Peter Mandl | Andreas Bakomenko | Johannes Weiß

Grundkurs Datenkommunikation

TCP/IP-basierte Kommunikation: Grundlagen, Konzepte und Standards

2. Auflage

► Mit Online-Service

STUDIUM

VIEWEG+ TEUBNER

Peter Mandl | Andreas Bakomenko | Johannes Weiß

Grundkurs Datenkommunikation

Peter Mandl | Andreas Bakomenko | Johannes Weiß

Grundkurs Datenkommunikation

TCP/IP-basierte Kommunikation: Grundlagen, Konzepte und Standards

2., überarbeitete und aktualisierte Auflage

Mit 256 Abbildungen und 21 Tabellen

STUDIUM

VIEWEG+
TEUBNER

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

Höchste inhaltliche und technische Qualität unserer Produkte ist unser Ziel. Bei der Produktion und Auslieferung unserer Bücher wollen wir die Umwelt schonen: Dieses Buch ist auf säurefreiem und chlorfrei gebleichtem Papier gedruckt. Die Einschweißfolie besteht aus Polyäthylen und damit aus organischen Grundstoffen, die weder bei der Herstellung noch bei der Verbrennung Schadstoffe freisetzen.

1. Auflage 2008
2., überarbeitete und aktualisierte Auflage 2010

Lektorat: Christel Roß | Maren Mithöfer

Vieweg+Teubner Verlag ist eine Marke von Springer Fachmedien.
Springer Fachmedien ist Teil der Fachverlagsgruppe Springer Science+Business Media.
www.viewegteubner.de

Umschlaggestaltung: KünkelLopka Medienentwicklung, Heidelberg
Druck und buchbinderische Verarbeitung: MercedesDruck, Berlin
Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier.
Printed in Germany

ISBN 978-3-8348-0810-3

Vorwort

Obwohl es Netzwerke zur Datenkommunikation noch gar nicht so lange gibt, sind sie heute die Basis für viele Anwendungssysteme. Im Zuge der ISO/OSI-Bemühungen in den 80er Jahren wurden einige grundlegende Konzepte für die Datenkommunikation erarbeitet. Jedoch waren die zum Teil sehr guten Ansätze in mancher Hinsicht zu komplex für eine Implementierung. So ganz nebenbei verbreitete sich seit dieser Pionierzeit die TCP/IP-Protokollfamilie immer mehr. Implementierungen waren schon bald auf Unix-Systemen verfügbar, und die Netze wurden immer größer. TCP/IP und die darüberliegenden Protokolle stellen heute einen Standard dar, ohne den viele Netzwerke nicht funktionieren könnten. Beispielsweise ist das globale Internet mit seinen Anwendungen ohne TCP/IP heute nicht mehr vorstellbar. In diesem Buch werden die Grundlagen, wichtige Konzepte und Standards TCP/IP-basierter Kommunikation anhand konkreter Protokolle vermittelt. Es soll dazu beitragen, den komplexen Sachverhalt zu strukturieren und verständlich zu machen. Dabei liegt der Schwerpunkt vor allem bei praktisch relevanten Themen, aber auch die grundlegenden Aspekte sollen erläutert werden. Das Buch behandelt insgesamt die folgenden acht Themenkomplexe:

1. Einführung in Referenzmodelle und Protokolle
2. Technische Grundlagen von Rechnernetzen
3. Ausgewählte Technologien und Protokolle unterer Schichten
4. Konzepte und Protokolle der Vermittlungsschicht
5. Konzepte und Protokolle der Transportschicht
6. Ausgewählte Anwendungsprotokolle
7. Grundlagen der mobilen Kommunikation
8. Entwicklung von Kommunikationsanwendungen

Kapitel 1 gibt eine grundlegende Einführung in Referenzmodelle und Protokolle, wobei auf das ISO/OSI-Referenzmodell und das TCP/IP-Referenzmodell eingegangen wird. Kapitel 2 fasst wichtige technische Grundlagen für das Verständnis der unteren beiden Schichten zusammen. In Kapitel 3 werden einige ausgewählte Netzwerktechnologien und Protokolle aus den unteren Schichten vorgestellt, die heute für den WAN-Zugang sowie im LAN von Bedeutung sind.

In Kapitel 4 und 5 wird detailliert auf die Vermittlungsschicht und die Transportschicht eingegangen, wobei vor allem die wichtigsten Kommunikationsprotokolle aus der TCP/IP-Protokollfamilie als Beispiele dienen.

In Kapitel 6 wird schließlich exemplarisch auf einige Anwendungsprotokolle bzw. -systeme eingegangen. Als Fallbeispiele dienen das Domain Name System, das World Wide Web und E-Mailsysteme. Dies sind Protokolle und Systeme, ohne die das Internet und das WWW heute kaum mehr vorstellbar wären. Weiterhin werden einige Multimedia-Protokolle, die zunehmend an Bedeutung gewinnen, diskutiert. Kapitel 7 befasst sich mit den Grundlagen von mobilen Systemen, die heute Einzug in verschiedenste Anwendungen halten, und Kapitel 8 zeigt schließlich auf, wie man Kommunikationsanwendungen auf Basis von Transportzugriffsschnittstellen wie Sockets programmiert.

In diesem Buch wird ein praxisnaher Ansatz gewählt. Der Stoff wird mit vielen Beispielen aus aktuell relevanten Protokollstandards angereichert. Für das Verständnis einiger Programmbeispiele sind grundlegende Kenntnisse von Programmiersprachen (C, C++, Java und C#) nützlich, jedoch können die wesentlichen Konzepte auch ohne tiefere Programmierkenntnisse verstanden werden. Vom Leser werden ansonsten keine weiteren Grundkenntnisse vorausgesetzt.

Der Inhalt des Buches entstand aus mehreren Vorlesungen über Datenkommunikation an der Hochschule für angewandte Wissenschaften – FH München.

Bedanken möchten wir uns sehr herzlich bei unseren Studentinnen und Studenten, die uns Feedback zum Vorlesungsstoff gaben. Auch den Gutachtern danken wir für ihre guten Verbesserungsvorschläge. Dem Verlag, insbesondere Frau Sybille Thelen, danken wir für die großartige Unterstützung im Projekt und für die sehr konstruktive Zusammenarbeit.

In der vorliegenden 2. Auflage wurden textliche Überarbeitungen sowie einige Aktualisierungen vorgenommen, und es wurden zahlreiche Fehler behoben. Der Aufbau des Buches blieb aber unverändert.
Fragen und Korrekturvorschläge richten Sie bitte an mandl@cs.hm.edu.

Für begleitende Informationen zur Vorlesung siehe www.prof-mandl.de.

München, März 2010 *Peter Mandl, Andreas Bakomenko, Johannes Weiß*

Inhaltsverzeichnis

1 Einführung in Referenzmodelle und Protokolle

Kommunikation ist der Austausch von Informationen nach bestimmten Regeln. Dies ist zwischen Menschen ähnlich wie zwischen Maschinen. Die Regeln fasst man in der Kommunikationstechnik unter dem Begriff *Protokoll* zusammen. Auch beim Telefonieren ist von den Beteiligten ein Protokoll einzuhalten, sonst funktioniert die Kommunikation nicht. Wenn z.B. beide Teilnehmer gleichzeitig reden, versteht keiner etwas. Ähnlich verhält es sich bei der Rechnerkommunikation.

In mehreren Gremien und Organisationen wurde in den letzten Jahrzehnten versucht, die komplexe Materie der Telekommunikation und der Datenübertragung in Modellen zu formulieren und zu standardisieren, einheitliche Begriffe einzuführen und Referenzmodelle für die Kommunikation zu schaffen. Einen wesentlichen Beitrag leistete hierzu die ISO (International Standardization Organization), aber auch die TCP/IP-Gemeinde und die ITU-T[1] setzen hier Standards.

In diesem Kapitel soll eine Einführung in die beiden wichtigsten Modelle der Datenkommunikation gegeben werden. Das ISO/OSI- und das TCP/IP-Referenzmodell werden erläutert. Weiterhin werden Protokollfunktionen, die in verschiedenen Ebenen zum Einsatz kommen können, aufgezeigt.

Zielsetzung des Kapitels

Ziel dieses Kapitels ist es, eine erste Einführung in die Problematik der Datenkommunikation zu vermitteln. Der Studierende soll grundlegende Begriffe kennenlernen und erläutern können.

Wichtige Begriffe

ISO/OSI-Referenzmodell, PDU, IDU, ICI, SDU, SAP, Diensterbringer, Dienstnehmer, Instanz, Kommunikationschichten, TCP/IP-Referenzmodell, Bestätigung, ACK, NACK, Flusssteuerung, Staukontrolle, Multiplexieren.

---

[1] ITU-T ist der Telekom-Sektor der International Telekommunication Union (ITU) und löste 1993 die CCITT ab. Die ITU-T dient der Standardisierung im Telekom-Umfeld.

1.1 Das ISO/OSI-Referenzmodell

Die gesamte Funktionalität, die hinter der Rechnerkommunikation steckt, ist zu komplex, um ohne weitere Strukturierung verständlich zu sein. Im Rahmen der Standardisierungsbemühungen der ISO hat man sich daher gemäß dem Konzept der virtuellen Maschinen mehrere Schichten ausgedacht, um die Materie etwas übersichtlicher zu beschreiben.

Das ISO/OSI-Referenzmodell (kurz: OSI-Modell) teilt die gesamte Funktionalität in sieben Schichten (siehe Abbildung 1-1) ein, wobei jede Schicht, der darüberliegenden Schicht Dienste für die Kommunikation bereitstellt. Die unterste Schicht beschreibt die physikalischen Eigenschaften der Kommunikation, und darüber liegen Schichten, die je nach Netzwerk teilweise in Hardware und teilweise in Software implementiert sind.

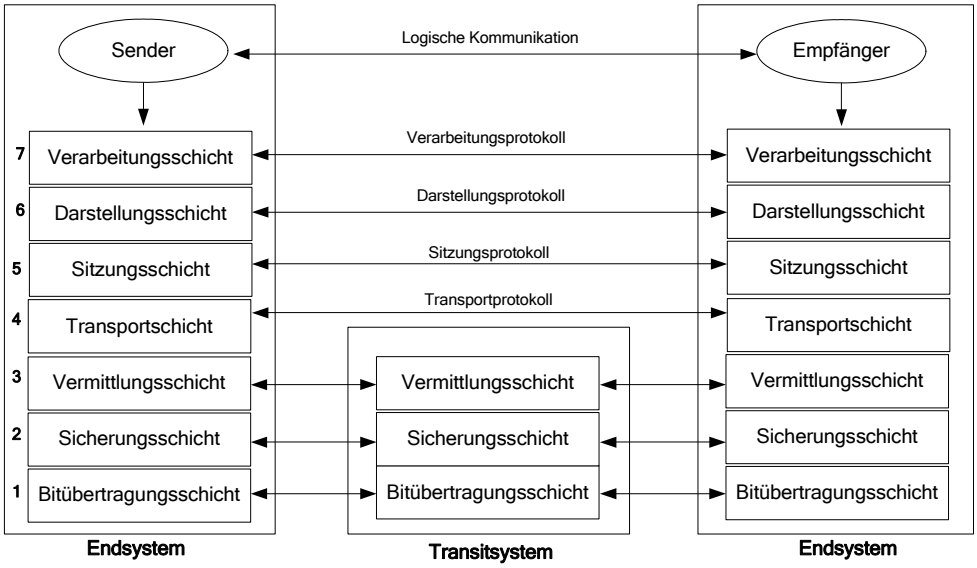


Abbildung 1-1: ISO/OSI-Referenzmodell

In einem Rechnersystem, das nach OSI ein offenes System darstellt, werden die einzelnen Schichten gemäß den Standards des Modells implementiert. Es ist aber nicht vorgeschrieben, dass alle Schichten, und innerhalb der Schichten alle Protokolle, implementiert sein müssen. Das Referenzmodell mit seinen vielen Protokollen beschreibt keine Implementierung, sondern gibt nur eine Spezifikation vor. Die eigentliche Kommunikationsanwendung, die in einem oder mehreren Betriebssystemprozessen ausgeführt wird, gehört nicht in eine Schicht, sondern nutzt nur die Dienste des Kommunikationssystems.

Die Schichten einer Ebene, die in verschiedenen Rechnersystemen (auch offene Systeme genannt) instanziert werden, kommunizieren horizontal miteinander über Protokolle. Jede Schicht innerhalb eines offenen Systems stellt der nächst höheren Dienste zur Verfügung. Das Konzept der virtuellen Maschine findet hier insofern Anwendung, als keine Schicht die Implementierungsdetails der darunterliegenden Schicht kennt und eine Schicht n immer nur die Dienste der Schicht n-1 verwendet.

Das Modell unterscheidet sog. Endsysteme und Transitsysteme (siehe hierzu Abbildung 1-1). Endsysteme implementieren alle Schichten des Modells, während Transitsysteme nur die Schichten 1 bis 3 realisieren und als Verbindungssysteme dienen, die unterschiedliche Teilstrecken zwischen den Endsystemen abdecken.

Die einzelnen Schichten haben im ISO/OSI-Referenzmodell folgende Aufgaben:

- Die *Bitübertragungsschicht (Schicht 1)* ist die unterste Schicht und stellt eine physikalische Verbindung bereit. Hier werden die elektrischen und mechanischen Parameter festgelegt. U.a. wird in dieser Schicht spezifiziert, welcher elektrischen Größe ein Bit mit Wert 0 oder 1 entspricht. Hier werden nur Bits bzw. Bitgruppen ausgetauscht.
- Die *Sicherungsschicht (Schicht 2)* sorgt dafür, dass ein Bitstrom einer logischen Nachrichteneinheit zugeordnet ist. Hier wird eine Fehlererkennung und -korrektur für eine Ende-zu-Ende-Beziehung zwischen zwei Endsystemen bzw. Transitsystemen unterstützt.
- Die *Netzwerk- oder Vermittlungsschicht (Schicht 3)* hat die Aufgabe, Verbindungen zwischen zwei Knoten ggf. über mehrere Rechnerknoten hinweg zu ermöglichen. Auch die Suche nach einem günstigen Pfad zwischen zwei Endsystemen wird hier durchgeführt (Wegewahl, Routing).
- Die *Transportschicht (Schicht 4)* sorgt für eine Ende-zu-Ende-Beziehung zwischen zwei Kommunikationsprozessen und stellt einen Transportdienst für die höheren Anwendungsschichten bereit. Die Schichten eins bis vier bezeichnet man zusammen auch als das *Transportsystem*.
- Die *Sitzungsschicht (Schicht 5)* stellt eine Sitzung (Session) zwischen zwei Kommunikationsprozessen her und regelt den Dialogablauf der Kommunikation.
- Die *Darstellungsschicht (Schicht 6)* ist im Wesentlichen für die Bereitstellung einer einheitlichen Transfersyntax zuständig. Dies ist deshalb wichtig, da nicht alle Rechnersysteme gleichartige Darstellungen für Daten verwenden. Manche nutzen z.B. den EBCDIC[2]- andere den ASCII[3]-Code und wieder an-

[2] EBCDIC (= Extended Binary Coded Decimal Interchange Code) wird zur Kodierung von Zeichen verwendet und wird z.B. von Mainframes genutzt.

[3] ASCII (=American Standard Code for Information Interchange) wird zur Kodierung von Zeichen verwendet und ist auch unter der Bezeichnung ANSI X3.4 bekannt.

dere den Unicode[4]. Auch die Byte-Anordnung bei der Integer-Darstellung (Little Endian und Big Endian Format) kann durchaus variieren. Unterschiedliche lokale Syntaxen werden in dieser Schicht in eine einheitliche, für alle Rechnersysteme verständliche Syntax, die Transfersyntax, übertragen.
- Die *Verarbeitungs-* oder *Anwendungsschicht (Schicht 7)* enthält schließlich Protokolle, die eine gewisse Anwendungsfunktionalität wie Filetransfer oder E-Mail bereitstellen. Die eigentliche Anwendung zählt nicht zu dieser Schicht. Hier sind viele verschiedene Protokolle angesiedelt.

Für jede Schicht gibt es im Modell mehrere Protokollspezifikationen für unterschiedliche Protokolle. In der Schicht 4 gibt es z.B. verschiedene Transportprotokolle mit unterschiedlicher Zuverlässigkeit.

Grob kann man auch festhalten: Die Schicht 2 dient der Verbindung zwischen zwei Rechnern (Endsystemen oder Zwischenknoten), sorgt sich also um eine Ende-zu-Ende-Verbindung zwischen zwei Rechnern. Die Schicht 3 unterstützt Verbindungen im Netzwerk (mit Zwischenknoten), also Ende-zu-Ende-Verbindungen zwischen zwei Rechnern (Endsysteme) über ein Netz. Die Schicht 4 kümmert sich um eine Ende-zu-Ende-Kommunikation zwischen zwei Prozessen auf einem oder unterschiedlichen Rechnern.

Eine konkrete Protokollkombination wird auch als *Protokollstack* bezeichnet. Der Begriff Stack wird deshalb verwendet, weil Nachrichten innerhalb eines Endsystems von der höheren zur niedrigeren Schicht übergeben werden, wobei jedes Mal Steuerinformation ergänzt wird. Diese Steuerinformation wird im empfangenden System in Schicht 1 beginnend nach oben gereicht und in umgekehrter Reihenfolge entfernt.

Darüber hinaus sind folgende Begriffe aus dem OSI-Modell wichtig:

- An einem *Dienstzugangspunkt (*oder *Service Access Point, SAP)* werden die Dienste einer Schicht bereitgestellt. Ein SAP ist dabei eine logische Schnittstelle, deren konkrete Realisierung nicht vorweggenommen wird und z.B. in einer Funktionsbibliothek oder in einem eigenen Prozess liegen könnte. Ein SAP wird durch eine SAP-Id identifiziert und adressiert.
- Ein *Dienst* ist eine Sammlung von Funktionen, die eine Schicht an einem SAP bereitstellt. Ein Dienst hat ggf. mehrere *Dienstelemente*. Beispielsweise verfügt der Dienst *connect* über die Dienstelemente *connect.request, connect.indication*, *connect.response* und *connect.confirmation*.
- Ein *Dienstnehmer* nutzt die Dienste einer darunterliegenden Schicht über einen SAP.
- Als *Diensterbringer* (Service Provider) bezeichnet man eine Schicht, die einen Dienst bereitstellt.

[4] Unicode ist ein hardware- und plattformunabhängiger Code zur Zeichenkodierung und wird z.B. in der Sprache Java verwendet.

– Unter einer *Instanz* versteht man eine Implementierung einer konkreten Schicht. Instanzen gleicher Schichten kommunizieren untereinander über ein gemeinsames Protokoll. Diese Kommunikation bezeichnet man im Gegensatz zur vertikalen Kommunikation aufeinander folgender Schichten auch als horizontale Kommunikation.

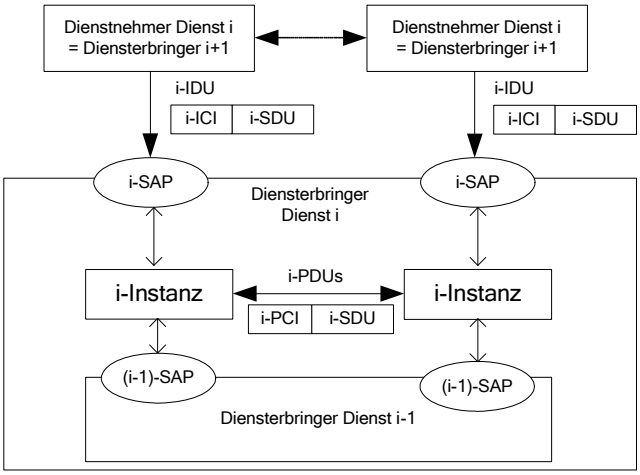


Abbildung 1-2: Hierarchische Dienstestruktur

Weitere Begriffe aus dem Modell sind PDU, IDU, ICI, SDU, PCI und IDU, die hier noch kurz erläutert und in Abbildung 1-3 veranschaulicht werden:

– Nachrichteneinheiten, also Datenpakete (siehe Abbildung 1-2 und Abbildung 1-3), die übertragen werden, bezeichnet man als Protocol Data Units (PDU).
– Unter einer IDU (Interface Data Unit) wird eine Schnittstelleneinheit verstanden, die eine Schicht n+1 einer Schicht n übergibt. Diese Nachricht besteht aus einer ICI (Interface Control Information) und einer SDU (Service Data Unit). Die ICI stellt die Steuerinformation für die Schicht n bereit und wird in der Schicht n interpretiert. Die SDU ist die eigentliche Nutzinformation, die übertragen werden soll.
– Die SDU wird von der Instanz in eine PDU eingepackt und mit einem Header, der sog. PCI (Protocol Control Information) versehen. Die Instanz der Sendeseite sendet die gesamte PDU logisch über das Netz zum Empfänger. Dort wird die PCI von der korrespondierenden Empfänger-Instanz interpretiert, und die SDU wird nach oben zur Schicht n+1 weitergereicht. Die eigentliche physikalische Übertragung findet natürlich in Schicht 1 statt, und aufgrund der Schichtenbearbeitung bildet sich eine Gesamt-PDU mit sieben PCIs, die tatsächlich übers Netz geht.

In Abbildung 1-2 ist die hierarchische Struktur eines Kommunikationssystems nach dem OSI-Modell verallgemeinert dargestellt. Ein Dienstnehmer der Schicht i, der gleichzeitig evtl. Diensterbringer für eine höhere Schicht i+2 ist, greift also über einen i-SAP auf den Diensterbringer i zu. Oft werden über einen i-SAP mehrere i-Verbindungen verwaltet und damit sog. Connection End Points (CEP) eingerichtet.

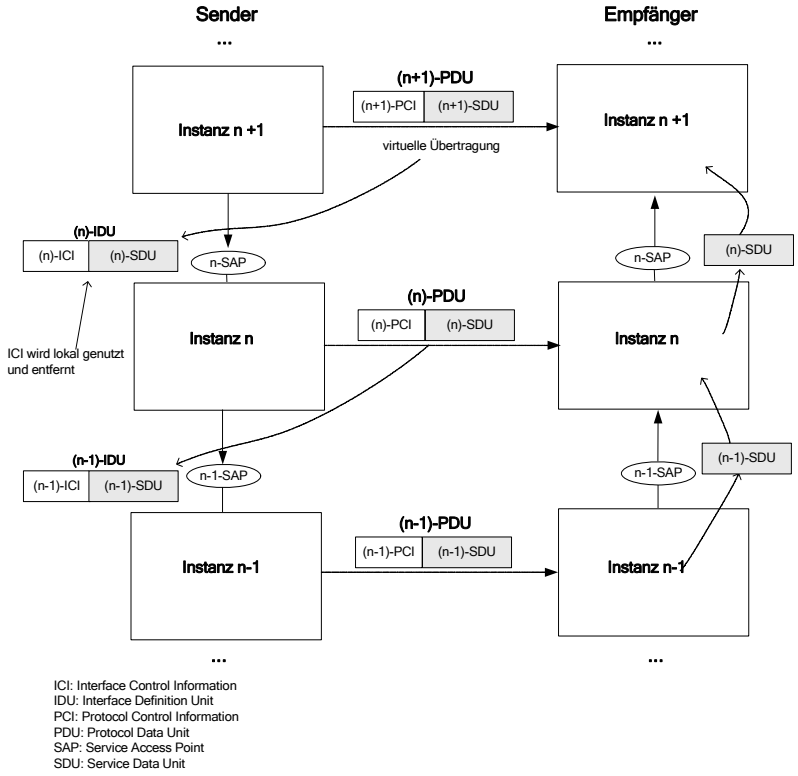


Abbildung 1-3: Beziehungen zwischen den Schichten

Die entsprechende i-Instanz führt den Dienst aus und sendet dabei ggf. wieder über eine darunterliegende Schicht eine Nachricht an den adressierten Partner. Der Diensterbringer i soll gegenüber der Schicht i+1 wie eine virtuelle Maschine arbeiten. Die ICI hat nur lokale Bedeutung und unterliegt damit auch nicht der OSI-Standardisierung.

Eine komplette i-PDU einschließlich der i-PCI wird um Steuerinformation (i-1)-ICI ergänzt und an die darunterliegende Schicht zur Beförderung weiter gereicht. Die i-PDU ist also identisch mit der (i-1)-SDU. Es gibt auch PDUs, die keinen SDU-Anteil enthalten, sondern nur der Steuerung dienen. Diese PDUs haben dann nur einen Header, also eine PCI und keinen Nutzdatenteil.

Die Abbildung 1-3 zeigt wie ein Sender stellvertretend in den Schichten n+1, n und n-1 arbeitet. Die Steuerinformationen[5] (ICI) haben nur lokale Bedeutung, während in den eigentlichen Nachrichten die Header (PCI) mitgesendet werden, die dann auf der Empfängerseite interpretiert und von der eigentlichen Nachricht (SDU) entfernt werden müssen. PCIs können entweder als Header (also vorneweg), als Trailer (am Ende der Nachricht) oder in zwei Teilen (Header und Trailer, siehe später z.B. das HDLC-Protokoll) gesendet werden. Für die logische Betrachtung spielt dies keine Rolle. Beim Sender wird in jeder Schicht von oben nach unten ein Header (PCI) ergänzt, und beim Empfänger wird diese wieder Schicht für Schicht von unten nach oben entfernt, bis in der höchsten Schicht 7 nur noch die eigentlichen Nutzdaten (SDU) vorliegen. Der Nachrichtenaustausch wird physikalisch in Schicht 1 abgewickelt, alle anderen Datenübertragungen zwischen den einzelnen Instanzen gleicher Schichten sind virtuell.

Die Begrifflichkeit wird noch schichtenspezifisch verfeinert. Jeder Bezeichnung (Instanz, PDU, IDU, SDU,...) wird nämlich ein Präfix vorneweg gestellt, das die konkrete Zuordnung zu einer Schicht anzeigt. Eine PDU der Schicht 4 erhält zum Beispiel das Präfix „T“ und wird als T-PDU bezeichnet. Ähnlich verhält es sich mit anderen PDUs („PH“ für Schicht 1, „DL“ für Schicht 2, „N“ für Schicht 3, „S“ für Schicht 5, „P“ für Schicht 6 und „A“ für Schicht 7).

Eine PDU der Anwendungsschicht, die bei Ausnutzung aller Schichten aus sieben Headern (PCI) und einem Nutzdatenteil besteht, ist in Abbildung 1-4 dargestellt. Hier wird auch deutlich, dass eine PDU einer niedrigeren Schicht i (mit i >= 1) alle Header der höheren Schichten i+1, i+2, usw. sowie den eigentlichen Nutzdatenteil (n-SDU) enthält. Man kann sich das Ganze auch so vorstellen, dass ein Brief beim Absender in ein Kuvert verpackt wird, dieses wieder in ein weiteres usw., also insgesamt sieben Kuverts verwendet werden, die dann auf der Empfängerseite ausgepackt werden müssen, um an die eigentliche Information zu kommen.

Zum Abschluss dieser etwas trockenen Begriffsdefinitionen soll nochmals in Erinnerung gerufen werden, was ein Protokoll im Gegensatz zu einem Dienst ist: Ein Protokoll ist eine Verhaltensrichtlinie, auf deren Grundlage Datenstationen untereinander kommunizieren können. In einem Protokoll werden bestimmte Spielregeln der Kommunikation vorgegeben, auf deren Basis gemäß dem OSI-Modell Schichten horizontal miteinander kommunizieren, also z.B. eine Instanz der Schicht 4 mit einer anderen Instanz der Schicht 4, die auf einem anderen Rechnersystem liegen kann, aber nicht muss.

[5] Steuerinformationen können auch für Dienste mit rein lokalen Auswirkungen genutzt werden, ohne dass eine PDU erzeugt und gesendet wird. Es sind einfach die Parameter, die am SAP bei einem Dienstaufruf übergeben werden.

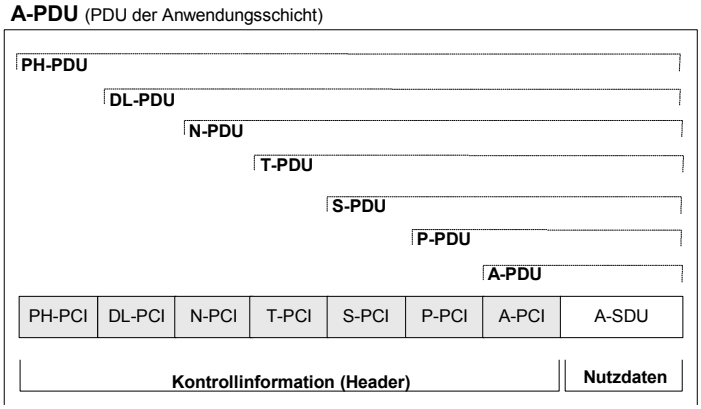


Abbildung 1-4: Eine PDU der Anwendungsschicht

Ein Dienst dagegen ist eine bestimmte Funktionalität, die ein Kommunikationssystem lokal an einer Schichtengrenze über einen SAP bereitstellt, und ist damit ein Abstraktionskonzept. Die Implementierung der diensterbringenden Schicht ist für den Dienstnutzer nicht relevant, wichtig ist die Dienstleistung. Dienstschnittstellen dienen der vertikalen Kommunikation zwischen aufeinanderfolgenden Schichten eines Rechnersystems. Ein Dienst setzt sich aus einem oder mehreren Dienstelementen zusammen, wobei das OSI-Modell die Begriffe Request, Indication, Response und Confirmation verwendet.

In Abbildung 1-5 ist die Beziehung zwischen Diensten bzw. Dienstelementen und einem Protokoll skizziert. Hier wird ein einfacher Verbindungsaufbau dargestellt. Prozess A setzt einen Verbindungsaufbauwunsch als Request (Dienstelement *connect.req*) am T-SAP ab. Die T-Instanz erzeugt daraufhin eine T-PDU (*connect.req-PDU*) und sendet sie zum gewünschten Empfänger (adressiert im *connect.req*). Die PDU kommt bei der T-Instanz des Empfängers an und wird an den Prozess B in Form eines *connect.ind* (Indication) weitergereicht. Dieser bestätigt den Verbindungsaufbauwunsch mit dem Dienstelement *connect.rsp* (Response). Die zugehörige T-Instanz wandelt das Dienstelement in eine *connect.rsp-PDU* um und sendet sie an den Prozess A. In der zugehörigen T-Instanz angekommen, wird ein Dienstelement *connect.cnf* (Confirmation) erzeugt und an den Prozess A weitergereicht. Damit steht die Verbindung und im Anschluss daran sendet der Prozess A eine Nachricht an Prozess B (*data.req,...*).

ISO/OSI-Protokolle- und Dienste haben sich eigentlich nicht richtig durchgesetzt, obwohl viele Dokumente zur Beschreibung von Protokollen für die einzelnen Schichten ausgearbeitet wurden. Beispiele hierfür sind das Transport-Protokoll OSI TP4, die Abstract Syntax Notation 1 mit den entsprechenden Kodierungsregeln für eine einheitliche Transportsyntax (ASN.1/BER) in Schicht 6 und viele Anwen-

dungsprotokolle wie etwa ein Remote Procedure Call (RPC), ein Remote Operation Service (ROSE), die Transaktionsprotokolle OSI TP (Transaction Protocol) und OSI CCR (Commitment, Concurrency und Recovery), sowie das Netzwerk- und Systemmanagement-Protokoll CMIP (Common Management Information Protocol). Weitere Informationen zu den OSI-Protokollen sind in (Gerdsen 1995) zu finden.

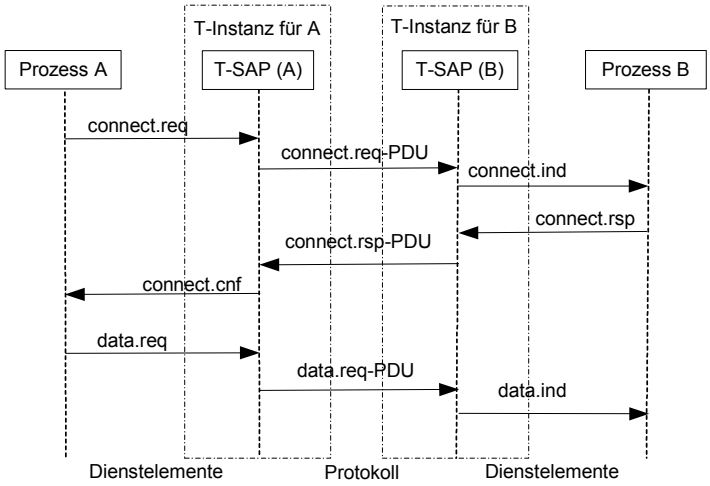


Abbildung 1-5: Beispiel des Zusammenspiels zwischen Dienstaufrufen und Protokoll

Anmerkung zu den ISO/OSI-Standards. Die ISO/OSI-Protokolle und -Dienste gelten heute als zu komplex und auch als etwas zu inflexibel insbesondere in den Schichten 5 und 6. Wie so oft haben sich leichtgewichtigere Ansätze in der Praxis durchgesetzt. Anwendung finden OSI-Protokolle heute noch im Telekom-Umfeld und hier vorwiegend mit CMIP. Ansonsten wurden sie weitgehend durch TCP/IP-Protokolle verdrängt bzw. wurden nie praxisrelevant. Das ISO/OSI-Referenzmodell dient heute aber als ein Modell, in das man neue Protokolle immer wieder einzuordnen versucht, da die grundlegende Idee der Schichtenbildung allgemein akzeptiert wird. Man streitet sich eher um die Anzahl der Schichten und deren Funktionalität (Tanenbaum 2003a).

Die TCP/IP-Protokollfamilie, deren Referenzmodell heute auch in vielen Bereichen als der Defacto-Standard gilt, soll im nächsten Abschnitt erläutert werden.

1.2 Das TCP/IP-Referenzmodell

Das von der Internet-Gemeinde entwickelte TCP/IP-Referenzmodell (vgl. hierzu Abbildung 1-6) ist heute der Defacto-Standard in der Rechnerkommunikation. Es hat vier Schichten, wobei die Schicht 3 (Internet-Schicht) und die Schicht 4 die tragenden Schichten sind. In der Schicht 3 wird neben einigen Steuerungs- und

Adressierungsprotokollen (ARP, ICMP)[6] im Wesentlichen das Protokoll IP (Internet Protocol) benutzt. In Schicht 4 gibt es zwei Standardprotokolle. Das mächtigere, verbindungsorientierte TCP (Transmission Control Protocol) und das leichtgewichtige, verbindungslose UDP (User Datagram Protocol). TCP gab dem Referenzmodell seinen Namen.

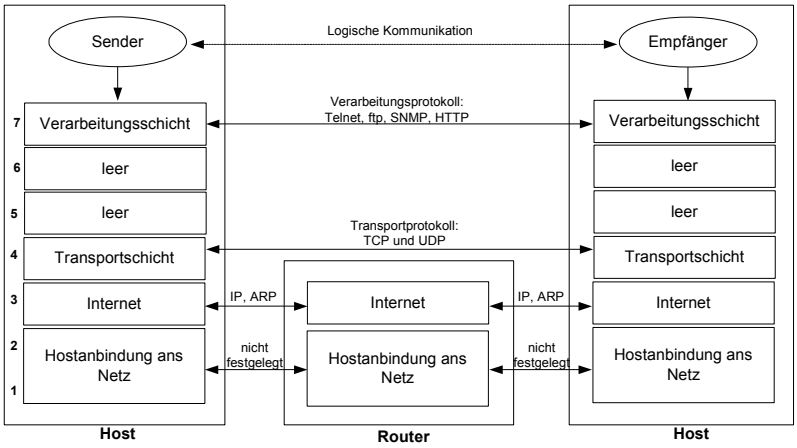


Abbildung 1-6: TCP/IP-Referenzmodell

Im Gegensatz zum OSI-Modell wird im TCP/IP-Modell nicht so streng zwischen Protokollen und Diensten unterschieden. Die Schichten 5 und 6 sind im Gegensatz zum OSI-Modell leer. Diese Funktionalität ist der Anwendungsschicht vorbehalten, d.h. die Verwaltung evtl. erforderlicher Sessions und die Darstellung der Nachrichten in einem für alle Beteiligten verständlichen Format muss die Anwendung selbst lösen. Kritiker des OSI-Modells sind der Meinung, dass gerade die Funktionalität dieser beiden Schichten sehr protokollspezifisch ist. Es gibt Protokolle, die brauchen hier gar keine Funktionalität, und andere, die hier sehr viel tun müssen. Über die unteren Schichten ist man sich prinzipiell einig. Alle Datenkommunikationsspezialisten sehen die Notwendigkeit für ein Transportsystem, wenn auch die Funktionalität je nach Protokolltyp hier sehr unterschiedlich sein kann. Beispielsweise gibt es verbindungsorientierte (wie TCP) und verbindungslose (wie UDP) Protokolle mit recht unterschiedlichen Anforderungen.

In den Schichten 1 und 2 legt sich das TCP/IP-Referenzmodell nicht fest. Hier wird die Anbindung an das Netzwerk gelöst, und dies kann ein LAN- ein WAN- oder ein MAN-Zugriff für jedes beliebige Netzwerk sein. Ein Unterschied ergibt sich dabei bei der Adressierung von Partnern. Während im OSI-Modell auf statische

[6] ARP = Address Resolution Protocol; ICMP = Internet Control Message Protocol.

Schicht-2-Adressierung gesetzt wird (Adressen müssen vorab a-priori konfiguriert sein), nützt man im TCP-Referenzmodell eine Art dynamischer Adressermittlung über das ARP-Protokoll. Hier wird zur Laufzeit eine Schicht-3-Adresse (IP-Adresse) auf eine Schicht-2-Adresse (auch MAC-Adresse, Medium Access Control) abgebildet.

1.3 Weitere Referenzmodelle

Weitere Referenzmodelle, die im Laufe der Zeit und meist bereits vor der OSI-Modellierung entstanden, sind z.B. SNA, DECnet und TRANSDATA.

SNA (Systems Network Architecture) von der Firma IBM ist ähnlich dem OSI-Modell schichtenartig aufgebaut und spielt auch heute noch eine wichtige Rolle, da bei den meisten Großfirmen die unternehmenskritischen Anwendungen auf IBM-Mainframes ablaufen, die in ein SNA-Netzwerk eingebunden sind.

Das Gegenstück zu SNA bei der Firma Siemens heißt TRANSDATA und das von der Firma DEC (heute Compaq bzw. Hewlett Packard), heißt DECnet. DECnet lehnt sich an das OSI-Modell an und wird von der DNA-Produktfamilie (Digital Network Architecture) benutzt.

Darüberhinaus wurde ein neues Referenzmodell für ein Breitbandsystem[7] auf Basis von ATM entwickelt. Es ist auch schichtenorientiert, aber hat nur vier Schichten, wobei die Schichten 1 und 3 jeweils nochmals in Sublayer aufgeteilt sind. Die Schichtung ist völlig anders als im OSI-Modell gelöst.

1.4 Klassische Protokollemechanismen

In der Protokollentwicklung werden in verschiedenen Schichten vergleichbare Protokollfunktionen (Protokollmechanismen) eingesetzt. Da viele dieser Mechanismen nicht nur in einer Schicht zu finden sind, sollen in diesem Kapitel vorab einige wichtige Protokollfunktionen erläutert werden. Die in mehreren Schichten genutzten Protokollmechanismen kann man Funktionsbereichen zuordnen.

Basisprotokollmechanismen. Unter diesen Funktionsbereich fallen der klassische *Datentransfer* und der sog. *Vorrangtransfer*. Der klassische Datentransfer dient der Übertragung der üblichen PDUs. Vorrang-PDUs können früher gesendete „normale“ PDUs überholen. Weiterhin gehört die *Verbindungsverwaltung* mit Verbin-

[7] Anmerkung zur Terminologie: Der Begriff „Breitbandnetze“ hat je nach Kontext etwas unterschiedliche Bedeutung. Man bezeichnet einerseits damit Netze, die trägermodulierte Technik verwenden, also auf einem Kabel mehrere Kanäle realisieren (siehe Kabelfernsehen). In Hochgeschwindigkeitsnetzen bezeichnet man zudem solche Netze als breitbandig, die eine hohe Bitrate (etwa von über 2,048 Mbit/s) bereitstellen und damit für Hochgeschwindigkeitsanforderungen geeignet sind.

dungsaufbau- (connect), -abbau (disconnect) und -abbruch (abort) in diesen Funktionsbereich.

Protokollmechanismen zur Fehlerbehandlung. Der *Sequenznummern*-Mechanismus ist hier anzusiedeln. Dieser Mechanismus dient der Fehlerbehandlung und der Auslieferung der Nachrichten in der richtigen Reihenfolge. Die aufeinander folgenden Nachrichten bzw. in TCP (wie wir unten noch sehen werden) der byteorientierte Nachrichtenstrom wird vom Sender durchnummeriert, so dass vom Empfänger Lücken erkannt werden können.

Die *Quittierung* ist auch in diesen Funktionsbereich einzuordnen. Empfangene Nachrichten werden vom Empfänger quittiert, indem zum Beispiel eine eigene Bestätigungs-PDU (ACK) gesendet wird. Es gibt dafür verschiedene Verfahren. Beispielsweise kann man einzeln oder gruppenweise (kumulativ) quittieren, oder man kann auch nur eine negative Quittierung (NACK = negative acknowledgement) senden, wenn ein Paket fehlt (negative acknowledgement). Auch Mischungen der einzelnen Verfahren sind möglich. Auch sog. Huckepack-Verfahren (picky back acknowledgement) kann ausgeführt werden. Dies bedeutet, dass eine Bestätigung für eine Nachricht mit der Antwort mitgesendet wird.

Zur Fehlerbehandlung gehören weiterhin *Zeitüberwachungsmechanismen* über Timer, *Prüfsummen* und *Übertragungswiederholung* sowie die *Fehlererkennung* und *Fehlerkorrektur* (forward error correction).

Oft wird eine Timerüberwachung für jede gesendete Nachricht verwendet. Ein Timer wird „aufgezogen", um das Warten auf eine ACK-PDU zu begrenzen. Hier ist die Zeitspanne von großer Bedeutung. Sie kann statisch festgelegt oder bei moderneren Protokollen dynamisch anhand des aktuellen Laufzeitverhaltens eingestellt werden. Übertragungswiederholung wird durchgeführt, wenn eine Nachricht beim Empfänger nicht angekommen ist. Der Sender geht z.B. in diesem Fall davon aus, wenn eine ACK-PDU nicht in der Timeout-Zeit ankommt.

Durch Redundanz in den Nachrichten können Fehler festgestellt und bei ausreichender Redundanz auch gleich im Empfänger korrigiert werden. In der Regel verwenden heutige Protokolle nur Fehlererkennungsmechanismen und fordern eine PDU bei Erkennen eines Fehlers erneut an.

Protokollmechanismen zur Längenanpassung. Längenanpassung kann notwendig sein, wenn die unterliegende Protokollschicht die PDU nicht auf einmal transportieren kann und daher eine Zerlegung dieser erfordert. Diesen Mechanismus bezeichnet man als Assemblierung und Deassemblierung. Beim Sender wird die PDU in mehrere Segmente aufgeteilt. Beim Empfänger muss die PDU dann wieder zusammengebaut, also reassembliert werden. Man spricht hier auch – je nach Schicht (z.B. im IP-Protokoll, Schicht 3) – von Fragmentierung und Defragmentierung.

Protokollmechanismen zur Systemleistungsanpassung. In diesen Funktionsbereich fallen Mechanismen wie die *Flusssteuerung*, die *Überlast-* und die *Ratensteue-*

rung. Die Flussteuerung (flow control) schützt den Empfänger vor Überlastung durch den Sender. Hierfür werden z.B Fenstermechanismen (sliding window) eingesetzt, die einem Sender einen gewissen, dynamisch veränderbaren Sendekredit einräumen. Der Sender darf dann so viele PDUs oder Byte ohne eine Bestätigung senden, wie sein Sendekredit vorgibt. Der Empfänger bremst den Sender aus, wenn sein Empfangspuffer zu voll wird und er mit der Verarbeitung, also der Weiterleitung an die nächsthöhere Schicht, nicht nachkommt. Die Überlastsicherung (congestion control) schützt das Netzwerk vor einer Überlastung. Der Sender wird über die Überlast informiert und drosselt daraufhin das Sendeaufkommen, bis die Überlast beseitigt ist. Dies ist meist ein sehr dynamischer Vorgang.

Protokollmechanismen zur Übertragungsleistungsanpassung. Hierunter fallen die Mechanismen zum *Multiplexieren* und *Demultiplexieren* (multiplexing, demultiplexing). Mehrere N-Verbindungen können auf eine (N-1)-Verbindung abgebildet werden (Multiplexen). Auf der Empfängerseite wird es umgedreht (Demultiplexen). Im Gegensatz dazu ist auch der umgekehrte Weg möglich. Man spricht hier auch von Teilung und Vereinigung. Eine N-Verbindung wird auf mehrere (N-1)-Verbindungen verteilt und beim Empfänger wieder vereint. Dies kann vorkommen, wenn eine höhere Schicht über eine höhere Übertragungsleistung verfügt als eine darunterliegende.

Nutzerbezogene Mechanismen. Hierunter kann man Mechanismen der *Dienstgütebehandlung* (quality of service) wie die Ratensteuerung (rate control) einordnen. Diese Mechanismen werden für Multimediaanwendungen benötigt. Dienstgüteparameter sind u.a. der Durchsatz, die Verzögerung und die Verzögerungssensibilität. Diese Parameter müssen zwischen Sender und Empfänger beim Verbindungsaufbau ausgehandelt werden. Unter Ratensteuerung versteht man z.B. das Aushandeln einer zulässigen Übertragungsrate zwischen Sender und Empfänger.

In den weiteren Kapiteln werden die verschiedenen Protokollfunktionen jeweils am konkreten Protokoll diskutiert.

1.5 Übungsaufgaben

1. Erläutern Sie das TCP/IP-Referenzmodell und vergleichen Sie es hinsichtlich seiner Schichten und Dienste mit dem ISO/OSI-Referenzmodell.
2. Nennen Sie Mechanismen zur Fehlerbehandlung in Protokollen.
3. Was ist eine A-PDU?
4. Was ist eine Instanz im ISO/OSI-Referenzmodell?

2 Technische Grundlagen von Rechnernetzen

Unter dem Begriff „technische Grundlagen“ fassen wir im Wesentlichen die nachrichtentechnischen Grundlagen der Schicht 1, auf denen die Datenübertragung basiert, sowie die Techniken, die in der Schicht 2 angesiedelt sind, zusammen. Wichtige Begriffe beider Schichten sollen im Folgenden kurz erläutert werden. Wir beginnen mit der Bitübertragungsschicht und einigen Aspekten der Nachrichtentechnik. Es wird erläutert, was ein Übertragungssystem und ein Übertragungskanal sind. Danach werden grundlegende Aspekte der digitalen Übertragung vorgestellt. Nach einem kurzen Einblick in Übertragungsmedien wird auf die strukturierte Verkabelung von Gebäuden eingegangen. Anschließend werden die Aufgaben der Sicherungsschicht erläutert, Medienzugriffsverfahren werden eingeführt, und im Speziellen werden Buszugriffsverfahren vorgestellt. CSMA-Protokolle, die bekanntesten Buszugriffsverfahren, werden etwas detaillierter behandelt.

Zielsetzung des Kapitels

Ziel dieses Kapitels ist es, einen Überblick über die beiden unteren Schichten der Datenkommunikation zu verschaffen. Der Studierende soll sich ein grundlegendes Verständnis über die Arbeitsweise und das Zusammenspiel verschiedener Techniken und Protokolle verschaffen.

Wichtige Begriffe

Übertragungssystem, Übertragungskanal, digitale Übertragung und PCM, strukturierte Verkabelung, Twisted Pair, UTP, STP, S/UTP, S/STP, XON/XOFF-Protokoll, Medienzugriffsverfahren, CSMA, CSMA/CD, p-persistent CSMA, Kollision, IEEE 802, LLC und MAC, Transferzeit, Laufzeit, Übertragungszeit.

2.1 Bitübertragungsschicht

2.1.1 Aufgaben und Einordnung

Wie der Name sagt, wird in der Bitübertragungsschicht eine Möglichkeit der physikalischen Übertragung von einzelnen Bits oder Bitgruppen über den direkten Zugang zum Übertragungsmedium bereitgestellt.

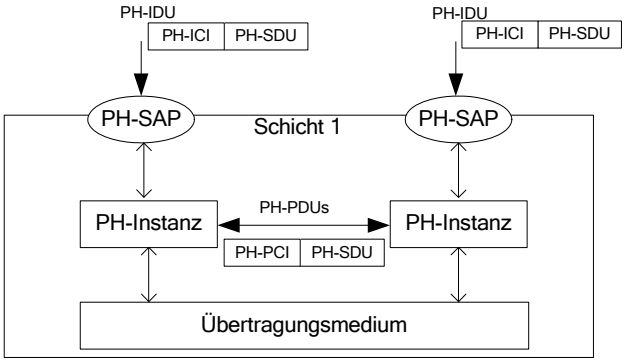


Abbildung 2-1: PH-Instanz

Die Dienste der Bitübertragungsschicht werden der nächsthöheren Schicht an einem PH-SAP angeboten (vgl. Abbildung 2-1). Die PH-Instanzen kommunizieren zur Erfüllung ihrer Aufgaben über PH-PDUs, wobei das Protokoll ein Bitübertragungsprotokoll ist. Als grundlegenden Dienst am PH-SAP kann man sich einen unbestätigten *ph_data*-Dienst vorstellen, der ein Bit oder einen ganzen Bitblock von der Schicht 2 zur Übertragung entgegennimmt. Zusätzliche Dienste gemäß dem ISO/OSI-Referenzmodell sind *ph_activate* zum Aufbau einer Verbindung auf der physikalischen Ebene, *ph_deactivate* zu Abbau und *ph_abort* zum Abbruch einer Verbindung. Am PH-SAP werden bei Nutzung des Dienstes PH-IDUs übergeben.

2.1.2 Grundbegriffe der Nachrichtenübertragung

Zum besseren Verständnis werden in diesem Absatz einige Begriffe aus der Nachrichtentechnik erläutert. Wir beginnen mit der Beschreibung eines Übertragungssystems und erläutern anschließend Begriffe wie Schrittgeschwindigkeit, Baudrate usw.

Übertragungssystem, Information und Signal. Ein Übertragungssystem überträgt Informationen von einer Quelle zu einer Senke. Ein nachrichtentechnischer Kanal dient der Übertragung von Informationen in Form von Bits mittels Transformation in elektrische oder optische Signale. Ein Signal ist dabei eine Wissensdarstellung (in Form einer physikalischen Größe), während eine Information einen Wissensinhalt (Beobachtungen, Meldungen,...) darstellt. Der Sender stellt Daten in den Kanal ein. Diese werden als Signale zum Empfänger übertragen.

Wandler. Ein Wandler wandelt das Quellensignal bei Bedarf in eine andere physikalische Form um. Ein typischer Wandler ist z.B. das Mikrofon, das Luftschwankungen in ein elektrisches Signal umwandelt. Wenn das Signal nach der Umwandlung noch nicht übertragbar ist, wird es an den Kanal angepasst (Kanalanpassung). Eine typische Anpassung wird bei Sprachsignalen durchgeführt, die über Funk zu

übertragen sind. Hier wird der Frequenzbereich (300 bis 3400 Hz) transponiert (z.B. auf über 100 kHz). Ein anderes Beispiel sind Computersignale, die über das Telefonnetz übertragen werden.

Übertragungskanal. Der Übertragungskanal (vgl. Abbildung 2-1) dient als Bindeglied zwischen Sender und Empfänger (Quelle und Senke). Analoge Übertragungskanäle sind durch eine Frequenzlage mit einer oberen und unteren Grenzfrequenz charakterisiert. Signalverfälschungen können durch additive und multiplikative Effekte auftreten (Eigenrauschen, Störsignale, Verzerrungen). Digitale Übertragungskanäle sind gekennzeichnet durch eine Übertragungsrate in der Einheit Bit/s und einer Bitfehlerquote.

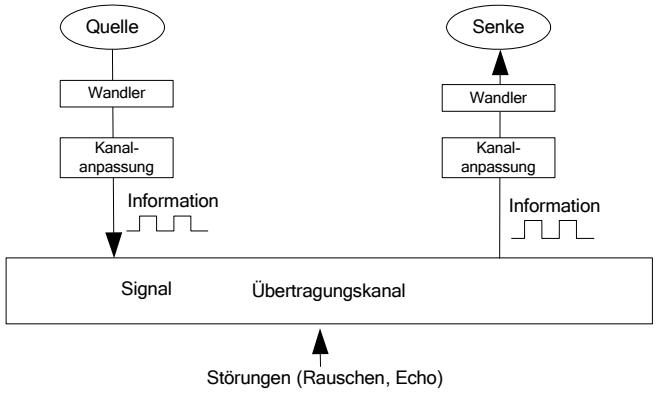


Abbildung 2-1: Grundstruktur eines Übertragungssystems

Man unterscheidet noch genauer zwischen Übertragungskanal und *Übertragungsmedium*. Übertragungsmedien, die gelegentlich auch physikalische Kanäle genannt werden, sind u.a. verdrillte Kupferleitungen, Koaxialkabel, Lichtwellenleiter (LWL), Infrarotstrecken, Laserstrahlen und Funkstrecken.

Schrittgeschwindigkeit, Wertigkeit, Bitrate, Kodiermethode (Meier 2002)**.** Die Zeit T, welche für die Übertragung von Information benötigt wird, hängt von der Kodiermethode und der Schrittgeschwindigkeit ab. Die Kodiermethode befasst sich mit der Abbildung eines bei der Quelle (beim Sender) eingegebenen Symbolvorrats M auf den hierfür auszugebenden Signalverlauf im Kanal. Die Anzahl der Symbole eines Codes heißt Wertigkeit von M (auch Anzahl der Signalstufen genannt) und wird mit Id(M) dargestellt.

Die Bitrate R (auch: Datenrate; Datenübertragungsrate oder Übertragungsgeschwindigkeit genannt) eines digitalen Signals wird in Bit/s angegeben. Um hier ein Gefühl für Größenordnungen zu vermitteln sollen einige Beispiele genannt werden:

- Zur Übertragung von Sprache in Telefonqualität benötigt man eine Bitrate von 64 Kbit/s.
- Für einen Stereo-Musik-Datenstrom benötigt man ohne Kompression eine Bitrate von ca. 1,4 Mbit/s (2 Kanäle mit je 44,1 kHz Abtastrate und 16 Bit Bandbreite)
- Für ein Farbvideo benötigt man 216 Mbit/s, mit Kompression kommt man mit 5-10 Mbit/s aus.
- Über einen Universal Serial Bus eines Personal Computers (USB3) können 5 Gbit/s übertragen werden.
- Über lokale Netzwerke z.B. auf Ethernet-Basis (siehe Kapitel 3) sind Bitraten im Gbit/s-Bereich machbar.

Die Bitrate ist der reziproke Wert (Kehrwert) für die Zeit, die benötigt wird, um ein Bit über einen Kanal zu versenden (auch als *Bitdauer* bezeichnet). Beträgt die Bitrate beispielsweise 10 Mbit/s, dann ergibt sich eine Bitdauer von 100 ns.

Die *Schrittgeschwindigkeit S* (Schrittrate oder Taktfrequenz) ist die Anzahl der Zustandsänderungen eines Signals pro Zeiteinheit und legt fest, wie oft sich der Wert des Signals ändert. Das Signal könnte hier eine Spannung sein und die Zeit in Sekunden angegeben werden. Man misst die Schrittgeschwindigkeit S in baud:

```
baud = bd = 1/s
```

Nur bei binären Signalen (0 oder 1) ist 1 baud = 1 Bit/s und M = 2, bei quaternärer Übertragung (M = 4) ist 1 baud = 2 Bit/s und bei einem achtwertigen Code ist 1 baud = 4 Bit/s und M = 8. Eine Leitung mit b Baud überträgt also nicht unbedingt mit einer Datenrate von b Bit/s, da jeder Signalwert nicht immer nur ein Bit überträgt, sondern evtl. sogar mehrere oder weniger Bits in einem Schritt übertragen werden können. Die Schrittgeschwindigkeit S darf also nicht mit der Bitrate R verwechselt werden. Es gilt:

```
R = S * ld(M) / 2 [Bit/s]
```

Hierbei ist zu beachten, dass die Einheit Bit/s ist und 1 Mbps = 1 Mbit/s = 1 Megabit/s = 1.000.000 Bit/s und nicht 2^{20} Bit/s ist.

Übertragungsstörungen. Da Übertragungsmedien nicht perfekt sind, kann es zu Übertragungsfehlern kommen. Mögliche Probleme bei der Übertragung können sein:

- Dämpfung: Hier ist der Energieverlust, der bei der Verbreitung eines Signals entsteht, gemeint. Man misst die Dämpfung in Dezibel pro Kilometer (db/km). Der Energieverlust ist abhängig von der Frequenz. Das Signal wird bei terrestrischen Medien logarithmisch mit der Entfernung schwächer.
- Laufzeitverzerrung: Hierunter versteht man das Überholen und damit Mischen von Bits.
- Rauschen: Dies ist die Beeinträchtigung der Übertragung durch uner-

wünschte Energie aus anderen Quellen wie etwa durch eng benachbarte Drähte (Nebensprechen durch induktive Kopplung).

Wie Abbildung 2-1 zeigt, wird ein Kanal durch Rauschen oder Echo usw. gestört. Das, was der Sender in den Kanal einstellt, soll natürlich beim Empfänger ankommen, und daher müssen Störungen möglichst erkannt und korrigiert werden.

Bandbreite. Die Bandbreite B eines Kanals ist in der Übertragungstechnik der Frequenzbereich, in dem ein System ohne übermäßige Dämpfung übertragen kann. Die Bandbreite wird in Hz gemessen. In der Datentechnik verwendet man den Begriff Bandbreite auch und meint damit die Gesamtkapazität des Netzwerks. Je mehr Information pro Zeiteinheit übertragen werden muss, desto mehr Bandbreite ist erforderlich.

Für einen rauschfreien Kanal gibt es nach Nynquist (1924) und Shannon (1948) eine maximale Bitrate mit eingeschränkter Bandbreite B. Das Nynquist-Theorem besagt, dass sich die Bitrate eines Kanals wie folgt ergibt:

$R < 2 * B * ld(M) / 2$ [Bit/s]

Beispiel: Betrachtet man z.B. eine Bandbreite von 3 kHz mit binärem Code M = 2, dann ergibt sich, dass ein rauschfreier Kanal mit dieser Bandbreite maximal 6000 Bit/s übertragen kann:

$R < 2 * 3000 * 1$ Bit/s → $R < 6000$ Bit/s.

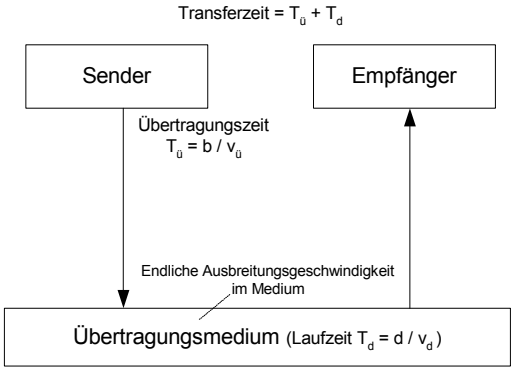


Abbildung 2-2: Laufzeit, Transferzeit und Übertragungszeit

Übertragungszeit, Laufzeit und Transferzeit. Interessant ist auch die Beziehung zwischen der Übertragungszeit, der Laufzeit und der Transferzeit eines Nachrichtenblocks (Abbildung 2-2).

Ein Block mit bestimmter Bitlänge b (in Bit) wird beim Sender in endlicher Geschwindigkeit $v_ü$ (in Bit/s) gesendet. Diese Zeit ist die Übertragungszeit $T_ü$. Sie gibt die Zeit an, die benötigt wird, bis der Bitblock an das Medium übergeben ist:

$T_ü = b / v_ü$ [s]

Der Bitblock kommt aufgrund der endlichen Signalgeschwindigkeit bzw. Ausbreitungsgeschwindigkeit v_d (gemessen in m/s) der Signale und einer endlichen Entfernung d (in m) erst nach einer bestimmten Laufzeit T_d beim Empfänger an:

$T_d = d / v_d$ [s]

Die Laufzeit ist abhängig vom Medium. Die Transferzeit T_t, also die gesamte Zeit vom Absenden des Bitblocks bis zur Ankunft beim Empfänger, ergibt sich als Summe von Übertragungszeit und Laufzeit:

$T_t = T_ü + T_d$ [s]

Bei niedrigen Übertragungsgeschwindigkeiten wird die Laufzeit gegenüber der Übertragungszeit vernachlässigbar, bei großen Entfernungen ist das allerdings nicht so. Die Laufzeit hängt vom Medium ab. Einige Medien und deren Ausbreitungsgeschwindigkeiten und Laufzeiten sind in Tabelle 2-1 dargestellt. Zum Vergleich: Schneller als die Lichtgeschwindigkeit, die ca. 300.000 km/s = 3 * 10^8 m/s beträgt, ist heute nicht möglich. Alle elektromagnetischen Strahlungen breiten sich im Vakuum mit Lichtgeschwindigkeit aus. In einem dichteren Medium wie Kupfer oder Glas sinkt die Geschwindigkeit

Medium	**~ Ausbreitungsgeschwindigkeit v_d in m/s**	**~ Laufzeit T_d in s/km**
Funkkanal	$3*10^8$ (näherungsweise)	$1/3*10^{-5}$ (0,00000333…)
Freiraum-Infrarot	$3*10^8$ (näherungsweise)	$1/3*10^{-5}$ (0,00000333…)
Glasfaserleitung (Quarzglas)	$2*10^8$	$5*10^{-6}$ (0,000005)
Basisband-Koaxialkabel (50/75 Ohm)	$2,3*10^8$	$4,3*10^{-6}$ (0,0000043…)
Zweidrahtleitung (verdrillt)	$2,5*10^8$	$4*10^{-6}$ (0,000004)

Tabelle 2-1: Typische Laufzeiten

Beispiel 1: Betrachten wir eine verdrillte Zweidrahtleitung mit einer Übertragungsgeschwindigkeit von 2.400 Bit/s und einer Ausbreitungsgeschwindigkeit von 2,5 * 10^8 m/s, über die ein Bitblock der Länge 100 Bit übertragen werden soll. Nehmen wir weiterhin an, dass die Leitung 100 km lang sei. Hier ergibt sich als Transferzeit:

$T_ü$ = 100 Bit / 2.400 Bit/s ~ 42 ms;

T_d = 100.000 m / (2,5 * 10^8 m/s) = 0,4 ms;

T_t = Tü + Td ~ 42 ms + 0,4 ms ~ 42,4 ms;

Hier ist also die Laufzeit im Vergleich zur Übertragungszeit vernachlässigbar.

Beispiel 2: Betrachten wir nun eine Funkverbindung über geostationäre Satelliten mit einer Übertragungsgeschwindigkeit von 2.400 Bit/s und einer Ausbreitungsgeschwindigkeit von 3 * 10^8 m/s. Zu übertragen ist ebenfalls ein Bitblock der Länge 100 Bit und die Leitung sei 36.000 km * 2 (hin und zurück) lang. Das Funksignal muss ja zum Satelliten und wieder zur Erde zurück. Dann ergibt sich als Transferzeit:

$T_ü$ = 100 Bit / 2.400 Bit/s ~ 42 ms;

$T_d = 72*10^6$ m / $3*10^8$ m/s) = $24*10^{-2}$ s (240ms);

$T_t = T_ü + T_d$ ~ 42 ms + 240 ms ~ 282 ms;

Bei diesem Beispiel ist es genau umgekehrt: Nicht die Laufzeit, sondern die Übertragungszeit ist vernachlässigbar.

Es sei nochmals angemerkt, dass die Bitrate nur die Zeit angibt, die benötigt wird, um die Daten in das Übertragungsmedium einzustellen. Je größer die Entfernung ist, desto weniger ist die Bitrate für die gesamte Transferzeit entscheidend.

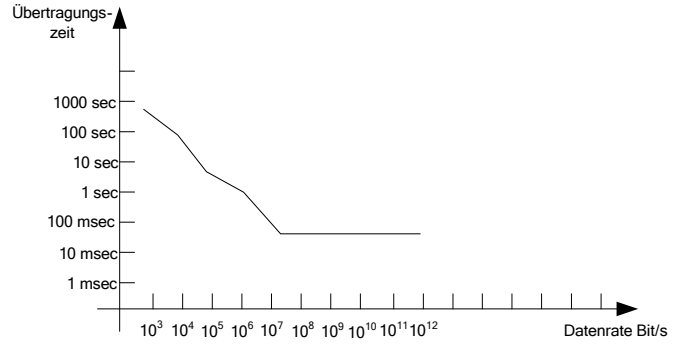


Abbildung 2-3: Übertragungszeit für eine 1-Mbyte-Datei bei einer Entfernung von 4.000 km nach (Zitterbart 1995)

Abbildung 2-3 zeigt am Beispiel der Übertragung einer 1-Mbyte-Datei über 4.000 km, dass sich die Übertragungszeit bei großen Distanzen auch durch deutliche Erhöhung der Bitrate nicht mehr wesentlich verbessern lässt. Man sieht, dass die Übertragungszeit bei Geschwindigkeiten von bis zu 1 Mbit/s von der Datenrate beherrscht wird, von da ab bestimmt vor allem die Rundreiseverzögerung von 40 ms die Übertragungszeit. Nur 1 ms wird bei 1 Gbit/s für das Einspeisen der Bits in die Leitung benötigt, der Rest wird für die Laufzeit benötigt.

Jitter und Verzögerungssensibilität. Eine insbesondere für die Multimedia-Datenübertragung wichtige Qualitätsgröße ist die Verzögerungsschwankung bzw. das Jitter (Flattern). Damit bezeichnet man die Schwankung in der Verzögerung nacheinander empfangener Datenpakete. Multimedia-Anwendungen (Audio, Vi-

deo) wie HDTV (High Definition TV) benötigen einen kontinuierlichen Datenfluss und damit eine minimale Verzögerungsschwankung. Man sagt, diese Anwendungen sind sensibel für Verzögerungsschwankungen. Deshalb ist dieser Parameter ein wichtiger Aspekt hinsichtlich der Dienstequalität derartiger Anwendungen. Weniger wichtig ist dies etwa bei klassischen Filetransfer-Anwendungen, da die Schwankungen hier nicht beim Anwender sichtbar werden.

2.1.3 Digitale Übertragung und Multiplexierung

Zur Übertragung eines Bitstroms gibt es grundsätzlich zwei Varianten, die analoge und die digitale Übertragung. Die analoge Übertragung funktioniert mit einer Menge an kodierten Werten aus einem kontinuierlichen und unendlichen Zeichenvorrat. Die digitalen Signale müssen bei analoger Übertragung auf ein Trägersignal aufmoduliert werden. Die digitale Übertragung hat gegenüber der analogen Übertragung einige Vorteile**.** Sie ist präziser, schneller und deutlich weniger störanfällig, und zudem zeichnet sie sich durch eine höhere Übertragungskapazität aus.

Modulationsarten. Als *Modulationsarten* sind Frequenzmodulation (z.B. eine Schwingung = 1, zwei Schwingungen = 0), Amplitudenmodulation (z.B. flache Schwingung = 0) und Phasenmodulation (z.B. vorgezogene Phase ist 1) möglich. Ein Beispiel für einen Standard zur analogen Übertragung ist die von der ITU-T genormte V24-Schnittstelle. Digitale Übertragung zeichnet sich durch einen endlichen Zeichenvorrat mit sprungartigem Übergang zwischen digitalen Zeichen aus. Beispielsysteme für die digitale Übertragung sind ISDN und ADSL.

In Bezug auf die Übertragungsgeschwindigkeit unterscheidet man zwischen *Schmalband-* und *Breitbandnetzen* und meint damit einfach zwei nicht klar voneinander abgegrenzte Geschwindigkeitsklassen. „Schmalbandige" Netze sind z.B. das heutige ISDN[1] (auch Schmalband-ISDN genannt) oder klassische LANs. Typische „breitbandige" Netze sind Gigabit-Ethernet und ATM-Netze. Während man bei Basisbandsystemen (Schmalbandsysteme) im Übertragungsmedium nur einen logischen Kanal nutzt, wird die Bandbreite in Breitbandsystemen auf mehrere Kanäle aufgeteilt, also eine Art Multiplexierung betrieben. Damit lässt sich insgesamt eine wesentlich höhere Datenrate bzw. Übertragungsgeschwindigkeit erzielen.

Multiplexierungsverfahren

Ein Übertragungskanal ist im Allgemeinen teuer. Wenn also die theoretische Grenze der Kanalkapazität noch nicht ausgeschöpft ist, kann man als Mittel der Kostensenkung für die Übertragung mehrere logische Verbindungen auf demselben geographischen Weg sog. *Multiplexierungsverfahren* (Multiplexing) einsetzen.

[1] ISDN = Integrated Service Digital Network: Dieses System wurde eingeführt, um Sprach- und Datenübertragung, also Telefonie und Rechnerkommunikation, über einen Netzzugang digital und integriert zu ermöglichen.

Hier bieten sich im Wesentlichen das *Frequenz- und Zeitmultiplexverfahren* (Frequency Division Multiplexing = *FDM* und Time Division Multiplexing = *TDM*) an (siehe hierzu Abbildung 2-4). Bei FDM wird das nutzbare Frequenzband des Übertragungsmediums in mehrere Teilbänder aufgeteilt. Bei TDM wird das Übertragungsmedium für bestimmte Zeitperioden einem Quellen-Senken-Paar zugeordnet. Die Übertragung erfolgt in Zeitperioden und daher ist eine Umschaltung erforderlich.

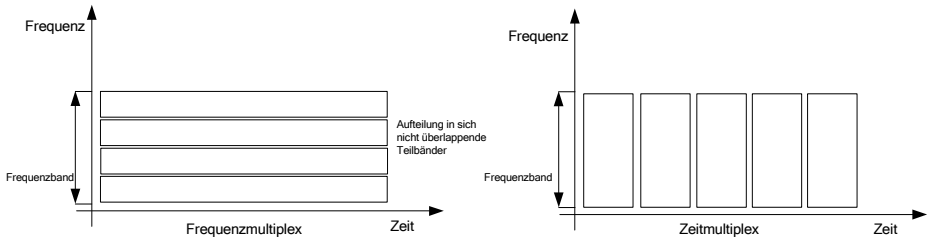


Abbildung 2-4: Frequenz- und Zeitmultiplexverfahren

Analoge Signale müssen zuerst digitalisiert werden, ehe sie in einem digitalen System weiterberarbeitet werden können. Ein analoges Signal ist zunächst zeit- und wertkontinuierlich und muss in ein zeit- und wertdiskretes Signal umgewandelt werden. Ein digitales Signal hat also ein diskretes Argument und einen diskreten Wertebereich und auch nur endlich viel Information.

Pulse Code Modulation (PCM)

Ein wichtiges TDM-basiertes Verfahren ist *PCM* (Pulse Code Modulation). PCM bildet die Grundlage für einige Kommunikationssysteme und ist der Standard für die digitale Kodierung von Audiodaten im Telefonnetz. Das Grundprinzip der A/D-Umwandlung (Analog-Digital-Umwandlung) wird im Folgenden beschrieben. Drei Bearbeitungsschritte sind im PCM-Verfahren für die Digitalisierung eines analogen Signals erforderlich:

- Abtastung
- Quantisierung
- Kodierung

Abtastung. Das Ziel der *Abtastung* ist zunächst, ein zeitkontinuierliches Signal in ein zeitdiskretes umzuwandeln. Nach Abbildung 2-5 wird das analoge Signal in konstanten Zeitabständen T abgetastet (sampling). Bei dieser Abtastung werden die Signalwerte übernommen und man erhält ein zeitdiskretes, aber noch wertkontinuierliches Signal. Dies ist noch kein digitales Signal. Die „Samples“ müssen also noch weiterverarbeitet werden, was bei der Quantisierung erfolgt.

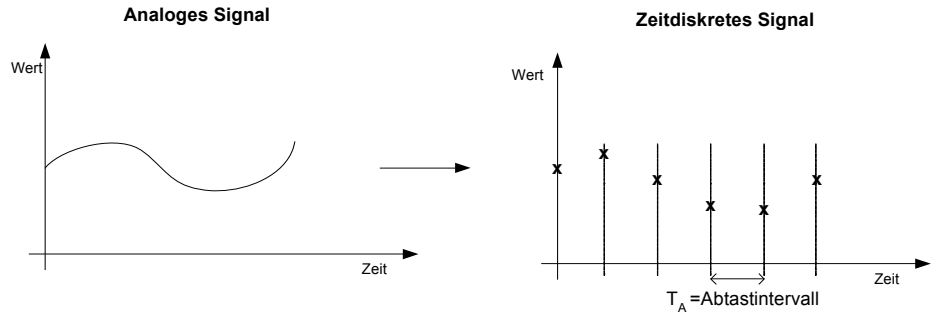


Abbildung 2-5: Abtastung eines analogen Signals

Beim *sampling* spielt als Grundidee das *Abtasttheorem von Shannon* eine bedeutende Rolle. Nach Shannon gilt, dass die Abtastfrequenz größer sein muss als die doppelte Bandbreite B des analogen Signals. Es muss also gelten:

$1 / T_A = f_A > 2 * B$

f_A ist die minimale Abtastfrequenz, T_A das Abtastintervall, und B ist die Bandbreite. Wird dies eingehalten, so steht beim Empfänger genügend Information zur Verfügung, um das Originalsignal fehlerfrei zu rekonstruieren. Musikinstrumente haben beispielsweise eine Bandbreite von etwa 20 kHz oder sogar mehr, was aber dann das menschliche Ohr gar nicht mehr hört. Bei CDs arbeitet man mit einer Abtastfrequenz von 44.1 kHz, also mehr als das Doppelte der Grenzfrequenz, und damit wird das Abtasttheorem mehr als erfüllt (Meyer 2002).

Quantisierung. Die *Quantisierung* hat das Ziel, aus einem wertkontinuierlichen Signal ein wertdiskretes Signal zu erzeugen. Dies geschieht im Prinzip durch Runden. Dazu wird der Wertebereich des analogen Signals in eine endliche Anzahl von Quantisierungsintervallen gegliedert. Jedes Intervall wird mit einem diskreten Wert belegt. Allen analogen Signalen des Intervalls wird derselbe diskrete Wert zugewiesen. Hierbei sind Quantisierungsfehler (Signalverfälschungen) wie Rauschen oder Quantisierungsrauschen möglich, die sich auf max. ½ des Intervalls belaufen und einen Informationsverlust als Preis für die Vorteile der Digitaltechnik zur Folge haben. Je kleiner das Quantisierungsintervall ist, umso geringer sind die Quantisierungsfehler.

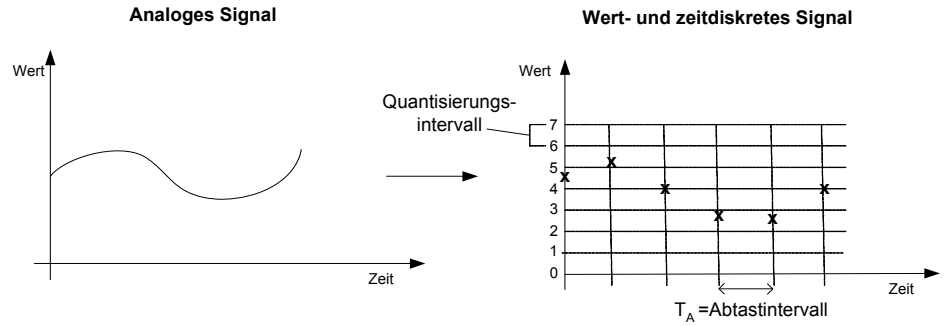


Abbildung 2-6: Quantisierung eines analogen Signals

In Abbildung 2-6 stellen die horizontalen Linien die möglichen ganzzahligen Werte dar. Ein A/D-Wandler wählt jeweils den nächstliegenden Wert als Code aus, was im Beispiel die Wertefolge „5 5 4 3 3 4" ergibt. In der PCM-Technik gibt es 256 Quantisierungsintervalle, in Abbildung 2-6 sind acht Quantisierungsintervalle dargestellt. Nach der Quantisierung ist ein Signal digital, also wert- und zeitdiskret.

Kodierung. Die *Kodierung* hat schließlich noch die Aufgabe, den Quantisierungsintervallen Codes zuzuordnen. Die PCM-Technik benötigt 8 Bit für die Kodierung, da 256 Quantisierungsintervalle kodiert werden müssen. Bei acht Quantisierungsintervallen, wie in der Abbildung skizziert, benötigt man drei Bit zur Kodierung.

Ein Haupteinsatzgebiet von PCM ist die Telefonie. Ein analoger Fernsprechkanal hat eine Bandbreite von 3100 Hz zwischen 300 Hz und 3400 Hz, also:

$f_A > 2 * 3100\ Hz \rightarrow f_A > 6200\ Hz$

Technisch wurde eine erhöhte Abtastfrequenz von 8000 Hz realisiert. Als Abtastintervall (Abtastperiode T_A) ergibt sich dann

$T_A = 1 / f_A = 125\ \mu s.$

Alle 125 µs wird das analoge Fernsprechsignal abgetastet, um es zu digitalisieren. Ein digitaler Sprachkanal hat dann folgende Bitrate:

$f_A * 8\ Bit = 8000\ Hz * 8\ Bit = 64\ Kbit/s$ (Anm: Hz = 1/s)

Die PCM-Technik ist heute insbesondere für die sog. dienstintegrierenden Netze für die gleichzeitige Übertragung von Daten, Bewegtbildern und Sprache von Bedeutung und wird z.B. bei ISDN eingesetzt. Ein ISDN-B-Kanal hat eine Bitrate von 64 Kbit/s und repräsentiert einen digitalen Sprachkanal.

Auf Basis der Grundeinheit von 64 Kbit/s wurde eine sog. Übertragungshierarchie definiert, in der jeweils mehrere Kanäle multiplexiert werden können (vgl. Tabelle 2-2). In PCM-30 sind dann z.B.

30 * 64 Kbit/s + 2 * 64 Kbit/s (Steuerinformation) = 2,048 Mbit/s

verfügbar. Entsprechend gibt es PCM-120 mit 8,448 MBit/s usw.

Tabelle 2-2: PCM-Übertragungshierarchie

PCM-Hierarchie	Übertragungsrate
PCM-30	2,048 Mbit/s
PCM-120	8,448 Mbit/s
PCM-480	34,368 Mbit/s
PCM-1920	139,264 Mbit/s
PCM-7680	564,992 Mbit/s

2.1.4 Quellen-, Kanal- und Leitungskodierung

Bevor Daten über einen Kanal übertragen werden, werden sie auf mehrfache Weise kodiert. Wie in Abbildung 2-7 zu sehen ist, unterscheidet man Quellenkodierung, Kanalkodierung und Leitungskodierung. Im Folgenden soll ein Überblick über diese Kodierungsverfahren gegeben werden.

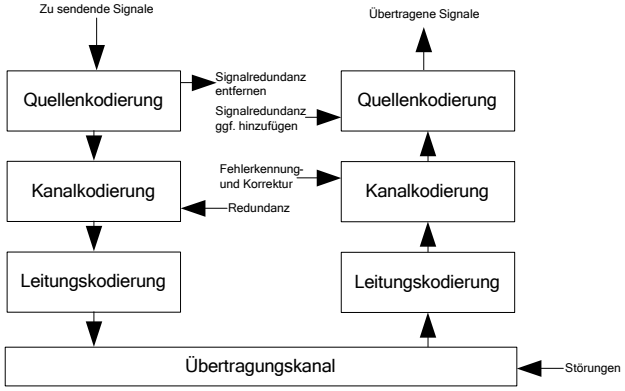


Abbildung 2-7: Kodierung von Nachrichten

Quellenkodierung

Die Quellenkodierung hat zum Ziel, Daten so zu kodieren, dass sie möglichst wenig Platz benötigen und dadurch im Sinne der Datenkommunikation auch mit möglichst geringer Übertragungsrate versendet werden können. Die Quellenkodierung sorgt aber auch für eine platzsparende Speicherung von Daten (z.B. Audio- und Video-Daten). Man spricht bei der Quellenkodierung auch von *Datenkomprimierung*.

Bei der Quellenkodierung können unwichtige Daten eliminiert werden. Dies führt zu einer verlustbehafteten Kodierung. Andere Verfahren dienen der verlustfreien Kodierung. Die Entropiekodierung ist z.B. eine verlustfreie Kodierung, bei der jedem Zeichen des Quellcodes unterschiedlich lange Bitfolgen zugeordnet werden.[2]

Kanalkodierung

Unter Kanalkodierung versteht man Verfahren, mit denen digitale Daten vor der Übertragung durch das Ergänzen von Redundanz gegen Übertragungsfehler geschützt werden. Die Redundanz wird am Eingang eines Übertragungskanals ergänzt und am Ausgang wieder entfernt.

Die Daten werden hierzu als Codewörter kodiert, wobei fehlererkennende (error detection) und fehlerkorrigierende (error correction) Codes unterschieden werden. Bei digitalen Standleitungen kann heute z.B. von einer sehr geringen Bitfehlerwahrscheinlichkeit von ca. 10^{-12} ausgegangen werden. Insbesondere bei drahtloser Übertragung ist aber die Bitfehlerwahrscheinlichkeit noch wesentlich höher. Zu den übertragenen Nettodaten wird vom Sender redundante Information ergänzt. Der Empfänger kann diese dann auswerten und ggf. Übertragungsfehler feststellen und korrigieren. Das Hinzufügen von Redundanz erfordert eine höhere Bitrate für die Datenübertragung. Daher ist abzuwägen, wieviel Redundanz notwendig ist, um für einen Übertragungskanal eine geeignete Fehlerkorrekturfähigkeit zu erreichen.

Klassische Verfahren zur Kanalkodierung sind der Einsatz von *Paritätsbits* und *zyklische Redundanzcodes*.

Einsatz von Paritätsbits. Unter *Parität* einer Bitfolge versteht man die Anzahl der Einsen in der Bitfolge. Setzt man z.B. auf eine gerade Parität, so wird ein den Nutzdaten zu ergänzendes Paritätsbit auf binär „1“ gesetzt, wenn die Anzahl der Einsen in der Nutzdaten ungerade ist. Die Berechnung ist sehr einfach, weshalb Paritätsbits häufig eingesetzt werden. Man berechnet z.B. ein Paritätsbit pro 8-Bit-Zeichen und ergänzt dies in der Nachricht.

Es hängt von der sog. *Hamming-Distanz* ab, wie viele Fehler durch Paritätsprüfung erkannt bzw. sogar korrigiert werden können. Die Distanz zweier Codewörter ist die Anzahl der Bitpositionen, in denen sich die beiden Codewörter unterscheiden. Die Hamming-Distanz eines Codes ist dann der Minimalwert der Distanzen zwischen allen möglichen Paaren von Codewörtern. Bei einer Hamming-Distanz von d können $d - 1$ Fehler erkannt und $t < (d - 1)/2$ Fehler korrigiert werden.[3] Mit einer

---

[2] Weitere Informationen zur Kompression von Audio- und Video-Daten sind in Kapitel 6 zu finden.

[3] Näheres hierzu findet man in (Sikora 2003). Hier finden sich auch weitere Wahrscheinlichkeitsangaben.

1-Bit-Paritätsprüfung kann zwar keine Fehlerkorrektur unterstützt werden, aber man erkennt einen Fehler bei einer ungeraden Anzahl von Bits in einer Zeichenfolge. Man erkennt dagegen nicht, wenn es sich um eine gerade Anzahl an Bitfehlern handelt.

Zyklische Redundanzcodes. *Zyklische Redundanzcodes* (auch CRC = Cyclic Redundancy Check) werden ebenfalls sehr häufig zur Fehlererkennung eingesetzt, und zwar insbesondere bei längeren, zu übertragenden Datenblöcken. Bekannte Einsatzbereiche sind die heutige USB-Schnittstelle[4] in PCs und Ethernet-Netzwerke (siehe Schicht 2). Hier werden die Bits der Nutzinformation von links nach rechts als Koeffizienten eines Polynoms aufgefasst. Dieses Polynom wird durch ein sog. Generatorpolynom vom Grad k dividiert. Dabei entsteht ein Restpolynom vom Grad k – 1. Dieser Divisionsrest (bzw. die Koeffizienten des Restpolynoms) stellt dann die Prüfsumme dar, die an die Nutzdaten angehängt wird. Der Empfänger teilt die empfangenen Daten (Nutzdatenbits und den Rest) ebenfalls durch das Generatorpolynom. Bei Fehlerfreiheit muss sich ein Rest von 0 ergeben.

Nutzt man CRC bei zeichenorientierten Protokollen, so betrachtet man die Bits der aufeinanderfolgenden Codewörter als Koeffizienten. Bei bitorientierten Protokollen werden die Datenbits als Koeffizienten des Poynoms verwendet. Ein zyklischer Code mit einem 16-Bit-CRC-Feld kann alle Fehler erkennen, bei denen bis zu 16 Bit in Folge fehlerhaft übertragen wurden, bei einem 32 Bit langen CRC-Feld kann man schon fehlerhafte Folgen bis zu 32 Bit erkennen. Fehler mit mehr als 32 Bit können ebenfalls mit sehr hoher Wahrscheinlichkeit erkannt werden (mit mehr als 99,99 % Wahrscheinlichkeit).

Leitungskodierung

Konvertiert man ein analoges Signal auf ein digitales oder umgekehrt, benötigt man Kodierungsverfahren. Man spricht hier auch von *Leitungskodierung*. Die Leitungskodierung legt bei der digitalen Datenübertragung fest, wie ein Signal physikalisch übertragen wird. Ein zu übertragendes Signal soll dabei möglichst optimal an das Übertragungsmedium angepasst werden. Es gibt verschiedene Verfahren zur Leitungskodierung, von denen drei im Folgenden kurz skizziert werden sollen.

Binäre Leitungscodes. Bekannt ist u.a. die reine *Bitkodierung*, für die man den logischen Zuständen 0 und 1 auf der physikalischen Leitung auch nur zwei verschiedene Zustände, wie z.B. Spannungen, zuordnet. Eine logische 1 kann als positive Spannung und eine logische 0 als negative Spannung dargestellt werden. Die reine Bitkodierung hat den Nachteil, dass bei langen Sequenzen der Takt verloren geht. Dieser unterliegt nämlich immer kleinen Schwankungen. Der Taktverlust hat zur Folge, dass es zu Missinterpretationen auf Empfängerseite kommen kann.

---

[4] USB steht für Universal Serial Bus und ist ein PC-Standard für den Anschluss von seriellen Geräten an den PC. Derartige Busse finden sich heue in jedem modernen PC.

Manchester-Kodierung. Ein anderes Verfahren wird als *Manchester*-Kodierung bezeichnet. Die Manchester-Kodierung (auch *Bi-Phase-Mark*) und andere Verfahren wie das *NRZI*- (Non Return to Zero Invert) oder das *MLT*-Verfahren (Multi Level Transmission) versuchen, den Takt beim Empfänger aus der Bitdarstellung wieder zurückzugewinnen. Bei der Manchester-Kodierung interpretiert man z.B. eine 1, wenn das Signal von hoch nach niedrig geht (fallende Flanke) und eine 0 im umgekehrten Fall (steigende Flanke). In Abbildung 2-8 ist stellvertretend die Manchester-Kodierung im Vergleich zur Binärkodierung dargestellt. Bei der Manchester-Kodierung wird die erforderliche Bitrate verdoppelt und die Schrittgeschwinfigkeit halbiert. Eine weitere Variante stellt die differenzierte Manchester-Kodierung dar, die hier nicht weiter erläutert werden soll.

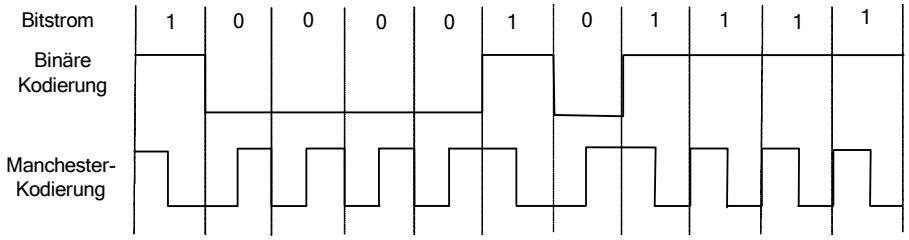


Abbildung 2-8: Manchester-Kodierung im Vergleich zur Binärkodierung

Block-Kodierung. Bei der *Block-Kodierung* werden *p* Bits eines Binärwortes zu einem Block mit der Bitlänge *q* zusammengefasst. Man bezeichnet diese Codes auch mit *pBqB*.

Ein typischer Blockcode, der u.a. auch bei verschiedenen Ethernet-Systemen (siehe Schicht 2) im Bereich der 100 Mbit/s-Ebene Einsatz findet, ist der *4B5B-Code*. Hier werden aus 4-Bit-Gruppen 5-Bit-Wörter kodiert. Dies ergibt im Gegensatz zur Manchester-Kodierung, bei der doppelt so viele Bit übertragen werden als man für die Information benötigt, nur eine Redundanz von 20 %. Der Manchester-Code kann auch als *1B2B-Code* bezeichnet werden. In Gigabit-Ethernets nutzt einen *8B10B*-Code.

2.1.5 Datenübertragungsmedien und Verkabelung

Ohne zu sehr auf die physikalischen Gegebenheiten einzugehen, sollen in diesem Abschnitt die wesentlichen Übertragungsmedien zusammengefasst, sowie typische Einsätze kurz diskutiert werden. In Tabelle 2-3 sind typische Übertragungsmedien beschrieben und dabei auch die möglichen Kanalkapazitäten angegeben.

Twisted Pair. *Twisted Pair* ist eine generelle Bezeichnung für Kupferkabel mit einem oder mehreren verdrillten Leitungspaaren. Twisted-Pair ist der älteste Kabeltyp, wird aber immer noch am meisten eingesetzt. Meistens nutzt man zwei verdrillte Paare, also vier Adern. Ein Paar wird zum Senden und das andere zum

Empfangen verwendet. Durch die „Verschlingung“ der Adern erreicht man eine Verringerung der Ein- und Ausstrahlung, da sich die Magnetfelder der beiden stromduchflossenen Leiter wechselseitig neutralisieren.

