstrukturen und es besitzt Schnittstellen, die speziell für die Arbeit mit Browsern und Webseiten konzipiert wurden. JavaScript kann auch Mausbewegungen über Bildschirmobjekten verfolgen – Grundlagen vieler Techniken, die Webseiten lebendiger machen, wie z.B. das plötzliche Einblenden eines Menüs. Darüber hinaus kann JavaScript über das DOM auf einzelne Seitenelemente zugreifen, es kann HTML und XML manipulieren und es kann asynchrone HTTP-Kommunikationen durchführen.

Bevor wir dem Thema der dynamischen Webinhalte den Rücken kehren, möchten wir das Wichtigste noch einmal kurz zusammenfassen, indem wir die angesprochenen Technologien in einer einzelnen Abbildung gegenüber stellen. Vollständige Webseiten können bei Bedarf unter Verwendung verschiedener Skripts auf dem Server dynamisch erzeugt werden. Diese Skripts können, wie in ▶ Abbildung 7.35 dargestellt, in Server-Erweiterungssprachen wie PHP, JSP oder ASP.NET geschrieben werden oder als separate CGI-Prozesse, die dann in jeder beliebigen Sprache implementiert werden können.

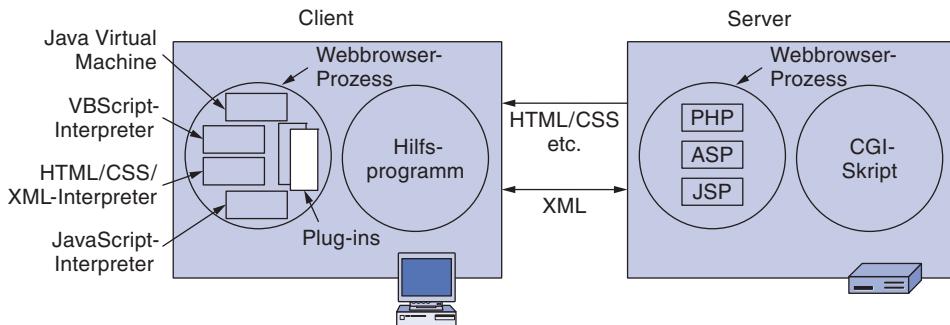


Abbildung 7.35: Verschiedene Möglichkeiten, dynamische Inhalte zu erzeugen.

Im Browser angekommen werden die Seiten dann als normale Seiten in HTML, CSS oder anderen MIME-Typen behandelt und angezeigt. Plug-ins, die im Browser laufen, sowie Hilfsprogramme, die außerhalb des Browsers ausgeführt werden, können installiert werden, um die Zahl der vom Browser unterstützten MIME-Typen zu erhöhen.

Dynamischer Inhalt kann auch auf dem Client erzeugt werden. Eingebettete Programme für Webseiten können in JavaScript, VBScript, Java und anderen Sprachen geschrieben werden. Diese Programme können beliebige Berechnungen ausführen und die Anzeige aktualisieren. Mithilfe von AJAX können Programme in Webseiten XML und andere Arten von Daten asynchron mit dem Server austauschen. Dieses Modell erlaubt die Erstellung von Webanwendungen der nächsten Generation, die wie herkömmliche Anwendersoftware aussehen, nur dass sie eben innerhalb eines Browsers ausgeführt werden und auf Daten zugreifen, die auf Servern im Internet liegen.

7.3.4 HTTP – HyperText Transfer Protocol

Nachdem wir nun gesehen haben, wie Webinhalte und Webanwendungen beschaffen sind, wird es Zeit, dass wir uns dem Protokoll zuwenden, mit dessen Hilfe all diese Daten zwischen Webservern und -clients übertragen werden. Das Protokoll heißt **HTTP (HyperText Transfer Protocol)** und ist in RFC 2616 definiert.

HTTP ist ein einfaches Anfrage-Antwort-Protokoll, das normalerweise auf TCP aufsetzt. HTTP gibt an, welche Nachrichten Clients an die Server senden dürfen und welche Antworten sie erhalten können. Die Anfrage- und Antwort-Header werden im ASCII-Format angegeben, ganz wie bei SMTP. Die Inhalte werden in einem MIME-ähnlichen Format angegeben, ebenfalls wie bei SMTP. Dieses einfache Modell war mitverantwortlich für den anfänglichen Erfolg des Webs, da es dessen Entwicklung und Ausbreitung erleichterte.

In diesem Abschnitt werden wir einen eingehenden Blick auf diejenigen Eigenschaften von HTTP werfen, die für die Art, wie es heute eingesetzt wird, von besonderer Bedeutung sind. Es sei aber angemerkt, dass die Rolle, die HTTP im Internet spielt, sich verändert. Grundsätzlich gehört HTTP zu den Protokollen der Anwendungsschicht, da es auf TCP aufsetzt und eng mit dem Web verbunden ist. Aus diesem Grund behandeln

wir es auch in diesem Kapitel. Auf der anderen Seite wird HTTP aber mehr und mehr wie ein Transportprotokoll verwendet, das es Prozessen ermöglicht, Inhalte über Netzwerkgrenzen hinaus auszutauschen. Dabei muss es sich bei diesen Prozessen keineswegs um einen Webbrower und einen Webserver handeln. Eine Software zum Abspielen von Videos und Songs könnte HTTP z.B. dazu nutzen, mit dem Server zu kommunizieren und Informationen über ein Album einzuholen. Antivirenprogramme können via HTTP die letzten Updates herunterladen. Entwickler können via HTTP Projektdateien übertragen. Geräte der Unterhaltungselektronik, wie z.B. digitale Fotorahmen, verfügen mittlerweile immer öfters über eingebettete HTTP-Server, die als Schnittstelle zur äußeren Welt dienen, und auch die Kommunikation von Rechner zu Rechner läuft immer häufiger über HTTP – beispielsweise könnte der Server einer Fluglinie, die komplette Urlaubspakete anbietet, mittels SOAP (eine auf HTTP aufbauende XML-RPC-Technologie) den Server eines Mietwagenverleihs kontaktieren, um für den Kunden ein Auto zu reservieren. Es steht zu erwarten, dass sich diese Trends fortsetzen werden, ebenso wie die Verbreitung von HTTP.

Verbindungen

Der übliche Weg, wie ein Browser einen Server kontaktiert, ist der Aufbau einer TCP-Verbindung zu Port 80 des Servers, obwohl dies formal nicht erforderlich ist. Der Einsatz von TCP hat den Vorzug, dass weder Browser noch Server sich um die Behandlung langer Nachrichten, um Fragen der Ausfallsicherheit oder der Netztüberlastung kümmern müssen. Dies wird alles von der TCP-Implementierung übernommen.

In den Anfangszeiten des Webs, also zu Zeiten von HTTP 1.0, wurde nach Errichtung der Verbindung eine einzige Anfrage überendet und eine einzige Antwort zurückgesendet. Dann wurde die TCP-Verbindung getrennt. In einer Welt, in der eine typische Webseite vollständig aus HTML-Text bestand, war diese Methode angemessen. Doch in nur wenigen Jahren nahm der Umfang der Webseiten so weit zu, dass eine durchschnittliche Seite eine Vielzahl von eingebetteten Links auf weitere Inhaltselemente wie Symbole, Bildern und anderen Attraktionen für das Auge enthielt. Die Errichtung einer separaten TCP-Verbindung zur Übertragung jedes einzelnen Elements wurde daher zu einem sehr teuren Verfahren.

Diese Entwicklung führte zu HTTP 1.1, das **dauerhafte Verbindungen** (*persistent connection*) unterstützt. Mit HTTP 1.1 ist es möglich, eine TCP-Verbindung einzurichten, eine Anfrage zu senden und eine Antwort zu erhalten und anschließend über die gleiche Verbindung noch weitere Anfragen zu senden und weitere Antworten zu erhalten. Man spricht in solchen Fällen auch von der **Wiederverwendung von Verbindungen** (*connection reuse*). Auf diese Weise entfallen die Kosten für Aufbau, Start und Freigabe weiterer TCP-Verbindungen und der relative, durch TCP verursachte Overhead pro Anfrage wird reduziert. Außerdem ist es möglich, Anfragen in Pipelines zu stellen, also die zweite Anfrage zu senden, noch bevor die Antwort auf die erste Anfrage zurückgekommen ist.

Die Leistungsunterschiede der drei angesprochenen Fälle sind in ▶ Abbildung 7.36 dargestellt. Teil a) zeigt drei Anfragen, die nacheinander abgearbeitet werden und jede eine eigene Verbindung aufbauen. Es könnte sich hier z.B. um eine Webseite handeln, in die zwei Bilder eingebettet sind, die auf demselben Server liegen. Da die URLs dieser Bilder erst beim Parsen der Hauptseite ermittelt werden, werden die Bilder auch erst nach der Hauptseite angefordert. Typische Seiten besitzen heutzutage um die 40 eingebettete Objekte, die für die Präsentation der Seite extra angefordert werden müssen. Da die Berücksichtigung von 40 Objekten die Abbildung allerdings ziemlich unübersichtlich gemacht hätte, begnügen wir uns hier mit nur zwei eingebetteten Objekten.

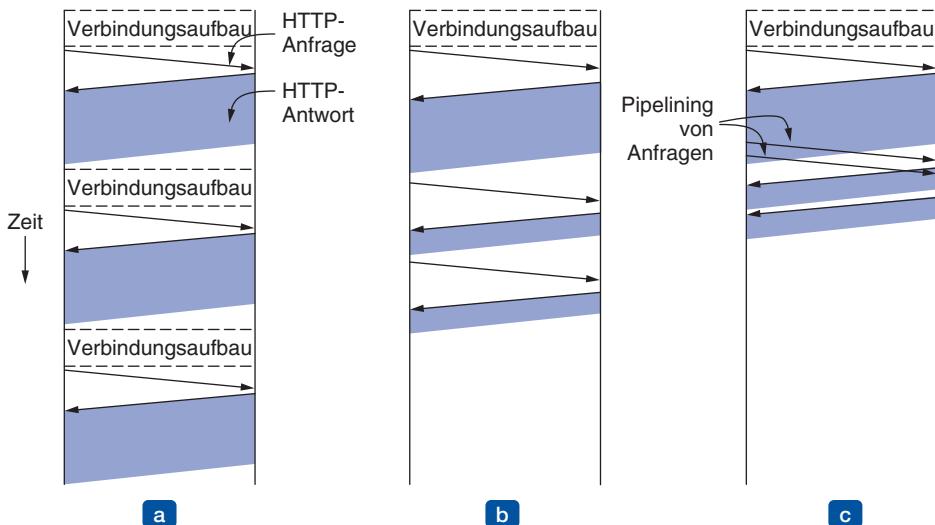


Abbildung 7.36: HTTP mit a) mehreren Verbindungen und sequenziellen Anfragen, b) mit einer dauerhaften Verbindung und sequenziellen Anfragen und c) mit einer dauerhaften Verbindung und Anfragen in Pipelines.

In ▶ Abbildung 7.36b wird die Seite über eine dauerhafte Verbindung angefordert. In diesem Fall wird zunächst die TCP-Verbindung geöffnet, dann werden nacheinander die bekannten drei Anfragen gesendet und erst anschließend wird die Verbindung wieder geschlossen. Beachten Sie, dass die Beschaffung der gewünschten Objekte in diesem Fall schneller vorstatten geht, und zwar aus zwei Gründen. Zum einen wird keine Zeit für den Aufbau zusätzlicher Verbindungen verschwendet. (Jede TCP-Verbindung benötigt zum Aufbau mindestens so viel Zeit wie für einen kompletten Austausch.) Zum anderen werden die Bilder schneller übertragen. Wieso? Wegen der Art und Weise, wie TCP mit Netzüberlastungen umgeht. TCP verwendet die Slow-Start-Prozedur, um den Durchsatz langsam zu erhöhen, bis mehr darüber bekannt ist, wie sich der Pfad durch das Netzwerk verhält. Die Konsequenz aus dieser Aufwärmphase ist, dass mehrere kurze TCP-Verbindungen ungleich mehr Zeit für die Übertragung der Daten benötigen als eine anhaltende TCP-Verbindung.

In ▶ Abbildung 7.36c schließlich gibt es eine dauerhafte Verbindung und eine Anfragen-Pipeline. Insbesondere die zweite und dritte Anfrage werden in schneller Folge gesendet – eben sobald die Hauptseite so weit übertragen wurde, dass die anzufor-

dernden Bilder identifiziert werden können. Die Antworten auf diese Anfragen folgen später. Dieses Verfahren reduziert die Leerlaufzeit des Servers und verbessert so noch einmal die Gesamtleistung.

Doch auch dauerhafte Verbindungen haben ihren Preis. So stellt sich für dauerhafte Verbindungen z.B. die Frage, wann die Verbindung geschlossen werden kann. Auf jeden Fall sollte die Verbindung zu einem Server so lange geöffnet bleiben, wie die Seite geladen wird. Doch was dann? Es ist nicht unwahrscheinlich, dass der Benutzer auf einen Link klickt, der eine weitere Seite von dem Server anfordert. Wenn die Verbindung geöffnet bleibt, kann die Anfrage umgehend gesendet werden. Eine Garantie dafür, dass der Client auch wirklich in Kürze eine weitere Anfrage an den Server schickt, gibt es aber nicht. In der Praxis lösen Server und Clients dies meist dadurch, dass sie dauerhafte Verbindungen so lange offen halten, bis diese für einen bestimmten, nicht zu langen Zeitraum (z.B. 60 Sekunden), nicht genutzt wurden oder bis es zu viele offene Verbindungen gibt und daher einige geschlossen werden müssen.

Dem aufmerksamen Leser wird vielleicht aufgefallen sein, dass wir einen Fall unterschlagen haben. Denkbar wäre es nämlich auch, eine Anfrage pro TCP-Verbindung zu senden, aber mehrere TCP-Verbindungen parallel auszuführen. Diese Technik der **parallelen Verbindungen** war vor dem Aufkommen dauerhafter Verbindungen unter den Browsern weitverbreitet. Sie hat zwar denselben Makel wie die sequenziellen Verbindungen – zusätzlicher Overhead – bietet aber eine wesentlich bessere Leistung, da Warte- und Leerlaufzeit durch das parallele Aufbauen und Hochfahren der Verbindungen reduziert werden. Von der Unterhaltung mehrerer TCP-Verbindungen zum selben Server ist jedoch abzuraten. TCP würde die Anpassung an die Netzauslastung für jede Verbindung separat durchführen, d.h., die Verbindungen würden in Konkurrenz zueinander treten, es würden mehr Pakete verloren gehen und zusammengenommen würden die Verbindungen aggressivere Netzbewerber darstellen als die Einzelverbindungen. Dauerhafte Verbindungen vermeiden dagegen zusätzlichen Overhead und führen nicht zu Problemen mit der Netzauslastung. Sie sind daher den parallelen Verbindungen überlegen und werden auch bevorzugt eingesetzt.

Methoden

Obwohl HTTP für die Nutzung im Web entwickelt wurde, entschied man sich mit Hinblick auf künftige objektorientierte Anwendungen dazu, das Protokoll absichtlich allgemeiner auszulegen als eigentlich nötig. Aus diesem Grunde unterstützt HTTP neben der Anforderung von Webseiten auch noch andere Operationen, im HTTP-Jargon als **Methoden** bezeichnet. Diese weise Voraussicht ermöglichte z.B. das Entstehen von SOAP.

Jede Anfrage besteht aus einer oder mehreren Zeilen ASCII-Text, wobei das erste Wort in der ersten Zeile der Name der angeforderten Methode ist. Die integrierten Methoden sind in ► Abbildung 7.37 aufgeführt. Beachten Sie, dass bei den Namen zwischen Groß- und Kleinschreibung unterschieden wird, d.h., *GET* ist korrekt, *get* ist nicht korrekt.

| Methode | Beschreibung |
|---------|--|
| GET | Lesen einer Webseite |
| HEAD | Lesen des Headers einer Webseite |
| POST | Anhängen an eine Webseite |
| PUT | Speichern einer Webseite |
| DELETE | Entfernen einer Webseite |
| TRACE | Echo-Wiedergabe für die eingehende Anfrage |
| CONNECT | Verbindung über einen Proxy |
| OPTIONS | Abfrage bestimmter Optionen zu einer Seite |

Abbildung 7.37: In HTTP integrierte Anfragemethoden.

Die *GET*-Methode fordert den Server auf, die Seite zu senden. (Wenn wir hier „Seite“ sagen, meinem wir im Grunde den allgemeinen Fall eines „Objekts“; um die erläuterten Konzepte zu verstehen, genügt es aber, sich unter einer Seite einfach die Inhalte einer Datei vorzustellen.) Die Seite ist entsprechend in MIME codiert. Die überwiegende Mehrzahl der Webserver-Anfragen sind vom Typ *GET*. Die übliche Form von *GET* lautet:

```
GET Dateiname HTTP/1.1
```

wobei *Dateiname* die Datei bezeichnet, die abgerufen werden soll, und 1.1 ist die verwendete Protokollversion.

Die *HEAD*-Methode fragt nur nach dem Nachrichten-Header, ohne die eigentliche Seite. Mithilfe dieser Methode können Informationen zur Indizierung gesammelt oder einfach eine URL auf Gültigkeit getestet werden.

Die *POST*-Methode wird zum Abschicken von Formularen verwendet. Sowohl *POST* als auch *GET* werden zudem für SOAP-Webservices verwendet. Ebenso wie *GET* beinhaltet auch *POST* eine URL. Statt aber einfach eine Seite anzufordern, lädt *POST* Daten – sprich den Inhalt des Formulars oder die RPC-Parameter – auf den Server hoch. Wie der Server die Daten dann verarbeitet, hängt von der URL ab. Grundsätzlich geht es aber darum, die Daten in passender Weise an das Objekt „anzuhängen“. Im Endeffekt wird dann z.B. ein Produkt gekauft oder eine Prozedur aufgerufen. Zum Schluss liefert die Methode eine Seite zurück, die über das Ergebnis der Aktion informiert.

Die verbleibenden Methoden werden beim Bewegen im Web eher selten verwendet. Die *PUT*-Methode ist das Gegenstück zu *GET*: statt eine Seite zu lesen, schreibt sie eine. Diese Methode ermöglicht den Aufbau einer Sammlung von Webseiten auf einem entfernten Server. Die Anfrage enthält die Seite. Sie kann mit MIME codiert sein. In diesem Fall können die Zeilen nach *PUT* möglicherweise Authentifizierungs-

Header enthalten, um nachzuweisen, dass der Anfrager zur Durchführung der angeforderten Operation berechtigt ist.

Die *DELETE*-Methode tut das, was man erwartet: Sie löscht die betreffende Seite bzw. meldet zumindest, dass der Webserver eingewilligt hat, die Seite zu löschen. Wie bei *PUT* spielen Authentifizierung und Berechtigungen hier eine wichtige Rolle.

Die *TRACE*-Methode dient dem Debuggen. Sie weist den Server an, die Anfrage zurückzusenden. Diese Methode ist nützlich, wenn Anfragen nicht korrekt verarbeitet werden und der Client wissen möchte, welche Anfrage der Server tatsächlich erhalten hat.

Die *CONNECT*-Methode erlaubt dem Benutzer eine Verbindung zu einem Webserver über ein zwischengeschaltetes Gerät, wie z.B. einen Webcache, herzustellen.

Die *OPTIONS*-Methode erlaubt dem Client, Informationen über eine Datei und die Methoden und Header, die mit der Seite verwendet werden können, abzufragen.

Auf jede Anfrage folgt eine Antwort, die aus einer Statuszeile und möglicherweise weiteren Informationen besteht (z.B. eine komplette Webseite oder Teile einer Webseite). Die Statuszeile enthält einen aus drei Ziffern bestehenden Statuscode, der angibt, ob der Anfrage wunschgemäß nachgekommen wurde bzw. welche Gründe es für das Scheitern gab. Die erste Ziffer dient der Unterteilung der Antworten in fünf Hauptgruppen, siehe ►Abbildung 7.38. Die 1xx-Codes kommen in der Praxis selten zum Einsatz. Die 2xx-Codes bedeuten, dass eine Anfrage erfolgreich bearbeitet wurde und der Inhalt (falls vorhanden) zurückgeliefert wurde. Die 3xx-Codes weisen den Client an, an anderer Stelle zu suchen, entweder mit einer anderen URL oder im eigenen Cache (dies wird später behandelt). Die 4xx-Codes besagen, dass die Anfrage aufgrund eines Client-Fehlers wie z.B. einer ungültigen Anfrage oder einer nicht vorhandenen Seite gescheitert ist. Die 5xx-Fehler schließlich zeigen an, dass der Server selbst ein Problem hat, entweder aufgrund eines Fehlers in seinem Code oder einer temporären Überlastung.

| Code | Bedeutung | Beispiel |
|------|---------------|--|
| 1xx | Information | 100 = Server stimmt zu, die Anfrage des Clients zu bearbeiten |
| 2xx | Erfolg | 200 = Anfrage erfolgreich; 204 = kein Inhalt vorhanden |
| 3xx | Umleitung | 301 = Seite wurde verlagert; 304 = Seite im Cache noch gültig |
| 4xx | Client-Fehler | 403 = verbotene Seite; 404 = Seite nicht gefunden |
| 5xx | Server-Fehler | 500 = interner Server-Fehler; 503 = versuche es später noch einmal |

Abbildung 7.38: Die Antwortgruppen des Statuscodes.

Nachrichten-Header

Auf die Zeile mit der Anfrage (also die Zeile mit der *GET*-Methode) können weitere Zeilen mit mehr Informationen folgen. Diese werden als **Anfrage-Header** (*request header*) bezeichnet. Diese Information kann mit den Parametern eines Prozeduraufrufs verglichen werden. Antworten können demgegenüber **Antwort-Header** (*response header*)

der) haben. Einige Header können in beiden Richtungen verwendet werden. Eine Auswahl der wichtigsten Header sehen Sie in ►Abbildung 7.39. Die Liste ist recht umfangreich, woraus Sie vielleicht zu Recht schließen werden, dass man oftmals mehrere Header in einer einzelnen Anfrage oder Antwort findet.

Der Header *User-Agent* erlaubt dem Client, den Server über seine Browser-Implementierung (z.B. *Mozilla/5.0* oder *Chrome/5.0.375.125*) zu informieren. Diese Information gestattet den Servern, ihre Antworten an den verwendeten Browser anzupassen – was angesichts der unterschiedlichen Ausstattung und abweichenden Verhaltensweisen der verschiedenen Browser recht nützlich sein kann.

Die vier *Accept*-Header teilen dem Server an, was der Client gewillt ist zu akzeptieren – für den Fall, dass der Client nur ein begrenztes Repertoire von Antworten akzeptiert. Der erste Header gibt die unterstützten MIME-Typen an (beispielsweise *text/html*). Der zweite gibt den Zeichensatz an (wie z.B. *ISO-8859-5* oder *Unicode-1-1*). Der dritte enthält die Komprimierungsmethoden (wie z.B. *gzip*). Der vierte enthält die natürliche Sprache (z.B. *Spanish*). Wenn dem Server verschiedene Seiten zur Auswahl zur Verfügung stehen, kann er anhand dieser Informationen genau die Seite zurückliefern, die der Client sucht. Kann er die Anfrage nicht erfüllen, schickt er einen Fehlercode und die Anfrage ist gescheitert.

Die Header *If-Modified-Since* und *If-None-Match* werden zum Caching verwendet. Mit ihrer Hilfe kann der Client mitteilen, dass eine Seite nur dann zurückgesendet werden soll, wenn die zwischengespeicherte Kopie nicht länger gültig ist. Das Thema Caching werden wir in Kürze noch ausführlicher behandeln.

Der *Host*-Header benennt den Server. Er stammt aus der URL. Der Header ist zwingend erforderlich. Er wird verwendet, weil einige IP-Adressen verschiedene DNS-Namen unterstützen können und der Server eine Möglichkeit haben muss, wie er herausfinden kann, an welchen Host die Anfrage übergeben werden soll.

Der Header *Authorization* ist für geschützte Seiten erforderlich. In diesem Fall muss der Client nachweisen, dass er die angeforderte Seite auch wirklich anzeigen darf – dazu verwendet er diesen Header.

Der falsch buchstabierte Header *Referer* wird vom Client verwendet, um die URL anzugeben, die auf die URL der aktuellen Anfrage verwies. In der überwiegenden Zahl der Fälle ist dies die URL der zuvor angezeigten Seite. Dieser Header ist vor allem für das Nachverfolgen von Benutzerbewegungen nützlich, da er dem Server mitteilt, wie ein Client zu der Seite gelangt ist.

Obwohl Cookies in RFC 2109 und nicht in RFC 2616 behandelt werden, gibt es auch für sie zwei Header. Der Header *Set-Cookie* ist die Methode, wie Server die Cookies an die Clients senden. Der Client soll das Cookie speichern und es bei nachfolgenden Anfragen unter Verwendung des *Cookie*-Headers an den Server zurückschicken. (Beachten Sie, dass es für Cookies noch eine jüngere Spezifikation mit neueren Headern gibt: RFC 2965. Diese wurde von der Industrie aber weitgehend abgelehnt und es gibt nur wenige Implementierungen.)

| Header | Typ | Inhalt |
|-------------------|---------|--|
| User-Agent | Anfrage | Information über den Browser und dessen Plattform |
| Accept | Anfrage | Seitentypen, die der Client verarbeiten kann |
| Accept-Charset | Anfrage | Zeichensätze, die der Client unterstützt |
| Accept-Encoding | Anfrage | Seitencodierungen, die der Client verarbeiten kann |
| Accept-Language | Anfrage | Natürliche Sprachen, die der Client verarbeiten kann |
| If-Modified-Since | Anfrage | Datum und Uhrzeit zur Überprüfung der Aktualität |
| If-None-Match | Anfrage | Zuvor gesendete Identifizierungsmarke zur Überprüfung der Aktualität |
| Host | Anfrage | DNS-Name des Servers |
| Authorization | Anfrage | Eine Liste der Anmeldeinformationen des Clients |
| Referer | Anfrage | Die vorangehende URL, von der die Anfrage ausging |
| Cookie | Anfrage | Sendet ein zuvor gesendetes Cookie an den Server zurück |
| Set-Cookie | Antwort | Cookie, dass der Client speichern soll |
| Server | Antwort | Informationen über den Server |
| Content-Encoding | Antwort | Wie der Inhalt codiert ist (z.B. gzip) |
| Content-Language | Antwort | Natürliche Sprache, die in der Seite verwendet wird |
| Content-Length | Antwort | Seitenlänge in Byte |
| Content-Type | Antwort | MIME-Typ der Seite |
| Content-Range | Antwort | Identifiziert einen Teil des Seiteninhalts |
| Last-Modified | Antwort | Zeit und Datum, an dem die Seite zuletzt geändert wurde |
| Expires | Antwort | Zeit und Datum, wann die Seite ungültig wird |
| Location | Antwort | Teilt dem Client mit, wohin er seine Anfrage senden soll |
| Accept-Ranges | Antwort | Zeigt an, dass der Server Anfragen zum Bytebereich akzeptiert |
| Date | Beides | Datum und Uhrzeit, zu der die Nachricht gesendet wurde |
| Range | Beides | Identifiziert einen Teil einer Seite |
| Cache-Control | Beides | Richtlinien für die Cache-Behandlung |
| ETag | Beides | Identifizierungsmarke für den Inhalt der Seite |
| Upgrade | Beides | Das Protokoll, auf das der Sender wechseln möchte |

Abbildung 7.39: Ausgewählte HTTP-Nachrichten-Header.

In Antworten werden noch zahlreiche weitere Header verwendet. Der *Server-Header* ermöglicht dem Server, bei Bedarf den eigenen Software-Build anzugeben. Die nächsten fünf Header, die alle mit *Content-* beginnen, erlauben dem Server, nähere Angaben zu der gesendeten Datei mitzuliefern.

Der Header *Last-Modified* gibt an, wann die Seite zum letzten Mal geändert wurde. Der Header *Expires* teilt mit, wie lange die Seite gültig bleibt. Beide Header sind für das Caching von Seiten von großer Bedeutung.

Der Header *Location* dient dem Server dazu, den Client darüber zu informieren, dass er eine andere URL versuchen sollte. Dies kann verwendet werden, wenn die Seite verlagert wurde oder um mehreren URLs den Verweis auf die gleiche Seite zu ermöglichen (unter Umständen auf verschiedenen Servern). Er wird auch in Unternehmen verwendet, deren Webhauptseite in der *com*-Domäne liegt, um Clients zu einer nationalen oder regionalen Seite auf der Basis ihrer IP-Adresse oder bevorzugten Sprache umzuleiten.

Ist eine Seite sehr groß, kann sie ein kleiner Client eventuell nicht auf einmal verarbeiten. Einige Server unterstützen Anfragen nach Byte-Bereichen, sodass die Seite in mehreren kleinen Einheiten abgerufen werden kann. Der Header *Accept-Ranges* gibt die Bereitschaft des Servers an, diese Art der Anfrage eines Teils einer Seite zu verarbeiten.

Bleiben noch die Header, die in beide Richtungen verwendet werden können. Der Header *Date* kann in beiden Richtungen verwendet werden und enthält Zeitpunkt und Datum, an dem die Nachricht gesendet wurde, während der Header *Range* einen Bytebereich für die in der Antwort enthaltene Seite angibt.

Der Header *ETag* wird für das Caching verwendet und enthält eine kurze Identifizierungsmarke, die als Name für den Inhalt der Seite fungiert. Der Header *Cache-Control* liefert weitere explizite Regeln, ob und wie Seiten im Cache zu speichern (bzw. nicht zu speichern) sind.

Der Header *Upgrade* schließlich dient zum Wechsel auf ein neueres Kommunikationsprotokoll, also z.B. eine zukünftige Version des HTTP-Protokolls oder einen sicheren Transportmechanismus. Mithilfe dieses Headers kann der Client ankündigen, was er unterstützen kann, und der Server kann angeben, was er verwendet.

Caching

Leute, die im Web surfen, kehren häufig zu Webseiten zurück, die sie bereits zuvor besucht haben, und verwandte Webseiten verwenden oft dieselben eingebetteten Ressourcen (wie z.B. Bilder für die Navigationselemente, allgemeine Stylesheets oder Skripts). Es wäre äußerst ineffizient, würden diese Ressourcen jedes Mal, wenn eine der betreffenden Seiten angezeigt wird, neu angefordert. Schließlich verfügt der Browser ja bereits über eine Kopie.

Das Horten dieser Seiten für den Fall, dass sie später noch einmal benötigt werden, bezeichnet man als **Caching**. Caching hat den Vorteil, dass im Cache zwischengespeicherte Seiten wiederverwendet werden können, ohne dass der Übertragungsvorgang wiederholt werden muss. HTTP unterstützt das Caching, indem es Clients hilft zu

erkennen, wann eine Seite problemlos wiederverwendet werden kann. Netzverkehr und Leerlauf werden auf diese Weise reduziert und die Leistung verbessert. Die Leistungsverbesserung hat aber einen Preis: die Browser müssen jetzt Seiten speichern. Allerdings zahlt sich die Investition fast immer aus, denn lokaler Speicher ist billig. Die Seiten werden üblicherweise auf der Festplatte abgelegt, sodass sie zur Verfügung stehen, wenn der Browser das nächste Mal ausgeführt wird.

Die Schwierigkeit beim HTTP-Caching ist zu entscheiden, ob eine neu angeforderte Seite noch genauso aussehen würde wie eine zuvor zwischengespeicherte Kopie der Seite. Allein anhand der URL kann diese Entscheidung nicht getroffen werden. Die URL könnte z.B. zu einer Seite führen, die die neuesten Nachrichten anzeigt. Die Inhalte einer solchen Seite werden ständig aktualisiert, während die URL immer dieselbe bleibt. Auf der anderen Seite könnte der Inhalt der Seite aber auch aus einer Liste der Götter aus der griechischen und römischen Mythologie bestehen. In diesem Fall würde sich der Inhalt der Seite vermutlich nicht ganz so häufig ändern.

HTTP kennt zwei Strategien, wie dieses Problem zu lösen ist. Beide Ansätze sind in ▶Abbildung 7.40 als alternative Abläufe zwischen Schritt 1 (Anfrage) und Schritt 5 (Antwort) dargestellt. Die erste Strategie basiert auf der Validierung der Seite (Schritt 2). Der Cache wird konsultiert, und wenn sich dabei herausstellt, dass es im Cache zu der angeforderten URL eine Kopie der Seite gibt, die noch aktuell (d.h. noch gültig) ist, besteht keine Notwendigkeit, die Seite erneut vom Server anzufordern. Stattdessen wird sofort die zwischengespeicherte Seite zurückgeliefert. Für die Entscheidung, ob die Seite noch gültig ist, werden lediglich das aktuelle Datum (einschließlich Uhrzeit) und der *Expires*-Header benötigt, der zusammen mit der zwischengespeicherten Seite zurückgeliefert wurde, als diese ursprünglich angefordert wurde.

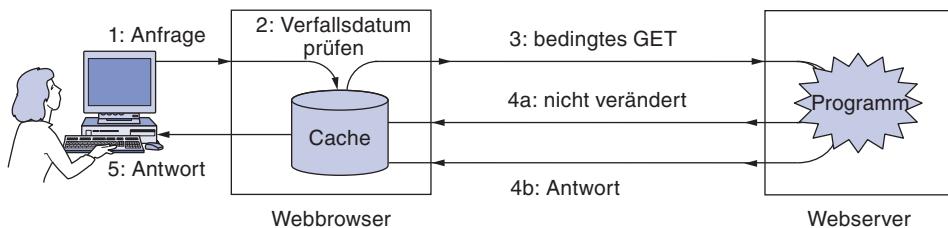


Abbildung 7.40: HTTP-Caching.

Allerdings sind nicht alle Seiten so zuvorkommend mit einem *Expires*-Header ausgestattet, der darüber informiert, wann die Seite wieder angefordert werden muss. Schließlich fallen Vorhersagen nicht immer leicht – vor allem nicht, wenn sie im Voraus getroffen werden müssen. In diesem Fall kann sich der Browser der Heuristik bedienen. Wurde die Seite z.B. im letzten Jahr überhaupt nicht geändert (laut *Last-Modified*-Header), kann mit ziemlicher Sicherheit davon ausgegangen werden, dass sie auch in der nächsten Stunde nicht verändert wird. Eine Garantie gibt es hierfür jedoch nicht, und manchmal erweist sich der heuristische Ansatz als trügerisch. Denken Sie z.B. an eine Seite mit Börsenkursen, die nach dem Schließen der Börse für mehrere Stunden unverändert bleibt, die aber in schneller Abfolge ständig aktualisiert

wird, sobald der Börsenhandel wieder aufgenommen wird. Wie gut sich eine Seite für die Zwischenspeicherung im Cache eignet, kann im Laufe der Zeit also stark variieren. Aus diesem Grund ist der heuristische Ansatz mit Vorsicht zu genießen, auch wenn er in der Praxis meist sehr gute Ergebnisse liefert.

Besonders lohnend ist das Caching, wenn der Browser im Cache Seiten findet, die noch nicht abgelaufen sind – bedeutet dies doch, dass der Server gar nicht kontaktiert werden muss. Leider tritt dieser Fall viel zu selten ein, denn die Server müssen beim Einsatz des *Expires*-Headers zurückhaltend sein, vor allem wenn Unsicherheit darüber besteht, wann eine Seite aktualisiert wird. So kann es leicht passieren, dass zwischengespeicherte Kopien immer noch aktuell sind, der Client aber keine Kenntnis darüber hat.

In solchen Fällen kommt die zweite Strategie zum Einsatz, die darin besteht, beim Server nachzufragen, ob die zwischengespeicherte Kopie noch gültig ist. Man bezeichnet diese Art von Anfrage als **bedingtes GET** (*conditional GET*), siehe Abbildung 7.40, Schritt 3. Kann der Server bestätigen, dass die zwischengespeicherte Kopie noch gültig ist, braucht er nur eine kurze, bejahende Antwort zu senden (Schritt 4a). Andernfalls sendet er die vollständige Antwort (Schritt 4b).

Für die Überprüfung der Gültigkeit zwischengespeicherter Seiten durch den Server stehen diverse Header-Felder zur Verfügung. Der Client kann an dem Header *Last-Modified* ablesen, wann die Seite das letzte Mal aktualisiert wurde. Er kann diese Zeitangabe über den Header *If-Modified-Since* zum Server schicken, um sich die Seite nur dann zurückliefern zu lassen, wenn sie in der Zwischenzeit geändert wurde.

Eine andere Möglichkeit ist, dass der Server zusammen mit der Seite einen *ETag*-Header schickt. Der *ETag*-Header enthält dann eine Marke, die als Kurzbezeichnung für den Inhalt der Seite fungiert, also quasi wie eine Prüfsumme, nur besser. (Eine solche Kurzbezeichnung kann z.B. ein kryptografischer Hash sein, siehe Kapitel 8.) Der Client kann dann zur Validierung der zwischengespeicherten Kopien einen *If-None-Match*-Header mit der Liste der zu den Kopien gehörenden Marken senden. Wenn irgendeine dieser Marken mit dem Inhalt übereinstimmt, mit dem der Server antworten würde, kann die entsprechende Kopie verwendet werden. Dieses Verfahren kann verwendet werden, wenn die Feststellung der Gültigkeit zu unbequem erscheint oder nicht zum gewünschten Ziel führt. Nehmen Sie z.B. den Fall, dass ein Server abhängig von der gewünschten Sprache und den bevorzugten MIME-Typen für dieselben URL unterschiedliche Inhalte sendet. In diesem Fall ist der Server nicht in der Lage, allein anhand des Datums der letzten Bearbeitung festzustellen, ob die zwischengespeicherte Seite noch aktuell ist.

Abschließend sei angemerkt, dass beide Caching-Strategien durch die Direktiven des Headers *Cache-Control* ausgehebelt werden können. Diese Direktiven dienen dazu, das Caching bei Bedarf einzuschränken (beispielsweise mit der Direktive *no-cache*). Typische Kandidaten sind z.B. dynamische Seiten, die jedes Mal, wenn sie angefordert werden, andere Inhalte präsentieren. Auch Seiten, die eine Autorisierung erfordern, werden üblicherweise nicht zwischengespeichert.

Es gäbe noch vieles, was man über das Caching erzählen könnte. Um das Thema aber nicht zu sehr ausufern zu lassen, beschränken wir uns auf zwei wichtige Punkte, die wir noch kurz ansprechen möchten. Erstens: Caching kann nicht nur im Browser, sondern auch noch an anderen Stellen stattfinden. Grundsätzlich können HTTP-Anfragen durch eine ganze Folge von Caches geführt werden. Kommen Caches außerhalb des Browsers zum Einsatz, spricht man von **Proxy-Caching**. Jede zusätzliche Cache-Stufe kann die Zahl der weiter oben in der Kette zu bearbeitenden Anfragen reduzieren. Viele ISPs und andere große Unternehmen betreiben Proxy-Caches, um die Vorteile des Seiten-Cachings für die Unterstützung der verschiedenen Benutzer zu nutzen. In Abschnitt 7.5 am Ende dieses Kapitels werden wir uns im Kontext des übergeordneten Themas der Inhaltsverteilung noch einmal auf das Proxy-Caching zurückkommen.

Zweitens: Caches können die Leistung erheblich verbessern, doch der Effekt ist nicht immer so groß, wie man es vielleicht erwarten würde. Dies liegt daran, dass es im Web neben den äußerst erfolgreichen und viel besuchten Dokumenten auch eine riesige Zahl eher unbekannter Dokumente gibt, die ebenfalls besucht werden und nicht selten recht umfangreich sind (z.B. Videos). Die lange Liste dieser unpopulären Dokumente belegt wertvollen Cache-Speicher und führt dazu, dass die Anzahl der Anfragen, die vom Cache bedient werden können, nur langsam mit der Größe des Caches steigt. Man geht davon aus, dass Web-Caches weniger als die Hälfte der Anfragen behandeln können (siehe Breslau et al., 1999).

Beispiel zur Verwendung von HTTP

Da HTTP ein ASCII-Protokoll ist, kann man von einem Terminal (nicht vom Browser) aus, recht einfach direkt mit einem Webserver kommunizieren. Man muss lediglich eine TCP-Verbindung zu Port 80 des Servers aufbauen. Am besten probieren Sie die folgende Befehlsfolge selbst einmal aus. Sie funktioniert in den meisten Unix-Shells wie auch der Eingabeaufforderung von Windows (sofern *Telnet* aktiviert ist).

```
telnet www.ietf.org 80
GET /rfc.html HTTP/1.1
Host: www.ietf.org
```

Diese Befehlsfolge baut eine Telnet-Verbindung (also eine TCP-Verbindung) zu Port 80 des Webservers der IETF, www.ietf.org, auf. Dann kommt der *GET*-Befehl, der den Pfad zu der URL und das Protokoll angibt. Probieren Sie ruhig, diverse Server und URLs Ihrer Wahl aus. Die nächste Zeile ist der Host-Header, der nicht fehlen darf, ebenso wenig wie die anschließende Leerzeile. Sie zeigt dem Server an, dass es keine weiteren Anfrage-Header gibt. Danach schickt der Server seine Antwort. Abhängig von dem angesprochenen Server und der verwendeten URL, werden Sie die verschiedensten Header und Seiten zu sehen bekommen.

7.3.5 Das mobile Web

Das Web wird von nahezu jeder Form von Rechner aus genutzt, Handys und Smartphones eingeschlossen. Über ein drahtloses Netzwerk im Web zu surfen, während man unterwegs ist, hat viele Vorteile, stellt aber auch eine große technische Herausforderung dar, da die meisten Webinhalte für knallige Präsentationen auf Desktop-Rechnern mit Breitbandverbindung gedacht sind. Wie sich der Zugang zum Web über mobile Endgeräte, das sogenannte **mobile Web**, im Laufe der Jahr entwickelt hat, ist Thema dieses Abschnitts.

Im Vergleich zu den Desktop-Rechnern auf der Arbeit oder zu Hause, wird das Surfen im Web über Handys durch verschiedene Aspekte erschwert:

- 1.** Relativ kleine Bildschirme schließen große Seiten und Bilder aus.
- 2.** Beschränkte Eingabefähigkeiten erschweren die Eingaben von URLs oder anderen umfangreicherer Eingaben.
- 3.** Die Netzwerkbandbreite über drahtlose Verbindungen ist begrenzt. Dies gilt besonders für Mobilfunknetze (z.B. 3G-Netze), die zudem meist recht teuer sind.
- 4.** Die Verbindungen können unter Aussetzern leiden.
- 5.** Die Rechenleistung wird durch verschiedene Faktoren wie Ladestatus der Akkus, Größe der Akkus, Wärmeentwicklung oder Kosten beschränkt.

Es ist daher keine gute Strategie, für das mobile Web einfach desktoporientierte Inhalte zu verwenden. Es sei denn, man nimmt in Kauf, dass man seinen Besucher ein eher frustrierendes Surferlebnis beschert.

Erste Ansätze für ein mobiles Web führten zur Entwicklung eines neuen Protokollstapels, der speziell auf die Bedürfnisse mobiler Endgeräte mit beschränkten Kapazitäten ausgelegt war. **WAP** (*Wireless Application Protocol*) ist wohl eines der bekanntesten Resultate dieser Strategie. Die WAP-Initiative wurde 1997 von einer Gruppe bedeutender Mobilfunkanbieter gestartet, darunter Nokia, Ericsson und Motorola. Dann aber geschah etwas Unerwartetes. Die nächsten zehn Jahre brachten enorme Verbesserungen bei der Netzwerkbreite und den technischen Fähigkeiten der mobilen Endgeräte, 3G-Datenservices kamen auf und auf dem Markt erschienen mobile Endgeräte mit größeren Farbdisplays, schnelleren Prozessoren und IEEE-802.11-Unterstützung (Wireless LAN). Plötzlich war es möglich, auf mobilen Endgeräten ganz gewöhnliche Webbrower auszuführen. Zwar reichte die Ausstattung mobiler Endgeräte damit noch immer nicht an die Fähigkeiten moderner Desktop-Rechner heran, und sie wird es wohl auch nie, doch viele der technischen Probleme, die zur Entwicklung eines eigenen Protokolls führten, hatten sich in Wohlgefallen aufgelöst.

Seitdem verfolgen immer mehr Anbieter einen anderen Ansatz: die Verwendung derselben Protokolle für Mobilgeräte und Desktop-Rechner und Websites, die entsprechend aufbereitete Inhalte liefern, wenn ein Besucher mit einem mobilen Gerät surft. Webserver können an den Anfrage-Headern ablesen, ob sie die Desktop- oder die Mobilversion einer Webseite zurücksenden sollen. Der Header *User-Agent*, der die

Browsersoftware identifiziert, ist in dieser Hinsicht besonders hilfreich. Ein Webserver, bei dem eine Anfrage eingeht, braucht also nur einen Blick in die Header zu werfen, um zu erkennen, ob er es mit einem iPhone-Nutzer zu tun hat, dem er eine Seite mit kleineren Bildern, weniger Text und einfacherer Navigation schickt, oder dem Benutzer eines Laptops, dem er die Vollversion der Seite zukommen lässt.

Das W3C unterstützt diesen Ansatz auf verschiedene Weise. Beispielsweise durch die Herausgabe offizieller Empfehlungen für mobile Webinhalte. Die erste Spezifikation enthält eine Liste von 60 solcher Empfehlungen (Rabin and McCathieNevile, 2008). Meist geht es bei diesen Empfehlungen um sinnvolle Schritte zur Reduzierung der Seitengröße, beispielsweise durch Komprimierung (da die Kosten für die Kommunikation höher sind als die Kosten für die Berechnung) oder optimale Nutzung des Caching. Site-Betreiber, vor allem wenn es sich um größere Sites handelt, werden ermutigt, spezielle Mobilversionen ihrer Inhalte zu erstellen, um mobile Webnutzer zu gewinnen. Zu deren Orientierung gibt es auch ein spezielles Logo, das Seiten kennzeichnet, die (gut) im mobilen Web betrachtet werden können.

Ein weiteres nützliches Hilfsmittel ist **XHTML Basic**, eine reduzierte HTML-Version. Die Sprache **XHTML Basic** ist eine Teilmenge von HTML, die speziell für Handys, Fernsehgeräte, PDAs, Getränkeautomaten oder Ähnliches, Mobilfunkempfänger, Autos, Spielautomaten und sogar Uhren konzipiert wurde. Aus diesem Grund gibt es keine Stylesheets, Skripts oder Frames, die meisten anderen Standard-Tags werden aber unterstützt. Die verfügbaren Tags werden in elf Module unterteilt. Einige sind obligatorisch, andere optional, alle sind in XML definiert. In ► Abbildung 7.41 sind die einzelnen Module und einige Beispiel-Tags aufgeführt.

| Modul | Erford. ? | Funktion | Tag-Beispiel |
|-------------------|-----------|-----------------------|---------------------------------------|
| Structure | Ja | Dokumentstruktur | body, head, html, title |
| Text | Ja | Information | br, code, dfn, em, hn, kbd, p, strong |
| Hypertext | Ja | Hyperlinks | a |
| List | Ja | Liste mit Elementen | dl, dt, dd, ol, ul, li |
| Forms | Nein | Ausfüllbare Formulare | form, input, label, option, textarea |
| Tables | Nein | Rechteckige Tabellen | caption, table, td, th, tr |
| Image | Nein | Bilder | img |
| Object | Nein | Applets, Maps etc. | object, param |
| Metainformationen | Nein | Zusatzinfos | meta |
| Link | Nein | Ähnlich wie <a> | link |
| Base | Nein | URL-Ausgangspunkt | base |

Abbildung 7.41: Die Module und Tags von XHTML Basic.

Allerdings sind nicht alle Seiten für das mobile Web geeignet. Ein alternativer Ansatz ist daher die **Inhaltstransformation** oder **Transcodierung**. Bei diesem Ansatz nimmt ein Computer, der zwischen den mobilen Endgeräten und dem Server sitzt, alle Anfragen der mobilen Geräte entgegen, fordert die entsprechenden Inhalte von den Servern an und transformiert diese dann in mobile Webinhalte. Eine einfache Transformation ist z.B. die Reduzierung der Größe von Bildern durch Konvertierung in eine niedrigere Auflösung. Daneben gibt es noch etliche weitere kleine, sinnvolle Transformationen. Die Transcodierung gibt es seit den ersten Tagen des mobilen Webs und sie wird seitdem mit einem Erfolg genutzt (siehe z.B. Fox et al., 1996). Wo allerdings beide Ansätze parallel verwendet werden, kommt es zu Konflikten zwischen den Entscheidungen, die Server und Transcodierer treffen. So kann es passieren, dass eine Website für das mobile Web eine spezielle Kombination von Bild und Text anbietet, der Transcodierer trotzdem aber noch einmal hingehnt und das Format des Bildes umwandelt.

Bisher haben wir uns ganz auf die Inhalte konzentriert und kaum etwas über die Protokolle gesagt. Dies ist insofern verständlich, als die Inhalte das Hauptproblem bei der Realisierung des mobilen Webs darstellen. Ein paar Worte sollten wir allerdings auch noch über die Protokolle verlieren. Die Protokolle HTTP, TCP und IP, die für das Web benutzt werden, können einen beachtlichen Teil der zur Verfügung stehenden Bandbreite allein für den von ihnen selbst verursachten Overhead, man denke z.B. an die Protokoll-Header, verbrauchen. Für WAP und andere Lösungsansätze wurden daher spezielle Protokolle definiert, um den Overhead zu reduzieren. Nun hat sich aber herausgestellt, dass dieser Aufwand weitgehend unnötig war. Technologien zur Komprimierung der Header, wie z.B. das in *Kapitel 6* beschriebene ROHC (*RObust Header Compression*), können den Overhead der Protokolle weitgehend eliminieren. Auf diese Weise ist es möglich, mit nur einem Satz von Protokollen (HTTP, TCP und IP) auszukommen und diesen sowohl für Links mit hoher als auch niedriger Bandbreite zu verwenden. Es muss lediglich für Links mit niedriger Bandbreite die Header-Komprimierung eingeschaltet werden.

7.3.6 Suchen im Web

Beschließen möchten wir das Thema Web mit einer Besprechung der Kategorie von Webanwendungen, die von allen wohl am erfolgreichsten ist: die Suche. 1998 gründeten Sergey Brin und Larry Page, damals noch Studenten in Stanford, ein Start-up-Unternehmen namens Google, mit dem Ziel eine bessere Suchmaschine anzubieten. Ihre Idee war damals ziemlich radikal: Anstatt die Trefferqualität einer Webseite daran zu messen, wie oft der gesuchte Begriff darin enthalten ist, entschieden sie, dass die Zahl der Links, die von anderen Seiten auf die betreffende Seite verweisen, ein viel besseres Maß sei. Nehmen Sie z.B. die Homepage von Cisco. Viele Seiten verweisen auf die Homepage von Cisco, weswegen diese Seite für jemanden, der nach „Cisco“ sucht, vermutlich interessanter ist als eine Seite außerhalb des Unternehmens, die das Wort „Cisco“ aus irgendeinem Grund häufiger enthält.

Wie sich herausstellte, lagen die beiden richtig. Es war möglich, eine bessere Suchmaschine zu schreiben, und die Benutzer strömten ihr nur so zu. Ausgestattet mit dem nötigen Risikokapital konnte Google schnell wachsen. Im Jahr 2004 wurde aus Google eine Aktiengesellschaft mit einer Marktkapitalisierung von 23 Milliarden US-Dollar. Im Jahr 2010 betrieb Google schätzungsweise mehr als eine Millionen Server in Datenzentren überall auf der Welt.

An sich handelt es sich bei der Websuche um eine ganz gewöhnliche Webanwendung, wenn auch eine besonders ausgereifte, an der seit den ersten Tagen des Webs geforscht wurde. Was die Websuche auszeichnet, ist, dass sie für die Benutzung des Webs mittlerweile so gut wie unverzichtbar ist. Es wird geschätzt, dass im Web jeden Tag über eine Milliarde Suchoperationen gestartet werden. Wonach auch immer die Menschen im Web suchen, fast immer benutzen sie als Ausgangspunkt die Seite einer Suchmaschine. Stellen Sie sich vor, Sie wollten herausfinden, wo Sie in Seattle Vegemite-Sandwiches kaufen können. Vermutlich wird Ihnen keine Website einfallen, auf der Sie die gewünschten Informationen finden oder von der aus Sie Ihre Suche beginnen könnten. Doch wenn Sie sich einer Suchmaschine bedienen, stehen die Chancen nicht schlecht, dass diese für Sie eine Seite findet und Sie schnell zu den gesuchten Informationen führt.

Um eine Websuche durchzuführen, steuert der Benutzer mit seinem Browser die URL einer Suchseite an. Zu den wichtigsten Suchseiten gehören Google, Yahoo! und Bing. Dann gibt der Benutzer die Suchbegriffe in ein Formular ein und schickt sie ab. Die Suchmaschine forscht daraufhin in ihrer internen Datenbank nach passenden Seiten, Bildern oder welche Art von Objekt auch immer gesucht wird, und liefert das Ergebnis in Form einer dynamisch erstellten Seite zurück. Der Benutzer braucht dann nur noch den Links zu den gefundenen Seiten zu folgen.

Die Websuche ist auch ein interessantes Untersuchungsobjekt, da sie Auswirkungen auf Entwurf und Nutzung der Netzwerke hat. Da wäre z.B. die Frage, wie Suchmaschinen Seiten finden. Dazu muss der Suchmaschine eine Seitendatenbank zur Verfügung stehen, in der sie recherchieren kann. Im Web kann jede HTML-Seite Links zu anderen Seiten enthalten und alles, was von Interesse ist (oder wonach zumindest gesucht werden könnte), ist irgendwie miteinander verknüpft. Das bedeutet, dass man theoretisch ausgehend von einer Handvoll von Seiten durch Verfolgen der enthaltenen Links zu allen anderen Seiten im Web gelangen kann. Diese Vorgehensweise wird als **Webcrawling** bezeichnet. Webcrawler werden von allen Suchmaschinen eingesetzt.

Ein interessanter Punkt ist die Frage, welche Art von Seiten durch Webcrawling überhaupt gefunden werden. Statische Dokumente anzufordern und Links nachzugehen ist einfach. Viele Webseiten enthalten aber Programme, die abhängig von den Interaktionen des Besuchers unterschiedliche Seiten bzw. Inhalte anzeigen. Die Onlinekataloge von Warenhäusern sind hierfür ein typisches Beispiel. Ein solcher Katalog kann dynamische Seiten enthalten, die aus einer Produktdatenbank und auf der Basis von Anfragen für verschiedene Produkte erstellt werden. Diese Art von Inhalten sind nicht so einfach zu durchwandern wie statische Seiten. Wie also kann ein Webcrawler diese dynamischen Seiten finden? Die Antwort ist: In den meisten Fällen findet er sie nicht. Man bezeich-

net diese Art von verborgenen Inhalten daher auch als das **Deep Web**. Wie dieses Deep Web durchsucht werden kann, ist derzeit Gegenstand der Forschung, siehe z.B. Madhavan et al. (2008). Daneben gibt es noch die Konvention, nach der Sites eine eigene Seite speziell für Webcrawler aufzusetzen (*robots.txt*), um diesen mitzuteilen, welche Teile der Site besucht werden sollten und welche nicht.

Interessant ist auch die Frage, wie die vom Webcrawler gefundenen Daten verarbeitet werden. Um indizierende Algorithmen auf der Masse der anfallenden Daten ausführen zu können, müssen die Seiten gespeichert werden. Genaue Schätzungen sind schwierig, aber man geht davon aus, dass die großen Suchmaschinen Indizes zu Dutzenden von Milliarden von Seiten unterhalten, alle aus dem sichtbaren Teil des Webs. Geht man weiter davon aus, dass die durchschnittliche Seitengröße schätzungsweise 320 KB beträgt, folgt daraus, dass zur Speicherung einer durch Webcrawling zusammengetragene Kopie des Webs um die 20 Petabyte oder 2×10^{16} Byte benötigt werden. So beeindruckend diese Zahl auch sein mag, eine solche Menge von Daten kann immer noch bequem in Internetdatenzentren gespeichert und verarbeitet werden (Chang et al., 2006). Rechnet man z.B., dass 1 TB Festplattenspeicher 20 Dollar kosten, dann kosten 2×10^4 TB gerade einmal 400 000 Dollar – für Unternehmen von der Größe von Google, Microsoft oder Yahoo! also leicht erschwinglich. Und während sich das Web immer weiter ausdehnt, fallen gleichzeitig die Kosten für Festplattenspeicher. Man kann also davon ausgehen, dass es für größere Unternehmen auch in Zukunft weiterhin möglich sein wird, das ganze Web zu speichern.

Das Speichern der Daten ist eine Sache, das Auswerten der Daten eine ganz andere. Sie können sich sicher vorstellen, wie sehr XML die programmgestützte Analyse der Datenstruktur erleichtert, während schnell zusammengestellte Formate meist viel Raum für Spekulationen lassen. Hinzukommen die Konvertierung zwischen verschiedenen Formaten und womöglich auch die Übersetzung in andere Sprachen. Und dabei ist die Entschlüsselung der Datenstruktur nur ein Teil des Problems. Der schwierigste Teil ist zu verstehen, was die Daten bedeuten. Doch gerade dadurch können entscheidende Informationen gewonnen und ein echter Mehrwert erzielt werden – beispielsweise in Form besserer Trefferseiten als Antwort auf Suchanfragen. Das ultimative Ziel aber ist, direkte Fragen beantworten zu können, wie z.B.: „Wo kann ich in meiner Stadt einen billigen, aber vernünftigen Minibackofen kaufen?“

Ein dritter Aspekt der Websuche ist, dass durch sie eine höhere, benutzerfreundliche Ebene der Benennung von Ressourcen etabliert wurde. Warum sich noch eine lange URL merken, wenn man die gewünschte Webseite genauso schnell und zuverlässig (vielleicht sogar noch zuverlässiger) mithilfe einer Websuche nach dem Namen einer Person findet (immer davon ausgehend, dass es den meisten Menschen leichter fällt, sich einen Namen als eine URL zu merken)? Diese Strategie wird immer erfolgreicher. Auf die gleiche Weise, wie die DNS-Namen die IP-Adressen der Rechner verdrängt haben, verdrängt nun die Websuche die URLs der Rechner. Und wenn sich in einem Suchbegriff Tipp- oder Rechtschreibfehler einschleichen, werden diese meist automatisch korrigiert, während sie für eine falsch eingetippte URL auch die falsche Seite erhalten.

Schließlich führt uns die Websuche eindringlich einen Aspekt vor Augen, der zwar weniger mit dem Entwurf von Netzwerken, dafür aber umso mehr mit dem Wachstum von Internetdiensten zu tun hat: in der Werbung steckt viel Geld. Werbung ist der ökonomische Motor, der zu dem spektakulären Wachstum der Suchmaschinenbranche geführt hat. Anders als bei der Werbung in Printmedien, können Suchmaschinen die Werbung darauf abstimmen, was der Benutzer sucht. Die Zielgerichtetetheit der Werbung wird also erhöht. Abgewandelte Auktionsverfahren werden eingesetzt, um die Suchanfragen mit den nützlichsten Werbeeinblendungen zu verbinden (Edelman et al., 2007). Leider hat auch dieses neue Modell seine Schattenseiten, etwa den **Klickbetrug** (*click fraud*) durch Programme, die Benutzer simulieren und auf Werbeeinblendungen klicken, um höhere Entgelte zu erzielen.

7.4 Streaming Audio und Video

Webanwendungen und das mobile Web sind aufregende, aber nicht die einzigen neuen Entwicklungen, an denen sich die Nutzer der Netze erfreuen können. Für viele Leute sind Audio und Video der heilige Gral der Netzwerknutzung. Allein bei der Erwähnung des Worts „Multimedia“ läuft sowohl den Technik-Freaks wie auch Geschäftsleuten das Wasser im Mund zusammen. Die erste Gruppe sieht in Multimedia riesige technische Herausforderungen in der Bereitstellung von Voice-over-IP und interaktivem Video-on-Demand für jeden Haushalt. Die zweite wittert riesige Gewinne.

Die Idee, Audio und Video über das Internet zu übertragen, gibt es seit den 1970er Jahren, wenn nicht sogar noch länger. Doch erst seit ungefähr 2000 ist eine ganz gewaltige Zunahme im **Echtzeit-Audio-** und **Echtzeit-Videoverkehr** (*real-time audio and video*) festzustellen. Echtzeitverkehr unterscheidet sich insofern vom normalen Webverkehr, als er in einer bestimmten vorgegebenen Geschwindigkeit abgespielt werden muss, um brauchbar zu sein. Denn wer möchte schon gern ein Video in Zeitlupe und mit vielen Aussetzern ansehen. Im Gegensatz dazu kann es im Webverkehr kurze Unterbrechungen geben und das Laden von Seiten kann – im Rahmen gewisser Grenzen – mehr oder weniger lang dauern, ohne dass dies ein größeres Problem darstellt.

Zwei Dinge haben diese Zunahme ermöglicht. Zum einen sind die Computer leistungsstärker geworden und mit Mikrofonen und Kameras ausgestattet, sodass die Eingabe, die Verarbeitung und die Ausgabe von Audio- und Videodaten erleichtert wird. Zum anderen steht inzwischen eine wahre Flut an Internetbandbreiten zur Verfügung. Langstreckenverbindungen im Kern des Internet laufen mit vielen GBit/s und Breitband und drahtloses IEEE-802.11-LAN erreichen die Benutzer in jedem Winkel des Internets. Diesen Entwicklungen ist es zu verdanken, dass Internetdienstanbieter (ISP) enorme Verkehrsströme über ihre Backbone-Netze schicken können und der normale Benutzer 100–1 000-mal schneller eine Verbindung zum Internet herstellen kann als mit einem Telefonmodem von 56 kbit/s.

Die Fülle an Bandbreite hatte eine Zunahme des Audio- und des Videoverkehrs zur Folge – allerdings aus unterschiedlichen Gründen. Telefonanrufe benötigen relativ wenig Bandbreite (an sich 64 kbit/s, im komprimierten Zustand jedoch weniger); den-

noch waren Telefondienste von jeher teuer. Unternehmen nutzten die Gelegenheit, die bestehende Bandbreite für eine Sprachübertragung über das Internet zu nutzen, um ihre Telefonrechnungen zu verringern. Start-up-Unternehmen wie Skype sahen eine Möglichkeit, Kunden über ihre Internetverbindungen kostenlose Telefongespräche anzubieten. Neu auf den Markt drängende Telefondienstanbieter sahen einen kostengünstigen Weg, traditionelle Sprachanrufe über die IP-Netztechnik zu leiten. Folge dieser Entwicklungen war eine explosionsartige Zunahme der Sprachdaten, die über Internetnetze übertragen wurden, was als **Voice-over-IP** oder **Internettelefonie** bezeichnet wird.

Anders als die Audioübertragung benötigt der Videoverkehr große Bandbreiten. Internetvideos hinreichender Qualität werden komprimiert mit einer Geschwindigkeit von ca. 1 Mbit/s codiert, und ein normaler DVD-Spielfilm hat eine Größe von 2 GB. In der Zeit, als es noch keinen Breitbandinternetzugang gab, war eine Übertragung von Spielfilmen über das Netz völlig ausgeschlossen. Diese Situation hat sich geändert. Mit dem Ausbau des Breitbandnetzes konnten die Benutzer zum ersten Mal zu Hause gestreamte Videos von einigermaßen guter Qualität anschauen. Und das lieben die Menschen. Es wird geschätzt, dass an jedem beliebigen Tag rund ein Viertel der Internetnutzer das öffentliche Internetvideoportal YouTube besuchen. Das Filmverleihgeschäft hat sich auf Online-Downloads verlagert. Und die schiere Größe der Videos hat den Internetverkehr allgemein verändert. Der größte Teil des Internetverkehrs entfällt bereits auf Videoübertragungen, die Schätzungen zufolge in wenigen Jahren 90 % des Internetverkehrs ausmachen (Cisco, 2010).

Angesichts der Tatsache, dass genügend Bandbreite zur Verfügung steht, um Audio- und Videodaten zu übertragen, liegt das Hauptproblem bei der Entwicklung von Streaming- und Konferenzanwendungen in der Netzverzögerung. Audio- und Videodaten sind auf Echtzeitdarstellung angewiesen, was bedeutet, dass sie mit einer vorgegebenen Geschwindigkeit abgespielt werden müssen, um nützlich zu sein. Große Verzögerungen bedeuten, dass Anrufe, die interaktiv sein sollten, dies nicht mehr sind. Dies Problem wird deutlich, wenn Sie jemals über ein Satellitentelefon angerufen haben, bei dem die Verzögerung von bis zu einer halben Sekunde ziemlich störend ist. Beim Abspielen von Musik und Filmen über das Netz spielt die absolute Verzögerung keine Rolle, da sie sich nur auf den Abspielanfang auswirkt. Anders sieht es hingegen mit der Varianz der Verzögerung, dem sogenannten **Jitter**, aus. Er muss vom Player maskiert werden oder Sie erhalten eine unverständliche Audio- bzw. ruckelige Videowiedergabe.

In diesem Abschnitt werden wir einige Strategien besprechen, wie das Verzögerungsproblem in den Griff zu bekommen ist, sowie Protokolle zum Einrichten von Audio- und Videositzungen. Nach einer Einführung in digitales Audio und Video stellen wir drei Fälle vor, für die unterschiedliche Entwürfe verwendet werden. Der erste und einfachste ist das Streamen von gespeicherten Medien, wie das Anschauen eines Videos auf YouTube. Der nächstschwierige Fall betrifft das Streamen von Live-Medien. Zwei Beispiele hierfür sind das Internetradio und IPTV, bei denen Rundfunk- und Fernsehstationen ihre Sendungen an viele Benutzer live im Internet ausstrahlen. Der letzte und schwierigste Fall ist ein Aufruf, wie er mit Skype oder, allgemeiner, mit einer interaktiven Audio- und Videokonferenz erfolgen könnte.

Nur am Rande sei bemerkt, dass der Begriff **Multimedia** oft im Zusammenhang mit dem Internet in der Bedeutung von Audio und Video verwendet wird. Wörtlich bezeichnet Multimedia zwei oder mehr Medien. Dieser Definition zufolge ist dieses Buch auch eine Multimediarstellung, da sie Text und Grafiken (die Abbildungen) enthält. Das ist jedoch wahrscheinlich nicht das, was Sie im Sinn hatten, sodass wir den Begriff „Multimedia“ verwenden, um zwei oder mehr **fortlaufende Medien** zu bezeichnen, das heißt, Medien, die während eines genau definierten Zeitraums abgespielt werden müssen. Bei den beiden Medien handelt es sich normalerweise um Video mit Audio, genau genommen um bewegte Bilder mit Ton. Viele bezeichnen auch reines Audio, wie Internettelefonie oder Internetradio als Multimedia, was es eindeutig nicht ist. Ein besserer Begriff für all diese Fälle ist **Streaming Media**. Nichtsdestotrotz werden wir uns der Mehrheit anschließen und Echtzeit-Audio auch als Multimedia betrachten.

7.4.1 Digitales Audio

Eine Klangwelle ist eine eindimensionale akustische Druckwelle. Tritt eine akustische Welle in das menschliche Ohr ein, vibriert das Trommelfell und damit auch die winzigen Knochen des Innenohrs, wodurch Nervenimpulse an das Gehirn gesendet werden. Diese Impulse werden vom Zuhörer als Klang wahrgenommen. Trifft eine akustische Welle auf ein Mikrofon, erzeugt das Mikrofon ein elektrisches Signal, das die Klangamplitude als Funktion der Zeit darstellt.

Der Frequenzbereich des menschlichen Ohres reicht von 20 Hz bis 20 000 Hz. Einige Tiere, vor allem Hunde, können viel höhere Frequenzen hören. Das Ohr hört logarithmisch, sodass das Verhältnis von zwei Klängen mit den Amplituden A und B konventionell in **dB (Dezibel)** als $10 \log_{10}(A/B)$ ausgedrückt wird. Definieren wir die untere Grenze der Hörbarkeit (ein Druck von etwa 20 μPascal) für eine 1-kHz-Sinuswelle als 0 dB, dann verläuft eine gewöhnliche Unterhaltung bei etwa 50 dB und die Schmerzgrenze liegt bei etwa 120 dB, ein Dynamikbereich mit einem Faktor von 1 Million.

Das Ohr ist erstaunlich empfänglich für Klangschwankungen, die nur ein paar Millisekunden dauern. Das Auge hingegen nimmt Änderungen des Lichtsignals in diesem Zeitraum nicht wahr. Aus dieser Beobachtung ergibt sich, dass Jitter von nur wenigen Millisekunden während einer Multimedia-Übertragung die wahrgenommene Tonqualität mehr beeinträchtigt als die wahrgenommene Bildqualität.

Digitales Audio ist die digitale Repräsentation einer Klangwelle, mit der diese wieder erzeugt werden kann. Klangwellen können mit einem **Analog/Digital-Wandler (A/D-Wandler, analog digital converter)** in digitale Form konvertiert werden. Ein A/D-Wandler nimmt eine elektrische Spannung als Eingabe und erzeugt eine Binärzahl als Ausgabe. In ► Abbildung 7.42a sehen wir ein Beispiel einer Sinuswelle. Um dieses Signal digital darzustellen, kann man es alle ΔT Sekunden abtasten, was durch die Höhe der Striche in ► Abbildung 7.42b dargestellt ist. Ist eine Klangwelle keine reine Sinuswelle, sondern eine lineare Überlagerung von Sinuswellen, wobei die höchste dargestellte Frequenzkomponente f ist, dann reicht es laut Nyquist-Theorem (siehe Kapitel 2) aus, in einer Frequenz von $2f$ abzutasten. Häufigeres Abtasten wäre sinnlos, weil die höheren Frequenzen, die dabei erkannt werden könnten, nicht vorhanden sind.

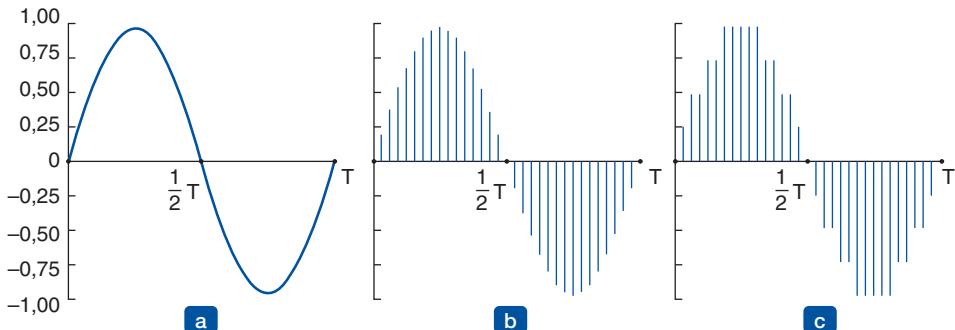


Abbildung 7.42: (a) Eine Sinuswelle. (b) Abtasten der Sinuswelle. (c) Quantisierung der Abtastwerte mit 4 Bit.

Der umgekehrte Prozess nimmt digitale Werte entgegen und erzeugt eine analoge elektrische Spannung. Hierzu benötigen Sie einen **Digital/Analog-Wandler (D/A-Wandler, digital analog converter)**. Ein Lautsprecher kann dann die analoge Spannung in akustische Wellen umwandeln, sodass die Menschen Töne hören können.

Digitale Samples sind nie genau. Die Abtastungen von ► Abbildung 7.42c erlauben nur neun Werte, von $-1,00$ bis $+1,00$ in Schritten von $0,25$. Ein 8-Bit-Sample würde 256 unterschiedliche Werte liefern, ein 16-Bit-Sample 65 536 unterschiedliche Werte. Der durch die endliche Bitzahl pro Sample eingebrachte Fehler heißt **Quantisierungsrauschen** (*quantization noise*). Ist dieses Rauschen zu hoch, kann es vom Ohr erkannt werden.

Zwei bekannte Beispiele für Klangabtastung sind das Telefon und CDs. Die im Telefonssystem verwendete Pulscodemodulation (PCM) basiert auf 8 000 8-Bit-Samples pro Sekunde. Die Skala ist nicht linear, um die empfundenen Verzerrungen zu minimieren, und mit nur 8 000 Samples pro Sekunde gehen die Frequenzen über 4 kHz verloren. In Nordamerika und Japan wird die **μ -law**-Codierung verwendet, während in Europa und im Rest der Welt die **A-law**-Codierung zugrunde gelegt wird. Jede Codierung liefert eine Datenrate von 64 000 Bit/s.

Audio-CDs sind digital und haben eine Abtastrate von 44 100 Samples/s. Das ist ausreichend, um Frequenzen von bis zu 22 050 Hz zu erfassen. Diese Qualität ist gut genug für Menschen, aber nicht zufriedenstellend für Hunde, die Musik lieben. Die Samples sind je 16 Bit und verlaufen linear über den Amplitudenbereich. Man beachte, dass 16-Bit-Samples nur 65 536 getrennte Werte ermöglichen, obwohl der dynamische Bereich des Ohrs über eine Million liegt. Das bedeutet, dass auch wenn Audio in CD-Qualität viel besser ist als Audio in Telefonqualität, bei der Verwendung von 16-Bit pro Sample mit hörbarem Quantisierungsrauschen zu rechnen ist (allerdings wird der volle dynamische Bereich nicht abgedeckt – CDs sollen ja keine Schmerzen verursachen). Einige besonders fanatische audiophile Musikliebhaber ziehen die alten Langspielplatten den CDs vor, da bei diesen Platten die Nyquist-Frequenz nicht bei 22 kHz abgeschnitten wird und kein Quantisierungsrauschen auftritt. (Allerdings müssen Sie mit Kratzern rechnen, wenn Sie nicht besonders sorgfältig damit umgehen.) Mit 44.100 Samples/s von je 16 Bit benötigt unkomprimiertes Audio von CD-Qualität eine Bandbreite von 705,6 kbit/s bei Mono und 1,411 Mbit/s bei Stereo.

Audiokomprimierung

Audiodaten werden oft komprimiert, um die Übertragungsraten und den Bedarf an Bandbreite zu reduzieren, auch wenn die Übertragungsraten für Audiodaten viel geringer sind als für Videodaten. Alle Komprimierungssysteme benötigen zwei Algorithmen: einen zum Komprimieren der Daten an der Quelle und einen zum Dekomprimieren am Ziel. In der Literatur werden diese Algorithmen als **Codier- und Decodieralgorithmen** bezeichnet. Wir werden uns an diese Terminologie halten.

Komprimierungsalgorithmen weisen bestimmte Asymmetrien auf, die Sie auf alle Fälle verstehen sollten. Bei vielen Anwendungen wird ein Multimediadokument nur einmal codiert (wenn es auf dem Multimediaserver gespeichert wird), aber Tausende Male decodiert (wenn es beim Kunden abgespielt wird). Diese Asymmetrie bedeutet, dass der Codieralgorithmus ruhig langsam sein und teure Hardware voraussetzen darf, solange der Decodieralgorithmus schnell ist und auf preisgünstiger Hardware läuft. Der Betreiber eines beliebten Audio- (oder Video-)Servers könnte durchaus bereit sein, einen Computer-Cluster zu kaufen, um seine ganze Bibliothek zu codieren, aber dies vom Kunden zu verlangen, damit er Musik hören oder Filme anschauen kann, dürfte keine große Begeisterung auslösen. Deshalb gibt es viele praktische Komprimierungssysteme, die den Schwerpunkt darauf legen, das Decodieren möglichst einfach und schnell zu machen, auch wenn das bedeutet, dass das Codieren langsam und kompliziert wird.

Für Live-Audio und -Video wie VoIP-Anrufe ist eine langsame Codierung hingegen nicht akzeptabel. Die Codierung muss in Echtzeit, d.h. möglichst zeitgleich erfolgen. Folglich verwendet Echtzeit-Multimedia andere Algorithmen oder Parameter als gespeicherte Audio- oder Videodaten auf Festplatte, oft sogar mit deutlich geringerer Komprimierung.

Eine zweite Asymmetrie ist, dass der Codier-/Decodierprozess nicht umkehrbar sein muss. Das heißt, nach dem Komprimieren, Übertragen und anschließenden Dekomprimieren einer Datei erwartet der Benutzer, dass er das Original bis zum letzten Bit zurückhält. Bei Multimedia besteht diese Anforderung nicht. Es ist normalerweise akzeptabel, wenn sich das Audio- oder Videosignal nach dem Codieren und Decodieren leicht vom Original unterscheidet, sofern es genauso klingt (oder aussieht). Wenn die decodierte Ausgabe nicht absolut der ursprünglichen Eingabe entspricht, wird das System als **verlustbehaftet** (*lossy*) bezeichnet. Sind hingegen Eingabe und Ausgabe identisch, ist das System **verlustlos** (*lossless*). Verlustbehaftete Systeme sind durchaus wichtig, da ein kleiner Informationsverlust durch das zu erreichende Komprimierungsverhältnis mehr als kompensiert wird.

In früheren Zeiten war Langstreckenbandbreite im Telefonnetz sehr teuer, sodass häufig **Vocoder** (*voice coder*, Sprachcodierer) zum Komprimieren von Audiodaten, speziell der gesprochenen Sprache, zum Einsatz kamen. Die menschliche Sprache bewegt sich normalerweise im Frequenzbereich von 600–6 000 Hz und wird durch einen mechanischen Prozess erzeugt, der von dem Vokaltrakt, der Zunge und dem Kiefer abhängt. Einige Vocoder basieren auf Modellen des Vokalsystems, um die Sprache auf einige Parameter (z.B. die Größe und Form von verschiedenen Hohlräumen) und die Datenrate auf unter 2,4 kbit/s zu reduzieren.

Wir werden uns auf Audio konzentrieren, wie es über das Internet geschickt wird, was normalerweise näher an CD-Qualität ist. Es wäre durchaus erstrebenswert, die Datenübertragungsraten für diese Art von Audio zu verringern. Mit 1,411 Mbit/s würde Stereo-Audio viele Breitbandverbindungen binden, was wenig Raum für Video und anderen Webverkehr ließe. Die Datenübertragungsrate kann mit Komprimierung um eine Größenordnung reduziert werden, und zwar ohne oder kaum wahrnehmbaren Qualitätsverlust.

Komprimierung und Dekomprimierung benötigen Signalverarbeitung. Glücklicherweise können digitalisierte Töne und Filme leicht von Software in Computern verarbeitet werden. Es gibt Dutzende von Programmen, mit denen Medien von den unterschiedlichsten Quellen aufgenommen, abgespielt, bearbeitet, gemischt und gespeichert werden können. Das hat dazu geführt, dass im Internet reichlich illegale Musik und Filme kursieren, was Künstler und Urheberrechtsinhaber veranlasst hat, eine große Zahl von Klagen einzureichen.

Es sind viele Komprimierungsalgorithmen für Audiodaten entwickelt worden. Die wahrscheinlich bekanntesten Formate sind **MP3** (*MPEG audio layer 3*) und **AAC** (*Advanced Audio Coding*), wie es in **MP4**-Dateien (*MPEG-4*) übertragen wird. Um Verwirrung zu vermeiden, möchten wir darauf hinweisen, dass MPEG sowohl Audio- als auch Videokomprimierung bietet. MP3 bezieht sich auf den Audiokompressionsteil (Teil 3) des MPEG-1-Standards, nicht die dritte Version von MPEG. Genau genommen wurde keine dritte Version von MPEG veröffentlicht, nur MPEG-1, MPEG-2 und MPEG-4. AAC ist der Nachfolger von MP3 und die Standard-Audiocodierung in MPEG-4. MPEG-2 erlaubt sowohl MP3- und AAC-Audio. Alles klar? Das Gute an Standards ist, dass man unter vielen wählen kann. Und wenn Sie keinen davon mögen, warten Sie einfach ein oder zwei Jahre.

Audiokomprimierung kann auf zwei Arten vorgenommen werden. Bei der **Wellenform-Codierung** (*waveform coding*) wird das Signal mathematisch über eine Fourier-Transformation in die Frequenzkomponenten umgewandelt. Abbildung 2.1(a) enthält eine Beispieldarstellung über die Zeit mit den Fourier-Amplituden. Die Amplitude jeder Komponente wird dann minimal codiert. Das Ziel ist, am anderen Ende die Wellenform möglichst exakt und mit möglichst wenigen Bits zu reproduzieren.

Die andere Methode, die **wahrnehmungsbezogene Codierung** (*perceptual coding*), nutzt gewisse Beschränkungen des menschlichen Gehörs, um ein Signal zu codieren, sodass es für einen Hörer gleich klingt, auch wenn es auf einem Oszilloskop anders aussieht. Die wahrnehmungsbezogene Codierung basiert auf der **Psychoakustik**, die sich damit beschäftigt, wie Menschen Klang wahrnehmen. MP3 und AAC basieren beide auf der wahrnehmungsbezogenen Codierung.

Die zentrale Eigenschaft der wahrnehmungsbezogenen Codierung ist, dass einige Klänge andere **überdecken** können. Stellen Sie sich vor, dass an einem warmen Sommertag ein Flötenkonzert live übertragen wird. Plötzlich taucht eine Gruppe Arbeiter in der Nähe auf; sie schalten ihre Presslufthämmer ein und beginnen, die Straße aufzurütteln. Niemand kann die Flöte mehr hören. Die Töne werden von den Pressluft-hämmern überlagert. Zu Übertragungszwecken ist es nun ausreichend, nur das von

den Presslufthämmern verwendete Frequenzband zu codieren, da die Zuhörer die Flöte sowieso nicht hören können. Dies wird als **Frequenzmaskierung** (*frequency masking*) bezeichnet – die Fähigkeit eines lauten Tons, auf einem Frequenzband einen leiseren Klang auf einem anderen Frequenzband zu überlagern, der ansonsten gut hörbar gewesen wäre. Selbst wenn die Presslufthämmer aufhören, ist die Flöte für kurze Zeit nicht hörbar, da das Ohr die Verstärkung reduziert, wenn sie zu hämmern beginnen, und es eine bestimmte Zeit dauert, diese wieder zu erhöhen. Dieser Effekt wird als **temporäre Maskierung** (*temporal masking*) bezeichnet.

Um diese Effekte etwas besser zu quantifizieren, stellen Sie sich Experiment 1 vor. Eine Person in einem ruhigen Raum setzt sich einen Kopfhörer auf, der mit der Soundkarte des Computers verbunden ist. Der Computer erzeugt eine reine Sinuswelle mit 100 Hz, erhöht aber langsam die Lautstärke. Die Person wird angewiesen, eine Taste zu drücken, wenn sie einen Ton hört. Der Computer zeichnet den aktuellen Lautstärkepegel auf und wiederholt das Experiment bei 200 Hz, 300 Hz und allen weiteren Frequenzen bis zur Obergrenze des menschlichen Gehörs. Wenn man hierfür über viele Personen einen Durchschnitt ermittelt, erhalten Sie den Log-Log-Graph aus ►Abbildung 7.43a. Er spiegelt wider, wie hoch die Lautstärke sein muss, damit ein Ton hörbar ist. Eine direkte Folge dieser Kurve ist, dass Frequenzen, deren Lautstärke unter die Hörbarkeitsgrenze fällt, nie codiert werden müssen. Wenn beispielsweise die Lautstärke bei 100 Hz 20 dB in Abbildung 7.43a beträgt, könnte dies bei der Ausgabe ohne wahrnehmbaren Qualitätsverlust wegfallen, da 20 dB bei 100 Hz unter die Hörbarkeitsgrenze fallen.

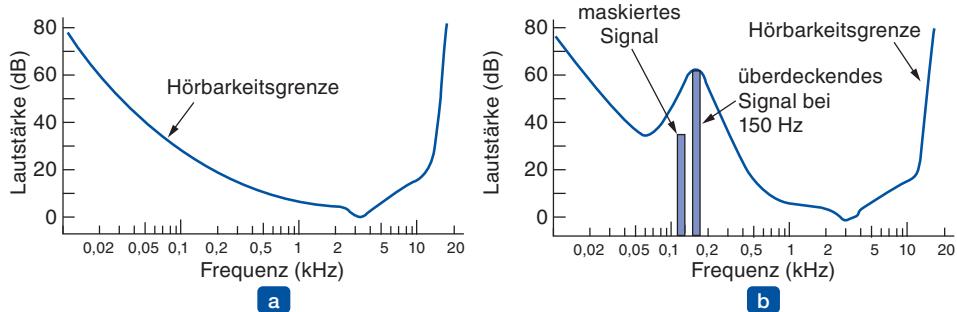


Abbildung 7.43: (a) Die Hörbarkeitsschwelle als Funktion der Frequenz. (b) Der Maskierungseffekt.

Betrachten wir nun Experiment 2. Der Computer führt erneut Experiment 1 aus, überdeckt dieses Mal aber die Testfrequenz mit einer Sinuswelle von z.B. 150 Hz und konstanter Amplitude. Dabei stellen wir fest, dass die Hörbarkeitsschwelle für Frequenzen nahe 150 Hz erhöht wird (►Abbildung 7.43b).

Diese neue Beobachtung hat Folgen, denn wenn wir wissen, welche Signale von lautstärkeren Signalen auf benachbarten Frequenzbändern überdeckt werden, können wir immer mehr Frequenzen im codierten Signal weglassen und somit Bits sparen. In Abbildung 7.43 kann das 125-Hz-Signal bei der Ausgabe komplett weggelassen werden, da niemand den Unterschied hören wird. Selbst nach Beenden eines lautstarken

Signals auf einem Frequenzband können wir dank unserer Kenntnis der temporären Maskierungseigenschaften die maskierten Frequenzen noch eine Zeit lang weglassen, da es dauert, bis sich das Ohr angepasst hat. Der Kern von MP3 und AAC ist die Fourier-Transformation des Tons, um die Lautstärke auf jeder Frequenz zu erhalten und dann nur die nicht überdeckten Frequenzen zu übertragen, wobei diese in möglichst wenig Bits codiert werden.

Mit dieser Hintergrundinformation verstehen wir nun, wie die Codierung ausgeführt wird. Die Audiokomprimierung erfolgt durch Abtasten der Wellenform mit einer Rate von 8 bis 96 kHz für AAC, oft mit 44,1 kHz, um CD-Klang nachzuahmen. Es können entweder ein (mono) oder zwei Kanäle (stereo) abgetastet werden. Als Nächstes wird die Ausgabebitrate gewählt. MP3 kann beispielsweise eine CD mit Rock 'n Roll ohne wahrnehmbaren Qualitätsverlust auf 96 kbit/s komprimieren. Nicht einmal Rockfans, die noch keinen Gehörschaden haben, würden etwas merken. Für ein Klavierkonzert ist AAC mit mindestens 128 kbit/s erforderlich. Dieser Unterschied ist darauf zurückzuführen, dass der Rauschabstand bei Rock 'n Roll viel höher ist als bei einem Klavierkonzert (zumindest vom technischen Standpunkt aus). Man kann auch niedrigere Ausgaberaten wählen und dafür einen gewissen Qualitätsverlust in Kauf nehmen.

Die Abtastungen werden dann in kleinen Gruppen verarbeitet. Jede Gruppe wird zuerst durch eine Reihe digitaler Filter geleitet, um Frequenzbänder zu erhalten. Diese Frequenzdaten werden in ein psychoakustisches Modell eingespeist, um die überdeckten Frequenzen zu bestimmen. Im nächsten Schritt wird das verfügbare Bit-Budget auf die Bänder aufgeteilt, wobei den Bändern mit der meisten nicht überdeckten Spektralleistung mehr Bits, den nicht überdeckten Bändern mit weniger Spektralleistung weniger Bits und den überdeckten Bändern keine Bits zugewiesen werden. Schließlich werden die Bits unter Verwendung der Huffman-Codierung codiert, die häufig auftretenden Werten kürzere Codes und weniger häufig auftretenden Werten lange Codes zuweist. Für den interessierten Leser, der tiefer in die Materie einsteigen möchte, sei Brandenburg (1999) empfohlen.

7.4.2 Digitales Video

Nachdem wir jetzt alles über das Ohr wissen, ist es an der Zeit, zum Auge überzugehen (nein, danach folgt kein Abschnitt über die Nase). Das menschliche Auge hat die Eigenschaft, dass ein Bild, wenn es auf die Netzhaut auftrifft, einige Millisekunden haften bleibt, bevor es verschwindet. Wird eine Bildfolge in 50 oder mehr Bildern pro Sekunde ausgegeben, kann das Auge nicht erkennen, dass es einzelne Bilder sind. Alle Videosysteme machen sich dieses Prinzip zunutze, um bewegte Bilder zu produzieren.

Die einfachste Darstellung von digitalem Video ist eine Folge von Rahmen, die jeweils aus einem rechteckigen Raster aus Bildelementen – den **Pixeln** – bestehen. Jedes Pixel kann ein einzelnes Bit sein, das entweder Schwarz oder Weiß darstellt. Die Qualität eines solchen Systems ist schrecklich. Versuchen Sie einmal, mit Ihrem Lieblingsbildeditor die Pixel eines Farbbildes in schwarz und weiß (und *nicht* in Grauschattierungen) umzuwandeln.

Im nächsten Schritt werden 8 Bit pro Pixel verwendet, um 256 Graustufen darzustellen. Dieses Schema ergibt ein Schwarzweißvideo hoher Qualität. Für Farbvideo verwenden gute Systeme 8 Bit pro RGB-Farbe (die roten, grünen und blauen Primärfarbakomponenten). Diese Darstellung ist möglich, weil jede Farbe aus der linearen Überlagerung von Rot, Grün und Blau in den entsprechenden Intensitäten erstellt werden kann. Bei 24 Bit pro Pixel gibt es ungefähr 16 Millionen Farben, was mehr ist, als das menschliche Auge unterscheiden kann.

Auf LCD-Farbbildschirmen für Computer und Fernsehen besteht jedes einzelne Pixel aus dicht nebeneinanderstehenden roten, grünen und blauen Teilpixeln. Rahmen werden durch das Setzen der Intensität der Teilpixel angezeigt und das Auge mischt die Farbkomponenten.

Übliche Bildübertragungsraten liegen bei 24 Rahmen/s (für 35mm-Kleinbildfilme), 30 Rahmen/s (gemäß NTSC-Standard des US-Fernsehens) und 25 Rahmen/s (beim PAL-System, wie es fast überall im Rest der Welt verwendet wird). (Für die, die es genau wissen wollen, sei erwähnt, dass NTSC-Farbfernsehen eigentlich mit 29,97 Rahmen/s läuft. Das ursprüngliche Schwarzweißsystem lief mit 30 Rahmen/s, aber als Farbe ins Spiel kam, benötigten die Techniker zur Signalisierung zusätzliche Bandbreite, sodass sie die Bildrate auf 29,97 reduzierten. NTCS-Videos, die für Computer bestimmt sind, verwendeten tatsächlich 30 Rahmen/s.) PAL wurde nach NTSC erfunden und verwendet tatsächlich 25 Rahmen/s. Und um das Ganze abzurunden, gibt es mit SECAM ein drittes System, das in Frankreich, dem frankophonen Afrika und Osteuropa verwendet wird. Es wurde über das kommunistische Ostdeutschland in Osteuropa eingeführt, damit die Ostdeutschen kein Westfernsehen (PAL) sehen konnten und nicht auf dumme Ideen kamen. Aber viele dieser Länder wechseln inzwischen zu PAL. Technik und Politik in Bestform.

Eigentlich sind für Fernsehsendungen 25 Rahmen/s nicht gut genug, um ein ruckelfreies Abspielen zu garantieren. Deshalb werden die Bilder in zwei **Felder** geteilt, eines mit ungeraden und eines mit geraden Abtastzeilen. Die beiden (halbauflösenden) Felder werden sequenziell gesendet, was fast 60 (NTSC) oder genau 50 (PAL) Felder pro Sekunde ergibt. Dieses System wird auch als **Interlacing** bezeichnet. Videos, die zum Betrachten auf Computern bestimmt sind, sind **progressiv**, d.h., sie verwenden kein Interlacing, da Computerbildschirme auf ihren Grafikkarten Puffer zur Verfügung haben, die es der CPU ermöglichen, 30 Mal/s ein neues Bild im Puffer abzulegen, während die Grafikkarte 50 oder sogar 100 Mal/s den Bildschirm neu zeichnet, um das Flimmern zu beseitigen. Analoge Fernseher haben keinen Rahmenpuffer, wie er bei Computern zu finden ist. Wenn ein mit Interlacing erzeugtes Video mit schneller Bewegung auf einem Computer abgespielt wird, werden kurze horizontale Zeilen neben scharfen Kanten sichtbar. Diesen Effekt nennt man **Kammeffekt (combing)**.

Die Rahmengrößen für Videos über das Internet schwanken stark, aus dem einfachen Grunde, dass größere Rahmen mehr Bandbreite benötigen, die nicht immer zur Verfügung steht. Ein Video mit niedriger Auflösung könnte 320×240 Pixel aufweisen und ein Video im Vollbildmodus 640×480 Pixel. Diese Abmaße sind nahe an denen der frühen Computermonitore bzw. NTSC-Fernsehen. Das **Seitenverhältnis (aspect ratio)** bzw. das

Breite-zu-Höhe-Verhältnis beträgt 4:3 und entspricht damit dem Standardfernsehen. **HDTV**-Videos (*high-definition television*) können mit $1\,280 \times 720$ Pixel heruntergeladen werden. Diese Breitbildauflösung hat ein Seitenverhältnis von 16:9, was sich eher dem 3:2-Seitenverhältnis von Filmen annähert. Zum Vergleich, Standard-DVD-Video hat normalerweise 720×480 Pixel und Video auf Blu-Ray-Discs ist normalerweise HDTV mit $1\,080 \times 720$ Pixel.

Im Internet ist die Anzahl der Pixel nur ein Teil der Geschichte, da Mediaplayer das gleiche Bild in verschiedenen Größen anzeigen können. Video ist nur ein weiteres Fenster auf dem Computerbildschirm, das vergrößert oder verkleinert werden kann. Die Aufgabe weiterer Pixel ist es, die Qualität des Bildes zu verbessern, sodass es nicht verschwommen aussieht, wenn es vergrößert wird. Viele Monitore können inzwischen jedoch Bilder (und damit Videos) anzeigen, die mehr Pixel haben als HDTV.

Videokomprimierung

Unseren bisherigen Ausführungen zu digitalem Video sollten Sie entnommen haben, dass die Komprimierung für das Senden von Videos über das Internet sehr wichtig ist. Sogar ein Video normaler Qualität mit 640×480 Pixelrahmen, 24 Bit Farbinformationen pro Pixel und 30 Rahmen/s benötigt über 200 Mbit/s. Dies übersteigt bei Weitem die Bandbreite, mit der die meisten Firmenbüros mit dem Internet verbunden sind, ganz zu schweigen von privaten Nutzern. Und dabei handelt es sich hier nur um einen einzigen Videostrom. Da die Übertragung nicht komprimierter Videos absolut außer Frage steht, zumindest über WAN-Netze, besteht die einzige Hoffnung darin, dass eine starke Komprimierung möglich ist. Zum Glück haben große Forschungsanstrengungen in den letzten Jahrzehnten viele Komprimierungstechniken und -algorithmen hervorgebracht, die die Videoübertragung möglich gemacht haben.

Für Video, das über das Internet übertragen wird, werden viele Formate verwendet, von denen einige firmenrechtlich geschützt, andere Standard sind. Die am weitesten verbreitete Codierung ist MPEG in verschiedenen Formen. Es ist ein offener Standard, der in Dateien mit der Endung mpg und mp4 sowie in anderen Containerformaten zu finden ist. In diesem Abschnitt werden wir uns mit MPEG beschäftigen, um zu zeigen, wie die Videokomprimierung funktioniert. Zu Beginn werden wir uns die Komprimierung von Standbildern mit JPEG anschauen. Ein Video ist schließlich nur eine Folge von Bildern (plus Ton). Eine Möglichkeit, Video zu komprimieren, besteht darin, die Einzelbilder nacheinander zu codieren. Als erste Annäherung sei erwähnt, dass MPEG nur die JPEG-Codierung von jedem Rahmen ist, plus einigen Extras, um die Redundanz aus den Rahmen zu entfernen.

Der JPEG-Standard

Der **JPEG**-Standard (Joint Photographic Experts Group) für die Komprimierung von Halbtontiefenbildern (z.B. Fotos) wurde von Fotoexperten unter der Schirmherrschaft von ITU, ISO und IEC sowie anderer Standardisierungsgremien entwickelt. Er ist weitverbreitet (suchen Sie nach Dateien mit der Extension jpg) und liefert für natürliche Bilder oft Kompressionsverhältnisse von 10:1 und besser.

JPEG ist im International Standard 10918 definiert. Es handelt sich mehr um eine Einkaufsliste als um einen einzelnen Algorithmus, aber von den vier definierten Modi ist für unsere Zwecke nur der verlustbehaftete sequenzielle Modus relevant. Des Weiteren konzentrieren wir uns darauf, wie JPEG normalerweise genutzt wird, um 24-Bit-RGB-Videobilder zu codieren. Um es einfach zu halten, werden einige Details und Optionen weggelassen.

Der Algorithmus wird in ► Abbildung 7.44 veranschaulicht. Schritt 1 ist die Blockvorbereitung. Der Genauigkeit halber lassen Sie uns annehmen, dass die JPEG-Eingabe ein RGB-Bild von 640×480 mit 24 Bit/Pixel ist (► Abbildung 7.45a). RGB ist nicht das beste Farbmodell, das sich für die Komprimierung anbietet. Das Auge reagiert viel empfindlicher auf die **Luminanz** (Helligkeit) des Videosignals als auf die **Chrominanz** (Farbe) des Videosignals. Folglich berechnen wir zuerst die Luminanz Y und dann die beiden Chrominanzsignale Cb und Cr aus den Komponenten R, G und B. 8-Bit-Werte, die von 0 bis 255 reichen, berechnen sich mit folgenden Formeln:

$$Y = 16 + 0,26R + 0,50G + 0,09B$$

$$Cb = 128 + 0,15R - 0,29G - 0,44B$$

$$Cr = 128 + 0,44R - 0,37G + 0,07B$$

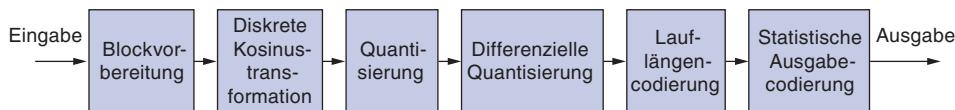


Abbildung 7.44: JPEG im verlustbehafteten sequenziellen Modus.

Für Y, Cb und Cr werden jeweils eigene Matrizen erstellt. Als Nächstes werden die Durchschnitte aus quadratischen Blöcken von vier Pixeln in den Cb- und Cr-Matrizen gebildet, um sie auf 320×240 zu reduzieren. Diese Reduktion ist verlustbehaftet, aber das Auge nimmt dies kaum wahr, da es stärker auf Luminanz als auf Chrominanz reagiert. Nichtsdestotrotz werden die Daten insgesamt um einen Faktor von zwei verdichtet. Nun wird 128 von jedem Element in allen drei Matrizen abgezogen, um 0 in die Mitte des Wertebereichs zu stellen. Schließlich wird jede Matrix in Blöcke von 8×8 aufgeteilt. Die Y-Matrix hat 4 800 Blöcke, die anderen beiden haben je 1 200 Blöcke (siehe ► Abbildung 7.45b).

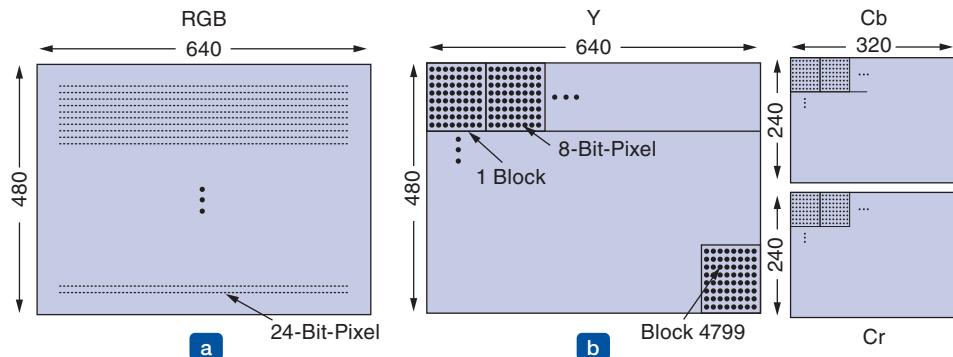


Abbildung 7.45: (a) RGB-Eingabedaten; (b) nach der Blockvorbereitung.

In Schritt 2 von JPEG wird auf jeden der 7 200 Blöcke getrennt eine **diskrete Kosinustransformation (DCT, discrete cosine transformation)** angewendet. Jede DCT-Ausgabe ist eine Matrix aus 8×8 DCT-Koeffizienten. DCT-Element (0, 0) ist der Durchschnittswert des Blocks. Die anderen Elemente geben an, wie viel spektrale Leistung in jeder räumlichen Frequenz vorhanden ist. Normalerweise fallen diese Elemente mit der Entfernung vom Ausgangspunkt (0, 0) schnell ab, wie ▶ Abbildung 7.46 zeigt.

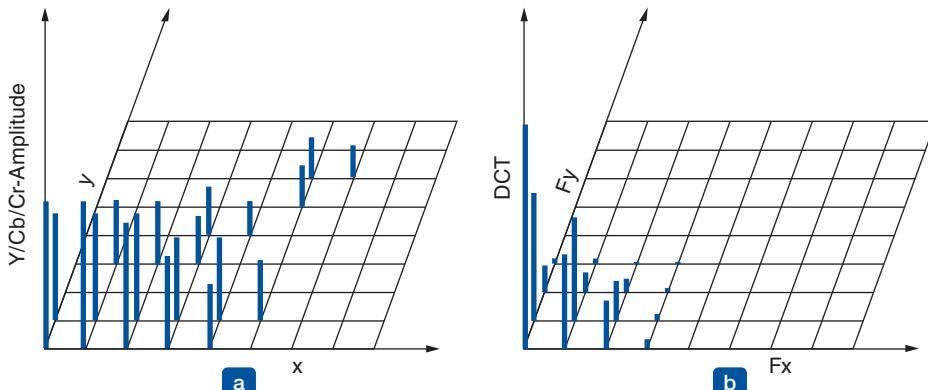


Abbildung 7.46: (a) Ein Block der Y -Matrix. (b) Die DCT-Koeffizienten.

Sobald die diskrete Kosinustransformation abgeschlossen ist, folgt in Schritt 3 die **Quantisierung**, bei der die weniger wichtigen DCT-Koeffizienten eliminiert werden. Diese (verlustbehaftete) Umwandlung erfolgt durch Dividieren der Koeffizienten in der 8×8 -DCT-Matrix durch eine Gewichtung, die einer Tabelle entnommen wird. Sind alle Gewichtungen 1, bewirkt die Umwandlung nichts. Steigen sie vom Ursprung aus stark an, fallen höhere räumliche Frequenzen schnell weg.

▶ Abbildung 7.47 zeigt ein Beispiel dieses Schritts. Hier sehen wir die anfängliche DCT-Matrix, die Quantisierungstabelle und das Ergebnis der Division jedes DCT-Elements durch das entsprechende Element der Quantisierungstabelle. Die Werte der Quantisierungstabelle gehören nicht zum JPEG-Standard. Jede Anwendung muss eigene Werte bereitstellen, sodass der Kompromiss zwischen Verlust und Komprimierung gesteuert werden kann.

| DCT-Koeffizienten | Quantisierungstabelle | Quantisierte Koeffizienten |
|----------------------|-------------------------|----------------------------|
| 150 80 40 14 4 2 1 0 | 1 1 2 4 8 16 32 64 | 150 80 20 4 1 0 0 0 |
| 92 75 36 10 6 1 0 0 | 1 1 2 4 8 16 32 64 | 92 75 18 3 1 0 0 0 |
| 52 38 26 8 7 4 0 0 | 2 2 2 4 8 16 32 64 | 26 19 13 2 1 0 0 0 |
| 12 8 6 4 2 1 0 0 | 4 4 4 4 8 16 32 64 | 3 2 2 1 0 0 0 0 |
| 4 3 2 0 0 0 0 0 | 8 8 8 8 8 16 32 64 | 1 0 0 0 0 0 0 0 |
| 2 2 1 1 0 0 0 0 | 16 16 16 16 16 16 32 64 | 0 0 0 0 0 0 0 0 |
| 1 1 0 0 0 0 0 0 | 32 32 32 32 32 32 32 64 | 0 0 0 0 0 0 0 0 |
| 0 0 0 0 0 0 0 0 | 64 64 64 64 64 64 64 64 | 0 0 0 0 0 0 0 0 |

Abbildung 7.47: Berechnung der quantisierten DCT-Koeffizienten.

Schritt 4 verkleinert den $(0, 0)$ -Wert jedes Blocks (der Wert in der oberen linken Ecke), indem er ihn durch den Betrag ersetzt, um den er sich vom entsprechenden Element im vorherigen Block unterscheidet. Da diese Elemente die Durchschnittswerte ihrer jeweiligen Blöcke sind, sollten sie sich langsam ändern, sodass die Verwendung der Differenzwerte die meisten auf kleine Werte reduzieren sollte. Aus den anderen Werten werden keine Differenzen berechnet.

Schritt 5 linearisiert die 64 Elemente und wendet die Lauflängencodierung auf die Liste an. Durch Abtasten des Blocks von links nach rechts und dann von oben nach unten werden die Nullen nicht verdichtet, sodass ein Zickzackabtastmuster verwendet wird (►Abbildung 7.48). Bei diesem Beispiel produziert das Zickzackmuster letztendlich 38 aufeinanderfolgende Nullen am Ende der Matrix. Diese Kette kann auf eine einzige Zahl reduziert werden, die besagt, dass 38 Nullen vorhanden sind, eine Technik, die als **Lauflängencodierung** (*run-length encoding*) bezeichnet wird.

| | | | | | | | |
|-----|----|----|---|---|---|---|---|
| 150 | 80 | 20 | 4 | 1 | 0 | 0 | 0 |
| 92 | 75 | 18 | 3 | 1 | 0 | 0 | 0 |
| 26 | 19 | 13 | 2 | 1 | 0 | 0 | 0 |
| 3 | 2 | 2 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Abbildung 7.48: Reihenfolge, in der die quantisierten Werte übertragen werden.

Damit haben wir eine Liste mit Zahlen, die das Bild (im Transformationsraum) darstellen. In Schritt 6 werden die Zahlen zum Speichern oder Übertragen nach Huffman codiert, wobei die gängigen Zahlen kürzere Codes als die nicht so häufig auftretenden Zahlen erhalten.

JPEG mag kompliziert erscheinen. Der Grund liegt darin, dass es wirklich kompliziert ist. Trotzdem wird die Technik häufig verwendet, weil sie ein Komprimierungsverhältnis von 10:1 oder noch besser liefert. Zum Decodieren eines JPEG-Bilds muss der Algorithmus rückwärts ausgeführt werden. JPEG ist in etwa symmetrisch: Die Decodierung dauert so lange wie die Codierung. Diese Eigenschaft gilt nicht für alle Komprimierungsalgorithmen, wie wir jetzt sehen werden.

Der MPEG-Standard

Wir kommen nun zum Kern der Sache: die **MPEG**-Standards (*Motion Picture Experts Group*). Auch wenn es viele firmeneigene Algorithmen gibt, definieren diese Standards die wichtigsten Algorithmen, die zur Komprimierung von Video verwendet werden.

Da Filme nicht nur Bilder sondern auch Ton enthalten, kann MPEG sowohl Audio- wie auch Videodateien komprimieren. Wir haben die Audiokomprimierung und Standbildkomprimierung bereits behandelt. Nun wenden wir uns der Videokomprimierung zu.

Der MPEG-1-Standard (zu dem auch MP3 Audio gehört) wurde das erste Mal 1993 international veröffentlicht und wird immer noch häufig verwendet. Sein Ziel war die Erzeugung von Ausgaben in Videorekorderqualität mit einer Komprimierung von 40:1 bei Raten von ungefähr 1 Mbit/s. Dieses Video eignet sich für breite Internetnutzung auf Websites. All denjenigen, die Videorekorder gar nicht mehr kennen zur Information: MPEG-1 wurde auch zum Speichern von Filmen auf CDs verwendet. Und wenn Sie CDs auch nicht kennen, lassen Sie uns einfach zu MPEG-2 übergehen.

Der MPEG-2-Standard wurde 1996 veröffentlicht und definiert die Komprimierung von Video in Fernsehqualität. Er ist heutzutage weitverbreitet, da er als Grundlage für Videos dient, die auf DVDs codiert werden (und unweigerlich ihren Weg ins Internet finden), sowie für digitales Fernsehen (wie DVB). Video in DVD-Qualität wird normalerweise mit Raten von 4–8 Mbit/s codiert.

Der MPEG-4-Standard umfasst zwei Videoformate. Das erste Format, das 1999 veröffentlicht wurde, codiert Video mit einer objektbasierten Darstellung. Dies erlaubt die Kombination von natürlichen und synthetischen Bildern und anderen Arten von Medien, zum Beispiel eine Wetterfee, die vor einer Wetterkarte steht. Diese Struktur ermöglicht es, Programme mit Filmdaten interagieren zu lassen. Das zweite Format, das es seit 2003 gibt, trägt die Bezeichnung **H.264** oder **AVC** (*Advanced Video Coding*). Es soll Videos bei gleichbleibender Qualität mit nur der Hälfte der Rate vorheriger Codierer codieren, um die Übertragung von Video über Netze noch besser zu unterstützen. Dieser Codierer wird für HDTV auf den meisten Blu-Ray-Discs verwendet.

Die Einzelheiten all dieser Standards sind umfangreich und vielfältig. Die späteren Standards haben außerdem viel mehr Funktionen und Codieroptionen als die älteren. Darauf wollen wir jedoch nicht näher eingehen. Größtenteils sind die Errungenschaften in der Videokomprimierung über die Zeit eher auf viele kleine Verbesserungen zurückzuführen als auf grundlegende Veränderungen in der Art und Weise, wie Video komprimiert wird. Deshalb werden wir die Gesamtkonzepte kurz anreißen.

MPEG komprimiert sowohl Audio als auch Video. Die Audio- und Videocodierer arbeiten unabhängig voneinander, was die Frage aufwirft, wie die zwei Ströme beim Empfänger synchronisiert werden. Dieses Problem wird durch einen einzigen Taktgeber gelöst, der den aktuellen Zeitwert an beide Codierer ausgibt. Diese Zeitstempel werden in die codierte Ausgabe eingebunden und bis zum Empfänger mit übertragen, der damit die Audio- und Videoströme synchronisieren kann.

MPEG-Videokomprimierung macht sich zwei Arten von Redundanzen zunutze, die in Filmen auftreten: räumliche und zeitliche. Die räumliche Redundanz kann genutzt werden, indem einfach jeder Rahmen getrennt mit JPEG codiert wird. Dieser Ansatz wird manchmal verfolgt, insbesondere wenn auf jeden Rahmen separat zugegriffen werden muss, etwa beim Editieren von Videoproduktionen. In diesem Modus werden JPEG-Komprimierungsstufen erreicht.

Zusätzliche Komprimierung kann erreicht werden, wenn man die Tatsache nutzt, dass aufeinanderfolgende Rahmen oft fast identisch sind. Dieser Effekt ist geringer als man zunächst annehmen sollte, weil viele Filmproduzenten alle drei oder vier Sekunden zwischen Szenen schneiden (stoppen Sie einmal einen Filmausschnitt und zählen Sie die Szenen). Dennoch eignet sich auch eine Folge von 75 sehr ähnlichen Rahmen für eine beträchtliche Reduzierung im Vergleich zur einfachen Codierung der einzelnen Rahmen mit JPEG.

Bei Szenen, in denen Kameraposition und Hintergrund unverändert bleiben, während sich ein oder zwei Darsteller langsam bewegen, sind fast alle Pixel von Rahmen zu Rahmen identisch. In diesem Falle genügt es durchaus, jeden Rahmen vom jeweils vorherigen abzuziehen und die JPEG-Komprimierung auf die Differenz anzuwenden. Bei Szenen, in denen die Kamera häufig geschwenkt oder viel gezoomt wird, versagt diese Technik kläglich. Hier ist eine Möglichkeit erforderlich, die diese Bewegungen kompensiert. Das genau leistet MPEG, und das ist auch der wichtigste Unterschied zwischen MPEG und JPEG.

Die MPEG-Ausgabe besteht aus drei Arten von Rahmen:

1. I-Bilder (intracodiert, *I-frame*): in sich geschlossene JPEG-codierte Standbilder
2. P-Bilder (prädiktiv, *P-frame*): blockweiser Unterschied zum letzten Rahmen
3. B-Bilder (bidirektional, *B-frame*): blockweise Unterschiede zwischen dem letzten und dem nächsten Rahmen

I-Bilder sind lediglich Standbilder. Sie können mit JPEG oder etwas Ähnlichem codiert werden. Es ist aus drei Gründen von Vorteil, wenn I-Bilder regelmäßig (d.h. ein- oder zweimal pro Sekunde) im Ausgabestrom erscheinen. Erstens kann MPEG für eine Multicast-Übertragung benutzt werden, in die sich Teilnehmer nach Wunsch ein-klinken. Hängen alle Rahmen von ihren Vorgängern ab, könnten alle, die den ersten Rahmen verpasst haben, die nachfolgenden Rahmen nicht decodieren. Zweitens, wenn ein Rahmen fehlerhaft empfangen würde, wäre keine weitere Decodierung mehr möglich. Drittens müsste für den Fall, dass keine I-Bilder verwendet würden, der Decoder beim schnellen Vor- oder Rücklauf jeden bis dato übertragenen Rahmen berechnen, um den vollen Wert des Rahmens an der Stopposition zu ermitteln.

P-Bilder codieren dagegen Unterschiede zwischen Rahmen. Sie basieren auf dem Konzept von **Makroblöcken**, die beispielsweise 16×16 Pixel im Luminanzraum und 8×8 Pixel im Chrominanzraum abdecken. Ein Makroblock wird dadurch codiert, dass im vorherigen Rahmen nach einem solchen Block oder einem leicht von ihm abweichenden Block gesucht wird.

► Abbildung 7.49 ist ein Beispiel für den sinnvollen Einsatz von P-Bildern. Hier sehen wir drei aufeinanderfolgende Rahmen, die den gleichen Hintergrund haben, aber in Bezug auf die Position der Person abweichen. Die Makroblöcke, die die Hintergrundszene enthalten, stimmen genau überein. Die Makroblöcke mit der kleinen Person müssen in Bezug auf die Position um ein unbekanntes Maß versetzt und somit verfolgt werden.

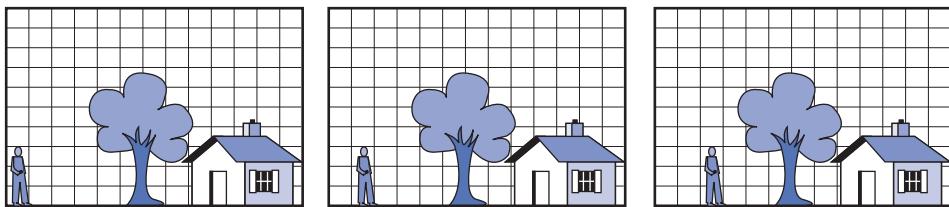


Abbildung 7.49: Drei aufeinanderfolgende Rahmen.

Die MPEG-Standards spezifizieren nicht, wie beziehungsweise wie weit gesucht werden muss oder ab wann eine Übereinstimmung eine Übereinstimmung ist. Das bleibt den einzelnen Implementierungen überlassen. Eine Implementierung sucht z.B. nach einem Makroblock an der aktuellen Position im vorherigen Rahmen und an allen anderen Positionen mit einem Offset von $\pm \Delta x$ in x- und $\pm \Delta y$ in y-Richtung. Für jede Position kann die Anzahl der Übereinstimmungen in der Luminanzmatrix berechnet werden. Die Position mit dem höchsten Wert würde als Gewinner deklariert, vorausgesetzt, sie läge oberhalb eines vordefinierten Schwellenwertes; andernfalls würde der Makroblock als fehlend gelten. Selbstverständlich sind viel anspruchsvollere Algorithmen möglich.

Wird ein Makroblock gefunden, dann wird er auf der Grundlage des Unterschieds seines aktuellen Wertes zum vorherigen Rahmen (für die Luminanz- und beide Chrominanzwerte) codiert. Diese Unterschiedsmatrizen werden dann wie gehabt der diskreten Kosinustransformation, der Quantisierung, der Laufängencodierung und der Huffman-Codierung unterzogen. Der für den Makroblock ermittelte Wert im Ausgabestrom ist der Bewegungsvektor (wie weit sich der Makroblock von der vorherigen Position in jede Richtung entfernt hat), gefolgt von der Codierung des Unterschieds. Befindet sich der Makroblock nicht im vorherigen Rahmen, dann wird der aktuelle Wert wie beim I-Bild codiert.

Daraus wird deutlich, dass dieser Algorithmus höchst asymmetrisch ist. Einer Implementierung steht es frei, jede plausible Position im vorherigen Rahmen auszuprobiieren, und zwar in dem verzweifelten Versuch, sämtliche Makroblöcke zu finden, wohin sie auch entchwunden sein mögen. Dieser Ansatz minimiert den codierten MPEG-Strom zulasten einer sehr langsamen Codierung. Dies mag für die einmalige Codierung eines Filmarchivs ausreichend sein, wäre aber fatal für Videokonferenzen in Echtzeit.

Ebenso kann jede Implementierung frei entscheiden, was ein „gefunder“ Makroblock ist. Durch diese Freiheit können die Implementierer bezüglich Qualität und Geschwindigkeit im Wettbewerb liegen, erzeugen aber immer eine kompatible MPEG-Ausgabe.

Soweit ist die Decodierung von MPEG unkompliziert. Die Decodierung von I-Bildern ist identisch mit der Decodierung von JPEG-Bildern. Die Decodierung von P-Bildern setzt voraus, dass der Decoder die vorherigen Rahmen zwischenspeichert, um dann den neuen Rahmen in einem zweiten Puffer aufzubauen, und zwar auf der Grundlage voll codierter Makroblöcke sowie solcher, die Unterschiede zu den vorherigen Rahmen enthalten. Der neue Rahmen wird Makroblock für Makroblock zusammengesetzt.

B-Bilder sind mit P-Bildern vergleichbar, außer dass bei ihnen der Referenzmakroblock in einem der vorherigen oder nachfolgenden Rahmen sein kann. Diese zusätzliche Freiheit ermöglicht eine verbesserte Bewegungskompensation. Dies ist beispielsweise nützlich, wenn sich Objekte vor oder hinter anderen Objekten bewegen. Für die Codierung von B-Bildern muss der Codierer eine Folge von Bildern gleichzeitig im Speicher halten: die vorherigen, den aktuell codierten und die zukünftigen Rahmen. Die Decodierung ist entsprechend ebenfalls komplizierter und verantwortlich für etwas Verzögerung. Dies liegt daran, dass ein gegebenes B-Bild erst decodiert werden kann, wenn die aufeinanderfolgenden Rahmen, von denen das B-Bild abhängt, decodiert sind. Also auch, wenn B-Bilder die beste Komprimierung ermöglichen, werden sie aufgrund der größeren Komplexität und Pufferanforderungen nicht immer verwendet.

Die MPEG-Standards enthalten viele Verbesserungen an diesen Techniken, um hohe Komprimierungsraten zu erzielen. Mithilfe von AVC können Videos im Verhältnis von 50:1 oder sogar höher komprimiert werden, was die Bandbreitenanforderungen um den gleichen Faktor reduziert. Weitere Informationen zu AVC finden Sie in Sullivan und Wiegand (2005).

7.4.3 Streaming von gespeicherten Medien

Kommen wir nun zu den Netzanwendungen. Als Erstes behandeln wir das Streaming von Medien, die bereits in Dateien gespeichert sind. Das bekannteste Beispiel ist das Betrachten von Videos über das Internet. Dabei handelt es sich um eine Form von **Video-on-Demand (VoD)**. Andere Formen von Video-on-Demand verwenden ein Providernetz, das vom Internet getrennt ist, um die Filme bereitzustellen (z.B. das Kabelnetz).

Im nächsten Abschnitt behandeln wir das Streaming von Live-Medien, z.B. IPTV und Internetradio. Anschließend widmen wir uns dem dritten Fall, der Echtzeitkonferenz. Beispiele dafür sind VoIP-Anrufe oder Videokonferenzen über Skype. Diese drei Fälle stellen zunehmend strengere Anforderungen an die Art und Weise, wie wir Audio und Video über das Netz liefern, da Verzögerung und Jitter eine immer stärkere Rolle spielen und beachtet werden müssen.

Das Internet ist voller Musik- und Video-Websites, die gespeicherte Mediendateien streamen. Eigentlich ist der einfachste Weg zur Behandlung von gespeicherten Medien, sie *nicht* zu streamen. Angenommen, Sie wollten als Konkurrenz zu Apples iTunes eine Website für den Online-Filmverleih aufbauen. Eine normale Website erlaubt Benutzern, Videos herunterzuladen und zu betrachten (natürlich nachdem sie bezahlt haben). Die Folge der Schritte ist in ► Abbildung 7.50 zu sehen und wird am nachfolgenden Beispiel erläutert.

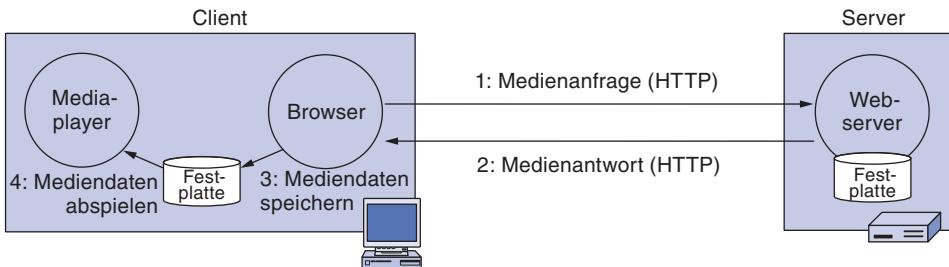


Abbildung 7.50: Mittels einfacher Downloads Medien über das Web abspielen.

Der Browser wird aktiv, wenn der Benutzer einen Film anklickt. In Schritt 1 sendet er eine HTTP-Anfrage für den Film an den Webserver, mit dem der Film verlinkt ist. In Schritt 2 ruft der Server den Film ab (der nur eine Datei im MP4- oder einem anderen Format ist) und sendet ihn zurück an den Browser. Anhand des MIME-Typs (z.B. `video/mp4`) entscheidet der Browser, wie die Datei angezeigt bzw. wiedergegeben werden soll. In diesem Fall ist es ein Mediaplayer, der als Hilfsprogramm angezeigt wird, obwohl es auch genauso gut ein Plug-in sein könnte. Der Browser speichert den gesamten Film in eine Arbeitsdatei auf Festplatte in Schritt 3. Dann ruft er den Mediaplayer auf und übergibt ihm den Namen der Arbeitsdatei. Zum Schluss, in Schritt 4, startet der Mediaplayer, liest die Datei ein und spielt den Film ab.

Dieser Ansatz ist im Grunde völlig korrekt; der Film wird abgespielt. Auch gibt es keine Schwierigkeiten mit Echtzeit im Netz, da lediglich eine Datei heruntergeladen wird. Das einzige Problem ist, dass der gesamte Film über das Netz übertragen werden muss, bevor mit dem Abspielen begonnen werden kann. Die meisten Kunden wollen jedoch nicht eine Stunde auf ihr „Video-on-Demand“ warten. Dieses Modell kann sogar für Audio problematisch sein. Stellen Sie sich vor, dass Sie ein Musikstück anhören wollen, bevor Sie ein Album kaufen. Wenn das Musikstück 4 MB groß ist (was für MP3-Lieder durchaus üblich ist) und die Breitbandkonnektivität 1 Mbit/s beträgt, dann wird der Benutzer erst einmal von einer halben Minute Stille begrüßt, bevor die Hörprobe beginnt. Mit diesem Modell werden sicherlich nicht viele Alben verkauft.

Um dieses Problem zu vermeiden, ohne die Arbeitsweise des Browsers zu ändern, können Websites das Design in ► Abbildung 7.53 verwenden. Die Seite, die mit dem Film verlinkt ist, ist nicht die eigentliche Filmdatei. Es handelt sich vielmehr um eine sogenannte Metadatei, eine sehr kurze Datei, die nur den Film angibt (und eventuell weitere Schlüsseldeskriptoren).

Eine typische Metadatei kann aus nur einer Zeile ASCII-Text bestehen und wie folgt aussehen:

```
rtsp://joes-video-server/video-0025.mp3
```

Der Browser erhält die Seite, jetzt eine einzeilige Datei, in Schritt 1 und 2. Dann startet er den Mediaplayer und übergibt wie gewohnt den Namen der einzeiligen Datei in Schritt 3. Der Mediaplayer liest die Metadatei und sieht die URL, wo der Film zu finden ist. Er kontaktiert *joes-video-server* und fragt nach dem Film in Schritt 4. Dann

wird der Film in Schritt 5 an den Medioplayer zurück gestreamt. Der Vorteil dieser Vorgehensweise ist, dass der Medioplayer fast direkt startet, denn es muss nur eine sehr kurze Metadatei heruntergeladen werden. Sobald dies geschieht, befindet sich der Browser nicht mehr in der Schleife. Die Mediendatei wird direkt an den Medioplayer geschickt, der mit dem Abspielen des Films beginnen kann, bevor die ganze Datei heruntergeladen wurde.

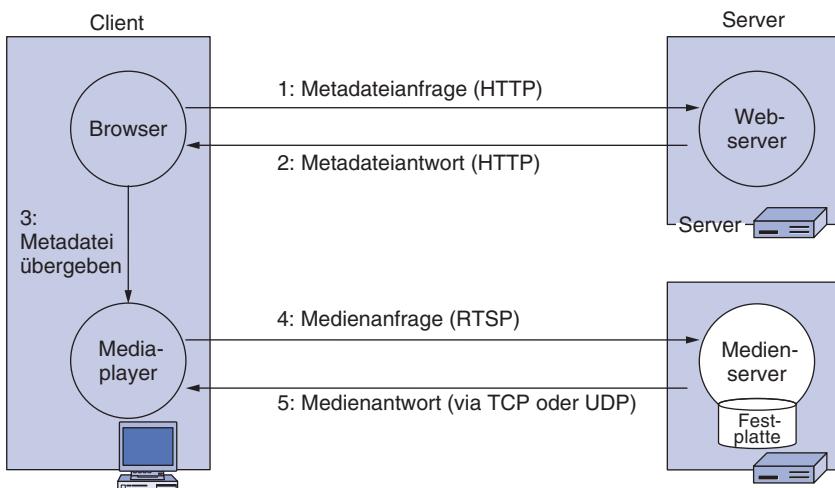


Abbildung 7.51: Mediendateien mit dem Web und einem Medienserver streamen.

Wir haben in ► Abbildung 7.51 zwei Server gezeigt, da der Server, der in der Metadatei steht, oft nicht der gleiche ist wie der Webserver. Im Allgemeinen ist es oft nicht einmal ein HTTP-Server, wenn auch ein spezialisierter Medienserver. In diesem Beispiel verwendet der Medienserver **RTSP** (*Real Time Streaming Protocol*, Echtzeit-Streaming-Protokoll).

Der Medioplayer muss vier Hauptaufgaben ausführen:

1. Verwaltung der Benutzerschnittstelle
2. Handhabung von Übertragungsfehlern
3. Dekomprimieren des Inhalts
4. Eliminierung von Jitter

Die meisten Medioplayer haben heutzutage eine glitzernde Oberfläche, die manchmal wie ein Stereoerstärker aussieht, mit Tasten, Knöpfen, Schiebereglern und grafischen Anzeigen. Sie verfügen häufig über austauschbare Fronten, sogenannte **Skins**, die der Benutzer für den Player auswählen kann. Der Medioplayer muss dies verwalten und mit dem Benutzer interagieren.

Die anderen Aufgaben sind verwandt und hängen von den Netzwerkprotokollen ab. Wir werden sie einzeln besprechen und beginnen mit der Behandlung von Übertragungsfehlern. Die Fehlerbehandlung hängt davon ab, ob für den Medientransport ein TCP-basierter

Transport wie HTTP oder ein UDP-basierter Transport wie RTP verwendet wird. Beide werden in der Praxis eingesetzt. Wenn ein TCP-basierter Transport verwendet wird, gibt es für den Medioplayer keine Fehler zu korrigieren, da TCP bereits durch die erneute Übertragung die nötige Zuverlässigkeit bietet. Diese Art Fehler zu behandeln, ist zumindest für den Medioplayer einfach, aber sie kompliziert später die Beseitigung von Jitter.

Alternativ kann auch ein UDP-basierter Transport wie RTP verwendet werden, um die Daten zu übertragen. Wir haben dieses Thema in Kapitel 6 bereits besprochen. Bei diesen Protokollen gibt es keine Neuübertragungen. Das bedeutet jedoch, dass Paketverlust aufgrund von Überlastung oder Übertragungsfehler dazu führt, dass nicht alle Mediendaten ankommen. Für dieses Problem ist der Medioplayer zuständig.

Lassen Sie uns versuchen zu verstehen, mit welchen Schwierigkeiten wir es zu tun haben. Der Verlust stellt ein Problem dar, weil Kunden Lücken in ihren Songs und Filmen nicht besonders schätzen. Das Problem ist jedoch nicht so groß wie bei der normalen Dateiübertragung, da der Verlust von ein paar Mediendaten die Präsentation für den Benutzer nicht unbedingt beeinträchtigen muss. Der Betrachter eines Videos wird kaum merken, ob in einer Sekunde gelegentlich 24 neue Rahmen statt 25 neuer Rahmen übertragen werden. Bei Audio können kurze Lücken beim Abspielen mit zeitnahen Tönen maskiert werden. Die Hörer werden diese Ersetzung kaum feststellen, es sei denn, sie horchen ganz genau hin.

Die Grundvoraussetzung der obigen Schlussfolgerung ist, dass die Lücken sehr kurz sind. Netzüberlastung oder Übertragungsfehler verursachen oft den Verlust eines ganzen Pakets, und Pakete gehen oft in kleinen Datenschüben verloren. Es gibt zwei Strategien, um die Auswirkung von Paketverlust auf die verlorenen Medien zu reduzieren: FEC und Verschachtelung (*interleaving*). Wir werden beide nacheinander besprechen.

FEC (*Forward Error Correction, Vorwärtsfehlerkorrektur*) ist einfach die bereits in *Kapitel 3* untersuchte Fehlerkorrekturcodierung, angewendet auf Anwendungsebene. Parität unter Paketen ist ein Beispiel (Shacham und McKenny, 1990). Für jeweils vier Datenpakete, die gesendet werden, kann ein fünftes **Paritätspaket** erstellt und mitgesendet werden. Dies wird in ► Abbildung 7.52 anhand der Pakete A, B, C und D gezeigt. Das Paritätspaket P enthält redundante Bits, die die Paritäts- oder Exklusiv-ODER-Summen der Bits in jedem der vier Datenpakete bilden. Im Idealfall kommen von den meisten Gruppen der fünf Pakete alle Pakete an. In diesem Fall wird das Paritätspaket einfach beim Empfänger entsorgt. Oder wenn nur das Paritätspaket verloren geht, resultiert kein Schaden daraus.

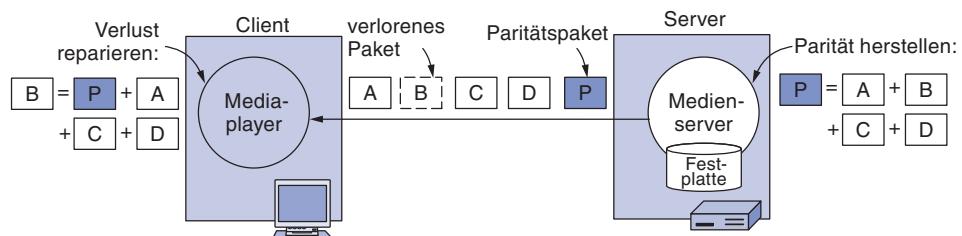


Abbildung 7.52: Mit einem Paritätspaket einen Verlust reparieren.

Gelegentlich kann jedoch ein Paket während der Übertragung verloren gehen, wie dies bei unserem Paket b in Abbildung 7.52 der Fall ist. Der Medioplayer empfängt nur die drei Datenpakete A , C und D sowie das Paritätspaket P . Gemäß dem Design können die Bits in dem fehlenden Datenpaket aus dem Paritätsbits rekonstruiert werden. Um genau zu sein, kann unter Verwendung von „+“ für die Exklusiv-ODER- oder Modulo-2-Addition B als $B = P + A + C + D$ rekonstruiert werden, und zwar durch die Eigenschaften des Exklusiv-ODER (d.h. $X + Y + Y = X$).

FEC kann die Höhe des Verlustes beim Medioplayer verringern, indem einige der Paketverluste repariert werden, was aber nur bis zu einem gewissen Grad gelingt. Wenn zwei Pakete von fünf verloren gehen, haben wir keine Möglichkeit mehr, die Daten zu rekonstruieren. Die andere Eigenschaft von FEC, auf die wir Sie hinweisen möchten, betrifft die Kosten, die für einen solchen Schutz aufzuwenden sind. Aus jeweils vier Paketen sind fünf Pakete geworden, sodass die Bandbreitenanforderungen für die Medien um 25 % gestiegen sind. Die Latenz des Decodierens ist ebenfalls gestiegen, da wir eventuell warten müssen, bis das Paritätspaket angekommen ist, bevor wir ein vorheriges Datenpaket rekonstruieren können.

Außerdem gibt es in der obigen Technik einen schlauen Trick. In Kapitel 3 haben wir behauptet, dass Parität der Fehlererkennung dient. Hier setzen wir sie für die Fehlerkorrektur ein. Wie kann sie beides leisten? Die Antwort ist, dass wir in diesem Fall wissen, welches Paket verloren ist. Die verlorenen Daten werden als **Lösung** (*erasure*) bezeichnet. Als wir in Kapitel 3 einen Rahmen betrachtet haben, der mit einigen fehlerhaften Bits empfangen wurde, wussten wir nicht, welche Bits fehlerbehaftet waren. Dieser Fall ist schwieriger als Lösungen. Bei Lösungen kann Parität Fehlerkorrektur bieten, ohne Lösungen jedoch nur Fehlererkennung. Wir werden bald, im Zusammenhang mit Multicast-Szenarien, einen weiteren unerwarteten Vorteil der Parität kennenlernen.

Die zweite Strategie, die wir vorstellen wollen, ist die sogenannte **Verschachtelungsstrategie** (*interleaving*). Dieser Ansatz basiert darauf, die Reihenfolge der Mediendaten vor der Übertragung durcheinanderzubringen oder zu verschachteln. Wenn dann ein Paket (oder ein Schub von Paketen) verloren gegangen ist, wird auf diese Weise der Verlust über die Zeit verteilt, indem die Verschachtelung wieder rückgängig gemacht wird. Es besteht also nicht die Gefahr, dass beim Abspielen eine einzige große Lücke entsteht. So kann ein Paket beispielsweise 220 Stereo-Samples enthalten, von denen jedes ein Paar 16-Bit-Werte enthält, die in der Regel für 5 ms Musik ausreichen. Wären die Samples der Reihenfolge nach gesendet worden, würde ein verlorenes Paket einen Ausfall der Musik von 5 ms bedeuten. Stattdessen werden die Samples wie in Abbildung 7.53 übertragen. Alle geraden Samples eines 10-ms-Intervalls werden in einem Paket gesendet, gefolgt von den ungeraden Samples im folgenden Paket. Der Verlust von Paket 3 bedeutet jetzt nicht eine 5-ms-Unterbrechung der Musik, sondern den Verlust von jedem zweiten Sample eines 10-ms-Intervalls. Dieser Verlust ist leicht zu kompensieren, da der Medioplayer die fehlenden Werte aus den vorhergehenden und nachfolgenden Werten interpolieren kann. Das Ergebnis ist eine schlechtere zeitliche Auflösung für 10 ms, aber keine auffällige Lücke in der Medienwiedergabe.

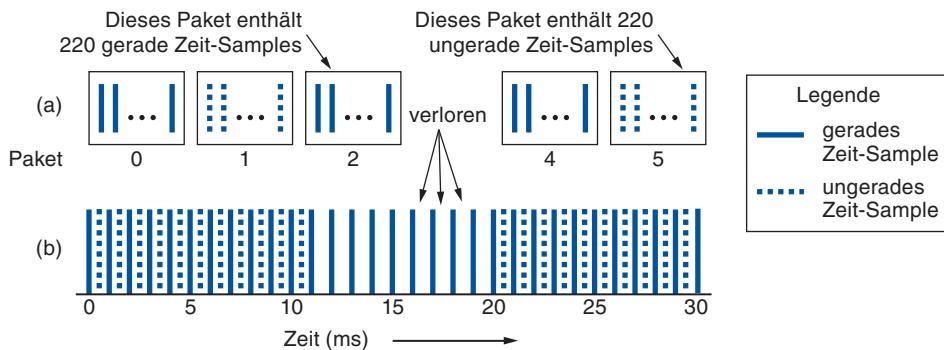


Abbildung 7.53: Wenn Pakete alternierende Samples enthalten, senkt der Verlust eines Pakets die zeitliche Auflösung, führt aber zu keiner Unterbrechung.

Diese Strategie funktioniert nur mit nicht komprimierten Samples. Die Verschachtelung kann jedoch auch nach der Komprimierung angewendet werden (über einen kurzen Zeitraum, keine einzelnen Samples), solange es möglich ist, im komprimierten Strom Samplegrenzen zu erkennen. RFC 3119 definiert eine Methode, die mit komprimierten Audiodaten funktioniert.

Verschachtelung ist im Einsatzfall eine interessante Technik, da sie im Gegensatz zu FEC keine zusätzliche Bandbreite benötigt. Allerdings trägt die Verschachtelung ebenso wie FEC zur Latenz bei, da auf eine Gruppe von Paketen gewartet werden muss, um die Verschachtelung rückgängig zu machen.

Die dritte Aufgabe des Mediaplayers ist die Dekomprimierung des Inhalts. Obwohl rechenintensiv, ist diese Aufgabe relativ unkompliziert. Die heikle Frage dabei ist, wie Mediendaten zu dekomprimieren sind, wenn das Netzprotokoll die Übertragungsfehler nicht korrigiert hat. Bei vielen Komprimierungsmethoden können die späteren Daten erst dekomprimiert werden, wenn die Daten davor dekomprimiert worden sind, da die späteren Daten in Relation zu den früheren Daten codiert werden. Beim UDP-basierten Transport kann es zu Paketverlust kommen. Deshalb muss der Codierprozess so ausgelegt sein, dass er das Decodieren trotz Paketverlust erlaubt. Dies ist genau der Grund, warum MPEG I-, P- und B-Bilder verwendet. Jedes I-Bild kann unabhängig von den anderen Rahmen decodiert werden, um den Verlust irgendwelcher früheren Rahmen wettzumachen.

Die vierte Aufgabe ist die Eliminierung von Jitter, die Krux aller Echtzeitsysteme. Die allgemeine Lösung, die wir in Abschnitt 6.4.3 beschrieben haben, ist die Verwendung eines Abspielpuffers. Alle Streaming-Systeme puffern zuerst die Mediendaten für 5–10 Sekunden, bevor sie mit der Wiedergabe beginnen, wie in ▶ Abbildung 7.54 zu sehen. Durch das Abspielen werden die Mediendaten regelmäßig aus dem Puffer entfernt, sodass Sie eine klare Audio- und eine reibungslose Videowiedergabe erhalten. Die Anfangsverzögerung bietet die Möglichkeit, den Puffer bis zur **Untergrenze (low-water mark)** zu füllen. Die Idee dahinter ist, dass die Daten so regelmäßig gesendet werden, dass der Puffer niemals völlig leer ist. Wenn dies passieren würde, käme es zu einem Stoppen der Medienwiedergabe. Der Vorteil eines Puffers liegt darin, dass, wenn die Daten

staubedingt manchmal langsam ankommen, die zwischengespeicherten Daten das Weiterabspielen gewährleisten, bis neue Mediendaten ankommen und der Puffer wieder gefüllt wird.

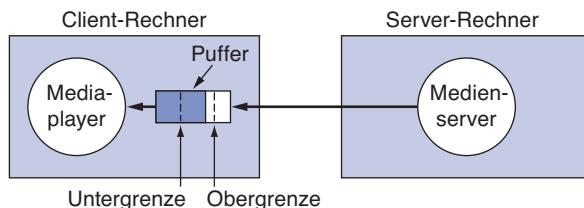


Abbildung 7.54: Der Medioplayer puffert die Eingaben vom Medienserver und spielt die Daten aus dem Puffer ab statt direkt aus dem Netz.

Wie viel Puffer benötigt wird und wie schnell der Medienserver die Daten sendet, um den Puffer zu füllen, hängt von den Netzprotokollen ab. Hier gibt es viele Möglichkeiten. Der größte Faktor im Entwurf ist, ob der verwendete Transport auf UDP oder TCP basiert.

Angenommen, es wird ein UDP-basierter Transport wie RTP verwendet, und weiterhin angenommen, es steht bei vernachlässigbarem Netzverkehr ausreichend Bandbreite zur Verfügung, um Pakete fast verlustfrei vom Medienserver zum Medioplayer zu schicken. In diesem Fall können die Pakete in genau der Geschwindigkeit gesendet werden, in der sie auch abgespielt werden. Jedes Paket durchquert das Netz und kommt nach einer Ausbreitungsverzögerung fast genau zu der Zeit im Medioplayer an, an der es wiedergegeben werden soll. Sie benötigen also nur einen kleinen Puffer, da es bei der Verzögerung nicht zu Schwankungen kommt. Wird mit Verschachtelung oder Vorwärtsfehlerkorrektur gearbeitet, muss der Puffer etwas größer ausfallen und sollte zumindest die Gruppe der Pakete umfassen, über die eine Verschachtelung oder Vorwärtsfehlerkorrektur ausgeführt wird. Allerdings ist die Pufferzunahme nur geringfügig.

Leider ist dieses Szenario aus zwei Gründen unrealistisch. Zum einen schwankt die Bandbreite über die Netzpfade, sodass der Medienserver vor dem Streamen der Daten normalerweise nicht weiß, ob genügend Bandbreite zur Verfügung steht. Dieses Problem lässt sich einfach dadurch lösen, dass Sie es dem Benutzer überlassen, die für seinen Internetanschluss günstigste Auflösung zu wählen. Oft gibt es nur zwei Stufen: hohe Qualität, die sagen wir mit 1,5 Mbit/s codiert wird, und niedrige Qualität, die mit 512 kbit/s oder weniger codiert wird.

Zweitens kommt es im Netz zu Jitter beziehungsweise Schwankungen in der Übertragungszeit der Medien-Samples. Oft liegt dies an einem hohen Verkehrsaufkommen im Netz, das nicht zuletzt von den Benutzern selbst verursacht wird, die im Internet surfen, während sie gleichzeitig einen gestreamten Film schauen. Dieser Verkehr ist der Grund für Fluktuationen beim Empfang der Mediendaten. Außerdem sind wir eher an der Ankunft von Videorahmen und Audio-Samples interessiert als an Paketen. Komprimiert können vor allem Videorahmen je nach ihrem Inhalt größer oder kleiner sein. Für die Codierung einer Action-Sequenz werden normalerweise mehr Bits benötigt als für die Codierung einer ruhigen Landschaft. Ist die Netzbänder konstant, variiert

die Rate der Medienzustellung über die Zeit. Je höher der Jitter oder die Verzögerungsschwankung von diesen Quellen, desto größer muss die Untergrenze des Puffers sein, um ein Leerlauf des Puffers zu vermeiden.

Gehen wir im Folgenden davon aus, dass ein TCP-basierter Transport wie HTTP zum Senden der Medien verwendet wird. Durch Neuübertragungen und Zurückhalten der Pakete, bis sie in der richtigen Reihenfolge sind, erhöht TCP den Jitter, der sich beim Medioplayer (vielleicht sogar erheblich) bemerkbar macht. Als Folge werden ein größerer Puffer und eine höhere Untergrenze benötigt. TCP hat jedoch auch einen Vorteil, denn es sendet die Daten so schnell, wie das Netz sie überträgt. Im Falle eines Verlustes können sich Mediendaten manchmal verzögern. Die meiste Zeit jedoch ist das Netz in der Lage, die Daten schneller zuzustellen, als der Player sie abspielen kann. In dieser Zeit kann der Puffer gefüllt und zukünftiger Pufferleerlauf verhindert werden. Wenn das Netz wesentlich schneller ist als die durchschnittliche Übertragungsrate – was oft der Fall ist –, dann wird sich der Puffer nach dem Starten schnell füllen, sodass ein leerer Puffer bald kein Problem mehr darstellt.

Bei TCP oder bei UDP und einer Übertragungsrate, die die Abspielrate übersteigt, stellt sich die Frage, wie weit hinter dem Abspielpunkt Medioplayer und Medienserver bereit sind, Daten bereitzustellen. Oft laden sie erst die ganze Datei herunter.

Doch durch das Bereitstellen der Daten weit über den Abspielpunkt hinaus wird Arbeit geleistet, die noch nicht benötigt wird, unter Umständen beträchtlichen Speicher belegt und nicht notwendig ist, um einen Pufferleerlauf zu vermeiden. Wenn dies nicht gewünscht wird, kann der Medioplayer als Lösung eine **Obergrenze** (*high-water mark*) im Puffer definieren. Der Server sendet im Grunde nur Daten, bis der Puffer bis zur Obergrenze voll ist. Der Medioplayer weist ihn dann an zu pausieren. Da die Daten weiterhin eintreffen, bis der Server die Pausenanforderung erhalten hat, muss der Abstand zwischen der Obergrenze und dem Ende des Puffers größer sein als das Produkt aus Bandbreite und Übertragungsverzögerung des Netzes. Nachdem der Server angehalten hat, beginnt sich der Puffer zu leeren. Wird die Untergrenze erreicht, weist der Medioplayer den Server an, wieder mit dem Senden zu beginnen. Die Untergrenze muss so gewählt werden, dass der Puffer nicht leer läuft (*buffer underrun*), d.h., auch hier muss das Bandbreite-Verzögerung-Produkt des Netzes berücksichtigt werden, wenn der Medienserver aufgefordert wird, mit dem Senden der Daten fortzufahren.

Für das Starten und Anhalten des Datenflusses muss der Medioplayer über eine Fernsteuerung verfügen. Diese wird über RTSP bereitgestellt. RTSP ist in RFC 2326 definiert und enthält einen Mechanismus, mit dem der Player den Server steuert. Dieser Mechanismus kann nicht nur den Datenstrom starten und stoppen, sondern er kann auch vorwärts und rückwärts an eine Position springen, angegebene Intervalle abspielen und mit schnellerer oder langsamer Geschwindigkeit abspielen. Er ist jedoch nicht für den Datenstrom verantwortlich, was normalerweise die Aufgabe von RTP über UDP oder RTP über HTTP über TCP ist.

Die wichtigsten Befehle für RTSP sind in ► Abbildung 7.55 aufgeführt. Sie weisen ein einfaches Textformat auf, wie es auch in HTTP-Nachrichten zu finden ist, und werden meist

über TCP übertragen. RTSP kann auch über UDP ausgeführt werden, da jeder Befehl bestätigt wird (und so erneut gesendet werden kann, wenn er nicht bestätigt wurde).

| Befehl | Server-Aktion |
|----------|--|
| DESCRIBE | Listet Medienparameter auf |
| SETUP | Errichtet logischen Kanal zwischen dem Player und dem Server |
| PLAY | Beginnt, Daten an den Client zu senden |
| RECORD | Beginnt, Daten vom Client zu akzeptieren |
| PAUSE | Unterbricht temporär das Senden der Daten |
| TEARDOWN | Gibt den logischen Kanal frei |

Abbildung 7.55: RTSP-Befehle vom Player zum Server.

Auch wenn sich TCP auf den ersten Blick nur schlecht für Echtzeitverkehr eignet, wird es in der Praxis oft verwendet. Der Hauptgrund ist, dass es weniger oft durch Firewalls blockiert wird als UDP, besonders wenn es auf dem HTTP-Port läuft. Die meisten Administratoren konfigurieren Firewalls so, dass ihre Netze vor unliebsamen Besuchern geschützt sind. Fast immer lassen diese die TCP-Verbindungen auf dem entfernten Port 80 für den HTTP- und Webverkehr durch. Das Blockieren dieses Ports ist keine gute Idee. Die meisten anderen Ports, einschließlich die für RSTP und RTP (Port 554 und 5004), werden jedoch blockiert. Um sicherzustellen, dass die gestreamten Mediendaten die Firewall überwinden, ist es für eine Website am einfachsten, sich zumindest vor der Firewall als HTTP-Server auszugeben, der eine normale HTTP-Antwort sendet.

Darüber hinaus bietet TCP noch weitere Vorteile. Um seinem Ruf eines zuverlässigen Protokolls gerecht zu werden, liefert TCP dem Client eine vollständige Kopie der Daten. Dadurch kann ein Benutzer leichter an einen bereits betrachteten Abspielpunkt zurückspringen, ohne sich Gedanken um den Verlust von Daten machen zu müssen. Außerdem puffert TCP in möglichst kurzer Zeit möglichst viele Daten. Wenn Puffer billig ist (was der Fall ist, wenn die Festplatte als Speichermedium verwendet wird), kann der Medioplayer den Film herunterladen, während der Benutzer ihn bereits anschaut. Wenn der Download abgeschlossen ist, kann der Benutzer den Film ohne weitere Unterbrechungen anschauen, auch wenn die Internetverbindung abreißen sollte. Diese Eigenschaft ist besonders für mobile Geräte hilfreich, da sich die Konnektivität mit einem Ortswechsel schnell ändern kann.

Der Nachteil von TCP ist die zusätzliche Startlatenz (aufgrund des TCP-Starts) und eine höhere Untergrenze. Dies ist jedoch ein vernachlässigbares Kriterium, solange die Netzbänderbreite wesentlich höher ausfällt als die Medierrate.

7.4.4 Streaming von Live-Medien

Im Web ist nicht nur das Abspielen aufgezeichneter Videos extrem populär. Inzwischen hat auch das Streaming in Echtzeit stark an Popularität gewonnen. Sobald es möglich war, Audio- und Videodaten über das Internet zu streamen, begannen Rundfunk- und Fernsehsender, ihre Inhalte nicht nur über den Äther, sondern auch über das Internet zu übertragen. Kurz darauf begannen Studentensender, ihre Signale über das Internet zu senden, und dann starteten *Studenten* ihre eigenen Internetübertragungen.

Heutzutage nutzen auch Privatpersonen und Firmen aller Größen das Live-Streaming von Audio- und Videodaten. Die sich ständig weiterentwickelnden Technologien und Standards machen diesen Bereich zu einem Zentrum von Innovationen. Die wichtigsten Fernsehstationen nutzen Live-Streaming für ihre Onlinepräsenz. Dies wird als **IPTV** (**IP Television**) bezeichnet. Aber auch Rundfunkstationen wie BBC nutzen das Internet zum Verbreiten ihrer Sendungen. Dies wird als **Internetradio** bezeichnet. Sowohl IPTV als auch Internetradio erreichen mit Ereignissen wie Modeschauen, Fußballweltmeisterschaften und Testspielen live von dem Melbourne Cricket Ground ihre Zuhörer in der ganzen Welt. Live-Streaming über IP wird von den Kabelanbietern für den Aufbau eigener Sendesysteme genutzt. Aber auch für den Betrieb von Web-sites, wie sie jedermann anbieten könnte, seien es Zoos oder Pornosite-Betreiber, wird es zunehmend eingesetzt. Bei dem gegenwärtigen Stand der Technik kann praktisch jeder schnell und mit geringem Aufwand Live-Streaming starten.

Ein Ansatz zum Live-Streaming besteht in der Aufzeichnung von Programmen auf Festplatte. Zuschauer oder Zuhörer können sich dann mit den Serverarchiven verbinden, ein beliebiges Programm auswählen und es zum Betrachten oder Anhören herunterladen. Ein **Podcast** ist eine Serienfolge, die auf diese Weise dem Betrachter verfügbar gemacht wird. Bei zeitlich terminierten Ereignissen ist es auch möglich, den Inhalt direkt nach der Live-Ausstrahlung zu speichern, sodass das Archiv bereits nach einer halben Stunde oder weniger nach dem Live-Feed abrufbar ist.

Dieser Ansatz ist genau der gleiche wie der, den wir für die gerade besprochenen Streaming-Daten verwendet haben. Er ist ganz einfach. Alle Techniken, die wir besprochen haben, lassen sich anwenden, und die Zuschauer können unter allen Programmen in den Archiven wählen.

Ein anderer Ansatz ist die Live-Übertragung über das Internet. Dabei schalten sich Zuschauer, wie beim Fernsehen, in einem laufenden Mediendatenstream zu. Mit den Mediaplayern haben die Benutzer jedoch zusätzlich die Möglichkeit, das Abspielen zu unterbrechen und zurückzuspalten. Die Live-Daten werden unterdessen weiter gestreamt und vom Player gespeichert, bis der Benutzer Zeit hat, sie anzusehen. Aus Sicht des Browsers entspricht dies dem Streamen von gespeicherten Mediendaten. Dem Player ist es egal, ob der Inhalt aus einer Datei kommt oder live gesendet wird, und normalerweise kann er das gar nicht unterscheiden (außer dass es bei einem Live-Stream nicht möglich ist vorzuspalten). In Anbetracht der Ähnlichkeit des Mechanismus, hat vieles von dem, was bereits gesagt wurde, auch in diesem Fall Gültigkeit, auch wenn es einige wichtige Unterschiede gibt.

Wichtig ist vor allem, dass clientseitig die Möglichkeit zum Puffern bestehen muss, um den Jitter zu glätten. Genau genommen, wird für das Live-Streaming oft sogar ein großer Puffer benötigt (unabhängig von der Überlegung, dass der Benutzer das Abspielen unterbricht). Wenn aus einer Datei gestreamt wird, können die Mediendaten mit einer Geschwindigkeit ausgestoßen werden, die größer als die Abspielrate ist. Damit wird schnell ein Puffer aufgebaut, der den Netzjitter kompensiert (und der Player stoppt das Streamen, wenn er keine Daten mehr puffern will). Im Gegensatz dazu werden beim Live-Streaming die Mediendaten immer mit genau der Geschwindigkeit übertragen, mit der sie erzeugt wurden, was der Abspielgeschwindigkeit entspricht. Sie können nicht schneller gesendet werden. In diesem Falle muss der Puffer groß genug sein, um den ganzen Netzjitter aufzufangen. In der Praxis reicht eine Startverzögerung von 10 bis 15 Sekunden, sodass dies kein großes Problem darstellt.

Der andere wichtige Unterschied ist, dass Live-Sendungen gleichzeitig meist Hunderte oder sogar Tausende Zuschauer haben, die den Inhalt alle zur selben Zeit betrachten wollen. Unter diesen Umständen ist Multicasting die naheliegende Lösung beim Live-Streaming. Dies ist beim Streamen von gespeicherten Mediendaten nicht erforderlich, da die Benutzer unterschiedliche Inhalte zu verschiedenen Zeiten streamen. In diesem Fall besteht das Streamen an viele Benutzer aus vielen einzelnen Streaming-Sitzungen, die zufällig zur selben Zeit stattfinden.

Multicast-Streaming funktioniert wie folgt. Der Server sendet jedes Medienpaket einmal mit IP-Multicasting an eine Gruppenadresse. Das Netz liefert jedem Mitglied der Gruppe eine Kopie des Pakets. Alle Clients, die den Stream empfangen möchten, müssen der Gruppe beitreten. Die Clients verwenden hierzu IGMP, anstatt eine RTSP-Nachricht an den Medienserver zu senden. Der Grund dafür ist, dass der Medienserver bereits den Live-Stream sendet (es sei denn, es ist noch kein Benutzer der Gruppe beigetreten). Was fehlt, sind Vorkehrungen, dass der Stream lokal empfangen wird.

Da Multicasting eine One-to-Many-Kommunikation ist (einer an viele), werden die Medien in RTP-Paketen über UDP übertragen. TCP kann nur zwischen einem einzelnen Sender und einem einzelnen Empfänger eingesetzt werden. Da UDP nicht so zuverlässig ist wie TCP, können einige Pakete verloren gehen. Um den Verlust an Mediendaten auf einem annehmbaren Niveau zu halten, können wir wie oben beschrieben Vorwärtsfehlerkorrektur und Verschachtelung verwenden.

Hinsichtlich der Fehlervorwärtskorrektur gibt es eine nützliche Interaktion für das Multicasting, die im Paritätsbeispiel in ► Abbildung 7.56 zu sehen ist. Wenn die Pakete per Multicasting übertragen werden, können verschiedene Clients verschiedene Pakete verlieren. Zum Beispiel hat Client 1 Paket *B* verloren, Client 2 das Paritätspaket *P*, Client 3 Paket *D* und Client 4 hat gar nichts verloren. Obwohl die Clients drei verschiedene Pakete verloren haben, kann in dem Beispiel jeder Client sein jeweilig verlorenes Datenpaket wiederherstellen. Voraussetzung ist jedoch, dass jedem Client nicht mehr als ein Paket abhanden gekommen ist – egal welches –, sodass das fehlende Paket durch die Paritätsberechnung rekonstruiert werden kann. Nonnenmacher et al. (1997) beschreiben, wie sich damit die Zuverlässigkeit erhöhen lässt.

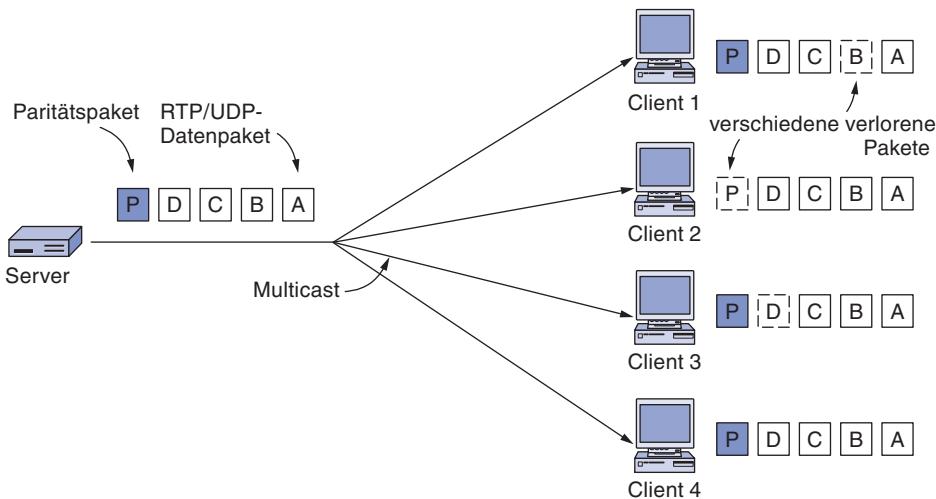


Abbildung 7.56: Multicast-Streaming mit einem Paritätspaket.

Für Server mit einer großen Zahl von Clients ist Multicasting von Mediendaten in RTP- und UDP-Paketen eindeutig die effizienteste Lösung. Im anderen Fall muss der Server N Ströme für N Clients übertragen, was für große Streaming-Ereignisse eine sehr große Netzbandbreite beim Server erfordert.

Es wird Sie vielleicht überraschen, aber die Praxis im Internet sieht anders aus. In der Regel stellt jeder Benutzer eine separate TCP-Verbindung zum Server her, über die die Mediendaten dann gestreamt werden. Für den Client unterscheidet sich dies nicht vom Streamen gespeicherter Mediendaten. Und wie beim Streamen gespeicherter Daten gibt es mehrere Gründe für diese auf den ersten Blick schlechte Wahl.

Der erste Grund ist, dass IP-Multicasting im Internet nicht überall verfügbar ist. Einige ISPs und Netze unterstützen Multicasting zwar intern, aber nicht über Netzgrenzen hinweg, was für das überregionale Streaming unerlässlich ist. Die anderen Gründe sind die bereits genannten Vorteile von TCP gegenüber UDP. Streaming mit TCP erreicht fast jeden Client im Internet, vor allem getarnt als HTTP, um die Firewalls zu überwinden. Und dank der zuverlässigen Medienzustellung können Benutzer leicht zurückspulen.

Es gibt jedoch eine wichtige Situation, in der UDP und Multicasting zum Streamen verwendet werden können: innerhalb eines Providernetzes. Ein Kabelbetreiber könnte sich beispielsweise entscheiden, TV-Kanäle mithilfe von IP-Technologie anstatt herkömmlicher Ausstrahlung an die Set-Top-Boxen der Kunden zu senden. Der Einsatz von IP zur Übertragung von Sendungen wird wie bereits erwähnt allgemein IPTV genannt. Da der Kabelbetreiber vollständige Kontrolle über das eigene Netz hat, kann er dieses so auslegen, dass es IP-Multicasting unterstützt und ausreichend Bandbreite für UDP-basierte Verteilung bereitstellt. Von all diesem bekommt der Kunde nichts mit, da die IP-Technologie innerhalb des sogenannten **Walled Garden** (ummauerter Garten) des Providers existiert. Dem Kunden wird der Eindruck vermittelt, er nutzt Kabelfernsehen, während dahinter IP verborgen ist, wobei die Set-Top-Box als Com-

puter fungiert, auf dem UDP läuft, und der Fernseher lediglich ein Monitor ist, der mit dem Computer verbunden ist.

Doch zurück zum Internet. Der Nachteil von Live-Streaming über TCP ist, dass der Server jedem Client eine eigene Kopie der Mediendaten senden muss. Für eine beschränkte Anzahl an Clients ist dies vertretbar, besonders für Audiodaten. Der Trick ist, den Server an einem Ort mit guter Internetkonnektivität aufzustellen, sodass genügend Bandbreite zur Verfügung steht. Dies bedeutet normalerweise, einen Server in einem Datenzentrum von einem Hosting-Provider zu mieten und nicht einen Server zu Hause zu nutzen, der nur Breitbandanschluss aufweist. Da der Konkurrenzdruck auf dem Hosting-Markt sehr hoch ist, muss dies noch nicht einmal sehr teuer sein.

Im Prinzip kann heutzutage jeder, sogar ein Student, einen Streaming-Medienserver (beispielsweise für einen Internetradiosender) einrichten und betreiben. Die Hauptkomponenten einer solchen Radiostation sind in ► Abbildung 7.57 aufgeführt.

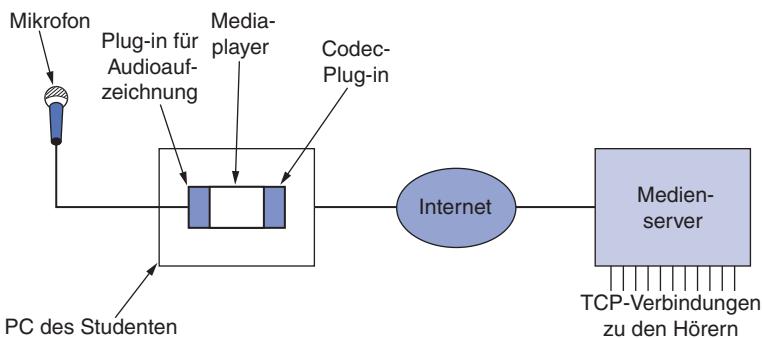


Abbildung 7.57: Ein Studentenradiosender.

Der vom PC empfangene Audiostrom wird dann als Podcast – entweder live oder aus einer gespeicherten Datei – über das Internet an einen Medienserver mit guter Netzkonnektivität übertragen. Der Server ist dafür zuständig, die Mediendaten auf sehr viele TCP-Verbindungen zu verteilen, und präsentiert außerdem eine Front-End-Website mit Seiten zu der Station und Links zu den Inhalten, die zum Streamen zur Verfügung stehen. Für die Verwaltung des Ganzen gibt es sowohl kommerzielle Softwarepakete als auch Open-Source-Pakete wie Icecast.

Im Falle von extrem vielen Clients ist es nicht ratsam, mit TCP von einem einzigen Server aus Mediendaten an jeden Client zu senden. Es gibt einfach nicht genug Bandbreite zu diesem einen Server. Große Streaming-Sites nutzen deshalb zum Streamen mehrere Server, die geografisch verteilt sind, sodass sich ein Client mit dem nächstgelegenen verbinden kann. Es handelt sich hierbei um ein Netz zur Verteilung von Inhalten, das wir am Ende dieses Kapitels näher untersuchen.

7.4.5 Echtzeitkonferenzen

Es gab mal eine Zeit, in der Telefonanrufe über das öffentliche Telefonsystem übertragen wurden und der Netzverkehr hauptsächlich aus der Übertragung von Sprache und nur selten von Daten bestand. Doch dann kamen das Internet und das Web. Der Datenverkehr wuchs und wuchs, bis 1999 genauso viele Daten wie Sprache übertragen wurden (da Sprache inzwischen digitalisiert ist, kann beides in Bit gemessen werden). 2002 war das Volumen des Datenverkehrs bereits eine Größenordnung höher als das Volumen der Sprachübertragung und es wächst immer noch exponentiell weiter, während die Sprachübertragung fast stagniert.

Die Folge dieses Wachstums war, dass das Telefonnetz auf den Kopf gestellt wurde. Sprache wird inzwischen mithilfe von Internettechnologien übertragen und benötigt nur einen winzigen Teil der Netzbандbreite. Diese revolutionäre Technologie wird als **Voice-over-IP** oder auch als **Internettelefonie** bezeichnet.

Voice-over-IP wird in mehreren Formen verwendet, die stark durch wirtschaftliche Faktoren bestimmt werden (auf gut Deutsch: Es spart Geld und wird deshalb von vielen Menschen genutzt). Eine Form des VoIP besteht darin, die Anrufe via normalen (altmodischen?) Telefonen, die sich an das Ethernet anschließen lassen, über das Netz zu senden. Pehr Anderson befand sich am MIT noch im Grundstudium, als er zusammen mit Freunden einen entsprechenden Prototyp als Projektarbeit vorstellte. Sie erhielten damals nur eine mittelmäßige Note. Zutiefst unzufrieden gründete Anderson 1996 seine eigene Firma namens NBX, die dieser Form des Voice-over-IP populär machte. Bereits drei Jahre später konnte er die Firma für 90 Millionen US-Dollar an 3Com verkaufen. Sein Ansatz erfreut sich vor allem bei Firmen und Unternehmen großer Beliebtheit, da sie keine separaten Telefonleitungen mehr benötigen und die bereits vorhandenen Netze nutzen können.

Ein anderer Ansatz besteht darin, mithilfe der IP-Technologie ein Fernnetz aufzubauen. In den Vereinigten Staaten kann auf ein solches Netz beispielsweise über eine bestimmte Vorwahl zugegriffen werden, um günstigere Ferngespräche zu führen. Sprachmuster werden in Pakete verpackt, dann in das Netz geschoben und beim Verlassen des Netzes wieder den Paketen entnommen. Da IP-Anlagen viel billiger als Telekommunikationsanlagen sind, können auch die Dienste billiger angeboten werden.

Nur am Rande sei erwähnt, dass die Preisdifferenz nicht nur technische Ursachen hatte. Jahrzehntelang gab es ein reguliertes Telefonmonopol, das den Telefongesellschaften neben den Kosten eine feste Profitrate garantierte. Das führte dazu, dass die Unternehmen versuchten, ihre Kosten in die Höhe zu treiben, indem sie beispielsweise Unmengen redundanter Hardware anschafften, mit der Begründung, die Zuverlässigkeit zu erhöhen (das Telefonsystem durfte nicht länger als 2 Stunden in 40 Jahren oder 3 Minuten pro Jahr ausfallen). Seit der Deregulierung hat dieser Effekt natürlich abgenommen, aber es gibt immer noch unverzichtbare alte Anlagen. Die IT-Branche hatte nie eine solche Vergangenheit, sie musste immer rationell und effizient sein.

Wir wollen uns hier jedoch auf die VoIP-Form konzentrieren, die für Benutzer am offensichtlichsten ist: von einem Computer aus einen anderen Computer anrufen.

Diese Kommunikationsform verbreitete sich rasch, da PCs zunehmend mit Mikrofon, Lautsprecher und Kamera ausgestattet wurden, die CPU schnell genug war, um Mediendaten zu verarbeiten, und sich den Nutzern in der Zwischenzeit die Möglichkeit bot, von zu Hause aus eine Breitbandinternetverbindung herzustellen. Bekanntestes Beispiel hierfür ist Skype – eine Software, die seit 2003 auf dem Markt ist. Skype und andere Firmen bieten zusätzlich Gateways, die es leichter machen, sowohl Telefonnummern als auch Computer mit IP-Adressen anzurufen.

Mit zunehmender Netzbандbreite kamen zu den Sprachanrufen Videoanrufe hinzu. Am Anfang wurden Videoanrufe primär von Firmen genutzt. Es wurden Videokonferenzsysteme entwickelt, um zwischen zwei oder mehr Orten eine Videoverbindung herzustellen, sodass Führungskräfte, die an unterschiedlichen Firmenstandorten arbeiten, sich während ihrer Sitzungen sehen können. Mit einem guten Breitbandinternetschluss und Videokomprimierungssoftware können auch Privatnutzer per Videokonferenz kommunizieren. Werkzeuge wie Skype, die zuerst nur für Audio ausgelegt waren, bieten inzwischen standardmäßig die Möglichkeit, bei den Anrufen per Video mit Freunden und Familienmitgliedern in der ganzen Welt zu kommunizieren, sodass sie sich nicht nur hören, sondern auch sehen können.

Aus unserer Sicht stellen Audio- oder Videoanrufe über das Internet gleichfalls ein Streaming-Problem dar, wenn auch viel beschränkter als das Streaming einer gespeicherten Datei oder eines Live-Events. Die zusätzliche Beschränkung ist die niedrige Latenz, die für eine bidirektionale Konversation benötigt wird. Das Telefonnetz erlaubt für eine akzeptable Nutzung eine Einweg-Latenz von nicht mehr als 150 ms; alles darüber wird von den Teilnehmern als ärgerliche Verzögerung wahrgenommen. (Internationale Anrufe können eine Latenz von bis zu 400 ms haben, was weit entfernt von einer positiven Benutzererfahrung ist.)

Diese niedrige Latenz ist schwer zu erreichen – mit Sicherheit nicht durch das Puffern von 5–10 Sekunden Mediendaten (wie es bei der Live-Ausstrahlung eines Sportereignisses der Fall wäre). Stattdessen müssen Video- und VoIP-Systeme mit einer Vielzahl von Techniken ausgelegt sein, um die Latenz zu minimieren. Dieses Ziel bedeutet, dass mit UDP als erste Wahl und nicht mit TCP gestartet werden muss, da TCP-Neubertragungen zu einer Verzögerung von mindestens einer Umlaufzeit führen. Einige Arten von Latenz lassen sich jedoch nicht reduzieren, auch nicht mit UDP. Die Entfernung zwischen Seattle und Amsterdam beträgt beispielsweise 8 000 km. Die Ausbreitungsverzögerung bei Lichtgeschwindigkeit für diese Distanz beträgt in Glasfaserkabeln 40 ms. Viel Glück, wenn Sie das schlagen wollen. In der Praxis ist die Ausbreitungsverzögerung durch das Netz größer, da eine längere Strecke zurückzulegen ist (die Bits folgen nicht der Großkreisdistanz) und Übertragungsverzögerungen hinzukommen, weil jeder IP-Router ein Paket speichert und weiterleitet. Diese feste Verzögerung schmälert das akzeptable Verzögerungsbudget.

Eine andere Ursache für Latenz ist mit der Paketgröße verbunden. Normalerweise lässt sich mit großen Paketen die Netzbändbreite am besten nutzen, da sie effizienter sind. Bei einer Audio-Samplingrate von 64 kbit/s würde es jedoch 125 ms dauern, um ein 1 KB-Paket zu füllen (und sogar noch länger, wenn die Samples komprimiert werden).

Diese Verzögerung würde den Großteil des gesamten Verzögerungsbudgets aufbrauchen. Hinzu kommt, dass die Übertragung eines 1-KB-Pakets 8 ms dauert, wenn es über eine Breitbandverbindung mit einer Rate von nur 1 Mbit/s gesendet wird. Addieren Sie dazu weitere 8 ms, wenn das Paket die Breitbandverbindung am anderen Ende durchläuft. Dies zeigt, dass große Pakete eindeutig nicht geeignet sind.

Stattdessen verwenden VoIP-Systeme kleine Pakete, um die Latenz auf Kosten der Bandbreiteneffizienz zu reduzieren. Sie verarbeiten Audio-Samples in kleineren Einheiten, normalerweise von 20 ms. Bei 64 kbit/s entspricht dies 160 Byte an Daten, mit Komprimierung weniger. Per Definition beträgt die Verzögerung von dieser Paketierung 20 ms. Die Übertragungsverzögerung verkürzt sich ebenfalls, da das Paket kleiner ist. In unserem Beispiel würde sie auf ungefähr 1 ms reduziert. Durch die Verwendung kurzer Pakete kann die Einweg-Verzögerung für ein Seattle-Amsterdam-Paket von unakzeptablen 181 ms ($40+125+16$) auf akzeptable 62 ms ($40+20+2$) gedrückt werden.

Und wir haben noch nicht über den Software-Overhead gesprochen, der ebenfalls das Verzögerungsbudget mindert. Dies gilt vor allem für Videodaten, die zur Anpassung an die verfügbare Bandbreite normalerweise einer Komprimierung bedürfen. Im Gegensatz zum Streamen einer gespeicherten Datei ist keine Zeit, einen rechenintensiven Codierer für hohe Komprimierungen einzusetzen.

Außerdem wird immer noch Puffer benötigt, um die Medien-Samples rechtzeitig abzuspielen (um unverständliches Audio oder ruckeliges Video zu vermeiden). Allerdings sollte die Größe des Puffers sehr klein gehalten werden, da die übrig gebliebene Zeit in unserem Verzögerungsbudget in Millisekunden gemessen wird.

Wenn ein Paket zu lang braucht und nicht ankommt, überspringt der Player einfach die fehlenden Samples; vielleicht spielt er benachbarte Geräusche ab oder wiederholt einen Rahmen, um den Verlust vor dem Benutzer zu verborgen. Es besteht eine Beziehung zwischen der Größe des Puffers zur Glättung des Jitters und der Menge an verloren gegangenen Daten. Ein kleinerer Puffer reduziert die Latenz, erhöht jedoch den jitterbedingten Verlust. Wenn der Puffer schrumpft, wird dem Benutzer der Datenverlust irgendwann auffallen.

Aufmerksame Leser werden vielleicht bemerkt haben, dass wir in diesem Abschnitt bisher noch nicht auf die Protokolle der Vermittlungsschicht eingegangen sind. Das Netz kann Latenz, zumindest aber Jitter, durch Dienstgütemechanismen reduzieren. Der Grund, dass dieses Thema noch nicht angesprochen wurde, ist, dass Streaming auch mit erheblicher Latenz funktioniert, sogar im Falle von Live-Streaming. Wenn Latenz nicht das Problem ist, reicht ein Puffer beim Zielrechner, um das Jitterproblem in den Griff zu bekommen. Für Echtzeitkonferenzen hingegen ist es normalerweise wichtig, dass das Netz die Verzögerung und den Jitter reduziert, damit das Verzögerungsbudget eingehalten wird. Die einzige Zeit, in der Jitter und Verzögerung nicht wichtig sind, ist, wenn es genügend Bandbreite gibt, sodass jeder einen guten Dienst erhält.

In Kapitel 5 haben wir zwei Dienstgütemechanismen beschrieben, die Ihnen helfen, dieses Ziel zu erreichen. Einer dieser Mechanismen sind die differenzierten Dienste (*differentiated services*, DS), bei denen die Pakete als Teil bestimmter Klassen markiert

werden, die im Netz jeweils unterschiedlich behandelt werden. Die entsprechende Markierung für VoIP-Pakete ist geringe Verzögerung. In der Praxis setzen Systeme den DS-Codepoint auf den bekannten Wert der Klasse *Expedited Forwarding* mit dem Diensttyp *Low Delay*. Dies ist besonders für Breitbandverbindungen nützlich, da diese zur Überlastung neigen, wenn Webverkehr oder anderer Verkehr um die Nutzung der Verbindung konkurrieren. Unter der Voraussetzung eines stabilen Netzpfades nehmen Verzögerung und Jitter durch die Überlastung zu. Jedes 1-KB-Paket benötigt auf einer 1-Mbit/s-Verbindung 8 ms und ein VoIP-Paket wird diese Verzögerungen übernehmen, wenn es in der Warteschlange dahinter steht. Mit der Markierung *Low Delay* springen die VoIP-Pakete an den Kopf der Warteschlange, d.h., sie umgehen damit die Web-pakete und reduzieren ihre Verzögerung.

Der zweite Mechanismus zur Verringerung der Verzögerung besteht darin sicherzustellen, dass genügend Bandbreite zur Verfügung steht. Wenn die verfügbare Bandbreite variiert oder die Übertragungsrate schwankt (wie beim komprimierten Video), sodass manchmal nicht genügend Bandbreite zur Verfügung steht, bilden sich Warteschlangen, was die Verzögerung erhöht. Dies passiert auch bei den differenzierten Diensten. Um sicherzustellen, dass ausreichend Bandbreite zur Verfügung steht, kann beim Netz eine Reservierung gemacht werden. Dies wird durch die integrierten Dienste gewährleistet. Leider wird davon nicht häufig Gebrauch gemacht. Stattdessen werden Netze auf einen zu erwartenden Verkehr ausgelegt oder Netzkunden gehen Dienstgütevereinbarungen für eine bestimmte Verkehrsichte ein. Um Überlastung und damit verbundene unnötige Verzögerungen zu vermeiden, müssen sich Anwendungen unterhalb dieses Wertes bewegen. Für gelegentliche Videokonferenzen von zu Hause kann der Benutzer eine Videoqualität quasi stellvertretend für den Bandbreitenbedarf wählen oder aber die Software testet den Netzpfad und wählt automatisch eine passende Qualität.

Alle obigen Faktoren können eine inakzeptable Latenz verursachen, sodass bei Echtzeitkonferenzen auch alle berücksichtigt werden müssen. Einen Überblick über Voice-over-IP und die Analyse dieser Faktoren finden Sie bei Goode (2002).

Nachdem wir das Problem der Latenz im Media-Streaming-Pfad diskutiert haben, wenden wir uns einem weiteren wichtigen Problem zu, mit dem sich Konferenzsysteme beschäftigen müssen: Wie werden Anrufe auf- und abgebaut? Wir werden zwei Protokolle vorstellen, die hierfür allgemein eingesetzt werden: H.323 und SIP. Skype ist ein weiteres wichtiges System, aber seine Funktionsweise ist Betriebsgeheimnis.

H.323

Schon bevor das Internet für Sprach- und Videoanrufe genutzt wurde, war allen Beteiligten klar, dass, wenn jeder Anbieter einen selbst entwickelten Protokollstapel hätte, das System nie funktionieren würde. Um dieses Problem zu vermeiden, taten sich mehrere Interessenten unter der Schirmherrschaft der ITU zusammen und erarbeiteten Standards. 1996 gab die ITU die Empfehlung **H.323** heraus, die den Titel erhielt „Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service“ (*Visuelle Telefonsysteme und Geräte für lokale*

Netze, die eine nicht garantierte Dienstgüte bereitstellen). Nur die Telefonbranche kann sich einen solchen Namen ausdenken. Bei der Überarbeitung 1998 wurde dieser Name bereits in „Packet-based Multimedia Communications Systems“ (Paketbasierte Multimedia-Kommunikationssysteme) geändert. H.323 war die Basis für die ersten weitverbreiteten Internetkonferenzsysteme. Es liegt inzwischen in der siebten Version von 2009 vor und ist die am weitesten verbreitete Lösung.

H.323 ist eher eine Architekturübersicht über Internettelefonie als ein spezielles Protokoll. Es verweist auf sehr viele andere spezielle Protokolle, die sich mit der Codierung von Sprache, dem Gesprächsaufbau, der Signalübertragung, der Datenübertragung und anderen Bereichen beschäftigen, anstatt diese Dinge selbst festzulegen. Das allgemeine Modell ist in ► Abbildung 7.58 aufgeführt. Im Zentrum steht ein **Gateway**, welches das Internet mit dem Telefonnetz verbindet. Internetseitig wird über H.323-Protokolle kommuniziert und telefonseitig über PSTN-Protokolle. Die Kommunikationsgeräte werden als **Terminals** (Endgeräte) bezeichnet. In einem LAN kann es einen **Gatekeeper** geben, der die unter seiner Hoheit befindlichen Endpunkte – als **Zone** bezeichnet – kontrolliert.

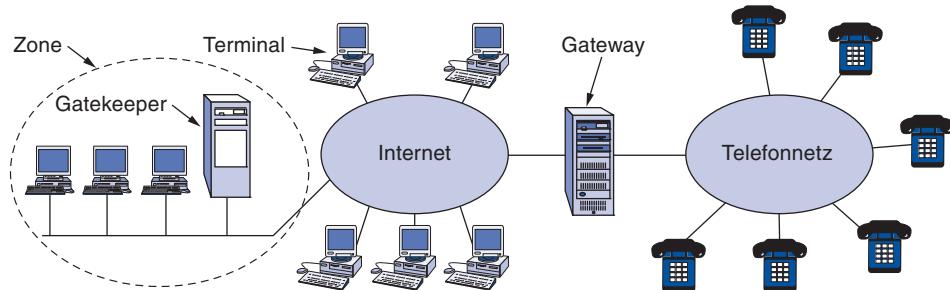


Abbildung 7.58: Das Architekturmodell von H.323 für die Internettelefonie.

Ein Telefonnetz benötigt verschiedene Protokolle, unter anderem ein Protokoll für die Codierung und Decodierung von Audio und Video. Standardtelefoniedarstellungen eines einzelnen Sprachkanals als 64 kbit/s digitales Audio (8 000 Samples mit 8 Bit pro Sekunde) sind in der ITU-Empfehlung **G.711** definiert. Alle H.323-Systeme müssen G.711 unterstützen. Es sind aber auch andere Sprachkomprimierungsprotokolle erlaubt (aber nicht vorgeschrieben). Sie verwenden verschiedene Komprimierungsalgorithmen und gehen verschiedene Kompromisse zwischen Qualität und Bandbreite ein. Für Video werden die bereits beschriebenen MPEG-Formen der Videokomprimierung unterstützt, einschließlich H.264.

Da mehrere Komprimierungsalgorithmen zulässig sind, ist ein Protokoll erforderlich, über das die Terminals verhandeln können, welchen sie verwenden. Dieses Protokoll heißt **H.245**. Hier werden auch andere Aspekte der Verbindung, wie die Bitübertragungsrate, verhandelt. RTPC ist für die Steuerung der RTP-Kanäle erforderlich. Darüber hinaus ist ein Protokoll zum Aufbau und zur Freigabe von Verbindungen, zur Bereitstellung der Wähltonen, zum Erzeugen der Klingeltöne und für die restlichen Dinge der Standardtelefonie erforderlich. Hier wird ITU **Q.931** verwendet. Außerdem benötigen die Terminals ein Protokoll für die Kommunikation mit dem Gatekeeper (falls vorhanden). Hierfür wird

H.225 verwendet. Der Kanal vom PC zum Gatekeeper, den dieses Protokoll verwaltet, wird als **RAS-Kanal (Registration/Admission/Status)** bezeichnet. Über diesen Kanal können Terminals unter anderem einer Zone beitreten oder sie verlassen, Bandbreite anfordern und sie zurückgeben oder Statusaktualisierungen bereitstellen. Schließlich wird ein Protokoll für die eigentliche Übertragung der Daten benötigt. Für diesen Zweck wird RTP über UDP verwendet. Es wird wie üblich von RTCP verwaltet. Die Positionierung dieses Protokolls wird in ►Abbildung 7.59 dargestellt.

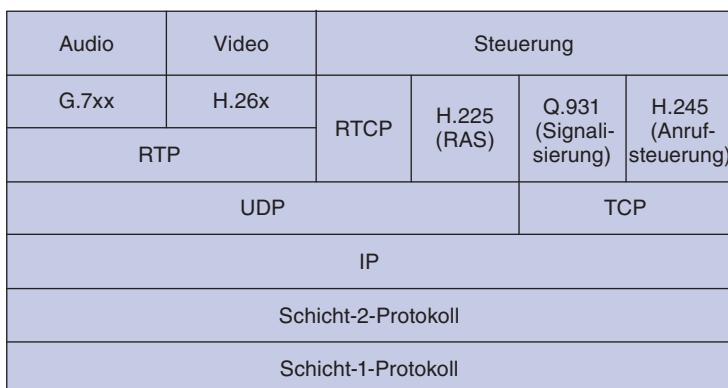


Abbildung 7.59: Der H.323-Protokollstapel.

Um das Zusammenspiel dieser Protokolle zu veranschaulichen, betrachten wir ein PC-Terminal in einem LAN (mit einem Gatekeeper), das ein entferntes Telefon anruft. Der PC muss zuerst den Gatekeeper ausfindig machen; daher sendet er ein UDP-Gatekeeper-Ermittlungspaket an Port 1718. Wenn der Gatekeeper antwortet, erfährt der PC die IP-Adresse des Gatekeepers. Der PC registriert sich beim Gatekeeper, indem er ihm eine RAS-Nachricht in einem UDP-Paket sendet. Nachdem dieses akzeptiert wurde, sendet der PC dem Gatekeeper eine RAS-Erlaubnisnachricht und fordert Bandbreite an. Erst nachdem Bandbreite zugewiesen wurde, kann der Gesprächsaufbau beginnen. Das Konzept der Bandbreitenanforderung im Voraus hat den Zweck, dass der Gatekeeper die Anzahl der Anrufe beschränken kann, um eine Überbelegung der Ausgangsleitungen zu vermeiden und um die erforderliche Dienstgüte bereitzustellen.

Eigentlich macht das Telefonsystem genau das Gleiche. Wenn Sie den Hörer heben, wird ein Signal an die lokale Teilnehmervermittlungsstelle gesendet. Wenn diese ausreichend ungenutzte Kapazität für einen neuen Anruf zur Verfügung hat, erzeugt sie einen Wählton. Wenn nicht, hören Sie nichts. Heutzutage ist das System so überdimensioniert, dass der Wählton fast immer sofort zu hören ist, während es früher oft einige Sekunden dauerte. Wenn Ihre Großkinder Sie also eines Tages fragen, warum es Wählton gibt, wissen Sie jetzt die Antwort. Doch wahrscheinlich gibt es bis dahin keine Telefone mehr.

Der PC errichtet anschließend eine TCP-Verbindung zum Gatekeeper, um den Gesprächsaufbau zu beginnen. Beim Gesprächsaufbau werden vorhandene Telefonnetzprotokolle verwendet, die verbindungsorientiert sind, sodass TCP erforderlich ist. Da

das Telefonsystem nichts hat, was RAS entspricht, damit Telefone ihre Anwesenheit bekannt machen können, konnten die H.323-Entwickler frei zwischen UDP und TCP für RAS wählen konnten. Sie entschieden sich für UDP mit dem niedrigeren Overhead.

Nachdem die Bandbreite zugeteilt ist, kann der PC über die TCP-Verbindung eine Q.931-Nachricht *SETUP* senden. Diese Nachricht enthält die Nummer des angerufenen Telefons (oder die IP-Adresse und den Port, wenn ein Computer angerufen wird). Der Gatekeeper antwortet mit einer Q.931-Nachricht *CALL PROCEEDING*, um den korrekten Eingang der Anfrage zu bestätigen, und leitet die *SETUP*-Nachricht an das Gateway weiter.

Das Gateway, das halb Computer und halb Telefonvermittlungsgerät ist, führt dann einen normalen Telefonanruf zu dem gewünschten (normalen) Telefon durch. Die Teilnehmervermittlungsstelle, an der das Telefon angeschlossen ist, bringt das angerufene Telefon zum Klingeln und sendet die Q.931-Nachricht *ALERT* zurück, um dem anrugenden PC mitzuteilen, dass der Klingelton erzeugt wird. Wenn die Person am anderen Ende das Telefon aufnimmt, sendet die Teilnehmervermittlungsstelle eine Q.931-Nachricht *CONNECT* zurück, um dem PC zu signalisieren, dass die Verbindung aufgebaut ist.

Wurde die Verbindung einmal aufgebaut, ist der Gatekeeper im Gegensatz zum Gateway nicht mehr in der Verbindung. Folgepakete umgehen den Gatekeeper und werden direkt an die IP-Adresse des Gateways geschickt. An diesem Punkt haben wir einen direkten Kanal zwischen den beiden Parteien. Es ist nur eine Verbindung auf der Bitübertragungsschicht zum Versenden von Bits, weiter nichts. Keine Seite weiß etwas von der anderen.

Mit dem H.245-Protokoll werden die Anrufparameter verhandelt. Es verwendet den H.245-Steuerungskanal, der immer offen ist. Jede Seite beginnt mit der Ankündigung ihrer Fähigkeiten, beispielsweise ob sie Video (H.323 kann auch Videokanäle verarbeiten) oder Telefonkonferenzen entgegennehmen kann, welche Codecs unterstützt werden usw. Wenn jede Seite weiß, was die andere verarbeitet, werden zwei unidirektionale Datenkanäle eingerichtet und jedem ein Codec und andere Parameter zugewiesen. Da jede Seite verschiedene Geräte besitzen kann, ist es durchaus möglich, dass die Codecs für den Empfangs- und den Rückkanal verschieden sind. Wenn alle Verhandlungen abgeschlossen sind, beginnt der Datenfluss unter Verwendung von RTP. Er wird mit RTCP verwaltet, das bei der Überlastungsüberwachung eine wichtige Rolle spielt. Sind Videodaten vorhanden, übernimmt RTCP die Audio-/Video-Synchronisation. Die verschiedenen Kanäle sind in ► Abbildung 7.60 dargestellt. Legt einer der Teilnehmer den Hörer auf, wird die Verbindung unter Verwendung des Q.931-Aufrufs „Call Signaling Channel“ abgebaut, nachdem der Anruf beendet ist, um Ressourcen freizugeben, die nicht länger benötigt werden.

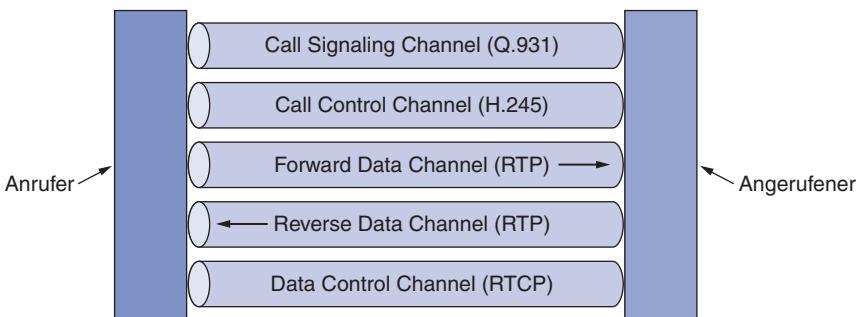


Abbildung 7.60: Logische Kanäle zwischen dem Anrufer und dem Angerufenen während eines Anrufs.

Wird der Anruf beendet, kontaktiert der anrufende PC wieder den Gatekeeper mit einer RAS-Nachricht, um die zugewiesene Bandbreite freizugeben. Alternativ kann er auch einen zweiten Anruf tätigen.

Bislang haben wir noch nichts über die Dienstgüte als Teil von H.323 gesagt, obwohl wir festgestellt haben, dass sie für den Erfolg von Echtzeitkonferenzen entscheidend ist. Der Grund hierfür ist, dass die Dienstgüte nicht in den Bereich von H.323 fällt. Wenn das zugrunde liegende Netz eine stabile, jitterfreie Verbindung vom anrufenden PC zum Gateway erzeugen kann, dann ist die Dienstgüte für den Anruf gut, andernfalls nicht. Der telefonseitige Teil des Anrufs ist immer jitterfrei, da das Telefonnetz so ausgelegt ist.

SIP – Session Initiation Protocol

H.323 wurde von der ITU (*International Telecommunication Union*) entwickelt. Für viele in der Internetgemeinde war dieses Protokoll ein typisches Produkt der Telekommunikationsbranche: groß, komplex und unflexibel. Daher rief die IETF ein Komitee ins Leben, das eine einfachere und modularere Lösung für Voice-over-IP erarbeiten sollte. Das bisher wichtigste Ergebnis ist **SIP** (*Session Initiation Protocol*, Sitzungsaktivierungsprotokoll). Die neueste Version wird in RFC 3261 aus dem Jahr 2002 beschrieben. Dieses Protokoll legt fest, wie über das Internet Telefonanrufe, Videokonferenzen und andere Multimedia-verbindungen aufgebaut werden. Im Unterschied zu H.323, das eine vollständige Protokollsuite darstellt, ist SIP nur ein Modul, das konzipiert wurde, um mit vorhandenen Internetanwendungen gut zusammenzuarbeiten. So werden Telefonnummern als URLs definiert, damit sie in Webseiten aufgenommen werden können, um mit einem Klick auf einen Link einen Telefonanruf zu starten (auf die gleiche Weise, wie mit einem Klick auf einen *mailto*-Link ein Programm gestartet wird, um eine E-Mail-Nachricht zu senden).

SIP kann Sitzungen mit zwei Parteien aufbauen (normale Telefonanrufe), Sitzungen mit mehreren Parteien (in denen jeder hören und sprechen kann) sowie Multi-cast-Sitzungen (ein Sender und viele Empfänger). Die Sitzungen können Audio, Video oder Daten enthalten, wobei Letzteres beispielsweise für Echtzeitspiele mit mehreren Spielern nützlich ist. SIP handhabt den Aufbau, die Verwaltung und die Beendigung von Sitzungen. Andere Protokolle wie RTP/RTCP werden für die Datenübertragung verwendet. SIP ist ein Protokoll der Anwendungsschicht und kann über UDP oder TCP ausgeführt werden.

SIP unterstützt eine Vielzahl von Diensten, einschließlich den angerufenen ausfindig machen (der eventuell nicht an seinem Heimrechner sitzt) und seine Fähigkeiten bestimmen, ebenso wie die Verbindung aufbauen und trennen. Im einfachsten Fall baut SIP eine Sitzung auf vom Computer des Anrufers zu dem des angerufenen. Diesen Fall untersuchen wir zuerst.

Die Telefonnummern werden in SIP als URLs unter Verwendung des *sip*-Schemas dargestellt, beispielsweise *sip:ilse@cs.university.edu* für eine Benutzerin namens Ilse an den mit dem DNS-Namen *cs.university.edu* angegebenen Host. SIP-URLs können auch IPv4- bzw. IPv6-Adressen oder konkrete Telefonnummern enthalten.

Das SIP-Protokoll ist ein textbasiertes Protokoll, das sich an HTTP anlehnt. Eine Partei sendet eine Nachricht im ASCII-Text, die aus einem Methodennamen in der ersten Zeile besteht, gefolgt von weiteren Zeilen mit Headern für die Parameterübergabe. Viele Header stammen aus MIME, damit SIP mit den bestehenden Internetanwendungen zusammenarbeiten kann. Die sechs in der Hauptspezifikation definierten Methoden werden in ► Abbildung 7.61 aufgeführt.

| Methode | Beschreibung |
|----------|--|
| INVITE | Initiierung einer Sitzung anfordern |
| ACK | Initiierung einer Sitzung bestätigen |
| BYE | Beendigung einer Sitzung anfordern |
| OPTIONS | Host über seine Möglichkeiten abfragen |
| CANCEL | Ausstehende Anforderung abbrechen |
| REGISTER | Umleitungsserver über den aktuellen Standort eines Benutzers informieren |

Abbildung 7.61: SIP-Methoden.

Um eine Sitzung einzurichten, erstellt ein Anrufer entweder eine TCP-Verbindung zum angerufenen und sendet eine *INVITE*-Nachricht, oder er sendet eine *INVITE*-Nachricht in einem UDP-Paket. In beiden Fällen beschreiben die Header in der zweiten und den folgenden Zeilen die Struktur des Nachrichtenhauptteils, der die Fähigkeiten des Anrufers, die Medientypen und die Formate angibt. Wenn der angerufene den Anruf entgegennimmt, antwortet er mit einem Antwortcode vom Typ HTTP (eine dreistellige Zahl unter Verwendung der Gruppen in Abbildung 7.38, 200 für die Annahme). Auf die Zeile mit dem Antwortcode kann der angerufene auch Informationen über seine Fähigkeiten, die Medientypen und Formate anhängen.

Die Verbindung wird über ein Dreiecks-Handshake aufgebaut, sodass der Anrufer mit einer *ACK*-Nachricht antwortet, um das Protokoll zu beenden und den Empfang der 200-Nachricht zu bestätigen. Jede Partei kann die Trennung einer Verbindung anfordern, indem eine Nachricht mit der *BYE*-Methode gesendet wird. Wenn die andere Seite dies bestätigt, wird die Sitzung beendet.

Die Methode *OPTIONS* dient zur Abfrage der Fähigkeiten eines Rechners. Sie wird in der Regel vor der Initiierung einer Sitzung verwendet, um herauszufinden, ob dieser Rechner überhaupt Voice-over-IP unterstützt oder welcher andere Sitzungstyp in Erwägung gezogen werden soll.

Die Methode *REGISTER* bezieht sich auf die Fähigkeit von SIP, einen Benutzer, der unterwegs ist, zu verfolgen und mit ihm eine Verbindung herzustellen. Die Nachricht wird an einen SIP-Location-Server gesendet, der im Blick hat, wer wo ist. Der Server kann dann später abgefragt werden, um den aktuellen Standort des Benutzers festzustellen. ►Abbildung 7.62 zeigt, wie die Umleitung funktioniert. Hier sendet der Anrufer eine *INVITE*-Nachricht an einen Proxy-Server, um die mögliche Umleitung zu verborgen. Der Proxy sucht dann, wo sich der Benutzer befindet, und sendet ihm die *INVITE*-Nachricht. Er fungiert damit als Vermittler für die nachfolgenden Nachrichten im Dreiwege-Handshake. Die Nachrichten *LOOKUP* und *REPLY* gehören nicht zu SIP. Es kann jedes gängige Protokoll verwendet werden, je nachdem, welcher Location-Server zum Einsatz kommt.

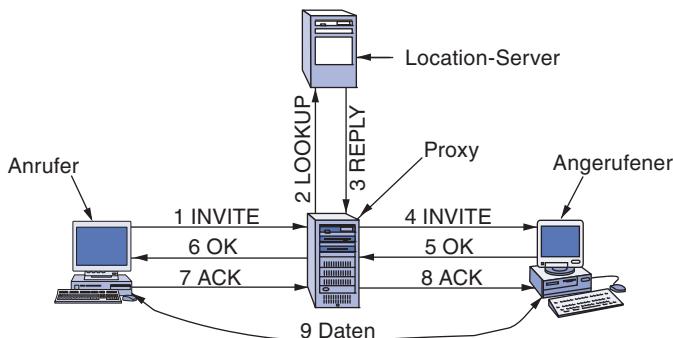


Abbildung 7.62: Einsatz eines Proxy-Servers und Umleitung mit SIP.

SIP zeichnet sich noch durch eine Vielzahl anderer Funktionen aus, auf die wir hier nicht weiter eingehen können, wie Anklopfen, Rufabschirmung, Verschlüsselung und Authentifikation. SIP kann darüber hinaus von einem Computer aus ein normales Telefon anrufen, wenn ein passendes Gateway zwischen dem Internet und dem Telefonsystem verfügbar ist.

Vergleich von H.323 und SIP

Sowohl H.323 als auch SIP erlauben Anrufe zwischen zwei und mehreren Parteien, die sowohl Computer wie auch Telefone als Endgeräte einsetzen. Beide unterstützen Aushandlung der Parameter, Verschlüsselung und RTP/RTCP-Protokolle. Eine Zusammenfassung der Ähnlichkeiten und Unterschiede ist in ►Abbildung 7.63 aufgeführt.

| Eigenschaft | H.323 | SIP |
|------------------------------------|---------------------------------|----------------------------------|
| Entworfen | ITU | IETF |
| Kompatibel mit PSTN | Ja | Größtenteils |
| Kompatibel mit dem Internet | Ja, mit der Zeit | Ja |
| Architektur | Monolithisch | Modular |
| Vollständigkeit | Kompletter Protokollstapel | SIP übernimmt nur das Einrichten |
| Verhandlung der Parameter | Ja | Ja |
| Signalisierung des Anrufs | Q.931 über TCP | SIP über TCP oder UDP |
| Nachrichtenformat | Binär | ASCII |
| Medienübertragung | RTP/RTCP | RTP/RTCP |
| Gespräche mit mehreren Teilnehmern | Ja | Ja |
| Multimedialkonferenzen | Ja | Nein |
| Adressierung | URL oder Telefonnummer | URL |
| Gesprächsbeendigung | Explizit oder TCP-Freigabe | Explizit oder Timeout |
| Instant Messaging | Nein | Ja |
| Verschlüsselung | Ja | Ja |
| Umfang des Standards | 1400 Seiten | 250 Seiten |
| Implementierung | Groß und komplex | Mittel |
| Status | Weitverbreitet, besonders Video | Alternativ, besonders Audio |

Abbildung 7.63: Vergleich von H.323 und SIP.

Obwohl die Funktionen sehr ähnlich sind, unterscheiden sich die beiden Protokolle konzeptionell sehr stark. H.323 ist ein typischer schwergewichtiger Standard der Telefonindustrie, der den gesamten Protokollstapel spezifiziert und genau definiert, was erlaubt ist und was nicht. Dieser Ansatz führt in jeder Schicht zu sehr gut definierten Protokollen und vereinfacht die Interoperabilität. Der hierfür gezahlte Preis ist ein großer, komplexer und rigider Standard, der schwer an zukünftige Anwendungen anzupassen ist.

Im Gegensatz dazu ist SIP ein typisches Internetprotokoll, bei dem nur kurze ASCII-Zeilen ausgetauscht werden. Es ist ein leichtes Modul, das mit anderen Internetprotokollen gut zusammenarbeitet, weniger gut allerdings mit den Signalübertragungsprotokollen des vorhandenen Telefonsystems. Da das IETF-Modell von Voice-over-IP stark modular ist, ist es flexibel und kann einfach an neue Anwendungen angepasst werden. Der Nachteil sind immer wieder auftauchende Interoperabilitätsprobleme, da die Betroffenen versuchen zu interpretieren, was der Standard bedeutet.

7.5 Content Delivery

Wie das Telefonnetz diente das Internet vor allem der Kommunikation. Schon früh kommunizierten Akademiker mit entfernten Rechnern, indem sie sich über das Netz einloggten und Aufgaben ausführten. Schon lange nutzen die Menschen E-Mail, um miteinander zu kommunizieren, und jetzt nutzen Sie darüber hinaus noch Video und Voice-over-IP. Seitdem das Web jedoch erwachsen geworden ist, dreht sich das Internet mehr um Inhalte (*content*) als um Kommunikation. Viele nutzen das Web für die Informationssuche, daneben gibt es einen enormen Zuwachs an Peer-to-Peer-Dateiaustausch, der auf die große Nachfrage nach Zugriff auf Filme, Musik und Programme zurückzuführen ist. Der Wechsel zu Inhalten war so ausgeprägt, dass der Großteil der Bandbreite heutzutage für die Übertragung gespeicherter Videos benötigt wird.

Da die Übertragung von Inhalten sich von der Übertragung von Kommunikation unterscheidet, sind auch die Anforderungen an das Netz andere. Wenn beispielsweise Sandra mit Jörg sprechen möchte, kann sie über VoIP sein Handy anrufen. Die Kommunikation muss über einen bestimmten Computer erfolgen. Es würde keinen Sinn machen, Pauls Computer anzurufen. Doch wenn Jörg das letzte Kricketspiel seiner Mannschaft sehen möchte, ist es ihm egal, von welchem Computer er das Video streamt, ob von Sandras oder Pauls Computer oder, was wahrscheinlicher ist, von einem unbekannten Server im Internet. Das heißt, bei Inhalt spielt der Ort keine Rolle, es sei denn er hat Einfluss auf die Leistung (und Verletzung des Urheberrechts).

Der andere Unterschied ist, dass einige Websites, die Inhalte bereitstellen, extrem populär geworden sind. Das beste Beispiel hierfür ist YouTube. Diese Plattform erlaubt den Benutzern, selbst erstellte Videos zu jedem beliebigen Thema für andere zum Betrachten hochzuladen, wovon viele Gebrauch machen, während der Rest der Welt die Videos betrachten will. Schätzungen zufolge macht YouTube mit all seinen Bandbreite konsumierenden Videos inzwischen fast 10 % des Internetverkehrs aus.

Kein einzelner Server ist leistungsstark oder zuverlässig genug, um eine solche überraschend hohe Nachfrage zu bewältigen. Deshalb bauen YouTube und andere große Content-Provider ihre eigenen Verteilernetze auf. Diese Netze nutzen Datenzentren in der ganzen Welt, um ihre Inhalte einer extrem großen Anzahl an Clients schnell und zuverlässig bereitzustellen.

Die Techniken, die zur Content-Verteilung verwendet werden, sind mit der Zeit entwickelt worden. Als das Web anfing zu wachsen, war es gerade diese Popularität, die fast seinen Ruin bedeutet hätte. Die steigende Nachfrage nach Content führte dazu, dass Server und Netze häufig überlastet waren. Viele nannten daraufhin das WWW auch das World Wide Wait.

Als Reaktion auf die Konsumentennachfrage wurden im Herzen des Internets sehr große Mengen an Bandbreite freigeschaltet und schnellere Breitbandanschlüsse bei den einzelnen Endabnehmern eingeführt. Diese Bandbreite war der Schlüssel zur Leistungsverbesserung, aber sie ist nur Teil der Lösung. Um die endlosen Verzögerungen zu verringern, entwickelten Wissenschaftler außerdem verschiedene Architekturen, um die Bandbreite optimal für die Content-Verteilung zu nutzen.

Eine dieser Architekturen ist das **CDN** (*Content Distribution Network*, Content-Verteilernetz), bei dem ein Provider mehrere Rechner verteilt an verschiedenen Orten im Internet einrichtet und sie dazu nutzt, um Inhalte an die Clients zu schicken. Hierfür entscheiden sich vor allem die großen Provider – die sogenannten „Big Player“. Eine alternative Architektur ist ein **P2P-Netz** (**Peer-to-Peer**). Hierbei bündeln mehrere Rechner ihre Ressourcen, um sich gegenseitig Inhalt zuzusenden, und zwar ohne separat zugeschaltete Server oder irgendeinen anderen zentralen Kontrollpunkt. Die Faszination dieser Idee liegt darin, dass viele kleine „Player“ durch diese Zusammenarbeit eine enorme Schlagkraft entwickeln können.

In diesem Abschnitt wollen wir uns dem Problem der Content-Verteilung im Internet zuwenden und einige Lösungen betrachten, die in der Praxis Anwendung finden. Nachdem wir kurz die Popularität von Content und den Internetverkehr diskutiert haben, beschreiben wir, wie leistungsstarke Webserver aufgebaut werden und wie mit Zwischenspeicherung die Leistungsfähigkeit der Webclients verbessert werden kann. Anschließend beschäftigen wir uns mit den zwei wichtigsten Architekturen zur Content-Verteilung: CDNs und P2P-Netze. Entwurf und Eigenschaften sind jeweils ganz unterschiedlich, wie wir dann sehen werden.

7.5.1 Content und Internetverkehr

Um ein gut funktionierendes Netz zu entwerfen und aufzubauen, müssen Sie wissen, wie hoch das zu erwartende Verkehrsvolumen ist, das über das Netz übertragen werden soll. Mit dem Übergang des Internetverkehrs auf Content-Übertragung wurden beispielsweise die Server aus den Firmenbüros in Internetdatenzentren verlagert, die eine große Anzahl an Rechnern mit ausgezeichneter Netzwerkkonnektivität bereitstellen. Selbst wenn Sie heutzutage nur einen kleinen Server betreiben wollen, ist es einfacher und billiger, einen virtuellen Server zu mieten, der in einem Internetdatenzentrum gehostet wird, als einen realen Rechner mit Breitbandanschluss von zu Hause oder dem Büro aus zu betreiben.

Zum Glück gibt es nur zwei Fakten, die Sie über den Internetverkehr unbedingt kennen müssen. Der erste Fakt ist, dass es sich schnell ändert, und zwar nicht nur im Detail, sondern auch im Gesamtaufbau. Vor 1994 bestand der Verkehr größtenteils aus E-Mail und der herkömmlichen FTP-Dateiübertragung (zum Schicken von Programmen und Datensätzen zwischen den Rechnern). Dann kam das Web und wuchs schnell exponentiell an. Der Webverkehr hatte den FTP- und E-Mailverkehr bereits weit hinter sich gelassen, als die Internetblase 2000 platzierte. Ungefähr ab diesem Zeitpunkt wurde damit begonnen, Musikdateien und später auch Filme von Peer zu Peer auszutauschen (*P2P file sharing*). 2003 dann bestand der Internetverkehr hauptsächlich aus P2P-Verkehr, der damit den Webverkehr ablöste. Und irgendwann gegen Ende des Jahrzehnts begannen CDNs, mit denen Videos von Websites wie YouTube gestreamt wurden, den P2P-Verkehr an Bedeutung abzulösen. Cisco schätzt, dass 2014 ungefähr 90 % des gesamten Internetverkehrs aus Videos in der einen oder anderen Art besteht (Cisco, 2010).

Doch nicht immer spielt das Verkehrsvolumen eine Rolle. Obwohl beispielsweise der VoIP-Verkehr boomte, noch bevor 2003 Skype aufkam, wird er immer nur eine kleine Zacke in der Kurve bleiben, da der Bandbreitenbedarf für Audio um zwei Größenordnungen kleiner ist als für Video. Allerdings belastet der VoIP-Verkehr das Netz in anderer Hinsicht, da er anfällig für Latenz ist. Als weiteres Beispiel seien die sozialen Onlinenetzwerke genannt, die quasi explodiert sind, seit Facebook 2004 gestartet wurde. 2010 erreichte Facebook zum ersten Mal mehr Benutzer im Web pro Tag als Google. Auch wenn man den Verkehr mal beiseite lässt (und es gibt wirklich sehr viel Verkehr), sind soziale Onlinenetzwerke wichtig, da sie die Art und Weise ändern, wie Menschen über das Internet interagieren.

Worauf wir hinweisen wollen, ist, dass es im Internetverkehr schnell und fast regelmäßig zu grundlegenden Veränderungen kommt. Was erwartet uns als Nächstes? Lesen Sie dies bitte in der sechsten Ausgabe dieses Buches – wir werden es Ihnen wissen lassen.

Der zweite wichtige Fakt ist, dass der Internetverkehr extrem verzerrt ist. Viele Eigenschaften, die wir kennen, bewegen sich um einen Durchschnittswert. Zum Beispiel haben die meisten Erwachsenen eine Durchschnittsgröße. Es gibt einige große und einige kleine Menschen, aber nur wenige sehr große oder sehr kleine Menschen. Bei solchen Eigenschaften ist es möglich, den Entwurf auf einen relativ kleinen Bereich zu beschränken, der dennoch die Mehrheit der Bevölkerung umfasst.

Beim Internetverkehr ist das nicht so. Seit Langem weiß man, dass es eine kleine Anzahl von Websites mit extrem viel Verkehr gibt und eine riesige Anzahl von Websites mit wesentlich weniger Verkehr. Diese Tatsache hat Einfluss auf die Netzwerkterminologie genommen. Frühere Aufsätze sprachen über den Verkehr in Form von **Paketzügen**, in der Annahme, dass Expresszüge mit einer großen Anzahl von Paketen plötzlich über eine Verbindung reisen würden (Jain und Routhier, 1986). Dies wurde als **Selbstähnlichkeit** formalisiert, die Sie sich für unsere Zwecke als Netzverkehr vorstellen können, der immer viele kurze und viele lange Lücken aufweist, auch wenn er auf verschiedenen Zeitskalen betrachtet wird (Leland et al., 1994). Spätere Abhandlungen bezeichneten lange Verkehrsflüsse als **Elefanten** und kurze Verkehrsflüsse als **Mäuse**. Die Idee ist, dass es nur wenige Elefanten, aber viele Mäuse gibt, dass aber Elefanten wichtig sind, weil sie so groß sind.

Wenn wir zum Web-Content zurückkehren, zeigt sich hier die gleiche Art der Verzerrung. Die Erfahrungen mit Videoverleihfirmen, öffentlichen Bibliotheken und anderen ähnlichen Organisationen haben gezeigt, dass nicht alle Filme gleich beliebt sind. Untersuchungen haben ergeben: Wenn N Spielfilme verfügbar sind, ist der Bruchteil aller Anfragen für den k -ten beliebtesten Film etwa C/k . Hier wird C berechnet, um die Summe auf 1 zu normalisieren, und zwar

$$C = 1 / (1 + 1/2 + 1/3 + 1/4 + 1/5 + \dots + 1/N)$$

Daher ist der beliebteste Film etwa siebenmal beliebter als der Film auf Platz 7. Dieses Ergebnis ist als das **Zipfsche Gesetz** (Zipf, 1949) bekannt. Es ist nach George Zipf, einem Linguistikprofessor der Harvard-Universität, benannt, der festgestellt hat, dass

die Häufigkeit eines Wortes in einem großen Text umgekehrt proportional seiner Position in der Reihenfolge ist. Beispielsweise wird das 40sthäufige Wort doppelt so oft verwendet wie das 80sthäufige Wort und dreimal so oft wie das 120sthäufige Wort.

In ▶ Abbildung 7.64a sehen Sie eine Zipf'sche Verteilung. Sie spiegelt die Aussage wider, dass es eine kleine Anzahl von beliebten Elementen und eine große Menge an weniger beliebten Elementen gibt. Um Verteilungen dieser Art zu erkennen, bietet es sich an, die Werte mit logarithmischer Skalierung auf beiden Achsen abzutragen (▶ Abbildung 7.64b). Das Ergebnis sollte eine gerade Linie sein.

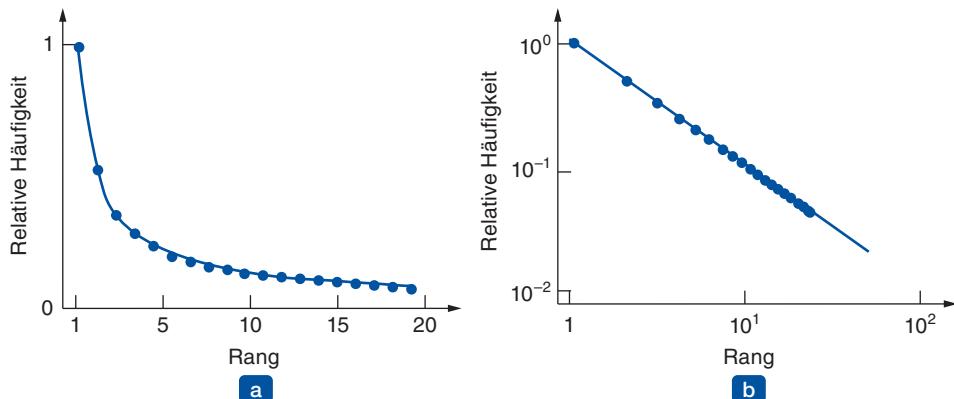


Abbildung 7.64: Zipf'sche Verteilung (a) mit linearer Skalierung; (b) mit Log-Log-Skalierung.

Bei der Untersuchung der Popularität von Webseiten stellte man fest, dass auch diese in etwa dem Zipf'schen Gesetz folgte (Breslau et al., 1999). Die Zipf'sche Verteilung ist nur ein Beispiel aus einer Familie von Verteilungen, die als **Potenzgesetze** bekannt sind. Potenzgesetze liegen vielen menschlichen Phänomenen zugrunde, wie der Verteilung der Stadtbevölkerung und des Wohlstandes. Sie neigen ebenfalls dazu, einige wenige große Player und sehr viele kleinere Player zu beschreiben und sie lassen sich ebenfalls als gerade Linie in einem Log-Log-Diagramm zeichnen. Man stellte schon bald fest, dass die Topologie des Internets grob mit Potenzgesetzen beschrieben werden konnte (Faloutsos et al., 1999). Daraufhin begannen die Wissenschaftler jede nur denkbare Eigenschaft des Internets auf einer logarithmischen Skala abzutragen, wobei sie bei jeder geraden Linie „Potenzgesetz!“ schrieben.

Wichtiger als eine gerade Linie in einem Log-Log-Diagramm ist, was diese Verteilungen für den Entwurf und die Nutzung der Netze bedeutet. In Anbetracht der vielen Content-Formen, die einer Verteilung nach dem Zipf'schen oder dem Potenzgesetz folgen, scheint es wichtig, dass Websites im Internet von der Beliebtheit her ebenfalls Zipf folgen. Dies wiederum bedeutet, dass eine durchschnittliche Site kein nützlicher Repräsentant ist. Sites sollten besser als populär oder nicht populär klassifiziert werden. Beide Arten von Sites sind auf ihre Art wichtig. Die populären Sites natürlich, weil die wenigen populären Sites für den Großteil des Verkehrs im Internet verantwortlich sein können. Vielleicht überrascht es Sie, dass auch die nicht so populären Sites eine Rolle spielen können. Das liegt daran, dass das Verkehrsvolumen der weniger populären Sites

zusammengenommen einen großen Anteil am Gesamtverkehr ausmachen kann, was wiederum darauf zurückzuführen ist, dass es so viele nicht populäre Sites gibt. Die Vorstellung, dass viele weniger populäre Optionen zusammen eine Rolle spielen können, wurde von Büchern wie *The Long Tail* (Anderson, 2008a) verbreitet.

Kurven, die so wie in Abbildung 7.64a abklingen, sind häufig, aber nicht alle gleich. Vor allem in Situationen, in denen die Abklingrate proportional der Menge des übrig gebliebenen Materials ist (z.B. bei instabilen radioaktiven Atomen), kommt es zu einer **exponentiellen Abklingrate**, die viel schneller abfällt als das Zipf'sche Gesetz. Die Anzahl der Elemente, hier der Atome, die nach der Zeit t übrig bleiben, wird normalerweise ausgedrückt als $e^{-t/\alpha}$, wobei die Konstante α festlegt, wie schnell das Abklingen erfolgt. Der Unterschied zwischen exponentiellem Abklingen und dem Zipf'schen Gesetz ist, dass Sie bei Ersterem getrost das Ende vom Ausläufer ignorieren können, während beim Zipf'schen Gesetz das Gesamtgewicht des Auslängers bedeutend ist und nicht ignoriert werden kann.

Um in dieser verzerrten Welt effektiv zu arbeiten, müssen wir beide Arten von Web-sites erstellen können. Weniger populäre Sites sind einfacher zu verwalten. Durch die Verwendung von DNS ist es möglich, dass viele verschiedene Sites eigentlich auf den-selben Computer im Internet zeigen, der alle Sites ausführt. Beliebte Sites hingegen sind nur schwer zu verwalten. Es gibt keinen Computer, der auch nur entfernt leis-tungsstark genug ist, und außerdem würde die Verwendung nur eines Computers bei Ausfall dazu führen, dass Millionen von Besuchern keinen Zugriff mehr auf die Site hätten. Um solche Sites zu verwalten, müssen wir Content-Verteilersysteme aufbauen. Dieser Aufgaben wollen wir uns als Nächstes widmen.

7.5.2 Serverfarmen und Webproxys

Die Webdesigns, die wir bisher gesehen haben, gehen von einem einzigen Server aus, der mit vielen Client-Rechnern spricht. Beim Aufbau großer Websites, die gute Leistungen zeigen, können wir die Verarbeitung entweder serverseitig oder clientseitig beschleuni-gen. Serverseitig ist es möglich, leistungsstärkere Webserver in Form einer Serverfarm zu erstellen, in denen ein Rechner-Cluster als ein einziger Server agiert. Clientseitig wird eine bessere Leistung mit besseren Techniken zur Zwischenspeicherung erreicht.