Heute unterscheidet man verschiedene Arten von verdrillten Kupferkabeln, die sich hauptsächlich hinsichtlich der Abschirmung unterscheiden. Man ordnet die Kabelarten auch nach Qualität in die Kategorien 1 bis 7, wobei die Kategorie 7 die besten Dämpfungswerte bei einer hohen Frequenz von 600 MHz besitzt. Bezieht man auch die gesamte Übertragungsstrecke mit in die Betrachtung ein, so unterscheidet der entsprechende Standard der ISO (EIA/TIA 568A) nochmals sog. Anwendungsklassen A bis F, wobei F Bandbreiten bis zu 750 MHz unterstützt.

Tabelle 2-3: Datenübertragungsmedien

Medium	**Aufbau**	**Kanalkapazität**
Twisted Pair	Verdrilltes 8-adriges Kupferkabelpaar (Telefonkabel)	1 Gbit/s
Koaxialkabel	Vom zylindrischen Leiter umschlossener Kupferdrahtkern	ca. 10 Mbit/s
Glasfaserkabel	Vom zylindrischen Glasmantel umschlossener Glaskern	mehrere Gbit/s
Mobiler Datenfunk	Terrestrischer Zellularfunk (z.B.: Funk-LAN)	einige Mbit/s (neuere WLAN-Technologie bis zu 54 Mbit/s und mehr)
Richtfunk	Terrestrisch und ortsfest	Bis zu 4 Mbit/s
Satellitenfunk	Geostationäre Nachrichtensatelliten	56 Kbit/s

In Abbildung 2-9 sind die verschiedenen Bauformen von Twisted-Pair-Kabeln dargestellt. Der einfachste Typ wird als UTP bezeichnet, der am besten abgeschirmte als S/STP. Bei S/STP sind sowohl die einzelnen Adernpaare abgeschirmt (Shield), als auch alle Adernpaare gemeinsam (Screen).

Koaxialkabel. Das *Koaxialkabe*l hat einen Innenleiter aus Kupfer und ist mit einer Isolierschicht, einem rohrförmigen Außenleiter zur Abschirmung und einer Außenisolierung umgeben. Koaxialkabel können bis zu einer Bandbreite von 2 GHz genutzt werden. Sie sind recht schwierig zu verlegen, verfügen aber über eine einfache Anschlusstechnik.

Glasfaserkabel. *Glasfaserkabel* (auch optische Kabel, Lichtwellenleiter oder LWL = Optical Fibre Cable) sind aus Glas oder Kunststofffasern hergestellt. Da Glas be-

kanntlich wiederum aus Sand hergestellt wird, steht für die Herstellung ein günstiges Rohmaterial zur Verfügung. Die Datenübertragung erfolgt über sehr kurze Laserlichtimpulse, die im Nanosekundenbereich gesendet werden, mit einer hohen Impulsrate, die bis zu 1 THz reicht.

Richtfunk. Bei *Richtfunk* wird drahtlos über elektromagnetische Wellen übertragen, die von einem Sender direkt und gebündelt und mit relativ kleiner Leistung auf eine entsprechende Antenne des Empfängers *gerichtet* sind. Man benötigt hierzu auf der Sender- und der Empfängerseite sog. Richtfunkantennen. Richtfunkantennen können z.B. auf Dächern höherer Häuser platziert werden. Bei der *Rundfunkübertragung* werden ebenfalls elektromagnetische Wellen zur Datenübertragung verwendet. Die Radiowellen werden gleichmäßig z.B. von Satelliten ausgestrahlt. Die Reichweite hängt u.a. von der Sendeleistung und der Empfindlichkeit der Empfänger ab. Die drahtlose Übertragung, die ja immer interessanter wird, ist in der Literatur ausführlich beschrieben. Die Arbeitsweise und der physikalische Aufbau der einzelnen Kabelarten ist ebenfalls der Literatur zu entnehmen und soll hier nicht weiter vertieft werden.

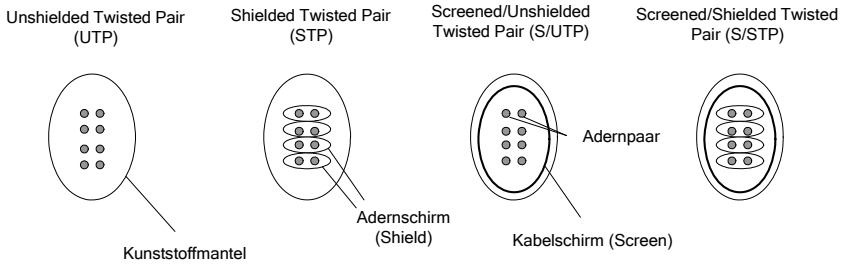


Abbildung 2-9: Twisted-Pair-Kabel: Bauformen

Die letzte Meile. Interessant und teuer bei der Vernetzung ist die *letzte Meile*, auch *local loop* oder *last mile* genannt. Dies ist der Anschluss der einzelnen Gebäude an ein Netzwerk einer Telefongesellschaft. Man spricht hier auch von Zugangsnetzen. In den letzten Jahrzehnten wurden die letzten Meilen vorwiegend mit Kupferleitungen erschlossen, während Weitdistanzübertragungen zwischen Vermittlungsrechnern der Telefongesellschaften über Multiplexverbindungen, in letzter Zeit immer mehr mit Glasfasern realisiert wurden. Die Kupferleitungen waren ursprünglich für Basisband-Sprachübertragungen über Distanzen bis zu acht Kilometer und nicht für die digitale Übertragung vorgesehen. Folglich mussten sich die Telefon-Gesellschaften überlegen, wie sie die Bandbreiten bis zum Gebäude verbessern konnten.

Zunächst wurden analoge Modems verwendet, mit denen man eine Bitrate von 56 Kbit/s erreichte. Später wurde ISDN entwickelt, und man konnte damit über das

bestehende Kupferkabel bereits standardmäßig 144 Kbit/s erreichen, bei Bedarf sogar 2 Mbit/s über den sog. ISDN-Primärmultiplexkanal.[5] Auch bei ADSL[6] werden die vorhandenen Kupferkabel nach einem speziellen Modulationsverfahren ausgenutzt. Über die Fernsehverkabelung schafften es die Fernsehgesellschaften[7], Koaxialkabel bis ins Haus zu verlegen und auch über die Stromversorgung (Starkstromverdrahtung) lassen sich einige Mbit/s erreichen. LWL sind heute für die letzte Meile (FTTH = Fiber To The Home) in der Regel immer noch zu teuer.

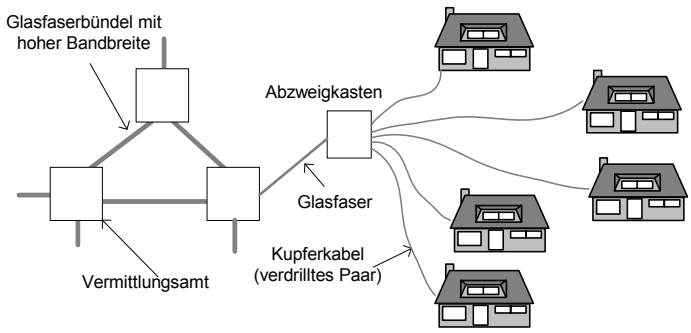


Abbildung 2-10: Typischer Einsatz von Kabelarten

In Abbildung 2-10 ist eine typische Verkabelung dargestellt, wie sie Telefongesellschaften durchführen. Die Vermittlungsknoten der Telefon-Gesellschaften werden heute über Glasfaserbündel vernetzt. In einem Ort oder einem Stadtteil befindet sich ein Abzweigkasten, von dem aus dann die einzelnen Häuser (die letzte Meile) über verdrillte Kupferkabel angeschlossen sind. Da man es hier oft nur mit einigen Hundert Metern zu tun hat, kann man über die Kupferkabel auch eine relativ hohe Bitrate erreichen. Man sieht also, dass eine komplette Vernetzung über Glasfaser bis hin zu den einzelnen Häusern heute noch nicht realisiert ist. Der Grund ist in den hohen Kosten zu sehen, die durch eine vollständige Glasfaservernetzung verursacht würden. Zudem gibt es ja schon eine hohe Abdeckung durch Kupferkabel.

Strukturierte Verkabelung. Im Gebäude vernetzt man heute nicht mehr über einen physikalischen Bus mit Koaxialkabel wie dies früher bei einem LAN oft noch der Fall war, sondern über ein strukturiertes und vereinheitlichtes Verkabelungs-

[5] ISDN-Primärmultiplexkanäle verfügen über eine Bitrate von 30 * 64 Kbit/s (B-Kanäle) + 1 * 64 Kbit/s Steuerkanal (D-Kanal) und ein weiterer für interne Zwecke, siehe Kapitel 3.

[6] Allgemeine Bezeichnung für die effiziente digitale Ausnutzung der Kupferverkabelung der letzten Meile ist xDSL, wobei x ein Platzhalter für „S“ = symmetrisch, „A“ = asymmetrisch usw. ist (Meyer 2002), siehe Kapitel 3.

[7] Das angewendete Verfahren heißt HFC (Hybrid Fiber Coaxial Cable), vgl. (Kurose 2002).

systems, wie es im Standard ISO/UEC -11801 beschrieben ist. Man spricht hier von *strukturierter Verkabelung* und möchte damit u.a. erreichen, dass möglichst wenig unterschiedliche Übertragungsmedien die Übertragung möglichst vieler Anwendungen erlauben und unterschiedliche LAN-Technologien durch einen einheitlichen Kabeltyp unterstützt werden.

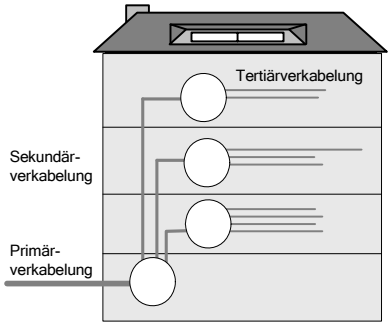


Abbildung 2-11: Strukturierte Verkabelung von Gebäuden

Man unterteilt bei der strukturierten Verkabelung die Gebäudeverkabelung, wie in Abbildung 2-11 dargestellt, in einen Primär-, einen Sekundär- und einen Tertiärbereich:

- Der Primärbereich ist der gebäudeübergreifende Bereich, der möglichst redundant auf Basis von Lichtwellenleitern verkabelt wird.
- Im Sekundärbereich installiert man ein Gebäude-internes Backbone auf Basis von Kupferkabeln oder Lichtwellenleitern.
- Den Tertiärbereich bildet eine *sternförmige* Verkabelung auf den einzelnen Etagen. Die Endgeräte werden mit Etagenverteilern verbunden.

Vorteil dieser Punkt-zu-Punkt-Vernetzung im Tertiärbereich ist, dass bei Ausfall eines Kabels, im Gegensatz zu einem Bussystem mit Koaxialkabel, nicht das ganze Netz lahmgelegt wird. Weiterhin verwendet man Verteiler mit Patchfeldern, mit denen man sehr flexibel und ohne weitere Verkabelung die einzelnen Steckdosen für die Endgeräte belegen kann.

Abschließend soll noch erwähnt werden, dass auch die Datenübertragung über Stromleitungen möglich ist. Man spricht hier von *Powerline Communication*. Bei diesem Verfahren werden die höheren Frequenzbereiche der Stromleitungen für die Übertragung verwendet. Heute sind z.B. bereits Geräte im Handel erhältlich, die über herkömmliche Stromsteckdosen LAN-Verbindungen ermöglichen. Dies ist also eine Variante, die verwendet werden kann, wenn eine andere Verkabelungsart oder eine Funkausleuchtung nur schwer möglich ist.

2.2 Sicherungsschicht

2.2.1 Aufgaben und Einordnung

Die Sicherungsschicht (auch Data Link Layer) stellt an einem DL-SAP (siehe Abbildung 2-12) einen Dienst zur Verfügung, der je nach Schicht-2-Implementierung etwas anders aussieht. Grundsätzlich wird eine zuverlässige Bitübertragung über die Datenverbindung zwischen zwei Rechnersystemen bereitgestellt. Mit „gesichert" meint man hier, dass überprüft wird, ob die Daten auch beim Empfänger angekommen sind und Übertragungsfehler erkannt und evtl. behoben werden (*Fehlersicherung*). Man spricht auch von einer gesicherten Ende-zu-Ende-Verbindung zwischen zwei Rechnersystemen.

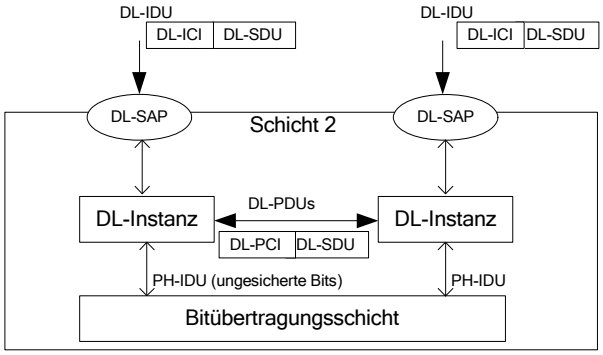


Abbildung 2-12: DL-Instanz

Für die Aufgabe der Fehlererkennung und evtl. Behebung werden sog. Error-Detection-Codes, also Prüfsummen, wie etwa Cyclic-Redundancy-Codes (CRC)[8], mitgesendet. Hier handelt es sich im Allgemeinen um redundante Information, die es ermöglicht, gewisse Fehler zu erkennen und bei ausreichender Redundanz sogar automatisch zu beheben. In der Praxis wird allerdings aus Aufwandsgründen selten eine Korrektur vorgenommen, ein erneutes Senden ist hier effizienter.

Die übertragenen Bits werden in logische Blöcke zusammengefasst. Zwischen Sender und Empfänger bzw. den zugeordneten DL-Instanzen erfolgt eine Blocksynchronisation. Aus dem Bitstrom werden also Nachrichtenbestandteile erkannt. Um eine Nachricht von einer folgenden abzutrennen, ist eine *Rahmung* („Framing") notwendig. Framing kann auf Zeichenebene erfolgen, indem man spezielle Start- und Stop-Zeichen definiert (z.B. die ASCII-Zeichen STX und ETX) und diese am Anfang bzw. am Ende der Nachricht sendet. Kommen diese Zeichen aber in den Nutzdaten vor, müssen sie über ein weiteres Zeichen „neutralisiert" werden, sonst

[8] Dahinter stecken Polynomial-Verfahren, die in (Tanenbaum 2003a) und (Kurose 2002) nachgelesen werden können.

geht die Synchronisation verloren. Hier müssen sich Sender und Empfänger einig sein, welches Zeichen dafür verwendet wird, wobei häufig das ASCII-Zeichen DLE verwendet wird. Diesen Mechanismus nennt man Character-Stuffing (Stopfen).

Daten, die ein Sender abschickt, dürfen nicht verloren gehen. Hier kann man in der Sicherungsschicht ebenfalls Maßnahmen ergreifen, wobei folgende *Protokollmechanismen* in Kombination angewendet werden:

- Bestätigungen senden
- Zeitüberwachungen (Timerüberwachung) für Bestätigungen durchführen
- Sendungswiederholung durchführen
- Erkennen von Duplikaten über Sequenzzähler und Verwerfen dieser

Es gibt unterschiedliche Sicherungsprotokolle. Man spricht hier von verbindungsorientierten und verbindungslosen Protokollen und meint damit, dass entweder vor dem Nutzdatenaustausch eine Verbindung aufgebaut werden muss oder nicht. Weiterhin gibt es Protokolle mit und ohne Bestätigung. Unbestätigte Protokolle übertragen Daten, ohne eine entsprechende DL-PDU ACK(knowlegde) vom Empfänger zu erhalten. Bei bestätigten Protokollen wird der Sender über das Ankommen einer DL-PDU vom Empfänger mit einer DL-PDU vom Typ ACK informiert.

Eine weitere Aufgabe der Sicherungsschicht ist die *Flusskontrolle* zur Vermeidung von Rückstaus und daraus resultierenden Datenverlusten. Eine empfangende DL-Instanz muss ja nicht immer genau so schnell empfangen können wie die sendende DL-Instanz überträgt. Die Folge wäre eine Überflutung der empfangenden DL-Instanz, was vermieden werden muss.

XON/XOFF-Protokoll. Ein klassischen Protokoll zur Flusssteuerung ist das in Abbildung 2-13 dargestellte *XON/XOFF-Protokoll*. Hier verwaltet jede DL-Instanz einen Empfangspuffer (vgl. Abbildung 2-14), in dem er empfangene Nachrichten einstellt. Wird der Puffer zu voll, ist also z.B. eine achtzig-prozentige Befüllung erreicht, so sendet die DL-Instanz auf Empfängerseite das Zeichen XOFF (ASCII-Zeichen). Die sendende DL-Instanz beendet das Senden, bis es ein XON empfängt. XON signalisiert dem Sender, dass der Empfänger wieder Platz in seinem Puffer hat. Die Puffergröße und auch der Befüllungsgrad, ab dem ein XOFF bzw. wieder ein XON gesendet wird, müssen natürlich gut abgestimmt sein.

Eine andere Möglichkeit zur Flusskontrolle ist der *Sliding-Window-Mechanismus*, in dem Sender und Empfänger darüber Buch führen, wie viele Nachrichten gesendet werden dürfen, bevor der Sender ausgebremst wird. Der Empfänger bestätigt ankommende Nachrichten oder Bytes über eine ACK-PDU und sendet zusätzlich die Information, wie groß das „Sendefenster“ für eine bestimmte Verbindung noch ist.

Weiterhin wird in Schicht 2 geregelt, wer als nächstes auf den Kanal zugreift (*Kanalsteuerung*), also welcher Kommunikationspartner wann das Übertragungsmedium nutzen darf.

Unter Flusskontrolle versteht man also die Sicherstellung, dass ein überlasteter Empfänger nicht von einem Sender mit Nachrichten überschwemmt wird, die er nicht verarbeiten kann. Durch geeignete Flusskontrollmechanismen kann man den Sender ausbremsen, wofür es mehrere Möglichkeiten gibt.

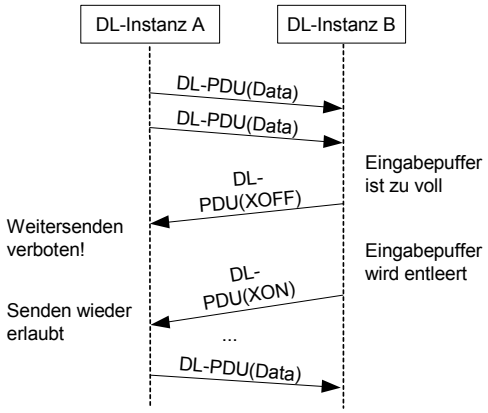


Abbildung 2-13: XON/XOFF-Protokoll

Ein typisches (in die Jahre gekommenes) Protokoll der Schicht 2 ist das zeichenorientierte *BSC*-Protokoll (Binary Synchronous Communication Protocol), das ursprünglich von IBM entwickelt wurde. „Zeichenorientiert" bedeutet hier, dass das Protokoll zeichenweise sendet, wobei die Kodierung nach dem ASCII-Zeichensatz erfolgt und die Transparenz der Daten durch DLE-Zeichen erreicht wird. Character-Stuffing ist also erforderlich.

Ein anderes Protokoll der Schicht 2 ist das bitorientierte *HDLC*-Protokoll (High Level Data Link Control Protocol), das ebenfalls von IBM entwickelt wurde und leistungsfähiger als BSC ist. *HDLC* wird weiter unten noch näher erläuert.

Da die Komplexität der Schicht 2 vor allem in Local Area Networks beträchtlich ist, teilt man diese funktional nochmals in zwei Sublayers, die MAC- und LLC-Schicht auf. Die MAC-Layer (Medium Access Control) oder 2a-Schicht regelt den Zugriff auf einen Mehrfachzugriffskanal. Die LLC-Layer (Logical Link Control) oder 2b-Schicht verwaltet die Verbindungen zwischen den DL-Instanzen.

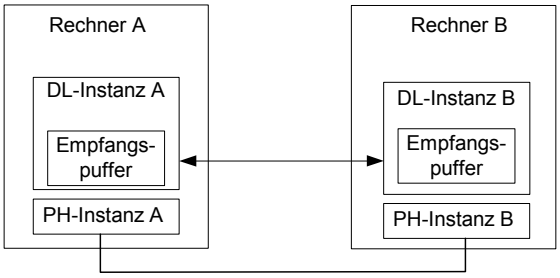


Abbildung 2-14: Empfangspuffer in der DL-Instanz

Einige der Mechanismen aus der Schicht 2, wie das Sliding-Window-Verfahren, finden auch in der Transportschicht Anwendung, gelten dort aber für eine Ende-zu-Ende-Kommunikation, die evtl. über mehrere Rechner geht und nicht wie in Schicht 2 „nur" zwischen zwei Rechnersystemen abgewickelt wird.

2.2.2 Topologien

Netzwerke können unterschiedlich strukturiert sein, und man unterscheidet hinsichtlich der Topologie vor allem Busnetze, Sternnetze, Baumnetze, Ringnetze und vermaschte Netze (vgl. Abbildung 2-15). Diese Topologien haben unterschiedliche Vor- und Nachteile. Sternförmige Netze sind z.B. um eine Zentrale gruppiert, was nachteilig bei einem Ausfall der Zentrale ist. In Busnetzen hängen alle Rechner an einem gemeinsamen Medium, was zu geringen Leitungskosten führt. Alle Rechner konkurrieren allerdings um das gleiche Medium (Wettkampf).

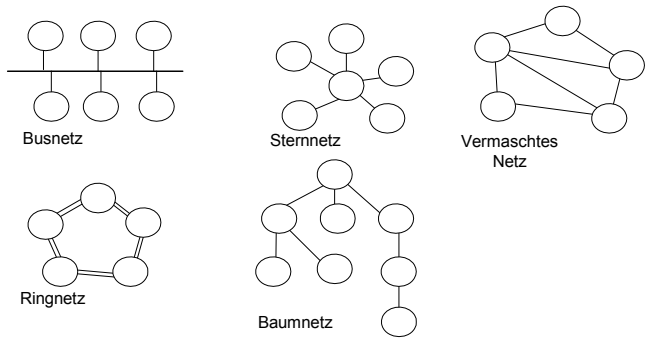


Abbildung 2-15: Netzwerktopologien

In Ringnetzen sind die Knoten ringförmig angeordnet, und die Übertragung erfolgt immer nur in eine Richtung. Meist sind die Leitungen zwischen den einzelnen Knoten auch doppelt ausgelegt. Baumnetze weisen eine hierarchische Organi-

sation mit einem Wurzelrechner aus. Fällt dieser aus, so ist das problematisch. Robust gegen Ausfall, dafür aber mit hohen Leitungskosten einhergehend sind sog. vermaschte Netze, wo ein Rechnerknoten immer mit mindestens zwei anderen verbunden ist. Es ist auch möglich, dass ein Netz wie ein Bus arbeiten, aber als Stern vernetzt ist. In LANs werden häufig logische Bussysteme eingesetzt.

2.2.3 Buszugriffsverfahren

In Bus-basierten Netzen sind sog. Buszugriffsverfahren für den Medienzugang erforderlich. Dies sind Mechanismen für den zeitgleichen Zugriff auf das Übertragungsmedium durch mehrere Knoten (Stationen). Diese Bus- oder allgemein Kanalzugriffsverfahren sind in der Schicht 2 und zwar in der MAC-Teilschicht angeordnet. Betrachtet werden im Folgenden vorwiegend Buszugriffsverfahren im LAN-Bereich.

Die Zuteilung des Busses muss eindeutig geregelt werden. Nach Abbildung 2-16 kann man die Buszugriffsverfahren in *gesteuerte* und *ungesteuerte* Verfahren einteilen. Gesteuerter Zugriff bedeutet, dass es einen klar definierten Steuerungsmechanismus für die Zuteilung des Busses gibt, während ein ungesteuertes Zugriffsverfahren zufällig ist. Ungesteuerter Zugriff erfordert eine Carrier-Sense-Multiple-Access-Strategie (kurz: *CSMA-Strategie*).

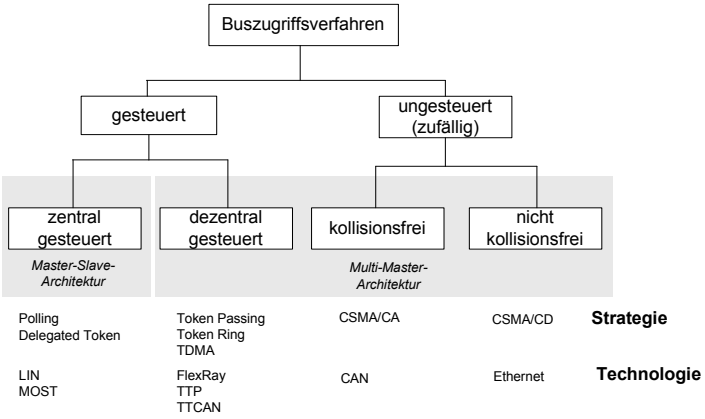


Abbildung 2-16: Einteilung der Buszugriffsverfahren nach (Schäuffele 2003)

Man unterscheidet hier wiederum in *kollisionsfreien* und *nicht kollisionsfreien* Buszugriff. Bei letzterem kann jeder Knoten zu jeder Zeit auf den Bus zugreifen, was zu Kollisionen führen kann. Daher ist eine Kollisionsstrategie erforderlich.

Strategien, die Kollisionen zwar erkennen und behandeln, aber nicht vermeiden, werden als *CSMA/Collision-Detecting-Strategien* (kurz: CSMA/CD) bezeichnet. Der bekannteste Technologievertreter dieser Strategie ist Ethernet. Eine Strategie, die

Kollisionen möglichst vermeiden kann, wird als *CSMA/Collision-Avoidance-Strategie* (kurz: CSMA/CA) bezeichnet. Dies kann man z.B. durch Nachrichtenpriorisierung erreichen. Nur die Nachrichten mit höchster Priorität dürfen gesendet werden. So wird dies z.B. bei CAN-Bussen (Controller Area Network) gemacht. CAN ist ein von der Firma Bosch Anfang der 80er Jahre für den Einsatz in Kraftfahrzeugen entwickeltes Feldbussystem, welches heute einen vielfältigen Einsatz bei Fahrzeugen vorweist.

Weiterhin kann man zwischen *Master-Slave-* und *Multi-Master-Architektur* unterscheiden. In einer Master-Slave-Architektur hat man einen Masterknoten, der die Steuerung des Buszugangs übernimmt. Bei einer Multi-Master-Architektur erfolgt die Zugangssteuerung dezentral durch alle Knoten. Master-Slave-Architekturen sind einfacher zu realisieren, sind aber nicht so ausfallsicher wie Multi-Master-Architekturen. Bei Ausfall des Masterknotens funktioniert der Bus in der Regel nicht mehr. Multi-Master-Verfahren können entweder token- oder zeitgesteuert sein. Ein Token ist hier eine spezielle Nachricht, die von Knoten zu Knoten weitergegeben wird. Hat ein Knoten das Token, darf er für eine definierte Zeitspanne den Bus zum Senden seiner Nachrichten nutzen. Danach gibt er das Token weiter. Bei zeitgesteuerten Verfahren erhält jeder Knoten einen festen Zeitschlitz für die exklusive Nutzung des Busses. Man spricht hier auch von *Time-Division-Multi-Access-Strategie* (kurz *TDMA*-Strategie). In diese Kategorie fallen z.B. *FlexRay*, *TTN* (Transit Television Network) und *TTCAN* (Time Triggered CAN): *FlexRay* ist ein Kommunikationssystem, das eine zuverlässige und echtzeitfähige Datenübertragung zwischen den elektrischen und mechatronischen Komponenten im Automobil darstellt. Mit einer Datenübertragungsrate von 10 Mbit/s ist FlexRay schneller als die heute in Kraftfahrzeugen eingesetzten Datenbusse wie CAN. Es wird vom FlexRay-Konsortium (BMW AG, DaimlerChrysler AG, ...) standardisiert.

In der Abbildung 2-156 sind auch typische Technologien dargestellt, die in die jeweiligen Kategorien passen. In betrieblichem Umfeld findet man in erster Linie die verschiedenen Varianten der Ethernet-Technologie, die CSMA/CD als Zugriffsstrategie nutzt. In Realzeitumgebungen, also in Systemumgebungen, in denen es mehr auf deterministisches Verhalten des Busses ankommt, findet man Feldbussysteme wie etwa CAN, LIN (Local Interconnect Network)[9], MOST (Media Oriented Systems Transport)[10] und FlexRay. Diese Bussysteme sind vorwiegend auch im

[9] LIN ist ein serieller Bus zur Vernetzung einfacher Aktoren und Sensoren im Bereich von Automobilanwendungen. Das Bussystem verfügt über ein einfaches Protokoll und eine einfache Ablaufsteuerung.

[10] MOST ist ein Multimedia-Glasfaserbus für Applikationen in der Automobilelektronik. Es definiert Protokolle sowie Hardware- und Software-Ebenen für die Übertragung von Steuerungs-, Echtzeit- und Paketdaten. MOST wird u.a. für die Kommunikation multimedialer Systeme im Auto eingesetzt (Überbegriff Infotainment im Fahrzeug).

Automotive-Umfeld, also in Autos (BMW,...) vorzufinden. Sie verbinden dort die verschiedenen Steuergeräte miteinander.

Eine etwas andere Klassifizierung unterscheidet folgende Zugriffsverfahren:

- *Wettkampfverfahren:* Man kann sich als Analogie vorstellen, dass eine Diskussionsgruppe diskutiert, in der alle gleichberechtigt reden dürfen**.** Typisches Verfahren ist hier auch CSMA/CD, das im LAN und zwar in Bustopologien eingesetzt wird.
- *Token-Passing-Verfahren:* Dieses Verfahren kann man mit einer Diskussionsgruppe vergleichen, in der derjenige reden darf, dem der Diskussionsleiter das Wort erteilt hat. Das Verfahren ist auch im LAN interessant. Token-Passing stammt aus der IBM-Welt und wird in Ringnetzen verwendet.
- *Distributed-Queue-Dual-Bus-Verfahren (DQDB):* Bei diesem Verfahren darf man innerhalb der Diskussionsgruppe reden, wenn man gemäß der Reihenfolge eingehender Wortmeldungen dran ist. DQDB wurde in MAN (Ringnetze) eingesetzt, ist heute aber nicht mehr verbreitet.

2.2.4 Fallbeispiel: CSMA-Protokolle

Bei Mehrfachzugriffskanälen (auch als Vielfachzugriffskanäle bezeichnet) pflanzen sich alle von einer Station gesendeten Signale in beide Richtungen des Kanals fort. In Ermangelung eines zentralen Controllers, können alle Stationen unabhängig voneinander arbeiten und gleichzeitig senden (Multiple Access = MA). Hier kommt es zu Kollisionen, die mit speziellen Aktionen bearbeitet werden müssen. Ein verteiltes Koordinationsschema ist also erforderlich, um Kollisionen zu erkennen und zu beseitigen (Tanenbaum 2003a).

Es gibt mehrere Algorithmen für den nebenläufigen Zugriff auf einen Mehrfachzugriffskanal im Wettkampfverfahren. *ALOHA* und seine verschiedenen Versionen waren Ende der 60er und in den 70er Jahren die bekanntesten Verfahren. Unter dem Begriff ALOHA werden gelegentlich auch alle Wettkampfverfahren zusammengefasst. ALOHA wurde ursprünglich von der Universität von Hawaii entwickelt. Das ALOHA-Verfahren wurde in einer sog. „*slotted*" und einer „*reinen*" Form betrieben. In der *slotted* Version wurden Zeitintervalle für Frames (Rahmen) definiert, die beim Senden einzuhalten waren. Das Senden eines Frames war also immer an einen festen Zeitschlitz gebunden. Im „reinen" (pure) ALOHA gab es keine Zeitschlitze, und damit konnte jede Station senden wann sie wollte. Eine Überprüfung des Kanals, ob er frei war, wurde beim reinen ALOHA nicht durchgeführt. Das führte je nach Netzwerkauslastung zu vielen Kollisionen und damit zu niedrigem Durchsatz. *Pure* ALOHA war aber nicht sehr effektiv, da alle Stationen, die senden wollten, ohne Rücksicht auf andere auch senden durften. *Slotted* ALOHA war etwas effektiver.

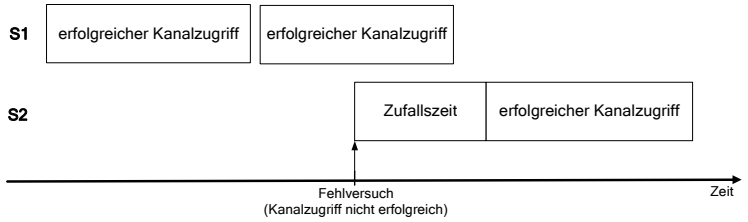


Abbildung 2-17: Szenario eines Kanalzugriffs bei non-persistent CSMA

Protokolle, die den Kanal (Träger, Carrier) dagegen vor dem Senden abhören und dann entscheiden, ob gesendet werden kann, heißen auch Trägererkennungsprotokolle (Carrier Sense Protocols) oder *CSMA*-Protokolle. Auch CSMA ist nicht kollisionsfrei. Es gibt verschiedene CSMA-Varianten. Man unterscheidet nicht persistente (non-persistent) und p-persistente[11] CSMA-Verfahren.

Non-persistent CSMA. Bei non-persistent *CSMA* wird vor dem Senden geprüft, ob der Kanal frei ist. Falls dies der Fall ist, wird gesendet. Ist dies nicht der Fall, wird eine zufällig verteilte Zeit gewartet und dann wieder von vorne begonnen. Dadurch wird die Wahrscheinlichkeit vermindert, dass ein belegter Kanal nach dem Freiwerden gleichzeitig durch mehrere wartende Stationen belegt wird und somit wiederum Kollisionen provoziert werden. Ein Kanal-Zugriffsszenario ist in Abbildung 2-17 skizziert. Hier greifen zwei Stationen S1 und S2 auf den Kanal zu. S1 ist zunächst erfolgreich, S2 hat, bevor sie den Kanal erfolgreich nutzen kann, erst einmal einen Fehlversuch und wartet daraufhin eine zufällig lange Zeit vor dem erneuten Kanalzugriff.

p-persistent CSMA. Beim *p-persistent CSMA* prüft eine Station vor dem Senden auch, ob der Kanal frei ist. Falls dies der Fall ist, wird das Paket nur mit der Wahrscheinlichkeit p sofort gesendet. Mit der Wahrscheinlichkeit 1 – p wird zunächst eine bestimmte Zeit gewartet und danach der Sendeversuch wiederholt.

Bei belegtem Kanal beobachtet die sendewillige Station den Kanal, bis dieser frei wird. Sobald er frei wird, wird wie gerade erläutert verfahren. Ein Szenario für einen typischen zeitlichen Verlauf bei p-persistent CSMA ist in Abbildung 2-18 skizziert. Im Beispiel nutzt zunächst die Station S1 den Kanal (Wahrscheinlichkeit p) und wartet beim zweiten Versuch (Wahrscheinlichkeit 1 – p) eine Zeit t, bis der Kanal genutzt wird. Die Station S2 hat zunächst einen Fehlversuch, weil S1 gerade sendet, versucht es beharrlich weiter und erhält dann schließlich den Kanal. Anschließend hat S1 ebenfalls einen Fehlversuch.

Die Zeit t ist so gewählt, dass ein Bit gerade Zeit hat, den Kanal zu durchlaufen. Dadurch wird erreicht, dass bei zwei oder mehreren sendewilligen Stationen nur

[11] Persistent = beharrend.

eine Station sendet. Die zweite hat in der Zeit t die Möglichkeit festzustellen, dass der Kanal belegt ist.

Die Wahrscheinlichkeit p kann in diesem Verfahren variiert werden, um die Übertragungskapazität bzw. die Verzögerungszeit zu optimieren.

1-persistent CSMA. Eine Spezialvariante ist das *1-persistent CSMA*. In dieser Variante hört eine Station den Kanal ab, wenn sie senden möchte. Ist der Kanal besetzt, wird gewartet, bis er frei ist, ansonsten wird ein Rahmen übertragen. Der Name kommt daher, weil eine Station mit einer Wahrscheinlichkeit von 1 sendet, wenn der Kanal frei ist.

Die CSMA-Verfahren sind effektiver als das reine und das slotted ALOHA-Verfahren, da sie sicherstellen, dass keine Station sendet, wenn der Kanal gerade belegt ist. Eine weitere Verbesserung ergibt sich, wenn eine Station bei Feststellung einer Kollision sofort aufhört zu senden. Diese Erweiterung wird als *CSMA/CD* bezeichnet, wobei „CD" für *Collision Detection* (Kollisionserkennung) steht.

Das CSMA/CD-Verfahren ist das bekannteste und wichtigste Medienzugangsverfahren bzw. Mehrfachzugriffsprotokoll und ist im LAN-Umfeld heute sehr verbreitet. Das Verfahren funktioniert nun so, dass alle Stationen, die an den Kanal (Bus) angeschlossen sind, das Übertragungsmedium abhören (Carrier Sense = CS) und ständig prüfen, ob andere Stationen senden (Busy Waiting). Eine Station (ein Rechner), die eine Nachricht senden möchte, kann dies völlig autark entscheiden und muss dabei einige Regeln beachten.

Bei CSMA/CD überwacht eine Station also zunächst das Medium. Ist es belegt, muss sie ihren Übertragungswunsch zurückstellen. Ist das Medium frei, erfolgt nach dem Verstreichen einer bestimmten Zeitspanne die Übertragung, ansonsten wartet die sendewillige Station eine zufällige Zeit. Ist der Kanal frei, sendet die Station und hört gleichzeitig das Medium ab. Stimmt das, was abgehört wird, nicht mehr mit dem überein, was gesendet wurde, liegt eine Kollision vor (Collision Detection = CD). In diesem Fall hören alle Stationen auf zu senden und warten eine zufällige Zeit, bis sie wieder anfangen zu senden.

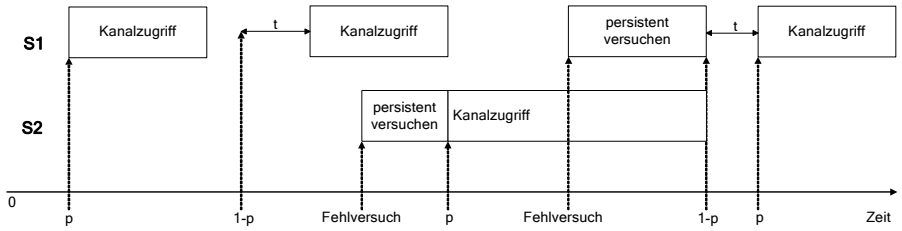


Abbildung 2-18: Szenario eines Kanalzugriffs bei p-persistent CSMA

Damit die Kanalbelegung oder eine Kollision auch von der am weitesten entfernten Station erkannt werden kann, ist eine Mindestnachrichtenlänge erforderlich, die von der Signallaufzeit, also von der Übertragungsgeschwindigkeit und der Kanallänge, abhängt. Das Senden einer Nachricht sollte doppelt so lange wie die Signallaufzeit sein. Damit wird auch eine Kollision erkannt, wenn die entfernteste Station kurz vor dem Ende eines Rahmens selbst zu senden beginnt.

Das CSMA/CD-Verfahren wird auch im Ethernet-Standard IEEE 802.3 in einer speziellen Variante verwendet. Es soll noch angemerkt werden, dass das verwendete Zugriffsverfahren auf das Medium (CSMA/CD) relativ einfach zu implementieren ist. Bei geringer Last erhält eine Station auch schnell den Zugriff auf das Medium. Jedoch können Stationen bei vielen Kollisionen auch benachteiligt werden. CSMA/CD ist kein deterministisches Verfahren, das einer Station den Zugriff innerhalb einer genau festgelegten Zeitspanne zusichert. Daher ist es nicht echtzeitfähig und auch nicht so gut für Multimedia-Anwendungen geeignet.

2.2.5 Überblick über konkrete Netzwerktechnologien

Konkrete Ausführungsformen bzw. Ausprägungen von Netzen stammen meist aus Entwicklungen bei konkreten Herstellerfirmen und wurden oft nachträglich standardisiert. Hierzu gehören u.a.:

- Ethernet als LAN-Standard (IEEE 802.3).
- Token Bus als LAN-Standard (IEEE 802.4) für logische Ringtopologien auf Basis eines physikalischen Busses (u.a. von General Motors für Automatisierungsaufgaben in der Fertigung konzipiert).
- Token Ring als LAN-Standard meist in IBM-Umgebung (IEEE 802.5).
- WLAN (Wireless LAN) als drahtloses Netzwerk (IEEE 802.11).
- DQDB als MAN-Standard (IEEE 802.6).
- FDDI bzw. FDDI II (Fiber Distributed Data Interface) als MAN-Standard, ein Doppelring auf Basis von Lichtwellenleitern, der Einsatz in Backbones findet (100 Mbit/s bei Entfernungen von mehr als 200 km, bis zu 1000 Stationen).
- ATM (Asynchronous Transfer Mode) als WAN/LAN-Standard im Telefonnetz oder auch für LAN-Backbones.

Die ersten fünf genannten Systembeispiele sind im IEEE-Normenwerk 802 für LAN und MAN zusammengefasst. ATM wird durch das ATM-Forum genormt. Wir wollen exemplarisch IEEE 802.3 und 802.11 etwas näher betrachten. In Abbildung 2-19 sind die im IEEE 802-Standard behandelten Normen skizziert. Man sieht in der Abbildung die etwas erweiterte Schichtung.

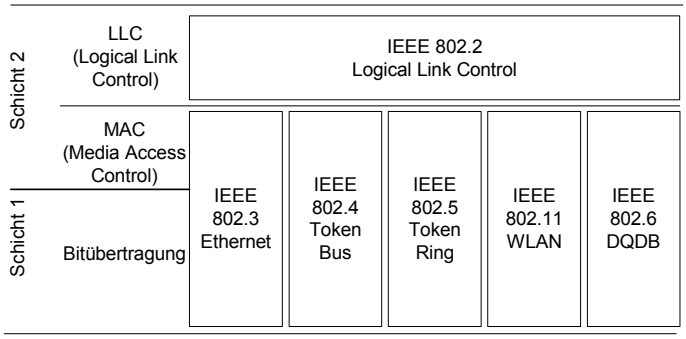


Abbildung 2-19: IEEE-Standardisierungsgruppen

Die Schicht 2 umfasst die IEEE-802.2-Norm für die LLC-Sublayer und reicht in die IEEE-802.3-Norm hinein. Diese Norm ist ständig in der Weiterentwicklung, und es gibt auch mittlerweile viele Normierungs-Untergruppen. Beispielsweise beschreibt die Unternorm *802.3ab* 1000Base-T für ein Gbit/s-Ethernet über Twisted-Pair und die Unternorm *802.3y* 100Base-T2 über 100 Mbit/s auf Basis eines Twisted-Pair-Kabels niedrigerer Qualität.

2.3 Übungsaufgaben

1. Unterscheiden Sie die Begriffe Transferzeit, Laufzeit und Übertragungszeit.
2. Was beschreibt die Schrittgeschwindigkeit und welche Einheit wird verwendet?
3. Was versteht man bei der Verkabelung unter S/STP?
4. Erklären Sie die Grundprinzipien der strukturierten Verkabelung.
5. Erläutern Sie den Unterschied zwischen p-persistent und 1-persistent CSMA?
6. Was versteht man unter gesteuerten Buszugriffsverfahren und welche Varianten gibt es hierfür?
7. Was versteht man unter ungesteuerten Buszugriffsverfahren und welche Varianten gibt es hierfür?
8. Wozu benötigt man in der Schicht 2 ein XON/XOFF-Protokoll?

3 Ausgewählte Technologien und Protokolle unterer Schichten

Bevor die höheren Kommunikationsschichten und die wichtigsten Protokolle daraus betrachtet werden, sollen vorab einige Protokolle, Technologien und Übertragungstechniken für den LAN- und WAN-Bereich, die gemäß ISO/OSI-Referenzmodell überwiegend in den unteren Schichten angesiedelt sind, anhand von Fallbeispielen betrachtet werden.

In diesem Kapitel wird ein knapper Überblick zu ausgewählten Themen gegeben. Stellvertretend für einen Standard der Schicht 1 wird *RS-232* diskutiert. Für die Schicht 2 wird das *HDLC*-Protokoll, das in vielen Netzwerken in mehreren Varianten vorzufinden ist, sowie das *PPP*-Protokoll vorgestellt. *Ethernet*, virtuelle LAN (VLAN) und wireless LAN (WLAN) werden beispielhaft als Technologien der unteren beiden Schichten im LAN-Bereich betrachtet. Mit *ISDN* wird ein Dienst, der weit verbreitet, aber nun schon in die Jahre gekommen ist, als Technologie und Zugang für Weitverkehrsnetze vorgestellt. Das *DSL*-Verfahren, das die gleichen Kabel wie ISDN verwendet, wird ebenfalls kurz skizziert. *PDH*, *SDH* und *SONET* sind Übertragungsverfahren, die in Weitverkehrsnetzen eingesetzt werden. Diese Verfahren sollen ebenfalls kurz vorgestellt werden. Am Ende des Kapitels wird noch *ATM* als WAN-Technologie eingeführt.

Zielsetzung des Kapitels

Ziel dieses Kapitels ist es, einen Überblick über ausgewählte Protokolle und Technologien unterer Kommunikationsschichten, insbesondere von RS-232, HDLC, PPP, Ethernet, WLAN und ATM zu geben. Der Studierende sollte die Funktionalität dieser Protokolle und Technologien erläutern können.

Wichtige Begriffe

RS-232, HDLC (NRM, ARM, ABM), PPP, Ethernet (100Base-T,…), Ethernet-Hub und Ethernet-Switch, WLAN, DSL, ATM, VLAN.

3.1 Bitübertragungsschicht: Der RS-232-Standard

Ein weit verbreiteter Standard für die Übertragung von Daten über kurze Entfernungen ist der von der Electronic Industries Association (EIA)[1] genormte RS-232-Standard. Der Standard hat die vollständige Bezeichnung RS-232-C und wird seit

[1] EIA ist ein Zusammenschluss von Herstellern elektronischer Geräte.

1991 auch EIA-232-C-Standard genannt. Die internationale Norm der CCITT heißt V.24. V24 stimmt aber mit RS-232-C weitgehend überein und soll uns als Beispiel für eine Schicht-1-Konvention dienen.

Im RS-232-Standard werden die Signal-Spannung, das Signal-Timing, ein Protokoll zum Austausch der Information sowie die mechanischen Verbindungen festgelegt. RS-232 definiert eine serielle, asynchrone Kommunikation zwischen zwei Rechnern oder zwischen einem Rechner und einem Modem. Die Übertragung erfolgt seriell (im Gegensatz zu parallel), also Bit für Bit nacheinander über einen Leiter (ein Pin ist für das Senden reserviert). Die Übertragung erfolgt asynchron in ganzen Zeichen. Asynchron bedeutet hier, dass sich Sender und Empfänger vor der Übertragung nicht synchronisieren. Abbildung 3-1 zeigt die typische Verbindung zwischen einem Rechner (DTE = Data Terminal Equipment) und einem Modem (DCE = Data Circuit-terminating Equipment). Auf der DCE-Seite ist die Pin-Belegung „female“, auf der DTE-Seite „male“.

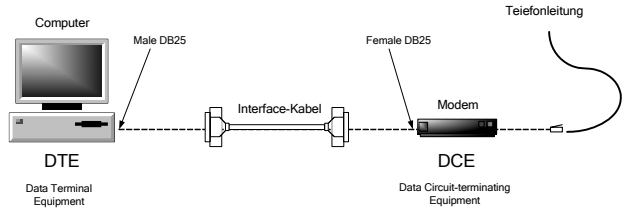


Abbildung 3-1: Kommunikation zwischen DTE und DCE in RS-232

Die Übertragung erfolgt in einer einfachen *Binärkodierung*. Eine Spannung zwischen –3V und –25V wird als logische '1', eine Spannung zwischen +3V und +25V als logische '0' betrachtet. Weiterhin gilt:

- Für die Übertragung eines Zeichens werden insgesamt neun Bit benötigt.
- Vor jeder Übertragung eines Zeichens wird ein 0-Bit, das sog. *Startbit* übertragen. Dieses 0-Bit ist notwendig, weil die Spannung nach der Übertragung auf einen 1-Bit-Wert stehen bleibt.
- Nach der Übertragung eines Zeichens wird ein Stopbit (0b1) gesendet.
- Ein Zeichen wird in sieben Bit kodiert.

Die Spannung zur Übertragung eines Bit wird beim Senden für eine vorgegebene Mindestzeit angelegt, damit der Empfänger für das Abgreifen genügend Zeit hat.

Der RS-232-Standard ist für Datenraten bis 20.000 Bit/s bei Kabellängen bis zu 15 m ausgelegt. Sender und Empfänger müssen die gleiche Signalgeschwindigkeit eingestellt haben, wobei 19.200 Baud üblicherweise als Obergrenze eingesetzt wird. Auch Baudraten von 300, 600, 1200, 4800 und 9600 werden verwendet. Um Fehler zu erkennen, misst der Empfänger eine anliegende Spannung mehrmals und ver-

gleicht die Messergebnisse miteinander. Stellt er Unterschiede fest, wird ein Framing-Error signalisiert.

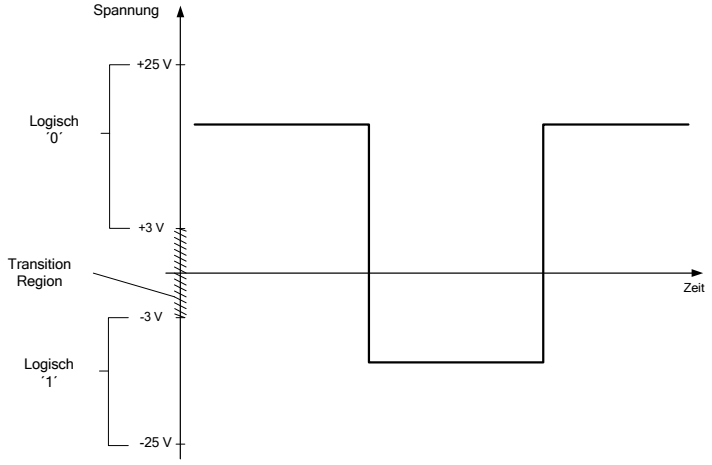


Abbildung 3-2: Kodierung bei RS-232

Kein Kabel leitet Elektrizität perfekt. Das *Signal-Timing* unterliegt daher einigen Konventionen, und beim Wechsel von 0b1 auf 0b0 oder umgekehrt muss einiges beachtet werden. Beispielsweise müssen sich Signale während eines Zustandswechsels zum entgegengesetzten Signalwertbereich bewegen, ohne die Richtung zu ändern (siehe die in der Abbildung 3-2 dargestellte Transition Region). Die Transitionszeit für den Übergang von einem Zustand in den anderen sollte weniger als eine Millisekunde betragen. Ein akzeptabler Puls sowie drei fehlerhafte Pulse sind in Abbildung 3-3 dargestellt. Ein fehlerhafter Puls führt möglicherweise zu einem Framing-Error.

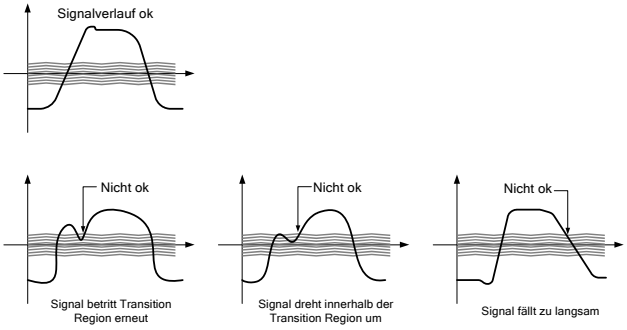


Abbildung 3-3: Saubere und fehlerhafte Transitionen bei RS-232

RS-232 beschreibt eine *Duplexübertragung*. Im Gegensatz zu einem Halbduplex-Verfahren steht je Richtung ein Leiter zur Verfügung. Die Steckergröße und die einzelnen Anschlüsse (Pins) sind genau festgelegt. Die Pins sind durchnummeriert. Für die Erdung wird für beide Richtungen ein Leiter verwendet.

Die Pin-Belegung sieht einen 25-poligen Stecker vor (DB25-Stecker). Der Standard gibt vor, wie die 25 Pins belegt sind. 22 Pins werden derzeit genutzt he Abbildung 3-4), neun davon werden meistens verwendet (Comer 2002):

- Pin 1 ist die Gehäusemasse (Erdung) für beide Richtungen.
- Mit *Data-Terminal-Ready* (DTR, Pin 20) bestätigt der Rechner dem Modem, dass er eingeschaltet ist (beim Hochfahren).
- Wird das Modem eingeschaltet, so sendet es eine 0b1 auf *Data-Set-Ready* (DSR, Pin 6).
- Wenn das Modem einen Träger (ein Signal) erkennt, so wird *Carrier-Detect* (Pin 8) gesendet.
- Mit *Request-to-Send* (RTS, Pin 4) wird durch den Rechner ein Sendewunsch signalisiert.
- *Clear-to-Send* (CTS, Pin 5) signalisiert die Empfangsbereitschaft des Modems.
- Mit *Transmit (*Pin 2) und *Receive* (Pin 3) wird gesendet und empfangen.

Die verfahrenstechnische Spezifikation, also das Protokoll ist ebenfalls festgelegt. Damit ist die zulässige Reihenfolge der Signale bzw. Ereignisse definiert. Das Protokoll sieht im Wesentlichen sog. Request-/Response-Paare vor. Wenn etwa der Rechner ein RTS-Signal anlegt, so muss das Modem mit CTS antworten, sofern es Daten empfangen kann.

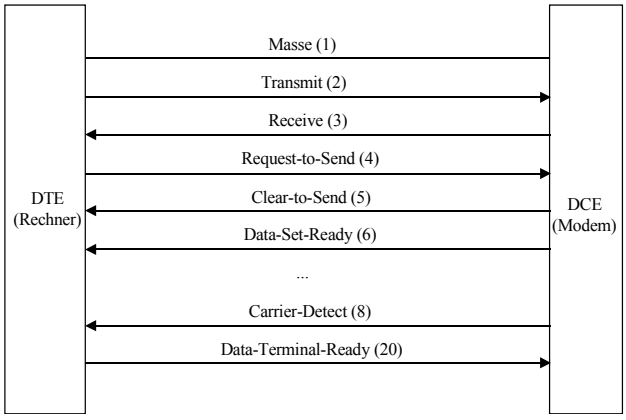


Abbildung 3-4: Pin-Belegung bei RS-232

Eine weitere Nutzungsvariante von RS-232 ist die direkte Verbindung zweier Rechner über ein sog. *Nullmodem*, das in einem Kabel mit gekreuzten Pins ausge-

legt ist. Hier wird z.B. der Transmit-Anschluss des einen Rechners mit dem Receive-Anschluss des anderen Rechners verbunden (Kreuzung).

Die Weiterentwicklung des RS-232-Standards ist die Normengruppe RS-449, die aus den Normen RS-449-A, RS-422-A und RS-423-A besteht. Dieser Standard unterstützt Übertragungsgeschwindigkeiten bis 2 Mbit/s über 60 m lange Kabel. Hier wird ein 37-poliger Stecker benutzt.

Es soll noch angemerkt werden, dass RS-232 für kurze Entfernungen definiert und daher auch nur eine einfache Kodierung mit positiver und negativer Spannung ausreichend ist. Diese Technik eignet sich nicht so ohne weiteres für große Entfernungen, da der Strom mit der zurückgelegten Strecke zunehmend schwächer wird. Durch den Widerstand im Kabel entsteht ein Signalverlust, die elektrische Energie wird in Wärmeenergie umgewandelt. Was man benötigt ist daher ein kontinuierlich schwingendes Signal (Sinuswellen), das auch als Träger bezeichnet wird. Der Träger schwingt ständig mit, auch wenn keine Daten gesendet werden. Die Information wird durch die sog. Modulation, also durch eine geringfügige Modifikation des Trägers durch den Sender übertragen. Diese Modifikation erkennt der Sender und kann die Information daraus dekodieren. Eine Art der Modulation ist z.B. die Veränderung der Phase (Phasenmodulation), andere werden als Amplituden- und Frequenzmodulation bezeichnet.

3.2 Protokolle und Technologien der Sicherungsschicht

3.2.1 HDLC-Protokoll

Ein bekanntes Protokoll der Schicht 2 ist das bitorientierte *HDLC*-Protokoll (High Level Data Link Control Protocol), das ursprünglich unter dem Namen SDLC (Synchronous Data Link Layer) in der SNA-Architektur (System Network Architecture) von IBM entstand. Die ISO nennt das Protokoll HDLC (ISO 33009 und ISO 4345), die CCITT übernahm das Protokoll und machte LAP (Link Access Protocol) daraus. Im X.25-Paketnetz wurde es auch in der Schicht 2 eingesetzt und unter dem Namen LAPB bekannt. HDLC ist exakt in der Schicht 2a (LLC, Locical Link Control) angesiedelt. HDLC findet vor allem in öffentlichen Netzen (WAN) Anwendung.

Das Protokoll bietet einen *verbindungsorientierten* Dienst. Es wird zur gesicherten Punkt-zu-Punkt-Verbindung zwischen zwei Rechnern, aber auch für Punkt-zu-Mehrpunkt-Verbindungen verwendet (ein Rechner steuert mehrere Stationen). Es führt folgende Aufgaben aus:

- Datenübertragung
- Flusskontrolle
- Reihenfolgekontrolle
- Fehlerkontrolle und Fehlerbenachrichtigung an die Schicht 3

HDLC-Stationstypen und Betriebsmodi. Man unterscheidet in HDLC *drei Stationstypen*: Primäre, sekundäre und kombinierte Stationen. Eine primäre Station steuert die Kommunikation, sekundäre Stationen antworten auf die Kommandos der primären Station. Zwischen primären und sekundären Stationen wird je eine eigene Verbindung verwaltet. Kombinierte Stationen können in beiden Rollen arbeiten.

HDLC unterscheidet weiterhin zwei Typen von *Link-Konfigurationen*. Bei unsymmetrischen Konfigurationen arbeiten mehrere Sekundärstationen mit einer Primärstation zusammen. Bei der symmetrischen Link-Konfiguration arbeiten zwei kombinierte Stationen. Die Datenübertragung geschieht jeweils im Voll- oder Halbduplex-Verfahren.

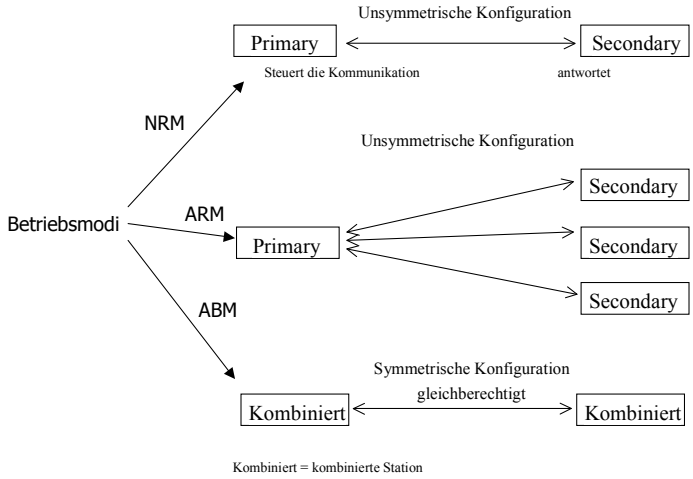


Abbildung 3-5: HDLC-Betriebsmodi

HDLC erlaubt durch die Kombination der Link-Konfigurationen mit dem Einsatz der vorhandenen Stationstypen mehrere *Betriebsmodi* (Abbildung 3-5):

- *Normal Mode* (NRM): Dieser Betriebsmodus wird bei unsymmetrischer Konfiguration verwendet. Nur die primäre Station löst die Übertragung aus, die sekundären antworten entsprechend.
- *Asynchronous Response Mode* (ARM): Dieser Betriebsmodus wird bei unsymmetrischer Konfiguration verwendet. Die sekundäre Station kann die Übertragung ohne Erlaubnis der primären beginnen.
- *Asynchronous Balanced Mode* (ABM): Dieser Betriebsmodus wird bei symmetrischer Konfiguration verwendet. Die Stationen sind jeweils kombiniert und können die Übertragung beliebig auslösen.

HDLC-Rahmenformat. HDLC ist ein *bitorientiertes* Protokoll. Das bedeutet, dass die Daten bitweise kodiert über die Leitung gehen und daher keine speziellen Zeichen für die Erreichung von Datentransparenz verwendet werden. Bei bitorientierten Protokollen wie HDLC hat man das Problem der Datentransparenz auf der Bitebene. Über eine Rahmung mit Hilfe eines speziellen Flags (Zeichen mit spezieller Bitfolge 0b01111110) am Anfang und am Ende jeder Nachricht wird nämlich die Nachrichtenbegrenzung kenntlich gemacht. Um dieses Flag in den Nutzdaten zu erkennen (Datentransparenz), verwendet man die *Bit-Stuffing*-Technik (Bitstopfen). Nach fünf aufeinanderfolgenden binären Einsen wird vom Sender (von der Schicht-2-Instanz) eine binäre Null ergänzt, die der Empfänger nach demselben Muster wieder entfernt.

Ähnlich verhält es sich bei den sog. zeichenorientierten Protokollen, wo im Wesentlichen ASCII-Zeichen durch die Leitung gehen und spezielle Zeichen als Steuerzeichen genutzt werden. Um Steuerzeichen in den Daten zu erkennen, wird *Character-Stuffing* verwendet, d. h. es werden „Entwertungszeichen wie das ASCII-Zeichen DLE vor die Sonderzeichen gesetzt. Dies wird vom Empfänger entsprechend interpretiert.

Der Aufbau des HDLC-Rahmens ist in Abbildung 3-6 dargestellt.

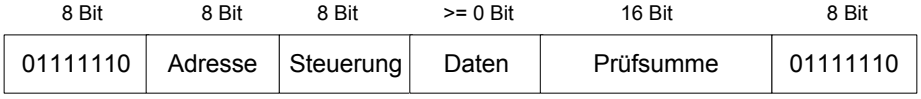


Abbildung 3-6: HDLC-Rahmenformat

Die einzelnen Felder spiegeln die Semantik des Protokolls wider und sollen deshalb kurz betrachtet werden:

- Am Anfang und am Ende einer HDLC-PDU sind jeweils die bereits bekannten Flags (0b01111110) zur Synchronisation zwischen Sender und Empfänger positioniert.
- Das Feld *Adresse* ist dann interessant, wenn mehr als zwei Rechner miteinander kommunizieren. Beispielsweise wird das Feld verwendet, um Terminals zu identifizieren, die von einem Master angesteuert werden.
- Das Feld *Steuerung* dient für verschiedene Zwecke und enthält u.a. eine Folgenummer und eine Bestätigungsnummer. Das Feld wird noch im Zusammenhang erläutert.
- Im Feld *Daten*, das auch leer sein kann, sind die Nutzdaten untergebracht (also die Schicht-3-PDU). Daten können beliebig lang sein.
- Das Feld *Prüfsumme* enthält eine Prüfsumme auf Basis eines zyklischen Redundanzcodes. Zur Prüfsummenberechnung für HDLC wird das 16-Bit-Standardgeneratorpolynom $G(x) = x^{16}+x^{12}+x^{5}+1$ verwendet, das von der

CCITT zum internationlen Standard CRC-CCITT erhoben wurde[2]. CRC-CCITT ist eine 16-Bit-Prüfsumme und wird für Codes mit einer Zeichenlänge von acht Bit verwendet. Erkannt werden damit alle 1-Bit-Fehler, alle 2-Bit-Fehler, alle ungeraden Bit-Fehler, alle Fehlerbündel[3] mit 16 oder weniger Bit, 99.997 % aller 17-Bit-Fehlerbündel und 99.998 % 18- und Mehr-Bit-Fehlerbündel[4]. Die Nutzung des Prüfsummenverfahrens kann in (Tanenbaum 2003a) nachgelesen werden.

Da die Kommunikation in HDLC verbindungsorientiert ist, gibt es eine Verbindungsaufbauphase, eine Phase des Informationsaustauschs und eine Phase der Verbindungsabbaus. Es gibt in HDLC drei Rahmenarten, die zur Kommunikation verwendet werden. Man teilt sie ein in folgende Typen:

- Informationsrahmen (I-Rahmen): I-Rahmen enthalten die Nutzdaten und Informationen zur Flusssteuerung und Fehlerbehebung.
- Überwachungsrahmen (S-Rahmen): Diese Rahmen werden zu Steuerzwecken benutzt, u.a., um Bestätigungen zu senden.
- Unnummerierte Rahmen (U-Rahmen): U-Rahmen werden für die Verbindungssteuerung benutzt („Unnumbered“ im Sinne der Folgenummernreihenfolge).

Es gibt verschiedene PDUs (sog. Kommandos), die nicht alle im Detail erläutert werden sollen. Für die Kennzeichnung der Rahmenart und des Kommandos werden die ersten fünf Bit des Feldes *Steuerung* verwendet, aber es werden nicht alle 32 Möglichkeiten genutzt. Die wichtigsten Teilfelder aus dem Feld *Steuerung* sind in Abbildung 3-7 skizziert.

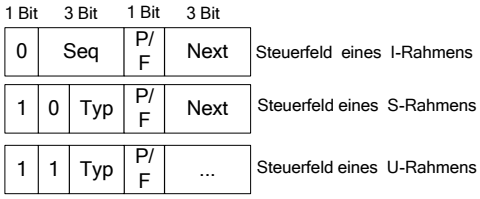


Abbildung 3-7: Steuerfelder (Tanenbaum 2003a)

Das Feld *Seq* beinhaltet die Folgenummer für einen Schiebefenster-Mechanismus (Sliding-Window) zur Flusskontrolle. Dies bedeutet, dass bis zu sieben noch nicht

---

[2] Ein weiterer Standard ist z.B. das CRC-12-Polynom, das bei einer Zeichenlänge von 6 Bit eingesetzt wird.

[3] 16-Bit-Fehlerbündel bedeutet, dass 16 Bit in Folge fehlerhaft sind.

[4] Bei Prüfsummenfeldern mit einer Breite von 32 Bit sind Fehlererkennungswahrscheinlichkeiten von 99,99999995 % möglich.

bestätigte Rahmen ausstehen dürfen. Die Nummerierung der Rahmen liegt im Bereich von 0 bis 7 (binär 0b000 bis 0b111 modulo 8, nach 7 wird also wieder 0 verwendet).

Das Feld *Next* beinhaltet die Bestätigungsnummer, die üblicherweise im Huckepackverfahren (Pickyback-Verfahren) genutzt wird. Es wird die Nummer des als nächstes erwarteten Rahmens gesendet.

Das Feld *P/F* (Poll/Final-Bit) dient der zyklischen Abfrage einer Gruppe von Stationen (Sekundärstationen) über einen steuernden Rechner (Primärstation).

Das Feld *Typ* gibt Auskunft über die verschiedenen Arten von Überwachungsrahmen (S-Rahmen, wobei es hier verschiedene Varianten gibt, die nicht alle aufgeführt werden sollen. Einige Beispiele für S-Rahmen sind:

- Positive Bestätigung, RECEIVE READY, RR (Typ 000).
- Negative Bestätigung, REJECT, REJ (Typ 001): Fordert eine Neuübertragung an.
- Negative Bestätigung, RECEIVE NOT READY, RNR (Typ 010): Bestätigt alle Rahmen bis zu *Next*, dient aber zum Ausbremsen des Senders. Wenn der Sender wieder übertragen darf, wird RR gesendet.
- Selektives Ablehnen, REJECT, REJ (Typ 100): Fordert eine Neuübertragung eines Rahmens an.

HDLC benutzt also eine Fenstertechnik mit einem einfachen Bestätigungsmechanismus. Es baut eine Verbindung auf und nach dem Informationsaustausch wieder ab. In Abbildung 3-8 ist der Ablauf einer Übertragung im ABM-Mode (gleichberechtigte Stationen) vereinfacht dargestellt, wobei die Nutzdaten vernachlässigt werden:

- Zunächst wird über einen U-Rahmen (SABM-Kommando) von Station A ein Verbindungswunsch für den ABM-Modus aufgebaut.
- Die Verbindung wird von Station B bestätigt (U-Rahmen UA = Unnumbered Acknowledge).
- Anschließend erfolgt die Datenübertragung über I-Rahmen. Der erste I-Rahmen (I, 0, 0) der Station A wird sofort von der Station B mit (S, RR, 1) bestätigt.
- Der zweite und dritte I-Rahmen ((I, 1, 0) und (I, 2, 0)) der Station A werden hintereinander, zunächst ohne Bestätigung übermittelt. Im Beispiel kommt ein I-Rahmen (I, 2, 0) beim Empfänger nicht an.
- Station B sendet daraufhin einen Rahmen und bestätigt im Huckepackverfahren, dass der Rahmen mit der Nummer 1 angekommen ist (I, 0, 2).
- Station A sendet einen Rahmen mit der Nummer 3, der nun von Station B abgewiesen wird. Über einen (S, REJ, 2)-Rahmen wird der Station A mitgeteilt, alle Rahmen ab Nummer 2 erneut zu senden.
- Dies wird von der Station A erledigt (I, 2, 1) usw.

– Am Ende der Übertragung wird der Verbindungsabbau mit einem U-Rahmen (DIS-Kommando) eingeleitet. Der Verbindungsabbauwunsch wird wiederum mit einem U-Rahmen (UA-Kommando) bestätigt.

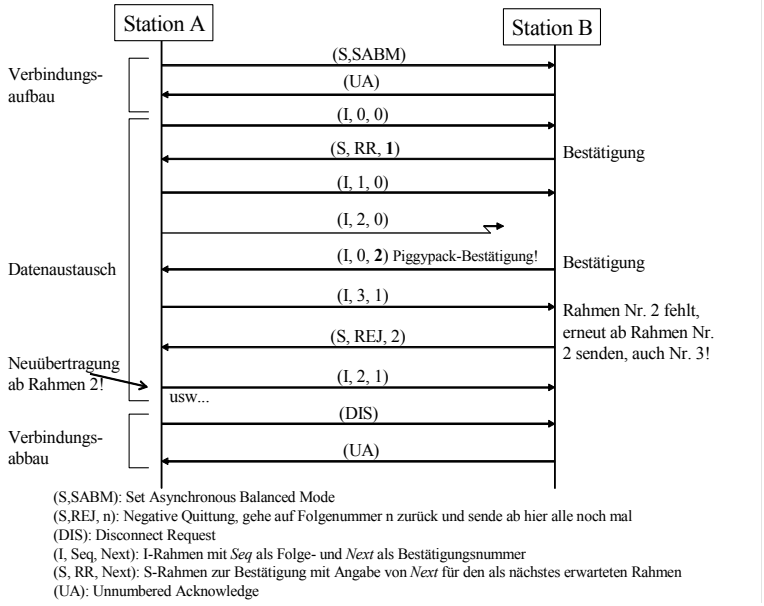


Abbildung 3-8: Sequenzdiagramm für eine typische HDLC-Kommunikation

3.2.2 Point-to-Point-Protocol (PPP)

Im Internet hat sich als Schicht-2-Protokoll für Punkt-zu-Punkt-Leitungen (also nicht für busorientierte Netze) das Protokoll PPP (RFC 1661, RFC 1662, RFC 1663) durchgesetzt. Das Protokoll wird neben dem immer weniger verbreiteten SLIP (Serial Line IP)[5] für den Internet-Zugang vom Dienstnutzer zum Internet-Provider genutzt. PPP wird also für die Verbindung zwischen einem PC im Homeoffice über Modem zum Internet-Provider eingesetzt. Ein weiteres Einsatzfeld ist die Kommunikation zwischen Internet-Routern über gemietete Leitungen. Die von PPP unterstützte Punkt-zu-Punkt-Verbindung ist eine Vollduplex-Verbindung. Über PPP wird u.a. und vorzugsweise IP als Schicht-3-Protokoll eingesetzt. PPP unterstützt wie HDLC eine Fehlerbehandlung und ermöglicht sogar das Aushandeln von IP-Adressen zur Verbindungsaufbauzeit.

[5] SLIP ist ein einfaches Schicht-2-Protokoll für Dial-Up-Telefonleitungen über Modem zum Internet. Es ist im RFC 1055 standardisiert. Über SLIP wird eine TCP/IP-Kommunikation ermöglicht.

PPP unterstützt einen sog. *Multiprotokoll-Framing-Mechanismus*, der über mehrere Schicht-1-Implementierungen wie Modemverbindungen, aber auch über bitseriellen HDLC-Verbindungen und auch über SDH/SONET-Verbindunge[6] eingesetzt werden kann.

PPP besteht aus folgenden Teilprotokollen:

- LCP (Link Control Protocol) zum Verbindungsaufbau und -abbau einer Schicht-2-Verbindung. Über LCP werden auch diverse Optionen ausgehandelt.
- NCP (Network Control Protocol) zum Aushandeln von Optionen der Netzwerkschicht. In der Regel wird hier IP unterstützt. Über NCP wird also auch die IP-Adresszuordnung ausgehandelt. Es sind aber auch andere Netzwerkprotokolle möglich.

Ein Rechner, der sich über ein Modem mit PPP ins Internet einwählen will, baut zunächst eine Modemverbindung auf. Wenn diese besteht, wird über LCP eine Schicht-2-Verbindung etabliert. Anschließend wird über NCP eine Schicht-3-Verbindung ausgehandelt und ggf. eine Netzwerkadresse (meist IP-Adresse[7]) vereinbart, über die der Rechner im weiteren Verlauf im Netzwerk adressierbar ist. Danach kann der Rechner beliebige Schicht-3-Nachrichten senden. Am Ende der Kommunikation wird die Verbindung wieder über den umgekehrten Weg abgebaut. Zunächst wird über NCP die Schicht-3-Verbindung abgebaut, anschließend über LCP die Schicht-2-Verbindung und schließlich auch die Modemverbindung.

Byte	1	1	1	1 oder 2	variabel	2 oder 4	1
	01111110	Adresse	Steuerung	Protokoll	Daten	Prüfsumme	01111110

Abbildung 3-9: PPP-Rahmenformat

Das PPP-Rahmenformat, das in Abbildung 3-9 dargestellt wird, ist dem HDLC-Rahmenformat sehr ähnlich, allerdings ist PPP im Unterschied zu HDLC zeichenorientiert (im Gegensatz zu bitorientiert) und verwendet das Zeichenstopfverfahren (Character-Stuffing), um Datentransparenz zu erreichen:

- Das Feld *Adresse* wird immer mit 0b11111111 belegt, um anzudeuten, dass alle Stationen den Frame akzeptieren. Man vermeidet damit die Zuordnung von eigenen Schicht-2-Adressen und deren Verwaltung.
- Das Feld *Steuerung* hat als Standardwert 0b00000011. Der Wert deutet einen „Unnumbered Frame“ an. Das bedeutet, dass PPP standardmäßig keine zuverlässige Übertragung mit Folgenummern und Bestätigung unterstützt.

[6] SDH und SONET werden weiter unten eingeführt.

[7] IP-Adressen werden in Kapitel 4 ausführlich behandelt.

„Numbered mode“ wie in HDLC kann aber auch genutzt werden. Die Felder *Steuerung* und *Adresse* können auch optional über LCP ausgeblendet werden, um zwei Byte in jedem Frame einzusparen.
- Das Feld *Protokoll* legt fest, welche Nachricht in den Nutzdaten liegt. Es gibt Codes für LCP, NCP, IP, IPX usw. Wenn das Feld mit 0b0 beginnt, handelt es sich um ein Paket der Netzwerkschicht, beginnt es mit 0b1, wird es als Nachricht betrachtet, die für das Aushandeln von Optionen dient. Das Feld hat normalerweise eine Länge von zwei Byte, kann aber über LCP-Mechanismen auf ein Byte heruntergesetzt werden.
- Im Feld *Daten* (auch *Payload* genannt) werden die Nutzdaten übertragen. Die Standardlänge ist 1.500 Byte, aber die Länge kann auch über LCP verhandelt werden. Auch die Länge des anschließenden Feldes *Prüfsumme* kann ausgehandelt werden.

Im RFC 1661 sind 11 LCP-PDUs definiert. Zum Aushandeln von Optionen gibt es z.B. die PDUs *Configure-Request*, *Configure-ACK*, *Configure-NAK* und *Configure-Reject* und zum Beenden einer Verbindung die PDUs *Terminate-Request* und *Terminate-ACK*. Für weitere Informationen wird auf die entsprechenden RFCs verwiesen.

3.2.3 Ethernet LAN

Eine der bekanntesten LAN-Technologien ist Ethernet, eine bus-orientierte Technologie. Das *Ethernet* wurde Anfang der 70er Jahre von R. Metcalfe[8] bei der Firma Xerox entwickelt und später von Xerox, DEC und Intel zum IEEE 802.3-Standard ausgebaut. Seine Vorgeschichte geht auf das ALOHA-System zurück. Dies ist ein an der Universität von Hawai entwickeltes Rundfunksystem auf Basis eines Mehrfachzugriffsprotokolls. Im Weiteren wird auf das Zugriffsverfahren und die Kollisionsbehandlung in Ethernet-basierten LANs, auf die verschiedenen Ethernet-Varianten und die eingesetzten Kabeltypen, auf den Aufbau der Ethernet-Rahmen (DL-PDUs), auf das Verfahren zur Prüfsummenberechnung, auf die Leitungskodierung und auf LAN-Switching eingegangen.

Zugriffs- und Kollisionsbehandlung. Ethernet ist von der Topologie her ein Bussystem und nutzt als Medienzugriffsverfahren *CSMA/CD*. Ethernet wird fälschlicherweise oft mit dem Zugriffsverfahren verwechselt. Der Ethernet-Standard umfasst aber mehr als nur das Zugriffsverfahren.

[8] R. Metcalfe ist der Gründer von 3COM, hat die Firma aber verkauft.

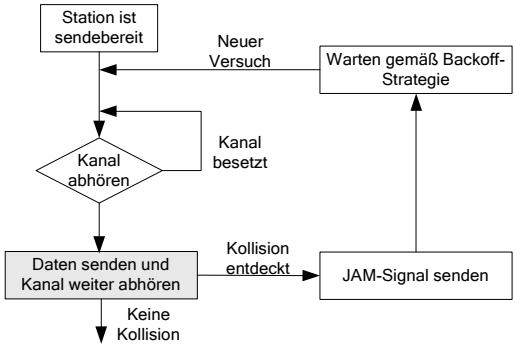


Abbildung 3-10: CSMA/CD-Algorithmus zur Kollisionsbehandlung

Der grobe Ablauf eines Sendevorgangs bei Ethernet ist grafisch in Abbildung 3-10 als Algorithmus dargestellt und sieht wie folgt aus:

- Zunächst überwacht eine Station das Medium (Carrier Sense). Ist es belegt, muss sie ihren Übertragungswunsch zurückstellen. Ist das Medium frei, erfolgt nach dem Verstreichen einer bestimmten Zeitspanne die Übertragung, wobei gleichzeitig die Übertragungsleitung (das Medium) abgehört wird.
- Die Datenübertragung beginnt mit einer Präambel, die ein binäres Muster 0b10101010 enthält, und dem anschließenden Senden der Daten. Wenn keine andere Station sendet, bis die Daten vollständig beim Empfänger sind und von diesem vom Netz gelesen wurden, ist die Übertragung erfolgreich. Alle Stationen erhalten das Paket und prüfen, ob es an sie adressiert ist. Ist das Paket bei der Zielstation angelangt, prüft diese das Paket auf Integrität (Kontrolle der korrekten Länge und Auswertung der Prüfsumme).
- Senden gleichzeitig mehrere Stationen, kommt es zu einer Überlagerung der Signale (Kollision). Angenommen zwei Stationen an entgegengesetzten Enden eines Segments beginnen zu senden. Die Signale breiten sich im Medium mit ca. 70% der Lichtgeschwindigkeit aus. Sobald sie sich überlappen, stören sie sich gegenseitig.
- Alle Stationen brechen die Übertragung bei Erkennen einer Kollision ab (Kollisionserkennung, Collision Detection = CD). Hat eine Station eine Kollision entdeckt, reagiert sie aber nicht nur mit der Unterbrechung der Übertragung, sondern erzeugt auch ein sog. 32-Bit-JAM-Signal (Störsignal), das nach links und rechts an alle Stationen weitergereicht wird. Das JAM-Signal hat den Inhalt 0b10101010 (und zwar viermal). Damit wird für die anderen Stationen die Erkennung einer Kollision erleichtert.
- Durch das Abhören nach dem Senden fällt die Störung der sendenden Station auf. Um auch zu gewährleisten, dass eine Kollision alle Stationen in einem Segment erreichen kann, muss die gesamte Transferzeit berücksichtigt

werden. Diese hängt von der maximalen Kabellänge ab und wird natürlich auch durch die PDU-Länge (Rahmengröße), für die eine minimale Größe festgelegt werden muss, beeinflusst.

– Nachdem die Stationen durch das JAM-Signal die Kollision erkannt haben, ermitteln diese zufallsabhängig und abhängig von der Zahl der aktuellen Sendeabbrüche der Station eine Zeitspanne, in der sie warten. Danach erfolgt ein erneuter Sendeversuch. Klappt es wieder nicht, wird die maximale Wartezeit verdoppelt und aus dem Intervall zwischen 0 und der maximalen Wartezeit eine Zufallswartezeit ausgewählt. Dies geschieht max. 16 mal, beim 10. Versuch wird die Wartezeit eingefroren. Dieses Verfahren wird als *binärer exponentieller Backoff* bezeichnet.

In Abbildung 3-11 sind vier Szenarien einer Kollision skizziert. Kollisionen müssen auch im kritischsten Fall erkannt werden, also auch dann, wenn die beiden Stationen gleichzeitig senden, die am weitesten auseinander liegen. Daher ist im Ethernet-Standard eine Mindestlänge der DL-PDU und eine maximale Entfernung zwischen den am weitesten entfernten Stationen vorgeschrieben. Die Station A sendet gemäß der Abbildung eine DL-PDU und überwacht das Medium für das Zweifache der maximalen Verteilungszeit. Station A geht davon aus, dass ihr Paket ordnungsgemäß übertragen ist, wenn in dieser Zeit keine Störungen auffallen.

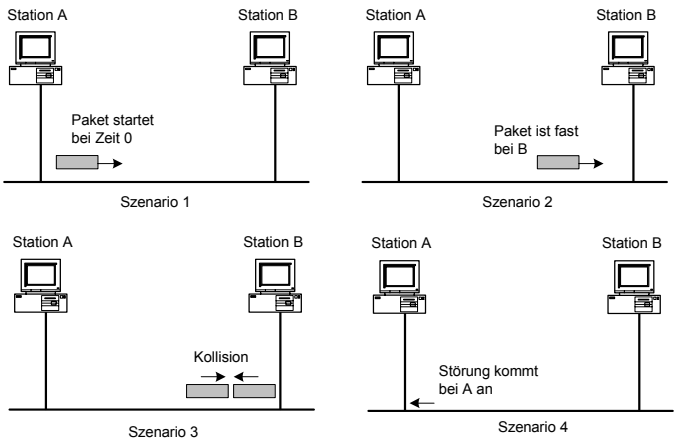


Abbildung 3-11: Kollisionen im Ethernet nach (Tanenbaum 2003a)

Der Zeitraum gemessen in sog. Bitzeiten (Zeit, die ein Bit benötigt), nach dem das Medium eindeutig durch eine Station als belegt gilt, wird als *Slot-Time* bezeichnet. Die Slot-Time korrespondiert in einem 10-Mbit/s-Ethernet mit einer minimalen Rahmengrößen von 64 Byte und ist daher 512 Bitzeiten (64 * 8 Bit = 512 Bit) lang. Innerhalb der Slot-Time muss eine Kollision erkannt werden, also müssen innerhalb dieser Zeit die konkurrierenden Stationen auch noch senden, da sie ansonsten

eine Kollision gar nicht bemerken würden. Damit wird auch deutlich, warum es eine minimale Rahmengröße geben muss.

Im Beispiel sendet die adressierte Empfängerstation B kurz vor Ablauf der Verteilungszeit selbst eine DL-PDU und es tritt in Szenario 3 eine Kollision auf. B erkennt die Kollision als erstes und sendet ein sog. JAM-Signal, um alle anderen Stationen zu warnen. Die Station A sieht das JAM-Signal kurz vor Ablauf der doppelten Verteilungszeit und kann darauf mit dem Warten und erneuten Senden gemäß CSMA/CD reagieren.

Betrachten wir hierzu die Abbildung 3-12. Die Instanz A (eine Ethernet-Station) sendet einen Ethernet-Rahmen an eine beliebige Station im Netzwerk. Die Instanz B ist physikalisch am weitesten von Instanz A entfernt. Das Erkennen einer Kollision kann bei Instanz B erst dann erfolgen, wenn das erste Bit des gesendeten Rahmens bei ihr ankommt. Bis dahin ist bereits die einfache Signallaufzeit im Medium vorüber. Entdeckt nun die Instanz B eine Kollision, so sendet es ein JAM-Signal, das aber auch wieder eine volle Signallaufzeit benötigt, bis es bei Instanz A ankommt. Erst nach der zweifachen Signallaufzeit ist eine Kollisionserkennung durch die Instanz A möglich. Das Kollisionssignal muss von der Instanz A noch während des Sendens des Ethernet-Rahmens, der die Kollision mitverursachte, empfangen werden. Andernfalls könnte die Kollision nicht erkannt werden.

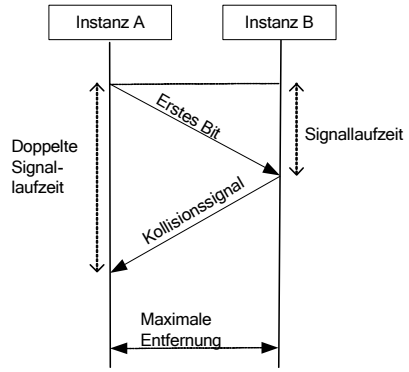


Abbildung 3-12: Kollisionsfenster bei Ethernet

Im Beispiel aus Abbildung 3-13 sendet die Station A zum Zeitpunkt t ein Datenpaket ab. Die Station B sendet ebenfalls zum Zeitpunkt t + (T - dt), also kurz bevor das Signal von A bei B ankommt ein Datenpaket. Damit nun die Station A die Kollision erkennen kann, vergeht nochmals die Zeit (T - dt), also insgesamt t + 2 * (T - dt). Wir vernachlässigen der Einfachheit halber dt (dt lassen wir gegen 0 gehen). Daraus ergibt sich, dass die Übertragung eines Datenpakets mindestens 2 * T betragen muss, damit eine Kollision sicher erkannt wird. Diese Zeitdauer entspricht der Slot-Time.

Die Signallaufzeit hängt natürlich von allen Komponenten des gesamten Übertragungswegs ab. Hierzu gehören neben den Kabeln auch Repeater. Ein Ethernet-LAN darf sich nicht beliebig ausdehnen und auch nur eine begrenzte Anzahl von Repeatern haben, ansonsten müssen die Datenpakete länger werden, um noch Kollisionen zu erkennen. Ähnliches trifft zu, wenn man die Geschwindigkeit der Übertragung heraufsetzt. Bei zunehmender Geschwindigkeit des Netzes muss die Mindestlänge entsprechend erhöht oder die maximale Kabellänge proportional verkürzt werden.

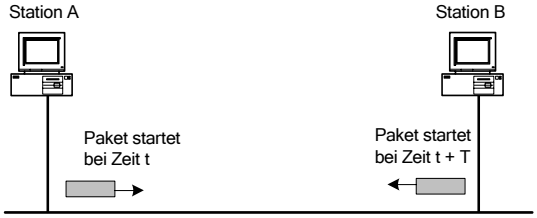


Abbildung 3-13: Signalausbreitungsgeschwindigkeit und Kollisionen

Die Länge des LAN-Segments und die Paketlänge sind also zwei wichtige Parameter, die harmonieren müssen, damit das LAN funktioniert. Bei einer maximalen Segmentlänge von 2500 m[9] (über vier Repeater möglich) muss bei einem 10-Mbit/s-Ethernet-LAN der kleinste zulässige Rahmen ca. 51.2 µs dauern. Dies entspricht den oben erwähnten 512 Bitzeiten (mit minimaler Rahmengröße von 64 Byte).

Ethernet-Varianten für verschiedene Kabeltypen. Ethernet unterstützt mehrere Kabeltypen. Historisch wurde der Kabeltyp *10Base5* verwendet. Hier handelt es sich um ein dickes Koaxialkabel (auch als Yellow Cable bekannt). „10" steht für die unterstützte Bitrate von 10 Mbit/s und „5" steht für ein maximales Segment von 500 m. Bei 10Base5 ist eine Station über ein sog. Dropkabel, das zu einem Transceiver führt, an den Bus angeschlossen.

10Base2 wurde auch als „Cheapernet" bezeichnet, da es etwas günstiger war. Es nutzt als Kabeltyp ein dünnes und damit leichter zu verlegendes Koaxialkabel, das von Station zu Station geschleift wird. Kabelbrüche oder lose Stecker sind bei *10Base5* und *10Base2* problematisch. Das Anhängen einer neuen Station führt dazu, dass das Netz unterbrochen werden muss. Für die einzelnen Varianten gibt es bestimmte Vorgaben, wie etwa die maximale Länge eines Segments oder der Mindestabstand zwischen Stationen. Abbildung 3-14 zeigt dies exemplarisch für *10Base5*.

[9] Unter Berücksichtigung der Drop-Kapel der Stationen und der Repeater sogar 2800 m (Rech 2008).

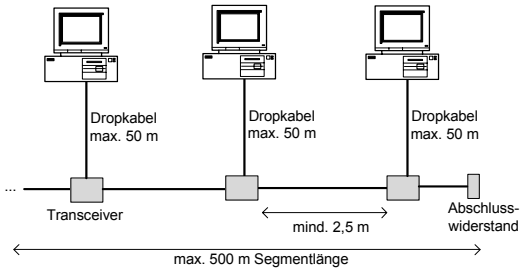


Abbildung 3-14: 10Base5-Konfiguration

Bei *10Base5* ist die maximale LAN-Länge 2500 m (nach (Rech 2008) sogar 2800 m), da man fünf Segmente über max. vier Repeater koppeln darf. Damit kann man max. 500 Stationen (100 pro Segment) miteinander verbinden.

Bei *10Base-T* stellt ein Hub den Bus dar. Alle Stationen sind mit einem max. 100 m langen, verdrillten Kupferkabel an den Hub angeschlossen. Dieses System hat gegenüber *10Base5* und *10Base2* den Vorteil, dass Kabelbrüche schnell entdeckt werden und nur noch lokale Probleme verursachen, das Netz aber weiter arbeiten kann. Abbildung 3-15 zeigt die Konfiguration eines *10Base-T*-basierten Netzes. Auch hier dürfen maximal vier Hubs anstelle von Repeatern verwendet werden.

10Base-F unterstützt schließlich als Kabeltyp ein Glasfaserkabel und wird zur sternförmigen Vernetzung von Gebäuden oder weit voneinander entfernten Hubs verwendet.

10Base5 und *10Base2* sind mittlerweile veraltet. Heute vernetzt man überwiegend sternförmig (strukturierte Verkabelung) und nutzt *10Base-T* bzw. dessen Nachfolger. Die Nachfolger werden *100Base-T (T4 und TX)* genannt und man bezeichnet diese Variante auch als *Fast Ethernet*.[10] *100Base-TX* ist heute ein Standard, wenn es um LAN-Vernetzung in Gebäuden geht.

Auch *10Base-F* hat sich zu *100Base-FX* weiterentwickelt. Bei *100Base-TX* und *100Base-FX* wird zudem die Koppelung nicht über einen Hub, sondern über einen Switch durchgeführt, was eine dedizierte Verbindung zwischen zwei Stationen ermöglicht. Dadurch wird eine Kollisionserkennung überflüssig.[11]

---

[10] Obwohl FDDI das LAN der Zukunft werden sollte, hat sich in den letzten Jahren hier die Ethernet-Technologie 100Base-T durchgesetzt und FDDI verdrängt.

[11] Bei 100VG-AnyLan verzichtet man ganz auf das Zugriffsverfahren CSMA/CD und stattet den Hub mit mehr Intelligenz für diese Aufgabe aus. Eine Station darf nur dann senden, wenn der zentrale Hub sie dazu auffordert.

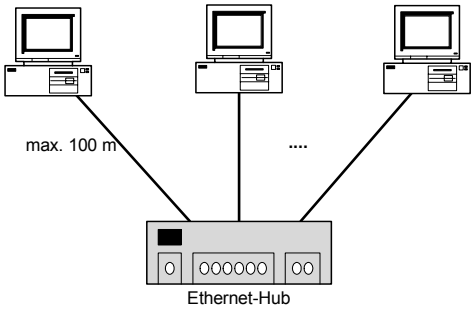


Abbildung 3-15: 10Base-T-Konfiguration

Die neueste Entwicklung ist das *Gigabit-Ethernet* mit einer Bitrate von 1000 Mbit/s und mehr (10 Gbit/s), wobei eine einfache Migration zwischen 10 Mbit/s-, 100 Mbit/s- und 1000 Mbit/s-Technologie angestrebt wurde.

Wer sich für weitere technische Daten und Parameter wie die maximale Segmentlänge oder die maximale Anzahl an möglichen Stationen für die einzelnen Ethernet-Varianten interessiert, der sei auf die Literatur (Rech 2008) verwiesen.

Ethernet-Rahmen (DL-PDUs). Der Aufbau der DL-PDUs, im Ethernet als Ethernet-Rahmen bezeichnet, ist in Abbildung 3-16 dargestellt. Die Präambel besteht aus dem Bitmuster 0b10101010 und dient der Synchronisation. Das Start-Kennzeichen (Start Frame Delimiter) besteht aus dem Bitmuster 0b10101011 und kennzeichnet den Anfang des Rahmens. Die Source- und die Destination-Adresse müssen für jede angeschlossene Station eindeutig sein. Die sechs Byte der Adresse sind nochmals wie folgt unterteilt:

- I/G-Feld (1 Bit): Individuelle oder Gruppenadresse für Multicast bzw. Broadcast.
- U/L-Feld (1 Bit): Kennzeichen, ob die Adresse vom Netzbetreiber oder der IEEE vergeben wurde.
- Herstellerkennung (22 Bit): Kennzeichnet den Hersteller der Ethernet-Karte eindeutig.
- Gerätenummer (24 Bit): Laufende Nummer, die der Gerätehersteller vergibt.

Im Längenfeld steht die Anzahl der Byte im anschließenden Nutzdaten-Bereich. Im Padding-Feld wird der Rahmen auf mindestens 64 Byte aufgefülllt, falls er kleiner ist. Hier werden die Präambel und der Start Frame Delimiter nicht mitgezählt, da diese Felder nur zur Synchronisierung und zur Kennzeichnung eines beginnenden Rahmens dienen und nicht zum eigentlichen Rahmen gehören.

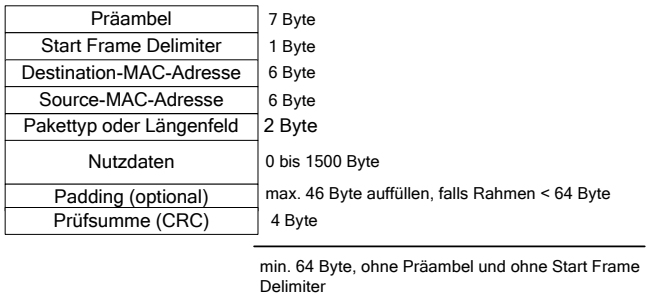


Abbildung 3-16: Ethernet-Rahmenformat

Prüfsummenberechnung. Die Prüfsumme wird nach dem CRC-Verfahren (Cyclic Redundancy Check) berechnet. Das Verfahren basiert auf einem Polynom vom Grade k–1. Es stellt im Prinzip einen 32-Bit-Hashcode der Nutzdaten dar. Wenn einige Bit wegen einer Kanalstörung im Kabel verfälscht werden, dann ist die Prüfsumme fast immer falsch. Der Empfänger merkt dies, indem er die Prüfsumme ebenfalls berechnet und sie mit der gesendeten vergleicht. Die Polynomberechnung wird mit XOR (exclusive OR) ohne Übertrag durchgeführt. Am Anfang der Polynomberechnung werden k Nullbit an die niederwertige Seite des Ethernet-Rahmens gehängt. Anschließend erfolgt anhand der Bitkette die Polynomberechnung.

Das Verfahren soll an folgendem Beispiel aus (Tanenbaum 2003a) verdeutlicht werden:

- Die Bitsumme des Ethernet-Beispielrahmens sei 0b11011011011.
- Das verwendete Polynom sei 10011 = $x^4 + x^1 + x^0$ (Bitposition auf 1 → Potenz geht in das Polynom ein).
- Ein Anhängen von k = 4 Nullbit ergibt: 110110110110**0000**.
- Die Berechnung der Prüfsumme:
  - 110110110110000 : 10011 = **1100001010, Rest 1110**.
- Übertragen wird der Ursprungsrahmen und der Rest, insgesamt also 0b110110110111**1110**.

Auf die Darstellung der exakten Berechnung der Prüfsumme soll hier nicht eingegangen werden. Sie kann in der angegebenen Literatur nachgelesen werden.

Leitungskodierung. Für die Leitungskodierung wird im Ethernet-Standard die *Manchester-Kodierung* verwendet. Jede Bit-Periode wird in zwei gleiche Intervalle unterteilt. Eine binäre Eins wird gesendet, indem die Spannung im ersten Intervall hoch und im zweiten Intervall niedrig gesetzt wird, bei der binären Null ist es umgekehrt. Im Vergleich zur direkten Binärkodierung wird also die doppelte Bandbreite benötigt. Als High-Signal wird 0,85 Volt und als Low-Signal –0,85 Volt verwendet. Ab dem Fast Ethernet wird die Manchester-Kodierung nicht mehr ver-

wendet, da eine Verdoppelung der Signalgeschwindigkeit bei höheren Datenraten sehr aufwändig ist. Im Gigabit-Ethernet wird beispielsweise das 8B/10B-Verfahren verwendet, in dem jeweils 8 Bit binäre Nutzdaten zur seriellen Übertragung in 10 Bit lange Kode-Gruppen umgewandelt werden (Rech 2008).

LAN-Switching. Switches, auch als LAN-Switches oder Switching-Hub bezeichnet nutzen im Gegensatz zu den anderen Ethernet-Techniken kein „shared" Medium mehr. Jede Station, die am Switch angeschlossen ist, erhält eine dedizierte Anschlussleitung. Die Stationen werden für die Dauer der Kommunikation jeweils paarweise miteinander verbunden.

Man unterscheidet *Store-and-Forward-Switches* und *Cut-Through-Switches*. Ein Store-and-Forward-Switch nimmt einen ganzen Rahmen auf, analysiert ihn und gibt ihn weiter. Ein Cut-Through-Switch gibt den Rahmen sofort nach der Identifikation der fast am Anfang des Rahmens stehenden Zieladresse an die adressierte Station weiter. Vorteil von Store-and-Forward-Switches ist, dass er fehlerhafte Rahmen erkennen kann. Dies ist bei Cut-Through-Switches nicht möglich. Dafür haben Cut-Through-Switches einen Leistungsvorteil. Heute beherrschen Switch-Produkte meist beide Varianten.

In Abbildung 3-17 ist ein LAN-Switch dargestellt. Ein Kreuzschienenverteiler im Inneren des Switches sorgt für das Schalten („Switching") der Verbindungen.

Diese Technik ermöglicht es auch, dass die für Ethernet-LANs typischen Kollisionen praktisch nicht mehr auftreten. Dadurch wird das Ethernet-LAN mit Switching-Technik auch für Echtzeitanwendungen immer interessanter (Popp 2004).

Die Switching-Technik entwickelte sich in den letzten Jahren insbesondere im TCP/IP-Umfeld sehr stark weiter. Daher kann man keine klaren Schichtengrenzen, wie diese im OSI-Modell vorgegeben sind, mehr ziehen. Switching auf Layer 3 bis 7 bedeutet, dass höhere Schichten in die Switching-Entscheidung einbezogen werden. Das heißt die Switching-Entscheidung muss, unabhängig von der Geschwindigkeit des Interfaces, zusätzliche Entscheidungskriterien der Schichten 3 bis 7 berücksichtigen. Im einfachsten Fall wird die Weiterleitungsentscheidung auf der Schicht 3 behandelt. Hier spricht man von einem *Layer-3-Switch*. Im Paket auf der Schicht 3 kann aber ebenso eine Information über die Priorität enthalten sein, mit der das Paket zu übertragen ist. Stand der Technik ist es bereits heute, Weiterleitungsentscheidungen mit Hilfe von Kriterien der Schichten 3 und 4 zu unterstützen. Mehrere Hersteller haben hierzu Verfahren in unterschiedlicher Ausprägung implementiert. Wenn auch noch Kriterien von Schicht-7-Protokollen berücksichtigt werden, spricht man von *Application-Switches*.

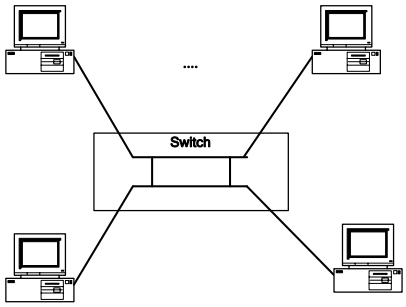


Abbildung 3-17: LAN-Switch

Virtuelle LAN. Als VLAN (Virtual Local Area Network) bezeichnet man ein logisches Netzwerk innerhalb eines physikalischen Netzwerkes. Man kann mit dieser Technik innerhalb eines physikalischen Netzwerks mehreren Benutzergruppen oder Organisationseinheiten eigene (virtuelle) LAN zuordnen.

Es gibt mehrere Realisierungsvarianten für VLANs. Beispielsweise können in der Schicht 2 MAC-Adressen oder Ports von intelligenten Hubs oder LAN-Switches zu Gruppen zusammengefasst werden[12]. Intelligente Hubs können beispielsweise im internen Bus Zeitschlitze im Zeitmultiplexverfahren auf verschiedene VLANs zuordnen. Diesen einzelnen Ports werden dabei die Zeitschlitze zugeordnet. Ähnliches ist mit LAN-Switches (sog. VLAN-taugliche Switches) möglich. In Abbildung 3-18 ist ein Beispiel für zwei VLANs innerhalb von zwei Layer-2-Switches durch Portzuordnung gegeben. Jeder Switch hat im Beispiel acht Ports. Die Stationen S1, S3, S5, S6 und S7 bis S14 sind dem VLAN 1 zugeordnet, die restlichen Stationen sind Mitglieder des VLAN 2. Die Switches sind miteinander verbunden. Ein Netzwerkadministrator muss die Ports den VLANs manuell zuordnen. Sollen Stationen der beiden VLANs miteinander kommunizieren, ist in der Schicht 3 ein Router erforderlich, da sie vollständig voneinander isoliert sind.

Auch in der Schicht 3 und in der Schicht 4 kann die VLAN-Gruppierung vorgenommen werden. Welche Station zu welchem VLAN gehört, kann auch über den Inhalt der Nachrichten festgestellt werden. Hier sind je nach Produkt mehrere Methoden möglich. Man kann beispielsweise alle Ethernet-Stationen mit einem bestimmten Wert im Feld Pakettyp einem VLAN zuordnen. Auf der Schicht 3 könnte man z.B. alle IP-Adressen eines Subnetzes einem VLAN zuordnen. In der Schicht 4 ist die Zuordnung über Transportadressen (z.B. TCP/UDP-Ports) mög-

[12] Siehe z.B. den Cisco VLAN Membership Policy Server (VMPS) oder das GARP (Generic Attribute Registration Protocol) VLAN Registration Protocol (GVRP).

lich[13]. Die Auswertung der Pakete ist allerdings mit relativ viel Aufwand verbunden.

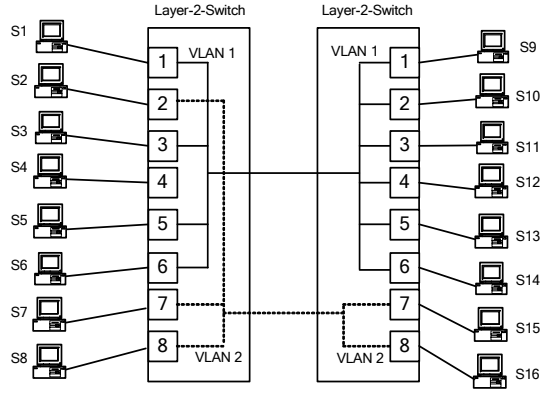


Abbildung 3-18: VLAN-Realisierung auf Basis von Portnummern

Die VLAN-Technik funktioniert auch in Ethernet-LANs mit der LAN-Switching-Technik. Der entsprechende IEEE-Standard heißt 802.1Q[14]. Jedem VLAN wird eine eindeutige Nummer, die sog. VLAN-Id zugeordnet. Gemäß dem IEEE-Standard 802.1Q wird hierfür das Ethernet-Paket um 4 Byte erweitert. Davon ist ein 12 Bit langes Feld zur Aufnahme der VLAN-Id vorgesehen, so dass insgesamt 4094 VLANs möglich sind. Die VLAN-Ids 0 und 4095 sind reserviert. Der erweiterte Ethernet-Rahmen ist in Abbildung 3-19 skizziert. Das VLAN-Feld hat wiederum vier Einzelfelder:

- Das Feld *TPID* (Tag Protocol Identifier) ist für das Ethernet belanglos und hat einen festen Wert 0x8100. Es wird für die Übertragung über andere Technologien wie z.B. Token Ring benötigt.
- Das Feld *User-Priorität* dient der Kennzeichnung zeitkritischer Frames
- Das Feld *CFI* (Canonical Format Indicator) wird auch nur für die Kommunikation über Token-Ring benötigt.
- Die *VLAN-Id* dient der Identifikation eines bestimmten VLAN.

Ein Gerät, das zum VLAN mit der VLAN-Id = 1 gehört, kann mit jedem anderen Gerät im gleichen VLAN kommunizieren, nicht jedoch mit einem Gerät in einem anderen VLAN. Die Kommunikation zwischen VLAN ist nur über Router möglich.

---

13 IP-Adressierung (Vermittlungsschicht) und TCP/UDP-Adressierung (Transportschicht) wird noch in den Kapiteln 4 und 5 erläutert.

14 Die Vorarbeiten leistete die Firma Cisco im De-Facto-Standard ISL (Intel Switch Link Protokoll).

Empfängt ein VLAN-fähiger Switch von einem Port einen Ethernet-Rahmen, in dem er ein VLAN-Tag erkennt, so wird dieser an den adressierten Zielport weitergeleitet. Falls sich an diesem Port ein Endgerät befindet, wird zuvor das Tag vom Switch entfernt und ein normaler Ethernet-Rahmen weiter gesendet. Umgekehrt wird einem Ethernet-Rahmen, der von einem Endgerät gesendet wurde, im Switch das VLAN-Tag hinzugefügt. Es gibt aber auch bereits Netzwerkkarten für Endgeräte, die diese Aufgabe erledigen.

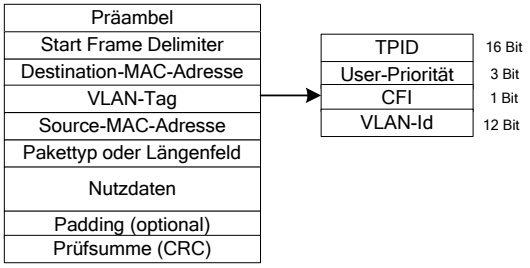


Abbildung 3-19: Ethernet-Rahmenformat mit VLAN-Erweiterung

3.2.4 Wireless LAN (WLAN)

Die Verbreitung von WLAN nach der Norm IEEE 802.11 nimmt in den letzten Jahren stark zu. Vor allem ermöglicht es in Hotels, Restaurants, Flughäfen usw. eine ideale Variante, Kunden mit eigenen Notebooks einen Netzzugang bzw. Internet-Zugang anzubieten. Aber auch z.B. für Handels-Unternehmen (Versand- und Großhandel mit Zentrallagern) ist ein Einsatz von WLAN in Lagersystemen etwa für mobiles Kommissionierpersonal gut geeignet. Da heutzutage oft WLAN-Router vom DSL-Anbieter bei einem Neuvertrag ausgeliefert werden, wächst die Verbreitung von WLAN auch im privaten Bereich.

In drahtlosen LANs nach IEEE 802.11 wird die modifizierte Form des CSMA/CD-Verfahrens eingesetzt, das als CSMA/CA bezeichnet wird. Die Abkürzung „CA“ steht für Collision Avoidance, also (möglichst) Vermeiden von Kollisionen oder zumindest die Wahrscheinlichkeit von Kollisionen gering halten. Das Medium wird hierzu abgehört, und es wird erst übertragen, wenn es frei ist. Zusätzlich wird ein Quittierungsverfahren eingesetzt. Von einer Station wird vor dem Senden einer Nachricht zunächst eine kurze Steuernachricht abgeschickt, die vom Empfänger bestätigt wird. Erst nach dem Empfang der Bestätigung beginnt der Sender die eigentlichen Daten zu senden. Alle anderen Stationen des Antennenbereichs warten nach Empfang der Steuernachricht auf das tatsächliche Datenpaket.

Das WLAN nach 802.11 kann in zwei grundlegenden Varianten realisiert werden. Man unterscheidet Infrastruktur- und Ad-hoc-Netzwerke:

Infrastruktur-Modus. In diesem Modus kommunizieren mehrere WLAN-Stationen über einen festen Zugangspunkt, der als Access-Point bezeichnet wird. Alle

Stationen und der für sie zuständige Access-Point werden zusammen als Basic Service Set (BSS) bezeichnet. Die Access-Points können auch den Zugang zu einem Festnetz verschaffen.

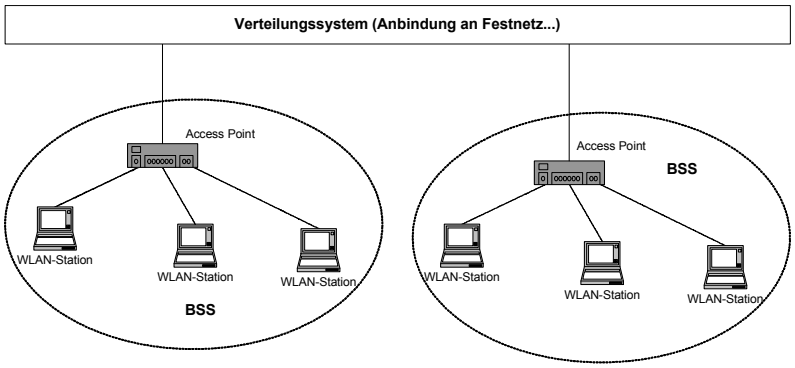


Abbildung 3-20: Typische WLAN-Vernetzung in Infrastruktur-Modus

Ad-hoc-Modus. Im Ad-hoc-Modus kommunizieren mehrere WLAN-Stationen untereinander direkt, also ohne Access Point in einem sog. IBSS (Independent Basic Service Set). Ein IBSS umfasst hier alle Stationen im selben Übertragungsbereich. In Abbildung 3-20 ist der Infrastrukturmodus mit zwei BSS skizziert, die über ein Verteilungssystem miteinander verbunden sind (Festnetz). In Abbildung 3-21 ist der Ad-hoc-Modus beispielhaft mit drei WLAN-Stationen in einem IBSS dargestellt.

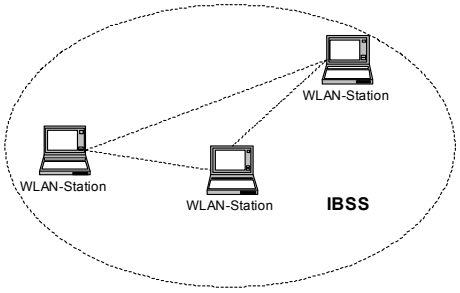


Abbildung 3-21: WLAN nach 803.11 im Ad-hoc-Modus

Funkkommunikation und Frequenzband. Die 802.11a-Norm ist für ein Frequenzband von 2,4 GHz vorgesehen und erreicht eine Bitrate von max. 11 Mbit/s. Die neue Variante 802.11g erreicht Datenraten von 54 Mbit/s und die Entwicklung ist hier noch nicht abgeschlossen.

Es soll noch bemerkt werden, dass Funkkommunikation wesentlich störanfälliger als die drahtgebundene Kommunikation ist. Kabel kann man gut abschirmen, das Funkmedium nicht. Funk ist sehr anfällig für jedwede elektromagnetische Störung und auch für Wettereinflüsse. Auch sind die Datenraten bei der Funkkommunikation wesentlich niedriger. Frequenzen sind knappe Ressourcen[15].

Die Entfernung, in der ein Signal noch empfangen werden kann, hängt von der Frequenz ab. Je höher die Frequenz, desto geringer ist die Reichweite und auch die Dämpfung nimmt zu. Schließlich muss bei der Funkkommunikation noch berücksichtigt werden, dass sich zwischen Sender und Empfänger meist irgendwelche Hindernisse befinden (Gebäude,....) und keine direkte Sichtverbindung vorhanden ist. Dies trifft besonders auf Städte zu. Je höher die Frequenz ist, desto mehr verhält sich das Signal wie Licht. Signale mit niedriger Frequenz durchdringen Objekte besser als Signale mit hoher Frequenz. Das Signal muss also ggf. mehrmals reflektiert werden, bis es beim Empfänger ankommt, und verliert dabei je nach Oberflächenbeschaffenheit des Hindernisses an Stärke, da es teilweise absorbiert wird.

3.3 Zugang zu öffentlichen Netzen (WAN-Technologien)

3.3.1 ISDN

ISDN steht für Integrated Services Digital Network und integriert verschiedenartige Kommunikationsdienste wie Sprache, Daten- und Bildkommunikation innerhalb eines Netzwerks über einen Netzwerkzugang. ISDN ist ein *digitales leitungsvermitteltes* System und wird überwiegend für die Telefonie verwendet. Es wird auch als *Schmalband*-ISDN bezeichnet. ISDN spielt in Deutschland und mit Euro-ISDN auch in Europa eine wichtige Rolle. In Deutschland wird es flächendeckend eingesetzt. Alle EU-Staaten verfügen ebenfalls über eine ISDN-Infrastruktur. In den USA wird ISDN allerdings nicht eingesetzt. Der Telekom-Anbieter AT&T bietet z.B. einen ISDN-ähnlichen Dienst unter dem Namen 5ESS in den USA an.

ISDN nutzt das PCM-Verfahren (Pulse Code Modulation). PCM bildet die Grundlage für einige Kommunikationssysteme und ist der Standard für die digitale Kodierung von Audiodaten im Telefonnetz. Auf Basis der Grundeinheit von 64 Kbit/s wurde eine sog. Übertragungshierarchie definiert, in denen jeweils mehrere Kanäle multiplexiert werden können. Die PCM-Übertragungshierarchie wurde bereits in Kapitel 2 dargestellt. In PCM-30 sind dann z.B. 30 * 64 Kbit/s + 2 * 64 Kbit/s (Steuerinformation) = 2,048 Mbit/s verfügbar. Entsprechend gibt es PCM-120 mit 8,448 MBit/s usw. Die PCM-Hierarchie wird für den internen Aufbau des ISDN-Netzwerkes genutzt.

[15] Die einheitliche Vergabe von Frequenzen regelt weltweit die ITU und in Europa die European Conference for Posts an Telecommunications (CEPT).

ISDN-Netzwerk. Das ISDN-Netzwerk ist hierachisch aufgebaut. Man unterscheidet digitale Fernvermittlungen (das sind Knotenrechner) und darunter digitale Ortsvermittlungen. Die Ortsvermittlungen hängen an den Fernvermittlungen, die wiederum auf höchster Ebene miteinander vernetzt sind. An den Ortsvermittlungen hängen schließlich die Telefonanbindungen für die Hausanschlüsse von Privatkunden bzw. von Unternehmen (siehe Abbildung 3-22). Zwischen einer digitalen Ortsvermittlung (Ortsvermittlungsstelle oder OvSt) und einer digitale Fernvermittlung wird z.B eine PCM-30-Verbindung genutzt, zwischen zwei digitalen Fernvermittlungen eine PCM-7680-Verbindung.

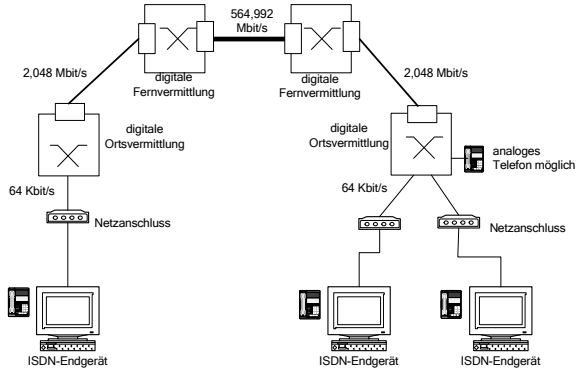


Abbildung 3-22: ISDN-Konfiguration nach (Zitterbart 1995)

Hausintern wird ein sog. S_0-Bus verlegt, an den unmittelbar ISDN-fähige Geräte (Telefone, Rechner mit ISDN-Adaptern, Faxgeräte) angeschlossen werden können. Ein S_0-Bus besteht aus vier Drähten, jeweils zwei je Senderichtung. Mit Hilfe eines Terminaladapters (TA) können an den S_0-Bus auch analoge Geräte angeschlossen werden.

Der Telekombetreiber (Carrier) installiert beim Endkunden eine Dose, den NT-Netzanschluss (auch NT1[16] genannt). Zwischen Endkunden und Ortsvermittlung werden verdrillte Kupferkabel verwendet (letzte Meile). Ein Endgerät (Rechner oder Telefon) wird vor Ort über einen ISDN-NT an das ISDN-Netz angeschlossen. Aus der Sicht des Endkunden ist der NT-Netzanschluss die Grenze zum öffentlichen Netz und damit der Netzwerkzugang. Unternehmen hängen meist eine Nebenstellenanlage (PBX = Private Branch eXchange) an den NT-Netzanschluss. Die PBX bildet dann den sog. NT2-Anschluss.

[16] NT steht für Network Termination. In Deutschland wird dieser Anschluss als NTBA (Netzterminator-Basisanschluss) bezeichnet.

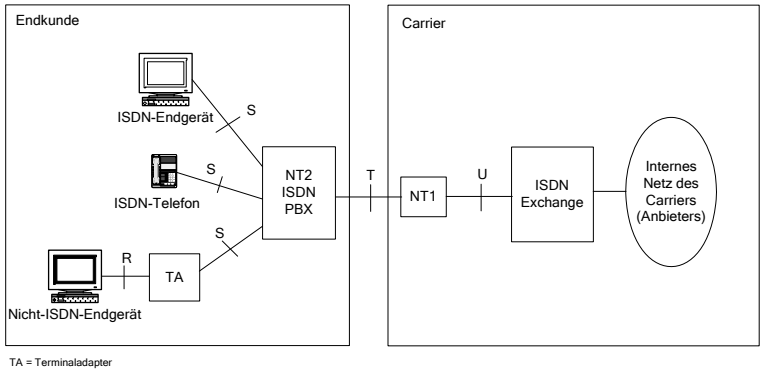


Abbildung 3-23: ISDN-Referenzkonfiguration

Die CCITT definiert verschiedene Schnittstellen, die als Bezugspunkte R, S, T und U bezeichnet werden (siehe Abbildung 3-23):

- Die T-Schnittstelle liegt im NT1-Stecker.
- Die S-Schnittstelle liegt zwischen einer ISDN-PBX und einen ISDN-Endgerät.
- R dient als Schnittstelle zu einem Terminaladapter (TA) und Endgeräten ohne ISDN.
- U ist die Schnittstelle zwischem den ISDN-Netzbetreibern und dem NT1-Anschluss.

ISDN-Schichtenmodell. Das ISDN-Schichtenmodell unterscheidet die verschiedenen Protokollstacks in den Endsystemen, in den Ortsvermittlungen und in den Fernvermittlungen. Weiterhin wird die Übertragung von Nutzdaten und von Signalisierungsinformation unterschieden. Für alle verwendeten Protokolle gibt es Empfehlungen der ITU. Zur Signalisierung zwischen den Fernvermittlungs-Instanzen wird das Signalisierungssystem SS7[17] (Signalling System Nr. 7) verwendet. SS7 und dessen Protokollstack sollen hier nicht weiter betrachtet werden. Vielmehr konzentrieren wir uns auf die Kommunikation zwischen den Endgeräten und den Ortsvermittlungs-Instanzen.

Schicht 1. Die Arbeitsweise der Schicht 1 wird durch die Spezifikationen I.430 und I.431 festgelegt. Die *ISDN*-Rahmen auf der physischen Ebene haben eine Länge von 48 Bit. Ein Rahmen wird in 250 µs übertragen. Von den 48 Bit sind 36 Datenbit, womit sich eine Datenrate von 144 Kbit/s errechnet. Zwischen einem empfangenen und einem gesendeten Rahmen liegt eine Zeitverzögerung von zwei Bit. Der ISDN-Rahmen ist in Abbildung 3-24 dargestellt. Ohne auf die Einzelbit näher ein-

[17] In Europa auch als C7 bekannt.

zugehen, wird deutlich, dass alle 250 µs im Zeitmultiplex-Verfahren 16 Bit je B-Kanal und 4 Bit für den D-Kanal gesendet werden.

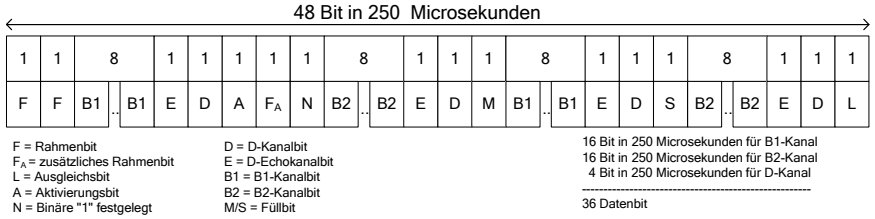


Abbildung 3-24: Schicht-1-Rahmen bei ISDN nach (Sikora 2003)

Schicht 2. Die Schicht 2 folgt den Empfehlungen Q.920 und Q.921 der ITU-T. Diese Schicht sorgt für eine zuverlässige Übertragung der DL-Rahmen und für das Verbindungsmanagement zwischen Endgerät und Vermittlungsstelle. Das Protokoll wird auch als LAPD (HDLC-Variante) bezeichnet. LAPD (Link-Access-Procedure-D-Kanal) übernimmt die Aufgaben der Rahmenbegrenzung und der Reihenfolgezuordnung, der Fehlererkennung und der -korrektur. Weiterhin wird eine Flusssteuerung durchgeführt. Der Aufbau der ISDN-LAPD-Rahmen ähnelt dem oben beschriebenen HDLC-Rahmen stark.

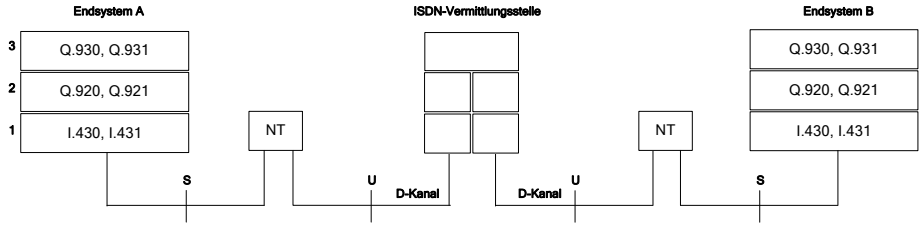


Abbildung 3-25: ISDN-Protokollstack nach (Sikora 2003)

In einem S_0-Bus ist der D-Kanal (4 D-Bit) für alle angeschlossenen Endgeräte nur einmal vorhanden. Der Zugriff auf den D-Kanal wird über Zugriffsprioritäten gesteuert. Ein Endgerät, das senden möchte, prüft vorher den Kanal, in dem es eine vorgegebene Anzahl an Einsen ausliest. Die Priorität einer Station nimmt mit zunehmender Anzahl an Einsen, die es vor dem Senden lesen muss, ab. Eine Station, die gesendet hat, erhöht die Anzahl der zu erkennenden Einsen um einen vorgegebenen Wert, damit die anderen Stationen als nächste das Medium erhalten können. Die Priorität wird wieder erhöht, sobald die eingestellte Anzahl an Einsen hintereinander am Bus erkannt wurden. Wenn mehr als ein Endgerät sendet, werden auftretende Kollisionen erkannt, indem jedes gesendete Bit auch wieder über das E-Bit gelesen wird. Sobald ein Endgerät eine Abweichung feststellt, beendet es das Senden und warten, bis es wieder an der Reihe ist.

Schicht 3. In der Schicht 3[18] werden die ITU-T-Empfehlung Q.930 und Q.931 verwendet. Die Protokolle sind zuständig für den Gesprächsaufbau zwischen den ISDN-Endsystemen sowie für die Kontrolle und für den Zugang zu den angebotenen Diensten. Weiterhin werden hier die ISDN-PDUs für die Kommunikation (ISDN-Meldungen genannt) definiert. Typische PDUs sind die CONN-PDU zum Aufbau der Verbindung, die CONNACK-PDU zur Verbindungsbestätigung und die DISC-PDU zum Einleiten eines Verbindungsabbaus.

Alle Q.x-Protokolle bilden das sog. *Digital Subscriber Signalling System No.1 (DSS1)*, das europaweit genutzt wird und auch als Euro-ISDN bezeichnet wird. Die Protokollstacks sind in Abbildung 3-25 skizziert. Weitere Details zu den Protokollen sind der Literatur zu entnehmen.

Die Telekom-Gesellschaften bieten verschiedene Dienstzugänge an. Der einfache Basisanschluss zur Ablösung des analogen Telefons bietet zwei B-Kanäle mit einer Bitrate von je 64 Kbit/s) und einen D-Kanal (Bitrate16 Kbit/s) für die Steuerung (2B+1D). Er wird auch als *S_0-Anschluss* bezeichnet und stellt eine Datenrate von 144 kbit/s bereit. Daneben wird noch der sog. Primärmultiplexkanal mit 30 Kanäle und einen D-Kanal mit 64 Kbit/s in Europa angeboten (30B+1D). Für einen Primärmultiplexanschluss (auch *S_{2M}-Anschluss* genannt) kann intern vom Carrier eine PCM-30-Verbindung verwendet werden. Der noch freie 32. 64-Kbit/s-Kanal wird u.a. für das Netzwerkmanagement und zur Synchronisation verwendet. Damit erreicht ein S_{2M}-Anschluss eine Datenrate von 2.048 kbit/s.

Abschließend sei noch bemerkt, dass der Standardisierungsprozess für ISDN ziemlich lange gedauert hat (seit 1984) und ISDN heute – trotz weiter Verbreitung – schon wieder als veraltet gilt.

3.3.2 DSL

Ein weiterer WAN-Zugangsdienst ist DSL. DSL steht für *Digital Subscriber Line* (digitale Teilnehmeranschlussleitung). DSL nutzt die gleichen Kupfer-Doppeladern (vier Drähte) wie ISDN, also herkömmliche Telefonleitungen, kann aber eine wesentlich höhere Datenrate erzielen. Man unterscheidet mehrere DSL-Verfahren und spricht oft auch insgesamt über die xDSL-Verfahren.

xDSL ist sehr gut für die Internet-Nutzung geeignet, (wenn man das typische Web-Surf-Verhalten betrachtet) und wird von den Telekomanbietern auch als Internet-Zugangsdienst angeboten. ADSL wird z.B. über den vorhandenen ISDN-Zugang angeboten, wobei Telefonieren und Datenübertragung gleichzeitig über dieselben

[18] Man sieht also, dass die ISDN-Architektur in einem 3-Schichtenmodell beschrieben ist. Wie wir noch sehen werden, ist z.B. in der TCP/IP-Architektur ein ISDN-WAN-Zugang in der Schicht 2 angesiedelt. Dies ist durchaus nichts Ungewöhnliches, da hier völlig eigenständige Architekturen vorliegen und ein IP-Netzwerk ISDN nur als Transportnetzwerk benutzt.

Kabel möglich ist. Man unterscheidet bei xDSL symmetrische und asymmetrische Verfahren, wobei bei letzteren die Downstream-Übertragung (vom Internet Service Provider zum Endkunden und Upstream-Übertragung für den umgekehrten Weg differenziert wird.

Beispiele für xDSL:

- ADSL steht für *Asymmetrical* DSL und verfügt über eine Downstream-Rate bis zu 8 Mbit/s. Im Upstream wird bis zu 640 Kbit/s erreicht. Man kann mit ADSL Distanzen von 2,7 bis 5,5 km bedienen. Diese Asymmetrie ist also durch die unterschiedlichen Bitraten gegeben. Weiterentwicklungen von ADSL (ADSL2, ADSL2+) ermöglichen sogar noch deutlich höhere Datenraten bei größeren Distanzen (Downstream: 25 Mbit/s, Upstream: 3 Mbit/s). Weiterentwicklungen von ADSL sind ADSL2 und ADSL2+. ADSL2 erweitert bei Kompatibilität zu ADSL die maximale Downstream-Rate auf 16 Mbit/s, ADSL2+ auf 25 Mbit/s[19].
- SDSL steht für *Symmetrical* DSL. Hier erreicht man sowohl „Downstream" als auch „Upstream" Datenraten von über 2 Mbit/s[20].
- VDSL steht für *Very High Bit Rate* DSL. Hier erreicht man im Downstream-Datenraten in der Größenordnung von 52 Mbit/s.

Man muss beachten, dass bei den einzelnen Verfahren die Datenrate und die Distanz umgekehrt proportional zueinander sind. VDSL verfügt z.B. über eine sehr hohe Datenrate bei kurzen Entfernungen.

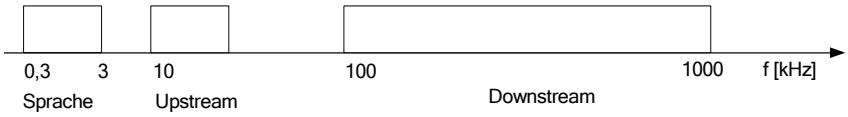


Abbildung 3-26: Frequenzbelegung bei ADSL nach (Sikora 2003)

xDSL-Anschlüsse bieten Vollduplexübertragung z.B. über ein Frequenzmultiplexverfahren. Bei ADSL sieht z.B. die Frequenzbelegung wie in Abbildung 3-26 dargestellt aus. Man sieht in der Abbildung, dass für Sprache, Upstream und Downstream disjunkte Frequenzbereiche reserviert sind.

Ein ADSL-Anschluss benötigt zwei Modems (auch ATU = ADSL Terminal Unit genannt) und zwei sog. Splitter (auch BBAE = Breitbandanschlusseinheit), die über eine Zweidrahtleitung miteinander verbunden sind. Ein Splitter ist im Hausanschluss und einer in der Vermittlungsstelle. Splitter sind *Frequenzweichen* und teilen den Frequenzbereich des Telefondienstes vom ADSL-Frequenzbereich ab. In Abbildung 3-27 ist ein typisches Beispiel eines ADSL-Zugangs für ein Hausnetz

[19] ADSL2 bzw. ADSL+ gelten auch als Voraussetzung für das Internet-Fernsehen (IPTV).

[20] Die von den Telekom-Betreibern angebotenen Datenraten erhöhen sich ständig.

beschrieben. An die ATU ist hier im Hausanschluss ein Ethernet-Hub angeschlossen, an den wiederum die einzelnen Rechner angeschlossen sind.

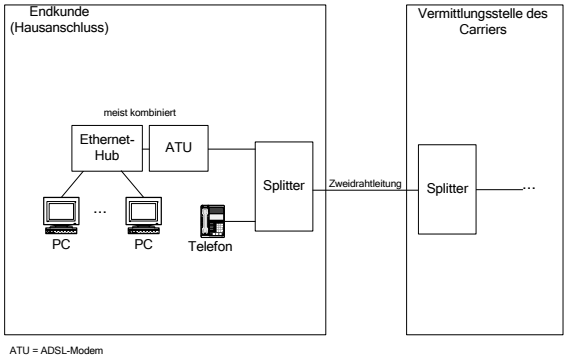


Abbildung 3-27: ADSL-Anschluss-Beispiel

Eine Besonderheit ist bei ADSL noch zu erwähnen. Die Modulationsmethode wird dynamisch ermittelt. Wenn die verwendete Frequenz über eine hohe Qualität verfügt, wird eine Modulationsmethode gewählt, die mehr Bit pro Baud kodiert. Bei niedriger Qualität werden entsprechend weniger Bit pro Baud kodiert.

Hinweis: Heute nutzt man für den Anschluss seines Heim-PCs ADSL-Modems, die neben der Modemfunktionalität auch mit Firewall-Technik, IP-Router-, DHCP- und NAT-Funktionalität (siehe Kapitel 4), Ethernet-Hub, usw. ausgestattet sind, also über eine komplette Anbindungsmöglichkeit ans Internet verfügen. Auch Kombinationen von ADSL-Modems mit WLAN-Technologien sind üblich.

3.3.3 PDH, SDH und SONET

In der Schicht 1 sind neben PCM (Pulse Code Modulation) weitere Multiplex-Techniken mit digitalen Übertragungshierarchien für Weitverkehrsverbindungen angesiedelt, die man hinsichtlich des zugrundeliegenden Übertragungsverfahrens in *synchrone*, *asynchrone* und *plesiochrone* (chronos = Zeit) Verfahren einteilt.

Übertragungsverfahren allgemein. Bei der synchronen Übertragung wird zwischen Sender und Empfänger der gleiche Takt für alle zu übertragenden Bit eingehalten, bei der asynchronen Übertragung gibt es keinen gemeinsamen Takt. Bei der synchronen Übertragung wird der Takt aus dem empfangenen Datensignal regeneriert, was beim Empfänger recht aufwändig ist.

Bei der asynchronen Übertragung schwingen zwar die Taktgeneratoren von Sender und Empfänger unterschiedlich und unabhängig voneinander, aber man synchronisiert nach jedem gesendeten Datenwort neu. Das kann man z.B. so bewerkstelligen, dass jede Gruppe von Nutzdaten-Bit durch sog. Synchronisations-Bit eingerahmt ist (Start- und Stop-Bit). Plesiochrone Übertragung ist fast, aber doch

nicht ganz synchron (Plesio steht für „fast“), das heißt, es ist ein leichtes Abweichen des Takts zwischen Quelle und Senke möglich, das auszugleichen ist.

Zwei wichtige Multiplex-Techniken mit Übertragungshierarchien sind PDH (Plesiochronous Digital Hierarchie) und SDH (Synchronous Digital Hierarchy). Insbesondere das europäische SDH, das in Nordamerika als SONET (Synchronous Optical NETwork) bezeichnet wird, ist ein modernes Transportverfahren für Weitverkehrsnetze. PDH und SDH/SONET sollen hier kurz eingeführt werden. Für eine weiterführende Erläuterung wird auf (Tanenbaum 2003a) verwiesen.

PDH-Technologie. Die seit den 60er Jahren entwickelten PDH-Netze benutzen ein TDM-Verfahren (Time Division Multiplexing, Zeitmulitplexverfahren). Die PDH-Hierarchie ist in Tabelle 3-1 dargestellt. Heute sind in Europa vorwiegend E1, E3 und E4 relevant, E5 wird bereits durch die SDH-Technologie verdrängt und E2 wird gerne übersprungen. Leider sind für PDH keine weltweiten Standards verfügbar. Die angebotenen Bitraten in den USA und in Japan unterscheiden sich von den europäischen.

Tabelle 3-1: PDH-Hierarchie in Europa

Bezeichnung nach ITU-T	Genaue Bitrate [Mbit/s]
E1	2,048 Mbit/s
E2	8,448 Mbit/s
E3	34,368 Mbit/s
E4	139,264 Mbit/s
E5	565,148 Mbit/s

PDH war nie ein weltweiter Standard. Nachteilig an PDH ist außerdem das komplizierte Verfahren, mit dem man einzelne Kanäle aus einer PDH-Hierarchie „herausfischen“ muss. Das Signal muss nämlich aufgebrochen werden, wobei alle Multiplexierungsstufen zu durchlaufen sind, um den Kanal zu lesen. Anschließend ist das Signal wieder neu zu multiplexieren. Dies liegt daran, dass PDH für jede Multiplexstufe einen eigenen Rahmen definiert. Das Multiplexen erfolgt zudem bitweise.

SDH- und SONET-Technologie. Die SDH-Technologie wurde 1990 durch die ITU-T festgelegt und behebt die PDH-Nachteile weitgehend. SDH entstand ursprünglich aus Forschungsarbeiten in den amerikanischen Bell-Labors und zwar dort unter dem Namen SONET. SDH ist für die Übertragung über *Lichtwellenleiter* (LWL) konzipiert und im Gegensatz zu PDH weitgehend in Software (in den Netzwerkkomponenten) realisiert. Die Vorteile gegenüber PDH sind:

- SDH ist ein weltweit gültiger Standard über alle Hierarchiestufen.

- Direkte Entnahme der einzelnen Zubringersignale, also der Zugriff auf einen einzelnen Kanal innerhalb der Multiplexierungshierarchie. SDH ist also wesentlich einfacher als PDH.
- Es wird byteweise multiplexiert.

SDH benötigt als synchrone Übertragungstechnik eine zentrale Taktversorgung, was jeweils landesweit realisiert wird.

Man kann sich vorstellen, dass die Nachrichtenzusammenstellung bei SDH vergleichbar mit Abläufen in einem Güterbahnhof ist. Der Transport der Daten erfolgt in Containern, die ineinander verpackbar sind. Die Datenübertragung erfolgt in Rahmen, sog. virtuellen Containern (VC), die als Synchronous Transport Modules (STM) bezeichnet werden. Ein STM-1-Rahmen hat neun Zeilen zu je 270 Byte und umfasst somit 19440 Bit (9 Zeilen * 270 Byte/Zeile * 8 Bit/Byte). Pro Sekunde werden 8000 STM-1-Rahmen übertragen, was eine nominale[21] Bitrate von 155,52 Mbits/s ergibt (19440 Bit/Rahmen * 8000 Rahmen/s = 155,52 Mbit/s). Jedes Byte unterstützt somit eine Datenübertragungsrate von 64 Kbit/s.

Die Rahmen werden multiplexiert. Vier STM-1-Rahmen bilden einen STM-4-Rahmen, vier STM-4-Rahmen einen STM-16-Rahmen usw. STM-16 stellt z.B. eine Datenrate von 2488,32 Mbit/s zur Verfügung. Heute sind bereits nominale Bitraten von 159252,480 Mbit/s (STM-1024) verfügbar.

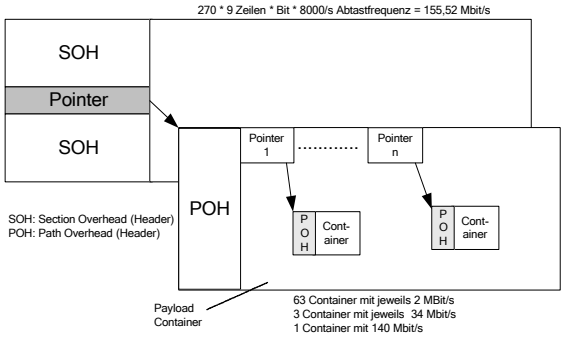


Abbildung 3-28: STM-1-Transportmodul

Der schnelle Zugriff auf die einzelnen Kanäle erfolgt durch die Auswertung des sog. Pointers. Abbildung 3-28 zeigt einen STM-1-Rahmen grob skizziert. Der Pointer im SOH (Section Overhead) zeigt auf den Beginn des größten Containers. Container sind entkoppelt vom übergeordneten Transportmodul. Eine neue Verschaltung geht daher sehr schnell ohne Zwischenpufferung.

[21] Die effektive Nutzlast ist etwas geringer, da man noch den Overhead wegrechnen muss.

3.3.4 ATM

Eine weitere, vorwiegend in öffentlichen Netzen eingesetzte WAN-Technologie[22] ist ATM (Abkürzung für *Asynchroner Transfer Modus*). ATM wurde ursprünglich auch als universelles Breitbandnetz bezeichnet und war als Weiterentwicklung von S-ISDN (Schmalband-ISDN, wie das heutige ISDN genannt wird) gedacht. Man sprach daher auch vom sog. Breitband-ISDN (B-ISDN). B-ISDN war ein Telekomdienst, der sowohl Telekommunikations- als auch Datendienste sowie Verteildienste (für Rundfunk und Fernsehen) in einem einheitlichen Netz anbot. Er sollte also in einer gemeinsamen Netzwerkinfrastruktur das Fernsprechnetz, das Fernsehnetz und Rechnernetze bedienen können. Heute wird der Begriff B-ISDN für Breitbandnetze nicht mehr verwendet. Die Netzkonzeption erwies sich als zu teuer.

Multiplexieren. Das „A" in ATM bezieht sich nach ITU-T auf das Multiplexieren von Daten über ein nicht starres Kanalkonzept für Verbindungen. Dies gilt im Unterschied zum synchronen Transfermodus, wo sich die Daten für eine Verbindung an genau definierten Stellen eines periodisch gesendeten Übertragungsrahmens befinden. Die Übertragung der Daten erfolgt in kleinen Paketen (ATM-Zellen). ATM benötigt schnelle Vermittlungsknoten (Synonyme: ATM-Knoten, ATM-Vermittler, ATM-Switch, ATM-Multiplexierer oder ATM-Router), um die Pakete schnell bewegen zu können. Dies wird u.a. dadurch erreicht, dass *keine Fehlerkorrektur* durchgeführt wird. Stellt man sich eine Kombination aus ATM und SDH vor, so passt die Analogie, dass SDH das Transportnetz und die ATM-Knoten die Rangierbahnhöfe darstellen.

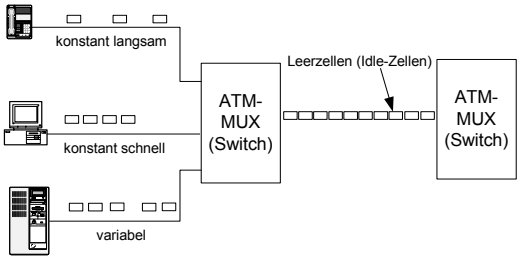


Abbildung 3-29: ATM-Funktionsweise nach (Kiefer 1999)

In Abbildung 3-29 wird die prinzipielle Arbeitsweise von ATM gezeigt. Unterschiedliche Endgeräte mit unterschiedlichen Anforderungen an die Kommunikationsverbindung werden unterstützt. Im Bild sind drei Teilnehmer mit unterschiedlichen Anforderungen an das Kommunikationssystem dargestellt. Der verantwort-

[22] ATM kann auch im LAN eingesetzt werden, ist jedoch dort nicht sehr verbreitet.

liche ATM-Multiplexierer überträgt die Daten durch eine flexible Zuordnung von Zellen zu Verbindungen.

Übermittlungsverfahren. ATM ist ein *paket- und verbindungsorientiertes* Übermittlungsverfahren und arbeitet mit virtuellen Verbindungen, d.h. zwischen zwei Teilnehmern muss zuerst ein Verbindungsaufbau durchgeführt werden, bevor kommuniziert werden kann. Man unterscheidet permanente (PVC = Permanent Virtual Circuit) und transiente (SVC = Switched Virtual Circuit) Verbindungen. Ein PVC ist wie eine Standleitung, die statisch und manuell vom Carrier konfiguriert wird und auch über Switch-Ausfälle hinweg bekannt bleibt. Nach einem System- oder Stromausfall eines beteiligten Switches wird ein PVC wieder aufgebaut. Ein SVC wird dynamisch aufgebaut, wenn dies von einem Endgerät gefordert wird.

Dienstqualität. ATM unterstützt auch bestimmte Anforderungen an die Dienstqualität in Netzen. Insbesondere in neueren Multimedia-Anwendungen ist beispielsweise oft eine gleichzeitige Übertragung von Sprache, Daten und (Bewegt-) Bildern erforderlich. Neben einem großen Bandbreitenbedarf (unkomprimierte Video-Ströme benötigen bis zu 210 Mbit/s) sind vor allem für Audio- und Videoübertragungen, aber auch für Simulationsdaten kontinuierliche Medienströme gefordert. Dies bedeutet, dass die Präsentation kontinuierlich über die Zeit verlaufen soll. Für die Übertragung von Sprach- und Videodaten wird also ein konkreter Zeitbezug (Echtzeitbezug) gefordert.

Folgende Übertragungsparameter sind dabei wichtig:

- Verzögerung: Eine akzeptable Verzögerung für Audio- und Videodaten liegt bei 150 ms.
- Verzögerungssensibilität (Jitter, Varianz der Verzögerung): Die Ankunftszeit darf nicht zu stark variieren (ca. 20 ns bei Audio).
- Konstante und auch hohe Bitrate.

Um für die Übertragung eine ausreichende Qualität zu erhalten, ist eine *Dienstqualität* in Netzen (*Qualitiy-of-Service*, QoS) von hoher Bedeutung. Hierunter versteht man die Fähigkeit eines Netzwerks, einen beliebigen Mix von Bandbreite, Verzögerung, Jitter und Paketverlust über den gesamten Kommunikationsweg (also auch über mehrere Teilnetze hinweg) zu kontrollieren und innerhalb sinnvoller Grenzen zu halten. Hierzu sind natürlich auch Maßnahmen zur Fluss- und Überlastkontrolle von enormer Bedeutung.

ATM-Paketaufbau. ATM dient der schnellen Paketvermittlung, wobei als Pakete sog. Zellen konstanter Länge von 53 Byte (48 Byte Nutzdaten und 5 Byte Overhead, siehe Abbildung 3-30) übertragen werden. Das Konzept bezeichnet man als *Zellenvermittlung*. Die Zellengröße von 53 Byte stellt einen Kompromiss zwischen den Anforderungen an die Daten- und die Sprachübertragung dar. Datenübertragung benötigt vor allem große Nutzdatenfelder, Sprachübertragung eine möglichst geringe Paketierzeit. Durch die Übertragung auf Basis kleiner Pakete (Zellen) ist man sehr flexibel und kann auch Echtzeitanforderungen befriedigen.

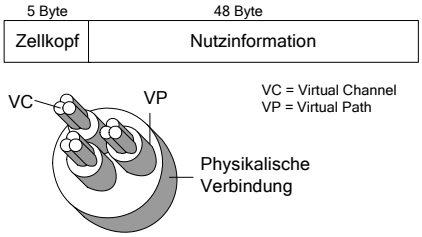


Abbildung 3-30: ATM-Zellenaufbau, ATM-Pfade und -Channels

Nach ITU-T I.361 gibt es zwei verschiedene Typen von ATM-Zellköpfen (Headern), die sich geringfügig unterscheiden[23]. In Abbildung 3-31 sind die Header der NNI- und der UNI-Zelle dargestellt. Im NNI-Zellenkopf sind vier Bit mehr für den VPI reserviert, dafür ist kein Feld *Flusskontrolle* vorhanden. Die Felder haben im Einzelnen folgende Bedeutung:

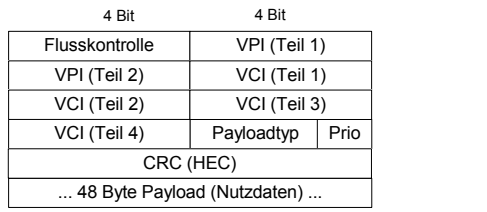
- Das Feld *Flusskontrolle* (auch GFC = Generic Flow Control) dient der Flusssteuerung zwischen Endgerät und Netz. [24]
- Die Felder *VPI* und *VCI* geben das Ziel der Zelle an. Der virtuelle Kanal (auch Circuit genannt) wird adressiert. *VPI* ist der *Virtual Path Identifier*, *VCI* der *Virtual Channel Identifier*. Beide Felder zusammen werden als Verbindungs-Identifikator verwendet (24 Bit).
- Das Feld *Payloadtyp* (3 Bit) gibt den Nutzdatentyp an. Hier wird zwischen Benutzerzellen und ungenutzten Zellen (Unassigned Cells) usw. unterschieden. Letztere werden u.a. zum Stopfen verwenden. Auch sog. O&M-Zellen (Operations and Management) sind für das Netzwerkmanagement vorgesehen.
- Das Feld *Prio* (1 Bit) gibt an, ob die Zelle bei einer Überlastsituation weggeworfen werden kann oder nicht. Der Wert B'0' entspricht der höheren Priorität. Zellen mit diesem Wert werden bei einer Überlastsituation zunächst verschont.
- Das Feld *CRC* (auch *HEC* = Header Error Control genannt) enthält eine Prüfsumme zur Synchronisation und Fehlererkennung, die von der Sendestation vor dem Absenden der Zelle ermittelt wird. Die Prüfung umfasst den Zellenkopf, genauer gesagt die ersten vier Byte davon. Einzelbitfehler können auch korrigiert werden, Zellen mit mehreren Fehlern werden verworfen. Für

[23] Die Header sind in der ITU-T-Empfehlung I.361 beschrieben.

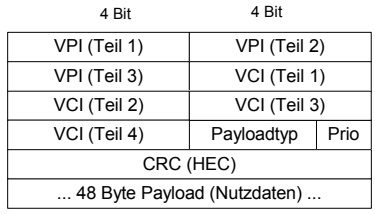
[24] ATM kann auch im LAN eingesetzt werden, ist dort aber selten vorzufinden. Das Feld *Flusskontrolle* dient hier der Regelung von Zugriffs- und Übertragungsrechten, soll aber nicht weiter betrachtet werden.

die Nutzdaten ist keine Fehlererkennung vorgesehen. Das Feld dient auch der Synchronisation des Empfängers auf den Zellenanfang. Die Ermittlung der Zellgrenzen wird über dieses Feld erreicht, indem der Empfänger so lange fünf Byte aus dem Bitstrom entnimmt, bis er eine gültige Prüfsumme findet. Er verschiebt den Ausschnitt aus dem Bitstrom so lange, bis er einen gültigen Zellkopf findet. Über diese Technik vermeidet man die Nutzung festgelegter Bitmuster für die Festlegung der Zellgrenzen.

Mehrere virtuelle, unidirektionale Verbindungen, sog. Datenkanäle (Virtual Channels) werden zum Transport verwendet und von den ATM-Knoten (Switches) zu virtuellen Pfaden zusammengefasst. Der VPI identifiziert den Pfad zwischen allen beteiligten Switches. Der VCI kann für jede Einzelverbindung zwischen den Switches anders sein. Damit ist es nicht notwendig, dass eine VCI in allen Einzelverbindungen geschaltet werden muss. Jeder Switch entlang des Pfads vergibt jeweils eine eigene VCI, die dem Pfad zugeordnet wird. Die Information wird in den Switches in „Routing-Tabellen“ gehalten.



a) Zellenheader für Network-Network-Interface (NNI)



a) Zellenheader für User-Network-Interface (UNI)

Abbildung 3-31: ATM-Zellenheader

ATM-Schichtenmodell. ATM ist in einem eigenen (recht komplexen und vom ISO/OSI-Modell abweichenden Referenzmodell beschrieben, das in Abbildung 3-32 skizziert ist. Das ATM-Referenzmodell sieht folgende Schichten vor:

- Üblicherweise werden SDH/SONET-Rahmen zur Datenübertragung auf der physischen Ebene (Schicht 1) verwendet.
- Die in der Regel auf physische SDH/SONET-Verbindungen aufsetzende ATM-Schicht ist für den Zellenaustausch zuständig.

- Oberhalb der ATM-Schicht liegt die AAL-Schicht (ATM Adaptation Layer), welche die Aufgabe hat, bestimmte Dienstklassen zu behandeln. Von den entwickelten AAL-Protokollen AAL1, AAL2, AAL3/4[25] und AAL5 sind heute vor allem AAL1 und AAL5 relevant. Die Schicht ist zweigeteilt. Die SAR-Schicht (Segmentation and Reassembly) erzeugt Nutzdatensegmente, die CS-Schicht (Convergence Sublayer) dient der Anpassung an die höheren Schichten. CS gibt es je nach Dienstklasse in unterschiedlichen Ausprägungen (VBR, UBR, CBR, ABR).
- ULP ist das darüberliegende Benutzerprotokoll (Upper Layer Protocol).

Auf die einzelnen PDUs soll an dieser Stelle nicht weiter eingegangen werden. Sie werden in den ATM-Zellen übertragen und benötigen natürlich eigene Steuerinformationen.

Je nach AAL-Protokollvariante werden im ATM-Referenzmodell diverse *Dienstklassen*, unterstützt, die verschiedene Qualitätskriterien erfüllen, wobei als Kriterien der Zeitbezug zwischen Sender und Empfänger, die Bitratenanforderung (konstant oder variabel) und die Verbindungsart (verbindungsorientiert, verbindungslos) herangezogen werden. Die Zusammensetzung des Zellenstroms aus Nutz- und Leerzellen wird als *Verkehrsprofil* bezeichnet. Dieses kann durch verschiedene Parameter beeinflusst werden: Normale Bitrate, Spitzenbitrate, Zellverteilung, Verzögerung, maximale erlaubte Zellverlustrate usw. Das Verkehrsprofil wird bei der Verbindungsaufnahme zwischen dem Endgerät und dem nächsten ATM-Knoten ausgehandelt.

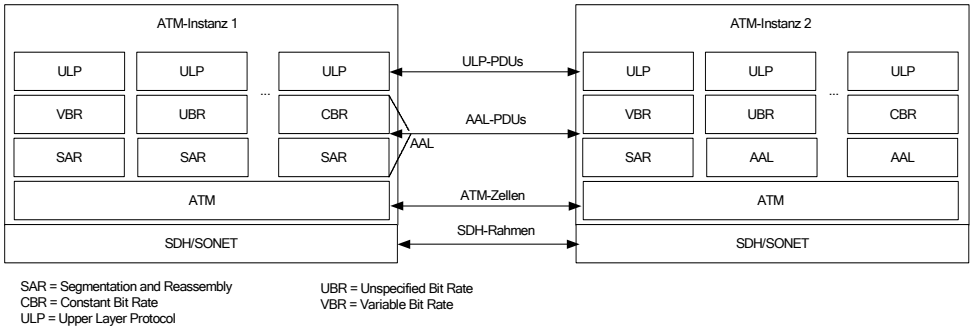


Abbildung 3-32: ATM-Schichtenmodell

[25] AAL3 und AAL4 wurden zusammengefasst. Eine Beschreibung der einzelnen AAL-Protokolle ist (Tanenbaum 2003a) zu entnehmen

Man spricht bei den verfügbaren und auszuhandelnden ATM-Verkehrsprofilen auch u.a. von:

- CBR (Constant Bitrate), typisch für Telefonie, Fax, Video-on-Demand, usw. Hier gilt das Verhältnis: Spitzenbitrate : mittlere Bitrate = 1.
- ABR ((Available Bitrate), typisch für burstartigen Datenverkehr. Hier gilt: Spitzenbitrate : mittlere Bitrate entschieden größer als 1.
- VBR ((Variable Bitrate), typisch für Multimedia-Übertragungen. Hier gilt: Spitzenbitrate : mittlere Bitrate größer oder gleich 1.
- UBR (Unspecified Bitrate), Best-Effort-Vergabe der Bitrate ohne Garantie.

Abschließende Bemerkung. Die zentralen Konzepte von ATM gelten unabhängig von speziellen Netzen und können daher auf verschiedene Basisnetzwerke abgebildet werden. ATM arbeitet z.B. mit einer Datenrate von 155,52 Mbit/s, und man kann zur Übertragung von ATM-Zellen einen STM-1-Rahmen aus einem SDH-Netz verwenden.

Die Netzinfrastruktur im Weitverkehrsbereich für öffentliche Breitbandnetze basiert heute noch auf den Schlüsseltechnologien SDH und ATM. Als Übertragungsmedium dienen Lichtwellenleiter (Glasfaser). SDH stellt die erforderliche Übertragungskapazität bereit. Basis der Transport-Infrastruktur sind SDH und ATM.

Es sei noch erwähnt, dass die ATM trotz der guten technischen Voraussetzungen, heute vor allem wegen der hohen Kosten dieser Technologie nicht genutzt wird. Tatsächlich scheint sich wohl ATM in letzter Zeit aufgrund der positiven Entwicklung anderer Technologien wie etwa der Ethernet-Technologie nicht mehr weiter zu verbreiten. Die Deutsche Telekom hat z.B. angekündigt, in Zukunft nicht mehr ATM als Basis für ihre xDSL-Infrastruktur einzusetzen, sondern Ethernet.

3.4 Übungsaufgaben

1. Welche HDLC-Betriebsmodi kennen Sie? Erläutern Sie einen davon.
2. Skizzieren Sie einen Ablauf einer HDLC-Kommunikation zwischen zwei Stationen im ABM-Mode
3. Was bedeutet LAN-Switching
4. Wozu dient das Protokoll PPP?
5. Können bei Ethernet, speziell bei 100Base-T Kollisionen auftreten und was macht man in heutigen Ethernet-Netzwerken gegen Kollisionen?
6. Was ist im WLAN nach IEEE 802.11 ein Infrastruktur-Modus?
7. Was bedeutet ADSL im Vergleich zu SDSL?

4 Konzepte und Protokolle der Vermittlungsschicht

In diesem Kapitel werden die Grundlagen der Vermittlungsschicht erläutert und konkret an der TCP/IP-Protokollfamilie beispielhaft skizziert. Zunächst werden die allgemeinen Aufgabenstellungen der Schicht 3 aufgezeigt. Hierzu gehören die Vermittlung, die Wegewahl und die Staukontrolle. Verschiedene Verfahren zur Lösung dieser Aufgabenstellungen werden gezeigt. Anschließend werden die wichtigsten Protokolle der Schicht 3 des Internets vorgestellt und auch der Aufbau und die Zusammenhänge des Internets beschrieben. Ausführlich wird auf die Adressierungsprobleme und auf Ansätze zur Lösung dieser Probleme (VLSM, NAT, CIDR) eingegangen. Neben dem Protokoll IP werden Steuer- und Kommunikationsprotokolle wie ICMP, ARP, NAT und DHCP beschrieben. Ebenso wird auf konkrete Routing-Protokolle wie RIP, OSPF und BGP eingegangen. Schließlich wird am Ende des Kapitels noch ein Einblick in die neueste Version des Internet-Protokolls IPv6 gegeben.

Zielsetzung des Kapitels

Ziel dieses Kapitels ist es, einen Überblick über die Aufgaben und die Funktionsweise der Schicht 3, insbesondere innerhalb der TCP/IP-Protokollfamilie zu geben. Der Studierende soll sich ein tiefergehendes Verständnis über die Arbeitsweise und das Zusammenspiel verschiedener Techniken und Protokolle der Schicht 3 verschaffen.

Wichtige Begriffe

Vermittlung, Leitungsvermittlung, Wegewahl, Routing, adaptives, statisches, zentralisiertes Routing, Router, Subnetzwerk, Metriken für die Routingentscheidung, Optimierungsprinzip, Peering, Transit, Provider, Autonomes System, Protokolle wie IP, ICMP, NAT, DHCP, ARP, RARP, Link-State-Verfahren, OSPF, Distanz-Vector-Verfahren, Routingprotokolle wie RIP, OSPF, IS-IS und BGP, MPLS, CIDR und VLSM, IPv6, VPN.

4.1 Grundlagen

In diesem Abschnitt wollen wir einige Mechanismen der Vermittlungsschicht detaillierter besprechen. Vorab soll ein kurzer Überblick über die Vermittlungsschicht, deren Aufgaben und Positionierung im Protokollstack gegeben werden:

- Die Vermittlungsschicht ist die unterste Schicht, die sich mit *Ende-zu-Ende-Übertragung* in einem Netzwerk befasst.

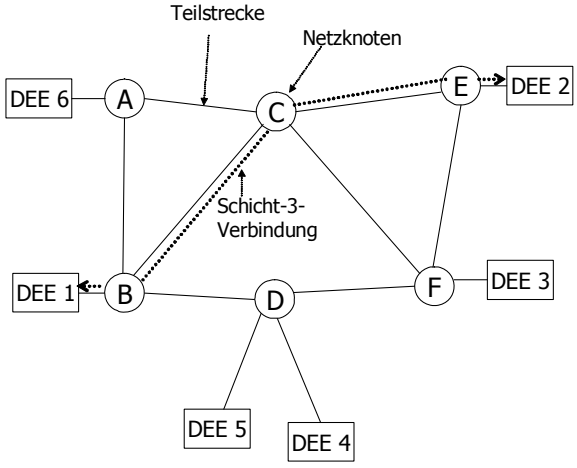
- Die Schnittstelle zur Vermittlungsschicht ist für WAN-Zugänge auch meist die *Netzbetreiberschnittstelle.*
- Die Endsysteme kommunizieren über einen oder mehrere Netzknotenrechner und über einen oder mehrere Übertragungswege (Teilstrecken) miteinander.
- Zu den Aufgaben der Vermittlungsschicht gehören im Wesentlichen die Wegewahl (auch Routing genannt), Multiplexen und Demultiplexen, die Staukontrolle (Congestion Control) sowie die Segmentierung und die Reassemblierung (auch als Fragmentierung und Defragmentierung bezeichnet).

Wir werden uns im Weiteren zunächst allgemein mit Vermittlungsverfahren, Wegewahl-Mechanismen und der Staukontrolle befassen, bevor wir im nächsten Teil das Internet mit seinen speziellen Schicht-3-Protokollen betrachten.

Besonderer Wert wird also auf die Mechanismen der Vermittlungsschicht des Internets mit ihren wesentlichen Protokollen gelegt. Weitere Protokolle, die der Vermittlungsschicht zugeordnet werden (wie z.B. IPX), können in der angegebenen Literatur nachgelesen werden.

4.1.1 Vermittlungsverfahren

In der Vermittlungsschicht haben wir es vor allem im WAN üblicherweise mit sog. Teilstreckennetzen zu tun. Dies sind im Wesentlichen Netzknoten, die über Leitungen miteinander verbunden sind. An diese werden die DEEs (Datenendeinrichtung), also die einzelnen Endsysteme angeschlossen.



DEE: Datenendeinrichtung
A, B,...,F: Knotenrechner, Netzknoten

Abbildung 4-1: Beispielnetzwerk

Den Gesamtvorgang der Verbindungsherstellung, des Haltens und des Abbauens einer Verbindung bezeichnet man als *Vermittlung* (engl. *Switching*). Im dem Beispielnetz aus Abbildung 4-1 wird eine Verbindung zwischen DEE1 und DEE2 über die Knoten B, C und E hergestellt.

Wir greifen im Folgenden die verschiedenen Vermittlungsverfahren auf und stellen Sie gegenüber. In Abbildung 4-2 wird zunächst in Leitungs- und Paketvermittlung (Paketvermittlung wird gelegentlich auch als Nachrichtenvermittlung bezeichnet) unterschieden.

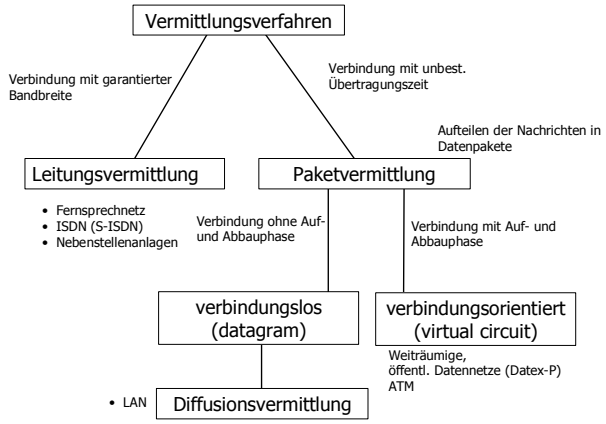


Abbildung 4-2: Vermittlungsverfahren

Leitungsvermittlung. In leitungsvermittelten Netzwerken werden alle auf dem Netzwerkpfad benötigten Ressourcen wie Bandbreite und Pufferspeicher in den Netzwerkknoten für die Dauer der Verbindung vorab reserviert.

Beispielnetze für die klassische Leitungsvermittlung sind das analoge Fernsprechnetz und das digitale ISDN. Bei diesem Switching-Verfahren, auch *circuit switching* oder *Durchschaltevermittlung* genannt, wird über die gesamte Verbindung ein physikalischer Verbindungsweg durch das Netzwerk geschaltet. Es wird eine feste Bandbreite garantiert und zwar unabhängig von dem, was tatsächlich übertragen wird.

Bei der Leitungsvermittlung ist es wahrscheinlich, dass Bandbreite unnötig reserviert wird, wenn z.B. das Nachrichtenaufkommen nicht immer konstant ist. Auch mit Blockierungen, also der Ablehnung eines Verbindungswunsches ist zu rechnen, wenn kein Verbindungsweg mehr frei ist.

Die teueren Leitungen zwischen den Netzknoten werden in heutigen Netzwerken mit Hilfe von Multiplexverfahren in mehrere „Sprechkreise“ aufgeteilt. Die verwendeten Verfahren sind Frequenzmultiplexen (FDM) und Zeitmultiplexen (TDM),

wobei sich heute das Zeitmultiplexverfahren durchsetzt, bei dem jeder Ende-zu-Ende-Verbindung zwischen zwei Knotenrechnern ein Zeitschlitz zugeordnet wird, über den eine bestimmte Bitrate möglich ist.

Paketvermittlung. Bei der Paketvermittlung werden immer komplette Nachrichten in einzelnen Datenpaketen zwischen den Netzwerkknoten ausgetauscht. Nachrichten werden erst weitergesendet, wenn sie vollständig sind. Man spricht hier auch von Store-and-Forward-Verfahren. Datagramme werden über eine vorher aufgebaute sog. virtuelle Verbindung übermittelt.

Bei der Paketvermittlung werden die Ressourcen nicht vorab reserviert, sondern zur Laufzeit dynamisch zugewiesen. Die Folge kann sein, dass zwei Kommunikationspartner auf eine Verbindung etwa in einer Warteschlange warten müssen, bis die erforderlichen Ressourcen verfügbar sind.

Das Netz überträgt die Nachrichten immer mit der Zieladresse evtl. über mehrere Knoten mit Zwischenspeicherung und zerlegt sie ggf. in einzelne Pakete (N-PDUs). Die Paketvermittlung ist für die Datenübertragung effizienter als die Leitungsvermittlung, jedoch wird keine Bandbreite garantiert. Daher gibt es aber auch kaum Blockierungen. Typische Beispiele für Paketvermittlungsnetze sind das Internet und das Breitband-ISDN auf Basis von ATM (sehr kurze Pakete, Zellen). In heutigen Netzwerken wird die Leitungsvermittlung immer mehr durch die Paketvermittlung ersetzt.

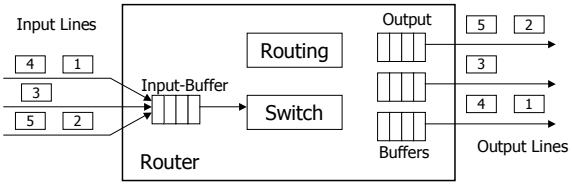
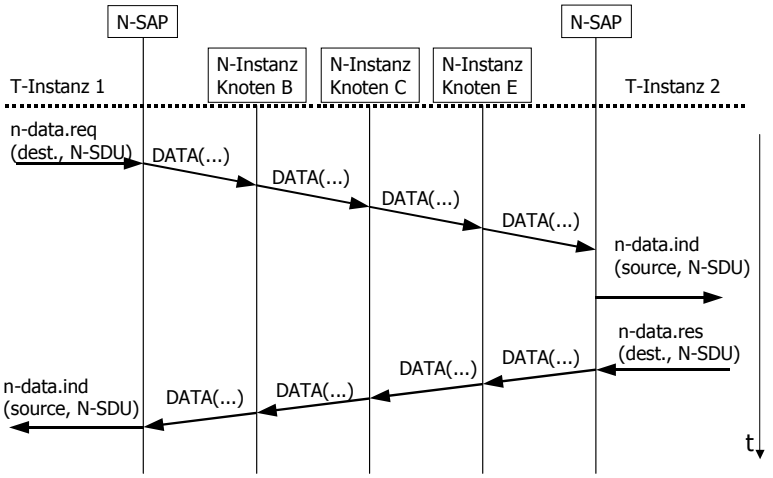


Abbildung 4-3: Prinzip der Paketvermittlung im Knoten nach (Gerdsen 1994a)

In Abbildung 4-3 ist dargestellt, wie ein Knotenrechner (Router) in einem paketvermittelten Netz grundsätzlich arbeitet. Über eine oder mehrere Eingangsleitungen werden ankommende Pakete entgegengenommen und über eine Switching- und Routing-Logik an die passende Ausgangsleitung weitergegeben.

Bei der Paketvermittlung unterscheidet man die verbindungslose und die verbindungsorientierte Variante. Bei der verbindungslosen Paketvermittlung werden Datagramme (N-PDUs) ohne vorhergehenden Verbindungsaufbau gesendet. Jedes Datagramm enthält die Quell- und die Zieladresse. Die Knoten ermitteln für jedes Datagramm einen optimalen Weg (Routing). Man bezeichnet dies auch als verbindungslose Vermittlung. Es ist nur ein einfacher data-Dienst zum Senden von Datagrammen erforderlich.

In Abbildung 4-4 ist dargestellt, wie eine Nachricht der T-Instanz 1 über mehrere Knoten B, C, E zum Ziel (T-Instanz 2) weitergeleitet wird. Die Antwort geht hier über den gleichen Weg zurück, was aber in der Schicht 3 je nach Konfiguration und Netzauslastung erfolgen kann.



N-SAP: Network Service Access Point

Abbildung 4-4: Nachrichtenfluss über mehrere N-Instanzen nach (Gerdsen 1994a)

Sonderfälle der Paketvermittlung sind *Virtual Circuits (VC).* Sie werden auch „scheinbare Verbindungen" genannt. Die Verbindung bleibt hier für die Dauer der Datenübertragung erhalten. Bei VC kann man sagen, dass kein physikalisches Durchschalten der Verbindung erfolgt, sondern die beim Verbindungsaufbau ermittelte Routing-Information in den Knoten verwendet wird. Ein typisches Netz, das mit Virtual Circuits arbeitet, ist das in die Jahre gekommene Datex-P-Netzwerk der Telekom. Ein moderneres Beispielnetz stellt ATM dar.

Man nennt Virtual Circuits auch einen verbindungsorientierten Dienst auf der Vermittlungsschicht. Die Kommunikation über VC wird mit entsprechenden Diensten in die drei Phasen Verbindungsaufbau (connect-Dienst), Datenübertragung (data-Dienst) und Verbindungsabbau (disconnect-Dienst) eingeteilt. Die Verbindung zwischen zwei Endsystemen wird schrittweise über Teilstrecken aufgebaut. Die Knoten müssen in der Verbindungsaufbauphase Informationen über das Mapping von eingehenden Paketen zu Ausgangsteilstrecken speichern, also eine gewisse Kontextverwaltung (Status und Verbindungstabellen) durchführen.

Schließlich kann als Spezialfall der paketorientierten, verbindungslosen Vermittlung auch das bekannte Diffussionsnetz angesehen werden, das z.B. im LAN Verwendung findet.

4.1.2 Wegewahl (Routing)

Die Wegewahl (Routing) ist eine der wesentlichen Aufgaben der Schicht-3-Instanzen in den Knoten des Vermittlungsnetzes. Sie wird bei paketorientierten Netzen notwendig, wenn in einem Netzwerk alternative Wege zwischen den Endsystemen vorhanden sind.

Verschiedene Routing-Kriterien und -Algorithmen sind möglich. Hierzu gehören die Suche der geringsten Entfernung oder die möglichst geringe Anzahl von Hops (Anzahl der zu durchlaufenden Knoten). Die Routing-Information wird in Routing-Tabellen in der Regel in den Knoten verwaltet. Routing-Verfahren lassen sich nach verschiedenen Klassifizierungen einteilen: Man unterscheidet z.B. globale oder dezentrale Verfahren, aber auch statische oder dynamische Verfahren.

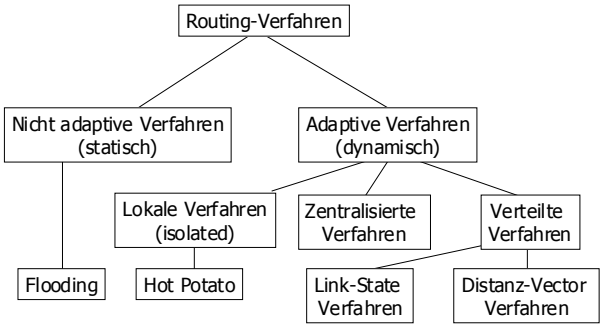


Abbildung 4-5: Klassifizierung der Routing-Verfahren

Das einfachste Verfahren ist *Flooding*. Hier wird jedes ankommende Paket an alle oder eine begrenzte Auswahl (*selektives Flooding*) an Ausgangsleitungen weitergereicht. Flooding ist enorm robust, und es wählt immer den optimalen Weg, da es ja alle auswählt. Leider ist das Verfahren aber nicht sehr leistungsfähig.

Man unterscheidet weiterhin folgende Verfahren:

- *Statische* Algorithmen: Hier gibt es keine Messungen der aktuellen Situation, sondern vor der Inbetriebnahme ermittelte Metriken. Eine statische Routing-Tabelle, die bei der Knotenkonfigurierung eingerichtet wird, enthält die fest vorgegebenen Routen, die ein Paket nehmen kann.
- *Dynamische* (adaptive) Algorithmen: Diese Verfahren nutzen Verkehrsmessungen zur Routenermittlung. Die Routing-Tabellen werden dynamisch über definierte Metriken angepasst. Die Optimierungskriterien können sich dynamisch verändern und werden im Algorithmus berücksichtigt.

Statische Routing-Algorithmen sind das Shortest-Path-Routing, Flooding und flussbasiertes Routing. Beim dynamischen Routing unterscheidet man, wie in Abbildung 4-5 zu sehen ist, wiederum folgende Möglichkeiten:

- *Isoliertes Routing*: Hier trifft jeder Knoten die Routing-Entscheidungen alleine. In diese Kategorie fällt auch das sog. Hot-Potato-Verfahren, bei dem jedes ankommende Datagramm grundsätzlich so schnell wie möglich an alle nicht überlasteten Ausgänge weiter geleitet wird.
- *Zentrales Routing* über einen zentralen Knoten (Routing-Kontroll-Zentrum): Die Zentrale ermittelt die Routing-Information und überträgt alle Routing-Tabellen an die einzelnen Knoten.
- *Dezentrales Routing* (verteilte Verfahren): Die Routing-Funktionalität liegt in jedem einzelnen Knoten. Die Routing-Tabellen werden im Zusammenspiel ermittelt.

Eine andere Klassifizierung unterteilt in zustandsabhängige und zustandsunabhängige Routing-Verfahren. Bei zustandsunabhängigen Verfahren wird der aktuelle Zustand des Netzes nicht berücksichtigt. Die aktuelle Belastung der Router oder die Bandbreitenauslastung spielt also keine Rolle. Die Wegewahl berücksichtigt nur die Entfernung zum Ziel. Zustandsabhängige Verfahren berücksichtigen dagegen die aktuelle Situation.

Zentrales Routing. Beim zentralen Routing gibt es ein sog. Routing-Control-Center (RCC), in dem die gesamte Routing-Information gesammelt wird (siehe Abbildung 4-6). Das Verfahren ist nicht fehlertolerant, aber konsistent, jedoch besteht beim zentralen Routing die Gefahr, dass Routing-Information an Aktualität verliert.

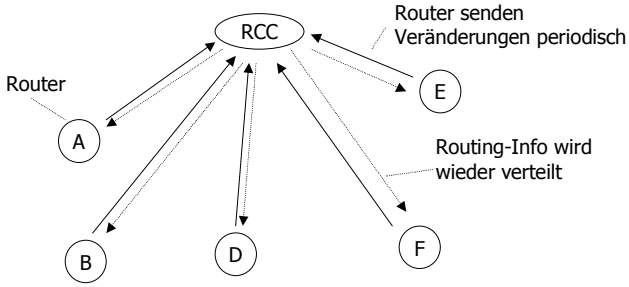


Abbildung 4-6: Zentrales Routing über ein RCC

Verteiltes Routing. Beim verteilten Routing unterscheidet man im Wesentlichen zwei verschiedene Verfahren. Zum einen ist das *Distance-Vector-Routing* oder Entfernungsvektorenverfahren ein sehr bekanntes und ursprünglich im ARPANET und auch nachher im Internet eingesetztes Verfahren. Zum anderen ist das heute ebenfalls im Internet eingesetzte Verfahren namens *Link-State-Routing* oder Ver-

bindungszustandsverfahren sehr bekannt. Es wird seit dem Ende der 70er Jahre im ARPANET bzw. im Internet eingesetzt.

Hierarchisches Routing. Schließlich ist auch noch das hierarchische Routing-Verfahren zu nennen. Große Netze haben (zu) große Routing-Tabellen, die mit langen Suchzeiten in den Routing-Tabellen einhergehen, zur Folge. Die Verringerung der Routing-Tabellen kann durch eine hierarchische Organisation erreicht werden. Als Hierarchiestufen sind z.B. Regionen geeignet. In Abbildung 4-7 ist ein Beispiel für ein nach Regionen aufgeteiltes Netzwerk dargestellt, wobei nur die Netzwerkknoten gezeigt werden.

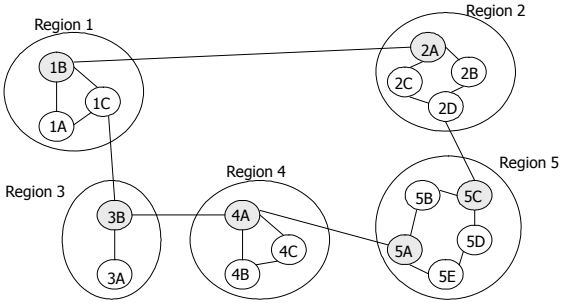


Abbildung 4-7: Hierarchisch organisiertes Routing

Würde in diesem Beispiel jeder Knoten jeden anderen erreichen müssen, käme man z.B. für den Knoten 1A in Region 1 gemäß der Routing-Tabelle aus Abbildung 4-8 auf 17 Tabelleneinträge.

Routing-Tabelle für 1A (vorher)

Ziel	Leitung	Teilstr.
1A	--	--
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Routing-Tabelle für 1A (nachher)

Ziel	Leitung	Teilstr.
1A	--	--
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Reduktion von 17 auf 7 Einträge!
Nachteil: Ansteigende Pfadlängen

Abbildung 4-8: Routing-Tabelle für hierarchisches Routing, Beispiel nach (Tanenbaum 2003a)

Bei hierarchischem Routing mit speziellen Knoten in jeder Region, die nach außen hin bekannt sind und Datagramme weiterleiten, kommt man nur noch auf sieben Einträge in der Routing-Tabelle von Knoten 1A. Nachteilig am hierarchischen Routing sind die möglicherweise ansteigenden Pfadlängen.

Die ansteigende Pfadlänge beim hierarchischen Routing kann am Beispiel der Pfadlänge zwischen den Knoten 1A und 5C gezeigt werden. Ohne hierarchisches Routing würde der Knoten 1A, wie in der Routingtabelle in der Abbildung 4-8 links angegeben, Pakete zum Knoten 5C über den Knoten 1B versenden. Betrachtet man die Abbildung 4-7, ergibt sich somit der Pfad 1A-1B-2A-2C-2D-5C mit einer Länge von 5 Teilstrecken. Die Routingtabelle beim hierarchischen Routing (Abbildung 4-8 rechts) gibt für Ziele der Region 5 den Knoten 1C als ersten Knoten vor. Betrachtet man wieder die Abbildung 4-7, ergibt sich der neue Pfad 1A-1C-3B-4A-5A-5B-5C mit einer Länge von 6 Teilstrecken. Durch das hierarchische Routing ist die Pfadlänge für die Verbindung zwischen den Knoten 1A und 5C somit von 5 auf 6 Teilstrecken angestiegen.

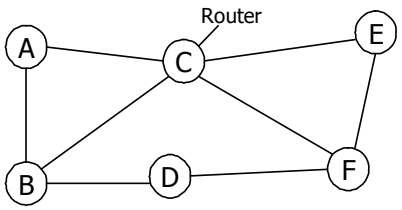


Abbildung 4-9: Optimierungsprinzip beim Routing

Optimierungsprinzip. Für das Routing gilt ein *Optimierungsprinzip,* das folgendes besagt: Wenn Router C auf dem optimalen Pfad zwischen *A* und *F* liegt, dann fällt der Pfad von *C* nach *F* ebenso auf diese Route (siehe Abbildung 4-9). Im Shortest-Path-Routing wird ein Graph des Teilnetzes statisch erstellt. Jeder Knoten im Graph entspricht einem Router, eine Kante entspricht einer Leitung zwischen zwei Routern. Die Kanten werden beschriftet („Pfadlänge"), die Metrik hierfür kann aus verschiedenen Kritierien (auch kombiniert) berechnet werden. Unter anderem sind folgende Kriterien möglich:

- Entfernung
- Bandbreite
- Durchschnittsverkehr
- Durchschnittliche Warteschlangenlänge in den Routern
- Verzögerung

Die Berechnung des kürzesten Pfads erfolgt dann über einen Optimierungsalgorithmus wie z.B. über Dijkstras Algorithmus (Tanenbaum 2003a). In der Abbildung

4-10 ist der Netzwerk-Graph mit Metriken skizziert. Berücksichtigt man die in der Abbildung als Kantenbeschriftungen angegebenen Metrikwerte, ergibt sich zwischen den Knoten *A* und *F* als kürzester Pfad *A-C-E-F* mit einer aufsummierten Pfadlänge (Addition der Werte) von 7.