Wir werden die einzelnen Techniken nacheinander beschreiben. Dabei ist jedoch zu beachten, dass keine Technik allein ausreicht, um die größten Websites aufzubauen. Für sehr beliebte Sites benötigen Sie die Content-Verteilermethoden, die wir in den folgenden Abschnitten beschreiben werden und die darin bestehen, Rechner an vielen verschiedenen Orten zu kombinieren.

Serverfarmen

Egal, wie viel Bandbreite ein Rechner zur Verfügung hat, er kann nur so viele Web-anfragen bearbeiten, wie diese Bandbreite erlaubt. Die Lösung besteht in diesem Fall darin, mehr als einen Computer als Webserver zu verwenden. Dadurch erhalten Sie das Modell der **Serverfarm** aus ►Abbildung 7.65.

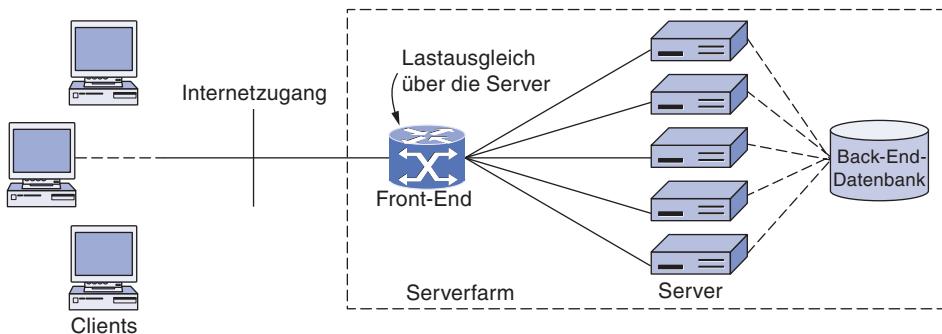


Abbildung 7.65: Eine Serverfarm.

Die Schwierigkeit bei diesem anscheinend sehr einfachen Modell ist, dass die Rechner, aus denen die Serverfarm besteht, für die Clients wie eine einzige logische Website aussehen müssen. Wenn dies nicht gelingt, haben wir nur verschiedene Websites eingerichtet, die parallel ausgeführt werden.

Es gibt mehrere mögliche Lösungen, um die Gruppe von Servern als eine Website erscheinen zu lassen. Alle Lösungen gehen davon aus, dass jeder Server eine Anfrage von jedem Client bearbeiten kann. Hierfür muss jeder Server eine Kopie der Website haben. Zu diesem Zweck werden die Server mit einer gemeinsamen Back-End-Datenbank verbunden, was durch eine gestrichelte Linie angezeigt wird.

Eine Lösung besteht darin, mithilfe des DNS die Anfragen auf die Server in der Serverfarm zu verteilen. Wenn eine DNS-Anfrage für die Web-URL erfolgt, liefert der DNS-Server eine rotierende Liste der IP-Adressen der Server zurück. Jeder Client versucht eine IP-Adresse, normalerweise die erste auf der Liste. Der Effekt ist, dass verschiedene Clients verschiedene Server kontaktieren, um auf dieselbe Website zuzugreifen – genau wie es beabsichtigt war. Die DNS-Methode ist das Kernstück der CDNs und wir werden später in diesem Abschnitt noch einmal darauf zu sprechen kommen.

Die anderen Lösungen basieren auf einem **Front-End**, das die eingehenden Anfragen auf den Pool von Servern in der Serverfarm verteilen. Dies geschieht auch dann, wenn der Client die Serverfarm unter Angabe einer bestimmten Ziel-IP-Adresse kontaktiert. Das Front-End ist normalerweise ein Switch der Sicherungsschicht oder ein IP-Router, das heißt ein Gerät, das Rahmen oder Pakete verarbeitet. Alle Lösungen basieren darauf (bzw. auf den Servern), um einen Blick auf die Header der Vermittlungs-, Transport- oder Anwendungsschicht zu werfen und diese auf unübliche Weise zu verwenden. Webanfragen und -antworten werden mit TCP übertragen. Für eine korrekte Funktionsweise muss das Front-End alle Pakete für eine Anfrage an denselben Server senden.

Ein einfacher Ansatz wäre, wenn das Front-End alle eingehenden Anfragen an alle Server schicken würde. Jeder Server beantwortet nach vorheriger Vereinbarung nur einen Teil der Anfragen. Zum Beispiel könnten 16 Server die Quell-IP-Adresse überwachen und auf die Anfrage nur antworten, wenn die letzten 4 Bit der Quell-IP-Adresse mit den für sie konfigurierten Selektoren übereinstimmen. Andere Pakete werden verworfen.

Auch wenn hierbei eingehende Bandbreite verschwendet wird, ist es bei Weitem nicht so ineffizient wie es klingt, da die Antworten oft viel länger sind als die Anfragen.

In einem eher allgemeinen Entwurf könnte das Front-End die IP-, TCP- und HTTP-Header der Pakete auswerten und die Pakete dann willkürlich einem Server zuweisen. Diese Zuweisung wird **Lastausgleich** (*load balancing*) genannt, da das Ziel darin besteht, die Last gleichmäßig auf die Server zu verteilen. Diese Strategie kann einfach oder komplex sein. Bei einer einfachen Strategie würden die Server beispielsweise der Reihe nach oder rundum (*round robin*) zugewiesen. Bei diesem Ansatz muss sich das Front-End die Zuordnung jeder Anfrage merken, sodass nachfolgende Pakete, die Teil derselben Anfrage sind, an denselben Server gesendet werden. Und soll die Site zuverlässiger sein als ein einzelner Server, muss das Front-End außerdem feststellen können, wann Server ausgefallen sind, damit es an diese Server keine weiteren Anfragen sendet.

Ähnlich wie bei NAT birgt dieser allgemeine Entwurf seine Gefahren oder ist zumindest schwach, insofern als wir damit das grundlegendste Prinzip der Schichtenprotokolle verletzen: Jede Schicht muss zu Kontrollzwecken ihren eigenen Header verwenden und darf die Daten der Nutzlast zu keinem Zweck anschauen und verwenden. Dennoch werden solche Systeme entworfen und, wenn sie irgendwann in der Zukunft aufgrund von Änderungen in den höheren Schichten nicht mehr funktionieren, sind die Entwickler überrascht. Das Front-End ist in diesem Fall ein Switch oder ein Router, kann aber auf der Basis von Daten der Transportschicht oder höher Maßnahmen ergreifen. Ein solches Gerät wird **Middlebox** genannt, weil es mitten im Netzwerkpfad zwischengeschaltet wird, wo es gemäß dem Protokollstapel nichts verloren hat. In diesem Fall wird das Front-End am besten als interner Teil einer Serverfarm betrachtet, der alle Schichten bis zur Anwendungsschicht beendet (und folglich alle Header-Daten für diese Schichten nutzen kann).

Nichtsdestotrotz hat sich dieses Design, wie bei NAT, in der Praxis bewährt. Kenntnisse der TCP-Header tragen zu einem besseren Lastausgleich bei, als dies nur mit den Informationen der IP-Header möglich wäre. Eine IP-Adresse könnte beispielsweise ein ganzes Unternehmen repräsentieren und viele Anfragen stellen. Erst die Informationen der TCP-Schicht oder höher verraten Ihnen, ob diese Anfragen auf verschiedene Server verteilt werden können.

Die HTTP-Header sind aus einem etwas anderen Grund interessant. Viele Webinteraktionen greifen auf Datenbanken zu und aktualisieren sie, zum Beispiel wenn ein Kunde seinen neuesten Kauf aufruft. Der Server, der diese Anfrage beantwortet, muss die Back-End-Datenbank abfragen. Es ist nützlich, alle nachfolgenden Anfragen von einem Benutzer auch an denselben Server zu leiten, da dieser bereits Daten über den Benutzer zwischengespeichert hat. Der einfachste Weg hierzu führt über die Verwendung von Web-Cookies (oder anderen Daten, die den Benutzer kennzeichnen), wobei die Cookies in den HTTP-Headern verborgen sind.

Obwohl wir diesen Entwurf bereits für Websites beschrieben haben, sei abschließend angemerkt, dass eine Serverfarm auch für andere Arten von Servern erstellt werden kann. Server, die Mediendaten über UDP streamen, sind ein Beispiel dafür. Die ein-

zige erforderliche Änderung ist, dass das Front-End in der Lage sein muss, einen Lastausgleich bei diesen Anfragen herzustellen (die andere Protokoll-Header-Felder haben als Webanfragen).

Webproxys

Webanfragen und -antworten werden mittels HTTP gesendet. In Abschnitt 7.3 haben wir beschrieben, wie Browser Antworten zwischenspeichern können, um sie dann zur Beantwortung zukünftiger Anfragen wiederzuverwenden. Der Browser verwendet verschiedene Header-Felder und -Regeln, um festzulegen, ob eine zwischengespeicherte Webseite-Kopie noch aktuell ist. Auf dieses Thema wollen wir hier nicht noch einmal eingehen.

Die Zwischenspeicherung verbessert die Leistung, indem sie die Antwortzeit verkürzt und die Netzwerklast reduziert. Wenn der Browser selbst feststellen kann, ob eine zwischengespeicherte Seite aktuell ist, kann die Seite direkt aus dem Cache geholt werden, ohne dass irgendein Netzverkehr anfällt. Und selbst wenn der Browser sich vom Server bestätigen lassen muss, dass die Seite aktuell ist, wird die Antwortzeit verkürzt und die Netzwerklast reduziert – vor allem für große Seiten –, da nur eine kleine Nachricht gesendet werden muss.

Das Beste, was der Browser jedoch machen kann, ist, alle vom Benutzer zuvor besuchten Websites zwischenspeichern. Vielleicht erinnern Sie sich von unserer Diskussion der Popularität, dass es neben einigen populären Seiten, die viele Menschen immer wieder besuchen, viele, viele weniger populäre Seiten gibt. In der Praxis beschränkt dies die Effektivität der Browserzwischenspeicherung, da es viele Seiten gibt, die von einem bestimmten Benutzer nur einmal besucht werden. Diese Seiten müssen immer vom Server abgerufen werden.

Eine Strategie, um Caches effektiver zu machen, besteht darin, dass mehrere Benutzer sich einen Cache teilen. Auf diese Weise kann eine Seite, die bereits für einen Benutzer übertragen wurde, einem anderen angezeigt werden, wenn dieser die gleiche Anfrage gestellt hat. Ohne die Zwischenspeicherung beim Browser müssten beide Benutzer die Seite vom Server abrufen. Diese gemeinsame Nutzung ist nicht möglich bei verschlüsseltem Verkehr, Seiten, die authentifiziert werden müssen, und nicht speicherbaren Seiten (z.B. aktuelle Aktienkurse), die von Programmen zurückgeliefert werden. Vor allem dynamische Seiten, die von Programmen erzeugt werden, sind eine zunehmende Spezies, für die eine Zwischenspeicherung unwirksam ist. Nichtsdestotrotz gibt es viele Seiten, die für viele Benutzer sichtbar sind und die immer gleich aussehen, egal welcher Benutzer die Anfrage macht (z.B. Bilder).

Für die gemeinsame Nutzung eines Cache wird ein [Webproxy](#) eingesetzt. Ein Proxy ist ein Stellvertreter, der im Auftrag von jemand anderem, z.B. dem Benutzer, handelt. Es gibt viele Arten von Proxys. Zum Beispiel antwortet ein ARP-Proxy auf ARP-Anfragen im Auftrag von einem Benutzer, der woanders ist, (und nicht für sich selbst antworten kann). Ein Webproxy holt Webanfragen im Auftrag seiner Benutzer. Die Webantworten werden normalerweise bei ihm zwischengespeichert, und da er von mehreren Benutzern genutzt wird, hat er einen erheblich größeren Cache als ein Browser.

Wenn eine Organisation ein Proxy verwendet, sieht das in der Regel so aus, dass die Organisation den Webproxy allen seinen Benutzern zur Verfügung stellt. Die Organisation kann beispielsweise ein Unternehmen oder ein Internetdienstanbieter (ISP) sein. Beide profitieren davon, wenn die Webanfragen ihrer Benutzer beschleunigt werden und der Bedarf an Bandbreite reduziert wird. Denn während für den Endbenutzer unabhängig von der Nutzung Flatrates üblich sind, werden die meisten Unternehmen und ISPs nach der Höhe der genutzten Bandbreite belangt.

Der Aufbau ist in ►Abbildung 7.66 zu sehen. Jeder Browser ist so konfiguriert, dass die Seitenanfragen an den Proxy geleitet werden und nicht an den eigentlichen Server der Seite. Wenn der Proxy die Seite bereits gespeichert hat, liefert er sie sofort zurück. Wenn nicht, holt er die Seite vom Server, legt sie für zukünftige Anfragen im Cache ab und überträgt sie an den Client, der sie angefordert hat.

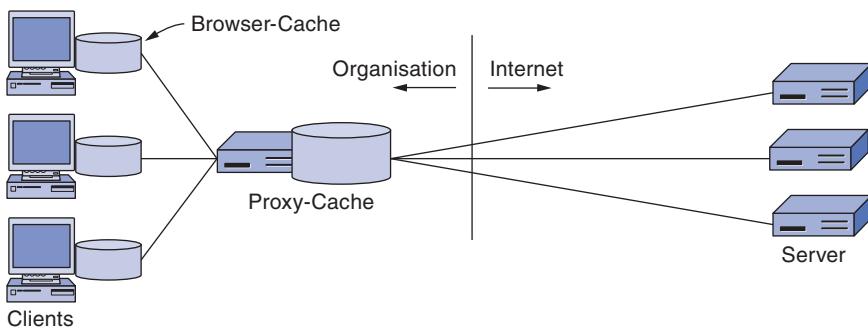


Abbildung 7.66: Ein Proxy-Cache zwischen Webbrowsersn und Webservern.

Clients senden ihre Webanfragen nicht nur an den Proxy anstatt an den eigentlichen Server, sondern führen unter Verwendung des Browser-Caches auch ihre eigene Zwischenspeicherung durch. Der Proxy wird erst konsultiert, wenn die Anfrage beim eigenen Browser-Cache vergeblich war. Das heißt, der Proxy ist eine zweite Ebene der Zwischenspeicherung.

Es können weitere Proxys hinzugefügt werden, die Ebenen ergänzen. Jeder Proxy (oder Browser) macht die Anfragen über einen **Upstream-Proxy**. Jeder Upstream-Proxy dient als Cache für die **Downstream-Proxys** (oder Browser). Auf diese Weise ist es möglich, dass Browser in einem Unternehmen einen Firmen-Proxy verwenden, der einen ISP-Proxy verwendet, der die Webserver direkt kontaktiert. Oft jedoch reicht eine Ebene der Proxy-Zwischenspeicherung, wie wir sie in Abbildung 7.66 gezeigt haben, aus, um in der Praxis den potenziellen Nutzen fast voll auszuschöpfen. Das Problem ist wieder der lange Ausläufer der Popularität. Studien zum Webverkehr haben gezeigt, dass die gemeinsame Nutzung eines Caches nur so lange von Vorteil ist, wie die Anzahl der Benutzer nicht die Größe eines kleinen Unternehmens erreicht (sagen wir 100 Personen). Je mehr Benutzer den Cache nutzen, desto geringer wird sein Nutzen, da die Anfragen nach weniger populären Seiten aufgrund von Speichermangel nicht zwischengespeichert werden können (Wolman et al., 1999).

Webproxys bieten zusätzliche Vorteile, die oft ein wichtiger Entscheidungsfaktor für ihren Einsatz sind. Ein Vorteil ist, dass sie Inhalte filtern. Der Administrator kann den Proxy so konfigurieren, dass er bestimmte Sites auf eine schwarze Liste stellt oder Anfragen anderweitig filtert. Viele Administratoren mögen es beispielsweise gar nicht, wenn Angestellte in ihrer Arbeitszeit auf YouTube surfen (oder noch schlimmer, Pornografie schauen), und setzen ihre Filter entsprechend. Ein weiterer Vorteil ist die Privatsphäre oder Anonymität, die die Proxys gewährleisten, da sie die Identität des Benutzers vor dem Server verbergen.

7.5.3 Content-Delivery-Netze

Serverfarmen und Webproxys helfen, große Websites zu erstellen und die Webleistung zu verbessern. Sie reichen aber nicht aus, wenn es um extrem populäre Websites geht, die ihren Inhalt global bereitstellen. Für diese Sites wird ein anderer Ansatz benötigt.

CDNs (*Content Delivery Network*) stellen die Idee der traditionellen Zwischenspeicherung im Web auf den Kopf. Anstatt dass die Clients nach einer Kopie der angefragten Seite in einem nahe gelegenen Cache suchen, ist es der Provider, der eine Kopie der angeforderten Seite in mehreren Knoten an verschiedenen Orten ablegt und den Client dazu bringt, den nächstgelegenen Knoten als Server zu verwenden.

Ein Beispiel für den Pfad, dem die Daten folgen, wenn sie über ein CDN verteilt sind, sehen Sie in ►Abbildung 7.67. Es ist ein Baum. Der ursprüngliche Server in dem CDN verteilt eine Kopie des Inhalts auf andere Knoten im CDN, die sich in diesem Beispiel in Sydney, Boston und Amsterdam befinden. Angezeigt wird dies durch die gestrichelten Linien. Clients holen sich dann die Seiten von dem nächstgelegenen Knoten in dem CDN. Dies wird durch durchgezogene Linien symbolisiert. Hierbei holen die beiden Clients aus Sydney die Seitenkopie, die in Sydney gespeichert ist. Sie holen die Seite nicht von dem ursprünglichen Server, der in Europa stehen kann.

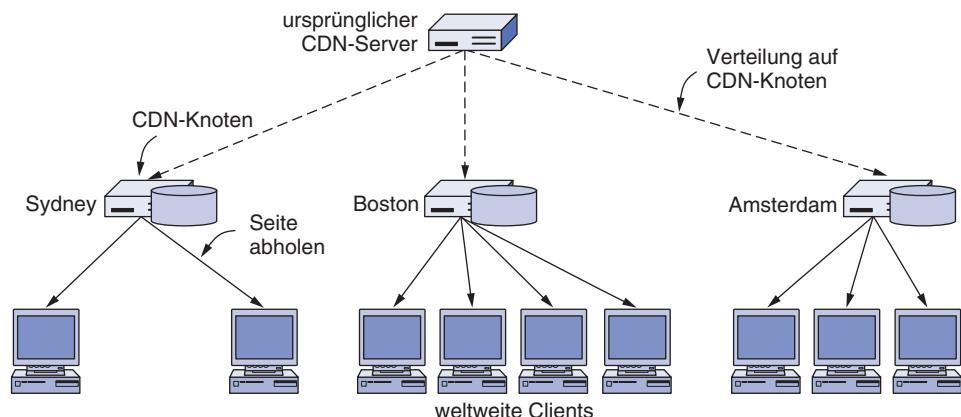


Abbildung 7.67: CDN-Verteilungsbaum.

Die Verwendung einer Baumstruktur hat drei Vorteile. Erstens kann die Verteilung der Inhalte auf beliebig viele Clients ausgedehnt werden, indem weitere Knoten in das CDN eingefügt werden beziehungsweise indem weitere Ebenen im Baum angelegt werden, wenn die Verteilung auf die CDN-Knoten zum Flaschenhals wird. Egal wie viele Clients es gibt – die Baumstruktur ist effizient. Der ursprüngliche Server ist nicht überlastet, da er über den Baum der CDN-Knoten mit den vielen Clients spricht; er muss nicht jede Anfrage nach einer Seite selbst beantworten. Zweitens, jeder Client erhält eine gute Leistung, da er die Seiten von einem nahe gelegenen Server abruft, anstatt von einem weit entfernten Server. Dementsprechend ist die Umlaufzeit zum Aufbau einer Verbindung kürzer, der TCP-Slow-Start läuft aufgrund der kürzeren Umlaufzeit schneller an und die Wahrscheinlichkeit sinkt, dass der kürzere Netzpfad durch überlastete Bereiche im Internet führt. Und schließlich wird die Gesamtlast auf dem Netz ebenfalls minimiert. Wenn die CDN-Knoten gut platziert sind, sollte der Verkehr für eine angeforderte Seite über jeden Teil des Netzes nur einmal laufen. Dies ist wichtig, da am Ende irgendwer für die Netzbандbreite zahlen muss.

Die Idee, einen Verteilungsbaum zu verwenden ist unkompliziert. Weniger einfach ist hingegen, wie die Clients zu organisieren sind, die diesen Baum verwenden. Proxyserver scheinen sich beispielsweise als Lösung anzubieten. Wenn in Abbildung 7.67 jeder Client konfiguriert würde, die CDN-Knoten Amsterdam, Boston oder Sydney als zwischenspeichernden Webproxy zu verwenden, würde die Verteilung diesem Baum folgen. Allerdings ist diese Strategie in der Praxis aus drei Gründen unbefriedigend. Der eine Grund ist, dass die Clients in einem gegebenen Teil des Netzes wahrscheinlich unterschiedlichen Organisationen angehören, sodass sie wahrscheinlich verschiedene Webproxys verwenden. Dabei sei noch einmal darauf hingewiesen, dass Caches normalerweise von einer und nicht von mehreren Organisationen genutzt werden, da zum einen der Nutzen der Zwischenspeicherung bei zunehmender Zahl der Clients sinkt und zum anderen Sicherheitsfragen zu bedenken sind. Zweitens kann es mehrere CDNs geben, wobei jeder Client aber nur einen einzigen Proxy-Cache verwendet. Da stellt sich die Frage, welches CDN ein Client als Proxy verwenden sollte. Und schließlich das aus Praxissicht vielleicht größte Problem: Die Webproxys werden von den Clients konfiguriert. Sie können so konfiguriert sein, dass sie von der Content-Verteilung eines CDN profitieren, müssen es aber nicht – und es gibt nichts, was das CDN dagegen machen kann.

Ein anderer einfacher Weg, einen Verteilungsbaum mit einer Ebene zu unterstützen, ist die Verwendung des **Spiegelns** (*mirroring*). Bei diesem Ansatz repliziert der ursprüngliche Server wie zuvor den Inhalt über die CDN-Knoten. Die CDN-Knoten in den verschiedenen Netzbereichen werden **Spiegel** (*mirror*) genannt. Die Webseiten auf dem ursprünglichen Server enthalten explizite Links zu den verschiedenen Spiegeln, die dem Benutzer normalerweise den Ort dieses Knotens verraten. Bei diesem Entwurf kann der Benutzer von Hand einen nahe gelegenen Spiegel zum Herunterladen des Inhalts auswählen. Normalerweise wird hierbei ein großes Softwarepaket auf Spiegeln gespeichert, die sich beispielsweise an der Ost- und Westküste der Vereinigten Staaten, in Asien und Europa befinden. Gespiegelte Sites sind im Allgemeinen

völlig statisch und die Auswahl an Sites bleibt für Monate, wenn nicht sogar Jahre unverändert. Diese Technik ist erprobt und hat sich bewährt. Allerdings setzt sie voraus, dass der Benutzer die Verteilung übernimmt, da die Spiegel in Realität verschiedene Websites sind, auch wenn sie miteinander verlinkt sind.

Der dritte Ansatz, der die Schwierigkeiten der ersten beiden Ansätze überwunden hat, verwendet das DNS (Domain Name System) und wird als **DNS-Umleitung** (*DNS redirection*) bezeichnet. Angenommen, ein Client möchte eine Seite mit der URL `http://www.cdn.com/page.html` abrufen. Um diese Seite zu holen, verwendet der Browser das DNS, um `www.cdn.com` zu einer IP-Adresse aufzulösen. Diese DNS-Suche erfolgt in gewohnter Manier. Über das DNS-Protokoll erfährt der Browser die IP-Adresse des Nameservers für `cdn.com`, den er anschließend kontaktiert, um ihn aufzufordern, `www.cdn.com` aufzulösen. Und jetzt das Clevere daran: Der Nameserver wird von dem CDN ausgeführt. Anstatt für jede Anfrage die gleiche IP-Adresse zurückzuliefern, schaut er auf die IP-Adresse des Clients, von dem die Anfrage kommt, und liefert entsprechend verschiedene Antworten zurück. Die Antwort ist die IP-Adresse des CDN-Knotens, der dem Client am nächsten liegt. Das heißt, wenn ein Client in Sydney den CDN-Nameserver auffordert, `www.cdn.com` aufzulösen, wird der Nameserver die IP-Adresse des CDN-Knotens in Sydney zurückliefern. Wenn jedoch ein Client in Amsterdam die gleiche Anfrage schickt, liefert der Nameserver die IP-Adresse des CDN-Knotens in Amsterdam zurück.

Diese Strategie ist gemäß der Semantik des DNS absolut legal. Wir haben bereits erfahren, dass Nameserver sich ändernde Listen von IP-Adressen zurückliefern können. Nach der Namensauflösung ruft der Sydney-Client die Seite direkt vom CDN-Knoten in Sydney ab. Weitere Seiten auf dem gleichen „Server“ werden aufgrund der DNS-Zwischenspeicherung ebenfalls direkt vom CDN-Knoten in Sydney abgeholt. Die gesamte Schrittfolge sehen Sie in ▶ Abbildung 7.68.

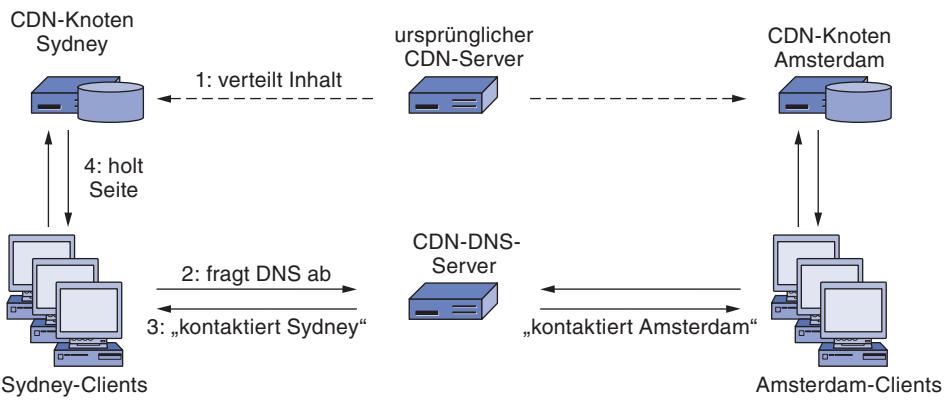


Abbildung 7.68: Clients mittels DNS zu den nächstgelegenen CDN-Knoten umleiten.

Eine komplexe Frage im obigen Prozess ist, was hier „nächstgelegener CDN-Knoten“ bedeutet und wie er ermittelt wird. Bei der Definition von „nächstgelegen“ spielt nicht unbedingt Geografie eine Rolle. Es gibt mindestens zwei Faktoren, die beim Abbilden eines Clients auf einen CDN-Knoten berücksichtigt werden müssen. Einer der Faktoren

ist die Entfernung im Netz. Der Client sollte einen kurzen Netzwerkpfad hoher Kapazität zum CDN-Knoten aufweisen. Das garantiert kurze Download-Zeiten. CDNs verwenden eine Zuordnungsliste, die sie zuvor berechnet haben, um zwischen der IP-Adresse eines Clients und seiner Position im Netz zu übersetzen. Der ausgewählte CDN-Knoten kann der sein, der per Luftlinie am nächsten gelegen ist oder nicht. Es zählt eine Kombination aus Länge und Kapazitätsbeschränkungen des Netzpfades. Der zweite Faktor ist die Last, die der CDN-Knoten bereits trägt. Wenn die CDN-Knoten überlastet sind, antworten sie nur langsam, wie beim überlasteten Webserver, den wir eigentlich zu vermeiden suchen. Folglich mag es notwendig sein, die Last auf die CDN-Knoten gleichmäßig zu verteilen, indem einige Clients Knoten zugeordnet werden, die zwar etwas weiter weg liegen, aber dafür weniger stark belastet sind.

Die Techniken, die entwickelt wurden, um das DNS zur Auslieferung von Inhalten zu verwenden, wurden von Akamai erstmalig 1998 kommerziell eingesetzt, als das Web unter der Last seines Anfangswachstums stöhnte. Akamai war der erste bedeutende CDN-Provider und wurde schnell zum Branchenführer. Wahrscheinlich noch schlauer als die Idee, Clients mithilfe des DNS mit nahe gelegenen Knoten zu verbinden, war die Anreizstruktur ihrer Geschäfte. Firmen zahlen Akamai für die Zustellung ihrer Inhalte an Clients, was die Antwortzeiten ihrer Websites beschleunigt, sodass die Kunden sie gerne wieder besuchen. Die CDN-Knoten müssen an Netzpositionen mit guter Konnektivität platziert werden, was anfänglich bedeutete: innerhalb des ISP-Netzes. Für ISPs ist ein CDN-Knoten in ihren Netzwerken nur von Vorteil, da der CDN-Knoten die Upstream-Netzbandbreite reduziert, die sie benötigen (und bezahlen müssen), genau wie es bei Proxy-Cache der Fall ist. Außerdem verbessert der CDN-Knoten die Reaktionsgeschwindigkeit für die ISP-Kunden, was den ISP für Kunden attraktiv macht und ihm Vorteile gegenüber anderen ISPs verschafft, die keinen CDN-Knoten haben. Angesichts dieser Vorteile (die keinen Cent kosten) bedarf die Installation eines CDN-Knotens keiner großen Überlegung seitens eines ISP. Es profitieren sozusagen alle: der Content-Provider, der ISP und die Kunden – und der CDN-Provider macht das Geld. Seit 1998 sind weitere Unternehmen in das Geschäft eingestiegen, sodass es heute eine wettbewerbsfähige Branche mit mehreren Providern ist.

Wie diese Beschreibung zeigt, bauen die meisten Firmen kein eigenes CDN auf, sondern nutzen stattdessen die Dienste eines CDN-Providers wie Akamai, um ihre Inhalte tatsächlich auszuliefern. Damit andere Firmen die Dienste eines CDN nutzen können, müssen wir noch einen letzten Schritt hinzufügen.

Nachdem der Vertrag unterzeichnet ist, dass ein CDN den Inhalt im Auftrag eines Website-Betreibers ausliefern darf, übergibt der Betreiber diesen Inhalt an das CDN. Dieser Content wird in die CDN-Knoten eingespeist. Zusätzlich überarbeitet der Betreiber alle seine Webseiten mit einem Link auf den Inhalt. Anstatt eine Verlinkung zu dem Inhalt auf ihrer Website herzustellen, werden die Seiten über das CDN mit dem Inhalt verlinkt. Lassen Sie uns als Beispiel den Quellcode für die *Fluffy Video*-Webseite betrachten, den Sie in ► Abbildung 7.69a finden. Nach der Bearbeitung sieht der Quellcode aus wie in ► Abbildung 7.69b und wird auf dem Server von *Fluffy Video* unter www.fluffyvideo.com/index.html gespeichert.

```

<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Fluffy Video's Product List </h1>
<p> Click below for free samples. </p>

<a href="koalas.mpg"> Koalas Today </a> <br>
<a href="kangaroos.mpg"> Funny Kangaroos </a> <br>
<a href="wombats.mpg"> Nice Wombats </a> <br>
</body>
</html>

```

a

```

<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Fluffy Video's Product List </h1>
<p> Click below for free samples. </p>

<a href="http://www.cdn.com/fluffyvideo/koalas.mpg"> Koalas Today </a> <br>
<a href="http://www.cdn.com/fluffyvideo/kangaroos.mpg"> Funny Kangaroos </a> <br>
<a href="http://www.cdn.com/fluffyvideo/wombats.mpg"> Nice Wombats </a> <br>
</body>
</html>

```

b

Abbildung 7.69: (a) Eine ursprüngliche Webseite. (b) Gleiche Webseite nach der Verlinkung zu dem CDN.

Wenn ein Benutzer der URL `www.fluffyvideo.com` in seinem Browser eingibt, liefert das DNS die IP-Adresse der eigenen Website von *Fluffy Video* zurück, wobei ganz normal die Hauptseite (HTML) abgerufen wird. Wenn der Benutzer auf einen der Hyperlinks klickt, fordert der Browser das DNS auf, `www.cdn.com` nachzuschlagen. Diese Suche kontaktiert den DNS-Server des CDN, der die IP-Adresse des nächstgelegenen CDN-Knotens zurückliefert. Dann sendet der Browser eine reguläre HTTP-Anfrage an den CDN-Knoten, z.B. für `/fluffyvideo/koalas.mpg`. Die URL identifiziert die zurückzuliefernde Seite, wobei der Pfad mit `fluffyvideo` beginnt, damit der CDN-Knoten die Anfragen für die verschiedenen Unternehmen, deren Inhalten er ausliefert, unterscheiden kann. Am Ende wird das Video zurückgeliefert und der Benutzer sieht niedliche kleine Kuscheltierchen.

Die Strategie hinter dieser Trennung von Inhalt, der vom CDN-Provider gehostet wird, und Einstiegsseiten, die vom Content-Besitzer gehostet werden, ist, dass der Content-Besitzer die Kontrolle behält, während das CDN die Masse der Daten überträgt. Die meisten Einstiegsseiten sind winzig, da sie nur aus HTML-Text bestehen. Diese Seiten enthalten oft Links zu großen Dateien wie Videos oder Bilder. Und es sind genau diese großen Seiten, die vom CDN ausgeliefert werden, auch wenn die Nutzung eines CDN absolut transparent für die Benutzer ist. Die Website sieht genauso aus, ist aber wesentlich schneller.

Es gibt für Websites, die sich mit anderen Sites ein CDN teilen, noch einen weiteren Vorteil. Es kann schwer sein, die zukünftige Nachfrage nach einer Website vorauszusagen. Häufig kommt es zu einem plötzlichen Anstieg der Nachfrage, die als **Flash Crowd** bezeichnet wird. Ein solcher Ansturm könnte erfolgen, wenn das neueste Produkt auf den Markt kommt, eine Modenschau oder ein anderes Ereignis organisiert wird oder das Unternehmen aus anderen Gründen in den Nachrichten ist. Sogar Websites, die zuvor unbekannt waren und kaum Besucher hatten, können plötzlich ins Zentrum des Interesses rücken, wenn sie aktuelle Inhalte bieten und von beliebten Sites verlinkt sind. Da die meisten Websites nicht darauf vorbereitet sind, solch einen starken Verkehrsanstieg zu bewältigen, stürzen viele von ihnen ab, wenn der Verkehr extrem ansteigt.

Durch die Verwendung eines CDN hat eine Website Zugang zu einer sehr großen Content-Auslieferungskapazität. Die größten CDNs haben überall auf der Welt Zehntausende von Servern installiert. Da (per Definition) nur eine kleine Zahl von Sites einen Flash Crowd erleben werden, können diese Sites die CDN-Kapazität nutzen, um die Last zu bewältigen, bis der Ansturm abklingt. Das heißt, das CDN kann schnell die Auslieferungskapazität einer Website hochfahren.

Die obige Diskussion beschreibt etwas vereinfacht die Funktionsweise von Akamai. In der Praxis spielen jedoch noch viele andere Faktoren eine Rolle. Die CDN-Knoten in unserem Beispiel sind normalerweise Rechner-Cluster. Die DNS-Umleitung erfolgt auf zwei Ebenen: Eine, um Clients auf ihren ungefähren Netzstandort abzubilden, und eine andere, um die Last auf die Knoten an diesem Standort zu verteilen. Wichtig sind sowohl Zuverlässigkeit als auch Leistung. Um einen Client von einem Rechner in einem Cluster zu einem anderen verschieben zu können, werden die DNS-Antworten auf der zweiten Ebene mit kurzen TTLs gegeben, sodass der Client die Auflösung nach einer kleinen Weile wiederholt. Zum Schluss sei erwähnt, dass CDNs auch die Erzeugung dynamischer Seiten, Streaming Media und mehr unterstützen, während wir uns hier auf die Auslieferung von statischen Objekten wie Bildern und Videos konzentriert haben. Weitere Informationen über CDNs finden Sie bei Dilley et al. (2002).

7.5.4 Peer-to-Peer-Netze

Nicht jeder kann einen 1 000-Knoten-CDN an Orten rund um die Welt einrichten, um seine Inhalte auszuliefern. (Eigentlich ist es dank der gut aufgestellten und konkurrenzstarken Hosting-Industrie nicht besonders schwer, weltweit 1 000 virtuelle Rechner zu mieten. Die Anmietung der Knotenkapazität ist jedoch nur der Anfang beim Einrichten eines CDN.) Zum Glück gibt es für uns andere eine Alternative, die leicht zu verwenden ist und enorme Mengen von Inhalten übertragen kann: das **P2P-Netz (Peer-to-Peer)**.

P2P-Netze hatten 1999 ihren großen Durchbruch. Ihre erste breite Anwendung diente der Massenkriminalität: 50 Millionen Napster-Nutzer tauschten urheberrechtlich geschützte Songs aus, ohne die Erlaubnis der Rechteinhaber einzuholen, bis Napster unter großen Auseinandersetzungen gerichtlich geschlossen werden musste. Nichts-

destotrotz hat die Peer-to-Peer-Technologie viele interessante und legale Anwendungsbereiche. Andere Systeme griffen die Technik auf und entwickelten sie weiter, was auf so breites Interesse bei den Nutzern stieß, dass der P2P-Verkehr den Webverkehr schnell in den Schatten stellte. Derzeit ist BitTorrent das beliebteste P2P-Protokoll. Es wird so stark für die Verteilung von (lizenzierten und lizenzzfreien) Videos und anderen Inhalten eingesetzt, dass es einen großen Anteil am gesamten Internetverkehr ausmacht. Wir wollen uns in diesem Abschnitt etwas eingehender damit beschäftigen.

Ein P2P-Dateiaustauschnetz basiert auf dem Grundgedanken, dass viele Rechner sich zusammenschließen und ihre Ressourcen bündeln, um ein System zum Verteilen von Inhalten (*content distribution system*) zu bilden. Bei den Rechnern handelt es sich oft um einfache PCs; es müssen keine Rechner in Internetdatenzentren sein. Die Rechner werden Peers genannt, weil jeder sowohl als Client auftreten kann, der von einem anderen Peer Inhalte abruft, als auch als Server dienen kann, der anderen Peers Inhalte bereitstellt. Das Interessante an den Peer-to-Peer-Systemen ist, dass es keine vorgegebene Infrastruktur gibt, wie es bei den CDNs der Fall ist. Jeder wird in die Verteilung von Inhalten mit eingebunden und oft gibt es keinen zentralen Kontrollpunkt.

Viele sind von der P2P-Technologie begeistert, weil vor allem der kleine Mann von der Straße davon profitiert. Der Grund ist nicht nur, dass ein großes Unternehmen erforderlich ist, um ein CDN zu unterhalten, während jeder mit einem Rechner einem P2P-Netz beitreten kann, sondern auch, dass P2P-Netze eine beträchtliche Kapazität haben, über die sie Inhalte verteilen können, die es durchaus mit den größten Web-sites aufnehmen.

Betrachten wir beispielsweise ein P2P-Netz mit N durchschnittlichen Benutzern, die jeweils über einen Breitbandanschluss von 1 Mbit/s verfügen. Dann beträgt die Gesamtkapazität des P2P-Netzes für das Hoch- bzw. Herunterladen sowie die Übertragungsrate zum Senden und Empfangen im Internet genau N Mbit/s. Außerdem können alle Benutzer gleichzeitig Daten hoch- und herunterladen, da jeder eine 1-Mbit/s-Verbindung in jede Richtung aufweist.

Es ist nicht direkt ersichtlich, dass dies der Wahrheit entspricht, aber es stellt sich heraus, dass die ganze Kapazität produktiv zum Verteilen des Inhalts verwendet werden kann, sogar für den Fall, dass eine einzelne Kopie einer Datei mit allen anderen Nutzern ausgetauscht wird. Um zu zeigen, wie dies geht, stellen Sie sich vor, dass die Benutzer in einem Binärbaum organisiert sind, bei dem jeder Benutzer, der nicht Blatt ist, die Kopie an zwei andere Benutzer sendet. Der Baum überträgt die einzelne Kopie der Datei an alle anderen Benutzer. Um die Upload-Bandbreite von möglichst vielen Benutzern jederzeit zu nutzen (und somit die große Datei mit geringer Latenz zu verteilen), müssen wir die Netzaktivität der Benutzer lenken. Stellen Sie sich weiterhin vor, die Datei besteht aus 1 000 Einzelteilen. Jeder Benutzer kann ein neues Teil von irgendwoher über sich im Baum empfangen und gleichzeitig das zuvor empfangene Teil im Baum nach unten weiterschicken. Sobald also die Pipeline gestartet ist und eine kleine Anzahl von Teilen (entsprechend der Tiefe des Baums) geschickt wurde, werden alle Benutzer, die nicht Blatt sind, beschäftigt sein, die Datei anderen Nutzern hochzuladen. Da es ungefähr $N/2$ Benutzer gibt, die nicht Blatt sind, beträgt die

Upload-Bandbreite dieses Baums $N/2$ Mbit/s. Wir können diesen Trick wiederholen und einen weiteren Baum erzeugen, der die anderen $N/2$ Mbit/s Upload-Bandbreite nutzt, indem wir die Rollen der Blatt- und Nicht-Blattknoten tauschen. Zusammen nutzt diese Konstruktion die gesamte Kapazität.

Dieses Argument bedeutet, dass P2P-Netze selbstskalierend sind. Ihre nutzbare Upload-Kapazität wächst zusammen mit der potenziellen Download-Nachfrage ihrer Nutzer. Sie sind in einem gewissen Sinne immer „groß genug“, ohne dass es dafür einer bestimmten Infrastruktur bedarf. Im Gegensatz dazu steht die Kapazität sogar einer großen Website fest und ist entweder zu groß oder zu klein. Betrachten wir dazu eine Website mit nur 100 Clustern, die jeweils eine Kapazität von 10 Gbit/s haben. Diese enorme Kapazität hilft nicht, wenn es nur wenige Nutzer gibt. Die Website kann die Daten an N Benutzer nicht schneller übertragen als N Mbit/s, da die Beschränkung bei den Nutzern liegt und nicht bei der Website. Und wenn es mehr als eine Million 1-Mbit/s-Nutzer gibt, kann die Website die Daten nicht schnell genug ausscheiden, um alle Nutzer mit dem Herunterladen zu beschäftigen. Das klingt nach einer hohen Anzahl Nutzer, aber große BitTorrent-Netze (z.B. Pirate Bay) behaupten, mehr als 10 000 000 Nutzer zu haben. Das entspräche in unserem Beispiel mehr als 10 Tbit/s!

Sie sollten diese Überschlagszahlen mit (großer) Vorsicht genießen, da sie die Situation etwas zu stark vereinfachen. Eine bedeutende Herausforderung für P2P-Netze ist es, die Bandbreite auch dann gut zu nutzen, wenn die Nutzer in allen Formen und Größen kommen und unterschiedliche Upload- und Download-Kapazitäten haben. Dennoch demonstrieren diese Zahlen das ungeheure Potenzial von P2P.

Es gibt einen weiteren Grund, warum P2P-Netze wichtig sind. Die CDNs und andere zentral bereitgestellte Dienste liefern den Providern einen wahren Schatz an persönlichen Daten – von den bevorzugt angesteuerten Websites und den genutzten Onlineshops, bis zu den Wohnorten und E-Mail-Adressen der Nutzer. Diese Daten können für einen besseren, personalisierten Dienst genutzt werden oder um die Privatsphäre der Nutzer auszukundschaften. Letzteres kann absichtlich erfolgen – sagen wir als Teil eines neuen Produkts – oder durch unabsichtliche Enthüllung oder Gefährdung. Bei P2P-Systemen gibt es keinen solchen Provider, der das ganze System beobachten kann. Das bedeutet nicht, dass P2P-Systeme notwendigerweise Datenschutz bieten, da die Nutzer sich im gewissen Maße trauen. Es bedeutet nur, dass sie eine andere Form von Datenschutz bieten können als zentral verwaltete Systeme. Zurzeit wird untersucht, inwieweit sich P2P-Systeme auch für andere Dienste außer Datenauschutzen lassen (z.B. Speicherung, Streaming), und erst die Zeit wird zeigen, ob dieser Vorteil von Bedeutung ist.

Die P2P-Technologie hat bei ihrer Entwicklung zwei verwandte Wege eingeschlagen. Auf der eher praktischen Seite gibt es die Systeme, die jeden Tag verwendet werden. Die bekanntesten Systeme basieren auf dem BitTorrent-Protokoll. Auf der eher akademischen Seite gibt es ein großes Interesse an DHT-Algorithmen (*Distributed Hash Table*, verteilte Hash-Tabelle), die dafür sorgen, dass die P2P-Systeme gute Leistung zeigen, ohne auf zentralisierte Komponenten angewiesen zu sein. Wir werden im Folgenden beide Technologien kurz vorstellen.

BitTorrent

Das BitTorrent-Protokoll wurde 2001 von Bram Cohen entwickelt und sollte den Austausch von Dateien zwischen mehreren Peers beschleunigen und erleichtern. Es gibt Dutzende von frei verfügbaren Clients, die dieses Protokoll „sprechen“, genauso wie es viele Browser gibt, die das HTTP-Protokoll „sprechen“, um mit Webservern zu kommunizieren. Das Protokoll ist unter www.bittorrent.org als offener Standard erhältlich.

In einem typischen Peer-to-Peer-System, wie es mit BitTorrent gebildet wird, verfügt jeder Benutzer über Daten, die für andere Benutzer von Interesse sein können. Hierbei kann es sich um freie Software, Musik, Videos, Fotos usw. handeln. Vor dem Austauschen von Inhalten müssen zuerst drei Probleme gelöst werden:

- 1.** Wie findet ein Peer andere Peers, die den Inhalt anbieten, den er gerne herunterladen möchte?
- 2.** Wie wird Inhalt von Peers repliziert, um jedem ein superschnelles Herunterladen zu bieten?
- 3.** Wie animieren Peers einander, Inhalte für andere hochzuladen bzw. für sich selbst herunterzuladen?

Das erste Problem besteht darin, dass zumindest am Anfang nicht alle Peers alle Inhalte haben. Der Ansatz, den BitTorrent verfolgt, besteht darin, dass jeder Content-Provider eine Content-Beschreibung erzeugt, die als **Torrent** (Sturzbach) bezeichnet wird. Dieser Torrent ist viel kleiner als der Inhalt und wird von einem Peer verwendet, um die Integrität der Daten zu verifizieren, die er von anderen Peers herunterlädt. Andere Benutzer, die den Inhalt herunterladen wollen, müssen sich zuerst den Torrent besorgen, indem sie ihn beispielsweise auf einer der Webseiten suchen, die für den Inhalt wirbt.

Torrents sind eigentlich nur Dateien in einem speziellen Format, die zwei wichtige Informationen enthalten: den Name eines **Trackers**, hinter dem sich ein Server verbirgt, der die Peers zum Inhalt des Torrents führt, und eine Liste gleich großer Teile oder **Segmente** (*chunk*), die den Inhalt bilden. Für verschiedene Torrents können verschiedene Segmentgrößen verwendet werden (normalerweise 64 KB bis 512 KB). Die Torrent-Datei enthält die Namen aller Segmente, und zwar in Form von 160-Bit SHA-1-Hashes. Wir werden auf das Thema der kryptografischen Hashes wie SHA-1 in Kapitel 8 näher eingehen. Im Moment reicht es, wenn Sie sich einen Hash als eine Art längere und sicherere Prüfsumme vorstellen. Da die Torrent-Datei nur die Segmentgröße und die Hashes enthält, ist sie drei Größenordnungen kleiner als der Inhalt, sodass sie schnell übertragen werden kann.

Um den Inhalt herunterzuladen, der in einem Torrent beschrieben wurde, kontaktiert ein Peer zuerst den Tracker für den Torrent. Der **Tracker** ist ein Server, der eine Liste aller anderen Peers verwaltet, die den Inhalt aktiv hoch- und herunterladen. Diese Peergruppe wird **Schwarm** (*swarm*) genannt. Die Mitglieder des Schwarms kontaktie-

ren den Tracker regelmäßig, um mitzuteilen, dass sie immer noch aktiv sind, bzw. wenn sie den Schwarm verlassen. Wenn ein neuer Peer den Tracker kontaktiert, um dem Schwarm beizutreten, erfährt er vom Tracker, welche anderen Peers dem Schwarm angehören. Sich die Torrent-Datei zu besorgen und den Tracker zu kontaktieren, sind die ersten beiden Schritte, um Inhalt herunterzuladen (► Abbildung 7.70).

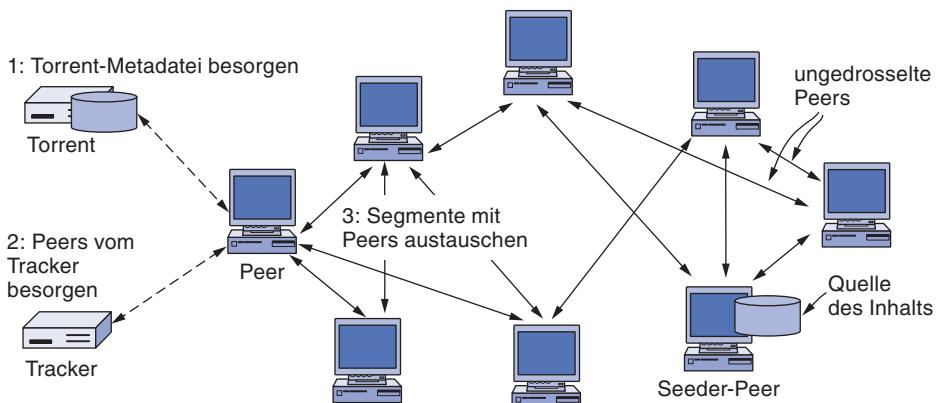


Abbildung 7.70: BitTorrent.

Das zweite Problem, das sich stellt, ist, wie sich beim Datenaustausch schnelle Download-Zeiten erreichen lassen. Am Anfang, wenn sich ein Schwarm bildet, haben einige Peers alle Segmente, aus denen der Inhalt besteht. Diese Peers werden **Seeder** (von *seed*, säen) genannt. Andere Peers, die dem Schwarm beitreten, haben keine Segmente. Dieses sind die Peers, die den Inhalt herunterladen.

Peers, die einem Schwarm angehören, laden fehlende Segmente von anderen Peers im Schwarm herunter und gleichzeitig Segmente zu anderen Peers hoch, die diese benötigen. Dieser Austausch ist in Abbildung 7.70 als letzter Schritt der Content-Verteilung dargestellt. Mit der Zeit sammelt der Peer weitere Segmente, bis er den ganzen Inhalt heruntergeladen hat. Der Peer kann den Schwarm jederzeit verlassen (und wieder zurückkehren). Normalerweise bleibt ein Peer noch kurz im Schwarm, nachdem er seinen eigenen Download beendet hat. Das ständige Kommen und Gehen der Peers kann die Fluktuationsrate in einem Schwarm relativ hochtreiben.

Damit obige Methode gut funktioniert, sollte jedes Segment bei möglichst vielen Peers zur Verfügung stehen. Wenn jeder Peer die Segmente in der gleichen Reihenfolge erhalten soll, würden wahrscheinlich viele Peers das nächste Segment von den Seedern erwarten. Dies hätte einen Engpass zur Folge. Um dieses Problem zu umgehen, tauschen Peers untereinander Listen ihrer Segmente aus. Anschließend wählen sie zum Herunterladen die Segmente, die nur schwer zu finden sind. Der Gedanke dahinter ist, dass das Herunterladen eines seltenen Segments eine Kopie erstellt, die dazu beiträgt, dass andere Peers das Segment leichter finden und herunterladen können. Wenn dies alle Peers machen, sind nach kurzer Zeit alle Segmente bei allen Peers im Schwarm verfügbar.

Das dritte Problem ist wahrscheinlich das interessanteste. CDN-Knoten dienen ausschließlich der Bereitstellung von Inhalten, P2P-Knoten nicht. P2P-Knoten werden aus den Rechnern der Benutzer gebildet, die unter Umständen stärker daran interessiert sind, Filme herunterzuladen, als anderen Benutzern mit ihren Downloads zu helfen. Knoten, die nur Ressourcen aus einem System ziehen, ohne selbst Beiträge zu leisten, werden **Trittbrettfahrer** oder **Leecher** (Blutegel) genannt. Wenn es zu viele von ihnen gibt, funktioniert das System nicht mehr gut. Bekanntlich hosteten frühere P2P-Systeme solche Peers (Saroui et al., 2003), sodass BitTorrent bemüht war, ihre Zahl zu minimieren.

In BitTorrent-Clients wird der Ansatz verfolgt, Peers zu belohnen, die gutes Upload-Verhalten zeigen. Jeder Peer wählt willkürlich die Peers, von denen er Segmente herunterlädt, während er gleichzeitig Segmente hochlädt. Mit einigen wenigen, die die beste Download-Leistung aufweisen, tauscht er dann die Daten, während er gleichzeitig weitere Peers testet, um gute Partner zu finden. Das zufällige Ausprobieren von Peers erlaubt außerdem Neulingen, sich erste Segmente zu besorgen, die sie dann mit anderen Peers tauschen können. Die Peers, mit denen ein Knoten gerade Segmente austauscht, befinden sich im sogenannten **Unchoked**-Zustand (ungedrosselt).

Mit der Zeit soll dieser Algorithmus Peers, die vergleichbare Upload- und Download-Raten haben, einander zuordnen. Je mehr Hilfe ein Peer anderen Peers gegenüber leistet, desto mehr Hilfe kann er erwarten. Eine Gruppe von Peers zu verwenden, trägt außerdem dazu bei, die Download-Bandbreite eines Peers zum Zwecke einer sehr guten Leistung voll auszunutzen. Umgekehrt wird ein Peer, der keine Segmente auf andere Peers hochlädt oder dies nur sehr langsam macht, früher oder später abgeschnitten oder gedrosselt (**Choked**-Zustand). Diese Strategie soll von einem antisozialen Verhalten abhalten, bei dem Peers nur Nutznießer eines Schwarms sind.

Der Choke-Algorithmus wird manchmal auch als Implementierung der **Geben-und-Nehmen**-Strategie beschrieben, der zur Zusammenarbeit in wiederholten Interaktionen anregt. Er hindert Clients jedoch nicht daran, das System zu überlisten (Piatek et al., 2007). Die Aufmerksamkeit, die diesem Thema geschenkt wird, und die Mechanismen, die es dem Gelegenheitsbenutzer schwerer machen, nur Nutznießer zu sein, haben sicher zum Erfolg von BitTorrent beigetragen.

Wie Sie unserer Diskussion entnehmen können, führt BitTorrent viele neue Begriffe ein. Es gibt Torrents, Schwärme, Leechers, Seeders und Trackers sowie Snubbing, Choking usw. Weitere Informationen finden Sie in dem kurzen Aufsatz zu BitTorrent (Cohen, 2003) und im Web unter www.bittorrent.org.

DHTs – verteilte Hash-Tabellen

Das Aufkommen der P2P-Datenaustauschnetze um die Jahrtausendwende weckte ein großes Interesse in der Forschungsgemeinde. Der Kern von P2P-Systemen ist, die zentral verwalteten Strukturen von CDNs und anderen Systemen zu vermeiden. Dies kann ein bedeutender Vorteil sein. Zentral verwaltete Komponenten werden zu einem Engpass, wenn das System sehr groß wird, und sind damit ein Single-Point-of-Failure. Zentrale Komponenten können auch als Kontrollpunkt verwendet werden (z.B. um

das P2P-Netz abzuschalten). Die frühen P2P-Systeme waren jedoch nur teilweise dezentralisiert bzw. wenn sie voll dezentralisiert waren, waren sie ineffizient.

Die traditionelle Form von BitTorrent, die wir hier gerade beschrieben haben, verwendet Peer-to-Peer-Übertragungen und einen zentralisierten Tracker für jeden Schwarm. Der Tracker scheint in einem Peer-to-Peer-System am schwersten zu dezentralisieren sein. Das Hauptproblem ist, herauszufinden, welche Peers den gesuchten speziellen Content haben. Zum Beispiel könnte jeder Benutzer ein oder mehr Datenelemente wie Songs, Fotos, Programme, Dateien usw. haben, die andere Benutzer gerne lesen oder sehen würden. Doch wie werden sie von diesen Benutzern gefunden? Das Erstellen eines Index, der angibt, wer was hat, ist einfach, aber er ist zentralisiert. Für jeden Peer einen eigenen Index anzulegen, ist auch keine Lösung. Er wäre dann zwar verteilt, aber es würde so viel Arbeit machen, die Indizes aller Peers aktuell zu halten (da Content im System verschoben wird), dass sich die Mühe nicht lohnt.

Die Forschungsgemeinde befasste sich mit der Frage, ob es möglich wäre, P2P-Indizes zu erstellen, die gänzlich verteilt, aber trotzdem leistungsstark sind. Mit leistungsstark meinen wir drei Dinge. Erstens, jeder Knoten speichert nur wenige Informationen über die anderen Knoten. Das bedeutet, dass es nicht sehr teuer ist, den Index aktuell zu halten. Zweitens, jeder Knoten kann schnell Einträge im Index nachschlagen. Ansonsten wäre er kein sehr nützlicher Index. Drittens, alle Knoten können den Index gleichzeitig nutzen, auch wenn andere Knoten kommen und gehen. Dies bedeutet, dass die Leistung des Index mit der Anzahl der Knoten wächst.

Die Antwort auf die Frage lautet „Ja“. 2001 wurden vier verschiedene Lösungen entwickelt. Sie lauten Chord (Stoica et al., 2001), CAN (Ratnasamy et al., 2001), Pastry (Rowstron und Druschel, 2001) und Tapestry (Zhao et al., 2004). Kurz danach kamen weitere Lösungen hinzu, einschließlich Kademia, die in der Praxis eingesetzt wird (Maymounkov und Mazières, 2002). Es handelt sich bei den Lösungen um verteilte Hash-Funktionen, die als **DHT** (*Distributed Hash Tables, verteilte Hash-Tabellen*) bezeichnet werden, da die Hauptaufgabe eines Index darin besteht, Schlüssel auf Werte abzubilden. Damit erhalten Sie eine Hash-Tabelle und die Lösungen sind natürlich verteilte Versionen.

DHTs geben der Kommunikation zwischen den Knoten eine regelmäßige Struktur. Dieses Verhalten unterscheidet sich damit ziemlich von dem der traditionellen P2P-Netze, die jede Verbindung nutzen, die die Peers aufbauen. Aus diesem Grund werden die DHTs auch **strukturierte P2P-Netze** genannt. Traditionelle P2P-Protokolle hingegen erstellen **unstrukturierte P2P-Netze**.

Die DHT-Lösung, die wir hier vorstellen, ist Chord. Stellen Sie sich vor, Sie wollten den zentralisierten Tracker, der normalerweise in BitTorrent verwendet wird, durch einen dezentralisierten Tracker ersetzen. Chord kann Ihnen dabei helfen. In diesem Szenario besteht der Gesamtindex aus einer Liste aller Schwärme, denen ein Rechner beitreten kann, um Inhalte herunterzuladen. Der Schlüssel, mit dem im Index gesucht wird, ist die Torrent-Beschreibung des Inhalts. Er identifiziert eindeutig einen Schwarm, von dem aus Inhalt heruntergeladen werden kann, als die Hashes aller

Inhaltssegmente. Der Wert, der im Index zu jedem Schlüssel gespeichert wird, ist die Liste der Peers, aus denen der Schwarm besteht. Diese Peers sind die Rechner, die zum Herunterladen des Inhalts kontaktiert werden können. Eine Person, die beispielsweise einen Film herunterladen möchte, hat nur die Torrent-Beschreibung. Die Frage, auf die DHTs eine Antwort liefern müssen, ist, wie eine Person ohne eine zentrale Datenbank herausfindet, welche Peers (von den Millionen BitTorrent-Knoten) den Film zum Herunterladen anbieten.

Chord besteht aus n teilnehmenden Knoten. In unserem Szenario sind es Knoten, die BitTorrent ausführen. Jeder Knoten hat eine IP-Adresse, über die er kontaktiert werden kann. Der Gesamtindex ist über die Knoten verteilt. Das bedeutet, dass jeder Knoten Teile vom Index speichert, die die anderen Knoten nutzen können. Die Hauptaufgabe von Chord ist das Navigieren im Index, wobei Identifikatoren in einem virtuellen Raum verwendet werden und keine IP-Adressen von Knoten oder Namen von Inhalten wie Filme. Vom Konzept her sind Identifikatoren einfach m -Bit-Zahlen, die in aufsteigender Reihenfolge in einem Ring angeordnet sind.

Um eine Knotenadresse in einen Identifikator zu verwandeln, wird ihr mit der Hash-Funktion $hash$ eine m -Bit-Zahl zugeordnet. Chord verwendet SHA-1 für $hash$. Dies ist der Hash, den wir bei der Beschreibung von BitTorrent erwähnt haben. Wir werden in *Kapitel 8* im Zusammenhang mit Kryptografie näher darauf eingehen. Hier soll es genügen zu erwähnen, dass es nur eine Funktion ist, die einen Byte-String variabler Länge als Argument übernimmt und eine extrem zufällig aussehende 160-Bit-Zahl erzeugt. Wir können damit also jede beliebige IP-Adresse in eine 160-Bit-Zahl, umwandeln, die **Knotenidentifikator** genannt wird.

In ▶ Abbildung 7.71a sehen Sie einen Ring von Knotenidentifikatoren für $m=5$. (Ignorieren Sie im Moment noch die Bögen in der Mitte.) Einige der Identifikatoren stehen für Knoten, die meisten jedoch nicht. In diesem Beispiel sind nur die Knoten mit den Identifikatoren 1, 4, 7, 12, 15, 20 und 27 tatsächliche Knoten (in der Abbildung farblich hervorgehoben); der Rest existiert nicht.

Lassen Sie uns dazu die Funktion $successor(k)$ als den Knotenidentifikator des ersten tatsächlichen Knotens definieren, der im Uhrzeigersinn im Kreis auf k folgt. Zum Beispiel ist $successor(6)=7$, $successor(8)=12$ und $successor(22)=27$.

Außerdem wird ein **Schlüssel** (key) erzeugt, indem aus einem Content-Namen mittels $hash$ (z.B. SHA-1) eine 160-Bit-Zahl erzeugt wird. In unserem Beispiel ist der Content-Name der Torrent. Um also $torrent$ (die Torrent-Beschreibungsdatei) in seinen Schlüssel umzuwandeln, berechnen wir $key=hash(torrent)$. Diese Berechnung ist nur ein lokaler Prozeduraufruf von $hash$.

Um einen neuen Schwarm zu starten, muss ein Knoten ein neues Schlüssel/Wert-Paar aus $(torrent, \text{ meine-IP-Adresse})$ in den Index einfügen. Hierzu bittet $successor(hash(torrent))$, meine-IP-Adresse zu speichern. Auf diese Weise wird der Index über die Knoten nach dem Zufallsprinzip verteilt. Zum Zwecke der Fehlertoleranz können mit p verschiedenen Hash-Funktionen die Daten in p Knoten gespeichert werden, doch wir wollen das Thema Fehlertoleranz hier nicht weiter vertiefen.

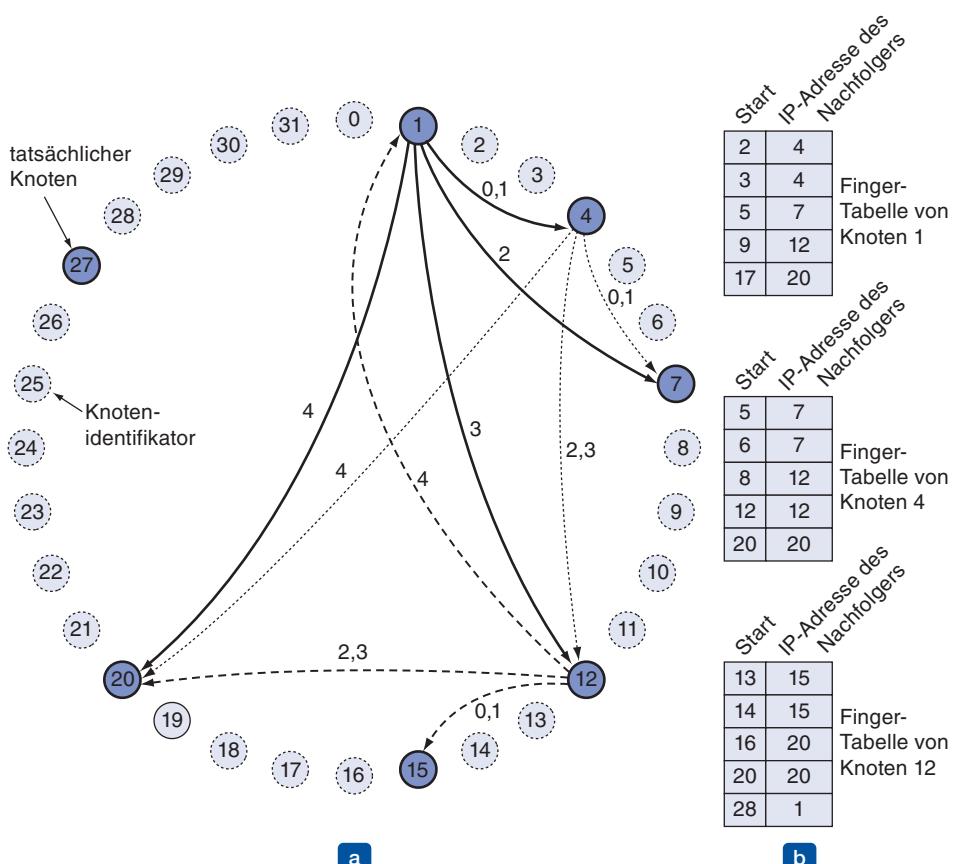


Abbildung 7.71: (a) Eine Gruppe von 32 Knotenidentifikatoren, die in einem Ring angeordnet sind. Die farblich hervorgehobenen Knoten entsprechen tatsächlichen Rechnern. Die Bögen zeigen die Finger von den Knoten 1, 4 und 12. Die Beschriftungen der Bögen sind die Tabellenindizes. (b) Beispiele der Finger-Tabellen.

Nachdem die DHT aufgebaut ist, möchte ein anderer Knoten einen Torrent finden, sodass er dem Schwarm beitreten und Inhalt herunterladen kann. Ein Knoten sucht nach *torrent*, indem er zuerst eine Hash-Berechnung darauf ausführt, um *key* zu erhalten, und dann mit *successor(key)* die IP-Adresse des Knotens ermittelt, der den entsprechenden Wert speichert. Der Wert ist die Liste der Peers im Schwarm. Der Knoten kann seine IP-Adresse der Liste hinzufügen und die anderen Peers kontaktieren, um mit dem BitTorrent-Protokoll Inhalt herunterzuladen.

Der erste Schritt ist einfach, der zweite nicht. Damit die IP-Adresse eines Knotens, der einem bestimmten Schlüssel entspricht, gefunden werden kann, muss jeder Knoten bestimmte Datenstrukturen verwalten. Eine dieser Strukturen ist die IP-Adresse des nachfolgenden Knotens im Ring der Knotenidentifikatoren. Zum Beispiel ist in ►Abbildung 7.71 der Nachfolger von Knoten 4 der Knoten 7 und der Nachfolger von Knoten 7 ist Knoten 12.

Die Suche kann jetzt wie folgt ablaufen: Der anfragende Knoten sendet ein Paket an seinen Nachfolger, zusammen mit seiner IP-Adresse und dem Schlüssel, nach dem er sucht. Das Paket wird im Ring weitergeleitet, bis es den Nachfolger des gesuchten Knotenidentifikators ermittelt hat. Dieser Knoten prüft, ob er irgendwelche Daten gespeichert hat, die dem Schlüssel entsprechen und liefert diese im Erfolgsfall an den anfragenden Knoten zurück, dessen IP-Adresse er kennt.

Das lineare Durchsuchen aller Knoten in einem großen P2P-System ist jedoch sehr ineffizient, da die durchschnittliche Zahl der erforderlichen Knoten pro Suche $n/2$ ist. Um die Suche stark zu beschleunigen, verwaltet jeder Knoten eine sogenannte **Finger-Tabelle**. Die Finger-Tabelle hat m Einträge mit einem Index von 0 bis $m-1$, die jeweils auf einen anderen tatsächlichen Knoten zeigen. Jeder Eintrag hat zwei Felder: *start* und die IP-Adresse von *successor(start)*, wie die drei Beispielknoten in ▶ Abbildung 7.71b zeigen.

Die Werte der Felder für Eintrag i bei einem Knoten mit dem Identifikator k berechnen sich wie folgt:

$$\begin{aligned} \textit{start} &= k + 2^i (\text{modulo } 2^m) \\ \text{IP-Adresse von } \textit{successor}[\textit{start}[i]] \end{aligned}$$

Beachten Sie, dass jeder Knoten die IP-Adressen von nur wenigen Knoten speichert und dass die meisten von ihnen in Bezug auf den Knotenidentifikator ziemlich in der Nähe liegen.

Mit der Finger-Tabelle läuft die Suche nach *key* bei Knoten k wie folgt ab. Wenn *key* zwischen k und *successor(k)* liegt, ist der Knoten, der Informationen über *key* enthält, *successor(k)* und die Suche endet. Andernfalls wird die Finger-Tabelle nach dem Eintrag durchsucht, dessen *start*-Feld der nächstgelegene Vorgänger von *key* ist. Dann wird eine Anfrage direkt an die IP-Adresse in diesem Finger-Tabelleneintrag gesendet, um sie aufzufordern, mit der Suche fortzufahren. Da sie näher an *key*, aber dennoch darunter liegt, stehen die Chancen gut, dass sie die Antwort mit nur wenigen zusätzlichen Abfragen zurückliefern kann. Da jede Suche die verbleibende Entfernung zum Ziel halbiert, kann gezeigt werden, dass die durchschnittliche Zahl der Suchläufe $\log_2 n$ ist.

Als erstes Beispiel betrachten wir die Suche nach *key*=3 bei Knoten 1. Da Knoten 1 weiß, dass 3 zwischen ihm und seinem Nachfolgeknoten 4 liegt, ist 4 der gewünschte Knoten und die Suche endet durch Zurückliefern der IP-Adresse von Knoten 4.

Als zweites Beispiel betrachten wir die Suche nach *key*=16 bei Knoten 1. Da 16 nicht zwischen 1 und 4 liegt, wird die Finger-Tabelle herangezogen. Der nächstgelegene Vorgänger zu 16 ist 9, sodass die Anfrage an die IP-Adresse des Eintrags zu 9, das heißt an Knoten 12, geleitet wird. Knoten 12 weiß die Antwort ebenfalls nicht selbst, sodass er nach dem nächstgelegenen Vorgängerknoten von 16 sucht und 14 findet, was die IP-Adresse von Knoten 15 zurückliefert. Dann wird dorthin eine Anfrage geschickt. Knoten 15 stellt fest, dass 16 zwischen ihm und seinem Nachfolgeknoten (20) liegt, sodass er dem Aufrufer die IP-Adresse von 20 zurückliefert, die ihren Weg zurück zu Knoten 1 findet.

Da die ganze Zeit Knoten hinzukommen und abwandern, benötigt Chord eine Methode, damit umzugehen. Wir nehmen an, dass das System, als es seinen Betrieb aufnahm, klein genug war, sodass die Knoten die Daten einfach direkt austauschen konnten, um den ersten Ring und die Finger-Tabellen zu erstellen. Danach wird ein automatisiertes Verfahren benötigt. Wenn ein neuer Knoten r dem Ring beitreten möchte, muss er einen der bestehenden Knoten kontaktieren und ihn auffordern, nach der IP-Adresse von $\text{successor}(r)$ zu suchen. Als Nächstes fragt der neue Knoten $\text{successor}(r)$ nach seinem Vorgängerknoten. Anschließend fordert der neue Knoten beide Knoten auf, r zwischen sich im Ring einzufügen. Wenn zum Beispiel in Abbildung 7.71 Knoten 24 neu zum Ring hinzustößen möchte, bittet es einen beliebigen Knoten, nach $\text{successor}(24)$ zu suchen, was 27 ergibt. Anschließend fragt er 27 nach seinem Vorgängerknoten (20). Nachdem er beiden über seine Existenz in Kenntnis gesetzt hat, verwendet 20 den Knoten 24 als Nachfolgeknoten und 27 verwendet ihn als Vorgängerknoten. Damit ist der Knoten 24 voll eingefügt.

Das hat jedoch zur Folge, dass viele Finger-Tabellen jetzt nicht mehr stimmen. Um sie zu korrigieren, führt jeder Knoten einen Hintergrundprozess aus, der regelmäßig jeden Finger neu berechnet, indem er successor aufruft. Wenn eine dieser Abfragen auf einen neuen Knoten stößt, wird der entsprechende Finger-Eintrag aktualisiert.

Wenn ein Knoten den Ring ordentlich verlässt, übergibt er seine Schlüssel dem Nachfolgeknoten und informiert seinen Vorgängerknoten über seinen Abgang, sodass dieser sich mit dem Nachfolgeknoten des scheidenden Knotens verbinden kann. Bei einem Knotenabsturz hingegen stellt sich das Problem, dass der Vorgängerknoten keinen gültigen Nachfolgeknoten mehr hat. Um diesem Problem entgegenzuwirken, merkt sich jeder Knoten nicht nur seinen direkten Nachfolgeknoten, sondern seine s direkten Nachfolgeknoten, damit er im Notfall $s-1$ aufeinanderfolgende ausgefallene Knoten überspringen und den Ring wieder verbinden kann.

Seit der Entwicklung von DHTs wurde in diesem Bereich enorm viel geforscht. Um Ihnen eine Vorstellung davon zu geben, wollen wir Ihnen eine Frage stellen: Was ist der am häufigsten zitierte Aufsatz zu Networking? Es wird Ihnen schwerfallen, einen Aufsatz zu finden, der häufiger zitiert wurde als die bahnbrechende Veröffentlichung zu Chord (Stoica et al., 2001). Trotz dieser großen Forschungsanstrengungen kommen erst jetzt allmählich Anwendungen mit DHTs auf den Markt. Einige BitTorrent-Clients verwendet DHTs, um einen vollständig dezentralisierten Tracker anzubieten, wie wir ihn bereits beschrieben haben. Große kommerzielle Cloud-Dienste wie Dynamo von Amazon setzen ebenfalls DHT-Techniken ein (DeCandia et al., 2007).

Zusammenfassung

Die **Namensgebung** im ARPANET war am Anfang sehr einfach: Es gab eine ASCII-Datei, die alle Namen der Hosts und die dazugehörigen IP-Adressen auflistete. Jede Nacht luden sich alle Rechner diese Datei herunter. Als sich jedoch aus dem ARPANET das Internet entwickelte und explosionsartig anwuchs, wurde ein wesentlich anspruchsvoller und dynamisches Namensgebungssystem benötigt. Derzeit wird im Internet ein hierarchisches System namens DNS (Domain Name System) verwendet. Es organisiert alle Rechner im Internet in sogenannten Baumstrukturen. Auf der obersten Ebene stehen die bekannten allgemeinen Domänen, wie *com* oder *edu*, sowie über 200 Domänen für einzelne Länder. DNS wird als verteiltes Datenbanksystem mit weltweit vorhandenen Servern implementiert. Durch Abfrage eines DNS-Servers kann ein Prozess zu einem Domänennamen im Internet die IP-Adresse erhalten, die er benötigt, um mit einem Rechner dieser Domäne zu kommunizieren.

E-Mail ist die erste Killeranwendung des Internets. Sie wird immer noch stark genutzt, denn jeder sendet E-Mails, vom kleinen Kind bis hin zu den Großeltern. Die meisten E-Mail-Systeme der Welt verwenden das in den RFCs 5321 und 5322 definierte Mailsystem. Die Nachrichten haben einfache ASCII-Header und mithilfe von MIME können viele Arten von Inhalten versendet werden. Die Mail wird zum Zustellen Nachrichtenübertragungsagenten übergeben und zum Anzeigen von einer Vielzahl von Benutzeragenten, einschließlich Webanwendungen, entgegengenommen. Die abgeschickte Mail wird mit SMTP versendet. Hierbei wird eine TCP-Verbindung vom sendenden zum empfangenden Nachrichtenübertragungsagenten errichtet.

Das **Web** ist die Anwendung, die von den meisten Menschen für das Internet gehalten wird. Ursprünglich war es ein System, um nahtlos Hypertextseiten (in HTML) über Rechner hinweg miteinander zu verknüpfen. Die Seiten werden mit HTTP heruntergeladen, indem eine TCP-Verbindung vom Browser zu einem Server hergestellt wird. Heutzutage wird viel Inhalt im Web dynamisch erzeugt, sei es beim Server (z.B. mit PHP) oder beim Browser (z.B. mit JavaScript). In Kombination mit Back-End-Datenbanken können mit dynamischen Serverseiten Webanwendungen wie E-Commerce und Suchmaschinen realisiert werden. Dynamische Browserseiten entwickeln sich zu umfangreichen Anwendungen mit allen Funktionen, wie E-Mail-Programme, die im Browser ausgeführt werden und die Webprotokolle für die Kommunikation mit den entfernten Servern verwenden.

Durch den verstärkten Einsatz von Zwischenspeicherung und dauerhaften Verbindungen wird die **Webleistung** verbessert. Eine große Herausforderung stellt die Verwendung des Webs auf mobilen Geräten dar, obwohl die Bandbreite und die Rechenleistung der mobilen Geräte zugenommen haben. Websites senden an Geräte mit kleinem Display oft speziell darauf zugeschnittene Versionen mit kleineren Bildern und einfacherer Navigation.

Die **Webprotokolle** werden zunehmend für die Kommunikation zwischen den Rechnern eingesetzt. Als Beschreibung des Inhalts, der von den Rechnern leicht zu verarbeiten ist, hat XML inzwischen HTML weitestgehend abgelöst. SOAP ist ein RPC-Mechanismus, der XML-Nachrichten über HTTP sendet.

Digitales Audio und Video sind seit 2000 die treibenden Kräfte bei der Entwicklung des Internets. Der größte Teil des **Internetverkehrs** besteht aus Videos. Viele dieser Videos werden von Web-sites über eine Kombination von Protokollen gestreamt (einschließlich RTP/UDP und RTP/HTTP/ TCP). Mediendaten werden live an viele Konsumenten übertragen. Dazu gehören Internetradio und Fernsehsender, die alle möglichen Ereignisse übertragen. Audio und Video werden außerdem für Echtzeitkonferenzen eingesetzt. Viele Anrufe verwenden Voice-over-IP anstelle des traditionel- len Telefonnetzes und umfassen Videokonferenz.

Es gibt eine kleine Zahl von extrem bekannten Websites sowie eine riesige Anzahl an weniger bekannten. Für den Ansturm auf die beliebten Websites wurden **Content-Verteilernetze** (CDNs) aufgebaut. CDNs leiten mittels DNS die Clients zu einem nahe gelegenen Server. Die Server befinden sich in Datenzentren rund um die Welt. Alternativ gibt es P2P-Netze, die es einer Gruppe von Rechnern erlauben, Inhalte wie Filme untereinander auszutauschen. Die Inhaltsverteilungskapazi-tät dieser Netze wächst mit der Anzahl der Rechner im Netz und kann durchaus mit den größten Sites mithalten.



Lösungs-
hinweise

Übungsaufgaben

- 1** Viele Firmenrechner haben drei verschiedene, weltweit eindeutige Identifikatoren. Wie lau-ten diese?
- 2** Warum steht in Abbildung 7.4 hinter *laserjet* kein Punkt?
- 3** Stellen Sie sich vor, ein Cyberterrorist hätte es geschafft, sämtliche DNS-Server auf der gan-zen Welt gleichzeitig zum Absturz zu bringen. Welche Möglichkeiten gäbe es danach noch, das Internet zu nutzen?
- 4** DNS verwendet UDP anstatt TCP. Geht ein DNS-Paket verloren, wird es nicht automatisch wiederhergestellt. Führt dies zu einem Problem und wenn ja, wie wird es gelöst?
- 5** John möchte einen originellen Domänennamen und verwendet ein randomisiertes Pro-gramm, um für sich einen sekundären Domänennamen zu erzeugen. Er möchte diesen Domänennamen in der allgemeinen Domäne *com* registrieren. Der erzeugte Domänenname ist 253 Zeichen lang. Erlaubt die *com*-Registrierungsstelle die Registrierung dieses Domänen-namens?
- 6** Kann ein Rechner mit einem DNS-Namen mehrere IP-Adressen besitzen? Wie ist das möglich?
- 7** Die Anzahl der Unternehmen, die eine Website haben, ist in den letzten Jahren erheblich gestiegen. Daher haben sich Tausende von Unternehmen in der *com*-Domäne registriert, sodass diese Domäne den Top-Level-Server stark belastet. Schlagen Sie eine Lösung vor, um dieses Problem zu lindern, ohne dass das Benennungsschema verändert wird (also ohne neue Top-Level-Domänennamen einzuführen). Ihre Lösung darf Änderungen am Client-Code vornehmen.
- 8** Einige E-Mail-Systeme unterstützen das Header-Feld *Content Return*: Es bestimmt, ob der Nachrichteninhalt im Falle einer Nichtzustellung zurückgesendet werden muss. Gehört die-ses Feld in den Umschlag oder in den Header?

- 9** E-Mail-Systeme müssen Verzeichnisse haben, damit man die E-Mail-Adresse von Personen nachsehen kann. Um solche Verzeichnisse zusammenzustellen, sollten Namen in Standardkomponenten (z.B. Vor- und Nachname) aufgeteilt werden, um die Suche zu ermöglichen. Diskutieren Sie einige Probleme, die zuerst gelöst werden müssen, damit ein Standard weltweit akzeptiert wird.
- 10** Eine große Rechtsanwaltskanzlei mit vielen Angestellten richtet für jeden Angestellten eine eigene E-Mail-Adresse an, die `<login>@kanzlei.com` lauten sollte. Leider hat die Kanzlei jedoch nicht explizit das Format des Logins definiert. Deshalb benutzen einige Angestellte ihren Vornamen, andere ihren Nachnamen und noch andere ihre Initialen als Login. Jetzt möchte die Kanzlei ein einheitliches Format vorgeben, zum Beispiel:
`vorname.nachname@kanzlei.com`
das für die E-Mail-Adressen aller ihrer Angestellten verwendet werden soll. Wie kann dies umgesetzt werden, ohne allzu viele Schwierigkeiten zu bereiten?
- 11** Eine Binärdatei ist 4 560 Byte lang. Wie lang wird sie, wenn sie mit der base64-Codierung verschlüsselt wird, wobei ein CR+LF-Paar (Zeilenschaltung und Zeilenvorschub) nach jeweils 110 gesendeten Byte und am Ende eingefügt wird?
- 12** Geben Sie fünf MIME-Typen an, die im Buch nicht erwähnt werden. Sie können diese Informationen in Ihrem Browser oder im Internet finden.
- 13** Angenommen, Sie möchten eine MP3-Datei an einen Freund senden, aber der ISP Ihres Freundes begrenzt das Volumen der eingehenden Mail auf 1 MB, die MP3-Datei ist aber 4 MB groß. Gibt es eine Möglichkeit, mit RFC 5322 und MIME dieser Situation Herr zu werden?
- 14** Nehmen Sie an, dass John gerade einen automatischen Weiterleitmechanismus für seine E-Mail-Adresse auf der Arbeit eingerichtet hat. Dieser Mechanismus empfängt alle seine geschäftlichen E-Mails und leitet sie an seine private E-Mail-Adresse weiter, die John mit seiner Frau teilt. Johns Ehefrau wusste davon nichts und hat einen Vacation Agent für ihr persönliches Konto aktiviert. Da John seine E-Mail weitergeleitet hat, hat er keinen Vacation-Dämon auf seinem Arbeitsrechner eingerichtet. Was passiert, wenn eine E-Mail an Johns E-Mail-Adresse auf der Arbeit eingeht?
- 15** In jedem Standard, wie RFC 5322, ist eine exakte Grammatik der zulässigen Aktionen erforderlich, damit verschiedene Implementierungen zusammenarbeiten können. Selbst einfache Elemente müssen sorgfältig definiert werden. SMTP-Header erlauben Leerraum zwischen Token. Geben Sie *zwei*/plausible alternative Definitionen für den Leerraum zwischen Token an.
- 16** Gehört der Vacation Agent zum Benutzeragenten oder zum Nachrichtenübertragungsagenten? Er wird natürlich mit dem Benutzeragenten eingerichtet, aber sendet der Benutzeragent auch die Antworten? Erklären Sie Ihre Antwort.
- 17** In einer einfachen Version des Chord-Algorithmus für die P2P-Suche verwenden die Suchläufe nicht die Finger-Tabellen. Stattdessen sind sie zu beiden Richtungen im Ring linear. Kann ein Knoten genau vorhersagen, in welche Richtung er suchen muss? Diskutieren Sie Ihre Antwort.

- 18** Benutzer können über IMAP E-Mail von einer entfernten Mailbox abrufen. Bedeutet dies, dass das interne Format der Mailboxen standardisiert sein muss, damit jedes IMAP-Programm auf der Clientseite die Mailbox auf jedem Server lesen kann? Erläutern Sie Ihre Antwort.
- 19** Betrachten Sie den Chord-Ring aus Abbildung 7.71. Angenommen, der Knoten 18 stellt plötzlich eine Onlineverbindung her. Welche der Finger-Tabellen aus der Abbildung sind betroffen und wie?
- 20** Verwendet Webmail POP3, IMAP oder keines von beiden? Wenn eines von beiden, warum wurde dies gewählt? Wenn keines von beidem, welchem steht es von der Konzeption her näher?
- 21** Wenn Webseiten versendet werden, werden MIME-Header vorangestellt. Warum?
- 22** Kann folgender Fall eintreten: Wenn ein Benutzer auf einen Link in Firefox klickt, wird ein bestimmtes Hilfsprogramm gestartet, wenn er aber auf den gleichen Link im Internet Explorer klickt, wird ein ganz anderes Hilfsprogramm gestartet, selbst wenn der MIME-Typ in beiden Fällen identisch ist. Erklären Sie Ihre Antwort.
- 23** Obwohl es nicht im Text erwähnt wurde, ist eine alternative Form einer URL die Verwendung der IP-Adresse anstatt des DNS-Namens. Verwenden Sie diese Information, um zu erklären, warum ein DNS-Name nicht mit einer Ziffer enden kann.
- 24** Stellen Sie sich vor, dass jemand in der Mathematikfakultät an der Universität Stanford gerade ein Dokument mit einem neuen Beweis geschrieben hat, das er über FTP zur Überprüfung an seine Kollegen verteilen will. Er stellt es in das FTP-Verzeichnis *ftp/pub/forReview/newProof.pdf*. Wie könnte die URL dafür lauten?
- 25** In Abbildung 7.22 verfolgt *www.aportal.com* die Benutzerpräferenzen in einem Cookie. Ein Nachteil dieser Methode ist, dass Cookies auf 4 KB beschränkt sind, sodass bei umfangreichen Informationen über Vorlieben, wie z. B. viele unterschiedliche Aktien, Sportteams und Arten von Nachrichten, Wetter für mehrere Städte, spezielle Informationen in verschiedenen Produktkategorien und mehr, das 4-KB-Limit erreicht werden kann. Entwerfen Sie eine alternative Möglichkeit, die Vorlieben zu verfolgen, bei der dieses Problem nicht auftritt.
- 26** Sloth Bank möchte Onlinebanking für seine bequemen Kunden vereinfachen. Wenn sich ein Kunde anmeldet und über ein Passwort authentifiziert, gibt die Bank ein Cookie zurück, das eine Kundennummer enthält. Auf diese Weise muss sich der Kunde in Zukunft nicht mehr selbst über das Passwort legitimieren. Was halten Sie von dieser Idee? Funktioniert das? Ist es eine gute Idee?
- 27** Betrachten Sie das folgende HTML-Tag:
- ```
<h1 title="Dies ist der Header"> HEADER 1 </h1>
```
- Unter welchen Bedingungen verwendet der Browser das Attribut *TITLE* und wie?
  - Inwiefern unterscheidet sich das *TITLE*-Attribut vom *A/F*-Attribut?
- 28** Wie machen Sie ein Bild in HTML anklickbar? Führen Sie ein Beispiel an.
- 29** Schreiben Sie eine HTML-Seite, die einen Link zu der E-Mail-Adresse *username@Domain-Name.com*. Was passiert, wenn ein Benutzer diesen Link anklickt?

- 30** Schreiben Sie eine XML-Seite für eine Universitäts-Registrierungsstelle, die mehrere Studenten aufführt, jeweils mit Name, Adresse und Durchschnittsnote.
- 31** Geben Sie für jede der folgenden Anwendungen mit Nennung der Gründe an, ob es (1) möglich und (2) besser ist, ein PHP-Skript oder JavaScript zu verwenden:
- a. Anzeige eines Kalenders für einen beliebigen angeforderten Monat ab September 1752
  - b. Anzeige des Flugplans von Amsterdam nach New York
  - c. Darstellung eines Polynoms mit vom Benutzer angegebenen Koeffizienten
- 32** Erstellen Sie ein Programm in JavaScript, das eine ganze Zahl größer 2 akzeptiert und angibt, ob es sich um eine Primzahl handelt. Beachten Sie, dass JavaScript *if* und *while*-Anweisungen kennt, die die gleiche Syntax wie in C und Java haben. Der Modulo-Operator ist %. Wenn Sie die Quadratwurzel von  $x$  benötigen, verwenden Sie *Math.sqrt(x)*.
- 33** Eine HTML-Seite sieht wie folgt aus:
- ```
<html> <body>
<a href="www.info-source.com/welcome.html"> Hier klicken für Info
</a>
</body> </html>
```
- Wenn der Benutzer auf den Hyperlink klickt, wird eine TCP-Verbindung eröffnet und es werden mehrere Zeilen an den Server gesendet. Listen Sie alle gesendeten Zeilen auf.
- 34** Mit dem Header *If-Modified-Since* kann geprüft werden, ob eine zwischengespeicherte Seite immer noch gültig ist. Es können Seiten mit Bildern, Klang und Videodateien wie auch HTML angefordert werden. Glauben Sie, dass die Effektivität dieser Technik für JPEG-Bilder im Vergleich zu HTML besser oder schlechter ist? Denken Sie sorgfältig darüber nach, was „Effektivität“ bedeutet, und erläutern Sie Ihre Antwort.
- 35** Am Tag einer großen Sportveranstaltung wie der Austragung der Meisterschaft in einer beliebten Sportart besuchen viele Personen die offizielle Website. Ist dies im gleichen Sinn eine Flash Crowd wie bei der Wahl in Florida im Jahre 2000? Warum oder warum nicht?
- 36** Ergibt es für einen ISP Sinn, als CDN zu fungieren? Wenn ja, wie funktioniert dies? Wenn nicht, was ist falsch an dieser Idee?
- 37** Angenommen, für Audio-CDs wird keine Komprimierung verwendet. Wie viele MB Daten muss die CD enthalten, um zwei Stunden Musik abzuspielen?
- 38** In Abbildung 7.42c wird durch die Verwendung von 4-Bit-Samples zur Darstellung von neun Signalwerten Quantisierungsrauschen verursacht. Das erste Sample bei 0 ist genau, die nächsten nicht. Wie groß ist der prozentuale Fehler der Samples bei 1/32, 2/32 und 3/32 der Folge?
- 39** Kann die für die Internettelefonie erforderliche Bandbreite anhand eines psychoakustischen Modells reduziert werden? Wenn ja, welche Bedingungen müssen hierfür erfüllt sein, damit es funktioniert? Wenn nicht, warum nicht?

- 40** Ein Audio-Streaming-Server hat eine unidirektionale „Entfernung“ zu einem Medioplayer von 100 ms. Er erzeugt mit 1 Mbit/s Ausgaben. Wenn der Medioplayer einen Puffer von 2 MB besitzt, welche Aussage kann man dann über die Untergrenze und Obergrenze des Puffers treffen?
- 41** Hat Voice-over-IP das gleiche Problem mit Firewalls wie Streaming Audio? Erläutern Sie Ihre Antwort.
- 42** Wie groß ist die Bitübertragungsrate für die Übertragung nicht komprimierter 1 200×800-Farbrahmens mit 16 Bit/Pixel bei 50 Rahmen/s?
- 43** Kann sich ein 1-Bit-Fehler in einem MPEG-Rahmen auf mehr als den Rahmen auswirken, in dem der Fehler vorhanden ist? Erklären Sie Ihre Antwort.
- 44** Betrachten Sie einen Videoserver für 50 000 Kunden, wobei jeder Kunde zwei Filme pro Monat sieht. Nehmen Sie an, dass zwei Drittel aller Filme um 20 Uhr gesendet werden. Wie viele Filme muss der Server während dieser Zeitspanne auf einmal übertragen? Wie viele OC-12-Verbindungen braucht der Server zum Netz?
- 45** Nehmen Sie an, dass das Zipf'sche Gesetz auf die Zugriffe auf einen Videoserver mit 10 000 Filmen anwendbar ist. Geben Sie einen Ausdruck für den Bruchteil aller Zugriffe auf den Speicher, wenn der Server die gängigsten 1 000 Filme im Speicher und die restlichen 9 000 auf Festplatte verwaltet. Schreiben Sie ein kleines Programm, um diesen Ausdruck numerisch auszuwerten.
- 46** Einige Cybersquatters (Webbesetzer) haben registrierte Domänenamen, die eine Falschschreibung einer bekannten Unternehmens-Site darstellen, wie *www.microsfot.com*. Nennen Sie mindestens fünf solche Domänen.
- 47** Viele Leute haben registrierte DNS-Namen, die aus einem *www.word.com* bestehen, wobei *word* ein normales Wort ist. Listen Sie für jede der folgenden Kategorien fünf Websites auf und fassen Sie kurz zusammen, was dahinter steckt (so ist beispielsweise *www.stomach.com* ein Gastroenterologe in Long Island). Die Liste der Kategorien lautet wie folgt: Tiere, Nahrungsmittel, Haushaltsgeräte und Körperteile. Bei der letzten Kategorie nehmen Sie bitte Körperteile oberhalb der Taille.
- 48** Schreiben Sie den Server in Abbildung 6.6 als echten Webserver unter Verwendung des *GET*-Befehls für HTTP 1.1 um. Er sollte auch die *Host*-Nachricht akzeptieren. Der Server sollte einen Cache mit vor kurzem von der Platte eingelesenen Dateien pflegen und Anforderungen, wann immer möglich, aus dem Cache bedienen.

8

ÜBERBLICK

Sicherheit in Netzen

| | |
|---|-----|
| 8.1 Kryptografie | 868 |
| 8.2 Algorithmen für die symmetrische Verschlüsselung | 881 |
| 8.3 Algorithmen für öffentliche Schlüssel | 897 |
| 8.4 Digitale Signaturen | 901 |
| 8.5 Verwaltung öffentlicher Schlüssel | 911 |
| 8.6 Kommunikationssicherheit | 919 |
| 8.7 Authentifizierungsprotokolle | 934 |
| 8.8 E-Mail-Sicherheit | 949 |
| 8.9 Sicherheit im Web | 954 |
| 8.10 Soziale Themen | 969 |

» In den ersten zwei Jahrzehnten ihrer Existenz wurden Rechnernetze vorwiegend von Universitäten für E-Mail und von Großunternehmen für die gemeinsame Nutzung von Ressourcen wie Drucken benutzt. Unter diesen Bedingungen wurde dem Faktor Sicherheit wenig Aufmerksamkeit gewidmet. Aber da inzwischen Millionen von „Normalbürgern“ Zugang zu Netzen haben, z.B. für E-Shopping und Onlinebanking, das Einreichen von Steuererklärungen usw., und sich eine Schwachstelle nach der anderen offenbart, ist Netzwerksicherheit zu einem sehr wichtigen Thema geworden. In diesem Kapitel werden wir die Sicherheitsaspekte in Netzen aus verschiedenen Blickwinkeln betrachten, uns mit etlichen Fallstricken beschäftigen, sowie viele Algorithmen und Protokolle vorstellen, mit denen die Sicherheit von Rechnernetzen verbessert werden kann.

Sicherheit ist ein breites Thema und beschäftigt sich mit einer Vielzahl von Sünden. In der einfachsten Form soll sichergestellt werden, dass neugierige Menschen nicht die Nachrichten anderer lesen oder, schlimmer noch, ändern. Außerdem hat Sicherheit die Aufgabe, unberechtigte Personen davon abzuhalten, auf entfernte Dienste zuzugreifen. Ebenso werden Möglichkeiten angegeben, um festzustellen, ob die Nachricht „Zahle bis Freitag oder ...“ wirklich vom Finanzamt und nicht von der Mafia ist. Sicherheit hat auch damit zu tun, dass seriöse Nachrichten erfasst und wiedergegeben werden, sowie mit Personen, die später abstreiten, bestimmte Nachrichten gesendet zu haben.

Die meisten Sicherheitsprobleme werden absichtlich verursacht, von Personen, die sich davon einen Vorteil versprechen, Aufmerksamkeit suchen oder jemandem schaden wollen. Eine Auswahl solcher Übeltäter ist in ► Abbildung 8.1 aufgeführt. Aus dieser Liste geht hervor, dass Netzsicherheit viel mehr verlangt, als das Netz von Programmierfehlern frei zu halten. Es bedeutet, dass man cleverer sein muss als die Eindringlinge, die meist intelligent, engagiert und gut ausgestattet sind. Es sollte auch klar sein, dass Maßnahmen, die gelegentliche Eindringlinge aufhalten, wenig Wirkung auf die ernsten Übeltäter haben. Den Polizeiakten kann man entnehmen, dass die meisten Angriffe nicht von Außenstehenden kommen, die eine Telefonleitung anzapfen, sondern von Insidern, die einen Groll gegen den Attackierten hegen. Dies sollte bei der Konzeption von Sicherheitssystemen berücksichtigt werden.

Probleme in Bezug auf die Netzsicherheit können grob in vier miteinander verflochene Bereiche unterteilt werden: Geheimhaltung, Authentifizierung, Nichtabstreitbarkeit und Integritätskontrolle. Geheimhaltung hat damit zu tun, dass Informationen vor dem Zugriff durch unberechtigte Benutzer geschützt werden. Dies ist der erste Gedanke beim Thema Sicherheit in Netzen. Authentifizierung hat die Aufgabe festzustellen, mit wem man es zu tun hat, bevor man vertrauliche Informationen preisgibt oder eine Geschäftsbeziehung eingeht. Nichtabstreitbarkeit dreht sich um Signaturen: Wie können Sie beweisen, dass der Kunde wirklich eine elektronische Bestellung für zehn Millionen Schraubenzieher für Linkshänder zu 89 Cent das Stück aufgegeben hat, wenn er später behauptet, es sei ein Stückpreis von 69 Cent vereinbart worden? Vielleicht behauptet er sogar, die Bestellung nie aufgegeben zu haben. Und schließlich

können Sie mithilfe von Integritätskontrolle sicherstellen, dass eine empfangene Nachricht von dem erwarteten Sender stammt und nicht von einer böswilligen Person im Laufe der Übertragung verändert oder aufgesetzt wurde.

| Übeltäter | Ziel |
|------------------------|---|
| Student | Hat Spaß daran, in der E-Mail anderer Leute zu schnüffeln |
| Hacker | Will das Sicherheitssystem auf die Probe stellen und Daten stehlen |
| Handelsvertreter | Erhebt den Anspruch, dass er ganz Europa vertritt, nicht nur Andorra |
| Geschäftsmann | Hat Interesse am Marketingplan der Konkurrenz |
| Ehemaliger Mitarbeiter | Tötet seine Rachegelüste aus, weil er entlassen wurde |
| Buchhalter | Will Geldbeträge von der Firma betrügerisch abzweigen |
| Börsenmakler | Streitet ab, dass er dem Kunden über E-Mail ein Versprechen gegeben hat |
| Identitätsdieb | Klaut Kreditkartennummern zum Verkaufen |
| Regierungsstelle | Hat Interesse an den militärischen Geheimnissen des Feindes |
| Terrorist | Hat Interesse an geheimen biologischen Waffen |

Abbildung 8.1: Auswahl böswilliger Eindringlinge und deren Gründe.

Alle vier Bereiche spielen auch in traditionellen Systemen eine Rolle, aber mit beträchtlichen Unterschieden. Integrität und Geheimhaltung werden erreicht, indem man Einschreiben verwendet und Dokumente wegsperrt. Postzüge zu überfallen ist heute schwieriger als zu Zeiten von Jesse James.

Normalerweise kann man auch den Unterschied zwischen einem Original und einer Fotokopie feststellen. Fertigen Sie als Test eine Fotokopie von einem gültigen Scheck an. Reichen Sie den Originalscheck am Montag bei Ihrer Bank ein. Nun versuchen Sie, die Fotokopie des Schecks am Dienstag einzureichen. Beobachten Sie den Unterschied im Verhalten des Bankangestellten. Bei elektronischen Schecks sind Original und Kopie nicht auseinanderzuhalten. Es kann eine Weile dauern, bis Banken gelernt haben, damit umzugehen.

Leute können andere am Gesicht, an der Stimme und an der Handschrift erkennen. Gültige Signaturen werden durch Unterschriften auf Papier mit Briefkopf, Siegeln usw. angegeben. Eine Fälschung kann in der Regel durch die Handschrift, die Tinte und durch Papierexperten nachgewiesen werden. Elektronisch ist das nicht möglich. Auch hier sind gute Lösungen dringend erforderlich.

Bevor wir uns mit den Lösungen befassen, betrachten wir, wo im Protokollstapel man Sicherheitsfragen im Netz einordnen kann. Wahrscheinlich gibt es nicht nur einen Ort. Jede Schicht muss etwas dazu beitragen. Auf der Bitübertragungsschicht kann das Anzapfen von Leitungen dadurch verhindert werden, dass Übertragungsleitungen

(oder besser Glasfaserkabel) in abgedichteten Röhren verlegt werden, die mit Schutzgas gefüllt sind. Ein Versuch, den Kabelkanal anzubohren, führt zum Entweichen von Gas und folglich zu einem Druckabfall, was wiederum einen Alarm auslöst. Diese Technik wird in einigen Militärsystemen verwendet.

Auf der Sicherungsschicht können Pakete einer Punkt-zu-Punkt-Verbindung verschlüsselt werden, wenn sie einen Rechner verlassen, und wieder entschlüsselt werden, wenn sie bei einem anderen Rechner ankommen. Alle Einzelheiten lassen sich auf der Sicherungsschicht erledigen, wobei die höheren Schichten davon nichts mitbekommen. Diese Lösung fällt weg, wenn die Pakete mehrere Router durchlaufen, weil in jedem Router eine Entschlüsselung anfällt, wodurch die Pakete Angriffen im Router ausgesetzt sind. Außerdem ist es nicht möglich, bestimmte Sitzungen zu schützen (z.B. für Online-Einkäufe mit Kreditkarte) und andere wiederum nicht. Trotzdem ist diese **Onlineverschlüsselung** (*link encryption*) meist nützlich und kann in jedem Netz leicht realisiert werden.

Auf der Vermittlungsschicht können Firewalls installiert werden, um Pakete zu schützen und Eindringlinge auszusperren. Auch die IP-Sicherheit wird auf dieser Schicht ausgeführt.

Auf der Transportschicht können komplette Verbindungen von Ende zu Ende bzw. von Prozess zu Prozess verschlüsselt werden. Zur Erzielung der maximalen Sicherheit ist die Sicherheit von Endpunkt zu Endpunkt erforderlich.

Themen wie Benutzeroauthentifizierung und die Nichtabstreitbarkeit können nur in der Anwendungsschicht behandelt werden.

Da Sicherheit nicht ausschließlich eine Schicht betrifft und wir das Thema nicht in jedem Kapitel dieses Buches ansprechen wollten, haben wir uns entschlossen, dem Thema ein eigenes Kapitel zu widmen.

Auch wenn dieses Kapitel lang, technisch und wichtig ist, ist es teilweise irrelevant. Es gibt ausreichend Belege, dass die meisten Sicherheitsfehler bei Banken auf inkompetente Mitarbeiter und lasche Sicherheitsprozeduren zurückzuführen sind, auf zahlreiche Implementierungsfehler, die unberechtigten Nutzern ein Eindringen in das Banksystem ermöglichen, sowie auf Social-Engineering-Angriffe, bei denen Kunden mit unlauteren Mitteln aufgefordert werden, ihre Kontodaten offenzulegen. All diese Sicherheitsprobleme sind stärker verbreitet als clevere Kriminelle, die Telefonleitungen anzapfen und dann chiffrierte Nachrichten entschlüsseln. Wenn eine Person irgendeine Bankfiliale mit einem auf der Straße gefundenen Geldausgabebeleg aufsuchen und behaupten kann, dass er seine PIN vergessen habe, und (im Namen guter Kundenbeziehungen) sofort eine neue erhält, so kann keine Verschlüsselung der Welt einen Missbrauch verhindern. In dieser Hinsicht öffnet einem das Buch von Ross Anderson (2008a) wirklich die Augen, da es Hunderte von Beispielen von Sicherheitsfehlern in verschiedensten Branchen dokumentiert, die fast alle auf – freundlich formuliert – nachlässigen Geschäftspraktiken oder Nichtbeachtung von Sicherheitsaspekten zurückzuführen sind. Dennoch ist die Kryptografie die technische Basis des E-Commerce, wenn alle diese anderen Faktoren beachtet wurden.

Mit Ausnahme der Sicherheit auf der Bitübertragungsschicht basiert fast die gesamte Sicherheit auf kryptografischen Prinzipien. Aus diesem Grund beginnen wir unsere Untersuchung der Sicherheit mit der ausführlichen Behandlung der Kryptografie. In Abschnitt 8.1 behandeln wir einige der grundlegenden Prinzipien. In Abschnitt 8.2 bis Abschnitt 8.5 stellen wir einige der wichtigsten Algorithmen und Datenstrukturen vor, die in der Kryptografie verwendet werden. Anschließend untersuchen wir im Detail, wie mit diesen Konzepten die Sicherheit in Netzen verwirklicht werden kann. Zum Abschluss gehen wir noch kurz auf technologische und gesellschaftliche Aspekte ein.

Bevor wir einsteigen, möchten wir noch kurz darauf eingehen, was nicht behandelt wird. Wir haben versucht, uns von den Themen her auf Networking und nicht auf Betriebssysteme und Anwendungen zu konzentrieren, obwohl es oftmals schwer ist, hier eine Linie zu ziehen. So gehen wir hier beispielsweise nicht näher auf Benutzerauthentifizierung mit biometrischen Verfahren, Sicherheit von Passwörtern, Angriffe durch Pufferüberlauf, trojanische Pferde, Fälschung der Anmeldung, Code-Injektion wie beispielsweise Cross-Site-Scripting, Viren, Würmer und Ähnliches ein. All diese Themen werden ausführlich in *Kapitel 9 in Modern Operating Systems* (Tanenbaum, 2007)¹ behandelt. Der interessierte Leser findet in diesem Buch die Systemaspekte der Sicherheit. Beginnen wir nun mit unserer Erkundungstour.



¹ Deutsche Ausgabe: „Moderne Betriebssysteme“, Pearson Studium, 2009.

8.1 Kryptografie

Kryptografie kommt aus dem Griechischen und bedeutet „Lehre vom geheimen Schreiben“. Sie hat eine lange und bewegte Geschichte, die Tausende von Jahren zurückreicht. Im folgenden Abschnitt stellen wir einige der Glanzlichter dieser Entwicklung vor als Hintergrundinformationen zu den nachfolgenden Themen. Wenn Sie die ganze Geschichte der Kryptografie kennenlernen möchten, kann das Werk von Kahn (1995) empfohlen werden. Eine umfassende Beschreibung aktueller Sicherheits- und Kryptografiealgorithmen, -protokolle und -anwendungen sowie verwandtes Material ist in Kaufman et al. (2002) enthalten. Weitere Informationen über den mathematischen Ansatz finden Sie in Stinson (2002). Einen weniger mathematischen Ansatz finden Sie in Burnet und Paine (2001).

In professionellen Kreisen wird zwischen Chiffren (*cipher*) und Codes (*code*) unterschieden. Eine **Chiffre** ist eine zeichen- oder bitweise Umwandlung, unabhängig von der linguistischen Struktur der Nachricht. Demgegenüber ersetzt ein **Code** ein Wort durch ein anderes Wort oder Zeichen. Codes werden nicht mehr eingesetzt, obwohl sie eine ruhmvolle Geschichte haben. Der erfolgreichste Code, der je entwickelt wurde, wurde von der amerikanischen Armee im Zweiten Weltkrieg an der pazifischen Front eingesetzt. Hier ließ man einfach Navajo-Indianer miteinander sprechen und dabei bestimmte Navajo-Begriffe für militärische Ausdrücke verwenden, wie *chay-dagahi-nail-tsaidi* (wörtlich: Schildkrötentöter) für Panzerabwehrwaffen. Die Navajo-Sprache ist sehr tonal, außerordentlich komplex und kennt keine Schrift. Und niemand in Japan kannte diese Sprache.

Im September 1945 beschrieb der *San Diego Union* den Code wie folgt: „Drei Jahre lang haben die Japaner, wo immer die Marines auch landeten, seltsame gurgelnde Laute gehört, zwischen die andere Töne eingestreut waren, die sich wie die Rufe tibetanischer Mönche in Kombination mit dem Ton einer Wärmflasche, die geleert wird, anhörten.“ Die Japaner haben den Code nie entschlüsselt und viele Navajo-Code-Sprecher wurden mit hohen militärischen Ehren für außergewöhnliche Leistung und Tapferkeit ausgezeichnet. Die Tatsache, dass die Amerikaner den japanischen Code entschlüsselten, diese aber nie den Navajo-Code, spielte bei den amerikanischen Siegen im pazifischen Raum einen wichtige Rolle.

8.1.1 Einführung in die Kryptografie

Historisch haben vier Gruppen die Kunst der Kryptografie ausgeübt und weiterentwickelt – das Militär, das Diplomatische Corps, die Tagebuchschreiber und die Verliebten. Das Militär ist die Gruppe, die diese Wissenschaft über die Jahrhunderte am stärksten geprägt hat. In militärischen Organisationen wurden geheime Nachrichten zur Verschlüsselung und Übertragung an schlecht bezahlte, rangniedrige Funker gegeben. Aufgrund der großen Menge an zu übermittelnden Nachrichten konnte diese Arbeit nicht von einer kleinen Elite von Spezialisten durchgeführt werden.

Vor Anbruch des Computerzeitalters war der eigentliche Maßstab für die Kryptografie die Fähigkeit des Funkers, die notwendigen Umwandlungen durchzuführen – und das oft mitten auf dem Schlachtfeld und mit sehr dürftiger Ausrüstung. Ein weiteres Problem war die schnelle Umstellung von einer kryptografischen Methode auf eine andere, weil dann viele Leute umgeschult werden mussten. Da jedoch die Gefahr bestand, dass ein Funker dem Feind in die Hände fiel, musste die kryptografische Methode bei Bedarf sofort umgestellt werden können. Diese widersprüchlichen Anforderungen führten zu dem in ►Abbildung 8.2 dargestellten Modell.

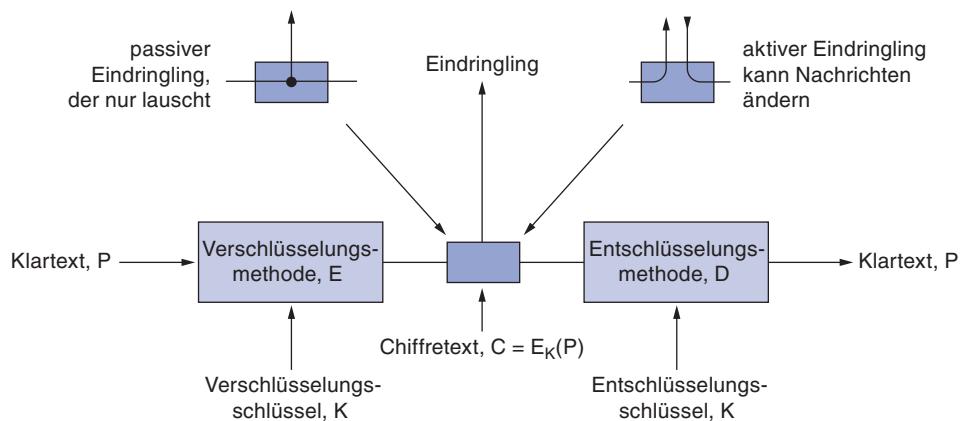


Abbildung 8.2: Das Verschlüsselungsmodell (für symmetrische Chiffren).

Die zu verschlüsselnden Nachrichten, auch **Klartext** (*plaintext*) genannt, werden hier durch eine Funktion umgewandelt, die von einem **Schlüssel** (*key*) parametrisiert wird. Das Ergebnis ist dann der **Chiffretext** (*ciphertext*), der (meist durch einen Boten oder über Funk) übertragen wird. Wir gehen davon aus, dass der Feind bzw. der **Eindringling** den ganzen verschlüsselten Text abhört und genau kopiert. Im Gegensatz zum eigentlichen Empfänger kennt der Feind den Schlüssel nicht und kann daher den Chiffretext nicht ohne Weiteres entschlüsseln. Manchmal kann der Eindringling den Kommunikationskanal nicht nur abhören (passiver Eindringling), sondern Nachrichten auch aufzeichnen und später nochmals abspielen, seine eigenen Nachrichten einspielen oder die Nachricht verändern, bevor sie beim Empfänger ankommt (aktiver Eindringling). Die Kunst, Chiffren zu entziffern, wird **Kryptoanalyse** (*cryptanalysis*) und die Kunst, diese zu entwickeln, **Kryptografie** (*cryptography*) genannt. Beide zusammen werden als **Kryptologie** (*cryptology*) bezeichnet.

Meist ist es nützlich, eine Notation zur Hand zu haben, die unverschlüsselten Text, verschlüsselten Text und Schlüssel verbindet. Wir verwenden $C=E_K(P)$, um zu bezeichnen, dass die Verschlüsselung des Klartextes P mit dem Schlüssel K zum Chiffretext C führt. Ebenso stellt $P=D_K(C)$ das Entschlüsseln von C dar, um wieder den Klartext zu erhalten. Es folgt dann, dass

$$D_K(E_K(P)) = P$$

Bei dieser Notation sind E und D lediglich mathematische Funktionen. Verwickelt ist hier, dass beides Funktionen von zwei Parametern sind und wir einen der Parameter (den Schlüssel) tiefgestellt haben, statt ihn als Argument zu verwenden, um ihn von der Nachricht zu unterscheiden.

Eine der wichtigsten Regeln der Kryptografie lautet, davon auszugehen, dass der Kryptoanalytiker die Methoden für die Verschlüsselung und Entschlüsselung kennt. Mit anderen Worten, er weiß, wie die Verschlüsselungsmethode E und die Entschlüsselung D in Abbildung 8.2 im Detail funktionieren. Der Aufwand für Entwurf, Test und Installation einer neuen Methode (die immer erforderlich ist, wenn die alte Methode gefährdet ist oder zumindest gefährdet scheint) hat die Geheimhaltung des Verschlüsselungsalgorithmus bisher immer bereitete. Einfach anzunehmen, dass geheim bleibt, was geheim ist, ist noch viel gefährlicher.

Hier kommt der Schlüssel ins Spiel. Er besteht aus einer (relativ) kurzen Zeichenkette, die eine von vielen möglichen Verschlüsselungsmethoden auswählt. Im Gegensatz zur grundlegenden Verschlüsselungsmethode, die unter Umständen nur alle paar Jahre geändert wird, kann der Schlüssel bei Bedarf jederzeit variiert werden. Also ist unser Grundmodell eine stabile und allseits bekannte Verschlüsselung, die von einem geheimen und leicht zu verändernden Schlüssel parametrisiert wird. Das Konzept, dass ein Kryptoanalytiker den Algorithmus kennt und das Geheimnis nur in den Schlüsseln liegt, wird als, **Kerckhoffs' Prinzip** bezeichnet, nach dem flämischen Militärkryptografen Auguste Kerckhoffs, der dies als Erster im Jahre 1883 formulierte (Kerckhoffs, 1883).

Kerckhoffs' Prinzip: Alle Algorithmen müssen öffentlich sein, nur die Schlüssel sind geheim.

Die allgemeine Bekanntheit des Algorithmus kann nicht genug betont werden. Der Versuch, den Algorithmus geheim zu halten, wird in der Branche als **Security by Obscurity** (Sicherheit durch Unklarheit) bezeichnet und funktioniert nie. Durch Veröffentlichen des Algorithmus erhält der Kryptograf kostenlose Beratung von vielen akademischen Kryptologen, die daran interessiert sind, das System zu knacken, damit sie Aufsätze veröffentlichen können, um aufzuzeigen, wie schlau sie sind. Ist es vielen Fachleuten in etlichen Jahren nach der Veröffentlichung nicht gelungen, den Algorithmus zu knacken, kann man ihn wohl als stabil bezeichnen.

Das eigentliche Geheimnis liegt im Schlüssel und seiner Länge. Man betrachte ein einfaches Kombinationsschloss. Das allgemeine Prinzip ist dabei, dass man nacheinander Zahlen eingibt. Jeder weiß das, aber der Schlüssel ist geheim. Eine Schlüssellänge von zwei Ziffern bedeutet, dass es 100 mögliche Kombinationen gibt. Eine Schlüssellänge von drei Ziffern bedeutet 1 000 mögliche Kombinationen, und eine von sechs Ziffern bietet eine Million Zahlenkombinationen. Je länger der Schlüssel, um so höher der **Arbeitsaufwand**, dem sich der Kryptoanalytiker gegenübergestellt sieht. Der Arbeitsaufwand zum Knacken des Systems durch vollständiges Durchsuchen des Schlüsselraums steigt exponentiell mit der Schlüssellänge. Die Geheimhaltung ergibt sich durch einen starken (aber öffentlichen) Algorithmus und einen langen Schlüssel. Um Ihren kleinen Bruder daran zu hindern, Ihre E-Mail zu lesen, reicht ein 64-Bit-Schlüs-

sel aus. Für normale kommerzielle Einsatzbereiche sind mindestens 128 Bit notwendig. Und um Regierungen vom Schnüffeln abzuhalten, sollten nicht weniger als 256 Bit, besser aber noch mehr, verwendet werden.

Aus Sicht des Kryptoanalytikers gibt es drei Varianten der Kryptoanalyse. Wenn er einen Chiffretext zur Verfügung hat, aber keinen Klartext dazu, nennt er sein Problem **Nur Chiffretext**. Dazu zählen die Kryptogramme in der Rätselspalte von Zeitungen. Hat der Kryptoanalytiker einen Chiffretext und den dazugehörigen Klartext, ist das der **bekannte Klartext**. Schließlich gibt es noch die Möglichkeit, dass er Teile eines Klartextes eigener Wahl verschlüsseln kann. Das ist der **gewählte Klartext**. Kryptogramme in Zeitungen wären ganz einfach zu lösen, wenn der Kryptoanalytiker fragen könnte: „Wie lautet die Verschlüsselung von ABCDEFGHIJKL?“

Neulinge im Bereich der Kryptografie glauben oft, dass eine Chiffre sicher ist, wenn sie einem Angriff aus der Nur-Chiffretext-Ecke widerstehen kann. Diese Annahme ist sehr naiv. In vielen Fällen kann der Kryptoanalytiker bestimmte Teile des Klartextes einfach erraten. Beispielsweise sagen viele Rechner bei der Anmeldung Folgendes: login. Mit einigen passenden Klartext/Chiffretext-Paaren ausgerüstet, wird der Job des Kryptoanalytikers wesentlich einfacher. Um tatsächliche Sicherheit zu garantieren, sollte der Kryptograf auf Nummer sicher gehen und das System so auslegen, dass es nicht zu knacken ist, auch wenn der Gegner beliebige Mengen von gewähltem Klartext verschlüsseln kann.

Verschlüsselungsmethoden wurden schon immer in zwei Kategorien unterteilt: Substitutionschiffren (*substitution cipher*) und Transpositionschiffren (*transposition cipher*). Im Folgenden werden wir kurz auf beide Kategorien eingehen, da sie uns Hintergrundinformationen zur modernen Kryptografie liefern.

8.1.2 Substitutionschiffren

Bei einer **Substitutionschiffre** wird jeder Buchstabe oder jede Buchstabengruppe durch einen anderen Buchstaben oder eine andere Buchstabengruppe ersetzt. Die älteste bekannte Substitutionschiffre ist die **Cäsar-Chiffre**, deren Erfindung Julius Cäsar zugeschrieben wird. Hier wird *a* zu *D*, *b* zu *E*, *c* zu *F*, ... und *z* zu *C*. Das Wort *attack* wird beispielsweise zu *DWWDFN*. In den folgenden Beispielen ist Klartext immer in Kleinbuchstaben und Chiffretext in Großbuchstaben dargestellt.

Eine einfache Verallgemeinerung der Cäsar-Chiffre ermöglicht die Verschiebung des Chiffrenalphabets um k Buchstaben anstatt um 3. In diesem Fall wird k der Schlüssel für eine allgemeine Methode, Alphabete zirkulär zu verschieben. Diese Chiffre hat vielleicht die Pompejaner zum Narren gehalten, seither ist aber niemand mehr darauf reingefallen.

Die nächste Verbesserung ist, jedes Zeichen des Klartextes (der Einfachheit halber gehen wir von 26 Buchstaben aus) auf einen anderen Buchstaben abzubilden. Zum Beispiel:

| | |
|--------------|---|
| Klartext: | a b c d e f g h i j k l m n o p q r s t u v w x y z |
| Chiffretext: | Q W E R T Y U I O P A S D F G H J K L Z X C V B N M |

Dieses Grundsystem heißt **monoalphabetische Substitution**. Hier ist der Schlüssel eine Buchstabenfolge mit 26 Zeichen, die dem vollständigen Alphabet entspricht. Mit dem obigen Schlüssel wird *attack* dann zu *QZZQEA* verschlüsselt.

Auf den ersten Blick wirkt dieses System sicher. Der Kryptoanalytiker kennt zwar das Grundsystem (Substitution Buchstabe für Buchstabe), er weiß aber nicht, welcher der $26! \approx 4 \times 10^{26}$ möglichen Schlüssel verwendet wurde. Alle durchzutesten, wie dies bei der Cäsar-Chiffre möglich ist, ist nicht sehr vielversprechend. Selbst ein Computer, der pro Lösung nur 1 ns benötigt, bräuchte 10 000 Jahre, um alle Möglichkeiten durchzurechnen.

Trotzdem kann die Chiffre problemlos gebrochen werden, auch wenn der Chiffretext nur kurz ist. Der Lösungsweg berücksichtigt die statistischen Eigenschaften natürlicher Sprachen. In der englischen Sprache ist beispielsweise das *e* der häufigste Buchstabe, dann kommen der Reihe nach *t*, *o*, *a*, *n*, *i* usw. Die häufigsten Kombinationen von zwei Buchstaben (**Digramme**) sind *th*, *in*, *er*, *re* und *an*. Die häufigsten Kombinationen von drei Buchstaben (**Trigramme**) sind *the*, *ing*, *and* und *ion*.

Versucht ein Kryptoanalytiker, eine monoalphabetische Chiffre zu brechen, fängt er damit an, die relative Häufigkeit aller Buchstaben im Chiffretext zu ermitteln. Dann könnte er versuchsweise dem häufigsten Buchstaben das *e* zuweisen und dem zweithäufigsten das *t*. Danach würde er nach einem häufig vorkommenden Trigramm mit der Struktur *tXe* suchen. Nun liegt nahe, dass das *X* ein *h* ist. Wenn auch die Struktur *thYt* oft vorkommt, ist das *Y* wahrscheinlich ein *a*. Nun kann er nach häufig vorkommenden Trigrammen der Struktur *aZW* suchen, die höchstwahrscheinlich *and* bedeutet. Indem er häufig vorkommende Buchstaben, Digramme und Trigramme errät, baut der Kryptoanalytiker Buchstabe für Buchstabe testweise einen Klartext auf.

Bei einer anderen Methode wird ein wahrscheinliches Wort oder eine Wortgruppe erraten. Ein Beispiel: Sehen Sie sich den folgenden Chiffretext einer englischen Buchhaltungsfirma an (der Text ist in Gruppen mit je fünf Buchstaben aufgeteilt):

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| CTBMN | BYCTC | BTJDS | QXBNS | GSTJC | BTSWX | CTQTZ | CQVUJ |
| QJSGS | TJQZZ | MNQJS | VLNSX | VSZJU | JDSTS | JQUUS | JUBXJ |
| DSKSU | JSNTK | BGAQJ | ZBGYQ | TLCTZ | BNYBN | QJSW | |

Ein wahrscheinlich in der Nachricht vorkommendes Wort könnte bei einer Buchhaltungsfirma *financial* sein. Mit dem Wissen, dass in dem Wort *financial* der Buchstabe *i* zweimal auftaucht und dazwischen vier andere Buchstaben stehen, machen wir uns auf die Suche. Wir finden zwölf Treffer an den Positionen 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 und 82. Aber nur bei zwei davon, 31 und 42, wird der nächste Buchstabe (der *n* im Klartext entspricht) an der richtigen Stelle wiederholt. Von diesen beiden steht nur bei 31 das *a* an der korrekten Position, sodass wir wissen, dass *financial* an der Position 30 beginnt. Ab diesem Punkt ist die Ableitung des Schlüssels einfach, wobei die Häufigkeitsverteilung der Buchstaben für englische Texte herangezogen wird und nach fast vollständigen Wörtern gesucht wird.

8.1.3 Transpositionschiffren

Substitutionschiffren behalten die Reihenfolge der Klartextsymbole bei und verschleiern diese nur. **Transpositionschiffren** vertauschen Buchstaben, verwenden jedoch keine Ersatzzeichen. In ▶ Abbildung 8.3 sehen Sie eine typische Transpositionschiffre, die sogenannte *Spaltentransposition*. Die Chiffre wird durch ein Wort (oder einen Ausdruck) verschlüsselt, das keine Buchstabenwiederholungen enthält. Im folgenden Beispiel lautet der Schlüssel MEGABUCK. Zweck dieses Schlüssels ist es, die Spalten zu nummerieren: Spalte 1 steht unter dem Schlüsselbuchstaben, der im Alphabet am weitesten vorne zu finden ist, usw. Der Klartext wird horizontal in Zeilen geschrieben und bei Bedarf zur Füllung der Matrix aufgefüllt. Der Chiffretext beginnt mit der Spalte, über der der (alphanumerisch) niedrigste Buchstabe steht.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|--|--|
| M | E | G | A | B | U | C | K | | |
| 7 | 4 | 5 | 1 | 2 | 8 | 3 | 6 | | |
| p | l | e | a | s | e | t | r | Klartext | |
| a | n | s | f | e | r | o | n | please transfer one million dollars to | |
| e | m | i | l | l | i | o | n | my swiss bank account six two two | |
| d | o | l | l | a | r | s | t | Chiffretext | |
| o | m | y | s | w | i | s | s | AFLLSKSOSELAWAIATOOSCTCLNMOMANT | |
| b | a | n | k | a | c | c | o | ESILYNTWRNNTSOWDPAEDOBUOERIRICXB | |
| u | n | t | s | i | x | t | w | | |
| o | t | w | o | a | b | c | d | | |

Abbildung 8.3: Eine Transpositionschiffre.

Um eine Transpositionschiffre brechen zu können, muss der Kryptoanalytiker zuerst einmal erkennen, dass er eine solche vorliegen hat. Wenn er die Verteilung der häufigen Buchstaben (*E, T, A, O, I, N* usw.) untersucht, kann er leicht erkennen, ob sie dem normalen Klartextmuster entsprechen. Trifft das zu, handelt es sich um eine Transpositionschiffre, in der ja jeder Buchstabe sich selbst darstellt, sodass die Häufigkeitsverteilung beibehalten wird.

Als Nächstes muss der Kryptoanalytiker die Spaltenzahl erraten. Oft kann aus dem Kontext der Nachricht ein Wort erraten werden, das wahrscheinlich in der Nachricht vorkommt. Ein Beispiel: Unser Kryptoanalytiker hat den Verdacht, dass im Klartext der vorliegenden Meldung der Ausdruck *milliondollars* vorkommt. Wie man sieht, treten wegen des Umbruchs die Digramme *MO*, *IL*, *LL*, *LA*, *IR* und *OS* im Chiffretext auf. Der Buchstabe *O* im Chiffretext folgt auf den Buchstaben *M* (d.h., diese Buchstaben stehen in Spalte 4 übereinander), weil sie im Testwort *milliondollars* einen Abstand zueinander haben, der der Schlüssellänge entspricht. Wird ein Schlüssel mit der Länge 7 verwendet, entstehen im Chiffretext stattdessen die Digramme *MD*, *IO*, *LL*, *LL*, *IA*, *OR* und *NS*. Je nach Schlüssellänge bildet sich also im Chiffretext eine andere Gruppe von Digrammen. Indem man alle Möglichkeiten durchspielt, kann man oft leicht die Schlüssellänge bestimmen.

Im letzten Schritt müssen nur noch die Spalten in der richtigen Reihenfolge angeordnet werden. Wenn die Spaltenzahl k klein ist, kann jedes der $k(k-1)$ Spaltenpaare auf die statistische Digrammverteilung der englischen Sprache hin untersucht werden. Der Kryptoanalytiker geht nun davon aus, dass das Spaltenpaar mit der größten Übereinstimmung richtig platziert ist. Dann wird jede Spalte als Nachfolger des gefundenen Paars durchprobiert. Die Spalte mit der besten Trigramm- und Digrammverteilung wird versuchsweise weiterverwendet. So wird auch ihre Nachfolgerin gefunden. Der Vorgang wird so lange wiederholt, bis eine Testreihenfolge entstanden ist. Die Wahrscheinlichkeit ist groß, dass der Klartext an diesem Punkt durchschaut wird (z.B. wenn *million* vorkommt, ist klar, wo der Fehler liegt).

Einige Transpositionschiffren akzeptieren einen Eingabeblock von fester Länge und produzieren einen Ausgabeblock von fester Länge. Um diese Chiffren vollständig zu beschreiben, benötigt man lediglich eine Liste, die die Reihenfolge darlegt, in der die Zeichen produziert werden. Die Chiffre in Abbildung 8.3 kann als Blockchiffre mit 64 Zeichen bezeichnet werden. Der Ausgabeblock lautet: 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, . . . , 62. Mit anderen Worten, das vierte Eingabezeichen a ist das erste Ausgabezeichen; danach kommt das zwölfte Zeichen f usw.

8.1.4 One-Time Pads

Die Ausarbeitung einer nicht knackbaren Chiffre ist eigentlich nicht schwer. Die Technik ist seit Jahrzehnten bekannt. Zuerst wählen Sie eine zufällige Bitfolge als Schlüssel. Dann konvertieren Sie den Klartext in eine Bitfolge, z.B. anhand seiner ASCII-Darstellung. Schließlich berechnen Sie das exklusive ODER (XOR) dieser beiden Zeichenfolgen Bit für Bit. Der resultierende Chiffretext kann nicht geknackt werden, da in einem ausreichend langen Chiffretext jeder Buchstabe gleich oft vorkommt, ebenso wie jedes Digramm und jedes Trigramm usw. Diese als **One-Time Pad** bezeichnete Methode ist gegen alle aktuellen und zukünftigen Angriffe gefeit, unabhängig davon, wie viel Rechenleistung ein Angreifer zur Verfügung hat. Der Grund liegt in der Informationstheorie: Es gibt einfach keine Information in der Nachricht, da alle möglichen Klartexte der gegebenen Länge gleich wahrscheinlich sind.

► Abbildung 8.4 enthält ein Beispiel zur Verwendung von One-Time Pads. Als Erstes wird die Nachricht 1, „I love you“ in 7-Bit-ASCII konvertiert. Dann wird ein einmal verwendetes (One-Time) Pad, Pad 1, gewählt und mit der Nachricht mit XOR verknüpft, um den Chiffretext zu erhalten. Ein Kryptoanalytiker könnte alle möglichen One-Time Pads ausprobieren, um herauszufinden, welcher Klartext jeweils daraus entsteht. So könnte beispielsweise das als Pad 2 aufgeführte One-Time Pad versucht werden, was zu Klartext 2, „Elvis lives“, führt, was plausibel ist oder auch nicht (aber über den Rahmen dieses Buches hinausgeht). Es gibt in der Tat für jeden 11 Zeichen langen ASCII-Klartext ein One-Time Pad, das ihn erzeugt. Dies ist mit der Aussage gemeint, dass im Chiffretext keine Informationen enthalten sind: Man kann eine beliebige Nachricht in der korrekten Länge daraus ableiten.

| | | | | | | | | | | | |
|--------------|---------|---------|---------|---------|---------|----------|----------|---------|---------|---------|---------|
| Nachricht 1: | 1001001 | 0100000 | 1101100 | 1101111 | 1110110 | 11000101 | 01000000 | 1111001 | 1101111 | 1110101 | 0101110 |
| Pad 1: | 1010010 | 1001011 | 1110010 | 1010101 | 1010010 | 1100011 | 0001011 | 0101010 | 1010111 | 1100110 | 0101011 |
| Chiffertext: | 0011011 | 1101011 | 0011110 | 0111010 | 0100100 | 0000110 | 0101011 | 1010011 | 0110000 | 0010011 | 0000101 |
| Pad 2: | 1011110 | 0000111 | 1101000 | 1010011 | 1010111 | 01000110 | 1000111 | 0111010 | 1001110 | 1110110 | 1110110 |
| Klartext 2: | 1000101 | 1101100 | 1110110 | 1101001 | 1110011 | 0100000 | 1101100 | 1101001 | 1110110 | 1100101 | 1110011 |

Abbildung 8.4: Die Verwendung eines One-Time Pads für die Verschlüsselung und die Möglichkeit, jeden möglichen Klartext aus dem Chiffertext durch Verwendung eines anderen Pads zu erhalten.

One-Time Pads sind in der Theorie hervorragend, weisen aber in der Praxis verschiedene Nachteile auf. Erstens kann man sich den Schlüssel nicht merken; deshalb muss er vom Sender und vom Empfänger aufgeschrieben werden. Aufgeschriebene Schlüssel sind selbstverständlich nicht wünschenswert, da sie abgefangen werden können. Zweitens ist die Gesamtmenge der Daten, die übertragen werden können, durch die Größe des verfügbaren Schlüssels begrenzt. Wenn ein Spion seinen Glückstag hat und eine Fülle an Daten entdeckt, kann er sie unter Umständen nicht mehr an die Zentrale zurückübertragen, weil der Schlüssel aufgebraucht wurde. Drittens ist die Methode empfindlich gegenüber verlorenen oder eingefügten Zeichen. Laufen Sender und Empfänger nicht mehr synchron, werden alle Daten von diesem Punkt an verstümmelt.

Mit der zunehmenden Verbreitung von Computern kann die One-Time-Pad-Methode für bestimmte Anwendungen sehr nützlich werden. Die Quelle des Schlüssels könnte beispielsweise eine spezielle DVD sein, die mehrere Gigabyte an Informationen enthält. Wenn sie in einer DVD-Filmbox transportiert wird und ein paar Minuten Video vorne drangehängt sind, schöpft niemand Verdacht, dass sich etwas Geheimnisvolles dahinter verbirgt. Bei den Geschwindigkeiten von Gigabit-Netzen ist das Einlegen einer anderen DVD alle 30 Sekunden selbstverständlich etwas mühsam. Zudem müssen die DVDs persönlich vom Sender zum Empfänger getragen werden, bevor Nachrichten gesendet werden können. Dies schränkt die Nützlichkeit in der Praxis natürlich ein.

Quantenkryptografie

Es gibt vielleicht eine Lösung für das Problem, wie das One-Time Pad über das Netz übertragen werden kann, und diese kommt interessanterweise aus einer gänzlich unerwarteten Ecke: der Quantenmechanik. Dieser Ansatz ist noch experimentell, aber erste Tests sind vielversprechend. Falls er sich perfektionieren und effizienter machen lässt, werden irgendwann einmal für die gesamte Kryptografie One-Time Pads eingesetzt werden, da diese nachweisbar sicher sind. Im Folgenden erläutern wir kurz, wie die **Quantenkryptografie** funktioniert. Im Besonderen beschreiben wir ein Protokoll, das nach seinen Autoren und seinem Erscheinungsjahr als **BB84** bezeichnet wird (Bennet und Brassard, 1984).

Eine Benutzerin, Alice, möchte eine sichere Verbindung über ein One-Time Pad zu einem zweiten Benutzer, Bob, aufbauen. Alice und Bob nennt man auch **Principals**, die Hauptakteure unserer Geschichte. Bob ist ein Banker, mit dem Alice Geschäfte tätigen will. Die Namen „Alice“ und „Bob“ werden für die Principals in praktisch jedem

Artikel und Buch über Kryptografie verwendet, seit Ron Rivest sie vor vielen Jahren eingeführt hat (Rivest et al., 1978). Kryptografen lieben die Tradition. Wenn wir hier „Andi“ und „Barbara“ verwenden würden, würde uns niemand mehr in diesem Kapitel etwas glauben. Also halten auch wir uns daran.

Wenn Alice und Bob ein One-Time Pad errichten könnten, wäre es ihnen möglich, darüber sicher zu kommunizieren. Die Frage lautet: Wie können sie dabei vorgehen, ohne zuvor DVDs auszutauschen? Wir können davon ausgehen, dass Alice und Bob sich jeweils am Ende eines Glasfaserkabels befinden, über das sie Lichtimpulse senden und empfangen. Ein unerschrockener Eindringling namens Trudy könnte aber das Kabel durchtrennen und aktiv anzapfen. Anschließend kann Trudy alle Bits in beide Richtungen lesen sowie in beide Richtungen falsche Nachrichten senden. Die Situation scheint für Alice und Bob aussichtslos zu sein, aber die Quantenkryptografie kann ein neues Licht auf dieses Thema werfen.

Die Quantenkryptografie basiert auf der Tatsache, dass Licht aus kleinen Paketen, sogenannten **Photonen**, besteht, die einige ganz bestimmte Eigenschaften haben. Darüber hinaus kann Licht polarisiert werden, indem es durch einen Polarisationsfilter geleitet wird, eine Tatsache, die Sonnenbrillenträgern und Fotografen wohlbekannt ist. Fällt ein Lichtstrahl (d.h. ein Strom von Photonen) durch einen Polarisationsfilter, sind alle austretenden Photonen in Richtung der Filterachse (z.B. vertikal) polarisiert. Wird der Strahl nun durch einen zweiten Polarisationsfilter geleitet, ist die Intensität des Lichts, das aus dem zweiten Filter austritt, proportional zum Quadrat des Kosinus des Winkels zwischen den Achsen. Sind die beiden Achsen lotrecht zueinander, gelangen keine Photonen durch. Die absolute Ausrichtung der beiden Filter spielt keine Rolle, nur der Winkel zwischen den Achsen zählt.

Um ein One-Time Pad zu erstellen, benötigt Alice zwei Gruppen von Polarisationsfiltern. Eine Gruppe besteht aus einem vertikalen und einem horizontalen Filter. Diese Einheit wird als **geradlinige Basis** (*rectilinear basis*) bezeichnet. Eine Basis ist nur ein Koordinatensystem. Die zweite Filtergruppe ist genauso aufgebaut, wird aber um 45 Grad gedreht, sodass ein Filter von unten links nach oben rechts verläuft und der andere Filter von oben links nach unten rechts. Diese Einheit wird als **diagonale Basis** bezeichnet. So verfügt Alice über zwei Basen, die sie schnell in ihren Strahl einbauen kann. In Wirklichkeit hat Alice jedoch keine vier separate Filter, sondern einen Kristall, dessen Polarisation elektrisch sehr schnell auf eine der vier zulässigen Richtungen umgeschaltet werden kann. Bob verfügt über die gleiche Ausrüstung wie Alice. Die Tatsache, dass Alice und Bob jeder zwei Basen zur Verfügung haben, ist für die Quantenkryptografie essenziell.

Alice weist nun einer Basis die Richtung 0 und der anderen eine 1 zu. Im unten stehenden Beispiel gehen wir davon aus, dass sie vertikal auf 0 und horizontal auf 1 setzt. Unabhängig davon wählt sie auch links unten nach rechts oben als 0 sowie oben links nach unten rechts als 1. Sie sendet Bob diese Einstellungen als Klartext.

Nun wählt Alice ein One-Time Pad, beispielsweise auf der Basis eines Zufallszahlengenerators (dies ist für sich ein komplexes Thema). Sie überträgt es Bit für Bit an Bob,

wobei für jedes Bit einer ihrer zwei Basen nach dem Zufälligkeitsprinzip ausgewählt wird. Um ein Bit zu senden, stößt die Photonenpistole ein Photon aus, das passend auf die Basis polarisiert ist, die für dieses Bit verwendet wird. Sie kann beispielsweise Basen diagonal, geradlinig, geradlinig, diagonal, geradlinig usw. wählen. Um ihr One-Time Pad 1001110010100110 mit diesen Basen zu senden, würde sie die Photonen wie in ▶ Abbildung 8.5a dargestellt senden. Wenn man das One-Time Pad und die Basenfolge kennt, so kann die Polarisation für jedes Bit eindeutig bestimmt werden. Bits, die jeweils von einem Photon getragen werden, werden **Qubits** genannt.

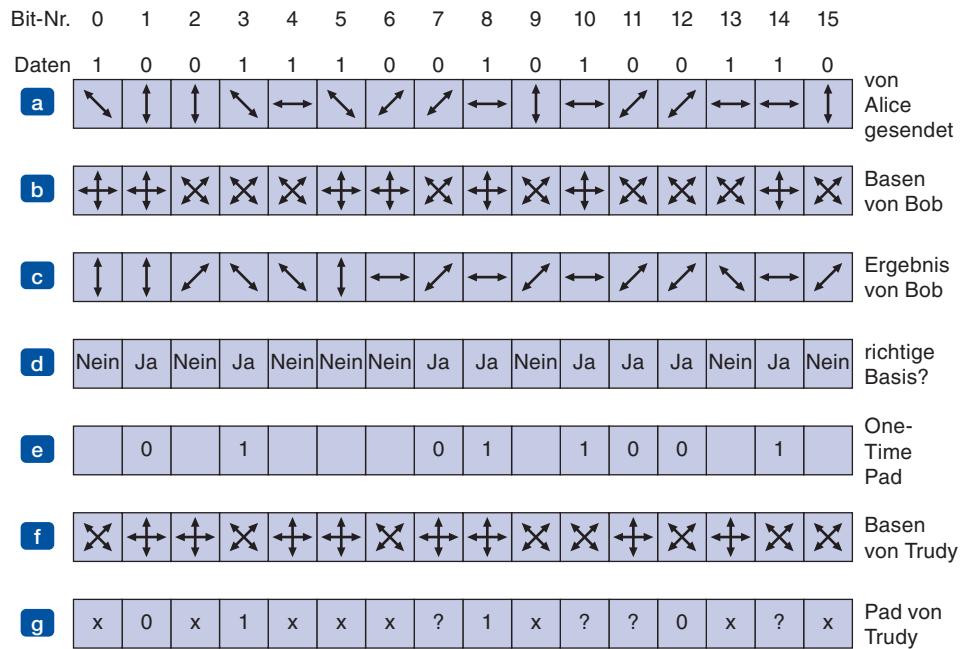


Abbildung 8.5: Beispiel für die Quantenkryptografie.

Bob weiß nicht, welche Basis er verwenden soll, daher wählt er für jedes an kommende Photon zufällig eine aus und verwendet sie, wie in ▶ Abbildung 8.5b dargestellt. Wenn er die richtige Basis auswählt, erhält er das richtige Bit. Wenn er die falsche Basis auswählt, erhält er ein Zufallsbit. Denn wenn ein Photon auf einen Filter auftrifft, der um 45 Grad zu seiner eigenen Polarisation polarisiert ist, springt es mit gleicher Wahrscheinlichkeit und zufällig in die Polarisation des Filters oder in eine Polarisation, die senkrecht zum Filter ist. Diese Eigenschaft der Photonen ist für die Quantenmechanik von grundlegender Bedeutung. So sind einige Bits korrekt, andere zufällig, aber Bob weiß nicht welche. Die Ergebnisse von Bob werden in ▶ Abbildung 8.5c dargestellt.

Wie findet Bob nun heraus, welche Basen richtig sind und welche falsch? Er teilt Alice im Klartext einfach mit, welche Basis er für jedes Bit verwendet hat, und sie sagt ihm im Klartext, welche richtig und falsch sind, wie in ▶ Abbildung 8.5d dargestellt. Mit dieser Information können beide eine Bitfolge der korrekten Treffer aufbauen, wie

in ▶ Abbildung 8.5e dargestellt. Im Durchschnitt ist diese Bitfolge halb so lang wie die ursprüngliche Bitfolge, da sie aber beide Parteien kennen, können sie sie als One-Time Pad verwenden. Alice muss also nur eine Bitfolge übertragen, die etwas länger als zweimal die gewünschte Länge ist, und sie und Bob erhalten ein One-Time Pad in der gewünschten Länge. Problem gelöst.

Aber, Moment! Wir haben Trudy vergessen. Angenommen sie möchte gerne wissen, was Alice sagt, und schneidet das Glasfaserkabel auf und installiert ihren eigenen Detektor und Sender. Leider weiß auch sie nicht, welche Basis für jedes Photon verwendet wurde. Das Beste, was Sie tun kann, ist, wie Bob für jedes Photon zufällig eine Basis auszuwählen. Ein Beispiel für ihre abgefangenen Werte ist in ▶ Abbildung 8.5f dargestellt. Wenn Bob (in Klartext) berichtet, welche Basen er verwendet hat, und Alice ihm (in Klartext) mitteilt, welche richtig sind, weiß Trudy, was richtig und was falsch ist. In Abbildung 8.5 hat sie es für die Bits 0, 1, 2, 3, 4, 6, 8, 12 und 13 getroffen. Aber sie weiß aus der Antwort von Alice in Abbildung 8.5d, dass nur die Bits 1, 3, 7, 8, 10, 11, 12 und 14 zum One-Time Pad gehören. Vier dieser Bits hat sie korrekt erraten (1, 3, 8 und 12) und das richtige Bit abgefangen. Bei den anderen vier (7, 10, 11 und 14) hat sie falsch geraten und kennt das übertragene Bit nicht. Daher weiß Bob, dass das One-Time Pad mit 01011001 beginnt, wie in Abbildung 8.5e, Trudy hat aber nur 01?1??0?, wie in ▶ Abbildung 8.5g.

Natürlich sind sich Alice und Bob bewusst, dass Trudy Teile ihres One-Time Pads abgefangen haben kann, sodass sie daran interessiert sind, dass Trudy möglichst wenig Informationen erhält. Hierzu können sie eine Transformation darauf ausführen. Sie könnten beispielsweise das One-Time Pad in 1 024-Bit-Blöcke aufteilen und diese quadrieren, um eine 2 048-Bit-Zahl zu erstellen, und die Verkettung dieser 2 048-Bit-Zahlen als One-Time Pad verwenden. Mit der teilweisen Kenntnis der übertragenen Bitfolge hat Trudy keine Möglichkeit, ein Quadrat zu erzeugen, und kann so nichts damit anfangen. Die Umwandlung des ursprünglichen One-Time Pad in ein anderes, das Trudy weniger Informationen zukommen lässt, wird als **Privacy Amplification** (Geheimhaltungsverstärkung) bezeichnet. In der Praxis werden hier anstatt der Quadratration komplexe Transformationen verwendet, bei denen jedes Ausgabebit von jedem Eingabebit abhängt.

Arme Trudy. Weder hat sie eine Ahnung, wie das One-Time Pad aussieht, noch kann sie ihre Anwesenheit länger geheim halten. Denn nun muss sie jedes empfangene Bit an Bob weitergeben, um ihn glauben zu lassen, er kommuniziere mit Alice. Das Problem ist, dass sie hier nichts anderes tun kann, als das empfangene Qubit zu übertragen, wobei sie die zum Empfang verwendete Polarisation benutzt. Hier liegt sie aber die Hälfte der Zeit falsch und erzeugt viele Fehler im One-Time Pad von Bob.

Wenn Alice dann mit der Übertragung der Daten beginnt, codiert sie sie mit einem aufwendigen Vorwärtsfehlerkorrekturcode. Aus der Sicht von Bob ist ein 1-Bit-Fehler im One-Time Pad das Gleiche wie ein 1-Bit-Übertragungsfehler. In beiden Fällen erhält er das falsche Bit. Wenn Vorwärtsfehlerkorrektur in ausreichendem Maße eingesetzt wird, kann er die ursprüngliche Nachricht trotz aller Fehler wiederherstellen. Aber er kann auch sehr einfach zählen, wie viele Fehler korrigiert wurden. Ist diese

Zahl weitaus höher als die erwartete Fehlerrate der eingesetzten Geräte, dann weiß er, dass Trudy die Leitung angezapft hat, und kann dementsprechend handeln (z.B. Alice mitteilen, auf Funk umzustellen, die Polizei anzurufen usw.). Könnte Trudy ein Photon klonen, sodass sie in der Lage wäre, ein Photon zu untersuchen und ein identisches an Bob zu senden, so könnte sie der Entdeckung entgehen. Derzeit besteht aber diese Möglichkeit nicht. Aber selbst wenn Trudy Photonen klonen könnte, mindert dies nicht den Wert der Quantenkryptografie zur Errichtung von One-Time Pads.

Obwohl die Quantenkryptografie bereits über eine Entfernung von 60 km in Glasfaserkabeln funktioniert, ist die Ausstattung komplex und teuer. Aber die Idee ist vielversprechend. Weitere Informationen über die Quantenkryptografie finden Sie in Mullins (2002).

8.1.5 Zwei Grundprinzipien der Verschlüsselung

Allen im weiteren Verlauf untersuchten Verschlüsselungssystemen liegen zwei Prinzipien zugrunde. Lesen Sie sorgfältig. Wenn Sie sie verletzen, geschieht dies auf eigenes Risiko.

Redundanz

Das erste Prinzip ist, dass alle verschlüsselten Nachrichten in irgendeiner Form Redundanz aufweisen, d.h. sie enthalten Informationen, die zum Verständnis der Nachricht nicht erforderlich sind. Ein Beispiel soll dies verdeutlichen. Betrachten wir eine Internetversandfirma namens The Couch Potato (TCP) mit einem Gesamtangebot von 60 000 Artikeln. Da die TCP-Programmierer sehr effizient sein möchten, entschließen sie sich, dass ihre Auftragsnachrichten aus einem 16 Byte langen Kundennamen, gefolgt von einem 3-Byte-Datenfeld (1 Byte für die Menge und 2 Byte für die Artikelnummer) bestehen. Die letzten 3 Byte sollen mit einem sehr langen Schlüssel, den nur der Kunde und TCP kennen, verschlüsselt werden.

Auf den ersten Blick scheint die Lösung sicher zu sein. In gewissem Sinne ist sie das auch, weil bei passiven Lauschangriffen die Nachrichten nicht entschlüsselt werden können. Leider weist sie einen gravierenden Fehler auf, durch den sie unbrauchbar wird. Nehmen wir an, dass eine kürzlich entlassene Mitarbeiterin, sich an ihrer Ex-Firma TCP rächen will. An ihrem letzten Arbeitstag nimmt sie die Kundenliste mit. Sie arbeitet die ganze Nacht an einem Programm, um fiktive Bestellungen mit echten Kundennamen zu erzeugen. Da sie keine Liste der Schlüssel hat, setzt sie in die letzten 3 Byte einfach zufällige Zahlen ein und sendet Hunderte von Bestellungen an TCP.

Beim Eintreffen dieser Nachrichten sucht der TCP-Computer den Schlüssel mithilfe des Kundennamens heraus und entschlüsselt die Nachricht. Leider ist fast jede 3-Byte-Nachricht gültig. Folglich beginnt der Computer mit dem Ausdrucken von Lieferanweisungen. Es mag zwar seltsam erscheinen, dass ein Kunde 837 Kinderschaukeln oder 540 Sandkästen bestellt, aber den Computer tangiert das nicht. Es könnte ja sein, dass der Kunde vorhat, eine Reihe von Spielplätzen zu bauen. Auf diese Weise kann ein aktiver Eindringling wie hier die gefeuerte Mitarbeiterin einen sehr großen Schaden anrichten, obwohl sie die von ihrem Computer erzeugten Nachrichten nicht versteht.

Das Problem kann durch das Hinzufügen von Redundanz in allen Nachrichten gelöst werden. Erstrecken sich Bestellungen z.B. über 12 Byte, von denen die ersten 9 Nullen sein müssen, funktioniert diese Attacke nicht mehr, weil die Ex-Angestellte keinen großen Strom gültiger Nachrichten erzeugen kann. Die Moral von der Geschichte ist, dass alle Nachrichten eine beträchtliche Redundanz aufweisen müssen, damit aktive Eindringlinge nicht massenweise zufallsgesteuerten Schrott versenden können, der von Computern als gültige Nachricht interpretiert wird.

Andererseits ist es für Kryptoanalytiker bei größerer Redundanz auch einfacher, Nachrichten aufzubrechen. Nehmen wir an, dass das Versandgeschäft ein sehr umkämpfter Markt ist und der größte Konkurrent von TCP, The Sofa Tuber, liebend gerne wissen möchte, wie viele Sandkästen TCP verkauft. Die Konkurrenz arrangiert also, dass die Telefonleitung von TCP angezapft wird. In dem ursprünglichen Schema mit 3-Byte-Nachrichten war die Kryptoanalyse fast unmöglich, weil der Kryptoanalytiker nach dem Raten eines Schlüssels keine Möglichkeit hatte festzustellen, ob er richtig geraten hatte, da schließlich fast jede Nachricht technisch zulässig wäre. Bei dem neuen 12-Byte-Schema ist es für den Kryptoanalytiker einfach, gültige von ungültigen Nachrichten zu unterscheiden.

Daraus leiten wir Folgendes ab:

Kryptografisches Prinzip 1: Nachrichten müssen eine gewisse Redundanz enthalten.

Anders ausgedrückt, der Empfänger muss bei der Entschlüsselung der Nachricht angeben können, ob das Erhaltene gültig ist, indem er sie einfach untersucht und eventuell noch eine einfache Berechnung ausführt. Diese Redundanz ist erforderlich, um aktive Eindringlinge davon abzuhalten, Müll zu senden und den Empfänger dazu zu bringen, den Müll in „KlarTEXT“ zu entschlüsseln und dementsprechend zu handeln. Andererseits vereinfacht gerade diese Redundanz passiven Eindringlingen, in das System einzubrechen, sodass hier eine gewisse Spannung vorliegt. Des Weiteren sollte Redundanz nie in Form von n Nullen am Anfang oder Ende einer Nachricht realisiert werden, da dies die Arbeit eines Kryptoanalytikers vereinfachen würde, der besser vorhersagbare Ergebnisse erhält, wenn er die Nachrichten mit kryptografische Algorithmen bearbeitet. Ein CRC-Polynom ist viel besser als eine Folge von Nullen, da der Empfänger dies sehr einfach prüfen kann, es aber mehr Arbeit für den Kryptoanalytiker bedeutet. Noch besser ist es, ein kryptografisches Hash zu verwenden, ein Konzept, das wir später behandeln. Im Moment reicht es, wenn Sie sich ein Hash als besseren CRC vorstellen.

Wenn wir kurz zur Quantenkryptografie zurückgehen, sehen wir auch, welche Rolle die Redundanz dort spielt. Da Trudy die Photonen abfängt, werden einige Bits im One-Time Pad von Bob falsch sein. Bob benötigt eine gewisse Redundanz bei den eingehenden Nachrichten, um festzustellen, ob Fehler vorhanden sind. Eine sehr drastische Form der Redundanz ist, die Nachricht zweimal zu wiederholen. Sind die beiden Kopien nicht identisch, weiß Bob, dass entweder das Rauschen im Glasfaserkabel sehr hoch ist oder jemand versucht, die Übertragung abzuhören. Natürlich ist es des Guten zu viel, alles zweimal zu senden. Ein Hamming- oder Reed-Solomon-Code ist sehr viel effizienter für die Fehlerentdeckung und -korrektur. Es sollte aber klar sein, dass eine

gewisse Redundanz erforderlich ist, um die gültigen Nachrichten von den ungültigen zu unterscheiden, besonders angesichts aktiver Eindringlinge.

Aktualität

Laut dem zweiten kryptografischen Prinzip müssen Maßnahmen ergriffen werden, um sicherzustellen, dass jede empfangene Nachricht als aktuell (d.h. vor sehr kurzer Zeit gesendet) verifiziert werden kann. Mit dieser Maßnahme werden aktive Eindringlinge gehindert, alte Nachrichten erneut abzuspielen. Ohne solche Maßnahmen könnte die gefeuerte Mitarbeiterin vom obigen Beispiel die Telefonleitung von TCP anzapfen und einfach alte gültige Nachrichten wiederholen. Es gilt also:

Kryptografisches Prinzip 2: *Eine Methode zur Vereitung von Replay-Angriffen (Wiedergabe alter Nachrichten) ist erforderlich.*

Eine geeignete Maßnahme ist das Einfügen eines Zeitstempels in jede Nachricht, der nur kurzfristig gültig ist, z.B. 10 Sekunden. Der Empfänger kann die Nachrichten nur für 10 Sekunden behalten und die neu ankommenen Nachrichten mit den früheren vergleichen, um Duplikate auszufiltern. Nachrichten, die älter sind als 10 Sekunden, werden entfernt, da auch Kopien, die mehr als 10 Sekunden später ankommen, als veraltet zurückgewiesen werden. Neben dem Zeitstempel gibt es weitere Methoden, die später beschrieben werden.

8.2 Algorithmen für die symmetrische Verschlüsselung

In der modernen Kryptografie werden die gleichen Grundprinzipien verwendet wie in der traditionellen Kryptografie – Transposition und Substitution –, aber mit einem anderen Schwerpunkt. Traditionell haben Kryptografen einfache Algorithmen verwendet. Heute trifft das Gegenteil zu: Das Ziel ist, den Verschlüsselungsalgorithmus so komplex auszulegen, dass sogar erfahrene Kryptoanalytiker mit einer riesigen Menge von verschlüsseltem Text ihrer Wahl nichts ohne Schlüssel anfangen können.

Die erste Klasse der Verschlüsselungsalgorithmen, die wir in diesem Kapitel behandeln, werden als **Algorithmen mit symmetrischen Schlüsseln** (*symmetric-key algorithm*) bezeichnet, da sie für die Ver- und Entschlüsselung denselben Schlüssel verwenden. Abbildung 8.2 zeigt die Verwendung eines symmetrischen Algorithmus. Wir werden uns hier vor allem auf **Blockchiffren** konzentrieren, die einen n -Bit-Block Klartext als Eingabe verwenden und diesen unter Verwendung eines Schlüssels in einen n -Bit-Block Chiffretext umwandeln.

Kryptografische Algorithmen können entweder in der Hardware (höhere Geschwindigkeit) oder in der Software (höhere Flexibilität) implementiert werden. Obwohl wir uns überwiegend mit Algorithmen und Protokollen beschäftigen, die von der eigentlichen Implementierung unabhängig sind, wollen wir doch kurz auf den Bau kryptografischer Hardware eingehen. Transpositionen und Substitutionen können mit einfachen Schaltungen implementiert werden. ►Abbildung 8.6a zeigt ein Gerät, das **P-Box** genannt wird (P steht für Permutation). Dieses Gerät führt eine Transposition für eine

8-Bit-Eingabe durch. Wenn die 8 Bit von oben nach unten als 01234567 bezeichnet sind, lautet die Ausgabe dieser P-Box 36071245. Bei der entsprechenden internen Verdrahtung kann eine P-Box jede beliebige Transposition praktisch in Lichtgeschwindigkeit ausführen, da keine Berechnung erforderlich ist, sondern nur Signale weitergeleitet werden. Der Entwurf folgt dem Prinzip von Kerckhoffs: Der Angreifer weiß, dass die allgemeine Vorgehensweise das Vertauschen von Bits ist. Was er nicht weiß, ist, an welcher Position die jeweiligen Bits landen.

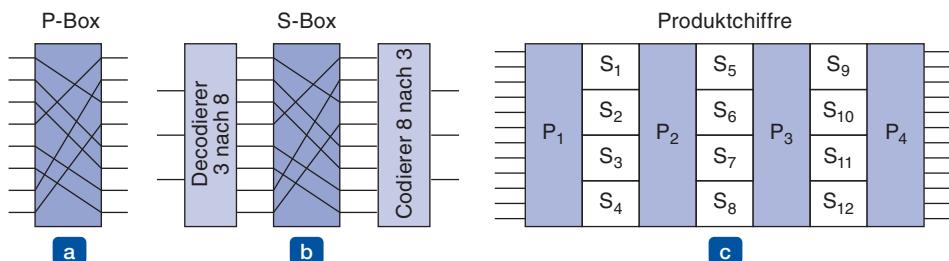


Abbildung 8.6: Grundelemente von Produktchiffren: (a) P-Box, (b) S-Box, (c) Produkt.

Substitutionen können mit **S-Boxen** durchgeführt werden, wie in ▶ Abbildung 8.6b dargestellt. In diesem Beispiel wird ein 3-Bit-Klartext eingegeben und ein 3-Bit-Chiffrentext ausgegeben. Die 3-Bit-Eingabe wählt eine der acht Leitungen aus, die aus der ersten Stufe herauskommen, und setzt sie auf 1. Alle anderen Leitungen stehen auf 0. Die zweite Stufe ist eine P-Box. In der dritten Stufe wird die ausgewählte Eingabeleitung wieder in Binärform verschlüsselt. Werden bei der angegebenen Verdrahtung die acht oktalen Zahlen 01234567 nacheinander eingegeben, würde die Ausgabefolge 24506713 lauten. Anders ausgedrückt, 0 wurde durch 2 ersetzt, 1 durch 4 usw. Auch hier kann durch eine entsprechende Verdrahtung der P-Box in der S-Box jede Substitution vorgenommen werden. Darüber hinaus kann ein solches Gerät hardwaremäßig gebaut werden und eine hohe Geschwindigkeit erzielen, da Codierer und Decodierer nur ein oder zwei Gatterverzögerungen (unterhalb des Nanosekundenbereichs) haben. Die Weiterleitungszeit über eine P-Box kann unter einer Pikosekunde liegen.

Die wahre Stärke dieser Grundelemente zeigt sich erst, wenn mehrere Boxen hintereinander geschaltet werden, sodass sie wie in ▶ Abbildung 8.6c eine **Produktchiffre** bilden. In diesem Beispiel werden zwölf Eingangsleitungen in der ersten Stufe (P_1) transponiert, also permutiert. In der zweiten Stufe wird die Eingabe dann in vier Gruppen von je 3 Bit aufgeteilt, die jeweils unabhängig voneinander ersetzt werden (S_1 bis S_4). Diese Anordnung zeigt eine Methode zur Annäherung einer größeren S-Box aus vielen kleineren S-Boxen. Ihr Vorteil liegt darin, dass kleine S-Boxen für eine Hardware-Implementierung praktischer sind (z.B. kann eine 8-Bit-S-Box als Nachschlagetabelle mit 256 Einträgen realisiert werden), während große S-Boxen zu unhandlich werden, um sie zu bauen (z.B. würde eine 12-Bit-S-Box mindestens $2^{12}=4\,096$ gekreuzte Drähte in der mittleren Stufe beanspruchen). Diese Methode ist zwar weniger allgemein, aber leistungsfähig genug. Durch Einbinden einer ausreichend großen Zahl von

Stufen in die Produktchiffre kann die Ausgabe das Ergebnis einer extrem komplizierten Funktion der Eingabe sein.

Produktchiffren, die k Bit Eingaben verarbeiten, um k Bit Ausgaben zu erzeugen, sind sehr gängig. In der Regel liegt k zwischen 64 und 256. Eine Hardware-Implementierung hat in der Regel mindestens zehn physikalische Stufen statt der in Abbildung 8.6c gezeigten sieben. Eine Software-Implementierung ist als Schleife mit mindestens acht Iterationen programmiert, wobei jede Schleife Substitutionen vom Typ S-Box auf Unterblöcken des zwischen 64 Bit und 256 Bit großen Datenblocks durchführt, gefolgt von einer Permutation, die die Ausgaben der S-Boxen mischt. Oftmals findet auch zu Beginn und am Ende eine spezielle Permutation statt. In der Literatur werden diese Iterationen als **Runden** (*round*) bezeichnet.

8.2.1 DES – Data Encryption Standard

Im Januar 1977 übernahm die US-Regierung eine von IBM entwickelte Produktchiffre als offizielle Norm für nicht geheime Nachrichten. Dies wiederum hat viele Hersteller dazu bewogen, diesen Verschlüsselungsalgorithmus, **DES** (*Data Encryption Standard*, Datenverschlüsselungsstandard), für ihre Sicherheitsprodukte zu übernehmen. Er ist in seiner ursprünglichen Form heute nicht mehr sicher, in einer modifizierten Version aber noch nützlich. Wie DES funktioniert, wird im Folgenden beschrieben.

►Abbildung 8.7a zeigt eine Übersicht über DES. Der Klartext wird zunächst in Blöcken von 64 Bit verschlüsselt, wobei ein Chiffretext mit einer Länge von 64 Bit entsteht. Der Algorithmus, der von einem 56-Bit-Schlüssel parametrisiert wird, ist in 19 Einzelschritte aufgeteilt. Der erste Schritt ist eine schlüsselunabhängige Transposition im 64-Bit-Klartext. Der letzte Schritt ist dann die genaue Umkehrung dieser Transposition. In der vorletzten Stufe werden die 32 Bit ganz links mit den 32 Bit ganz rechts vertauscht. Die restlichen 16 Stufen sind in ihrer Funktionsweise identisch, werden jedoch von verschiedenen Funktionen des Schlüssels parametrisiert. Dieser Algorithmus ist so ausgelegt, dass die Nachricht mit dem gleichen Schlüssel entschlüsselt werden kann, mit dem sie verschlüsselt wurde – eine Eigenschaft, die bei allen symmetrischen Verschlüsselungen erforderlich ist. Die einzelnen Schritte laufen lediglich in der umgekehrten Reihenfolge ab.

Der Einsatz einer dieser Zwischenstufen ist in ►Abbildung 8.7b dargestellt. Jede Stufe verarbeitet zwei 32-Bit-Eingaben und produziert zwei 32-Bit-Ausgaben. Die linke Ausgabe ist einfach eine Kopie der rechten Eingabe. Die rechte Ausgabe ist ein bitweises XOR auf der linken Eingabe und einer Funktion, welche die rechte Eingabe und den Schlüssel dieser Stufe, K_i , als Funktionswerte hat. Die gesamte Komplexität des Algorithmus liegt in dieser Funktion.

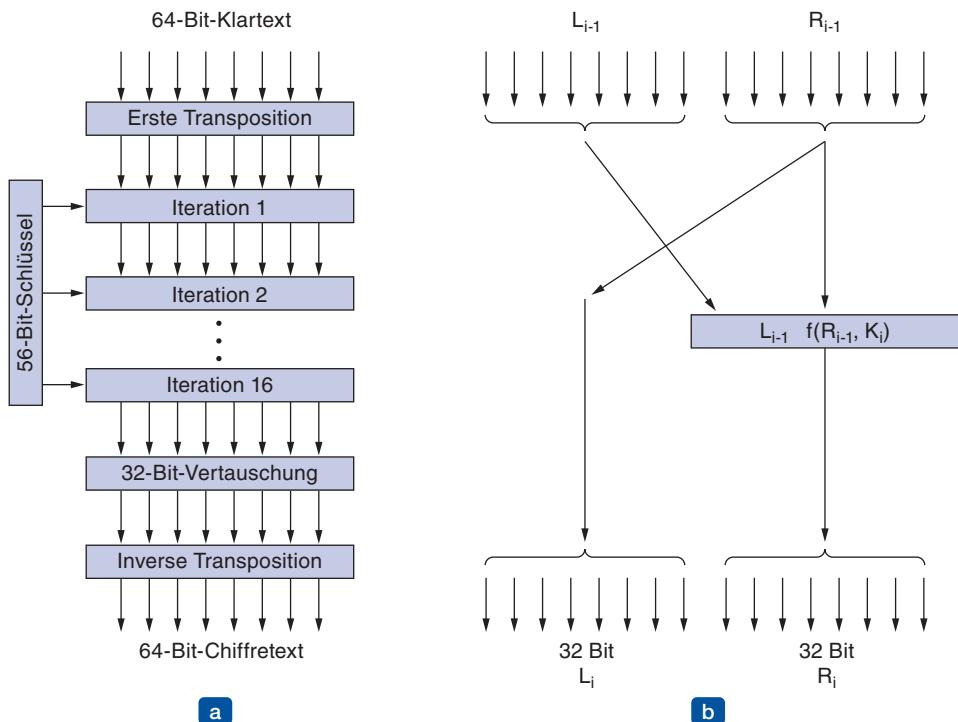


Abbildung 8.7: Der DES-Standard: (a) allgemeine Übersicht, (b) Detailansicht einer Iteration.
Das + im Kreis steht für exklusives ODER.

Die Funktion besteht aus vier Stufen, die nacheinander ausgeführt werden. Zuerst wird eine Zahl E mit einer Länge von 48 Bit konstruiert, indem die 32-Bit-Zahl R_{i-1} durch eine feste Transpositions- und Verdoppelungsregel erweitert wird. Zweitens werden E und K_i über XOR verknüpft. Dieses Ergebnis wird dann in acht Gruppen zu je sechs Bit unterteilt. Jede dieser acht Gruppen gelangt in eine eigene S-Box. Jede der 64 möglichen Eingaben in eine S-Box wird auf eine 4-Bit-Ausgabe abgebildet. Schließlich werden diese 8×4 Bit durch eine P-Box geschleust.

Bei jeder der 16 Iterationen wird ein anderer Schlüssel verwendet. Bevor der Algorithmus beginnt, wird eine 56-Bit-Transposition auf den Schlüssel angewendet. Vor jeder Iteration wird der Schlüssel dann in zwei 28-Bit-Einheiten geteilt, die beide um eine Zahl nach links rotiert werden, die von der Nummer der Iteration abhängig ist. K_i wird ermittelt, indem auf diesen rotierten Schlüssel nochmals eine 56-Bit-Transposition angewendet wird. Eine andere 48-Bit-Teilmenge der 56 Bit wird extrahiert und in jeder Runde permuiert.

DES wird manchmal mit einem Verfahren namens **Whitening** wirksamer gemacht. Dabei wird jeder Klartextblock, bevor er in DES eingespeist wird, mit einem zufälligen 64-Bit-Schlüssel XOR-verknüpft. Dann wird ein zweiter 64-Bit-Schlüssel mit dem resultierenden Chiffretext über XOR verkettet, bevor dieser versendet wird. Whitening kann ganz einfach durch die umgekehrten Operationen entfernt werden (wenn der

Empfänger die zwei Whitening-Schlüssel besitzt). Da dieses Verfahren effektiv die Schlüssellänge um weitere Bit erhöht, wird die ausgiebige Suche im Schlüsselraum sehr viel zeitaufwendiger. Beachten Sie, dass für jeden Block der gleiche Whitening-Schlüssel verwendet wird (d.h. es gibt nur einen Whitening-Schlüssel).

DES wurde vom ersten Tag an unter heftigen Kontroversen entwickelt. Es gründete auf einer von IBM entwickelten und patentierten Chiffre namens *Lucifer*, wobei die IBM-Chiffre nicht auf einem 56-Bit-, sondern auf einem 128-Bit-Schlüssel basierte. Als die US-Regierung eine Chiffre für nicht geheime Zwecke standardisieren wollte, wurde IBM „eingeladen“, die Angelegenheit mit dem amerikanischen Geheimdienst NSA, dem Codeknackerarm der US-Regierung, zu „diskutieren“. NSA als größte Arbeitgeber für Mathematiker und Kryptologen weltweit ist so geheim, dass in der Branche folgender Witz die Runde macht:

Frage: Was bedeutet die Abkürzung NSA?

Antwort: No Such Agency (Keine solche Behörde).

NSA ist die Abkürzung von *National Security Agency* (Behörde für nationale Sicherheit).

Nach diesen Diskussionen reduzierte IBM den Schlüssel von 128 auf 56 Bit und entschloss sich, den Prozess, mit dem DES entwickelt wurde, für sich zu behalten. Viele hatten damals den Verdacht, dass die Schlüssellänge reduziert wurde, um sicherzustellen, dass nur NSA DES knacken konnte, aber keine Organisation mit einem kleineren Budget. Der Zweck der Geheimhaltung des Entwurfs war angeblich der, eine Hintertür zu verbergen, die es NSA noch einfacher machen würde, DES aufzubrechen. Als ein NSA-Mitarbeiter IEEE diskret bat, eine geplante Konferenz über Kryptografie abzusagen, machte das viele noch unruhiger. NSA stritt alles ab.

1977 entwarfen zwei Stanford-Wissenschaftler, Diffie und Hellman (1977), einen Rechner, um DES aufzubrechen. Ihren Schätzungen zufolge ließe sich ein solcher Rechner für 20 Millionen US-Dollar realisieren. Mit einem kurzen Klartext und dem passenden Chiffertext konnte dieser Rechner durch erschöpfende Suche des 2^{56} -Eingabeschlüsselraums in weniger als einem Tag den Schlüssel finden. Heute gibt es einen solchen Rechner bereits, er ist zu kaufen und kostet weniger als 8 000 Euro (Kumar et al., 2006).

Triple DES

Bereits 1979 erkannte IBM, dass die DES-Schlüssellänge zu kurz ist, und entwickelte eine Methode, sie durch dreifache Verschlüsselung deutlich zu erhöhen (Tuchman, 1979). Die gewählte Methode, die inzwischen im internationalen Standard 8732 veröffentlicht wurde, ist in ► Abbildung 8.8 aufgezeigt. Hier werden zwei Schlüssel und drei Stufen benutzt. In der ersten Stufe wird der Klartext mittels DES in der üblichen Weise mit K_1 verschlüsselt. In der zweiten Stufe wird DES im Entschlüsselungsmodus mit K_2 als Schlüssel ausgeführt. Dann folgt eine weitere Verschlüsselung mit K_1 .

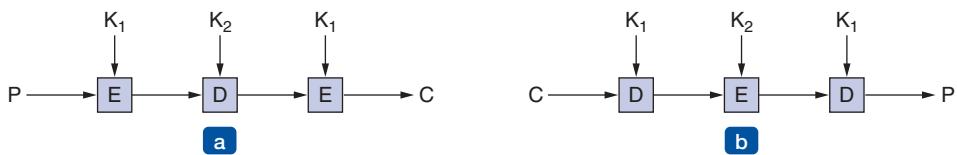


Abbildung 8.8: (a) Dreifache Verschlüsselung mit DES. (b) Entschlüsselung.

Dieses Design wirft zwei Fragen auf. Erstens: warum werden nur zwei Schlüssel verwendet anstatt drei? Zweitens: warum wird **EDE** (*Encrypt Decrypt Encrypt*, Verschlüsselung Entschlüsselung Verschlüsselung) anstelle von **EEE** (*Encrypt Encrypt Encrypt*, Verschlüsselung Verschlüsselung Verschlüsselung) verwendet? Der Grund, warum nur zwei Schlüssel benutzt werden, liegt darin, dass auch extrem paranoide Kryptografen zugeben, dass 112 Bit derzeit für kommerzielle Anwendungen ausreichen. (Und unter Kryptografen wird Paranoia als Eigenschaft, nicht als Fehler betrachtet.) Eine Verlängerung auf 168 Bit würde nur unnötigen Overhead für die Verwaltung und Übertragung eines weiteren Schlüssels bedeuten.

Durch Verschlüsselung, Entschlüsselung und erneute Verschlüsselung soll die Abwärtskompatibilität mit vorhandenen DES-Systemen, die mit einem Schlüssel arbeiten, sichergestellt werden. Sowohl die Ver- als auch die Entschlüsselungsfunktion sind Zuordnungen zwischen Gruppen von 64-Bit-Zahlenblöcken. Aus kryptografischer Sicht sind die zwei Zuordnungen gleich stark. Durch Verwendung von EDE anstelle von EEE kann ein Rechner, der die dreifache Verschlüsselung anwendet, mit einem anderen kommunizieren, der nur mit einem Schlüssel arbeitet, indem einfach $K_1 = K_2$ gesetzt wird. Durch diese Eigenschaft kann allmählich auf die dreifache Verschlüsselung umgestellt werden, was für akademische Kryptografen kein Thema ist, für IBM und seine Kunden aber große Bedeutung hat.

8.2.2 AES – Advanced Encryption Standard

Als sich abzeichnete, dass DES, selbst mit Triple DES, nicht mehr lange von großem Nutzen sein würde, entschied **NIST** (*National Institute of Standards and Technology*), eine Behörde des amerikanischen Handelsministeriums, dass die Regierung einen neuen kryptografischen Standard für nicht geheime Einsatzbereiche benötigte. NIST waren die Kontroversen um DES bekannt, sodass klar war, dass nicht einfach ein neuer Standard bekannt gegeben werden konnte, da sonst jeder, der sich auch nur ansatzweise in Kryptografie auskannte, automatisch angenommen hätte, die NSA hätte daran mitgewirkt, um alle damit verschlüsselten Nachrichten lesen zu können. Unter diesen Bedingungen würde vermutlich niemand den Standard einsetzen, und er würde sang- und klanglos dahinscheiden.

So schlug NIST für einen staatlichen Apparat einen erstaunlichen Weg ein: Es sponserte einen kryptografischen Wettbewerb. Im Januar 1997 wurden Wissenschaftler aus der ganzen Welt eingeladen, Vorschläge für einen neuen Standard einzureichen, der als **AES** (*Advanced Encryption Standard*, erweiterter Verschlüsselungsstandard) bezeichnet wurde. Die Wettbewerbsregeln lauteten:

1. Der Algorithmus muss eine symmetrische Blockchiffre sein.
2. Das gesamte Design muss öffentlich sein.
3. Schlüssel der Länge 128, 192 und 256 Bit müssen unterstützt werden.
4. Die Implementierung muss sowohl in Software wie auch in Hardware möglich sein.
5. Die Algorithmen müssen öffentlich oder zu nicht diskriminierenden Bedingungen lizenziert sein.

Es wurden 15 ernsthafte Vorschläge eingereicht, die in öffentlichen Konferenzen vorgestellt wurden. Die Teilnehmer wurden sogar ausdrücklich aufgefordert, nach Fehlern in den Vorschlägen zu suchen. Im August 1998 wählte NIST die fünf Finalisten vor allem unter Berücksichtigung von Sicherheit, Effizienz, Einfachheit, Flexibilität und Speicheranforderungen (wichtig für eingebettete Systeme). Es folgten weitere Konferenzen, um den Besten zu ermitteln.

Im Oktober 2000 gab NIST bekannt, dass es sich für Rijndael von Joan Daemen und Vincent Rijmen entschieden hatte. Der Name Rijndael, ausgesprochen (ungefähr) Rheindoll, leitet sich aus den Nachnamen der Urheber ab: Rijmen + Daemen. Im November 2001 wurde Rijndael der Standard der amerikanischen Regierung, der als Federal Information Processing Standard FIPS 197 veröffentlicht wurde. Aufgrund der außergewöhnlichen Offenheit des Wettbewerbs, der technischen Eigenschaften von Rijndael und der Tatsache, dass die Gewinner zwei junge belgische Kryptografen waren (die wohl kaum eine Hintertür eingebaut haben, um der NSA eine Freude zu machen), hat sich Rijndael als führende kryptografische Chiffre etabliert. AES-Ver- und Entschlüsselung ist inzwischen Teil des Befehlssatzes einiger Mikroprozessoren (z.B. Intel).

Rijndael unterstützt Schlüssellängen und Blockgrößen von 128 bis 256 Bit in 32-Bit-Schritten. Schlüssellänge und Blocklänge können unabhängig voneinander ausgewählt werden. AES gibt aber vor, dass die Blockgröße 128 Bit und die Schlüssellänge 128, 192 oder 256 Bit betragen muss. Es ist zweifelhaft, ob 192-Bit-Schlüssel jemals verwendet werden, sodass AES de facto zwei Varianten hat: einen 128-Bit-Block mit einem 128-Bit-Schlüssel und einen 128-Bit-Block mit einem 256-Bit-Schlüssel.

In unserer unten stehenden Behandlung des Algorithmus untersuchen wir nur den Fall 128/128, da dieser höchstwahrscheinlich der kommerzielle Standard werden wird. Ein 128-Bit-Schlüssel erlaubt einen Schlüsselraum von $2^{128} \approx 3 \times 10^{38}$ Schlüssel. Selbst wenn es der NSA gelingt, einen Rechner mit 1 Milliarde paralleler Prozessoren zu bauen, von denen jeder einen Schlüssel pro Pikosekunde auswerten kann, würde solch ein Rechner etwa 10^{10} Jahre benötigen, um den Schlüsselraum zu durchsuchen. Bis dahin ist die Sonne erloschen, sodass die dann Lebenden die Ergebnisse bei Kerzenschein lesen müssen.

Rijndael

Aus mathematischer Sicht basiert Rijndael auf der Galois-Feldtheorie, durch die er einige beweisbare Sicherheitseigenschaften erhält. Man kann dies aber auch als C-Code betrachten, ohne in die Mathematik einzusteigen.

Wie DES verwendet Rijndael Substitution und Permutationen sowie mehrere Runden. Die Anzahl der Runden hängt von der Schlüsselgröße und der Blockgröße ab und beträgt 10 für 128-Bit-Schlüssel mit 128-Bit-Blöcken und bis zu 14 für den größten Schlüssel oder den größten Block. Aber im Unterschied zu DES verwenden alle Operationen ganze Bytes, um die effiziente Implementierung in Hardware und Software zu ermöglichen. Der Code wird ansatzweise in ▶ Abbildung 8.9 beschrieben.

```
#define LENGTH 16                                     /* Bytezahl des Datenblocks oder
                                                       Schlüssels */
#define NROWS 4                                       /* Zeilenzahl im Array state */
#define NCOLS 4                                       /* Spaltenzahl im Array state */
#define ROUNDS 10                                      /* Anzahl der Iterationen */
typedef unsigned char byte;                         /* vorzeichenloser
                                                       8-Bit-Integer */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
    int r;                                         /* Schleifenindex */
    byte state[NROWS][NCOLS];                      /* aktueller Zustand */
    struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1]; /* Rundenschlüssel */

    expand_key(key, rk);                           /* erstelle die Runden-
                                                       schlüssel */
    copy_plaintext_to_state(state, plaintext);     /* initialisiere den aktuellen
                                                       Zustand */
    xor_roundkey_into_state(state, rk[0]);          /* Schlüssel und Zustand mit XOR
                                                       verknüpfen */

    for (r = 1; r <= ROUNDS; r++) {
        substitute(state);                         /* wende S-Box auf jedes Byte
                                                       an */
        rotate_rows(state);                       /* rotiere Zeile i um i Bytes */
        if (r < ROUNDS) mix_columns(state);       /* Mischfunktion */
        xor_roundkey_into_state(state, rk[r]);     /* Schlüssel und Zustand mit XOR
                                                       verknüpfen */
    }
    Copy_state_to_ciphertext(ciphertext, state);/* Ergebnis zurückgeben */
}
```

Abbildung 8.9: Kurzfassung von Rijndael.

Die Funktion *rijndael* verfügt über drei Parameter. Diese sind: *plaintext*, ein Array von 16 Byte mit Eingabedaten, *ciphertext*, ein Array von 16 Byte, das die verschlüsselte Ausgabe aufnimmt, und *key*, der 16-Byte-Schlüssel. Bei der Berechnung wird der aktuelle Zustand der Daten in dem Byte-Array *state* gehalten, dessen Größe *NROWS*×*NCOLS* beträgt. Mit 128-Bit-Blöcken ist dieses Array 4×4 Byte groß. Bei 16 Byte kann der ganze 128-Bit-Datenblock gespeichert werden.

Das *state*-Array wird auf den Klartext initialisiert und dann in jedem Schritt der Berechnung modifiziert. In einigen Schritten wird eine Byte-für-Byte-Substitution durchgeführt. In anderen werden die Bytes im Array vertauscht. Es werden auch andere Transformationen verwendet. Am Ende werden die Inhalte von *state* als Chiffretext zurückgegeben.

Der Code beginnt mit der Erweiterung des Schlüssels in elf Arrays, die die gleiche Größe wie *state* haben. Sie werden in *rk*, einem Array von Strukturen, gespeichert, die jeweils ein *state*-Array enthalten. Eines davon wird bei Beginn der Berechnung verwendet, und die anderen zehn werden eines pro Runde in den zehn Runden verwendet. Die Berechnung der Rundenschlüssel aus dem Verschlüsselungsschlüssel ist zu kompliziert, sodass wir nicht weiter darauf eingehen. Es muss genügen zu wissen, dass die Rundenschlüssel durch wiederholte Rotation und XOR-Verknüpfungen verschiedener Schlüsselbits erzeugt werden. Diese Details werden alle in Daemen und Rijmen (2002) beschrieben.

Im nächsten Schritt wird der Klartext in das *state*-Array kopiert, sodass es in den folgenden Runden verarbeitet werden kann. Es wird spaltenweise kopiert, wobei die ersten vier Byte in Spalte 0 gehen, die nächsten vier Byte in Spalte 1 usw. Sowohl die Spalten als auch die Zeilen werden beginnend mit 0 nummeriert, obwohl die Runden mit 1 beginnend nummeriert werden. Das erste Setup der 12-Byte-Arrays der Größe 4×4 ist in ▶ Abbildung 8.10 dargestellt.

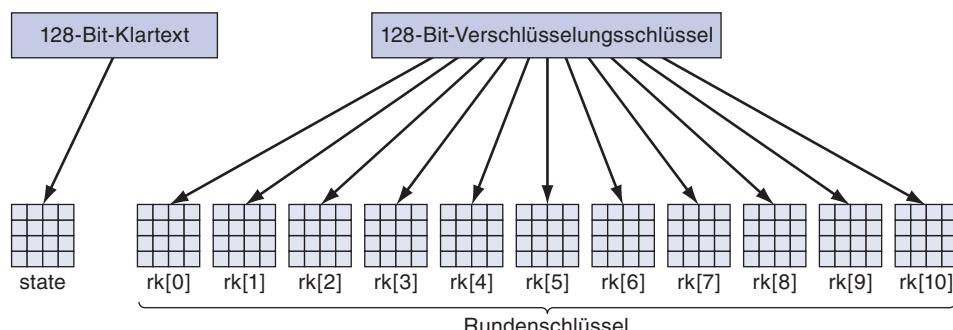


Abbildung 8.10: Erstellen der Arrays *state* und *rk*.

Es gibt noch einen weiteren Schritt, bevor die Hauptberechnung beginnt. *rk[0]* wird mit *state* Byte für Byte mit XOR verknüpft. Anders ausgedrückt, jedes der 16 Byte in *state* wird durch das XOR von sich selbst und dem entsprechenden Byte in *rk[0]* ersetzt.

Nun kommt das Wichtigste. Die Schleife führt zehn Iterationen aus, eine pro Runde, wobei *state* in jeder Runde transformiert wird. Die Inhalte jeder Runde bestehen aus vier Schritten. Im ersten Schritt wird eine Byte-für-Byte-Substitution für *state* durchgeführt. Jedes Byte wird wiederum als ein Index in eine S-Box verwendet, um seinen Wert durch die Inhalte dieses S-Box-Eintrags zu ersetzen. Dieser Schritt ist eine direkte monoalphabetische Substitutionschiffre. Im Unterschied zu DES, das mehrere S-Boxen einsetzt, hat Rijndael nur eine.

Im zweiten Schritt rotiert jede der vier Zeilen nach links. Die Zeile 0 wird um 0 Byte rotiert (also nicht geändert), Zeile 1 wird um 1 Byte rotiert, Zeile 2 um 2 Byte und Zeile 3 um 3 Byte. Dieser Schritt verstreut die Inhalte der aktuellen Daten über den Block, analog zu den Permutationen in Abbildung 8.6.

Im dritten Schritt wird jede Spalte unabhängig von den anderen gemischt. Das Mischen wird über eine Matrizenmultiplikation vorgenommen, bei der die neue Spalte das Produkt der alten Spalte und einer konstanten Matrix ist. Die Multiplikation wird unter Verwendung des endlichen Galois-Körpers $GF(2^8)$ durchgeführt. Obwohl dies kompliziert klingen mag, existiert ein Algorithmus, der es ermöglicht, dass jedes Element der neuen Spalte mit zweimaligem Tabellenlesen und drei XOR-Verknüpfungen berechnet wird (Daemen und Rijmen, 2002, Anhang E).

Schließlich wird im vierten Schritt der Schlüssel dieser Runde im *state*-Array mit XOR verknüpft, zur Verwendung in der nächsten Runde.

Da jeder Schritt reversibel ist, kann die Entschlüsselung durch umgekehrtes Ausführen des Algorithmus erfolgen. Es gibt aber auch einen Trick, bei dem die Entschlüsselung durch Ausführen des Verschlüsselungsalgorithmus mit unterschiedlichen Tabellen vorgenommen werden kann.

Der Algorithmus wurde nicht nur für höchste Sicherheit, sondern auch für höchste Geschwindigkeit entwickelt. Eine gute Software-Implementierung auf einem 2-GHz-Rechner sollte hier eine Verschlüsselungsrate von 700 Mbit/s erzielen. Dies ist schnell genug, um über 100 MPEG-2-Videos in Echtzeit zu verschlüsseln. Die Hardware-Implementierungen sind noch schneller.

8.2.3 Chiffriermodi

Trotz aller Komplexität ist AES (oder DES bzw. in diesem Zusammenhang jede Blockchiffre) im Grunde eine monoalphabetische Substitutionschiffre mit großen Zeichen (128-Bit-Zeichen für AES und 64-Bit-Zeichen für DES). Gibt man den gleichen Klartextblock erneut ein, kommt der gleiche Chiffretextblock am Ende heraus. Wenn man den Klartext *abcdefgh* 100-mal mit dem gleichen DES-Schlüssel verschlüsselt, erhält man 100-mal den gleichen Chiffretext. Ein Eindringling kann diese Eigenschaft nutzen, um die Chiffre aufzubrechen.

Electronic-Code-Book-Modus

Um zu zeigen, wie diese monoalphabetische Substitutionschiffre zur teilweisen Entschlüsselung von Chiffren genutzt werden kann, verwenden wir (Triple) DES, da 64-Bit-Blöcke einfacher dargestellt werden können als 128-Bit-Blöcke. Aber AES hat genau das gleiche Problem. Der einfachste Einsatz von DES zur Verschlüsselung von langen Passagen Klartext ist, diesen in aufeinanderfolgende 8-Byte-Blöcke (64 Bit) aufzuteilen und sie nacheinander mit dem gleichen Schlüssel zu verschlüsseln. Der letzte Block wird bei Bedarf auf 64 Bit aufgefüllt. Diese Technik wird als **ECB-Modus** (*Electronic Code Book*) bezeichnet, in Analogie zu den altmodischen Codebüchern, in

denen jedes Klartextwort gefolgt von seinem Chiffretext (in der Regel eine fünfstellige Dezimalzahl) aufgelistet war.

In ► Abbildung 8.11 haben wir den Anfang einer Computerdatei, die Jahresboni auflistet, die eine Firma ihren Mitarbeitern auszahlen will. Diese Datei besteht aus aufeinanderfolgenden 32-Byte-Datensätzen, einem pro Mitarbeiter, in dem gezeigten Format: 16 Byte für den Namen, 8 Byte für den Posten und 8 Byte für den Bonus. Jeder der sechzehn 8-Byte-Blöcke (von 0 bis 15 durchnummiert) wird durch (Triple) DES verschlüsselt.

| Name | Posten | Bonus |
|-------------------------------|-------------------|--------------------------------|
| A d a m s , L | e s l i e | C l e r k |
| B I a c k , R | o b i n | B o s s |
| C o l l i n s , | K i m | M a n a g e r |
| D a v i s , B | o b b i e | J a n i t o r |
| | | \$ 1 0 |
| | | \$ 5 0 0 , 0 0 0 |
| | | \$ 1 0 0 , 0 0 0 |
| | | \$ 5 |

Byte ←————— 16 —————→ 8 ←————— 8 —————→ 8 ←—————

Abbildung 8.11: Der Klartext einer Datei, die als 16 DES-Blöcke verschlüsselt wird.

Leslie hatte gerade Streit mit dem Chef und erwartet in Bezug auf den Bonus nicht viel. Kim wird vom Chef bevorzugt, und jeder weiß das. Leslie kann sich nach der Verschlüsselung Zugriff auf die Datei verschaffen, bevor diese an die Bank gesendet wird. Kann sich Leslie gegen diese unfaire Behandlung wehren, auch wenn ihr nur die verschlüsselte Datei zur Verfügung steht?

Gar kein Problem! Leslie braucht nur eine Kopie von Chiffretextblock 12 (der den Bonus für Kim enthält) zu kopieren und ihn mit Chiffretextblock 4 (ihrem eigenen Bonus) auszutauschen. Auch ohne zu wissen, was in Block 12 steht, kann Leslie davon ausgehen, dass ihr Weihnachtsfest in diesem Jahr üppiger ausfallen wird. (Kopieren des achten Chiffretextblocks ist auch eine Möglichkeit, wird aber mit größerer Wahrscheinlichkeit aufgedeckt. Außerdem will Leslie ja nicht übertreiben.)

Cipher-Block-Chaining-Modus

Um diese Art von Attacken abzuwehren, können alle Blockchiffren auf unterschiedliche Weise verkettet werden. So führt das Ersetzen eines Blocks, wie Leslie dies getan hat, dazu, dass der auf einen ersetzen Block folgende verschlüsselte Klartext unverständlich wird. Eine Möglichkeit der Verkettung ist das **Cipher Block Chaining**. Bei dieser Methode (► Abbildung 8.12) wird jeder Klartextblock vor der Verschlüsselung mit XOR mit dem vorherigen Chiffretextblock verknüpft. Folglich wird der gleiche Klartextblock nicht mehr auf den gleichen Chiffretextblock abgebildet und die Verschlüsselung ist keine große monoalphabetische Substitutionschiffre mehr. Der erste Block wird mit XOR mit einem zufällig gewählten **Initialisierungsvektor** (**IV**, *Initialization Vector*) verknüpft, der (im Klartext) mit dem Chiffretext übertragen wird.

Das Beispiel in Abbildung 8.12 zeigt auf, wie das Cipher Block Chaining funktioniert. Wir beginnen mit der Berechnung $C_0 = E(P_0 \text{ XOR } IV)$. Dann berechnen wir $C_1 = E(P_1 \text{ XOR } C_0)$ usw. Die Entschlüsselung verwendet ebenso XOR, um den Prozess umzukehren, wobei $P_0 = IV \text{ XOR } D(C_0)$ ist usw. Die Verschlüsselung von Block i ist also eine Funktion des gesamten Klartextes in den Blöcken 0 bis $i - 1$. Deshalb erzeugt der gleiche Klartext je nachdem, wo er vorkommt, einen anderen Chiffretext. Eine Umwandlung der Art, wie Leslie sie durchführte, ergibt in den zwei von ihr manipulierten Bonusfeldern nur Unsinn. Ein scharfsinniger Sicherheitsmanager eines Unternehmens kann an dieser Besonderheit wahrscheinlich erkennen, wo man mit der Untersuchung beginnen muss.

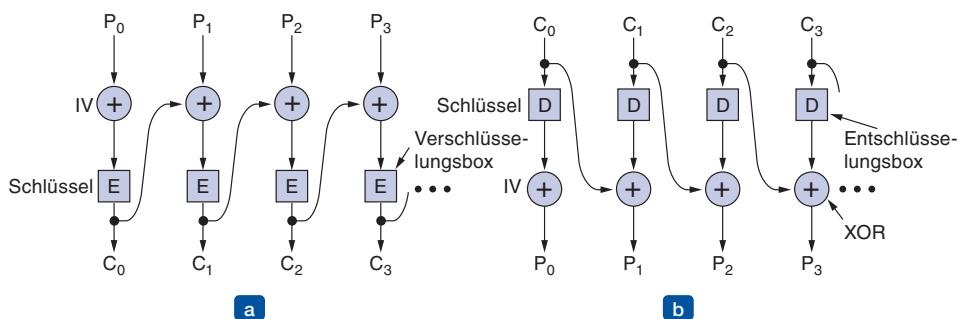


Abbildung 8.12: Cipher Block Chaining: (a) Verschlüsselung, (b) Entschlüsselung.

Cipher Block Chaining hat auch den Vorteil, dass der gleiche Klartextblock nicht zum gleichen Chiffretextblock führt, was die Kryptoanalyse erschwert. Das ist überhaupt der wichtigste Grund für die Anwendung dieser Technik.

Cipher-Feedback-Modus

Cipher Block Chaining hat aber auch einen Nachteil. Bevor die Verschlüsselung beginnen kann, muss ein ganzer 64-Bit-Block ankommen. In Umgebungen mit interaktiven Terminals, wo die Leute Zeilen mit weniger als 8 Zeichen eintippen und dann stoppen und auf eine Antwort warten können, ist diese Technik ungeeignet. Für eine byteweise Verschlüsselung eignet sich der sogenannte **Cipher-Feedback-Modus** unter Verwendung von (Triple) DES (►Abbildung 8.13) besser. Bei AES ist das Konzept das gleiche, nur wird ein 128-Bit-Schieberegister verwendet. In dieser Abbildung wird der Zustand des Verschlüsselungsrechners dargestellt, nachdem die Bytes 0 bis 9 verschlüsselt und übertragen wurden. Kommt Klartextbyte 10 an (siehe ►Abbildung 8.13a), arbeitet der DES-Algorithmus mit dem Inhalt des 64-Bit-Schieberegisters, um einen 64-Bit-Chiffretext zu erzeugen. Das linke Byte dieses Chiffretextes wird mit P_{10} XOR-verknüpft. Dieses Byte wird über die Übertragungsleitung gesendet. Darüber hinaus wird das Schieberegister um 8 Bit nach links verschoben, sodass C_2 am linken Ende herausfällt und C_{10} an der gerade am rechten Ende von C_9 frei gewordenen Stelle eingefügt wird.

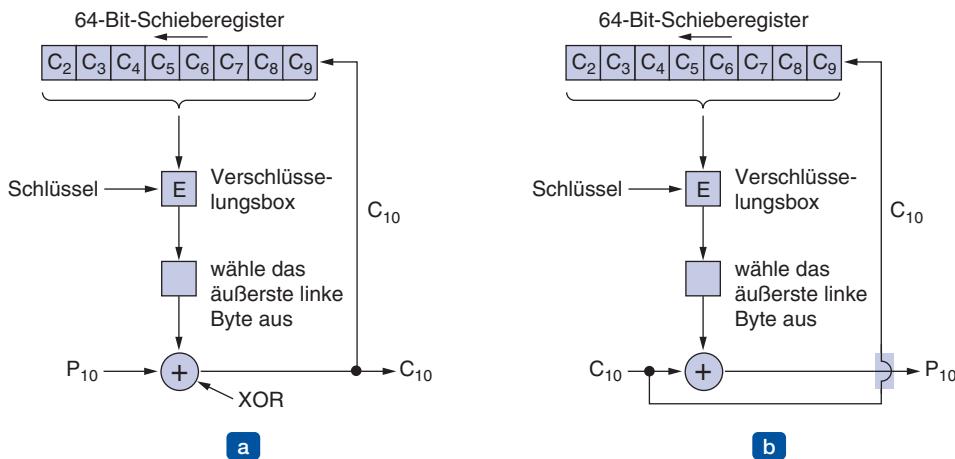


Abbildung 8.13: Cipher-Feedback-Modus: (a) Verschlüsselung, (b) Entschlüsselung.

Der Inhalt des Schieberegisters hängt von der gesamten Vorgeschiede des Klartextes ab. Deshalb wird ein Muster, das sich mehrmals im Klartext wiederholt, im Chiffretext jedes Mal anders verschlüsselt. Wie beim Cipher Block Chaining ist ein Initialisierungsvektor erforderlich, um den Ball ins Rollen zu bringen.

Die Entschlüsselung beim Cipher-Feedback-Modus erfolgt auf die gleiche Weise wie die Verschlüsselung. Insbesondere ist der Inhalt des Schieberegisters *verschlüsselt* und nicht *entschlüsselt*, sodass das ausgewählte Byte, das mit C_{10} XOR-verknüpft wird, um P_{10} zu erhalten, dasselbe ist, das ursprünglich mit P_{10} XOR-verknüpft wurde, um C_{10} zu erzeugen. Solange die zwei Schieberegister identisch sind, funktioniert das Entschlüsseln richtig. Es ist in ► Abbildung 8.13b dargestellt.

Ein Problem beim Cipher-Feedback-Modus ist, dass bei der Übertragung versehentlich invertiertes Bit des Chiffretextes alle 8 zu entschlüsselnden Byte verstümmelt, die sich zusammen mit dem fehlerhaften Byte im Schieberegister befinden. Wird das fehlerhafte Byte aus dem Schieberegister geschoben, wird wieder der richtige Klartext erzeugt. Somit bleiben die Auswirkungen eines einzelnen invertierten Bit mehr oder weniger lokal und ruinieren nicht den Rest der Nachricht. Sie beschädigen aber so viele Bits, wie das Schieberegister breit ist.

Stream-Cipher-Modus

Aber es gibt natürlich Anwendungen, bei denen es nicht akzeptabel ist, dass ein 1-Bit-Fehler bei der Übertragung 64 Bit des Klartextes unbrauchbar macht. Für diese Anwendungen gibt es eine vierte Option, den **Stream-Cipher-Modus**. Hier wird ein Initialisierungsvektor mit einem Schlüssel verschlüsselt, um einen Ausgabeblock zu erhalten. Der Ausgabeblock wird dann mit dem Schlüssel verschlüsselt, um einen zweiten Ausgabeblock zu erhalten. Dieser Block wird dann verschlüsselt, um einen dritten Block zu erhalten usw. Die (beliebig lange) Folge von Ausgabeblöcken, die als **Schlüsselstrom** (*keystream*) bezeichnet wird, wird wie ein One-Time Pad behandelt und mit dem Klar-

text XOR-verknüpft, wie in ▶ Abbildung 8.14a gezeigt. Beachten Sie, dass der Initialisierungsvektor (IV) nur im ersten Schritt verwendet wird. Danach wird die Ausgabe verschlüsselt. Beachten Sie auch, dass der Schlüsselstrom unabhängig von den Daten ist, sodass er bei Bedarf im Voraus berechnet werden kann. Er ist völlig unempfindlich gegenüber Übertragungsfehlern. Die Entschlüsselung ist in ▶ Abbildung 8.14b dargestellt.

Bei der Entschlüsselung wird der gleiche Schlüsselstrom auf der Empfängerseite generiert. Da der Schlüsselstrom nur vom IV und dem Schlüssel abhängt, wird er von Übertragungsfehlern im Chiffretext nicht berührt. Daher erzeugt ein 1-Bit-Fehler im übertragenen Chiffretext nur einen 1-Bit-Fehler im entschlüsselten Klartext.

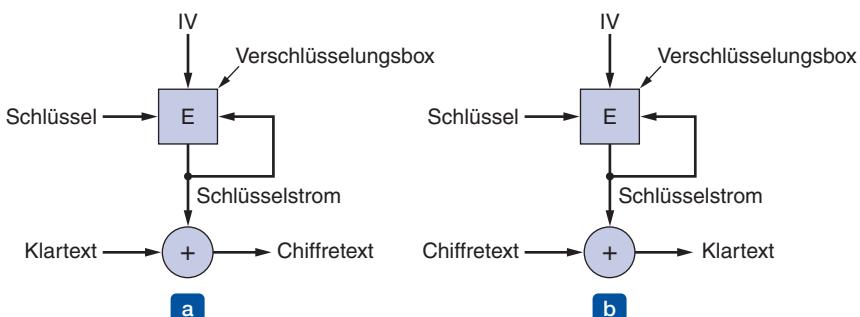


Abbildung 8.14: Stromchiffre: (a) Verschlüsselung, (b) Entschlüsselung.

Es ist wichtig, dass bei einer Stromchiffre nie das gleiche Schlüssel/IV-Paar zweimal verwendet wird, da es jedes Mal den gleichen Schlüsselstrom erzeugt. Wenn der gleiche Schlüsselstrom zweimal verwendet wird, ist der Chiffretext einem **Schlüsselstrom-Reuse-Angriff** ausgesetzt. Stellen Sie sich vor, dass ein Klartextblock P_0 mit dem Schlüsselstrom verschlüsselt wird, um $P_0 \oplus K_0$ zu erhalten. Später wird ein zweiter Klartextblock Q_0 mit dem gleichen Schlüsselstrom verschlüsselt, um $Q_0 \oplus K_0$ zu erhalten. Ein Eindringling, der diese beiden Chiffretextblöcke abfängt, kann einfach beide mit XOR verknüpfen, um $P_0 \oplus Q_0$ zu erhalten, wodurch der Schlüssel entfernt wird. Der Eindringling hat nun das XOR der beiden Klartextblöcke. Ist einer davon bekannt oder kann erraten werden, so kann auch der andere ermittelt werden. In jedem Fall kann das XOR der beiden Klartextströme durch das Heranziehen der statistischen Eigenschaften der Nachricht angegriffen werden. Bei englischem Text beispielsweise ist das häufigste Zeichen im Strom höchstwahrscheinlich das XOR von zwei Leerzeichen, gefolgt von dem XOR eines Leerzeichens und dem Buchstaben „e“ usw. Kurz gefasst: Wenn das XOR von zwei Klartexten vorliegt, hat der Kryptoanalytiker gute Chancen, beide zu entziffern.

Counter-Modus

Ein Problem, das bei allen Modi außer dem Electronic-Code-Book-Modus auftritt, ist, dass der Zufallszugriff auf verschlüsselte Daten unmöglich ist. Angenommen, eine Datei wird über ein Netz übertragen und dann verschlüsselt auf Platte gespeichert. Dies kann eine vernünftige Arbeitsweise sein, wenn der Empfänger ein Notebook ist,

das gestohlen werden kann. Wenn alle wichtigen Dateien verschlüsselt gespeichert werden, reduziert dies den Schaden erheblich, wenn der Computer und damit die Information in die falschen Hände geraten.

Auf Festplattendateien wird aber häufig in nicht sequenzieller Folge zugegriffen, besonders auf Dateien in Datenbanken. Wird eine Datei mithilfe von Cipher Block Chaining verschlüsselt, dann müssen bei einem Zugriff auf einen Zufallsblock zuerst alle Blöcke entschlüsselt werden – eine aufwendige Voraussetzung. Aus diesem Grund wurde ein anderer Modus, der sogenannte **Counter-Modus**, entwickelt, der in ►Abbildung 8.15 dargestellt wird. Hier wird der Klartext nicht direkt verschlüsselt. Stattdessen werden der Initialisierungsvektor und eine Konstante verschlüsselt und der resultierende Chiffrettext wird mit dem Klartext über XOR verknüpft. Indem der Initialisierungsvektor für jeden neuen Block um 1 erhöht wird, kann ein beliebiger Block in der Datei einfach entschlüsselt werden, ohne dass zuerst alle Vorgänger entschlüsselt werden müssen.

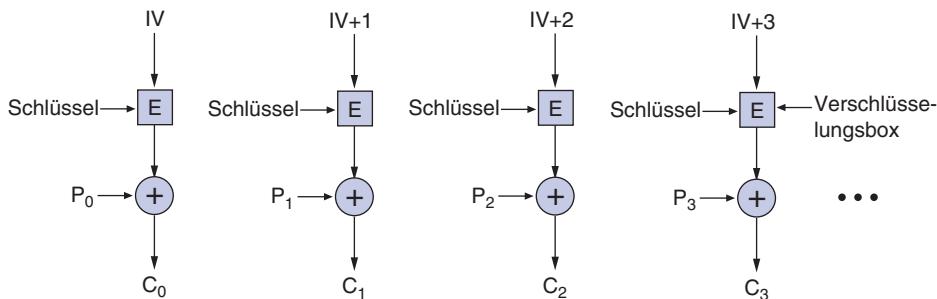


Abbildung 8.15: Verschlüsselung mit dem Counter-Modus.

Obwohl der Counter-Modus nützlich ist, weist er auch eine Schwachstelle auf, die Sie kennen sollten. Angenommen, der gleiche Schlüssel K wird in Zukunft noch einmal verwendet (mit einem anderen Klartext, aber dem gleichen IV) und ein Angreifer erhält den gesamten Chiffrettext beider Durchläufe. Die Schlüsselströme sind in beiden Fällen gleich, sodass die Chiffre einem Schlüsselstrom-Reuse-Angriff ausgesetzt sind, wie wir bei der Stromverschlüsselung sahen. Der Kryptoanalytiker muss nur die beiden Chiffretexte mit XOR verknüpfen, um den kryptografischen Schutz zu entfernen und das XOR der Klartexte zu erhalten. Diese Schwachstelle bedeutet nicht, dass der Counter-Modus ein schlechtes Konzept ist. Sie sollten nur darauf achten, dass Schlüssel und Initialisierungsvektoren unabhängig voneinander und zufällig gewählt werden. Selbst wenn der gleiche Schlüssel zufällig zweimal verwendet wird, ist der Klartext sicher, wenn die IV verschieden sind.

8.2.4 Weitere Chiffren

DES und (AES) Rijndael sind die bekanntesten symmetrischen kryptografischen Algorithmen und die Standardwahl der Hersteller, und sei es auch nur aus Gründen der Haftung. (Niemand wird Sie zur Rechenschaft ziehen, wenn Sie AES in Ihrem Produkt

verwenden und AES geknackt wird. Anders sieht es hingegen aus, wenn Sie eine nicht standardisierte Chiffre verwenden, die später gebrochen wird.) Dennoch sollte erwähnt werden, dass es eine Vielzahl von anderen symmetrischen Verschlüsselungen gibt. Einige davon sind in verschiedene Produkte eingebettet. Einige der häufiger verwendeten Chiffren sind in ►Abbildung 8.16 aufgeführt. Es ist sogar möglich, Chiffren zu kombinieren, z.B. AES mit Twofish, sodass beide Chiffren geknackt werden müssen, um an die Daten zu gelangen.

| Chiffre | Autor | Schlüssellänge | Kommentar |
|----------------|--------------------------|----------------|---|
| DES | IBM | 56 Bit | Heutzutage zu schwach |
| RC4 | Ronald Rivest | 1–2 048 Bit | Vorsicht: Einige Schlüssel sind schwach |
| RC5 | Ronald Rivest | 128–256 Bit | Gut, aber patentiert |
| AES (Rijndael) | Daemen und Rijmen | 128–256 Bit | Beste Lösung |
| Serpent | Andersen, Biham, Knudsen | 128–256 Bit | Sehr stark |
| Triple DES | IBM | 168 Bit | Gut, aber allmählich veraltet |
| Twofish | Bruce Schneier | 128–256 Bit | Sehr stark, weitverbreitet |

Abbildung 8.16: Einige häufig verwendete symmetrische kryptografische Algorithmen.

8.2.5 Kryptoanalyse

Bevor wir uns von dem Thema der symmetrischen Kryptografie abwenden, lohnt sich ein Blick auf vier Entwicklungen im Bereich der Kryptoanalyse. Die erste ist die **differenzielle Kryptoanalyse** (*differential cryptanalysis*; Biham und Shamir, 1997). Mit dieser Technik lässt sich jede Blockchiffre attackieren. Ausgangsbasis sind zwei Klartextblöcke, die sich nur in ganz wenigen Bits unterscheiden. Dann wird aufmerksam beobachtet, was bei jeder internen Iteration im Verlauf der Verschlüsselung passiert. Meist treten einige Muster häufiger auf als andere, was zu Wahrscheinlichkeitsattacken führen kann.

Die zweite erwähnenswerte Entwicklung ist die **lineare Kryptoanalyse** (*linear cryptanalysis*; Matsui, 1994). Sie kann DES mit nur 2^{43} bekannten Klartexten aufbrechen. Bei dieser Methode werden bestimmte Bits im Klartext und dem Chiffertext mit XOR verknüpft und die Ergebnisse untersucht. Wird das wiederholt ausgeführt, sollte die eine Hälfte der Bits Nullen und die andere Einsen sein. Meist weisen Chiffren aber in der einen oder der anderen Richtung eine Verschiebung auf. Diese Verschiebung, auch wenn sie noch so klein ist, lässt sich nutzen, um den Arbeitsaufwand zu reduzieren. Ausführliche Informationen sind in dem Aufsatz von Matsui zu finden.

Die dritte Entwicklung basiert auf der Analyse des elektrischen Stromverbrauchs, um geheime Schlüssel zu finden. Die Computer stellen ein 1-Bit in der Regel mit 3 Volt dar sowie ein 0-Bit mit 0 Volt. Demzufolge verbraucht die Verarbeitung einer 1 mehr

elektrische Energie als die Verarbeitung einer 0. Wenn ein kryptografischer Algorithmus aus einer Schleife besteht, in der die Schlüsselbits der Reihe nach verarbeitet werden, kann ein Angreifer, der den n -GHz-Haupttakt durch einen niedrigen (z.B. 100-Hz-)Takt ersetzt und Abgreifklemmen auf den Strom- und Massepins der CPU anbringt, den von jeder Rechneranweisung benötigten Stromverbrauch genau messen. Aus diesen Daten ist die Ableitung eines Schlüssels erstaunlich einfach. Diese Art der Kryptoanalyse kann nur vereitelt werden, indem der Algorithmus sorgfältig in Assembler codiert wird, um den Stromverbrauch unabhängig vom Schlüssel und auch unabhängig von allen einzelnen Rundenschlüsseln zu machen.

Die vierte Entwicklung ist die Analyse des Zeitverhaltens. Kryptografische Algorithmen sind voller `if`-Anweisungen, die Bits in den Rundenschlüsseln testen. Wenn die Verarbeitung der `then`- und `else`-Teile unterschiedlich lange Zeit im Anspruch nimmt, ist es auch möglich, durch Verringern des Takts und Prüfen der Dauer der verschiedenen Schritte die Rundenschlüssel abzuleiten. Wenn einmal alle Rundenschlüssel bekannt sind, kann damit in der Regel der ursprüngliche Schlüssel berechnet werden. Der Stromverbrauch und das Zeitverhalten können auch gleichzeitig analysiert werden, um die Aufgabe zu vereinfachen. Wenngleich die Analyse des Stromverbrauchs und des Zeitverhaltens exotisch scheinen mag, so sind dies in der Praxis leistungsstarke Techniken, die jede Chiffre aufbrechen können, die nicht speziell dafür ausgelegt wurde, diesen Techniken standzuhalten.

8.3 Algorithmen für öffentliche Schlüssel

Historisch ist die Verteilung der Schlüssel bei den meisten Kryptosystemen seit jeher der schwächste Punkt. Wenn ein Eindringling den Schlüssel stehlen konnte, war das System wertlos, gleichgültig, wie gut es war. Dabei sind Kryptologen immer davon ausgegangen, dass zum Verschlüsseln und Entschlüsseln dieselben Schlüssel verwendet werden (oder einfach voneinander abgeleitet werden können). Jedoch musste der Schlüssel an alle Benutzer des Systems verteilt werden. Daher schien es eher ein inhärentes Problem zu sein. Schlüssel müssen einerseits vor Diebstahl geschützt, andererseits aber auch verteilt werden. Man kann sie nicht in einem Banksafe wegschließen.

1976 schlugen die beiden Wissenschaftler Diffie und Hellman von der Stanford-Universität ein völlig anderes Kryptosystem vor, eines, bei dem sich die Schlüssel für die Ver- und die Entschlüsselung unterschieden (1976). Der zweite Schlüssel konnte nicht vom ersten abgeleitet werden. In ihrem Vorschlag musste der (mit Schlüssel versehene) Verschlüsselungsalgorithmus E und der (ebenfalls mit Schlüssel versehene) Entschlüsselungsalgorithmus D folgende drei Anforderungen erfüllen:

- 1.** $D(E(P)) = P$.
- 2.** Es ist extrem schwierig, D aus E abzuleiten.
- 3.** E kann nicht durch eine gewählte Klartextattacke gebrochen werden.

Die erste Anforderung besagt: Wenn wir D auf eine verschlüsselte Nachricht $E(P)$ anwenden, erhalten wir die Originalnachricht im Klartext P . Ohne diese Eigenschaft könnte der legitime Empfänger den Chiffretext nicht entschlüsseln. Die zweite Anforderung spricht für sich selbst. Die dritte Anforderung ist nötig, weil Eindringlinge eventuell nach Herzenslust mit dem Algorithmus experimentieren können, wie wir gleich sehen werden. Unter diesen Umständen besteht kein Grund, dass der Schlüssel zum Verschlüsseln nicht öffentlich sein soll.

Die Methode funktioniert wie folgt: Eine Person, nennen wir sie Alice, möchte geheime Nachrichten empfangen. Sie entwirft zuerst zwei Algorithmen, die die obigen Anforderungen erfüllen. Der Verschlüsselungsalgorithmus und der Schlüssel von Alice werden öffentlich bekannt gemacht, daher der Name **Kryptografie mit öffentlichem Schlüssel** (*public key cryptography*). Alice könnte beispielsweise ihren öffentlichen Schlüssel auf ihrer Homepage im Web angeben. Wir verwenden die Notation E_A für den Verschlüsselungsalgorithmus, der von Alices öffentlichem Schlüssel parametrisiert wird. Entsprechend lautet der (geheime) Entschlüsselungsalgorithmus, der von Alices privatem Schlüssel parametrisiert wird, D_A . Bob macht das Gleiche; er gibt E_B bekannt, hält aber D_B geheim.

Lassen Sie uns sehen, ob wir das Problem lösen können, einen sicheren Kanal zwischen Alice und Bob aufzubauen, die vorher nie miteinander Kontakt hatten. Der Schlüssel für die Verschlüsselung von Alice E_A und derjenige von Bob E_B befinden sich in einer öffentlich zugänglichen Datei. Nun nimmt Alice ihre erste Nachricht P , berechnet $E_B(P)$ und sendet sie an Bob. Bob entschlüsselt sie durch Anwenden seines Geheimschlüssels D_B (d.h., er berechnet $D_B(E_B(P))=P$). Niemand anders kann die verschlüsselte Nachricht $E_B(P)$ lesen, weil das Verschlüsselungssystem als stark gilt und es zu schwierig ist, D_B vom öffentlich bekannten E_B abzuleiten. Zum Senden der Antwort, R , überträgt Bob $E_A(R)$. Alice und Bob können nun sicher kommunizieren.

An dieser Stelle wollen wir ein paar Worte zur Terminologie loswerden. Die Kryptografie mit öffentlichen Schlüsseln setzt voraus, dass jeder Benutzer zwei Schlüssel hat: einen öffentlichen Schlüssel, der von allen zum Verschlüsseln von Nachrichten an den Benutzer verwendet wird, und einen privaten Schlüssel, den der Benutzer zum Entschlüsseln benötigt. Wir nennen sie in diesem Buch durchgängig *öffentlich* (*public*) bzw. *privat* (*private*) und unterscheiden sie von den *geheimen* Schlüsseln, die in konventionellen Umgebungen mit symmetrischen Schlüsseln verwendet werden.

8.3.1 RSA

Der einzige Haken ist, dass wir Algorithmen finden müssen, die die drei obigen Anforderungen erfüllen. Aufgrund der erheblichen Vorteile der Kryptografie mit öffentlichen Schlüsseln sind viele Wissenschaftler eifrig bei der Arbeit. Einige Algorithmen wurden bereits veröffentlicht. Eine gute Methode wurde von einer Arbeitsgruppe am MIT entdeckt (Rivest et al., 1978) und ist nach den Anfangsbuchstaben der drei Entwickler (Rivest, Shamir, Adleman) mit **RSA** benannt. Dieser Algorithmus hat über 30 Jahre alle Versuche, ihn zu knacken, überstanden und gilt als sehr stark. Sehr viel

praktische Sicherheit basiert darauf. Dafür wurde Rivest, Shamir und Adleman 2002 sogar der ACM Turing Award verliehen. Der größte Nachteil ist, dass der Algorithmus einen Schlüssel von mindestens 1 024 Bit benötigt, um eine gute Sicherheit zu gewährleisten (gegenüber 128 Bit für Algorithmen mit symmetrischen Schlüsseln), was ihn ziemlich langsam macht.