Die zugrundeliegende Problemstellung stammt aus der Graphentheorie und kann wie folgt beschrieben werden: Finde für einen Startknoten *s* und einen Endknoten *e* eines gewichteten Graphen G mit der Knotenmenge *V*, der Kantenmenge *E* und der Kostenfunktion *k* einen Weg zwischen *s* und *e* mit minimalen Kosten. Die Kostenfunktion *k* bezieht sich auf eine Kante zwischen zwei Knoten.

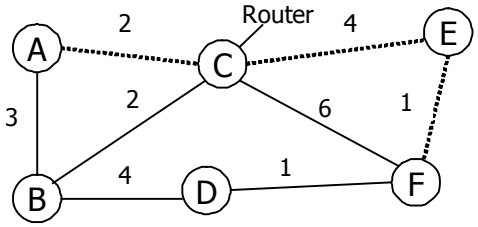


Abbildung 4-10: Beispiel für eine Shortest-Path-Berechnung

Distance-Vector-Verfahren. Im *Distance-Vector-Verfahren* (andere Bezeichnung: Bellman-Ford-Routing nach Bellman, 1957 und Ford, 1962) führt jeder Router eine dynamisch aktualisierte Routing-Tabelle mit allen Zielen. Der Begriff „Entfernungsvektor" bzw. „Distance-Vector" kommt daher, dass eine Route zu einem Ziel aus einer Kombination aus Entfernung und Richtung (Vektor) angegeben wird. Die Entfernung ist dabei eine metrische Bewertung der Route nach einem bestimmten Verfahren, in das verschiedene Größen einbezogen werden können. Das Verfahren setzt voraus, dass ein Router die Entfernung zu allen Zielen kennt, wobei die Berechnung mit Hilfe der Nachbarknoten ausgeführt wird. Die Routing-Tabelleneinträge enthalten die bevorzugte Ausgangsleitung zu einem Ziel. Als Metrik kann z.B. die Verzögerung in ms oder die Anzahl der Teilstrecken (Hops) bis zum Ziel verwendet werden: Benachbarte Router tauschen Routing-Information aus.

Nachteilig bei diesem Verfahren ist, dass *Schleifen* möglich sind und dadurch Pakete ewig kreisen können. Das Problem ist auch als *Count-to-Infinity-Problem* bekannt. Zudem hat das Verfahren eine *schlechte Konvergenz*, was bedeutet, dass sich schlechte Nachrichten z.B. über nicht mehr verfügbare Routen, sehr langsam im Netz verbreiten. Dies liegt daran, dass kein Knoten über vollständige Informationen (Verbindungen und deren Kosten) zum Netz verfügt. Kein Knoten verfügt (außer bei direkten Nachbarn) über den vollständigen Pfad zu einem entfernten Knoten, sondern kennt nur den Nachbarn, an den er ein Paket senden muss, damit es beim gewünschten Zielknoten ankommt.

Jeder Knoten sieht beim Distance-Vector-Routing nur seine Nachbarn. Einige Zeit nach dem Netzstart, nach der sog. Konvergenzdauer, verfügen alle Knoten über die optimalen Routing-Tabellen. Jeder Router legt beim Start zunächst einen Vektor mit der Entfernung 0 für jeden bekannten Router an. Alle anderen Ziele bekommen die Entfernung „unendlich". Durch das zyklische Versenden der bekannten Entfernungsvektoren breiten sich die Informationen im Netz aus. Mit jeder neuen Information werden die optimalen Routen in den Routern neu berechnet.

Wie man im Beispiel aus Abbildung 4-11 sieht, enthalten die Einträge in den Routing-Tabellen den Zielknoten, die Distanz zum Zielknoten und die Kosten (hier in Hops). Im Beispiel sind die Routing-Tabellen von Knoten A und E dargestellt. A lernt hier z.B. von den benachbarten Knoten, dass der optimale Weg zu D über E in 5 Hops ist.

Knoten A

Ziel	Distanz	Nächster Knoten	Hops
B	4	B	1
C	3	C	1
E	2	E	1
D	5	E	2

Knoten E

Ziel	Distanz	Nächster Knoten	Hops
A	2	A	1
B	3	B	1
C	4	B	2
D	3	D	1

...

Nach einiger Zeit (Konvergenzdauer) verfügen alle Router über optimale Routing-Tabellen

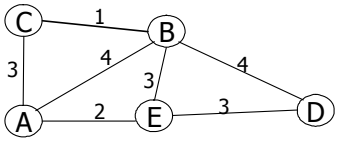


Abbildung 4-11: Beispiel-Routing-Tabellen im Distance-Vector-Verfahren

Die Eigenschaften des Distance-Vector-Routing lassen sich kurz wie folgt zusammenfassen:

- Das Verfahren ist verteilt. Die Nachbarn tauschen Informationen über die nächste Umgebung aus, die Router berechnen die besten Pfade aus ihrer Sicht und tauschen erneut Informationen aus.
- Das Verfahren ist iterativ, es wird zyklisch wiederholt.
- Die einzelnen Knoten arbeiten selbstständig und völlig unabhängig voneinander.
- Gute Nachrichten verbreiten sich schnell, schlechte eher langsam.
- Bei Ausfall eines Links ist evtl. keine Terminierung mehr sichergestellt.

Link-State-Verfahren. Im *Link-State-Verfahren* verwaltet jeder Router eine Kopie der gesamten Netzwerktopologie in einer Link-State-Datenbasis, also nicht nur der nächsten Umgebung wie beim Distance-Vector-Verfahren. Jeder Knoten kennt also alle Kosteninformationen des gesamten Netzwerks. Jeder Router verteilt die lokale Information per Flooding an alle anderen Router im Netz und damit kennt jeder Router alle anderen. Die Berechnung der optimalen Routen erfolgt meist dezentral, wobei jeder einzelne Knoten den absolut kürzesten Pfad errechnet. Auch eine zentrale Berechnung der optimalen Wege mit einer netzweiten Verteilung ist im Link-State-Verfahren möglich. Wichtig ist nur, dass bei der Berechnung die Informationen zum gesamten Netz einbezogen werden. Die Berechnung der kürzesten Pfade wird z.B. über einen Shortest-Path-Algorithmus (z.B. Dijkstra's Algorithmus) durchgeführt.[1]

Da alle Knoten die gesamte Topologie kennen und somit jeder Knoten die gleiche Information über die Topologie besitzt, sind keine Schleifen und eine schnelle Reaktion auf Topologieänderungen möglich.

Dieses Verfahren wird deshalb als Link-State-Verfahren bezeichnet, weil es die globale Zustandsinformation des Netzes mit den Kosten aller Verbindungsleitungen (Links) kennen muss. Die Knoten kennen zwar anfangs noch nicht alle anderen Knoten, aber durch den Empfang von sog. Link-State-Broadcasts wird die Information unter den Knoten ausgetauscht.

In sehr großen Netzen ist es sinnvoll, eine gewisse Hierarchie im Netz einzuführen (siehe hierarchisches Routing) und innerhalb kleinerer Regionen oder Teilnetzen ein einheitliches Routing-Verfahren zu nutzen. Zum Austausch der Routing-Information zwischen den Regionen können dann wieder eigene Verfahren verwendet werden, die komprimierte Information austauschen. Ein Link-State-Verfahren würde über alle Rechner im Internet nicht funktionieren. Wir werden im Weiteren noch sehen, dass im globalen Internet mehrere Verfahren zur Anwendung kommen, aber natürlich auch eine Hierarchisierung durchgeführt wurde.

4.1.3 Staukontrolle (Congestion Control)

Zu viele Pakete in einem Netz, also mehr als die maximale Übertragungskapazität zulässt, führen zum Abfall der Leistung und damit zur Überlastung (Congestion) oder Verstopfung des Netzes. Die Ursachen hierfür können vielfältig sein:

- Viele Pakete zu einer Zeit führen zu langen Warteschlangen in den Netzknoten
- Langsame Prozessoren in den Netzknoten
- Zu wenig Speicher in den Netzknoten

[1] Dijkstra hat sich im Jahre 1959 damit befasst. Der Algorithmus von Dijkstra für das Link-State-Verfahren ist in (Kurose 2002) ausführlich beschrieben. Ein anderer Algorithmus hierfür ist der sog. Prim-Algorithmus.

Eine Überlastung kann zu einem Teufelskreis führen. Pakete gehen verloren oder werden evtl. in Netzknoten verworfen und werden von den Sendern erneut verschickt. Die Sendungswiederholungen erhöhen wiederum die Last, und so geht es weiter, bis das Netz vollständig überlastet ist. Bei übermäßiger Verkehrsbelastung des Netzes fällt – wie in Abbildung 4-12 zu sehen ist – die Leistung rapide ab. Durch Staukontrolle sollen Verstopfungen bzw. Überlastungen im Netz vermieden werden. Maßnahmen können in den Schichten 2 bis 4 ergriffen werden. Möglichkeiten der Staukontrolle sind u.a.:

- Lokale Steuerung (gehört zur Schicht 2), da sie sich auf Einzelleitungen bezieht
- Ende-zu-Ende-Steuerung zwischen Endsystemen (Schicht 4)
- Globale Steuerung über das gesamte Netz (Schicht 3)

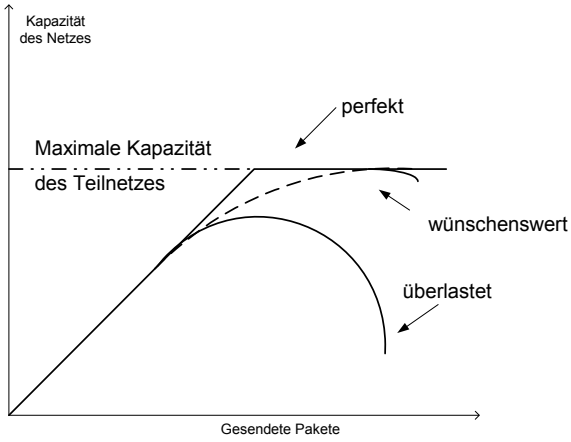


Abbildung 4-12: Netzwerkbelastung und Leistung im Netz nach (Tanenbaum 2003a)

Im Gegensatz zur Flusssteuerung ist die Staukontrolle ein Mechanismus mit *netzglobalen* Auswirkungen. Wir wollen uns in diesem Abschnitt mit den Möglichkeiten in der Schicht 3 befassen. Bei der Diskussion der Transportschicht wird auf die dort vorhandenen Möglichkeiten eingegangen.

Eine wesentliche Ursache für Überlastungen sind *Verkehrsspitzen*. Man kann diese bereits im Vorfeld eindämmen, indem man das sog. *Traffic Shaping* als Maßnahme zur Regulierung der durchschnittlichen Datenübertragungsrate von Endsystemen einsetzt.

Die Überwachung der Endsysteme wird als Traffic Policing bezeichnet und kann durch den Netzbetreiber ausgeführt werden. Diese Maßnahme eignet sich besser für virtuelle Verbindungen (VC) als für datagrammorientierte Netze. Zum Einsatz

kommt der Leaky-Bucket-Algorithmus, dessen Prinzip in der Abbildung 4-13 skizziert ist und wie folgt funktioniert:

Endsysteme (Hosts) verfügen über Netzwerkschnittstellen in Kernel und Netzwerkkarten mit einer internen Warteschlange, die hier als Leacky Bucket bezeichnet wird (rinnender Eimer). Wenn die Warteschlange voll ist, wird ein neues Paket schon im Endsystem verworfen und belastet das Netz nicht.

Die erforderlichen Parameter müssen in einer sog. Flussspezifikation für virtuelle Verbindungen beim Verbindungsaufbau zwischen Sender und Empfänger ausgehandelt werden. Man einigt sich beim Verbindungsaufbau über die max. Paketgröße, die max. Übertragungsrate usw. Die Abbildung 4-13 zeigt die Analogie eines rinnenden Eimers mit einer Netzwerkschnittstelle in einem Host, die den Ausgang der Pakete reguliert. Auch beim Wassereimer tropft das Wasser konstant durch die Öffnung.

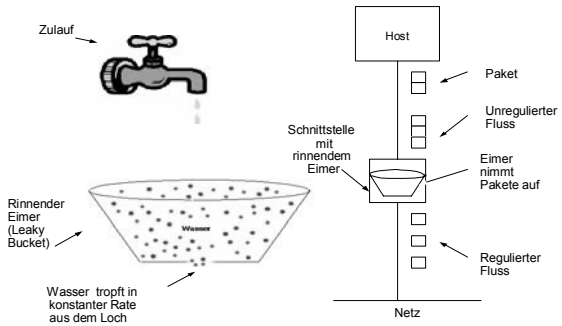


Abbildung 4-13: Leaky-Bucket-Verfahren zur Staukontrolle nach (Tanenbaum 2003a)

Für Netze mit virtuellen Verbindungen und auch für datagrammorientierte Netze gibt es noch weitere Möglichkeiten der Überlastungskontrolle. Hierfür wird auf die Literatur verwiesen (Tanenbaum 2003a).

4.2 Das Internet und das Internet-Protokoll IPv4

4.2.1 Überblick

In Abbildung 4-14 sind die wichtigsten Bestandteile (Protokolle, Softwarekomponenten, Adressierungsmechanismen) der Vermittlungsschicht im Internet in Abgrenzung zu den benachbarten Schichten dargestellt. Da das Internet ein paketorientiertes Netzwerk ist, ist die Wegewahl (Routing) eine der wichtigsten Aufgaben der Vermittlungsschicht im Internet. Im Internet bzw. in Netzwerken, die IP nutzen, wird das komplette Routing über verschiedene Protokolle abgewickelt.

Die Vermittlungsknoten verwalten jeweils eine Routing-Tabelle. Schließlich übernimmt das Internet-Protokoll die eigentlichen Übertragungsaufgaben sowie die

Adressierung. Das Internet-Protokoll (IP), als wichtigstes Protokoll der Internet-Vermittlungschicht ist ein paketvermitteltes (datagrammorientiertes) und verbindungsloses Protokoll. IP dient der Beförderung von Datagrammen von einer Quelle zu einem Ziel evtl. über verschiedene Zwischenknoten (IP-Router). Datagramme werden ggf. während des Transports zerlegt und am Ziel wieder zusammengeführt, bevor sie der Schicht 4 übergeben werden.

IP stellt einen ungesicherten verbindungslosen Dienst zur Verfügung, d.h. es existiert keine Garantie der Paketauslieferung. Die Übertragung erfolgt nach dem *Best-Effort-Prinzip (*Auslieferung nach bestem Bemühen), wobei jedes Paket des Datenstroms isoliert behandelt wird. Ein IP-Paket wird in einem oder – sofern eine Fragmentierung erforderlich ist – ggf. in mehreren Rahmen der zugrundeliegenden Schicht 2 transportiert, für den Längenrestriktionen bestehen. Bei Ethernet ist z.B. eine Länge von 1500 Byte üblich.

In die Vermittlungsschicht ordnen wir auch spezielle Protokolle zur Unterstützung und Optimierung der Adressierung und Adresskonfigurierung wie DHCP (Dynamic Host Configuration Protocol) und NAT (Network Address Translation) ein. Diese Zuordnung ist in der Literatur nicht immer üblich, passt aber aus unserer Sicht recht gut.

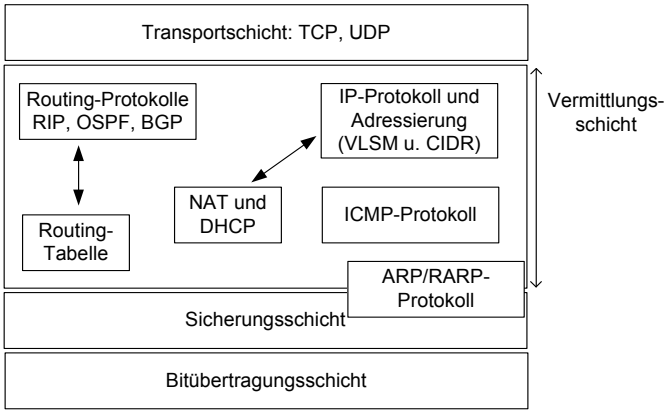


Abbildung 4-14: Das Innere der Vermittlungsschicht im Internet nach (Kurose 2002)

Wir werden im Weiteren die wichtigsten Komponenten der Vermittlungsschicht ausführlich diskutieren. In der Abbildung 4-14 ist zu erkennen, dass ARP (Address Resolution Protocol) und RARP (Reverse Address Resolution Protocol) an der Schnittstelle zwischen Schicht 2 und 3 angesiedelt wurden. Diese Protokolle kümmern sich um das Mapping der MAC-Adressen auf die IP-Adressen, und wir behandeln die Problematik ebenfalls in diesem Kapitel. Das ICMP-Protokoll dient als Steuerprotokoll zur Übertragung von Fehlermeldungen und wird ebenfalls weiter unten diskutiert.

4.2.2 Aufbau des Internets

Das globale Internet stellt sich heute als eine Sammlung von vielen Einzelnetzen dar, die weltweit über viele Kommunikationsverbindungen miteinander verbunden sind. Die Einzelnetze werden auch als autonome Systeme bezeichnet.

Autonome Systeme

Autonome Systeme (AS) sind eigenständig von verschiedensten Organisationen verwaltete Teilnetze. Es gibt derzeit mehr als 110.000 autonome Systeme weltweit. Das Internet ist eine hierarchische Organisation des gesamten Netzwerks, das die autonomen Systeme miteinander verbindet. Die großen AS bilden gemeinsam ein globales Backbone (Netzwerkrückgrad), das wiederum aus kleineren Backbones zusammengesetzt ist. Typische autonome Systeme sind Institutionen (Universitäten,...) und regionale Internet-Provider. AS können für sich regionale Netze, aber auch globale Netze betreiben. An den Backbones hängen regionale Netze, und an den regionalen Netzen hängen die Netze von Unternehmen, Universitäten, Internet Service Providern (ISP), usw. Beispielsweise wird der Zugriff eines in Deutschland platzierten Hosts über das Internet auf einen Server in den USA über mehrere Teilnetze transportiert. In der Abbildung 4-15 sind die Backbones aus den USA und Europa skizziert. Sie sind über transatlantische Standleitungen[2] verbunden. Die schwarzen Punkte repräsentieren IP-Router, von denen es eine ganze Menge in jedem AS gibt.

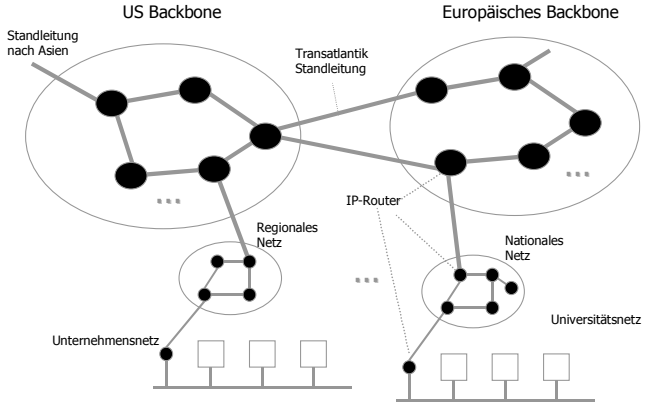


Abbildung 4-15: Das Internet als Sammlung vieler autonomer Teilnetze (Tanenbaum 2003a)

AS-Nummern. Autonome Systeme haben sich in den letzten Jahrzehnten unterschiedlich entwickelt und verfügen z.B. über verschiedene Routing-Strategien.

[2] Die Kabel sind tatsächlich durch den Atlantik gezogen.

Jedes AS hat eine eindeutige Nummer (*AS-Nummern*, *ASN*), von denen einige hier genannt werden sollen:

- AS11, Harward-University
- AS20633, Universität Frankfurt
- AS1248, Nokia
- AS2022, Siemens
- AS3680, Novell
- AS4183, Compuserve
- AS6142, Sun
- AS12816, Münchner Wissenschaftsnetz (MWN)

Die AS-Nummern sind 32 Bit lang[3] und werden von der IANA verwaltet. IANA delegiert die Zuteilung wiederum an regionale Registrierungsinstanzen wie RIPE NCC (Europa und Asien), ARIN (Nordamerika) und AfriNIC (Afrika). Ein ISP, der von IANA als solcher anerkannt werden will, muss mindestens mit zwei anderen AS über ein adaptives Routingprotokoll verbunden sein (mehr zum Routing weiter unten).

Kommunikation über Peering- und Transitabkommen

Tier-x-AS. Man unterscheidet je nach Größe und Bedeutung *Tier-1-*, *Tier-2-* und *Tier-3-AS*, die im Internet auch als *Provider* und gelegentlich auch als *Carrier* bezeichnet werden. Die Begriffe sind nicht eindeutig definiert und hängen von den Rollen ab, die ein AS im „Internet-Markt" ausfüllt:

- *Tier-1-AS* sind Betreiber von globalen Internet-Backbones. Hierzu gehören z.B. AT&T (US-amerikanischer Telekomanbieter), AOL (US-amerikanischer Online-Dienst, NTT (Nippon Telegraph and Telephone Corporation) und Verizon Communications (US-amerikanischer Telekomanbieter). Es gibt nur wenige Tier-1-AS[4].
- *Tier-2-AS* sind Betreiber großer, überregionaler Netze. Typische Tier-2-Provider sind die Deutsche Telekom, die France Telecom und Tiscali (Telekom-Unternehmen, Italien).
- *Tier-3-AS* sind kleinere, lokale Provider mit Endkundengeschäft. Hierzu gehören z.B. M-net in Bayern (Hauptgesellschafter sind die Münchner Stadtwerke), Hansenet (Hamburg) und Versatel (Berlin). Tier-3-AS werden auch als *Edge-Networks* bezeichnet.

Eine offizielle „Ernennung" zu einem Tier-1-AS gibt es nicht. Dies hängt von den Eigenschaften eines AS ab. Es gibt nur wenige Tier-1- und Tier-2-AS, aber viele Tier-3-AS, die, etwas vereinfacht, in Abbildung 4-16 dargestellt sind. Normalerweise benötigt ein Tier-3-AS eine Anbindung an einen Tier-2-AS, um im weltwei-

[3] Seit Januar 2009.

[4] Aktuell ca. 12 weltweit.

ten Internet mit allen anderen AS kommunizieren zu können. Dies ist aber nicht zwingend notwendig, da es auch andere Verbindungsmöglichkeiten zwischen AS gibt. Ebenso ist ein Tier-2-AS mit mindestens einem Tier-1-AS verbunden. Die Tier-1-AS bilden weitgehend das globale Backbone.

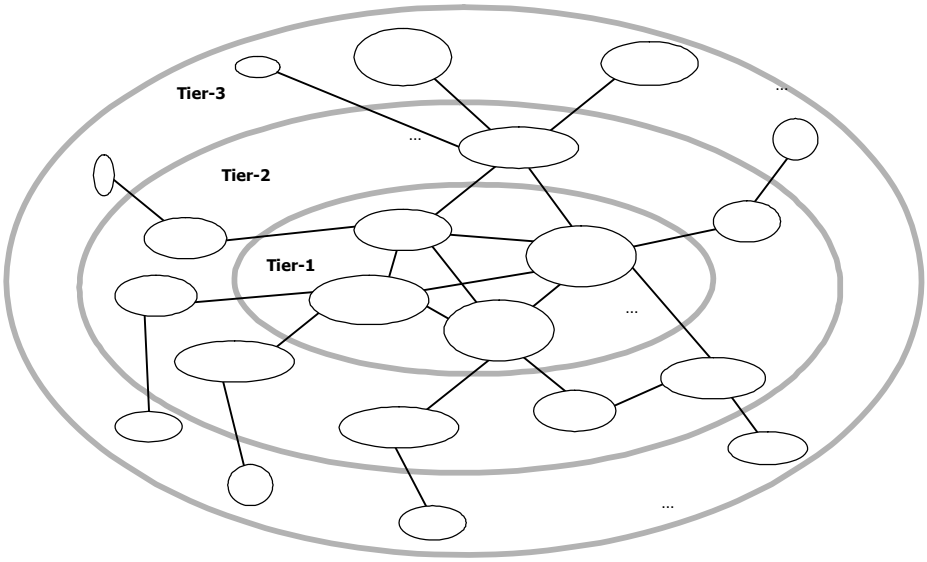


Abbildung 4-16: Vereinfachte Darstellung der Beziehungen von AS (Kurose 2008)

Internet-Knoten. Die größeren AS werden auch als *Internet-Knoten*, *Peering-Points*, *Internet Exchanges*, *Internet Exchange Points*, *Network Access Points* (*NAP*) oder kurz als *IX* bezeichnet. Sie stellen sog. *Peering-* oder auch *Transit-Dienste* zur Verfügung, die es kleineren AS ermöglichen, Daten auch mit anderen AS auszutauschen. AS, die nur Transit-Dienste anbieten und keine Transit-Dienste hinzukaufen, nennt man auch *Transit-AS*.[5]

Peering- und Transitabkommen. Die Provider vereinbaren, wie miteinander abgerechnet wird. Gleichgestellte Tier-1-AS führen üblicherweise einen kostenlosen Datenaustausch miteinander durch. Hier spricht man auch von *Peering-Abkommen*. Ansonsten werden die Kosten meist nach Transferleistung abgerechnet, und man spricht in diesem Fall von *Transit-Abkommen*. Auch Peering-Abkommen zwischen Tier-1- und Tier-2- oder sogar Tier-3-AS sind üblich, wenn beide Partner daraus einen Vorteil ziehen können. Je mehr Vereinbarungen ein AS mit anderen hat, desto besser sind die Kommunikationsverbindungen, die ein AS wiederum den bei

[5] Das Routing zwischen den Peering-Points erfolgt meist über das Border Gateway Protocol (BGP), siehe weiter unten.

ihm angeschlossenen AS bieten kann. Tier-3-AS sind normalerweise in der Rolle der Kunden, Tier-1-AS sind in der Rolle von Providern bzw. Peers, Tier-2-AS sind meist Provider für Tier-3-AS, können aber auch Peers zu anderen AS sein.

Öffentliche Peering-Punkte. In den letzten Jahren haben sich auch große öffentliche (public) Peering-Punkte entwickelt und werden oft von mehreren kleineren AS unterstützt. Nicht alle AS nutzen diese, aber mittlerweile wird doch sehr viel Internet-Verkehr über öffentliche Peering-Punkte abgewickelt.[6] Die *DE-CIX Management GmbH* betreibt z.B. den derzeit zweitgrößten europäischen IX in Frankfurt, *DE-CIX* genannt[7]. Weitere Internet-Knoten in Deutschland sind z.B. *B-CIX* (Berlin Commercial Internet Exchange) in Berlin sowie *ALP-IX* in München.

Die Abbildung 4-17 zeigt vereinfacht, wie heutige AS und öffentliche Internet Exchange Points über Peering- und Transit-Abkommen miteinander verbunden sind.

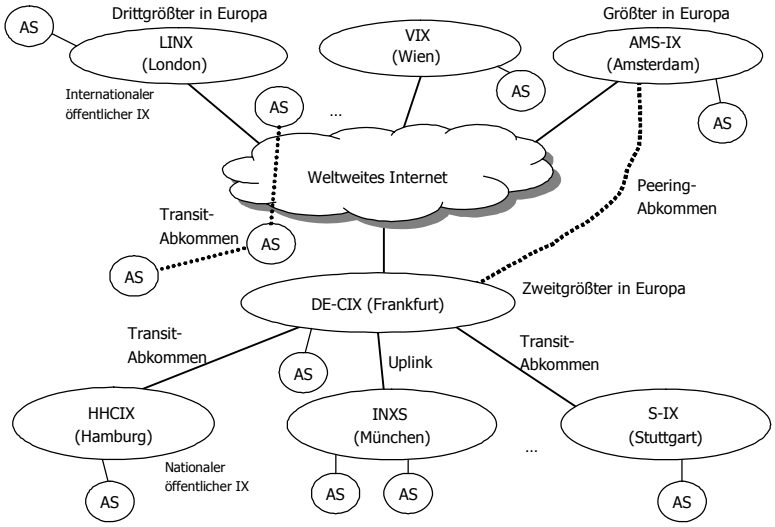


Abbildung 4-17: Vereinfachte Sicht auf die Verbindungen im Internet

Die autonomen Systeme werden je nach Marktrolle als *Peer*, *Provider* und/oder *Kunde* bezeichnet, je nachdem, wie sie agieren. In der Rolle eines Providers bieten sie anderen AS Zugang zum weltweiten Internet an und verlangen hierfür Gebühren. In der Kundenrolle nutzen sie Dienste anderer Provider. Wenn ein Provider

---

6 Die Top-80 Internet-Knoten in Europa sind unter der URL http://www.alrond.com/de zu finden, Stand: 23.05.2009.

7 Siehe http://www.de-cix.de, Stand: 23.05.2009.

nicht selbst Kunde ist und mit anderen Providern gleichberechtigten Datenaustausch betreibt, spricht man von einem Peer. Dies sind in der Regel Tier-1-AS.

Links. Die Anbindung eines AS an das globale Internet wird über Kommunikationsverbindungen, die als Links bezeichnet werden, erreicht. Je nachdem, wie ein AS an das Internet angebunden ist, unterscheidet man:

- *Stub AS:* Diese sind nur über einen Link an einen Provider (ein anderes AS) angebunden. Laut Internet-Richtlinien sollte dies gar nicht sein.
- *Multihomed-Stub-*AS*:* Diese sind aus Gründen der Ausfallsicherheit über mindestens zwei Links an einen Provider angebunden.
- Multihomed-AS*:* Diese AS sind zur Erhöhung der Ausfallsicherheit über mehrere Links an mindestens zwei Provider angebunden.
- *Transit-AS:* Transit-AS stellen Zwischensysteme dar, die für den Übergang von einem zum anderen AS dienen. Sie arbeiten als reine Transportnetze.

Die Abbildung 4-18 stellt die verschiedenen Varianten der Anbindung an das Internet nochmals dar.

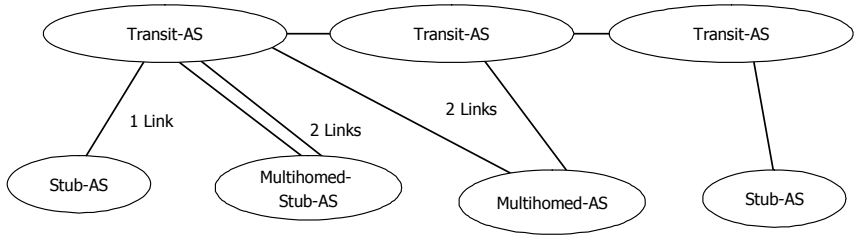


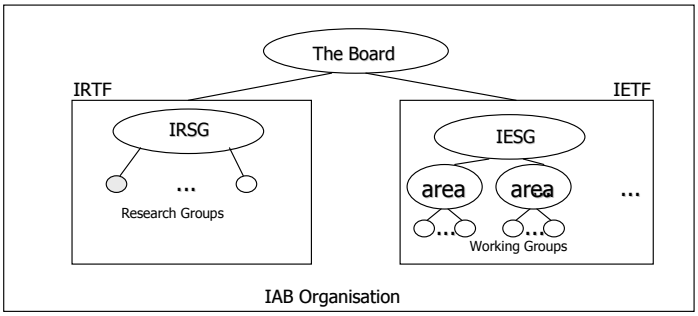
Abbildung 4-18: Beispiel für verschiedene Links von autonomen Systemen untereinander

4.2.3 Standardisierung im Internet

Für die Weiterentwicklung des Internet ist das IAB (Internet Activity, heute: Architecture Board) zuständig, das bereits *1983* gegründet und *1989* umorganisiert wurde. Das Board hat mehrere Bereiche und Gruppen (siehe Abbildung 4-19**):**

- Das *IAB* (Board) bestimmt die Richtlinien der Politik.
- Die *IETF* kümmert sich in verschiedenen Bereichen (areas) um kurz- und mittelfristige Probleme.
- Die *IESG* koordiniert die IETF Working Groups.
- Die *IRTF* ist ein Forschungsbereich, der die TCP/IP-Forschungsthemen koordiniert.
- Die *IRSG* koordiniert die Forschungsaktivitäten der einzelnen Gruppen.

Die Working Groups setzen sich aus freiwilligen Mitarbeitern zusammen.



IRTF: Internet Research Task Force
IETF: Internet Engineering Task Force
IRSG: Internet Research Steering Group
IESG: Internet Engineering Steering Group

Abbildung 4-19: Organisation des Internet Architecture Boards

Das NIC (Network Information Center, gesprochen „Nick“) ist zuständig für die Dokumentation und die Verwaltung der umfangreichen Information über Protokolle, Standards, Services, usw. Das NIC verwaltet das Internet und auch die Domänennamen (www.nic.net). In Deutschland ist die DENIC (www.denic.de) als nationale Vertretung eingerichtet.

4.2.4 Adressierung in Internet-basierten Netzen

Ein Rechnersystem wird im Internet als Host bezeichnet und besitzt in der Regel einen Hostnamen als symbolische Bezeichnung. Die IP-Adresse ist nicht an den Host, sondern an den Netzwerkzugang gebunden. Verfügt ein Host also z.B. über einen Ethernet-Adapter, so ist diesem in einem IP-Netzwerk eine IP-Adresse zugeordnet. Demnach ist es möglich, dass ein Host mehrere IP-Adressen besitzt, die jeweils die Netzzugänge eindeutig adressieren. Ein IP-Router verfügt z.B. mind. über zwei IP-Adressen.

IP-Adressen sind in der derzeit aktuellen IP-Version 4 (IPv4) 32 Bit lange Adressen. Es gibt also insgesamt 2^{32} = 4294967296 IP-Adressen. Eine Adresse besteht aus dem Tupel aus Netzwerknummer und Hostnummer. Die IP-Adressen von Quelle und Ziel werden in jedem IP-Paket mit übertragen.

Um unterschiedlich große Organisationen zu unterstützen, entschieden sich die Designer des Internet, den Adressraum in Klassen aufzuteilen. Man spricht hier von *klassenweiser Adressierung*. Man unterscheidet fünf verschiedene Adressformate, die in die Klassen A, B, C, D und E eingeteilt sind. Eine IP-Adresse wird in gepunkteten Dezimalzahlen (dotted decimal), jeweils ein Byte als Dezimalzahl zwischen 0 und 255, angegeben.

Beispiel: Die hexadezimale Darstellung einer IP-Adresse ist 0xC0290614. Die korrespondierende Dezimaldarstellung ist 192.41.6.20, mit 0xC0 = 192, 0x29 = 41 usw.

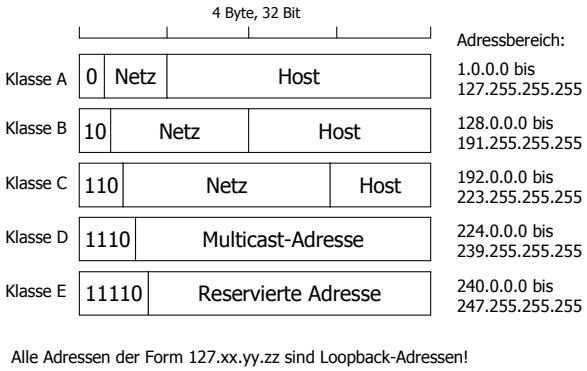


Abbildung 4-20: IP-Adressformate

Die Klasse einer Adresse erkennt man, wie in Abbildung 4-20 dargestellt, an den ersten Bit. Eine Klasse-A-Adresse erkennt man z.B. daran, dass sie mit einer binären Null beginnt, eine Klasse-B-Adresse beginnt mit binär 0b10 usw. Diese Information nutzen auch die Router aus, um eine Zieladresse zu interpretieren. In der Abbildung kann man erkennen, dass sich die Adressklassen im Wesentlichen darin unterscheiden, dass sie unterschiedliche Längen für die Netzwerknummern haben. Eine Klasse-A-Adresse hat z.B. nur ein Byte für die Netzwerknummer, während eine Klasse-B-Adresse zwei Byte und eine Klasse-C-Adresse drei Byte für die Netzwerknummer hat.

Heute spricht man auch von /8-Netzwerken, wenn man Klasse-A-Netzwerke meint und entsprechendes gilt für die Klasse B (/16) und die Klasse C (/24). Die Zahl hinter dem Schrägstrich gibt die Anzahl der Bit für die Netzwerknummer an. Die Schreibweise wird auch als *Präfix-Längen-Schreibweise* oder *Präfix-Notation* bezeichnet:

- Es gibt insgesamt ($2^7 - 2$) Netzwerke der Klasse A, da sieben Bit für die Netzwerknummer reserviert sind. Jedes Klasse-A-Netzwerk kann ($2^{24} - 2$) Hostadressen unterstützen. Bei der Berechnung werden zwei Netzwerkadressen abgezogen, da das Netzwerk 0.0.0.0 als Standard-Route und das Netzwerk 127.0.0.0 als Loopback-Adresse verwendet wird. Auch beim Hostanteil dürfen nicht alle Adressen ausgenutzt werden. Die Adressen mit nur Nullen und nur Einsen haben eine Sonderbedeutung.
- Für Klasse-B-Netze stehen 14 Bit für den Netzwerkanteil zur Verfügung, zwei Bit („10“) sind zur Identifikation der Klasse reserviert. Dies ergibt 16.384 Klasse-B-Netze. Jedes Klasse-B-Netzwerk kann maximal ($2^{16} - 2$) Hosts enthalten.

– In der Klasse C stehen 21 Bit für den Netzwerkanteil zur Verfügung (drei Bit sind für die Identifikation reserviert („110“). Dies ergibt ($2^{21} - 2$) verfügbare Klasse-C-Netzwerke.

In Tabelle 4-1 sind die Adressumfänge nochmals grob zusammengefasst.

Tabelle 4-1: Adressumfänge der IP-Klassen

Klasse	Anzahl Netze	Max. Anzahl Hosts je Netz	Anteil am IP-Adressraum	Adressen insgesamt
A (/8)	126 ($2^7 - 2$)	16777214 ($2^{24} - 2$)	50 %	2147483638 (2^{31})
B (/16)	16384 (2^{14})	65534 ($2^{16} - 2$)	25 %	1073741824 (2^{30})
C (/24)	2097152 (2^{21})	254 ($2^8 - 2$)	12,5 %	536870912 (2^{29})

Damit sind 87,5 % des IP-Adressraums festgelegt. Der Rest wird den Klassen D und E sowie den Adressen zur privaten Nutzung zugeordnet. Die Klasse D ist für Multicasting-Anwendungen reserviert, und die Klasse E wird im Moment nicht benutzt. Einige IP-Adressen haben eine besondere Bedeutung und sollen hier erwähnt werden. Hierzu gehören:

– Die niedrigste IP-Adresse ist 0.0.0.0. Diese Adresse hat die besondere Bedeutung „ein bestimmtes Netz“ oder „ein bestimmter Host“ und wird evtl. beim Bootvorgang eines Hosts benötigt.
– Eine Adresse mit einer Netzwerknummer bestehend aus lauter binären Einsen und einem beliebigen Hostanteil ermöglicht es Hosts, auf das eigene Netzwerk zu verweisen, ohne die Netzwerknummer zu kennen.
– Die höchste IP-Adresse ist 255.255.255.255 (–1). Sie wird als Broadcast-Adresse verwendet. Beim Broadcast werden alle Hosts eines Netzwerks mit einem Datagramm angesprochen. Alle Hosts nehmen also Pakete mit der Zieladresse 255.255.255.255 vom Netz und leiten sie an die Anwendungsprozesse weiter, die darauf warten. Wenn die darunterliegende Schicht 2 Broadcasting unterstützt, dann geht auch physikalisch nur ein Paket durch das Netz. Dies ist für bestimmte Anwendungen sehr interessant, da die Netzwerkbelastung niedriger als bei Einzelnachrichten ist.
– Alle Adressen, die mit 127. beginnen, sind sog. Loopback-Adressen und für die interne Hostkommunikation reserviert. Pakete mit Zieladressen in diesem Bereich werden nicht auf das Netz gelegt.

Broadcast

Beim *Broadcasting* unterscheidet man nochmals zwischen direktem (gerichteten, directed) und begrenzten (limited) Broadcast. Der direkte Broadcast ermöglicht das Senden einer Broadcast-Nachricht an ein beliebiges Netzwerk im Internet und zwar direkt von einem Host eines anderen Netzwerks aus. Der zuständige Router

sendet das Paket über den entsprechenden Pfad zum Zielrouter, der dann den Broadcast in seinem lokalen Netzwerk sendet.

Die Adresse für den direkten Broadcast enthält die Netzwerknummer und im Hostteil lauter binäre Einsen. Wie in Abbildung 4-21 dargestellt, sendet der Host H_1 einen Broadcast an ein anderes Netz, in dem die zwei Hosts H_2 und H_3 die Nachricht erhalten. Sowohl das Quell-, als auch das Zielnetz sind Klasse-C-Netze. Die Ziel-Broadcast-Adresse in „dotted decimal" ist 130.6.20.255. Ein begrenzter Broadcast bezieht sich auf das lokale Netzwerk und wird von den Routern nicht durchgelassen. Die „limited" Broadcast-Adresse ist in diesem Fall für alle Netze gleich und ist – wie bereits weiter oben erläutert – 255.255.255.255. Aus den Ausführungen zu den Spezialadressen ergibt sich, dass es keine Host-Adresse geben kann, dessen Hostanteil nur aus binären Einsen oder Nullen besteht.

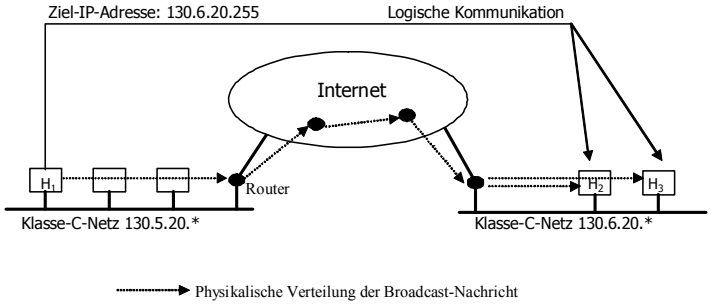


Abbildung 4-21: Beispiel für einen direkten Broadcast

Das Weiterleiten von directed Broadcast-Nachrichten ist in vielen Routern deaktiviert, da es für Angriffe (z.B. Smurf-Attacken) genutzt werden kann.[8]

Private Adressen

Eine weitere Besonderheit sind die sog. *privaten IP-Adressen*. Private Adressen sind im RFC 1918 definiert. Hierzu gehören:

- 10.0.0.0 mit einem Netzwerkanteil von 8 Bit (Netzwerkmaske 255.0.0.0/8, Bereich 10.x.x.x)[9]
- 172.16.0.0 mit einem 12-Bit-Netzwerkanteil (Netzwerkmaske 255.240.0.0/12, Bereich 172.16.x.x -172.31.x.x)

[8] Bei einem Smurf-Angriff sendet ein Angreifer Nachrichten (siehe ICMP unten) an die directed Broadcast-Adresse eines Netzwerks.

[9] Die Hintergründe der Netzwerkmaske werden etwas weiter unten in diesem Abschnitt aufgeklärt.

– 192.168.0.0 mit einem Netzwerkanteil von 16 Bit (Netzwerkmaske 255.255.0.0/16, Bereich 192.168.x.x)

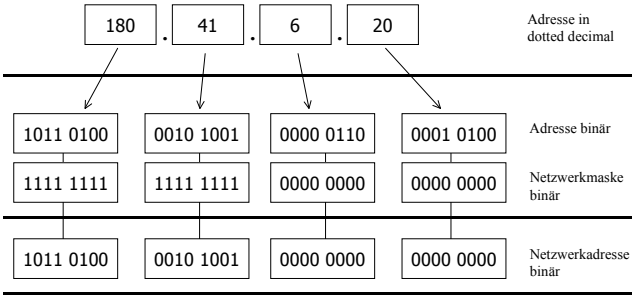
Hierbei handelt es sich um Adressen, die jeder Router besonders behandelt. Pakete mit einer derartigen Zieladresse werden von den Routern nicht weitergeleitet und verlassen daher niemals das lokale Netzwerk. Diese Adressen können von jedem Netzwerk für interne Zwecke verwendet werden und sind daher nicht mehr global eindeutig. Heute nutzt man die privaten Adressen gerne für die Adressvergabe im Intranet, während man nach außen hin zum globalen Internet ein Masquerading über NAT (wird noch später erläutert) ausführt.

Netzwerkmasken und Routing

Die Router müssen wissen, welche Adressklasse in einem IP-Paket als Ziel adressiert ist. Da aber die Adressklasse nicht im IP-Header steht, muss der Router diese Information auf andere Weise herausfinden. Die Lösung ist, dass in den Routing-Tabelleneinträgen noch eine zusätzliche Information mit aufgenommen wird. Es handelt sich um die *Netzwerkmaske*.[10].

Die Netzwerkmaske ist ein 32 Bit breites Feld, in dem für jedes Adressbit genau ein Bit zugeordnet ist. Steht ein Bit auf Binär ‚1', so ist das Bit in der IP-Adresse der Netzwerknummer zugeordnet. Bei den klassenbehafteten A/B/C-Adressen können dies natürlich nur alle Bit der Netzwerknummer sein.

Wenn ein Router ein Paket empfängt, legt er die Netzwerkmaske über die IP-Adresse und ermittelt so die Netzwerk- und die Hostadresse. In Abbildung 4-22 ist ein Beispiel für die Nutzung der Netzwerkadresse angegeben.



Logische Und-Verknüpfung der IP-Adresse mit der Netzmaske
Die Netzwerkadresse ist demnach: 180.41.0.0
Netzwerkmaske in dotted decimal: 255.255.0.0

Abbildung 4-22: Netzwerkmaske und deren Einsatz

[10] Routing-Tabellen werden auch in den Hosts geführt. In Windows-Systemen kann man sich z.B. mit dem Kommando *route print* die aktuellen Routing-Tabelleneinträge in einem Host anschauen.

Ein Router führt eine logische Und-Operation aus und ermittelt so die Netzwerknummer bzw. Netzwerkadresse. Im Beispiel handelt es sich um eine Klasse-B-Adresse.

Routing-Protokolle übermitteln heute noch die vier Byte lange Netzwerkmaske in Ihren PDUs.

4.2.5 Subnetze und deren Adressierung

Durch die Klasseneinteilung ergab sich eine gewisse Verschwendung von Adressen. Die zweistufige Adressierung (Netzwerknummer, Hostnummer) führte zudem zu Problemen, wenn eine Organisation sein internes Netzwerk strukturieren wollte. Jede Aufgliederung des Netzwerks ging nur über die Beantragung einer zusätzlichen Netzwerknummer. Die Routing-Tabellen in den Backbone-Routern begannen zu wachsen.

Im RFC 950 wurden schließlich im Jahre 1985 Teilnetze (Subnets) definiert, um die organisatorischen Probleme abzumildern und auch um die Routing-Tabellen im globalen Internet zu entlasten. Durch die *Subnetzadressierung* wurde eine bessere organisatorische Gliederung der internen Netze ermöglicht.

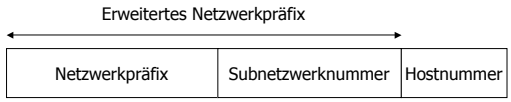


Abbildung 4-23: Erweitertes Netzwerkpräfix bei Subnetzadressierung

Die Hostadresse wird hierbei zur Unterteilung des Netzwerks in mehrere Unternetze (Subnetze) in zwei Teile zerlegt und zwar in die Teilnetznummer (Subnet) und die Hostnummer, woraus sich insgesamt eine dreistufige Hierarchie ergibt. Die gesamte IP-Adresse wird bei Subnetting über das Tupel *(Netzwerknummer, Subnetzwerknummer, Hostnummer)* angegeben (siehe Abbildung 4-23).

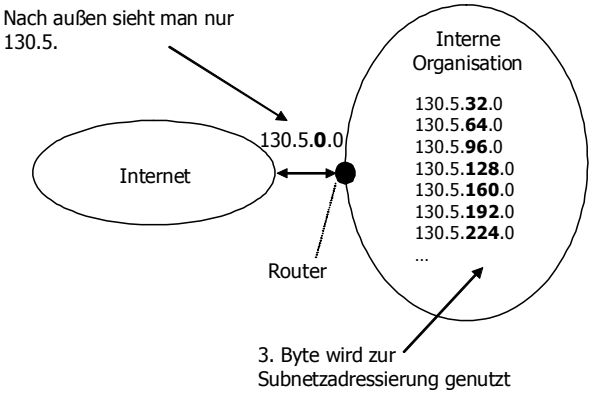


Abbildung 4-24: Reduktion der Routereinträge durch Subnetze

Wie die Abbildung 4-24 zeigt, wird außerhalb des Teilnetzes die Aufgliederung allerdings nicht sichtbar. Im globalen Internet gibt es also nur einen Routeneintrag für die Netzwerknummer, nicht aber für die Subnetzwerknummern. Interne IP-Router berücksichtigen aber die Subnetzadresse. Die Netzwerkmaske wird auch hier als Bitmaske verwendet, um die Bit der Subnetzwerknummer zu identifizieren. Insgesamt sind im Beispiel 254 Subnetze möglich, da die Subnetzwerknummer die Werte 1 bis 254 annehmen kann.

Der lokale Administrator besitzt alle Freiheiten zur Bildung von Subnetzen, ohne die Komplexität auf die externen Internet-Router zu übertragen. Bei einer Klasse-B-Adresse wäre folgende Struktur denkbar:

- Die ersten beiden Byte sind die Netzwerknummer.
- Das dritte Byte gibt die Organisationseinheit im Unternehmen an (Subnetz-Adresse).
- Das vierte Byte erlaubt die Nummerierung der Hosts.

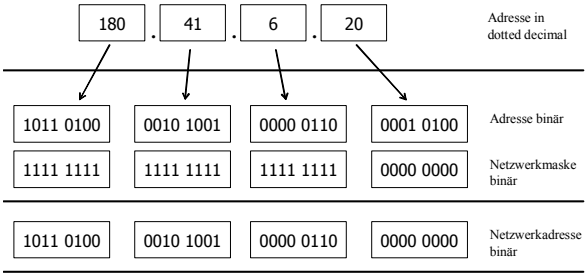


Abbildung 4-25: Einsatz der Subnetzadressierung

Damit erkennt ein Administrator schon an der Adresse, welcher Organisationseinheit ein Host zugeordnet ist. Durch die Subnetzadressierung ergeben sich zwar nicht mehr Adressen, aber sie trägt wesentlich zur unternehmensinternen Strukturierung bei, ohne dass dies nach außen dringt. Betrachten wir das Beispiel in Abbildung 4-25. Hier ist die Netzmaske 255.255.255.0 bei einer Klasse-B-Adresse. Das dritte Byte dient also der Subnetz-Bildung und das vierte bleibt für die Hostnummer.

Die Nutzung von Adressteilen für die Subnetzbildung ist nicht an die Byte-Grenzen gebunden. Die Klasse-C-Adresse 193.1.1.0 kann man z.B. mit einer Netzwerkmaske von 255.255.255.224 (/27) ausstatten. In diesem Fall sind die drei höherwertigen Bit des Hostanteils für das Subnetzwerk reserviert, und es bleiben noch 5 Bit für den Hostanteil übrig (siehe Abbildung 4-26).

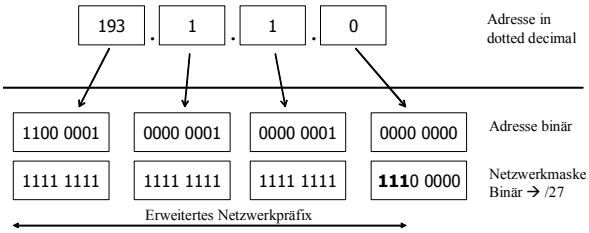


Abbildung 4-26: Subnetzwerk mit 27-Bit erweitertes Netzwerkpräfix

Damit lassen sich in diesem Beispiel acht /27-Teilnetze bilden, die folgende Subnetz-Adressen zugeordnet bekommen:

```
Basisnetz: 11000001.00000001.00000001.00000000 = 193.1.1.0/24
Teilnetz 0:11000001.00000001.00000001.00000000 = 193.1.1.0/27
Teilnetz 1:11000001.00000001.00000001.00100000 = 193.1.1.32/27
Teilnetz 2:11000001.00000001.00000001.01000000 = 193.1.1.64/27
Teilnetz 3:11000001.00000001.00000001.01100000 = 193.1.1.96/27
Teilnetz 4:11000001.00000001.00000001.10000000 = 193.1.1.128/27
Teilnetz 5:11000001.00000001.00000001.10100000 = 193.1.1.160/27
Teilnetz 6:11000001.00000001.00000001.11000000 = 193.1.1.192/27
Teilnetz 7:11000001.00000001.00000001.11100000 = 193.1.1.224/27
```

Das Basisnetz 193.1.1.0/24 wird also in acht Teilnetze mit jeweils 30 nutzbaren IP-Adressen aufgeteilt. Dies ergibt insgesamt 240 Adressen. Die Adressen mit lauter Nullen und Einsen können in keinem /27-Netzwerk genutzt werden. Es sind also von den maximal möglichen 254 Adressen des /24-Subnetzes weitere 14 für die Nutzung ausgeschlossen. Defakto verliert man also durch die Subnetz-Aufteilung ein paar IP-Adressen.

Subnetting wird auch als FLSM-Subnetting (Fixed Length Subnet Masks) im Gegensatz zu VLSM (das im Folgenden erläutert wird) bezeichnet. Damit ist gemeint, dass eine eingestellte Subnetz-Adresse in einem Netzwerk starr ist. Teilnetz 0 hat im Beispiel die Adresse 193.1.1.0/27. Im ganzen Subnetz muss diese Subnetz-Adresse gelten, und sie kann auch nicht mehr weiter unterteilt werden.

4.2.6 VLSM und CIDR

In den Routing-Tabellen der wichtigen Internet-Router, der sog. Backbone-Router, ist seit den 90er Jahren ein exponentielles Wachstum zu verzeichnen. Im Jahre 1990 gab es etwas mehr als 2.000 Routen, 1992 bereits mehr als 8.000 und 1995 waren es schon mehr als 30.000 Routeneinträge in den Backbone-Routern.

Das Internet wächst immer mehr (in den 90er Jahren verdoppelte sich seine Größe in weniger als einem Jahr), was vor allem durch die enorme Nutzung des World Wide Web verursacht wird. Es zeichnete sich bereits seit Anfang der 90er Jahre

aufgrund des nicht effektiv ausgenutzten Adressraums eine gewisse *Adressenknappheit* aus.

Die Netzwerknummern wurden zudem in der Vergangenheit nicht sehr effizient verteilt. Klasse A- und B-Adressen wurden an Organisationen vergeben, die überhaupt nicht so viele Adressen benötigten. Aufgrund der Klasseneinteilung kann allerdings auch tatsächlich nicht der ganze Adressbereich sinnvoll ausgenutzt werden. Beispielsweise musste ein Unternehmen mit nur 10 Rechnern eine Klasse-C-Adresse nutzen und vergeudete damit 244 (256-10-2) IP-Adressen[11], die anderweitig genutzt werden könnten. Dies ist auch ein Grund dafür, warum IP-Adressen knapp sind. Daher werden heute IP-Adressen vom NIC bzw. in Deutschland von der DENIC etwas restriktiver zugewiesen.

Einige Verbesserungsvorschläge für die Adressenproblematik wurden erarbeitet:

- Mit dem *VLSM-Konzept* (Variable Length Subnet Masks) wurden variable Längen der Subnetz-Masken innerhalb einer Organisation möglich. Eine bedarfsgerechte Aufteilung des Adressraums im Intranet wird dadurch unterstützt.
- Mit *CIDR* (Classless InterDomain Routing), auch als „classless IP" bezeichnet, wurde das Konzept der Klasseneinteilung auch im globalen Internet fallen gelassen. VLSM wird damit auch von den Routern im globalen Internet unterstützt. Damit wird eine bedarfsgerechte Aufteilung des Adressraums im globalen Internet vereinfacht.
- In der Praxis schafft *NAT* (Network Address Translation) auch eine gewisse Erleichterung.
- Die neue Version des Internet-Protokolls *IPv6* mit Adresslängen von 128 Bit soll in Zukunft richtige Abhilfe schaffen.

VLSM (Variable Length Subnet Masks) besagt prinzipiell, dass einem IP-Netzwerk mehr als eine Netzwerkmaske bzw. eine variabel lange Teilnetzmaske zugewiesen werden kann. Damit kann eine Organisation den ihr zugewiesenen Adressraum noch effektiver als mit Subnetzadressen nutzen. Eine feinere Aufteilung der Adressen auf interne Netze ist möglich, was wiederum hilft, Adressen einzusparen.

Man ergänzt nach der IP-Adresse einen Schrägstrich und dahinter die Anzahl an Bit, die (linksbündig) für die Netzwerknummer verwendet werden. Die Notation lautet also wie folgt: *<Netzwerkadresse>/<Anzahl Masken-Bit>*

Beispieladressen:

- 180.41.6.0/16 ist eine klassische Klasse-B-Adresse mit zwei Byte für die Netzwerknummer und zwei Byte für die Hostnummer.
- 180.41.6.0/24 ist ein Pendant zur in Abbildung 4-25 gezeigten Subnetzadresse mit einem Netzwerkanteil von 24 Bit und 8 Bit für die Hostnummer.

[11] Zwei Adressen (lauter Nullen oder Einsen) können nicht für Hosts genutzt werden.

- 180.41.6.0/25 entspricht vom Adressbereich einer halben Klasse-C-Adresse. Es verbleiben 7 Bit für den Hostanteil.
- 180.41.6.0/28 belässt nur noch 4 Bit für die Hostnummer und kann für kleine Netze eingesetzt werden. Damit wird ein Klasse-C-Netzwerk noch einmal unterteilt. Insgesamt könnte man mit /28 genau 16 kleine Netze dieser Art unterstützen. Davon können aber zwei nicht genutzt werden.

Subnetz-Organisation bei VLSM

Der Vorteil von VLSM für eine Organisation soll an einem typischen Klasse-B-Netzwerk (/16-Netzwerk) beispielhaft diskutiert werden: Arbeitet man ohne interne Subnetze, hat man nur ein Netzwerk für die ganze Organisation. Wenn man z.B. Subnetting mit einem /22-Netzwerkpräfix verwendet, so bedeutet dies, dass 64 Subnetze (6 Bit) gebildet werden können. Jedes dieser Subnetze kann über 1.022 ($2^{10} - 2$) Hostadressen verfügen. Wenn allerdings ein Teilnetz der Organisation weniger als 1.022, z.B. nur 50 Hostadressen benötigt, ist die Aufteilung bei Subnetting auch viel zu starr. Mehr als eine Teilnetzwerkmaske ist nicht möglich, und es werden viele IP-Adressen verschenkt.

Wäre die Teilnetzmaske noch weiter unterteilbar, so könnte man unterhalb des /22-Teilnetzes z.B. noch ein /26-Teilnetz bilden und hätte für die Abteilung mit 50 Hosts eine vernünftige Größe ($2^6 - 2 = 62$ Adressen) ohne zu große Adressenverschwendung. VLSM erlaubt dies, indem der Adressraum einer Organisation rekursiv aufteilbar ist. Zusätzliche, über die Klasse-B-Netzwerkadresse hinausgehende Routinginformation wird aber nicht nach außen ins globale Internet gegeben, die Subnetze sind also nur in der Organisation bekannt, und die Router werden nicht belastet.

Beispiel: In Abbildung 4-27 ist ein Beispiel für eine hierarchische Aufteilung des Teilnetzes 11.0.0.0/8 skizziert. Das Teilnetz ist zunächst in 254 /16-Teilnetze unterteilt und diese wiederum in /24-Teilnetze. In der Abbildung sind auf der untersten Ebene schließlich sechs nutzbare /27-Teilnetze unter 11.1.2.0/24 dargestellt. Die anderen Teilnetze können ebenfalls untergliedert werden und auch die /27-Teilnetze sind weiterhin unterteilbar.

Mit VLSM ist eine bedarfsgerechte Aufteilung des Adressraums möglich. Bei der Netzwerkplanung muss man sich überlegen wie viele Subnetze und wie viele IP-Adressen man innerhalb eines Subnetzes benötigt. Daraus ergibt sich die Anzahl der Bit, die man noch dem Netzwerkpräfix hinzufügen muss. Hat man beispielsweise von seinem ISP die Adresse 143.26.0.0/16 für die eigene Organisation zugewiesen bekommen und die Organisation ist in 14 Subnetze mit jeweils max. 4.000 Rechnern strukturiert, so ist es sinnvoll, vier Bit für die Subnetz-Adressierung zu verwenden. Damit kann man nämlich 14 Subnetze bilden und erhält jeweils ein /20-Subnetz mit 4.094 IP-Adressen. Gibt es einige kleinere Subnetze in der Organisation, kann man ein /20-Subnetz auch nochmals unterteilen.

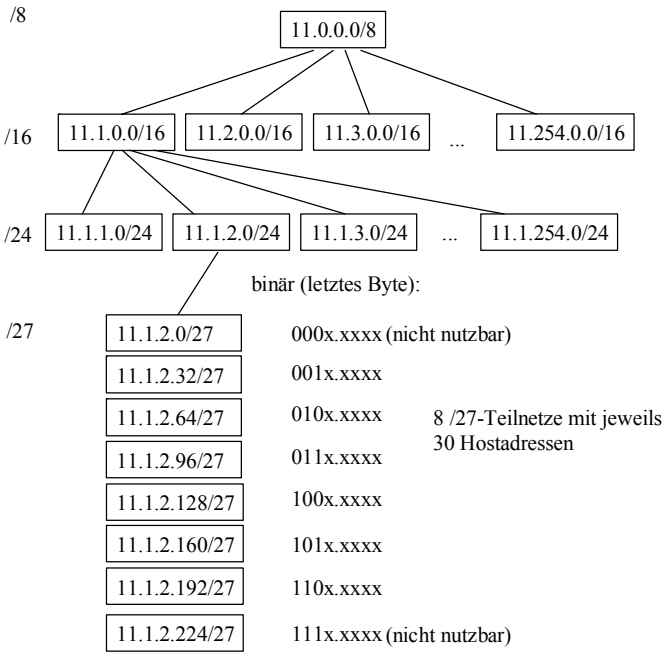


Abbildung 4-27: Beispiel für hierarchische VLSM-Aufteilung eines Netzwerkbereichs

Routing bei VLSM

Die internen Router der Organisation müssen aber VLSM unterstützen. RIP-1, eines der bekanntesten Routing-Protokolle, überträgt auch beim Informationsaustausch zwischen den Routern keine Netzwerkmaske, und damit wird die Subnetz-Information nicht übertragen. RIP-1 ermitteln die Netzwerkmasken für nicht bekannte Netzwerke anhand der Netzwerknummer, die sich aus der Klassenzugehörigkeit ergibt. Es sind also erweiterte Routingprotokolle erforderlich. Die Folgeversion von RIP-1 wird als RIP-2 bezeichnet und unterstützt VLSM bereits. Auch das Routing-Protokoll OSPF unterstützt bereits das Weiterleiten des erweiterten Netzwerkpräfix.[12]

Beispiel: In Abbildung 4-28 ist ein Beispiel für eine strukturierte Nutzung von VLSM in einer internen Organisation gegeben. Der Router, der ins globale Internet routet, gibt nur eine Route (11/8) für die der Organisation zugewiesene Netzwerknummer bekannt.

[12] Die Routing-Protokolle werden noch erläutert. RIP und OSPF werden innerhalb von autonomen Systemen verwendet.

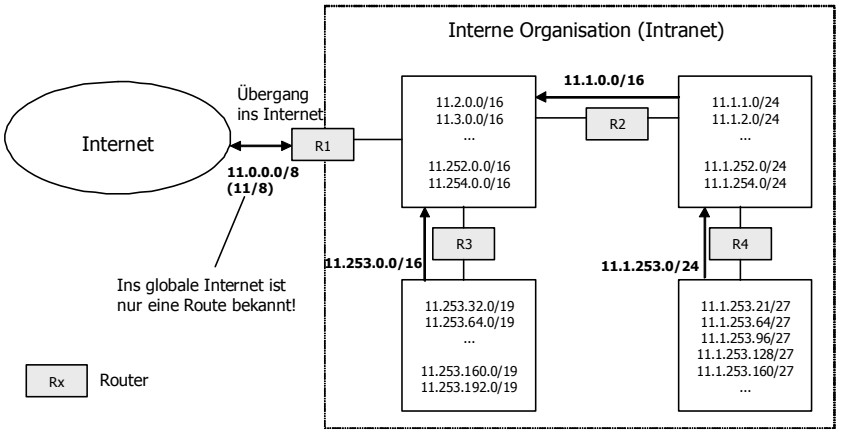


Abbildung 4-28: VLSM-Beispielkonfiguration

Auch intern fassen Router die Teilnetze zusammen. Der Router R4 fasst z.B. sechs /27-Teilnetze zusammen. Nach außen gibt der Router nur Informationen über das Teilnetz 11.1.253.0/24. Router R2 stellt R4 also alle Pakete durch, die an das gesamte Teilnetz 11.1.253.0/24 adressiert sind. Wie R4 weiterroutet ist R2 nicht bekannt. Auch der Router R2 fasst alle hinter ihm liegenden Routen in der Teilnetzwerkadresse 11.1.0.0/16 und R3 alle Teilnetze zur Route 11.253.0.0/16 zusammen.

CIDR im globalen Internet:

CIDR[13] (RFCs 1518/1519) ist ein Internet-Standard (RFC-Protokolltyp = Proposed-Standard-Protokoll) und bedeutet "Classless Inter Domain Routing". Mit diesem Konzept begegnet man der Adressknappheit auch im globalen Internet. CIDR funktioniert wie VLSM, die Netzwerknummern sind also nicht starr nach Klassen sondern flexibel, bitweise einstellbar. Mit CIDR wird eine IP-Adresse wie bei VLSM in Netzwerkpräfix-Notation beschrieben. Damit hat man im globalen Internet neben einer optimaleren Adressenorganisation auch die Vorteile der Routen-Aggregation zur Entlastung der Routing-Tabellen.

Das Konzept von CIDR geht noch etwas weiter:

- Die restlichen, noch nicht vergebenen Klasse-C-Netze (bei der CIDR-Einführung ca. 2 Millionen) werden in Blöcken variabler Länge vergeben.
- Die Vergabe der Blöcke wird von den ISPs (Internet Service Provider) verwaltet.
- CIDR unterstützt sog. *geographische Zonen* für die verbleibenden Klasse-C-Netze.

[13] CIDR wird als „caider" ausgesprochen.

Braucht z.B. ein Standort (Unternehmen, Institut) 2.000 Adressen, erhält er vom ISP acht aufeinanderfolgende Klasse-C-Netze zugewiesen, was 2.048 Adressen entspricht. Damit kann auf eine Klasse-B-Adresse verzichtet werden. Braucht ein Unternehmen weniger Adressen, als mit einer Klasse-C-Adresse adressierbar sind, so wird auch die Klasse-C-Adresse vom ISP noch einmal unterteilt und man spart erneut Adressen ein.

CIDR hat sich heute zum Standard entwickelt und wird von allen Routern und den entsprechenden Routing-Protokollen unterstützt. Alle Router müssen einen Weiterleitungsalgorithmus implementieren, der auf der längst möglichen Übereinstimmung der Netzwerkmaske basiert. Die Routing-Protokolle müssen die Netzwerkpräfixe mit der Routing-Information (Routenankündigung) übertragen.

Beispiel für sinnvolle Netzwerkgestaltung mit CIDR/VLSM: Der Adressbereich 180.41.224.0/24 kann mit CIDR und VLSM vielfältig aufgeteilt werden. In der Tabelle 4-2 ist eine beispielhafte Aufteilung dargestellt. Bräuchte man etwa einen Adressbereich mit 25 Hostadressen, käme z.B. 180.41.224.192/27 mit insgesamt 30 gültigen Hostadressen in Frage. Man könnte diesen Adressbereich auch noch in zwei Bereiche zu je 14 Adressen aufteilen. Dies funktioniert, indem man statt 27 Bit nun 28 Bit für die Maske verwendet, woraus sich folgende Bereiche ergeben:

- 180.41.224.192/28 (gültige Adressen 180.41.224.193 bis 180.41.224.206)
- 180.41.224.208/28 (gültige Adressen 180.41.224.209 bis 180.41.224.222)

Durch die Aufteilung verliert man aber nochmals zwei Hostadressen, da in beiden Bereichen nun jeweils 14 Hostadressen gültig sind. Dies reicht aber immer noch, um die Anforderung zu erfüllen.

Tabelle 4-2: Beispiel für eine Aufteilung des Adressraums mit CIDR/VLSM

Adressbereich	Binäre Netzwerkadresse	von Adresse	bis Adresse	Adressen
180.41.224.0/25	**11000000.00101001.11100000.0**0000000	180.41.224.0	180.41.224.127	128 - 2
180.41.224.128/26	**11000000.00101001.11100000.10**000000	180.41.224.128	180.41.224.191	64 – 2
180.41.224.192/27	**11000000.00101001.11100000.110**00000	180.41.224.192	180.41.224.223	32 – 2
180.41.224.224/28	**11000000.00101001.11100000.1110**0000	180.41. 224.224	180.41.224.239	16 – 2
180.41.224.240/29	**11000000.00101001.11100000. 11110**000	180.41.224.240	180.41.224.247	8 – 2
180.41.224.248/30	**11000000.00101001.11100000. 11111**00	180.41.224.248	180.41.224.251	4 – 2
180.41.224.252/30	**11000000.00101001.11100000. 11111**00	180.41.224.252	180.41.224.255	4 – 2

Die Tabelle 4-2 zeigt eine typische Aufteilung eines Klasse-C-Netzwerks in weitere Subnetze. Jede Adresse darf natürlich nur einmal vorkommen. Überschneidungen

darf es nicht geben, und der Adressbereich darf natürlich auch nur einmal vergeben werden. Die Vergabe der Adressen muss entsprechend geplant werden. Das Klasse-C-Netzwerk wird in zwei Subnetze mit 126 Adresse (/25) aufgeteilt. Das erste mit der Adresse 180.41.224.0/25 wird nicht mehr weiter untergliedert und das zweite mit der Adresse 180.41.224.128/25 wird weiter zerlegt. Schrittweise werden dem Netzwerkpräfix immer mehr Bit hinzugefügt. Die fett dargestellten Bit sind dem Netzwerkpräfix zugeordnet, die restlichen Bit können für den Hostanteil verwendet werden.

Geographische Zonen:

Ein positiver Nebeneffekt ergibt sich bei Einsatz von CIDR durch eine Verbesserung im globalen Internet-Routing und zwar nicht nur durch die Routen-Aggregation.

Die verbleibenden Klasse-C-Netze werden in acht gleich große Adressblöcke mit einer Größe von 131.072 Adressen unterteilt und sog. *Areas* (geographische Zonen) zugeordnet (RFC 1466). Aktuell sind u.a. vier Areas mit folgenden Adressen aus dem Klasse-C-Adressbereich festgelegt:

- Europa: 194.0.0.0 bis 195.255.255.255
- Nordamerika: 198.0.0.0 bis 199.255.255.255
- Zentral- und Südamerika: 200.0.0.0 bis 201.255.255.255
- Pazifik-Länder: 202.0.0.0 bis 203.255.255.255

Durch eine feste Zuordnung von Klasse-C-Adressbereichen zu geographischen Zonen kann nun z.B. ein europäischer Router anhand der Zieladresse feststellen, ob ein Paket in Europa bleibt oder z.B. direkt zu einem amerikanischen Router weitergeleitet werden soll. Damit wird eine Optimierung der Wegewahl erreicht.

31-Bit-Präfixes. Bei VLSM/CIDR werden durch die Nichtausnutzung von /31-Subnetzen IP-Adressen verschenkt. Tatsächlich erhält man z.B. aus einem Klasse-C-Netzwerk weniger IP-Adressen als bei der klassenweisen Adressvergabe. Ursprünglich war die Nutzung eines /31-Subnetzes nicht zulässig, da hiermit nur zwei Hostadressen gebildet werden können. Die erste besteht aus einer binären ‚0' und die zweite aus einer ‚1' im Hostanteil. Adressen mit lauter Nullen oder Einsen sind aber nicht für Hostadressen zugelassen, da sie mit Spezialadressen wie der Broadcastadresse in Konflikt stehen. Im RFC 3021 ist die Nutzung von 31-Bit-Präfixes aber nun eingeschränkt zugelassen worden. Als sinnvolle Anwendung ist der Einsatz bei der Verbindung zweier Router angegeben. Damit können nun auch diese Adressen eingesetzt werden, was als Mittel zur Linderung der Adressknappheit genutzt werden kann.

4.2.7 IP-Protokoll-Header

In diesem Abschnitt werfen wir einen Blick auf die IP-PCI, also den IP-Header mit all seinen Feldern. Wie in Abbildung 4-29 unschwer zu erkennen ist, handelt es sich aufgrund des variabel langen Optionsteils um einen variabel langen Header.

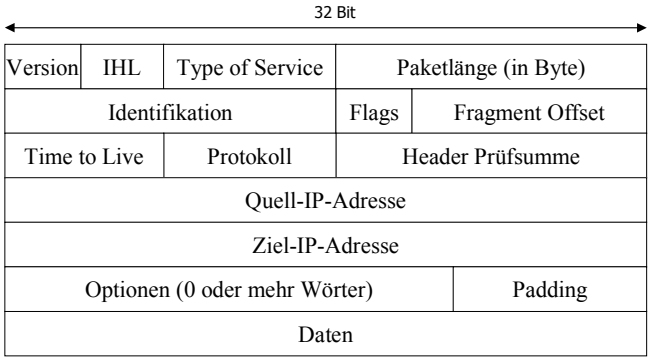


Abbildung 4-29: IP-Header

Folgende Felder sind im IP-Header enthalten:

- *Version:* Enthält die genutzte IP-Version (derzeit 4).
- *IHL*: Gibt die Länge des IP-Headers gemessen in 32-Bit-Worten an. Dieses Längenfeld ist aufgrund der variablen Länge des Optionsfeldes nötig. Der Header hat eine Länge von mindestens fünf 32-Bit-Worten ohne Berücksichtigung der Optionen und maximal 15 Worten bei Optionsangabe, also max. 60 Byte.
- *Type of Service*: Die Bedeutung dieses Feldes wurde mehrfach geändert:
  - Im RFC 791 wurde es auf drei Einzelfelder aufgeteilt. In den ersten drei Bit konnte man eine Priorität angeben (000=Standard, 001=Priorität, ..., 101=kritisch, ...), in den nächsten drei Bit einen Service-Typen (hoher Durchsatz, niedrige Verzögerung, ...) und die letzten beiden Bit wurden nicht belegt.
  - Nach einigen Änderungen wurden die ersten sechs Bit im Rahmen der Festlegungen differenzierter Services im Internet im RFC 2474 neu definiert. Zur Definition differenzierter Services können 64 verschiedene Klassen angegeben werden.
  - Die letzten beiden Bit werden heute gemäß RFC 2481 für einen expliziten Congestion-Notification-Mechanismus (ECN) verwendet. Über diesen Mechanismus können sich benachbarte Router über Warteschlangen-Engpässe informieren, um Überlastsituationen auf der IP-Ebene zu behandeln.

- *Paketlänge*: Gesamtlänge des Datenpakets in Byte inkl. des Headers. Die Maximallänge beträgt 65.535 Byte.
- *Identifikation*: Alle Fragmente eines Datagramms erhalten hier den gleichen Wert (mehr dazu weiter unten).
- *Flags:* (3 Bit) Dient der Kontrolle der Fragmentierung (mehr dazu bei der Fragmentierung weiter unten). Die Flags geben an, ob das Feld geteilt werden muss und weitere Pakete folgen oder ob das aktuelle Paket das letzte ist. Es sind drei Flags, wovon das erste unbenutzt ist. Die benutzten Flags heißen DF und MF und haben folgende Bedeutung:
  - Falls DF=1 ist, dann ist eine Fragmentierung des Pakets nicht erlaubt. Dieses Flag ist für Router von Bedeutung, um zu entscheiden, ob eine Fragmentierung erfolgen darf.
  - Falls MF=0 (More Fragments) ist, dann handelt es sich um das letzte Fragment im Datagramm. Falls MF=1 ist, folgen noch weitere Fragmente. Dieses Flag ist für das Zielsystem wichtig, um das Ende des Datagramms zu erkennen.
- *Fragment Offset (FO):* Dieses Feld dient der korrekten Herstellung der Ursprungssequenz, da Pakete das Ziel in unterschiedlicher Reihenfolge erreichen. Das Feld dient der Ermittlung der relativen Lage des Fragments im Datagramm (angegeben in Byte-Offset dividiert durch 8) und ist 13 Bit lang. Damit lassen sich genau 2^{13} * 8 Byte = 8.192 * 8 Byte darstellen, was mit der maximalen Paketlänge korrespondiert. Das kleinste Fragment hat demnach eine Länge von 8 Byte (ohne Header).
- *Time to live (TTL):* Das Feld gibt an, wie lange ein Datagramm im Internet verbleiben darf. Es dient dazu, zu alte Pakete vom Netz zu nehmen. Bei einem Wert von 0 wird das Paket verworfen und eine ICMP-Nachricht (ICMP wird weiter unten noch erläutert) zum Quellhost gesendet. Ursprünglich war die Angabe der Zeit in Sekunden (max. 255 s) gedacht, ein Paket sollte also max. 255 Sekunden leben und dann muss es ausgeliefert sein. Heute wird es aber als Hop-Count genutzt. Jeder Router, den ein Paket passiert, subtrahiert 1 von diesem Feld. Der Initialzustand hängt meist von der Konfigurierung im Host ab.
- *Protokoll:* Das Feld definiert das darüberliegende Protokoll, an welches die Daten des Pakets weitergereicht werden (6=TCP, 89=OSPF,...) und ist wichtig für die Zuordnung ankommender Pakete an die entsprechenden Transportinstanzen.
- Header-Prüfsumme: Dies ist eine Prüfsumme, die der Fehlererkennung im IP-Header dient. Das Feld sichert also nicht Daten in höheren Protokollen. Es muss für jede Teilstrecke neu berechnet werden, da sich der TTL-Wert immer verändert.
- *Quell-IP-Adresse* und *Ziel-IP-Adresse:* Hier werden jeweils die 32 Bit langen IP-Adressen für die Quelle und das Ziel eingetragen.

- *Optionen*: In diesem Feld stehen zusätzliche, optionale Angaben. Beispiel sind „Loose Source Routing“ als Möglichkeit, den Weg eines Paketes durch das Internet aufzuzeichnen, und „Strict Source Routing“, wobei für die Pakete die Pfadvorgabe eingehalten werden muss. Das Feld wird selten verwendet.
- *Padding*: Wenn eine Option genutzt wird, muss das Datagramm bis zur nächsten 32-Bit-Grenze mit Nullen aufgefüllt werden. Dies wird bei Bedarf in diesem Feld erledigt.
- *Daten:* Hier sind die Nutzdaten der höheren Schicht enthalten.

Die Bedeutung der Felder für die Fragmentierung wird im Zusammenhang verständlich.

4.2.8 IP-Fragmentierung und -Reassemblierung

Das Internet unterstützt eine Vielzahl von Netzwerkzugängen (Schicht 2). Im LAN ist z.B. der Ethernet-Standard ein typischer Netzwerkzugang, im WAN-Bereich gibt es u.a. ISDN, ADSL, ATM usw. All diese Netzwerkzugänge übertragen ihre Daten aus IP-Sicht in Schicht-2-PDUs, in denen die übertragbare Nutzdatenlänge jeweils unterschiedlich ist. Ethernet-Pakete haben z.B. eine Maximallänge von 1.500 Byte. Viele WAN können nicht mehr als 576 Byte in einer PDU übertragen. Diese Größe wird als maximale Transfereinheit (MTU) bezeichnet.

Auf einer Route zwischen einem Zielrechner und einem Quellrechner kann es nun sein, dass verschiedene Schicht-2-Verbindungen zu durchlaufen sind, die unterschiedliche MTU-Längen aufweisen. Wenn ein IP-Datagramm größer ist, als die MTU des Netzwerkzugangs, über den es gesendet werden soll, dann muss der Router dieses Paket zerlegen, und der Zielknoten muss alle Teile (Fragmente) wieder zusammenbauen. Erst wenn alle Fragmente eines IP-Datagramms wieder zusammengebaut sind, kann es am Zielknoten an die Transportschicht weitergereicht werden.

Diese Aufgabe nennt man bekanntlich Fragmentierung (Assemblierung) und Defragmentierung (Reassemblierung). Im IP-Header sind die Informationen enthalten, die notwendig sind, um diese Aufgabe zu erfüllen. Beim Zusammensetzen des IP-Datagramms im Zielknoten muss u.a. erkennbar sein, welches Fragment nun zu welchem IP-Datagramm gehört und in welcher Reihenfolge die Fragmente zusammengebaut werden müssen. Außerdem müssen Regeln für Fehlersituationen definiert sein, z.B. was zu tun ist, wenn ein Fragment nicht im Zielrechner ankommt.

Sobald ein IP-Router eine Fragmentierung initiiert hat, laufen in einem Knoten einige Aktivitäten und Überprüfungen ab:

- Das DF-Flag wird überprüft, um festzustellen, ob eine Fragmentierung erlaubt ist. Ist das Bit auf „1“ gesetzt, wird das Paket verworfen.

- Ist das DF-Flag nicht gesetzt, wird entsprechend der zulässigen Paketgröße das Datenfeld des Ur-Paketes in mehrere Teile zerlegt (fragmentiert).
- Alle neu entstandenen Pakete – mit Ausnahme des letzten Paketes – weisen als Länge immer ein *Vielfaches von 8 Byte* auf.
- Alle Datenteile werden in neu erzeugte IP-Pakete eingebettet. Die Header dieser Pakete sind Kopien des Ursprungskopfes mit einigen Modifikationen.
- Das MF-Flag wird in allen Fragmenten mit Ausnahme des letzten auf „1“ gesetzt.
- Das Fragment-Offset-Feld erhält Angaben darüber, wo das Datenfeld in Relation zum Beginn des nicht fragmentierten Ur-Paketes platziert ist.
- Enthält das Ur-Paket Optionen, wird abhängig vom *Type*-Feld entschieden, ob die Option in jedes Paketfragment aufgenommen wird (z.B. Protokollierung der Route).
- Die Headerlänge (IHL) und die Paketlänge sind für jedes Fragment neu zu bestimmen
- Die Headerprüfsumme wird für jedes Fragment neu berechnet.

Die Zielstation setzt die Fragmente eines Datagramms wieder zusammen. Die Zusammengehörigkeit entnimmt sie dem Identifikationsfeld, das von einer Fragmentierung unberührt bleibt. Bei der Defragmentierung wird wie folgt vorgegangen:

- Die ankommenden Fragmente werden zunächst gepuffert. Bei Eintreffen des ersten Fragments wird ein Timer gestartet.
- Ist der Timer abgelaufen bevor alle Fragmente eingetroffen sind, wird alles, was bis dahin gesammelt wurde, verworfen.
- Im anderen Fall wird das komplette IP-Datagramm am N-SAP zur Transportschicht hochgereicht.

Erst wenn alle Fragmente am Zielhost angekommen sind (und dies kann durchaus in anderer Reihenfolge sein), kann die Reassemblierung abgeschlossen werden.

Es ist natürlich wünschenswert, dass die Fragmentierung auf ein Minimum reduziert wird, da es sowohl in den Routern als auch in den Endknoten zusätzlichen Overhead bedeutet. Im Internet ist definiert, dass jedes Sicherungsprotokoll eine MTU-Länge von *576 Byte* unterstützen soll. Fragmentierung kann also vermieden werden, wenn die Transportprotokolle (TCP und UDP) kleinere Segmente verwenden.

Beispiel nach (Kurose 2002): Betrachten wir den Fall, dass ein Datagramm mit 4.000 Byte (3.980 Byte Nutzdaten und 20 Byte IP-Header) Länge und natürlich auch inkl. der Header höherer Protokolle bei einem Router ankommt und über eine Netzwerkverbindung mit einer MTU von 1500 Byte weitergeleitet werden muss. Das Original-Datagramm, das eine Identifikation von 777 hat, wird in drei Fragmente zerlegt, in denen die IP-Header-Felder gemäß Tabelle 4-3 gefüllt werden müssen (siehe hierzu den IP-Header). Im Feld Fragment Offset (FO) steht tatsächlich die Byte-Position dividiert durch 8. Also steht z.B. im FO-Feld des zweiten

Fragments der Wert 185. Bis auf das letzte Fragment eines Datagramms haben alle Fragmente eine durch 8 teilbare Anzahl an Byte. Bei einer MTU von 1500 Byte können abzüglich des minimalen IP-Headers 1480 Byte IP-Nutzdaten übertragen werden.

Tabelle 4-3: Beispiel für eine IP-Fragmentierung

Fragment	Byteanzahl im Datenfeld	Identifikation	Fragment-Offset in Byte (im FO-Feld)	Flags
1	1480	777	0 (0)	MF=1
2	1480	777	1480 (185)	MF=1
3	1020	777	2960 (370)	MF=0

Ein Fragment wird bei der Übertragung wie ein normales Datagramm behandelt und kann somit selbst wieder in Fragmente zerlegt werden. Werden beispielsweise die Fragmente der Tabelle 4-3 vom nächsten Router über eine Verbindung mit einer MTU von 1024 Byte übertragen, ist eine weitere Fragmentierung notwendig. Da die Fragmente als eigenständige Datagramme behandelt werden, kann es zu einer unerwartet hohen Anzahl von Fragmenten kommen. Wie in der Tabelle 4-4 zu sehen ist, werden sechs Fragmente erzeugt, da jedes Fragment nochmals in zwei Fragmente aufgeteilt wird. Wäre schon bei der ersten Fragmentierung die MTU bei 1024 Byte, würden nur 4 (3980 / 1000 <= 4) Fragmente erzeugt werden. Wie schon beschrieben, kann die Zielstation auch bei mehrmaliger Fragmentierung das ursprüngliche Datagramm über den Fragment-Offset wiederherstellen.

Tabelle 4-4: Beispiel für eine IP-Fragmentierung

Fragment	Byteanzahl im Datenfeld	Identifikation	Fragment-Offset in Byte (im FO-Feld)	Flags
1 (1.1)	1000	777	0 (0)	MF=1
2 (1.2)	480	777	1000 (125)	MF=1
3 (2.1)	1000	777	1480 (185)	MF=1
4 (2.2)	480	777	2480 (310)	MF=1
5 (3.1)	1000	777	2960 (370)	MF=1
6 (3.2)	20	777	3960 (495)	MF=0

4.2.9 Routing im Internet

Für die Wegewahl werden im Internet verschiedene Verfahren eingesetzt, und man muss prinzipiell zwischen der Wegewahl innerhalb autonomer Systeme (Intra-AS-Routing) und der Wegewahl im globalen Internet, also zwischen den autonomen Systemen (Inter-AS-Routing), unterscheiden. Jedes autonome System kann intern eigene Routing-Algorithmen verwenden, die in IP-Routern ablaufen. Ein IP-Router verbindet mindestens zwei Netzwerke und verfügt daher über zwei Netzwerkadapter.

Routing-Tabellen

Jeder IP-Router und auch jeder Host verwaltet eine Routing-Tabelle mit Routing-Informationen. Ein Eintrag in der Routing-Tabelle (siehe Tabelle 4-5) spezifiziert eine Route. Bestandteile sind das Netzwerkziel (Netzwerkadresse des Ziels der Route), die Netzwerkmaske oder evtl. die Länge der Subnetz-Maske bei Einsatz von VLSM bzw. CIDR, der nächste Router auf der Route (auch Gateway genannt), der Ausgangsport (auch als Interface bezeichnet) und eine Metrik. Einige Anmerkungen hierzu:

- Auch Hosts verfügen über eine Routing-Tabelle, in der z.B. das Default-Gateway mit Zieladresse 0.0.0.0, also die Standard-Route statisch eingetragen ist.
- In der Spalte *Netzwerkziel* kann auch eine Hostadresse stehen. Man spricht in diesem Fall von einer *Hostroute*, und die Spalte *Netzwerkmaske* enthält in diesem Fall den Wert „255.255.255.255“.
- In der Spalte *Metrik* werden die Kosten der Route angegeben. Bei RIP (siehe weiter unten) wird hier z.B. die Anzahl an Hops zum Ziel angegeben. Ein Hop kann als das Absenden eines Pakets aus einem Router/Host interpretiert werden. Die Anzahl der Hops gibt also die Anzahl der Absendevorgänge an, bis das Paket am Ziel ist.[14]
- In der Spalte *Ausgangsport* steht üblicherweise die dem Interface zugeordnete IP-Adresse.

[14] Hinweis: Diese Metrik wird bei Windows z.B. anhand des Netzwerkanschlusses automatisch ermittelt. Eine Metrik von 20 wird öfter für einen 100 MBit-LAN-Anschluss vergeben.

Tabelle 4-5: Typischer Aufbau einer Routing-Tabelle in IP-Netzen

Netzwerk-ziel	Netzwerk-maske	Nächster Router	Ausgangsport = Schnittstelle	Metrik
...	...	...	...	...

Grundsätzliches zur Routenbestimmung. Bei dynamischen bzw. zustandsbehafteten Routing-Protokollen werden zwischen den Routern Informationen zur Aktualisierung der Routing-Tabellen ausgetauscht. Diese Informationen bezeichnet man auch als Routing-Informationen. Welche Informationen konkret ausgetauscht werden, hängt vom eingesetzten Routing-Protokoll ab.

Die Bestimmung der Route läuft nach einem definierten Algorithmus ab, der grob skizziert werden soll:

- Die Zieladresse eines ankommenden und weiterzuleitenden IP-Pakets wird mit allen Einträgen der Routing-Tabelle verglichen. Hierzu wird eine bitweise Und-Operation zwischen der Zieladresse im Paket und der Netzwerkmaske aus allen Routeneinträgen, die in der Routing-Tabelle stehen, durchgeführt. Anschließend wird das Ergebnis jeweils mit der Spalte *Netzwerkziel* desselben Eintrags in der Routing-Tabelle verglichen. Wenn das Ergebnis der Und-Operation mit dem Netzwerkziel übereinstimmt, repräsentiert der entsprechende Routing-Eintrag eine potenzielle Route.
- Aus allen potenziellen Routen wird die Route mit der längsten Übereinstimmung bei der Und-Operation ausgewählt. Dies ist die Route mit den meisten übereinstimmenden Bit.
- Bei gleichwertigen Routen wird die Route mit dem besten (kleinsten) Wert aus der Spalte *Metrik* ausgewählt. Gibt es hier auch mehrere Kandidaten, wird zufällig entschieden.