Die RSA-Methode basiert auf einigen Prinzipien der Zahlentheorie. Nachfolgend eine Übersicht. Für ausführliche Informationen empfiehlt sich die Lektüre des Dokuments der Arbeitsgruppe.

- 1.** Wähle zwei Primzahlen p und q (in der Regel 1 024 Bit).
- 2.** Berechne $n=p \times q$ und $\varphi=(p-1) \times (q-1)$.
- 3.** Wähle eine Zahl, die teilerfremd zu φ ist, und nenne sie d .
- 4.** Finde e so, dass $e \times d = 1 \pmod{\varphi}$.

Sind diese Parameter im Voraus berechnet, können wir mit der Verschlüsselung beginnen. Wir teilen den Klartext (der als Bitkette gilt) in Blöcke auf, sodass jede Klartextnachricht P im Intervall $0 \leq P < n$ liegt. Dies erreichen wir, indem wir den Klartext in Blöcken zu je k Bit anordnen, wobei k die größte ganze Zahl ist, auf die die Bedingung $2^k < n$ zutrifft.

Um eine Nachricht P zu verschlüsseln, berechnen wir $C=P^e \pmod{n}$. Um C zu entschlüsseln, berechnen wir $P=C^d \pmod{n}$. Man kann beweisen, dass für alle P im angegebenen Intervall die Ver- und Entschlüsselungsfunktionen zueinander invers sind. Um die Nachricht zu verschlüsseln, benötigen wir die Werte e und n . Für die Entschlüsselung der Nachricht benötigen wir d und n . Der öffentliche Schlüssel besteht daher aus dem Wertepaar (e,n) , während der private Schlüssel aus dem Paar (d,n) besteht.

Die Sicherheit der Methode hängt mit dem Problem zusammen, große Zahlen in Faktoren zu zerlegen. Kann der Kryptoanalytiker die (öffentlich bekannte) Zahl n in Faktoren zerlegen, so könnte er p und q und damit auch φ herausfinden. Mit φ und e und mithilfe des Algorithmus von Euklid könnte er d errechnen. Zum Glück ist es den Mathematikern in den letzten 300 Jahren nicht gelungen, große Zahlen in Faktoren zu zerlegen. Alle in diesem Zusammenhang bekannten Fakten weisen darauf hin, dass dies ein ausnehmend schwieriges Problem ist.

Nach Rivest und Kollegen erfordert das Zerlegen einer 500-stelligen Zahl 10^{25} Jahre mit Brachialgewalt (ein sogenannter Brute-Force-Angriff). Dabei wird in beiden Fällen vom besten bekannten Algorithmus und einem Rechner mit einer Befehlausführungszeit von $1\mu\text{s}$ ausgegangen. Mit einer Million parallel ausgeführter Chips, die jeweils eine Befehlausführungszeit von 1 ns aufweisen, würde es noch 10^{16} Jahre dauern. Auch wenn Computer weiterhin pro Jahrzehnt um eine Größenordnung schneller werden, wird es viele Jahre dauern, bis die Faktorisierung einer 500-stelligen Zahl durchführbar ist. Und wenn es dann soweit ist, können unsere Nachfahren p und q einfach größer wählen.

► Abbildung 8.17 zeigt ein einfaches Lehrbeispiel des RSA-Algorithmus. Für dieses Beispiel haben wir $p=3$ und $q=11$ gewählt, sodass $n=33$ und $z=20$ ist. Ein passender Wert für d ist $d=7$, da 7 und 20 keine gemeinsamen Faktoren haben. Mit diesen Entscheidungen kann e durch Lösung der Gleichung $7e \equiv 1 \pmod{20}$ gefunden werden, was $e=3$ ergibt. Der Chiffrentext C für eine Klartextnachricht P wird dargestellt durch $C=P^3 \pmod{33}$. Der Chiffrentext wird vom Empfänger mit der Formel $P=C^7 \pmod{33}$ entschlüsselt. Die Abbildung zeigt als Beispiel die Verschlüsselung des Klartextes „SUZANNE“.

| Klartext (P) | | Chiffrentext (C) | | Nach der Entschlüsselung | |
|--------------|-----------|------------------|-----------------|--------------------------|-----------------|
| Symbolisch | Numerisch | P^3 | $P^3 \pmod{33}$ | C^7 | $C^7 \pmod{33}$ |
| S | 19 | 6859 | 28 | 13492928512 | 19 |
| U | 21 | 9261 | 21 | 1801088541 | 21 |
| Z | 26 | 17576 | 20 | 1280000000 | 26 |
| A | 01 | 1 | 1 | 1 | 01 |
| N | 14 | 2744 | 5 | 78125 | 14 |
| N | 14 | 2744 | 5 | 78125 | 14 |
| E | 05 | 125 | 26 | 8031810176 | 05 |

Berechnung des Senders
Berechnung des Empfängers

Abbildung 8.17: Beispiel des RSA-Algorithmus.

Da die in diesem Beispiel gewählten Primzahlen sehr klein sind, muss P kleiner sein als 33. Somit kann jeder Klartextblock nur ein Zeichen enthalten. Das Ergebnis ist eine nicht sehr beeindruckende monoalphabetische Substitutionschiffre. Hätten wir stattdessen p und $q \approx 2^{512}$ gewählt, wäre $n \approx 2^{1024}$. Damit könnte jeder Block bis zu 1 024 Bit oder 128 8-Bit-Zeichen enthalten, im Gegensatz zu acht Zeichen bei DES und 16 Zeichen bei AES.

RSA ist in dem hier aufgeführten Beispiel mit einem symmetrischen Algorithmus im ECB-Modus vergleichbar: Der gleiche Eingabeblock ergibt den gleichen Ausgabeblock. Deshalb ist eine Form der Verkettung für die Datenverschlüsselung erforderlich. In der Praxis verwenden die meisten RSA-Systeme die Kryptografie mit öffentlichen Schlüsseln, um einmalig zu verwendende Schlüssel für Sitzungen zu verteilen, in denen dann mit symmetrischen Algorithmen wie AES oder Triple DES gearbeitet wird. RSA ist zu langsam für die tatsächliche Verschlüsselung großer Datenmengen, wird aber sehr oft zur Verteilung von Schlüsseln eingesetzt.

8.3.2 Weitere Algorithmen für öffentliche Schlüssel

RSA wird zwar häufig verwendet, ist aber keinesfalls der einzige bekannte Algorithmus mit öffentlichen Schlüsseln. Der erste Algorithmus mit öffentlichen Schlüsseln war der *Rucksack-Algorithmus* (*knapsack algorithm*; Merkle und Hellman, 1978). Er basiert auf dem Konzept, dass jemand eine große Menge von Objekten mit je einer anderen Gewichtung besitzt. Der Inhaber codiert die Nachricht durch Auswahl einer geheimen Teilmenge der Objekte, dann packt er sie in den Rucksack. Die Gesamt-

gewichtung der Objekte im Rucksack wird veröffentlicht, ebenso wie die Liste aller möglichen Objekte und deren Gewichtung. Die Objektliste im Rucksack bleibt geheim. Mit bestimmten zusätzlichen Einschränkungen galt das Herausfinden der möglichen Objektliste mit der gegebenen Gewichtung als rechnerisch unmöglich und bildete die Grundlage für den Algorithmus mit öffentlichen Schlüsseln.

Ralph Merkle, der Erfinder des Algorithmus, war überzeugt, dass sein Algorithmus absolut sicher sei. Er bot jedem, dem es gelänge, den Algorithmus zu knacken, eine Belohnung von 100 US-Dollar. Adi Shamir (das „S“ in RSA) gelang dies und er holte sich seine Belohnung. Unbeirrt machte Merkle an der Verbesserung seines Algorithmus weiter und erhöhte seine Belohnung für das Knacken der neuen Version auf 1 000 US-Dollar. Ron Rivest (das „R“ in RSA) knackte prompt die neue Fassung und verdiente sich so die Belohnung. Merkle hielt sich danach bei der folgenden Version mit einer weiteren Belohnung von 10 000 US-Dollar zurück, sodass Leonard Adleman (das „A“ in RSA) leer ausging. Insgesamt wird der Rucksack-Algorithmus nicht als sicher betrachtet und in der Praxis nicht mehr eingesetzt.

Weitere Algorithmen mit öffentlichen Schlüsseln basieren auf der Aufgabe, diskrete Logarithmen zu berechnen. Algorithmen, die auf diesem Prinzip gründen, wurden von El Gamal (1985) und Schnorr (1991) erfunden.

Daneben gibt es noch weitere Methoden, z.B. jene, die auf elliptischen Kurven basieren (Menezes und Vanstone, 1993). Die beiden wichtigsten Kategorien sind jedoch die, die auf der Schwierigkeit der Faktorisierung großer Zahlen bzw. der Berechnung diskreter Logarithmen modulo einer großen Primzahl beruhen. Diese Aufgaben gelten naturgemäß als schwer zu lösen – viele Mathematiker haben sich daran seit vielen Jahren die Zähne ausgebissen, ohne einen größeren Durchbruch zu erzielen.

8.4 Digitale Signaturen

Die Echtheit vieler juristischer, finanzieller und anderer Dokumente wird durch die darauf vorhandene handschriftliche Unterschrift gewährleistet. Fotokopien zählen nicht. Bevor jedoch Computernachrichtensysteme den physikalischen Transport von Papier-und-Tinte-Dokumenten ablösen können, muss eine Methode gefunden werden, um Dokumente fälschungssicher zu unterzeichnen.

Einen Ersatz für die handschriftliche Unterschrift zu finden, ist nicht einfach. Grundsätzlich wird ein System benötigt, mit dem eine Partei eine „unterschriebene“ Nachricht unter folgenden Voraussetzungen an eine andere Partei senden kann:

- 1.** Der Empfänger kann die Identität des Senders überprüfen.
- 2.** Der Sender kann den Inhalt der Nachricht später nicht verleugnen.
- 3.** Der Empfänger kann die Nachricht nicht irgendwie selbst fabriziert haben.

Die erste Voraussetzung ist beispielsweise bei Banksystemen wichtig. Wenn der Computer eines Kunden den Computer der Bank auffordert, eine Tonne Gold zu kaufen,

muss der Bankcomputer nachprüfen können, dass der anweisende Computer wirklich zu der Firma gehört, deren Konto belastet werden soll. Anders ausgedrückt muss die Bank den Kunden authentifizieren (und der Kunde muss die Bank authentifizieren).

Die zweite Voraussetzung soll die Bank vor Betrug schützen. Wenn die Bank nun die Tonne Gold kauft und gleich darauf der Goldpreis stark fällt, könnte ein unehrlicher Kunde behaupten, er habe diese Anweisung nie erteilt, und die Bank verklagen. Wenn die Bank die Nachricht vor Gericht vorlegt, behauptet der Kunde, er habe sie nie gesendet. Die Eigenschaft, dass keine Vertragspartei später abstreiten kann, den Vertrag unterzeichnet zu haben, wird als **Nichtabstreichbarkeit** (*nonrepudiation*) bezeichnet. Die digitalen Signaturschemata, die wir im Folgenden untersuchen, helfen, dies sicherzustellen.

Die dritte Voraussetzung ist nötig, um den Kunden in dem Fall zu schützen, dass der Goldpreis plötzlich in die Höhe schießt und die Bank versucht, eine unterzeichnete Nachricht dergestalt zu manipulieren, dass der Kunde nicht eine Tonne, sondern nur einen Barren Gold gekauft hat. In diesem Fälschungsszenario behält die Bank das restliche Gold einfach für sich.

8.4.1 Signaturen mit symmetrischen Schlüsseln

Bei einem der Ansätze für digitale Signaturen ist eine zentrale Stelle vorgesehen, die alles weiß und der alle vertrauen. Nennen wir sie hier einfach *Big Brother* (*BB*). Jeder Benutzer wählt einen Geheimschlüssel und trägt ihn persönlich zu *BB*. Nur Alice und *BB* kennen also den Geheimschlüssel K_A von Alice.

Möchte Alice eine unterschriebene Klartextnachricht P an ihren Banker Bob senden, so erzeugt sie $K_A(B, R_A, t, P)$, wobei B die Identität von Bob ist, R_A eine von Alice gewählte Zufallszahl, t ein Zeitstempel, um die Aktualität sicherzustellen, und $K_A(B, R_A, t, P)$ die Nachricht K_A , die mit ihrem Schlüssel verschlüsselt wurde. Sie sendet die Nachricht dann, wie in ▶ Abbildung 8.18 dargestellt. *BB* erkennt, dass die Nachricht von Alice kommt, entschlüsselt sie und sendet eine Nachricht an Bob. Diese Nachricht enthält den Klartext aus Alices Nachricht und die unterzeichnete Nachricht $K_{BB}(A, t, P)$. Bob führt jetzt die Anfrage von Alice aus.

Was passiert, wenn Alice später leugnet, die Nachricht gesendet zu haben? Schritt 1 ist, dass jeder jeden verklagt (zumindest in den USA). Wird der Fall schließlich vor Gericht verhandelt und leugnet Alice immer noch, die Nachricht gesendet zu haben, wird Bob vom Richter befragt, wie er mit Sicherheit behaupten kann, dass die strittige Nachricht von Alice und nicht von Trudy stammt. Bob verweist zuerst darauf, dass *BB* eine Nachricht ohne Verschlüsselung mit K_A erst gar nicht von Alice angenommen hätte, sodass keine Möglichkeit für Trudy bestanden haben kann, *BB* eine falsche Nachricht von Alice zu senden, ohne dass *BB* dies sofort entdeckt.

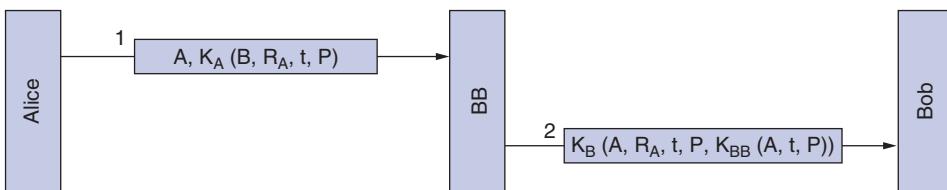


Abbildung 8.18: Digitale Signaturen mit Big Brother.

Bob legt dann sehr dramatisch das Beweisstück A vor: $K_{BB}(A, t, P)$. Er sagt, dass dies eine von BB unterschriebene Nachricht sei, was belege, dass Alice P an Bob gesendet hat. Der Richter bittet BB (dem jeder vertraut), Beweisstück A zu entschlüsseln. Als BB nachweist, dass Bob die Wahrheit sagt, entscheidet das Gericht zugunsten von Bob. Klage abgewiesen.

Ein mögliches Problem beim Signaturprotokoll in Abbildung 8.18 ist, dass Trudy eine der Nachrichten erneut abspielt. Um dieses Problem zu minimieren, werden durchgängig Zeitstempel benutzt. Des Weiteren kann Bob alle kürzlich eingegangenen Nachrichten daraufhin prüfen, ob R_A in einer benutzt wurde. Trifft das zu, wird die Nachricht als Wiederholung abgewiesen. Bob weist sehr alte Nachrichten auf der Grundlage des Zeitstempels ab. Um sich gegen Instant-Replay-Attacken zu schützen, prüft Bob R_A in jeder ankommenden Nachricht, um festzustellen, ob eine solche Nachricht von Alice in der letzten Stunde eingegangen ist. Trifft das nicht zu, kann Bob sicher davon ausgehen, dass es sich um eine neue Anfrage handelt.

8.4.2 Signaturen mit öffentlichen Schlüsseln

Der Einsatz der Kryptografie mit geheimen Schlüsseln für digitale Signaturen hat das inhärente Problem, dass jeder Big Brother vertrauen muss. Außerdem kann Big Brother alle unterschriebenen Nachrichten lesen. Die offenkundigen Kandidaten, die als Big Brother agieren könnten, sind Regierungen, Banken oder Anwälte. Das sind genau die drei Gruppen, die vielen Bürgern alles andere als Vertrauen einflößen. Somit wäre es schön, wenn die Unterzeichnung elektronischer Dokumente auch ohne eine „vertrauenswürdige Stelle“ möglich wäre.

Hier kann die Kryptografie mit öffentlichen Schlüsseln einen wichtigen Beitrag leisten. Nehmen wir an, dass die Ver- und Entschlüsselungsalgorithmen mit öffentlichen Schlüsseln das Merkmal $E(D(P))=P$ zusätzlich zu dem üblichen Merkmal $D(E(P))=P$ haben. (RSA hat dieses Merkmal, deshalb darf man davon ausgehen, dass diese Anforderung nicht übertrieben ist.) Unter der Annahme, dass dies der Fall ist, kann Alice eine unterschriebene Klartextnachricht P durch Übertragung von $E_B(D_A(P))$ an Bob senden. Beachten Sie, dass Alice ihren eigenen (privaten) Schlüssel D_A wie auch den öffentlichen Schlüssel von Bob E_B kennt, sodass Alice eine Nachricht erstellen kann.

Wenn Bob die Nachricht erhält, wandelt er sie, wie üblich, mit seinem privaten Schlüssel um, was $D_A(P)$ ergibt, wie in ►Abbildung 8.19 dargestellt. Er speichert die-

sen Text an einem sicheren Ort und entschlüsselt ihn mit E_A , sodass er den Original-
klartext erhält.

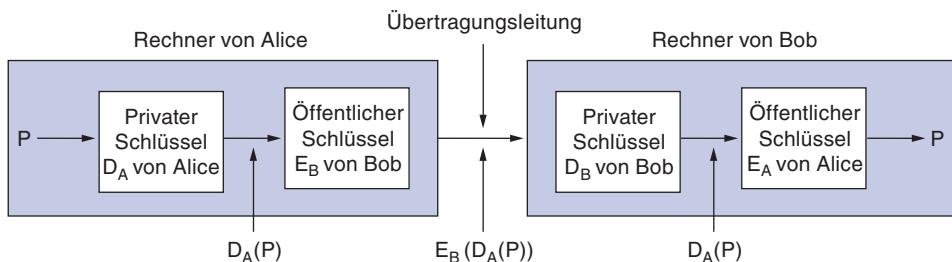


Abbildung 8.19: Digitale Signaturen mithilfe öffentlicher Schlüssel.

Um zu sehen, wie die Signatureigenschaft funktioniert, nehmen wir an, dass Alice im Nachhinein leugnet, Nachricht P an Bob gesendet zu haben. Wird der Fall vor Gericht verhandelt, kann Bob P und $D_A(P)$ als Beweisstücke vorlegen. Der Richter kann leicht prüfen, dass Bob tatsächlich eine gültige mit D_A verschlüsselte Nachricht hat, indem er E_A darauf anwendet. Da Bob nicht weiß, wie der private Schlüssel von Alice lautet, konnte er eine mit diesem Schlüssel verschlüsselte Nachricht nur erhalten haben, wenn sie wirklich von Alice gesendet wurde. Alice wird sich wegen Meineid und Betrug im Gefängnis wahrscheinlich ihre Gedanken über Algorithmen mit öffentlichen Schlüsseln machen.

Die Kryptografie mit öffentlichen Schlüsseln für digitale Signaturen ist zwar eine elegante Methode, wirft aber je nach der Umgebung, in der sie eingesetzt wird, verschiedene Probleme auf. Zum einen kann Bob nur nachweisen, dass eine Nachricht von Alice gesendet wurde, solange D_A geheim bleibt. Gibt Alice ihren Geheimschlüssel preis, ist das Argument nicht mehr stichhaltig, weil dann jeder die Nachricht gesendet haben könnte, auch Bob selbst.

Ein Problem kann beispielsweise entstehen, wenn Bob der Börsenmakler von Alice wäre. Alice beauftragt Bob, eine bestimmte Aktie zu kaufen. Unmittelbar danach fällt der Kurs. Um Ansprüche gegen Bob geltend zu machen, rennt Alice zur Polizei und behauptet, man habe bei ihr eingebrochen und den PC, der ihren Schlüssel speichert, gestohlen. Je nach Land ist sie eventuell haftbar, insbesondere wenn sie behauptet, den Einbruch erst später bei Rückkehr von der Arbeit bemerkt zu haben.

Ein anderes Problem bei elektronischen Signaturen kann entstehen, wenn Alice ihren Schlüssel ändert. Die Tatsache, dass sie das macht, ist absolut legal und wahrscheinlich periodisch sogar empfehlenswert. Wird die Sache später – wie oben beschrieben – vor Gericht verhandelt, wendet der Richter den *aktuellen* E_A auf $D_A(P)$ an und stellt fest, dass daraus nicht P erzeugt wird. Bob wird an dieser Stelle ziemlich dumm dastehen.

Im Prinzip kann für digitale Signaturen jeder Algorithmus mit öffentlichen Schlüsseln verwendet werden. Der De-facto-Industriestandard ist der RSA-Algorithmus. Viele Sicherheitsprodukte basieren darauf. 1991 schlug das NIST aber eine Variante des Algorithmus mit öffentlichen Schlüsseln von El Gamal für seinen neuen Standard

DSS (*Digital Signature Standard*) vor. Die Sicherheit bei El Gamal basiert auf der Schwierigkeit, diskrete Logarithmen zu berechnen, nicht auf der Faktorisierung großer Zahlen.

Wie gewöhnlich, wenn die Regierung versucht, kryptografische Standards zu diktieren, gab es auch in diesem Fall viel Aufruhr. DSS wurde aus folgenden Gründen kritisiert:

1. DSS ist zu geheim (NSA legte das Protokoll zur Verwendung für El Gamal aus).
2. DSS ist zu langsam (10- bis 40-mal langsamer als RSA beim Prüfen von Signaturen).
3. DSS ist zu neu (El Gamal war noch nicht umfassend analysiert worden).
4. DSS ist zu unsicher (fester 512-Bit-Schlüssel).

In einer späteren Überarbeitung wurde der vierte Punkt ausgebügelt. Heute sind bis zu 1 024 Bit zulässig. Dennoch behalten die ersten beiden Punkte ihre Gültigkeit.

8.4.3 Message Digests

An Methoden für elektronische Signaturen wird heftig kritisiert, dass sie oft zwei getrennte Funktionen koppeln: Authentifizierung und Geheimhaltung. Meist ist nur Authentifizierung, nicht aber Geheimhaltung erforderlich. Auch ist der Erhalt einer Exportlizenz oft einfacher, wenn das betreffende System nur die Authentifizierung, aber nicht die Geheimhaltung bereitstellt. Nachfolgend wird ein Authentifizierungsschema beschrieben, bei dem nicht die gesamte Nachricht verschlüsselt werden muss.

Dieses Schema gründet auf dem Konzept einer Ein-Wege-Hash-Funktion, die einen beliebig langen Klartext nimmt und daraus eine Bitfolge fester Länge berechnet. Diese Hash-Funktion *MD*, die auch **Message Digest** (Nachrichten-Kurzfassung) genannt wird, hat drei wichtige Eigenschaften:

1. Mit gegebenem *P* ist es einfach, *MD(P)* zu berechnen.
2. Mit gegebenem *MD(P)* ist es effektiv unmöglich, *P* zu ermitteln.
3. Mit gegebenem *P* kann niemand *P'* finden, sodass *MD(P')=MD(P)*.
4. Eine Änderung der Eingabe in nur einem einzigen Bit erzeugt eine ganz andere Ausgabe.

Um Kriterium 3 zu erfüllen, muss die Hash-Tabelle mindestens 128 Bit lang sein, vorausgesetzt länger. Um das Kriterium 4 zu erfüllen, muss die Hash-Tabelle die Bits sehr gründlich vermischen, nicht unähnlich den symmetrischen Verschlüsselungsalgorithmen, die wir kennengelernt haben.

Die Berechnung eines Message Digests aus einem Stück Klartext ist viel schneller als die Verschlüsselung des gleichen Klartextes durch einen Algorithmus mit öffentlichen Schlüsseln. Deshalb führen Message Digests bei Algorithmen für digitale Signaturen zu höheren Geschwindigkeiten. Um zu sehen, wie dies funktioniert, betrachten wir

nochmals das Signaturprotokoll in Abbildung 8.18. Anstatt P mit $K_{BB}(A, t, P)$ zu unterschreiben, berechnet BB hier den Message Digest durch Anwenden von MD auf P , was $MD(P)$ ergibt. BB bindet dann $K_{BB}(A, t, MD(P))$ als fünftes Element in die mit K_B verschlüsselte Liste ein, die anstelle von $K_{BB}(A, t, P)$ an Bob gesendet wird.

Ergibt sich zwischen den Parteien eine Streitigkeit, kann Bob sowohl P als auch $K_{BB}(A, t, MD(P))$ vorlegen. Nachdem Big Brother das für den Richter entschlüsselt hat, hat Bob $MD(P)$, was garantiert echt ist, sowie das vermeintliche P . Da es für Bob effektiv unmöglich ist, eine andere als die durch diese Hash-Tabelle vorgegebene Nachricht zu finden, kann der Richter leicht davon überzeugt werden, dass Bob die Wahrheit sagt. Diese Art der Verwendung von Message Digests spart sowohl Verschlüsselungszeit als auch Kosten für den Transport und das Speichern von Nachrichten.

Message Digests funktionieren auch in Kryptosystemen mit öffentlichen Schlüsseln, wie in ►Abbildung 8.20 gezeigt. Hier berechnet Alice zuerst den Message Digest ihres Klartextes. Dann unterschreibt sie den Message Digest und sendet ihn mit dem Klartext an Bob. Falls Trudy P auf dem Transit ändert, kann Bob das erkennen, wenn er $MD(P)$ berechnet.

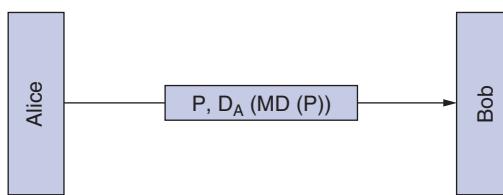


Abbildung 8.20: Digitale Signaturen mit Message Digests.

SHA-1 und SHA-2

Es wurde eine Vielzahl von Message-Digest-Funktionen vorgeschlagen. Eine der am häufigsten verwendeten Funktionen ist **SHA-1** (*Secure Hash Algorithm 1*, sicherer Hash-Algorithmus; NIST, 1993). Wie alle Message Digest mischt er die Bits ausreichend stark, sodass jedes Ausgabebit von jedem Eingabebit betroffen ist. SHA-1 wurde von NSA entwickelt und von NIST in FIPS 180-1 abgesegnet. Es verarbeitet Eingabedaten in 512-Bit-Blöcken und erzeugt einen 160 Bit großen Message Digest. Wie Alice im Regelfall eine nicht geheime, aber unterzeichnete Nachricht an Bob sendet, wird in ►Abbildung 8.21 gezeigt. Hierbei wird ihre Klartextnachricht in den SHA-1-Algorithmus eingespeist, um einen 160-Bit-SHA-1-Hash zu erhalten. Alice unterzeichnet den Hash mit ihrem privaten RSA-Schlüssel und sendet beide, d.h. die Klartextnachricht und den unterzeichneten Hash, an Bob.

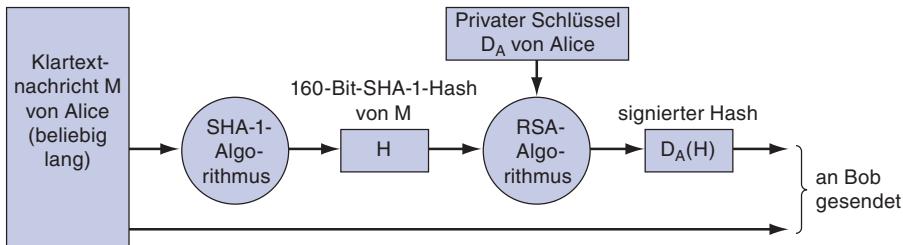


Abbildung 8.21: Einsatz von SHA-1 und RSA zur Unterzeichnung nicht geheimer Nachrichten.

Nach Erhalt der Nachricht berechnet Bob den SHA-1-Hash selbst und wendet den öffentlichen Schlüssel von Alice darauf an, um den ursprünglichen Hash-Wert H zu erhalten. Stimmen beide überein, wird die Nachricht als gültig betrachtet. Da für Trudy keine Möglichkeit besteht, die (Klartext-)Nachricht zu verändern, während sie übertragen wird, und eine neue zu erzeugen, die den Hash-Wert H hat, kann Bob Änderungen, die Trudy an der Nachricht vorgenommen hat, sehr leicht entdecken. Bei Nachrichten, deren Integrität wichtig ist, deren Inhalte aber nicht geheim sind, wird häufig das Schema in Abbildung 8.21 verwendet. Es garantiert bei relativ geringem Berechnungsaufwand, dass alle Änderungen, die an Klartextnachrichten während der Übertragung vorgenommen wurden, mit großer Wahrscheinlichkeit entdeckt werden können.

Betrachten wir nun kurz, wie SHA-1 funktioniert. Zu Beginn wird die Nachricht aufgefüllt, indem ein 1-Bit am Ende gefolgt von so vielen Nullen angefügt wird, wie erforderlich sind, um aus der Länge ein Vielfaches von 512 Bit zu machen. Dann wird eine 64-Bit-Zahl, die die Nachrichtenlänge vor dem Auffüllen enthält, mittels ODER mit den niedrigstwertigen 64 Bit verknüpft. In ►Abbildung 8.22 wird die Nachricht mit dem Auffüllen der rechten Seite gezeigt, da englischer Text und Zahlen von links nach rechts gehen (d.h. unten rechts wird in der Regel als Ende der Zahl interpretiert). Bei Rechnern entspricht diese Ausrichtung Big-Endian-Rechnern wie SPARC und IBM 360 und Nachfolgern, aber SHA-1 fügt immer Auffüllbits am Ende der Nachricht ein, unabhängig davon, welcher Endian-Rechner verwendet wird.

Während der Berechnung verwaltet SHA-1 fünf 32-Bit-Variablen, H_0 bis H_4 , in denen sich der Hash aufbaut. Diese werden in ►Abbildung 8.22b gezeigt. Sie werden auf Konstanten initialisiert, die im Standard angegeben werden.

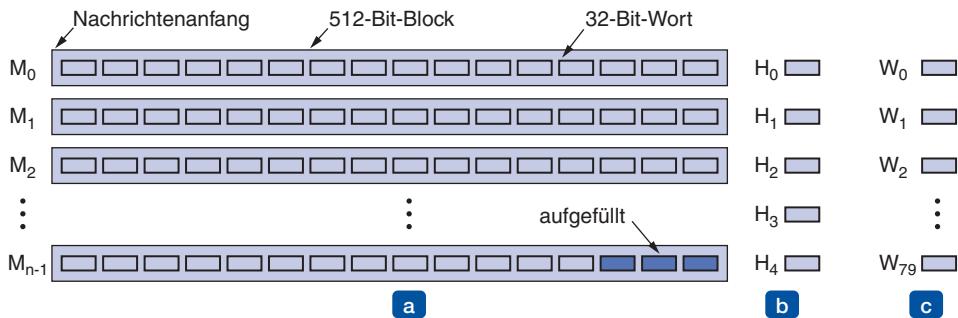


Abbildung 8.22: (a) Eine Nachricht, die auf ein Vielfaches von 512 Bit aufgefüllt ist.
(b) Die Ausgabevariablen. (c) Das Wort-Array.

Jeder der Blöcke M_0 bis M_{n-1} wird nun der Reihe nach verarbeitet. Im aktuellen Block werden die 16 Wörter zuerst an den Anfang eines 80-Wort-Hilfsarrays W kopiert, wie in ► Abbildung 8.22c gezeigt.

Dann werden die weiteren 64 Wörter in W mithilfe der folgenden Formel aufgefüllt:

$$W_i = S^1(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \leq i \leq 79)$$

wobei $S^b(W)$ für die Rotation des 32-Bit-Worts W um b Bits nach links steht. Als Nächstes werden die fünf Hilfsvariablen A bis E mit den Werten von H_0 bis H_4 initialisiert.

Die eigentliche Berechnung kann in Pseudo-C-Code wie folgt ausgedrückt werden:

```
for (i = 0; i < 80; i++) {
    temp = S5(A) + fi(B, C, D) + E + Wi + Ki;
    E = D; D = C; C = S30(B); B = A; A = temp;
}
```

wobei die Konstanten K_i im Standard definiert sind. Die Mischfunktionen f_i werden definiert als

$$\begin{aligned} f_i(B,C,D) &= (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) & (0 \leq i \leq 19) \\ f_i(B,C,D) &= B \text{ XOR } C \text{ XOR } D & (20 \leq i \leq 39) \\ f_i(B,C,D) &= (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) & (40 \leq i \leq 59) \\ f_i(B,C,D) &= B \text{ XOR } C \text{ XOR } D & (60 \leq i \leq 79) \end{aligned}$$

Wenn alle 80 Iterationen der Schleife durchlaufen sind, werden A bis E zu H_0 bis H_4 hinzugezählt.

Nachdem nun der erste 512-Bit-Block verarbeitet wurde, wird der nächste in Angriff genommen. Das W -Array wird wieder von dem neuen Block initialisiert, wobei H aber so bleibt wie gehabt. Ist dieser Block fertig, wird mit dem nächsten begonnen usw., bis alle 512-Bit-Nachrichtenblöcke behandelt wurden. Wenn der letzte Block fertig ist, werden die fünf 32-Bit-Wörter im H -Array als kryptografisches 160-Bit-Hash ausgegeben. Der vollständige C-Code für SHA-1 findet sich in RFC 3174.

Inzwischen sind neuere Versionen von SHA-1 entwickelt worden, die Hashes für 224, 256, 384 und 512 Bit erzeugen. Diese Entwicklungen werden unter der Bezeichnung SHA-2 zusammengefasst. Diese Hashes sind nicht nur länger als SHA-1-Hashes, sondern auch die Digest-Funktion wurde geändert, um einige potenzielle Schwächen von SHA-1 auszubügeln. Noch ist SHA-2 relativ neu und nicht weitverbreitet, doch das wird sich sicherlich bald ändern.

MD5

Der Vollständigkeit halber wollen wir hier noch einen weiteren Digest erwähnen, der recht populär ist. **MD5** ist die fünfte einer Reihe von Hash-Funktionen, die von Ronald Rivest entwickelt wurden (Rivest, 1992). Der Ablauf ist wie folgt: Zuerst beginnt die Funktion mit dem Auffüllen der Nachricht auf eine Länge von 448 Bit (modulo 512). Dann wird die Originallänge der Nachricht als 64-Bit-Ganzzahl angehängt, sodass die Länge der Gesamteingabe einem Vielfachen von 512 Bit entspricht. In jeder Berech-

nungsrunde wird ein 512-Bit-Eingabeblock mit dem 128-Bit-Puffer durchgemischt. Sicherheitshalber wird zum Mischen eine aus der Sinusfunktion gebildete Tabelle herangezogen. Sinn und Zweck, eine bekannte Funktion zu verwenden, liegt darin, den Verdacht zu vermeiden, dass der Entwickler eine clevere Hintertür eingebaut hat, die nur er benutzen kann. Dieser Prozess wird so lange fortgesetzt, bis alle Eingabeblocks aufgebraucht sind. Der Inhalt des 128-Bit-Puffers bildet den Message Digest.

Nach mehr als zehn Jahren steter Verwendung und Untersuchung haben sich Schwächen in MD5 offenbart, unter anderem in Form von verschiedenen Nachrichten mit dem gleichen Hash (Sotirov et al., 2008). Das versetzt jeder Digest-Funktion den Todesstoß, da es bedeutet, dass der Digest nicht mehr sicher für eine Nachricht verwendet werden kann. Damit betrachtet die Sicherheitsgemeinde MD5 als geknackt. Der Algorithmus sollte, wo es möglich ist, ersetzt werden und neue Systeme sollten ihn in ihrem Design nicht mehr vorsehen. Allerdings wird Ihnen MD5 in bestehenden Systemen noch begegnen.

8.4.4 Die Geburtstagsattacke

In der Welt der Kryptografie ist nichts so, wie es zu sein scheint. Man möchte vermuten, dass 2^m Operationen nötig sind, um einen Message Digest mit m Bit zu unterminieren. In Wirklichkeit reichen $2^{m/2}$ Operationen oft aus, indem man die sogenannte **Geburtstagsattacke** (*birthday attack*) anwendet. Diese Vorgehensweise wurde von Yuval (1979) in seinem inzwischen klassischen Aufsatz „How to Swindle Rabin“ veröffentlicht.

Das Konzept für diese Attacke stammt von einer Technik, die Mathematikdozenten oft in ihren Wahrscheinlichkeitskursen präsentieren. Die Frage lautet: Wie viele Studenten müssen in einer Klasse sein, bevor die Wahrscheinlichkeit, dass zwei Leute am gleichen Tag geboren sind, $1/2$ überschreitet? Die meisten Studenten gehen davon aus, dass die Antwort über 100 lautet. De facto sagt die Wahrscheinlichkeitstheorie aber nur 23. Mit 23 Leuten kann man $(23 \times 22)/2 = 253$ verschiedene Paare bilden, die jeweils eine Trefferwahrscheinlichkeit von $1/365$ haben. So gesehen ist das dann nicht mehr überraschend.

Allgemein gibt es für jede Zuordnung zwischen Eingaben und Ausgaben mit n Eingaben (Leuten, Nachrichten usw.) und k möglichen Ausgaben (Geburtstagen, Message Digests usw.) $n(n-1)/2$ Eingabepaare. Ist $n(n-1)/2 > k$, stehen die Chancen, mindestens einen Treffer zu erhalten, ziemlich gut. Somit gibt es mit einiger Wahrscheinlichkeit bei $n > \sqrt{k}$ einen Treffer. Dieses Ergebnis bedeutet, dass ein Message Digest von 64 Bit wahrscheinlich aufgebrochen werden kann, indem man etwa 2^{32} Nachrichten erzeugt und nach zweien mit dem gleichen Message Digest sucht.

Hier ein praktisches Beispiel. Die Informatikfakultät der State-Universität hat eine freie Dozentenstelle, für die sich zwei wissenschaftliche Mitarbeiter bewerben, die wir hier Tom und Dick nennen. Tom wird als Erster zum Bewerbungsgespräch eingeladen, weil er zwei Jahre vor Dick eingestellt wurde. Wenn er erfolgreich ist, hat Dick keine Chance

mehr. Tom weiß, dass die Fakultätsleiterin Marilyn seine Arbeit sehr schätzt. Deshalb bittet er sie, ihm ein Empfehlungsschreiben an den Dekan zu verfassen, der über Toms Fall entscheidet. Nach dem Versenden werden alle Briefe vertraulich.

Marilyn weist ihre Sekretärin Ellen an, dem Dekan einen Brief mit dem besagten Anliegen zu schreiben. Wenn der Brief fertig ist, will Marilyn ihn durchsehen, den Message Digest von 64 Bit berechnen, ihn damit unterschreiben und an den Dekan senden. Ellen kann den Brief später per E-Mail versenden. Bedauerlicherweise für Tom hat Ellen eine Affäre mit Dick, sodass sie den Brief zu dessen Gunsten verfasst. Sie schreibt folgenden Brief mit 32 Optionen in Klammern.

Dear Dean Smith,

This [letter | message] is to give my [honest | frank] opinion of Prof. Tom Wilson, who is [a candidate | up] for tenure [now | this year]. I have [known | worked with] Prof. Wilson for [about | almost] six years. He is an [outstanding | excellent] researcher of great [talent | ability] known [worldwide | internationally] for his [brilliant | creative] insights into [many | a wide variety of] [difficult | challenging] problems.

He is also a [highly | greatly] [respected | admired] [teacher | educator]. His students give his [classes | courses] [rave | spectacular] reviews. He is [our | the Department's] [most popular | best-loved] [teacher | instructor].

[In addition | Additionally] Prof. Wilson is a [gifted | effective] fund raiser. His [grants | contracts] have brought a [large | substantial] amount of money into [the | our] Department. [This money has | These funds have] [enabled | permitted] us to [pursue | carry out] many [special | important] programs, [such as | for example] your State 2000 program. Without these funds we would [be unable | not be able] to continue this program, which is so [important | essential] to both of us. I strongly urge you to grant him tenure.

Zum Nachteil für Tom schreibt Ellen anschließend noch einen zweiten Brief:

Dear Dean Smith,

This [letter | message] is to give my [honest | frank] opinion of Prof. Tom Wilson, who is [a candidate | up] for tenure [now | this year]. I have [known | worked with] Tom for [about | almost] six years. He is a [poor | weak] researcher not well known in his [field | area]. His research [hardly ever | rarely] shows [insight in | understanding of] the [key | major] problems of [the | our] day.

Furthermore, he is not a [respected | admired] [teacher | educator]. His students give his [classes | courses] [poor | bad] reviews. He is [our | the Department's] least popular [teacher | instructor], known [mostly | primarily] within [the | our] Department for his [tendency | propensity] to [ridicule | embarrass] students [foolish | imprudent] enough to ask questions in his classes.

[In addition | Additionally] Tom is a [poor | marginal] fund raiser. His [grants | contracts] have brought only a [meager | insignificant] amount of money into [the | our] Department. Unless new [money is | funds are] quickly located, we may have to cancel some essential programs, such as your State 2000 program. Unfortunately, under these [conditions | circumstances] I cannot in good [conscience | faith] recommend him to you for [tenure | a permanent position].

Nun setzt sich Ellen an ihren Computer und berechnet die 2^{32} Message Digests der zwei Briefe über Nacht. Die Chancen stehen gut, dass ein Digest des ersten Briefs mit

einem des zweiten Briefs übereinstimmt. Andernfalls kann sie noch ein paar Optionen einfügen und es am Wochenende erneut versuchen. Nehmen wir an, sie findet einen Treffer. Nennen wir den „guten“ Brief *A* und den „schlechten“ *B*.

Ellen sendet Brief *A* nun per E-Mail an Marilyn zur Genehmigung. Brief *B* hält sie ganz geheim und zeigt ihn niemand. Marilyn genehmigt ihn selbstverständlich, berechnet ihr Message Digest von 64 Bit, unterschreibt und sendet den unterschriebenen Digest per E-Mail an den Dekan Smith. Unabhängig davon schickt Ellen den Brief *B* per E-Mail an den Dekan (nicht Brief *A*, was sie eigentlich tun sollte).

Nach dem Erhalt des Briefs und dem unterschriebenen Message Digest führt der Dekan den Message-Digest-Algorithmus auf Brief *B* aus, nimmt die Empfehlungen von Marilyn an und feuert Tom. Der Dekan erkennt nicht, dass es Ellen gelungen ist, zwei Briefe mit dem gleichen Message Digest zu generieren und ihm einen anderen zu senden als den, den Marilyn gesehen und genehmigt hat. (Wahlweises Ende: Ellen sagt Dick, was sie getan hat. Dick ist empört und macht mit ihr Schluss. Ellen ist sauer und gesteht alles Marilyn. Marilyn ruft den Dekan an. Tom erhält am Ende die Anstellung.) Mit SHA-1 ist die Geburtstagsattacke schwierig, weil es sogar bei der absurd Geschwindigkeit von einer Billion Digests pro Sekunde über 32 000 Jahre dauern würde, um alle 2^{80} Digests der zwei Briefe mit je 80 Varianten zu berechnen, und auch dann ist kein Treffer garantiert. Mit einer Cloud von 1 000 000 parallel arbeitenden Chips werden aus den 32 000 Jahren 2 Wochen.

8.5 Verwaltung öffentlicher Schlüssel

Durch die Verschlüsselung mit öffentlichen Schlüsseln können Personen, die keinen gemeinsamen Schlüssel haben, sicher miteinander kommunizieren. Darüber hinaus ist auch die Unterzeichnung von Nachrichten möglich, ohne dass eine vertrauenswürdige dritte Partei beteiligt ist. Zudem ermöglichen unterzeichnete Message Digests, dass der Empfänger die Integrität der empfangenen Nachrichten leicht und sicher verifizieren kann.

Hier gibt es aber noch ein Problem, über das wir zu schnell hinweggegangen sind. Wenn Alice und Bob einander nicht kennen, wie erhalten sie dann die gegenseitigen öffentlichen Schlüssel, um den Kommunikationsprozess einzuleiten? Die offensichtliche Lösung – Veröffentlichung des öffentlichen Schlüssels auf der eigenen Website – funktioniert aus dem folgenden Grund nicht. Angenommen Alice möchte den öffentlichen Schlüssel von Bob auf seiner Website nachsehen. Wie geht sie dabei vor? Sie beginnt, indem sie die URL von Bob eintippt. Ihr Browser schlägt dann die DNS-Adresse von Bobs Homepage nach und sendet an sie eine GET-Anforderung, wie in ► Abbildung 8.23 gezeigt. Unglücklicherweise fängt Trudy die Anforderung ab und antwortet mit einer gefälschten Homepage, vermutlich einer Kopie der Homepage von Bob, wobei aber der öffentliche Schlüssel von Bob durch den öffentlichen Schlüssel von Trudy ersetzt wird. Wenn Alice nun die erste Nachricht mit E_T verschlüsselt, entschlüsselt Trudy diese, liest sie, verschlüsselt sie mit dem öffentlichen Schlüssel von

Bob und sendet sie an Bob, der keine Ahnung davon hat, dass Trudy seine eingehenden Nachrichten liest. Noch schlimmer ist, dass Trudy die Nachrichten verändern könnte, bevor sie sie für Bob erneut verschlüsselt. Daher ist logischerweise ein Vorgehen erforderlich, mit dem sichergestellt werden kann, dass öffentliche Schlüssel sicher ausgetauscht werden können.

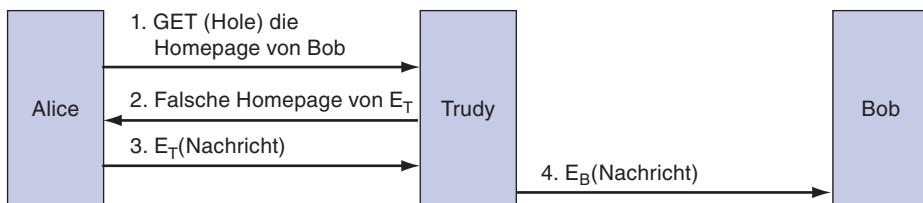


Abbildung 8.23: Eine Möglichkeit, wie Trudy die Verschlüsselung mit einem öffentlichen Schlüssel unterlaufen kann.

8.5.1 Zertifikate

Für einen ersten Versuch, öffentliche Schlüssel sicher zu verteilen, könnten wir uns eine Schlüsselverteilung vorstellen, die 24 Stunden täglich online verfügbar ist und öffentliche Schlüssel bei Anforderung bereitstellt. Eines der vielen Probleme ist, dass diese Lösung nicht skalierbar ist und das **Schlüsselverteilungszentrum** sicherlich schnell ein Engpass wird. Des Weiteren würde die Sicherheit im Internet zusammenbrechen, wenn das Zentrum je abstürzen sollte.

Aus diesen Gründen wurde eine andere Lösung entwickelt, bei der das Schlüsselverteilungszentrum nicht die ganze Zeit online sein muss. Es muss de facto überhaupt nicht online sein. Stattdessen werden öffentliche Schlüssel, die Personen, Unternehmen und anderen Organisationen gehören, zertifiziert. Eine Organisation, die öffentliche Schlüssel zertifiziert, wird als **Zertifizierungsstelle** (*CA, Certification Authority*) bezeichnet.

Als Beispiel nehmen wir an, dass Bob Alice und anderen ihm teilweise unbekannten Personen gestatten möchte, mit ihm sicher zu kommunizieren. Er kann dann mit seinem öffentlichen Schlüssel und seinem Pass oder Führerschein zu der Zertifizierungsstelle gehen und die Zertifizierung beantragen. Die Zertifizierungsstelle stellt dann ein Zertifikat, wie das in ►Abbildung 8.24, aus und unterzeichnet den SHA-1-Hash mit dem privaten Schlüssel der Zertifizierungsstelle. Bob zahlt eine Gebühr an die Zertifizierungsstelle und erhält eine Diskette mit dem Zertifikat und dem signierten Hash.

I hereby certify that the public key
 19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
 belongs to
 Robert John Smith
 12345 University Avenue
 Berkeley, CA 94702
 Birthday: July 4, 1958
 Email: bob@superdupernet.com

SHA-1-Hash des obigen Zertifikats, der mit dem privaten Schlüssel der Zertifizierungsstelle unterzeichnet ist

Abbildung 8.24: Ein mögliches Zertifikat und der unterzeichnete Hash.

Die primäre Aufgabe eines Zertifikats ist, den öffentlichen Schlüssel mit dem Namen eines Principals (Person, Unternehmen usw.) zu verbinden. Zertifikate selbst sind nicht geheim oder geschützt. Bob könnte sich beispielsweise entschließen, sein neues Zertifikat mit einem Link auf der Hauptseite seine Website zu stellen, der besagt: Klicken Sie hier, um das Zertifikat für meinen öffentlichen Schlüssel zu erhalten. Ein Klick würde sowohl das Zertifikat als auch den Signaturblock (den unterzeichneten SHA-1-Hash des Zertifikats) zurückliefern.

Gehen wir das Szenario in Abbildung 8.23 noch einmal durch. Wenn Trudy die Anforderung von Alice für die Homepage von Bob abfängt, wie kann sie vorgehen? Sie kann ihr eigenes Zertifikat und den Signaturblock auf der gefälschten Seite einfügen. Wenn aber Alice das Zertifikat liest, sieht sie sofort, dass sie nicht mit Bob kommuniziert, da der Name von Bob fehlt. Trudy kann die Homepage von Bob während der Übertragung verändern, indem sie den öffentlichen Schlüssel von Bob durch ihren eigenen ersetzt. Wenn aber Alice den SHA-1-Algorithmus für das Zertifikat ausführt, erhält sie einen Hash, der nicht mit dem übereinstimmt, den sie erhält, wenn sie den bekannten öffentlichen Schlüssel der Zertifizierungsstelle auf den Signaturblock anwendet. Da Trudy den privaten Schlüssel der Zertifizierungsstelle nicht kennt, hat sie keine Möglichkeit, einen Signaturblock zu erstellen, der den Hash der geänderten Webseite mit ihrem öffentlichen Schlüssel enthält. Auf diese Weise kann Alice sichergehen, dass sie den öffentlichen Schlüssel von Bob und nicht von Trudy oder jemand anderem besitzt. Und wie versprochen muss die Zertifizierungsstelle nicht immer zur Verifikation online sein, sodass ein möglicher Engpass vermieden wird.

Wenngleich es die Standardaufgabe eines Zertifikats ist, einen öffentlichen Schlüssel an einen Principal zu binden, kann ein Zertifikat auch dazu benutzt werden, einen öffentlichen Schlüssel an ein **Attribut** zu binden. Ein Zertifikat könnte beispielsweise besagen: Der öffentliche Schlüssel gehört einer Person über 18. Es könnte als Nachweis verwendet werden, dass der Eigentümer des privaten Schlüssels nicht minderjährig ist und damit auf Inhalte zugreifen kann, die nicht für Minderjährige gedacht sind, wobei aber die Identität der Person nicht preisgegeben wird. In der Regel wird die Person, die das Zertifikat besitzt, dieses an Websites, Principals oder Prozesse senden, für die die Altersgrenze wichtig ist. Diese Sites, Principals oder Prozesse erzeugen dann eine Zufallszahl und verschlüsseln sie mit dem öffentlichen Schlüssel im

Zertifikat. Wenn der Eigentümer dies entschlüsseln und zurücksenden kann, so ist das der Beweis, dass der Eigentümer tatsächlich das in dem Zertifikat angegebene Attribut besitzt. Alternativ könnte eine Zufallszahl erzeugt werden, um einen Sitzungsschlüssel für die nachfolgende Unterhaltung zu erzeugen.

Ein weiteres Beispiel für eine Situation, in der ein Zertifikat ein Attribut enthalten kann, ist ein objektorientiertes Verteilungssystem. Jedes Objekt besitzt in der Regel mehrere Methoden. Der Eigentümer des Objekts könnte jedem Kunden ein Zertifikat zur Verfügung stellen, das ein Bitmuster enthält, welche Methoden der Kunde aufrufen darf, wobei das Bitmuster mit einem unterzeichneten Zertifikat an den öffentlichen Schlüssel gebunden wird. Auch hier gilt, wenn der Zertifikatinhaber den Besitz des entsprechenden privaten Schlüssels nachweisen kann, kann er die in dem Bitmuster freigegebenen Methoden ausführen. Dieser Ansatz zeichnet sich dadurch aus, dass die Identität des Eigentümers nicht bekannt sein muss, was vor allem in Situationen nützlich ist, in denen Datenschutz wichtig ist.

8.5.2 X.509

Wenn jeder, der eine Signatur benötigt, mit einer anderen Art von Zertifikat zu der Zertifizierungsstelle liefe, würde die Verwaltung all dieser verschiedenen Formate schnell aus dem Ruder laufen. Um dieses Problem zu lösen, wurde ein Standard für Zertifikate entwickelt und von der ITU genehmigt. Dieser Standard wird als [X.509](#) bezeichnet und kommt im Internet häufig zum Einsatz. Seit der ersten Standardisierung hat er drei Versionen erlebt. Wir behandeln hier Version 3.

X.509 wurde sehr von der OSI-Welt beeinflusst und hat einige der schlechtesten Funktionen übernommen (wie Benennung und Codierung). Erstaunlicherweise hat die IETF die OSI-Konzepte bei X.509 durchgehen lassen, obwohl sie in fast allen anderen Bereichen, von den Rechneradressen und den Transportprotokollen bis hin zu den E-Mail-Formaten, OSI ignoriert hat und bestrebt war, die Sache richtig zu machen. Die IETF-Version von X.509 wird in RFC 5280 beschrieben.

X.509 ist im Grunde eine Möglichkeit, Zertifikate zu beschreiben. Die wichtigsten Felder eines Zertifikats sind in ▶ Abbildung 8.25 aufgeführt. Die angegebenen Beschreibungen sollen eine allgemeine Vorstellung davon vermitteln, was die Felder tun. Weitere Informationen finden Sie im Standard selbst oder im RFC 2459. Wenn Bob beispielsweise in der Kreditabteilung der Money Bank arbeitet, könnte seine X.500-Adresse wie folgt lauten:

/C=US/O=MoneyBank/OU=Loan/CN=Bob/

wobei *C* für das Land steht, *O* für die Organisation, *OU* für seine Abteilung und *CN* für seinen Namen. Zertifizierungsstellen und andere Entitäten werden ähnlich benannt. Ein erhebliches Problem bei X.500-Namen ist, dass Alice, wenn sie versucht, *bob@moneybank.com* zu kontaktieren, und ein Zertifikat mit dem X.500-Namen erhält, diesem Zertifikat nicht entnehmen kann, dass es sich auf den Bob bezieht, den sie kontaktieren möchte. Zum Glück sind ab Version 3 DNS-Namen anstatt X.500-Namen zulässig, sodass dieses Problem irgendwann verschwindet.

| Feld | Bedeutung |
|---------------------|---|
| Version | Angabe der X.509-Version. |
| Serial Number | Diese Zahl bestimmt zusammen mit dem Namen der Zertifizierungsstelle das Zertifikat eindeutig. |
| Signature Algorithm | Der zur Signierung des Zertifikats verwendete Algorithmus. |
| Issuer | X.500-Name der Zertifizierungsstelle. |
| Validity Period | Die Gültigkeitsdauer mit Anfangs- und Endzeitpunkt. |
| Subject Name | Der Name des Zertifikatinhabers, dessen Schlüssel zertifiziert wird. |
| Public Key | Der öffentliche Schlüssel des Zertifikatinhabers und die Kennung des verwendeten Algorithmus. |
| Issuer ID | Eine optionale Kennung, die die Ausgabestelle des Zertifikats eindeutig bestimmt. |
| Subject ID | Eine optionale Kennung, die den Zertifikatinhaber eindeutig festlegt. |
| Extensions | Es wurden viele Erweiterungen definiert. |
| Signature | Die Signatur des Zertifikats (mit dem privaten Schlüssel der Zertifizierungsstelle unterschrieben). |

Abbildung 8.25: Die wichtigsten Felder eines X.509-Zertifikats.

Zertifikate werden mit OSI **ASN.1** (*Abstract Syntax Notation 1*) codiert, die man sich als eine Art Struktur in C vorstellen kann, aber mit einer ganz besonderen und ausführlichen Notation. Weitere Informationen über X.509 finden Sie in Ford und Baum (2000).

8.5.3 PKI – Infrastruktur für öffentliche Schlüssel

Es ist klar, dass eine einzige Zertifizierungsstelle, die weltweit alle Zertifikate ausgibt, nicht funktionieren kann. Sie würde unter der Last zusammenbrechen und darüber hinaus eine zentrale Fehlerquelle darstellen. Eine mögliche Lösung könnten mehrere Zertifizierungsstellen sein, die alle von der gleichen Organisation betrieben werden und alle den gleichen privaten Schlüssel zur Unterzeichnung der Zertifikate verwenden. Dies würde sicherlich das Last- und Fehlerproblem lösen, führt aber zu einem neuen Problem: Schlüssellecks (*key leakage*). Wenn Dutzende von Servern in der ganzen Welt verteilt sind, die alle den privaten Schlüssel der Zertifizierungsstelle enthalten, erhöht dies die Gefahr erheblich, dass der private Schüssel gestohlen oder anderweitig bekannt wird. Da eine Preisgabe des Schlüssels die weltweite elektronische Sicherheitsinfrastruktur lahmlegen würde, ist es sehr riskant, eine zentrale Zertifizierungsstelle einzurichten.

Darüber hinaus muss man fragen, welche Organisation als Zertifizierungsstelle in Frage käme? Es gibt wohl kaum eine Behörde, die weltweit von allen als legitim und vertrauenswürdig akzeptiert würde. In einigen Ländern würden die Leute darauf bestehen, dass dies der Staat übernimmt, wohingegen in anderen Ländern die Mehrheit dafür wäre, dies nicht in staatliche Hände zu legen.

Aus diesen Gründen wurde eine andere Vorgehensweise zur Zertifizierung öffentlicher Schlüssel entwickelt. Diese wird als **PKI** (*Public Key Infrastructure*) bezeichnet. In diesem Abschnitt fassen wir die allgemeine Funktionsweise zusammen, obwohl es viele Vorschläge gibt, sodass sich die Details im Laufe der Zeit noch weiterentwickeln werden.

Eine PKI besteht aus mehreren Komponenten, einschließlich Benutzer, Zertifizierungsstellen, Zertifikate und Verzeichnisse. Die Aufgabe der PKI ist es, diese Komponenten zu strukturieren und Standards für verschiedene Dokumente und Protokolle zu definieren. Eine besonders einfache Form einer PKI ist eine Hierarchie von Zertifizierungsstellen, wie in ► Abbildung 8.26 dargestellt. In diesem Beispiel haben wir drei Ebenen, in der Praxis können es aber durchaus weniger oder mehr sein. Die oberste Zertifizierungsstelle, die Wurzel (*root*), zertifiziert die Zertifizierungsstellen auf der zweiten Ebene, die wir als **regionale Zertifizierungsstellen** (*RA*, *Regional Authority*) bezeichnen, da sie sich auf eine bestimmte geografische Region wie einem Land oder einen Kontinent beziehen. Der Begriff ist nicht standardisiert, aber für die verschiedenen Ebenen im Baum gibt es keine wirklichen Standardbegriffe. Diese regionalen Zertifizierungsstellen zertifizieren ihrerseits die eigentlichen Zertifizierungsstellen, die die X.509-Zertifikate an Organisationen und Personen ausgeben. Wenn die oberste Stelle eine neue regionale Zertifizierungsstelle autorisiert, erzeugt sie ein X.509-Zertifikat, das die Zertifizierung der regionalen Zertifizierungsstelle bestätigt und den öffentlichen Schlüssel der regionalen Zertifizierungsstelle enthält. Das unterzeichnete Zertifikat wird dann der regionalen Zertifizierungsstelle ausgehändigt. Genauso verfährt die regionale Zertifizierungsstelle, wenn sie eine neue Zertifizierungsstelle bestätigt: Sie erstellt und unterzeichnet einen öffentlichen Schlüssel, der die Genehmigung darstellt und den öffentlichen Schlüssel der Zertifizierungsstelle enthält.

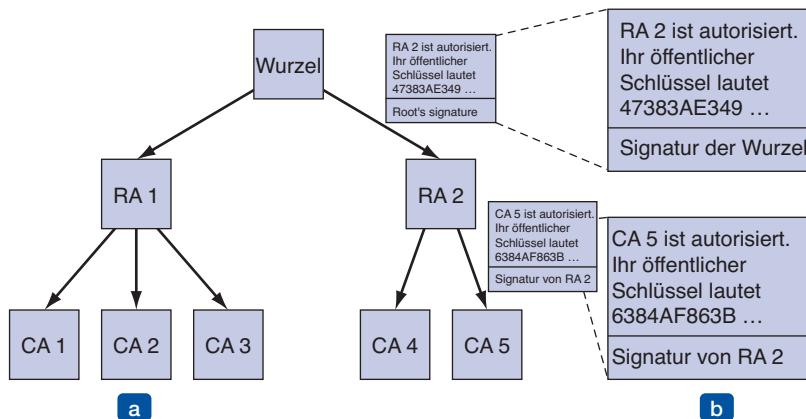


Abbildung 8.26: (a) Hierarchische PKI. (b) Eine Kette aus Zertifikaten.

Die PKI funktioniert wie folgt: Angenommen Alice benötigt den öffentlichen Schlüssel von Bob, um mit ihm zu kommunizieren. Sie sucht und findet ein entsprechendes Zertifikat, das von der Zertifizierungsstelle 5 (CA 5) unterzeichnet wurde. Alice hat aber noch nie etwas von CA 5 gehört. Nach allem, was sie weiß, könnte CA 5 die zehnjährige Tochter von Bob sein. Sie könnte zu CA 5 gehen und sagen: „Beweisen Sie Ihre Legiti-

mität.“ CA 5 antwortet mit einem Zertifikat, das sie von der regionalen Zertifizierungsstelle 2 (RA 2) erhalten hat. Es enthält den öffentlichen Schlüssel von CA 5. Nachdem Alice damit über den öffentlichen Schlüssel von CA 5 verfügt, kann sie prüfen, ob das Zertifikat von Bob tatsächlich von CA 5 unterzeichnet wurde und somit legal ist.

Es sei denn, RA 2 ist der zwölfjährige Sohn von Bob. Daher muss sie im nächsten Schritt bei RA2 um einen Nachweis der Legitimität anfragen. Die Antwort ist ein Zertifikat, das von der obersten Stelle unterzeichnet ist und den öffentlichen Schlüssel von RA2 enthält. Nun ist Alice sicher, dass sie den öffentlichen Schlüssel von Bob hat. Wie aber findet Alice den öffentlichen Schlüssel der obersten Stelle? Durch Zauberei. Man geht davon aus, dass jeder den öffentlichen Schlüssel der obersten Stelle kennt. So könnte beispielsweise ihr Browser den öffentlichen Schlüssel der obersten Stelle gleich ab Lieferung integriert haben.

Bob ist ein freundlicher Mensch und möchte Alice nicht zu viel Arbeit verursachen. Er weiß, dass sie CA 5 und RA 2 prüfen muss; daher übergibt er ihr, um ihr die Arbeit zu erleichtern, die beiden erforderlichen Zertifikate zusammen mit seinem. Nun kann sie mit ihrem Wissen des öffentlichen Schlüssels der obersten Stelle die obersten Zertifikate und den hierin enthaltenen öffentlichen Schlüssel prüfen, um den zweiten zu verifizieren. Auf diese Weise muss Alice niemanden kontaktieren, um die Verifikation auszuführen. Da alle Zertifikate unterschrieben sind, kann sie Fälschungsversuche an den Inhalten leicht entdecken. Eine Zertifikatkette, die zur obersten Stelle zurückgeht, wird manchmal auch als **Zertifizierungspfad** (*certification path*) oder **Vertrauenskette** (*chain of trust*) bezeichnet. Diese Technik kommt in der Praxis häufig zum Einsatz.

Natürlich haben wir immer noch das Problem, wer die oberste Stelle betreibt. Die Lösung ist hier, nicht eine oberste Stelle zu haben, sondern viele, die wiederum ihre eigenen regionalen Zertifizierungsstellen und deren untergeordneten Zertifizierungsstellen haben. Moderne Browser speichern öffentliche Schlüssel für über 100 oberste Zertifizierungsstellen, die manchmal als **Sicherheitsanker** (*trust anchor*) bezeichnet werden. Auf diese Weise kann vermieden werden, dass es nur eine einzige international vertrauenswürdige Einrichtung gibt.

Aber nun besteht das Problem, wie der Browser-Hersteller entscheidet, welche Sicherheitsanker vertrauenswürdig sind und welche eher nicht. Es bleibt letztlich dem Benutzer überlassen, dem Browser-Hersteller zu vertrauen, dass dieser eine gute Auswahl trifft und nicht einfach alle Sicherheitsanker aufnimmt, die bereit sind, eine Aufnahmegebühr zu zahlen. Viele Brower gewähren den Benutzern Einsicht in die Schlüssel der obersten Zertifizierungsstelle (in der Regel in Form von Zertifikaten, die von der obersten Zertifizierungsstelle unterzeichnet wurden) und erlauben, diejenigen zu entfernen, die nicht vertrauenswürdig erscheinen.

Verzeichnisse

Ein weiteres Thema für jede PKI ist, wo die Zertifikate (und deren Ketten zurück zu einem Sicherheitsanker) gespeichert werden. Eine Möglichkeit ist, dass jeder Benutzer seine eigenen Zertifikate speichert. Dies ist zwar sicher (d.h., Benutzer können die

unterzeichneten Zertifikate nicht verändern, ohne entdeckt zu werden), aber unbedeutsam. Ein alternativer Vorschlag ist die Verwendung des DNS als Zertifikatverzeichnis. Bevor Alice also Bob kontaktiert, muss sie wahrscheinlich die IP-Adresse unter Verwendung des DNS nachschlagen. Warum sollte dann das DNS nicht gleich die gesamte Zertifikatkette zusammen mit seiner IP-Adresse zurückgeben?

Manche Leute glauben, dass dies eine gute Lösung sei. Andere wiederum bevorzugen spezielle Verzeichnisserver, deren einzige Aufgabe die Verwaltung der X.509-Zertifikate ist. Solche Verzeichnisse könnten Nachschlagedienste bereitstellen, die sich der Eigenschaften der X.500-Namen bedienen. Theoretisch könnte ein solcher Verzeichnisdienst Anfragen wie die folgende beantworten: „Erstelle mir eine Liste aller Personen namens Alice, die in der Vertriebsabteilung eines Unternehmens in Amerika oder Kanada tätig sind.“

Annullieren von Zertifikaten

In der realen Welt gibt es ebenso viele Zertifikate wie Pässe oder Führerscheine. Manchmal können diese Zertifikate entzogen werden, wie Führerscheine bei Fahrten mit zu viel Promille oder anderen Verletzungen der Verkehrsregeln. Das gleiche Problem tritt auch in der digitalen Welt auf: Die Vergabestelle eines Zertifikats kann dies unter Umständen zurückziehen, wenn die Organisation, die es besitzt, auf irgendeine Weise Missbrauch damit treibt. Es kann auch zurückgezogen werden, wenn der private Schlüssel des Objekts preisgegeben wurde oder, noch schlimmer, wenn der private Schlüssel der Zertifizierungsstelle in irgendeiner Weise manipuliert wurde. Daher muss eine PKI auch den Fall der Annulierung berücksichtigen, was alles etwas komplizierter macht.

Ein erster Schritt in diese Richtung ist, dass jede Zertifizierungsstelle in regelmäßigen Abständen eine **Sperrliste der Zertifikate** (*certificate revocation list*) ausgibt, die sie zurückgezogen hat. Da Zertifikate Ablaufdaten enthalten, muss die Sperrliste nur die Seriennummern der Zertifikate enthalten, die noch nicht abgelaufen sind. Ist das Datum abgelaufen, wird ein Zertifikat automatisch ungültig, sodass keine Unterscheidung zwischen den abgelaufenen und den wirklich zurückgezogenen erforderlich ist. In beiden Fällen können sie nicht mehr verwendet werden.

Bedauerlicherweise bedeutet die Einführung einer Sperrliste, dass ein Benutzer, der ein bestimmtes Zertifikat nutzen möchte, sich die Sperrliste besorgen muss, um zu sehen, ob das Zertifikat annulierte wurde. Ist dies der Fall, sollte es nicht verwendet werden. Steht das Zertifikat aber nicht in der Liste, könnte es kurz nach deren Veröffentlichung annulierte worden sein. Daher ist die einzige Möglichkeit, hier wirklich sicherzugehen, die Zertifizierungsstelle direkt zu fragen. Bei der nächsten Verwendung des gleichen Zertifikats muss die Zertifizierungsstelle wieder gefragt werden, da das Zertifikat wenige Sekunden zuvor annulierte worden sein könnte.

Eine weitere Komplikation ist, dass das annulierte Zertifikat möglicherweise auch wieder aktiviert worden sein kann, wenn es beispielsweise aufgrund von nicht bezahlten Gebühren gesperrt wurde, die aber in der Zwischenzeit gezahlt wurden.

Wenn man die Annullierung (und möglicherweise die Reaktivierung) einführt, verliert man eine der besten Eigenschaften, die Zertifikate haben, nämlich dass sie ohne Kontaktieren der Zertifizierungsstelle verwendet werden können.

Und wo sollten Sperrlisten gespeichert werden? Ein guter Ort wäre dort, wo auch die Zertifikate gespeichert sind. Eine Strategie der Zertifizierungsstelle kann darin bestehen, die Sperrlisten in periodischen Abständen auszugeben, damit die Verzeichnisse sie verarbeiten können, indem sie die annullierten Zertifikate entfernen. Wenn die Zertifikate nicht in Verzeichnissen gespeichert werden, können die Sperrlisten an verschiedenen Stellen im Netz zwischengespeichert werden. Da eine Sperrliste ein signiertes Dokument ist, wird ein Fälschungsversuch leicht entdeckt.

Wenn die Zertifikate eine lange Lebensdauer haben, werden auch die Sperrlisten lang. Wenn beispielsweise Kreditkarten fünf Jahre gültig sind, ist die Anzahl der ausstehenden Annullierungen viel größer, als wenn alle drei Monate neue Karten ausgegeben werden. Eine Standardlösung für lange Sperrlisten ist, ab und zu eine Masterliste auszugeben, zwischendurch aber häufiger Aktualisierungen zu veröffentlichen. Dies reduziert die für die Verteilung der Sperrlisten erforderliche Bandbreite.

8.6 Kommunikationssicherheit

Damit haben wir unsere Untersuchung der verfügbaren Werkzeuge abgeschlossen und Ihnen die wichtigsten Techniken und Protokolle vorgestellt. Der Rest dieses Kapitels beschreibt, wie diese Techniken in der Praxis zur Gewährleistung der Netzsicherheit eingesetzt werden. Am Ende des Kapitels gehen wir noch auf die sozialen Aspekte der Sicherheit ein.

Die folgenden vier Abschnitte sind dem Thema Kommunikationssicherheit gewidmet, also wie die Bits unerkannt und ohne Änderung von der Quelle zum Ziel gelangen und wie unerwünschte Bits draußen vor der Tür bleiben. Dies sind beileibe nicht die einzigen Sicherheitsthemen in der Netzwerktechnik, aber sie gehören sicherlich zu den wichtigsten, sodass wir damit beginnen wollen.

8.6.1 IPsec

Der IETF war seit Jahren bekannt, dass es so gut wie keine Sicherheit im Internet gab. Diese zu ergänzen, war nicht einfach, weil ein Krieg darüber entbrannte, wo diese Eingriffe genau erfolgen sollten. Die meisten Sicherheitsexperten sind der Meinung, dass für absolute Sicherheit die Verschlüsselung und Integritätsprüfung von Ende-zu-Ende durchgeführt werden müssen (also in der Anwendungsschicht). Das bedeutet, der Quellprozess verschlüsselt und/oder schützt die Integrität der Daten und sendet diese an den Zielprozess, wo sie entschlüsselt und/oder verifiziert werden. Irgendwelche Manipulationen, die zwischen diesen beiden Prozessen oder in den beiden Betriebssystemen vorgenommen werden, können so entdeckt werden. Das Problem bei diesem Ansatz ist, dass alle Anwendungen geändert werden müssen, da sie jetzt für die

Sicherheit zuständig sind. Angesichts dieses Aufwands besteht der nächstbeste Ansatz darin, die Verschlüsselung in der Transportschicht beziehungsweise in einer neuen Schicht zwischen der Anwendungsschicht und der Transportschicht vorzunehmen, sodass immer noch Ende-zu-Ende gewährleistet ist, aber keine Anwendungen geändert werden müssen.

Die gegenteilige Ansicht ist, dass Benutzer von Sicherheit keine Ahnung haben und sich nicht richtig schützen können und darüber hinaus auch niemand vorhandene Programme modifizieren möchte, sodass die Vermittlungsschicht die Pakete authentifizieren und/oder verschlüsseln sollte, ohne Einbezug der Benutzer. Nach Jahren harter Kämpfe gewann diese Ansicht ausreichend Anhänger, sodass ein Sicherheitsstandard für die Vermittlungsschicht definiert wurde. Teilweise wurde damit argumentiert, dass bei einer Verschlüsselung auf der Vermittlungsschicht sicherheitsbewusste Benutzer immer noch die Möglichkeit hätten, selbst Maßnahmen zu ergreifen, während Benutzer ohne Sicherheitsbewusstsein auf alle Fälle geschützt wären.

Das Ergebnis dieser Auseinandersetzung war ein Entwurf namens **IPsec** (*IP security*, IP-Sicherheit), der unter anderem in den RFCs 2401, 2402 und 2406 beschrieben wurde. Nicht alle Benutzer wollen eine Verschlüsselung (da sie berechnungstechnisch aufwendig ist). Anstatt jedoch die Verschlüsselung optional zu machen, entschied man sich für eine obligatorische Verschlüsselung mit der Option, einen Null-Algorithmus zu verwenden. Der Null-Algorithmus wird in RFC 2410 beschrieben und zeichnet sich durch Einfachheit, leichte Implementierbarkeit und hohe Geschwindigkeit aus.

Der gesamte IPsec-Entwurf ist ein Framework von mehreren Diensten, Algorithmen und Granularitäten. Es gibt mehrere Dienste, weil nicht jeder den Preis dafür zahlen möchte, alle Dienste immer bereit zu haben, sodass die Dienste à la carte verfügbar sind. Die wichtigsten Dienste sind Datenschutz, Datenintegrität, Schutz vor Replay-Angriffen (Eindringling spielt die Konversation erneut ab). Alle diese Dienste basieren auf einer symmetrischen Verschlüsselung, da hohe Leistung äußerst wichtig ist.

Der Grund, warum es mehrere Algorithmen gibt, ist, dass ein Algorithmus, der heute als sicher gilt, morgen aufgebrochen werden kann. Durch diese Unabhängigkeit von einem bestimmten Algorithmus kann das IPsec-Framework weiterbestehen, selbst wenn einer der Algorithmen später gebrochen wird.

Der Grund, warum es verschiedene Granularitäten gibt, ist es zu ermöglichen, unter anderem eine einzelne TCP-Verbindung, den gesamten Verkehr zwischen zwei Hosts oder den gesamten Datenverkehr zwischen zwei sicheren Routern zu schützen.

Ein etwas überraschender Aspekt bei IPsec ist, dass es verbindungsorientiert ist, obwohl es sich in der IP-Schicht befindet. So überraschend ist es wiederum nicht, weil zur Gewährleistung von Sicherheit auf alle Fälle ein Schlüssel für eine bestimmte Zeitspanne eingerichtet und verwendet werden muss – im Grunde eine Art von Verbindung, nur unter einem anderen Namen. Die Verbindungen amortisieren außerdem die Einrichtungskosten über viele Pakete. Eine „Verbindung“ im Zusammenhang mit IPsec wird als **Sicherheitsassoziation** (*security association*) bezeichnet. Eine Sicherheitsassoziation ist eine Simplexverbindung zwischen zwei Endpunkten,

mit der ein Sicherheitsidentifikator verknüpft ist. Ist in beiden Richtungen sicherer Datenverkehr erforderlich, so sind zwei Sicherheitsassoziationen erforderlich. Sicherheitsidentifikatoren werden in Paketen übertragen, die auf diesen sicheren Verbindungen reisen. Mit ihnen werden Schlüssel und andere relevante Informationen nachgeschlagen, wenn ein sicheres Paket ankommt.

Technisch gesehen besteht IPsec aus zwei Teilen. Der erste Teil beschreibt zwei neue Header, die zu Paketen hinzugefügt werden können, um den Sicherheitsidentifikator, die Daten zur Integritätskontrolle und andere Informationen zu übertragen. Der andere Teil namens **ISAKMP** (*Internet Security Association and Key Management Protocol*) bezieht sich auf das Einrichten von Schlüsseln. ISAKMP ist ein Framework. Das Hauptprotokoll, das die ganze Arbeit macht, ist **IKE** (*Internet Key Exchange*). Es sollte die Version 2 von IKE verwendet werden, die in RFC 4306 beschrieben wird, da die frühere Version sehr fehlerhaft war, wie Perlman und Kaufman aufzeigten (2000).

IPsec kann in einem von zwei Modi eingesetzt werden. Im **Transportmodus** wird der IPsec-Header direkt nach dem IP-Header eingefügt. Das Feld *Protocol* im IP-Header wird geändert um anzugeben, dass auf den normalen IP-Header ein IPsec-Header folgt (vor dem TCP-Header). Der IPsec-Header enthält Sicherheitsinformationen, vor allem den Sicherheitsidentifikator, eine neue Sequenznummer und möglicherweise eine Integritätsprüfung der Nutzdaten.

Im **Tunnelmodus** wird das gesamte IP-Paket, einschließlich Header und Daten, im Hauptteil eines neuen IP-Pakets mit einem vollständig neuen IP-Header gekapselt. Der Tunnelmodus ist nützlich, wenn der Tunnel an einem anderen Ort als dem Endziel endet. In einigen Fällen ist das Ende des Tunnels ein Sicherheits-Gateway, beispielsweise eine Firewall in einem Unternehmen. Dies ist in der Regel bei einem VPN (virtuelles privates Netzwerk) der Fall. In diesem Modus kapselt und entkapselt das Sicherheits-Gateway Pakete, wenn sie durch das Gateway gehen. Indem der Tunnel bei diesem sicheren Rechner endet, brauchen die Rechner im Unternehmens-LAN IPsec nicht zu kennen. Nur das Sicherheits-Gateway muss es kennen.

Der Tunnelmodus ist ebenfalls nützlich, wenn ein Bündel von TCP-Verbindungen zusammengefasst und als verschlüsselter Strom behandelt wird, da dies einen Eindringling davon abhält zu erkennen, wer wie viele Pakete an wen sendet. Es ist manchmal schon eine wertvolle Information, wenn man weiß, wie viele Daten in welche Richtung fließen. Wenn beispielsweise in einer Militärkrise der Datenverkehr zwischen dem Pentagon und dem Weißen Haus stark abfällt, aber der Datenverkehr zwischen dem Pentagon und einer Militärbasis tief in den Rocky Mountains von Colorado entsprechend zunimmt, kann ein Eindringling daraus wertvolle Schlüsse ziehen. Die Untersuchung der Flussmuster von Paketen, selbst wenn sie verschlüsselt sind, nennt man **Datenverkehrsanalyse** (*traffic analysis*). Der Tunnelmodus ist eine Möglichkeit, diese Analyse bis zu einem bestimmten Grad zu vereiteln. Der Nachteil des Tunnelmodus ist, dass hier ein zusätzlicher IP-Header hinzugefügt wird, was die Paketgröße erheblich erhöht. Demgegenüber wirkt sich der Transportmodus nicht sehr auf die Paketgröße aus.

Der erste neue Header ist der **Authentifizierungs-Header** (*authentication header*). Er bietet Integritätsprüfung und Sicherheit gegen Replay-Angriffe, aber keinen Datenschutz (also keine Verschlüsselung). Die Verwendung dieses Headers im Transportmodus ist in ►Abbildung 8.27 dargestellt. In IPv4 steht er zwischen dem IP-Header (einschließlich beliebiger Optionen) und dem TCP-Header. In IPv6 ist er nur ein zusätzlicher Erweiterungs-Header und wird auch als solcher behandelt. Das Format ist dem eines standardmäßigen IPv6-Erweiterungs-Headers sehr ähnlich. Die Nutzdaten müssen für den Authentifizierungsalgorithmus eventuell wie gezeigt auf eine bestimmte Länge aufgefüllt werden.

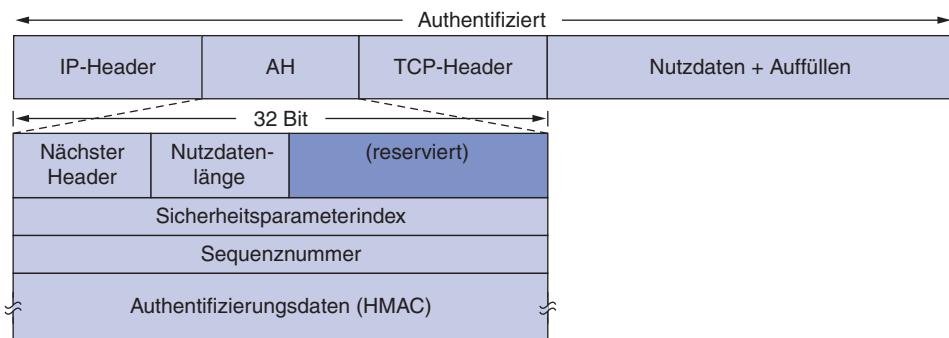


Abbildung 8.27: Der Authentifizierungs-Header von IPsec im Transportmodus für IPv4.

Untersuchen wir nun kurz den Authentifizierungs-Header. Im Feld *Next Header* wird der vorherige Wert des IP-Felds *Protocol* gespeichert, bevor er durch 51 ersetzt wurde, um anzugeben, dass ein Authentifizierungs-Header folgt. In den meisten Fällen steht hier der Code für TCP (6). Die *Payload Length* gibt die Anzahl der 32-Bit-Wörter im Authentifizierungs-Header minus 2.

Der *Security Parameters Index* ist der Verbindungsidentifikator. Er wird vom Sender eingefügt, um einen bestimmten Datensatz in der Datenbank des Empfängers anzugeben. Dieser Datensatz enthält den gemeinsamen Schlüssel, der bei dieser Verbindung verwendet wird, sowie andere Informationen über diese Verbindung. Wenn das Protokoll von der ITU anstatt der IETF erfunden worden wäre, würde dieses Feld den Namen *Virtual Circuit Number* tragen.

Das Feld *Sequence Number* dient zur Nummerierung der Pakete, die auf einer Sicherheitsassoziation gesendet wurden. Alle Pakete erhalten eine eindeutige Nummer, selbst erneute Übertragungen. Anders ausgedrückt, ein neu übertragenes Paket erhält eine andere Nummer als das Original (selbst wenn seine TCP-Sequenznummer die gleiche ist). Der Zweck dieses Feldes ist es, Replay-Angriffe zu entdecken. Diese Sequenznummern dürfen sich nicht wiederholen. Sind alle 2^{32} erschöpft, muss eine neue Sicherheitsassoziation erstellt werden, um die Kommunikation fortzusetzen.

Schließlich kommen wir noch zu *Authentication Data*, ein Feld variabler Länge, das die digitale Signatur der Nutzdaten enthält. Wird die Sicherheitsassoziation errichtet, verhandeln die beiden Seiten, welcher Signaturalgorithmus verwendet wird. In der

Regel wird hier keine Verschlüsselung mit öffentlichem Schlüssel verwendet, da die Pakete extrem schnell verarbeitet werden müssen und alle bekannten Algorithmen für öffentliche Schlüssel zu langsam sind. Da IPsec auf der symmetrischen Verschlüsselung basiert und Sender und Empfänger einen gemeinsamen Schlüssel aushandeln, bevor eine Sicherheitsassoziation errichtet wird, wird der gemeinsame Schlüssel bei der Berechnung der Signatur verwendet. Eine einfache Möglichkeit ist, den Hash über das Paket plus dem gemeinsamen Schlüssel zu berechnen. Der gemeinsame Schlüssel wird natürlich nicht übertragen. Ein solches Schema wird als **HMAC** (*Hashed Message Authentication Code*, Nachrichtenauthentifizierungscode auf Basis einer Hashfunktion) bezeichnet. Er kann viel schneller berechnet werden, als erst SHA-1 und dann auf dem Ergebnis RSA auszuführen.

Der Authentifizierungs-Header erlaubt keine Verschlüsselung der Daten, sodass er am nützlichsten ist, wenn eine Integritätsprüfung, aber kein Datenschutz benötigt wird. Eine bemerkenswerte Eigenschaft des Authentifizierungs-Headers ist, dass die Integritätsprüfung einige Felder im IP-Header beinhaltet, und zwar diejenigen, die sich nicht verändern, während die Pakete von Router zu Router gesendet werden. Das Feld *Time to Live* verändert sich bei jeder Teilstrecke, sodass es sich nicht für die Integritätsprüfung eignet. Es wird aber die IP-Quelladresse in die Prüfung mit aufgenommen, sodass ein Eindringling den Ursprung des Pakets unmöglich fälschen kann.

Der alternative IPsec-Header ist **ESP** (*Encapsulating Security Payload*, Kapselung der Sicherheitsnutzdaten). Er wird sowohl für den Transport- als auch den Tunnelmodus verwendet und ist in ▶ Abbildung 8.28 dargestellt.

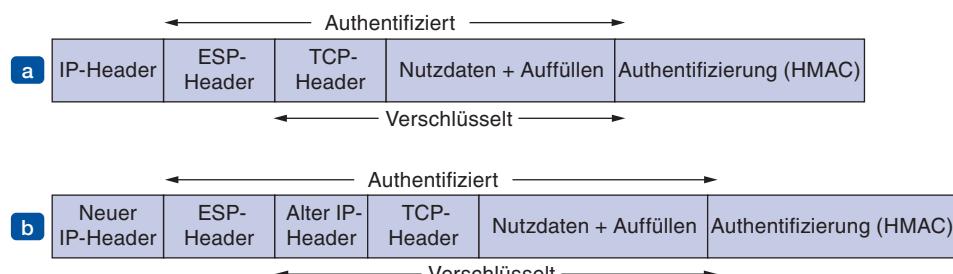


Abbildung 8.28: (a) ESP im Transportmodus. (b) ESP im Tunnelmodus.

Der ESP-Header besteht aus zwei 32-Bit-Wörtern. Es sind die vom Authentifizierungs-Header (AH) bekannten Felder *Security Parameters Index* und *Sequence Number*. Ein drittes Wort, das im Allgemeinen darauf folgt (aber technisch nicht zum Header gehört), ist der *Initialization Vector*, der für die Datenverschlüsselung verwendet wird. Wird die Null-Verschlüsselung eingesetzt, fällt dies weg.

ESP stellt wie AH auch HMAC-Integritätsprüfungen bereit, die aber, anstatt in den Header aufgenommen zu werden, nach den Nutzdaten folgen, wie in Abbildung 8.28 gezeigt. Wird HMAC ans Ende gesetzt, ist dies für die Hardware-Implementierung vorteilhaft. HMAC kann berechnet werden, während die Bits über die Netzwerkschnittstelle hinausgehen und am Ende angehängt werden. Deshalb haben Ethernet und

andere LANs ihre CRCs in einem Trailer anstatt im Header. Bei AH muss das Paket gepuffert und die Signatur berechnet werden, bevor das Paket gesendet werden kann, was die Anzahl der sendbaren Pakete/Sekunde eventuell reduziert.

In Anbetracht der Tatsache, dass ESP genauso viel und mehr als AH kann und effizienter zu booten ist, stellt sich die Frage: Warum braucht man AH überhaupt? Die Antwort hat überwiegend historische Gründe. Ursprünglich war AH nur für die Integrität zuständig und ESP für die Geheimhaltung der Daten. Später wurde ESP um die Integrität erweitert, aber die Entwickler von AH wollten es nach der ganzen Arbeit nicht einfach in der Versenkung verschwinden lassen. Ihr einziges echtes Argument ist, dass AH einen Teil des IP-Headers prüft, was ESP nicht macht. Doch abgesehen davon ist es ein schwaches Argument. Ein weiteres schwaches Argument ist, dass ein Produkt, das nur AH, aber nicht ESP unterstützt, weniger Schwierigkeiten hat, eine Exportlizenz zu erwerben, da es keine Verschlüsselung unterstützt. AH wird in Zukunft vermutlich verschwinden.

8.6.2 Firewalls

Die Möglichkeit, beliebige Rechner an beliebigen Stellen der Welt miteinander zu verbinden, hat Licht- und Schattenseiten. Für den Einzelnen ist es ein riesiger Spaß, zu Hause am PC im Internet zu surfen. Für die Sicherheitsbeauftragten in Unternehmen ist es ein Alptraum. Die meisten Unternehmen haben online umfangreiche vertrauliche Informationen gespeichert, seien es Betriebsgeheimnisse, Produktentwicklungspläne, Marketingstrategien, Finanzanalysen usw. Die Preisgabe solcher Informationen an einen Konkurrenten kann gravierende Folgen haben.

Neben dem Risiko, dass Daten in falsche Hände geraten können, besteht auch die Gefahr, dass unerwünschte Daten in das System eingeschleust werden. Insbesondere Viren, Würmer und anderes digitales Ungeziefer können Sicherheitsvorkehrungen durchbrechen, wertvolle Daten zerstören und dem Systemadministrator viel Zeit kosten, um die von ihnen hinterlassene Unordnung wieder zu richten. Meist werden die Rechner durch sorglose Mitarbeiter infiziert, die mal schnell ein witziges neues Spiel ausprobieren wollen.

Deshalb sind Mechanismen erforderlich, um die „guten“ Bits zu schützen und die „bösen“ auszusperren. Eine Methode ist IPsec. Mit diesem Ansatz können Daten bei der Übertragung zwischen sicheren Bereichen geschützt werden. Allerdings ist IPsec keine Hilfe, um digitale Schädlinge und Eindringlinge von Unternehmens-LANs fernzuhalten. Dies wird über Firewalls erreicht, die wir im Folgenden näher betrachten.

Firewalls sind nur die moderne Version einer mittelalterlichen Sicherheitsmaßnahme: ein tiefer Schutzgraben rund um die Burg. Jeder, der die Burg betreten oder verlassen wollte, musste über die Zugbrücke und wurde von der Wache am Eingangstor kontrolliert. Bei Netzen ist die gleiche Vorgehensweise möglich. Ein Unternehmen kann viele LANs zusammenschließen, jedoch fließt der gesamte Verkehr nach außen und nach innen durch eine elektronische Zugbrücke (Firewall), wie in ▶ Abbildung 8.29 ersichtlich ist. Es gibt keinen anderen Weg.

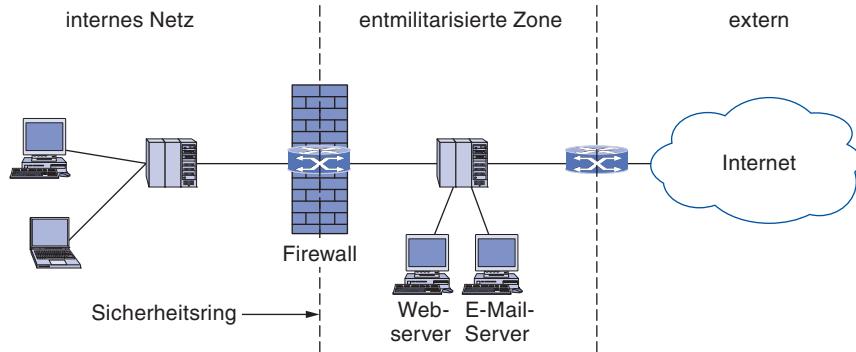


Abbildung 8.29: Eine Firewall, die ein internes Netz schützt.

Die Firewall fungiert als **Paketfilter**. Sie inspiziert jedes einzelne eingehende und ausgehende Paket. Pakete, die den Kriterien entsprechen, die der Netzadministrator in Regeln formuliert hat, werden normal weitergeleitet. Pakete, die den Test nicht bestehen, werden gnadenlos fallen gelassen.

Die Filterkriterien werden normalerweise als Regeln oder Tabellen formuliert. Diese listen die erlaubten und blockierten Quellen und Ziele auf und definieren die Standardregeln, die auf die Pakete anzuwenden sind, wenn sie an andere Rechner gesendet werden oder von anderen Rechnern kommen. Im gängigen Fall einer TCP/IP-Einstellung besteht eine Quelle oder ein Ziel aus einer IP-Adresse und einem Port. Ports geben an, welcher Dienst gewünscht wird. So wird TCP-Port 25 für Mail und TCP-Port 80 für HTTP verwendet. Einige Ports können einfach blockiert werden. Beispielsweise kann ein Unternehmen eingehende Pakete für alle IP-Adressen in Verbindung mit TCP-Port 79 blockieren. Für den Finger-Dienst war es einmal populär, die E-Mail-Adressen von Personen nachzuschlagen; dieser Dienst wird heute nicht mehr häufig genutzt.

Andere Ports lassen sich nicht so einfach blockieren. Die Schwierigkeit ist, dass Netzadministratoren nach Sicherheit streben, aber die Kommunikation nach außen nicht abtrennen können. Das wäre für die Sicherheit wahrscheinlich viel einfacher und besser, aber es gäbe Benutzerbeschwerden ohne Ende. Da kommt eine Anordnung wie die **entmilitarisierte Zone (DMZ, DeMilitarized Zone)** in Abbildung 8.29 gerade recht. Diese Zone ist der Teil des Unternehmensnetzes, der außerhalb des Sicherheitskreises liegt. Hier ist alles möglich. Wird ein Rechner wie beispielsweise ein Webserver in der entmilitarisierten Zone platziert, können Computer im Internet diesen Rechner kontaktieren, um die Unternehmenswebsite zu browsen. Jetzt kann die Firewall so konfiguriert werden, dass sie den eingehenden TCP-Verkehr auf Port 80 blockiert, sodass kein Computer im Internet über diesen Port die Rechner des internen Netzes attackieren kann. Damit eine Verwaltung des Webservers möglich ist, kann die Firewall eine Regel definiert haben, die Verbindungen zwischen internen Rechnern und dem Webserver zulässt.

Im Rüstungswettlauf mit den Angreifern sind die Firewalls mit der Zeit immer raffinierter geworden. Am Anfang wendeten die Firewalls eine Regel an, die unabhängig für jedes Paket gesetzt wurde. Es erwies sich jedoch als schwierig, Regeln zu schreiben, die nützliche Funktionalität erlauben, aber allen ungewollten Verkehr blockie-

ren. **Zustandsorientierte Firewalls** ordnen Pakete Verbindungen zu und verwenden TCP/IP-Headerfelder, um diese Verbindungen zu verfolgen. Dadurch ist es möglich, Regeln zu erstellen, die zum Beispiel einem externen Webserver erlauben, Pakete an einen internen Host zu senden, aber nur, wenn der interne Host zuerst eine Verbindung mit dem externen Webserver hergestellt hat. Eine solche Regel ist mit zustandslosen Entwürfen nicht möglich, die alle Pakete von dem externen Server entweder durchlassen oder verwerfen müssen.

Ein weiterer Grad an Verbesserung gegenüber der zustandslosen Verarbeitung wurde erreicht, als die Firewalls **Anwendungs-Gateways** implementierten. Hierbei schaut die Firewall direkt in die Pakete und gibt sich nicht mit den TCP-Headern zufrieden, um festzustellen, was die Anwendung macht. Mit dieser Fähigkeit ist es möglich, HTTP-Verkehr zum Webbrowsen von HTTP-Verkehr für P2P-Dateiaustausch zu unterscheiden. Administratoren können Regeln schreiben, um Unternehmen vor P2P-Dateiaustausch zu verschonen und gleichzeitig das Browsen im Web zuzulassen, das für das Geschäft absolut wichtig ist. Bei all diesen Methoden ist es möglich, nicht nur den eingehenden Verkehr, sondern auch den ausgehenden Verkehr zu kontrollieren, um zu verhindern, dass sensible Dokumente nach außen geschickt werden.

Aus der obigen Diskussion dürfte klar sein, dass Firewalls die Standardschichtung der Protokolle verletzen. Sie sind in der Vermittlungsschicht angesiedelt, linsen aber zum Filtern in die Transport- und die Anwendungsschicht. Das macht sie sehr anfällig. Beispielsweise neigen Firewalls dazu, sich bei der Portnummerierung auf die Standardkonventionen zu verlassen, um zu entscheiden, welche Art von Verkehr in einem Paket übertragen wird. Oft werden Standardports verwendet, aber nicht von allen Rechnern und auch nicht von allen Anwendungen. Einige P2P-Anwendungen wählen die Ports dynamisch, um zu vermeiden, erkannt (und blockiert) zu werden. Eine Verschlüsselung mit IPsec oder anderen Schemata verbirgt Informationen auf höherer Ebene vor der Firewall. Und schließlich kann eine Firewall den Computern, die darüber kommunizieren, nicht einfach mitteilen, welche Richtlinien angewendet werden und warum ihre Verbindung abgebrochen wird. Sie muss so tun, als handele es sich um ein Kabelbruch. Aus all diesen Gründen betrachten Netzwerk-Puristen Firewalls als ein Makel in der Architektur des Internets. Da jedoch das Internet für einen Computer ein gefährlicher Ort ist, sind Firewalls durchaus eine Hilfe und werden uns mit Sicherheit auch weiterhin gute Dienste leisten.

Selbst wenn die Firewall perfekt konfiguriert ist, gibt es noch eine Fülle von Sicherheitsproblemen. Wenn eine Firewall beispielsweise so konfiguriert ist, dass nur Pakete von speziellen Netzen eingehen dürfen (z.B. von den anderen Zweigstellen des Unternehmens), kann ein Eindringling von außerhalb der Firewall falsche Quelladressen angeben, um diese Prüfung zu umgehen. Möchte ein Insider geheime Dokumente nach außen versenden, kann er sie verschlüsseln oder sogar fotografieren und die Fotos als JPEG-Dateien senden, was jeden E-Mail-Filter umgeht. Und wir haben noch gar nicht die Tatsache erwähnt, dass auch wenn drei Viertel aller Angriffe von außen kommt, es die Attacken von innen sind, beispielsweise von verärgerten Mitarbeitern, die normalerweise viel größeren Schaden anrichten (Verizon, 2009).

Ein weiteres Problem von Firewalls ist es, dass sie nur einen einzigen Verteidigungsring bieten. Wird dieser Ring durchbrochen, ist alles möglich. Aus diesem Grund werden Firewalls oft auf mehreren Ebenen zur Verteidigung eingesetzt. So könnte beispielsweise eine Firewall den Eingang zu dem internen Netz überwachen und jeder Rechner in seiner eigenen Firewall ausgeführt werden. Leser, die glauben, dass eine Sicherheitsmaßnahme ausreicht, haben offensichtlich in letzter Zeit an keinen Linienflug auf einer internationalen Strecke teilgenommen.

Darüber hinaus gibt es eine ganze andere Angriffsklasse, gegen die Firewalls keinen Schutz bieten. Die Grundidee einer Firewall ist, Eindringlinge davon abzuhalten, hineinzukommen und geheime Daten daran zu hindern, hinauszugelangen. Leider gibt es Leute, die nichts Besseres zu tun haben, als zu versuchen, bestimmte Sites zum Absturz zu bringen. Hierzu senden sie legitime Pakete in großen Mengen an das Ziel, bis dieses unter der Last zusammenbricht. Um beispielsweise eine Website stillzulegen, kann ein Eindringling ein SYN-Paket mit TCP senden, um eine Verbindung aufzubauen. Die Site weist der Verbindung dann einen Tabellenplatz zu und sendet ein SYN + ACK-Paket als Antwort. Wenn der Eindringling nicht antwortet, bleibt der Tabellenplatz eine gewisse Zeit erhalten, bis ein Timeout auftritt. Sendet der Eindringling Tausende von Verbindungsanfragen, dann wird der gesamte Tabellenplatz gefüllt und keine legitimen Verbindungen können mehr durchgehen. Angriffe, bei denen der Eindringling beabsichtigt, das Ziel zum Absturz zu bringen, anstatt Daten zu stehlen, werden als **DoS**-Angriffe (*Denial of Service, Dienstverweigerung*) bezeichnet. In der Regel haben diese Pakete falsche Quelladressen, sodass die Eindringlinge nicht so leicht aufgespürt werden können. DoS-Angriffe auf größere Websites kommen im Internet recht häufig vor.

Eine noch üblere Variante ist die, bei der der Eindringling bereits Hunderte von Rechnern weltweit aufgebrochen hat und dann alle anweist, gleichzeitig dasselbe Ziel anzugreifen. Dies erhöht nicht nur die Schlagkraft des Eindringlings, sondern reduziert auch die Chance, dass er aufgespürt wird, da die Pakete von vielen Rechnern stammen, die unverdächtigen Benutzern gehören. Solch ein Angriff wird als **DDoS**-Angriff (*Distributed Denial of Service*) bezeichnet. Dieser Angriff ist sehr schwer abzuwehren. Selbst wenn der angegriffene Rechner eine falsche Anfrage schnell erkennen kann, benötigt er eine gewisse Zeit, um diese zu verarbeiten und zu verwerfen. Wenn ausreichend Anfragen pro Sekunde eingehen, ist die CPU die ganze Zeit mit deren Behandlung beschäftigt.

8.6.3 Virtuelle private Netze

Viele Unternehmen haben Niederlassungen und Produktionsstätten in verschiedenen Städten, manchmal sogar in verschiedenen Ländern. In früheren Zeiten, als es noch keine öffentlichen Datennetze gab, mieteten diese Unternehmen von der Telefongesellschaft Standleitungen zwischen einigen oder allen Standortpaaren. Manche Unternehmen verfolgen diese Strategie noch heute. Ein Netzwerk, das aus Unternehmensrechnern und Standleitungen besteht, wird als **privates Netz** bezeichnet.

Private Netze funktionieren prima und sind sehr sicher. Wenn nur Standleitungen verfügbar sind, kann kein Datenverkehr das Unternehmen verlassen, weil es keine undichten Stellen gibt. Eindringlinge müssten die Leitungen physisch anzapfen, was nicht so einfach ist. Das Problem bei privaten Netzen ist, dass die Miete einer T1-Leitung mehrere Tausend Euro im Monat kostet und T3-Leitungen noch viel teurer sind. Als öffentliche Datennetze und später das Internet aufkamen, wollten viele Unternehmen ihren Datenverkehr (und eventuell auch den Sprachverkehr) auf ein öffentliches Netz übertragen, ohne dabei auf die Sicherheit von privaten Netzen zu verzichten.

Diese Forderung führte bald zur Erfindung von **virtuellen privaten Netzen (VPNs)**, *(Virtual Private Network)*, bei denen es sich um sogenannte Overlay-Netze handelt, die auf den öffentlichen Netzen aufsetzen, aber die meisten Eigenschaften von privaten Netzen aufweisen. Sie werden als „virtuell“ bezeichnet, weil sie nur in der Vorstellung vorhanden sind, so wie virtuelle Verbindungen keine wirklichen Leitungen sind und virtueller Speicher kein echter Speicherchip.

Beliebt ist der Ansatz, die VPNs direkt auf dem Internet aufzusetzen. Ein gängiger Entwurf ist, jedes Büro mit einer Firewall auszustatten und Tunnel durch das Internet zwischen allen Niederlassungen aufzubauen, wie in ► Abbildung 8.30a dargestellt. Ein weiterer Vorteil, das Internet zur Konnektivität zu verwenden, ist, dass die Tunnel bei Bedarf so eingerichtet werden können, dass der Rechner eines Angestellten, der zu Hause arbeitet oder beruflich unterwegs ist, mitberücksichtigt wird, sofern diese Person einen Internetanschluss hat. Diese Flexibilität ist viel größer als bei Standleitungen, doch aus der Perspektive der Rechner im VPN sieht die Topologie genauso aus wie beim privaten Netz (► Abbildung 8.30b). Wenn das System hochgefahren wird, muss jedes Firewall-Paar die Parameter seiner Sicherheitsassoziation verhandeln, einschließlich der Dienste, Modi, Algorithmen und Schlüssel. Wird IPsec zum Tunneling verwendet, so kann der gesamte Datenverkehr zwischen zwei Niederlassungen in einer authentifizierten, verschlüsselten Sicherheitsassoziation zusammengefasst und somit Integritätskontrolle, Geheimhaltung und sogar eine bedeutende Immunität gegenüber Datenverkehrsanalysen bereitgestellt werden. Viele Firewalls haben VPN-Funktionen integriert, aber diese können auch von ganz normalen Routern übernommen werden. Da Firewalls vor allem der Sicherheit dienen, ist es normal, dass die Tunnel an den Firewalls beginnen und enden, was eine klare Trennung zwischen Unternehmen und Internet herstellt. So sind Firewalls, VPNs und IPsec mit ESP im Tunnelmodus eine natürliche Kombination, die in der Praxis weit verbreitet ist.

Wurde die Sicherheitsassoziation errichtet, dann kann der Verkehr zu fließen beginnen. Für einen Router im Internet ist ein Paket, das in einem VPN-Tunnel reist, nur ein einfaches Paket. Das einzige Ungewöhnliche daran ist das Vorhandensein des IPsec-Headers nach dem IP-Header. Da aber diese Extra-Header sich nicht auf den Weiterleitungsprozess auswirken, kümmern sich Router nicht darum.

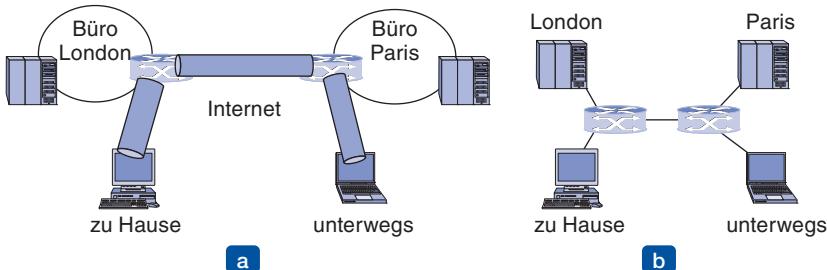


Abbildung 8.30: (a) Ein virtuelles privates Netz. (b) Topologie aus der Sicht von innen.

Ein anderer Ansatz, der immer populärer wird, besteht darin, den ISP das VPN einzurichten zu lassen. Unter Verwendung von MPLS (beschrieben in *Kapitel 5*) können die Pfade für den VPN-Verkehr über das ISP-Netz zwischen den Zweigniederlassungen eingerichtet werden. Diese Pfade halten den VPN-Verkehr von dem anderen Internetverkehr getrennt und erhalten garantiert eine bestimmte Bandbreite oder andere Dienstqualität.

Ein bedeutender Vorteil, ein VPN so aufzubauen, ist, dass es für die gesamte Benutzersoftware völlig transparent ist. Die Firewalls errichten und verwalten die Sicherheitsassoziationen. Die einzige Person, die dieses Setup kennt, ist der Systemadministrator, der die Sicherheits-Gateways konfigurieren und verwalten muss, oder der ISP-Administrator, der die MPLS-Pfade konfigurieren muss. Für jeden anderen ist es wieder wie ein privates Netzwerk mit Standleitungen. Weitere Informationen über VPNs finden Sie in Lewis (2006).

8.6.4 Drahtlose Sicherheit

Es ist erstaunlich einfach, ein System auszulegen, das dank VPNs und Firewalls logisch völlig sicher ist, aber praktisch leck ist wie ein Sieb. Diese Situation kann eintreten, wenn einige der Rechner drahtlos sind und über Funk laufen, da hier die Firewall in beiden Richtungen „überflogen“ wird. Die Reichweite von IEEE-802.11-Netzen beträgt oftmals ein paar Hundert Meter, sodass jeder, der ein Unternehmen ausspionieren möchte, sich einfach am Morgen auf den Angestelltenparkplatz stellen kann, mit einem Notebook mit IEEE-802.11-Unterstützung im Wagen, das alle herumschwirrenden Informationen aufzeichnet. Am späten Nachmittag ist die Festplatte voller wertvoller Nachrichten. Theoretisch sollte diese undichte Stelle nicht auftreten. Theoretisch sollte aber auch niemand eine Bank ausrauben.

Die Ursache für dieses Sicherheitsproblem liegt größtenteils bei den Herstellern, die versuchen, ihre drahtlosen Basisstationen (Zugangspunkte) benutzerfreundlich zu gestalten. In der Regel nimmt ein Benutzer das Gerät aus dem Karton und steckt den Stecker in die Steckdose. Das Gerät funktioniert sofort – meistens fast ohne jegliche Sicherheit, sodass vertrauliche Informationen für jeden im Funkbereich zugänglich sind. Wenn es dann an ein Ethernet angeschlossen wird, erscheint auch noch der gesamte Datenverkehr im Ethernet auf dem Parkplatz. Drahtlosigkeit ist der Traum

eines jeden Schnüfflers: freie Daten, ohne dass man dafür etwas tun muss. Daher ist es natürlich logisch, dass die Sicherheit bei drahtlosen Systemen noch viel wichtiger ist als bei kabelbasierten. In diesem Abschnitt werden einige Möglichkeiten betrachtet, wie die Sicherheit in Funknetzen erhöht werden kann. Weitere Informationen finden Sie in Nichols und Lekkas (2002).

IEEE-802.11-Sicherheit

Ein Teil des IEEE-802.11-Standards, der ursprünglich **IEEE 802.11i** hieß, legt ein Sicherheitsprotokoll auf der Sicherungsschicht fest, das einen drahtlosen Knoten daran hindert, Nachrichten zwischen einem anderen drahtlosen Knotenpaar zu lesen oder zu beeinträchtigen. Im Handel ist es auch unter dem Namen **WPA2** (*WiFi Protected Access 2*) bekannt. Einfaches WPA ist eine Interimslösung, die nur einen Teil von IEEE 802.11i implementiert. Es sollte zugunsten von WPA2 vermieden werden.

Bevor wir näher auf IEEE 802.11i eingehen, wollen wir darauf hinweisen, dass es ein Ersatz ist für **WEP** (*Wired Equivalent Privacy*, Datenschutz-Äquivalent zu kabelgebundenen Übertragungen), der ersten Generation von IEEE-802.11-Sicherheitsprotokollen. WEP wurde von einem Netznormenausschuss entwickelt – eine Vorgehensweise, die sich grundlegend von der unterschied, die beispielsweise das NIST beim Entwurf von AES verfolgte. Die Ergebnisse waren verheerend. Was war falsch an WEP? Aus Sicht der Sicherheit so ziemlich alles, wie sich nach und nach zeigte. Zum Beispiel wurden bei WEP geheime Daten verschlüsselt, indem sie mit der Ausgabe einer Stromchiffre XOR-verknüpft wurden. Leider führte eine schwache Schlüsselstromerzeugung dazu, dass die Ausgabe oft wiederverwendet wurde. Dies wiederum ermöglichte es, das System auf einfache Weise auszuhebeln. Außerdem basierte die Integritätsprüfung auf einem 32-Bit-CRC-Wert, um ein weiteres Beispiel zu nennen. Hierbei handelt es sich um einen effizienten Code zur Erkennung von Übertragungsfehlern, der kryptografisch nicht stark genug ist, um Angreifer abzuwehren.

Diese und andere Entwurfsfehler machten es sehr leicht, WEP zu knacken. Den ersten praktischen Beweis lieferte Adam Stubblefield während seiner Praktikantenzeit bei AT&T (Stubblefield et al., 2002). Es gelang ihm in nur einer Woche, einen von Fluhrer et al. (2001) grob umrissenen Angriff zu codieren und zu testen, wobei er einen Großteil der Zeit damit verbrachte, seine Vorgesetzten zu überzeugen, ihm für seine Versuche eine WiFi-Karte zu genehmigen. Derzeit gibt es kostenlose Software, die WEP-Kennwörter in nur einer Minute knackt, sodass von der Verwendung von WEP dringend abgeraten wird. WEP schützt zwar vor gelegentlichem Zugriff, bietet jedoch keine wirkliche Sicherheit in irgendeiner Form. Als klar war, dass WEP leicht zu brechen war, wurde auf die Schnelle die IEEE-802.11i-Gruppe zusammengestellt, die im Juni 2004 einen formalen Standard vorlegte.

Im Folgenden wollen wir den Standard IEEE-802.11i beschreiben, der echte Sicherheit bietet, wenn er ordnungsgemäß eingerichtet und verwendet wird. Es gibt zwei geläufige Szenarien, in denen WPA2 Verwendung findet. Im ersten Szenario hat ein Unternehmen einen separaten Authentifizierungsserver einschließlich einer Benutzernamen- und Kennwortdatenbank eingerichtet, mit deren Hilfe festgelegt wird, ob

ein drahtloser Client auf das Netz zugreifen darf. In diesem Fall verwenden Clients Standardprotokolle, um sich gegenüber dem Netz zu authentifizieren. Die beiden wichtigsten Protokolle sind **IEEE 802.1X**, mit dem der Zugangspunkt dem Client erlaubt, in Dialog mit dem Authentifizierungsserver zu treten, und das Ergebnis beobachtet, sowie **EAP** (*Extensible Authentication Protocol*; RFC 3748), das verrät, wie der Client und der Authentifizierungsserver interagieren. Eigentlich ist EAP ein Framework und andere Standards definieren die Protokollnachrichten. Wir wollen uns an dieser Stelle jedoch nicht eingehender mit den vielen Details dieses Austausches befassen, da wir im Rahmen dieses Buches nur einen Überblick bieten können.

Das zweite Szenario sind Rechner im privaten Umfeld, wo es keinen Authentifizierungsserver gibt. Stattdessen gibt es nur ein gemeinsames Kennwort, mit dem die Clients auf das Drahtlosnetz zugreifen. Dieses Szenario ist weniger komplex als das mit einem Authentifizierungsserver, weshalb es bevorzugt zu Hause und in kleineren Betrieben verwendet wird. Leider ist es aber auch weniger sicher. Der Hauptunterschied ist, dass bei einem Authentifizierungsserver jeder Client einen Schlüssel zur Datenverschlüsselung erhält, der den anderen Clients nicht bekannt ist. Bei einem gemeinsamen Kennwort wird für jeden Client ein eigener Schlüssel abgeleitet, aber alle Clients haben das gleiche Kennwort und können die Schlüssel der anderen ableiten, wenn sie wollen.

Die Schlüssel, die verwendet werden, um den Verkehr zu verschlüsseln, werden als Teil eines Authentifizierungs-Handshakes berechnet. Der Handshake erfolgt direkt, nachdem der Client eine Verbindung zu einem Drahtlosnetz hergestellt und sich beim Authentifizierungsserver, sofern vorhanden, ausgewiesen hat. Zu Beginn des Handshakes hat der Client entweder das gemeinsame NetzKennwort oder sein Kennwort für den Authentifizierungsserver. Dieses Kennwort wird zur Ableitung eines Masterschlüssels verwendet. Allerdings wird der Masterschlüssel nicht direkt zum Verschlüsseln der Pakete eingesetzt. Es ist in der Kryptografie üblich, für jeden Nutzungszeitraum einen neuen Sitzungsschlüssel abzuleiten, den Schlüssel für verschiedene Sitzungen zu ändern und den Masterschlüssel so wenig wie möglich zu exponieren. Und genau dieser Sitzungsschlüssel wird im Handshake berechnet.

Der Sitzungsschlüssel wird mit dem 4-Pakete-Handshake berechnet (►Abbildung 8.31). Zuerst sendet der Zugangspunkt eine Zufallszahl zur Identifizierung. Zufallszahlen, die nur einmal in Sicherheitsprotokollen wie diesem verwendet werden, werden **Nonces** genannt. Dies ist eine Zusammenfassung der englischen Bezeichnung „number used once“ (Nummer nur einmal verwendet). Der Client wählt außerdem seine eigene Nonce. Er verwendet die Nonces, seine MAC-Adresse und die des Zugangspunkts sowie den Masterschlüssel zur Berechnung eines Sitzungsschlüssels K_S . Der Sitzungsschlüssel wird in mehrere Teile aufgespalten, die jeweils für einen anderen Zweck eingesetzt werden; dieses Detail haben wir ausgelassen. Jetzt verfügt der Client über Sitzungsschlüssel, aber der Zugangspunkt nicht. Deshalb sendet der Client seine Nonce an den Zugangspunkt und dieser führt die gleiche Berechnung aus, um die gleichen Sitzungsschlüssel abzuleiten. Die Nonces können offen versendet werden, da sich davon die Schlüssel nicht ableiten lassen – es sei denn, man verfügt über zusätzliche geheime Informationen. Die Nachricht vom Client wird mit einer

Integritätsprüfung namens **MIC** (*Message Integrity Check*) auf der Basis eines Sitzungsschlüssels geschützt. Der Zugangspunkt kann prüfen, ob die MIC korrekt ist, und damit steht fest, dass die Nachricht tatsächlich vom Client gekommen sein muss, nachdem er die Sitzungsschlüssel berechnet hat. MIC ist nur ein anderer Name für einen Nachrichtenauthentifizierungscode (*Message Authentication Code*), wie in einem HMAC. Aufgrund der Verwechslungsgefahr mit MAC-Adressen (*Medium Access Control*) wird bei Netzprotokollen oft der Begriff MIC bevorzugt.

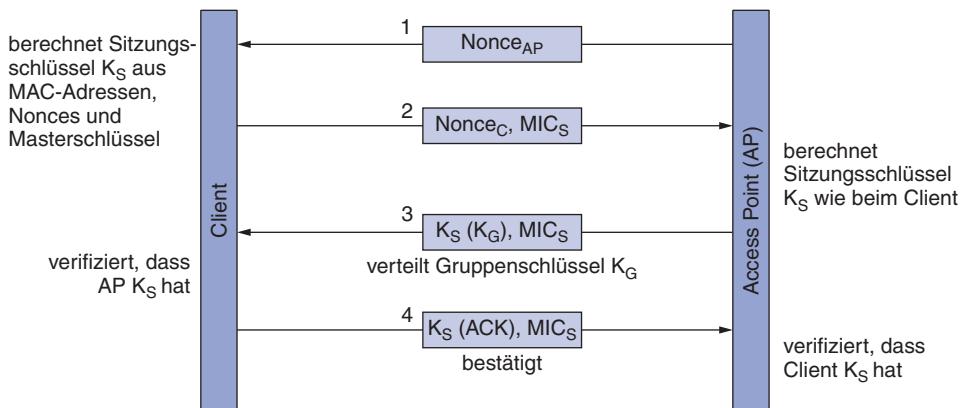


Abbildung 8.31: Der Handshake zum Einrichten des IEEE-802.11i-Schlüssels.

In den letzten zwei Nachrichten verteilt der Zugangspunkt einen Gruppenschlüssel K_G an den Client und der Client bestätigt die Nachricht. Über den Empfang dieser Nachrichten kann der Client verifizieren, dass der Zugangspunkt die korrekten Sitzungsschlüssel hat, und umgekehrt. Der Gruppenschlüssel wird für den Funk- und Multicastverkehr auf dem IEEE-802.11-LAN verwendet. Da aufgrund des Handshakes jeder Client über eigene Verschlüsselungsschlüssel verfügt, kann der Zugangspunkt keinen dieser Schlüssel verwenden, um Pakete an alle drahtlosen Clients zu senden; folglich müsste eine separate Kopie an jeden Client unter Verwendung seines Schlüssels gesendet werden. Stattdessen wird ein gemeinsamer Schlüssel verteilt, sodass der Funkverkehr nur einmal gesendet und trotzdem von allen Clients empfangen wird. Er muss aktualisiert werden, wenn Clients das Netz verlassen oder ihm beitreten.

Zum Schluss kommen wir zu dem Teil, dass die Schlüssel tatsächlich zur Gewährleistung von Sicherheit verwendet werden. Es gibt zwei Protokolle in IEEE 802.11i, die Vertraulichkeit, Integrität und Authentifizierung einer Nachricht erlauben. Wie WPA war eines der Protokolle, mit Namen **TKIP** (*Temporary Key Integrity Protocol*), nur eine Zwischenlösung. Es wurde entworfen, um die Sicherheit auf alten und langsamen IEEE-802.11-Karten zu verbessern, sodass zumindest ein gewisser Grad an Sicherheit, der besser war als bei WEP, als Firmware-Upgrade bereitgestellt werden konnte. Doch auch TKIP ist inzwischen gebrochen worden, sodass Sie mit dem anderen empfohlenen Protokoll, **CCMP**, besser dran sind. Wofür steht CCMP? Es ist die Abkürzung für *Counter mode with Cipher block chaining Message authentication code Protocol*. Wir nennen es hier einfach CCMP. Sie können es nennen wie Sie wollen.

CCMP funktioniert relativ einfach. Es basiert auf einer AES-Verschlüsselung mit 128-Bit-Schlüssel und -Blockgröße. Den Schlüssel erhält man über den Sitzungsschlüssel. Für die Vertraulichkeit werden die Nachrichten mit AES im Counter-Modus verschlüsselt. (Zur Erinnerung: Wir haben die Chiffriermode in Abschnitt 8.2.3 besprochen.) Diese Modi verhindern, dass die gleiche Nachricht jedes Mal mit den gleichen Bits verschlüsselt wird. Der Counter-Modus kombiniert die Verschlüsselungsdaten mit einem Zähler. Für die Integrität wird die Nachricht einschließlich der Header-Felder mit dem Cipher-Block-Chaining-Modus verschlüsselt und der letzte 128-Bit-Block wird als MIC behalten. Anschließend werden die Nachricht (im Counter-Modus verschlüsselt) und die MIC gesendet. Der Client und der Zugangspunkt können beide diese Verschlüsselung ausführen beziehungsweise sie verifizieren, wenn ein drahtloses Paket empfangen wird. Bei Funk- oder Multicast-Nachrichten wird die gleiche Prozedur mit dem Gruppenschlüssel verwendet.

Sicherheit in Bluetooth

Bluetooth hat eine erheblich geringere Reichweite als IEEE 802.11, sodass es nicht vom Parkplatz aus angegriffen werden kann. Dennoch ist Sicherheit immer noch ein wichtiges Thema. Stellen Sie sich vor, dass der Rechner von Alice mit einer drahtlosen Bluetooth-Tastatur ausgestattet ist. Wenn keine Sicherheitsvorkehrungen getroffen wurden und Trudy sich zufällig im Büro nebenan befindet, könnte sie alles lesen, was Alice eintippt, einschließlich der gesamten ausgehenden E-Mail. Sie könnte auch alle Informationen von Alices Rechner abfangen, die an einen Bluetooth-Drucker gesendet werden, der neben ihr steht (wie eingehende E-Mail und vertrauliche Berichte). Zum Glück besitzt Bluetooth ein ausgefeiltes Sicherheitsschema, um die Angriffe der Trudys dieser Welt abzuwehren. Wir fassen die wichtigsten Funktionen im Folgenden zusammen.

Bluetooth Version 2.1 und höher hat vier Sicherheitsmodi, die von keiner Sicherheit bis zur vollständigen Datenverschlüsselung und Integritätskontrolle reichen. Wie bei IEEE 802.11 besteht bei deaktivierter Sicherheit (die Voreinstellung) keine Sicherheit. Die meisten Benutzer aktivieren die Sicherheit erst, nachdem bereits ein ernsthafter Sicherheitsbruch aufgetreten ist. Auf gut Deutsch heißt das, erst zu handeln, wenn das Kind bereits in den Brunnen gefallen ist.

Bluetooth unterstützt Sicherheit in mehreren Schichten. Auf der Bitübertragungsschicht bietet das Frequenzsprungverfahren etwas Sicherheit, aber da einem Bluetooth-Gerät, das in ein Piconetz eintritt, die Frequenzsprungfolge mitgeteilt werden muss, ist diese Folge offensichtlich kein Geheimnis. Die echte Sicherheit beginnt, wenn ein neu hinzugekommener Slave den Master um einen Kanal bittet. Vor Bluetooth 2.1 wurde davon ausgegangen, dass zwei Geräte einen geheimen Schlüssel teilen, der vorab eingerichtet wurde. In einigen Fällen werden beide vom Hersteller fest vorgegeben (wie bei einem Headset und Mobiltelefon, die als Einheit verkauft werden). In anderen Fällen hat ein Gerät (wie das Headset) einen fest vorgegebenen Schlüssel, und der Benutzer muss den Schlüssel in ein anderes Gerät (wie das Mobiltelefon) als Dezimalzahl eingeben. Diese gemeinsamen Schlüssel werden auch als **Passkeys** (Hauptschlüssel) bezeichnet. Leider wird für diese Schlüssel oft „1234“ oder

ein ähnlich vorhersagbarer Wert vorgegeben, der auf alle Fälle aus vier Ziffern besteht, was genau 10^4 Möglichkeiten zulässt. Bei einer einfachen sicheren Verbindung (Pairing) in Bluetooth 2.1 wählen die Geräte einen Code aus einem sechsstelligen Bereich, sodass der Passkey weniger leicht vorhersagbar, wenn auch längst nicht sicher ist.

Um einen Kanal einzurichten, prüfen Master und Slave, ob der jeweils andere den Passkey kennt. Wenn ja, verhandeln sie, ob der Kanal mit einer Verschlüsselung, mit einer Integritätskontrolle oder mit beidem versehen sein soll. Dann wählen sie einen zufälligen 128-Bit-Sitzungsschlüssel aus, von dem einige Bits öffentlich sein können. Die Abschwächung dieses Schlüssels ist auf die Erfüllung von staatlichen Beschränkungen in verschiedenen Ländern zurückzuführen, die darauf abzielen, den Export oder die Verwendung von Schlüsseln zu verhindern, die länger sind, als sie die Regierung aufbrechen kann.

Die Verschlüsselung verwendet eine Stromchiffre namens E_0 , die Integritätskontrolle verwendet **SAFER+**. Beides sind traditionelle symmetrische Blockchiffren. SAFER+ wurde auch im AES-Wettbewerb eingereicht, schied aber bereits in der ersten Runde aus, weil es langsamer war als andere Kandidaten. Bluetooth wurde abgeschlossen, bevor die AES-Chiffre ausgewählt wurde. Ansonsten hätte es höchstwahrscheinlich Rijndael verwendet.

Die eigentliche Verschlüsselung verwendet die Stromchiffre in Abbildung 8.14, wobei der Klartext mit dem Schlüsselstrom über XOR verknüpft wird, um den Chiffretext zu erzeugen. Leider kann E_0 (wie RC4) einige fatale Schwächen aufweisen (Jakobsson und Wetzel, 2001). Selbst wenn diese Chiffre noch nicht bei der Verfassung dieses Buches aufgebrochen sein sollte, so geben doch die Ähnlichkeiten mit der A5/1-Chiffre, deren spektakuläre Fehler den gesamten GSM-Telefondatenverkehr gefährden, Anlass zu Besorgnis (Biryukov et al., 2000). Es verblüfft viele Leute (wie auch die Autoren dieses Buchs), dass in dem fortwährenden Katz-und-Maus-Spiel der Kryptografen und Kryptoanalytiker die Kryptoanalytiker oftmals die Gewinner sind.

Ein weiteres Sicherheitsproblem ist, dass Bluetooth nur Geräte, aber keine Benutzer authentifiziert, sodass der Diebstahl eines Bluetooth-Gerätes dem Dieb unter Umständen Zugriff auf die Bankkonten usw. eines Benutzers geben kann. Bluetooth implementiert aber auch Sicherheitsfunktionen in den oberen Schichten, sodass selbst im Fall des Aufbruchs der Sicherheit in der Sicherungsschicht die Sicherheit teilweise erhalten bleibt, vor allem bei Anwendungen, die einen PIN-Code erfordern, der manuell über eine Art Tastatur eingegeben werden muss, um die Transaktion abzuschließen.

8.7 Authentifizierungsprotokolle

Authentifizierung (Echtheitsprüfung) ist die Technik, durch die ein Prozess prüft, ob die Kommunikationspartner die sind, die sie sein sollten. Die Überprüfung der Identität eines entfernten Prozesses angesichts eines böswilligen aktiven Eindringlings ist erstaunlich schwierig und erfordert komplexe Protokolle auf der Grundlage der Kryptografie. In diesem Abschnitt werden einige der zahlreichen Authentifizierungsprotokolle beschrieben, die in unsicheren Rechnernetzen verwendet werden.

Einige Leute verwechseln Autorisation mit Authentifizierung oder setzen beides gleich. *Authentifizierung* hat mit der Frage zu tun, ob man wirklich mit einem spezifischen Prozess kommuniziert. *Autorisation* betrifft die Dinge, zu deren Ausführung ein Prozess berechtigt ist. Ein Client-Prozess nimmt beispielsweise Verbindung mit einem Dateiserver auf und sagt: „Ich bin Scotts Prozess und möchte die Datei *cookbook.old* löschen.“ Aus Sicht des Dateiservers müssen zwei Fragen beantwortet werden:

- 1.** Handelt es sich wirklich um einen Prozess von Scott (Authentifizierung)?
- 2.** Ist Scott berechtigt, die Datei *cookbook.old* zu löschen (Autorisation)?

Erst wenn beide Fragen zufriedenstellend geklärt wurden, kann die angeforderte Aktion ausgeführt werden. Die erste Frage ist die wichtigere. Weiß ein Dateiserver, mit wem er es zu tun hat, kann er die entsprechenden Berechtigungen in einer lokalen Tabelle oder Datenbank nachschlagen. Aus diesem Grund konzentrieren wir uns auf Authentifizierung.

Alle Authentifizierungsprotokolle basieren auf dem folgenden allgemeinen Modell: Alice beginnt mit dem Versenden einer Nachricht an Bob oder an ein vertrauenswürdiges **Schlüsselverteilungszentrum** (*Key Distribution Center, KDC*), bei dem Ehrlichkeit vorausgesetzt wird. In verschiedenen Richtungen werden mehrere weitere Nachrichten ausgetauscht. Ein Eindringling mit schlechten Absichten namens Trudy kann sich in diese Nachrichten einklinken, sie ändern oder erneut abspielen, um Alice und Bob zu hintergehen oder einfach nur um zu stören.

Dennoch kann Alice nach Beendigung des Protokolls sicher sein, dass sie mit Bob kommuniziert und umgekehrt. Des Weiteren richten die beiden bei den meisten Protokollen auch einen geheimen **Sitzungsschlüssel** (*session key*) für die bevorstehende Konversation ein. In der Praxis wird der gesamte Verkehr aus Leistungsgründen anhand von symmetrischen Schlüsseln (normalerweise AES oder Triple DES) verschlüsselt, obwohl auch öffentliche Schlüssel für die Authentifizierungsprotokolle und für die Einrichtung des Sitzungsschlüssels verwendet werden.

Für jede neue Verbindung wird ein neuer, zufällig gewählter Sitzungsschlüssel benutzt, was dem Zweck dient, das Verkehrsvolumen zu minimieren, das mit den geheimen oder öffentlichen Schlüsseln der Benutzer ausgetauscht wird, die Menge an Chiffretext zu reduzieren, zu dem sich ein Eindringling eventuell Zugang verschafft, und den Schaden gering zu halten, wenn ein Prozess abstürzt und der dabei geschriebene Prozess-Dump in die falschen Hände gerät. In diesem Fall kann man nur hoffen, dass außer dem Sitzungsschlüssel keine anderen Schlüssel präsent sind. Alle permanenten Schlüssel sollten nach dem Aufbau der Sitzung sorgfältig auf null gesetzt werden.

8.7.1 Authentifizierung auf der Basis eines gemeinsamen geheimen Schlüssels

Bei unserem ersten Authentifizierungsprotokoll gehen wir davon aus, dass Alice und Bob bereits über einen gemeinsamen Schlüssel K_{AB} verfügen. Dieser gemeinsame Schlüssel wurde eventuell telefonisch oder persönlich vereinbart, jedenfalls nicht über das (unsichere) Netz.

Das Protokoll basiert auf einem Prinzip, das viele Authentifizierungsprotokolle aufweisen: Eine Partei sendet eine Zufallszahl an die andere, die sie dann auf spezielle Weise umwandelt und das Ergebnis zurückschickt. Solche Protokolle nennt man **Challenge-Response-Protokolle** (Aufforderung-Antwort-Protokolle). In diesem und den weiteren Authentifizierungsprotokollen wird folgende Notation benutzt:

A, B sind die Identitäten von Alice und Bob.

R_i sind die Aufforderungen; die auffordernde Partei wird durch Tiefstellung bezeichnet.

K_i sind Schlüssel, wobei i den Inhaber bezeichnet.

K_S ist der Sitzungsschlüssel.

Die Nachrichtenabfolge in unserem ersten Beispiel eines Authentifizierungsprotokolls mit gemeinsamem Schlüssel ist in ► Abbildung 8.32 ersichtlich. Mit Nachricht 1 sendet Alice ihre Identität A an Bob in einer Weise, die Bob versteht. Bob hat selbstverständlich keine Möglichkeit festzustellen, ob diese Nachricht von Alice oder Trudy stammt. Er wählt deshalb eine Aufforderung (*challenge*) in Form einer großen Zufallszahl R_B und sendet sie als Nachricht 2 im Klartext an „Alice“ zurück. Alice verschlüsselt die Nachricht dann mit dem Schlüssel, den sie mit Bob gemeinsam vereinbart hat, und sendet den Chiffertext $K_{AB}(R_B)$ in Nachricht 3 zurück. Wenn Bob diese Nachricht sieht, erkennt er sofort, dass sie von Alice kommt, da Trudy K_{AB} nicht kennt und die Nachricht daher nicht erzeugt haben kann. Da R_B zufällig aus einem großen Zahlenraum (z.B. 128-Bit-Zufallszahlen) gewählt wurde, ist es sehr unwahrscheinlich, dass Trudy R_B und die Antwort aus einer früheren Sitzung kennt. Ebenso ist es unwahrscheinlich, dass sie die korrekte Antwort auf irgendeine Aufforderung erraten kann.

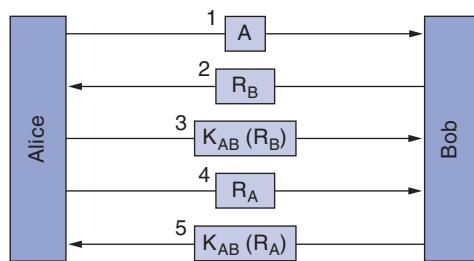


Abbildung 8.32: Zweiwege-Authentifizierung mithilfe des Challenge-Response-Protokolls.

Bob ist also inzwischen sicher, dass er es mit Alice zu tun hat, was bei Alice noch nicht der Fall ist. Sie weiß nur, dass Trudy in Nachricht 1 eingedrungen sein und R_B zurückgeschickt haben könnte. Vielleicht ist Bob letzte Nacht verstorben. Um auch ihrerseits sicherzustellen, mit wem sie es zu tun hat, wählt sie eine Zufallszahl R_A und sendet sie in Nachricht 4 als Klartext an Bob. Wenn Bob mit $K_{AB}(R_A)$ antwortet, weiß Alice, dass sie mit Bob spricht. Möchten die beiden jetzt einen Sitzungsschlüssel einrichten, wählt sich Alice einen K_S aus und sendet ihn mit K_{AB} verschlüsselt an Bob.

Das Protokoll von Abbildung 8.32 enthält fünf Nachrichten. Schauen wir einmal, ob wir hier clever sind und einige entfernen können. Ein Ansatz ist in ►Abbildung 8.33 dargestellt. Hier wartet Alice nicht auf Bobs Kontaktaufnahme, sondern leitet selbst ein Challenge-Response-Protokoll ein. Bob antwortet auf Alices Aufforderung, sendet seinerseits aber selbst eine. Das gesamte Protokoll kann von fünf auf drei Nachrichten reduziert werden.

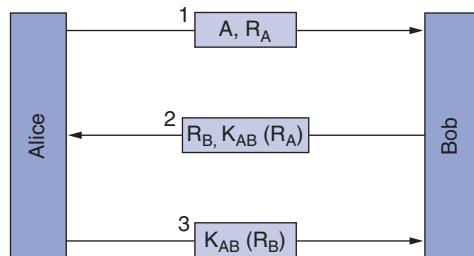


Abbildung 8.33: Gekürzte Form des Zweiwege-Authentifizierungsprotokolls.

Stellt das neue Protokoll eine Verbesserung gegenüber dem alten dar? In einer Hinsicht schon: Es ist kürzer. Leider ist es aber falsch. Unter bestimmten Umständen kann Trudy dieses Protokoll durch die sogenannte **Reflexionsattacke** (*reflection attack*) angreifen. Insbesondere kann sie es knacken, wenn es möglich ist, zu Bob mehrere Sitzungen gleichzeitig aufzubauen. Dieser Fall würde z.B. zutreffen, wenn Bob eine Bank wäre und bereit, viele gleichzeitige Verbindungen von Bankautomaten entgegenzunehmen.

Trudys Reflexionsattacke ist in ►Abbildung 8.34 dargestellt. Sie beginnt damit, sich als Alice auszugeben, und sendet R_T . Bob antwortet wie üblich mit seiner eigenen Aufforderung R_B . Jetzt hängt Trudy fest. Welche Möglichkeiten hat sie? Sie kennt $K_{AB}(R_B)$ nicht.

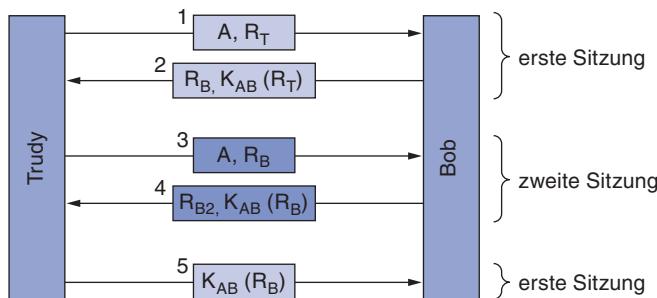


Abbildung 8.34: Trudys Reflexionsattacke.

Sie kann eine zweite Sitzung mit Nachricht 3 öffnen, mit der sie R_B aus Nachricht 2 als ihre Aufforderung bereitstellt. Bob entschlüsselt dies ruhig und sendet $K_{AB}(R_B)$ in Nachricht 4 zurück. Wir haben die Nachrichten in der zweiten Sitzung schattiert, um sie hervorzuheben. Nun hat Trudy die fehlende Information, kann die erste Sitzung vervollständigen und die zweite abbrechen. Bob ist überzeugt, dass Trudy Alice ist. Als sie nach dem Kontostand ihres Girokontos fragt, erhält sie deshalb prompt Aus-

kunft. Sie fordert ihn auf, die gesamte Summe auf ein Geheimkonto in der Schweiz zu überweisen, was er geflissentlich durchführt.

Die Moral von der Geschichte ist:

Das Design eines korrekten Authentifizierungsprotokolls ist viel komplizierter, als es aussieht.

Die folgenden vier allgemeinen Regeln helfen hier oftmals, die häufigsten Fallstricke zu vermeiden:

1. Der Initiator muss vor der antwortenden Seite seine Identität nachweisen. In diesem Fall hat Bob wertvolle Informationen preisgegeben, bevor Trudy den Nachweis ihrer Identität erbracht hat.
2. Der Initiator und die antwortende Seite müssen für den Nachweis verschiedene Schlüssel verwenden, auch wenn das bedeutet, dass mit zwei gemeinsamen Schlüsseln K_{AB} und K'_{AB} gearbeitet wird.
3. Der Initiator und die antwortende Seite müssen die Zufallszahl für ihre jeweilige Aufforderung aus unterschiedlichen Mengen ziehen. Der Initiator muss beispielsweise gerade und die antwortende Seite ungerade Zahlen verwenden.
4. Machen Sie das Protokoll resistent gegen Angriffe, die eine zweite parallele Sitzung aufbauen und dann die Informationen aus dieser Sitzung in der ersten Sitzung verwenden.

Wenn eine dieser Regeln verletzt wird, kann das Protokoll häufig ausgehebelt werden. Hier wurden alle vier Regeln missachtet, mit den entsprechend verheerenden Folgen.

Gehen wir nun zurück und sehen uns Abbildung 8.32 noch einmal genauer an. Ist es sicher, dass das Protokoll nicht Ziel einer Reflexionsattacke wird? Nun, das hängt von verschiedenen Faktoren ab. Es ist ziemlich raffiniert. Trudy war mit ihrer Reflexionsattacke erfolgreich, weil es möglich war, eine zweite Sitzung mit Bob zu eröffnen, und er darin ausgetrickst wurde, seine eigenen Fragen zu beantworten. Was passiert, wenn Alice keine Person am Rechner, sondern ein ganz normaler Rechner wäre, der mehrere Sitzungen akzeptiert? Sehen wir uns kurz einmal an, was Trudy tun kann.

Um zu sehen, wie der Angriff von Trudy funktioniert, betrachten wir ►Abbildung 8.35. Alice beginnt mit der Bekanntgabe ihrer Identität in Nachricht 1. Trudy fängt diese Nachricht ab und beginnt ihre eigene Sitzung mit Nachricht 2, in der sie behauptet, Bob zu sein. Auch hier haben wir wieder die schattierten Nachrichten in der zweiten Sitzung. Alice antwortet auf Nachricht 2: „Sie behaupten, Bob zu sein? Beweisen Sie dies in Nachricht 3“. An diesem Punkt hängt Trudy, weil sie nicht beweisen kann, dass sie Bob ist.

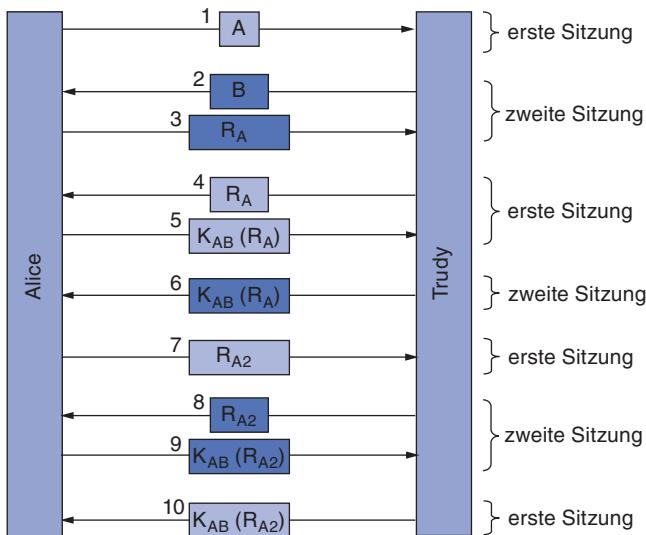


Abbildung 8.35: Eine Reflexionsattacke im Protokoll von Abbildung 8.32.

Was tut Trudy nun? Sie geht zur ersten Sitzung zurück, wo sie jetzt eine Aufforderung senden muss, und sendet das R_A , das sie in Nachricht 3 erhalten hat. Alice antwortet freundlich in Nachricht 5 und gibt Trudy so die Informationen, die sie benötigt, um die Nachricht 6 in der zweiten Sitzung zu senden. An diesem Punkt hat Trudy praktisch gewonnen, weil sie erfolgreich auf die Aufforderung von Alice in der zweiten Sitzung geantwortet hat. Sie kann die erste Sitzung nun abbrechen, eine beliebige alte Nummer für den Rest von Sitzung 2 senden und hat eine authentifizierte Sitzung mit Alice in der zweiten Sitzung.

Aber Trudy ist gemein und will es ihnen wirklich zeigen. Anstatt eine alte Nummer zu senden, um die zweite Sitzung fertig aufzubauen, wartet sie, bis Alice die Nachricht 7 sendet, die Aufforderung von Alice für die erste Sitzung. Trudy weiß natürlich nicht, wie sie antworten soll, sodass sie die Reflexionsattacke erneut einsetzt und R_{A2} als Nachricht 8 zurücksendet. Alice verschlüsselt praktischerweise R_{A2} in Nachricht 9. Trudy wechselt nun in Sitzung 1 zurück und sendet Alice die gewünschte Nummer in Nachricht 10, kopiert aus dem, was Alice in Nachricht 9 gesendet hat. An dieser Stelle hat Trudy zwei vollständig authentifizierte Sitzungen mit Alice.

Der Angriff hat ein etwas anderes Ergebnis als ein Angriff bei dem Protokoll mit den drei Nachrichten, wie in Abbildung 8.34 gezeigt. Dieses Mal hat Trudy zwei authentifizierte Verbindungen mit Alice. Im vorherigen Beispiel hatte sie nur eine authentifizierte Verbindung mit Bob. Auch hier hätte der Angriff aufgehalten werden können, wenn wir alle allgemeinen oben erörterten Regeln für Authentifizierungsprotokolle befolgt hätten. Eine detaillierte Erörterung dieser Art von Angriffen und wie sie abgewehrt werden ist in Bird et al. (1993) enthalten. Sie zeigen auch, wie man systematisch Protokolle erstellen kann, die nachweisbar korrekt sind. Da bereits das einfachste dieser Protokolle relativ kompliziert ist, wollen wir hier eine andere Klasse von Protokollen zeigen, die auch funktioniert.

Das neue Authentifizierungsprotokoll wird in ►Abbildung 8.36 (Bird et al., 1993) dargestellt. Es verwendet einen HMAC vom gleichen Typ wie IPsec. Alice beginnt mit dem Senden einer Nonce R_A an Bob als Nachricht 1. Bob antwortet, indem er seine eigene Nonce R_B auswählt und diese zusammen mit einem HMAC zurücksendet. Der HMAC ist aus einer Datenstruktur aufgebaut, die aus der Nonce von Alice, der von Bob, ihren Identitäten und dem gemeinsamen geheimen Schlüssel K_{AB} besteht. Aus dieser Datenstruktur wird dann im HMAC ein Hash erzeugt, beispielsweise mit SHA-1. Erhält Alice Nachricht 2, hat sie nun R_A (das sie selbst gewählt hat), R_B , das als Klartext ankommt, zwei Identitäten und den geheimen Schlüssel K_{AB} , der bekannt ist, sodass sie den HMAC selbst berechnen kann. Stimmt dieser mit dem HMAC in der Nachricht überein, weiß sie, dass sie mit Bob spricht, da Trudy K_{AB} nicht kennt und daher nicht herausbekommen kann, welcher HMAC gesendet werden soll. Alice antwortet Bob mit einem HMAC, der nur die beiden Nonces enthält.

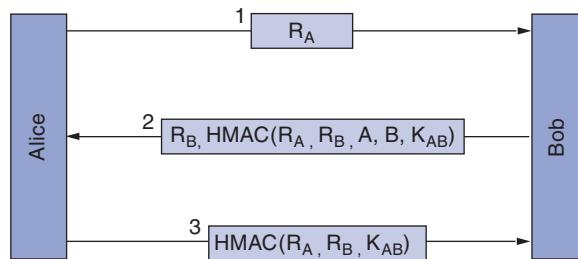


Abbildung 8.36: Authentifizierung mit HMACs.

Kann Trudy dieses Protokoll irgendwie unterlaufen? Nein, weil sie keine der beiden Parteien zwingen kann, einen von ihr gewählten Wert zu verschlüsseln oder dafür einen Hash zu bilden, wie dies in Abbildung 8.34 und Abbildung 8.35 passiert ist. Beide HMACs enthalten Werte, die von der sendenden Partei gewählt wurden, was Trudy nicht kontrollieren kann.

Für dieses Konzept können nicht nur HMACs verwendet werden. Ein alternatives Schema, das oftmals verwendet wird, besteht darin, nicht den HMAC über eine Reihe von Elementen zu berechnen, sondern diese Elemente sequenziell mit Cipher Block Chaining zu verschlüsseln.

8.7.2 Einrichten eines gemeinsamen Schlüssels: Das Schlüsselaustauschprotokoll von Diffie und Hellman

Bisher sind wir davon ausgegangen, dass Alice und Bob einen gemeinsamen Geheimschlüssel haben. Angenommen sie haben keinen (da es bislang keine universell akzeptierte PKI zur Signierung und Verteilung von Zertifikaten gibt). Wie können sie einen solchen Schlüssel einrichten? Eine Möglichkeit wäre, dass Alice Bob anruft und ihm ihren Schlüssel telefonisch durchgibt. Er würde wahrscheinlich aber so reagieren: „Wie kann ich wissen, dass Sie Alice und nicht Trudy sind?“ Die beiden könnten ein Treffen vereinbaren, zu dem beide ihren Personalausweis, den Führerschein, drei Kreditkarten usw. mitbringen, aber viel beschäftigt, wie sie sind, würde ein beiden geneh-

mer Termin wahrscheinlich erst in Wochen zustande kommen. Glücklicherweise gibt es, so unglaublich sich das anhört, eine Möglichkeit für völlig fremde Leute, am helllichten Tag einen gemeinsamen Geheimschlüssel zu vereinbaren, auch wenn Trudy fleißig alle Nachrichten aufzeichnet.

Das Protokoll, mit dem dies möglich ist, heißt **Diffie-Hellman-Schlüsselaustausch** (Diffie und Hellman, 1976). Es funktioniert wie folgt: Alice und Bob müssen sich auf zwei große Zahlen n und g einigen, wobei n eine Primzahl ist, ebenso wie $(n-1)/2$, und auf g bestimmte Bedingungen zutreffen. Diese Zahlen können öffentlich sein, sodass jeder von ihnen n und g auswählen und dem anderen mitteilen kann. Nun wählt Alice eine große Zahl x (z.B. 1 024 Bit groß) und hält sie geheim. Das Gleiche macht Bob mit y .

Alice leitet das Schlüsselaustauschprotokoll ein, indem sie Bob eine Nachricht mit $(n, g, g^x \bmod n)$ sendet (siehe ▶ Abbildung 8.37). Bob antwortet Alice mit der Nachricht $g^y \bmod n$. Alice erhebt die von Bob erhaltene Zahl in die x -te Potenz mod n und erhält so $(g^y \bmod n)^x \bmod n$. Bob führt die gleiche Operation durch und erhält $(g^x \bmod n)^y \bmod n$. Nach dem Gesetz der modularen Arithmetik ergeben beide Berechnungen $g^{xy} \bmod n$. Und siehe da, Alice und Bob haben damit einen Geheimschlüssel $g^{xy} \bmod n$.

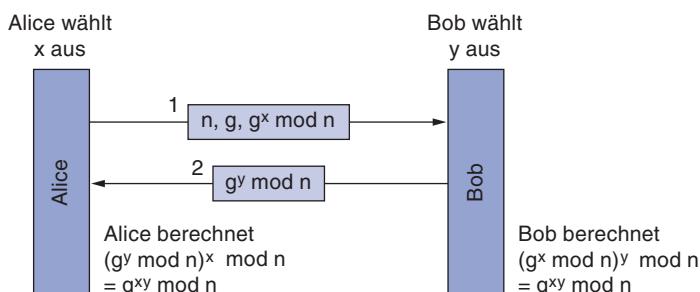


Abbildung 8.37: Das Schlüsselaustauschprotokoll von Diffie und Hellman.

Trudy zapft selbstverständlich auch diese beiden Nachrichten an. Sie kennt g und n aus Nachricht 1. Wenn sie x und y berechnen könnte, wäre sie in der Lage, den Geheimschlüssel herausfinden. Zu ihrem Pech kann sie mit $g^x \bmod n$ allein x nicht ermitteln. Für die Berechnung diskreter Logarithmen modulo einer sehr großen Primzahl ist kein praktischer Algorithmus bekannt.

Um das obige Beispiel zu konkretisieren, verwenden wir die (völlig unrealistischen) Werte $n=47$ und $g=3$. Alice wählt $x=8$ und Bob $y=10$. Beide werden geheim gehalten. Alices Nachricht an Bob lautet $(47, 3, 28)$, weil $3^8 \bmod 47 28$ ergibt. Bobs Nachricht an Alice ist (17) . Alice berechnet $17^8 \bmod 47$, was 4 ergibt. Bob berechnet $28^8 \bmod 47$, was 4 ergibt. Alice und Bob haben unabhängig voneinander ermittelt, dass der geheime Schlüssel nun 4 ist. Trudy muss die Gleichung $3^x \bmod 47 = 28$ lösen, was bei kleinen Zahlen wie dieser durch erschöpfende Suche durchgeführt werden kann. Aber wenn alle Zahlen Hunderte von Bits lang sind, funktioniert dies nicht mehr. Alle derzeit bekannten Algorithmen brauchen einfach zu lange, selbst auf massiv-parallelen, blitzschnellen Rechnern.

Trotz der Eleganz des Diffie-Hellman-Algorithmus gibt es ein Problem: Wenn Bob das Tripel $(47, 3, 28)$ erhält, kann er nicht unbedingt sicher sein, dass es von Alice und nicht von Trudy ist. Er hat keine Möglichkeit, das festzustellen. Trudy kann diese Tatsache ausnutzen und sowohl Alice als auch Bob täuschen (► Abbildung 8.38). Während Alice und Bob x bzw. y wählen, wählt Trudy eine eigene Zufallszahl z . Alice sendet Nachricht 1 an Bob. Trudy fängt diese ab und sendet Nachricht 2 an Bob mit dem richtigen g und n (die ohnehin öffentlich sind), aber mit ihrem eigenen z anstelle von x . Sie sendet auch Nachricht 3 an Alice zurück. Später sendet Bob Nachricht 4 an Alice, die von Trudy wiederum abgefangen und zurückbehalten wird.

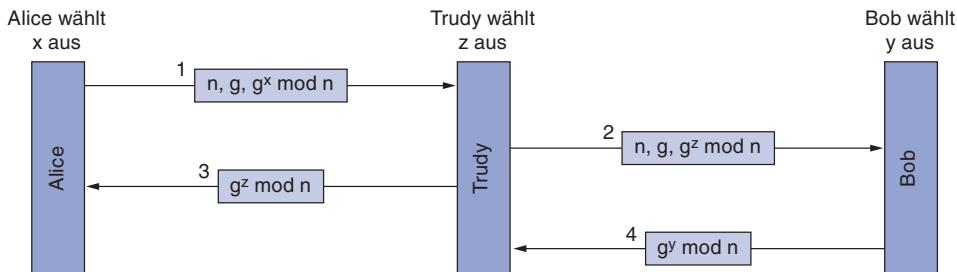


Abbildung 8.38: Der Bucket-Brigade- oder Man-in-the-Middle-Angriff.

Jetzt führt jeder die modulare Arithmetik durch. Alice berechnet den Geheimschlüssel als $g^{xz} \text{ mod } n$, und Trudy macht das Gleiche (für die Nachrichten an Alice). Bob berechnet $g^{yz} \text{ mod } n$, und Trudy macht wieder das Gleiche (für die Nachrichten an Bob). In der Annahme, mit Bob zu kommunizieren, richtet Alice (mit Trudy) einen Geheimschlüssel ein. Bob macht das Gleiche. Jede Nachricht, die Alice über die verschlüsselte Sitzung sendet, wird von Trudy abgefangen, gespeichert, nach Wunsch geändert und dann (wahlweise) an Bob weitergegeben. Die gleiche Prozedur erfolgt in entgegengesetzter Richtung. Trudy sieht alles und kann alle Nachrichten willkürlich ändern, während Alice und Bob überzeugt sind, über einen sicheren Kanal zu kommunizieren. Aus diesem Grund wird diese Attacke auch als **Man-in-the-Middle-Angriff** (Mittelsmann-Angriff) bezeichnet. Sie wird auch **Bucket-Brigade-Angriff** (Eimerketten-Angriff) genannt, weil sie Ähnlichkeit mit der Feuerwehr aus alten Zeiten hat.

8.7.3 Authentifizierung mithilfe eines Schlüsselverteilungszentrums

Wir haben gesehen, dass die Einrichtung eines Geheimschlüssels mit einem Fremden fast, aber nicht ganz funktioniert. Eventuell lohnt sich der ganze Aufwand überhaupt nicht. Um auf diese Weise mit n Leuten zu kommunizieren, braucht man n Schlüssel. Bei populären Personen wäre die Verwaltung der Schlüssel eine echte Belastung, insbesondere, wenn jeder Schlüssel auf einer eigenen Kunststoffchipkarte gespeichert werden muss.

Ein anderer Ansatz wird mit der Einführung eines vertrauenswürdigen Schlüsselverteilungszentrums (*Key Distribution Center, KDC*) verfolgt. Bei diesem Modell hat jeder Benutzer einen einzelnen Schlüssel gemeinsam mit dem KDC. Die Authentifizierung und Sitzungsschlüssel gehen nun über das KDC. Das einfachste bekannte KDC-

Authentifizierungsprotokoll mit zwei Parteien und einem vertrauenswürdigen KDC wird in ▶ Abbildung 8.39 dargestellt.

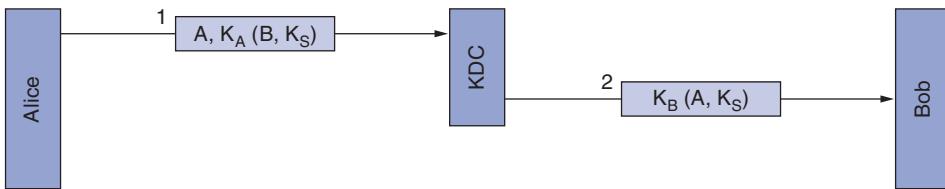


Abbildung 8.39: Ein erster Versuch zu einem Authentifizierungsprotokoll mit einem KDC.

Das Konzept für dieses Protokoll ist einfach: Alice wählt einen Sitzungsschlüssel K_S und teilt dem KDC mit, dass sie mittels K_S mit Bob kommunizieren will. Diese Nachricht ist mit dem Geheimschlüssel K_A , den nur Alice und das KDC kennen, verschlüsselt. Das KDC entschlüsselt diese Nachricht, entnimmt die Identität von Bob und den Sitzungsschlüssel. Dann sendet es eine Nachricht mit der Identität von Alice und dem Sitzungsschlüssel an Bob. Diese Nachricht wird mit K_B , dem Geheimschlüssel von Bob und dem KDC, verschlüsselt. Bob entschlüsselt diese Nachricht und erfährt, dass Alice mit ihm kommunizieren will und welchen Schlüssel sie dafür verwenden möchte.

Die Authentifizierung erfolgt hier kostenfrei. Das KDC weiß, dass Nachricht 1 nur von Alice kommen kann, da außer ihr und dem KDC selbst niemand den Geheimschlüssel kennt. Bob weiß, dass Nachricht 2 vom KDC, dem er vertraut, kommen muss, da niemand sonst seinen Geheimschlüssel kennt.

Leider weist dieses Protokoll einen schwerwiegenden Schwachpunkt auf. Trudy braucht Bares und denkt sich eine „ganz normale“ Arbeit aus, die sie für Alice erbringen kann. Sie bewirbt sich und erhält den Job. Nach beendeter Arbeit fordert Trudy Alice dann höflich auf, mittels Banküberweisung zu bezahlen. Alice errichtet dann einen Sitzungsschlüssel mit ihrem Banker Bob. Dann sendet sie Bob eine Nachricht, in der sie ihn auffordert, Geld auf das Konto von Trudy zu überweisen.

Inzwischen ist Trudy wieder damit beschäftigt, im Netz zu schnüffeln. Sie kopiert Nachricht 2 von Abbildung 8.39 und den darauffolgenden Überweisungsauftrag. Später sendet sie beides erneut an Bob, der dieser Übertragung entnimmt, dass Alice Trudy noch einmal beschäftigt haben muss, weil sie wohl gute Arbeit leistet. Bob überträgt den gleichen Betrag vom Konto von Alice auf das von Trudy. Einige Zeit später, nach dem 50. Nachrichtenpaar, nimmt Bob mit Trudy Verbindung auf, um ihr ein Darlehensangebot zu machen, da ihr Geschäft offensichtlich phantastisch läuft. Dieses Problem nennt man **Replay-Attacke**.

Für eine Replay-Attacke gibt es mehrere Lösungen. Erstens kann man in jede Nachricht einen Zeitstempel einbinden. Erhält dann jemand eine überholte Nachricht, kann sie unverzüglich verworfen werden. Leider ist das problematisch, weil nicht alle Uhren im Netz gleich ticken, sodass eventuell auch der Zeitstempel kein verlässlicher Hinweis ist. Trudy kann mit ein wenig Aufwand Synchronisationslücken im Netz ausmachen und sie für ihre Replay-Nachrichten nutzen.

Die zweite Lösung ist, eine Nonce in jede Meldung einzufügen. Jede Partei muss sich an alle vorherigen Nonces erinnern und alle Nachrichten, die eine bereits benutzte Nonce enthalten, verwerfen. Diese Nonces müssen über viele Jahre gespeichert werden, weil Trudy eventuell eine fünf Jahre alte Nachricht abspielt. Stürzt eine Maschine ab, sodass die Liste der Nonces verloren geht, ist das Feld wieder frei für Replay-Attacken. Zeitstempel und Nonces können kombiniert werden, um einzuschränken, wie lang man sich Nonces merken muss. Allerdings wird das Protokoll hierdurch viel komplizierter.

Ein anspruchsvollerer Ansatz zur gegenseitigen Authentifizierung ist ein mehrwegiges Challenge-Response-Protokoll. Ein bekanntes Beispiel hierfür ist das **Needham-Schroeder-Authentifizierungsprotokoll** (Needham und Schroeder, 1978), von dem eine Variante in ▶ Abbildung 8.40 dargestellt ist.

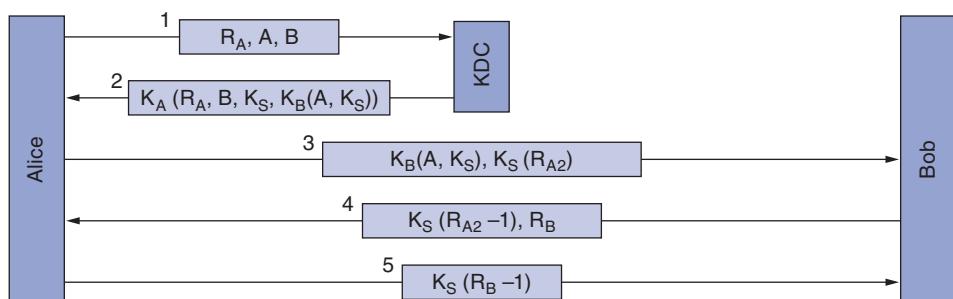


Abbildung 8.40: Das Needham-Schroeder-Authentifizierungsprotokoll.

Das Protokoll beginnt damit, dass Alice dem KDC die beabsichtigte Kommunikation mit Bob mitteilt. Diese Nachricht enthält eine große Zufallszahl R_A als Nonce. Das KDC sendet Nachricht 2 mit Alices Zufallsnummer, einem Sitzungsschlüssel und einem Ticket zurück, das sie an Bob senden kann. Mit der Zufallszahl R_A soll für Alice der Nachweis erbracht werden, dass Nachricht 2 neu und kein Replay ist. Bobs Identität ist ebenfalls enthalten für den Fall, dass Trudy das B in Nachricht 1 durch ihre eigene Identität ersetzt. In diesem Fall würde das KDC das Ticket am Ende von Nachricht 2 mit K_T anstelle von K_B verschlüsseln. Das mit K_B verschlüsselten Ticket ist in der verschlüsselten Nachricht eingebettet, sodass es auf dem Weg zu Alice nicht durch etwas anderes ersetzt werden kann.

Alice sendet das Ticket mit einer neuen Zufallszahl R_{A2} , mit Sitzungsschlüssel K_S verschlüsselt, an Bob. Mit Nachricht 4 sendet Bob $K_S(R_{A2}-1)$ zurück, um Alice zu beweisen, dass er der echte Bob ist. Das Zurücksenden von $K_S(R_{A2})$ hätte nicht funktioniert, weil Trudy dies gerade aus Nachricht 3 gestohlen haben könnte.

Nach Erhalt der Nachricht 4 ist Alice nun überzeugt, dass sie mit Bob kommuniziert und bislang keine Replays verwendet wurden. Schließlich hat sie R_{A2} gerade erst vor ein paar Millisekunden erzeugt. Mit Nachricht 5 soll Bob davon überzeugt werden, dass er mit der echten Alice verhandelt und keine Replays im Spiel sind. Da hier jede Partei für jede Nachricht eine Aufforderung und eine Antwort erzeugt, werden Replay-Angriffe ausgeschlossen.

Obwohl dieses Protokoll ziemlich stabil zu sein scheint, entdeckt man auf den zweiten Blick eine kleine Schwäche. Sollte es Trudy je gelingen, einen alten Sitzungsschlüssel im Klartext in die Hände zu bekommen, kann sie eine neue Sitzung zu Bob aufbauen und ihm Nachricht 3 vorspielen, mit der sie ihn davon überzeugt, dass sie Alice ist (Denning und Sacco, 1981). Diesmal könnte sie Alices Bankkonto plündern, ohne auch nur eine einzige „ganz normale“ Arbeit erbracht zu haben.

Needham und Schroeder haben später ein Protokoll veröffentlicht, in dem dieses Problem behoben wurde (Needham und Schroeder, 1987). In der gleichen Zeitschriftausgabe haben Otway und Rees (1987) ebenfalls ein Protokoll veröffentlicht, das dieses Problem mit weniger Zeitaufwand löst. ►Abbildung 8.41 zeigt das leicht modifizierte Otway-Rees-Protokoll.

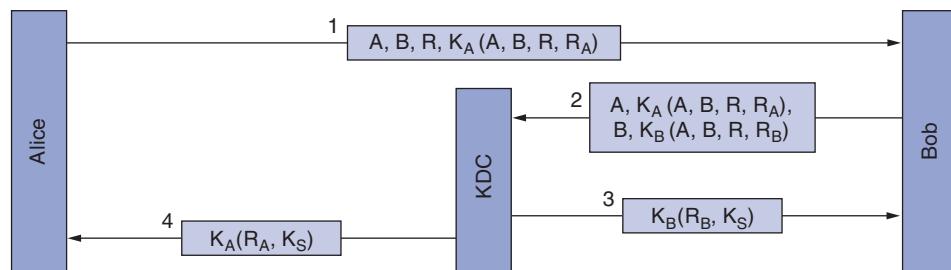


Abbildung 8.41: Das Otway-Rees-Authentifizierungsprotokoll (etwas vereinfacht).

Beim Otway-Rees-Protokoll beginnt Alice damit, dass sie ein Zufallszahlenpaar erzeugt: R als gemeinsamen Identifikator und R_A als Aufforderung an Bob. Nachdem Bob diese Nachricht erhalten hat, bildet er aus dem verschlüsselten Teil von Alices Nachricht eine neue und eine analoge von ihm. Beide mit K_A und K_B verschlüsselten Teile identifizieren Alice und Bob und enthalten sowohl den gemeinsamen Bezeichner als auch eine Aufforderung.

Das KDC prüft, ob R in beiden Teilen gleich ist. Dies trifft zum Beispiel nicht zu, wenn Trudy R in Nachricht 1 gefälscht oder einen Teil von Nachricht 2 ersetzt hat. Wenn die beiden R übereinstimmen, glaubt das KDC, dass die Anforderungsnachricht von Bob gültig ist. Es erzeugt einen Sitzungsschlüssel und verschlüsselt ihn zweimal, einmal für Alice und einmal für Bob. Jede Nachricht enthält die Zufallszahl des Empfängers als Nachweis, dass sie vom KDC und nicht von Trudy erzeugt wurden. Damit besitzen Alice und Bob den gleichen Sitzungsschlüssel und können mit der Übertragung beginnen. Beim ersten Austausch von Datennachrichten können beide erkennen, dass die jeweils andere Partei eine identische Kopie von K_S besitzt, sodass die Authentifizierung damit abgeschlossen ist.

8.7.4 Authentifizierung mit Kerberos

Ein in vielen echten Systemen (wie Windows 2000 und später) benutztes Authentifizierungsprotokoll ist **Kerberos**, das auf einer Variante von Needham-Schroeder basiert. Das Protokoll wurde nach dem mehrköpfigen Hund aus der griechischen Mythologie

benannt, der den Eingang zum Hades bewacht (sicherlich, um unerwünschte Besucher fernzuhalten). Kerberos wurde am MIT mit dem Ziel entwickelt, Workstation-Benutzern sicheren Zugriff auf Netzressourcen zu ermöglichen. Der größte Unterschied zwischen Kerberos und dem Needham-Schroeder-Protokoll liegt in der Annahme, dass alle Uhren relativ synchron sind. Das Protokoll hat mehrere Überarbeitungen durchlaufen. V5 ist die heute in der Industrie am meisten verwendete Version und wird in RFC 4120 definiert. Die Vorgängerversion V4 wurde zurückgezogen, nachdem ernste Fehler gefunden wurden (Yu et al., 2004). V5 zeichnet sich gegenüber V4 durch viele kleine Änderungen aus sowie durch einige verbesserte Funktionen, z.B. dass es nicht mehr auf dem inzwischen veralteten DES aufsetzt. Weitere Informationen finden Sie in Neuman und Ts'o (1994).

Zusätzlich zu Alice (einer Client-Workstation) umfasst Kerberos drei Server:

- Authentifizierungsserver (AS): Prüft Benutzer beim Anmelden.
- Ticketausgabe-Server (TGS, *Ticket-Granting Server*): Gibt die zum Nachweis der Identität benutzten Tickets aus.
- Bob-Server: Führt die Arbeit aus, die Alice anfordert.

Der AS ist mit einem KDC dahingehend vergleichbar, dass er gemeinsam mit jedem Benutzer ein geheimes Passwort teilt. Der TGS hat die Aufgabe, Tickets auszugeben, die die wahren Server davon überzeugen können, dass der Inhaber eines TGS-Tickets tatsächlich der ist, der er behauptet zu sein.

Um eine Sitzung zu beginnen, setzt sich Alice an eine beliebige öffentliche Workstation und tippt ihren Namen ein. Die Workstation sendet ihren Namen und den Namen der TGS im Klartext an den AS (►Abbildung 8.42). Was zurückkommt, ist ein Sitzungsschlüssel und ein Ticket $K_{TGS}(A, K_S, t)$, das für den TGS bestimmt ist. Diese Elemente werden mit dem Geheimschlüssel von Alice verschlüsselt, sodass sie nur von Alice entschlüsselt werden können. Erst bei Ankunft von Nachricht 2 fordert die Workstation das Passwort von Alice an. Mit dem Passwort wird dann K_A erzeugt, um Nachricht 2 zu entschlüsseln sowie den darin enthaltenen Sitzungsschlüssel zu erhalten.

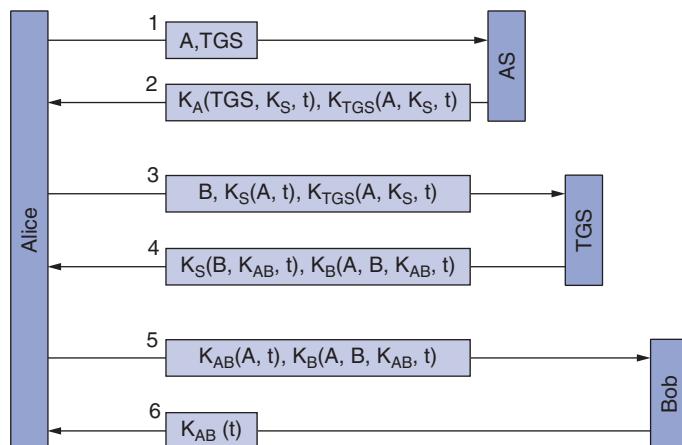


Abbildung 8.42: Ablauf von Kerberos V4.

An diesem Punkt überschreibt die Workstation Alices Passwort, um sicherzustellen, dass es sich höchstens ein paar Millisekunden in der Workstation befindet. Versucht Trudy, sich als Alice anzumelden, wird sie von der Workstation aufgrund eines falschen Passworts abgewiesen, weil sich der Standardteil von Nachricht 2 geändert hat.

Nachdem sich Alice angemeldet hat, kann sie der Workstation mitteilen, dass sie mit dem Dateiserver Bob Kontakt aufnehmen möchte. Die Workstation sendet Nachricht 3 an den TGS und fordert ein Ticket für den Einsatz mit Bob an. Das wichtigste Element in dieser Anforderung ist $K_{TGS}(A, K_S, t)$, das mit dem geheimen Schlüssel des TGS verschlüsselt wird und als Beweis dient, dass der Sender wirklich Alice ist. Der TGS antwortet Alice in Nachricht 4 mit der Erzeugung eines Sitzungsschlüssels K_{AB} , den sie zur Kommunikation mit Bob verwenden kann. Davon werden zwei Versionen zurückgeschickt. Die erste ist nur mit K_S verschlüsselt und kann von Alice gelesen werden. Die zweite ist ein weiteres Ticket, das mit dem Schlüssel von Bob K_B verschlüsselt ist, sodass Bob es lesen kann.

Trudy kann Nachricht 3 kopieren und versuchen, sie wiederzuverwenden, wird aber aufgrund des verschlüsselten Zeitstempels t , der mit der Nachricht gesendet wird, zurückgewiesen. Sie kann den Zeitstempel nicht durch einen neueren ersetzen, weil sie K_S , den Sitzungsschlüssel von Alice für die Kommunikation mit TGS, nicht kennt. Auch wenn Trudy Nachricht 3 schnell abspielt, erhält sie lediglich eine Kopie von Nachricht 4, die sie beim ersten Mal nicht entschlüsseln konnte und auch diesmal nicht entschlüsseln kann.

Nun kann Alice K_{AB} über das neue Ticket an Bob senden und eine Sitzung mit ihm aufbauen (Nachricht 5). Auch dieser Informationsaustausch erhält einen Zeitstempel. Die optionale Antwort (Nachricht 6) ist der Nachweis, dass Alice wirklich mit Bob, nicht mit Trudy, kommuniziert.

Nach dieser Abfolge von Nachrichten kann Alice mit Bob unter dem Schutzmantel von K_{AB} kommunizieren. Entschließt sie sich später, mit dem Server Carol zu kommunizieren, muss sie lediglich Nachricht 3 zum TGS wiederholen, nur diesmal C anstelle von B angeben. Der TGS antwortet prompt mit einem mit K_C verschlüsselten Ticket, das Alice an Carol senden kann, die es als Nachweis der Identität von Alice annimmt.

Der springende Punkt all dieser Mühe ist, dass Alice nun im gesamten Netz sicher auf Server zugreifen kann. Ihr Passwort muss nie im Netz verteilt werden. Es lag nur einige Millisekunden lang auf ihrer eigenen Workstation vor. Jeder Server ist aber für seine eigene Autorisation zuständig. Wenn Alice ihr Ticket Bob vorlegt, erhält er lediglich den Beweis, von wem es kommt. Wozu Alice berechtigt ist, liegt bei Bob.

Da nicht zu erwarten war, dass die ganze Welt nur einem einzigen Authentifizierungs-server trauen würde, trafen Kerberos-Entwickler Vorkehrungen für mehrere **Bereiche** (*realm*) mit je einem eigenen AS und TGS. Um ein Ticket für einen Server in einem entfernten Bereich zu erhalten, würde Alice ihren eigenen TGS um ein Ticket bitten, das der TGS im entfernten Bereich akzeptiert. Hat sich der entfernte TGS beim lokalen TGS (auf die für lokale Server übliche Weise) registriert, gibt der lokale TGS Alice ein für den entfernten TGS gültiges Ticket. Mit diesem Ticket kann sie auf die Server in

diesem Bereich zugreifen. Zu beachten ist hier, dass zwei Parteien aus verschiedenen Bereichen, die miteinander kommunizieren wollen, dem TGS des jeweils anderen vertrauen müssen. Ansonsten ist keine Kommunikation möglich.

8.7.5 Authentifizierung mithilfe öffentlicher Verschlüsselung

Die gegenseitige Authentifizierung kann auch mit öffentlichen Schlüsseln erfolgen. Zu Beginn benötigt Alice den öffentlichen Schlüssel von Bob. Wenn eine PKI mit einem Verzeichnisserver existiert, der die Zertifikate für öffentliche Schlüssel ausgibt, kann Alice nach dem Schlüssel von Bob fragen (in ▶ Abbildung 8.43 als Nachricht 1). Die Antwort in Nachricht 2 ist ein X.509-Zertifikat, das den öffentlichen Schlüssel von Bob enthält. Nachdem Alice verifiziert hat, dass die Signatur korrekt ist, sendet sie Bob eine Nachricht mit ihrer Identität und einer Nonce.

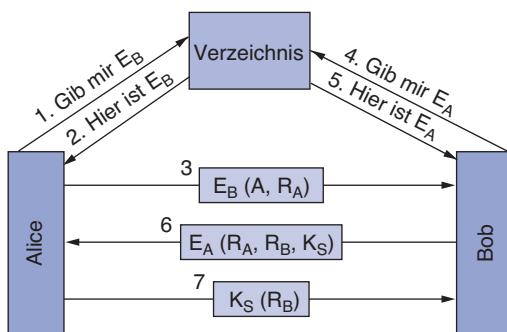


Abbildung 8.43: Gegenseitige Authentifizierung mithilfe öffentlicher Schlüssel.

Erhält Bob diese Nachricht, kann er zunächst nicht wissen, ob sie von Alice oder Trudy kommt. Er spielt aber mit und fragt den Verzeichnisserver nach dem öffentlichen Schlüssel für Alice (Nachricht 4), den er sodann erhält (Nachricht 5). Er sendet Alice in Nachricht 6 das R_A von Alice, seine eigene Nonce R_B und einen vorgeschlagenen Sitzungsschlüssel K_S .

Alice erhält Nachricht 6 und entschlüsselt sie mithilfe ihres privaten Schlüssels. Das R_A in der Nachricht nimmt sie mit Freuden zur Kenntnis, denn jetzt weiß sie, dass die Nachricht von Bob sein muss, da Trudy keine Möglichkeit hat, R_A zu ermitteln. Außerdem ist dadurch sichergestellt, dass die Nachricht aktuell ist, weil Alice eben erst R_A an Bob gesendet hat. Alice stimmt der Sitzung durch Zurücksenden von Nachricht 7 zu. Wenn Bob R_B im Sitzungsschlüssel, den er gerade erzeugt hat, sieht, weiß er, dass Alice Nachricht 6 erhalten und R_A geprüft hat. Jetzt freut sich auch Bob.

Wie kann Trudy dieses Protokoll unterlaufen? Sie könnte Nachricht 3 fabrizieren und sich Bob gegenüber als Alice ausgeben. Alice würde dann aber ein R_A erhalten, das sie nicht gesendet hat, und die Sitzung abbrechen. Trudy kann Nachricht 7 an Bob nicht fälschen, weil sie weder R_B noch K_S kennt und diese ohne Alices privaten Schlüssel auch nicht ermitteln kann. Diesmal hat sie Pech.

8.8 E-Mail-Sicherheit

Eine zwischen zwei entfernten Standorten ausgetauschte E-Mail passiert auf ihrem Weg in der Regel Dutzende von Rechnern. Jeder davon könnte die Nachricht lesen und zur späteren Verwendung aufzeichnen. In der Praxis gibt es keinen Datenschutz – unabhängig von dem, was viele denken. Dennoch möchten viele, dass die von ihnen versendeten E-Mails nur von den angegebenen Empfängern und sonst von niemandem gelesen werden, weder von ihrem Chef noch von der Regierung. Dieser Wunsch hat verschiedene Leute und Gruppen dazu angeregt, die an früherer Stelle beschriebenen kryptografischen Prinzipien auch auf E-Mail anzuwenden. In den folgenden Abschnitten untersuchen wir das weitverbreitete, sichere E-Mail-System PGP und gehen dann kurz auf ein anderes ein: S/MIME. Weitere Informationen zum sicheren Versenden von E-Mail findet Sie in Kaufman et al. (2002) sowie bei Schneier (1995).

8.8.1 PGP – Pretty Good Privacy

Unser erstes Beispiel, **PGP** (*Pretty Good Privacy*), ist im Wesentlichen das geistige Produkt einer einzelnen Person: Phil Zimmermann (Zimmermann, 1995a, 1995b). Zimmermann ist ein Verfechter des Datenschutzes, dessen Motto lautet: „Wird Datenschutz kriminalisiert, dann haben nur Kriminelle Datenschutz.“ Bei PGP, das 1991 vorgestellt wurde, handelt es sich um ein komplettes E-Mail-Sicherheitspaket, das Datenschutz, Authentifizierung, digitale Signaturen und Komprimierung in bedienungsfreundlicher Form bietet. Des Weiteren wird das Komplettspaket mitsamt dem ganzen Quellcode kostenlos über das Internet verteilt. Aufgrund der Qualität, des Preises (Nulltarif) und der einfachen Verfügbarkeit für MS-DOS/Windows, Unix, Linux und Macintosh wird es heute häufig eingesetzt.

PGP verschlüsselt Daten mit einer Blockchiffre namens **IDEA** (*International Data Encryption Algorithm*), die 128-Bit-Schlüssel verwendet. Sie wurde in der Schweiz entwickelt, als DES bereits als veraltet galt und AES noch nicht erfunden war. Von der Konzeption her funktioniert IDEA ähnlich wie DES und AES: Sie mischt die Bits in mehreren Runden, aber die Details der Mischfunktionen unterscheiden sich von DES und AES. Bei der Schlüsselverwaltung wird RSA verwendet und die Datenintegrität verwendet MD5; beide Themen haben wir bereits erörtert.

PGP hat von Beginn an zu vielen Kontroversen Anlass gegeben (Levy, 1993). Da Zimmermann niemanden daran hinderte, PGP für jedermann zugänglich ins Internet zu stellen, behauptete die amerikanische Regierung, dass Zimmermann gegen die amerikanischen Gesetze zum Export von Rüstungsgütern verstößen habe. Die Untersuchung der amerikanischen Regierung dauerte fünf Jahre, wurde aber schließlich aus zwei Gründen eingestellt. Erstens hatte Zimmermann PGP nicht selbst ins Internet gestellt, sodass sein Anwalt anführte, dass er nie etwas exportiert habe (wobei dann noch der Sachverhalt zu untersuchen war, ob das Erstellen einer Website überhaupt als Export zu bewerten ist). Zweitens erkannte die Regierung schließlich, dass zum Gewinnen eines Prozesses das Gericht überzeugt werden musste, dass eine Website mit einem herunterladbaren Datenschutzprogramm unter das Waffenhandelsgesetz fällt, das den Export von Kriegsmate-

rialien wie Panzer, U-Boote, Militärflugzeuge und Atomwaffen verbietet. Außerdem waren die jahrelangen Negativschlagzeilen auch nicht gerade von Vorteil.

Nebenbei bemerkt sind die US-Exportregeln grotesk, um es freundlich auszudrücken. Die Regierung betrachtete das Ablegen von Code auf einer Website als illegalen Export und machte Zimmermann fünf Jahre lang das Leben schwer. Wenn andererseits jemand den vollständigen PGP-Quellcode in C als Buch veröffentlichte (in großer Schrift mit einer Prüfsumme auf jeder Seite, um das Scannen zu vereinfachen) und dann das Buch exportierte, war das für die Regierung in Ordnung, weil Bücher nicht als Waffen eingestuft werden. Das Schwert ist mächtiger als die Feder, zumindest für Uncle Sam.

Ein weiteres Problem von PGP war, dass es gegen Patentrechte verstieß. Das Unternehmen RSA Security, Inc., das das RSA-Patent hielt, behauptete, dass PGP durch die Verwendung des RSA-Algorithmus ihr Patent verletzte. Der Konflikt wurde aber mit Einführung von Version 2.6 und höher beigelegt. Darüber hinaus verwendet PGP noch einen weiteren patentierten Verschlüsselungsalgorithmus, IDEA, dessen Einsatz anfangs ein paar Probleme verursachte.

Da PGP als Open-Source-Quellcode vorliegt, haben diverse Einzelpersonen und Gruppen den Quellcode weiterentwickelt und eigene Versionen erzeugt. Einige Versionen zielen darauf ab, das Waffenexportgesetz zu umgehen, andere konzentrierten sich auf die Vermeidung von patentierten Algorithmen und wieder andere wollten PGP in eine kommerzielle Closed-Source-Software umwandeln. Als Folge davon hat PGP, trotz der leichten Liberalisierung des Waffenexportgesetzes (andernfalls hätten auch Produkte, die AES verwenden, nicht exportiert werden dürfen) und des seit September 2000 abgelaufenen RSA-Patents, damit zu kämpfen, dass verschiedene inkompatible Versionen von PGP unter verschiedenen Namen in Umlauf sind. Die nachfolgende Erörterung behandelt das klassische PGP, d.h. die älteste und einfachste Version. Eine andere populäre Version, Open PGP, wird in RFC 2440 beschrieben. Eine weitere ist GNU Privacy Guard.

PGP bedient sich absichtlich bestehender kryptografischer Algorithmen, anstatt neue zu erfinden. Es basiert größtenteils auf Algorithmen, die bereits umfassend von Experten geprüft wurden und nicht von einer Regierungsbehörde mit dem Ziel der Abschwächung entworfen und beeinflusst wurden. Für regierungskritische Benutzer ist dies ein großes Plus.

PGP unterstützt Textkomprimierung, Geheimhaltung und digitale Signaturen und bietet umfangreiche Schlüsselverwaltungsfunktionen, seltsamerweise aber keine E-Mail-Unterstützung. Es funktioniert eher wie ein Präprozessor, der Klartext als Eingabe nimmt und als Ausgabe unterzeichneten Chiffertext in base64-codierter Form erzeugt. Diese Ausgabe kann dann natürlich per E-Mail verschickt werden. Einige PGP-Implementierungen rufen als letzten Schritt einen Benutzeragenten auf, der für das eigentliche Senden der Nachricht zuständig ist.

Um zu sehen, wie PGP funktioniert, betrachten wir das Beispiel in ► Abbildung 8.44. Hier möchte Alice eine unterschriebene Nachricht P im Klartext sicher an Bob senden. Alice und Bob haben private (D_X) und öffentliche (E_X) RSA-Schlüssel. Nehmen

wir an, dass beide den öffentlichen Schlüssel des jeweils anderen kennen. Auf die PGP-Schlüsselverwaltung wird später ausführlich eingegangen.

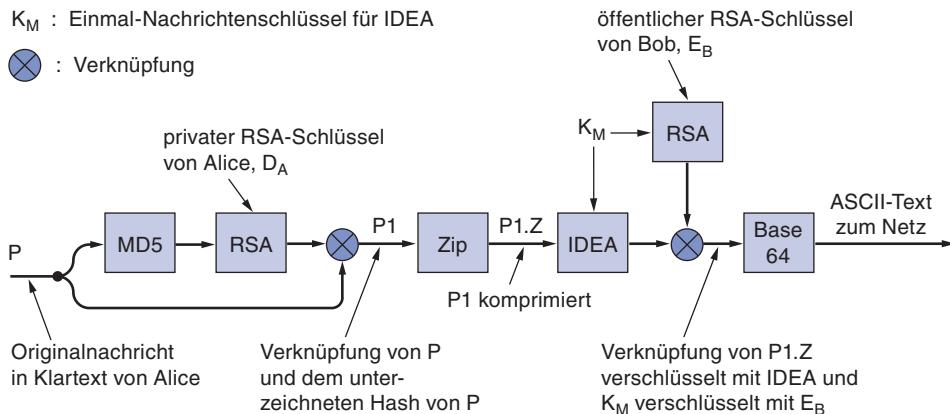


Abbildung 8.44: PGP beim Senden einer Nachricht.

Alice beginnt durch Starten des PGP-Programms auf ihrem Rechner. PGP berechnet für ihre Nachricht P zuerst den MD5-Hashwert und verschlüsselt diesen mit Alices privaten RSA-Schlüssel D_A . Wenn Bob die Nachricht erhält, kann er den Hashwert mit dem öffentlichen Schlüssel von Alice entschlüsseln und prüfen. Sollte auf dieser Stufe eine dritte Person (z.B. Trudy) den Hash abfangen und ihn mit dem bekannten öffentlichen Schlüssel von Alice entschlüsseln, wäre MD5 trotzdem so stark, dass es rechnerisch unmöglich ist, mit dem gleichen MD5-Hash eine andere Nachricht zu produzieren.

Der verschlüsselte Hash und die Originalnachricht werden zu einer Nachricht P_1 verkettet und mit dem ZIP-Programm komprimiert, das auf dem Ziv-Lempel-Algorithmus (Ziv und Lempel, 1977) basiert. Wir nennen die Ausgabe dieses Schrittes $P_{1.Z}$.

Als Nächstes fordert PGP Alice auf, Zufallszeichen einzugeben. Sowohl der Inhalt als auch die Tippgeschwindigkeit werden herangezogen, um einen 128 Bit großen IDEA-Nachrichtenschlüssel K_M zu erzeugen. (In der PGP-Literatur wird dieser Schlüssel Sitzungsschlüssel genannt, was ein wenig irreführend ist, da es keine Sitzung gibt.) K_M wird benutzt, um $P_{1.Z}$ mit IDEA im Cipher-Feedback-Modus zu verschlüsseln. Darüber hinaus wird K_M mit Bobs öffentlichem Schlüssel E_B verschlüsselt. Diese zwei Komponenten werden dann verkettet und in base64 konvertiert (siehe den Abschnitt über MIME in Kapitel 7). Die sich daraus ergebende Nachricht enthält nur Buchstaben, Ziffern und die Zeichen +, / und =, was bedeutet, dass sie in den Body-Teil einer RFC-822-Nachricht eingefügt werden kann. Man kann somit davon ausgehen, dass sie unverändert ankommt.

Bob erhält die Nachricht, kehrt die base64-Codierung um und entschlüsselt den IDEA-Schlüssel mit seinem privaten RSA-Schlüssel. Mithilfe dieses Schlüssels entschlüsselt er die Nachricht und erhält $P_{1.Z}$. Nach der Dekomprimierung trennt Bob den Klartext vom verschlüsselten Hash und entschlüsselt das Hash mit dem öffentlichen Schlüssel von Alice. Stimmt das Klartext-Hash mit seiner eigenen MD5-Berechnung überein, weiß er, dass P die richtige Nachricht ist und von Alice stammt.

Man beachte, dass RSA hier nur an zwei Stellen benutzt wird: zum Verschlüsseln des 128 Bit großen MD5-Hash und zum Verschlüsseln des gleich großen IDEA-Schlüssels. RSA ist zwar langsam, muss aber hier nur eine mäßige Datenmenge von 256 Bit verschlüsseln. Des Weiteren sind alle 256 Klartext-Bits extrem zufällig, sodass Trudy viel Arbeit hätte, den richtigen Schlüssel herauszubekommen. Der aufwendige Teil der Verschlüsselung wird von IDEA durchgeführt, das um einige Größenordnungen schneller ist als RSA. PGP bietet also Sicherheit, Komprimierung und eine digitale Signatur – und das alles effizienter als bei dem in Abbildung 8.19 aufgezeigten Schema.

PGP unterstützt vier RSA-Schlüssellängen. Die Wahl der geeigneten Länge liegt beim Benutzer. Die Längen sind wie folgt:

- 1. Casual (384 Bit):** Kann einfach aufgebrochen werden.
- 2. Commercial (512 Bit):** Kann von Organisationen, die mit drei Buchstaben abgekürzt werden, geknackt werden.
- 3. Military (1 024 Bit):** Kann von keinem auf der Welt geknackt werden.
- 4. Alien (2 048 Bit):** Kann von keinem in unserer Galaxie geknackt werden.

Da RSA nur für zwei kleine Berechnungen benutzt wird, sollte für jede Mail die Alien-Schlüssellänge gewählt werden.

Das Format einer PGP-Nachricht ist in ►Abbildung 8.45 dargestellt. Daneben gibt es noch zahlreiche andere Formate. Die Nachricht besteht aus drei Teilen: dem IDEA-Schlüssel, der Signatur und der Nachricht. Der Schlüsselteil enthält nicht nur den Schlüssel, sondern auch einen Schlüsselidentifikator, da Benutzer mehrere öffentliche Schlüssel haben können.

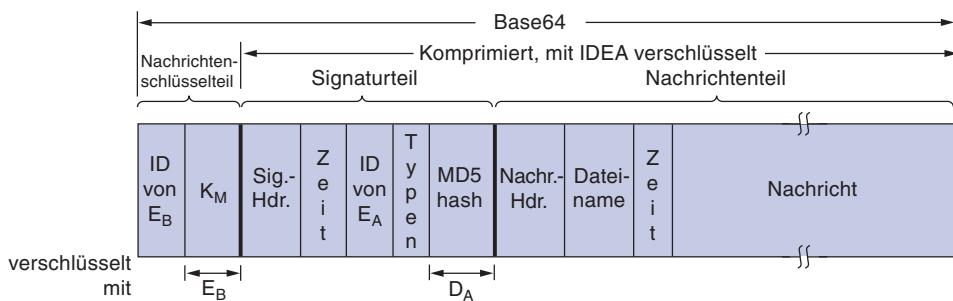


Abbildung 8.45: Eine PGP-Nachricht.

Der Signaturteil enthält einen Header, mit dem wir uns hier nicht befassen. Dem Header folgt ein Zeitstempel, der Identifikator für den öffentlichen Schlüssel des Senders, mit dem der Signatur-Hash entschlüsselt werden kann, einige Typinformationen zu den benutzten Algorithmen (sodass irgendwann einmal MD6 und RSA2 benutzt werden können, sofern sie erfunden werden) und der verschlüsselte Hash.

Der Nachrichtenteil enthält ebenfalls einen Header sowie den Standardnamen der Datei, der verwendet wird, wenn der Empfänger die Datei auf Festplatte schreibt, einen Zeitstempel für die Erstellung der Nachricht und schließlich die Nachricht selbst.

Der Schlüsselverwaltung wurde in PGP viel Aufmerksamkeit gewidmet, da dies die Achillesferse aller Sicherheitssysteme ist. Die Schlüsselverwaltung funktioniert wie folgt: Jeder Benutzer verwaltet lokal zwei Datenstrukturen: einen privaten und einen öffentlichen Schlüsselbund. Der **private Schlüsselbund** (*private key ring*) enthält ein oder mehrere persönliche Schlüsselpaare, die aus einem privaten und einem öffentlichen Schlüssel bestehen. Durch die Unterstützung mehrerer Paare kann der Benutzer seine öffentlichen Schlüssel periodisch oder bei Verdacht auf Missbrauch ändern, ohne dass aktuelle Nachrichten (bei der Übertragung oder in Bearbeitung) dadurch ungültig werden. Jedes Paar hat einen Identifikator. Auf diese Weise kann der Sender einer Nachricht dem Empfänger mitteilen, welcher öffentliche Schlüssel zum Verschlüsseln der Nachricht benutzt wurde. Nachrichtenidentifikatoren bestehen aus den 64 niederwertigen Bits des öffentlichen Schlüssels. Die Benutzer sind dafür verantwortlich, dass bei ihren öffentlichen Schlüsselidentifikatoren keine Konflikte entstehen. Die privaten Schlüssel auf Platte werden mit einem speziellen (beliebig langen) Passwort verschlüsselt, um sie vor Attacken zu schützen.

Der **öffentliche Schlüsselbund** (*public key ring*) eines Benutzers enthält die öffentlichen Schlüssel seiner Gesprächspartner. Sie werden benötigt, um die Nachrichtenschlüssel, die zu einer Nachricht gehören, zu verschlüsseln. Jeder Eintrag im öffentlichen Schlüsselbund enthält nicht nur den öffentlichen Schlüssel, sondern auch seinen 64-Bit-Identifikator und einen Hinweis darauf, wie stark das Vertrauen des Benutzers in den Schlüssel ist.

Hierbei stellt sich folgendes Problem: Nehmen wir an, dass öffentliche Schlüssel in einem sogenannten „Schwarzen Brett“ verwaltet werden. Eine Möglichkeit für Trudy, Bobs geheime E-Mail zu lesen, wäre eine Attacke auf das Schwarze Brett und das Ersetzen von Bobs öffentlichem Schlüssel durch einen ihrer Wahl. Holt sich Alice später den vermeintlich Bob gehörenden Schlüssel, kann Trudy einen Bucket-Brigade-Angriff auf Bob starten.

Um solche Angriffe zu verhindern oder zumindest deren Folgen einzugrenzen, muss Alice wissen, wie weit sie dem Element namens „Bobs Schlüssel“ in ihrem eigenen öffentlichen Schlüsselbund vertrauen kann. Wenn Bob ihr den Schlüssel persönlich auf einer Diskette übergeben hat, kann sie von hoher Sicherheit ausgehen. Es ist dieser dezentralisierte, benutzergesteuerte Ansatz für die Verwaltung öffentlicher Schlüssel, der PGP vor zentralisierten PKI-Schemata auszeichnet.

Dennoch besorgen sich Benutzer manchmal öffentliche Schlüssel von einem vertrauenswürdigen Schlüsselserver. Aus diesem Grund unterstützte PGP nach der Standardisierung von X.509 neben dem traditionellen Schlüsselbundverfahren von PGP auch diese Zertifikate. Inzwischen bieten alle aktuellen Versionen von PGP Unterstützung für X.509.

8.8.2 S/MIME

Das Engagement der IETF im Bereich E-Mail-Sicherheit trägt die Bezeichnung **S/MIME** (*Secure/MIME*) und wird in den RFCs 2632 bis 2643 beschrieben. Es bietet Authentifizierung, Datenintegrität, Geheimhaltung und Nichtabstrebbarkeit. Darüber hinaus ist es ziemlich flexibel und unterstützt verschiedene Verschlüsselungsalgorithmen. Die Bezeichnung S/MIME lässt darauf schließen, dass es sich gut in MIME integrieren lässt und die verschiedenen Nachrichtentypen schützt. Es wurden beispielsweise mehrere neue MIME-Header definiert, die der Aufnahme digitaler Signaturen dienen.

S/MIME weist keine feste Zertifikathierarchie auf, die mit einer einzigen Wurzel beginnt – was eines der politischen Probleme war, die ein früheres System namens PEM (*Privacy Enhanced Mail*) zum Scheitern verurteilte. Stattdessen können die Benutzer mehrere Sicherheitsanker verwenden. Solange ein Zertifikat zu einem bestimmten Sicherheitsanker zurückverfolgt werden kann, dem der Benutzer vertraut, wird es als gültig betrachtet. S/MIME verwendet Standardalgorithmen und -protokolle, die wir bereits behandelt haben, sodass wir hier nicht weiter darauf eingehen. Weitere Informationen finden Sie in den RFCs.

8.9 Sicherheit im Web

Wir haben gerade zwei wichtige Bereiche erörtert, in denen Sicherheit benötigt wird: Kommunikation und E-Mail. Sie können sich diese als Suppe und Vorspeise vorstellen. Nun kommt das Hauptgericht: Sicherheit im Web. Das Web ist der Ort, in dem sich heutzutage die meisten Truds tummeln und ihre Schurkereien aushecken. In den folgenden Abschnitten betrachten wir einige der Probleme und Themen im Zusammenhang mit der Sicherheit im Web.

Die Sicherheit im Web kann grob in drei Teile unterteilt werden. Erstens, wie werden die Objekte und Ressourcen sicher bezeichnet? Zweitens, wie können sichere und authentifizierte Verbindungen errichtet werden? Drittens, was passiert, wenn eine Website einem Client ein Stück ausführbaren Code schickt? Nachdem wir uns einige Bedrohungen näher angesehen haben, untersuchen wir diese Fragen.

8.9.1 Bedrohungen der Sicherheit

Man liest beinahe wöchentlich über Probleme mit der Sicherheit von Websites. Die Lage sieht wirklich ziemlich düster aus. Betrachten wir exemplarisch ein paar Vorfälle. Erstens gibt es zahlreiche Unternehmen, deren Homepage von Crackern angegriffen und durch eine neue Homepage ersetzt wurde. (Die Boulevardpresse bezeichnet Leute, die sich in Computer „einhacken“, oftmals als „Hacker“. Viele Programmierer reservieren diese Bezeichnung jedoch für großartige Programmierer. Wir bezeichnen diese Eindringlinge deshalb lieber als „Cracker“.) Zu den gecrackten Websites gehören die von Yahoo, der U.S. Army, des CIA, der NASA und der *New York Times*. In den meisten Fällen haben Cracker nur einen witzigen Spruch hinterlassen, und die Sites waren innerhalb weniger Stunden wieder repariert.

Betrachten wir nun die schwerwiegenderen Fälle. Unzählige Sites wurden durch Denial-of-Service-Angriffe lahmgelegt, wobei der Cracker die Site mit so viel Verkehr flutet, dass sie nicht mehr auf normale Anfragen reagieren kann. Oftmals wird dieser Angriff von sehr vielen Rechnern gestartet, die der Cracker bereits aufgebrochen hat (DDoS-Angriffe). Diese Angriffe sind so häufig, dass sie keine Nachricht mehr wert sind, aber der von ihnen verursachte Geschäftsausfall kann den angegriffenen Sites Tausende von Euros kosten.

1999 brach ein schwedischer Cracker in die Website von Microsoft Hotmail ein und erzeugte eine gespiegelte Site, die jedem, der den Namen eines Hotmail-Benutzers eingab, erlaubte, die aktuellen und archivierten E-Mails des betreffenden Benutzers zu lesen.

In einem anderen Fall brach ein 19-jähriger Russe namens Maxim in eine E-Commerce-Website ein und stahl 300 000 Kreditkartennummern. Dann wandte er sich an die Besitzer der Site und teilte ihnen mit, dass er alle Kreditkartennummern ins Internet stellen werde, wenn sie ihm nicht 100 000 US-Dollar zahlten. Sie gaben dieser Erpressung nicht nach und er stellte tatsächlich die Kreditkartennummern ins Internet und fügte so unschuldigen Opfern schweren Schaden zu.

In einem anderen Fall schickte ein 23-jähriger Student aus Kalifornien eine Pressemeldung an eine Nachrichtenagentur, die fälschlicherweise behauptete, dass die Emulex Corporation einen riesigen Quartalsverlust verbuchen würde und der CEO mit sofortiger Wirkung zurücktrete. Innerhalb von ein paar Stunden fiel die Aktie des Unternehmens um 60%, sodass Aktionäre über 2 Milliarden US-Dollar verloren. Der Eindringling machte einen Gewinn von einer Viertelmillion US-Dollar, indem er seine Optionen auf die Aktien kurz vor dem Absenden der Ankündigung verkaufte. Wenn dieses Ereignis auch kein Einbruch in eine Website war, so ist doch klar, dass eine solche Ankündigung auf die Homepage eines großen Unternehmens eine ähnliche Wirkung hätte.

Wir könnten diese Beispiele (leider) endlos fortsetzen. Doch ist es nun an der Zeit, die technischen Fragen in Zusammenhang mit der Sicherheit im Web zu erörtern. Weitere Informationen zu den verschiedensten Sicherheitsproblemen finden Sie in Anderson (2008a), Stuttard und Pinto (2007) und Schneier (2004). Außerdem wird eine Suche im Internet Ihnen eine Flut weiterer ähnlicher Fälle liefern.

8.9.2 Sichere Namensvergabe

Beginnen wir mit etwas sehr Grundlegendem: Alice möchte die Website von Bob besuchen. Sie gibt die URL von Bob in ihren Browser ein und ein paar Sekunden später wird die Webseite angezeigt. Aber ist es die von Bob? Vielleicht. Auch hier könnte Trudy wieder am Werk sein. Sie könnte beispielsweise alle ausgehenden Pakete von Alice abfangen und sie untersuchen. Wenn sie eine GET-HTTP-Anfrage abfängt, die die Webseite von Bob anfordert, könnte sie selbst zur Website von Bob gehen, sich die Seite holen und diese nach Belieben modifizieren und die gefälschte Seite an Alice zurücksenden. Alice würde dies nicht merken. Noch schlimmer: Trudy könnte die Preise im E-Store von Bob senken, damit seine Produkte sehr attraktiv aussehen, und Alice dazu bringen, ihre Kreditkartennummer an „Bob“ zu senden, um etwas zu erwerben.

Ein Nachteil dieses klassischen Man-in-the-Middle-Angriffs ist, dass Trudy in der Lage sein muss, den ausgehenden Datenverkehr von Alice abzufangen und den eingehenden Datenverkehr zu fälschen. In der Praxis muss sie entweder die Telefonleitung von Alice oder von Bob anzapfen, da Glasfaser-Backbones hierfür viel zu kompliziert sind. Doch auch wenn ein aktives Anzapfen der Leitung möglich ist, so ist es doch ziemlich arbeitsaufwendig, und Trudy ist zwar clever, aber faul. Darüber hinaus gibt es einfachere Wege, Alice auszutricksen.

DNS-Spoofing

Angenommen, Trudy kann das DNS-System oder auch nur den DNS-Cache bei Alices ISP knicken und die IP-Adresse von Bob (sagen wir 36.1.2.3) mit ihrer (Trudys) IP-Adresse (sagen wir 42.9.9.9) ersetzen. Das führt zu einem Angriff, wie er in ▶ Abbildung 8.46a dargestellt ist: In diesem Szenario fordert Alice (1) vom DNS die IP-Adresse von Bob an, (2) erhält diese, (3) fragt Bob nach seiner Homepage und (4) erhält auch diese. Nachdem Trudy den DNS-Datensatz von Bob durch ihre eigene IP-Adresse ersetzt hat, erhalten wir die Situation in ▶ Abbildung 8.46b. Wenn Alice nun die Adresse von Bob nachschlägt, erhält sie die von Trudy, sodass der gesamte an Bob gerichtete Datenverkehr an Trudy geht. Trudy kann nun einen Man-in-the-Middle-Angriff starten, ohne eine Telefonleitung anzapfen zu müssen. Stattdessen muss sie nur in den DNS-Server einbrechen und einen Datensatz ändern – was sehr viel einfacher ist.

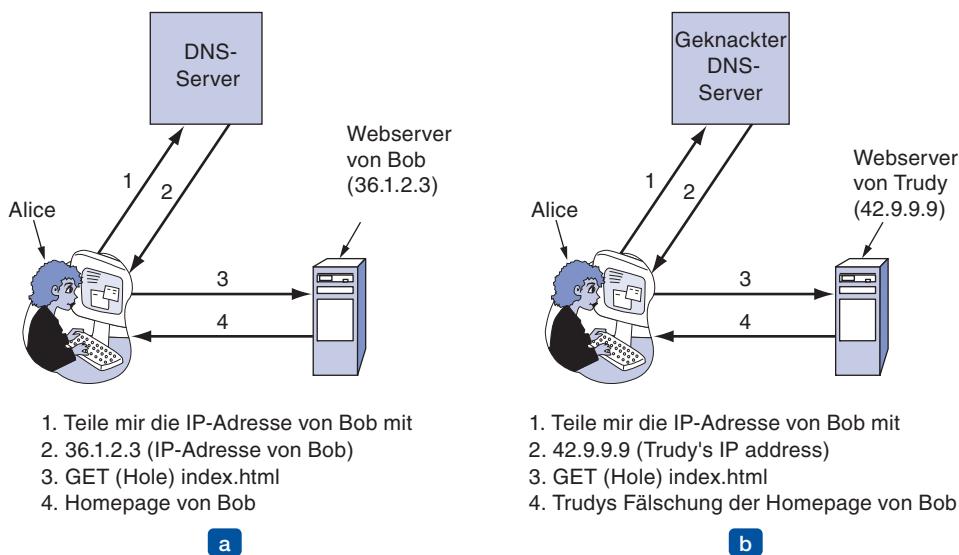


Abbildung 8.46: (a) Normale Situation. (b) Ein Angriff, bei dem das DNS aufgebrochen und der Datensatz von Bob verändert wird.

Wie kann Trudy das DNS täuschen? Wie sich herausstellt, ist dies relativ einfach. Kurz zusammengefasst kann Trudy den DNS-Server beim ISP von Alice dazu verleiten, eine Anfrage nach Bobs Adresse auszusenden. Da das DNS leider UDP verwendet, kann der DNS-Server nicht wirklich überprüfen, wer die Antwort bereitgestellt hat. Trudy kann diese Eigenschaft für sich nutzen, indem sie die erwartete Antwort fälscht und

somit eine falsche IP-Adresse in den Cache des DNS-Servers einspeist. Der Einfachheit halber gehen wir davon aus, dass der ISP von Alice anfangs keinen Eintrag für Bobs Website, *bob.com*, hat. Wenn er einen hat, kann Trudy warten, bis ein Timeout auftritt (oder es mit anderen Tricks versuchen).

Trudy startet ihren Angriff, indem sie eine Nachschlageanforderung an den ISP von Alice sendet, mit der sie nach der IP-Adresse von *bob.com* fragt. Da hier kein Eintrag für diesen DNS-Namen vorhanden ist, fragt der Cache-Server beim Top-Level-Server für die *.com*-Domäne an, ob dieser einen solchen Eintrag gespeichert hat. Doch bevor der *.com*-Server antworten kann, ist Trudy ihm schon zuvorgekommen und hat folgende falsche Antwort zurückgesendet: „*bob.com* ist 42.9.9.9“, wobei sie ihre IP-Adresse angibt. Wenn ihre falsche Antwort zuerst beim ISP von Alice ankommt, wird diese im Cache abgelegt, und die echte Antwort wird als unaufgeforderte Antwort auf eine Anfrage, die nicht länger vorhanden ist, zurückgewiesen. Das Ausricksen eines DNS-Servers, um eine falsche IP-Adresse zu installieren, wird als **DNS-Spoofing** (DNS-Manipulation) bezeichnet. Ein Cache, der falsche, manipulierte IP-Adressen enthält, wird **vergifteter Cache** (*poisoned cache*) genannt.

Tatsächlich sind aber die Dinge nicht so einfach. Als Erstes prüft der ISP von Alice, ob die Antwort die korrekte IP-Adresse des Top-Level-Servers enthält. Da Trudy aber alle beliebigen Informationen in das IP-Feld eintragen kann, kann sie diesen Test leicht bestehen, da die IP-Adressen der Top-Level-Server öffentlich bekannt sind.

Zweitens: damit DNS-Server erkennen, welche Antwort zu welcher Anforderung gehört, tragen alle Anforderungen eine Folgennummer. Um den ISP von Alice zu täuschen, muss Trudy die aktuelle Sequenznummer kennen. Am einfachsten kann Trudy diese in Erfahrung bringen, wenn sie selbst eine Domäne registriert, wie beispielsweise *trudy-the-intruder.com*. Nehmen wir an, dass deren IP-Adresse auch 42.9.9.9 ist. Sie erstellt auch einen DNS-Server für die neue Domäne *dns.trudy-the-intruder.com*. Auch dieser hat die IP-Adresse von Trudy, 42.9.9.9, da Trudy nur einen Rechner hat. Nun muss sie den ISP von Alice über ihren DNS-Server informieren. Das ist auch einfach. Alles, was sie tun muss, ist, den ISP von Alice nach *foobar.trudy-the-intruder.com* zu fragen, wodurch der ISP von Alice durch Anfrage beim Top-Level-*.com*-Server herausfindet, wer Trudys neue Domäne bedient.

Ist *dns.trudy-the-intruder.com* sicher im Cache von Alices ISP, dann kann der eigentliche Angriff beginnen. Trudy fragt nun den ISP von Alice nach *www.trudy-the-intruder.com*. Der ISP sendet natürlich eine Anfrage an Trudys DNS-Server. Diese Anfrage enthält die Sequenznummer, nach der Trudy sucht. Schnell wie der Wind bittet Trudy den ISP von Alice, Bob nachzuschlagen. Sie beantwortet sofort ihre eigene Frage, indem sie dem ISP eine gefälschte Nachricht, angeblich vom Top-Level-*.com*-Server, sendet: „*bob.com* ist 42.9.9.9“. Diese gefälschte Antwort enthält eine Sequenznummer, die um eins höher ist als die, die sie gerade erhalten hat. Sie kann hier auch eine zweite Fälschung mit einer um zwei höheren Sequenznummer senden, eventuell auch ein Dutzend mehr, wobei sie die Sequenznummern erhöht. Eine davon wird passen. Der Rest wird verworfen. Wenn die gefälschte Antwort bei Alice ankommt, wird sie im Cache abgelegt. Wenn die echte Antwort später eingeht, wird sie zurückgewiesen, weil keine Anfrage aussteht.

Wenn nun Alice `bob.com` nachschlägt, wird ihr mitgeteilt, 42.9.9.9, die Adresse von Trudy, zu verwenden. Trudy hat ganz bequem vom eigenen Wohnzimmer aus einen erfolgreichen Man-in-the-Middle-Angriff gestartet. Die hierzu erforderlichen Schritte sind in ►Abbildung 8.47 dargestellt. Diese spezielle Angriffsart kann abgewehrt werden, wenn DNS-Server zufällige Identifikatoren in ihren Anforderungen verwenden, anstatt nur Sequenznummern hochzuzählen, aber es scheint, dass jedes Mal, wenn ein Loch gefüllt ist, sich ein anderes auftut. Vor allem ist von Nachteil, dass die Identifikatoren nur 16 Bit groß sind, sodass man problemlos alle Zahlen durchgehen kann, vor allem wenn dafür ein Computer eingesetzt wird.

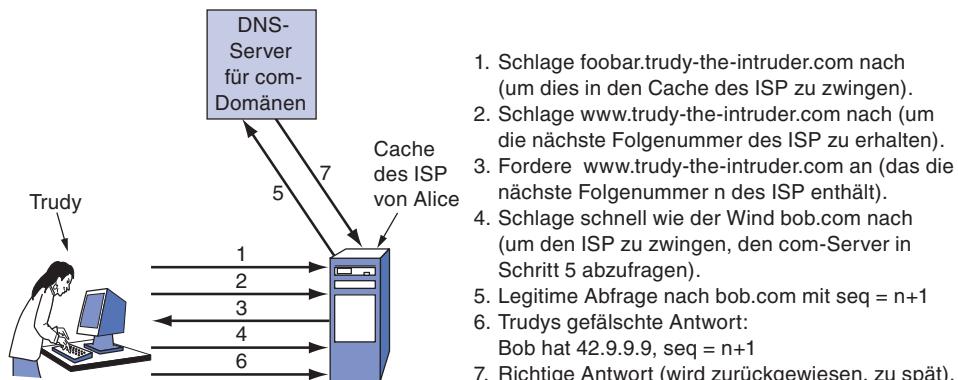


Abbildung 8.47: Wie Trudy den ISP von Alice täuscht.

Sicheres DNS

Das eigentliche Problem ist, dass das DNS zu einer Zeit entwickelt wurde, als das Internet nur einigen Hundert Universitäten zu Recherchezwecken diente, und weder Alice, noch Bob oder Trudy durften hier mitmachen. Sicherheit war zu dieser Zeit kein Thema. Wichtig war, dass das Internet gut funktionierte. Diese Situation hat sich im Laufe der Jahre radikal geändert, sodass im Jahre 1994 die IETF eine Arbeitsgruppe einrichtete, deren Aufgabe es war, das DNS grundlegend sicher zu machen. Das (noch laufende) Projekt wurde unter dem Namen **DNSsec** (*DNS Security, DNS-Sicherheit*) bekannt; seine Ergebnisse sind in RFC 2535 zusammengefasst. Leider wird DNSsec noch nicht überall eingesetzt, sodass unzählige DNS-Server immer noch mit Spoofing-Angriffen verwundbar sind.

DNSsec ist von der Konzeption her sehr einfach. Es basiert auf der Verschlüsselung mit öffentlichen Schlüsseln. Jede DNS-Zone (im Sinn von Abbildung 7.5) hat ein Paar privater/öffentlicher Schlüssel. Jede von einem DNS-Server gesendete Information ist mit dem privaten Schlüssel der Ausgangszone signiert, sodass der Empfänger die Authentizität verifizieren kann. DNSsec bietet drei fundamentale Dienste:

1. Nachweis, woher die Daten stammen.
2. Verteilung von einem öffentlichen Schlüssel.
3. Authentifizierung der Transaktionen und Anforderungen.

Der Hauptdienst ist der erste, der verifiziert, dass die zurück erhaltenen Daten vom Besitzer der Zone genehmigt wurden. Der zweite ist beim sicheren Speichern und Abrufen öffentlicher Schlüssel wichtig. Der dritte ist erforderlich, um Playback- und Spoofing-Angriffe abzuwehren. Beachten Sie, dass Geheimhaltung nicht als Dienst angeboten wird, weil alle Informationen im DNS als öffentlich betrachtet werden. Da man davon ausgeht, dass die Einführung von DNSsec mehrere Jahre dauert, ist die Fähigkeit, dass Server mit hohem Sicherheitsstandard mit Servern mit niedrigem Sicherheitsstandard zusammenarbeiten, von grundlegender Bedeutung. Dies impliziert, dass das Protokoll nicht geändert werden kann. Sehen wir uns einige Details etwas genauer an.

DNS-Datensätze werden in Gruppen namens **RRSets** (*Resource Record Set*) zusammengefasst, wobei alle Datensätze mit gleichem Namen, gleicher Klasse und gleichem Typ eine Gruppe bilden. Ein RRSet kann mehrere Datensätze *A* enthalten, beispielsweise wenn ein DNS-Name auf eine primäre IP-Adresse und eine sekundäre IP-Adresse abgebildet wird. Die RRSets werden um einige neue Datensatztypen erweitert (im Folgenden erläutert). Aus jedem RRSet wird kryptografisch ein Hash erzeugt (z.B. mit SHA-1). Der Hash wird mit dem privaten Schlüssel der Zone unterzeichnet (z.B. mit RSA). Die an die Clients gerichtete Übertragungseinheit ist das unterzeichnete RRSet. Bei Empfang eines unterzeichneten RRSets kann der Client verifizieren, ob dies mit dem privaten Schlüssel der Ausgangszone unterzeichnet wurde. Stimmt die Signatur, werden die Daten akzeptiert. Da jedes RRSet eine eigene Signatur umfasst, können RRSets überall zwischengespeichert werden, selbst auf nicht vertrauenswürdigen Servern, ohne die Sicherheit zu gefährden.

DNSsec führt verschiedene neue Datensatztypen ein. Der erste ist der Datensatz *KEY*. Dieser Datensatz enthält den öffentlichen Schlüssel einer Zone, eines Benutzers, Hosts oder anderen Principals, den zur Unterzeichnung verwendeten kryptografischen Algorithmus und das zur Übertragung verwendete Protokoll sowie ein paar weitere Bits. Der öffentliche Schlüssel wird unverschlüsselt gespeichert. X.509-Zertifikate werden aufgrund ihrer Größe nicht verwendet. Im Algorithmusfeld steht eine 1 für die MD5/RSA-Signaturen (die bevorzugte Wahl) beziehungsweise andere Werte für andere Kombinationen. Das Protokollfeld kann die Verwendung von IPSec oder einem anderen Sicherheitsprotokoll, falls verwendet, angeben.

Der zweite neue Datensatztyp ist *SIG*. Dieser enthält den signierten Hash gemäß dem Algorithmus, der im Datensatz *KEY* angegeben wurde. Die Signatur gilt für alle Datensätze im RRSet, einschließlich jeglicher vorhandener *KEY*-Datensätze, sich selbst aber ausgeschlossen. Er enthält auch die Zeitangaben, wann die Gültigkeit der Signatur beginnt und abläuft, sowie den Namen des Unterzeichners und ein paar weitere Angaben.

Der Entwurf von DNSsec sieht vor, dass der private Schlüssel einer Zone offline gehalten werden kann. Ein- oder zweimal pro Tag können die Datenbanken der Zone manuell (z.B. auf CD-ROM) auf einen nicht ans Netz angeschlossenen Rechner übertragen werden, auf dem sich der private Schlüssel befindet. Alle RRSets können dort signiert und die so erzeugten *SIG*-Datensätze auf den primären Server der Zone über CD-ROM zurücktransportiert werden. Auf diese Weise kann der private Schlüssel auf einer CD-

ROM gespeichert werden, die in einem Safe aufbewahrt wird, außer wenn sie in den nicht angeschlossenen Rechner zur Unterzeichnung der täglich neuen RRSets eingelegt wird. Nach der Signatur werden alle Kopien des Schlüssels aus dem Speicher und von der Platte gelöscht und die CD-ROM wieder in den Safe zurückgelegt. Dieses Verfahren reduziert die elektronische Sicherheit auf die physische Sicherheit, da diese für viele besser nachvollziehbar ist.

Das Vorunterzeichnen von RRSets beschleunigt die Beantwortung von Anforderungen erheblich, da keine Verschlüsselung während der Anforderung vorgenommen werden muss. Der Nachteil ist, dass sehr viel Festplattenspeicher erforderlich ist, um alle Schlüssel und Signaturen in den DNS-Datenbanken zu speichern. Einige Datensätze können sich durch eine Signatur um das Zehnfache vergrößern.