Für alle ankommenden Pakete, zu denen kein konkretes Netzwerkziel in der Routing-Tabelle gefunden wird, kann die Default-Route (Standardroute) angewendet werden, um das Paket weiterzuleiten. Diese Route wird von vorneherein festgelegt und im Router bzw. im Host konfiguriert. Das Netzwerkziel der Standardroute hat den Wert „0.0.0.0“ und die Netzwerkmaske „0.0.0.0“ (siehe auch das folgende Beispiel).

Beispiel: Über das Kommando *netstat -r* kann in den meisten Betriebssystemen der aktuelle Inhalt der Routing-Tabelle ausgegeben werden. Die folgende Ausgabe wurde auf einem Windows-basierten Host mit der IP-Adresse 10.28.16.21 (privates IP-Netzwerk) erzeugt:

```
> netstat -r
```

```
Aktive Routen:
Netzwerkziel     Netzwerkmaske    Gateway        Schnittstelle    Anzahl
0.0.0.0          0.0.0.0          10.28.1.253    10.28.16.21      20
127.0.0.0        255.0.0.0        127.0.0.1      127.0.0.1        1
10.28.16.21      255.255.255.255  127.0.0.1      127.0.0.1        20
224.0.0.0        240.0.0.0        10.28.16.21    10.28.16.21      20
255.255.255.255  255.255.255.255  10.28.16.21    10.28.16.21      1
...
Standardgateway: 10.28.1.253
```

Die Abbildung 4-30 zeigt die Einbettung des ausgewählten Hosts in das Beispielnetzwerk. Es handelt sich um ein Klasse-A-Netzwerk mit einem Netzwerkanteil von 8 Bit.

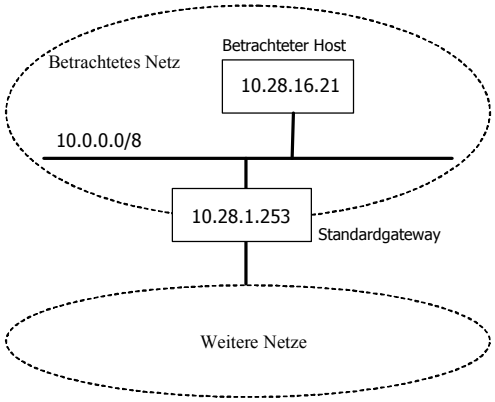


Abbildung 4-30: Netwerkausschnitt zum Routing-Beispiel

In dem Host gibt es genau eine Ethernetkarte (LAN-Adapter), also eine Schnittstelle (physikalischer Ausgangsport) nach außen. Die Spalte mit der Bezeichnung „Gateway" gibt den nächsten Router an. Als Standardgateway ist der Router 10.28.1.253 konfiguriert. Die Einträge werden im Folgenden einzeln betrachtet.

Standardroute. Der erste Eintrag gibt die *Standardroute* an. Sie hat immer das Netzwerkziel 0.0.0.0 und die Netzwerkmaske 0.0.0.0 (/0).

```
Netzwerkziel     Netzwerkmaske    Gateway        Schnittstelle    Anzahl
0.0.0.0          0.0.0.0          10.28.1.253    10.28.16.21      20
```

Bedeutung: Jedes Paket mit einer IPv4-Zieladresse, für die eine bitweise logische Und-Operation mit 0.0.0.0 ausgeführt wird, führt immer zu dem Ergebnis 0.0.0.0. Die Standardroute führt daher zu einer Übereinstimmung mit jeder IPv4-Zieladresse. Wenn die Standardroute die längste übereinstimmende Route ist, lautet die Adresse des nächsten Knotens 10.28.1.253 (Standard-Gateway) und die

Schnittstelle für den nächsten Knoten ist der Netzwerkadapter mit der IPv4-Adresse 10.28.16.21 (einziger LAN-Adapter).

Loopback-Route. Der nächste Eintrag gibt die sog. Loopback-Route an. Als Netzwerkziel wird für diese Route üblicherweise 127.0.0.0 angegeben und als Netzwerkmaske 255.0.0.0 (/8).

```
Netzwerkziel     Netzwerkmaske    Gateway        Schnittstelle    Anzahl
127.0.0.0        255.0.0.0        127.0.0.1      127.0.0.1        1
```

Bedeutung: Für alle Pakete, die an Adressen in der Form 127.x.y.z gesendet werden, wird die Adresse des nächsten Knotens auf 127.0.0.1 (die Loopback-Adresse) gesetzt. Die Schnittstelle für den nächsten Knoten ist die Schnittstelle mit der Adresse 127.0.0.1 (die Loopback-Schnittstelle). Die Pakete werden nicht in das Netzwerk gesendet, sondern verbleiben im Host.

Hostroute. Die Route mit der eigenen IP-Adresse als Netzwerkziel (10.28.16.21) und der Netzwerkmaske 255.255.255.255 (/32) wird immer als Hostroute bezeichnet.

```
Netzwerkziel     Netzwerkmaske    Gateway        Schnittstelle    Anzahl
10.28.16.21      255.255.255.255  127.0.0.1      127.0.0.1        20
```

Bedeutung: Für alle (von einer lokalen Anwendung) an die Adresse 10.28.16.21 (eigene IP-Adresse) gesendeten IPv4-Pakete wird die Adresse des nächsten Knotens auf 127.0.0.1 gesetzt. Damit erfolgt eine Kommunikation über den Kernel, ohne das Netzwerk zu belasten. Die Pakete verbleiben im Host. Die verwendete Schnittstelle für den nächsten Knoten ist also die Loopback-Schnittstelle.

Route für Multicast-Verkehr. Der Eintrag mit dem Netzwerkziel 224.0.0.0 und der Netzwerkmaske 240.0.0.0 (/4) ist eine Route für den Multicast-Verkehr, der von diesem Host gesendet wird.

```
Netzwerkziel     Netzwerkmaske    Gateway        Schnittstelle    Anzahl
224.0.0.0        240.0.0.0        10.28.16.21    10.28.16.21      20
```

Bedeutung: Für alle von diesem Host ausgehenden Multicast-Pakete wird die Adresse des nächsten Knotens auf die Adresse 10.28.16.21 gesetzt, und für die Schnittstelle des nächsten Knotens wird der LAN-Adapter mit der IP-Adresse 10.28.16.21 festgelegt.

Limited-Broadcast-Adresse. Der Eintrag mit dem Netzwerkziel 255.255.255.255 und der Netzwerkmaske 255.255.255.255 (/32) ist eine Hostroute, die der limited Broadcast-Adresse entspricht

```
Netzwerkziel     Netzwerkmaske    Gateway        Schnittstelle    Anzahl
255.255.255.255  255.255.255.255  10.28.16.21    10.28.16.21      1
```

Bedeutung: Für alle an die IP-Adresse 255.255.255.255 gesendeten IPv4-Pakete wird die Adresse des nächsten Knotens auf die IP-Adresse 10.28.16.21 gesetzt, und

die Schnittstelle des nächsten Knotens ist der LAN-Adapter mit der Adresse 10.28.16.21.

IGP und EGP

Die Routing-Protokolle für autonome Systeme werden als Interior-Gateway-Protokolle (*IGP*) bezeichnet, die zwischen autonomen Systemen als Exterior-Gateway-Protokolle (*EGP*).

Das Routing Information Protocol (RIP)[15] ist ein älteres IGP-Protokoll für kleinere Netze. Es ist ein Distance-Vector-Protokoll mit den bereits diskutierten Nachteilen wie dem Count-to-Infinity-Problem. RIP ist ein zustandsunabhängiges Routing-Protokoll und berücksichtigt damit nicht die aktuelle Netzwerksituation.

Nachfolger von RIP ist seit ca. 1990 das *Open Shortest Path First Protocol (OSPF*, RFC 1247). OSPF wird von der Internet-Gemeinde empfohlen. Das Netz wird hier als gerichteter Graph abstrahiert. Die Kanten zwischen den Knoten werden mit Kosten gewichtet (Entfernung, Verzögerung,...) und die Entscheidung über das Routing erfolgt anhand der Kosten. OSPF ist ein zustandsabhängiges Routing-Protokoll.

Bei EGP werden andere Ziele verfolgt als bei IGP. Beispiele für Routing-Kriterien sind hier u.a.:

- Nicht alle Pakete zwischen den AS dürfen befördert werden.
- Für den Transitverkehr werden Gebühren verrechnet.
- Wichtige Informationen nicht durch unsichere autonome Systeme senden.

Hierzu sind Routing-Regeln erforderlich, die vom Routing-Protokoll unterstützt werden müssen. Im Internet wird als EGP das *Border-Gateway-Protokoll (BGP)*, ein Distance-Vector-Protokoll, empfohlen (RFC 1771) und stellt quasi den De-facto-Standard dar. Im BGP werden nicht nur Kosten pro Ziel verwaltet, sondern es wird auch eine Buchführung über die Nutzung der Verbindungen durchgeführt.

[15] RIP oder RIP-1 (Version 1) ist im RFC 1058 genormt, die weiterentwickelte Version 2 von RIP (RIP-2) im RFC 2453. Ein weiteres, bekanntes IGP-Protokoll ist IGRP (Interior Gateway Routing Protocol) von der Firma Cisco, also eine firmeneigene Lösung. Auch gibt es eine Erweiterung dieses Protokolls, die EIFRP (Enhanced IGRP) heißt. Cisco entwickelte das Protokoll, um die Einschränkungen von RIP zu umgehen. Es läuft allerdings nur unter Cisco-Routern. Das OSI-Routing-Protokoll IS-IS ist ebenfalls ein IGP.

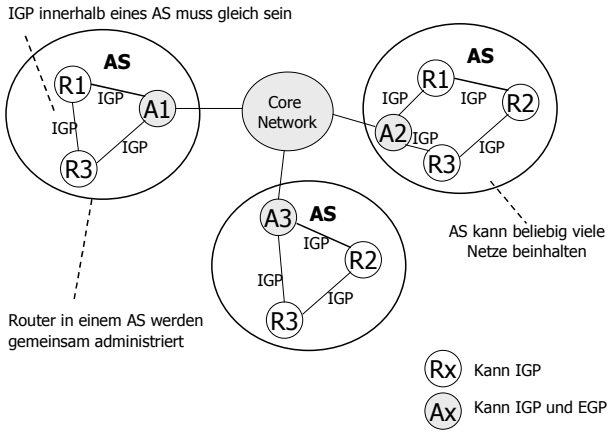


Abbildung 4-31: Routing im Internet

In Abbildung 4-31 sind die Zusammenhänge zwischen IGP und EGP im Internet dargestellt. Die drei autonomen Systeme in der Abbildung arbeiten alle über eigene IGP. Im globalen Internet wird EGP verwendet. In jedem AS ist ein Router, der sowohl IGP nach innen, als auch EGP nach außen betreibt.

Routing Information Protocol, Version 1 (RIP-1)

Betrachten wir zunächst RIP (RFC 1058) etwas genauer. RIP wurde ursprünglich von XEROX entwickelt und ist ein leicht zu implementierendes Distance-Vector-Protokoll. Es nutzt UDP zum Austausch der Routing-Information unter den Routern, wobei eine Request-PDU zur Anfrage (ein sog. Advertisement) der Routing-Information bei einem Nachbar-Router und eine entsprechende Response-PDU definiert ist. Zur Kommunikation wird der UDP-Port 520 verwendet. Beim Start eines Routers sendet dieser zunächst an all seine Nachbar-Router eine Request-PDU und fordert damit die Routing-Informationen an. Die Antwort erhält er in zielgerichteten Response-PDUs (also nicht über Broadcast-, sondern über Unicast-Nachrichten). Im normalen Betrieb sendet ein Router unaufgefordert seine Routing-Informationen und nutzt hierzu ebenfalls die Response-PDUs. In einer Response-PDU können max. 25 Tabelleneinträge übertragen werden.

Als Metrik wird die Anzahl der Sprünge (Hops) von Router zu Router verwendet. Die Hops werden in den Paketen in einem Feld im IP-Header (TTL) gezählt. Bei RIP ist der maximale Hop-Count 15. Eine höhere Angabe wird als „unendlich“ interpretiert. Die Implementierung unter Unix liegt beispielsweise in einem

Prozess namens *routed*[16] (Routing Dämon). Heute ist RIP zwar nicht mehr der empfohlene Standard im Internet, es wird aber in kleinen Netzen immer noch stark verwendet.

Die Router tauschen alle 30 Sekunden über einen Broadcast Advertisements in Form von unaufgeforderten Response-PDUs aus, in denen die komplette Routing-Tabelle an alle Nachbar-Router übertragen wird. Da der Broadcast auf MAC-Ebene erfolgt und dies eine recht hohe Netzwerkbelastung mit sich bringt, ist RIP-1 für den Einsatz im WAN nicht optimal.

Wenn ein Router 180 Sekunden nichts von einem seiner Nachbarn hört, gilt dieser als nicht erreichbar. Die Routing-Tabelle wird daraufhin aktualisiert, d.h. die Metrik der Route zu diesem Router wird mit dem Wert 16 belegt, und die Routenänderung wird an die anderen Nachbarn propagiert. Das tatsächliche Entfernen der Route aus der Routing-Tabelle erfolgt im Anschluss. Zusätzlich wird der Timeout für die Überwachung des Nachbarn auf 120 Sekunden verkürzt. Dieser Mechanismus wird auch als Garbage-Collection bezeichnet.

Das Problem der *langsamen Konvergenz* und der Schleifen („*Count-to-Infinity-Problem*“) ist in RIP-1 gegeben. Die Konvergenzzeit ist die Zeit, die benötigt wird, bis alle Router die aktuelle Vernetzungsstruktur kennengelernt haben. Wenn man davon ausgeht, dass alle Router ihre Routing-Informationen alle 30 Sekunden verteilen, kann die Verbreitung einer neuen Information in einem Netz mit mehreren Subnetzen durchaus mehrere Minuten dauern.

Beispiel: Sendet der Router R1, wie in Abbildung 4-32 dargestellt, eine neue Routen-Information, so dauert es im Beispiel 90 Sekunden bis die Information beim Router R5 angekommen ist. Wie man erkennen kann, ist die Konvergenzzeit zufallsabhängig, da das Eintreffen des Ereignisses „Neue Route bekannt“ noch nicht sofort zum Senden einer Advertisement-PDU führt. Bis zum nächsten Broadcast dauert es im Mittel 15 Sekunden. Im Beispiel sendet R1 die neue Information nach 15 Sekunden, R2 sendet sie 15 Sekunden nach dem Empfang weiter, R3 wartet 20 Sekunden usw.

Um die Konvergenzzeit bei RIP zu verringern gibt es drei Möglichkeiten: Die *Split-Horizon-Technik (*geteilter Horizont), die *Split-Horizon-Technik mit Poison-Reverse* (vergifteter Rückweg) und sog. *Triggered Updates* (ereignisgesteuerte Routen-Aktualisierungen).[17]

---

[16] Etwas verwirrend mag es sein, dass eine Aufgabe der Vemittlungsschicht über einen Protokoll der Schicht 4 (UDP) abgewickelt wird. Dies liegt an der Unix-Historie, da *routed* eigentlich als Anwendung in der Anwendungsschicht platziert ist und daher einen Transportdienst verwenden kann.

[17] Welche Methoden in den Routern nun implementiert sind, ist dem Hersteller überlassen.

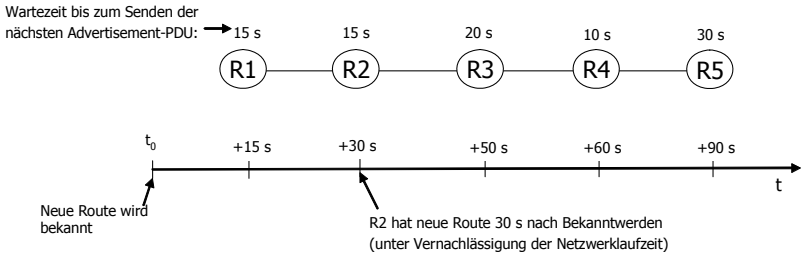


Abbildung 4-32: Konvergenzzeit bei RIP-1

Bei der *Split-Horizon-Technik* wird die Konvergenzzeit dadurch verringert, dass in den Routing-Tabellen zusätzlich die Information verwaltet wird, woher (von welchem Router) die Routing-Information stammt. Ein Router darf keine Route an ein Subnetz propagieren, die er über dasselbe Subnetz gelernt hat. Bei der *Split-Horizon-Technik mit Poison-Reverse* werden zwar alle Routen propagiert, jedoch werden Routen, die aus einem Subnetz erlernt wurden, an dieses mit einer Metrik von 16 Hops gesendet. Damit sind sie als „nicht erreichbar" markiert. Bei Nutzung der *Triggered-Updates*-Methode werden Routen-Aktualisierungen unmittelbar nach dem Eintreffen dieser Ereignisse weitergeleitet, d.h. es wird nicht gewartet, bis der 30-Sekunden-Timer abläuft. Dies erhöht zwar die Netzwerkbelastung, aber die Konvergenzzeit wird deutlich verringert.

Beispiel: Betrachten wir den Netzausschnitt aus Abbildung 4-33. Was passiert ohne und was mit *Split-Horizon*, wenn die Verbindung zwischen Router R1 und Router R2 ausfällt?

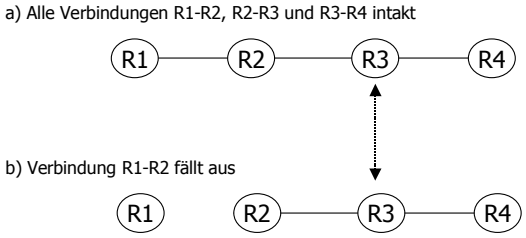


Abbildung 4-33: Verbindungsabbruch bei RIP

Ohne *Split Horizon*:

- R3 hat noch die Routing-Information, dass R1 über einen Hop erreichbar ist.
- R3 propagiert diese Info an R2, also an den Router, über den R1 erreicht wurde.
- R2 glaubt dies und sendet Pakete zu R1 nun über R3.

– Es entsteht ein Ping-Pong-Effekt, also eine Routing-Schleife bis der Hop-Count = 16 ist, dann erst wird R1 als nicht erreichbar markiert.

Mit *Split Horizon*:

– R3 weiß, woher die Routing-Information für R1 kommt (von R2).
– Die Route mit höheren Kosten wird nicht zurückpropagiert.

Bei Anwendung von *Split-Horizon* können Routing-Schleife und somit das *Count-To-Infinity-Problem* vermieden werden. Gleiches gilt bei der Anwendung der *Split-Horizon-Technik mit Poison-Reverse.* Weist die Netzwerktopologie jedoch Schleifen (Kreise) auf, kann auch bei Anwendung der *Split-Horizon-Technik* das *Count-To-Infinity-Problem* auftreten.

RIP-1-PDU-Aufbau. RIP-1-PDUs werden über UDP ausgetauscht. Die Kommunikation erfolgt immer zwischen benachbarten Routern. Eine RIP-1-PDU hat den in Abbildung 4-34 skizzierten Aufbau. Die PDU besteht aus zwei Teilen, einem RIP-1-Header und einer Tabelle mit maximal 25 Routing-Einträgen. Wenn ein Router mehr als 25 Routen propagieren möchte, muss er mehrere Nachrichten senden.

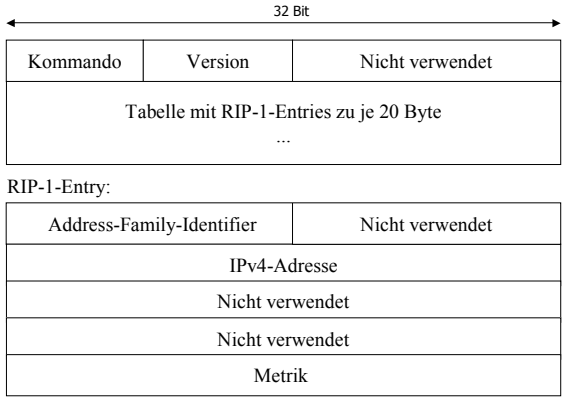


Abbildung 4-34: RIP-1-PDU

Der RIP-1-Header enthält zwei genutzte Felder:

– *Kommando*: In diesem Feld wird angegeben, ob es sich um eine RIP-Request- (0x01) oder um eine RIP-Response-PDU (0x02) handelt.
– *Version*: Angabe der verwendeten RIP-Version (0x01 = RIP-1).

Ein RIP-1-Entry enthält drei Felder zur Angabe jeweils einer Route:

– *Address-Familiy-Identifier (AFI)*: Dieses Feld gibt die Adressierungsart an und enthält für IP-Adressen immer den Wert 0x02. RIP wurde ursprünglich unabhängig von der Netzwerkschicht konzipiert.

- *IPv4-Adresse*: In diesem Feld steht die IP-Adresse der Route, also des Netzwerkziels.
- *Metrik*: In diesem Feld wird die Anzahl der Hops zum Netzwerkziel übertragen. Steht der Wert 16 in diesem Feld, bedeutet dies, dass das angegebene Netzwerkziel nicht erreichbar ist.

Routing Information Protocol, Version 2 (RIP-2)

Wie bereits erwähnt hat RIP-1 einige Schwächen. Hierzu gehört z.B., dass die RIP-1-Advertisements über Broadcast auf der MAC-Ebene übertragen werden, was zu hoher Netzwerkbelastung führt. Weiterhin wird VLSM von RIP-1 nicht unterstützt. Schließlich wird in den RIP-1-Entries die Subnetzmaske der Netzwerkziele nicht mit übertragen, weshalb ein Router eigenständig einfache Annahmen über die Subnetzmaske treffen muss. Beispielsweise werden die ersten drei Bit der IP-Adresse des Netzwerkziels analysiert, um die Netzwerkklasse zu ermitteln und damit die Subnetzmaske herzuleiten.

Das Protokoll RIP-2 (RFC 2453) ist eine Weiterentwicklung von RIP-1 und ist ebenfalls ein Distance-Vector-Routing-Protokoll, auch Bellman-Ford- oder Ford-Fulkerson-Algorithmus genannt. RIP-2 hat im Vergleich zu RIP-1 einige Erweiterungen bzw. Verbesserungen, um die genannten Schwächen zu kompensieren. Trotzdem ist RIP-2 kompatibel zu RIP-1, da RIP-1-Router RIP-2-spezifische Felder im RIP-Header überlesen.

Folgende Funktionen sind neu bzw. verbessert worden:

- RIP-2 unterstützt im Gegensatz zu RIP-1 „classless IP-Routing“ mit variabel langen Subnetzmasken (VLSM und CIDR).
- RIP-2 unterstützt eine Router-Authentifizierung als Sicherheitsinstrument für die Aktualisierung von Routing-Tabellen. Die Authentifizierung vermeidet, dass Router anderen nicht berechtigten Routern ihre Routing-Informationen weiterleiten. Die Authentifizierung erfolgt über ein Kennwort.
- RIP-1 verwendet Broadcasting zum Verbreiten von Routing-Informationen. RIP-1-PDUs werden aber nicht nur von Routern empfangen, sondern auch von den angeschlossenen Endgeräten[18]. RIP-2 verwendet dagegen Multicasting, wobei die RIP-2-PDUs über eine Klasse-D-Adresse (224.0.0.9) versendet werden. Endgeräte werden also durch RIP-2-PDUs nicht beeinträchtigt.

Das Erlernen der Routen von den Nachbar-Routern erfolgt bei RIP-2 nach dem gleichen Prinzip wie bei RIP-1. Auch der maximale Hop-Count ist weiterhin auf 15 eingestellt. Die Methoden *Split-Horizon*, *Split-Horizon mit Poison-Reverse* und *Triggered-Updates* werden unterstützt.

[18] MAC-Broadcasts werden von jedem Rechner im Netz bis zur Schicht 2 behandelt.

RIP-2-PDU-Aufbau. RIP-2-PDUs haben einen ähnlichen Aufbau wie RIP-1-PDUs (siehe Abbildung 4-35). Der Header ist identisch, im Feld *Version* steht allerdings der Wert 0x02.

In den Routen-Einträgen gibt es einige Unterschiede. Die Felder *Route-Tag*, *Subnet-Mask* und *Next-Hop* wurden ergänzt:

- *Route-Tag*: Dieses Feld wird nur benutzt, wenn ein RIP-2-Router auch Routen aus anderen Routing-Protokollen, z.B. von einem BGP-Router (siehe unten) übernimmt. In diesem Fall kann im Route-Tag ein Kennzeichen übertragen werden, das die Herkunft angibt. Dies könnte z.B. die Nummer des AS sein. Das Feld kann innerhalb eines AS frei verwendet werden, die Router müssen sich nur über den Inhalt einig sein.
- *Subnet-Mask*: Dieses Feld enthält die Subnetzmaske des Netzwerkziels und unterstützt damit VLSM und CIDR.
- *Next-Hop:* Über dieses Feld kann ein Router eine direkte Route zu einem Host (Host-Route) bekannt geben. Das Feld beinhaltet die Adresse des Hosts. Mit dieser Information in der Routing-Tabelle sendet ein Router ankommende IP-Pakete, die an den Host adressiert sind, nicht über einen Router, sondern direkt an den angegebenen Host weiter.

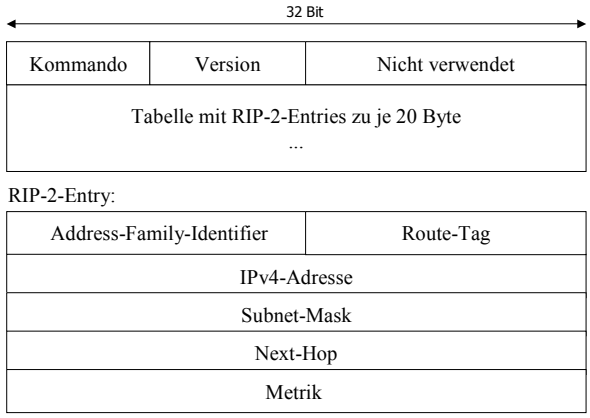


Abbildung 4-35: RIP-2-PDU

Die Authentifizierung wird über den ersten Routen-Eintrag in der Tabelle vorgenommen. Die Felder werden hierfür speziell belegt. Im *AFI*-Feld wird der Wert 0xFFFF, im Feld *Route-Tag* ein Authentifizierungstyp, und in den restlichen 16 Byte werden die Angaben für die Authentifizierung eingetragen. Möglich sind hier entweder ein Kennwort oder eine MD5-Prüfsumme (Message Digest 5 ist ein

Hashcode-Verfahren[19]). Wird ein einfaches, unverschlüsseltes Kennwort verwendet, steht im Feld *Route-Tag* der Wert 0x0001.

OSPF und OSPFv2

Während für kleinere Netze heute oft noch RIP bzw. die statische Konfigurierung der Routing-Tabellen verwendet wird, ist OSPF für große Unternehmensnetze gedacht. OSPF ist ein offener Standard (Open SPF) für ein Link-State-Protokoll. Im Gegensatz zu RIP handelt es sich hier um ein zustandsorientiertes Routing-Protokoll. Der „Link State" ist der Zustand einer Verbindung zwischen zwei IP-Routern. Aktuell liegt OSPF in der Version 2 (OSPFv2) vor (RFC 2328). Jeder Router führt bei OSPF eine Datenbasis (Link-State-Datenbank oder Verbindungszustandsdatenbank genannt) mit allen Routing-Informationen des Netzes. Die Routing-Information zu allen Routen wird also in jedem Router geführt. Die Kommunikation zum Austausch der Routing-Information erfolgt bei OSPF zwischen allen unmittelbaren Nachbarn (bei kleineren Netzen) oder zwischen herkömmlichen und sog. designierten Nachbarn (große Netze).

Jeder Router erzeugt aus seiner Sicht einen SPF-Baum (Shortest-Path-First), der auch als Spanning-Tree (überspannender Baum) bezeichnet wird, für das ganze Netzwerk. In diesem Baum stellt der Router die Wurzel dar. Die Berechnung erfolgt beispielsweise auf Basis des Dijkstra-Algorithmus. Die Verzweigungen im Baum stellen die günstigsten Routen zu allen bekannten Subnetzen bzw. anderen Routern dar. Auf Basis des SPF-Baums wird die Routing-Tabelle erzeugt. In Abbildung 4-36 ist ein Beispiel eines Spanning-Trees für einen Router A dargestellt.

Jede Kante des Baums stellt gewissermaßen einen Subnetzübergang und damit den „Link" zwischen zwei Routern dar. Diese Links müssen mit Kosten versehen werden, wobei als Faktoren z.B. die Belastung, die Übertragungsrate oder die Verzögerung möglich sind.

---

[19] Hash-Funktion, die von Prof. R. L. Rivest am MIT entwickelt wurde. MD5 erzeugt aus einer Nachricht variabler Länge eine Ausgabe fester Länge (128 Bit). Die Eingabenachricht wird in 512-Bit-Blöcke aufgeteilt. MD5 wird unter anderem zur Integritätsprüfung von Nachrichten benutzt. Der Sender erzeugt einen Hash-Code aus einer Nachricht, der Empfänger erzeugt ihn ebenfalls für die empfangene Nachricht und vergleicht ihn mit dem gesendeten Wert.

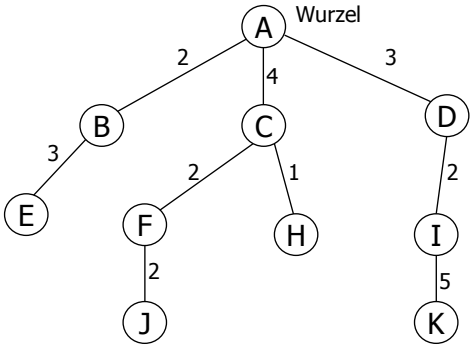


Abbildung 4-36: Spanning-Tree eines OSPF-Routers

Damit alle Router die vollständige Topologie des Netzes kennen, müssen sie sich synchronisieren. Bei OSPF sieht dies so aus, dass jeder Router mit seinen Nachbarn kommuniziert und jede Veränderung, die er von einem Nachbarn erfährt, an alle anderen Nachbarn weiterleitet. Die Kommunikation erfolgt dabei in Broadcast-Netzwerken wie herkömmlichen Ethernet-LANs über IP-Multicast-Adressen oder über Punkt-zu-Punkt-Verbindungen zwischen zwei Routern (z.B. mit designierten Routern, siehe unten). Bei Broadcastnetzen hört jeder Router die zugeordnete Multicast-Adresse ab.

Zum Aufbau von Nachbarschaften (Adjacency) sendet – etwas vereinfacht dargestellt – ein Router beim Start seinen Nachbarn sog. *Hello-PDUs* zu. Nicht alle angrenzenden Router werden auch zu Nachbarn (sog. adjacents). Anhand der Antworten, die auch in Form von Hello-PDUs eintreffen, entscheidet der neu gestartete Router, welche Router seine Nachbarn sind. Die Nachbarn senden dem neuen Router ihre Routing-Informationen in Form *von Database-Description-PDUs*.

Auf ähnliche Weise läuft es ab, wenn ein neuer Router zu einem bestehenden IP-Netzwerk hinzukommt. Jeder Router sendet nach der Hello-Phase seine eigene Routing-Information als sog. LSA (Link-State-Advertisements). Die von den Nachbarroutern empfangenen LSAs werden jeweils an die anderen Nachbarrouter weitergeleitet. Damit wird das ganze Netzwerk mit allen Routing-Informationen überflutet. Zum Abgleich der Link-State-Datenbank ist ein zyklischer Austausch (alle 30 min) der Topologieänderungen mit den Nachbarn vorgesehen.

Eine Lebendüberwachung wird ebenfalls periodisch unter den Nachbarn durchgeführt. Jeder Router teilt seinen Nachbarn in regelmäßigen Hello-PDUs mit, dass er noch aktiv ist. Kommt 40 Sekunden lang keine Hello-PDU eines Nachbarn, so gilt dieser als ausgefallen. Der Router, der den Ausfall bemerkt, sendet eine *Link-State-Update-PDU* an alle anderen Nachbarn, die auch mit einer *Link-State-Ack-PDU* bestätigt wird.

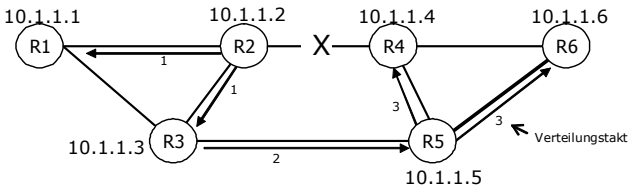


Abbildung 4-37: Routenaustausch beim Link-State-Verfahren

Die Verteilung einer Veränderung im Netz geht – nachdem sie erkannt worden ist – relativ schnell, das Protokoll arbeitet also mit recht hoher Konvergenz (siehe Abbildung 4-37). Im Beispiel ist die Verbindung zwischen 10.1.1.2 und 10.1.1.4 ausgefallen. Die Link-State-Updates werden über das ganze Netz verteilt. Nachdem die Link-State-Datenbanken aller Router synchronisiert sind, gibt es nur noch eine kürzeste Route zu jedem Router.

Unterstützung großer Netze. Für große Netzwerke kann man eine Aufteilung in mehrere autarke OSPF-Bereiche vornehmen, damit die Berechnung der optimalen Routen in den beteiligten Routern nicht zu aufwändig wird. Die Netzbereiche werden auch als *Areas* bezeichnet und erhalten eine *Area-Id* zur Identifikation. Die Schreibweise für die Area-Id ist analog zur IP-Adresse in *dotted decimal* notiert (Beispiel: 0.0.3.1).

Areas dienen der Hierarchisierung eines autonomen Systems. Alle Areas sind über ein OSPF-Backbone miteinander verbunden (siehe hierzu Abbildung 4-38), das die Area-Id 0.0.0.0 besitzt. Backbones werden auch als Area 0 bezeichnet.

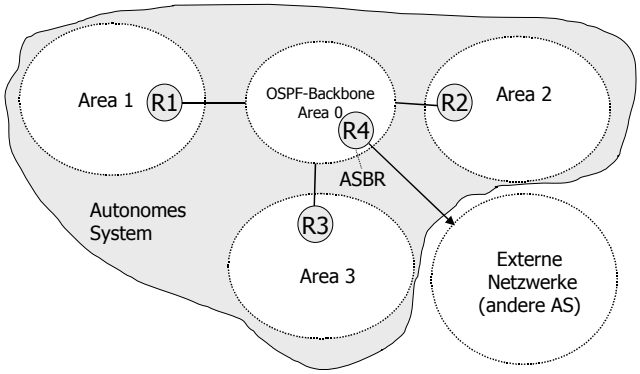


Abbildung 4-38: OSPF-Backbone

In einer Area verwenden alle Router den gleichen Shortest-Path-First-Algorithmus. Jeder Router in einer Area hat die gleiche Link-State-Datenbank. Router kennen

nur die Router aus ihrer Area. Ein Router der Area muss jedoch am OSPF-Backbone hängen und zwar über einen sog. Grenz- oder Area-Grenz-Router.

Bei OSPF gibt es vier Router-Klassen:

- *Interne Router* der Area, die nur Intra-AS-Routing durchführen und nach außen nicht in Erscheinung treten.
- *Area-Grenz-Router*: Dies sind Router an Bereichsgrenzen (Area-Grenzen), die zwei oder mehrere Areas innerhalb eines AS verbinden.
- *Backbone-Router*: Diese befinden sich im Backbone und führen das Routing innerhalb des Backbones durch, sie sind aber selbst keine AS-Grenz-Router.
- *AS-Grenz-Router*: (Area Boundary Router = ASBR). Sie vermitteln zwischen mehreren autonomen Systemen und tauschen mit diesen Routing-Informationen aus.

Ein Area-Grenz-Router kennt die LSA-Datenbanken aller Areas, an denen er angebunden ist. Alle *Area-Grenz-Router* sind im Backbone enthalten. Das Backbone dient primär dazu, den Verkehr zwischen den Bereichen im AS weiterzuleiten. Von einem autonomen System aus werden externe Routen zu anderen AS über ASBR ausgetauscht und dadurch nur in diesen verwaltet. In Abbildung 4-38 ist z.B. der Router R4 ein *ASBR*, während R1, R2 und R3 Area-Grenz-Router sind.

Jeder Router erhält zur Identifikation eine *Router-Id*. Die Übertragung der Routing-Informationen erfolgt direkt oder über spezielle Multicast-Adressen. OSPF-PDUs werden auch nur zwischen Nachbarn ausgetauscht. Ein Weiterrouten außerhalb des eigenen Netzes wird unterstützt.

Die Aufteilung eines großen AS hat den Vorteil, dass die Größe der Link-State-Datenbanken verringert wird und damit auch die Routing-Tabellen reduziert werden.

Designierte Router. Wenn es in einem Netzwerk viele OSPF-Router gibt, ist die Netzwerkbelastung sehr groß. Bei einem Netzwerk mit n Routern müssen n(n-1)/2 Nachbarschaften aufgebaut werden. In Broadcast-Netzwerken (LAN) wird daher ein sog. designierter Router und ein Ersatz-Router (designierter Backup-Router) zur Synchronisation der Routing-Information bestimmt.

Der designierte Router ist für die Verteilung der Routing-Information (in Form von LSAs) zuständig und wird mit einer sog. Priorität gekennzeichnet, die höher ist als bei herkömmlichen OSPF-Routern. Über diese Information kann ein designierter Router im Netz identifiziert werden. Eine direkte Kommunikation der OSPF-Router ist damit nicht mehr erforderlich. Es wird nur noch mit dem designierten Router oder seinem Stellvertreter kommuniziert. Die Anzahl der Nachbarschaften reduziert sich bei Einsatz eines designierten Routers somit auf n-1. Damit wird die Netzwerkbelastung, die sich ansonsten aufgrund von Broadcasts ergeben würde, reduziert.

OSPF-PDU-Typen. Im OSPF-Protokoll sind fünf OSPF-PDU- bzw. Paketypen definiert:

- Hello-PDU zur Feststellung der Nachbarn.
- Database-Description-PDU zur Bekanntgabe der neuesten Link-State-Datenbanken.
- Link-State-Request-PDU zum Anfordern von Routing-Informationen bzw. Änderungen von den Nachbarn.
- Link-State-Update-PDU zum Verteilen von Routing-Informationen an die Nachbarn.
- Link-State-Acknowledgement-PDU zur Bestätigung eines Updates.

Die OSPF-PDU verfügt über einen Header und die Nutzdaten. In Abbildung 4-39 ist der Grobaufbau der OSPF-PDUs dargestellt. OSPF-PDUs werden im Gegensatz zu RIP direkt über IP gesendet (siehe IP-Header, Protokoll = 89). Damit müsste man OSPF eigentlich der Schicht 4 zuordnen.

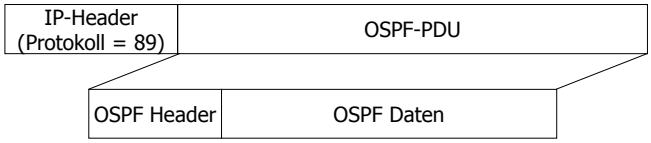


Abbildung 4-39: Grobaufbau der OSPF-PDU

In Abbildung 4-40 ist der Header etwas verfeinert dargestellt:

- Im Feld *Version* wird der Wert „2“ eingetragen.
- Das Feld *Typ* gibt den genauen Inhalt des Pakets an (Hello-PDU, Database-Description-PDU,...).
- Im Feld *Länge* steht die Gesamtlänge der PDU in Byte (inkl. Header).
- Das Feld *Router-Id* identifiziert den sendenden Router. In *Area-Id* wird der Bereich angegeben, in dem die PDU erzeugt wurde.
- Im Feld *Prüfsumme* wird eine Prüfsumme über den PDU-Inhalt ohne den Inhalt des Feldes *Auth* (1+2) angegeben.
- *Auth-Typ* gibt die Art der Identifikation an. Mögliche Angaben sind hier entweder eine passwort-gestützte Authentifizierung oder eine kryptografische Summe.
- Im Feld *Auth* (im Bild aus Darstellungsgründen aufgeteilt auf *Auth*(1) und (2)) wird je nach Inhalt des Feldes *Auth-Typ* die eigentliche Authentifikation eingetragen.

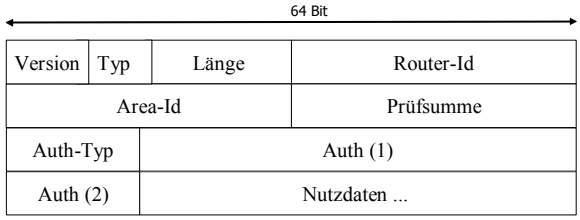


Abbildung 4-40: OSPFv2-Header

Auf den Aufbau der Hello-PDU soll exemplarisch eingegangen werden. Im Feld *Netzwerkmaske* wird die Netzwerk- oder Subnetzmaske des lokalen OSPF-Router-Ports angegeben, über den die PDU gesendet wird. Im Feld *Intervall* wird in Sekunden angegeben, in welchen Abständen eine Hello-PDU gesendet wird. Im Feld *Options* werden einige Bit für optionale Angaben reserviert, auf die hier nicht näher eingegangen werden soll. Das Feld *Prio* gibt die Router-Priorität an und wird bei Einsatz von dedizierten Routern verwendet. Designierte Router haben eine höhere Priorität als konventionelle OSPF-Router. Im Feld *Router-Dead-Intervall* wird die Zeit in Sekunden angegeben, die verstreichen darf, bis ein Nachbar-Router als ausgefallen gilt.

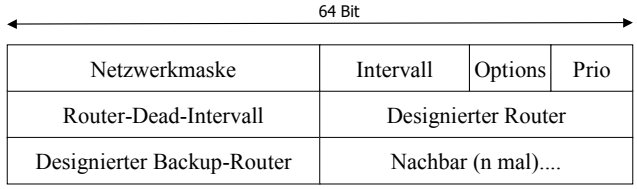


Abbildung 4-41: OSPFv2-Hello-PDU

Das Feld *Designierter Router* ist nur ausgefüllt, wenn es sich beim Sender um einen designierten Router handelt. In diesem Fall steht die IP-Adresse des Ports in diesem Feld. Für das Feld *Designierter Backup-Router* gilt das gleiche. Der Wert 0.0.0.0 wird verwendet, wenn der sendende Router weder designierter noch designierter Backup-Router ist. Im Feld *Nachbar* wird schließlich eine Liste von Ids der Nachbar-Router angegeben, die dem sendenden Router bereits bekannt sind.

Die Hello-PDU (Abbildung 4-41) wird verwendet, um bei der Initialisierung eines Routers alle Nachbarn zu ermitteln und um in regelmäßigen Abständen zu prüfen, ob die Verbindungen zu den Nachbarn noch verfügbar sind. Weiterhin wird die PDU eingesetzt, um in Broadcast-Netzen einen designierten Router und einen Backup-Router zu bestimmen.

In Abbildung 4-42 ist eine Initialisierungsphase für eine OSPF-Instanz 1 dargestellt, die zwei Nachbarn mit Hello-PDUs anspricht. Beide antworten – etwas vereinfacht dargestellt – mit einer Hello-PDU und übergeben ihre Router-Ids.

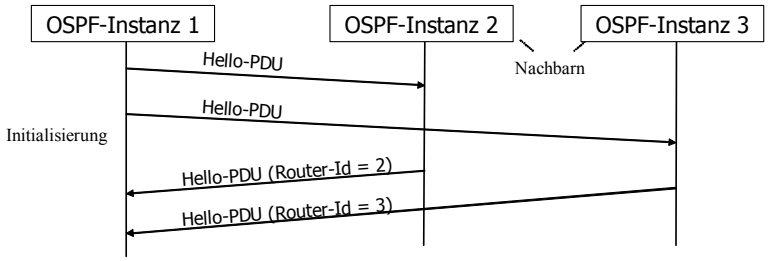


Abbildung 4-42: OSPF-Hello-Protokoll

Wenn die Nachbarschaften aufgebaut sind, werden die Routing-Informationen (die Link-State-Databases) ständig synchronisiert. Dies geschieht über Database-Description-PDUs.

Über Link-State-Request-PDUs können von den Nachbarn gezielt Informationen (Link State Advertisements = LSA) zu Links angefordert werden. Antworten werden über Link-State-Update-PDUs gesendet, und diese werden wiederum durch Link-State-Ack-PDUs bestätigt. In der Link-State-, der Link-State-Update- und der Link-State-Ack-PDU können Listen von Link-States angegeben werden he Abbildung 4-43).

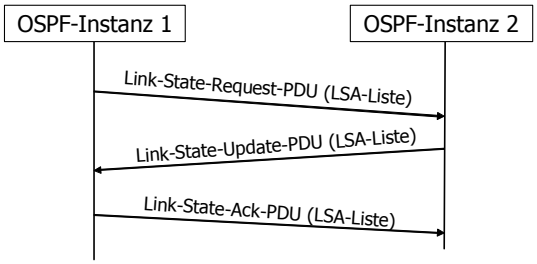


Abbildung 4-43: OSPF-Link-State-Request-Bearbeitung

Die Einzelheiten zu den OSPF-PDUs können in den einzelnen RFCs nachgelesen werden. Auf eine detaillierte Darstellung wird hier verzichtet.

IS-IS

Das ISO-Protokoll IS-IS (Intermediate System to Intermediate System intra-domain routing information exchange protocol) wird wie OSPF in großen Netzen als IGP genutzt. Während sich Protokolle wie RIP und OSPF stark am Internet-Standard

orientieren, baut IS-IS auf den OSI-Standards auf. IS-IS ist in der aktuellen Version im internationalen Standard ISO/IEC 10589:2002 beschrieben.

Die grundlegende Funktionsweise von IS-IS und OSPF ist sehr ähnlich. Auch IS-IS verwaltet eine Datenbank (Link State Database), in der die Verbindungsinformationen der gesamten Topologie erfasst werden. Auf Basis dieser Daten wird wie bei OSPF ein Spanning-Tree mit der jeweils günstigsten Verbindung zu jedem Knoten errechnet. Um die für die Berechnung des Spanning-Tree benötigten Information zu erlangen, werden auch bei IS-IS zwischen den Routern (bei IS-IS spricht man von einem Intermediate System) Nachrichten mit Verbindungsinformationen (Link State Packet) ausgetauscht. Bevor zwischen zwei Intermediate Systems (IS) eine Synchronisation der Verbindungsinformationen startet, wird wie bei OSPF eine Nachbarschaft (Adjacency) aufgebaut. Dazu sendet ein IS, zum Beispiel nach dem Start, eine *IS-to-IS-Hello-PDU* (*IIH-PDU*), um andere IS zu finden. Ein benachbarter IS empfängt die IIH-PDU, prüft die in der IIH-PDU enthaltenen Informationen (z.B. System ID des Senders) und sendet eine IIH-PDU als Bestätigung zurück. Die IIH-PDUs werden auch verwendet, um bei einer aufgebauten Nachbarschaft periodisch die Qualität der Verbindung zwischen den IS zu prüfen.

Wie schon angesprochen ist die grundlegende Funktionsweise von IS-IS und OSPF sehr ähnlich. Im Folgenden werden daher nur einige Unterschiede[20] zwischen den beiden Interior-Gateway-Protokollen kurz beschrieben.

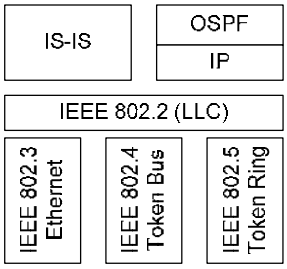


Abbildung 4-44: Beispielhafter Protokollstapel von IS-IS und OSPF

Bei der Betrachtung der Protokollstapel ist zu sehen, dass IS-IS direkt auf einem Protokoll der Schicht zwei (z.B. Ethernet) aufsetzt. Wie Abbildung 4-44 zeigt, nutzt OPSF im Gegensatz dazu das IP-Protokoll aus der Schicht drei, um Nachrichten zu versenden. IS-IS muss somit die verschiedenen Protokolltypen der Schicht zwei unterstützen. Abhängig von der eingesetzten Technologie muss IS-IS die Nachrichten entsprechend dem jeweiligen Protokoll der Schicht zwei übergeben. Auf den

[20] Eine ausführliche Beschreibung der Unterschiede zwischen IS-IS und OSPF ist in „IS-IS and OSPF Difference Discussions“ von Bhatia, Manral und Ohara (Internet Draft, 2005) zu lesen.

Dienst der Fragmentierung muss IS-IS verzichten. Ein OSPF-Router kann dagegen ohne Rücksicht auf die MTU (Maximum Transmission Unit) Nachrichten an IP übergeben. Sind die Nachrichten zu lang, wird eine IP-Fragmentierung durchgeführt. IS-IS muss eine solche Fragmentierung entweder selbst durchführen, oder die Pakete müssen unter Beachtung der MTU erstellt werden. Durch die direkte Nutzung der Dienste aus der Schicht zwei wird kein zusätzlicher Protokoll-Header benötigt. Durch die Vermeidung des zusätzlichen Overheads werden die Nachrichten des IS-IS-Paketes etwas kleiner.

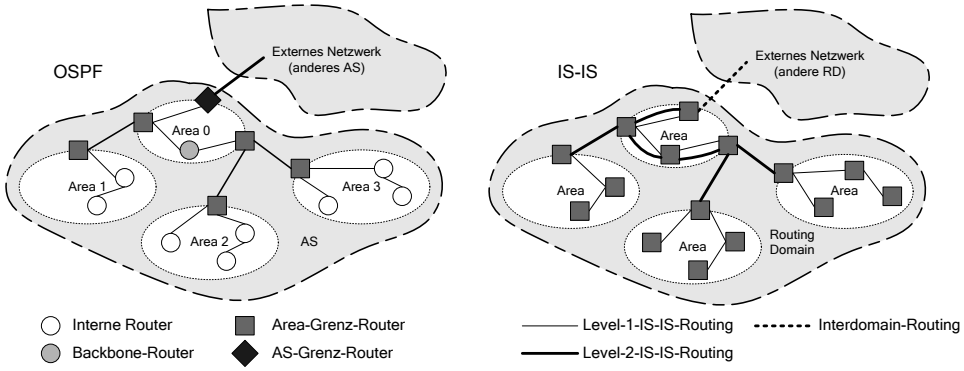


Abbildung 4-45: Vergleich der OSPF und IS-IS Topologie

IS-IS unterteilt ähnlich dem OSPF das Netzwerk in kleinere Bereiche (Areas) um die Vorteile von hierarchischem Routing zu nutzen. Wie schon weiter oben am Beispiel von OSPF beschrieben, werden durch das hierarchische Routing die Routing-Tabelle kürzer und somit die Berechung der günstigsten Pfade einfacher. Wie in Abbildung 4-45 zu sehen ist, werden bei IS-IS die Bereichsgrenzen über die Verbindungen definiert. Die Verbindungen (Links) zwischen zwei IS werden in drei Klassen aufgeteilt:

- *Level-1-IS-IS-Routing* zwischen Intermediate Systems innerhalb einer Area.
- *Level-2-IS-IS-Routing* zwischen Intermediate Systems innerhalb einer Area und zwischen Areas einer Routing-Domäne.
- *Interdomain-Routing* zwischen eigenständigen Routing-Domänen (Autonomen Systemen).

Im Gegensatz zu OSPF gibt es bei IS-IS keine gesonderte Backbone-Area. Die Verbindung der Areas wird über Level-2-IS-IS-Routing realisiert. Eine Verbindung kann bei IS-IS neben einem Level-1- auch für ein Level-2-IS-IS-Routing genutzt werden.

Das IS-IS-Protokoll wird gegenwärtig, wie das OSPF-Protokoll, in großen IP Netzwerken eingesetzt. IS-IS wird von den meisten Tier-1-ISPs als Routing-Protokoll gewählt. Viele große ISPs wechseln zu IS-IS, da es im Vergleich zu OSPF als stabi-

ler und einfacher zu implementieren gilt. OSPF ist als Routing Protokoll bei mittleren bis großen ISPs stark verbreitet. Bei IP-basierten Unternehmensnetzwerken wird fast ausschließlich OSPF verwendet.

Es soll noch festgehalten werden, dass die Entscheidung, welches Routing-Protokoll verwendet wird, von der grundlegenden Entscheidung, welche Protokollstandards im Netz genutzt werden, abhängt. IS-IS setzt nämlich auf dem OSI-Standard und OSPF auf dem Internet-Standard auf. Diese Abhängigkeit ergibt sich zum Beispiel aus den Unterschieden bei der Adressierung der Knoten. Wird in einem Netzwerk das Internet-Protokoll (IP) eingesetzt, bevorzugt man OSPF oder RIP (siehe dazu Abbildung 4-44). Werden hingegen im Netzwerk ISO-Protokolle (wie etwa ISO-9542-Standard, ISO End System to Intermediate System Protocol) eingesetzt, ist IS-IS die bevorzugte Wahl. Es existieren aber auch Vorschläge wie der RFC 1195, um IS-IS für IP-basiertes Routing oder in gemischten Umgebungen zu nutzen.

Border-Gateway-Protokolle BGP und BGP-4

BGP (RFCs 1654, 1771, 1772 und 1773) bzw. dessen neuere Version BGP-4 (Border Gateway Protocol, auch BGPv4)[21] ist ein EGP, das heute im Internet eingesetzt wird. Das ältere BGP-Protokoll ist in RFC 1163 beschrieben. In der derzeit eingesetzten Version 4 ist es im RFC 1771 spezifiziert. BGP ist ein Pfadvektorprotokoll (Path Vector Protocol) und ermöglicht das Routing zwischen verschiedenen autonomen Systemen (AS) und ähnelt vom Grundprinzip her dem Distance-Vector-Verfahren. Im Gegensatz zum Distance-Vector-Verfahren verwendet BGP jedoch keine Kosteninformation (wie die Anzahl der Hops), sondern nur die Pfadinformation.

Derzeit ist im Internet nur BGP als spezielle Ausprägung eines EGP im Einsatz. Die Pfade werden hier auf AS-Ebene, nicht auf Router-Ebene verwaltet. Jeder BGP-Router führt eine Datenbank mit Routen zu allen erreichbaren autonomen Systemen. Die heutigen Routing-Tabellengrößen umfassen ca. 200.000 Einträge für ca. 26.000 autonome Systeme. Unmittelbar benachbarte Router bezeichnet man auch als *Peers* bzw. *BGP-Speaker*. Die Verbindung zwischen *Peers* wird als *Peer-Verbindung* bezeichnet[22].

Kernstück des BGP-Protokolls ist eine UPDATE-Nachricht. Mit Hilfe von UPDATE-PDUs teilen sich Router die Existenz neuer Routen über sog. Announcements mit und kommunizieren auch den Wegfall bestehender Routen (Withdrawals). Der Empfänger einer UPDATE-Nachricht entscheidet anhand seiner Routing-Policies,

---

[21] BGP gibt es erst seit 1989, BGP-4 seit 1993.

[22] BGP kann auch innerhalb eines AS verwendet werden und wird dann als IBGP bezeichnet.

was mit den UPDATE-Informationen passieren soll (verarbeiten und Routen anpassen, nur weiterleiten).

BGP-Router kennen also die beste Route zu einem anderen AS als vollständigen Pfad. Ein BGP-Router informiert periodisch alle Nachbar-BGP-Router genau über die zu nutzenden Routen. Zyklen in Routen werden bei Übernahme der Information gefunden, indem geprüft wird, ob die eigene AS-Nummer in der Route ist. Falls dies der Fall ist, wird die Route nicht akzeptiert. Damit tritt auch das Count-to-Infinity-Problem nicht auf. BGP-Router überwachen sich gegenseitig über ein Heartbeat-Protokoll, um Ausfälle schnell zu erkennen.

Die Routing-Information wird in den BGP-Routern in sog. *RIB* (Routing Information Base) abgelegt. BGP-Router verwenden zur Auswahl der besten Routen eine *Routing-Policy* mit unterschiedlichen Regelwerken, in die Sicherheitsaspekte, Kostenaspekte und evtl. Sperren von Routen für bestimmte Absender und Empfänger eine Rolle spielen. Eine Route, welche die Regeln verletzt, wird auf „unendlich" gesetzt. Eine konkrete Vorgabe zur Strategie ist aber nicht gegeben. Dies wird vielmehr den einzelnen Routern (bzw. deren Administratoren) überlassen.

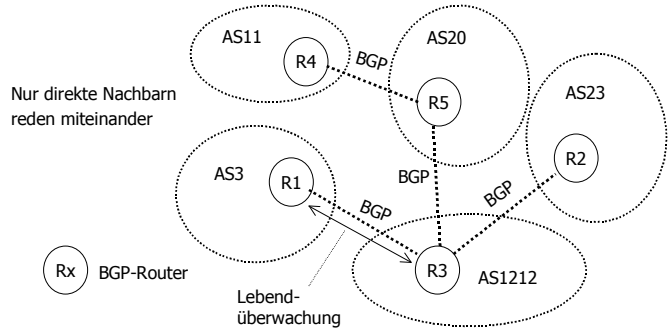


Abbildung 4-46: BGP-Router-Kommunikation

Die Arbeitsweise der BGP-Router ist in Abbildung 4-46 dargestellt. In jedem AS ist (mind.) ein BGP-Router, der als Stellvertreter des AS dient und mit den anderen BGP-Routern kommuniziert. BGP-4 wird also zwischen AS-Grenzroutern (ASBR) verwendet.

BGP-PDUs. BGP nutzt TCP (Port 179)[23] als Transportprotokoll für den Nachrichtenaustausch (verbindungsorientiert!). Folgende BGP-PDUs sind definiert:

- OPEN-PDU: Diese PDU wird nach dem TCP-Verbindungsaufbau gesendet und dient der Identifikation und Authentifizierung sowie dem Parameter-

[23] Auch hier wird ein Transportprotokoll für eine Schicht-3-Aufgabe verwendet.

austausch (Timer für die Zeitüberwachung zwischen Heartbeat- und Update-PDUs, BGP-Identifier des sendenden Routers, Identifikation des Autonomen Systems,...).
- UPDATE-PDU: Mit dieser PDU wird ein Advertisement, also ein Pfad zu einem bestimmten Ziel an den Nachbarn gesendet. Auch bei Änderungen der Routen wird eine UPDATE-PDU gesendet.
- KEEPALIVE-PDU: Diese PDU dient als Bestätigungs-PDU für die OPEN-PDU (OPEN-Response). Ein Nachbar teilt mit der PDU auch mit, dass er noch am Leben ist.
- NOTIFICATION-PDU: Diese PDU dient dazu, einen Nachbarn darüber zu informieren, dass in einer vorhergehenden PDU ein Fehler war oder dass der sendende BGP-Router die Verbindung schließen möchte.

Die nähere Beschreibung des PDU-Aufbaus der BGP-4-PDUs kann im RFC 1771 nachgelesen werden. BGP-4 unterstützt im Gegensatz zum Vorgänger BGP auch CIDR und die Aggregation von Routen. Heute wird im Internet fast ausschließlich BGP-4 eingesetzt.

Multicast-Routing

Der Vollständigkeit halber sollen zum Themenbereich „Routing im Internet“ noch Routing-Protokolle für das Multicast-Routing erwähnt werden. Das Protokoll *IGMP* (Inter Group Management Protocol) dient im Internet als Mechanismus, über den Anwendungen (Multicast-Clients) sich dynamisch in Multicast-Gruppen registrieren können. Multicasting stellt eine wichtige Technik für Video- und Audiokonferenzen im Internet dar.

RIP und OSPF werden für das Routing von Unicast-Nachrichten verwendet. Für das Routing von Multicast-Nachrichten im Internet sind spezielle Routing-Protokolle erforderlich, die in sog. Multicast-Routern implementiert werden.

Multicast-Routing-Protokolle, die im Internet verwendet werden, sind z.B. DVMRP (Distance Vector Multicast Protocol, RFC 1075) und MOSPF (Multicast OSPF, RFC 1854). DVRMP setzt auf RIP auf, und MOSPF ist eine Erweiterung von OSPF. Daher nutzt DVRMP auch ein von RIP abgeleitetes Distance-Vector-Protokoll und MOSPF ein Link-State-Verfahren.

Ein Zusammenschluss von multicastfähigen Routern im Internet entstand aus einer Initiative der IETF mit dem Ziel, Multicast-Techniken auszuprobieren. Als Bezeichnung für das Netzwerk wurde *Mbone* (Multicast Backbone on the Internet) verwendet. Heute beherrschen nahezu alle Router auch Multicast-Routing-Protokolle, weshalb das spezielle Mbone nicht mehr weiterentwickelt wird. Multicast-Router (M-Router) sind im Prinzip heute Standard-Router, die neben RIP/OSPF noch das Routing von Multicast-Nachrichten unterstützen.

Auf die Funktionsweise der Protokolle, des Mbone-Systems und dessen spezielle Routing-Protokolle soll an dieser Stelle nicht weiter eingegangen werden.

Multi-Protocol Label Switching (MPLS)

Bevor das Thema Routing abgeschlossen werden kann, soll noch auf die MPLS-Technik eingegangen werden, die in erster Linie für die Kommunikation zwischen den Routern von ISPs genutzt wird und daher für die externen Internet-Nutzer kaum sichtbar wird.

MPLS vereint unterschiedliche Protokolle in einem Router und kombiniert die Vorteile von Switching mit Routing. Mit MPLS versucht man, die Vorteile von virtuellen Leitungen (Virtual Circuits) mit den Vorteilen von Datagrammen zu verbinden, um damit einen Leistungsgewinn zu erzielen. Insbesondere die aufwändige Suche nach Einträgen in den Routing-Tabellen sollte beschleunigt werden. Dazu wurden zu den IP-Paketen sog. Labels ergänzt, die von den Routern für eine schnelle Routing-Entscheidung verwendet werden.

Ein MPLS-fähiger Router weist jedem Präfix in der Routing-Tabelle ein aus seiner Sicht eindeutiges Label zu, das er seinen Nachbar-Routern mitteilt. Die notwendige Advertisement-PDU wird im *Label Distribution Protocol* (*LDP*, RFC 3036) definiert. Damit werden die Nachbar-Router aufgefordert, diese Labels an alle IP-Pakete zu hängen, die an eine IP-Adresse mit dem zugeordneten Präfix adressiert werden. Bei ankommenden Paketen muss der Router nun nicht mehr die aufwändige Suche in der Routing-Tabelle über den IP-Zieladress-Suchalgorithmus (größte Übereinstimmung) durchführen, sondern kann den Ausgangsport anhand des Labels wesentlich schneller ermitteln.

MPLS arbeitet zwischen den Schichten 2 und 3 des OSI-Schichtenmodells als Zwischenschicht. Urprünglich sind die Funktionalitäten von MPLS durch das Feature "Tag Switching" in den Routern von Cisco bekannt geworden. Mittlerweile gibt es eine Fülle von RFCs, die sich mit MPLS befassen. Für eine weitergehende Betrachtung sei stellvertretend der RFC 3031 genannt, der die MPLS-Architektur beschreibt.

4.3 Steuer- und Konfigurationsprotokolle im Internet

In diesem Abschnitt werden wichtige Steuer- und Konfigurationsprotokolle für die Schicht 3 des Internets erörtert. Diese Protokolle sind essentieller Bestandteil internet-basierter Netzwerke.

4.3.1 Internet Control Message Protocol (ICMP)

Das Internet Control Message Protocol (ICMP, RFC 792) ist ein Steuerprotokoll, das von Hosts und Routern benutzt wird. Es dient der Übertragung von unerwarteten Ereignissen und auch für Testzwecke in der Vermittlungsschicht.

ICMP gehört in die Internet-Vermittlungsschicht und nutzt für die Datenübertragung das IP-Protokoll. ICMP-PDUs werden daher wie TCP- oder UDP-Pakete als IP-Nutzdaten übertragen.

In Abbildung 4-47 ist der Aufbau einer ICMP-Nachricht mit Header und Nutzdatenteil dargestellt. Im Header sind der Nachrichtentyp und ein Code definiert. Weiterhin ist im Header eine Prüfsumme zu finden. Im Nutzdatenteil wird ein Teil des ursprünglichen IP-Pakets übertragen, und zwar der IP-Header und 64 Bit des Nutzdatenteils, welches das Ziel nicht erreicht hat.

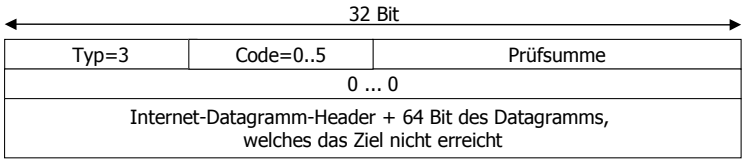


Abbildung 4-47: ICMP-Nachricht

Stellt ein IP-Router ein Problem fest, sendet er eine ICMP-Nachricht direkt an den Absender. Einige typische Beispiele von ICMP-Nachrichten sollen stellvertretend erwähnt werden:

- „Destination unreachable“ wird von einem Router abgesetzt, der ein Datagramm nicht ausliefern kann (Typ = 3, Code = 1).
- „Network unreachable“ wird von einem Router abgesetzt, wenn ein adressiertes Netzwerk von einem Router nicht erreichbar ist (Typ = 3, Code = 0).
- „Source quench“ wird von einem Router gesendet, wenn sein Speicherplatz erschöpft ist (Typ = 4, Code = 0).
- Das *ping*-Kommando verwendet ICMP-PDUs (Echo Request, Echo Reply), um zu prüfen, ob ein bestimmter Host oder Router erreichbar ist (Typ = 0, Code = 0).
- Das *traceroute*-Kommando, das zur Ermittlung einer Route zwischen zwei Rechnern dient, verwendet u.a. die ICMP-Nachricht „Time Exceeded“ (Typ=11).

Alle derzeit definierten ICMP-Nachrichten sind dem entsprechenden RFC zu entnehmen.

4.3.2 ARP und RARP

Bisher sind wir davon ausgegangen, dass jeder Host neben den IP-Adressen[24] der Zielhosts auch die Schicht-2-Adressen kennt, die für die Adressierung in der Sicherungsschicht notwendig sind. Für die Adressierung im Ethernet braucht man z.B. eine 48 Bit lange MAC-Adresse (Beispiel: für eine Ethernet-Adresse: 1A-23-F9-CD-06-9B). In anderen Netzwerk-Architekturen (siehe z.B. im ISO/OSI-Modell) werden die MAC-Adressen statisch per Vorabkonfiguration mit den Schicht-3-Adressen

[24] Das können auch mehrere je Host sein, da die IP-Adressen an das Netzwerkinterface gekoppelt sind.

gekoppelt. Im internetbasierten LAN ist dies nicht so. Jeder Host kennt initial nur seine eigene(n) MAC-Adresse(n) und nicht die der Partnerhosts.

Wie bekommt ein Host also die MAC-Adresse eines Partners? Er muss sie sich zur Laufzeit besorgen, und hierfür dient das Steuerprotokoll ARP, das eigentlich zur Hälfte in die Schicht 2 und zur anderen Hälfte in die Schicht 3 gehört. ARP (Address Resolution Protocol) dient dem *dynamischen Mapping* von IP-Adressen auf Schicht-2-Adressen (MAC-Adressen).

ARP funktioniert im Wesentlichen wie folgt:

- Wenn ein Zielhost adressiert wird, der bisher noch nicht oder schon längere Zeit nicht mehr adressiert wurde, wird ein *ARP-Request* in einem limited Broadcast im LAN versendet. In dem ARP-Request ist die IP-Zieladresse eingetragen. Mit diesem ARP-Request fragt der Host alle anderen Rechner im LAN, wer denn diese Adresse kennt.
- Der Zielhost oder ein anderer antwortet mit einem *ARP-Reply* und übergibt dabei seine MAC-Adresse.

Damit die MAC-Adressen nicht ständig neu angefragt werden müssen, führt jeder Host einen ARP-Cache und merkt sich darin die Schicht-2-Adressen zu den entsprechenden IP-Adressen, mit denen er gerade zu tun hat. Der ARP-Cache wird aber periodisch bereinigt, um evtl. Inkonsistenzen zu vermeiden. Jeder Eintrag hat hierzu ein TTL-Feld (Time-To-Live). Wird ein Eintrag eine bestimmte Zeit (z.B. 20 Minuten) nicht benötigt, so wird er aus dem ARP-Cache gelöscht.

ARP-PDU-Aufbau. In Abbildung 4-48 ist der ARP-Header mit seinen Feldern dargestellt. Das Protokoll ist sehr allgemeingültig gehalten, wird aber überwiegend für Ethernet-Netzwerke verwendet.

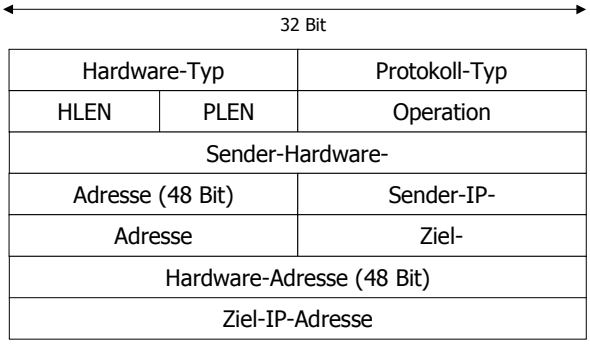


Abbildung 4-48: ARP/RARP-Header

Die Felder bedeuten im Einzelnen:

- *Hardware-Typ:* Hier wird der Hardware-Adresstyp angegeben, wobei derzeit nur „1“ für Ethernet definiert ist.
- *Protokoll-Typ:* Dieses Feld enthält den Adresstyp des höherliegenden Protokolls. Wird ARP mit IP verwendet, steht in dem Feld der Wert 0x0800.
- *HLEN:* Das Feld enthält die Länge der Hardware-Adresse.
- *PLEN:* Dieses Feld enthält die Länge der Schicht-3-Adresse (z.B. IP-Adressenlänge).
- *Operation:* In diesem Feld steht die Protokoll-Operation, die ausgeführt werden soll. Mögliche Inhalte sind:
    - 1 = ARP-Request
    - 2 = ARP-Response
    - 3 = RARP-Request
    - 4 = RARP-Response
- Die restlichen Feldnamen sind sprechend und enthalten die Adressinformation (*Sender-Hardware-Adresse*, *Ziel-Hardware-Adresse* = MAC-Adresse und *Ziel-IP-Adresse*).

Die Belegung der einzelnen Felder hängt von der aktuellen Protokolloperation ab. Wird eine MAC-Adresse gesucht, so ist z.B. die IP-Adesse ausgefüllt.

Der Header wird auch noch für ein weiteres Steuerprotokoll, und zwar für das *Reverse ARP (RARP*, RFC 903*)* verwendet. Dieses Protokoll wird benötigt, wenn – im Gegensatz zu ARP – zu einer MAC-Adresse eine IP-Adresse gesucht wird. Klassischer Anwendungsfall ist, wenn eine plattenlose Workstation hochfährt und ihre IP-Adresse benötigt. In diesem Fall sendet sie einen RARP-Request, und ein zuständiger RARP-Server beantwortet die Anfrage.[25]

ARP-Proxy. Ein Problem gibt es noch beim Einsatz von ARP zur MAC-Adressenermittlung: Wenn der Zielhost nicht im lokalen Netz ist, kann er auch nicht antworten. In diesem Fall übernimmt der zuständige IP-Router die Rolle eines Stellvertreters (ARP-Proxy) und sendet seine MAC-Adresse an den anfragenden Host. Über die IP-Adresse des gewünschten Zielhosts kann er aus seiner Routing-Tabelle die gewünschte Route bestimmen.

Der Router im Zielnetz muss nun bei Ankunft des ersten IP-Pakets für den betroffenen Zielhost dessen MAC-Adresse ermitteln, sofern er sie noch nicht kennt. Dies erledigt er ebenfalls über einen ARP-Request. Der Zusammenhang ist in dem typi-

[25] Mittlerweile wird für diese Aufgabe dem BOOTP-Protokoll der Vorzug gegeben, das in RFC 951 beschrieben ist. RARP hat nämlich den Nachteil, dass es *limited* Broadcasts sendet und damit in jedem Teilnetz ein RARP-Server stehen muss. Im Gegensatz dazu benutzt BOOTP UDP-Nachrichten, welche die Router weiterleiten. Noch fortschrittlicher ist DHCP.

schen Campusnetz in Abbildung 4-49 dargestellt. Möchte in diesem Netz z.B. ein Host des Informatik-Netzes mit einem Host im BWL-Netz kommunizieren, agiert der Router mit der IP-Adresse 192.31.65.1 als ARP-Proxy.

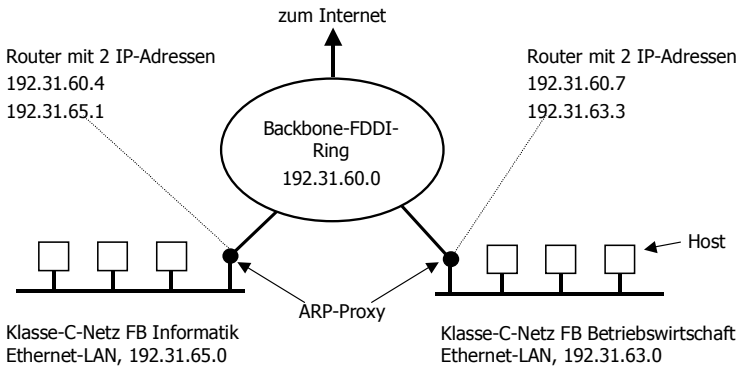


Abbildung 4-49: Netzwerkbeispiel für einen ARP-Request über einen ARP-Proxy nach (Tanenbaum 2003a)

4.3.3 NAT und IP-Masquerading

Mit *NAT* (Network Address Translation) werden grundsätzlich Adressen eines privaten Netzes über Abbildungstabellen öffentlich registrierten IP-Adressen zugeordnet. Dies führt dazu, dass interne Rechner keine öffentlichen IP-Adressen haben müssen. Die Zuordnung kann 1:1 erfolgen, d.h. für jede interne IP-Adresse wird eine eigene externe IP-Adresse zugeordnet. Mit *IP-Masquerading* oder PAT (Port and Address Translation)[26] werden alle internen IP-Adressen auf genau eine öffentlich registrierte Adresse abgebildet. Um diese Abbildung durchführen zu können, werden Portnummern benutzt.

NAT/PAT wird heute auch in *DMZ* (De-Militarized Zone) eingesetzt, damit interne Rechner mit der Außenwelt ohne Verletzung der Sicherheitskriterien im Netzwerk kommunizieren können. Ohne zu stark auf sicherheitsrelevante Themen im Internet einzugehen, soll NAT im Zusammenhang mit DMZ kurz erläutert werden.

Eine DMZ ist in einem Netzwerk wie z.B. einem Unternehmens- oder einem Universitätsnetz, das an das globale Internet angeschlossen ist, erforderlich, um einen gewissen Schutz vor Angriffen auf interne Hosts zu bieten. Eine DMZ[27] ist gewissermaßen ein Zwischennetz zwischen dem globalen Internet und dem internen

[26] Auch als 1:n-NAT bezeichnet.

[27] Der Begriff stammt aus dem Militärischen (z.B. de- oder entmilitarisierte Zone zwischen Nord- und Südkorea).

Netzwerk, wie dies z.B. in Abbildung 4-50 skizziert ist, das einen Sicherheitsbereich darstellt.

Möchte ein Host aus dem internen Netzwerk mit einem Host im Internet kommunizieren, so geht dies nur über die DMZ. In der DMZ stehen die Rechner, die nach außen hin sichtbar sein sollen.

Spezielle NAT-Server oder IP-Router, welche die NAT-Aufgaben zusätzlich übernehmen, arbeiten nach außen hin als *Stellvertreter* (Proxies) für alle internen Hosts. Sie tauschen bei ausgehenden und entsprechend bei ankommenden IP-Paketen die IP-Adressen und die Ports der Transportschicht aus. Bei ausgehenden IP-Paketen wird in das Feld *Quell-IP-Adresse* im IP-Header die IP-Adresse und im TCP-Paket der *Quellport* durch den NAT-Server eingetragen. Bei ankommenden IP-Paketen wird das Feld *Ziel-IP-Adresse* ausgetauscht. Durch dieses Mapping wird nach außen hin nur noch der NAT-Server gesehen, und die interne Netzwerkstruktur bleibt verborgen. Damit können auch keine internen Hosts direkt adressiert werden, was zur Erhöhung der Sicherheit beiträgt.

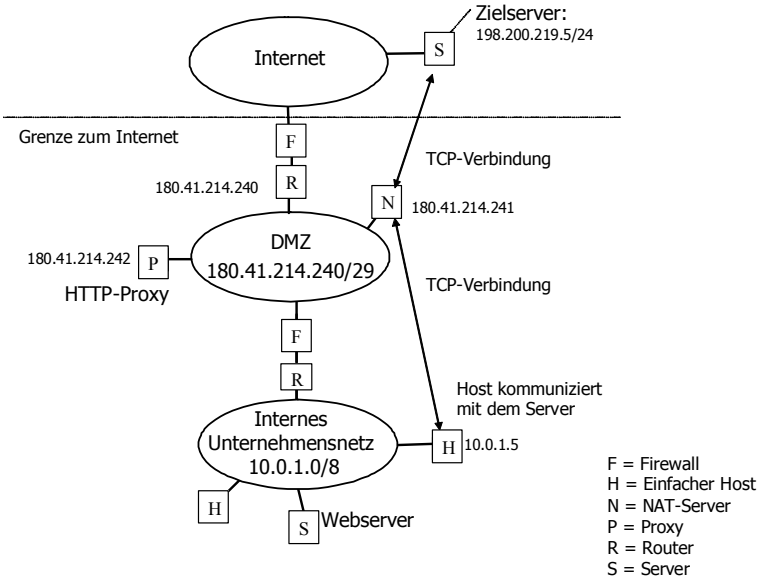


Abbildung 4-50: Beispielnetz mit DMZ

Um nun zu zeigen, wie das Ganze funktioniert, soll nochmals das Beispielnetz aus Abbildung 4-50 herangezogen werden. Ein Host mit der IP-Adresse 10.0.1.5 möchte gerne mit einem Server in einem anderen Netzwerk eine TCP-Verbindung aufbauen. Die Kommunikation geht für alle Beteiligten unbemerkt über den NAT-Server. Der Zielrechner (bzw. dessen Netzwerkanbindung) hat die Adresse

198.200.219.5. In der DMZ ist eine IP-Netzwerkadresse mit maximal 14 Hostadressen (16-2) zugewiesen. Die DMZ enthält verschiedene Rechner, meist Router und Proxy-Server, aber auch den NAT-Server.[28] Man sieht also, dass die interne Netzstruktur verborgen werden kann.

Bei Verbindungsaufbauwünschen von außen, die an einen internen Server gerichtet sind, wird ähnlich verfahren. Dadurch wird bei TCP die Verbindung immer unterbrochen, und es gibt keine echte Ende-zu-Ende-Verbindung mehr. Es werden sogar zwei TCP-Verbindungen aufgebaut, eine zwischen Quellhost und NAT-Server und eine zwischen NAT-Server und Zielhost. Dies ist allerdings nur dem NAT-Server bekannt.

Schon beim Verbindungsaufbau werden die IP-Pakete, die nach außen gehen, verändert. Der NAT-Server ändert die Connect-Request-PDU, indem er in das Feld *Quell-IP-Adresse* seine eigene Adresse einträgt. Er verwaltet in einer Tabelle, welches Mapping er durchgeführt hat, um auch PDUs von der Gegenseite zuordnen zu können. Die Connect-Response-PDU wird in der gleichen Weise maskiert, wobei hier die Adresse des Quellhosts eingetragen wird (vgl. hierzu Abbildung 4-51).

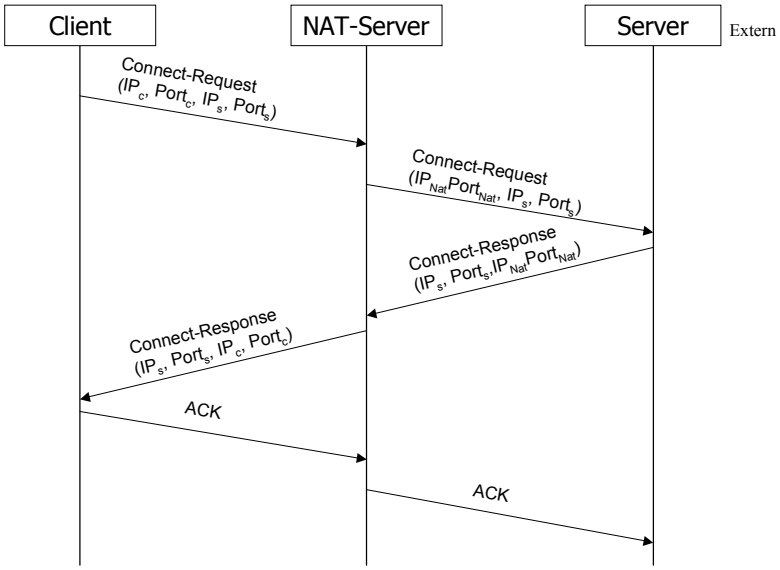


Abbildung 4-51: Sequenzdiagramm zur TCP-Verbindungsaufbau mit NAT

[28] Die Bedeutung einer Firewall, eines Proxy-Servers und eines Paketfilters kann (Tanenbaum 2003a) entnommen werden.

Selbstverständlich gibt es in dem Beispielnetz auch noch Firewalls, ggf. Paketfilter, Router und Proxy-Server. Diese Rechner spielen aber in Bezug auf NAT keine große Rolle, bis auf die Tatsache, dass die NAT-Funktionalität auch im Router sein kann. Firewalls, Proxies und Paketfilter sichern das Netzwerk ab.

Die Begriffe NAT, PAT und IP-Masquerading werden oft nicht eindeutig verwendet. Man spricht von *IP-Masquerading*, weil die internen IP-Adressen eines Netzwerks mit der IP-Adresse des NAT-Servers maskiert werden. Nach außen hin sind die lokalen IP-Adressen somit gar nicht sichtbar. Damit hat NAT natürlich auch noch den Vorteil, dass man massiv Netzwerkadressen einsparen kann. Intern kann man nämlich eine beliebige, nach außen nicht sichtbare Netzwerknummer verwenden. Hier eignen sich vor allem die privaten IP-Adressen (10.0.0.0/8 usw.), die ohnehin nach außen nicht geroutet werden. Für ein Netzwerk (z.B. ein Unternehmensnetz) benötigt man somit nach außen hin nur noch eine offizielle IP-Adresse. Wenn man einen Mailserver, einen WWW-Server bzw. deren Proxies und ggf. weitere Dienste von außen adressierbar machen möchte, kommen noch ein paar weitere offizielle Adressen hinzu.

Abschließend sei noch erwähnt, dass in einem gut gesicherten Netz alle Anwendungsdienste in der DMZ durch Proxies vertreten werden. Ein direkter Kontakt eines internen Rechners mit dem Internet ist damit ausgeschlossen.

4.3.4 Dynamic Host Configuration Protocol (DHCP)

Netzwerkadministratoren müssen sich u.a. um die IP-Adressvergabe kümmern. Jeder einzelne Host im Netz benötigt mindestens eine IP-Adresse. Bei manueller Konfiguration ist das bereits bei kleineren Netzwerken eine nicht zu unterschätzende Aufgabe. Für jeden Rechner muss eine manuelle Konfiguration der IP-Parameter ausgeführt werden. Bei mehreren Tausend Rechnern im Netz ist diese Aufgabe nicht mehr manuell abzuwickeln, Automatismen sind erforderlich.

Zu diesem Zweck wurde das Dynamic Host Configuration Protocol (DHCP, RFCs 2131 und 2132) entwickelt. Wie der Name schon sagt, handelt es sich bei diesem Protokoll um einen Mechanismus, der es ermöglicht, dass Hosts dynamisch (meist beim Startvorgang) eine IP-Adresse und weitere IP-Parameter von einem DHCP-Server anfordern können.

Neben der IP-Adresse kann ein Host, DHCP-Client genannt, die Subnetzmaske, die Adresse des DNS-Servers (wird weiter unten erläutert), des zuständigen IP-Routers und weitere Parameter wie den zuständigen Web- und Mailserver besorgen. Die Übertragung kann vollständig ohne manuellen Eingriff erfolgen.

DHCP ist eine Weiterentwicklung des BOOTP-Protokolls und ist mit BOOTP kompatibel. Ein DHCP-Server verwaltet die Adressinformation und kann verschiedene Dienste anbieten. Entweder kann einem DHCP-Client eine IP-Adresse vollautomatisch auf unbegrenzte Zeit oder für eine begrenzte Zeit (dynamische Zuweisung) zugewiesen werden. Nach Ablauf einer vorgegebenen Zeit muss dann

der DHCP-Client erneut anfragen. Die DHCP-Server führen Buch über die vergebenen und noch vorhandenen IP-Adressen.

Es soll noch erwähnt werden, dass mit DHCP einem Client per Server-Konfiguration auch eine feste IP-Adresse zugewiesen werden kann. Der Client wird dann immer mit der gleichen Netzwerkkonfiguration versorgt. Wird dies nicht explizit angegeben, so wird für den Client eine IP-Adresse aus einem IP-Adresspool dynamisch zugewiesen.

DHCP-PDU-Aufbau. Im Folgenden wollen wir einen Blick auf die DHCP-PDU werfen, die zum Austausch der Informationen zwischen Client und Server verwendet wird (Abb. 4-52).

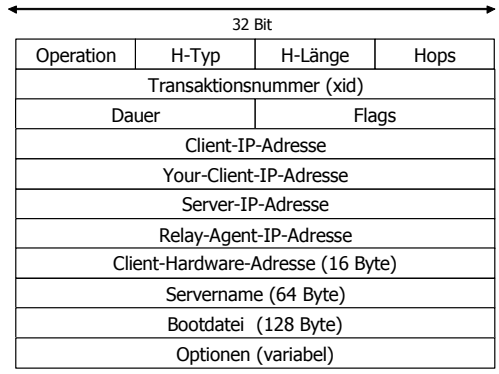


Abbildung 4-52: DHCP-PDU

Im Einzelnen enthält die DHCP-PDU folgende Felder: [29]

- *Operation:* Nachrichtenoption (1=Bootrequest, 2=Bootreply).
- *H-Typ*: Gibt Informationen über die Hardware des verwendeten Netzes (z.B. 1=10-MB-Ethernet).[30]
- *H-Länge*: Gibt die Länge der Hardwareadresse (MAC) an (z.B. 6 Byte bei 10-MB-Ethernet).
- *Hops*: Standardmäßig auf 0, kann beim Booten über Relay-Agents verwendet werden.
- *Transaktionsnummer*: Fortlaufende Identifikation der Requests.
- *Dauer*: Anzahl der Sekunden, die beim Client seit Adressanfrage bzw. Erneuerungsanfrage vergangen sind.
- *Flags*: Das erste Bit zeigt an, ob es sich um ein Multicast-Paket handelt. (TCP/IP-Stacks müssen Unicasts nicht annehmen, wenn noch keine IP-

[29] Vgl. http://www.ietf.org/rfc/rfc2131.txt, Stand: 06.07.2009.

[30] Vgl. http://www.ietf.org/rfc/rfc1700.txt, Stand: 6.07.2009.

Adresse konfiguriert ist.) Die folgenden Bits sind für den zukünftigen Gebrauch reserviert und werden derzeit nicht beachtet.

- *Client-IP-Adresse*: Dieses Feld wird nur gesetzt, wenn der Client bereits eine IP-Adresse besitzt und z.B. einen REFRESH-Prozess anstößt.
- *Your-Client-IP-Adresse*: Die vorgeschlagene IP-Adresse des DHCP-Servers.
- *Server-IP-Adresse*: Die Adresse des DHCP-Servers.
- *Relay-Agent-IP-Adresse*: Falls die Kommunikation über einen Relay-Agent, also einen Vermittler zwischen zwei Subnetzen, erfolgt, wird hier dessen IP-Adresse eingetragen.
- *Client-Hardware-Adresse*: Hardware-Adresse des Clients.
- *Servername*: Hostname des Servers (optional).
- *Bootdatei*: Name einer Bootdatei, die vom Client über TFTP geladen und ausgeführt werden soll.
- *Optionen*: Dieses Feld ist von variabler Länge, mindestens jedoch 312 Byte lang und beinhaltet sowohl Pflicht- als auch Kannfelder. Neben den DHCP-spezifischen Einstellungen, wie z.B. *Nachrichtentyp* und *Lease-Zeit*, werden hier auch verschiedene Optionen zur Konfiguration des Netzwerkadapters, insbesondere zur Konfiguration von TCP/IP aufgelistet (z.B. DNS-Server, TTL, MTU, IP-Routen, etc.). Für detaillierte Informationen sei auf RFC 1355 verwiesen.

Die Tabelle 4-6 zeigt die verschiedenen Nachrichtentypen, die im Rahmen einer DHCP-Kommunikation versendet werden können, sowie die zugehörigen Werte im entsprechenden Optionsfeld der DHCP-PDU.

Tabelle 4-6: Die Option „Nachrichtentyp“ und ihre Werte

Wert	Nachrichtentyp
1	DHCPDISCOVER
2	DHCPOFFER
3	DHCPREQUEST
4	DHCPDECLINE
5	DHCPACK
6	DHCPNAK
7	DHCPRELEASE
8	DHCPINFORM

Ablauf der Kommunikation. Der prinzipielle Ablauf einer dynamischen Anforderung einer IP-Adresse durch einen DHCP-Client ist in Abbildung 4-53 dargestellt:

- Der DHCP-Client sendet beim Bootvorgang eine Discover-PDU über direkten Broadcast ins Netz.
- Ein zuständiger DHCP-Server, der natürlich doppelt ausgelegt werden muss (er sollte kein Single Point of Failure sein), bietet dem Client eine Netzwerkkonfiguration in einer Offer-PDU an.

- Falls der Client annimmt, sendet er seinerseits eine Request-PDU direkt an den nun bekannten DHCP-Server.
- Der DHCP-Server bestätigt dies nochmals mit einer ACK-PDU.
- Die Zeit, die der DHCP-Client die Netzwerkkonfiguration nutzen darf, die sog. Lease-Zeit, wird ebenfalls in Form von zwei Parametern T1 und T2 mit übergeben. T1 gibt standardmäßig 50 % der Lease-Zeit an, T2 87,5 %. Wie in der Abbildung 4-53 zu sehen ist, wird nach Ablauf von T1 erneut eine Request-PDU zum DHCP-Server gesendet (REFRESH), wenn die Netzwerkkonfiguration noch länger erwünscht ist. Kommt eine ACK-PDU vom Server, läuft die Lease-Zeit erneut an. Kommt keine ACK-PDU, wird nach Ablauf des Timers T2 erneut ein Discovery über einen Broadcast eingeleitet.