Erhält ein Client-Prozess ein unterzeichnetes RRSet, muss er den öffentlichen Schlüssel der Ursprungszone anwenden, um den Hash zu entschlüsseln, den Hash selbst berechnen und die beiden Werte vergleichen. Stimmen sie überein, werden die Daten als gültig betrachtet. Diese Prozedur wirft aber die Frage auf, wie der Client den öffentlichen Schlüssel der Zone erhält. Eine Möglichkeit ist, diesen von einem vertrauenswürdigen Server über eine sichere Verbindung (wie über IPsec) zu beziehen.

In der Praxis wird aber erwartet, dass Clients mit den öffentlichen Schlüsseln aller Top-Level-Domänen vorkonfiguriert werden. Wenn Alice nun die Website von Bob besuchen möchte, kann sie das DNS nach dem RRSet von *bob.com* fragen, das seine IP-Adresse und einen *KEY*-Datensatz mit dem öffentlichen Schlüssel von Bob enthält. Dieses RRSet wird von der Top-Level-*com*-Domäne unterzeichnet, sodass Alice die Gültigkeit sehr leicht verifizieren kann. Ein Beispiel der Inhalte dieses RRSets ist in ▶Abbildung 8.48 dargestellt.

| Domänenname | Lebensspanne | Klasse | Typ | Wert |
|-------------|--------------|--------|-----|-------------------------------|
| bob.com. | 86400 | IN | A | 36.1.2.3 |
| bob.com. | 86400 | IN | KEY | 3682793A7B73F731029CE2737D... |
| bob.com. | 86400 | IN | SIG | 86947503A8B848F5272E53930C... |

Abbildung 8.48: Beispiel für ein RRSet für *bob.com*. Der *KEY*-Datensatz ist der öffentliche Schlüssel von Bob. Der *SIG*-Datensatz ist der unterzeichnete Hash des Top-Level-*com*-Servers der Datensätze *A* und *KEY*, um deren Authentizität zu verifizieren.

Mit der verifizierten Kopie von Bobs öffentlichem Schlüssel bewaffnet kann Alice nun den DNS-Server von Bob (von Bob betrieben) nach der IP-Adresse von *www.bob.com* fragen. Dieses RRSet wird mit dem privaten Schlüssel von Bob unterzeichnet, sodass Alice die Signatur auf dem von Bob zurückgegebenen RRSet verifizieren kann. Wenn es Trudy gelingt, ein falsches RRSet in einen der Caches einzuspeisen, kann Alice die fehlende Authentizität entdecken, weil der darin enthaltene *SIG*-Datensatz nicht korrekt ist.

DNSsec bietet auch ein kryptografisches Verfahren, um eine Antwort an eine spezielle Anforderung zu binden, damit Täuschungen der Art, wie Trudy sie in Abbildung 8.47 versucht hat, vermieden werden. Diese (optionale) Maßnahme fügt der Antwort einen Hash der Anforderungsnachricht hinzu, der mit dem privaten Schlüssel des Antwortenden unterzeichnet ist. Da Trudy den privaten Schlüssel des Top-Level-*com*-Servers nicht kennt, kann sie keine Antwort auf eine Anforderung fälschen, die der ISP von Alice dorthin gesendet hat. Selbst wenn Trudy ihre Antwort als Erste zurückerhält, wird diese aufgrund der ungültigen Signatur für die Hash-Anforderung zurückgewiesen.

DNSsec unterstützt auch ein paar andere Datensatztypen. So kann beispielsweise der Datensatz *CERT* dazu verwendet werden, Zertifikate (wie X.509) zu speichern. Dieser Datensatz wurde bereitgestellt, weil viele das DNS in eine PKI umwandeln wollten. Ob dies tatsächlich eintritt, bleibt abzuwarten. Wir beenden unsere Erörterung von DNSsec hier. Weitere Informationen finden Sie im RFC 2535.

8.9.3 SSL – Secure Sockets Layer

Die sichere Benennung ist ein guter Start, aber Websicherheit umfasst noch viel mehr. Der nächste Schritt betrifft sichere Verbindungen. Im Folgenden betrachten wir, wie man sichere Verbindungen verwirklichen kann. Nichts im Zusammenhang mit Sicherheit ist einfach; das gilt auch hierfür.

Als das Web in den Blickpunkt der Öffentlichkeit geriet, wurde es anfangs nur für die Verteilung statischer Seiten eingesetzt. Es dauerte aber nicht lange, bis Unternehmen auf die Idee kamen, das Internet für finanzielle Transaktionen wie Einkäufe mit Kreditkarten, Onlinebanking und Aktienhandel zu nutzen. Diese Anwendungen waren der Grund, dass der Ruf nach sicheren Verbindungen immer lauter wurde. Deswegen führte Netscape Communications Corp, der zu dieser Zeit führende Browseranbieter, 1995 ein Sicherheitspaket namens **SSL** (*Secure Sockets Layer*) ein. Diese Software und das dazugehörige Protokoll sind inzwischen weitverbreitet und werden unter anderem von Firefox, Safari und dem Internet Explorer eingesetzt, sodass es sich lohnt, dass wir uns näher damit befassen.

SSL errichtet eine sichere Verbindung zwischen zwei Sockets. Dies beinhaltet:

- 1.** Verhandlung der Parameter zwischen Client und Server.
- 2.** Gegenseitige Authentifizierung von Client und Server.
- 3.** Geheime Kommunikation.
- 4.** Schutz der Datenintegrität.

Wir haben diese Punkte bereits kennengelernt, sodass wir nicht nochmals darauf eingehen.

Die Positionierung von SSL im Protokollstapel wird in ►Abbildung 8.49 dargestellt. Es wird im Grunde eine neue Schicht zwischen der Anwendungsschicht und der Transportschicht eingefügt, welche Anforderungen vom Browser akzeptiert und diese

an TCP für die Übertragung zum Server übergibt. Ist die sichere Verbindung einmal errichtet, besteht die Hauptaufgabe von SSL in der Komprimierung und Verschlüsselung. HTTP über SSL wird als **HTTPS** (*Secure HTTP*) bezeichnet, obwohl es das HTTP-Standardprotokoll ist. Manchmal ist es auf einem neuen Port (443) anstatt auf dem Standardport (80) verfügbar. Nebenbei bemerkt ist der Einsatz von SSL nicht auf Webbrowser beschränkt, aber dies ist die am meisten genutzte Anwendung. Darüber hinaus bietet es auch gegenseitige Authentifizierung.

- Anwendungsschicht (HTTP)
- Sicherheitsschicht (SSL)
- Transportschicht (TCP)
- Vermittlungsschicht (IP)
- Sicherungsschicht (PPP)
- Bitübertragungsschicht (Modem, ADSL, Kabelfernsehen)

Abbildung 8.49: Schichten (und Protokolle) für einen privaten Benutzer, der mit SSL im Internet browsst.

SSL hat mehrere Versionen durchlaufen. Im Folgenden erörtern wir nur Version 3, da dies die am meisten verwendete Version ist. SSL unterstützt eine Vielzahl von verschiedenen Optionen. Zu diesen Optionen gehören das Vorhandensein oder Fehlen der Komprimierung, die verwendeten kryptografischen Algorithmen und einige Themen im Zusammenhang mit Exportrestriktionen für Kryptografie. Letzteres dient vor allem dazu, dass eine fundierte Verschlüsselung nur verwendet wird, wenn sich beide Endpunkte einer Verbindung in den Vereinigten Staaten befinden. In den anderen Fällen sind die Schlüssel auf 40 Bit beschränkt, was Kryptografen als Witz betrachten. Netscape wurde gezwungen, diese Restriktionen umzusetzen, um eine Exportlizenz von der US-Regierung zu erhalten.

SSL besteht aus zwei Teilprotokollen, eines zur Errichtung einer sicheren Verbindung und eines zu dessen Verwendung. Beginnen wir damit, wie diese sicheren Verbindungen errichtet werden. Das Protokoll für den Verbindungsauflauf wird in ► Abbildung 8.50 dargestellt. Es beginnt mit Nachricht 1, wenn Alice eine Verbindungsanforderung an Bob sendet. Die Anforderung gibt die SSL-Version von Alice und ihre Präferenzen bezüglich der Komprimierung und der kryptografischen Algorithmen an. Sie enthält auch eine Nonce, R_A , die später verwendet wird.

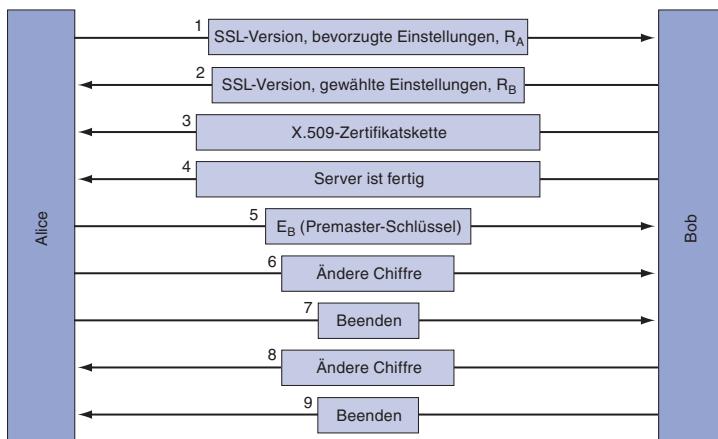


Abbildung 8.50: Eine vereinfachte Version des SSL-Teilprotokolls zum Verbindungsaufbau.

Nun ist Bob an der Reihe. In Nachricht 2 wählt Bob einen Algorithmus aus den verschiedenen, von Alice unterstützten Algorithmen aus und sendet seine eigene Nonce R_B . Dann sendet er in Nachricht 3 ein Zertifikat, das seinen öffentlichen Schlüssel enthält. Für den Fall, dass dieses Zertifikat nicht von einer bekannten Zertifizierungsstelle unterzeichnet ist, wird eine Zertifikatskette mitgeschickt, mit der eine solche Stelle abgeleitet werden kann. Alle Browser einschließlich der von Alice werden bereits mit etwa 100 öffentlichen Schlüsseln geliefert. Wenn also Bob eine Kette errichten kann, die sich zu einem dieser Schlüssel zurückverfolgen lässt, ist Alice in der Lage, den öffentlichen Schlüssel von Bob zu verifizieren. An diesem Punkt kann Bob einige andere Nachrichten senden (wie eine Anforderung nach dem Zertifikat von Alices öffentlichem Schlüssel). Ist Bob fertig, sendet er Nachricht 4, um Alice mitzuteilen, dass sie an der Reihe ist.

Alice antwortet, indem sie einen zufälligen 384-Bit-**Premaster-Schlüssel** (premaster key) wählt und diesen verschlüsselt mit Bobs öffentlichem Schlüssel an Bob schickt (Nachricht 5). Der eigentliche Sitzungsschlüssel zur Verschlüsselung der Daten wird aus dem Premaster-Schlüssel in Kombination mit den Nonces auf komplexe Weise abgeleitet. Nachdem Nachricht 5 empfangen wurde, können sowohl Alice wie auch Bob den Sitzungsschlüssel berechnen. Aus diesem Grund teilt Alice Bob mit, auf die neue Chiffre zu wechseln (Nachricht 6) und dass sie mit dem Teilprotokoll zum Sitzungsaufbau fertig ist (Nachricht 7). Bob bestätigt ihr daraufhin den Empfang dieser Nachrichten (Nachrichten 8 und 9).

Aber obwohl Alice nun weiß, wer Bob ist, weiß Bob nicht, wer Alice ist (außer Alice hat einen öffentlichen Schlüssel und ein entsprechendes Zertifikat, was eher unwahrscheinlich ist für eine Einzelperson). Da kann die erste Nachricht von Bob durchaus die Aufforderung an Alice sein, sich mit einem vorher definierten Benutzernamen und Passwort anzumelden. Das Anmeldeprotokoll liegt aber außerhalb von SSL. Wenn dies einmal ausgeführt wurde, über welche Verfahren auch immer, kann die Datenübertragung beginnen.

Wie oben erwähnt, unterstützt SSL mehrere kryptografische Algorithmen. Der stärkste Algorithmus verwendet Triple DES mit drei getrennten Schlüsseln für die Verschlüsselung und SHA-1 für die Nachrichtenintegrität. Diese Kombination ist relativ langsam, sodass sie meistens für Onlinebanking und andere Anwendungen genutzt wird, bei denen die höchste Sicherheit erforderlich ist. Bei normalen E-Commerce-Anwendungen wird RC4 mit einem 128-Bit-Schlüssel zur Verschlüsselung und MD5 zur Nachrichtenauthentifizierung verwendet. RC4 nimmt den 128-Bit-Schlüssel als Ausgangsbasis und erweitert ihn zur internen Verwendung auf eine viel größere Zahl. Dann erzeugt er mit dieser internen Zahl einen Schlüsselstrom. Der Schlüsselstrom wird mit XOR mit dem Klartext verknüpft, um eine klassische Stromchiffre bereitzustellen, wie in Abbildung 8.14 dargestellt wird. Die Exportversionen verwenden ebenfalls RC4 mit 128-Bit-Schlüsseln, von denen jedoch 88 Bit öffentlich sind, um die Chiffre leicht aufbrechbar zu machen.

Bei dem eigentlichen Transport wird ein zweites Teilprotokoll verwendet, wie in Abbildung 8.51 dargestellt. Die Nachrichten aus dem Browser werden zuerst in bis zu 16 KB große Teile zerlegt. Falls Komprimierung aktiviert ist, wird jede Einheit einzeln komprimiert. Danach wird der geheime Schlüssel aus den beiden Nonces und dem Premaster-Schlüssel mit dem komprimierten Text verkettet und aus dem Ergebnis mit dem vereinbarten Hashing-Algorithmus (in der Regel MD5) ein Hash gebildet. Dieser Hash wird an jedes Fragment als MAC angehängt. Das komprimierte Fragment plus MAC wird dann mit einem vereinbarten symmetrischen Verschlüsselungsalgorithmus verschlüsselt (in der Regel durch eine XOR-Verknüpfung mit dem RC4-Schlüsselstrom). Schließlich wird ein Fragment-Header angehängt und das Fragment über die TCP-Verbindung übertragen.

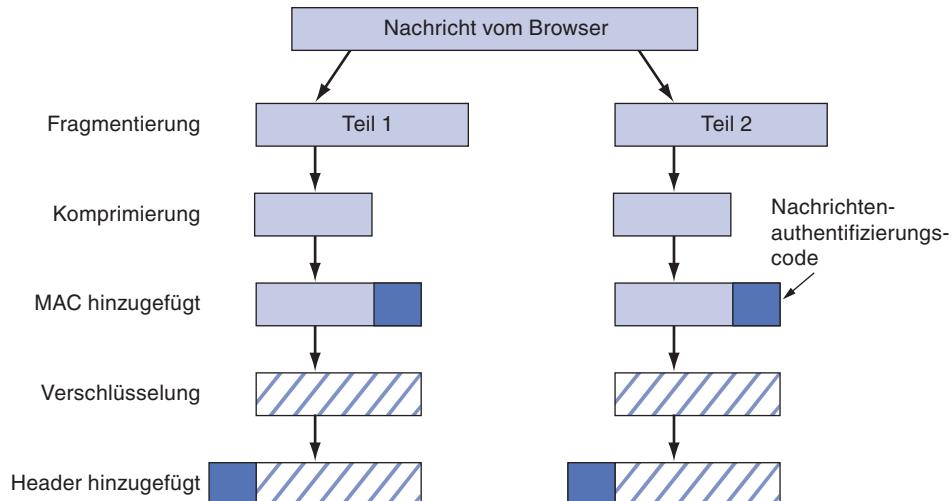


Abbildung 8.51: Datenübertragung mit SSL.

Allerdings ist ein Wort der Vorsicht angebracht. Da bekannt ist, dass RC4 einige schwache Schlüssel hat, die einfach aufgedeckt werden können, steht die Sicherheit

bei SSL mit RC4 auf wackligen Beinen (Fluhrer et al., 2001). Browser, in denen der Benutzer die Chiffre-Suite auswählen kann, sollten per Konfiguration immer Triple DES mit 168-Bit-Schlüsseln und SHA-1 verwenden, selbst wenn diese Konfiguration langsamer ist als RC4 und MD5. Oder noch besser, die Benutzer sollten ein Browser-Upgrade verwenden, das den Nachfolger von SSL unterstützt, den wir nachfolgend besprechen werden.

Ein weiteres Problem mit SSL ist, dass die Principals eventuell keine Zertifikate haben, und selbst wenn sie welche haben, verifizieren diese nicht immer, dass die verwendeten Schlüssel dazu passen.

1996 übergab Netscape Communications Corp. SSL an die IETF zur Standardisierung. Das Ergebnis war **TLS** (*Transport Layer Security*). Es wird in RFC 5246 beschrieben.

TLS wurde auf SSL Version 3 aufgebaut. Die an SSL vorgenommenen Änderungen waren relativ gering, aber ausreichend, dass SSL Version 3 und TLS nicht miteinander arbeiten können. So wurde beispielsweise die Art der Ableitung eines Sitzungsschlüssels aus dem Premaster-Schlüssel und den Nonces geändert, um den Schlüssel stärker zu machen (also schwerer entschlüsselbar). Aufgrund dieser Inkompatibilität implementieren die meisten Browser beide Protokolle, wobei TLS im Bedarfsfall zur Verhandlung auf SSL zurückgreift. Dies wird als SSL/TLS bezeichnet. Die erste TLS-Implementierung erschien 1999; August 2008 wurde Version 1.2 definiert. Sie unterstützt stärkere Chiffre-Suiten (vor allem AES). SSL ist immer noch stark auf dem Markt vertreten, obwohl TLS es wahrscheinlich mehr und mehr ersetzen wird.

8.9.4 Sicherheit bei mobilem Code

Benennungen und Verbindungen sind zwei eng mit der Websicherheit in Verbindung stehende Bereiche. Aber es gibt noch weitere. In den Anfangszeiten, als die Webseiten nur statische HTML-Dateien waren, enthielten sie keinen ausführbaren Code. Heutzutage enthalten sie oftmals kleine Programme wie Java-Applets, ActiveX-Steuerelemente und JavaScripts. Der Download und die Ausführung von solchem **mobilen Code** stellt offensichtlich ein erhebliches Sicherheitsrisiko dar, sodass verschiedene Methoden entwickelt wurden, um die Gefahren zu minimieren. Wir sehen uns nun einige der von mobilem Code hervorgerufenen Probleme sowie Lösungsansätze an.

Sicherheit von Java-Applets

Java-Applets sind kleine Java-Programme, die in einer stapelverarbeitungsorientierten Maschinensprache namens **JVM** (*Java Virtual Machine*) kompiliert werden. Sie können auf eine Webseite gestellt werden, um mit der Seite heruntergeladen zu werden. Nachdem die Seite geladen wurde, werden die Applets in einen JVM-Interpreter im Browser eingefügt, wie in ▶ Abbildung 8.52 gezeigt.

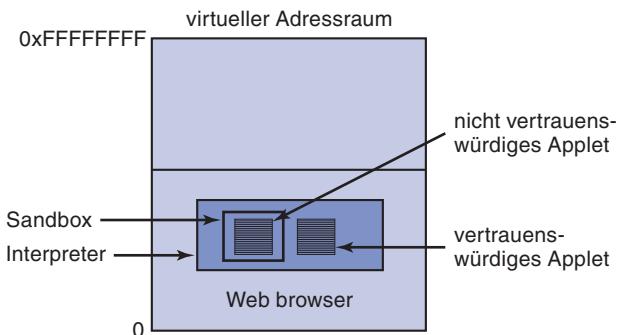


Abbildung 8.52: Applets können von einem Webbrowser interpretiert werden.

Der Vorteil der Ausführung von interpretiertem Code gegenüber kompiliertem Code ist, dass jede Anweisung vom Interpreter untersucht wird, bevor sie ausgeführt wird. Hierdurch kann der Interpreter prüfen, ob die Anweisungsadresse gültig ist. Darüber hinaus werden auch Systemaufrufe abgefangen und interpretiert. Was mit diesen Aufrufen geschieht, hängt von den Sicherheitsregeln ab. Wenn beispielsweise ein Applet vertrauenswürdig ist (z.B. von der lokalen Platte stammt), können seine Systemaufrufe kritiklos ausgeführt werden. Wenn aber ein Applet nicht vertrauenswürdig ist (weil es z.B. aus dem Internet stammt), kann es in eine sogenannte **Sandbox** gekapselt werden, um das Verhalten einzuschränken und um die Versuche, Systemressourcen zu verwenden, abzufangen.

Versucht ein Applet, eine Systemressource zu belegen, wird der Aufruf an einen Sicherheitsmonitor zur Genehmigung übergeben. Der Monitor untersucht den Aufruf und entscheidet unter Berücksichtigung der lokalen Sicherheitsregeln, ob der Aufruf zugelassen oder zurückgewiesen wird. Auf diese Weise ist es möglich, den Applets Zugriff auf einige, aber nicht alle Ressourcen zu gewähren. Leider funktioniert dieses Sicherheitsmodell in der Realität oft schlecht und weist immer wieder andere Fehler auf.

ActiveX

ActiveX-Steuerelemente sind x86-Binärprogramme, die in Webseiten eingebettet werden können. Wird eines angetroffen, wird geprüft, ob es ausgeführt werden soll. Besteht es den Test, dann wird es ausgeführt. Da es nicht interpretiert oder in eine Sandbox gesteckt wird, hat es so viel Macht wie jedes andere Benutzerprogramm und kann potenziell großen Schaden anrichten. Daher besteht die gesamte Sicherheit in der Entscheidung, ob das ActiveX-Steuerelement ausgeführt werden soll. Rückblickend ist die ganze Idee eine einzige riesige Sicherheitslücke.

Die von Microsoft dafür gewählte Methode basiert auf dem Konzept der **Code-Signierung** (*code signing*). Jedes ActiveX-Steuerelement wird mit einer digitalen Signatur ergänzt – einem Hash des Codes, der von seinem Erzeuger mit öffentlichen Schlüsseln unterschrieben wird. Wenn ein ActiveX-Steuerelement auftaucht, verifiziert der

Browser zuerst die Signatur, um sicherzustellen, dass es während der Übertragung nicht gefälscht wurde. Ist die Signatur korrekt, prüft der Browser seine internen Tabellen, um herauszufinden, ob der Erzeuger des Programms vertrauenswürdig ist oder ob es einen Zertifizierungspfad zu einem vertrauenswürdigen Erzeuger gibt. Trifft das zu, so wird das Programm ausgeführt, andernfalls nicht. Das System von Microsoft zur Verifizierung von ActiveX-Steuerelementen wird als **Authenticode** bezeichnet.

Es ist nützlich, die Java- und ActiveX-Ansätze einander gegenüberzustellen. Bei Java wird nicht versucht festzustellen, wer das Applet geschrieben hat. Stattdessen stellt ein Laufzeitinterpreter sicher, dass das Applet nichts tut, was der Rechnerbesitzer verboten hat. Demgegenüber wird bei der Code-Signierung nicht versucht, das Laufzeitverhalten des mobilen Codes zu überwachen. Wenn dieser aus einer vertrauenswürdigen Quelle stammt und nicht bei der Übertragung geändert wurde, wird er ausgeführt. Es wird nichts unternommen, um festzustellen, ob der Code bösartig ist oder nicht. Wenn der ursprüngliche Programmierer *beabsichtigt* hat, dass der Code die Festplatte formatiert und dann das Flash-ROM löscht, sodass der Rechner nie wieder gestartet werden kann, und wenn dieser Programmierer als vertrauenswürdig eingestuft wurde, dann wird der Code ausgeführt und der Rechner zerstört (es sei denn, ActiveX-Steuerelemente wurden im Browser deaktiviert).

Viele Leute sind der Ansicht, dass es eher bedenklich ist, einem unbekannten Softwareunternehmen zu vertrauen. Um das Problem zu veranschaulichen, hat ein Programmierer in Seattle ein Softwareunternehmen gegründet und es als vertrauenswürdig zertifizieren lassen. Er schrieb dann ein ActiveX-Steuerelement, das einen Rechner sauber herunterfuhr und verteilte dieses ActiveX-Steuerelement. Es fuhr die Rechner nur herunter, aber sie konnten wieder neu gestartet werden, sodass kein Schaden entstand. Er versuchte nur, dieses Problem in der Öffentlichkeit bekannt zu machen. Die offizielle Antwort war, das Zertifikat für dieses spezielle ActiveX-Steuerelement zurückzuziehen, aber das zugrunde liegende Problem wurde damit nicht gelöst (Garfinkel und Spafford, 2002). Da es keine Möglichkeit gibt, Tausende von Softwareunternehmen zu überwachen, die mobilen Code schreiben können, steht bei der Code-Signierung der schlimmste Fall noch aus.

JavaScript

JavaScript besitzt kein formales Sicherheitsmodell, aber eine lange Geschichte fehlerhafter Implementierungen. Jeder Hersteller handhabt die Sicherheit auf andere Weise. So verwendete beispielsweise Netscape Navigator Version 2 etwas Ähnliches wie das Java-Modell, was in Version 4 aber zugunsten eines Code-Signierungsmodells ganz abgeschafft wurde.

Das grundlegende Problem ist, dass die Ausführung von fremdem Code auf einem Rechner potenziell Ärger verursachen kann. Unter dem Sicherheitsaspekt ist dies so, wie wenn man einen Einbrecher ins Haus lässt und dann versucht, ihn sorgfältig zu überwachen, damit er nicht von der Küche ins Wohnzimmer entkommen kann. Wenn etwas Unerwartetes passiert und man für einen Moment abgelenkt wird, können schlimme Dinge passieren. Die Diskrepanz besteht hier darin, dass mobiler Code

lebendige Grafiken und schnelle Interaktion ermöglicht, was viele Websitedesigner für wichtiger halten als Sicherheit, insbesondere, wenn andere mit ihren Rechnern das Risiko tragen.

Browsererweiterungen

Neben der Erweiterung von Webseiten mit Code gibt es einen florierenden Markt für **Browsererweiterungen**, **Add-ons** und **Plug-ins**. Dabei handelt es sich um Computerprogramme, die die Funktionalität der Webbrowser erweitern. Plug-ins bieten oft die Fähigkeit, einen bestimmten Inhaltstyp, wie PDFs oder Flash-Animationen, zu interpretieren oder anzuzeigen. Erweiterungen und Add-ons bieten neue Browserfunktionen wie eine bessere Kennwortverwaltung oder Möglichkeiten, mit Seiten zu interagieren, zum Beispiel indem eigene Notizen hinzugefügt werden, oder bequemeres Einkaufen für verwandte Artikel.

Eine Erweiterung, Add-on oder Plug-in zu installieren, ist ganz einfach: Wenn Sie beim Browsen im Internet auf etwas stoßen, was Sie gerne hätten, folgen Sie einfach dem Link, um das Programm zu installieren. Damit laden Sie Code vom Internet herunter, der bei Ihnen im Browser installiert wird. Alle diese Programme werden für Frameworks geschrieben, die je nach zu ergänzendem Browser unterschiedlich ausfallen. Eigentlich jedoch werden sie Teil der Trusted Computing Base des Browsers. Das heißt, wenn der installierte Code fehlerhaft ist, kann dies negative Auswirkungen für den ganzen Browser haben.

Es gibt noch zwei weitere offenkundige Schadensarten. Zum einen kann sich das Programm bösartig verhalten, indem es beispielsweise persönliche Daten ausspioniert und an einen entfernten Server sendet. Soweit der Browser weiß, hat der Benutzer die Erweiterung genau zu diesem Zweck installiert. Das zweite Problem ist, dass Plug-ins den Browser befähigen, neue Inhaltstypen zu interpretieren. Oft ist dieser Inhalt eine voll entwickelte Programmiersprache. PDF und Flash sind gute Beispiele. Wenn Benutzer Seiten mit PDF- und Flash-Inhalt anschauen, führen die Plug-ins in ihren Browsern den PDF- und Flash-Code aus. Dieser Code sollte besser sicher sein, denn oft weisen die Rechner Schwächen auf, die ausgenutzt werden können. Aus diesen Gründen sollten Add-ons und Plug-ins nur installiert werden, wenn sie absolut benötigt werden, und auch nur von vertrauenswürdigen Anbietern.

Viren

Viren sind eine andere Form von mobilem Code. Im Unterschied zu den obigen Beispielen werden Viren nie eingeladen. Der Unterschied zwischen einem Virus und normalem mobilen Code besteht darin, dass Viren geschrieben sind, um sich selbst zu reproduzieren. Wenn ein Virus auf einem Rechner eingeht, entweder über eine Webseite, über einen E-Mail-Anhang oder anderweitig, dann beginnt er in der Regel, ausführbare Programme auf der Festplatte zu infizieren. Wird eines dieser Programme ausgeführt, so übernimmt der Virus die Steuerung und versucht, sich in der Regel auf andere Rechner zu verbreiten, beispielsweise durch das Senden von E-Mails an jeden Kontakt im Adressbuch des Opfers. Einige Viren infizieren den Boot-Sektor der Fest-

platte, sodass der Virus ausgeführt wird, wenn der Rechner hochgefahren wird. Viren sind im Internet ein riesiges Problem geworden und haben Milliarden Euro an Schaden angerichtet. Es gibt hierfür offensichtlich keine Lösung. Vielleicht schafft eine ganz neue Generation von Betriebssystemen, die auf sicheren Microkernen basiert und eine bessere Abschottung von Benutzern, Prozessen und Ressourcen bietet, hier Abhilfe.

8.10 Soziale Themen

Das Internet samt seiner Sicherheitstechnologie ist ein Bereich, in dem soziale Themen, öffentlich-rechtliche Belange und Technik frontal aufeinandertreffen, oftmals mit erheblichen Konsequenzen. Im Folgenden werden kurz drei Bereiche angeprochen: Datenschutz, Redefreiheit und Urheberrecht. Natürlich können wir diese Themen nur im Abriss behandeln. Weitere Informationen finden Sie in Anderson (2008a), Garfinkel und Spafford (2002) und Schneier (2004). Auch das Internet enthält eine Fülle von Materialien. Geben Sie nur einmal Wörter wie „Datenschutz“ (*privacy*), „Zensur“ (*censorship*) und „Urheberrecht“ (*copyright*) in eine Suchmaschine ein. Auch die Website dieses Buchs enthält einige Links.

8.10.1 Datenschutz

Haben Personen ein Recht auf Datenschutz? Gute Frage. Der vierte Zusatzartikel der amerikanischen Verfassung verbietet dem Staat die willkürliche Durchsuchung von Wohnung, Papiere und Eigentum und schränkt die Bedingungen ein, unter denen Durchsuchungsbefehle erteilt werden können. So war der Schutz der Privatsphäre zumindest in den USA die letzten 200 Jahre geregelt.

In den letzten zehn Jahren hat sich aber einiges verändert: zum einen wurde es den Regierungen einfacher gemacht, ihre Bürger auszuspionieren, zum anderen haben sich die Möglichkeiten der Bürger verbessert, sich dagegen zu schützen. Wenn der Staat im 18. Jahrhundert die Papiere von Bürgern durchsuchen wollte, musste er einen Polizisten zu Pferd zur Farm des betreffenden Bürgers schicken, um Einsicht in bestimmte Papiere zu fordern. Dies war sehr mühselig. Heutzutage bieten Telefongesellschaften und ISPs bei Vorlage eines Durchsuchungsbefehls bereitwillig Telefonüberwachung. Dies vereinfacht das Leben der Polizei und ein Sturz vom Pferd ist auch ausgeschlossen.

Die Kryptografie hat alles verändert. Jeder, der sich die Mühe macht, PGP herunterzuladen und zu installieren, und einen gut gehüteten Schlüssel in Alien-Stärke einsetzt, kann ziemlich sicher sein, dass niemand im Universum seine E-Mail liest – mit oder ohne Durchsuchungsbefehl. Regierungen sind sich dessen durchaus bewusst und alles andere als erfreut darüber. Echter Datenschutz bedeutet, dass es sehr viel schwieriger ist, Kriminelle aller Arten auszuspionieren, aber es ist auch viel schwieriger, Journalisten und politische Gegner auszuspionieren. Folglich haben einige Regierungen die Verwendung oder den Export der Kryptografie untersagt. In Frankreich war beispielsweise vor 1999 die gesamte Kryptografie verboten – es sei denn, die Regierung hatte einen Schlüssel.

Frankreich war kein Einzelfall. Im April 1993 gab die amerikanische Regierung die Absicht bekannt, einen Hardware-Kryptoprozessor, den **Clipper-Chip**, zum Standard für die gesamte vernetzte Kommunikation zu machen. Auf diese Weise, so wurde argumentiert, könne die Privatsphäre der Personen geschützt werden. Nebenbei wurde erwähnt, dass der Chip der Regierung die Möglichkeit einräumte, den gesamten Datenverkehr über ein Schema namens **Key Escrow** zu entschlüsseln, das der Regierung Zugriff auf alle Schlüssel gewährte. Aber man versprach, nur bei Vorlage eines gültigen Durchsuchungsbefehls zu schnüffeln. Folge war natürlich ein riesiger Aufruhr, bei dem die Datenschutzvertreter den ganzen Plan verdammten und die offiziellen Gesetzesvertreter ihn lobten. Irgendwann gab die Regierung nach und ließ das Vorhaben fallen.

Wer sich für das Thema elektronischer Datenschutz interessiert, findet auf der Website der Electronic Frontier Foundation unter www.eff.org viele weitere Informationen.

Anonyme Remailer

Über Technologien wie PGP und SSL können zwei Parteien eine sichere, authentifizierte Kommunikation aufbauen, die nicht von Dritten überwacht oder gestört wird. Manchmal jedoch besteht der beste Datenschutz darin, auf Authentifizierung zu verzichten und stattdessen die Kommunikation anonym zu halten. Anonymität kann bei Punkt-zu-Punkt-Nachrichten, Newsgruppen oder beidem erstrebenswert sein.

Betrachten wir einige Beispiele. Als Erstes seien hier politische Dissidenten genannt, die in autoritären Regimen leben und oft anonym kommunizieren möchten, um einer Gefangenschaft oder Ermordung zu entgehen. Zweitens werden illegale Praktiken in Unternehmen, Ausbildungsstätten, Behörden und anderen Organisationen oftmals durch Informanten enthüllt, die lieber anonym bleiben möchten, um nicht belangt zu werden. Drittens gibt es viele Personen mit unpopulären sozialen, politischen und religiösen Ansichten, die miteinander per E-Mail oder in Newsgruppen kommunizieren wollen, ohne ihre Identität preiszugeben. Viertens gibt es solche, die anonym über ihre Probleme mit Alkoholismus, Geisteskrankheiten, sexueller Belästigung, Kindesmissbrauch diskutieren wollen oder als Mitglied einer verfolgten Minderheit an Newsgruppen teilzunehmen wünschen. Es gibt aber natürlich noch viele andere Beispiele.

Betrachten wir ein spezielles Beispiel. In den 1990er Jahren versendeten einige Kritiker einer religiösen Minderheit ihre Ansichten über einen **anonymen Remailer** an eine USENET-Newsgruppe. Ein anonymer Remailer ist ein Server, der es seinen Benutzern erlaubt, Pseudonyme zu erstellen und unter diesen Pseudonymen E-Mails über den Server zu versenden oder Mitteilungen zu verfassen, sodass niemand sagen kann, woher die Nachrichten eigentlich stammen. Einige Beiträge enthielten Texte, die die religiöse Gruppe als Geschäftsgeheimnisse und urheberrechtlich geschützte Dokumente bezeichnete. Die religiöse Gruppierung reagierte mit einer Beschwerde bei den lokalen Behörden, dass ihre Geschäftsgeheimnisse offenbart und das Urheberrecht verletzt worden sei, beides Straftaten, die bis zum Server zurückverfolgt werden konnten. In dem sich anschließenden Gerichtsprozess wurde der Serverbetreiber gezwungen, die Zuordnungsdaten preiszugeben, die die wahre Identität der Personen enthiel-

ten, die die Beiträge lanciert hatten. (Dies war übrigens nicht das erste Mal, dass eine religiöse Gruppe nicht erfreut darüber war, dass ihre Geheimnisse ans Tageslicht kamen: William Tyndale wurde im Jahre 1536 auf dem Scheiterhaufen verbrannt, weil er die Bibel ins Englische übersetzt hatte.)

Die Internetgemeinde war zum großen Teil außer sich über diesen Vertrauensbruch. Alle waren sich darin einig, dass ein anonymer Remailer, der eine Zuordnung zwischen den echten E-Mail-Adressen und den Pseudonymen (als Typ-1-Remailer bezeichnet) erlaubt, nicht viel wert sei. Dieser Fall regte verschiedene Leute dazu an, anonyme Remailers zu entwerfen, die auch juristischen Zwangsmaßnahmen standhielten.

Diese neuen Remailers werden oft als **Cypherpunk-Remailer** bezeichnet und funktionieren wie folgt: Der Benutzer erzeugt eine E-Mail-Nachricht einschließlich der RFC-822-Header (außer natürlich *From:*), verschlüsselt diese mit dem öffentlichen Schlüssel des Remailers und sendet sie an den Remailer. Hier werden die äußeren RFC-822-Header abgestreift, der Inhalt wird entschlüsselt und die Nachricht erneut versendet. Der Remailer verwaltet weder Konten noch Protokolle, sodass der Server keine Spuren der durchlaufenden Nachrichten aufweist, selbst wenn er irgendwann konfisziert werden sollte.

Viele Benutzer, die wirklich anonym bleiben wollen, leiten ihre Anfragen über mehrere anonyme Remailers, wie in ▶ Abbildung 8.53 gezeigt. Hier möchte Alice Bob eine wirklich, wirklich, wirklich anonyme Valentinskarte senden, sodass sie drei Remailers verwendet. Sie erstellt die Nachricht M und versieht sie mit einem Header, der Bobs E-Mail-Adresse enthält. Dann verschlüsselt sie alles mit dem öffentlichen Schlüssel von Remailer 3, E_3 (durch die horizontale Schraffur angegeben). An diese hängt sie einen Header mit der E-Mail-Adresse von Remailer 3 in Klartext an. Dies ist die Nachricht, die in der Abbildung zwischen den Remailern 2 und 3 steht.

Dann verschlüsselt sie diese Nachricht mit dem öffentlichen Schlüssel des Remailers 2, E_2 (durch die vertikale Schraffur dargestellt), und hängt vorne einen Klartext-Header an, der die E-Mail-Adresse von Remailer 2 enthält. Die Nachricht wird in Abbildung 8.53 zwischen 1 und 2 dargestellt. Dann verschlüsselt sie diese Nachricht mit dem öffentlichen Schlüssel des Remailers 1, E_1 , und hängt vorne einen Klartext-Header an, der die E-Mail-Adresse von Remailer 1 enthält. Dies ist die rechts von Alice stehende Nachricht, die dann tatsächlich übertragen wird.

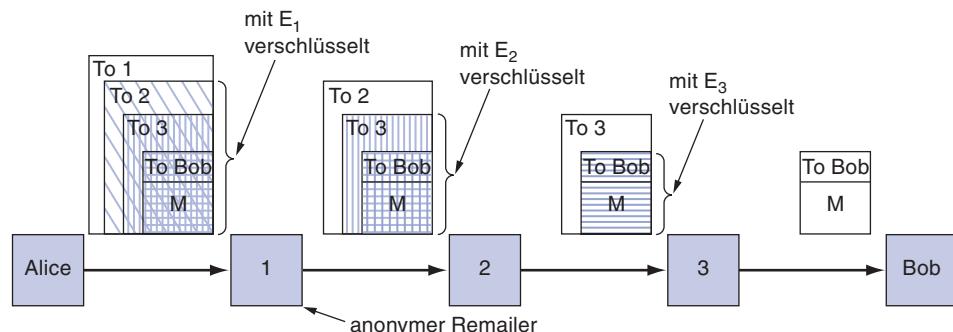


Abbildung 8.53: Wie Alice über drei Remailers Bob eine Nachricht sendet.

Wenn die Nachricht bei Remailer 1 ankommt, wird der äußere Header entfernt. Der Hauptteil wird dann entschlüsselt und an Remailer 2 gesendet. Ähnlich wird mit den beiden anderen Remailern verfahren.

Obwohl es extrem schwierig ist, die eigentliche Nachricht zu Alice zurückzuverfolgen, ergreifen viele Remailer zusätzliche Sicherheitsmaßnahmen. So halten sie beispielsweise Nachrichten eine Zufallszeitspanne zurück, ergänzen oder entfernen Datenmüll am Anfang oder Ende und ordnen Nachrichten neu an, damit man schwerer ermitteln kann, welche Nachrichtenausgabe eines Remailers zu welcher Eingabe gehört, und um die Analyse des Datenverkehrs zu erschweren. Eine Beschreibung eines solchen Remailers finden Sie in Mazières und Kaashoek (1998).

Anonymität ist nicht auf E-Mail beschränkt. Es existieren auch Dienste, die anonymes Surfen im Web ermöglichen, und zwar mit der gleichen Vorgehensweise: Jeder Knoten kennt nur den nächsten Knoten in der Kette. Diese Methode wird **Onion-Routing** genannt, da jeder Knoten eine andere Schicht von der „Zwiebel“ entfernt, um zu ermitteln, wohin das nächste Paket zu senden ist. Der Benutzer konfiguriert seinen Browser so, dass der Anonymizer-Dienst als Proxy verwendet wird. Ein bekanntes Beispiel für ein solches System ist Tor (Dingledine et al., 2004). Ab der Konfiguration laufen alle HTTP-Anforderungen durch das Anonymizer-Netz, das die Seite anfordert und zurückschickt. Die Website betrachtet den Exit-Knoten des Anonymizer-Netzes als Quelle der Anforderung, und nicht den Benutzer. Solange das Anonymizer-Netz kein Protokoll aufzeichnet, kann niemand feststellen, wer welche Seite angefordert hat.

8.10.2 Redefreiheit

Datenschutz bezieht sich auf das Bedürfnis von Personen, das, was andere Personen von ihnen sehen und wissen, zu beschränken. Ein zweites soziales Schwerpunktthema ist die Redefreiheit und deren Pendant, die Zensur, die Regierungen ausüben, um zu kontrollieren, welche Informationen ihre Bürger lesen und veröffentlichen dürfen. Das Web mit seinen Abermillionen Seiten ist zu einem Paradies für Zensoren geworden. Je nach Art und Ideologie des Regimes können Websites mit folgenden Inhalten gesperrt werden:

1. Kinder- und jugendgefährdende Inhalte
2. Hassinhalte, die auf bestimmte ethnische, religiöse, sexuelle oder andere Gruppen gerichtet sind
3. Informationen über Demokratie und demokratische Werte
4. Beschreibungen historischer Ereignisse, die der Version der Regierung widersprechen
5. Anleitungen zum Aufbrechen von Schlössern, Bauen von Waffen, Verschlüsseln von Nachrichten usw.

Die übliche Reaktion ist, die „schlechten“ Sites zu sperren.

Manchmal fallen die Ergebnisse anders aus als erwartet. So haben beispielsweise einige öffentliche Bibliotheken Webfilter auf ihren Rechnern installiert, um sie kinderfreundlich zu machen, z.B. durch das Blockieren pornografischer Sites. Die Filter haben Sites auf ihren schwarzen Listen, prüfen die Seiten aber auch auf anstößige Wörter, bevor sie sie anzeigen. In einem Fall in Loudoun County, Virginia, blockierte der Filter die Suche eines Förderers der Bibliothek nach Brustkrebs, weil das Wort „Brust“ vorkam. Der Förderer verklagte daraufhin Loudoun County. Dagegen haben in Livermore in Kalifornien Eltern eine öffentliche Bibliothek verklagt, weil sie *keinen* Filter installiert hatte und ihr zwölfjähriger Sohn beim Betrachten pornografischer Inhalte erwischt wurde. Was soll eine Bibliothek tun?

Vielen Leuten ist entgangen, dass das World Wide Web ein weltweites Web ist. Es umspannt die ganze Welt. Nicht alle Länder sind bezüglich der Inhalte im Web einer Meinung. So hat beispielsweise im November 2000 ein französisches Gericht Yahoo, ein kalifornisches Unternehmen, dazu verurteilt, die Anzeige von Auktionen mit Nazi-Erinnerungsstücken für französische Besucher der Yahoo-Website zu unterbinden, da der Besitz dieser Gegenstände gegen das französische Gesetz verstößt. Yahoo hat dies bei einem amerikanischen Gericht angefochten und Recht erhalten. Dennoch ist noch lange nicht geklärt, welches Recht in einem solchen Fall anzuwenden ist.

Stellen Sie sich einmal Folgendes vor: Was würde passieren, wenn ein Gericht in Utah Frankreich anweisen würde, alle Websites mit Wein zu sperren, weil diese nicht mit den viel strenger Alkoholgesetzen von Utah in Einklang stehen? Angenommen, China würde fordern, dass alle Websites, die demokratische Inhalte verbreiten, auf den Index gehören, weil dies nicht im Interesse des Staates ist. Gelten die iranischen Religionsgesetze für das liberale Schweden? Kann Saudi-Arabien Websites blockieren, die Frauenrechte behandeln? Das Thema ist eine echte Büchse der Pandora.

Ein passender Kommentar von John Gilmore lautet: „Das Internet interpretiert Zensur als technischen Defekt und umschifft sie.“ Eine konkrete Implementierung ist beispielsweise der **Eternity Service** (Anderson, 1996). Hiermit soll sichergestellt werden, dass veröffentlichte Informationen nicht zurückgezogen oder umgeschrieben werden können, wie dies in der Sowjetunion unter der Herrschaft von Josef Stalin an der Tagesordnung war. Um den Eternity Service zu nutzen, gibt der Benutzer an, wie lange das Material aufbewahrt werden soll, bezahlt eine umfangs- und zeitabhängige Gebühr und lädt das Material auf den Server. Danach kann keiner die Informationen mehr bearbeiten oder entfernen, nicht einmal derjenige, der sie auf den Server gestellt hat.

Wie könnte ein solcher Dienst implementiert werden? Das einfachste Modell ist ein Peer-to-Peer-System, in dem gespeicherte Dokumente auf Dutzenden von teilnehmenden Servern hochgeladen werden, von denen jeder einen Teil der Gebühr erhält und damit auch einen Anreiz, sich dem System anzuschließen. Die Server sollten über viele Gerichtsbarkeiten verteilt werden, um eine maximale Ausfallsicherheit zu gewährleisten. Listen mit zehn zufällig ausgewählten Servern sollten sicher an mehreren Orten gespeichert werden, sodass, wenn einige abgeschaltet werden, andere noch weiterexistieren. Eine Staatsgewalt, die ein Dokument auf einem Server zerstören möchte, könnte nie sicher sein, dass sie alle Kopien gefunden hat. Das System könnte

auch selbstreparierend in dem Sinn sein, dass wenn bekannt wird, dass einige Kopien zerstört wurden, die restlichen Sites neue Speicherorte suchen, um diese zu ersetzen.

Der Eternity Service war der erste Vorschlag für ein zensurresistentes System. Seither wurden weitere vorgeschlagen und in einigen Fällen auch implementiert. Es wurden verschiedene neue Funktionen hinzugefügt, wie Verschlüsselung, Anonymität und Fehlertoleranz. Oftmals werden die gespeicherten Daten in mehrere Fragmente aufgeteilt, wobei jedes Fragment auf vielen Servern gespeichert wird. Zu diesen Systemen gehören Freenet (Clarke et al., 2002), PASIS (Wylie et al., 2000) sowie Publius (Waldman et al., 2000). Weitere Arbeiten werden in Serjantov (2002) beschrieben.

Viele Länder versuchen zunehmend, den Export immaterieller Werte zu regulieren, wozu auch immer häufiger Websites, wissenschaftliche Artikel, E-Mails, Telefon-Helptdesks und anderes mehr gehören. Selbst Großbritannien mit einer jahrhundertelangen Tradition der Redefreiheit überlegt nun ernsthaft, sehr restriktive Gesetze einzuführen, nach denen beispielsweise technische Diskussionen zwischen einem britischen Professor und seinen ausländischen Studenten an der Universität von Cambridge als gesetzlich geregelter Export betrachtet werden, der eine staatliche Lizenz erfordert (Anderson, 2002). Man muss natürlich nicht betonen, dass solche politischen Maßnahmen sehr umstritten sind.

Steganografie

In Ländern, in denen Zensur allgegenwärtig ist, versuchen Dissidenten oftmals ihre Spuren mit diversen Technologien zu verwischen. Kryptografie ermöglicht das Versenden geheimer Nachrichten (eventuell sogar illegalerweise), aber wenn die Regierung der Ansicht ist, dass Alice eine verdächtige Person ist, kann allein die Kommunikation mit Bob auch diesen zu einer verdächtigen Person machen, da repressive Regierungen das Konzept der transitiven Hülle verstehen, auch wenn sie einen Mangel an Mathematikern haben. Anonyme Remailer sind hier eine Möglichkeit, aber wenn sie in dem betreffenden Land gesperrt sind und Nachrichten an ausländische Remailer einer Exportlizenz der Regierung bedürfen, sind auch sie keine große Hilfe. Das Web kann hier jedoch weiterhelfen.

Personen, die geheim kommunizieren möchten, versuchen oftmals zu verbergen, dass sie überhaupt kommunizieren. Die Kunst, Nachrichten zu verbergen, wird als **Steganografie** bezeichnet, was sich aus dem Griechischen ableitet und „geheimes Schreiben“ bedeutet. Ja, es war also auch schon den alten Griechen bekannt. Herodot beschreibt einen General, der den Kopf eines Boten rasierte, dort eine Nachricht eintätowierte und das Haar wieder nachwachsen ließ, bevor er ihn wegschickte. Moderne Verfahren gehen im Grunde nach dem gleichen Schema vor, nur haben sie eine höhere Bandbreite, eine geringere Latenzzeit und benötigen keinen Friseur.

Betrachten Sie als Beispiel ► Abbildung 8.54a. Das Foto, vom Autor in Kenia aufgenommen, enthält drei Zebras mit einer Akazie im Hintergrund. ► Abbildung 8.54b scheint das gleiche Foto mit den drei Zebras und der Akazie zu sein, jedoch mit einer zusätzlichen Besonderheit. Es enthält den ungetilgten Text von fünf Werken von Shakes-

peare: *Hamlet*, *König Lear*, *Macbeth*, *Der Kaufmann von Venedig* und *Julius Caesar*. Diese Texte belegen zusammen über 700 KB Speicher.

Wie funktioniert dieser steganografische Kanal? Das ursprüngliche Farbbild hat $1\,024 \times 768$ Pixel. Jedes Pixel besteht aus drei 8-Bit-Zahlen, je eine für den roten, grünen und blauen Wert dieses Pixels. Die Farbe des Pixels wird durch eine lineare Überlagerung der drei Farben erzeugt. Die steganografische Codierungsmethode verwendet das niederwertigste Bit eines jeden RGB-Farbwerts als verborgenen Kanal. So hat jedes Pixel Platz für 3 Bit geheime Informationen, eine im roten, eine im grünen und eine im blauen Wert. Bei einem Bild dieser Größe können bis zu $1\,024 \times 768 \times 3$ Bit oder 294 912 Byte geheime Informationen darin gespeichert werden.

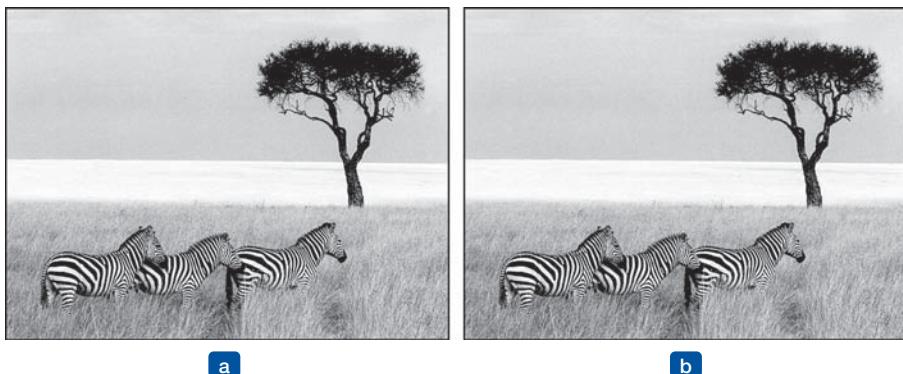


Abbildung 8.54: (a) Drei Zebras und ein Baum (b) Drei Zebras, ein Baum und der vollständige Text von fünf Stücken von William Shakespeare.

Der volle Text der fünf Stücke und eine kurze Anmerkung kommen auf 734 891 Byte. Der Text wurde zuerst mit einem Standardkomprimierungs-Algorithmus auf etwa 274 KB komprimiert. Das komprimierte Ergebnis wurde mit IDEA verschlüsselt und in das niederwertigste Bit eines jeden Farbwerts eingefügt. Wie man sieht (oder besser, nicht sieht), ist das Vorhandensein der Information völlig unsichtbar. Sie ist auch unsichtbar in der großen Farbversion des Fotos. Das Auge kann 21-Bit-Farbe von 24-Bit-Farbe so gut wie nicht unterscheiden.

Wenn man die beiden Bilder in Schwarzweiß mit einer niedrigen Auflösung ansieht, erkennt man nicht, wie leistungsstark diese Technik ist. Damit Sie ein besseres Gefühl für die Arbeitsweise der Steganografie erhalten, hat der Autor eine Demonstration vorbereitet, einschließlich des hochauflösenden Farbbildes in Abbildung 8.54b mit den fünf eingebetteten Stücken. Die Demonstration enthält Werkzeuge zum Einfügen und Extrahieren von Text in Bildern und befindet sich auf der Website zum vorliegenden Buch.

Um die Steganografie für geheime Kommunikation zu nutzen, könnten Dissidenten eine Website erstellen, die voller politisch korrekter Bilder ist, wie Fotografien des Großen Führers, lokale Sport-, Film- und Fernsehstars usw. Natürlich wären die Bilder mit steganografischen Nachrichten versetzt. Wenn die Nachrichten zuerst komprimiert und dann verschlüsselt werden, würde auch jemand, der das Vorhandensein

vermutet, enorme Schwierigkeiten haben, die Nachricht von weißem Rauschen zu unterscheiden. Natürlich sollten die Bilder aktuelle Scans sein. Nur ein Bild aus dem Internet zu kopieren und dann ein paar Bits zu ändern, ist zu offensichtlich.

Bilder sind bei Weitem nicht der einzige Träger für steganografische Nachrichten. Auch Audiodateien eignen sich hierfür. Verborgenen Informationen lassen sich in VoIP-Aufrufen beispielsweise durch die Manipulation von Paketverzögerungen, Verzerren des Audio, ja sogar in den Header-Feldern der Pakete übertragen (Lubacz et al., 2010). Selbst das Layout und die Anordnung der Tags in einer HTML-Datei können Informationen enthalten.

Obwohl wir die Steganografie im Abschnitt über die Redefreiheit behandelt haben, hat sie zahlreiche andere Einsatzbereiche. Eine häufige Verwendung ist, dass die Besitzer von Bildern geheime Nachrichten in diesen codieren, die ihre Urheberrechte dokumentieren. Wird solch ein Bild gestohlen und auf einer Website platziert, kann der gesetzmäßige Besitzer die steganografische Nachricht enthüllen, um vor Gericht zu beweisen, wem das Bild gehört. Dieses Verfahren nennt man auch **Watermarking** (*Wasserzeichen*). Es wird in Piva et al. (2002) erörtert.

Weitere Informationen über Steganografie finden Sie in Wayner (2008).

8.10.3 Urheberrechte

Datenschutz und Zensur sind zwei Bereiche, in denen Technologie auf Politik trifft. Ein dritter Bereich ist das Urheberrecht. Das **Urheberrecht** (*copyright*) gibt den Urhebern von geistigem Eigentum, wie Schriftstellern, Künstlern, Komponisten, Musikern, Fotografen, Filmschaffenden und anderen, das exklusive Recht, ihr geistiges Eigentum eine gewisse Zeitspanne exklusiv zu nutzen. Dies ist in der Regel die Lebenszeit des Autors plus 50 beziehungsweise 75 Jahre im Falle von Unternehmenseigentum. Nachdem das Urheberrecht eines Werks abläuft, ist das Werk öffentlich kostenfrei nutzbar. Das Gutenberg-Projekt (www.promo.net/pg) beispielsweise hat Tausende von öffentlich verfügbaren Werken (z.B. von Goethe, Schiller, Heine) im Web veröffentlicht. 1998 verlängerte der amerikanische Kongress auf Anforderung von Hollywood das Urheberrecht um zwanzig Jahre, wobei behauptet wurde, dass ohne diese Verlängerung kein Kreativschaffender mehr produktiv sein würde. Patente dagegen laufen nur zwanzig Jahre und es wird trotzdem ständig Neues erfunden.

Das Urheberrecht rückte in den Mittelpunkt als Napster, eine Musiktauschbörse, 50 Millionen Mitglieder hatte. Obwohl Napster im Prinzip keine Musik kopierte, waren die Gerichte der Auffassung, dass die Pflege einer zentralen Datenbank, in der verwaltet wurde, wer welches Stück hatte, eine Beihilfe zu einer Rechtsverletzung darstellte, das heißt, Napster half anderen Personen, diese Rechtsverletzung auszuführen. Während niemand ernsthaft behauptet, dass das Urheberrecht eine schlechte Idee ist (obwohl viele behaupten, dass die Zeitspanne viel zu lang ist, da sie große Unternehmen gegenüber der Öffentlichkeit begünstigt), wirft die nächste Musiktauschgeneration bereits größere ethische Probleme auf.

Betrachten Sie beispielsweise ein Peer-to-Peer-Netz, in dem Personen legale Dateien tauschen (Public-Domain-Musik, Heimvideos, religiöse Traktate, die keine Geschäftsgeheimnisse sind, usw.) sowie einige, die urheberrechtlich geschützt sind. Nehmen wir weiterhin an, dass jeder die ganze Zeit über ADSL oder Kabel online ist. Jeder Rechner hat einen Index dessen, was auf der Platte gespeichert ist, plus eine Liste der anderen Mitglieder. Jemand, der ein spezielles Stück sucht, kann zufällig ein Mitglied auswählen und prüfen, ob dieses das gewünschte Stück hat. Wenn nicht, kann er alle anderen Mitglieder in der Liste dieser Person durchgehen sowie alle Mitglieder in deren Listen usw. Computer sind für diese Art von Arbeit sehr gut geeignet. Wenn das gesuchte Objekt gefunden wird, muss der Suchende es nur kopieren.

Ist das Werk urheberrechtlich geschützt, besteht die Möglichkeit, dass der Anfordernde eine Gesetzesübertretung begeht (obwohl bei internationalen Übertragungen die Frage nicht klar ist, welches Gesetz gilt). Wie sieht es aber für den Anbieter aus? Ist es ein Verbrechen, Musik zu haben, für die man gezahlt und die man legal auf die Festplatte heruntergeladen hat, wo andere sie finden können? Wenn Sie eine unverschlossene Hütte irgendwo auf dem Land haben und ein IP-Dieb steigt ein mit Notebook und Scanner unter dem Arm, scannt ein urheberrechtlich geschütztes Buch ein auf die Festplatte des Notebooks und verschwindet, sind *Sie* dann schuldig, dass Sie nicht das Urheberrecht eines anderen geschützt haben?

Aber es sieht nach noch mehr Ärger an der Urheberrechtsfront aus. Es gibt einen großen Kampf zwischen Hollywood und der Computerbranche. Hollywood möchte das gesamte geistige Eigentum streng schützen und die Computerbranche möchte nicht die Polizei für Hollywood sein. Im Oktober 1998 verabschiedete der Kongress das **Digital Millennium Copyright Act** (DMCA, Urheberrechtsgesetz für das digitale Jahrtausend), das das Umgehen von vorhandenen Schutzmechanismen oder die Anleitung zum Umgehen vorhandener Schutzmechanismen zum kriminellen Akt erklärt. Eine ähnliche Gesetzgebung wurde in der Europäischen Union erlassen. Wenngleich praktisch niemand der Meinung ist, dass Piraten in Fernost urheberrechtlich geschützte Werke duplizieren dürfen, sind viele Leute der Ansicht, dass DMCA die Balance zwischen den Interessen der Urheberechtsbesitzer und dem öffentlichen Interesse verschiebt.

Ein typisches Beispiel: Im September 2000 sponserte ein Konsortium der Musikindustrie, das den Auftrag hatte, ein nicht knackbares System zum Onlineverkauf von Musik zu erarbeiten, einen Wettbewerb, bei dem die Teilnehmer eingeladen wurden, das System aufzubrechen (was genau das Richtige für jedes neue Sicherheitssystem ist). Ein Team von Sicherheitsforschern aus verschiedenen Universitäten, das von Prof. Edward Felten von Princeton geleitet wurde, nahm die Herausforderung an und knackte das System. Sie schrieben dann einen Artikel über ihre Ergebnisse und reichten diesen auf einer USENIX-Sicherheitskonferenz ein, wo er einer Begutachtung unterzogen und akzeptiert wurde. Vor der Präsentation des Artikels erhielt Felten einen Brief von der Recording Industry Association of America, worin den Autoren gedroht wurde, sie im Rahmen von DMCA gerichtlich zu belangen, falls sie den Artikel veröffentlichten.

Ihre Antwort war, dass sie eine Klage erhoben, um ein amerikanisches Gericht entscheiden zu lassen, ob die Veröffentlichung wissenschaftlicher Artikel über Forschungen im Bereich Sicherheit immer noch legal war. Nachdem ein wirkliches Gericht darüber zu befinden hatte, zog die Branche die Drohung zurück und das Gericht wies die Klage von Felten ab. Die Branche war zweifelsohne von der Schwachheit ihres Falls motiviert: Sie hatten Leute ermuntert, ihr System aufzubrechen, und dann jemandem gedroht, ihn zu verklagen, der ihre Herausforderung angenommen hatte. Nach der zurückgezogenen Klage wurde der Artikel veröffentlicht (Craver et al., 2001). Eine neue Konfrontation ist so gut wie sicher.

In der Zwischenzeit haben illegal heruntergeladene Songs und Filme das massive Wachstum der Peer-to-Peer-Netze begünstigt. Das hat den Inhabern des Urheberrechts nicht gefallen, die mit Bezug auf das DMCA Maßnahmen ergriffen haben. Heutzutage gibt es automatische Systeme, die Peer-to-Peer-Netze durchsuchen und Netzwerkbetreiber und Benutzer warnen, die in Verdacht stehen, das Urheberrecht zu verletzen. In den Vereinigten Staaten werden diese Warnungen als **DMCA Takedown Notices** bezeichnet. Diese Suche ist ein Wettrüsten, da es schwer ist, zuverlässig die zu fangen, die das Urheberrecht verletzen. Sogar ihr Drucker kann versehentlich für den Schuldigen gehalten werden (Piatek et al., 2008).

Ein verwandtes Thema ist der Umfang der **Doktrin der angemessenen Verwendung** (*fair use doctrine*), der von den Gerichten in verschiedenen Ländern festgelegt wurde. Diese Doktrin besagt, dass Käufer eines urheberrechtlich geschützten Werks bestimmte begrenzte Rechte haben, das Werk zu kopieren. Hierzu gehört auch das Recht, Teile daraus für wissenschaftliche Zwecke zu zitieren, es als Lehrmaterial an Schulen und Universitäten zu verwenden und in einigen Fällen Sicherungskopien zur persönlichen Verwendung zu machen, falls das Original ausfällt. Was eine „angemessene“ Verwendung ist, entscheidet erstens, ob es sich um einen kommerziellen Einsatz handelt, zweitens welcher Prozentsatz des Ganzen kopiert wird und drittens die Auswirkung des Kopierens auf den Verkauf des Werks. Da DMCA und ähnliche Gesetze in der Europäischen Union die Umgehung von Maßnahmen zum Urheberrechtschutz verbieten, verbieten diese Gesetze auch die legale angemessene Nutzung. In Wirklichkeit nimmt das DMCA den Benutzern ihre historischen Rechte und gibt den Verkäufern von Inhalten mehr Macht. Ein größerer Schlagabtausch wird unvermeidbar sein.

Eine weitere in Arbeit befindliche Entwicklung, die sogar DMCA in den Schatten stellt, was die Verschiebung der Balance zwischen den Urheberrechtbesitzern und den Benutzern angeht, ist das **Trusted Computing**, wie es von industriellen Gremien wie **TCG** (*Trusted Computing Group*), angeführt von Unternehmen wie Intel und Microsoft, befürwortet wird. Das Konzept besteht darin, auf der Ebene unterhalb des Betriebssystems Unterstützung zu bieten, um das Benutzerverhalten auf verschiedene Weise sorgfältig zu überwachen (z.B. das Abspielen von raubkopierter Musik), mit dem Ziel, unerwünschtes Verhalten zu unterbinden. Dies wird mithilfe eines kleinen Chips namens **TPM** (*Trusted Platform Module*) erreicht, der relativ manipulationssicher ist. Die meisten PCs, die heutzutage verkauft werden, sind mit einem TPM ausgestattet. Das System erlaubt Software, die von den Inhalteanbietern geschrieben wurde, die PCs auf

eine Art und Weise zu manipulieren, die von den Benutzern nicht geändert werden kann. Dies wirft die Frage auf, wem beim Trusted Computing noch vertraut werden kann. Mit Sicherheit nicht dem Benutzer. Man muss natürlich nicht betonen, dass die sozialen Folgen dieser Maßnahme immens sind. Es ist schön, dass die Industrie der Sicherheit schließlich Aufmerksamkeit zollt. Aber es ist beklagenswert, dass dies ganz auf die Durchsetzung der Urheberrechtsgesetze gerichtet ist und weniger darauf, Viren, Cracker, Eindringlinge und andere Sicherheitsthemen, die den meisten Leuten Sorgen machen, in den Griff zu kriegen.

Kurz gefasst: Die Gesetzgeber und Rechtsanwälte werden auch in den nächsten Jahren noch damit beschäftigt sein, ein Gleichgewicht zwischen den ökonomischen Interessen der Urheberrechtsbesitzer und dem öffentlichen Interesse zu finden. Cyberspace unterscheidet sich nicht von der materiellen Welt: Ständig ist eine Gruppe im Interessenkonflikt mit einer anderen, was zu Machtkämpfen und Rechtsstreitigkeiten und auch (hoffentlich) zu einer Art Lösung führt, zumindest bis wieder neue umstrittene Technologien aufkommen.

Zusammenfassung

Kryptografie ist ein Werkzeug, mit dem Informationen geheim gehalten werden können. Darüber hinaus sichert deren Verwendung die Integrität und Authentizität der Daten. Alle modernen kryptografischen Systeme basieren auf dem Kerckhoffs'schen Prinzip, dass der Algorithmus bekannt, aber der Schlüssel geheim ist. Viele kryptografische Algorithmen verwenden komplexe Transformationen, die Substitutionen und Permutationen enthalten, um Klartext in Chiffertext umzuwandeln. Wenn die Quantenkryptografie irgendwann praxistauglich ist, kann die Verwendung von One-Time Pads tatsächlich zu nicht aufbrechbaren Kryptosystemen führen.

Kryptografische Algorithmen können in Algorithmen mit symmetrischen Algorithmen mit öffentlichen Schlüsseln unterteilt werden. Algorithmen mit symmetrischen Schlüsseln vertauschen die Bits in mehreren Runden, die von einem Schlüssel parametrisiert werden, um solchermaßen Klartext in Chiffertext zu verwandeln. AES (Rijndael) und Triple DES sind die bekanntesten symmetrischen kryptografischen Algorithmen. Diese Algorithmen können unter anderem im elektronischen Code-Book-Modus, im Cipher-Block-Chaining-Modus, Stream-Cipher-Modus, Counter-Modus verwendet werden.

Algorithmen mit öffentlichen Schlüsseln haben die Eigenschaft, dass zur Verschlüsselung und zur Entschlüsselung jeweils ein anderer Schlüssel verwendet wird. Hierbei kann der Entschlüsselungsschlüssel nicht aus dem Verschlüsselungsschlüssel abgeleitet werden. Diese Eigenschaften erlauben die Veröffentlichung des öffentlichen Schlüssels. Der bedeutendste Algorithmus mit öffentlichen Schlüsseln ist RSA, der seine Stärke aus der Tatsache ableitet, dass es sehr schwierig ist, große Zahlen zu faktorisieren.

Rechtsgültige, kommerzielle und andere Dokumente müssen unterzeichnet werden. Folglich wurden einige Schemata für **digitale Signaturen** entwickelt, die Algorithmen mit sowohl symmetrischen wie auch öffentlichen Schlüsseln verwenden. In der Regel wird für Nachrichten, die unterzeichnet werden sollen, mithilfe von Algorithmen wie SHA-1 ein Hash gebildet, wobei dann diese Hashes anstatt der Originalnachricht unterzeichnet werden.

Öffentliche Schlüssel können mit Zertifikaten verwaltet werden. Dies sind Dokumente, die einen Principal an einen öffentlichen Schlüssel binden. Die Zertifikate werden von einer vertrauenswürdigen Stelle unterzeichnet oder von jemandem, der (rekursiv) von einer vertrauenswürdigen Stelle genehmigt wurde. Die Wurzel dieser Kette muss vorab bekannt sein. Browser haben in der Regel viele oberste Zertifikate bereits integriert.

Diese kryptografischen Werkzeuge können im sicheren Netzverkehr eingesetzt werden. IPsec verschlüsselt in der Vermittlungsschicht den Paketfluss von Host zu Host. **Firewalls** können den Datenverkehr in und aus einer Organisation prüfen, oftmals auf der Basis des verwendeten Protokolls und der Ports. **Virtuelle private Netze** können die alten Standleitungsnetze simulieren, um bestimmte Sicherheitseigenschaften zur Verfügung zu stellen. Und schließlich haben drahtlose Netze hohe Sicherheitsansprüche, damit die Nachrichten nicht von jedem gelesen werden können, und IEEE 802.11i bietet diese Sicherheit.

Wenn zwei Parteien eine Sitzung aufbauen, müssen sie sich gegenseitig **authentifizieren** und, wenn erforderlich, einen gemeinsamen **Sitzungsschlüssel** errichten. Es existieren verschiedene Authentifizierungsprotokolle, einschließlich einiger, die eine vertrauenswürdige dritte Partei verwenden, Diffie-Hellman, Kerberos und die Kryptografie mit öffentlichen Schlüsseln.

Die Sicherheit von E-Mail kann über eine Kombination der hier behandelten Technologien erreicht werden. **PGP** komprimiert beispielsweise Nachrichten und verschlüsselt sie dann mit einem geheimen Schlüssel. Es sendet den geheimen Schlüssel, der mit dem öffentlichen Schlüssel des Empfängers verschlüsselt wurde. Darüber hinaus wird von der Nachricht ein Hash gebildet und der signierte Hash gesendet, um die Integrität der Nachricht zu verifizieren.

Ein weiteres wichtiges Thema ist **Sicherheit im Web**. Hier sei als Erstes die sichere Benennung genannt. DNSsec kann helfen, DNS-Spoofing zu vermeiden. Die meisten E-Commerce-Websites verwenden SSL/TLS, um sichere, authentifizierte Sitzungen zwischen Client und Server zu errichten. Außerdem werden zur Behandlung von mobilem Code verschiedene Techniken eingesetzt, wie **Sandboxing** und **Code-Signierung**.

Das Internet wirft viele Fragen auf, vor allem in Bereichen, in denen Technologie und öffentliches Politik im Widerstreit stehen. Zu diesen Bereichen gehören unter anderem **Datenschutz**, **Redefreiheit** und **Urheberrecht**.



Übungsaufgaben

- 1** Knacken Sie folgende monoalphabetische Substitutionschiffre. Der Klartext besteht nur aus Buchstaben. Es ist ein sehr bekannter Ausschnitt aus einem Gedicht von Lewis Carroll:

mvyy bek mnyx n yvjjyr snijrh invq n muvjydt je n idnvy
 jurhri n fehfevir pyeir oruvdq ki ndq uri jhrnqvdt ed zb jnvy
 lrr uem rnrhyb jur yeoirhi ndq jur jkhjyri nyy nqlndpr
 Jurb nhr mnvjydt ed jur iuvdtyr mvyy bek pezr ndq wevd jur qndpr
 mvyy bek, medj bek, mvyy bek, medj bek, mvyy bek wevd jur qndpr
 mvyy bek, medj bek, mvyy bek, medj bek, medj bek wevd jur qndpr

- 2** Eine affine Chiffre gehört zu den monoalphabetischen Substitutionschiffren, bei der die Buchstaben eines Alphabets der Größe m zuerst auf Ganzzahlen im Bereich 0 bis $m-1$ abgebildet werden. Anschließend wird die Ganzzahl, die jeweils für einen Klartextbuchstaben steht, in eine Ganzzahl umgewandelt, die für den entsprechenden Chiffretextbuchstaben steht. Die Verschlüsselungsfunktion für einen einzigen Buchstaben lautet:

$$E(x) = (ax + b) \bmod m$$

wobei m die Größe des Alphabets ist und a und b der Schlüssel der Chiffre und teilerfremd sind. Trudy findet heraus, dass Bob einen Chiffretext mithilfe einer affinen Chiffre erzeugt hat. Sie erhält eine Kopie des Chiffretextes und stellt fest, dass der häufigste Buchstabe des Chiffretextes „R“ und der zweithäufigste „K“ ist. Zeigen Sie, wie Trudy den Code aufbrechen kann, um den Klartext zu erhalten.

- 3** Knacken Sie folgende Spaltentausch-Transpositionschiffre. Der (englische) Klartext stammt aus einem bekannten Computerfachbuch, deshalb dürfte darin das Wort „Computer“ vorkommen. Der Klartext besteht nur aus Buchstaben (keine Leerzeichen). Der Chiffretext ist zur einfacheren Lesbarkeit in Blöcke von je fünf Zeichen unterteilt.

aauan cvlre runn dltme aeepb ytust iceat npmey iicgo gorch srsoc
 nntii imiha oofpa gsivt tpsit lboir otoex

- 4** Alice verwendete eine Transpositionschiffre, um ihre Nachrichten an Bob zu verschlüsseln. Für zusätzliche Sicherheit hat sie die Transpositionschiffre mithilfe einer Substitutionschiffre verschlüsselt und die verschlüsselte Chiffre in ihrem Computer gespeichert. Trudy ist es gelungen, in den Besitz des Schlüssels der verschlüsselten Transpositionschiffre zu gelangen. Kann Trudy die Nachricht von Alice an Bob entschlüsseln? Begründen Sie Ihre Antwort.

- 5** Finden Sie ein 77-Bit One-Time Pad, das den Text „Hello World“ aus dem Chiffretext von Abbildung 8.4 erstellt.

- 6** Sie sind ein Spion und haben bequemerweise eine Bibliothek mit einer riesigen Anzahl Bücher zur Hand. Ihr Verbindungsman hat ebenfalls eine solche Bibliothek. Sie haben sich auf *Lord of the Rings* als One-Time Pad geeinigt. Erläutern Sie, wie Sie damit einen unendlich langen One-Time Pad erzeugen können.

- 7 Die Quantenkryptografie setzt eine Photonenpistole voraus, die bei Bedarf ein einzelnes Photon abschießt, das 1 Bit trägt. Berechnen Sie bei dieser Aufgabe, wie viele Photonen ein Bit auf einer 250-Gbit/s-Glasfaserverbindung trägt. Nehmen Sie an, dass die Länge des Photons gleich seiner Wellenlänge ist, die für diese Aufgabe 1 μm ist. Die Lichtgeschwindigkeit in Glasfasern ist 20 cm/ns.
- 8 Wenn Trudy bei Verwendung der Quantenkryptografie Photonen abfängt und neu erzeugt, sind einige davon falsch. Dies führt zu Fehlern im One-Time Pad von Bob. Wie hoch ist der Prozentsatz der falschen Bits im One-Time Pad von Bob im Durchschnitt?
- 9 Einem grundlegenden kryptografischen Prinzip zufolge müssen alle Nachrichten Redundanz aufweisen. Wir wissen aber auch, dass Eindringlinge sich die Redundanz zunutze machen können, um leichter festzustellen, ob der geratene Schlüssel korrekt ist. Betrachten Sie zwei Arten der Redundanz. Bei der ersten enthalten die ersten n Bit des Klartexts ein bekanntes Muster. Bei der zweiten enthalten die letzten n Bit der Nachricht einen Hash über die Nachricht. Sind diese beiden vom Sicherheitsstandpunkt aus gleichwertig? Erläutern Sie Ihre Antwort.
- 10 In Abbildung 8.6 wechseln sich die P- und S-Boxen ab. Diese Anordnung ist zwar vom ästhetischen Gesichtspunkt ansprechend, aber ist sie auch sicherer als eine Anordnung, bei der erst alle P-Boxen und dann alle S-Boxen kommen?
- 11 Denken Sie sich eine Attacke auf DES aus, wobei Ihnen bekannt ist, dass der Klartext ausschließlich aus ASCII-Großbuchstaben sowie Leerzeichen, Komma, Punkt, Semikolon, Zeilenumbruch und Zeilenrücklauf besteht. Über die Paritätsbits des Klartexts ist nichts bekannt.
- 12 Im Text haben wir berechnet, dass ein Dechiffrierrechner mit einer Million Prozessoren, der einen Schlüssel in 1 Nanosekunde berechnen könnte, 10^{16} Jahre benötigen würde, um die 128-Bit-Version von AES aufzubrechen. Lassen Sie uns berechnen, wie lange es dauert, bis dieser Zeitraum auf 1 Jahr reduziert werden kann, was immer noch eine lange Zeit ist. Um dies zu erreichen, benötigen wir Computer, die 10^{16} -mal schneller sind. Angenommen, das Gesetz von Moore (die Rechenleistung verdoppelt sich alle 18 Monate) ist weiterhin gültig, wie viele Jahre benötigt ein Parallelrechner, um den Zeitraum zum Knacken der Chiffre auf ein Jahr zu senken?
- 13 AES unterstützt einen 256-Bit-Schlüssel. Wie viele Schlüssel hat AES-256? Schauen Sie, ob Sie eine Zahl gleicher Größenordnung in der Physik, Chemie oder Astronomie finden können. Nutzen Sie das Internet, um nach großen Zahlen zu suchen. Ziehen Sie aus Ihrer Suche eine Schlussfolgerung.
- 14 Nehmen Sie an, dass eine Nachricht mit DES im Counter-Modus verschlüsselt wurde. Ein Bit des Chiffretexts in Block C_i wird versehentlich bei der Übertragung von einer 0 in eine 1 umgewandelt. Wie viel Klartext wird dadurch verstümmelt?
- 15 Betrachten wir noch einmal Cipher Block Chaining. Statt der Umwandlung eines 0-Bits in ein 1-Bit wird diesmal ein zusätzliches 0-Bit in den Chiffretext nach Block C_i eingefügt. Wie viel Klartext wird dadurch verstümmelt?
- 16 Vergleichen Sie den Cipher-Block-Chaining-Modus mit dem Cipher-Feedback-Modus in Bezug auf die Zahl der Verschlüsselungsvorgänge, die nötig sind, um eine große Datei zu übertragen. Welcher Modus ist effizienter und um wie viel?

- 17** Verwenden Sie ein RSA-Kryptosystem mit öffentlichem Schlüssel, wobei $a=1$, $b=2 \dots$ $y=25$, $z=26$ ist.
- Listen Sie fünf gültige Werte für d auf, wenn $p=5$ und $q=13$.
 - Finden Sie e , wenn $p=5$, $q=31$ und $d=37$.
 - Finden Sie e und verschlüsseln Sie „hello“, wenn $p=3$, $q=11$ und $d=9$.
- 18** Alice und Bob verwenden RSA-Verschlüsselung mit öffentlichem Schlüssel, um miteinander zu kommunizieren. Trudy findet heraus, dass Alice und Bob eine der Primzahlen genutzt haben, mit der die Anzahl n ihrer öffentlichen Schlüsselpaare festgelegt wird. In anderen Worten, Trudy hat festgestellt, dass $n_a=p_a \times q$ und $n_b=p_b \times q$. Wie kann Trudy sich diese Information zunutze machen, um Alices Code zu knacken?
- 19** Betrachten Sie die Verwendung des Counter-Modus, wie in Abbildung 8.15 dargestellt, aber mit $IV=0$. Bedroht die Verwendung von 0 die Sicherheit der Chiffre im Allgemeinen?
- 20** In Abbildung 8.20 wird dargestellt, wie Alice Bob eine unterzeichnete Nachricht senden kann. Wenn Trudy P ersetzt, kann Bob dies feststellen. Was passiert aber, wenn Trudy sowohl P als auch die Signatur ersetzt?
- 21** Digitale Signaturen haben eine potentielle Schwäche, was nicht zuletzt auf faule Benutzer zurückzuführen ist. Bei E-Commerce-Transaktionen könnte ein Vertrag erstellt werden, den der Benutzer mit seinem SHA-1-Hash unterzeichnen soll. Wenn der Benutzer nicht verifiziert, dass Vertrag und Hash übereinstimmen, könnte der Benutzer versehentlich einen anderen Vertrag unterzeichnen. Angenommen die Mafia versucht, diese Schwäche gewinnbringend für sich auszunutzen. Sie richtet eine Bezahl-Website ein (Pornografie, Glücksspiele usw.) und fragt neue Kunden nach deren Kreditkartennummer. Anschließend sendet die Mafia dem Kunden einen Vertrag, in dem der Kunde bestätigen soll, dass er diesen Dienst nutzen möchte und mit Kreditkarte bezahlt. Sie fordert den Kunden auf, diesen Vertrag zu unterzeichnen, wobei sie weiß, dass die meisten den Vertrag unterzeichnen werden, ohne zu prüfen, ob der Vertrag und der Hash übereinstimmen. Zeigen Sie, wie die Mafia Diamanten von einem seriösen Internetjuwelier kaufen kann, die dann einem arglosen Kunden berechnet werden.
- 22** Eine Mathematikklasse hat 25 Schüler. Angenommen, alle Schüler wurden in der ersten Hälfte des Jahres, d. h. zwischen dem 1. Januar und dem 30. Juni geboren. Wie hoch ist die Wahrscheinlichkeit, dass mindestens zwei Schüler am gleichen Tag Geburtstag haben? Gehen Sie davon aus, dass niemand am 29. Februar geboren wurde, sodass es 181 mögliche Geburtstage gibt.
- 23** Nachdem Ellen Marilyn gegenüber ihre miese Tour in Sachen Beförderung von Tom gestanden hat, unternimmt Marilyn Anstrengungen zur Schadensbegrenzung. Sie entscheidet, den Inhalt künftiger Nachrichten auf ein Diktiergerät aufzusprechen und sie von ihrer neuen Sekretärin abtippen zu lassen. Dann möchte sie die Nachrichten an ihrem Terminal durchsehen, um sicherzustellen, dass sie nicht verändert wurden. Kann die neue Sekretärin immer noch die Nachrichten mit einer Geburtstagsattacke fälschen, und wenn ja, wie? *Tipp:* Sie kann.
- 24** Betrachten Sie den misslungenen Versuch von Alice in Abbildung 8.23, den öffentlichen Schlüssel von Bob zu erhalten. Angenommen, Bob und Alice nutzen bereits einen geheimen Schlüssel, aber Alice möchte noch den öffentlichen Schlüssel von Bob. Besteht eine Möglichkeit, diesen sicher zu erhalten? Falls ja, wie?

- 25** Alice möchte mit Bob über öffentliche Schlüssel kommunizieren. Sie errichtet eine Verbindung zu jemandem, von dem sie hofft, dass es Bob ist. Sie fragt ihn nach seinem öffentlichen Schlüssel, den er ihr in Klartext zusammen mit einem X.509-Zertifikat zusendet, das von der obersten Zertifizierungsstelle unterzeichnet wurde. Alice hat bereits den öffentlichen Schlüssel der obersten Zertifizierungsstelle. Welche Schritte führt Alice aus, um zu verifizieren, dass sie mit Bob spricht? Nehmen Sie an, dass Bob sich nicht darum kümmert, mit wem er spricht (z. B. kann Bob ein öffentlicher Dienst sein).
- 26** Angenommen, ein System verwendet PKI, das auf einer Baum-Hierarchie von Zertifizierungsstellen basiert. Alice möchte mit Bob kommunizieren und erhält ein Zertifikat von Bob, das von der Zertifizierungsstelle X nach der Errichtung eines Kommunikationskanals mit Bob unterzeichnet wurde. Angenommen, Alice hat von X noch nie etwas gehört. Welche Schritte führt Alice aus, um zu verifizieren, dass sie mit Bob spricht?
- 27** Kann IPsec mit AH im Transportmodus verwendet werden, wenn einer der Rechner hinter einer NAT-Box hängt? Erklären Sie Ihre Antwort.
- 28** Alice möchte eine Nachricht an Bob unter Verwendung von SHA-1-Hashes senden. Sie fragt Sie, welchen Signaturalgorithmus sie verwenden sollte. Welchen Algorithmus würden Sie empfehlen?
- 29** Geben Sie einen Grund an, warum eine Firewall zur Untersuchung des eingehenden Datenverkehrs konfiguriert werden sollte. Geben Sie einen Grund an, warum sie zur Untersuchung des ausgehenden Datenverkehrs konfiguriert werden sollte. Sind Sie der Ansicht, dass die Untersuchungen erfolgreich sind?
- 30** Angenommen, eine Organisation verwendet VPN, um ihre Standorte über das Internet sicher zu verbinden. Jim, ein Benutzer in der Organisation, verwendet das VPN, um mit seiner Chefin Mary zu kommunizieren. Beschreiben Sie eine Kommunikationsart zwischen Jim und Mary, die weder einer Verschlüsselung noch eines anderen Sicherheitsmechanismus bedarf, sowie eine weitere Kommunikationsart, die einer Verschlüsselung oder eines anderen Sicherheitsmechanismus bedarf. Erklären Sie Ihre Antwort.
- 31** Ändern Sie eine Nachricht in dem Protokoll von Abbildung 8.34 geringfügig ab, um es gegen den Reflexionsangriff immun zu machen. Erläutern Sie, warum Ihre Änderung funktioniert.
- 32** Der Schlüsselaustausch nach Diffie-Hellman wird benutzt, um einen geheimen Schlüssel zwischen Alice und Bob einzurichten. Alice sendet Bob $(227, 5, 82)$. Bob antwortet mit (125) . Die Geheimnummer x von Alice ist 12 und Bobs Geheimnummer y ist 3 . Zeigen Sie, wie Alice und Bob den Geheimschlüssel berechnen.
- 33** Zwei Benutzer können einen gemeinsamen geheimen Schlüssel mit dem Diffie-Hellman-Algorithmus aufbauen, auch wenn sie sich nie gesehen haben, keine Geheimnisse teilen und keine Zertifikate haben.
- Erklären Sie, warum dieser Algorithmus anfällig für einen Man-in-the-Middle-Angriff ist.
 - Welchen Einfluss hätte es auf diese Anfälligkeit, wenn n oder g geheim wären?
- 34** Warum wird im Protokoll von Abbildung 8.39 A in Klartext zusammen mit dem verschlüsselten Sitzungsschlüssel gesendet?

- 35** In dem Needham-Schroeder-Protokoll erzeugt Alice zwei Aufforderungen, R_A und R_{A2} . Das scheint übertrieben zu sein. Hätte nicht eine genügt?
- 36** Nehmen Sie an, dass ein Unternehmen zur Authentifizierung Kerberos verwendet. Wie sieht es mit der Sicherheit und Dienstverfügbarkeit aus, wenn der Authentifizierungsserver oder der Ticketausgabe-Server ausfällt?
- 37** Alice verwendet das Authentifizierungsprotokoll mit öffentlichem Schlüssel aus Abbildung 8.43, um die Kommunikation mit Bob zu authentifizieren. Beim Senden der Nachricht 7 hat Alice jedoch vergessen, R_B zu verschlüsseln. Trudy kennt jetzt den Wert von R_B . Müssen Alice und Bob die Authentifizierungsprozedur mit neuen Parametern wiederholen, um eine sichere Kommunikation sicherzustellen? Erläutern Sie Ihre Antwort.
- 38** Bei dem Authentifizierungsprotokoll mit öffentlichen Schlüsseln von Abbildung 8.43 ist R_B in Nachricht 7 mit K_S verschlüsselt. Ist diese Verschlüsselung notwendig oder hätte es genügt, sie im Klartext zurückzusenden? Erklären Sie Ihre Antwort.
- 39** POS-Terminals, die mit Magnetstreifenkarten und PIN-Codes arbeiten, haben eine fatale Schwäche: Ein böswilliger Händler kann seinen Kartenleser so manipulieren, dass er alle auf Karten befindlichen Informationen und den PIN-Code erfasst und speichert, um künftig betrügerische Transaktionen durchzuführen. Terminals der nächsten Generation werden mit Karten arbeiten, die komplett mit CPU, Tastatur und einem winzigen Display ausgestattet sind. Denken Sie sich für dieses System ein Protokoll aus, das bösartige Händler nicht aufbrechen können.
- 40** Ist es möglich, Multicasting bei einer PGP-Nachricht zu verwenden? Welche Beschränkungen sind anzuwenden?
- 41** Gehen Sie von der Annahme aus, dass alle Benutzer im Internet PGP einsetzen. Kann eine PGP-Nachricht an eine beliebige Internetadresse gesendet und korrekt decodiert werden? Erläutern Sie Ihre Antwort.
- 42** Der Angriff in Abbildung 8.47 lässt einen Schritt aus. Dieser Schritt ist für einen erfolgreichen Angriff nicht erforderlich, aber er könnte den potenziellen Verdacht im Nachhinein reduzieren. Welcher Schritt fehlt?
- 43** Das SSL-Datentransportprotokoll enthält zwei Nonces wie auch einen Premaster-Schlüssel. Welchen Sinn, wenn überhaupt, hat die Verwendung der Nonces?
- 44** Gegeben sei ein Bild von 2 048×512 Pixel. Sie wollen eine Datei von 2,5 MB verschlüsseln. Welchen Teil der Datei können Sie in diesem Bild verschlüsseln? Welchen Teil könnten Sie verschlüsseln, wenn Sie die Datei auf ein Viertel ihrer Originalgröße komprimieren würden? Zeigen Sie Ihre Berechnungen.
- 45** Das Bild in Abbildung 8.54b enthält den ASCII-Text von fünf Shakespeare-Werken. Könnte man anstatt Text auch Musik in den Zebras verstecken? Wenn ja, wie würde dies funktionieren und wie viel könnte in dem Bild untergebracht werden? Wenn nicht, warum?

- 46** Sie erhalten eine Textdatei von 60 MB, die mittels Steganografie in den niederwertigen Bits einer jeden Farbe in einer Bilddatei verschlüsselt werden soll. Wie groß muss das Bild sein, um die ganze Datei zu verschlüsseln? Welche Größe wird benötigt, wenn die Datei zuvor auf ein Drittel ihrer Originalgröße komprimiert wird? Geben Sie Ihre Antwort in Pixel und zeigen Sie Ihre Berechnungen. Gehen Sie davon aus, dass die Bilder ein Seitenverhältnis von 3:2 haben, z. B. 3 000×2 000 Pixel.
- 47** Alice war eine eifrige Benutzerin eines anonymen Typ-1-Remailers. Sie postete viele Nachrichten in ihrer Lieblings-Newsgruppe *alt.fanclub.alice*, und jeder wusste, dass alles von ihr stammte, da alle Nachrichten das gleiche Pseudonym trugen. Wenn der Remailer korrekt arbeitet, kann Trudy Alice nicht verkörpern. Nachdem die Typ-1-Remailer alle abgeschaltet wurden, wechselte Alice zu einem Cypherpunkt-Remailer und startete einen neuen Thread in ihrer Newsgruppe. Entwickeln Sie für Alice eine Möglichkeit, um Trudy davon abzuhalten, neue Nachrichten als Alice in der Newsgruppe zu posten.
- 48** Durchsuchen Sie das Internet nach einem interessanten Fall im Bereich Datenschutz und schreiben Sie einen einseitigen Bericht darüber.
- 49** Durchsuchen Sie das Internet nach einem Rechtsfall im Bereich Urheberrecht gegen angemessene Verwendung und schreiben Sie einen einseitigen Bericht darüber.
- 50** Schreiben Sie ein Programm, das seine Eingaben verschlüsselt, indem es diese mit einem Schlüsselstrom über XOR verknüpft. Suchen oder schreiben Sie einen möglichst guten Zufallszahlengenerator, um den Schlüsselstrom zu erzeugen. Das Programm sollte als Filter fungieren, Klartext von der Standardeingabe entgegennehmen und Chiffretext auf der Standardausgabe ausgeben (und umgekehrt). Das Programm sollte einen Parameter annehmen, den Schlüssel, der die Eingabe für den Zufallszahlengenerator bildet.
- 51** Schreiben Sie eine Prozedur, die den SHA-1-Hash eines Datenblocks berechnet. Diese Prozedur sollte zwei Parameter haben: einen Zeiger auf den Eingabepuffer und einen Zeiger auf einen 20-Byte-Ausgabepuffer. Um die exakte Spezifikation von SHA-1 kennenzulernen, durchsuchen Sie das Internet nach der vollständigen Spezifikation FIPS 180-1.
- 52** Schreiben Sie eine Funktion, die einen Strom von ASCII-Zeichen entgegennimmt und diese Eingabe unter Verwendung einer Substitutionschiffre mit dem Cipher-Block-Chaining-Modus verschlüsselt. Die Blockgröße sollte 8 Byte betragen. Das Programm sollte Klartext von der Standardeingabe entgegennehmen und den Chiffretext auf der Standardausgabe ausgeben. Bei diesem Problem dürfen Sie jedes beliebige System wählen, um festzulegen, dass das Ende der Eingabe erreicht ist und/oder wann der Block zur Vervollständigung aufgefüllt werden sollte. Sie können jedes Ausgabeformat wählen, solange es eindeutig ist. Das Programm sollte zwei Parameter empfangen:
- einen Zeiger auf den Initialisierungsvektor
 - eine Zahl k , die die Verschiebung der Substitutionschiffre angibt, derzufolge jedes ASCII-Zeichen durch das k -te Zeichen davor im Alphabet verschlüsselt wird.
- Wenn beispielsweise $k=3$ ist, wird A durch D ersetzt, B durch E usw. Legen Sie angemessene Annahmen hinsichtlich dem Erreichen des letzten Zeichens im ASCII-Satz zugrunde. Achten Sie darauf, in Ihrem Code deutlich alle Annahmen zu dokumentieren, die Sie über die Eingabe und den Verschlüsselungsalgorithmus machen.

53 Der Zweck dieser Aufgabe ist, Ihnen ein besseres Verständnis der Mechanismen von RSA zu geben. Schreiben Sie eine Funktion, die als Parameter die Primzahlen p und q entgegen nimmt, mit diesen Parametern öffentliche und private RSA-Schlüssel berechnet und n , z , d und e auf der Standardausgabe ausgibt. Die Funktion sollte auch einen Strom von ASCII-Zeichen annehmen und diese Eingabe mit den berechneten RSA-Schlüsseln verschlüsseln. Das Programm sollte Klartext von der Standardeingabe entgegennehmen und den Chiffertext auf der Standardausgabe ausgeben. Die Verschlüsselung sollte zeichenweise ausgeführt werden, das heißt, jedes Zeichen aus der Eingabe sollte unabhängig von den anderen Zeichen in der Eingabe verschlüsselt werden. Bei diesem Problem dürfen Sie jedes beliebige System wählen, um festzulegen, dass das Ende der Eingabe erreicht ist. Sie können jedes AusgabefORMAT wählen, solange es eindeutig ist. Achten Sie darauf, in Ihrem Code deutlich alle Annahmen zu dokumentieren, die Sie über die Eingabe und den Verschlüsselungsalgorithmus machen.

Leseempfehlungen und Bibliografie

- | | | |
|-----|---|-----|
| 9.1 | Empfehlungen für weiterführende Literatur | 990 |
| 9.2 | Alphabetische Bibliografie | 997 |

9

ÜBERBLICK

» Unsere Untersuchung von Rechnernetzen ist nun abgeschlossen, aber das ist erst der Anfang. Viele interessante Themen wurden nicht so ausführlich behandelt, wie sie das eigentlich verdient hätten. Andere wurden aus Platzmangel ganz weggelassen. In diesem Kapitel finden Sie Vorschläge für weiterführende Literatur und ein Literaturverzeichnis für den Fall, dass Sie Ihr Wissen über Rechnernetze noch vertiefen wollen. «

9.1 Empfehlungen für weiterführende Literatur

Auf dem Markt ist umfassende Literatur über alle Aspekte von Rechnernetzen erhältlich. Zwei Zeitschriften, die Artikel auf diesem Gebiet veröffentlichen, sind *IEEE/ACM Transactions on Communications* und *IEEE Journal on Selected Areas in Communications*.

IEEE gibt außerdem die drei Fachzeitschriften *IEEE Internet Computing*, *IEEE Network Magazine* und *IEEE Communications Magazine* heraus, die Berichte, Tutorials und Fallstudien über Netzwerke enthalten. Die ersten beiden beschäftigen sich mit der Architektur, den Standards und der Software, die dritte geht in Richtung Kommunikationstechnologie (Lichtwellenleiter, Satelliten etc.).

Jährlich oder alle zwei Jahre finden verschiedene Konferenzen statt, die viele Berichte über Netze veröffentlichen. Insbesondere sollten Sie nach folgenden Konferenzen Ausschau halten: *SIGCOMM*, *NSDI (Symposium on Networked Systems Design and Implementation)*, *MobiSys (Conference on Mobile Systems, Application and Services)*, *SOSP (Symposium on Operating Systems Principles)* und *OSDI (Symposium on Operating Systems Design and Implementation)*.

Es folgt nun eine Liste mit zusätzlichen Literaturempfehlungen, die nach den Kapiteln in diesem Buch aufgebaut ist. Viele davon sind Bücher oder Buchkapitel, einige auch Tutorials und Übersichten. Die vollständige Biografie folgt in Abschnitt 9.2.

9.1.1 Einführung und allgemeine Werke

Comer: *The Internet Book, 4. Auflage*

Dieses Buch ist für jeden ein Leckerbissen, der nach einer leichten Einführung in das Internet sucht. Comer beschreibt Geschichte, Wachstum, Technologien, Protokolle und Dienste im Internet in einer Sprache, die auch der Neuling versteht. Das Buch deckt aber derart viel Material ab, dass es auch für technisch versierte Leser interessant ist.

Computer Communication Review, Ausgabe zum 25. Jubiläum, Jan. 1995

In dieser speziellen Ausgabe wurden einige wichtige Aufsätze aus den Jahren bis 1995 zusammengetragen, die eine Perspektive aus erster Hand ermöglichen, wie sich das Internet entwickelt hat. Darunter sind Artikel zur Entwicklung von TCP, Multicasting, DNS, Ethernet und der allgemeinen Architektur.

Crovella und Krishnamurthy: *Internet Measurement*

Woran erkennen wir, wie gut das Internet wirklich funktioniert? Diese Frage ist nicht einfach zu beantworten, weil niemand für das Internet verantwortlich ist. Dieses Buch beschreibt die Techniken, die entwickelt wurden, um die Arbeitsweise des Internets zu messen – von der Netzinfrastruktur bis zu Anwendungen.

IEEE Internet Computing, Jan.–Feb. 2000

Die erste Ausgabe von *IEEE Internet Computing* im neuen Jahrtausend hat genau das gemacht, was man erwarten würde: Es hat die Leute, die an der Erstellung des Internets im alten Jahrtausend beteiligt waren, gefragt, wie es im neuen Jahrtausend wohl weitergehen wird. Zu den befragten Experten gehören Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy und andere. Wie gut diese Vorhersagen waren, kann man nun, über ein Jahrzehnt später, hier nachlesen.

Kipnis: „Beating the System: Abuses of the Standards Adoption Process“

Standardisierungsgremien versuchen, gerecht und herstellerneutral zu sein, aber leider gibt es Unternehmen, die versuchen, das System auszunutzen. Es ist beispielsweise wiederholt vorgekommen, dass ein Unternehmen an der Erarbeitung eines Standards mitwirkt und dann, nachdem dieser anerkannt ist, ankündigt, dass er auf einem eigenen Patent basiert und dass nur bestimmte Unternehmen Lizenzen erhalten, andere aber nicht, wobei der Preis alleine von diesem Unternehmen bestimmt wird. Wenn Sie die dunklen Seiten der Standardisierung kennenlernen möchten, ist dieser Artikel ein exzellenter Einstieg.

Naughton: *A Brief History of the Future*

Wer hat eigentlich das Internet erfunden? Viele Leute haben dies für sich beansprucht. Und dies rechtmäßig, weil viele Leute auf verschiedene Arten daran beteiligt waren. So zum Beispiel Paul Baran, der einen Bericht über Paketvermittlung geschrieben hat, die Mitarbeiter verschiedener Universitäten, die die APRANET-Architektur entworfen haben, die Angestellten von BBN, die die ersten IMPs programmierten, nicht zuletzt Bob Kahn und Vint Cerf, die TCP/IP entwickelt haben, und so weiter. Dieses Buch erzählt die Geschichte des Internets (zumindest bis zum Jahr 2000), versehen mit vielen Anekdoten.

9.1.2 Bitübertragungsschicht

Bellamy: *Digital Telephony*, 3. Auflage

Möchten Sie auf die Entwicklung des anderen wichtigen Netzes – das Telefonnetz – zurückblicken, so finden Sie in diesem maßgebenden Buch alles, was Sie schon immer über das Telefonsystem wissen wollten – und mehr. Besonders interessant sind die Kapitel über Übertragungstechniken und Multiplexing, digitale Vermittlung, Glasfaser, Mobiltelefonie und DSL.

Hu und Li: „Satellite-Based Internet: A Tutorial“

Der Internetzugang über Satellit ist anders als die Nutzung über terrestrische Leitungen. Hier gibt es nicht nur das Problem der Übertragungsverzögerung, auch Routing und die Vermittlung sind hier anders. In diesem Artikel untersuchen die Autoren die Themen im Zusammenhang mit dem Einsatz von Satelliten für den Internetzugang.

Joel: „Telecommunications and the IEEE Communications Society“

Wenn Sie eine kompakte, aber erstaunlich umfassende Geschichte der Telekommunikation suchen, die mit dem Telegrafen beginnt und mit IEEE 802.11 endet, ist dieser Artikel genau richtig. Darüber hinaus werden Radio, Telefone, analoge und digitale Vermittlung, Unterwasserkabel, digitale Übertragungen, Fernsehausstrahlungen, Satelliten, Kabelfernsehen, optische Kommunikation, Mobiltelefone, Paketvermittlung, ARPANET und das Internet behandelt.

Palais: *Fiber Optic Communication, 5. Auflage*

Bücher über Glasfasertechnik sind meist für den Fachmann geschrieben. Dieses hier ist eine wohltuende Ausnahme. Behandelt werden Themen wie Wellenleiter, Lichtquellen, Lichtdetektoren, Koppler, Modulationstechniken, Rauschen und vieles mehr.

Su: *The UMTS Air Interface in RF Engineering*

Dieses Buch bietet einen detaillierten Überblick auf eines der wichtigsten zellulären 3G-Systeme. Im Mittelpunkt stehen die Luftschnittstelle bzw. drahtlose Protokolle, die zwischen mobilen Geräten und der Netzinfrastruktur eingesetzt werden.

Want: *RFID Explained*

Das Buch von Want ist eine leicht zu lesende Einführung darin, wie die ungewöhnliche Technologie der RFID-Bitübertragungsschicht funktioniert. Es behandelt alle Aspekte von RFID, einschließlich der möglichen Anwendungen. Einige praktische Einsatzbeispiele von RFID-Installationen sowie die Erfahrung, die daraus gewonnen wurde, werden ebenfalls besprochen.

9.1.3 Sicherungsschicht

Kasim: *Delivering Carrier Ethernet*

Heute ist Ethernet nicht nur eine Technologie für den lokalen Einsatz. Die neue Mode ist, Ethernet als eine Langstreckenverbindung für Carrier-grade-Ethernet zu benutzen. Dieses Buch trägt Artikel zusammen, die das Thema eingehend beleuchten.

Lin und Costello: *Error Control Coding, 2. Auflage*

Codes zum Entdecken und Korrigieren von Fehlern sind für zuverlässige Rechnernetze von zentraler Bedeutung. Dieses beliebte Lehrbuch erklärt einige der wichtigsten Codes, von einfachen linearen Hamming-Codes bis hin zu komplexeren Paritätsprüfcodes wie LDPC (*Low-Density Parity Check Code*). Es wird hierbei versucht, mit einem Minimum an Algebra auszukommen, doch dies ist immer noch eine Menge.

Stallings: *Data and Computer Communications, 9. Auflage*

Teil 2 behandelt digitale Datenübertragung und eine Vielzahl von Verbindungen, einschließlich Fehlererkennung, Fehlerkorrektur mit Neuübertragungen und Flusskontrolle.

9.1.4 MAC-Teilschicht

Andrews et al.: *Fundamentals of WiMAX*

Dieses Buch behandelt die WiMAX-Technologie umfassend und genau – angefangen bei der Idee des drahtlosen Breitbandzugangs über die drahtlosen Techniken, die OFDM und mehrere Antennen verwenden, bis hin zum Mehrfachzugangssystem. Der Aufbau im Stil eines Lernprogramms bietet den besten Zugang, den Sie zu diesem schweren Stoff finden werden.

Gast: *802.11 Wireless Networks, 2. Auflage*

Wer sich eine lesbare Einführung in Technologie und Protokolle von IEEE 802.11 wünscht, wird hier fündig. Das Buch beginnt bei der MAC-Teilschicht, führt dann in Fragestellungen zu unterschiedlichen Bitübertragungsschichten ein und behandelt außerdem das Thema Sicherheit. Die zweite Auflage ist jedoch nicht neu genug, um viel zu IEEE 802.11n sagen zu können.

Perlman: *Interconnections, 2. Auflage*

Mit diesem kompetenten und unterhaltsamen Buch ist Perlman eine hervorragende Arbeit über Bridges und Router gelungen. Die Autorin hat den Algorithmus entwickelt, der in der Spannbaum-Bridge von IEEE 802 verwendet wird. Sie ist eine der weltweit führenden Experten in verschiedenen Netzthemen.

9.1.5 Vermittlungsschicht

Comer: *Internetworking with TCP/IP, Band 1, 5. Auflage*

Comer hat das Standardwerk über die TCP/IP-Protokollsuite geschrieben, das nun in der fünften Auflage erschienen ist. In der ersten Hälfte des Buchs werden IP und verwandte Protokolle der Vermittlungsschicht beschrieben. Die anderen Kapitel behandeln vor allem die höheren Schichten und sind ebenso lesenswert.

Grayson et al.: *IP Design for Mobile Networks*

Traditionelle Telefonnetze und das Internet befinden sich auf einem Kollisionskurs – und die mit IP implementierten Mobilfunknetze befinden sich mittendrin. Dieses Buch erklärt, wie ein Netz, das die IP-Protokolle verwendet, entworfen wird und wie es mobile Telefondienste unterstützt.

Huitema: *Routing in the Internet, 2. Auflage*

Wenn Sie ein tief gehendes Verständnis von Routing-Protokollen gewinnen wollen, ist dies ein sehr gutes Buch für Sie. Sowohl aussprechbare Algorithmen (wie RIP und CIDR) als auch unaussprechbare (wie OSPF, IGRP und BGP) werden sehr ausführlich behandelt. Neuere Entwicklungen sind nicht enthalten, da dies ein älteres Buch ist, aber dafür werden die behandelten Themen sehr gut erklärt.

Koodli und Perkins: *Mobile Inter-networking with IPv6*

Zwei wichtige Entwicklungen der Vermittlungsschicht werden in einem Band erklärt: IPv6 und mobiles IP. Beide werden ausführlich dargestellt. Perkins war einer der treibenden Kräfte hinter mobilem IP.

Nucci und Papagiannaki: *Design, Measurement and Management of Large-Scale IP Networks*

Wir haben viel darüber gesprochen, wie Netze arbeiten, doch nicht darüber, wie Sie Netze entwerfen, installieren und verwalten würden, wenn Sie ein ISP wären. Dieses Buch füllt die Lücke und stellt moderne Methoden zum Traffic-Engineering vor und behandelt, wie ISPs Dienste zur Netznutzung anbieten.

Perlman: *Interconnections, 2. Auflage*

In den Kapiteln 12 bis 15 beschreibt Perlman viele Aspekte des Entwurfs von Algorithmen für Unicast- und Multicast-Routing sowohl für WANs wie auch für Netzwerke aus LANs. Aber der bei Weitem beste Teil des Buches ist Kapitel 18, in dem die Autorin ihre jahrelange Erfahrung im Bereich der Netzprotokolle in einem informativen und unterhaltsamen Kapitel zusammenfasst. Es ist eine Pflichtlektüre für Protokollentwickler.

Stevens: *TCP/IP Illustrated, Band 1*

In den Kapiteln 3 bis 10 werden TCP und die dazugehörigen Protokolle (ARP, RARP und ICMP) umfassend behandelt und durch zahlreiche Beispiele verdeutlicht.

Varghese: *Network Algorithmics*

Wir haben viel Zeit investiert, darüber zu sprechen, wie Router und andere Netzelemente miteinander interagieren. Dieses Buch ist anders: Hier geht es darum, wie Router praktisch entworfen werden, um Pakete mit erstaunlichen Geschwindigkeiten weiterzuleiten. Wer Interesse an diesem Insiderblick und an verwandten Fragestellungen hat, für den ist dies das richtige Buch. Der Autor ist eine Autorität im Bereich intelligenter Algorithmen, die in der Praxis zur Implementierung von Elementen für Hochgeschwindigkeitsnetze in Soft- und Hardware eingesetzt werden.

9.1.6 Transportschicht

Comer: *Internetworking with TCP/IP, Band 1, 5. Auflage*

Wie oben erwähnt, hat Comer ein Standardwerk über die TCP/IP-Protokollsuite geschrieben. Die zweite Hälfte des Buchs behandelt UDP und TCP.

Farrell und Cahill: *Delay- and Disruption-Tolerant Networking*

Dieses kurze Buch ist für diejenigen geeignet, die einen tiefer gehenden Einblick in Architektur, Protokolle und Anwendungen von sogenannten „Challenged Networks“ wünschen, die unter schwierigen Verbindungsverhältnissen arbeiten müssen. Die Autoren haben an der Entwicklung von DTNs innerhalb der IETF DTN Research Group mitgewirkt.

Stevens: *TCP/IP Illustrated, Band 1*

In den Kapiteln 17 bis 24 wird TCP umfassend behandelt und durch zahlreiche Beispiele verdeutlicht.

9.1.7 Anwendungsschicht

Berners-Lee et al.: „The World Wide Web“

Unternehmen Sie eine Reise in die Vergangenheit und erleben Sie das Web und seine Zukunft der Perspektive seines Erfinders und einiger seiner Kollegen von CERN. Der Artikel konzentriert sich auf die Webarchitektur, URLs, HTTP und HTML sowie auf künftige Trends und vergleicht das Web mit anderen verteilten Informationssystemen.

Held: *A Practical Guide to Content Delivery Networks, 2. Auflage*

Dieses Buch bietet eine bodenständige Darstellung, wie CDNs arbeiten, wobei besonders die praktischen Überlegungen beim Entwurf und Betrieb eines CDN mit guter Performanz hervorgehoben werden.

Hunter et al.: *Beginning XML, 4. Auflage*

Es gibt viele, viele Bücher zu HTML, XML und Webdiensten. Dieses 1000-Seiten-Buch deckt vermutlich das meiste von dem ab, was Sie wissen wollen. Es erklärt nicht nur, wie man in XML und XHTML schreibt, sondern auch, wie Webdienste entwickelt werden, die in der Praxis häufig eingesetzt werden.

Krishnamurthy und Rexford: *Web Protocols and Practice*

Es ist schwer, ein umfassenderes Buch über alle Aspekte des Webs zu finden als dieses. Wie man erwarten würde, behandelt es Clients, Server, Proxys und Caching. Es gibt auch Kapitel über Datenverkehr im Web und Messtechniken sowie ferner Kapitel über die aktuelle Forschung und die Optimierung des Webs.

Simpson: *Video Over IP, 2. Auflage*

Der Autor betrachtet aus der Weitwinkelperspektive, wie IP-Technologie benutzt werden kann, um Videodaten über Netze zu transportieren, sowohl über das Internet als

auch über private Netze, die zum Übertragen von Video entworfen wurden. Interessanterweise ist dieses Buch eher auf Video-Fachleute ausgerichtet, die etwas über Netze lernen möchten, als umgekehrt.

Wittenburg: *Understanding Voice Over IP Technology*

Dieses Buch beschreibt, wie Voice-over-IP funktioniert – vom Übertragen der Audiodaten mit den IP-Protokollen und Fragen der Dienstgüte, über die SIP- und H.323-Protokollsuites. Angesichts des Materials ist die Darstellung zwangsläufig ausführlich, aber zugänglich aufbereitet und in überschaubare Einheiten aufgeteilt.

9.1.8 Netzsicherheit

Anderson: *Security Engineering, 2. Auflage*

Dieses Buch bietet eine wundervolle Mischung einerseits aus der Darstellung von Sicherheitstechniken und andererseits ein Verständnis dafür zu entwickeln,** wie diese (sinnvoll und missbräuchlich) eingesetzt werden. Es ist technischer als *Secrets and Lies*, aber weniger technisch als *Network Security* (siehe unten). Nach einer Einführung in die grundlegenden Sicherheitstechniken sind ganze Kapitel verschiedenen Anwendungen, wie Einsatz im Bankbereich, Befehle für und Steuerung von Atomkraftwerken, Sicherheitsdruck, Biometrik, elektronische Kriegsführung, Telekommunikationssicherheit, E-Commerce und Urheberrechtsschutz gewidmet.

Ferguson et al.: *Cryptography Engineering*

In vielen Büchern können Sie nachlesen, wie die populären kryptografischen Algorithmen funktionieren. Dieses Buch erzählt Ihnen, wie Sie Kryptografie verwenden – warum kryptografische Protokolle auf genau diese Art entworfen werden und wie sie zu einem System zusammengesetzt werden, welches Ihre Sicherheitsziele erfüllen wird. Es ist ein ziemlich kompaktes Buch, das Pflichtlektüre für jeden Entwickler ist, dessen Systeme mit Kryptografie zu tun haben.

Fridrich: *Steganography in Digital Media*

Steganografie geht zurück auf das antike Griechenland, wo Wachs auf leeren Tafeln geschmolzen wurde, sodass geheime Nachrichten auf dem darunterliegenden Holz angebracht werden konnten, bevor der Wachs erneut aufgetragen wurde. Heutzutage stellen Videos, Audiodaten und andere Inhalte im Internet unterschiedliche Träger für geheime Nachrichten dar. Es werden hier verschiedene moderne Techniken zum Verstecken und Auffinden von Informationen in Bildern besprochen.

Kaufman et al.: *Network Security, 2. Auflage*

Dieses kompetente und stellenweise witzige Buch ist eine technischere Informationsquelle für Netzsicherheitsalgorithmen und Protokolle. Algorithmen und Protokolle für symmetrische und öffentliche Schlüssel, Nachrichten-Hashes, Authentifizierung, Kerberos, PKI, IPsec, SSL/TLS und E-Mail-Sicherheit werden ausführlich behandelt. Kapitel 26 über Sicherheitsfolklore ist ein wahrer Schatz. Beim Thema Sicherheit

steckt der Teufel in den Details. Jeder, der plant, ein Sicherheitssystem zu entwerfen, das auch wirklich verwendet wird, wird aus den praxisbezogenen Ratschlägen in diesem Kapitel eine Menge lernen.

Schneier: *Secrets and Lies*

Wenn Sie *Cryptography Engineering* von vorne bis hinten durchgelesen haben, wissen Sie alles, was man über kryptografische Algorithmen wissen muss. Wenn Sie dann *Secrets and Lies* von vorne bis hinten durchlesen (wozu man erheblich weniger Zeit braucht), erfahren Sie, dass kryptografische Algorithmen nicht die ganze Geschichte sind. Die meisten Sicherheitsmängel treten nicht aufgrund fehlerhafter Algorithmen oder zu kurzer Schlüssel auf, sondern durch Fehler in der Sicherheitsumgebung. Dieses Buch ist als nicht technische und sehr faszinierende Darstellung der Computersicherheit im weitesten Sinn sehr empfehlenswert.

Skoudis und Liston: *Counter Hack Reloaded, 2. Auflage*

Die beste Methode, einen Hacker zu stoppen, ist, wie ein Hacker zu denken. Dieses Buch zeigt, wie Hacker ein Netz sehen, und argumentiert, dass Sicherheit eine Funktion des gesamten Netzentwurfs sein sollte, nicht ein nachträglicher Einfall, der auf einer speziellen Technologie basiert. Es werden beinahe alle gängigen Angriffe behandelt, einschließlich „Social Engineering“-Arten, die die Unkenntnis der Sicherheitsmaßnahmen von Benutzern ausnutzen.

9.2 Alphabetische Bibliografie

Abramson, N.: „Internet Access Using VSATs“, *IEEE Commun. Magazine*, Vol. 38, S. 60–68, Juli 2000.

Ahmadi, S.: „An Overview of Next-Generation Mobile WiMAX Technology“, *IEEE Commun. Magazine*, Vol. 47, S. 84–88, Juni 2009.

Allman, M., und Paxson, V.: „On Estimating End-to-End Network Path Properties“, *Proc. SIGCOMM '99 Conf.*, ACM, S. 263–274, 1999.

Anderson, C.: *The Long Tail: Why the Future of Business is Selling Less of More*, überarbeitete und aktualisierte Auflage, New York, Hyperion, 2008a.

Anderson, R. J.: *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2. Auflage, New York, John Wiley & Sons, 2008b.

Anderson, R. J.: „Free Speech Online and Offline“, *IEEE Computer*, Vol. 25, S. 28–30, Juni 2002.

Anderson, R. J.: „The Eternity Service“, *Proc. Pragocrypt Conf.*, CTU Publishing House, S. 242–252, 1996.

Andrews, J., A. Ghosh, und R. Muhammed: *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*, Upper Saddle River, NJ, Pearson Education, 2007.

- Astely, D., E. Dahlman, A. Furuskär, Y. Jading, M. Lindstrom und S. Parkvall: „LTE: The Evolution of Mobile Broadband“, *IEEE Commun. Magazine*, Vol. 47, S. 44–51, April 2009.
- Ballardie, T., P. Francis, P. und J. Crowcroft: „Core Based Trees (CBT)“, *Proc. SIGCOMM '93 Conf.*, ACM, S. 85–95, 1993.
- Baran, P.: „On Distributed Communications: I. Introduction to Distributed Communication Networks“, *Memorandum RM-420-PR*, Rand Corporation, August 1964.
- Bellamy, J.: *Digital Telephony*, 3. Auflage, New York, John Wiley & Sons, 2000.
- Bellman, R. E.: *Dynamic Programming*, Princeton, NJ, Princeton University Press, 1957.
- Bellovin, S.: „The Security Flag in the IPv4 Header“, *RFC 3514*, April 2003.
- Belsnes, D.: „Flow Control in the Packet Switching Networks“, *Communications Networks*, Uxbridge, England: Online, S. 349–361, 1975.
- Bennet, C. H., und Brassard, G.: „Quantum Cryptography: Public Key Distribution and Coin Tossing“, *International Conf. on Computer Systems and Signal Processing*, S. 175–179, 1984.
- Beresford, A., und Stajano, F.: „Location Privacy in Pervasive Computing“, *IEEE Pervasive Computing*, Vol. 2, S. 46–55, Januar 2003.
- Berghel, H. L.: „Cyber Privacy in the New Millennium“, *IEEE Computer*, Vol. 34, S. 132–134, Januar 2001.
- Berners-Lee, T., A. Cailliau, A. Loutonen, H. F. Nielsen, H. F. und A. Secret: „The World Wide Web“, *Commun. of the ACM*, Vol. 37, S. 76–82, August 1994.
- Bertsekas, D., und R. Gallager: *Data Networks*, 2. Auflage, Englewood Cliffs, NJ, Prentice Hall, 1992.
- Bhatti, S. N., und Crowcroft, J.: „QoS Sensitive Flows: Issues in IP Packet Handling“, *IEEE Internet Computing*, Vol. 4, S. 48–57, Juli–August 2000.
- Biham, E., und A. Shamir: „Differential Fault Analysis of Secret Key Cryptosystems“, *Proc. 17th Ann. Int'l Cryptology Conf.*, Berlin, Springer-Verlag LNCS 1294, S. 513–525, 1997.
- Bird, R., I. Gopal, A. Herzberg, P. A. Janson, S. Kutten, R. Molva und M. Yung: „Systematic Design of a Family of Attack-Resistant Authentication Protocols“, *IEEE J. on Selected Areas in Commun.*, Vol. 11, S. 679–693, Juni 1993.
- Birrell, A. D., und B. J. Nelson: „Implementing Remote Procedure Calls“, *ACM Trans. on Computer Systems*, Vol. 2, S. 39–59, Feb. 1984.
- Biryukov, A., A. Shamir und D. Wagner: „Real Time Cryptanalysis of A5/1 on a PC“, *Proc. Seventh Int'l Workshop on Fast Software Encryption*, Berlin, Springer-Verlag LNCS 1978, S. 1–8, 2000.
- Blaze, M., und S. Bellovin: „Tapping on My Network Door“, *Commun. of the ACM*, Vol. 43, S. 136, Oktober 2000.

- Boggs, D., J. Mogul und C. Kent: „Measured Capacity of an Ethernet: Myths and Reality“, *Proc. SIGCOMM '88 Conf.*, ACM, S. 222–234, 1988.
- Borisov, N., I. Goldberg und D. Wagner: „Intercepting Mobile Communications: The Insecurity of 802.11“, *Seventh Int'l Conf. on Mobile Computing and Networking*, ACM, S. 180–188, 2001.
- Braden, R.: „Requirements for Internet Hosts – Communication Layers“, RFC 1122, Oktober 1989.
- Braden, R., D. Borman und C. Partridge: „Computing the Internet Checksum“, *RFC 1071*, September 1988.
- Brandenburg, K.: „MP3 and AAC Explained“, *Proc. 17th Intl. Conf.: High-Quality Audio Coding*, Audio Engineering Society, S. 99–110, August 1999.
- Bray, T., J. Paoli, C. Sperberg-McQueen, E. Maler, F. Yergeau und J. Cowan: „Extensible Markup Language (XML) 1.1 (Second Edition)“, *W3C Recommendation*, September 2006.
- Breslau, L., P. Cao, L. Fan, G. Phillips und S. Shenker: „Web Caching and Zipf-like Distributions: Evidence and Implications“, *Proc. INFOCOM Conf.*, IEEE, S. 126–134, 1999.
- Burleigh, S., A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott und H. Weiss: „Delay-Tolerant Networking: An Approach to Interplanetary Internet“, *IEEE Commun. Magazine*, Vol. 41, S. 128–136, Juni 2003.
- Burnett, S., und S. Paine: *RSA Security's Official Guide to Cryptography*, Berkeley, CA, Osborne/McGraw-Hill, 2001.
- Bush, V.: „As We May Think“, *Atlantic Monthly*, Vol. 176, S. 101–108, Juli 1945.
- Capetanakis, J. I.: „Tree Algorithms for Packet Broadcast Channels“, *IEEE Trans. on Information Theory*, Vol. IT–5, S. 505–515, Sept. 1979.
- Castagnoli, G., S. Brauer und M. Herrmann: „Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits“, *IEEE Trans. on Commun.*, Vol. 41, S. 883–892, Juni 1993.
- Cerf, V. und R. Kahn: „A Protocol for Packet Network Interconnection“, *IEEE Trans. on Commun.*, Vol. COM–2, S. 637–648, Mai 1974.
- Chang, F., J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Handra, A. Fikes und R. Gruber: „Bigtable: A Distributed Storage System for Structured Data“, *Proc. OSDI 2006 Symp.*, USENIX, S. 15–29, 2006.
- Chase, J. S., A. J. Gallatin und K. G. Yocom: „End System Optimizations for High-Speed TCP“, *IEEE Commun. Magazine*, Vol. 39, S. 68–75, April 2001.
- Chen, S. und K. Nahrstedt: „An Overview of QoS Routing for Next-Generation Networks“, *IEEE Network Magazine*, Vol. 12, S. 64–69, Nov./Dez. 1998.
- Chiu, D. und R. Jain: „Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks“, *Comput. Netw. ISDN Syst.*, Vol. 17, S. 1–4, Juni 1989.

Cisco: „Cisco Visual Networking Index: Forecast and Methodology, 2009–2014“, Cisco Systems Inc., Juni 2010.

Clark, D. D.: „The Design Philosophy of the DARPA Internet Protocols“, *Proc. SIGCOMM '88 Conf.*, ACM, S. 106–114, 1988.

Clark, D. D.: „Window and Acknowledgement Strategy in TCP“, *RFC 813*, Juli 1982.

Clark, D. D., V. Jacobson, J. Romkey und H. Salwen: „An Analysis of TCP Processing Overhead“, *IEEE Commun. Magazine*, Vol. 27, S. 23–29, Juni 1989.

Clark, D. D., S. Shenker und L. Zhang: „Supporting Real-Time Applications in an Integrated Services Packet Network“, *Proc. SIGCOMM '92 Conf.*, ACM, S. 14–26, 1992.

Clarke, A. C.: „Extra-Terrestrial Relays“, *Wireless World*, 1945.

Clarke, I., S. G. Miller, T. W. Hong, O. Sandberg und B. Wiley: „Protecting Free Expression Online with Freenet“, *IEEE Internet Computing*, Vol. 6, S. 40–49, Jan.–Feb. 2002.

Cohen, B.: „Incentives Build Robustness in BitTorrent“, *Proc. First Workshop on Economics of Peer-to-Peer Systems*, Juni 2003.

Comer, D. E.: *The Internet Book*, 4. Auflage, Englewood Cliffs, NJ, Prentice Hall, 2007.

Comer, D. E.: *Internetworking with TCP/IP*, Vol. 1, 5. Auflage, Englewood Cliffs, NJ, Prentice Hall, 2005.

Craver, S. A., M. Wu, B. Liu, A. Stubblefield, B. Swartzlander, D. W. Wallach, D. Dean und E. W. Felten: „Reading Between the Lines: Lessons from the SDMI Challenge“, *Proc. 10th USENIX Security Symp.*, USENIX, 2001.

Crovella, M. und B. Krishnamurthy: *Internet Measurement*, New York, John Wiley & Sons, 2006.

Daemen, J. und V. Rijmen: *The Design of Rijndael*, Berlin, Springer-Verlag, 2002.

Dalal, Y. und R. Metcalfe: „Reverse Path Forwarding of Broadcast Packets“, *Commun. of the ACM*, Vol. 21, S. 1040–1048, Dez. 1978.

Davie, B. und A. Farrel: *MPLS: Next Steps*, San Francisco, Morgan Kaufmann, 2008.

Davie, B. und Y. Rekhter: *MPLS Technology and Applications*, San Francisco, Morgan Kaufmann, 2000.

Davies, J.: *Understanding IPv6*, 2. Auflage, Redmond, WA, Microsoft Press, 2008.

Day, J. D.: „The (Un)Revised OSI Reference Model“, *Computer Commun. Rev.*, Vol. 25, S. 39–55, Okt. 1995.

Day, J. D. und H. Zimmermann: „The OSI Reference Model“, *Proc. of the IEEE*, Vol. 71, S. 1334–1340, Dez. 1983.

Decandia, G., D. Hastorin, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall und W. Vogels: „Dynamo: Amazon's Highly Available

Key-value Store“, *Proc. 19th Symp. on Operating Systems Prin.*, ACM, S. 205–220, Dez. 2007.

Deering, S. E.: „SIP: Simple Internet Protocol“, *IEEE Network Magazine*, Vol. 7, S. 16–28, Mai/Juni 1993.

Deering, S. und D. Cheriton: „Multicast Routing in Datagram Networks and Extended LANs“, *ACM Trans. on Computer Systems*, Vol. 8, S. 85–110, Mai 1990.

Demers, A., S. Keshav und S. Shenker: „Analysis and Simulation of a Fair Queueing Algorithm“, *Internetwork: Research and Experience*, Vol. 1, S. 3–26, Sept. 1990.

Denning, D. E. und G. M. Sacco: „Timestamps in Key Distribution Protocols“, *Commun. of the ACM*, Vol. 24, S. 533–536, Aug. 1981.

Devarapalli, V., R. Wakikawa, A. Petrescu und P. Thubert: „Network Mobility (NEMO) Basic Support Protocol“, *RFC 3963*, Jan. 2005.

Diffie, W. und M. E. Hellman: „Exhaustive Cryptanalysis of the NBS Data Encryption Standard“, *IEEE Computer*, Vol. 10, S. 74–84, Juni 1977.

Diffie, W. und M. E. Hellman, M. E.: „New Directions in Cryptography“, *IEEE Trans. on Information Theory*, Vol. IT–2, S. 644–654, Nov. 1976.

Dijkstra, E. W.: „A Note on Two Problems in Connexion with Graphs“, *Numer. Math.*, Vol. 1, S. 269–271, Okt. 1959.

Dilley, J., B. Maggs, J. Parikh, H. Prokop, R. Sitaraman und B. Wheil: „Globally Distributed Content Delivery“, *IEEE Internet Computing*, Vol. 6, S. 50–58, 2002.

Dingledine, R., N. Mathewson, P. Syverson: „Tor: The Second-Generation Onion Router“, *Proc. 13th USENIX Security Symp.*, USENIX, S. 303–320, Aug. 2004.

Donahoo, M. und K. Calvert: *TCP/IP Sockets in C*, 2. Auflage, San Francisco, Morgan Kaufmann, 2009.

Donahoo, M. und K. Calvert: *TCP/IP Sockets in Java*, 2. Auflage, San Francisco, Morgan Kaufmann, 2008.

Donaldson, G. und D. Jones: „Cable Television Broadband Network Architectures“, *IEEE Commun. Magazine*, Vol. 39, S. 122–126, Juni 2001.

Dorfman, R.: „Detection of Defective Members of a Large Population“, *Annals Math. Statistics*, Vol. 14, S. 436–440, 1943.

Dutcher, B.: *The NAT Handbook*, New York, John Wiley & Sons, 2001.

Dutta-Roy, A.: „An Overview of Cable Modem Technology and Market Perspectives“, *IEEE Commun. Magazine*, Vol. 39, S. 81–88, Juni 2001.

Edelman, B., M. Ostrovsky und M. Schwarz: „Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords“, *American Economic Review*, Vol. 97, S. 242–259, März 2007.

El Gamal, T.: „A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms“, *IEEE Trans. on Information Theory*, Vol. IT-1, S. 469–472, Juli 1985.

EPCglobal: *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHz – 960 MHz, Version 1.2.0*, Brüssel, EPCglobal Inc., Okt. 2008.

Fall, K.: „A Delay-Tolerant Network Architecture for Challenged Internets“, *Proc. SIGCOMM 2003 Conf.*, ACM, S. 27–34, Aug. 2003.

Faloutsos, M., P. Faloutsos und C. Faloutsos: „On Power-Law Relationships of the Internet Topology“, *Proc. SIGCOMM '99 Conf.*, ACM, S. 251–262, 1999.

Farrell, S. und V. Cahill: *Delay- and Disruption-Tolerant Networking*, London, Artech House, 2007.

Fellows, D. und D. Jones: „DOCSIS Cable Modem Technology“, *IEEE Commun. Magazine*, Vol. 39, S. 202–209, März 2001.

Fenner, B., M. Handley, H. Holbrook und I. Kouvelas: „Protocol Independent Multicast-Sparse Mode (PIM-SM)“, *RFC 4601*, Aug. 2006.

Ferguson, N., B. Schneider und T. Kohno: *Cryptography Engineering: Design Principles and Practical Applications*, New York, John Wiley & Sons, 2010.

Flanagan, D.: *JavaScript: The Definitive Guide*, 6. Auflage, Sebastopol, CA, O'Reilly, 2010.

Fletcher, J.: „An Arithmetic Checksum for Serial Transmissions“, *IEEE Trans. on Commun.*, Vol. COM-0, S. 247–252, Jan. 1982.

Floyd, S., M. Handley, J. Padhye und J. Widmer: „Equation-Based Congestion Control for Unicast Applications“, *Proc. SIGCOMM 2000 Conf.*, ACM, S. 43–56, Aug. 2000.

Floyd, S. und V. Jacobson: „Random Early Detection for Congestion Avoidance“, *IEEE/ACM Trans. on Networking*, Vol. 1, S. 397–413, Aug. 1993.

Fluhrer, S., I. Mantin und A. Shamir: „Weakness in the Key Scheduling Algorithm of RC4“, *Proc. Eighth Ann. Workshop on Selected Areas in Cryptography*, Berlin, Springer-Verlag LNCS 2259, S. 1–24, 2001.

Ford, B.: „Structured Streams: A New Transport Abstraction“, *Proc. SIGCOMM 2007 Conf.*, ACM, S. 361–372, 2007.

Ford, L. R. Jr. und D.R. Fulkerson: *Flows in Networks*, Princeton, NJ, Princeton University Press, 1962.

Ford, W. und M. S. Baum: *Secure Electronic Commerce*, Upper Saddle River, NJ, Prentice Hall, 2000.

Forney, G. D.: „The Viterbi Algorithm“, *Proc. of the IEEE*, Vol. 61, S. 268–278, März 1973.

Fouli, K. und M. Maler: „The Road to Carrier-Grade Ethernet“, *IEEE Commun. Magazine*, Vol. 47, S. S30–S38, März 2009.

Fox, A., S. Gribble, E. Brewer und E. Amir: „Adapting to Network and Client Variability via On-Demand Dynamic Distillation“, *SIGOPS Oper. Syst. Rev.*, Vol. 30, S. 160–170, Dez. 1996.

Francis, P.: „A Near-Term Architecture for Deploying Pip“, *IEEE Network Magazine*, Vol. 7, S. 30–37, Mai/Juni 1993.

Fraser, A. G.: „Towards a Universal Data Transport System“, *IEEE J. on Selected Areas in Commun.*, Vol. 5, S. 803–816, Nov. 1983.

Fridrich, J.: *Steganography in Digital Media: Principles, Algorithms und Applications*, Cambridge, Cambridge University Press, 2009.

Fuller, V. und T. Li: „Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan“, *RFC 4632*, Aug. 2006.

Gallagher, R. G.: „A Minimum Delay Routing Algorithm Using Distributed Computation“, *IEEE Trans. on Commun.*, Vol. COM–5, S. 73–85, Jan. 1977.

Gallagher, R. G.: „Low-Density Parity Check Codes“, *IRE Trans. on Information Theory*, Vol. 8, S. 21–28, Jan. 1962.

Garfinkel, S. mit G. Spafford: *Web Security, Privacy und Commerce*, Sebastopol, CA, O'Reilly, 2002.

Gast, M.: *802.11 Wireless Networks: The Definitive Guide*, 2. Auflage, Sebastopol, CA, O'Reilly, 2005.

Gershenfeld, N. und R. Krikorian und D. Cohen: „The Internet of Things“, *Scientific American*, Vol. 291, S. 76–81, Okt. 2004.

Gilder, G.: „Metcalfe's Law and Legacy“, *Forbes ASAP*, Sept. 13, 1993.

Goode, B.: „Voice over Internet Protocol“, *Proc. of the IEEE*, Vol. 90, S. 1495–1517, Sept. 2002.

Goralski, W. J.: *SONET*, 2. Auflage, New York, McGraw-Hill, 2002.

Grayson, M., K. Shatzkamer und S. Wainner: *IP Design for Mobile Networks*. Cisco Press, Indianapolis, 2009.

Grobe, K. und J. Elbers: „PON in Adolescence: From TDMA to WDM-PON“, *IEEE Commun. Magazine*, Vol. 46, S. 26–34, Jan. 2008.

Gross, G., M. Kaycee, A. Lin, A. Malis und J. Stephens: „The PPP Over AAL5“, *RFC 2364*, Juli 1998.

Ha, S., I. Rhee und X. Lisong: „CUBIC: A New TCP-Friendly High-Speed TCP Variant“, *SIGOPS Oper. Syst. Rev.*, Vol. 42, S. 64–74, Juni 2008.

Hafner, K. und M. Lyon: *Where Wizards Stay Up Late*. Simon & Schuster, New York, 1998.

Halperin, D., T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno und W. Maisel: „Pacemakers and Implantable Cardiac Defibrillators: Soft-

ware Radio Attacks and Zero-Power Defenses“, *IEEE Symp. on Security and Privacy*, S. 129–142, Mai 2008.

Halperin, D., W. Hu, A. Sheth und D. Wetherall: „802.11 with Multiple Antennas for Dummies“, *Computer Commun. Rev.*, Vol. 40, S. 19–25, Jan. 2010.

Hamming, R. W.: „Error Detecting and Error Correcting Codes“, *Bell System Tech. J.*, Vol. 29, S. 147–160, April 1950.

Harte, L., S. Kellogg, R. Dreher und T. Schaffnit: *The Comprehensive Guide to Wireless Technology*. APDG Publishing, Fuquay-Varina, 2000.

Hawley, G. T.: „Historical Perspectives on the U.S. Telephone Loop“, *IEEE Commun. Magazine*, Vol. 29, S. 24–28, März 1991.

Hecht, J.: *Understanding Fiber Optics*. Prentice Hall, Upper Saddle River, 2005.

Held, G.: *A Practical Guide to Content Delivery Networks*, 2. Auflage. CRC Press, Boca Raton, 2010.

Heusse, M., F. Rousseau, G. Berger-Sabbatel, A. Duda: „Performance Anomaly of 802.11b“, *Proc. INFOCOM Conf.*, IEEE, S. 836–843, 2003.

Hiertz, G., D. Denteneer, L. Stibor, Y. Zang, X. Costa und B. Walke: „The IEEE 802.11 Universe“, *IEEE Commun. Magazine*, Vol. 48, S. 62–70, Jan. 2010.

Hoe, J.: „Improving the Start-up Behavior of a Congestion Control Scheme for TCP“, *Proc. SIGCOMM '96 Conf.*, ACM, S. 270–280, 1996.

Hu, Y. und V. O. K. Li: „Satellite-Based Internet: A Tutorial“, *IEEE Commun. Magazine*, Vol. 30, S. 154–162, März 2001.

Huitema, C.: *Routing in the Internet*, 2. Auflage. Prentice Hall, Englewood Cliffs, 1999.

Hull, B., V. Bychokovsky, K. Chen, M. Goraczko, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan und S. Madden: „CarTel: A Distributed Mobile Sensor Computing System“, *Proc. Sensys 2006 Conf.*, ACM, S. 125–138, Nov. 2006.

Hunter, D., J. Rafter, J. Fawcett, E. van der List, D. Ayers, J. Duckett, A. Watt und L. McKinnon: *Beginning XML*, 4. Auflage. Wrox, New Jersey, 2007.

Irmer, T.: „Shaping Future Telecommunications: The Challenge of Global Standardization“, *IEEE Commun. Magazine*, Vol. 32, S. 20–28, Jan. 1994.

ITU (International Telecommunication Union): *ITU Internet Reports 2005: The Internet of Things*. ITU, Genf, Nov. 2005.

ITU (International Telecommunication Union): *Measuring the Information Society: The ICT Development Index*. ITU, Genf, März 2009.

Jacobson, V.: „Compressing TCP/IP Headers for Low-Speed Serial Links“, *RFC 1144*, Feb. 1990.

- Jacobson, V.: „Congestion Avoidance and Control“, *Proc. SIGCOMM '88 Conf.*, ACM, S. 314–329, 1988.
- Jain, R. und S. Routhier: „Packet Trains—Measurements and a New Model for Computer Network Traffic“, *IEEE J. on Selected Areas in Commun.*, Vol. 6, S. 986–995, Sept. 1986.
- Jakobsson, M. und S. Wetzel: „Security Weaknesses in Bluetooth“, *Topics in Cryptology: CT-RSA 2001*, Springer-Verlag LNCS 2020, Berlin, S. 176–191, 2001.
- Joel, A.: „Telecommunications and the IEEE Communications Society“, *IEEE Commun. Magazine*, 50th Anniversary Issue, S. 6–14 und 162–167, Mai 2002.
- Johnson, D., C. Perkins und J. Arkko: „Mobility Support in IPv6“, *RFC 3775*, Juni 2004.
- Johnson, D. B., D. Maltz und J. Broch: „DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks“, *Ad Hoc Networking*, Addison-Wesley, Boston, S. 139–172, 2001.
- Juang, P., H. Oki, Y. Wang, M. Martonosi, L. Peh und D. Rubenstein: „Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet“, *SIGOPS Oper. Syst. Rev.*, Vol. 36, S. 96–107, Okt. 2002.
- Kahn, D.: *The Codebreakers, 2. Auflage*. Macmillan, New York, 1995.
- Kamoun, F. und L. Kleinrock: „Stochastic Performance Evaluation of Hierarchical Routing for Large Networks“, *Computer Networks*, Vol. 3, S. 337–353, Nov. 1979.
- Karn, P.: „MACA – A New Channel Access Protocol for Packet Radio“, *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.*, S. 134–140, 1990.
- Karn, P. und C. Partridge: „Improving Round-Trip Estimates in Reliable Transport Protocols“, *Proc. SIGCOMM '87 Conf.*, ACM, S. 2–7, 1987.
- Karp, B. und H. T. Kung: „GPSR: Greedy Perimeter Stateless Routing for Wireless Networks“, *Proc. MOBICOM 2000 Conf.*, ACM, S. 243–254, 2000.
- Kasim, A.: *Delivering Carrier Ethernet*. McGraw-Hill, New York, 2007.
- Katabi, D., M. Handley und C. Rohrs: „Internet Congestion Control for Future High Bandwidth-Delay Product Environments“, *Proc. SIGCOMM 2002 Conf.*, ACM, S. 89–102, 2002.
- Katz, D. und P. S. Ford: „TUBA: Replacing IP with CLNP“, *IEEE Network Magazine*, Vol. 7, S. 38–47, Mai/Juni 1993.
- Kaufman, C., R. Perlman und M. Speciner: *Network Security, 2. Auflage*. Prentice Hall, Englewood Cliffs, 2002.
- Kent, C. und J. Mogul: „Fragmentation Considered Harmful“, *Proc. SIGCOMM '87 Conf.*, ACM, S. 390–401, 1987.
- Kerckhoffs, A.: „La Cryptographie Militaire“, *J. des Sciences Militaires*, Vol. 9, S. 5–38, Jan. 1883 und S. 161–191, Feb. 1883.

- Khanna, A. und J. Zinky: „The Revised ARPANET Routing Metric“, *Proc. SIGCOMM '89 Conf.*, ACM, S. 45–56, 1989.
- Kipnis, J.: „Beating the System: Abuses of the Standards Adoption Process“, *IEEE Commun. Magazine*, Vol. 38, S. 102–105, Juli 2000.
- Kleinrock, L.: „Power and Other Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications“, *Proc. Intl. Conf. on Commun.*, S. 43.1.1–43.1.10, Juni 1979.
- Kleinrock, L. und F. Tobagi, F.: „Random Access Techniques for Data Transmission over Packet-Switched Radio Channels“, *Proc. Nat. Computer Conf.*, S. 187–201, 1975.
- Kohler, E., H. Handley und S. Floyd: „Designing DCCP: Congestion Control without Reliability“, *Proc. SIGCOMM 2006 Conf.*, ACM, S. 27–38, 2006.
- Koodli, R. und C. E. Perkins: *Mobile Inter-networking with IPv6*. John Wiley & Sons, New York, 2007.
- Koopman, P.: „32-Bit Cyclic Redundancy Codes for Internet Applications“, *Proc. Intl. Conf. on Dependable Systems and Networks*. IEEE, S. 459–472, 2002.
- Krishnamurthy, B. und J. Rexford: *Web Protocols and Practice*. Addison-Wesley, Boston, 2001.
- Kumar, S., C. Paar, J. Pelzl, G. Pfeiffer und M. Schimmler: „Breaking Ciphers with COPACOBANA: A Cost-Optimized Parallel Code Breaker“, *Proc. 8th Cryptographic Hardware and Embedded Systems Wksp.*, IACR, S. 101–118, Okt. 2006.
- Labovitz, C., A. Ahuja, A. Bose und F. Jahanian: „Delayed Internet Routing Convergence“, *IEEE/ACM Trans. on Networking*, Vol. 9, S. 293–306, Juni 2001.
- Lam, C. K. M. und B. C. Y. Tan: „The Internet Is Changing the Music Industry“, *Commun. of the ACM*, Vol. 44, S. 62–66, Aug. 2001.
- Laoutaris, N., G. Smaragdakis, P. Rodriguez und R. Sundaram: „Delay Tolerant Bulk Data Transfers on the Internet“, *Proc. SIGMETRICS 2009 Conf.*, ACM, S. 229–238, Juni 2009.
- Larmo, A., M. Lindstrom, M. Meyer, G. Pelletier, J. Torsner und H. Wiemann: „The LTE Link-Layer Design“, *IEEE Commun. Magazine*, Vol. 47, S. 52–59, April 2009.
- Lee, J. S. und L. E. Miller: *CDMA Systems Engineering Handbook*. Artech House, London, 1998.
- Leland, W., M. Taqqu, W. Willinger und D. WILSON: „On the Self-Similar Nature of Ethernet Traffic“, *IEEE/ACM Trans. on Networking*, Vol. 2, S. 1–15, Feb. 1994.
- Lemon, J.: „Resisting SYN Flood DOS Attacks with a SYN Cache“, *Proc. BSDCon Conf.*, USENIX, S. 88–98, 2002.
- Levy, S.: „Crypto Rebels“, *Wired*, S. 54–61, Mai/Juni 1993.

- Lewis, M.: *Comparing, Designing und Deploying VPNs*. Cisco Press, Indianapolis, 2006.
- Li, M., D. Agrawal, D. Ganesan und A. Venkataramani: „Block-Switched Networks: A New Paradigm for Wireless Transport“, *Proc. NSDI 2009 Conf.*, USENIX, S. 423–436, 2009.
- Lin, S. und D. Costello: *Error Control Coding, 2. Auflage*. Pearson Education, Upper Saddle River, 2004.
- Lubacz, J., W. Mazurczyk und K. Szczypiorski: „Vice over IP“, *IEEE Spectrum*, S. 42–47, Feb. 2010.
- Macedonia, M. R.: „Distributed File Sharing“, *IEEE Computer*, Vol. 33, S. 99–101, 2000.
- Madhavan, J., D. Ko, L. Lot, V. Gangpathy, A. Rasmussen und A. Halevy: „Google’s Deep Web Crawl“, *Proc. VLDB 2008 Conf.*, VLDB Endowment, S. 1241–1252, 2008.
- Mahajan, R., M. Rodrig, D. Wetherall und J. Zahorjan: „Analyzing the MAC-Level Behavior of Wireless Networks in the Wild“, *Proc. SIGCOMM 2006 Conf.*, ACM, S. 75–86, 2006.
- Malis, A. und W. Simpson: „PPP over SONET/SDH“, *RFC 2615*, Juni 1999.
- Massey, J. L.: „Shift-Register Synthesis and BCH Decoding“, *IEEE Trans. on Information Theory*, Vol. IT-5, S. 122–127, Jan. 1969.
- Matsui, M.: „Linear Cryptanalysis Method for DES Cipher“, *Advances in Cryptology – Eurocrypt 1993 Proceedings*, Springer-Verlag LNCS 765, Berlin, S. 386–397, 1994.
- Maufer, T. A.: *IP Fundamentals*, Prentice Hall, Upper Saddle River, 1999.
- Maymounkov, P. und D. Mazières: „Kademlia: A Peer-to-Peer Information System Based on the XOR Metric“, *Proc. First Intl. Wksp. on Peer-to-Peer Systems*, Springer-Verlag LNCS 2429, Berlin, S. 53–65, 2002.
- Mazières, D. und M. F. Kaashoek: „The Design, Implementation und Operation of an Email Pseudonym Server“, *Proc. Fifth Conf. on Computer and Commun. Security*, ACM, S. 27–36, 1998.
- McAfee Labs: *McAfee Threat Reports: First Quarter 2010*. McAfee Inc., 2010.
- Menezes, A. J. und S. A. Vanstone: „Elliptic Curve Cryptosystems and Their Implementation“, *Journal of Cryptology*, Vol. 6, S. 209–224, 1993.
- Merkle, R. C. und M. Hellman: „Hiding and Signatures in Trapdoor Knapsacks“, *IEEE Trans. on Information Theory*, Vol. IT-4, S. 525–530, Sept. 1978.
- Metcalfe, R. M.: „Computer/Network Interface Design: Lessons from Arpanet and Ethernet“, *IEEE J. on Selected Areas in Commun.*, Vol. 11, S. 173–179, Feb. 1993.
- Metcalfe, R. M. und D. R. Boggs: „Ethernet: Distributed Packet Switching for Local Computer Networks“, *Commun. of the ACM*, Vol. 19, S. 395–404, Juli 1976.
- Metz, C: „Interconnecting ISP Networks“, *IEEE Internet Computing*, Vol. 5, S. 74–80, März–April 2001.

- Mishra, P. P., H. Kanakia und S. Tripathi: „On Hop by Hop Rate-Based Congestion Control“, *IEEE/ACM Trans. on Networking*, Vol. 4, S. 224–239, April 1996.
- Mogul, J. C.: „IP Network Performance“, *Internet System Handbook*, D. C. Lynch und M. Y. Rose (Hrg.), Addison-Wesley, Boston , S. 575–575, 1993.
- Mogul, J. und S. Deering: „Path MTU Discovery“, *RFC 1191*, Nov. 1990.
- Mogul, J. und G. Minshall: „Rethinking the Nagle Algorithm“, *Comput. Commun. Rev.*, Vol. 31, S. 6–20, Jan. 2001.
- Moy, J.: „Multicast Routing Extensions for OSPF“, *Commun. of the ACM*, Vol. 37, S. 61–66, Aug. 1994.
- Mullins, J.: „Making Unbreakable Code“, *IEEE Spectrum*, S. 40–45, Mai 2002.
- Nagle, J.: „On Packet Switches with Infinite Storage“, *IEEE Trans. on Commun.*, Vol. COM-5, S. 435–438, April 1987.
- Nagle, J.: „Congestion Control in TCP/IP Internetworks“, *Computer Commun. Rev.*, Vol. 14, S. 11–17, Okt. 1984.
- Naughton, J.: *A Brief History of the Future*. Overlook Press, Woodstock, 2000.
- Needham, R. M. und M. D. Schroeder: „Using Encryption for Authentication in Large Networks of Computers“, *Commun. of the ACM*, Vol. 21, S. 993–999, Dez. 1978.
- Needham, R. M. und M. D. Schroeder: „Authentication Revisited“, *Operating Systems Rev.*, Vol. 21, S. 7, Jan. 1987.
- Nelakuditi, S. und Z.-L. Zhang: „A Localized Adaptive Proportioning Approach to QoS Routing“, *IEEE Commun. Magazine Vol. 40*, S. 66–71, Juni 2002.
- Neuman, C. und T. Ts'o: „Kerberos: An Authentication Service for Computer Networks“, *IEEE Commun. Mag.*, Vol. 32, S. 33–38, Sept. 1994.
- Nichols, R. K. und P. C. Lekkas: *Wireless Security*. McGraw-Hill, New York, 2002.
- NIST: „Secure Hash Algorithm“, *U.S. Government Federal Information Processing Standard 180*, 1993.
- Nonnenmacher, J., E. Biersack und D. Towsley: „Parity-Based Loss Recovery for Reliable Multicast Transmission“, *Proc. SIGCOMM '97 Conf.*, ACM, S. 289–300, 1997.
- Nucci, A. und D. Papagiannaki: *Design, Measurement and Management of Large-Scale IP Networks*. Cambridge University Press, Cambridge, 2008.
- Nugent, R., R. Munakana, A. Chin, R. Coelho und J. Puig-Suari: „The CubeSat: The Pico-Satellite Standard for Research and Education“, *Proc. SPACE 2008 Conf.*, AIAA, 2008.
- Oran, D.: „OSI IS-IS Intra-domain Routing Protocol“, *RFC 1142*, Feb. 1990.
- Otway, D. und O. Rees: „Efficient and Timely Mutual Authentication“, *Operating Systems Rev.*, S. 8–10, Jan. 1987.

- Padhye, J., V. Firoiu, D. Towsley und J. Kurose: „Modeling TCP Throughput: A Simple Model and Its Empirical Validation“, *Proc. SIGCOMM '98 Conf.*, ACM, S. 303–314, 1998.
- Palais, J. C.: *Fiber Optic Communications*, 5. Auflage. Prentice Hall, Englewood Cliffs, 2004.
- Parameswaran, M., A. Susarla und A. B. Whinston: „P2P Networking: An Information-Sharing Alternative“, *IEEE Computer*, Vol. 34, S. 31–38, Juli 2001.
- Parekh, A. und R. Gallagher: „A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case“, *IEEE/ACM Trans. on Networking*, Vol. 2, S. 137–150, April 1994.
- Parekh, A. und R. Gallagher: „A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case“, *IEEE/ACM Trans. on Networking*, Vol. 1, S. 344–357, Juni 1993.
- Partridge, C., J. Hughes und J. Stone: „Performance of Checksums and CRCs over Real Data“, *Proc. SIGCOMM '95 Conf.*, ACM, S. 68–76, 1995.
- Partridge, C., T. Mendez und W. Milliken: „Host Anycasting Service“, *RFC 1546*, Nov. 1993.
- Paxson, V. und S. Floyd: „Wide-Area Traffic: The Failure of Poisson Modeling“, *IEEE/ACM Trans. on Networking*, Vol. 3, S. 226–244, Juni 1995.
- Perkins, C.: „IP Mobility Support for IPv4“, *RFC 3344*, Aug. 2002.
- Perkins, C. E.: *RTP: Audio and Video for the Internet*. Addison-Wesley, Boston, 2003.
- Perkins, C. E. (Hrg.): *Ad Hoc Networking*. Addison-Wesley, Boston, 2001.
- Perkins, C. E.: *Mobile IP Design Principles and Practices*. Prentice Hall, Upper Saddle River, 1998.
- Perkins, C. E. und E. Roye: „The Ad Hoc On-Demand Distance-Vector Protocol“, *Ad Hoc Networking*, C. Perkins (Hrg.), Addison-Wesley, Boston, 2001.
- Perlman, R.: *Interconnections*, 2. Auflage. Addison-Wesley, Boston, 2000.
- Perlman, R.: *Network Layer Protocols with Byzantine Robustness*, Doktorarbeit, M.I.T., 1988.
- Perlman, R.: „An Algorithm for the Distributed Computation of a Spanning Tree in an Extended LAN“, *Proc. SIGCOMM '85 Conf.*, ACM, S. 44–53, 1985.
- Perlman, R. und C. Kaufman: „Key Exchange in IPsec“, *IEEE Internet Computing*, Vol. 4, S. 50–56, Nov.–Dez. 2000.
- Peterson, W. W. und D. T. Brown: „Cyclic Codes for Error Detection“, *Proc. IRE*, Vol. 49, S. 228–235, Jan. 1961.

- Piatek, M., T. Kohno und A. Krishnamurthy: „Challenges and Directions for Monitoring P2P File Sharing Networks – or Why My Printer Received a DMCA Takedown Notice“, *3rd Workshop on Hot Topics in Security*, USENIX, Juli 2008.
- Piatek, M., T. Isdal, T. Anderson, A. Krishnamurthy und V. Venkataramani: „Do Incentives Build Robustness in BitTorrent?“, *Proc. NSDI 2007 Conf.*, USENIX, S. 1–14, 2007.
- Piscitello, D. M. und A. L. Chapin: *Open Systems Networking: TCP/IP and OSI*. Addison-Wesley, Boston, 1993.
- Piva, A., F. Bartolini und M. Barni: „Managing Copyrights in Open Networks“, *IEEE Internet Computing*, Vol. 6, S. 18–26, Mai 2002.
- Postel, J.: „Internet Control Message Protocols“, *RFC 792*, Sept. 1981.
- Rabin, J. und C. McCathieNevile: „Mobile Web Best Practices 1.0“, *W3C Recommendation*, Juli 2008.
- Ramakrishnam, K. K., S. Floyd und D. Black: „The Addition of Explicit Congestion Notification (ECN) to IP“, *RFC 3168*, Sept. 2001.
- Ramakrishnam, K. K. und R. Jain: „A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer“, *Proc. SIGCOMM '88 Conf.*, ACM, S. 303–313, 1988.
- Ramaswami, R., S. Kumar und G. Sasaki: *Optical Networks: A Practical Perspective*, 3. Auflage. Morgan Kaufmann, San Francisco, 2009.
- Ratnasamy, S., P. Francis, M. Handley, R. Karp und S. Shenker: „A Scalable Content-Addressable Network“, *Proc. SIGCOMM 2001 Conf.*, ACM, S. 161–172, 2001.
- Rieback, M., B. Crispo und A. Tanenbaum: „Is Your Cat Infected with a Computer Virus?“, *Proc. IEEE Percom*, S. 169–179, März 2006.
- Rivest, R. L.: „The MD5 Message-Digest Algorithm“, *RFC 1320*, April 1992.
- Rivest, R. L., A. Shamir und L. Adleman: „On a Method for Obtaining Digital Signatures and Public Key Cryptosystems“, *Commun. of the ACM*, Vol. 21, S. 120–126, Feb. 1978.
- Roberts, L. G.: „Extensions of Packet Communication Technology to a Hand Held Personal Terminal“, *Proc. Spring Joint Computer Conf.*, AFIPS, S. 295–298, 1972.
- Roberts, L. G.: „Multiple Computer Networks and Intercomputer Communication“, *Proc. First Symp. on Operating Systems Prin.*, ACM, S. 3.1–3.6, 1967.
- Rose, M. T.: *The Simple Book*, Prentice Hall, Englewood Cliffs, 1994.
- Rose, M. T.: *The Internet Message*. Prentice Hall, Englewood Cliffs, 1993.
- Rowstron, A. und P. Druschel: „Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility“, *Proc. 18th Int'l Conf. on Distributed Systems Platforms*, Springer-Verlag LNCS 2218, London, S. 329–350, 2001.

Ruiz-Sanchez, M. A., E. W. Biersack und W. Dabbous: „Survey and Taxonomy of IP Address Lookup Algorithms“, *IEEE Network Magazine*, Vol. 15, S. 8–23, März–April 2001.

Saltzer, J. H., D. P. Reed und D. D. Clark: „End-to-End Arguments in System Design“, *ACM Trans. on Computer Systems*, Vol. 2, S. 277–288, Nov. 1984.

Sample, A., D. Yeager, P. Powledge, A. Mamishev und J. Smith: „Design of an RFID-Based Battery-Free Programmable Sensing Platform“, *IEEE Trans. on Instrumentation and Measurement*, Vol. 57, S. 2608–2615, Nov. 2008.

Saroui, S., K. Gummade und S. Gribble: „Measuring and Analyzing the Characteristics of Napster & Gnutella Hosts“, *Multim. Syst.*, Vol. 9, S. 170–184, Aug. 2003.

Schaller, R.: „Moore’s Law: Past, Present and Future“, *IEEE Spectrum*, Vol. 34, S. 52–59, Juni 1997.

Schneier, B.: *Secrets and Lies*. John Wiley & Sons, New York, 2004.

Schneier, B.: *E-Mail Security*. John Wiley & Sons, New York, 1995.

Schnorr, C. P.: „Efficient Signature Generation for Smart Cards“, *Journal of Cryptology*, Vol. 4, S. 161–174, 1991.

Scholtz, R. A.: „The Origins of Spread-Spectrum Communications“, *IEEE Trans. on Commun.*, Vol. COM-0, S. 822–854, Mai 1982.

Schwartz, M. und N. Abramson: „The AlohaNet: Surfing for Wireless Data“, *IEEE Commun. Magazine*, Vol. 47, S. 21–25, Dez. 2009.

Seifert, R. und J. Edwards: *The All-New Switch Book*. John Wiley & Sons, New York, 2008.

Senn, J. A.: „The Emergence of M-Commerce“, *IEEE Computer*, Vol. 33, S. 148–150, Dez. 2000.

Serjantov, A.: „Anonymizing Censorship Resistant Systems“, *Proc. First Int’l Workshop on Peer-to-Peer Systems*, Springer-Verlag LNCS 2429, London, S. 111–120, 2002.

Shacham, N. und P. McKenney: „Packet Recovery in High-Speed Networks Using Coding and Buffer Management“, *Proc. INFOCOM Conf.*, IEEE, S. 124–131, Juni 1990.

Shaikh, A., J. Rexford und K. Shin: „Load-Sensitive Routing of Long-Lived IP Flows“, *Proc. SIGCOMM ’99 Conf.*, ACM, S. 215–226, Sept. 1999.

Shalunov, S. und R. Carlson: „Detecting Duplex Mismatch on Ethernet“, *Passive and Active Network Measurement*, Springer-Verlag LNCS 3431, Berlin, S. 3135–3148, 2005.

Shannon, C.: „A Mathematical Theory of Communication“, *Bell System Tech. J.*, Vol. 27, S. 379–423, Juli 1948; und S. 623–656, Okt. 1948.

Shepard, S.: *SONET/SDH Demystified*. McGraw-Hill, New York, 2001.

Shreedhar, M. und G. Varghese: „Efficient Fair Queueing Using Deficit Round Robin“, *Proc. SIGCOMM ’95 Conf.*, ACM, S. 231–243, 1995.

- Simpson, W.: *Video Over IP, 2. Auflage*. Focal Press, Burlington, 2008.
- Simpson, W.: „PPP in HDLC-like Framing“, *RFC 1662*, Juli 1994b.
- Simpson, W.: „The Point-to-Point Protocol (PPP)“, *RFC 1661*, Juli 1994a.
- Siu, K. und R. Jain: „A Brief Overview of ATM: Protocol Layers, LAN Emulation und Traffic“, *ACM Computer Communications Review*, Vol. 25, S. 6–20, April 1995.
- Skoudis, E. und T. Liston: *Counter Hack Reloaded, 2. Auflage*. Prentice Hall, Upper Saddle River, 2006.
- Smith, D. K. und R. C. Alexander: *Fumbling the Future*. William Morrow, New York, 1988.
- Snoeren, A. C. und H. Balakrishnan: „An End-to-End Approach to Host Mobility“, *Int'l Conf. on Mobile Computing and Networking*, ACM, S. 155–166, 2000.
- Sobel, D. L.: „Will Carnivore Devour Online Privacy“, *IEEE Computer*, Vol. 34, S. 87–88, Mai 2001.
- Sotirov, A., M. Stevens, J. Appelbaum, A. Lenstra, D. Molnar, D. Osvik and B. de Weger: „MD5 Considered Harmful Today“, *Proc. 25th Chaos Communication Congress*, Verlag Art d'Ameublement, 2008.
- Southey, R.: *The Doctors*. Longman, Brown, Green and Longmans, London, 1848.
- Spurgeon, C. E.: *Ethernet: The Definitive Guide*. O'Reilly, Sebastopol, 2000.
- Stallings, W.: *Data and Computer Communications, 9. Auflage*. Pearson Education, Upper Saddle River, 2010.
- Starr, T., M. Sorbara, J. Coiffi und P. Silverman: „DSL Advances“, Prentice Hall, Upper Saddle River, 2003.
- Stevens, W. R.: *TCP/IP Illustrated: The Protocols*. Addison Wesley, Boston, 1994.
- Stinson, D. R.: *Cryptography Theory and Practice, 2. Auflage*, CRC Press, Boca Raton, 2002.
- Stoica, I., R. Morris, D. Karger, M. F. Kaashoek und H. Balakrishnan: „Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications“, *Proc. SIGCOMM 2001 Conf.*, ACM, S. 149–160, 2001.
- Stubblefield, A., J. Ioannidis und A. D. Rubin: „Using the Fluhrer, Mantin und Shamir Attack to Break WEP“, *Proc. Network and Distributed Systems Security Symp.*, ISOC, S. 1–11, 2002.
- Stuttard, D. und M. Pinto: *The Web Application Hacker's Handbook*. John Wiley & Sons, New York, 2007.
- Su, S.: *The UMTS Air Interface in RF Engineering*, McGraw-Hill, New York, 2007.
- Sullivan, G. und T. Wiegand: „Tree Algorithms for Packet Broadcast Channels“, *Proc. of the IEEE*, Vol. 93, S. 18–31, Jan. 2005.

- Sunshine, C. A. und Y. K. Dalal: „Connection Management in Transport Protocols“, *Computer Networks*, Vol. 2, S. 454–473, 1978.
- Tan, K., J. Song, Q. Zhang und M. Sridharn: „A Compound TCP Approach for High-Speed and Long Distance Networks“, *Proc. INFOCOM Conf.*, IEEE, S. 1–12, 2006.
- Tanenbaum, A. S.: *Modern Operating Systems*, 3. Auflage. Prentice Hall, Upper Saddle River, 2007.
- Tanenbaum, A. S. und van Steen, M.: *Distributed Systems: Principles and Paradigms*, NJ: Prentice Hall, Upper Saddle River, 2007.
- Tomlinson, R. S.: „Selecting Sequence Numbers“, *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop*, ACM, S. 11–23, 1975.
- Tuchman, W.: „Hellman Presents No Shortcut Solutions to DES“, *IEEE Spectrum*, Vol. 16, S. 40–41, Juli 1979.
- Turner, J. S.: „New Directions in Communications (or Which Way to the Information Age)“, *IEEE Commun. Magazine*, Vol. 24, S. 8–15, Okt. 1986.
- Ungerboeck, G.: „Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction“, *IEEE Commun. Magazine*, Vol. 25, S. 5–11, Feb. 1987.
- Valade, J.: *PHP & MySQL for Dummies*, 5. Auflage. John Wiley & Sons, New York, 2009.
- Varghese, G.: *Network Algorithmics*. Morgan Kaufmann, San Francisco, 2004.
- Varghese, G. und T. Lauck: „Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility“, *Proc. 11th Symp. on Operating Systems Prin.*, ACM, S. 25–38, 1987.
- VERIZON BUSINESS: *2009 Data Breach Investigations Report*. Verizon, 2009.
- Viterbi, A.: *CDMA: Principles of Spread Spectrum Communication*. Prentice Hall, Englewood Cliffs, 1995.
- Von Ahn, L., B. Blum und J. Langford: „Telling Humans and Computers Apart Automatically“, *Commun. of the ACM*, Vol. 47, S. 56–60, Feb. 2004.
- Waitzman, D., C. Partridge und S. Deering: „Distance Vector Multicast Routing Protocol“, *RFC 1075*, Nov. 1988.
- Waldman, M., A. D. Rubin und L. F. Cranor: „Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System“, *Proc. Ninth USENIX Security Symp.*, USENIX, S. 59–72, 2000.
- Wang, Z. und J. Crowcroft: „SEAL Detects Cell Misordering“, *IEEE Network Magazine*, Vol. 6, S. 8–9, Juli 1992.
- Want, R.: *RFID Explained*. Morgan Claypool, San Rafael, 2006.
- Warneke, B., M. Last, B. Liebowitz und K. S. J. Pister: „Smart Dust: Communicating with a Cubic Millimeter Computer“, *IEEE Computer*, Vol. 34, S. 44–51, Jan. 2001.

- Wayner, P.: *Disappearing Cryptography: Information Hiding, Steganography und Watermarking*, 3. Auflage. Morgan Kaufmann, San Francisco, 2008.
- Wei, D., J. Cheng, S. Low und S. Hedge: „FAST TCP: Motivation, Architecture, Algorithms, Performance“, *IEEE/ACM Trans. on Networking*, Vol. 14, S. 1246–1259, Dez. 2006.
- Weiser, M.: „The Computer for the Twenty-First Century“, *Scientific American*, Vol. 265, S. 94–104, Sept. 1991.
- Welbourne, E., L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska und G. Borriello: „Building the Internet of Things Using RFID“, *IEEE Internet Computing*, Vol. 13, S. 48–55, Mai 2009.
- Wittenburg, N.: *Understanding Voice Over IP Technology*. Delmar Cengage Learning, Clifton Park, 2009.
- Wolman, A., G. Voelker, N. Sharma, N. Cardwell, A. Karlin und H. Levy: „On the Scale and Performance of Cooperative Web Proxy Caching“, *Proc. 17th Symp. on Operating Systems Prin.*, ACM, S. 16–31, 1999.
- Wood, L., W. Ivancic, W. Eddy, D. Stewart, J. Northam, C. Jackson und A. da Silva Curiel: „Use of the Delay-Tolerant Networking Bundle Protocol from Space“, *Proc. 59th Int'l Astronautical Congress*, Int'l Astronautical Federation, S. 3123–3133, 2008.
- Wu, T.: „Network Neutrality, Broadband Discrimination“, *Journal on Telecom. and High-Tech. Law*, Vol. 2, S. 141–179, 2003.
- Wylie, J., M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kiliccote und P. K. Khosla: „Survivable Information Storage Systems“, *IEEE Computer*, Vol. 33, S. 61–68, Aug. 2000.
- Yu, T., S. Hartman und K. Raeburn: „The Perils of Unauthenticated Encryption: Kerberos Version 4“, *Proc. NDSS Symposium*, Internet Society, Feb. 2004.
- Yuval, G.: „How to Swindle Rabin“, *Cryptologia*, Vol. 3, S. 187–190, Juli 1979.
- Zacks, M.: „Antiterrorist Legislation Expands Electronic Snooping“, *IEEE Internet Computing*, Vol. 5, S. 8–9, Nov.–Dez. 2001.
- Zhang, Y., L. Breslau, V. Paxson und S. Shenker: „On the Characteristics and Origins of Internet Flow Rates“, *Proc. SIGCOMM 2002 Conf.*, ACM, S. 309–322, 2002.
- Zhao, B., H. Ling, J. Stribling, S. Rhea, A. Joseph und J. Kubiatowicz: „Tapestry: A Resilient Global-Scale Overlay for Service Deployment“, *IEEE J. on Selected Areas in Commun.*, Vol. 22, S. 41–53, Jan. 2004.
- Zimmermann, P. R.: *The Official PGP User's Guide*. M.I.T. Press, Cambridge, 1995a.
- Zimmermann, P. R.: *PGP: Source Code and Internals*, M.I.T. Press, Cambridge, 1995b.
- Zipf, G. K.: *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, Boston, 1949.
- Ziv, J. und Z. Lempel: „A Universal Algorithm for Sequential Data Compression“, *IEEE Trans. on Information Theory*, Vol. IT–3, S. 337–343, Mai 1977.

Register

Numerics

100Base-FX 341
100Base-T4 341
100Base-TX 341
10-Gigabit-Ethernet 346
 Kabelarten 347
1-Bit-Schiebefensterprotokoll 274
1G-Mobiltelefon 202, 204
1G-System 91
1-persistentes CSMA 314
2,5G 217
2G-Mobiltelefon 202, 208
2G-System 91
3G-Mobiltelefon 202, 212
3GPP 104
3G-System 91
4B/5B 162
64B/66B-Codierung 346
8B/10B 164
8B/10B-Codierung 345

A

AAC 799
AAL5 297
About-Protokoll 740
Access Grant Channel 212
ACL 378
ActiveX 769
ActiveX-Steuerelement 966
Adaptiver Algorithmus 421
Adaptives Frequenzsprungverfahren 377
Adaptives Tree-Walk-Protokoll 323
Add-on 968
Ad-hoc-Netz
 Routing 448
Ad-hoc-Netzwerk 97, 350
Adressierung 58, 581
ADSL 126, 183, 295
Advanced Mobile Phone System
 siehe AMPS
Advanced Networks and Services
 siehe ANS
AES 363
AES (Advanced Encryption Standard) 886
AIFS 359
AIMD 611
AJAX 770
Aktives RFID 101

A-law 190
A-law-Codierung 795
Algorithmus
 adaptiver 421
 mit öffentlichen Schlüsseln 897
 mit symmetrischen Schlüsseln 881
 nicht adaptiver 421
Algorithmus von Dijkstra 424
ALOHA 99, 309
 reines 309
Alternate Mark Inversion siehe AMI
AMI 164
Amplitudenmodulation 165
AMPS 91, 204
Amtsleitung siehe Teilnehmeranschluss-
 leitung
Anfrage-Antwort-Dienst 62
Anonymer Remailer 970
ANS 86
ANSNET 86
Antenne
 sektorisierte 216
Anwendungs-Gateway 926
Anwendungsschicht 70, 73, 694
Anycast-Routing 444
AP siehe Zugangspunkt
Apache 767
Apokalypse der zwei Elefanten 77
Applet 769
APSD 358
ARP 533
ARPA 81
ARPANET 71, 80
ARQ 596
ARQ-Protokoll 269
AS siehe autonomes System
AS-Grenzrouter 543
ASK siehe Amplitudenmodulation
ASP.NET 767
AS-Pfad 548
Association 362
Assured Forwarding 486
Asymmetric Digital Subscriber Line
 siehe ADSL
Asynchrone E/A 774
ATM 296
 ATM Adaption Layer 5 siehe AAL5
Audio, digitales 794
Audiotranskription 796
Auslöschungskanal 245

- Authenticode 967
Authentifizierung 60
 mit öffentlicher Verschlüsselung 948
 Schlüsselverteilungszentrum 942
Authentifizierungsprotokoll 934
 KDC 943
 Kerberos 945
Autokorrelation 215
Autonegotiation 342
Autonomes System 495, 540
 Backbone-Bereich 543
 Backbone-Router 543
Autoresponder 714
Autorisation 935
Autoritativer Datensatz 704
AVC 805
- B**
- Backbone 89
Backbone-Bereich 543
Backbone-Router 543
Backward-Learning-Algorithmus 389
Bandbreite 122
 analoge 125
 digitale 125
Bandbreiteneffizienz 160
Bandbreitenzuordnung 605
 Effizienz 605
 Konvergenz 608
 Max-Min-Fairness 606
Bandbreite-Verzögerung-Produkt 278, 314, 676
Baran, Paul 81
Barker-Folge 352
Base64-Codierung 720
Basis
 diagonale 876
 geradlinige 876
Basisbandsignal 124, 165
Basisbandübertragung 159
Basisoperation 63
Basisstation 42, 97
Baudrate 161
BB84 875
BCCH 211
Beacon-Rahmen 358
Bellman-Ford-Algorithmus 428
Bent Pipe 149
Benutzeragent (E-Mail) 709
Benutzerprofil 37
Bereichswahl 222
Berkeley-Socket 573
- Berners-Lee, Tim 734
Bestätigter Datagrammdienst 62
Bestätigung
 Dup-Ack 655
 kumulative 283, 634, 645
 selektive 636, 658
 verzögerte 643
Bestätigungsrahmen 68
BGP 495, 546
 Peering 547
Binärer Countdown 320
Binäres exponentielles Backoff 334
Bipolare Codierung 164
Bitmusterprotokoll 318
Bitrate 161
Bitstopfen 241
BitTorrent 849
Bitübertragungsschicht 68
Blockchiffre 881
Blockcode 246
Bluetooth 41
 Anwendungen 374
 Architektur 373
 Funkschicht 376
 Protokollstapel 375
 Rahmenstruktur 378
 Sicherheit 933
 Verbindungsschicht 377
Bookmark 741
Botnet 713
BPSK 166
Breitbanddienst 183
Breitbandzugang 89
Bridge 386, 395
Broadcast Control Channel siehe BCCH
Broadcasting 40
Broadcast-Routing 439
Broadcast-Sturm 399, 662
Broadcast-Verbindung 39
Browser 736
Browsererweiterung 968
BSC 209
Bucket-Brigade-Angriff 942
Bündel-Protokoll 683
Bytestopfen 240
- C**
- Cache
 vergifteter 957
Caching 783
Call-by-Reference 620
CAPTCHA 38

- Care-of-Adresse 446, 553
 Carrier Extension 344
 Carrier Sense Multiple Access siehe CSMA
 Carrier-grade-Ethernet 349
 Cäsar-Chiffre 871
 CCCH 212
 CCITT 105
 CCK 352
 CCMP 932
 CDMA 93, 140, 170, 208
 CDMA2000 213
 CDN siehe Content-Delivery-Netz
 CERN 734
 CGI 765
 Challenge-Response-Protokoll 936
 Chiffre 868
 Chiffertext 869
 Chiffriermodus 890
 Chipfolge 170
 Choked-Zustand 851
 Choke-Paket 459
 Chord 852
 Chromatische Dispersion 135
 Chrome 736
 CIDR 510
 Cipher Block Chaining 891
 Cipher-Feedback-Modus 892
 Clark, David 109
 Clark, Wesley 82
 Classless InterDomain Routing siehe CIDR
 Client 25
 Client-Server-Modell 25
 Clipper-Chip 970
 Cloud Computing 763
 CMTS 221
 Code 868
 - 4B/5B- 162
 - 8B/10B- 164
 - Gray- 167
 - Hamming- 248
 - LDPC- 252
 - linearer 246
 - Manchester- 161
 - mobiler 965
 - Reed-Solomon- 251
 - systematischer 246
 - Walsh- 171
 Codec 189
 Codemultiplexverfahren 93
 Codemultiplexverfahren siehe CDMA
 Coderate 246
 Code-Signierung 966
- Codierung
 - Algorithmus 796
 - bipolare 164
 - Psychoakustik 797
 - wahrnehmungsbezogen 797
 - Wellenform 797
 Comité Consultatif International
 Télégraphique et Téléphonique
 siehe CCITT
 Common Control Channel siehe CCCH
 Complementary Code Keying siehe CCK
 Content Delivery 832
 Content-Delivery-Netz 841
 Cookie 37, 748
 - Spyware 752
 - von Drittanbietern 752
 Copyright 976
 Core-Based Tree 443
 Counter-Modus 895
 Count-to-Infinity-Problem 430
 CRC 255
 CSMA 99, 314
 - 1-persistentes 314
 - mit Kollisionserkennung 316
 - nicht persistentes 315
 - p-persistentes 315
 CSMA/CA 354
 CSMA/CD 316
 - mit binärem exponentiellen Backoff 334
 CSNET 85
 CSS 762
 CTS-Rahmen 327
 CubeSat 157
 Cut-through-Schaltung 61
 Cut-through-Switching 390
 Cybersquatting 698
 Cypherpunk-Remailer 971
- D**
- DAG 423
 D-AMPS 208
 Darstellungsschicht 70
 Datagramm 415
 Datagrammdienst 61
 - bestätigter 62
 Datagrammnetz 415
 Datenfluss 466
 Datenrahmen 68
 Datenschutz 969
 Datenübertragung
 - theoretische Grundlagen 121
 Datenübertragungsrate, maximale 125

- Datenverkehrsanalyse 921
Datenzustellung 363
Davies, Donald 82
DCCH 211
DCF 355
DCF InterFrame Spacing siehe DIFS
Decodieralgorithmus 796
Dedicated Control Channel siehe DCCH
Deep Web 791
De-facto-Standard 103
Default-freie Zone 510
Default-Gateway 535
Deficit-Round-Robin 476
De-jure-Standard 104
Denial of Service 927
DES (Data Encryption Standard) 883
Designerter Router 433
Desktop-Sharing 26
Dezimalpunktschreibweise 507
DHCP 536
DHT 851
Dialogsteuerung 70
Dienst
 Basisoperation 63
 differenzierter 606
 verbindungsloser 60
 verbindungsorientierter 60
Dienstgüte 59, 466
 klassenbasierte 484
Dienstgütvereinbarung 468
Differenzierter Dienst 484, 823
Diffie-Hellman-Schlüsselaustausch 941
DIFS 359
Digital Millennium Copyright Act
 siehe DMCA
Digital Subscriber Line Access Multiplexer
 siehe DSLAM
Digital Subscriber Line siehe DSL
Digitale Modulation 159
Digitale Signatur 901
Digitales Audio 794
Diskrete Kosinustransformation 803
Disparität 164
Dispersion
 chromatische 135
Dissociation 362
Distanzvektoralgorithmus 428
Distributed Denial of Service 927
Distribution 363
DIX-Standard 330
DMCA 977
DMCA Takedown Notice 37, 978
DMT siehe Mehrtonverfahren
DNS 73, 85, 618, 695
 Datensätze 959
 Namensaflösung 704
 Namensraum 696
 Nameserver 703
 Registrierungsstelle 697
 Resolver 695
 sicheres 958
 Umleitung 843
 Zone 703
DNSsec 958
DNS-Spoofing 956
DOCSIS 221
DOM 771
Domain Name Service siehe DNS
Domain Name System siehe DNS
Domäne 696
 Ressourcendatensatz 699
Domänennamensdienst siehe DNS
Dotcom-Ära 735
Downstream-Proxy 840
Downward-Multiplexing 602
Drahtloses LAN 96
Dreiwege-Handshake 589, 829
Drosseln des Verkehrs 458
DSL 88, 183
DSLAM 88, 186
DSL-Modem 88
DSSS siehe Spreizbandtechnik
Dup-Ack 655
Durchgangsvermittlung 176
Durchlassbandsignal 124
Durchlassbereich 159, 164
DVMRP 443
DWDM 197
Dynamic-Frequency-Selection 364
Dynamischer Routing-Algorithmus 421
Dynamisches HTML 767

E

- EAP 931
Early-Exit-Routing 551
eBGP siehe externes BGP
Echtzeitkonferenz 821
Echtzeittransportprotokoll 622
ECMP 542
ECN 460, 610, 634, 659
E-Commerce 27
 Modelle 30
EDGE 217
EIFS 360
Einflusslänge 249

- Ein-Wege-Hash-Funktion siehe Message Digest
- Electronic Commerce siehe E-Commerce
- Electronic-Code-Book-Modus 890
- Elektronischer Produktcode siehe EPC
- E-Mail 26, 708
- Base64-Codierung 720
 - beantworten 714
 - Benutzeragent 709
 - Body 711
 - erstellen 714
 - ESMTP 727
 - Header 711
 - IMAP 731
 - Mail Submission 709, 725, 728
 - Mailingliste 710, 715
 - MIME-Nachrichten 718
 - Nachrichtenformat 716
 - Nachrichtenübertragung 725
 - Nachrichtenübertragungsagent 709
 - POP3 733
 - Quoted-Printable-Codierung 721
 - RFC5322 716
 - Sicherheit 949
 - SMTP 710, 725
 - Umschlag und Inhalt 710
 - versenden 715
 - Webmail 733
- Emoticon 708
- Empfangsfenster 272
- Ende-zu-Ende-Argument 596
- Ende-zu-Ende-Argumentation 414
- Entmilitarisierte Zone 925
- Envelope 710
- EPC 380
- EPC Gen 2 380
- Architektur 381
 - Bitübertragungsschicht 382
 - Tag-Identifizierungsschicht 383
- EPON 188
- ESMTP 727
- Eternity Service 973
- Ethernet 43
- Bitübertragungsschicht 329
 - DIX-Standard 330
 - Leistung 335
 - MAC-Teilschichtprotokoll 331
- EuroDOCSIS 222
- Expedited Forwarding 485
- Explicit Congestion Notification siehe ECN
- Exponentiell gewichteter gleitender Durchschnitt 459
- Exposed-Terminal-Problem 327
- Externes BGP 550
- Externes Gateway-Protokoll 495, 540
- ## F
- Fair Queueing 474
- Faltungscode 249
- Fast Ethernet 339
- FCFS-Algorithmus 474
- FDD 206, 368
- FDDI 320
- FDM siehe Frequenzmultiplexverfahren
- FEC 244, 539
- Fehlerbehebung 58
- Fehlererkennung 58
- Fehlererkennungscode 244, 252
- Fehlerkorrekturcode 244, 246
- Blockcode 246
 - Fehlersyndrom 249
 - Fehlerüberwachung 242, 595
 - Fernleitung 176
 - Fernnetz 47
 - Fernnetzbetreiber 178
 - Fernvermittlungsstelle 176
 - FHSS 139
 - Fiber to the Home 132
 - Fiber to the Home siehe FttH
 - FIFO-Algorithmus 474
 - File Transfer Protocol siehe FTP
 - Firefox 736
 - Firewall 866, 924
 - Anwendungs-Gateway 926
 - DDoS 927
 - DoS 927
 - entmilitarisierte Zone 925
 - Paketfilter 925
 - zustandsorientierte 926 - Flagbyte 239
 - Flash Crowd 846
 - Fletcher-Prüfsumme 255
 - Flusskontrolle 59, 243, 595
 - basiert auf der Übertragungsrate 243
 - feedbackbasierte 243 - Flussspezifikation 478
 - Fluten 427
 - Forwarding-Algorithmus 51
 - Fourieranalyse 121
 - Fourierreihe 121
 - Frame-Bursting 344
 - Frequency Division Duplex siehe FDD
 - Frequency Hopping Spread Spectrum siehe FHSS
 - Frequenz 138

- Frequenzbereich
ISM 145
politische Regelungen 144
U-NII-Band 146
White Space 146
Zuweisung 220
- Frequenzduplexverfahren 368
- Frequenzmaskierung 798
- Frequenzmodulation 166
- Frequenzmultiplexverfahren 167
- Frequenzspezifikation 170
- Frequenzsprungtechnik mit
Spektrumsspezifikation siehe FHSS
- Frequenzsprungverfahren 139, 376, 933
adaptives 377
- Frequenzwiederholung 92
- Front-End 837
- Frequenzmultiplexverfahren
orthogonales 168
- FSK siehe Frequenzmodulation
- FTP 73, 520, 740
- FttH 89, 187
- Funkübertragung 141
- Funkzugangsnetz 93
- Fuzzball 85
- G**
- G.dmt 185
- G.lite 187
- Gallagher, Robert 252
- Gatekeeper 825
- Gateway 52, 397, 825
- Geburtstagsattacke 909
- Geheimhaltungsverstärkung 878
- Gemeinschaftsantennen-Fernsehen 217
- Gemischmodus 339
- Generatorpolynom 256
- GEO-Satellit 151
- Geostationärer Satellit 150
- Geotagging 34
- Gerichteter azyklischer Graph siehe DAG
- Gesetz von Moore 132
- GGSN 94
- Gigabit-Ethernet 342
Kabelarten 344
- Gigabit-Netz, Protokoll 674
- Glasfaserkabel 135
- Glasfaserknoten 218
- Glasfaserleiter 132
- Global Positioning System siehe GPS
- Global System for Mobile communications
siehe GSM
- Globalstar 156
- Gmail 37
- Go-back-N-Protokoll 277
- GPON 188
- GPRS 94
- GPS 34
- GPS-Satellit 154
- Gratuitous ARP 535, 555
- Gray-Code 167
- Grenzrouter 543
- GSM 91, 208
- Gutenberg-Projekt 976
- H**
- H.245 825
- H.323 824
- H.323 und SIP
Vergleich 830
- Halbduplex 129
- Hamming-Abstand 247
- Hamming-Code 248
- Handover 95, 206
Hard 95, 216
Soft 95, 216
- Hard Handover 95, 216
- Hard-Decision-Decodierung 250
- Harmonische 121
- Hash-Tabelle, verteilte 851
- HDLC 292
- HDLC-Protokoll 241
- Header 56, 262
Komprimierung 672
Prediction 671
robuste Komprimierung 674
- Heimatagent 446, 553
- Heimatstandort 445
- HFC-System 218
- HF-RFID 101
- Hidden-Terminal-Problem 327
- Hierarchisches Routing 437
- Hilfsprogramm 743
- HLR 209
- Home Subscriber Server siehe HSS
- Hop-by-Hop-Rückstau 461
- Host 47
- Hot-Potato-Routing 551
- Hotspot 32
- HSS 95
- HTML 741, 753
dynamisches 767
- Eingabefeld 759
- Formular 758

- HTML 1.0 756
 HTML 2.0 756
 HTML 3.0 756
 HTML 4.0 756, 757
 Tags 753
HTTP 70, 737, 740, 775
 Beispiel 786
 Caching 783
 Methode 778
 Nachrichten-Header 780
 Verbindungen 776
Hub 153, 337, 395
Huckepacktransport 272
Hyperlink 736
Hypertext 735
HyperText Transfer Protocol siehe HTTP
- I**
- IAB** 109
ICANN 508, 696
ICMP 72, 531
IEEE 107
IEEE 802.11 43, 96
 Architektur 349
 Bitübertragungsschicht 351
 Dienste 362
 MAC-Teilschichtprotokoll 353
 Protokollstapel 349
 Rahmenstruktur 360
IEEE 802.11a 352
IEEE 802.11b 352
IEEE 802.16 364
 Architektur 366
 Bitübertragungsschicht 367
 MAC-Teilschichtprotokoll 369
 Protokollstapel 366
 Rahmenstruktur 371
IEEE 802.16 siehe WiMAX
IETF 109
IGMP 552
IKE 921
IMAP 731
IMP 82
IMT-2000 213
IMTS 204
inetd 629
Infrarotübertragung 147
Ingress-Filter 555
Instant Messaging 29
Institute of Electrical and Electronics Engineers siehe IEEE
Integration 363
Integrierter Dienst 481
Integrität 60
Interdomain-Protokoll 495
Interdomain-Routing 540
Interface Message Processor siehe IMP
Interlacing 800
Interleaving 253
International Telecommunications Union siehe ITU
Internationaler Standard 107
Internes BGP 550
Internes Gateway-Protokoll 495, 540
Internet 23, 51, 80
 über Kabel 218
Internet Activities Board siehe IAB
Internet Architecture Board siehe IAB
Internet Control Message Protocol siehe ICMP
Internet Engineering Task Force siehe IETF
Internet Explorer 736
Internet Protocol siehe IP
Internet Research Task Force siehe IRTF
Internet Service Provider siehe Internetdienstanbieter
Internet Society 109
Internetarchitektur 87
Internetdämon 629
Internetdienstanbieter 37, 50, 88
Internetknoten 89, 547
Internetradio 817
Internetschicht 71
Internetstandard 110
Internettelefonie 26, 793, 821
Internettransportprotokoll
 TCP 628
Internetverbindung 88
Internetwork 49, 51, 487
 Multicasting 551
 Routing 494
Internetworking 59
Internetzugang 88
Interplanetarisches Internet 40
Intradomain-Protokoll 495
Intradomain-Routing 540
IP 72, 502
IP TeleVision 31
IP-Adresse 507
 Dezimalpunktschreibweise 507
 Leasing 536
 Präfix 507
IPsec 920
 Sicherheitsassoziation 920
 Transportmodus 921
 Tunnelmodus 921
IP-Telefonie 26

- IPTV 817
IPTV siehe IP TeleVision
IPv4 503
IPv6 520
 Erweiterungs-Header 526
 Hauptheader 523
 Kontroversen 529
IrDA-Standard 147
Iridium 155
IRTF 109
IS 107
IS-95 208
ISAKMP 921
IS-IS 541
IS-IS-Protokoll 436
ISM-Band 97, 145
ISO 106
ISO/OSI-Referenzmodell 66
ISP siehe Internetdienstanbieter
ISP-Netz 50
Iterative Anfrage 706
ITU 105
ITU-D 105
ITU-R 105
ITU-T 105
IXP siehe Internetknoten
- J**
- Java-Applet 965
JavaScript 767
 Sicherheit 967
Jitter 467, 625
JPEG 801
JSP 767
Jumbogramm 528
Jumborahmen 346
JVM 769
JVM (Java Virtual Machine) 965
- K**
- Kabel
 Glasefaser 135
 Kategorie 3 129
 Kategorie 5 128
 Kategorie 6 129
 Kategorie 7 129
 Koaxial- 130
 Twisted-Pair 128
 UTP 129
Kabelkopfstelle 47
Kabelmodem 221
Kabelmodemabschlussystem 89
- Kammeffekt 800
Kanalgebundene Zeichengabe 191
Kanalzuordnung
 statische 305
Kapazität 126
Karn-Algorithmus 648
Kategorie-3-Kabel 129
Kategorie-5-Kabel 128
Kategorie-6-Kabel 129
Kategorie-7-Kabel 129
KDC-Authentifizierungsprotokoll 943
Keepalive-Timer 648
Kerberos 945
Kerckhoffs, Auguste 870
Kernnetz 93
Key Escrow 970
Key siehe Schlüssel
Klartext 869
Klassenbasierte IP-Adressierung 513
Kleinrock, Len 83
Klickbetrug 792
Knotenidentifizierer 853
Koaxialkabel 130
Kommunikationssicherheit 919
Kommunikationssubnetz 47
Komander 190
Konkurrenzsystem 309
Konstellationsdiagramm 166
Konvergenz 430, 608
Kopplung, kapazitive 163
Kreuzkorrelation 214
Kryptoanalyse 869, 896
 differenzielle 896
 lineare 896
Kryptografie 868
 Aktualität 881
 Arbeitsaufwand 870
 Chiffretext 869
 Chiffriermodus 890
 Einführung 868
 Kerckhoffs? Prinzip 870
 Klartext 869
 mit öffentlichem Schlüssel 898
 One-Time Pad 874
 Privacy Amplification 878
 Quantenkryptografie 875
 RSA 898
 Schlüssel 869
 symmetrische Verschlüsselung 881
Kryptologie 869
Kumulative Bestätigung 645
Kürzester Pfad 423

L

L2CAP 376
 Label Switching 418
 LAN 42
 drahtloses 96
 Lastabwurf 455, 463
 Lastausgleich 838
 Lauflängencodierung 804
 LCP 292
 LDPC-Code 252
 Leaky Bucket 457
 Leaky-Bucket-Algorithmus 469
 Leasing 536
 Leecher 851
 Leitung
 T1- 162
 Leitungscode 160
 Leitungsvermittlung 198
 LEO-Satellit 154
 LER 538
 LF-RFID 101
 Lichtgeschwindigkeit 138
 Lichtübertragung 147
 Link 377, 735
 Link-State-Routing 431
 LLC 350, 361
 Local Area Network siehe LAN
 Location Privacy 37
 Logical Link Control siehe LLC
 Lokales Netz 42
 Long Fat Network 674
 Long Term Evolution siehe LTE
 Longest-Prefix-Match 513
 LSR 538
 LTE 96, 217, 366
 Luftschnittstelle 93, 209

M

MAC 68, 304
 MACA 327
 MAHO 212
 Mail Submission 709, 728
 Mailbox 710
 Mailprogramm 711
 Mailserver 709
 Mailto-Protokoll 740
 MAN 46
 Manchester-Code 161
 MANET siehe Ad-hoc-Netz
 Man-in-the-Middle-Angriff 942, 956
 Marshaling 619

Max-Min-Fairness 606
 M-Commerce 34
 MD siehe Message Digest
 MD5 908
 Medium
 magnetisches 127
 Medium Access Control siehe MAC
 Mehrfachzugriffskanal 304
 Mehrfachzugriffsnetz 542
 Mehrtonverfahren 185
 Mehrwegeempfang 143
 Mehrwegeföhrung 98
 Mehrwegempfang 98
 MEO-Satellit 154
 Message Digest (MD) 905
 Messung
 Netzleistung 662
 Metropolitan Area Network siehe MAN
 MIC 932
 Middlebox 838
 Middleware 23
 Mikrowellenübertragung 143
 Mikrozelle 206
 MIME 718, 719
 MIMO 353
 Minizeitscheibe 222
 Mirroring 842
 μ -law 190
 μ -law-Codierung 795
 Mobile Commerce siehe M-Commerce
 Mobile Vermittlungsstelle 94
 Mobil Code 965
 Mobiles IP 553
 Mobiles Web 787
 Mobilfunkvermittlungsstelle 206
 Mobiltelefon 202
 1G 202, 204
 2G 202, 208
 3G 202, 212
 Mockapetris, Paul 78
 Modem 89
 Modulation
 digitale 159
 Moore, Gesetz von 132
 MOSPF 442
 MP3 799
 MPEG 804
 MPEG-1 805
 MPEG-2 805
 MPLS 418, 537
 MSC siehe Mobilfunkvermittlungsstelle
 MSS 636

- MTSO siehe Mobilfunkvermittlungsstelle
MTU 632
Multiaccess Channel siehe Mehrfachzugriffskanal
Multicasting 40, 331, 623
 Internetwork 551
Multicast-Routing 441
Multidestination-Routing 439
Multihoming 548
Multi-Hop-Netz 102
Multimedia 792
Multimode-Faser 133
Multiple Access with Collision Avoidance
 siehe MACA
Multiple Input Multiple Output siehe MIMO
Multiplexing 159, 601
 Downward- 602
 inverses Multiplexing 602
 SCTP 602
 statistisches 59
 Upward- 601
Multiprotokoll-Router 492
Multithreading 745
- N**
- Nachrichtenformat 716
Nachrichtenübertragung 725
Nachrichtenübertragungsagent (E-Mail) 709
Nagle-Algorithmus 643
Namensauflösung 704
Namensgebung 58
Namensraum 696
Nameserver 701, 703
 iterative Anfrage 706
 rekursive Anfrage 706
NAP 86
Napster 976
NAT 516
NAT-Box 517
NAT-Durchdringung 519
National Institute of Standards and
 Technology siehe NIST
NAV 356
N_Bell, Alexander Graham 126, 174
N_Clarke, Arthur C. 150
NCP 292
Near Field Communication siehe NFC
Network Access Point siehe NAP
Network Address Translation siehe NAT
Netz
 der 3. Generation 91
 Klassifizierung 41
lokales 42
Overlay 928
privates 927
Protokoll 53
Schicht 53
skalierbares 59
soziales 30
verzögerungstolerantes 679
Netzarchitektur 54
Netzbelegungsvektor 356
Netzbeschleuniger 258
Netzdienstanbieter 50
Netzentwurf, zellulärer 92
Netzmaske 507
Netzneutralität 36
Netzüberlastung 59
Netzwerkkarte 244, 258
Netzzugangspunkt 86
Netzzugangsschicht 71
NFC 34
N_Fourier, Jean-Baptiste 121
N_Gray, Elisha 174
N_Hertz, Heinrich 138
Nicht adaptiver Algorithmus 421
Nichtabstrebbarkeit 902
Nicht-persistentes CSMA 315
NID 186
NIST 107
N_Lamarr, Hedy 140
N_Maxwell, James Clerk 138
N_Nyquist, Henry 125
Node B
Nonce 931
Non-Return-to-Zero Inverted siehe NRZI
Non-Return-to-Zero siehe NRZ
N_Revere, Paul 147
NRZ 159
NRZI 162
NSAP 581
NSFNET 85
N_Shannon, Claude 125
- O**
- Obergrenze 815
OFDM 98, 168, 352
OFDMA 368
Öffentliches Telefonnetz 174
Olsen, Ken 27
One-Time Pad 874
Onion-Routing 972
Onlineverschlüsselung 866
Optimalitätsprinzip 422
Optischer Richtfunk 147

- Orthogonales Frequenzmultiplex-verfahren 98
 Orthogonales Frquenzmultiplexing 168
 Ortsnetzbereich 178
 Ortsnetzbetreiber 178
 OSI-Referenzmodell 66
 OSPF 540
 - Bereich 543
 - interner Router 543
 OSPF-Protokoll 436, 540
 OUI 331
 Overlay 494
 Overlay-Netz 928
 Overprovisioning 465
- P**
- P2P
 - Choked-Zustand 851
 - Chord 852
 - Knotenidentifizierer 853
 - Leecher 851
 - Netz 846
 - Schwarm 849
 - Seeder 850
 - Tracker 849
 - Unchoked-Zustand 851
 Packet over SONET 291
 Paging Channel 212
 Pairing 372, 377
 Paket 39, 60
 Paketfragmentierung 496
 Paket-Scheduling-Algorithmus 473
 Paketvermittlung 200
 - Store-and-forward 413
 PAN 41
 Paritätsbit 252
 Paritätspaket 811
 PAR-Protokoll 269
 Passives optisches Netz siehe PON
 Passives RFID 101
 Path MTU 496
 Path MTU Discovery 499
 PAWS 590, 636, 675
 P-Box 881
 PCF 355
 PCM 190
 PCS 208
 Peering 89, 547
 Peer-to-Peer-Kommunikation 28
 Peer-to-Peer-Netz
 - Urheberrechte 977
 Per-Hop Behavior 484
- Persistence-Timer 648
 Personal Area Network siehe PAN
 Pfad-Vektor-Protokoll 548
 Pfadverlust 142
 PGP (Pretty Good Privacy) 949
 Phasenmodulation 166
 Phishing 38
 PHP 765
 Piconetz 373
 PIM 444, 552
 Pipelining 278
 PKI (Public Key Infrastructure) 915
 Plug-in 742, 968
 Podcast 817
 Point of Presence siehe POP
 Point-to-Point-Protokoll siehe PPP
 Poisson-Modell 307
 Polynomcode 255
 PON 188
 POP 89, 178
 POP3 733
 Port
 - reservierter 629
 Portmapper 583
 Post, Telegraph & Telephone siehe PTT
 Potenzgesetz 835
 p-persistentes CSMA 315
 PPP 240, 291
 PPP over ATM siehe PPPoA
 PPPoA 297
 Präfix 507
 Premaster-Schlüssel 963
 Primitive
 - Transportschicht 570
 Privacy Amplification 878
 Privates Netz 927
 Problem der drei Bären 514
 Produktchiffre 882
 Protokoll
 - 1-Bit-Schiebefenster- 274
 - ARQ- 269
 - Go-back-N- 277
 - HDLC- 241
 - LCP 292
 - mit eingeschränkter Konkurrenz 322
 - mit selektiver Wiederholung 284
 - NCP 292
 - PAR- 269
 - PPP 240
 - Präambel 242
 - Schiebefenster- 272
 - SLIP 292
 - Stop-and-Wait- 266

- Protokoll mit Trägerprüfung 314
Protokollhierarchie 53
Protokollsicht 58
Protokollstack 55
Provisioning 454
Proxy ARP 535, 555
Proxy-Caching 786
Prozedurauftrag, entfernter 618
Prüfsumme 254, 617
PSK siehe Phasenmodulation
PTT 104
Pulscodemodulation 190
Punkt-zu-Punkt-Verbindung 39
Push-to-Talk-System 204
- Q**
- Q.931 825
QAM siehe Quadraturamplitudenummodulation
QAM-16 166
QAM-64 166
QoS-Routing 477
QoS-Traffic-Scheduling 363
QPSK 166
Quadraturamplitudenummodulation 166
Quality of Service siehe Dienstgüte
Quantenkryptografie 875
Quantisierung 803
Quantisierungsräuschen 795
Qubit 877
Quelle-Senke-Baum 422
Quellport 518
Quoted-Printable-Codierung 721
- R**
- R 101
Radio Frequency IDentification siehe RFID
Radio Network Controller siehe RNC
Rahmen 235
 Bildung 238
Rahmenbildung 238
Rahmen-Header 262
Random Access Channel 212
Random Access Channel siehe
 Mehrfachzugriffskanal
Random Early Detection siehe RED
Ranging 369
Ratenanomalität 360
Ratenanpassung 352
- Rauschabstand siehe Signal-Rauschverhältnis
Real-time Transport Protocol siehe RTP
Reassociation 362
Rechnerallgegenwart siehe Ubiquitous Computing
Rechnernetz
 Definition 22
 Einsatz 24
RED 464
Redefreiheit 972
Redundanzprüfung
 zyklische 255
Reed-Solomon-Code 251
Reflexionsattacke 937
Registrierungsstelle 697
Reines ALOHA 309
Rekursive Anfrage 706
Remailer, anonymer 970
Remote Procedure Call siehe RPC
Repeater 330, 395
Replay-Attacke 943
Request For Comments siehe RFC
Resolver 695
Ressource
 gemeinsame Nutzung 24
Ressourcendatensatz 699
Retransmission-Timer 646
Reverse Lookup 701
Reverse Path Forwarding 439
RFC 109
RFID 32, 100, 380
 aktives 101
 HF 101
 LF 101
 passives 101
 UHF 101
Richtfunk
 optischer 147
Rijndael 888
Rivest, Ronald 908
RNC 93
Robbed-Bit Signaling 191
Roberts, Larry 82
Root-Nameserver 705
Round-Robin-Fair-Queueing 474
Routen-Aggregation 511
Router 48, 397
 angrenzender 544
 designierter 433, 544

- Routing 58
 Ad-hoc-Netz 448
 Anycasting 444
 Broadcasting 439
 hierarchisches 437
 Internetwork 494
 mobiler Host 445
 Multicasting 441
 Multidestination- 439
 nach dem kürzesten Pfad 423
 Reverse Path Forwarding 439
 statisches 421
 Tunneling 447
 unter Verkehrsberücksichtigung 455
 verkehrsgewahres 454
 Routing-Algorithmus 51, 417, 420
 dynamischer 421
 RPC 619
 RPR 320
 RSA 898
 RSVP 481
 RTCP 625
 RTP 73, 622
 RTP-Header 624
 RTSP-Protokoll 740
 RTS-Rahmen 327
 Rucksack-Algorithmus 900
 Rückstreuung 382
- S**
- S/MIME (Secure/MIME) 954
 S-ALOHA 312
 Sandbox 966
 Satellit
 CubeSat 157
 GEO- 151
 geostationär 150
 GPS- 154
 LEO- 154
 MEO- 154
 Statistisches Zeitmultiplexing 170
 S-Box 882
 Scatternet 373
 Schaltung 60
 Schiebefensterprotokoll 272, 596
 1-Bit- 274
 Schlüssel 869
 öffentlicher 897
 symmetrischer 881
 Schlüsselbund, öffentlich 953
 Schlüsselbund, privat 953
 Schlüsselstrom 893
 Schlüsselverteilungszentrum 912, 935, 942
 Schwarm 849
 SCO 378
 Scrambler 163
 SCTP 602
 SDH 193
 Security by Obscurity 870
 Seeder 850
 Segmentgröße, maximale (MSS) 636
 Segment-Header 633
 Segmentverarbeitung 669
 Selektive Wiederholung 280
 Sendefenster 272
 Sensorsnetz 35, 102
 Server 25
 Serverfarm 90, 836
 Server-Housing 90
 Server-Stub 619
 Session-Routing 420
 SGSN 94
 SHA (Secure Hash Algorithm) 906
 SHA-1 906
 Short InterFrame Spacing siehe SIFS
 Short Message Service siehe SMS
 Sicherheit 864
 ActiveX-Steuerelement 966
 Add-on 968
 Bluetooth 933
 Browsererweiterung 968
 CCMP 932
 drahtlose 929
 E-Mail 949
 IEEE-802.11-Standard 930
 im Web 954
 Java-Applet 965
 JavaScript 967
 MIC 932
 mobiler Code 965
 PGP 949
 Plug-in 968
 S/MIME 954
 TKIP 932
 WEP 930
 WPA2 930
 Sicherheitsanker 917
 Sicherheitsassoziation 920
 Sicherheitsband 168
 Sicherheitsproblem 864
 Sicherungsschicht 68
 Entwurfsaspekte 235
 SIFS 359

| | |
|--|--|
| SIG 372 | G.dmt 185 |
| Signal | IrDA 147 |
| symmetrisches 163 | IS-95 208 |
| Signal-Rausch-Verhältnis 126 | V.32- 182 |
| Signatur | V.32bis- 182 |
| digitale 901 | V.34- 182 |
| mit öffentlichem Schlüssel 903 | V.34bis- 182 |
| mit symmetrischem Schlüssel 902 | V.90- 183 |
| Signaturblock 715 | V.92- 183 |
| Silly Window Syndrome 644 | Standardentwurf 110 |
| SIM-Karte 96, 209 | Standardvorschlag 110 |
| Simple Mail Transfer Protocol siehe SMTP | Statisches Routing 421 |
| Simplex 129 | Statistisches Multiplexing 59 |
| Simplexprotokoll | STDIM siehe statisches Zeitmultiplexing |
| mit Stop-and-Wait 265 | Steganografie 974 |
| utopisches 263 | Steuerungsgesetz 611 |
| Singlemode-Faser 133 | Stop-and-Wait-Protokoll 266, 596 |
| SIP-Methode 829 | Stop-and-Wait-Schiebefensterprotokoll 272 |
| SIPP 522 | Store-and-forward 413 |
| SIP-Protokoll 740, 828 | Store-and-forward-Paketvermittlung 413 |
| Sitzungsschicht 70 | Store-and-forward-Schaltung 60 |
| Sitzungsschlüssel 935 | Stream-Cipher-Modus 893 |
| SLIP 292 | Streaming |
| Slotted ALOHA siehe S-ALOHA | Interleaving 812 |
| Slow-Start 652 | Internetradio 817 |
| Smiley 708 | IPTV 817 |
| SMS 34 | Live-Medien 817 |
| SMTP 73, 710, 725 | Löschnung 812 |
| SOAP 773 | Medien 808 |
| Socket 84, 629 | Obergrenze 815 |
| Soft Handover 95, 216 | Paritätspaket 811 |
| Soft-Decision-Decodierung 250 | Podcast 817 |
| Soliton 135 | Untergrenze 813 |
| SONET 193 | Verschachtelungsstrategie 812 |
| Packet over 291 | Vorwärtsfehlerkorrektur 811 |
| Soziales Netzwerk 30 | Walled Garden 819 |
| Spaltentransposition 873 | Strowger-Schaltung 199 |
| Spam 708 | STS-1 194 |
| Spannbaum 392, 440 | Stub-Bereich 543 |
| Gedicht 394 | Stub-Netz 548 |
| SPE 195 | Subdomäne 696 |
| Special Interest Group siehe SIG | Subnetting 509 |
| Sperrliste 918 | Subnetz 508 |
| Spezielle IP-Adressierung 516 | Subscriber Identity Module siehe SIM-Karte |
| Spiegeln 842 | Substitution, monoalphabetische 872 |
| Splitter 186 | Substitutionschiffre 871 |
| Spreizbandtechnik 140 | Supernetz 511 |
| Spyware 752 | Switch 43, 395 |
| SSL (Secure Sockets Layer) 961 | Switched Ethernet 43, 337 |
| Stadtnetz 46 | Symbol 160 |
| Standard | Symbolrate 161 |
| De-facto- 103 | Symmetrischer Schlüssel 881 |
| De-jure- 104 | Symmetrisches Signal 163 |

- Synchronisation 70
 SYN-Cookie 638
 SYN-Flood 638
 Systementwurf 666
 Systemwiederherstellung 602
- T**
- T1-Leitung 162
 T1-Träger 190
 Tag 753
 Tag Switching 537
 Taildrop 474
 Taktrückgewinnung 161
 Talkspur 627
 TCM 182
 TCP 72, 628
 - Dienstmodell 629
 - Protokoll 632
 - Schiebefenster 642
 - Segment 632
 - Segment-Header 633
 - Slow-Start 652
 - Timerverwaltung 646
 - Überlastungsfenster 649
 - Überlastungsüberwachung 649
 - Übertragungsregeln 642
 - Verbindungsaufbau 637
 - Verbindungs freigabe 638
 - Zukunft 659
- TCP NewReno 658
 TCP Reno 657
 TCP Tahoe 655
 TCP/IP-Referenzmodell 70
 TDD 368
 TDM siehe Zeitmultiplexverfahren
 Teilnehmeranschlussleitung 175
 Teilnehmervermittlungsstelle 175
 Telefonmodem siehe Modem
 Telefondienst
 - Aufbau 174
 - öffentliche 174
- Temporäre Maskierung 798
 Textnachricht 33
 Thick Ethernet 330
 Thin Ethernet 330
 Third Generation Partnership Project
 - siehe 3GPP
- Tier-1-Netz 501
 Time Division Duplex siehe TDD
 Timing-Wheel 672
 TKIP 932
 Token Bucket 457
- Token-Bucket-Algorithmus 469
 Token-Bus 320
 Token-Ring-Protokoll 319
 Token-Verwaltung 70
 Top-Level-Domäne 696
 Traceroute 532
 Tracker 849
 Traffic-Engineering 456
 Traffic-Policing 468
 Traffic-Shaping 468
 Tragbare Computer 35
 Trägerfrequenzanlage 31, 46, 130
 Trägerprüfung 308
 Transcodierung 789
 Transit 90
 Transitdienst 546
 Transmission Control Protocol siehe TCP
 Transmit-Power-Control 364
 Transponder 149
 Transportadresse 581
 Transportinstanz 567
 Transportmodus 921
 Transportprotokoll 580
 Transportschicht 69, 72, 566
 - Adressierung 581
 - Dienste 567
 - Dienstprimitive 569
 - Leistungsaspekte 660
 - Multiplexing 601
 - Verbindung freigeben 591
 - Verbindungs aufbau 584
- Transpositionschieffre 873
 Trellis Coded Modulation siehe TCM
 Triangel-Routing-Problem 556
 Triangle-Routing 447
 Triple DES 885
 Trusted Computing 978
 TSAP 581
 Tunneling 447, 493
 Tunnelmodus 921
 Twisted-Pair-Kabel 128
 Twitter 29
 TXOP 360
- U**
- Übergabepunkt 89
 Überlastung 452
 - Choke-Paket 459
 - ECN 460
 - Hop-by-Hop-Rückstau 461
 - Zugangssteuerung 455
- Überlastungsfenster 649

- Überlastungskollaps 452
Überlastungsüberwachung 604
AIMD 611
Effizienz 605
Konvergenz 608
Max-Min-Fairness 606
Prinzipien 454
Senderate regulieren 609
Steuerungsgesetz 611
Überlastungsvermeidung 458
Übertragung
Funk 141
im Durchlassbereich 159, 164
Infrarot 147
Licht- 147
Mikrowelle 143
Übertragungseinheit, maximale 632
Übertragungsleitung 48
Ubiquitous Computing 31
UDP 72, 616
DNS 618
Einführung 616
entfernter Prozeduraufruf 618
Prüfsumme 617
Server-Stub 619
UHF-RFID 101
Ultrabreitbandtechnik 141
UMTS 91, 213
Unchoked-Zustand 851
Underlay-Technik 141
Unicasting 39, 623
U-NII-Band 146
Universal Mobile Telecommunications System siehe UMTS
Universal Resource Identifier siehe URI
Universal Resource Name siehe URN
Universal Serial Bus siehe USB
Untergrenze 813
Unternehmensnetzwerk 42
Upstream-Proxy 840
Upward-Multiplexing 601
Urheberrecht 976
URI 741
URL (Uniform Resource Locator) 738
URN 741
USB 162
User Datagram Protocol siehe UDP
UTP-Kabel 129
UWB siehe Ultrabreitbandtechnik
- V**
- V.32bis-Standard 182
V.32-Standard 182
V.34bis-Standard 182
V.34-Standard 182
V.90-Standard 183
V.92-Standard 183
VBScript 769
VC-Netz 415
Verbindungsauflaufbau 584
Verbindungsauflaufbauprotokoll 583
Verbindungsausnutzung 278
Verbindungsfreigabe 591
Verbindungsleitung 176
Verbindungsloser Dienst 60
Implementierung 415
Verbindungsorientierter Dienst 60
vergifteter Cache 957
Vermittlungselement 48
Vermittlungsschicht 69
Dienste 414
Entwurfsaspekte 413
Versatz siehe Interleaving
Verschlüsselung
AES 886
Chiffriermodus 890
Cipher Block Chaining 891
Cipher-Feedback-Modus 892
Counter-Modus 895
DES 883
Electronic-Code-Book-Modus 890
Grundprinzipien 879
mit öffentlichen Schlüsseln 911
Redundanz 879
Schlüsselstrom 893
Stream-Cipher-Modus 893
Triple DES 885
Verteiltes System 23
Vertraulichkeit 60
Verzögerung durch Warteschlangenbildung 201
Video
AVC 805
Chrominanz 802
digitales 799
diskrete Kosinustransformation 803
Einführung 799
HDTV 801
Interlacing 800
JPEG 801
Kammeffekt 800
Lauflängencodierung 804

- Luminanz 802
 - MPEG 804
 - Video-on-Demand 808
 - Virtuelle Verbindung 50, 296, 415
 - Virtuelles LAN siehe VLAN
 - Virtuelles privates Netz 24, 50
 - Virtuelles privates Netz (VPN) 928
 - Virus
 - Sicherheit 968
 - Viterbi-Algorithmus 250
 - VLAN 44, 386, 397
 - IEEE-802.1Q-Standard 401
 - VLR 209
 - Vocoder 796
 - Voice-over-IP 26, 61, 793, 821
 - VoIP siehe Voice-over-IP
 - Vollduplex 129
 - Vorrangdaten 631
 - Vorwärtsfehlerkorrektur 244, 615, 811
 - VPN 494
 - VPN siehe virtuelles privates Netz
 - VSAT 152
- W**
- W3C 110, 735
 - Wählleitung 88
 - Walsh-Code 171
 - WAN 47
 - WAP 787
 - Watermarking 976
 - WCDMA 92, 213
 - WDM siehe Wellenlängenmultiplexing
 - Web
 - WAP 787
 - Webanwendung 25
 - Webcrawling 790
 - Webmail 733
 - Webseite 735
 - dynamische 763
 - dynamische Erzeugung 764, 767
 - statische 752
 - Webserver 744
 - Apache 767
 - Websicherheit 954
 - Website
 - Bedrohung 954
 - Webtracking 750
 - Weighted Fair Queueing siehe WFQ
 - Weiterleitung 420
 - Wellenform-Codierung 797
 - Wellenlänge 138
 - Wellenlängenmultiplexing 196
 - WEP 100, 363, 930
 - WFQ 476
 - White Space 146
 - Whitening 884
 - Wide Area Network siehe WAN
 - Wideband CDMA siehe WCDMA
 - Wideband Code Division Multiple Access
 - siehe WCDMA
 - Wiederherstellung 602
 - Wiederholung, selektive 280
 - WiFi 43, 97
 - WiFi Alliance 103
 - WiFi Protected Access siehe WPA
 - Wiki 30
 - Wikipedia 30
 - WiMAX 47, 96, 217, 364
 - WiMAX Forum 365
 - Wired Equivalent Privacy siehe WEP
 - World Wide Web 23, 734
 - Architektur 735
 - Bookmark 741
 - Browser 736
 - Clientseite 738
 - Cookie 748
 - Grundmodell 737
 - HTTP 737
 - Hyperlink 736
 - Hypertext 735
 - Link 735
 - mobiles 787
 - Serverseite 744
 - Suche 789
 - URL 738
 - Webcrawling 790
 - Webseite 735
 - World Wide Web Consortium siehe W3C
 - Wormhole-Routing 390
 - WPA 100
 - WPA2 100, 362, 930
 - WWW siehe World Wide Web
- X**
- X.400 715
 - X.509 914
 - XCP 610
 - xDSL 183
 - XHTML 773
 - XHTML Basic 788
 - XML 771
 - XSLT 773

Z

- Zeichengabe
 - kanalgebundene 191
- Zeitduplexverfahren 368
- Zeitmultiplexing 190
- Zeitmultiplexverfahren 169
 - statistisches 170
- Zelle 205
- Zellulärer Netzentwurf 92
- Zensur 969
- Zertifikat 912
 - annullieren 918
 - Kette 917
 - PKI 915
 - Sicherheitsanker 917
 - speichern 917
 - Sperrlisten 918
 - X.509 914
- Zertifizierungsstelle 912
- Zielport 518
- Zipf'sches Gesetz 834
- Zone 703
- Zugangspunkt 42, 97
- Zugangssteuerung 455, 457, 477
- Zwei-Armeen-Problem 591
- Zwischengespeicherter Datensatz 704
- Zyklische Redundanzprüfung 255

FIFTH EDITION

COMPUTER NETWORKS



TANENBAUM | WETHERALL

Copyright

Daten, Texte, Design und Grafiken dieses eBooks, sowie die eventuell angebotenen eBook-Zusatzdaten sind urheberrechtlich geschützt. Dieses eBook stellen wir lediglich als **persönliche Einzelplatz-Lizenz** zur Verfügung!

Jede andere Verwendung dieses eBooks oder zugehöriger Materialien und Informationen, einschließlich

- der Reproduktion,
- der Weitergabe,
- des Weitervertriebs,
- der Platzierung im Internet, in Intranets, in Extranets,
- der Veränderung,
- des Weiterverkaufs und
- der Veröffentlichung

bedarf der **schriftlichen Genehmigung** des Verlags. Insbesondere ist die Entfernung oder Änderung des vom Verlag vergebenen Passwortschutzes ausdrücklich untersagt!

Bei Fragen zu diesem Thema wenden Sie sich bitte an: info@pearson.de

Zusatzdaten

Möglicherweise liegt dem gedruckten Buch eine CD-ROM mit Zusatzdaten bei. Die Zurverfügungstellung dieser Daten auf unseren Websites ist eine freiwillige Leistung des Verlags. **Der Rechtsweg ist ausgeschlossen.**

Hinweis

Dieses und viele weitere eBooks können Sie rund um die Uhr und legal auf unserer Website herunterladen:

<http://ebooks.pearson.de>