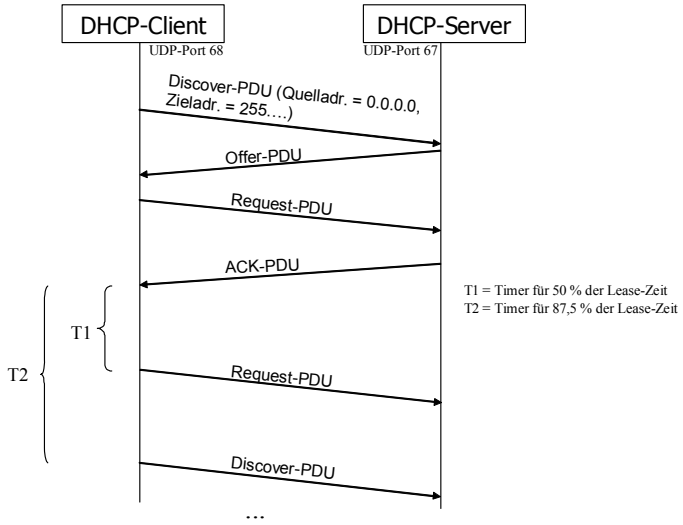


Abbildung 4-53: Ablauf einer DHCP-Kommunikation

Damit der DHCP-Mechanismus schon beim Booten funktioniert, muss das Betriebssystem gewisse Vorkehrungen treffen. Trickreich ist auch, wie der DHCP-Server den DHCP-Client in der Offer-PDU adressiert. Er hat ja noch keine IP-Adresse des Clients, muss aber ein IP-Datagramm senden, in dem die Offer-PDU übertragen wird. Die Lösung hierfür ist, dass der Server direkt an die MAC-Adresse sendet. Da manche TCP-/IP-Stacks diese Pakete jedoch nicht annehmen, können die Offer-/ACK-/NAK-PDUs auch als Multicast gesendet werden. Der Client muss hierzu im Feld „Flags“ das Multicast-Bit gesetzt haben. Die Kommunikation zwischen DHCP-Client und -Server wird über UDP abgewickelt. Für den DHCP-Client ist für die Kommunikation der UDP-Port 68 und für den DHCP-Server der UDP-Port 67 reserviert.

Nachteilig an dem Verfahren war ursprünglich, dass in jedem Subnetz ein DHCP-Server platziert sein musste. Dieses Problem ist jedoch gelöst, da heutige IP-Router auch als sog. DHCP-Relay-Agents auftreten, die DHCP-Nachrichten an DHCP-Server in anderen Netzwerken weiterleiten.

Die weiteren Nachrichtentypen haben folgende Bedeutung:

- *Decline*: Nach dem Empfang der ACK-PDU überprüft der Client noch einmal, ob die ihm zugewiesene Adresse wirklich frei ist. Dazu kann er z.B. eine ARP-Anfrage senden. Wenn die Adresse bereits vergeben ist, muss der Client eine Decline-PDU an den Server senden und die DHCP-Anfrage erneut beginnen.
- *Release*: Mit einer Release-PDU gibt der Client die ihm zugewiesene IP-Adresse wieder frei. Der Server kann über diese danach wieder frei verfügen.
- *Inform*: Eine Inform-PDU wird vom Client an den Server gesendet, wenn der Client bereits eine IP-Adresse eingetragen hat (z.B. durch manuelle Konfiguration) und nur noch Konfigurationsdaten (z.B. Gateway, DNS-Server) erfragen will. In diesem Fall wieder ein DHCP-Request mit gesetzter Client-IP-Adresse gesendet.

Beispiel einer DHCP-Serverkonfiguration unter Linux. Unter Linux wird eine DHCP-Serverimplementierung als Dämonprozess bereitgestellt. Der Dämon hat den Namen *dhcpd* und wird über eine Konfigurationsdatei namens /etc/dhcpd.conf konfiguriert. Eine typische Konfiguration lässt sich beispielsweise wie folgt beschreiben:

```
default-lease-time 600; # 10 Minuten
max-lease-time 7200; # 2 Stunden

option domain-name "isys.com";
option domain-name-servers 192.168.1.1 192.168.1.2;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option subnet-mask 255.255.255.0;
subnet 192.168.1.0 netmask 255.255.255.0
{
        range 192.168.1.10 192.168.1.20;
        range 192.168.1.100 192.168.1.200;
}
host mandl
        hardware ethernet 00:00:45:12:EE:E4;
        fixed-address 192.168.1.21;
```

In den Optionsangaben sind die Parameter eingetragen, die an den DHCP-Client bei einem Request neben der IP-Adresse übertragen werden. In den Ranges sind die IP-Adressbereiche angegeben, die der DHCP-Server vergeben darf. Der Host

mit dem Hostnamen „mandl“ und der angegebenen Ethernet-Adresse ist im Beispiel statisch einer festen IP-Adresse zugewiesen.

Beispiel für DHCP-Einsatz. Wie in Abbildung 4-54 zu sehen ist, melden sich die einzelnen Hosts (meist Arbeitsplätze, seltener Server) beim DHCP-Server über Broadcast mit einer DHCP-Anfrage.

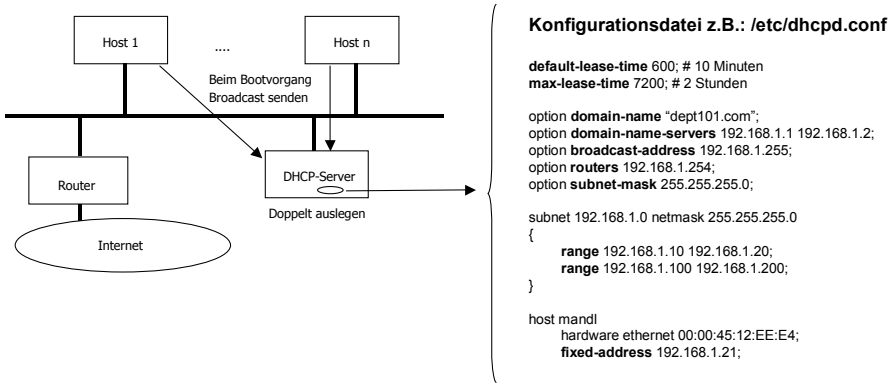


Abbildung 4-54: Beispiel für den DHCP-Einsatz

Der DHCP-Server weiß aufgrund seiner Konfiguration, wie er die Anfragen zu beantworten hat. Im Beispiel ist für die dynamische Vergabe von IP-Adressen eine Range von 192.168.1.10 bis 192.168.1.20 und eine weitere Range von 192.168.1.100 bis 192.168.1.200 festgelegt. Weiterhin ist ein Host (Hostname mandl) mit einer festen IP-Adresse belegt.

4.4 Das neue Internet-Protokoll IPv6

4.4.1 Ziele der IPv6-Entwicklung

Hauptziel der Entwicklung eines neuen IP-Protokolls (IPv6, oder IPnG) war es, die Adressproblematik umfassend und langfristig zu lösen. Zukunftsszenarien sind u.a., dass jeder Fernseher bald ein Internet-Knoten (Video-on-Demand) sein soll und dass es Millionen von drahtlosen Systemen im Internet gibt, die alle adressierbar sind. Mit IPv4-Adressen ist dies nicht möglich, IPv6 soll das Adressproblem aber zumindest für lange Zeit lösen. Weitere Subziele der IPv6-Entwicklung sind:

- Die Vereinfachung des Protokolls zur schnelleren Bearbeitung von Paketen in Routern.
- Der Umfang der Routing-Tabellen soll reduziert werden.
- Anwendungstypen wie Multimedia-Anwendungen (Echtzeitanwendungen) sollen unterstützt werden.
- Flussmarken sollen unterstützt werden.

- Eine höhere Sicherheit in der Schicht 3 (Datenschutz, Authentifikation) soll erreicht werden.
- Das Multicasting soll besser unterstützt werden.
- Zur Unterstützung mobiler Hosts soll die Möglichkeit geschaffen werden, dass Hosts ihr Heimatnetz verlassen können.
- IPv6 soll die Möglichkeit der Weiterentwicklung des Protokolls vereinfachen.

Die Koexistenz mit IPv4 ist für eine Migration unbedingt erforderlich und wird daher angestrebt, da es unmöglich ist, das ganze Internet auf einen Zug umzustellen. Das Protokoll ist fertig konzipiert (RFC 2460) und derzeit nur partiell im Einsatz, also (noch) nicht besonders verbreitet. Wir wollen in diesem Abschnitt nur die wichtigsten Merkmale kurz diskutieren.

4.4.2 IPv6-Adressstruktur und -Adressraum

Auch in IPv6 kennzeichnet eine Adresse nicht einen Host, sondern ein Interface (physikalischer Port), also den Netzwerkanschluss des Hosts. Es wird für jedes Interface eine IP-Adresse vergeben.

IPv6 unterstützt 16-Byte-Adressen (128 Bit) mit einer neuen Notation. Der grundlegende Aufbau einer IPv6-Adresse mit 128 Bit ist in Abbildung 4-55 sowohl unstrukturiert als auch zunächst grob strukturiert in einen Netzwerk- und einen Interface-Anteil dargestellt.

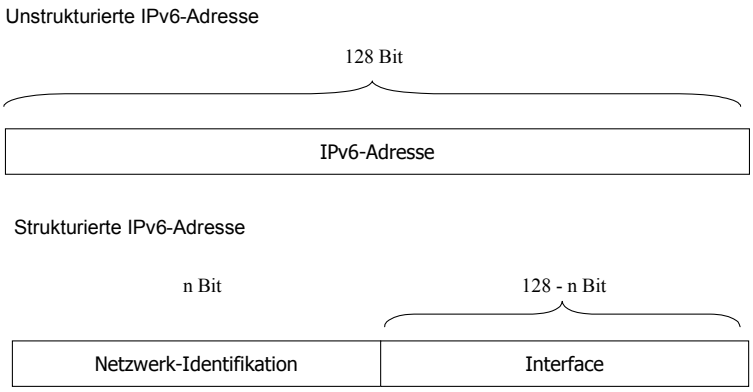


Abbildung 4-55: Grundlegender Aufbau einer IPv6-Adresse

Anmerkung: Eine anschauliche Analogie zur Anzahl der möglichen IPv6-Adressen (2^{128}) ist die folgende: Wenn die ganze Welt mit Computern bedeckt wäre, könnte man mit IPv6 ca. $7*10^{23}$ IP-Adressen pro m^2 abdecken. Dies ergibt sich wie folgt: Der Äquatorradius der Erde r ist $6{,}378*10^6$ m. Die Erdoberfläche S ist $4 * r^2 * Pi$. Daraus folgt $S = 5{,}112*10^{14}$ Quadratmeter (annähernd, da die Erde nicht wirklich eine Kugel ist). Die theoretisch mögliche Anzahl an Adressen in IPv6 ist 2^{128} also $3{,}40*10^{38}$. Dies ergibt schließlich $6{,}65*10^{23}$ IPv6 Adressen pro Quadratmeter oder ca.

665 Billiarden IPv6 Adressen pro Quadratmillimeter. Im Vergleich zu IPv4 ergibt sich eine Vergrößerung des Adressraums um 2^{96}.

Es gibt verschiedene Klassen von Adressen:

- *Unicast-Adressen*: Dies ist der traditionelle Adresstyp zum Adressieren des Netzanschlusses eines Hosts oder Routers.
- *Multicast-Adressen*: Kennzeichnen eine Reihe von Endsystemen, also eine Gruppe von Interfaces, z.B. für Gruppenkommunikation. Ein Paket wird an alle Netzanschlüsse zugestellt, die einer Multicast-Adresse zugeordnet sind. Die Aufgabe der Broadcast-Adressen (IPv4) wird bei IPv6 von Multicast-Adressen übernommen
- *Anycast-Adressen*: Kennzeichnen eine Gruppe von Netzwerkanschlüssen, die meist einer funktionalen Gruppe angehören (z.B.: alle Router). Ein mit einer Anycast-Adresse versehenes Datagramm wird einem beliebigen Rechner aus der Menge zugestellt. Es wird versucht, den „nächstliegenden" Rechner aus der Gruppe zu erreichen.

Die Adressen sind von der Darstellung her in acht Gruppen zu je vier Hex-Zahlen (16 Bit) abgetrennt durch Doppelpunkte aufgeteilt.

Beispiel:

„8000:0000:0000:0000:0123:5555:89AB:CDEF"

Führende Nullen können in jeder Gruppe weggelassen werden (z.B. 0 statt 0000, 3 statt 0003). Gruppen mit lauter Nullen können durch zwei Doppelpunkte „::" ersetzt werden. Das Symbol „::" ist nur an einer Stelle der Adresse erlaubt, sonst geht die Eindeutigkeit verloren.

Beispiele:

„8A00:0000:0123:0005:89AB:CDEF:0000:0000"

wird zu

„8A00:0:123:5:89AB:CDEF::"

und

„::65:78C1:9A:6008"

entspricht

„0000:0000:0000:0000:0065:78C1:009A:6008"

IPv4-Adressen können ebenfalls auf die neue Schreibweise abgebildet werden. Die ersten 12 Byte werden wie eben beschrieben in sechs Gruppen zu je vier Hex-Zahlen abgetrennt durch Doppelpunkte aufgeteilt. Für die verbleibenden 4 Byte wird, zur Darstellung der IPv4-Adresse, die klassische Schreibweise verwendet. Alternativ kann eine IPv4-Adresse als normale IPv6-Adresse geschrieben werden. Hierzu müssen jeweils zwei Byte der IPv4-Adresse zu einer Hex-Zahlen-Gruppe umgerechnet werden. Die IPv4-Adressen sind mit 0xFFFF gekennzeichnet.

Beispiel:

„::FFFF:192.168.0.1

entspricht

::FFFF:C0A8:1"

Eine klassenweise Aufteilung in A-, B-, C-Klassen usw. gibt es bei IPv6 nicht. Die 128-Bit-Adresse wird aber üblicherweise in eine Netzidentifikation und in eine Identifikation für den Host strukturiert. Meist werden die ersten n Bit (üblicherweise die ersten 64 Bit) als Netzidentifikation verwendet, und der Rest wird der Hostid zugeordnet. Die Präfixlänge wird in CIDR-Notation an die Adresse gehängt.

Beispiel:

„2001::0123:5555:89AB:CDEF/64"

Auch in einer URL kann eine IPv6-Adresse notiert werden, wobei eckige Klammern als Trennzeichen verwendet werden. Dies ist notwendig, da das Zeichen „:" auch in der URL vorkommt.

Beispiel:

„http://[2001::0123:5555:89AB:CDEF/64]:8080"

Im Anschluss werden einige IPv6-Adressformate diskutiert. Es soll aber erwähnt werden, dass die Diskussion um die Adressstrukturierung noch nicht abgeschlossen ist. Dies erkennt man auch an der Fülle an neuen RFCs, die es zu diesem Thema gibt. Die folgende Beschreibung bezieht sich auf den RFC 4291 von Februar 2006.

IPv6-Adress-Sonderformen. Es gibt einige IPv6-Adressen mit Sonderformen. Diese Sonderformen werden in den ersten Bit der Adresse kodiert. Diese Kodierungen können von den Routern ausgewertet werden.

- Das Präfix ::/128 (alle 128 Bit stehen auf 0) entspricht der IPv4-Adresse 0.0.0.0 und steht für die undefinierte Adresse. Eine andere Darstellungsform ist 0:0:0:0:0:0:0:0. Diese Adresse kann beim Bootvorgang eines Hosts verwendet werden, um eine Adresskonfiguration durchzuführen, und darf nicht als Empfängeradresse benutzt werden.
- Das Präfix ::1/128 gibt die Loopback-Adresse (in IPv4 z.B. 127.0.0.1) an. Eine andere Darstellungsform ist 0:0:0:0:0:0:0:1. Die Loopback-Adresse wird wie bei IPv4 verwendet, um Pakete an sich selbst zu senden. IPv6-Pakete mit der Loopback-Adresse als Zieladresse werden nicht an die angeschlossenen Netze weitergeleitet. Sie dürfen keinem Interface zugewiesen werden.
- Das Präfix FF00::/8 (die ersten acht Bit stehen auf 1) weist auf eine Multicast-Adresse hin.
- Das Präfix FF80::/10 weist auf eine Link-Local-Adresse hin (siehe weiter unten).

Alle Adressen, die nicht mit einem der eben beschriebenen Präfixe beginnen, werden als globale Unicast-Adressen behandelt. Anycast-Adressen werden aus dem Adressbereich der Unicast-Adressen erstellt. Durch das Zuweisen einer Unicast-Adresse zu mehreren Interfaces wird aus ihr eine Anycast-Adresse. Im Folgenden werden die Adressformen näher beschrieben.

Global Unicast-Adressen. Global Unicast-Adressen dienen dazu, einen Host (Knoten) im Internet global eindeutig zu identifizieren. Eine Unicast-Adresse hat folgenden Aufbau (siehe Abbildung 4-56).

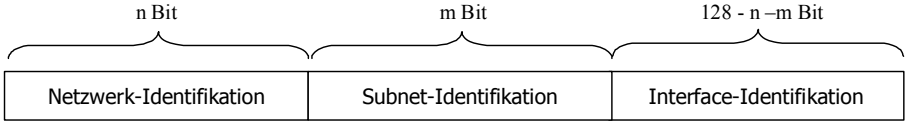


Abbildung 4-56: Aufbau einer globalen Unicast-Adresse

Die Netzwerkidentifikation (das sog. *Global Routing Prefix)* kann dazu verwendet werden, den Adressbereich einer Organisation, z.B. eines Internet Providers, oder eines Unternehmens zu identifizieren. Diese Information nutzen Router zur Optimierung. Das Präfix wird üblicherweise nochmals hierarchisch strukturiert. Adressbereiche einer gemeinsamen Route werden in den Routing-Tabellen der Internet-Router zusammengefasst. Auch die *Subnet-Identifikation* kann für individuelle Adressierungshierarchien nochmals, z.B. in geographische Adressbereiche untergliedert werden. Die *Interface-Identifikation* dient der Adressierung eines Hosts innerhalb eines Subnetzes. Sie wird manuell zugewiesen oder mit Hilfe eines speziellen Verfahrens automatisch (Autokonfiguration) generiert[31]. Die Interface-Identifikation kann auch zufällig erzeugt werden, um Sicherheitsprobleme zu vermeiden (RFC 4941). Ein Beispiel einer Untergliederung der einzelnen Adressbestandteile ist in Abbildung 4-57 dargestellt. Diese Unicast-Adressen sind für Provider geeignet und bestehen aus einem öffentlichen und einem privaten Anteil. Die einzelnen Adressbestandteile haben dabei folgende Bedeutung:

– *Registry-Id* ist die internationale Registrierungs-Id; ICANN[32] hat die Registry-Id 0b10000, RIPE[33] hat 0b01000 und APNIC[34] hat 0b00100 usw.

[31] Dieses Verfahren wird als EUI-64-Verfahren bezeichnet. Es beschreibt die automatische Erweiterung der 48-Bit-langen MAC-Adresse auf einen 64-Bit-langen Interface-Identifier. Die 64-Bit-basierte EUI-64-Adresse wird vom IEEE festgelegt. EUI-64-Adressen werden entweder einem Netzwerkadapter zugewiesen oder von IEEE-802-Adressen abgeleitet.

[32] ICANN ist die Abkürzung für Internet Corporation for Assigned Names and Numbers.

[33] RIPE steht für Reesau IP Europeen Regional Internet Registry for Europe, der europäischen Registrierungsorganisation des Internets.

[34] APNIC steht für Asia Pacific Internet Information Center.

- *Nat. Registry-Id* ist die Identifikation einer nationalen Organisation.
- *Provider-Id* identifiziert den Anbieter der Internet-Dienste, den ISP. Die Provider-Id kann variabel lang sein. Ein großer, weltweit agierender Provider erhält z.B. eine kleine Id (p ist klein). Damit bleiben mehr Bit für die Sub-Provider dieses Anbieters übrig.
- *Subscriber-Id* kennzeichnet einen privaten Netzbetreiber und kann mit der Netzwerk-Id in IPv4 verglichen werden.
- *Intra-Subscriber-Id* dient der privaten Nutzung und kann nochmals zur Strukturierung innerhalb eines privaten Netzes verwendet werden. Hierzu wird der Adressanteil in eine *Subnet-Id* und eine *Interface-Id* zerlegt, was bei IPv4 ebenfalls der *Subnet-Id* und der *Host-Id* entspricht.

Eine international agierende Registrierungsorganisation kann also mehrere nationale Organisationen mit Adressen versorgen und diese wiederum jeweils mehrere nationale Organisationen. Die öffentlichen Adressanteile ermöglichen eine globale Lokalisierung auf der Erde, womit ein weltweites hierarchisches Routing ermöglicht wird.

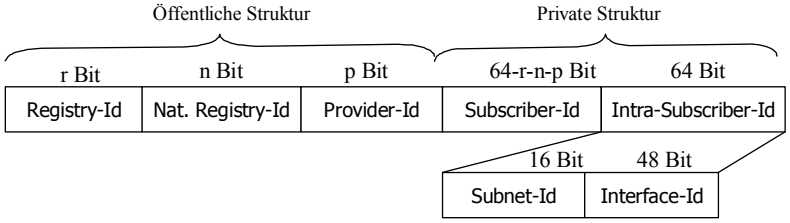


Abbildung 4-57: Beispiel einer Struktur einer Unicast-Adresse

Globale Unicast-Adressen dienen also im Wesentlichen der Hierarchisierung. Damit kann eine weltweite Hierarchie an Adressen aufgebaut werden, die letztendlich im Vergleich zu IPv4 zur Reduzierung der Einträge in den Routing-Tabellen führt.

IPv4-Mapped Adressen. Zur Abbildung von IPv4 Adressen gibt es einen speziellen Adressbereich der globalen Unicast-Adressen. IPv4-Mapped-Adressen benutzen immer den Präfix ::FFFF/96. Die verbleibenden 32 Bit werden zu Kodierung der IPv4 Adresse verwendet, die natürlich global eindeutig sein muss.

Link-lokale Adressen. Jede Link-lokale Adresse besteht aus dem Präfix FE80::/10 und dem 64 Bit langen Interface-Identifier. Link-lokale Adressen werden von jedem Host beim Systemstart erzeugt und auf das lokale Subnetz beschränkt. Sie sind nur für den Einsatz innerhalb des eigenen Netzwerkes bestimmt. Eine Link-lokale Adresse darf daher von Routern nicht in andere Netze weitergeleitet werden. Beispielhafte Einsatzbereiche sind die automatische Adresskonfiguration oder das *Neighbor Discovery*.

Anycast-Adressen. Eine Unicast-Adresse, die mehr als einem Interface zugeordnet wird, bezeichnet man als Anycast-Adresse. Alle Knoten die einer Anycast-Gruppe hinzugefügt werden, müssen explizit dafür konfiguriert werden. Beim Routing wird versucht den nächsten Knoten der Anycast-Gruppe zu erreichen. Eine Anycast-Adresse besitzt, analog zu den anderen Adressen, einen Präfix zur Identifikation des Netzwerks. Innerhalb des durch das Präfix identifizierten Netzwerkes, müssen sich alle Knoten befinden, die mit einer Anycast-Adresse adressiert werden. Innerhalb des Netzwerkes einer Anycast-Gruppe muss die vollständige Anycast-Adresse als gesonderter Eintrag in den Routing-Tabellen geführt werden. Soll eine Anycast-Gruppe gebildet werden, die über das gesamte Internet verteilt ist, muss ein Präfix mit der Länge 0 gewählt werden. Für solche globalen Anycast-Adressen müssen die entsprechenden Einträge in den Routing-Tabellen aller Router verwaltet werden. Die Verwendung dieser globalen Anycast-Adressen wird daher wohl vollständig untersagt oder stark eingeschränkt.

Eine beispielhafte Anycast-Adresse ist die vordefinierte Adresse aller Router eines Teilnetzes. Ergänzt man die Identifikation eines Netzwerks mit einer „leeren" (alle Bit sind auf 0 gesetzt) Identifikation für den Host, erhält man die Anycast-Adresse der lokalen Router. Ein Paket das an diese Anycast-Adresse versendet wird, erreicht in der Regel direkt einen der nächsten Router. Es ist somit möglich, nur mit dem Wissen der Netzwerkidentifikation, einen der Router im lokalen Netzwerk über die Anycast-Adresse anzusprechen.

Multicast-Adressen. Multicast-Nachrichten werden z.B. für die Anwendungen *Neighbor Discovery*, *DHCP* und für die Unterstützung des Routings eingesetzt. Eine Multicast-Adresse darf wie bei IPv4 nicht als Absenderadresse benutzt werden. Der Aufbau sieht folgendermaßen aus (siehe Abbildung 4-58):

IPv6-Multicast-Adressen beginnen mit dem Format-Präfix 0xFF00/8 (binär 0b11111111). Anschließend folgt ein Feld namens *Flag*, von denen das letzte Bit angibt, ob es sich um eine temporär vergebene oder um eine sog. „well-known", also eine ständig zugeordnete Multicast-Adresse handelt. Die ersten drei Bit sind für zukünftige Verwendungen reserviert und müssen derzeit auf Null gesetzt sein.

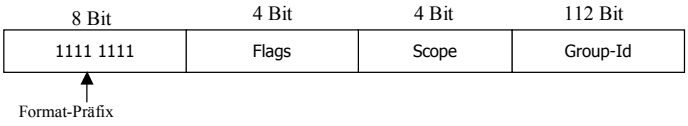


Abbildung 4-58: IPv6-Multicast-Adressen

Im Feld *Scope* ist in einem Halbbyte der Gültigkeitsbereich festgelegt (Werte: 0x0 – 0xF). Der Wert 0xE bedeutet z.B., dass es sich um eine Multicast-Adresse handelt, die alle Rechner adressiert. Dies entspricht der IPv4-Broadcastadresse Das Feld *Scope* legt somit fest, wie weit sich ein Multicast-Paket ausbreiten darf. Mögliche Werte sind u.a.:

- Der Wert 0x1 kennzeichnet eine Multicast-Adresse, die sich nur auf das Interface eines Rechners bezieht. Die Pakete, die an diese Adresse gesendet werden, sind knotenlokal und verlassen den Knoten nie. So gekennzeichnete Adressen sind also vergleichbar mit den IPv4-Loopback-Adressen.
- Der Wert 0x2 gibt an, dass es sich um eine Link-lokale Adresse handelt. Die Pakete, die an diese Adresse gesendet werden, werden von Routern grundsätzlich nie weitergeleitet und können deshalb das Subnetz nicht verlassen.

Im Feld *Group-Id* ist schließlich die Identifikation der Multicast-Gruppe enthalten. Im RFC 3306 wird die Strukturierung dieses Feldes weiter diskutiert.

4.4.3 Der IPv6-Header

In Abbildung 4-59 ist der IPv6-Header dargestellt. Er ist nun im Vergleich zu IPv4 etwas einfacher strukturiert. Es gibt z.B. keine Optionen mehr. Dafür gibt es in IPv6 ausgelagerte Erweiterungs-Header, die bei Bedarf verwendet werden.

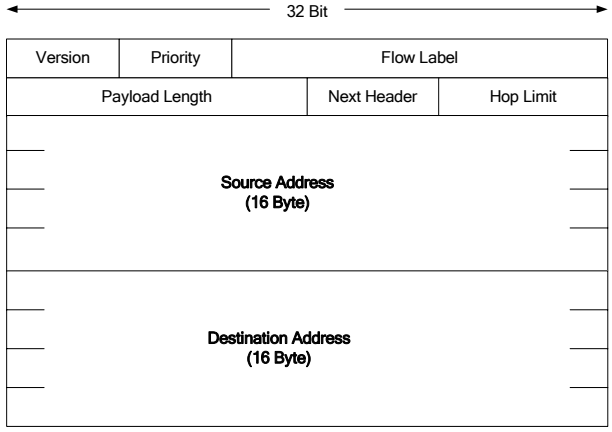


Abbildung 4-59: IPv6-Header

Folgende Felder sind im IPv6-Header enthalten:

- *Version:* Versionsnummer des Internet-Protokolls (6)
- *Priorität*: Der Wert der Priorität dient als Information für Router und ist interessant bei Überlastsituationen. Folgende Werte sind möglich:
  - 0–7 = Verkehrsarten mit Staukontrolle
  - 0 = nicht charakterisierter Verkehr
  - 4 = stoßartiger Verkehr (ftp, NFS)
  - 6 = interaktiver Verkehr (telnet)
  - 8–15 = Verkehrsarten ohne Staukontrolle (8 z.B. für Videoanwendungen)

- *Flussmarke*: Identifikation des „Flusses“, falls ungleich 0. Flussmarken werden im Quellknoten in die IPv6-PDU eingetragen. Die Quelladresse kennzeichnet in Verbindung mit der Zieladresse und der Flussmarke einen Fluss.
- *Payload Length***:** Nutzdatenlänge ohne die 40 Byte des IPv6-Headers.
- *Next Header*: Verweis auf ersten Erweiterungs-Header. Der letzte Erweiterungs-Header verweist auf den Protokolltyp der nächst höheren Schicht (siehe IPv4-Feld *Protokoll*).
- *Hop Limit*: Verbleibende Lebenszeit des Pakets in Hops. Jeder Router zählt das Hop-Limit bei Ankunft eines IP-Pakets um 1 herunter. Dies entspricht dem *TTL*-Feld in IPv4. Der Name entspricht jetzt der eigentlichen Nutzung im Internet.
- *Source und Destination Adresse*: IPv6-Adressen der Quelle und des Ziels.

Tabelle 4-7: IPv6-Erweiterungs-Header

Erweiterungs-Header	Beschreibung
Optionen für Teilstrecken (Hop-by-Hop)	Verschiedene Informationen für Router
Routing	Definition einer vollen oder teilweisen Route
Fragmentierung	Verwaltung von Datengrammfragmenten
Authentifikation	Echtheitsüberprüfung des Senders
Verschlüsselte Sicherheitsdaten	Informationen über verschlüsselten Inhalt
Optionen für Ziele	Zusätzliche Informationen für das Ziel

In Tabelle 4-7 sind die in IPv6 definierten Erweiterungs-Header zusammengefasst. Header und Erweiterungs-Header sind miteinander verkettet**,** wobei jeder Typ max. einmal vorkommen kann. Die Erweiterungen werden nicht in den Routern bearbeitet, sondern nur in den Endsystemen. Eine Ausnahme hierzu ist der Routing-Erweiterungs-Header. Die Reihenfolge der Erweiterungs-Header in einem IP-Paket ist genau festgelegt.

Ein Beispiel für eine Kette aus einem IPv6-Header mit einem Erweiterungs-Header und einer anschließenden TCP-PDU ist in Abbildung 4-60 dargestellt.

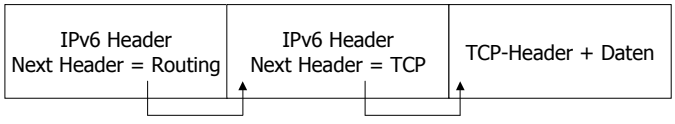


Abbildung 4-60: Verkettung der IPv6-Erweiterungs-Header

Die Erweiterungs-Header für das Routing und für die Fragmentierung sollen im Folgenden beispielhaft diskutiert werden.

Routing-Header. Der Routing-Header dient dem Quellhost zur Festlegung (Vorabdefinition) des Weges (Pfads) bis zum Ziel. Es wird sowohl striktes, als auch loses Routing, d.h ein voller Pfad oder ausgewählte Router zugelassen.

Der Routing-Header enthält folgende Felder:

- *Next Header:* Verweis auf den nächsten Erweiterungsheader in der Kette.
- *Anzahl Adressen:* Anzahl der folgenden IP-Adressen, die besucht werden müssen.
- *Next Address*: Index innerhalb der folgenden IP-Adressliste auf die nächste zu besuchende Adresse.
- *Bitmuster:* Bitmap, in der für jede IP-Adresse ein Bit vorhanden ist. Bei einer Bitfolge mit Werten von 1 müssen die korrespondierenden Adressen unmittelbar aufeinander besucht werden. Ansonsten können auch andere Router dazwischen liegen.
- *Adressliste:* Bis zu 24 IP-Adressen, die durchlaufen werden müssen (Abb. 4-61).

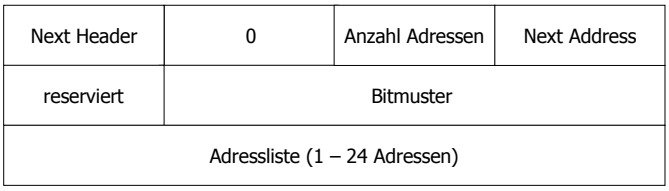


Abbildung 4-61: IPv6-Routing-Header

Fragmentierungs-Header. Der Fragmentierungs-Header wird verwendet, um größere Dateneinheiten zu senden, also wenn die IP-PDU-Länge größer als die MTU des Pfades ist.

Nach Abbildung 4-62 sind im Fragmentierungs-Header folgende Felder enthalten:

- *Next Header:* Verweis auf den nächsten Erweiterungsheader in der Kette.
- *Fragment Offset*: Position der Nutzdaten relativ zum Beginn der PDU (Ursprungs-Dateneinheit).
- *Identifikation*: Identifikation (Id) der PDU.
- *M:* Dies ist das sog. More-Flag, M = 1 bedeutet, dass weitere Fragmente folgen.

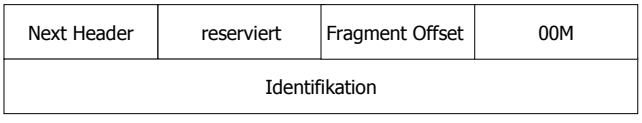


Abbildung 4-62: IPv6-Fragmentierungs-Header

Im Gegensatz zu IPv4 erfolgt die Fragmentierung bei IPv6 nur im Quellknoten, die Router segmentieren dagegen nicht. Dies hat eine geringere Routerbelastung zur Folge, die natürlich auf Kosten der Hosts geht.

4.4.4 Flussmarken

Ein interessanter Aspekt in IPv6 sind sog. Flussmarken. Ziel von Flussmarken ist der Aufbau von Pseudoverbindungen zwischen Quelle und Ziel mit definierbaren QS-Merkmalen wie Verzögerung und Bandbreite. Ressourcen können reserviert werden, was für Datenströme von Echtzeitanwendungen sehr hilfreich ist. Ein „Fluss“ wird durch die Quell- und Zieladresse sowie durch eine Flussnummer eindeutig identifiziert.

Die Flexibilität von Datagramm-Netzen soll bei diesem Mechanismus mit den Vorteilen der virtuellen Verbindungen kombiniert werden. Router führen für die Verarbeitung von „Flüssen“ eine Sonderbehandlung durch.

Man sollte allerdings erwähnen, dass das Thema noch stark in der Experimentierphase ist, weshalb hier auch nicht weiter darauf eingegangen wird.

4.4.5 Neighbor Discovery

Das Neighbor Discovery Protocol (ND-Protokoll, RFC 2461) dient bei IPv6 zur Unterstützung der automatischen Konfiguration von Endsystemen. Im ND-Protokoll spricht man von *Links*, wenn man einen Netzwerkanschluss meint, und von Link-Adresse, wenn man von der Adresse des Netzwerkanschlusses spricht. Dabei spielt es keine Rolle, um welchen physikalischen Netzwerkanschluss es sich handelt. Das ND-Protokoll ist sowohl für LANs als auch für verbindungsorientierte Netze (ISDN, ATM,...) konzipiert. In einem Ethernet-LAN ist die Link-Adresse eine MAC-Adresse, in einem ISDN-Netzwerk ist es die ISDN-Rufnummer und in ATM-Netzwerken eine ATM-Adresse.

Das ND-Protokoll löst einige Probleme, die aus der IPv4-Konfiguration bekannt sind und dort in eigenen Protokollen behandelt werden. Hierzu gehören u.a.:

- Das Auffinden von Routern im gleichen Link (Subnetz). Dies wird als *Router Discovery* bezeichnet.
- Die dynamische Zuordnung von Konfigurationsparametern wie der maximalen MTU und dem Hop-Limit an IPv6-Endsysteme. Hier spricht man von *Parameter Discovery*.
- Die automatische IP-Adress-Konfiguration für Interfaces zur Laufzeit (Neighbor Solicitation). Hier ist auch die Abbildung der bisherigen 48-Bit-MAC-Adressen auf die EUI-64-Bit-Adressen von Bedeutung.
- Die dynamische Adress-Auflösung für Layer-2-Adressen, wie es heute im ARP-Protokoll abgewickelt wird.
- Die optimale MTU wird in IPv6 vom sendenden Endsystem eingestellt und muss zur Laufzeit gefunden und optimiert werden. Die Suche nach der op-

timalen MTU zwischen Sender und Empfänger wird als Path MTU Discovery bezeichnet.

Beispielhaft sollen zwei Aspekte betrachtet werden: Die Ermittlung von Konfigurationsparametern vom Router und das Auffinden von Routern im Netz. Der Literatur können weitere interessante Funktionen wie *Neighbor-Solicitation* entnommen werden (Wiese 2002).

Router-Discovery. Wenn ein Endsystem seinen nächsten Router sucht, sendet es eine *Router-Solicitation*-Nachricht über Multicast an die Adresse „FF02::2". Damit werden alle Router angesprochen. Die Router antworten mit einer *Router-Advertisement*-Nachricht. Damit unterstützt das ND-Protokoll das Auffinden des verantwortlichen Routers zur Laufzeit, und man muss diese Information nicht im Endsystem manuell konfigurieren oder über DHCP ermitteln. In einem IPv6-Subnetz können im Gegensatz zu IPv4 mehrere Router aktiv sein. Das ND-Protokoll nutzt zur Abwicklung seiner Aufgaben einige ICMPv6-Nachrichten. Das Ermitteln des nächsten Routers im Subnetz wird z.B. über eine ICMPv6-Nachricht (Router Discovery) durchgeführt.

Parameter Discovery. Ein Host eines Subnetzes kann zum Startzeitpunkt eine Nachricht ins Subnetz senden, die als *Router-Solicitation*-Nachricht[35] bezeichnet wird. Die Nachricht wird an die feste Multicast-Adresse „FF02::1"[36] (Scope = „link-lokal", also im Subnetz) gesendet. Ein Router antwortet mit einer *Router-Advertisement*-Nachricht an die Link-Adresse des Endsystems. Folgende Parameter kann eine *Router-Advertisement*-Nachricht u.a. übertragen:

- *Max-Hop-Limit*: Dies ist der Wert „Hop-Limit", der in die IPv6-PDUs eingetragen wird.
- *Retransmission-Timer*: Zeit in Millisekunden, die seit dem Absenden der letzten *Solicitation*-Nachricht abgelaufen ist.
- *Managed-Address-Configuration-Flag*: Über diese Information wird angezeigt, dass der Router eine *stateful Configuration* (siehe unten) mit DHCPv6 unterstützt.
- Über ein *Optionsfeld* wird z.B. vom Router auch die *MTU-Size* übermittelt. Mit dieser Information kann das Endsystem zur Vermeidung von Fragmentierung beitragen, was ja in IPv6 Aufgabe der Endsysteme ist.

In IPv4 wird diese Aufgabe vom DHCP-Server prinzipiell mit übernommen, sofern nicht eine statische Einstellung der Parameter im Endsystem vorgenommen wird. Jeder IPv6-Router sendet zudem periodisch seine Parameter im Subnetz über eine *Router-Advertisement*-Nachricht auch über die Multicast-Adresse „FF02::2".

---

[35] Solicitation = Bewerbung, Ansuchen.

[36] Diese Multicast-Adresse wird auch *Solicited-Multicast-Adresse* genannt.

Wenn in IPv6 beim Parameter Discovery kein Router antwortet, kann das Protokoll DHCPv6 verwendet werden.

4.4.6 Automatische Adresskonfiguration

Die automatische Konfiguration von Endsystemen, d.h. die Versorgung der Endsysteme mit IPv6-Adressen, wird in IPv6 über zwei Varianten unterstützt: *Stateless* und *stateful Autoconfiguration*. Beide Möglichkeiten sollen im Folgenden kurz erläutert werden.

Stateless Autoconfiguration. Bei dieser Variante suchen sich die Endsysteme eines IPv6-Subnetzes automatisch ohne Unterstützung durch einen dedizierten DHCP-Server ihre IP-Adressen. Das Verfahren funktioniert nur innerhalb von IPv6-Subnetzen. Die IPv6-Adresse eines Subnetzes setzt sich aus zwei Teilen, einem Präfix und einem sog. Link-Token zusammen. Das Link-Token repräsentiert die einem Endsystem bereits zum Startzeitpunkt bekannte MAC-Adresse (im Falle eines LAN). Nur das Präfix muss ermittelt werden, der Rest ist dem Endsystem bekannt.

Für diese Konfigurationsart wird ICMPv6 als Protokoll verwendet. ICMPv6-Pakete werden hierfür wiederum in IPv6-Pakete eingebettet, die als Hop-Limit den Wert 255 enthalten. Eine Weiterleitung der Pakete durch einen Router ist damit nicht zugelassen. Der Ablauf ist grob in Abbildung 4-63 skizziert.

Wenn ein Endsystem eine IPv6-Adresse benötigt, sendet es zunächst in einem ersten Schritt die eigene Link-Adresse, also die MAC-Adresse in einer ICMPv6-Nachricht mit dem Typ 135 über die sog. *Solicited-Multicast-Adresse* „FF02::1“. Diese Nachricht wird als *Neighbor-Solicitation* bezeichnet. Dabei werden alle Endsysteme und Router des Subnetzes angesprochen. Wenn nun ein Rechner im Subnetz die gesendete MAC-Adresse (das Link-Token) ebenfalls verwendet, sendet dieser eine *Neighbor-Advertisement*-Nachricht direkt an die Link-Adresse des anfragenden Endsystems. Damit kann also festgestellt werden, ob ein Link-Token bereits verwendet wird. Ist dies der Fall, muss eine manuelle Konfliktauflösung erfolgen.

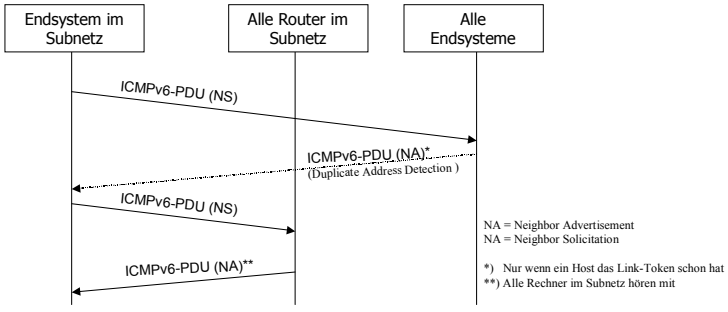


Abbildung 4-63: Ablauf bei stateless Autoconfiguration

Wenn das Link-Token eindeutig ist, sich also kein anderer Rechner beschwert, wird in einem zweiten Schritt eine weitere *Neighbor-Solicitation*-Nachricht an die spezielle *All-Routers-Multicast-Adresse* gesendet. Diese Nachricht ist für alle lokalen Router bestimmt. Mindestens ein Router antwortet an die *Solicited-Multicast-Adresse* mit einer *Neighbor-Advertisement*-Nachricht, in der er das Präfix zur Ergänzung der IPv6-Adresse an das Endsystem überträgt. Alle Rechner im Netz hören die Nachricht mit, und die IPv6-Adresse ist nun im Subnetz bekannt.

Durch dieses Verfahren wird auch ein vereinfachtes Renumbering von IP-Adressen möglich, da die Vergabe der Adressen zeitlich über eine Lease-Zeit begrenzt wird.

Stateless Autoconfiguration ist risikoreich, da ein Router auch ausfallen kann und dann kein Endsystem mehr in der Lage ist, im Subnetz zu kommunizieren. Eine Ergänzung der Variante um einen DHCPv6-Server ist daher durchaus sinnvoll.

Stateful Autoconfiguration. Die zustandsbehaftete Autoconfiguration wird über das Protokoll DHCPv6 (RFC 2462) unterstützt. Wie bei DHCP wird hierfür ein DHCP-Server benutzt, der die Adressen und sonstige Konfigurationsparameter (Router-Adresse, DNS-Servername,...) verwaltet.

Das Endsystem fungiert als DHCP-Client und kommuniziert über das DHCPv6-Protokoll mit einem DHCPv6-Server. Ist der Server nicht im Subnetz, kann er im DHCPv6-Modell über einen *DHCPv6-Agenten*, der auf einem Router liegt, indirekt angesprochen werden. Ein DHCPv6-Agent wird auch als *DHCPv6-Relay* bezeichnet.

Das DHCPv6-Protokoll ähnelt dem DHCP-Protokoll stark und wird daher an dieser Stelle nicht weiter ausgeführt.

4.4.7 Anpassung wichtiger Protokolle an IPv6

Bei Nutzung von IPv6 sind auch die meisten Steuer- und Routing-Protokolle entsprechend anzupassen. Im Folgenden sollen einige der neuen Protokolle kurz vorgestellt werden. Neben den hier genannten Anpassungen ist für IPv6 auch die TCP/UDP-Socket-Schnittstelle anzupassen, da hier IP-Adressen als Parameter für manche Funktionen verwendet werden, deren Länge sich ändert.

RIPng. RIPng (RIP next generation, RFC 2462) ist eine Anpassung des Routing-Protokolls RIP-2 an IPv6. Insbesondere wird hier die Adressierung auf die neue Adresslänge erweitert. RIPng bleibt aber ein Distance-Vector-Protokoll. Der Austausch der Routing-Informationen und auch der maximale Hop-Count von 15 ändern sich nicht. Zur Vermeidung des Count-to-Infinity-Problems werden ebenfalls die bei RIP-1 und RIP-2 vorgeschlagenenen Methoden verwendet.

Die wesentlichen Veränderungen sollen kurz dargestellt werden:

- Die Präfixlänge der Subnetzmaske und nicht mehr die Subnetzmaske selbst wird in den Routing-Informationen übermittelt. Damit ist es auch möglich,

dass RIPng in Netzwerken eingesetzt wird, in denen mehrere Präfixlängen verwendet werden.
- Durch Angabe eines „nächsten Hops“ kann der nächste Router für ein Netzwerkziel direkt angegeben werden. Im Gegensatz zu RIP-2 dient die Angabe zur Adressierung eines nächsten Routers und nicht eines Hosts.

Der Aufbau einer RIPng-PDU ist in Abbildung 4-64 skizziert. Der Aufbau des Headers entspricht dem von RIP-2. Die Tabelle mit Routing-Informationen wird als *RTE*-Tabelle (*Routing Table Entry*) bezeichnet. Die Länge der RTE-Tabelle ist nur durch die MTU beschränkt. In den RTEs gibt es einige Unterschiede zu RIP-2. Im Feld *IPv6-Präfix* wird das Netzwerkziel als IPv6-Adresse übermittelt. Die gültige Präfix-Länge wird im Feld *Präfix-Länge* in einem Byte übermittelt. Die restlichen Felder entsprechen den Feldern der RIP-2-PDU.

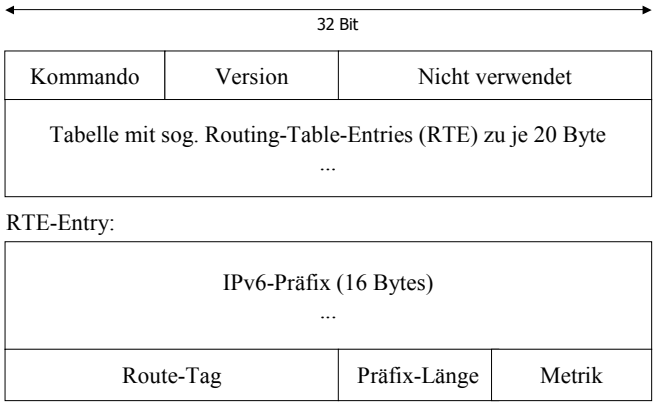


Abbildung 4-64: RIPng-PDU

Die Information für einen nächsten Hop wird im Feld IPv6-Präfix eingetragen und zusätzlich als solche im Feld *Metrik* mit dem Wert 0xFF gekennzeichnet.

OSPFng. OSPFng (OSPF next generation) ist eine Anpassung von OSPFv2 an IPv6. Prinzipiell gibt es allerdings kaum Änderungen. Die OSPFng-Pakete werden - wie bei OSPFv2 - direkt in IP-Paketen übertragen. Es werden auch die gleichen PDU-Typen unterstützt wie bei OSPFv2.

ICMPv6. Auch in IPv6-Umgebungen ist ein Hilfsprotokoll wie ICMP notwendig. ICMPv6 ist die Anpassung von ICMP an IPv6, allerdings sind einige Funktionen ergänzt worden. ICMPv6 wird wie das bisherige ICMP neben der Übertragung von Fehlermeldungen auch für Diagnoseinformationen verwendet. Darüber hinaus wird es zur Unterstützung der automatischen Adresskonfiguration eingesetzt.

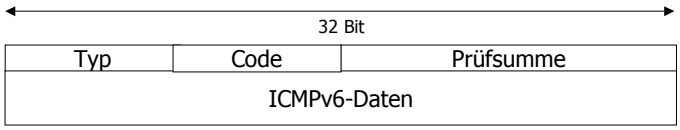


Abbildung 4-65: ICMPv6-PDU

Der ICMPv6-Nachricht (siehe Abbildung 4-65) wird in der IPv6-Nachricht, in der sie eingebettet ist, mit einer Next-Header-Information = 58 eingeleitet. Die Felder entsprechen der herkömmlichen ICMP-PDU. Die Typ-Angabe ist neu organisiert:

- Typ = 1 bedeutet z.B. „Destination Unreachable Message"
- Typ = 2 beinhaltet die Information „Packet too big Message"
- Typ = 128 beinhaltet die Information „Echo Request Message" und wird vom Kommando *ping* genutzt.
- usw.

Neu sind vor allem die Nachrichtentypen 130, 131 und 132 für die Gruppen-Membership-Nachrichten und die Typen 133 bis 173 zur Unterstützung der automatischen Adresskonfiguration. Mit einer Nachricht vom Typ 130 sendet z.B. ein Multicast-Knoten einem M-Router eine Anfrage, ob er an der Multicast-Gruppe teilnehmen darf.

ARPv6. Ist bei IPv6 im Prinzip nicht mehr notwendig, da die MAC-Adressen in die Interface-Ids eingetragen und von daher statisch festgelegt werden können.

DHCPv6. Dieses Protokoll wird in IPv6-Umgebungen für „*stateful Autoconfiguration*" verwendet. Die Funktionsweise wurde bereits beschrieben. Das Protokoll hat nur einige Erweiterungen gegenüber DHCP.

4.4.8 Migrationaspekte und abschließende Bemerkungen

Zu Migrationszwecken werden heute die IP-Router (Dual-IP-Stack) so ausgelegt, dass sie beide Protokolle (IPv4 und IPv6) beherrschen, wobei u.a. verschiedene Szenarien betrachtet werden müssen:

- Ein IPv4-Netzwerk muss in eine IPv6-Umgebung integriert werden.
- Ein IPv6-Netz muss an ein IPv4-Netz angeschlossen werden.
- Zwei IPv6-Netze kommunizieren über ein IPv4-Netz. Hier kann ein Tunneling-Mechanismus genutzt werden, d.h. IPv6-Nachrichten werden in IPv4-Paketen eingepackt und übertragen.

Die meisten Hersteller von Betriebssystemen (Windows, Linux, Solaris, HP-UX, AIX) und Routern (Cisco) unterstützen heute IPv6, allerdings häufig nicht standardmäßig, sondern als experimenteller oder gesondert zu installierender Protokollstack.

Die meisten Anwendungen sind inzwischen ebenfalls IPv6-fähig. Die Nutzung von IPv6 im privaten Netzwerk ist also möglich. Im globalen Internet ist IPv6 noch nicht etabliert und funktioniert nur über Tunneling-Mechanismen. In Deutschland wurde das *JOIN*-Projekt zur Evaluation von IPv6 abgewickelt. *6Win* hieß der erste IPv6-Backbone in Deutschland, der vom Verein zur Förderung des Deutschen Forschungsnetzes (DFN[37]) aufgebaut wurde. Ein anderes Netzwerk wurde von der Deutschen Telekom unterhalten und ein drittes, das weltweite 6Bone, ist bereits seit Juni 2005 wieder eingestellt.

Ein großer Vorteil von IPv6 ist, dass im Gegensatz zu IPv4 in IPv6 schon Sicherheitsmechanismen im Protokoll spezifiziert sind.[38] Vor allem sind zwei Mechanismen verfügbar:

- Authentifizierung: Der MD5-Algorithmus (Message Digest) kann zur Authentifizierung der Partner verwendet werden.
- Verschlüsselung: Die Verschlüsselung des Nutzdatenteils wird mit einer Variante des DES-Verschlüsselungsalgorithmus unterstützt. DES (Data Encryption Standard) ist ein symmetrisches Verschlüsselungsverfahren.

Damit kann auf IPv4-Erweiterungen wie z.B. zur Gewährleistung von Netzwerksicherheit in IPv6 verzichtet werden. Auf die Sicherheitsaspekte soll an dieser Stelle nicht weiter eingegangen werden. In der angegebenen Literatur gibt es einige interessante Ausführungen hierzu.

Viele wichtige Aspekte für moderne Hochleistungsnetze sowie Anforderungen an mobile Systeme und Sicherheitsprotokolle sind in IPv6 bereits integriert. Ein Umstieg brächte also für einige Anwendungstypen enorme Vorteile. Allerdings ist abzusehen, dass die komplette Umstellung von IPv4 auf IPv6 möglicherweise noch lange dauern wird, zumal auch die Adressprobleme durch Mechanismen wie NAT und CIDR im Moment zumindest eingedämmt sind. Migrationskonzepte sind erforderlich, da die Umstellung nicht auf einmal möglich ist. Für die nächsten Jahre ist wohl eine Koexistenz von IPv4 und IPv6 zu erwarten. Dies erfordert je nach Netzwerksituation Migrationstechniken wie zum Beispiel die Kopplung von IPv6-Netze über ein IPv4-Netzwerk und das Betreiben von Dual-IP-Routern und Endsystemen.

Es soll noch erwähnt werden, dass neben den Schicht-3-Protokollen für eine Migration nach IPv6 auch höhere Protokolle angepasst werden müssen. Im folgenden Kapitel werden zum Beispiel sog. Pseudoheader in UDP und in TCP erläutert. Diese nutzen IP-Adressen der Quell- und Zielhosts zur Berechnung von Prüfsum-

---

[37] DFN = Deutsches Forschungsnetz.

[38] In IPv4 gibt es keine derartigen Mechanismen, weshalb das IPsec-Protokoll (IP-Security) eingeführt wurde. Das Protokoll stellt auch für IPv4 Authentifizierungs- und Verschlüsselungsmechanismen bereit.

men. Da die IPv6-Adressen wesentlich länger sind, müssen die Prüfsummenberechnungen angepasst werden. Auch auf DNS, das ebenfalls weiter unten beschrieben wird, hat eine IPv6-Umstellung Auswirkungen. Dort wurden neue DNS-Recordtypen eingeführt, mit denen IPv6-Adressen angegeben werden können. Es gibt also auch IPv6-fähige UDP-, TCP- und DNS-Varianten.

4.5 Virtual Private Networks

Ein Virtuelles Privates Netz (VPN) ist ein Netzwerk, das zum Transport privater Daten ein öffentliches Netz (zum Beispiel das Internet) nutzt. Im IP-Umfeld werden heute VPNs immer mehr eingesetzt, da sie eine direkte Verbindung zwischen den einzelnen Netzwerken überflüssig machen. Teilnehmer eines VPN können Daten wie in einem internen LAN austauschen. Die einzelnen Teilnehmer selbst müssen hierzu nicht direkt verbunden sein.

Das Problem ist jedoch die Gewährleistung einer abhörsicheren Übertragung über das doch recht unsichere globale Internet. Die Verbindung über das öffentliche Netz (meist das globale Internet) muss daher verschlüsselt werden. Eine Verbindung der Netze wird über einen sog. Tunnel zwischen VPN-Client und VPN-Server ermöglicht (IP-Tunneling).

VPNs werden oft verwendet, um Mitarbeitern außerhalb einer Organisation oder Firma den Zugriff auf das interne Netz zu geben. Dabei baut ein Endsystem des Mitarbeiters eine sog. VPN-Verbindung zu dem ihm bekannten VPN-Gateway des Unternehmens auf. Über diese Verbindung ist es dann möglich, so zu arbeiten, als ob man sich im lokalen Netz der Firma befindet. Aber auch die Anbindung eines ganzen Netzes beispielsweise einer Filiale an ein Unternehmensnetzwerk eines Mutterunternehmens, wie dies in Abbildung 4-66 skizziert wird, ist über ein VPN sinnvoll. Wenn zwei gleichberechtigte Netze über ein VPN verbunden werden, wird auf beiden Seiten ein VPN-Gateway verwendet. Diese bauen dann untereinander eine VPN-Verbindung auf. Andere Rechner in einem lokalen Netz verwenden nun jeweils das lokale VPN-Gateway, um Daten in das andere Netz zu senden. So lassen sich zum Beispiel zwei weit entfernte Standorte einer Firma verbinden. Die VPN-Gateways sorgen für die Authentifizierung der Teilnehmer und die Abbildung der IP-Adressen auf die lokale Umgebung.

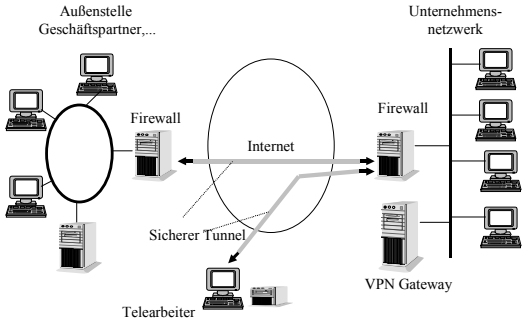


Abbildung 4-66: Ein VPN-Beispiel

Das bekannteste Protokoll (es ist wohl eher eine Gruppe von Protokollen) zum Betreiben eines VPN im globalen Internet ist *IPSec* (IP Security), das entwickelt wurde, um die Schwächen von IPv4 zu beheben. IPSec entstand auch im Zuge der Entwicklung von IPv6 und ist in mehreren RFCs spezifiziert (RFCs 2401, 2402, 2406, 2407, 2408 und 2409, wobei RFC 2401 das Basisdokument ist). IPSec enthält Mechanismen zur Authentifizierung über Schlüsselaustausch und zur Verschlüsselung von Daten. Das Protokoll ist ziemlich komplex und soll an dieser Stelle nicht weiter vertieft werden.

4.6 Übungsaufgaben

1. Nennen Sie zwei Aufgaben der Vermittlungsschicht und beschreiben Sie diese kurz!
2. Was unterscheidet die Vermittlungsverfahren „Leitungsvermittlung“ und „Paketvermittlung“?
3. Warum werden virtuelle Verbindungen in der Schicht 3 auch scheinbare Verbindungen genannt?
4. Erläutern Sie den Unterschied zwischen statischen und dynamischen Routing-Mechanismen! Nennen Sie je ein konkretes Verfahren hierzu!
5. Was versteht man unter einem zentralen Routing-Verfahren? Handelt es sich hier um ein statisches oder um ein adaptives Routing-Verfahren?
6. Welche Vorteile bringt ein hierarchisches Routing-Verfahren?
7. Erläutern Sie kurz das Optimierungsprinzip beim Routing!
8. Was versteht man im Distance-Vector-Routing-Verfahren unter dem Count-to-Infinity-Problem und wie verhält sich das Verfahren im Hinblick auf Konvergenz? Begründen Sie Ihre Entscheidung!
9. Welche Art von Routing-Verfahren ist das Distance-Vector-Verfahren und wann wird es in IP-Netzen auch heute noch eingesetzt?

10. Nennen Sie drei mögliche Metriken, die ein Routing-Verfahren zur Ermittlung der optimalen Routen nutzen kann!
11. Wie sieht ein einzelner Router im Link-State-Routing-Verfahren die aktuelle Netzwerktopologie?
12. Sind im Link-State-Routing-Verfahren Schleifen möglich? Begründen Sie Ihre Entscheidung!
13. Was versteht man unter einem Leaky-Bucket-Verfahren und wozu dient es?
14. Was ist ein autonomes System im Internet und welche Arten von autonomen Systemen kennen Sie?
15. Was ist im globalen Internet ein Transit-AS?
16. Beschreiben Sie kurz den Dienst, den IP für die darüberliegenden Schichten zur Verfügung stellt im Hinblick auf die Übertragungssicherheit!
17. Welches Vermittlungsverfahren verwendet die Internet-Schicht?
18. Was versteht man unter einem Sink Tree im Sinne der Wegewahl?
19. Was versteht man unter NAT (Network Address Translation) und welche Vorteile bietet das Verfahren?
20. Welche Aufgabe verrichtet ein NAT-Router im Rahmen der Adressierung für ankommende und abgehende IP-Pakete?
21. Warum muss ein NAT-Router vor dem Weiterleiten eines vom globalen Internet ankommenden oder vom Intranet abgehenden IP-Paketes die Checksumme im IP-Header jedesmal neu berechnen?
22. Nennen Sie den Unterschied zwischen der klassischen Subnetz-Adressierung und dem Classless Interdomain Routing (CIDR)! Welche Vorteile bringt CIDR für die Adressenknappheit im Internet (Begründung)?
23. Wozu wird in IPv4-Netzen im Router das Wissen über eine Netzwerkmaske für jedes angeschlossene Netz benötigt?
24. Was bedeutet in CIDR die Darstellung 132.10.1.8/24?
25. Wie findet ein Host innerhalb eines LANs (IPv4-Netzwerks) die MAC-Adresse eines Partner-Hosts, wenn er das erste Mal ein IP-Paket an diesen senden will?
26. Wie findet ein Host die MAC-Adresse eines Partner-Hosts, der nicht im eigenen LAN, sondern irgendwo in einem entfernten LAN, das aber über einen Router erreichbar ist, liegt?
27. Über welches Steuerprotokoll wird eine Ping-Nachricht abgesetzt? Verwendet das besagte Steuerprotokoll TCP, UDP oder direkt IP zur Nachrichtenübertragung?

28. Erläutern Sie den Unterschied zwischen limited Broadcast und directed Broadcast in IP-Netzen! Wann benötigt man z.B. diese Broadcast-Varianten (Nennen Sie je ein Beispiel)?
29. Was kann die Vermittlungsschicht zur Vermeidung bzw. zum Abbau von Stausituationen im Netz beitragen?
30. Kann man innerhalb eines autonomen Systems im globalen Internet unterschiedliche Routing-Verfahren verwenden? Begründen Sie Ihre Entscheidung!
31. Warum werden in IPv4 IP-Adressen „verschenkt" und wie werden im derzeitigen globalen Internet IP-Adressen eingespart? Nennen Sie zwei Einsparvarianten!
32. Welche Bedeutung hat das TTL-Feld im IP-Header und wie wird es in einem Router im Rahmen der Bearbeitung eines ankommenden IP-Pakets bearbeitet?
33. Wozu benötigt eine IP-Instanz das Feld Protokoll aus dem IP-Header?
34. Beschreiben Sie kurz den Protokollmechanismus der Fragmentierung am Beispiel von IPv4 und gehen Sie dabei auf die genutzten Felder Identifikation, Fragment Offset und Flags ein!
35. Welches Problem im Routing-Protokoll RIP versucht die Split-Horizon-Technik zu lösen und wie funktioniert diese Technik?
36. Im globalen Internet setzt man prinzipiell zwei verschiedene Routing-Verfahren ein (EGP und IGP). Erläutern Sie den Unterschied zwischen EGP und IGP und stellen Sie dar, wo beide Routing-Verfahren Verwendung finden? Nennen Sie je ein konkretes Routing-Protokoll für die beiden Verfahren!
37. Nennen Sie drei Ziele der IPv6-Entwicklung!
38. Welchen Sinn haben im IPv6-Protokoll die sog. Erweiterungsheader? Nennen Sie zwei Erweiterungsheader und beschreiben Sie kurz deren Aufgabe!
39. Wozu sollen im IPv6-Protokoll Flussmarken dienen?
40. Host A sendet in einem IPv4-Netzwerk seinem Partnerhost B ein IP-Paket der Länge 5000 Byte. 20 Byte davon enthält der IP-Header des Pakets (minimaler IP-Header ohne Optionen). Es gelten folgende Bedingungen:
    - Das IP-Paket muss von A nach B drei IP-Netze durchlaufen, Host A liegt im Netz 1, Netz 2 ist ein Transitnetz und Host B liegt in Netz 3
    - Netz 1 und Netz 2 werden durch Router R1 verbunden
    - Netz 2 und Netz 3 werden durch Router R2 verbunden
    - Netz 1 hat eine MTU von 2048 Byte
    - Netz 2 hat eine MTU von 1024 Byte
    - Netz 3 hat eine MTU von 576 Byte

    ▪ Skizzieren Sie die gesamte Netzwerktopologie!

- Wie viele IP-Fragmente verlassen R1 für das besagte IP-Paket in Richtung Netzwerk mit der Nummer 2?
- Wie viele IP-Fragmente verlassen R2 für das besagte IP-Paket in Richtung Netzwerk mit der Nummer 3?
- In welchem System (Host oder Router) werden die IP-Fragmente wieder zusammengebaut? Wie wird in diesem System erkannt, welche IP-Fragmente zum ursprünglichen IP-Paket gehören?

41. Eine Organisation hat von seinem ISP den IPv4-Adressblock 131.42.0.0/16 (classless) zugewiesen bekommen. Die Organisation möchte gerne ihr Netzwerk intern wie folgt aufteilen:

    – 1 Subnetz mit bis zu 32000 Rechnern
    – 15 Subnetze mit bis zu 2000 Rechnern
    – 8 Subnetze mit bis zu 250 Rechnern

    Der Adressblock wird zunächst in die zwei Adressblöcke 131.42.0.0./17 und 131.42.128.0/17 aufgeteilt. Zeigen Sie auf, wie die Organisation intern die Adressen weiter aufteilen könnte, um obiges Ziel zu erreichen. Hinweis: Alle beteiligten Router beherrschen CIDR.

42. Was passiert, wenn ein IPv4-Fragment, also ein Teil eines IPv4-Pakets, in einem Netzwerk landet, dessen MTU kleiner ist als die Länge des Fragments?

43. Wo (auf welchem Rechner) werden IPv4-Fragmente wieder zum ursprünglich abgesendeten IPv4-Datagramm reassembliert?

44. Was versteht man im Sinne der IP-Adressvergabe unter einem multihomed Host?

45. Wie lauten die entsprechenden Netzwerkmasken für die CIDR-Präfixnotationen /16, /20 und /24?

46. Aus dem Adressbereich 11.1.253/24 eines VLSM-Teilnetzes sollen /27-Teilnetze herausgeschnitten werden. Wie lauten die Teilnetzwerkadressen? Wie viele Adressen bleiben pro /27-Teilnetz?

47. Erläutern Sie, wie ein neu in ein Netzwerk hinzukommender OSPF-Router seine Routing-Information aufbaut und verwaltet. Gehen Sie dabei auf den Begriff des Spanning-Tree und auf die nachbarschaftliche Beziehung der OSPF-Router ein.

48. Ein Problem bei Routing-Protokollen ist das Konvergenzverhalten bzw. die Konvergenzdauer bei Änderungen der Netzwerktopologie oder bei Änderungen von Routen. Wie ist das Konvergenzverhalten bei den Routing-Protokollen RIP-2 und OSPFv2? Welche Mechanismen nutzt RIP-2 zur Verbesserung der Konvergenz? Sind in beiden Routing-Protokollen Endlosschleifen (Count-to-Infinity-Problem) möglich?

49. Wie funktioniert bei IPv6 prinzipiell die automatische Adresskonfiguration?

5 Konzepte und Protokolle der Transportschicht

Transportprotokolle wie TCP und UDP sind die Basis für die höherwertigen Kommunikationsmechanismen. Was man in den höheren Schichten sonst nicht sieht, aber sich im Hintergrund der Kommunikation abspielt, soll in diesem Kapitel dargestellt werden, um auch die Komplexität der Kommunikationsabläufe einschätzen zu können.

In diesem Kapitel wird zunächst die Funktionalität von Transportschichten allgemein betrachtet. Hierbei werden Themen wie das Verbindungsmanagement und die Datenübertragung einschließlich der Flusskontroll- und der Staukontrollmechanismen betrachtet. In den anschließenden Abschnitten werden die konkreten TCP- und UDP-Protokollmechanismen erläutert. Spezielle Lösungen dieser beiden Protokolle, insbesondere des weitaus komplexeren TCP, wie etwa das Slow-Start-Verfahren, das spezielle Timer-Management usw. werden dargestellt. Die Protokollheader beider Protokolle werden ebenfalls diskutiert.

Zielsetzung des Kapitels

Der Studierende soll verstehen, wie Transportprotokolle grundsätzlich arbeiten und wie im Speziellen die Protokolle TCP und wie UDP funktionieren.

Wichtige Begriffe

Drei-Wege-Handshake, Go-Back-N-Verfahren, Kumulatives Quittierungsverfahren, NAK-Verfahren, Sliding-Windows-Verfahren, Slow-Start-Verfahren, UDP- und TCP-Ports.

5.1 Grundlagen

In diesem Abschnitt werden grundlegende Mechanismen der Transportschicht als Basis für die anschließende konkrete Betrachtung der Protokolle TCP und UDP zunächst allgemein diskutiert.

5.1.1 Transportdienste

Die Transportschicht stellt den anwendungsorientierten Schichten einen Transportdienst für eine Ende-zu-Ende-Beziehung zur Verfügung. Höhere Schichten nutzen die Dienste zum Austausch von Nachrichten über einen Dienstzugang (T-SAP, vgl. Abbildung 5-1). Je nach Ausprägung der Transportschicht ist der Dienst *zuverlässig* oder *unzuverlässig* im Sinne der Datenübertragung.

Man unterscheidet weiterhin grundsätzlich zwischen *verbindungsorientierten* und *verbindungslosen* Transportdiensten. Bei ersteren wird zwischen den Kommunikationspartnern vor dem eigentlichen Datenaustausch eine Transportverbindung etabliert, bei letzteren ist dies nicht notwendig.

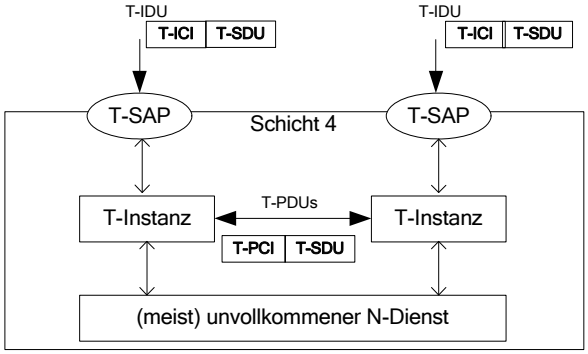


Abbildung 5-1: Dienste der Transportschicht

Verbindungsorientierte Transportdienste sind durch drei Phasen gekennzeichnet:

- Verbindungsaufbau
- Datenübertragung
- Verbindungsabbau

Der Sender adressiert den Empfänger beim Verbindungsaufbau und sendet beim Datenaustausch in den T-PDUs (auch als Segmente bezeichnet) nur noch seine Verbindungs-Identifikation mit. Für eine Verbindung muss bei beiden Kommunikationspartnern in den Transportinstanzen ein gemeinsamer Verbindungskontext verwaltet werden, der sich für jedes Protokoll in einem Zustandsautomaten beschreiben lässt.

Verbindungslose Dienste sind meist durch folgende Eigenschaften charakterisiert:

- Der Verlust von Datenpaketen ist möglich
- Die Daten können ggf. verfälscht werden
- Die Reihenfolge ist nicht garantiert
- Die Adressierungsinformation muss in allen T-PDUs enthalten sein

Verbindungsorientierte Protokolle sind komplexer und daher aufwändiger zu implementieren, bieten aber in der Regel eine höhere Zuverlässigkeit und garantieren meist eine fehlerfreie und reihenfolgerichtige Auslieferung der Daten beim Empfänger.

Nicht jede Anwendung benötigt einen verbindungsorientierten Transportdienst und entsprechende Zuverlässigkeit. Dies ist z.B. für neuere Anwendungstypen wie etwa Videoübertragungen (Streaming) der Fall. Hier wird oft auf verbindungs-

lose Dienste zurückgegriffen, da ein begrenzter Datenverlust vertretbarer als Verzögerungen bzw. Verzögerungsschwankungen in der Datenübertragung ist.

Die grundlegenden Protokollfunktionen, die in der Transportschicht angesiedelt werden, sind je nach Protokoll unterschiedlich. Grundsätzlich sind folgende Aufgaben zu erfüllen:

- Verbindungsmanagement und Adressierung
- Zuverlässiger Datentransfer
- Flusskontrolle
- Staukontrolle
- Multiplexierung und Demultiplexierung
- Fragmentierung (bzw. Segmentierung) und Defragmentierung

Welche Mechanismen die T-Instanz implementiert, hängt sehr stark von der darunterliegenden N-Schicht ab. Ist diese schon sehr zuverlässig, kann die Implementierung in der T-Instanz einfacher sein.

5.1.2 Verbindungsmanagement und Adressierung

In Abbildung 5-2 ist die Kommunikation zwischen zwei Anwendungsprozessen in zwei verschiedenen Rechnersystemen (Hosts) dargestellt. Die Anwendungsprozesse sind über die zugehörigen T-SAPs adressierbar, während die Transportschicht N-SAPs zur Adressierung der Rechnersysteme nutzt. Die Schicht-4-Adresse wird auch als Transportadresse bezeichnet. In der Abbildung ist zu erkennen, dass ein Anwendungsprozess neben dem N-SAP noch ein weiteres Identifikationsmerkmal aufweisen muss, da ja mehr als ein Anwendungsprozess eines Hosts über denselben N-SAP kommunizieren kann. Bei TCP ist dies z.B. die sog. Portnummer.

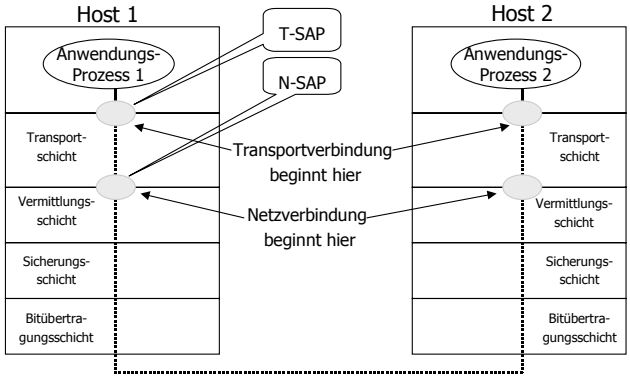


Abbildung 5-2: Adressierung über T-SAP

Eine T-Instanz unterstützt in der Regel mehrere T-SAPs. Transportadressen sind meist sehr kryptisch, weshalb oft symbolische Adressen benutzt werden, die über sog. Naming-Services (Directory Services) auf Transportadressen abgebildet werden, ohne dass der Anwendungsprogrammierer diese kennen muss. Ein Beispiel für einen klassischen Naming-Service ist das im Internet unbedingt erforderliche DNS (Domain Name System).

Verbindungsorientierte Dienste erfordern einen Verbindungsaufbau, in dem zunächst auf beiden Seiten ein Verbindungskontext aufgebaut wird. Die Endpunkte der Verbindung werden im ISO/OSI-Jargon auch als Connection End Points (CEP) bezeichnet.

Verbindungsaufbau: Beim Verbindungsaufbau muss sichergestellt werden, dass keine Duplikat-PDUs alter Verbindungen erneut beim Empfänger ankommen. Hierzu sind entsprechende Protokollmechanismen wie etwa eine Folgenummer (Sequenznummern) als fortlaufender Zähler der abgesendeten Nachrichten kombiniert mit einer maximalen Paketlebensdauer erforderlich.

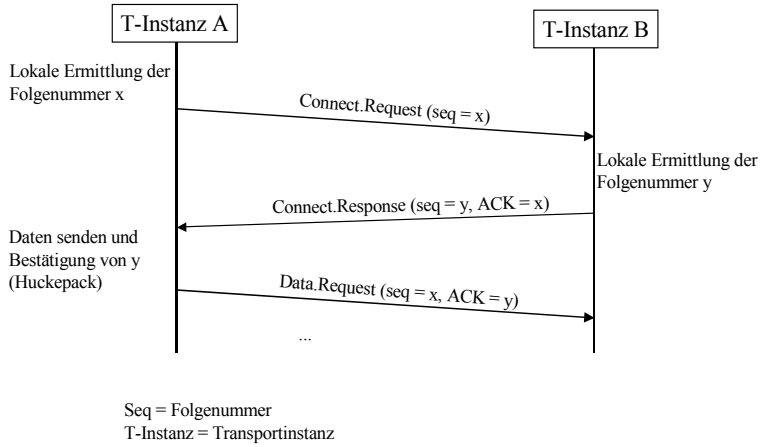


Abbildung 5-3: Drei-Wege-Handshake-Protokoll für den normalen Verbindungsaufbau

Ein Verbindungsaufbau erfolgt meist über einen bestätigten Dienst, in dem Folgenummern vereinbart werden. Ein klassisches Verbindungsaufbauprotokoll ist das Drei-Wege-Handshake-Protokoll, das in Abbildung 5-3 skizziert ist. Der Ablauf sieht wie folgt aus:

- Host A (bzw. die T-Instanz in Host A) initiiert den Verbindungsaufbau mit einer Connect-Request-PDU und sendet dabei eine vorher ermittelte Folgenummer (seq=x) mit. Mit der Connect-Request-PDU wird auch die Transportadresse gesendet, um den Empfänger zu identifizieren.

- Host B (bzw. die T-Instanz in Host B) bestätigt den Verbindungsaufbauwunsch mit einer ACK-PDU, bestätigt dabei die Folgenummer und sendet seine eigene Folgenummer (seq=y) mit der ACK-PDU an Host A.
- Host A bestätigt die Folgenummer von Host B mit der ersten Data-PDU im Piggypacking-Verfahren, und damit ist die Verbindung aufgebaut.

Wie in Abbildung 5-4 dargestellt, können beim Verbindungsaufbau gewisse Fehlersituationen auftreten. Beispielsweise kann eine alte Connect-Request-PDU bei Host B ankommen, der dann nicht feststellen kann, ob dies eine gültige PDU ist. Host A muss erkennen, dass es sich um einen Connect-Response für einen Connect-Request handelt, der nicht bzw. in der Vergangenheit von ihm initiiert wurde. Er muss die ACK-PDU dann negativ beantworten und die Verbindung zurückweisen (Reject).

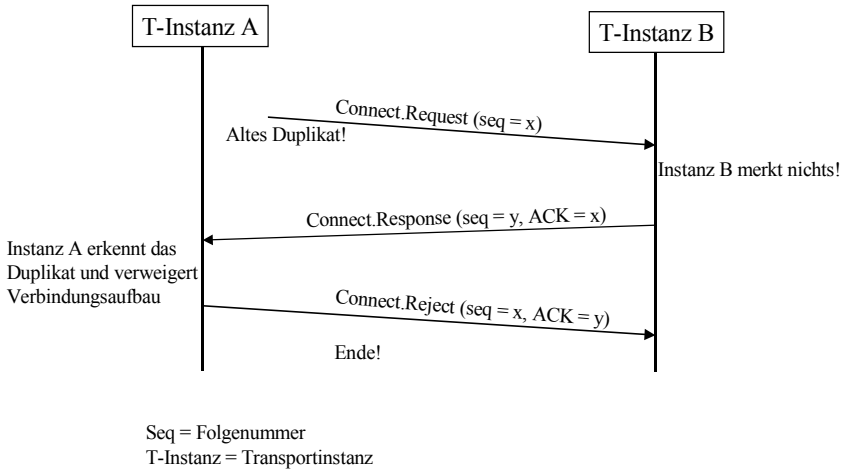


Abbildung 5-4: Verbindungsaufbau, Fehlerfall 1: Connect-Request-Duplikat taucht auf

Eine noch kompliziertere Fehlersituation kann auftreten, wenn zusätzlich zum Connect-Request-Duplikat noch ein altes ACK-Duplikat auftritt. Auch hier darf die Verbindung nicht zustande kommen (vgl. Abbildung 5-5).

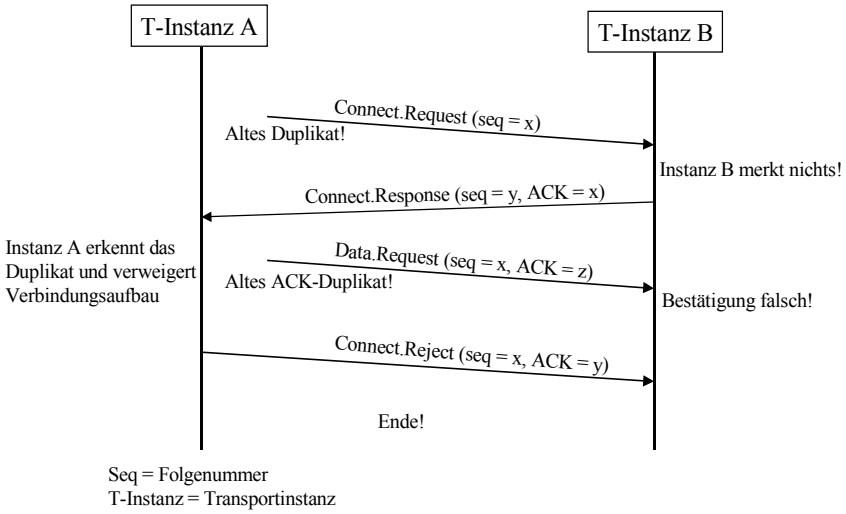


Abbildung 5-5: Verbindungsaufbau, Fehlerfall 2: CR- und ACK-Duplikate tauchen auf

Verbindungsabbau: Auch an einen ordnungsgemäßen Verbindungsabbau werden einige Anforderungen gestellt: Beim Verbindungsabbau dürfen keine Nachrichten verloren gehen. Ein Datenverlust kann nämlich vorkommen, wenn eine Seite einen Verbindungsabbau initiiert, die andere aber vor Erhalt der Disconnect-Request-PDU noch eine Nachricht sendet. Diese Nachricht ist dann verloren (Datenverlust). Daher ist ein anspruchsvolles Verbindungsabbau-Protokoll notwendig. Auch hier verwendet man gerne einen Drei-Wege-Handshake-Mechanismus, in dem beide Seiten[1] ihre „Senderichtung" abbauen.

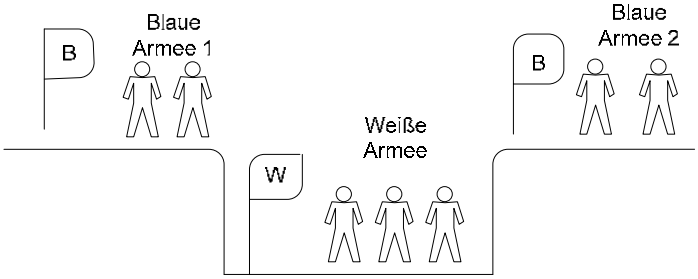


Abbildung 5-6: Das Zwei-Armeen-Problem aus (Tanenbaum 2003a)

[1] Wir gehen hier von einer Vollduplex-Verbindung aus.

Das Problem wird durch das Zwei-Armeen-Beispiel transparent. Die Armee der Weißröcke lagert in einem Tal. Auf zwei Anhöhen lagert jeweils ein Teil der Armee der Blauröcke. Die Blauröcke können nur gemeinsam gewinnen und müssen ihren Angriff synchronisieren. Zur Kommunikation der beiden Teilarmeen der Blauröcke existiert ein unzuverlässiger Kommunikationskanal, und zwar Boten, die zu Fuß durch das Tal rennen müssen (vgl. Abbildung 5-6).

Die eine Seite der Blauröcke sendet einen Boten mit einer Nachricht los. Damit sie aber sicher sein kann, dass er angekommen ist, muss die zweite Seite die Nachricht über einen weiteren Boten bestätigen. Was ist aber, wenn der nicht ankommt? Und wenn er ankommt, woher weiß es dann die zweite Seite? Kein Protokoll ist hier absolut zuverlässig, denn es wird immer eine Seite geben, die unsicher ist, ob die letzte Nachricht angekommen ist.

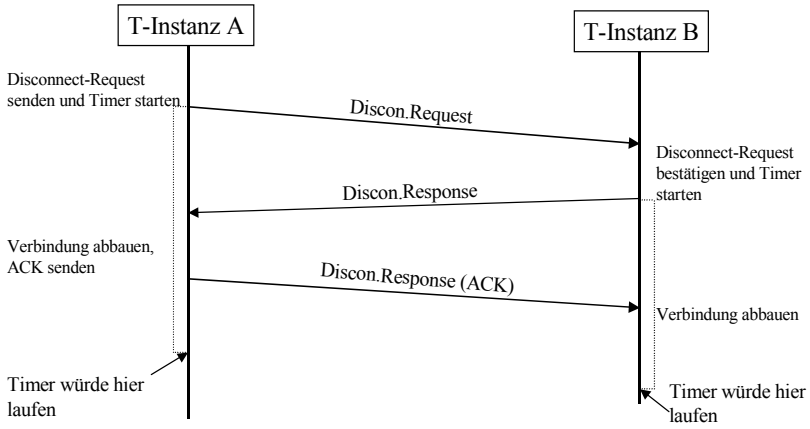


Abbildung 5-7: Szenario beim Verbindungsabbau, normaler Ablauf nach (Tanenbaum 2003a)

Übertragen auf den Verbindungsabbau bedeutet dies, dass beim Drei-Wege-Handshake jederzeit ein Disconnect-Request oder eine Bestätigung verloren gehen kann. Man löst das Problem in der Praxis pragmatisch über eine *Timerüberwachung* mit begrenzter Anzahl an Nachrichtenwiederholungen (wird bei Protokollen oft so gehandhabt). Dies liefert dann keine unfehlbaren, aber doch ganz zufriedenstellende Ergebnisse. Aber ein Problem bleibt bestehen: Falls der erste Disconnect-Request und n weitere verloren gehen, baut der Sender die Verbindung ab und der Partner weiß nichts davon. In diesem Fall liegt eine *halboffene* Verbindung vor. Aber auch dieses Problem kann gelöst werden, indem beim Senden einer Nachricht nach einer bestimmten Zeit die Verbindung abgebaut wird, wenn keine Antwort zurückkommt. Der Partner baut dann auch die Verbindung irgendwann wieder ab. In den folgenden Abbildungen sind einige Szenarien für die Timerüberwa-

chung beim Verbindungsabbau skizziert, die in einem verbindungsorientierten Transportprotokoll behandelt werden sollten. Abbildung 5-7 stellt den normalen Verbindungsabbau als Drei-Wege-Handshake dar. In Abbildung 5-8 ist ein Szenario skizziert, in dem ein Timer abläuft. In diesem Fall trennt die Instanz B die Verbindung beim Ablauf des Timers.

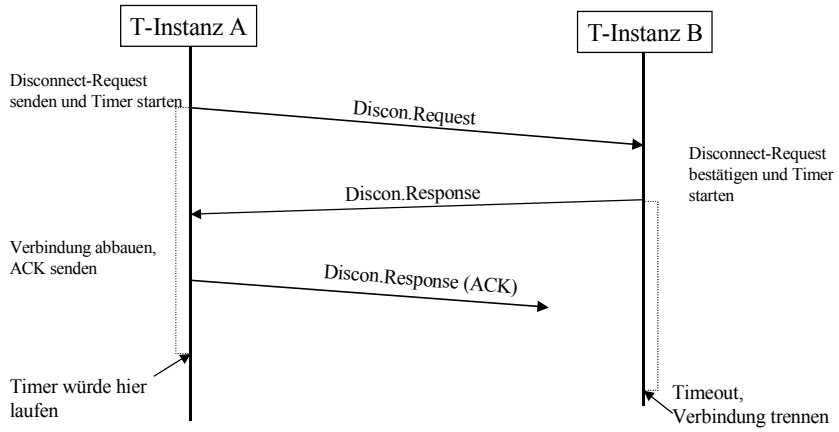


Abbildung 5-8: Szenario beim Verbindungsabbau, Timerablauf

In Abbildung 5-9 geht aus irgendeinem beliebigen Grund (z.B. Netzwerküberlastung) die Disconnect-Response-PDU der Instanz B verloren und kommt daher nicht bei Instanz A an. In diesem Fall läuft bei Instanz A der Timer ab, und die Disconnect-Request-PDU wird erneut gesendet.

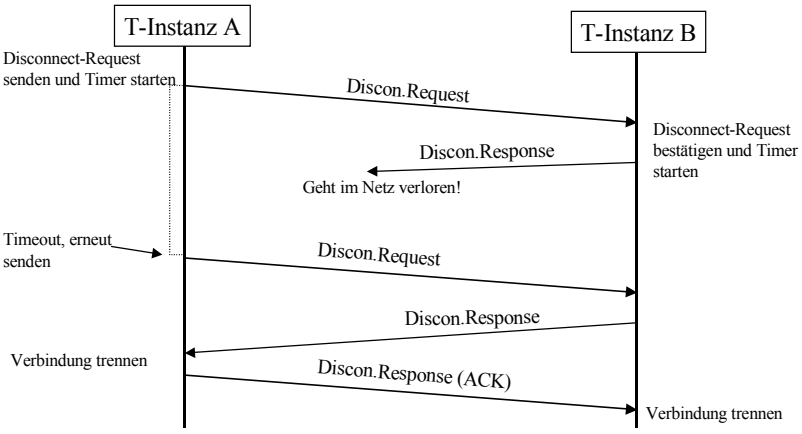


Abbildung 5-9: Szenario beim Verbindungsabbau, Disconnect-Responses gehen verloren

Schließlich gehen in dem Szenario, das in Abbildung 5-10 skizziert ist, die Disconnect-Response-PDU der Instanz B und eine erneut gesendete Disconnect-Request-PDU der Instanz A verloren. Hier greift auf beiden Seiten der Timeout-Mechanismus. Die Verbindungen werden selbstständig von den beiden Instanzen getrennt, wenn die Wartezeit auf die Antwortnachricht des jeweiligen Partners abgelaufen ist. Wie oft eine Nachricht bei einem Timeout-Ereignis wiederholt wird, hängt vom jeweiligen Transportprotokoll ab. Es gibt sicherlich noch weitere Fehlervariationen, die aber alle über die Timerüberwachung lösbar sind.

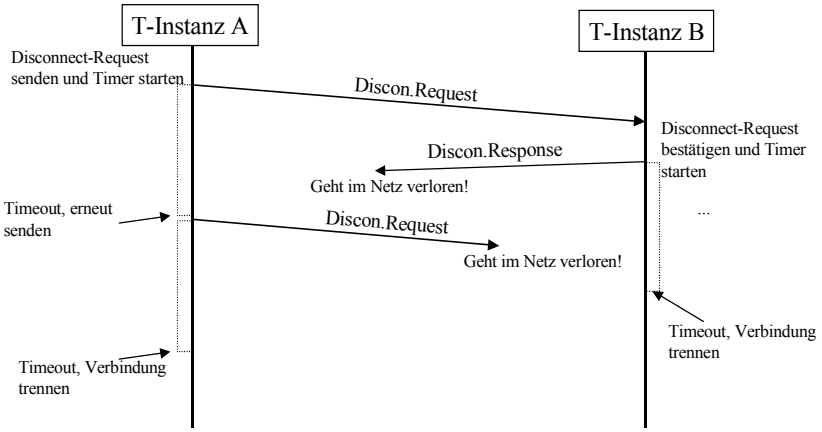


Abbildung 5-10: Szenario beim Verbindungsabbau, zwei Disconnect-PDUs gehen verloren

5.1.3 Zuverlässiger Datentransfer

Bei einem zuverlässigen Datentransfer, wie ihn sowohl TCP als auch OSI TP 4 gewährleistet, werden die Daten in der richtigen Reihenfolge, vollständig und ohne Fehler übertragen. Probleme werden erkannt und vom Transport-Provider, sofern möglich, gelöst oder über die Diensteschnittstelle nach oben weiter gemeldet. Weiterhin werden bei einem zuverlässigen Datentransfer Duplikate vermieden. Ein zuverlässiger Datentransfer erfordert daher geeignete Protokollmechanismen zur Fehlererkennung und Fehlerbehebung, ein Empfänger-Feedback (Bestätigungen), eine Möglichkeit der Neuübertragung von Daten sowie eine geeignete Flusskontrolle.

Quittierungsverfahren

Was das Transportprotokoll tatsächlich leisten muss, um Zuverlässigkeit zu gewährleisten, hängt vom Dienst der Schicht 3 ab. Bei einem absolut zuverlässigen Kanal der Schicht 3 muss die Schicht 4 natürlich weniger tun als bei verlustbehafteten Kanälen.

Um sicher zu sein, dass Nachrichten in einem verlustbehafteten Kanal richtig ankommen, sind Quittierungsverfahren erforderlich. Hier gibt es gravierende Unterschiede hinsichtlich der Leistung. Einfache Verfahren quittieren jede Data-PDU mit einer ACK-PDU (Acknowledge), was natürlich nicht sehr leistungsfähig ist. Der Sender muss bei diesem Stop-and-Wait-Protokoll immer warten, bis eine Bestätigung eintrifft, bevor er weitere Data-PDUs senden kann. Aufwändigere Verfahren ermöglichen das Senden mehrerer Daten und sammeln die Bestätigungen ggf. kumuliert ein. Hier spricht man auch von *Pipelining*.

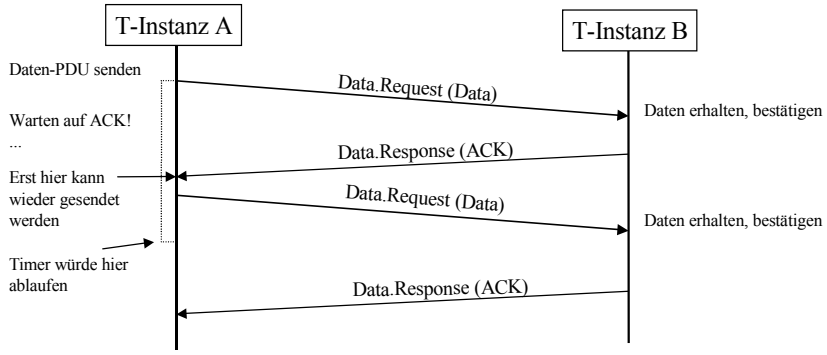


Abbildung 5-11: Positiv-selektives Quittungsverfahren

Man unterscheidet bei Quittungsverfahren:

- *Positiv-selektives* Quittierungsverfahren wie das Stop-and-Wait-Protokoll: Hier wird vom Empfänger eine Quittung (ACK) pro empfangener Nachricht gesendet. Dies hat einen hohen zusätzlichen Nachrichtenverkehr zur ge. Abbildung 5-11 zeigt dieses Verfahren.
- *Positiv-kumulatives* Quittierungsverfahren: Eine Quittung wird für mehrere Nachrichten verwendet. Diese Reduzierung der Netzlast hat aber den Nachteil, dass Information über einen Datenverlust verspätet an den Sender übertragen wird.
- *Negativ-selektives* Quittierungsverfahren: Es werden vom Empfänger nur selektiv nicht empfangene, also verlorengegangene, Nachrichten erneut vom Sender angefordert. Alle Nachrichten, bei denen keine Nachfrage kommt, gelten beim Sender als angekommen. Dieses Verfahren reduziert die Netzlast weiter. Ein Verlust von negativen Quittungen (Nachfragen des Empfängers) ist jedoch möglich und muss behandelt werden.
- Eine *Kombination der Verfahren*: In Hochleistungsnetzen verwendet man z.B. das positiv kumulative kombiniert mit dem negativ selektiven Verfahren.

Das positiv-kumulative Verfahren ist in Abbildung 5-12 angedeutet. Im Szenario werden von Instanz A zwei Nachrichten gesendet und durch Instanz B gemeinsam

bestätigt. Instanz A muss in diesem Fall nicht nach jeder Nachricht auf eine Bestätigung warten.

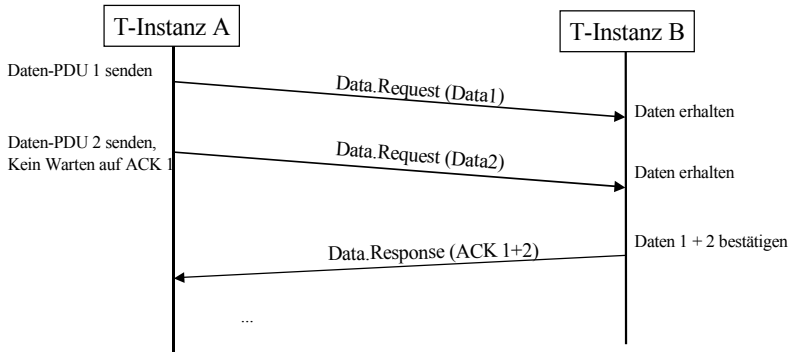


Abbildung 5-12: Positiv-kumulatives Quittungsverfahren

In Abbildung 5-13 wird ein Beispiel für eine negativ-selektive Quittung dargestellt. Die 2. Nachricht kommt bei Instanz B nicht an und wird bei Eintreffen der Nachricht 3 nochmals angefordert. Falls die NAK-Nachricht (Not-Acknowledge) nicht beim Sender ankommt, weiß dieser vom Verschwinden der Nachricht 2 nichts. Daher muss die Instanz B durch Zeitüberwachung dafür sorgen, dass die NAK-Nachricht erneut gesendet wird.

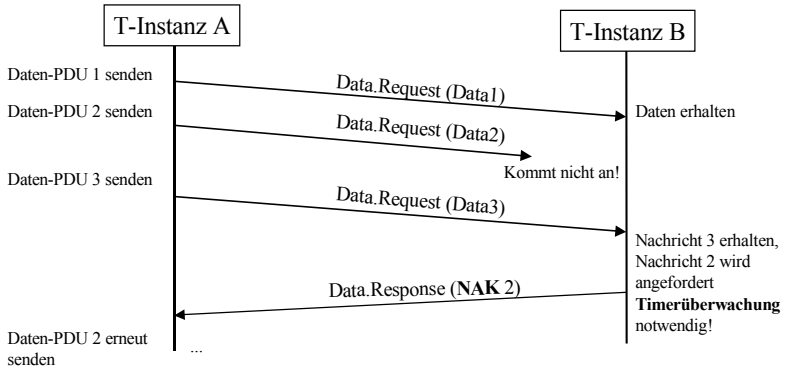


Abbildung 5-13: Negativ-selektives Quittungsverfahren

Übertragungswiederholung

Verlorengegangene Nachrichten müssen erneut übertragen werden. Diesen Vorgang nennt man *Übertragungswiederholung*. Die positiven Quittierungsverfahren benötigen für jede gesendete PDU eine Timerüberwachung, damit nicht endlos auf

eine ACK-PDU gewartet wird. Die Übertragungswiederholung ist dabei auf zwei Arten üblich. Entweder man verwendet eine *selektive Wiederholung* oder ein *Go-Back-N-Verfahren*:

- Beim selektiven Verfahren werden nur die negativ quittierten Nachrichten wiederholt. Der Empfänger puffert die nachfolgenden Nachrichten, bis die fehlende da ist. Erst wenn dies der Fall ist, werden die Daten am T-SAP nach oben zur nächsten Schicht weitergereicht. Nachteilig ist dabei, dass eine hohe Pufferkapazität beim Empfänger erforderlich ist. Von Vorteil ist, dass die reguläre Übertragung während der Wiederholung fortgesetzt werden kann.
- Beim Go-Back-N-Verfahren werden die fehlerhafte Nachricht sowie alle nachfolgenden Nachrichten erneut übertragen. Nachteilig ist hier, dass die reguläre Übertragung unterbrochen wird. Von Vorteil ist, dass beim Empfänger nur geringe Speicherkapazität erforderlich ist.

Sender und Empfänger müssen sich über das verwendete Verfahren einig sein. Die genaue Vorgehensweise ist also in der Protokollspezifikation festzulegen.

Für beide Verfahren gilt generell, dass der Sender Nachrichten über einen gewissen Zeitraum zur Übertragungswiederholung bereithalten muss. Es kann ja jederzeit vorkommen, dass eine Nachricht erneut angefordert wird. Beim Stop-and-Wait-Verfahren ist dies relativ einfach. Der Sender muss nur eine Nachricht speichern und kann sie bei Empfang der ACK-PDU verwerfen. Schwieriger ist es beim positiv-kumulativen und beim negativ-selektiven Verfahren. Beim negativ-selektiven Verfahren ist es für den Sender problematisch festzustellen, wann eine gesendete Nachricht aus dem Puffer eliminiert werden kann. Der Sender weiß ja nie genau, ob die Nachrichten beim Empfänger angekommen sind oder nicht. Deshalb wird dieses Verfahren in seiner reinen Form auch selten verwendet.

In Abbildung 5-14 ist ein Go-Back-N-Verfahren skizziert, wobei eine selektiv, negative Quittierung (Empfänger meldet aktiv fehlende PDUs) angenommen wird. Die Data-PDU mit Nummer 2 wird von Instanz B angemahnt. Daraufhin werden von Instanz A die Data-PDUs mit der Nummer 2 und die ebenfalls bereits gesendete (aber angekommene) PDU mit Folgenummer 3 erneut gesendet.

Die Netzbelastung ist in langsamen Netzen relativ gering. Anders dagegen ist dies in Netzen mit hoher Pfadkapazität, da bei entsprechender Fenstergröße (siehe Flusskontrolle) viele PDUs gesendet werden können, bevor die erste Negativ-Quittung beim Sender eintrifft. Dies führt dann ggf. zur unnötigen Übertragungswiederholung vieler PDUs.

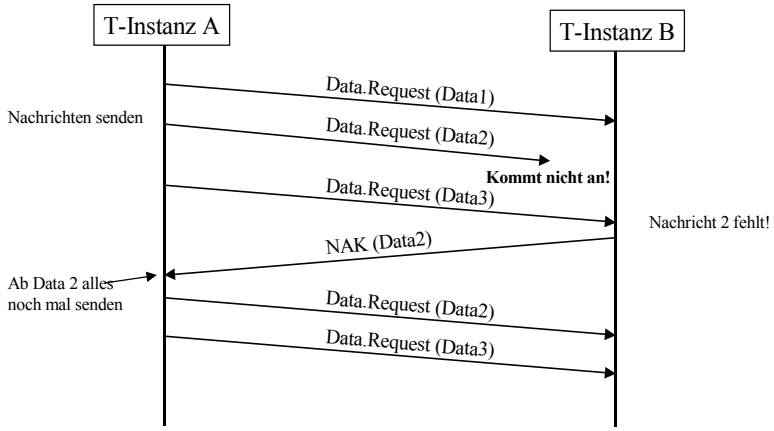


Abbildung 5-14: Negativ-selektive Quittung und Go-Back-N-Verfahren

Abschließend soll noch erwähnt werden, dass ein zuverlässiger Datentransfer noch keine Verarbeitungssicherheit und schon gar keine Transaktionssicherheit, gewährleistet. Zuverlässiger Datentransfer bedeutet „lediglich", dass die von einem Sendeprozess abgesendeten Nachrichten bei der T-Instanz des Empfängers ankommen, also dort in den Empfangspuffer eingetragen werden. Ob sie vom Empfängerprozess abgearbeitet werden oder gar zu den gewünschten Datenbankzugriffen führen, ist nicht gesichert. Hierfür werden höhere Protokolle, sog. Transaktionsprotokolle benötigt, die in der Anwendungsschicht platziert und noch komplexer als Transportprotokolle sind. Man findet Realisierungen von Transaktionsprotokollen in heutigen Datenbankmanagementsystemen.

5.1.4 Flusskontrolle

Unter Flusskontrolle versteht man in der Schicht 4 in etwa das Gleiche wie in der Schicht 2, mit dem Unterschied, dass man in der Schicht 4 eine Ende-zu-Ende-Verbindung ggf. über ein komplexes Netzwerk verwaltet, während die Schicht 2 eine Ende-zu-Ende-Verbindung zwischen zwei Rechnersystemen unterstützt. Letzteres ist wesentlich einfacher in den Griff zu bekommen.

Durch die Steuerung des Datenflusses soll eine Überlastung des Empfängers vermieden werden. Traditionelle Flusskontroll-Verfahren sind:

- *Stop-and-Wait-Verfahren:* Dies ist das einfachste Verfahren, wobei eine Kopplung von Fluss- und Fehlerkontrolle durchgeführt wird. Die nächste Nachricht wird erst nach einer erfolgreichen Quittierung gesendet (Fenstergröße 1).
- *Fensterbasierte Flusskontrolle* (Sliding-Window-Verfahren): Der Empfänger vergibt einen sog. *Sendekredit*, also eine max. Menge an Nachrichten oder Byte, die unquittiert an ihn gesendet werden dürfen. Der Sendekredit reduziert

sich bei jedem Senden. Der Empfänger kann den Sendekredit durch positive Quittungen erhöhen. Der Vorteil dieses Verfahrens ist, dass ein kontinuierlicher Datenfluss und höherer Durchsatz als bei Stop-and-Wait möglich ist.

Für die fensterbasierte Flusskontrolle werden in den Transportinstanzen für jeden Kommunikationspartner vier Folgennummern-Intervalle verwaltet, die grob in Abbildung 5-15 dargestellt sind:

- Das linke Intervall sind Folgenummern, die gesendet und vom Empfänger bereits bestätigt wurden.
- Das zweite Intervall von links stellt alle Folgenummern dar, die gesendet, aber vom Empfänger noch nicht bestätigt wurden. Der Zeiger *base* verweist auf den Anfang dieses Intervalls.
- Das nächste Intervall gibt alle Folgenummern an, die noch verwendet werden dürfen, ohne dass eine weitere Bestätigung vom Empfänger eintrifft. Der Zeiger *nextseqnum* zeigt auf den Anfang des Intervalls.
- Das vierte Intervall zeigt die Folgenummern, die noch nicht verwendet werden dürfen.

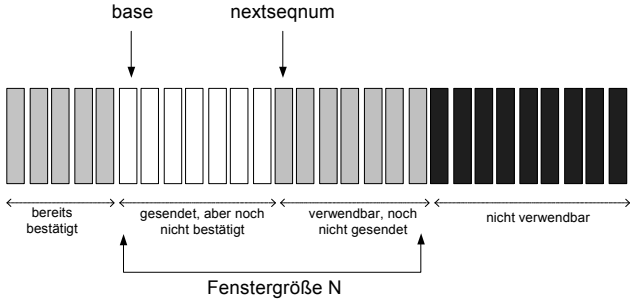


Abbildung 5-15: Intervallverwaltung für die fensterbasierte Flusskontrolle nach (Kurose 2002)

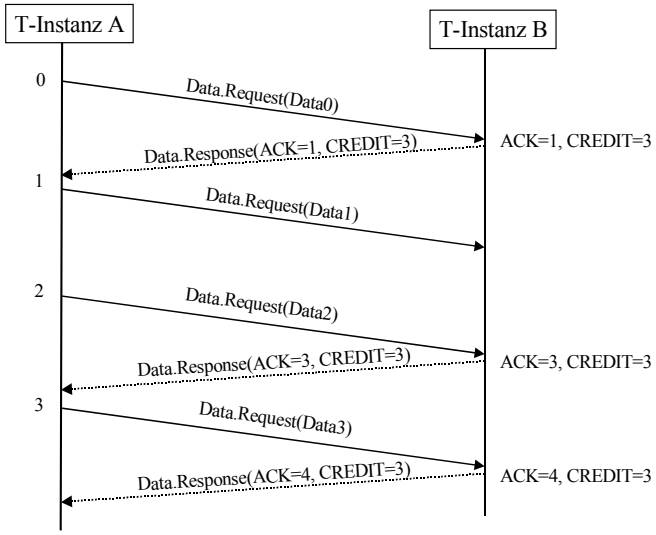
Die beiden inneren Intervalle bilden zusammen das aktuelle Fenster, geben also den verfügbaren Sendekredit an. Trifft nun eine Bestätigung des Empfängers ein, wandert das Fenster um die Anzahl der bestätigten Folgenummern nach rechts, d.h. der Zeiger *base* wird nach rechts verschoben. Gleichzeitig wird eine entsprechende Anzahl an Folgenummern aus den bisher nicht verwendbaren Folgenummern dem dritten Intervall zugeordnet. Der Zeiger *nextseqnum* wird nicht verändert. Erst wenn wieder Nachrichten gesendet werden, wird dieser entsprechend der Anzahl an benutzten Folgenummern nach rechts verschoben.

Die Größe des Sendekredits ist entscheidend für die Leistung des Protokolls. Zu große Kredite gewährleisten keinen richtigen Schutz der Ressourcen beim Empfänger. Eine zu starke Limitierung führt zu schlechterer Leistung. Die Größe des

Sendekredits hängt auch von der Pfadkapazität ab, also davon, wie viele Pakete im Netzwerk gespeichert werden können.

In Abbildung 5-16 ist ein Beispiel für eine fensterbasierte Flusskontrolle in einem Netz mit niedriger Pfadkapazität gezeigt. Der anfänglich bereitgestellte Sendekredit wird immer wieder schnell genug durch den Empfänger bestätigt, so dass es zu keinen Wartezeiten im Sender kommt.

Bei Netzwerken mit hoher Pfadkapazität führen kleine Sendekredite häufig dazu, dass der Sender durch das Warten auf Kreditbestätigungen blockiert wird. Dies ist in Abbildung 5-17 skizziert. Bei einem Sendekredit von drei entstehen beträchtliche Wartezeiten im Sender und auch im Empfänger. Die Instanz A hat bei kleinen Krediten häufig keinen Sendekredit, ist wieder einer da, entstehen Burst-Übertragungen (Übertragung von großen Mengen) in Fenstergröße.



Anfänglicher Sendekredit ist 3
Kumulative Quittierung
Kontinuierliches Senden von Daten möglich, da immer Sendekredit verfügbar ist

Abbildung 5-16: Fensterbasierte Flusskontrolle bei geringer Pfadkapazität nach (Zitterbart 1996)

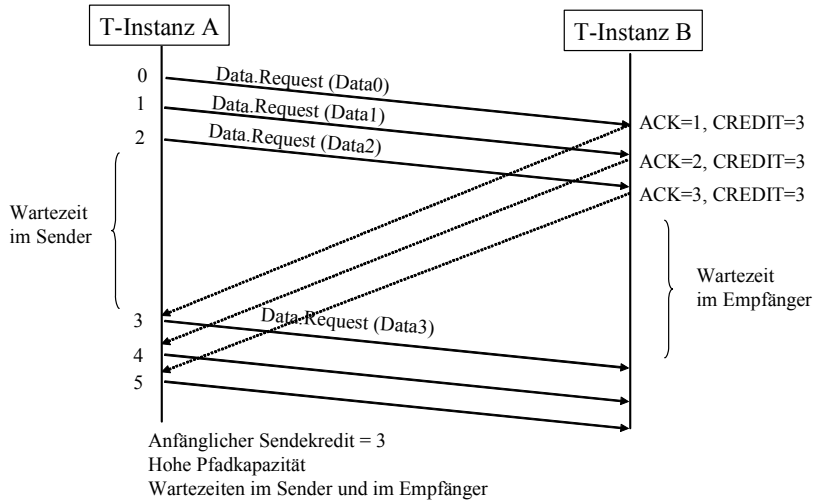


Abbildung 5-17: Fensterbasierte Flusskontrolle bei hoher Pfadkapazität nach (Zitterbart 1996)

5.1.5 Staukontrolle

Durch Staukontrolle (Congestion Control) sollen Verstopfungen bzw. Überlastungen im Netz vermieden werden. Maßnahmen zur Staukontrolle können in den Schichten 2 bis 4 durchgeführt werden und beziehen sich in der Schicht 4 überwiegend auf die Ende-zu-Ende-Steuerung zwischen Endsystemen.

Staukontrolle ist ein Mechanismus mit netzglobalen Auswirkungen. Einige Netzwerkprotokolle (wie ATM ABR) liefern am N-SAP gewisse Informationen, andere (wie das Internet-Protokoll) nicht.

In der Transportschicht hat man die Möglichkeit, aus Sicht einer bestimmten Ende-zu-Ende-Verbindung Maßnahmen zu ergreifen, wenn ein Engpass erkannt wird. Eine typische Maßnahme ist die Drosselung des Datenverkehrs durch Zurückhalten von Paketen.

Bei der Behandlung des Transportprotokolls TCP wird auf das Thema Staukontrolle am konkreten Protokollbeispiel eingegangen.

5.1.6 Multiplexierung und Demultiplexierung

In vielen Fällen werden von einem Host zu einem anderen mehrere Transportverbindungen für eine oder sogar mehrere Anwendungen benötigt. Auf der Schicht 3 genügt aber eine Netzwerkverbindung, um alle Transportverbindungen zu bedienen.

Um N-Verbindungen besser zu nutzen, kann die Transportschicht eine N-Verbindung für mehrere Transportverbindungen verwenden. Diesen Mechanismus nennt man Multiplexieren oder Multiplexen bzw. den umgekehrten Vorgang Demultiplexieren bzw. Demultiplexen.

Die T-Instanz hat die Aufgabe, den Verkehr, der über verschiedene T-SAPs läuft, an einen N-SAP zu leiten. Umgekehrt werden ankommende Pakete, welche die Schicht 3 an die Schicht 4 weiterleitet, entsprechend den einzelnen T-SAPs zugeordnet. Dies geschieht mit Hilfe der Adress-Information.

5.1.7 Fragmentierung/Segmentierung und Defragmentierung

Eine T-SDU, die an einem T-SAP übergeben wird, hat üblicherweise eine beliebige Länge. Eine T-PDU hat aber oft nur eine bestimmte Maximallänge, auch MSS (Maximum Segment Size) genannt. In diesem Fall muss die T-Instanz zu große T-SDUs in mehrere Segmente zerstückeln und einzeln übertragen. Diesen Vorgang nennt man Fragmentierung oder in der Schicht 4 auch Segmentierung.

Beim Empfänger ankommende Segmente, welche die N-Instanz einer T-Instanz liefert, müssen entsprechend wieder zusammengebaut werden, um eine vollständige T-SDU zu erhalten und nach oben weiterreichen zu können. Diesen Vorgang nennt man Defragmentierung oder Reassemblierung. Zuverlässige Transportprotokolle stellen sicher, dass auch keine Fragmente verloren gehen. Eine Schicht tiefer kann der gleiche Mechanismus übrigens nochmals angewendet werden.

Zur Optimierung versuchen leistungsfähige Protokolle jedoch, die Fragmentierung in Grenzen zu halten, da mit diesem Mechanismus viel Overhead verursacht wird.

5.2 Transmission Control Protocol (TCP)

5.2.1 Einordnung und Aufgaben

TCP ist ein Transportprotokoll und gibt der TCP/IP-Protokollfamilie seinen Namen. Es ermöglicht eine Ende-zu-Ende-Beziehung zwischen kommunizierenden Anwendungsinstanzen. TCP ist sehr weit verbreitet und als Industrienorm akzeptiert. Es ist ein offenes, frei verfügbares Protokoll und damit nicht an einen Hersteller gebunden. Neben UDP ist TCP das Transportprotokoll im Internet, auf das die meisten Anwendungen basieren.

Als Transportzugriffsschnittstelle dient die Socket-Schnittstelle. Ein T-SAP wird in TCP durch einen Port mit einer Portnummer identifiziert. Ein Anwendungsprozess benötigt also einen lokalen TCP-Port und kommuniziert über diesen mit einem anderen Anwendungsprozess, der ebenfalls über einen TCP-Port adressierbar ist. Wie in Abbildung 5-18 gezeigt, kann ein Anwendungsprozess durchaus mit mehreren anderen Anwendungsprozessen über den gleichen oder über verschiedene Ports kommunizieren.

TCP ist ein verbindungsorientiertes, zuverlässiges Protokoll. Die Protokollmechanismen von TCP werden in diesem Abschnitt beschrieben. Grob gesagt kümmert sich TCP um die Erzeugung und Erhaltung einer gesicherten Ende-zu-Ende-Verbindung zwischen zwei Anwendungsprozessen auf Basis von IP. TCP garantiert weiterhin die Reihenfolge der Nachrichten, und die vollständige Auslieferung, übernimmt die Fluss- und Staukontrolle und kümmert sich um die Multiplexierung sowie die Segmentierung.

TCP stellt also sicher, dass Daten

- nicht verändert werden,
- nicht verloren gehen,
- nicht dupliziert werden und
- in der richtigen Reihenfolge eintreffen.

TCP ermöglicht die Kommunikation über *vollduplex-fähige*, *bidirektionale* virtuelle Verbindungen zwischen Anwendungsprozessen. Das bedeutet, beide Kommunikationspartner können zu beliebigen Zeiten Nachrichten senden, auch gleichzeitig.

Wichtig ist auch, dass TCP am T-SAP eine Strom-orientierte Kommunikation (Stream) im Unterschied zur blockorientierten Übertragung (siehe OSI TP4) unterstützt. Daten werden also von einem Anwendungsprozess Byte für Byte in einen Bytestrom geschrieben. Die TCP-Instanz kümmert sich um den Aufbau von Segmenten, die übertragen werden. Dies bleibt für Anwendungsprozesse transparent. Andere Transportdienste erwarten ihre Daten in festen Blöcken.

Maßnahmen zur Sicherung der Übertragung sind die Nutzung von Prüfsummen, Bestätigungen, Zeitüberwachungsmechanismen mit verschiedenen Timern, Nachrichtenwiederholung, Sequenznummern für die Reihenfolgeüberwachung und das Sliding-Windows-Prinzip zur Flusskontrolle.

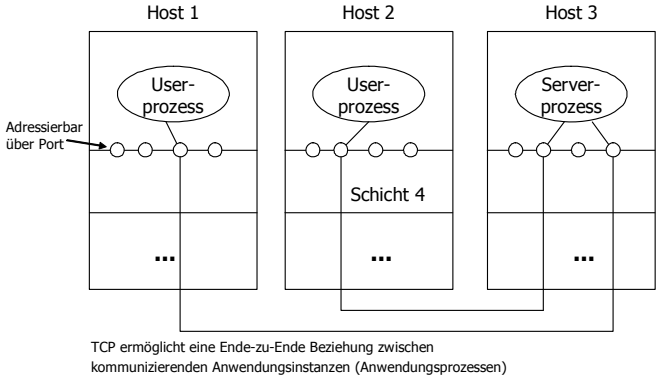


Abbildung 5-18: Ende-zu-Ende-Beziehung in TCP

TCP nutzt prinzipiell folgende Protokollmechanismen:

- Drei-Wege-Handshake-Verbindungsauf- und –abbau.
- Positives, kumulatives Bestätigungsverfahren mit Timerüberwachung für jede Nachricht.
- Implizites negatives Bestätigungsverfahren (NAK-Mechanismus): Bei drei ankommenden Duplikat-ACK-PDUs wird beim Sender das Fehlen des folgenden Segments angenommen. Ein sog. Fast-Retransmit-Mechanismus führt zur Neuübertragung des Segments, bevor der Timer abläuft.
- Go-Back-N zur Übertragungswiederholung.
- Fluss- und Staukontrolle.

5.2.2 TCP-Header

TCP sieht den Datenstrom (Stream) als eine Sequenz von Octets (Byte) und unterteilt diese zur Übertragung in Segmente. Ein Segment besteht aus einem mind. 20 Byte langen TCP-Header.

Die Begrenzung der Segmentlänge für jede Teilstecke ist aber durch die MTU (Maximum Transfer Unit) für jede Teilstrecke eines Netzes gegeben. In Abbildung 5-19 ist dargestellt, dass in der TCP-Schicht Segmente ausgetauscht werden, in der IP-Schicht sog. IP-Fragmente.

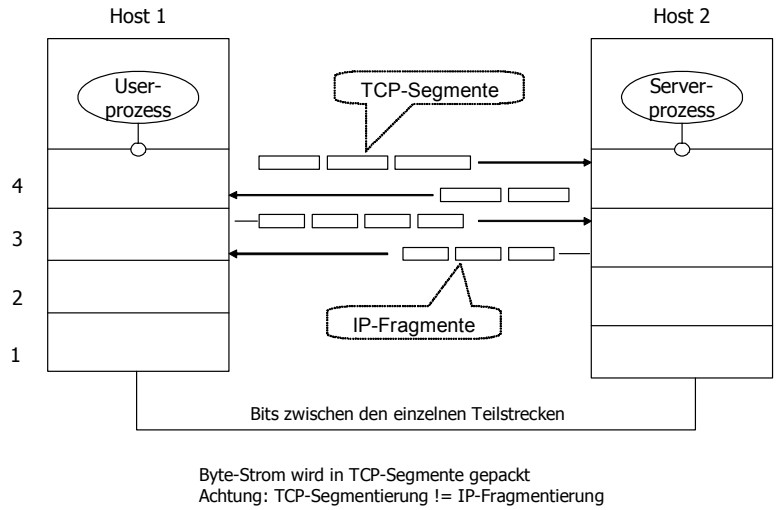
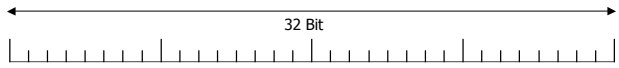


Abbildung 5-19: TCP-Segmente für den Datenaustausch

Der in der Regel 20 Byte lange TCP-Header (T-PCI) enthält die gesamte Steuerinformation, die TCP für den Verbindungsaufbau, den zuverlässigen Datentransfer und den Verbindungsabbau benötigt. Dazu können noch wahlweise bestimmte

Optionen kommen, was dazu führt, dass der Header auch länger als 20 Byte sein und damit auch eine variable Länge aufweisen kann. Im Anschluss an den Header werden bei TCP-Data-PDUs die Daten übertragen. Abbildung 5-20 zeigt den TCP-Header.



Quellport			Zielport
Folgenummer			
Bestätigungsnummer			
Offset	Res.	Flags	Zeitfenstergröße
Prüfsumme			Urgent-Zeiger
Optionen (0 oder mehrere 32-Bit-Wörter)			Padding
Daten (optional)			

Abbildung 5-20: TCP-Protokoll-Header (T-PCI)

Im Einzelnen enthält der TCP-Header folgende Felder:

- *Quell- und Zielport*: Portnummer des Anwendungsprogramms des Senders (Quelle) und des Empfängers (Ziel).
- *Folgenummer*: Nächstes Byte innerhalb des TCP-Streams, das der Sender absendet.
- *Bestätigungsnummer*: Gibt das als nächstes erwartete Byte im TCP-Strom des Partners an und bestätigt damit den Empfang der vorhergehenden Byte.
- *Offset*: Gibt die Länge des TCP-Headers in 32-Bit-Worten an. Dies ist nötig, da der Header aufgrund des variablen Optionenfeldes keine fixe Länge aufweist.
- *Reserviert (Res.):* Hat noch keine Verwendung.
- *Flags*: Steuerkennzeichen für diverse Aufgaben.
- *Zeitfenstergröße*: Erlaubt es einem Empfänger, in einer ACK-PDU seinem Partner den vorhandenen Pufferplatz in Byte zum Empfang der Daten mitzuteilen (auch Window-Size).
- *Prüfsumme*: Verifiziert das Gesamtpaket (Header+Daten) auf Basis eines einfachen Prüfsummenalgorithmus.
- *Urgent-Zeiger*: Beschreibt die Position (Byteversatz ab der aktuellen Folgenummer), an der dringliche Daten vorgefunden werden. Diese Daten werden vorrangig behandelt. Der Zeiger wird allerdings in der Praxis selten genutzt.

- *Optionen*: Optionale Angaben zum Aushandeln bestimmter Verbindungsparameter.
- *Padding*: Aufüllen auf Wortgrenze, wenn das Optionsfeld kleiner als ein Vielfaches von 4 Byte ist.
- *Daten*: Nutzlast, die auch fehlen kann.

Die *Folgenummer* und die *Bestätigungsnummer* dienen der Flusskontrolle und der Synchronisation zwischen Sender und Empfänger bzgl. der übertragenen Daten. Die TCP-Flusskontrolle wird noch behandelt.

Die Flags im TCP-Header haben eine besondere Bedeutung. Diese sind in Tabelle 5-1 beschrieben. Ursprünglich waren es sechs Flags, in neueren TCP-Implementierungen werden bereits acht Flags unterstützt. Hinzu kamen mit RFC 2481 die Flags CWR und ECN.[2]

Tabelle 5-1: TCP-Flags

Flag (1 Bit)	Bedeutung
CWR	Nutzung für die explizite Staukontrolle (Congestion Window Reduced).
ECE	Nutzung für die explizite Staukontrolle (ECN-Echo).
URG	Kennzeichen, das angibt, dass das Urgent-Zeiger-Feld gefüllt ist.
ACK	Bestätigung (z.B. bei Verbindungsaufbau genutzt), d.h. die Bestätigungsnummer hat einen gültigen Wert.
PSH	Zeigt Push-Daten an. Ankommende Daten dürfen beim Empfänger nicht zwischengespeichert werden, sondern sind sofort an den Empfängerprozess weiter zu leiten.
RST	Dient zum - Rücksetzen der Verbindung (sinnvoll z.B. nach dem Absturz eines Hosts). Es ist keine Kontextinformation mehr verfügbar. - Abweisen eines Verbindungsaufbauwunsches - Abweisen eines ungültigen Segments
SYN	Wird genutzt beim Verbindungsaufbau. Der Host, der die Verbindung aufbaut, sendet zunächst eine SYN-PDU.
FIN	Wird genutzt beim Verbindungsabbau. Der Host, der die Verbindung abbauen möchte, sendet als erstes eine FIN-PDU.

Das Feld *Prüfsumme* dient der Überprüfung des TCP-Headers, der Daten und eines Pseudoheaders, der in Abbildung 5-21 skizziert ist. Alle 16-Bit-Wörter werden in ihr Einer-Komplement umgewandelt und anschließend addiert.

[2] Einen ECN-Mechanismus gibt es nun ebenso für IP zum Austausch von Stauinformationen zwischen Routern (kurz erwähnt in Kapitel 4).

Der *Pseudoheader* enthält die IP-Adresse des Quell- und des Zielrechners, die Protokollnummer (Nummer 6 = TCP) und die Länge der TCP-PDU in Byte. Sinn und Zweck des Protokollheaders ist es, beim Empfänger fehlgeleitete PDUs zu erkennen. Dies kann die empfangende TCP-Instanz anhand der IP-Adresse prüfen. Informationen des Schicht-3-Protokolls werden in der Schicht 4 verwendet, was allerdings der Protokollschichtung widerspricht. Zudem kann die Angabe der Länge der TCP-PDU genutzt werden, um festzustellen, ob die ganze PDU beim Empfänger angekommen ist.

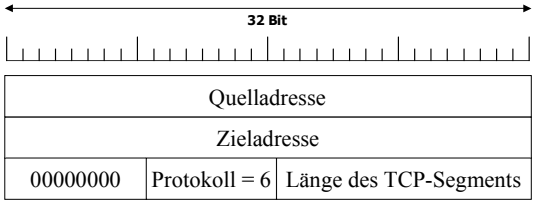


Abbildung 5-21: TCP-Pseudo-Header

Optionen. Die Optionen wurden relativ selten verwendet, aber mittlerweile dienen sie dem Aushandeln einiger wichtiger TCP-Funktionen. Sie bestehen entweder aus einem einzelnen Byte (Optionsnummer) oder haben eine variable Form. Jede zulässige Option ist mit einer Typangabe, einem Längenfeld und den eigentlichen Optionsdaten gekennzeichnet (siehe Abbildung 5-22).

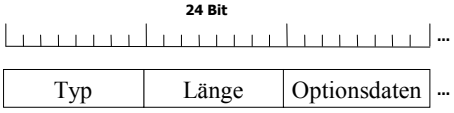


Abbildung 5-22: Grundlegender Aufbau des TCP-Optionsfeldes

Optionen werden in den entsprechenden RFCs mit Abkürzungen bezeichnet. Zu den Optionen gehören u.a. MSS, WSOPT, SACK, SACKOK und TSOPT:

– *Maximum Segment Size (MSS)*, Typ = 2, die Optionsdaten sind 4 Byte lang (RFC 793): Diese Option dient für das Aushandeln der maximalen Segmentlänge beim Verbindungsaufbau (SYN-Flag muss gesetzt sein). Für die MSS-Angabe stehen 16 Bit zur Verfügung, d.h. ein Segment kann maximal 64 KB groß sein. Beide Hosts übermitteln beim Verbindungsaufbau ihr Maximum und der kleinere der beiden Werte wird genommen. Als grundlegende Vereinbarung gilt, dass alle Hosts TCP-Segmente von 536+20 Byte akzeptieren müssen. Wird also nichts weiter vereinbart, gilt diese MSS für die Verbindung.

- *Windows-Scale-Option* (WSOPT), Typ = 3, die Optionsdaten sind 3 Byte lang (RFC 1323): Diese Option dient zum Aushandeln der maximalen Fenstergröße für das Sliding-Window-Verfahren (RFC 1323). Für Leitungen mit hoher Bandbreite sind nämlich 64 KB (siehe 16 Bit im Feld *Zeitfenstergröße*) als maximale Fenstergröße zu klein, da ein Sender dann evtl. zu lange warten muss, bis er erneut senden darf. Die Fenstergröße kann auf 2^{30} Bit (= 1 GB) skaliert werden, in dem der Inhalt des Feldes *Zeitfenstergröße* um max. 14 Bit nach links verschoben wird. Der Skalierungsfaktor wird in dieser Option übertragen (siehe Abbildung 5-23). Die Realisierung der Skalierung erfolgt durch Linksshift des Feldes *Zeitfenstergröße* um so viele Bit wie im Skalierungsfaktor angegeben werden (max. 14). Das Aushandeln der Fenstergröße kann nur beim Verbindungsaufbau (SYN-PDUs) erfolgen und ist für beide Senderichtungen möglich. Die Begrenzung des Skalierungsfaktors auf max. 14 Bit hat mit dem Wertebereich der Sequenznummer zu tun. Die Fenstergröße muss kleiner als der Abstand zwischen linkem und rechtem Ende des Sequenznummernbereichs sein.
- Zwei Optionen für die *selektive Wiederholung* (SACKOK und SACK), Typ = 4 und 5, Optionsdaten sind bei SACKOK 2 Byte und bei SACK variabel lang (RCF 2018): Anstelle des Go-Back-N-Verfahrens kann selektive Wiederholung für die Verbindung genutzt werden. Diese Vorgehensweise wird mit der Option SACKOK festgelegt. Mit der Option SACK kann man erlauben, dass in einer ACK-PDU eine Liste von Sequenznummernbereichen übergeben wird, die bereits empfangen wurden. Die Liste ist variabel lang und enthält zur Angabe der bestätigten Nachrichten Nummernpaare. Die erste Nummer eines Paares gibt dabei die erste Sequenznummer des Bereichs an, die empfangen wurde. Die zweite Nummer gibt die erste nicht empfangene Sequenznummer an.
- *Timestamps Option* (TSOPT), Typ = 8, die Optionsdaten sind 10 Byte lang: Dieses Feld besteht aus zwei Zeitstempeln zu je 4 Byte: Einem sog. Timestamp (TSval) und Timestamp-Echo-Reply (TSecr). Letzteres Feld ist nur bei ACK-Segmenten erlaubt. Mit diesen Werten können sich die TCP-Instanzen einer Verbindung über die sog. Round-Trip-Time (RTT) informieren. Man kann also damit die Zeit messen, die benötigt wird, um eine TCP-PDU zu senden und um die Bestätigung zu erhalten, dass sie beim Empfänger angekommen ist. Eine TCP-Instanz kann z.B. mit einer SYN-PDU oder einer Data-PDU den TSval-Wert setzen. Die Partner-TCP-Instanz antwortet immer auf die letzte Anforderung innerhalb einer ACK-PDU mit einem TSecr-Wert. Damit tauschen beide Instanzen ihre Timestamps zum Absendezeitpunkt aus.

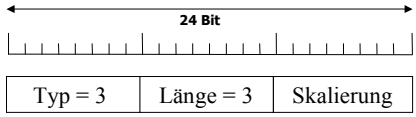


Abbildung 5-23: Aufbau der WSOPT-Option

Wichtige Felder werden in der weiteren Beschreibung der TCP-Protokollmechanismen ausführlich im Zusammenhang beschrieben.

5.2.3 Adressierung

Anwendungsprozesse sind über Transportadressen adressierbar, die als *Sockets* bezeichnet werden. Ein Socket ist ein Tupel der Form (IP-Adresse, TCP-Portnummer). Meist erfolgt die Kommunikation in einem Client-/Server-Paradigma. Ein Serverprozess stellt einen Dienst bereit und bietet ihn über eine Portnummer an. Ein Port (Abk. für Portnummer) ist eine 16-Bit-Integerzahl. Wenn ein Clientprozess die IP-Adresse und die Portnummer des Dienstes sowie das zugehörige Protokoll kennt, kann er mit dem Server kommunizieren und dessen Dienste nutzen.

IANA[3] verwaltet die Ports und definiert sog. *well-known* (wohlbekannte) Ports mit einem Nummernbereich von 0 bis 1023, *registrierte* Ports (Nummernbereich von 1024 bis 49151) und *dynamische* bzw. *private* Ports (Nummernbereich von 49152 bis 65535).

Es gibt eine Reihe von well-known Ports für reservierte Services. Diese sind innerhalb der TCP/IP-Gemeinde bekannt und werden immer gleich benutzt. Wichtige TCP-Ports (well-known Ports) und dazugehörige Dienste sind z.B.:[4]

- Port 23, Telnet (Remote Login)
- Ports 20 und 21, ftp (File Transfer Protocol)
- Port 25, SMTP (Simple Mail Transfer Protocol)
- Port 80, HTTP (Hyper Text Transfer Protocol)

Registrierte Ports werden durch IANA bestimmte Anwendungen fest zugeordnet. Die Hersteller der Anwendungen müssen die Zuordnung wie Domänennamen beantragen. Folgende Anwendungen sind z.B. als TCP-Ports registriert:

- Port 1109, Kerberos POP
- Port 1433, Microsoft SQL Server
- Port 1512, Microsoft Windows Internet Name Service (wins)

---

[3] IANA = Internet Assigned Numbers Authority.

[4] In Unix- und Windows-Systemen sind Services mit reservierten Ports meist in einer Konfigurationsdatei wie z.B. /etc/services festgelegt.

Dynamische bzw. private Ports können beliebig verwendet werden. Die Zuordnung wird von IANA nicht registriert. Mit diesen Portnummern ist es möglich, eigene Dienste zu definieren. Hier können in einem Netz auch Überlappungen auftreten. Beispielsweise kann ein Server 1 einen Dienst mit der Portnummer 52000 anbieten und ein Server 2 einen anderen Dienst mit derselben Portnummer 52000. Dies ist kein Problem, sofern die Anwendungsdomänen disjunkt sind. Auf einem Rechner kann ein Port aber nur einmal vergeben werden.

Eine Verbindung wird durch ein Paar von Endpunkten eindeutig identifiziert (Socket Pair). Dies entspricht einem Quadrupel, das die IP-Adresse von Host 1, die Portnummer, die in Host 1 genutzt wird, die IP-Adresse von Host 2 und die Portnummer auf der Seite von Host 2 enthält.

Dadurch ist es möglich, dass ein TCP-Port auf einem Host für viele Verbindungen genutzt werden kann, was in einigen Anwendungen auch intensiv genutzt wird.

Beispiel: Der HTTP-Port 80 wird für viele Verbindungen eines HTTP-Servers (hier im Beispiel mit der IP-Adresse 195.214.80.76) mit beliebig vielen Web-Clients (Browsern) verwendet. Mögliche TCP-Verbindungen aus Sicht des HTTP-Servers sind:

```
((195.214.80.76, 80) (196.210.80.10,6000))
((195.214.80.76, 80) (197.200.80.11,6001))
...
```

Man erkennt, dass serverseitig die Portnummer 80 für jede TCP-Verbindung mit Web-Clients genutzt wird und die Socket Pairs trotzdem alle eindeutig sind.

Wie schon angedeutet nutzt man zur Ermittlung der Schicht-3-Adresse (IP-Adresse), die Bestandteil der Transportadresse ist, einen Naming-Service wie DNS. In Programmen verwendet man sinnvollerweise symbolische Namen, die als Hostnamen bezeichnet werden.

5.2.4 Flusskontrolle

Bei der Einrichtung einer TCP-Verbindung sorgen die TCP-Instanzen beider Kommunikationspartner dafür, dass für die Verbindung jeweils Pufferbereiche für abzusendende und zu empfangende Daten eingerichtet werden. Aufgabe der Flusskontrolle ist es, dass der Empfangspuffer immer ausreichend ist. In Abbildung 5-24 ist ein Empfangspuffer skizziert. Die TCP-Instanz muss über dessen Befüllungsgrad Buch führen. Es ist eine der wichtigsten Aufgaben von TCP, eine optimale Fenstergröße für Transportverbindungen bereitzustellen.

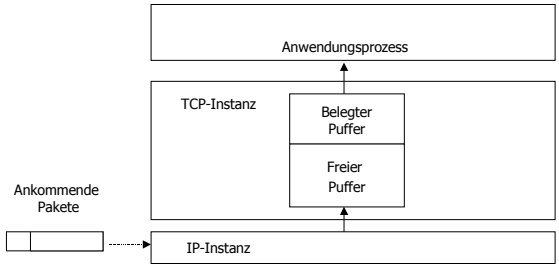


Abbildung 5-24: Empfangspuffer für TCP-Verbindungen

TCP verwendet für die Flusskontrolle einen Sliding-Window-Mechanismus. Dieser erlaubt die Übertragung von mehreren TCP-Segmenten, bevor eine Bestätigung eintrifft, sofern die Übertragung die vereinbarte Fenstergröße nicht überschreitet.

Bei TCP funktioniert Sliding-Window auf der Basis von Octets (Byte), worin sich auch der Streams-Gedanke widerspiegelt. Die Octets (Byte) eines Streams sind sequenziell nummeriert. Die Flusskontrolle wird beiderseits durch den Empfänger gesteuert, der dem jeweiligen Partner mitteilt, wie viel freier Platz noch in seinem Empfangspuffer ist. Der Partner darf nicht mehr Daten senden und muss das Senden eines Segments bei Bedarf verzögern.

Zur Fensterkontrolle wird im TCP-Header das Feld *Zeitfenstergröße* verwendet. In jeder ACK-PDU sendet eine TCP-Instanz in diesem Feld die Anzahl an Byte, die der Partner aktuell senden darf, ohne dass der Empfangspuffer überläuft. Wenn der Empfänger eine Nachricht mit einem Wert von 0 im Feld *Zeitfenstergröße* sendet, so bedeutet dies für den Sender, dass er sofort mit dem Senden aufhören muss. Erst wenn der Empfänger wieder ein Zeitfenster > 0 sendet, darf der Sender erneut Nachrichten senden.

In Abbildung 5-25 ist dies beispielhaft skizziert. Anfänglich ist der Empfangspuffer auf der Empfängerseite leer. In diesem Beispiel ist der Puffer 4 KB groß. In der TCP-Instanz auf der Empfängerseite läuft der Puffer voll, da der Anwendungsprozess die Daten vermutlich nicht oder zu langsam zur Verarbeitung abholt. Dies kann viele Gründe haben. Beispielsweise kann der Host zu stark ausgelastet sein oder ein Anwendungsprozess wartet auf ein Ereignis. Der Sender wird dadurch blockiert und wartet, bis der Empfänger wieder freien Pufferplatz meldet, in dem er eine zusätzliche ACK-PDU absendet, in der im Feld „Fenstergröße“ (WIN) 2 KB angegeben werden.

Die Flusskontrollmechanismen wurden mehrmals optimiert, und es gibt in der TCP-Dokumentation eine Reihe von RFCs mit Vorschlägen für Optimierungs-Algorithmen. Einige Besonderheiten, die auch ausführlich in der angegebenen Literatur erläutert sind, sollen hier kurz skizziert werden:

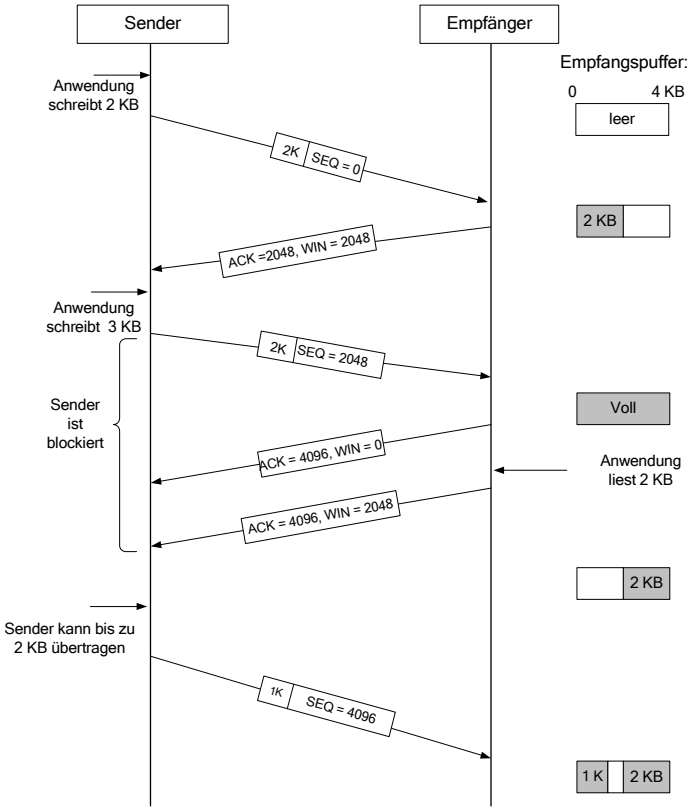


Abbildung 5-25: Sliding-Window-Mechanismus bei TCP nach (Tanenbaum 2003a)

Nagle-Algorithmus. Ein Algorithmus, der sich durchgesetzt hat, ist der *Nagle-Algorithmus* (RFC 896 und RFC 1122), der bei allen TCP-Implementierungen verwendet wird. Nagle versuchte aus Optimierungsgründen zu verhindern, dass viele kleine Nachrichten gesendet werden, da diese schlecht für die Netzauslastung sind. Kleine Nachrichten werden also bei Bedarf zusammengefasst und zwar nach folgendem Prinzip:

- Wenn vom Anwendungsprozess an der Socket-Schnittstelle die Daten im Stream Byte für Byte ankommen, wird zunächst nur das erste Byte gesendet und die restlichen werden im Sendepuffer gesammelt.
- Kommt dann die ACK-PDU für das eine Byte an, werden alle anstehenden Byte gesendet.
- Anschließend wird erneut gesammelt und so geht es weiter.

Dies ist allerdings nicht immer optimal und insbesondere schlecht bei Anwendungen mit direktem Echo der Dialogeingaben. Beispiele dafür sind X-Windows-

Anwendungen, da hier für jede Mausbewegung eine Übertragung stattfinden muss, um den Cursor auf dem Bildschirm zu bewegen. Geschieht dies nicht, bewegt sich der Cursor auf dem Bildschirm nicht synchron zur Bewegung der Maus und ggf. ruckartig. In diesem Fall ist es besser, abhängig vom Anwendungstyp den Nagle-Algorithmus auszuschalten. Weitere Beispielanwendungen für diese Problematik sind *SSH* und *Telnet*.

Ist dieses Verhalten für eine Anwendung problematisch, so lässt sich der Nagle-Algorithmus unter POSIX-kompatiblen Betriebssystemen mit der setsockopt-Option TCP_NODELAY abschalten. In der Praxis wird das zum Beispiel bei interaktiven Sitzungsprotokollen wie Telnet oder SSH getan, um die Reaktionszeit der Gegenseite auf Tastatureingaben oder bei Bildschirmausgaben zu verkürzen.

Silly-Window-Syndrom. Ein anderes Problem ist als das *Silly-Window-Syndrom* bekannt. Es tritt auf, wenn ein empfangender Anwendungsprozess die Daten byteweise ausliest, der Sender aber in größeren Nachrichten sendet. In diesem Fall wird der Empfangspuffer immer um ein Byte geleert, und die TCP-Instanz beim Empfänger sendet daraufhin eine ACK-PDU mit dem Hinweis, dass wieder ein Byte übertragen werden kann. Dies verhindert *Clarks* Lösung, indem es das Senden einer ACK-PDU bis zu einer vernünftigen Fenstergröße zurückhält. Die Mechanismen von Nagle und Clark ergänzen sich in einer TCP-Implementierung.

5.2.5 Datenübertragung

In diesem Abschnitt behandeln wir die normale Datenübertragung in TCP. Betrachten wir zunächst die Datenübertragung bei bereits vorhandener Verbindung.

Der Ablauf einer Übertragung eines Datensegments sieht, wie in Abbildung 5-26 dargestellt, bei TCP im Normalfall wie folgt aus:

- Der Sender stellt ein Segment zusammen, sendet es ab und zieht anschließend für jedes einzelne Segment einen Timer zur Zeitüberwachung auf. Die Zeit für den Timer wird dynamisch anhand der aktuellen RTT (Round Trip Time) ermittelt. Dies ist die Zeit, die die Ende-zu-Ende-Übertragung eines Segments einschließlich der Bestätigung dauert.
- Im Normalfall erhält der Empfänger das Paket und bestätigt es in einem kumulativen Verfahren. Damit wird versucht, die Netzlast zu reduzieren. Immer dann, wenn es möglich ist, gleich mehrere Segmente zu bestätigen, wird dies gemacht. Es wird sogar eine Verzögerung der Bestätigung unterstützt, um noch auf weitere Segmente zu warten.
- Der Sender erhält die ACK-PDU und löscht daraufhin den Timer.

Eine ACK-PDU zeichnet sich dadurch aus, dass das ACK-Flag gesetzt ist. Nun gibt es natürlich einige Fehlerszenarien, die TCP bearbeiten muss. Die wichtigsten werden kurz skizziert:

Szenario „Erfolgreiche Übertragung". In Abbildung 5-26 wird die Übertragung zweier Segmente dargestellt, die ordnungsgemäß (in diesem Falle jedes für sich) bestätigt werden. Bei Ankunft der Bestätigung wird der Timer in der Instanz 1 gelöscht, und die Übertragung ist damit abgeschlossen.

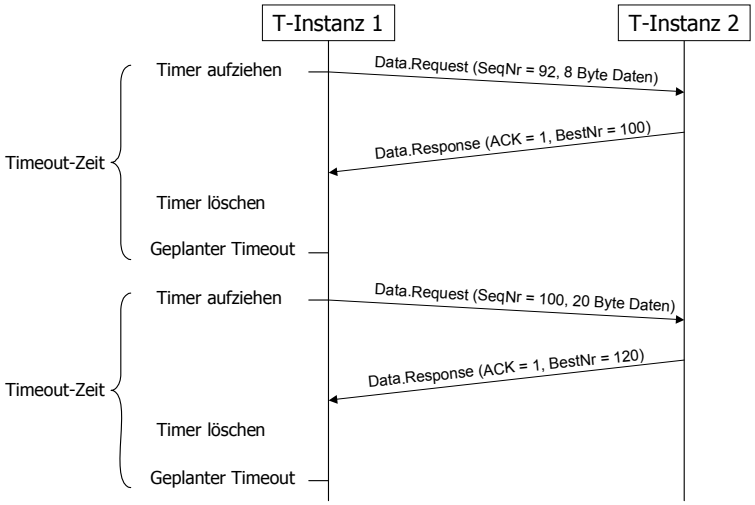


Abbildung 5-26: Normale Übertragung eines TCP-Segments

Szenario „Timer läuft ab". Läuft im Sender z.B. der Timer ab, bevor eine Bestätigung angekommen ist, wird das Segment noch einmal gesendet. Der genaue Grund, warum das Segment nicht bestätigt wurde, ist dem Sender nicht bekannt. Es kann sein, dass das Ursprungssegment verloren gegangen ist, es kann aber auch sein, dass die ACK-PDU nicht angekommen ist.

In Abbildung 5-27 läuft dagegen der Timer vor Ankunft der ACK-PDU in Instanz 1 ab. Die ACK-PDU geht im Netz verloren. Das Segment wird erneut übertragen, und die zweite Übertragung wird von Instanz 2 bestätigt. Die zweite Bestätigung trifft rechtzeitig ein.

Es sei hier nochmals darauf hingewiesen, dass ein Segment bei der Übertragung im Netz durchaus verloren gehen kann. Immerhin kann es sein, dass ein Router, durch den das Segment weitergeleitet werden soll, gerade überlastet ist. Außerdem kann eine Ende-zu-Ende-Kommunikation zwischen zwei Anwendungsprozessen über mehrere Router ablaufen. TCP stellt aber sicher, dass ein verlorengegangenes Segment noch einmal übertragen wird. Die empfangende TCP-Instanz gibt nur T-SDUs nach oben zum Anwendungsprozess weiter, wenn diese einen lückenlosen Datenstrom (Stream) ergeben.

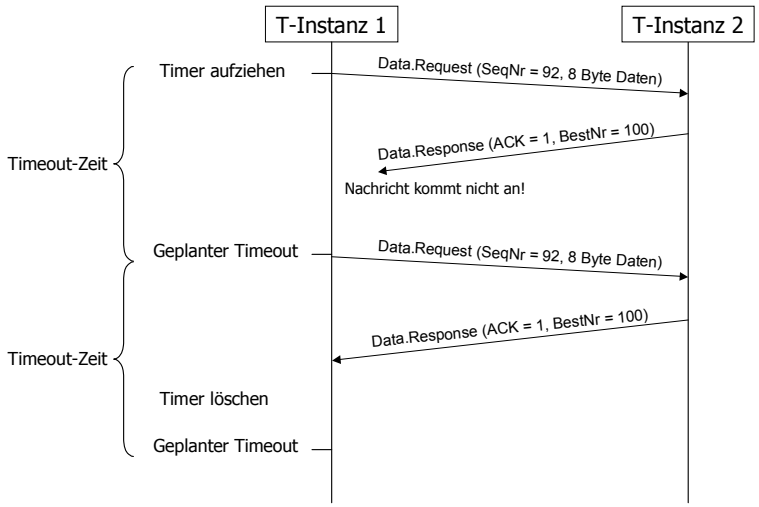


Abbildung 5-27: Timerablauf bei der Übertragung eines TCP-Segments

Szenario „Empfänger vermisst ein Segment". Ein sog. *implizites NAK* (Not Acknowledge) wird in TCP ebenfalls unterstützt. Wenn der Empfänger ein Segment vermisst, so sendet er die vorhergehende Bestätigung noch einmal. Erhält der Sender viermal die gleiche Bestätigung (also drei ACK-Duplikate)[5], so nimmt er an, dass auf das Segment folgende Segmente nicht angekommen sind und sendet diese vor Ablauf des Timers erneut (siehe Abbildung 5-28).

Das dreimalige Empfangen einer gleichen Bestätigung kann – je nach TCP-Implementierung – auch Auswirkungen auf die Staukontrolle haben. Dieser Vorgang wird als *Fast Retransmit* bezeichnet (siehe hierzu auch den mit dem *Fast Retransmit* gemeinsam behandelten Mechanismus, der als *Fast-Recovery* bezeichnet wird).

Szenario „Sender erhält kein ACK, aber Sendewiederholung wird durch kumulatives ACK vermieden". Für die Garantie der Reihenfolge und der Vollständigkeit wird bei TCP der Einsatz von Sequenznummern unterstützt, die im Sinne des Datenstrom-Konzepts auf einzelnen Bytes, nicht auf TCP-Segmenten basieren. Im TCP-Header wird hierfür das Feld *Folgenummer* verwendet. Die initiale Sequenznummer wird beim Verbindungsaufbau für beide Kommunikationsrichtungen festgelegt. Sie enthält jeweils die laufende Nummer des als nächstes erwarteten Byte im Stream der Verbindung.

[5] Siehe hierzu RFC 2581, früher waren es laut RFC 2001 nur zwei ACK-Duplikate.

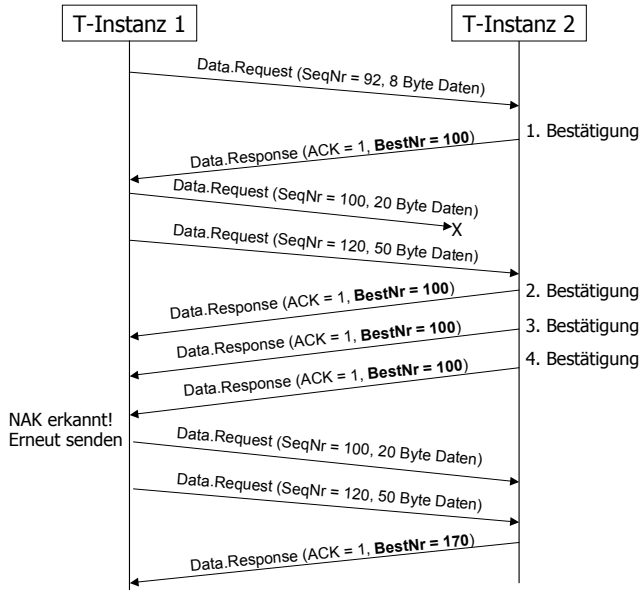


Abbildung 5-28: Empfänger sendet implizites NAK

Die Bestätigung der ordnungsgemäßen Ankunft eines Segments wird in der ACK-PDU über das Feld *Bestätigungsnummer* durchgeführt. In diesem Feld steht jeweils die als nächstes erwartete Sequenznummer des Partners, womit der Empfang aller vorhergehenden Byte im Stream bestätigt wird. Eine Bestätigung muss von der empfangenden TCP-Instanz nicht unbedingt sofort gesendet werden, sofern noch Platz im Empfangspuffer ist. Hier besteht eine gewisse Implementierungsfreiheit.

Eine Ausnahme bilden sog. „Urgent-Daten“, die immer gesendet werden können, auch wenn der Empfangspuffer voll ist, wobei diese allerdings in der Regel nicht verwendet werden.

Wie bereits angedeutet, werden TCP-Segmente wenn möglich kumulativ bestätigt. Dies hat den positiven Nebeneffekt, dass ggf. auch verloren gegangene ACK-PDUs durch folgende ACK-PDUs erledigt werden. Dies ist z.B. im Sequenzdiagramm aus Abbildung 5-29 deutlich zu sehen. In diesem Szenario geht eine ACK-PDU verloren und wird durch die folgende kompensiert.

In den RFCs 1122 und 2581 werden einige Implementierungs-Empfehlungen für die Quittierungsmechanismen von TCP-Segmenten gegeben. Einige Beispiele sollen hierzu erwähnt werden:

– Der Empfänger eines TCP-Segments sollte maximal 500 ms auf weitere Segmente warten, um diese kumulativ zu bestätigen.

- Kommt ein TCP-Segment außerhalb der Reihe mit einer Sequenznummer bei einer TCP-Instanz an, die höher ist als die erwartete, so soll ein Duplikat-ACK gesendet werden. Damit weiß der Sender sofort, dass er die fehlenden TCP-Segmente erneut senden muss.
- Kommt ein TCP-Segment bei einer TCP-Instanz an, das eine Lücke schließt, so wird dies sofort, ohne Verzögerung bestätigt, damit die sendende TCP-Instanz nicht behindert wird.

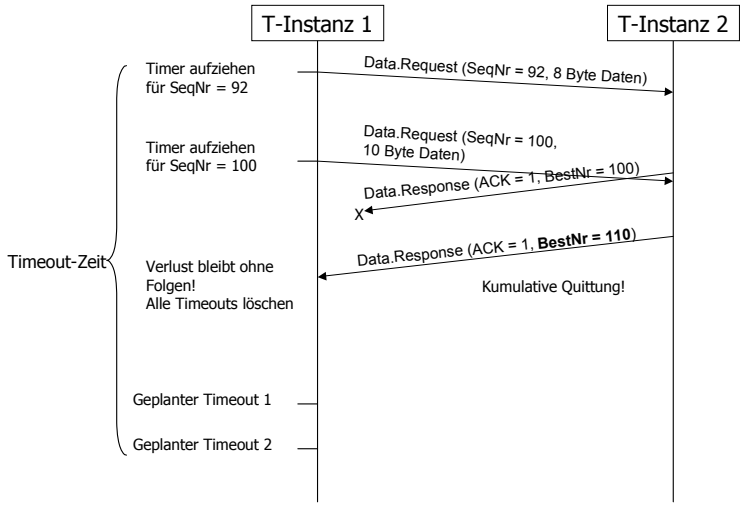


Abbildung 5-29 Kumulative Quittierung in TCP

5.2.6 Verbindungsmanagement

Bisher wurde vorausgesetzt, dass eine TCP-Verbindung existiert. Nun wird die Verbindungsaufbauphase näher betrachtet. Beim Verbindungsaufbau erfolgt eine Synchronisation zwischen einem aktiven Partner, der den Verbindungsaufbauwunsch absetzt, und einem passiven Partner, der auf einen Verbindungsaufbauwunsch wartet. Während des Verbindungsaufbaus werden einige Parameter ausgehandelt, die für die weitere Kommunikation als Basis dienen. Hierzu gehören die Größe der Flusskontrollfenster, die initialen Folgenummern für beide Partner und die Anzahl der Segmente, die im Rahmen der Staukontrolle anfänglich gesendet werden dürfen. TCP verwendet ein Drei-Wege-Handshake-Protokoll für den Verbindungsaufbau wie es vereinfacht in Abbildung 5-30 dargestellt ist:

- Der aktive Partner – als Instanz 1 bezeichnet – (im Sinne einer Client-/Server-Architektur auch als Client bezeichnet) beginnt den Verbindungsaufbau je nach Socket-Implementierung z.B. mit einem Connect-Aufruf. Die zuständige TCP-Instanz richtet daraufhin lokal einen Kontext ein und ermittelt eine initiale Folgenummer, die sie im Feld *Folgenummer* mit einer Connect-Request-PDU an den Server sendet. Diese PDU ist durch ein gesetztes SYN-Flag markiert. Im TCP-Header wird weiterhin der adressierte Port und der eigene Port eingetragen.
- Der passive Partner – als Instanz 2 bezeichnet – wartet schon auf einen Verbindungsaufbauwunsch, in dem er an der Socket-Schnittstelle einen *Listen*-Aufruf getätigt hat. Die zuständige TCP-Instanz auf der passiven Seite baut ebenfalls einen Kontext auf und sendet eine Connect-Response-PDU, in der das SYN-Flag und das ACK-Flag auf 1 gesetzt sind. Weiterhin wird die Folgenummer des passiven Partners dadurch bestätigt, das sie um 1 erhöht und in das Feld „Bestätigungsnummer" eingetragen wird. Dies bedeutet, dass das nächste erwartete Byte die ermittelte Folgenummer enthalten muss. Zudem berechnet die TCP-Instanz vor dem Absenden der *Connect-Response-PDU* ebenfalls eine Folgenummer für die von ihr gesendeten Daten und nimmt sie in die PDU im Feld *Folgenummer* mit auf. Zu beachten ist auch, dass der passive Partner die Portnummer noch verändern kann.
- Sobald der aktive Partner die ACK-PDU erhält, stellt er seinerseits eine ACK-PDU zusammen, in der das SYN-Flag auf 0 und das ACK-Flag auf 1 gesetzt sind und im Feld *Bestätigungsnummer* die um 1 erhöhte Folgenummer des aktiven Partners eingetragen wird. Das ACK kann auch im Piggypacking mit dem ersten anstehenden Datensegment gesendet werden.

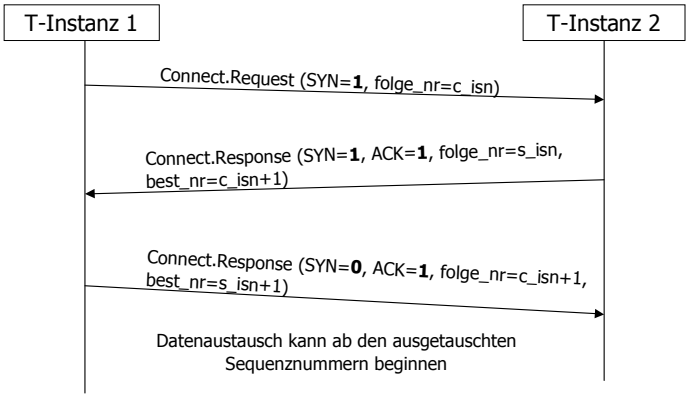


Abbildung 5-30: Normaler Verbindungsaufbau in TCP nach (Tanenbaum 2003a)

Sollte der passive Partner auf den Verbindungswunsch nicht antworten, wird in der Regel ein erneuter Connection-Request abgesetzt. Dies wird je nach Konfigurierung der TCP-Instanz mehrmals versucht, wobei eine dreimalige Wiederholung häufig anzutreffen ist.

Bei jedem Datenaustausch werden im weiteren Verlauf der Kommunikation die Felder *Quell-* und *Zielport*, die *Folgenummer* und die *Bestätigungsnummer* sowie das Feld *Fenstergröße* gesendet. Der aktive Partner sendet in der *Connect-Request-PDU* seine Portnummer als Quellport und den adressierten TSAP als Zielport im TCP-Header. Der passive Part kann seinen Port in der Connect-Response-PDU noch mal verändern. Quell- und Zielport werden jeweils aus Sicht des Senders belegt.

Beim Verbindungsaufbau sind Kollisionen möglich. Zwei Hosts können z.B. gleichzeitig versuchen, eine Verbindung zueinander aufzubauen. TCP sorgt aber dafür, dass in diesem Fall nur eine TCP-Verbindung mit gleichen Parametern aufgebaut wird. In Abbildung 5-31 ist der normale Verbindungsaufbau dem Verbindungsaufbau mit einer Kollision gegenübergestellt. Dieser Fall ist natürlich schon allein aufgrund der asymmetrischen Client-/Server-Kommunikation, wie sie meistens in TCP verwendet wird, recht unwahrscheinlich.

TCP sorgt auch für die Vermeidung einer weiteren, ebenfalls recht unwahrscheinlichen, Kollisionsart von Folgenummern aufeinander folgender Verbindungen zwischen gleichen Hosts. Bei TCP kann nämlich das für Sequenznummern typische Problem auftreten, dass eine Verbindung abgebrochen und zufällig gleich wieder mit den gleichen Ports eine Verbindung aufgebaut wird und darüber hinaus eine Nachricht der alten Verbindung noch unterwegs ist. In diesem Fall kann es zur Verwendung einer noch in der alten Verbindung genutzten Sequenznum-

mer kommen. TCP beugt hier vor, indem die Folgenummern Zeitgeber-gesteuert mit einem bestimmten Zyklus (ursprünglich 4,6 Stunden) ermittelt werden. Dies schließt derartige Kollisionen so gut wie aus.

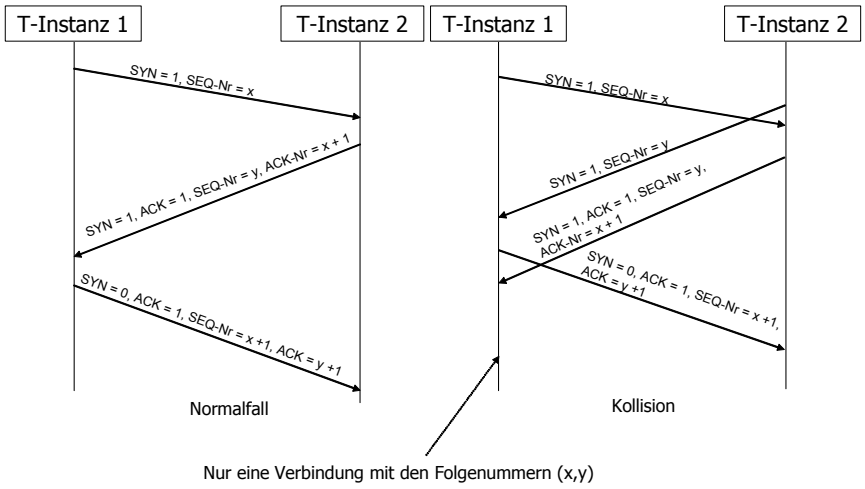


Abbildung 5-31: Kollision beim TCP-Verbindungsaufbau

Abschließend soll der Vollständigkeit halber noch erwähnt werden, dass ein Verbindungsaufbauwunsch vom passiven Partner auch sofort abgewiesen werden kann, wenn z.B. kein Prozess im Listen-Zustand ist. In diesem Fall sendet die passive TCP-Instanz eine Reset-PDU (mit gesetztem RST-Bit), und man kann in der Regel davon ausgehen, dass die Kommunikationsanwendung nicht richtig initialisiert ist oder aber ein größeres Problem in der TCP-Instanz vorliegt.

Sobald die Kommunikation, die theoretisch beliebig lange dauern kann, abgeschlossen ist, wird von einer Seite (egal von welcher) ein Verbindungsabbau initiiert. Dies geschieht an der Socket-Schnittstelle über einen *close*-Aufruf. Die initiierende Seite startet einen etwas modifizierten Drei-Wege-Handshake-Mechanismus. Jede der beiden Verbindungsrichtungen der Vollduplex-Verbindung wird abgebaut, d.h. beide Seiten bauen ihre „Senderichtung" ab. Der Ablauf ist in Abbildung 5-32 skizziert und verläuft wie folgt:

- Der aktiv abbauende Partner (T-Instanz 1) sendet zunächst ein Segment mit FIN-Flag = 1.
- Der passive Partner (T-Instanz 2) antwortet mit einem ACK und informiert die Anwendung über den Verbindungsabbauwunsch.

- Wenn die Partner-Anwendung die *close*-Funktion an der Socketschnittstelle aufruft, sendet die TCP-Instanz ein weiteres TCP-Segment mit gesetztem FIN-Flag.
- Der aktive Partner sendet abschließend ein Segment mit ACK-Flag = 1.

In der Abbildung sind die Zustandsübergänge in den Zustandsautomaten der beteiligten TCP-Instanzen dargestellt. Man sieht an den Zustandsübergängen, dass der Verbindungsabbau nicht trivial ist. Die Verbindung ist einseitig abgebaut, wenn eine Seite den Zustand CLOSE erreicht hat. In diesem Fall sendet sie keine Daten mehr über diese Verbindung. Haben beide Seiten den Zustand CLOSE erreicht ist die Verbindung vollständig abgebaut.

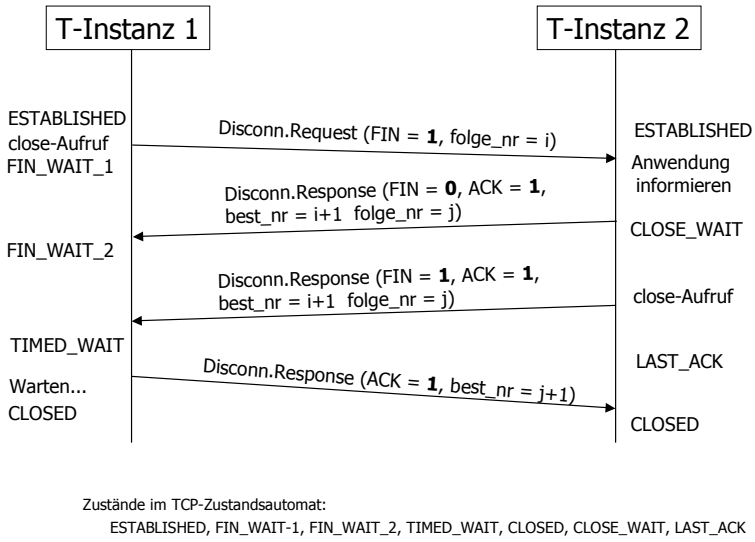


Abbildung 5-32 TCP-Verbindungsabbau

Eine Besonderheit des Verbindungsabbaus beim aktiven Partner ist, dass er nach allen Bestätigungen des gegenüberliegenden Partners die Verbindung noch nicht abschließt, sondern erst noch in den Zustand *TIMED_WAIT* geht. Hier wartet er eine bestimmte Zeit auf Pakete, die evtl. noch im Netz unterwegs sind, damit nichts verloren geht. Es könnte ja sein, dass ein Datensegment des Partners durch seine ACK-PDUs während des Verbindungsabbaus überholt wurde. Alle Segmente mit den Folgenummern < i bzw. < j müssen noch beim jeweiligen Zielhost ankommen. In diesem Zustand verweilt die TCP-Instanz eine doppelte Paketlebensdauer lang (siehe auch Timed-Wait-Timer). Nach Ablauf eines Timers wird der Zustand *CLOSE* eingenommen.

Laut Zustandsautomat gibt es mehrere Varianten des Verbindungsabbaus. Es geht auch mit drei Segmenten, wenn die passive Seite von der Anwendung sofort einen *close*-Aufruf erhält. Beim passiven Partner kann es aber eine Weile dauern, bis die Anwendung auf das *close*-Ereignis reagiert. Die Anwendung könnte ja evtl. sogar eine Benutzereingabe erfordern, was vom Programm abhängt.

Auch eine abnormale Beendigung einer Verbindung ist möglich. In diesem Fall wird ein TCP-Segment mit RST-Bit = 1 gesendet, und der Empfänger bricht die Verbindung sofort ab.

5.2.7 Staukontrolle

Im Jahre 1986 gab es im Internet massive Stausituationen. Aus diesem Grund suchte man für das Thema Staukontrolle (auch Congestion Control oder Überlastkontrolle) eine Lösung, die man auch ein paar Jahre später standardisierte. Seit 1989 gehört ein Mechanismus zur Staukontrolle zum TCP-Standard.

IP ist nicht in der Lage, der TCP-Schicht irgendwelche Informationen zur Netzauslastung nach oben zu geben. Also muss TCP selbst dafür sorgen, dass es Stausituationen wenn möglich präventiv erkennt und darauf reagiert. TCP kann dies nur auf Basis der einzelnen Verbindungen, also für jede Ende-zu-Ende-Beziehung separat. Staukontrolle darf nicht mit Flusskontrolle verwechselt werden. Während die Staukontrolle Netzprobleme zu vermeiden versucht, kümmert sich die Flusskontrolle darum, dass keine Empfangspuffer in den Endsystemen überlaufen.

Ein Paketverlust oder eine zu spät ankommende ACK-PDU wird von TCP als Auswirkung einer Stausituation im Netz interpretiert. Die Annahme von TCP ist, dass Netze prinzipiell stabil sind und daher eine fehlende Bestätigung nach dem Senden einer Nachricht auf ein Netzproblem zurückzuführen ist. In diesem Fall wird die Datenübertragung gedrosselt.

Das verwendete Verfahren wird als reaktives *Slow-Start-Verfahren* (Langsamstart-Verfahren, RFC 1122) bezeichnet, weil es mit einem kleinen, sog. Überlastfenster beginnt. Das Verfahren baut auf das Erkennen von Datenverlusten auf. Bestätigungen dienen als Taktgeber für den Sender. Die Übertragungsrate wird im Überlastfall bei diesem Verfahren vom Sender massiv gedrosselt. Die gesendeten Datenmengen werden kontrolliert. Im Gegensatz zur Flusskontrolle drosselt bei der Staukontrolle der Sender die Datenübertragung. Alle TCP-Implementierungen müssen das Slow-Start-Verfahren unterstützen.

TCP funktioniert nun so, dass es sich für jede Verbindung an die optimale Datenübertragungsrate herantastet. Die maximale Menge an Daten, die ohne Bestätigung gesendet werden darf, ist durch die Größe des sog. *Überlastfensters* bzw. Staukontrollfensters begrenzt, das nun zusätzlich zum Empfangsfenster der Flusskontrolle verwaltet wird. Es gilt:

```
Sendekredit für eine TCP-Verbindung =
min {Überlastungsfenster, Empfangsfenster}
```

Dies bedeutet, dass der Sendekredit einer Verbindung nicht größer als das Minimum aus der Größe des Überlastfensters und des Empfangsfensters (Fenstergröße der Flusskontrolle) sein darf.

Die Staukontrolle läuft, wie in Abbildung 5-33 zu sehen ist, in zwei Phasen ab:

- *Slow-Start-Phase* (nach dieser Phase ist das Verfahren benannt): Sender und Empfänger einigen sich beim Verbindungsaufbau auf eine erste zu sendende Anzahl an Segmenten mit der vereinbarten TCP-Segmentlänge (z.B. 1024 Byte).
- Der Sender sendet zunächst ein Segment dieser Länge. Kommt eine ACK-PDU hierfür rechzeitig, wird die Anzahl der Segmente, die gesendet werden dürfen (also das Überlastungsfenster) verdoppelt. Exakt sieht es so aus, dass jedes bestätigte Byte verdoppelt wird. Das geht bis zu einem Schwellwert so weiter, was zu einer exponentiellen Steigerung der Größe des Überlastungsfensters führt.
- Nach dem Erreichen des Schwellwerts (Grenzwert) geht das Verfahren in die sog. *Probing-* oder *Überlastvermeidungsphase* (Congestion Avoidance) über. In dieser Phase erhöht sich bei jeder empfangenen Quittung die Anzahl der zulässigen Segmente nur noch linear um 1.

Man sieht auch in der Abbildung, dass das Slow-Start-Verfahren gar nicht so langsam ist, sondern nur mit einem kleinen Überlastfenster beginnt. Die Steigerung der Überlastfenstergröße ist exponentiell, solange alles gut geht. Als Obergrenze gilt natürlich die Fenstergröße, die auch dynamisch ermittelt wird. Die Fenstergröße kann allerdings durch Nutzung der Option WSOPT (siehe TCP-Header) bis zu 1 GB groß sein. Aufgrund eines Engpasses auf der Empfangsseite kann immer eine Drosselung des Datenverkehrs initiiert werden (siehe hierzu das Thema Flusskontrolle).

Die Ermittlung der Größe des Sendekredits erfolgt immer in beiden TCP-Instanzen separat, also für beide Kommunikationsrichtungen, und kann natürlich auch voneinander abweichen.

Was passiert nun, wenn eine ACK-PDU nicht rechtzeitig eintrifft und ein Timeout abläuft? TCP geht davon aus, dass z.B. ein weiterer Sender im Netz hinzugekommen ist und sich die Verbindung mit diesem neuen Sender (und natürlich den schon vorhandenen Teilnehmern) die Pfadkapazität teilen muss, oder aber, dass irgendwo ein Netzproblem vorliegt. Als Reaktion darauf wird der Schwellwert auf die Hälfte des aktuellen Staukontrollfensters reduziert, die Segmentgröße wieder auf das Minimum heruntergesetzt, und das Verfahren beginnt von vorne, also wieder mit einem Slow-Start. Die Reaktion ist also massiv, die Last wird sofort stark reduziert.

Abbildung 5-33 zeigt ein Beispiel für eine Entwicklung des Überlastungsfensters. Die Ursprungsgröße des Überlastungsfensters ist hier ein TCP-Segment mit der Länge 1 KB. Der erste Schwellwert liegt bei 32 KB. Ab dieser Überlastfenstergröße

steigt das Fenster linear um jeweils eine MSS. Bei einer Größe des Überlastungsfensters von 40 KB tritt ein Timeout auf, die ACK-PDU bleibt also aus. Der Schwellwert wird dann auf 20 KB gesetzt und das Verfahren beginnt von vorne.

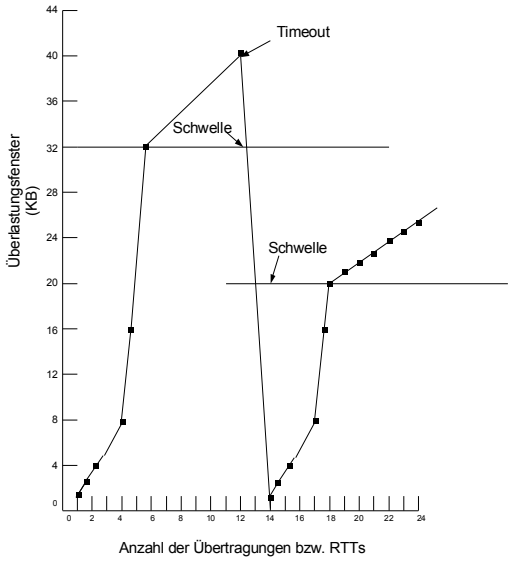


Abbildung 5-33: Staukontrolle bei TCP nach (Tanenbaum 2003a)

Man sieht an diesem Verfahren, dass auch hier die Timerlänge ganz entscheidend ist. Ist ein Timer zu lang, kann es zu Leistungsverlusten kommen; ist er zu kurz wird die Last durch erneutes Senden nochmals erhöht. TCP führt daher die Berechnung anhand der Umlaufzeit eines Segments dynamisch durch und versucht immer eine der Situation angemessene RTT zu ermitteln. Die Algorithmen zur Berechnung der Umlaufzeit sind im RFC 1122 festgehalten. Der Slow-Start-Algorithmus wird auch – zumindest in der Congestion-Avoidance-Phase – als AIMD-Algorithmus (Additive-Increase, Multiplicative-Decrease) bezeichnet, weil TCP die Übertragungsrate additiv erhöht, wenn der TCP-Pfad in Ordnung ist und multiplikativ senkt, wenn ein Datenverlust erkannt wird.

Ein ergänzendes Verfahren zur Staukontrolle in TCP ist der *Fast-Recovery-Algorithmus*, der den mehrfachen Empfang von Bestätigungen für eine TCP-PDU nutzt (RFC 2581). Wenn der Sender vier Quittierungen (drei ACK-Duplikate) für eine TCP-PDU empfängt, geht er von einem Paketverlust aus und veranlasst die sofortige Sendewiederholung der PDU. Die Sendeleistung wird entsprechend angepasst. Der Schwellwert wird auf die Hälfte des aktuellen Staukontrollfensters reduziert. Da nur von einem Paketverlust und nicht von einem Stau im Netzwerk

ausgegangen wird (die dem verlorengegangenen Segment folgenden Segmente sind ja im Puffer des Empfängers), wird das Staukontrollfenster auf einen Wert über dem Schwellwert gesetzt und es wird zudem mit der Stauvermeidungsphase fortgefahren.

Die Staukontrolle versucht man in letzter Zeit auch noch über einen anderen Mechanismus zu verbessern. Im RFC 2481 wurden zwei weitere Flags im TCP-Header vorgeschlagen, die für die Ende-zu-Ende-Signalisierung von Stauproblemen, genauer zur expliziten Stausignalisierung zwischen Endpunkten (ECN = Explicit Congestion Notification), verwendet werden sollen. Dies sind die Flags CWR und ECN. Während des Verbindungsaufbaus handeln hierfür die beiden Partner über die Flags CWR und ECN aus, ob sie für die Transportverbindung eine Ende-zu-Ende-Stausignalisierung einrichten. Der aktive TCP-Partner setzt bei der SYN-Nachricht, also in der Connect-Request-PDU, auch das ECE-Flag und das CWR-Flag.

Der TCP-Partner antwortet in der Connect-Response-PDU mit einer Bestätigung, in der das SYN-Flag, das ACK-Flag und das ECE-Flag gesetzt sind, nicht aber das CWR-Flag. Ist kein ECN gewünscht, dann wird vom Partner in der Antwortnachricht weder das CWR- noch das ECE-Flag gesetzt. Wie die Signalisierung der Stausituationen dann erfolgt, ist noch weitgehend in der Experimentierphase und soll daher nicht weiter ausgeführt werden.

5.2.8 Timer-Management

Die richtige Einstellung von Timeout-Werten ist in allen Protokollen, so auch in TCP, sehr wichtig für die Leistungsfähigkeit. Aus diesem Grund soll an dieser Stelle nochmals kurz auf die Timer-Problematik eingegangen werden. TCP verwendet z.B. folgende Timer:

- *Retransmission Timer*: Dieser Timer dient der Überwachung der Übertragung der TCP-Segmente nach dem sog. *Karn-Algorithmus* und wird für jede einzelne Übertragung verwendet. Wenn ein Timer abläuft, wird er verlängert und das TCP-Segment wird erneut gesendet (Übertragungswiederholung).
- *Keepalive Timer*: Dieser Timer wird verwendet, um zu überprüfen, ob ein Partner, der schon längere Zeit nichts gesendet hat, noch lebt. Läuft der Timer ab, überprüft eine Seite durch das Senden einer Nachricht, ob der Partner noch lebt. Kommt keine Antwort zurück, wird die Verbindung abgebaut.
- *Timed-Wait-Timer*: Dieser Timer wird für den Verbindungsabbau benötigt und läuft über die doppelte Paketlebensdauer (Siehe TCP-Zustandsautomaten). Der Timer soll sicherstellen, dass alle Pakete bei einem Verbindungsabbau noch übertragen werden.

Der für die Leistungsfähigkeit des Protokolls wichtigste Timer ist ohne Zweifel der Retransmission-Timer, der bei jedem Senden eines Segments angewendet wird. Er

wird gestoppt, wenn die ACK-PDU rechtzeitig ankommt, im anderen Fall wird das Segment erneut übertragen.

Wie in Abbildung 5-34 dargestellt, ist die optimale Länge eines Timers dann gegeben, wenn sich die Wahrscheinlichkeitsdichte der Ankunftszeiten von Bestätigungen wie in Bild (a) verhält. Dies ist aber in der Schicht 2 viel einfacher zu erreichen als in der Schicht 4. In Schicht 2 ist die erwartete Verzögerung gut vorhersehbar. In Schicht 4 ist dies weitaus komplizierter, da die Rundreisezeit (Round Trip Time, RTT) über mehrere Router gehen kann und sich dies auch noch dynamisch ändert. Eine Überlastung des Netzes verändert die Situation in wenigen Sekunden. Die Wahrscheinlichkeitsdichte der Ankunftszeiten von Bestätigungen ist bei TCP eher durch die Funktion in Bild (b) beschreibbar. Ein Timer-Intervall T1 führt hier zu unnötigen Neuübertragungen, und T2 führt zu Leistungseinbußen durch zu lange Verzögerungen.

Die RTT ist im Wesentlichen die Verzögerungszeit, also die Zeit, die benötigt wird, bis die Quittung für eine Nachricht empfangen wird. Diese Zeit wird auch als Roundtrip-Zeit bezeichnet. Mit Hilfe der RTT kann die Timeout-Berechnung erfolgen. Die RTT verändert sich aber dynamisch je nach Lastsituation. Aus diesem Grund wird bei TCP ein sehr dynamischer Algorithmus verwendet, der das Timeout-Intervall auf der Grundlage von ständigen Messungen der Netzlast immer wieder anpasst. Dieser Algorithmus geht im Wesentlichen von der Messung der RTT aus. Bei jedem Roundtrip wird die neue RTT aus der angesammelten RTT und der zuletzt gemessenen RTT ermittelt. Die Werte gehen über einen Glättungsfaktor, der typischerweise bei 7/8 liegt, in die Berechnung ein.

Aus diesem Wert wird dann nach einem Berechnungsschema, das auf *Jacobson* und *Karn* zurückzuführen ist und in der Literatur nachgelesen werden kann, ein neuer Timeout-Wert ermittelt.

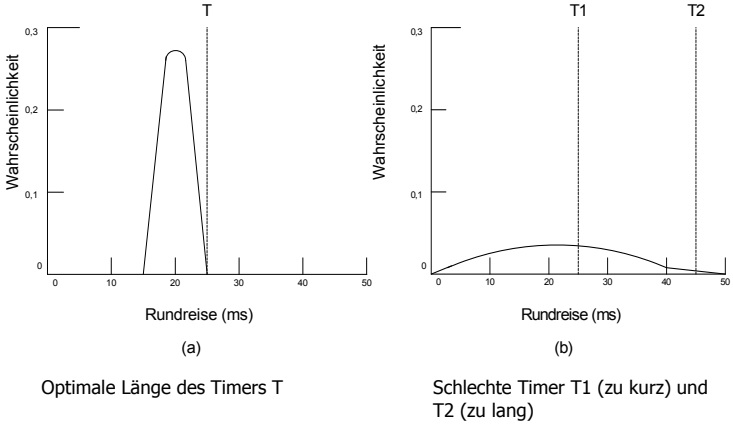


Abbildung 5-34: Timerproblematik bei TCP nach (Tanenbaum 2003a)

Gehen Datenpakete verloren, so müssen sie erneut gesendet werden. RTT wird von der letzten Wiederholungssendung bis zum ACK ermittelt, wodurch die RTT zu niedrig angesetzt wird. Neuere Implementierungen ziehen daher bei der Ermittlung von RTT die wiederholten Datenpakete nicht mehr mit ein und haben die Bestimmung des Timeouts weiter optimiert (Tanenbaum 2003a).

5.2.9 TCP-Zustandsautomat

Wie bereits mehrfach erwähnt, kann jedes Protokoll durch einen Zustandsautomaten dargestellt werden. Heute verwendet man zur Protokollspezifikation erweiterte endliche Automaten (Extended Finite State Machines, EFSM) oder Communicating Extended Finite State Machines (CEFSM). Protokollinstanzen werden als endliche Automaten der Klasse CEFSM beschrieben. Als formale Beschreibungssprache, welche CEFSM nutzt, dient z.B. SDL (Specification and Description Language)[6].

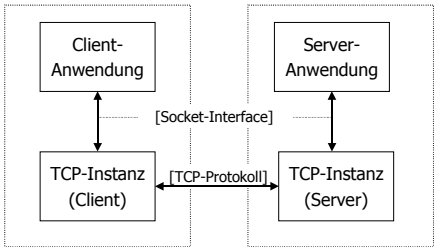


Abbildung 5-35: Vereinfachtes Systemmodell für das TCP-Protokoll

Wir wollen hier lediglich eine grobe Darstellung des TCP-Protokollautomaten geben. Ein vereinfachtes Systemmodell ist in Abbildung 5-35 dargestellt. Auf der Client-Seite kommuniziert eine Client-Anwendung lokal mit einer TCP-Instanz über ein Socket-Interface. Auf der Serverseite verhält es sich ähnlich. Die beiden TCP-Instanzen kommunizieren über TCP miteinander. Selbstverständlich sind in einer vollständigen Protokollspezifikation auch noch andere Ereignisquellen wie z.B. ein Zeitgeber für die Timerbearbeitung notwendig. Dies wird für unsere Betrachtung jedoch vernachlässigt.

Für alle Kommunikationskomponenten ist jeweils ein Protokollautomat zu spezifizieren. Wir zeigen im Folgenden (Abbildung 5-36) den vereinfachten Protokollautomaten für die TCP-Instanz (zusammengefasst für beide Kommunikationsseiten). Die fetten, durchgezogenen Linien in der Abbildung zeigen die normalen Zustandsübergänge im Client, die fett gestrichelten Linien die normalen Zustands-

[6] Eine sehr gute Einführung zur Protokollspezifikation mit SDL ist in (Gerdsen 1994a) zu finden.

übergänge im Server an. Die fein gezeichneten Linien deuten Spezialübergänge an. Die Notation des Automaten ist wie folgt zu lesen:

- Die Kästen repräsentieren Zustände.
- Ein Zustandsübergang wird durch eine beschriftete Kante dargestellt.
- Die Beschriftung am Zustandsübergang gibt das auslösende Ereignis (z.B. eine ankommende PDU, den Aufruf einer Funktion des Anwendungsprozesses oder ein Timeout) an. Nach einem folgenden Schrägstrich wird die beim Zustandsübergang ausgeführte Aktion angegeben (z.B. ACK für das Senden einer ACK-PDU oder nur ein Minuszeichen, wenn nichts getan wird).
- Je nach aktueller Rolle eines TCP-Partners werden unterschiedliche Zustände durchlaufen. Einige der Zustände werden nur im aktiven Partner verwendet. Hierzu gehören die Zustände *FIN_WAIT_1*, *FIN_WAIT_2*, *CLOSING* und *TIMED_WAIT*, die für den aktiven Verbindungsabbau genutzt werden. Andere werden nur im passiven Partner verwendet. Hierzu gehören z.B. die Zustände *LISTEN* beim passiven Verbindungsaufbau sowie *CLOSED_WAIT* und *LAST_ACK* beim passiven Verbindungsabbau.

Interessant sind vor allem die Zustandsübergänge beim Verbindungsauf- und abbau. Clientseitig (also im aktiven Partner) wird durch das Senden einer Connect-Request-PDU (SYN) in den Zustand SYN_SENT übergegangen. Dies ist der erste Schritt im Drei-Wege-Handshake. Nach Empfang eines SYN und eines ACK (ACK-PDU) wird ebenfalls ein ACK gesendet und der Zustand auf ESTABLISHED gewechselt. Die Verbindung ist nun auf der Clientseite eingerichtet. Auf der Serverseite (passiver Partner) spielt sich unterdessen Folgendes ab: Der Server befindet sich ursprünglich im Zustand LISTEN, d.h. er wartet auf ankommende Verbindungswünsche. Trifft eine Connection-Request-PDU ein (SYN-Bit gesetzt), antwortet der Server mit SYN+ACK (ACK-PDU) und geht in den Zustand SYN_RCVD. Nach dem erneuten Empfang einer ACK-PDU wird der Zustand auf ESTABLISHED gesetzt. Damit ist auch die Verbindung auf der Serverseite eingerichtet, und die Datenübertragung kann in diesem Zustand beginnen. Der Zustand ESTABLISHED wird nur im Fehlerfall (im vereinfachten Zustandsautomaten nicht dargestellt) oder beim Beenden der Verbindung verlassen.

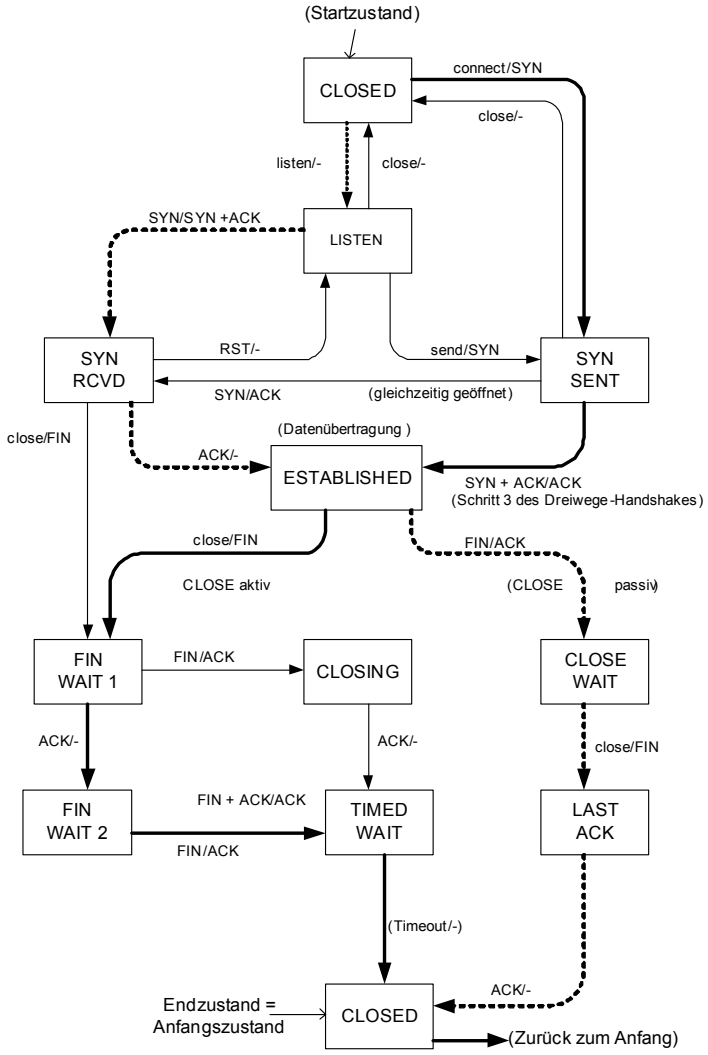


Abbildung 5-36: Vereinfachter TCP-Protokollautomat nach (RFC 793)

Der Verbindungsabbau ist noch etwas komplizierter, aber auch hier spiegelt sich der Drei-Wege-Handshake wider. Ein aktiver Partner (aktiv im Sinne des Verbindungsabbaus) muss den Verbindungsabbau initiieren. Der Anwendungsprozess leitet den Verbindungsabbau durch Aufruf der *close*-Funktion ein. Im Normalfall geht der aktive Partner nach dem Senden einer Disconnect-Request-PDU (FIN-Bit = 1) in den Zustand FIN_WAIT1. Schon in diesem Zustand wird keine weitere

Data-PDU mehr gesendet, es dürfen aber noch welche empfangen werden. Der passive Partner empfängt eine Disconnet-Request-PDU, bestätigt diese und meldet der Anwendung den gewünschten Verbindungsabbau. Die Anwendung hat nun Zeit auf den Verbindungsabbau zu reagieren und bestätigt den Verbindungsabbau mit der close-Funktion. TCP sendet daraufhin auch an den aktiven Partner eine Disconnect-Request-PDU und wechselt in den Zustand LAST_ACK, um auf die letzte Bestätigung zu warten. Der aktive Partner wechselt schon nach der erhaltenen Bestätigung in den Zustand FIN_WAIT2 und wartet auf den Verbindungsabbauwunsch. Nach dem Erhalt der Disconnect-Request-PDU und dem Senden einer erneuten ACK-PDU wird im aktiven Partner in den Zustand TIMED_WAIT übergegangen. In diesem Zustand wird der Timed-Wait-Timer aufgezogen und zwar auf die doppelte Paketlebensdauer. Erst nach Ablauf des Timers wird der Zustand CLOSED eingenommen, und damit ist die Verbindung beendet. Der passive Partner wechselt sofort nach dem Erhalten der letzten Bestätigung in den Zustand CLOSED.

Beim Verbindungsabbau werden insgesamt vier Nachrichten ausgetauscht. Es handelt sich somit um einen modifizierten Drei-Wege-Handshake, da der Server (passive Partner) mit zwei Nachrichten auf den Verbindungsabbauwunsch reagiert.

Der Grund für den Zustand *TIMED_WAIT* ist – wie schon weiter oben angesprochen – darin zu sehen, dass noch zur Sicherheit so lange gewartet werden soll, bis tatsächlich kein Paket mehr unterwegs sein kann. Deshalb wird die aktuell doppelte Paketlebensdauer als Wartezeit verwendet.

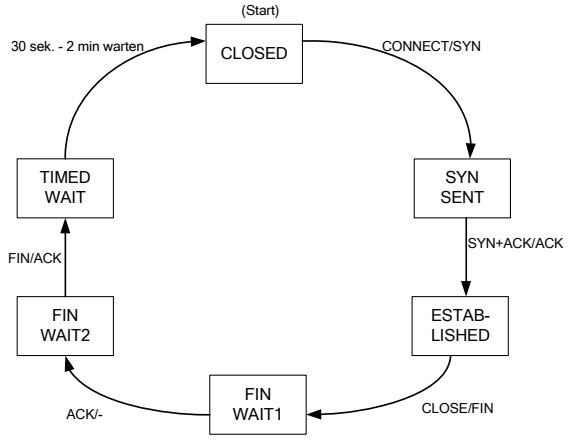


Abbildung 5-37: TCP-Zustandsautomat des Clients

Neben dem eben beschriebenen Verbindungsaufbau und -abbau sieht der TCP-Protokollautomat noch spezielle Varianten wie beispielsweise den gleichzeitigen Verbindungsaufbau- und -abbau vor.

In Abbildung 5-37 und Abbildung 5-38 sind die Zustandsübergänge nochmals getrennt für den TCP-Client (aktiver Partner beim Verbindungsaufbau) und den TCP-Server (passiver Partner beim Verbindungsaufbau) dargestellt.

Der Status wird mit allen anderen Variablen im sog. TCB (Transmission Control Block (RFC 793) durch die TCP-Instanz verwaltet. Andere Variablen, die je TCP-Verbindung und Kommunikationsseite im TCB liegen, sind die Fenstergröße, die Größe des Überlastungsfensters, die verschiedenen Zeiger für die Fensterverwaltung, die verschiedenen Timer usw.

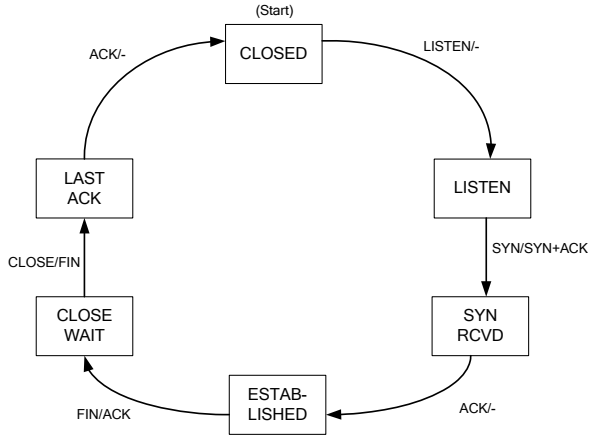


Abbildung 5-38: TCP-Zustandsautomat des Servers

Es soll noch erwähnt werden, dass im TCP-Zustandsautomaten der Sende- und Empfangsvorgang nicht mehr verfeinert sind. Er konzentriert sich hauptsächlich auf das Verbindungsmanagement. Die eigentliche Datenübertragung erfolgt im Zustand *ESTABLISHED*. Auch hier könnte man eine Verfeinerung vornehmen, die z.B. die Timerüberwachung, das Bestätigungs- und das Übertragungswiederholungsverfahren inkl. dem *Fast Retransmit* beschreibt.

5.3 User Datagram Protocol (UDP)

5.3.1 Einordnung und Aufgaben

Im Gegensatz zu TCP ist UDP ein unzuverlässiges, verbindungsloses Transportprotokoll. Dieses Transportprotokoll hat sich im Gegensatz zu TCP seit seiner ersten Spezifikation kaum verändert. Es ist wie folgt charakterisiert:

- Es werden keine Empfangsbestätigungen für Pakete gesendet
- UDP-Nachrichten, die bei UDP als Datagramme bezeichnet werden, können jederzeit verloren gehen
- Eingehende Pakete werden nicht in einer Reihenfolge sortiert

– Es gibt keine Fluss- und keine Staukontrolle.

Maßnahmen zur Erhöhung der Zuverlässigkeit wie Bestätigungen, Timerüberwachung und Übertragungswiederholung, müssen im darüberliegenden Anwendungsprotokoll ergriffen werden. Lediglich die Segmentierung und auch ein Multiplexieren mehrerer UDP-Kommunikationsbeziehungen über einen darunterliegenden IP-SAP wird von UDP übernommen. Trotzdem bietet UDP – richtig eingesetzt – einige Vorteile:

– Bei UDP ist keine explizite Verbindungsaufbau-Phase erforderlich und entsprechend auch kein Verbindungsabbau. UDP-basierte Protokolle sind recht einfach zu implementieren. UDP muss keine Kontextverwaltung durchführen, sondern arbeitet im Wesentlichen zustandslos. Da es keine Verbindungen gibt, braucht man für jeden Kommunikationsprozess genau einen Port (hier ein UDP-Port). Ein Anwendungsprozess erzeugt einen UDP-Socket und kann Nachrichten senden und empfangen.
– Man kann unter Umständen eine bessere Leistung erzielen, aber nur, wenn TCP im Anwendungsprotokoll nicht nachgebaut werden muss, sondern eine gewisse Unzuverlässigkeit in Kauf genommen werden kann.
– Mit UDP kann auch Multicasting und Broadcasting genutzt werden, d.h. eine Gruppe von Anwendungsprozessen kann mit wesentlich weniger Nachrichtenverkehr kommunizieren, als bei Nutzung einzelner Punkt-zu-Punkt-Verbindungen. Beispielsweise kann man mit Multicast/Broadcast eine Nachricht, die ein Produzenten-Prozess erzeugt, mit einer einzigen UDP-PDU an mehrere Partner senden.
– Die Senderate wird bei UDP nicht wie bei TCP gedrosselt (siehe Staukontrolle), wenn eine Netzüberlast vorliegt. Es gehen dann zwar evtl. Nachrichten verloren, aber für Audio- und Video-Ströme oder sonstige Multimedia-Anwendungen kann dies günstiger sein[7].

In den UDP-Datagrammen wird immer die T-SAP-Adresse des Senders und des Empfängers gesendet. Der T-SAP ist bei UDP ein UDP-Port. Der Wertebereich der in einem Rechner verfügbaren UDP-Ports ist disjunkt zum Wertebereich für TCP-Ports und wird von den Protokollimplementierungen völlig selbstständig verwaltet.

Zu den wichtigen UDP-Portnummern, die auch als well-known Ports definiert sind, gehören u.a.:

– 53, DNS (Domain Name Service)
– 69, TFTP (Trivial File Transfer Protocol)
– 161, SNMP (Simple Network Management Protocol)
– 520, RIP (Routing Information Protocol)

[7] Wie bereits erläutert tolerieren diese Anwendungen einen gewissen Datenverlust und verkraften diesen eher als zu große und unkontrollierbare Verzögerungen und Jitter.

Im Gegensatz zu TCP bietet eine UDP-Instanz an der Schnittstelle zum Anwendungsprozess auch keinen Stream-Mechanismus. Datagramme werden in festen Blöcken gesendet.

5.3.2 UDP-Header

Der UDP-Protokoll-Header (siehe Abbildung 5-39**)** ist verglichen mit TCP wesentlich einfacher und besteht aus nur acht Byte. Neben der Adressierungsinformation enthält er noch die variable Länge des UDP-Datagramms sowie eine optionale Prüfsumme.

Die Felder des UDP-Headers sollen kurz einzeln beschrieben werden:

- *UDP-Quell-Portnummer*: Nummer des sendenden Ports.
- *UDP-Ziel-Portnummer*: Nummer des empfangenden Ports, also des adressierten Partners.
- *Länge*: Hier wird die Größe des UDP-Paketes inkl. des Headers in Byte angegeben. Dies ist notwendig, da UDP-PDUs keine fixe Länge aufweisen.
- *Prüfsumme* (optional): Die Prüfsumme ist optional und prüft das Gesamtpaket (Daten+Header) in Verbindung mit einem Pseudo-Header (siehe unten). Es wird ein Einer-Kompliment zu der Summe aller 16-Bit-Wörter ermittelt.[8]
- *Daten*: Nettodaten des Datagramms.

Durch die Angabe der Datagramm-Länge (16 Bit breites Längenfeld) ist die Länge einer UDP-PDU begrenzt. Die Länge eines UDP-Pakets ist minimal 8 Byte (nur der UDP-Header ohne Prüfsumme) und maximal $2^{16} - 1 = 65.535$ Byte. Die Nettodatenlänge ergibt sich aus folgender Berechnung:

```
2^16 - 1 - 8 (Header) Byte = 65.527 Byte
```

Es ist sinnvoll, die UDP-Nachrichten nicht länger als in der maximal möglichen IP-Paketlänge zu versenden, da sonst in der IP-Schicht fragmentiert wird und die Wahrscheinlichkeit eines Datenverlusts größer wird.

Da die Prüfsummenberechnung tatsächlich etwas komplizierter ist und auch noch der sog. Pseudo-Header im Spiel ist, soll hier nochmals kurz darauf eingegangen werden. Die Prüfsumme ist die einzige Möglichkeit, die intakte Übertragung beim Empfänger zu verifizieren. Ziel des Prozederes ist es auch, beim Empfänger herauszufinden, ob das Paket den richtigen Empfänger gefunden hat.

Man spricht von einem Pseudo-Header, weil vor dem Berechnen der Prüfsumme ein virtueller Header gebildet wird, der um die Felder aus Abbildung 5-39 ergänzt wird. Das Paket wird auf eine durch 16 Bit teilbare Größe aufgefüllt. Die ergänzten Felder sind Teile des IP-Headers.

[8] Einzelheiten zur Prüfsummenberechnung finden sich im RFC 1071.

Der Empfänger muss bei Empfang einer UDP-Nachricht, die eine Prüfsumme enthält, folgendes unternehmen:

- Die IP-Adressen aus dem ankommenden IP-Paket lesen.
- Der Pseudo-Header muss zusammengebaut werden.
- Die Prüfsumme muss ebenfalls berechnet werden.
- Die mitgesendete Prüfsumme mit der berechneten vergleichen.

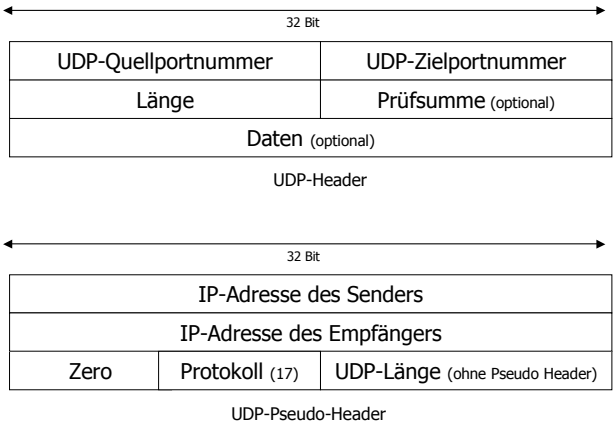


Abbildung 5-39: UDP-Header und UDP-Pseudo-Header

Wenn die beiden Prüfsummen identisch sind, dann hat man durch diesen Trick die Gewissheit, dass das Datagramm seinen Zielrechner und auch den richtigen UDP-Port erreicht haben muss. Eine Fehlerbehebung findet aber nicht statt. Ein fehlerhaftes Paket wird nicht an den Anwendungsprozess weitergegeben, sondern in den meisten Implementierungen verworfen.[9]

5.3.3 Datenübertragung

Die Datenübertragung ist in UDP relativ einfach. Eine UDP-basierte Kommunikationsanwendung muss nur dafür sorgen, dass die Kommunikationsprozesse ihre Ports kennen, über die Socket-Schnittstelle jeweils ein UDP-(Datagramm)-Socket erzeugen und dann können sie sich Datagramme zusenden. Das Senden der Daten erfolgt dabei völlig asynchron. Das darüberliegende Anwendungsprotokoll muss die eigentliche Arbeit ausführen.

[9] Manche UDP-Implementierungen scheinen ein fehlerhaftes UDP-Datagramm auch an den Anwendungsprozess weiter zu geben und eine Warnung zu ergänzen.

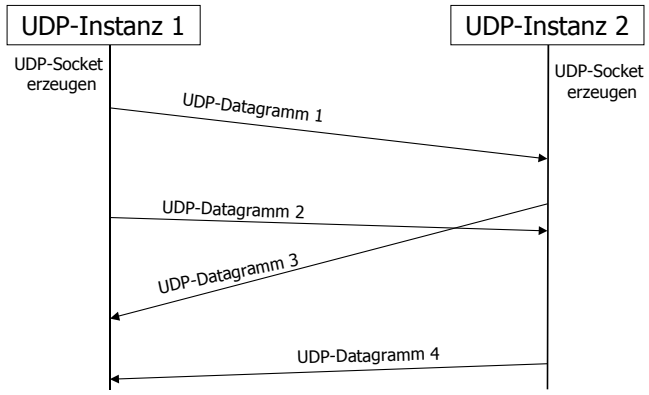


Abbildung 5-40 Einfache Kommunikation über UDP

Die UDP-Instanz wickelt bei Bedarf eine Segmentierung/Desegmentierung ab, aber übernimmt zusätzlich zur Übertragung der Datagramme keine weiteren Maßnahmen. In Abbildung 5-40 ist eine typische Kommunikation zweier über UDP kommunizierender Transportinstanzen dargestellt, die sich gegenseitig wahlfrei sich überlappende Datagramme zusenden. Die gesamte Protokolllogik liegt in den Anwendungs-PDUs. Die beiden in der Abbildung kommunizierenden Prozesse senden hier jeweils zwei PDUs über UDP zu ihren Kommunikationspartnern. Bestätigungen werden von UDP nicht versendet. Diese müssen in den Anwendungs-PDUs versendet werden.

Aufgrund des einfachen Protokolls wird an dieser Stelle auf die Beschreibung des Protokollautomaten verzichtet.

5.4 Abschließende Bemerkung

TCP ist eindeutig das weit komplexere Transportprotokoll der TCP/IP-Protokollfamilie. Die TCP-Entwicklung ist aufgrund der sich ändernden Anforderungen (mobile Systeme, multimediale Anwendungen) noch nicht abgeschlossen. Es wird auch heute noch ständig weiterentwickelt. Es soll daher noch abschließend auf neue Ansätze für das Congestion Control und auf weitere TCP-Funktionen im Umfeld der High-Preformance-Netzwerke hingewiesen werden. Hierzu gehören Ansätze wie TCP *Reno*, TCP *BIC*, TCP *CUBIC*, *Hamilton* TCP, TCP *Highspeed* und TCP *Scalable*. Diese Ansätze sollen hier nicht weiter vertieft werden. Einen ersten Einstieg in neue Themen der TCP-Forschung findet man im WWW. [10]

[10] http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/index.htm (letzter Zugriff am 10.6.2007).

5.5 Übungsaufgaben

1. Nennen Sie vier typische Protokollfunktionen, die in der Transportschicht implementiert sein sollten!
2. Beschreiben Sie einen Drei-Wege-Verbindungsaufbau und erläutern Sie kurz, wie man durch diese Art des Verbindungsaufbaus Duplikate erkennen kann!
3. Was ist eine Ende-zu-Ende-Verbindung im Sinne der Transportschicht?
4. Nennen Sie je einen Vorteil für ein verbindungsloses und ein verbindungsorientiertes Protokoll der Transportschicht!
5. Was will man mit einer Timerüberwachung beim Verbindungsabbau einer Transportverbindung erreichen?
6. Bei der Fehlerbehandlung nutzt man in einer gesicherten Transportschicht u.a. das positiv selektive und das negativ selektive Quittierungsverfahren. Erläutern Sie die beiden Verfahren!
7. Zur Fehlerbehebung nutzt man in der Transportschicht das Verfahren der Übertragungswiederholung. Nennen Sie hierzu zwei Verfahren und erläutern sie diese kurz! Was muss der Sender tun, damit eine Übertragungswiederholung möglich ist?
8. Was ist bei TCP ein Port und was ist bei TCP ein well-known Port?
9. TCP ist datenstromorientiert (stream-ortientiert). Was bedeutet das?
10. Was sind TCP-Segmente, wie groß sind diese mindestens und warum?
11. Wie wird in TCP eine Verbindung eindeutig identifiziert?
12. Wie viele TCP-Segmente werden für die Übertragung von 100 Byte Nutzdaten durchs Netz gesendet? Erläutern Sie dies anhand eines Time-Sequence-Diagramms!
13. Welche Quittierungsvariante wird bei TCP eingesetzt, damit der Empfänger den ordnungsgemäßen Empfang einer Nachricht bestätigen kann? Was macht der Sender eines Pakets, wenn er keine Quittung vom Empfänger erhält?
14. Erläutern Sie das Silly-Window-Syndrom bei TCP!
15. Wozu dient das Slow-Start-Verfahren bei TCP? Beschreiben Sie das Verfahren kurz!
16. Wozu dienen die beiden Zustände TIMED_WAIT und CLOSE_WAIT im TCP-Zustandsautomaten? Gehen Sie dabei kurz auf den Verbindungsabbau ein!
17. Nennen Sie zwei Timer, die bei TCP verwendet werden, und beschreiben Sie kurz deren Aufgabe!
18. Was unternimmt die empfangende UDP-Instanz, wenn eine UDP-PDU ankommt? Gehen Sie hier auf den Sinn des UDP-Pseudo-Headers ein!

19. Nennen Sie zwei Vorteile von UDP im Vergleich zu TCP!
20. Wozu dienen die Stati SYN_RECVD und SYN_SENT des TCP-Zustandsautomaten?
21. Welchen Sinn hat der UDP-Pseudo-Header?
22. Übernimmt eine UDP-Instanz die Aufgabe der Segmentierung eines langen Datagramms oder muss diese Aufgabe das Anwendungsprotokoll erledigen?
23. Erläutern Sie kurz, was bei einer negativen Quittierung für eine Transport-PDU passiert, wenn in einem Netzwerk mit hoher Pfadkapazität eine fensterbasierte Flusskontrolle mit einem großen Fenster angewendet wird und die Übertragungswiederholung im Go-Back-N-Verfahren erfolgt.

6 Ausgewählte Anwendungsprotokolle

Das Transportsystem ist nur vorhanden, weil es den eigentlich nutzbringenden Kommunikationssystemen, die in der Anwendungsschicht liegen, als Kommunikationsbasis dienen soll. Die höheren Kommunikationsprotokolle unterstützen, wie der Name sagt, konkrete Anwendungssysteme. In diesem Kapitel werden daher nach einem Überblick über TCP- und UDP-basierte Protokolle der Anwendungsschicht einige Anwendungssysteme und deren Kommunikationsprotokolle betrachtet. Als Fallbeispiele dienen das Domain Name System (DNS), Webbasierte Systeme mit HTTP als Standardprotokoll sowie E-Mail-Services mit den entsprechenden Protokollen SMTP, POP3 und IMAP4 und einige Anwendungsprotokolle für multimediale Systeme.

Zielsetzung des Kapitels

Ziel dieses Kapitels ist es, einen Überblick über die Aufgaben und die Funktionsweise höherer Protokolle der TCP/IP-Familie und deren Einbettung in die Referenzmodelle zu verschaffen.

Wichtige Begriffe

DNS, Domänen, Top-Level-Domains (TLDs), Domain-Server, Root-Name-Server, DNSSEC, HTTP, AJAX, XMLHttpRequest, HTML, WWW, SMTP, IMAP4, POP3, Codec, RTP, RTCP, RTSP, SIP, UPnP, MPEG.

6.1 Überblick über TCP/UDP-Anwendungsprotokolle

Bevor auf spezielle Anwendungsprotokolle der TCP/IP-Protokollfamilie eingegangen wird, soll vorab nochmals ein Überblick über einige wichtige Protokolle und deren Einordnung in die Protokollfamilie gegeben werden.

Wir sehen in Abbildung 6-1 einige Protokolle in der Anwendungsschicht, die entweder auf TCP oder UDP basieren. In der Transportschicht sind die wichtigsten Protokolle TCP und UDP, die wiederum auf IPv4/IPv6 aufsetzen.

Es gibt aber auch durchaus Protokolle, die mehrere andere Basisprotokolle nutzen. DNS kann sowohl über TCP als auch über UDP arbeiten. In der Vermittlungsschicht sind schließlich IPv4 bzw. IPv6 die zentralen Protokolle, über die alle anderen Protokolle ihre Nachrichten austauschen.

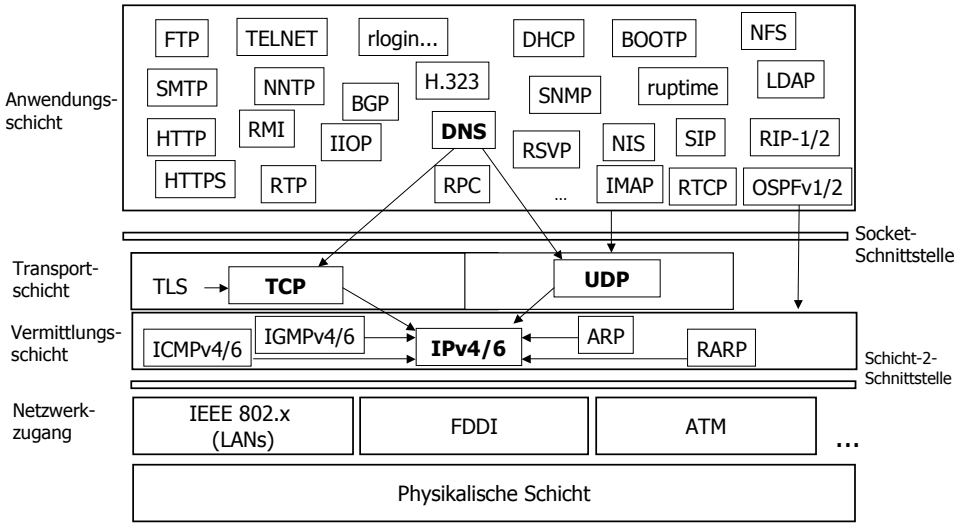


Abbildung 6-1: Wichtige Protokolle der TCP/IP-Familie

In Tabelle 6-1 sind die Protokollbezeichnungen einiger wichtiger Protokolle der TCP/IP-Protokollfamilie kurz erläutert. Zu jedem Protokoll wird angegeben, auf welchem Basisprotokoll (TCP, UDP, IP) es aufsetzt. Hinter den meisten Anwendungsprotokollen stecken auch zum Teil komplizierte verteilte Systeme. DNS ist z.B. ein verteiltes Datenhaltungssystem. FTP setzt eine Client-/Server-Beziehung zwischen einem FTP-Server und einem FTP-Client, der eine Datei übertragen möchte, voraus. Auch SNMP ist zwar der Name eines Protokolls, aber dahinter verbirgt sich ein Manager-Agenten-System mit SNMP-Agenten, die Management-Informationen verwalten, und SNMP-Clients (Management-Anwendungen), welche die Information lesen und teilweise auch verändern können.

Wir wollen im Folgenden nur eine kleine Auswahl an Protokollen aus der Anwendungsschicht, und zwar DNS, HTTP und E-Mail-Protokolle besprechen, ohne die man sich das Internet gar nicht mehr vorstellen kann. Weiterhin werden wir auf einige Multimedia-Protokolle eingehen. Andere Anwendungsprotokolle wie RIP und OSPF, die zwar formal in der Anwendungsschicht angesiedelt sind, funktional aber in die Schicht 3 gehören, wurden bereits an den entsprechenden Stellen behandelt. Informationen zu den anderen Protokollen sind in der Literatur und in den einschlägigen RFCs nachzulesen.

Tabelle 6-1: Protokolle der TCP/IP-Familie

Protokoll	Transport	Beschreibung
ARP	IP	Address Resolution Protocol
ICMP	IP	Internet Control Message Protocol
IGMP	IP	Internet Group Management Protocol
OSPF, OSPFv2	IP	Open Shortest Path First Protocol
RARP	IP	Reverse Address Resolution Protocol
RSVP	IP	Resource Reservation Protocol
BGP, BGP-4	TCP	Border Gateway Protocol
FTP	TCP	File Transfer Protocol
HTTP (HTTPS)	TCP	Hypertext Transfer Protocol (over TLS)
IIOP	TCP	Internet Inter ORB Protocol
NNTP	TCP	Network News Transfer Protocol
SMTP	TCP	Simple Mail Transfer Protocol
rcp, rlogin,..	TCP	Berkeley Remote-Kommandos
RMP (RMI)	TCP	Java Remote Method Invocation Protocol
TELNET	TCP	Sitzung mit entferntem Host
TLS	TCP	Transport Layer Security (SSLv3), Vorgänger SSL
X.11	TCP	C/S-Protokoll des X-Windows-Systems
BOOTP	UDP	Bootstrap-Protokoll für diskless Computer
DHCP	UDP	Dynamic Host Configuration Protocol
LDAP	UDP	Ligthtweight Directory Access Protocol
NFS	UDP	Network File System
NIS, NIS+	UDP	Network Information Service
RIP-1, RIP-2	UDP	Routing Information Protocol
SNMP	UDP	Simple Network Management Protocol
RTP	UDP	Realtime Transport Protocol
RTCP	UDP	Realtime Transport Control Protocol
RPC	UDP, TCP	Remote Procedure Call (z.B. Sun ONC)
H.323	UDP, TCP	ITU-T-Standard für multimediale Kommunikation
RPC	UDP, TCP	Remote Procedure Call (z.B. Sun ONC)
DNS	UDP, TCP	Domain Name System
SIP	UDP, TCP	Session Initiaton Protocol

6.2 Domain Name System (DNS)

In diesem Abschnitt wird das Domain Name System erläutert, das heute im Internet für die Abbildung von symbolischen Namen (Domainnamen usw.) auf Adressen (IP-Adressen usw.) zuständig ist.

6.2.1 Systemüberblick

Im ARPANET mit seinen wenigen hundert Hosts war die Verwaltung der Rechneradressen noch einfach. Es gab eine Datei *hosts.txt* auf einem Verwaltungsrechner, die alle IP-Adressen des Netzwerks enthielt. Der Namensraum war flach. Die Datei wurde nachts auf die anderen Hosts kopiert, und das war der ganze Konfigurationsaufwand, der zu betreiben war.

Als das Netz immer größer wurde, war das nicht mehr so einfach zu handhaben und man führte im Jahre 1983 DNS (Domain Name System) ein. DNS dient der Abbildung von Hostnamen auf IP-Adressen sowie E-Mail-Adressen und ist ein sog. Internet-Directory-Service. Das DNS wurde 1983 von Paul Mockapetris entworfen und im RFC 882 beschrieben. Der RFC 882 wurde inzwischen von den RFCs 1034 und 1035 abgelöst.

DNS ist ein hierarchisches Namensverzeichnis für IP-Adressen (Adressbuch des Internets) und verwaltet eine statische Datenbasis, die sich über zahlreiche Internet-Hosts erstreckt. Konzeptionell ist das Internet in mehrere hundert *Domänen* aufgeteilt. Die Domänen sind wiederum in *Teildomänen* (Subdomains) untergliedert usw. Es ist allerdings eine *rein organisatorische* und keine physikalische Einteilung.

Man spricht auch von einem weltweit verteilten Namensraum (vgl. Abbildung 6-2), der als Baum strukturiert ist. Die Blätter des Baumes sind die Hosts.

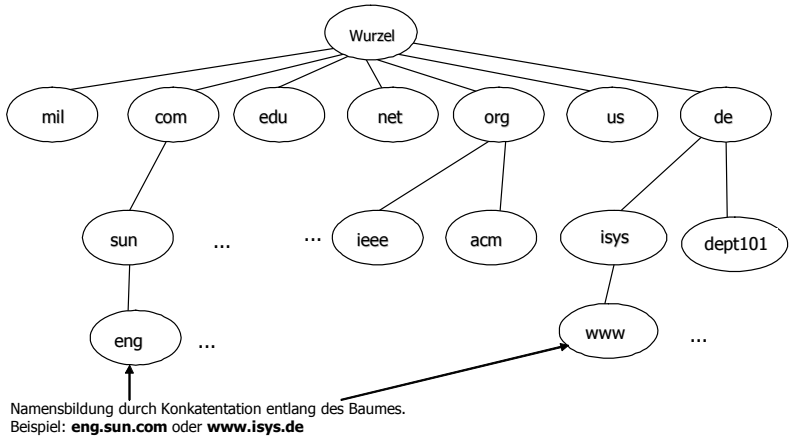


Abbildung 6-2: Hierarchischer DNS-Baum

Man unterscheidet verschiedene Domaintypen:

- Geographische oder länderspezifische (Country-Code, ccTLDs) Domains wie de, at, us, uk, gb, usw. Für jedes Land ist nach ISO 3166 ein Code mit zwei Buchstaben vorgesehen. Es gibt derzeit über 200 ccTLDs. Für die Europäische Union wurde *.eu* ebenfalls dieser Art von Domains zugeordnet, obwohl *.eu* als eine Ausnahme behandelt wird (Liste der Ausnahmen zu ISO 3166).
- Allgemeine Domains für Organisationen (*generic* oder *gTLDs*)
- Infrastruktur-Domains als Sonderfall (spezielle Domain .arpa)

Man nennt diese Domains auch Top-Level-Domains (ccTLDs für geographische Domains und TLDs für die anderen). Einige der sog. Top-Level-Domains sind in der Tabelle 6-2 aufgeführt. Diese werden auch als „nicht-gesponserte" Domains bezeichnet, da ihre Verwaltung durch die ICANN erfolgt. Heute ist die *ICANN* (Internet Corporation for Assigned Names and Numbers) und die Internet-Community für die Pflege und Vergabe der Domain-Namen verantwortlich. Will man eine Domäne reservieren, übernimmt meist ein ISP (Internet Service Provider) stellvertretend diese Aufgabe.

Tabelle 6-2: Einige Generic Top-Level-Domains (gTLDs)

Domain	**Beschreibung**
com	Kommerzielle Organisationen (sun.com, ibm.com, ...)
edu	Bildungseinrichtungen (fhm.edu)
gov	Amerikanische Regierungsstellen (nfs.gov)
mil	Militärische Einrichtung in den USA (navy.mil)
net	Netzwerkorganisationen (nsf.net)
org	Nichtkommerzielle Organisationen
bis	Business, für Unternehmen
arpa	TLD des ursprünglichen Arpanets, die heute als sog. *Address and Routing Parameter Area* verwendet wird (auch als "Infrastruktur-Domain" bezeichnet).
pro	Professions, Berufsgruppen der USA, Deutschlands und des Vereinigten Königreichs

Alle TLDs sind in der obersten Ebene des Namensbaums platziert. Diese Ebene des DNS-Namensraums wird vom InterNIC[1] (Internet Network Information Center) administriert. InterNIC überträgt die Verantwortung zur Verwaltung von Domänen an öffentliche und private Organisationen. Diese Organisationen setzen DNS-

[1] Siehe hierzu http://www.Internic.com.

Server für die Verwaltung ein. Jedes Land hat seine eigene Registrierungspolitik. Die Domäne *.de* wurde ursprünglich technisch und auch administrativ an den Universitäten in Dortmund und Karlsruhe verwaltet und wurde ab 1996 mit der Gründung der DENIC eG (Deutsche Network Information Center) nach Frankfurt überführt. Die DENIC ist heute eine eingetragene Genossenschaft.

„Gesponserte" Domains werden von unabhängigen Organisationen in Eigenregie kontrolliert und auch finanziert. Diese Organistationen haben das Recht, eigene Richtlinien für die Vergabe von Domainnamen anzuwenden. Zu den „gesponserten" TLDs gehören:

- *.aero*: Aeronautics, für in der Luftfahrt tätige Organisationen, weltweiter Einsatz.
- *.coop*: Steht für *cooperatives* (Genossenschaften), weltweiter Einsatz.
- *.mobi*: Darstellung von Webseiten speziell für mobile Endgeräte, weltweiter Einsatz.
- *.museum*: Museen, weltweiter Einsatz.
- *.name*: Nur für natürliche Personen oder Familien (Privatpersonen), weltweiter Einsatz.
- *.travel*: Für die Reise-Industrie (z.B. Reisebüro, Fluggesellschaften etc.).

Die Infrastruktur-TLD *.arpa* war ursprünglich nur als temporäre Lösung bei der Einrichtung des DNS im Internet gedacht, jedoch stellte sich die spätere Auflösung dieser Domain als problematisch heraus. Die Subdomain *in-addr.arpa* ist heute weltweit im Einsatz, um das Auflösen einer IP-Adresse in einen Domainnamen zu ermöglichen.[2]

Es gibt im globalen Internet heute auch Organisationen, die alternative DNS-Server betreiben, über die zusätzlich zu den vom ICANN kontrollierten TLDs weitere TLDs verfügbar sind. Diese Adressen sind aber für herkömmliche Internet-Nutzer nicht erreichbar. Auch werden sie von Suchmaschinen wie Google ignoriert. Ein weiterer Nachteil ist, dass die Namensräume zweier Betreiber kollidieren können. Folgende Organisationen sind noch verfügbar:

- *OpenNIC* versucht die alternativen Systeme zusammenzuführen, betrachtet jedoch die ICANN-TLDs als vorrangig und akzeptiert weder in Konflikt stehende noch private Namensräume. OpenNIC verfügt über eigene TLDs mit den Bezeichnungen *.glue*, *.indy*, *.geek*, *.null*, *.oss* und *.parody*.
- *AlterNIC* stellt die folgenden TLDs *.exp*, *.llc*, *.lnx*, *.ltd*, *.med*, *.nic*, *.noc*, *.porn* usw. zur Verfügung.
- Das *Free Community Network* verwendet als einzige TLD .fcn.
- *Pacific Root* bietet TLDs, die über *OpenNIC*-Name-Server erreichbar sind: *ais*, *.bali*, *.belize*, *.bio*, .cal, *.career*, *.chem*, *.children*, *.costarica*, *.ind*, *.job*, *.lib*, *.medic*, *.nomad*, *.npo*, *.ppp*, *.sat*, *.satcom*, *.satnet*, *.scuba*, *.socal*, *.stream*, *.work* und *.www*.

[2] Siehe auch „reverse Lookup" weiter unten.

Des Weiteren gibt es auch das europäische *Open Root Server Network.* Es stellt eine unabhängige Alternative zu den ICANN-Root-Name-Servern[3] bereit.

DNS ist eine baumförmige weltweite Vernetzung von Name-Servern (DNS-Servern), die gemeinsam die verteilte DNS-Datenbank bilden. Jeder Knoten im DNS-Baum hat einen Namen und stellt eine Domäne bzw. eine Subdomäne. Der Baum kann mit einem Verzeichnis in einem Dateisystem verglichen werden. In Abbildung 6-3 sind zwei Beispiele für Unternehmensnetze mit einem entsprechenden Subdomänen-Baum. Beispielsweise hat die Hochschule München eine Aufteilung der Domäne *hm* in Subdomänen realisiert. Die Fakultät Informatik (*cs*: Computer Science) hat z.B. einen eigenen Adressraum zu verwalten.

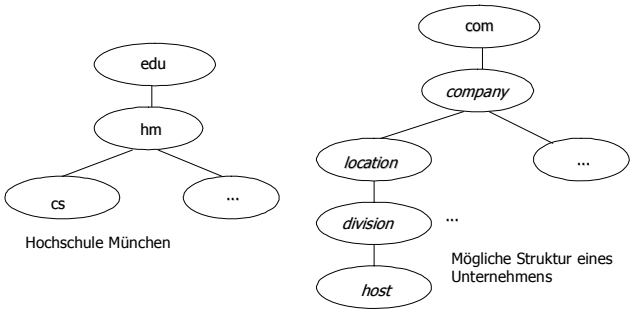


Abbildung 6-3: Beispiele für eine Domäne mit untergeordneten Subdomänen

6.2.2 DNS-Zonen und deren Verwaltung

Das Herzstück von DNS bilden die sog. DNS-Root-Name-Server (kurz: Root-Name-Server). Diese stehen weltweit zur Verfügung, um eine DNS-Anfrage zu beantworten bzw. Informationen über die weitere Suche zu geben. Ein Root-Name-Server verfügt über eine Referenz-Datenbank aller von der ICANN freigegebenen *Top-Level-Domains (TLD)* und die wichtigsten Referenzen auf die sog. Top-Level-Domain-Server.

Ein Root-Name-Server kennt immer einen DNS-Server, der eine Anfrage beantworten oder zumindest an den richtigen DNS-Server weiterleiten kann. Er weiß, wo die einzelnen Top-Level-Domain-Server zu finden sind. Die Root-Domain-Server sind demnach für die Namensauflösung von großer Bedeutung. Wären alle Root-Domain-Server nicht mehr verfügbar, dann könnte man im Internet nur noch sehr eingeschränkt kommunizieren.

Es gibt derzeit weltweit 13 sog. Root-Name-Server (A bis M), von denen 10 in Nordamerika, einer in Stockholm, einer in London und einer in Tokio (M-Knoten) stehen. Da die Root-Name-Server vielfach das Ziel von Angriffen waren, wurde

[3] Die Aufgabe der DNS-Root-Name-Server wird weiter unten in diesem Kapitel erläutert.

die Ausfallsicherheit durch die Nutzung von *Anycast* als Adressierungsart in den letzten Jahren nochmals wesentlich erhöht.

Einige Root-Name-Server bestehen heute somit nicht mehr aus einem Serversystem, sondern aus mehreren Serversystemen, die zu einem logischen Server zusammengeschlossen sind. Diese Server (Nodes) befinden sich an verschiedenen Standorten um die ganze Welt verteilt und sind per Anycast über dieselbe IP-Adresse erreichbar. Bei Anycast antwortet der Root-Name-Server auf eine DNS-Anfrage, der über die kürzeste Route erreichbar ist. Anfang 2007 nutzten bereits sechs Root-Name-Server das Anycast-Verfahren. Der Root-Name-Server F bestand z.B. aus 33, der Root-Name-Server K aus 16 Serverrechnern. Anfang 2009 gab es insgesamt bereits mehr als 120 Root-Name-Server weltweit.

Gemäß RFC 2870 muss jeder Root-Name-Server mit der dreifachen Last des am stärksten belasteten Root-Name-Servers umgehen können. Das bedeutet, dass ein Root-Name-Server im Normalbetrieb nur maximal ein Drittel seiner Kapazität ausnutzen darf. Fallen zwei Drittel der Kapazität eines Root-Name-Servers aus, soll das noch betriebsfähige Drittel trotzdem alle Anfragen beantworten können.

Da die Abhängigkeit von den in Nordamerika platzierten Root-Name-Servern sehr hoch war, hatte sich seit dem Jahre 2002 eine europäische Initiative, das *ORSN* (Open Root Server Network) etabliert, um für eine gewisse Autarkie des Netzwerks in Europa zu sorgen. Die ORSN wollte gerne in mehreren europäischen Ländern, allerdings max. 13 Root-Name-Server, einrichten. Das Netzwerk wurde auch aufgebaut, allerdings wurde der Betrieb im Jahre 2008 mangels Interesse der Betreiber wieder eingestellt.

Ein Internet Service Provider (ISP) oder ein privater ISP (z.B. ein Unternehmen) verfügt über einen lokalen DNS-Server. Die Anfrage eines Hosts geht zunächst zum lokalen DNS-Server. Ein Host kennt dessen Adresse meist aufgrund einer manuellen[4] oder einer auomatischen (siehe DHCP) Konfiguration.

Ein einzelner DNS-Server verwaltet jeweils sog. *Zonen* des DNS-Baums, wobei eine Zone an einem Baumknoten beginnt und die darunterliegenden Zweige beinhaltet. Ein DNS-Server (bzw. die entspr. Organisation) kann die Verantwortung für Subzonen an einen weiteren DNS-Server delegieren. Die DNS-Server kennen jeweils ihre Nachbarn in der darunter- und darüberliegenden Zone.

[4] Unter Windows kann die IP-Adresse des lokalen DNS-Servers z.B. in der Systemsteuerung (Netzwerk) im entsprechenden Register eingetragen werden.

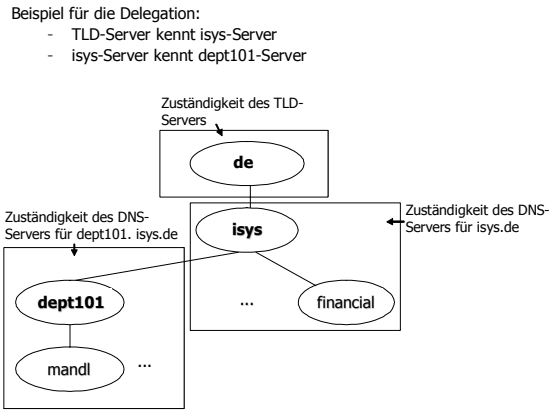


Abbildung 6-4: DNS-Name-Server und Zonen

In Abbildung 6-4 ist z.B. ein DNS-Teilbaum dargestellt, der drei Zonen zeigt. Die oberste Zone mit dem Domänennamen .de wird vom TLD-Server verwaltet. Die Domäne isys und alles was darunter liegt wird an die Firma iSYS delegiert. Unter isys sind zwei Zonen angelegt. Eine Zone für die Domäne isys und eine weitere Zone für den Teilbaum der Abteilung *dept101*. In beiden Zonen sind DNS-Server vorhanden. Eine Anfrage aus einem anderen Netz kann vom TLD-Server an den iSYS-DNS-Server weitergeleitet werden, der dann seinerseits ggf. den DNS-Server aus *dept101* nutzt. Die *dept101*-Zone ist nach außen hin nicht bekannt.

6.2.3 Namensauflösung

Eine Anwendung, die eine Adresse benötigt, wendet sich lokal an einen *Resolver* (Library), der die Anfrage an den lokalen DNS-Server (auch Name-Server) richtet, wie das Beispiel in Abbildung 6-5 zeigt. Unter Unix bzw. Linux sind die Name-Server-Dienste als *BIND* (Berkeley Internet Name Domain) implementiert. Im DNS-Server läuft der Dämonprozess *named*. BIND stützt sich auf mehrere Konfigurationsdateien, die weiter unten erläutert werden. DNS-Server sind also Programme, die Anfragen zu ihrem Domain-Namensraum beantworten.

Man unterscheidet zwischen autoritativen und nicht-autoritativen DNS-Servern. Der einem Host direkt zugeordnete DNS-Server wird als autoritativer DNS-Server bezeichnet. Oft sind die lokalen und autoritativen DNS-Server identisch. Ein autoritativer DNS-Server verfügt immer über die Adressen der direkt zugeordneten Hosts und ist verantwortlich für eine Zone. Seine Informationen über diese Zone werden deshalb als gesichert angesehen. Für jede Zone existiert mindestens ein

autoritativer DNS-Server, der sog. Primary Name-Server. Dieser wird in einer Konfigurationsdatei[5] bekannt gemacht.

Ein nicht-autoritativer Name-Server bezieht seine Informationen über eine Zone von anderen Name-Servern. Derartige Informationen sind allerdings nicht gesichert. Da sich DNS-Informationen nur selten ändern, speichern nicht-autoritative DNS-Server die erhaltenen Informationen in einem lokalen Cache im Hauptspeicher ab, damit diese bei einer erneuten Anfrage schneller vorliegen. Jeder Eintrag besitzt aber ein Verfalldatum (TTL-Datum = time to live), nach dessen Ablauf eine Löschung aus dem Cache erfolgt. Der TTL-Wert wird durch einen autoritativen Server bestimmt und anhand der Änderungswahrscheinlichkeit des Eintrags ermittelt. DNS-Daten, die sich häufig ändern, erhalten eine niedrige TTL. Das kann aber auch bedeuten, dass der DNS-Server nicht immer richtige Informationen liefert.

Der Kommunikationsanwendung bleibt die Art und Weise, wie die Adresse besorgt wird, verborgen. Sie ruft lediglich eine Methode oder Funktion auf, wie z.B. *gethostbyname* in der Sprache C, die eine IP-Adresse zu einem Hostnamen besorgt und dabei implizit die Methode *resolve* nutzt.

Der Resolver kann die Anfrage entweder lokal befriedigen, wenn er die IP-Adresse in seinem Cache gespeichert hat, oder er setzt einen Request an den ihm zugeordneten lokalen DNS-Server ab. Der DNS-Server prüft, ob er die Adresse in seinem Cache hat. Falls ja, dann gibt er diese in einer Response-Nachricht an den Resolver zurück. Falls er die Adresse nicht hat, sendet er seinerseits einen Request an den nächsten DNS-Server.

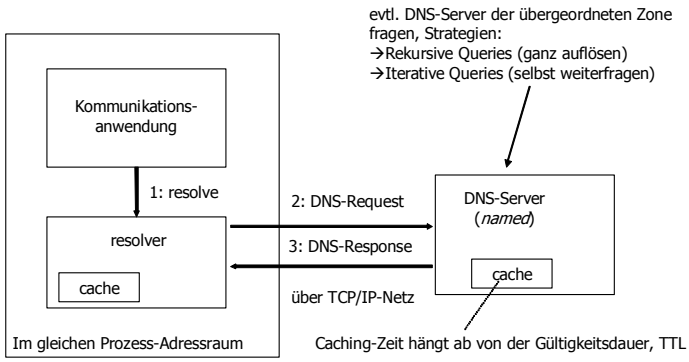


Abbildung 6-5: Adressauflösung im DNS

[5] Sog. Zonendatei, im SOA Resource Record.

Für die Kommunikation der DNS-Komponenten untereinander sind gleichermaßen der TCP- und der UDP-Port mit der Nummer 53 reserviert. DNS-Anfragen werden normalerweise auf dem UDP-Port abgesetzt und auch beantwortet. Falls die Antwort aber größer als 512 Byte war, wurde diese früher auf dem TCP-Port übermittelt. Mit Einführung von EDNS (Extended DNS) als DNS-Erweiterung wurde auch die zehnfache PDU-Größe (RFC 2671) möglich.

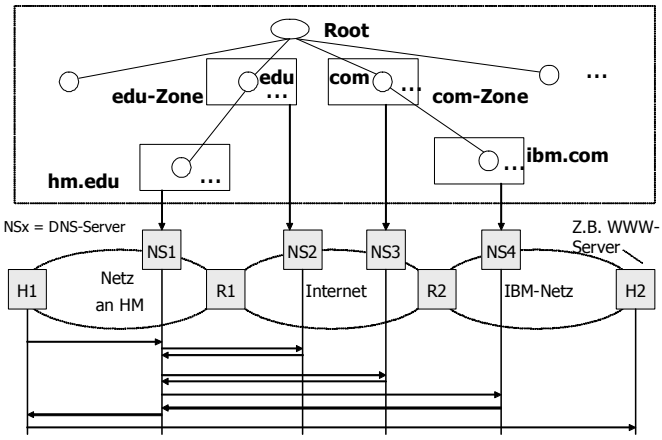


Abbildung 6-6: Iterative Auflösung von Hostnamen

Der DNS-Server kann entweder eine *rekursive* oder eine *iterative* Anfrage (Query) absetzen. In Abbildung 6-6 ist z.B. eine iterative Auflösung skizziert. Der Host *H1* im Netz der Hochschule München (*hm.edu*) möchte mit dem Rechner *H2* im IBM-eigenen Netz unter *ibm.com* kommunizieren und wendet sich zur Adressermittlung zunächst an den lokalen DNS-Server *NS1*. Dieser fragt in der übergeordneten edu-Zone nach, für die der DNS-Server *NS2* zuständig ist. *NS2* sendet an *NS1* die Information zurück, dass der DNS-Server *NS3* für diese Aufgabe zuständig ist. *NS1* stellt nun iterativ direkt eine Anfrage an *NS3* und der verweist schließlich auf *NS4*. *NS4* ist der lokale und autoritative DNS-Server für Host *H2* und kann die Anfrage beantworten. Resolver-Implementierungen von Clients nutzen üblicherweise eine rekursive Auflösung, während Name-Server eine iterative Auflösung verwenden.

Jeder DNS-Server ist genau einer Zone zugeordnet. Man sieht in der Abbildung, dass die DNS-Organisation nichts mit dem Routing zu tun hat. Die eingezeichneten Router leiten zwar die IP-Pakete für die DNS-Anfragen weiter, kennen aber die DNS-Hierarchie nicht. Für die Namensauflösung werden in diesem Beispiel acht DNS-Nachrichten durch das Netz gesendet.

Bei einer rekursiven Anfrage sieht die Kommunikation zwischen den DNS-Servern etwas anders aus (vgl. Abbildung 6-7). Jeder angefragte DNS-Server gibt die An-

frage an den nächsten DNS-Server weiter und erhält irgendwann das Ergebnis zurück, das er dann seinerseits an den anfragenden DNS-Server bzw. Host weiterreicht. Die Anzahl der Nachrichten, die durch das Netz gesendet werden, bleibt allerdings gleich. Bei einer weiteren Anfrage wäre die IP-Adresse von H2 bereits in den Caches der einzelnen DNS-Server und sogar im Resolver-Cache von H1 und die Anfrage würde wesentlich schneller gehen.

Eine Optimierung könnte sich ergeben, wenn der autoritative DNS-Server bei der Suche sofort einen Root-Name-Server kontaktiert. Die Adressen der Root-Name-Server sind allen DNS-Servern bekannt. Der adressierte Root-Name-Server könnte evtl. schon den autoritativen DNS-Server kennen.

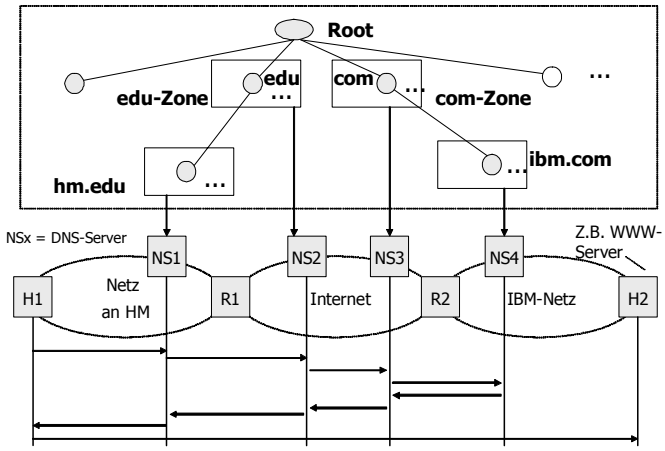


Abbildung 6-7: Rekursive Auflösung von Hostnamen

Das DNS-Protokoll ermöglicht sowohl die iterative als auch die rekursive Anfrage, und auch Mischformen für eine Anfrage sind möglich. Wann welcher Typ angewendet wird, entscheidet der anfragende DNS-Server. Anfragen an den Root-Name-Server werden immer iterativ ausgeführt, um ihn nicht zu stark zu belasten. Damit ein nicht-autoritativer Name-Server Informationen über andere Teile des Namensraumes finden kann, bedient er sich einer entsprechenden Suchstrategie, die im Folgenden nochmals kurz zusammengefasst werden soll:

- Er delegiert, wie bereits erläutert, Teile des Namensraumes einer Domain an Subdomänen, die per Konfigurierung eingerichtet werden müssen. Die Subdomänen erhalten eigene DNS-Server. Ein DNS-Server kennt alle DNS-Server der Subdomänen und gibt Anfragen ggf. an diese weiter.
- Falls der angefragte Namensraum außerhalb der eigenen Domäne liegt, wird die Anfrage an einen fest konfigurierten DNS-Server weitergeleitet, der dann ggf. seinerseits weitersucht.

– Falls kein DNS-Server antwortet, wird ein DNS-Root-Name-Server kontaktiert. Die Namen der IP-Adressen der DNS-Root-Name-Server sind in einer statischen Konfigurationsdatei auf jedem DNS-Serversystem hinterlegt.

Es soll noch erwähnt werden, dass bei einer Befriedigung eines Folgezugriffs auf die gleiche Adresse über einen Cache das Ergebnis als „nicht-autoritativ" gekennzeichnet ist, da es nicht vom autoritativen Server kommt. Der Eintrag im Cache lebt auch nur eine bestimmte, konfigurierbare Zeit und wird dann aus Aktualitätsgründen wieder gelöscht.

6.2.4 Inverse Auflösung von IP-Adressen

Meistens wird DNS verwendet, um zu einem Domänennamen die zugehörige IP-Adresse zu ermitteln. Oft gibt es aber auch die umgekehrte Situation, in der ein Benutzer oder ein Programm nur über eine IP-Adresse eines Partnerrechners verfügt und den zugehörigen Hostnamen benötigt. Man braucht in diesen Fällen den Hostnamen der besseren Lesbarkeit wegen, um ihn z.B. zur Diagnose in Logdateien einzutragen. DNS-Anfragen, die dazu dienen, eine IP-Adresse auf einen Domänennamen abzubilden, werden als inverse Anfragen bzw. als Reverse Lookup (*inverse* oder *reverse* Abbildung) bezeichnet.

Es ist allerdings sehr zeitaufwändig bei einer inversen Anfrage den gesamten Domänen-Baum nach einer IP-Adresse zu durchsuchen, zumal nicht bekannt ist, in welchem Zweig des Baumes sich der gesuchte Eintrag befindet. Aus diesem Grund wurde eine eigenständige Domäne für inverse Zugriffe geschaffen, die als in-addr.arpa-Domäne bezeichnet und durch InterNIC verwaltet wird. Unterhalb dieser Domäne gibt es nur vier Subdomänen-Ebenen, so dass man die Auflösung eines Domänennamens in wenigen Schritten erledigen kann.

Die Knoten der Domäne in-addr.arpa sind nach Zahlen in der für IP-Adressen üblichen Repräsentation benannt. Die Domäne in-addr.arpa hat 256 Subdomänen und die Subdomänen haben jeweils wieder 256 Subdomänen. In der untersten (vierten) Stufe werden die vollen Hostnamen eingetragen.

Die Subdomänen der Ebene 1 in der *in-addr.arpa-Domäne* haben als Bezeichnung eine Zahl zwischen 0 und 255 und repräsentieren die erste Komponente einer IP-Adresse. Die nächste Ebene im Baum repräsentiert die zweite Komponente einer IP-Adresse usw. In Abbildung 6-8 ist eine Adressauflösung für die IP-Adresse 195.214.80.70 beispielhaft dargestellt. Beginnend mit dem niedrigsten Byte der IP-Adresse wird in den Baum eingestiegen. Jedes Byte wird als Index in einer Ebene verwendet, bis man in der vierten Ebene angelangt ist und den zugeordneten Hostnamen erhält.

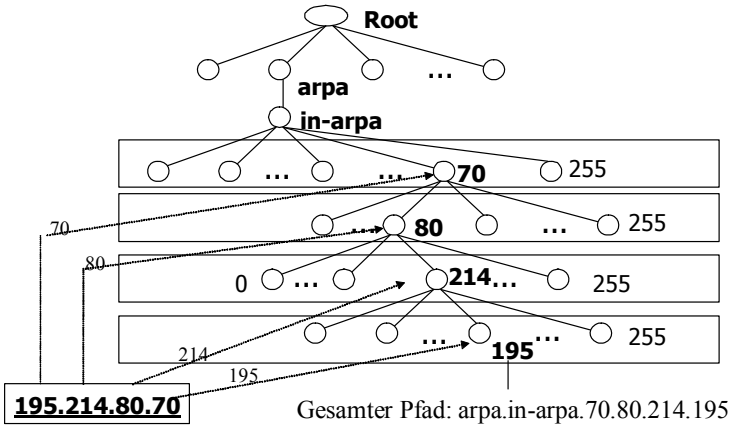


Abbildung 6-8: Reverse Abbildung von IP-Adressen auf Hostnamen

Alle lokalen Netze (Subnetze) sollte man bei Änderungen immer in der Domäne *in-addr.arpa* eintragen lassen, um die umgekehrte Abbildung zu ermöglichen.

Reverse Lookup bei IPv6. Bei Einsatz von IPv6 wird für einen inversen Lookup die Dömäne *ip6.arpa* verwendet. Die DNS-Unterstützung für IPv6 ist im RFC 3596 geregelt. Beispielsweise wird die IPv6-Adresse

4321:0:1:2:3:4:567:89ab

für einen Reverse Lookup wie folgt notiert:

b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.IP6.ARPA.

6.2.5 DNS-Konfiguration

Jede Zone muss einen DNS-Server bereitstellen und sollte diesen auch zur Erhöhung der Ausfallsicherheit doppelt auslegen. DNS bietet hierzu die notwendigen Mechanismen zur Synchronisation eines primären mit einem sekundären DNS-Server. In Abbildung 6-9 ist ein Beispiel einer möglichen Konfiguration dargestellt, ohne hier auf Sicherheitsaspekte einzugehen. Wie man sieht, gibt es in dem Ausschnitt eines Unternehmensnetzes zwei lokale DNS-Server im Netz mit den Adressen 195.214.80.70 und 195.214.80.71. Der nächste zugeordnete DNS-Server liegt beim ISP und hat die IP-Adresse 195.143.108.2.[6]

Heute trennt man netzinterne DNS-Server von den extern zugänglichen DNS-Servern und legt nur die externen in eine demilitarisierte Zone (DMZ). Der externe DNS-Server dient dann als *Forwarder* und alle DNS-Nachrichten werden aus-

[6] Dies soll nur als Beispiel dienen und muss nicht mit der Realität übereinstimmen.

schließlich über diesen gesendet. Damit werden auch externe Angreifer daran gehindert, dass sie interne Adressinformationen ermitteln können. Nur die nach außen zugänglichen Rechner (z.B. WWW-Server oder WWW-Proxy) sollen über den Forwarder abfragbar sein. Ein DNS-Forwarder ist also ein DNS-Server, der zur Auswertung einer DNS-Anfrage extern kommuniziert. Alle internen DNS-Server müssen so konfiguriert werden, dass sie Anfragen, die sie nicht selbst beantworten können, an den Forwarder senden.

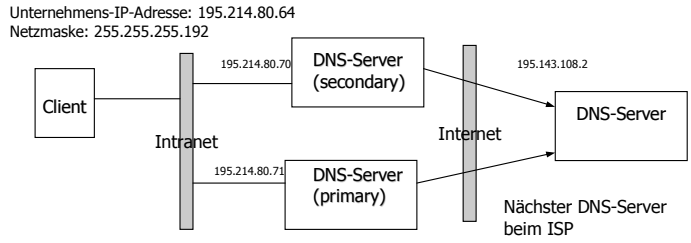


Abbildung 6-9: Doppelte Auslegung der DNS-Server in einem Unternehmensnetz

Aus Redundanz- und auch aus Lastverteilungsaspekten heraus werden in größeren Netzen autoritative DNS-Server fast immer in ein Server-Cluster eingebettet, wobei die Zonendaten identisch auf einem oder mehreren Sekundärservern liegen. Die Synchronisation zwischen dem Primär- und den Sekundärservern erfogt über spezielle Zonentransfernachrichten. Die DNS-Datenbasis wird in Dateien verwaltet. Unter Unix oder Linux werden für BIND u.a. folgende Dateien/Dateitypen[7] zur Konfiguration benötigt:

- *named.conf* und *named-boot*: Diese Datei enthält globale Parameter für den DNS-Server (Bind-Optionen, Forwarder, Zonen-Struktur, usw.) und wird unmittelbar nach dem Start des DNS-Servers eingelesen. In den Optionen werden unter *forwarders* die DNS-Server eingetragen, an welchen eine DNS-Anfrage gesendet werden soll, wenn sie nicht direkt beantwortet werden kann. Trägt man „*forwarder first*“ ein, so wird jede DNS-Anfrage zuerst an die angegebenen DNS-Server weitergeleitet, bevor sie an einen DNS-Root-Name-Server gesendet wird. Trägt man „*forwarder only*“ ein, werden die DNS-Root-Name-Server gar nicht gefragt.
- *named.cache* oder *named.root*: In dieser Datei sind die weltweit eindeutigen DNS-Root-Name-Server mit ihren IP-Adressen hinterlegt.
- *db.127.0.0* oder *named.127.0.0*: Forward-Lookup-Datei für die Abbildung von *localhost* auf die Loopback-Adresse.

[7] Die Namen der Dateien variieren je nach System, sind auch zum Teil frei zu vergeben und nur exemplarisch dargestellt.

- *db.<domain>* oder *named.<domain>:* Forward-Lookup-Datei für alle Rechnernamen der Domäne <domain>, Beispiel: *db.dept101.isys-software.de.*
- *db<network>* oder *named.<network>:* Reverse-Lookup-Datei für das Subnetzwerk mit der im Dateinamen enthaltenen IP-Adresse <network>, Beispiel: *db.192.168.2*.

Im Folgenden wird eine Konfigurationsdatei *named.conf* exemplarisch grob dargestellt, ohne auf Details einzugehen. Der Optionsteil enthält u.a. Angaben zum Forwarder sowie über Rechner, die Anfragen machen dürfen. Anschließend sind die Zonen beschrieben. Zu jeder Zone wird der Verweis auf die zugehörigen db-Dateien (für die Hosttabellen und für inverse Auflösung) angegeben.

```
options {
 directory "/var/named";
 forwarders { … };
 …
};

# Obligatorische Zone für die DNS-Root-Name-Server
zone "." IN {
 type hint;
 file "named.root";
};

# Festlegen des Loopback (forward und reverse)
zone "localhost" IN {
 type master;
 file "db.localhost";
};
zone "0.0.127.IN-ADDR.ARPA"  IN {
 type master;
 file "db.127.0.0";
};

# Die Zonen (forward und reverse)
zone "isys-software.de" IN {
 type master;
 file "db.isys-software.de";
};
…
zone "dept101.isys-software.de" IN {
 type master;
 file "db.dept101.isys-software.de";
};
```

```
zone "2.168.192.IN-ADDR.ARPA" IN {
 type master;
 file "db.192.168.2";

};

...
```

Informationen des DNS werden in sog. *Resource Records* (RR) verwaltet. Der Aufbau eines Resource Records wird meist mit folgendem Inhalt angegeben: [8]

```
(Name, Type, Class, Time-to-live, Value)
```

Die Felder sollen hier nur kurz erläutert werden:

- *Name*: Name des IP-Knotens, zu dem der RR gehört.
- *Type*: Typ des Records (A = IP-Adresse, NS = Name-Server-Record für autorisierten DNS-Server, MX = Mail-Server-Record, Verteiler für Mail-Server, SOA = Beginn einer Zone, CNAME = Aliasname, PTR = Angabe eines Zeigers auf eine Domäne).
- *Class*: immer IN (Internet).
- *Time-to-live*: Stabilität (Gültigkeitsdauer) des Records als Integerzahl (je höher desto stabiler). Der Wert ist für das Caching wichtig (kurz: TTL).
- *Value*: Wert des Records je nach Typ: z.B. bei Typ = A die IP-Adresse, bei Typ NS der Name des Name-Servers, bei Typ MX der Name des E-Mail-Servers,...).

Ohne weiter auf Details einzugehen, wird im Folgenden ein Auszug einer *db*-Datei mit Hostnamen gezeigt. Am Anfang der Datei steht ein sog. SOA-Eintrag (Start of Authority), welcher die administrierte Zone beschreibt und u.a. eine Bearbeitungsnummer für die Versionspflege enthält. Weiterhin sind einige Parameter aufgeführt, die für den DNS-Server wichtig sind. Beispielsweise wird der Standard-TTL-Wert in Sekunden (hier ein Tag) angegeben. Ansonsten sind nur NS- und A-Records in der Datei. Die TTL-Angaben sind in den einzelnen IN-Records weggelassen, es gilt die Default-Angabe.[9]

```
# Standard TTL, 2 Tage = 2D
$TTL 2D
# @ bedeutet, dass die Zone aus named.conf entnommen wird
@ IN  SOA softie.isys-software.de. root.softie.isys-software.de. (
        2002121115     ; serial YYYYMMDDCC
        10800          ; refresh after 3 hours
        3600           ; retry after 1 hour
```

[8] Dem Wert geht optional eine Längenangabe voraus.

[9] Die Bedeutung des Platzhaltersymbols „@“ und weitere Details können in der BIND-Dokumentation des jeweiligen Systems nachgelesen werden.

```
        604800         ; expire after 1 week)

IN  NS  softie.isys-software.de.
IN  NS  softie2.isys-software.de.

localhost  IN  A      127.0.0.1

; Router
grandcentral   IN  A  192.168.2.1

; Server
backup         IN  A  192.168.2.6
hardy          IN  A  192.168.2.7
sun101         IN  A  192.168.2.8
firedept       IN  A  192.168.2.9
…
; Workstations
andreas        IN  A  192.168.2.10
lalinea        IN  A  192.168.2.11
pflaume        IN  A  192.168.2.12
ulysses        IN  A  192.168.2.13
…
```

Eine weitere Anwendung von DNS ist die Unterstützung des SMTP-Protokolls[10], das für die Kommunikation im SMTP-basierten E-Mail-System wichtig ist. DNS hält alle Informationen für eine korrekte Übermittlung von E-Mails. Eine E-Mail-Adresse im SMTP-System hat bekanntlich das Format user@host.domain.suffix. Mit DNS kann der Hostname aufgelöst werden. Dazu wird vor dem Absenden einer E-Mail eine DNS-Anfrage abgesetzt, mit der ein MX-Record des Zielrechners ermittelt wird. Anschließend wird mit dieser Information eine Namensauflösung durch einen weiteren DNS-Request initiiert, um den zugeordneten A-Record zu besorgen.

Interessant ist hier, dass in den MX-Einträgen noch sog. Gewichte für die Mail-Server einer Zone angegeben werden können. Je niedriger das Gewicht ist, umso eher wird ein Mail-Server ausgewählt (inverse Zusendereihenfolge). Damit kann ein Ausfallkonzept aufgebaut werden. Normalerweise wird der Mail-Server einer Domäne adressiert, der das niedrigste Gewicht hat. Fällt ein Mail-Server aus, wird der nächste in der Liste adressiert. MX-Einträge könnten z.B. wie folgt aussehen:

```
IN MX 5  mail1.isys-software.de
IN MX 10 mail2.isys-software.de
```

[10] SMTP steht für Simple Mail Transfer Protocol.

In diesem Beispiel würde zunächst der Mail-Server *mail1* angesprochen. Wenn dieser nicht antwortet, wird der Mail-Server *mail2* adressiert.

6.2.6 DNS-Nachrichten

Die DNS-Nachrichten für das Request-/Response-Protokoll setzen sich aus mehreren Sektionen (Sections) zusammen (RFCs 1035 und 1036):

- Der *Header* besteht aus den ersten sechs Feldern (Header Section).
- Die *Question Section* enthält Felder zur Spezifikation der Anfrage. Die Anfrage wird in einem Resource Record (RR), aber nur mit den ersten drei Feldern (*Name, Type, Class*) formuliert.
- Die *Answer Section* enthält die Antwort eines Name-Servers in Form von Resource Records. Eine Antwort kann mehrere RR ausgeben, z.B. weil ein Hostname mehrere IP-Adressen haben kann.[11]
- Die *Authority Section* enthält die RRs von autorisierten Name-Servern.
- Die *Additional Information Section* enthält zusätzliche Informationen zur Anfrage oder zur Antwort wie etwa die Zeit, die für die Anfrage benötigt wurde, den Zeitpunkt der Bearbeitung und die Länge der Anfrage- sowie der Antwortnachricht.

Wie die Abbildung 6-10 zeigt, hat der DNS-Header 12 Byte mit sechs Feldern die folgende Bedeutung haben:

- *Identifikation*: Dies ist eine Id der Anwendung, welche die Abfrage abgesetzt hat und wird in die Antwort-Nachricht übernommen. Somit ist eine Zuordnung möglich.
- *Parameters:* Das Feld hat mehrere Statusfelder und Flags:
  - Anfrage/Antwort-Flag: 0 = Anfrage, 1 := Response
  - Opcode: Operationscode, der die Art der Anfrage angibt: 0 = Standard-Query, 1 = inverse Query, 2 = Serverstatus abfragen
  - AA-Flag: Flag, das angibt, ob die Antwort von einem autoritativen Server stammt
  - TC-Flag: Flag das anzeigt, ob die Nachricht geteilt wurde (> 512 Byte)
  - RD-Flag: Flag, das gesetzt wird, wenn in der Anfrage eine rekursive Auflösung gewünscht wird
  - RA-Flag: Gibt an, ob ein Server rekursives Auflösen unterstützt
  - Rcode: Antwort-Code des Servers, 0 = kein Fehler, 2 bis 5 = verschiedene Fehlertypen, Fehlertyp 3 bedeutet z.B. „Domain existiert nicht“
- *QDcount:* Anzahl der Einträge in der Question Section.
- *ANcount:*Anzahl an Resource Records (RR) in der Answer Section.
- *NScount:* Anzahl an Resource Records (RR) in der Authority Section.
- *ARcount:* Anzahl an RR in der Additional Information Section.

[11] Die IPv4-Adresse ist an die Netzwerkschnittstelle gebunden.

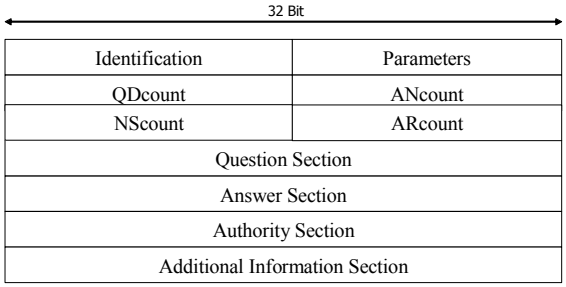


Abbildung 6-10: Aufbau der DNS-PDU

6.2.7 Sicheres DNS

Das klassische DNS verwendet keine Sicherheitsmechanismen. Dadurch ist es potenziellen Angreifern möglich, die Kommunikation zwischen DNS-Client und DNS-Server abzuhören und zu manipulieren, um im Internet z.B. gefälschte Zuordnungen von Namen auf IP-Adressen in Umlauf zu bringen. Man bezeichnet diese Angriffsart als *DNS-Spoofing*. Bei einer *Man-in-the-Middle-Attacke* braucht ein Angreifer lediglich den Datenverkehr für DNS-Anfragen über den Standard-Port 53 abzuhören und die Anfragen mit gefälschten Daten zu beantworten. Da der Client weder die Möglichkeit hat, falsche Einträge zu erkennen, noch über sichere Informationen verfügt, um den Absender zu authentifizieren, werden die falschen DNS-Einträge als korrekt angenommen und verwendet.

Um derartigen Sicherheitsproblemen zu begegnen, wurde *DNSSEC* (DNS Security Extension) erstmals in RFC 2535 im Jahr 1999 definiert und im Jahr 2004 komplett neu überarbeitet (RFC 4033). Ziele von DNSSEC sind vor allem die Sicherstellung der Authentizität der Herkunft der DNS-Pakete, die Datenintegrität und eine sichere Verteilung von öffentlichen Schlüsseln. Über einen Authentifizierungsmechanismus ist eine Prüfung möglich, ob ein Sender von Nachrichten auch derjenige ist, für den er sich ausgibt. Datenintegritätsmechanismen stellen sicher, dass Daten auf dem Weg nicht verfälscht werden. Da für die sichere Kommunikation sog. Signaturen benötigt werden, ist ein Verteilungsmechanismus für Schlüssel erforderlich. DNSEC nutzt ein Public-Key-Verschlüsselungsverfahren und unterstützt verschiedene Verschlüsselungsalgorithmen (u.a. RSA/MD5 und RSA/SHA-1)[12]. Um die oben genannten Anforderungen zu erfüllen, wurden neue Resource-Record-Typen (DNSKEY, RRSIG, usw.) definiert.

Als erste Top-Level-Domain führte Schweden im Jahr 2005 DNSSEC für die *.se*-Domäne ein und bietet seitdem DNSSEC als zusätzlichen Service für Inhaber ihrer

[12] Mehr zu Verschlüsselungsverfahren ist in (Tanenbaum 2003a) zu finden.

Subdomänen an[13]. Durch die Einführung von DNSSEC in Schweden wurde der Grundstein für einen sicheren Einsatz von DNS gelegt. Inwieweit diese Möglichkeit von anderen Internet-Teilnehmern genutzt wird, wird die Zukunft zeigen. Für eine tiefere Betrachtung von DNSSEC sei auf die RFCs 4033, 4034, und 4035 verwiesen.

6.3 Das World Wide Web

6.3.1 Einführung

HTML (Hypertext Markup Language) und HTTP (Hypertext Transfer Protocol) sind die wichtigsten Komponenten des WWW (World Wide Web). HTML ist die Auszeichnungssprache (Markup Language), in der Dokumente (auch HTML-Seiten genannt) beschrieben werden. Sie können über sog. HTML-Links (Hyperlink) beliebig verknüpft werden. *HTTP* ist ein Anwendungsprotokoll, das auf Basis von TCP als textbasiertes Kommunikationsprotokoll konzipiert wurde. Textbasiert bedeutet in diesem Zusammenhang, dass alle Informationen textbasiert übertragen werden.

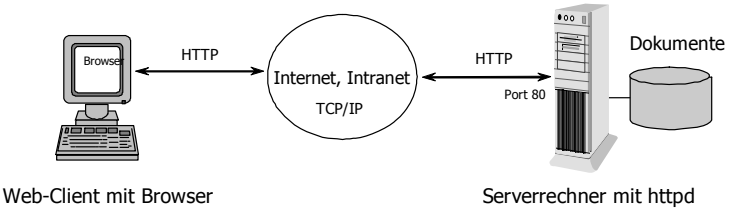


Abbildung 6-11: Web-Modell

Das Web-Kommunikationsmodell sieht weiterhin zwei wichtige Anwendungskomponenten vor (siehe Abbildung 6-11):

- *Web-Browser* auf der Clientseite als grafische Benutzeroberfläche
- *Web-Server* auf der Serverseite zur Verwaltung und Generierung der Dokumente.

HTML ist eine recht einfache Markup-Language und durch das W3C-Konsortium[14] genormt. Eine HTML-Seite beginnt mit dem Tag <HTML> und endet mit </HTML>. Es gibt einige Möglichkeiten, um eine Bildschirmseite mit Kontrollelementen zu versehen.

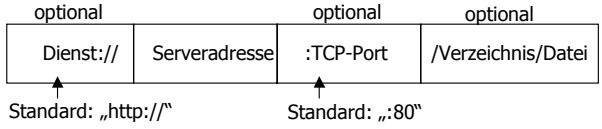
---

[13] Siehe http://www.dnssec.net/why-deploy-dnssec, letzter Zugriff am 07.07.2009.

[14] Siehe Word Wide Web Consortium, www.w3c.org.

Der Stil einer Oberfläche wird durch eine HTML-Ergänzung, CSS (Cascading Style Sheets) genannt, gesteuert. Mit CSS kann man das Look&Feel einer Oberfläche unabhängig von den HTML-Seiten beschreiben. Da aber HTML und CSS nicht zu unseren Schwerpunkten zählen, werden diese Sprachen im Folgenden nicht weiter betrachtet.

Für die Kommunikation ist von Bedeutung, dass der ganze HTML-Code in lesbarer Form vom Web-Server zum Web-Client in den HTTP-PDUs übertragen wird. Weiterhin gibt es Möglichkeiten, in den PDUs auch Parameter aus den Eingabefeldern der HTML-Seiten an den Server zu übertragen.



Beispiel 1: www.isys-software.de
Beispiel 2: http://www.hm.edu:8080/mandl/doc

Abbildung 6-12: URL-Aufbau

Die Adressierung der HTML-Seiten auf den Web-Servern erfolgt über Uniform Resource Locators (URL) oder allgemeingültiger über sog. Uniform Resource Identifier (URI). URI ist ein allgemeinerer Begriff für alle Adressierungsmuster, die im WWW unterstützt werden. Der Aufbau einer URL ist in Abbildung 6-12 dargestellt. Eine URL besteht aus den folgenden Feldern:

- *Dienst*: Gibt den Diensttyp an. Bei Webseiten, die über das HTTP-Protokoll adressiert werden, ist dies HTTP.
- *Server*: Hostname des Web-Servers.
- *TCP-Port*: Portnummer, unter welcher der Web-Server auf Anfragen wartet. Das Feld muss nicht angegeben werden. Der Standardport ist 80.
- *Verzeichnis*/*Datei*: Name des Verzeichnisses und der angesprochenen Datei. Es wird unter einem konfigurierten Pfad gesucht.

Bei den einzelnen Feldern ist auf die Trennzeichen zu achten. Zwischen Servername und Portnummer muss z.B. ein „:“ stehen.

Dieser Adressierungsmechanismus wurde für statische Webseiten erfunden, dient aber heute auch zur Adressierung von Anwendungen, die dynamisch HTML-Content erzeugen und/oder komplexe Operationen im Server ausführen.

Wir betrachten nun der Vollständigkeit halber die Modell-Komponenten Web-Server und Web-Browser und konzentrieren uns anschließend vorwiegend auf das Protokoll HTTP.

6.3.2 Web-Server und Proxy-Cache-Server

Ein Web-Server wartet auf ankommende Verbindungsaufbauwünsche standardmäßig auf TCP-Port 80, indem er die Socket-Funktion *listen* aufruft. Ein klassischer TCP-Verbindungsaufbauwunsch wird entgegengenommen. Steht die Verbindung, wird vom Web-Browser ein HTTP-Request erzeugt, der vom Web-Server empfangen und üblicherweise in einem eigenen Thread oder Prozess bearbeitet wird. Das gewünschte Dokument wird lokalisiert (wir nehmen hier die Adressierung einer statischen Web-Seite an, wozu im konfigurierten *Root-Verzeichnis* des Web-Servers gesucht wird.

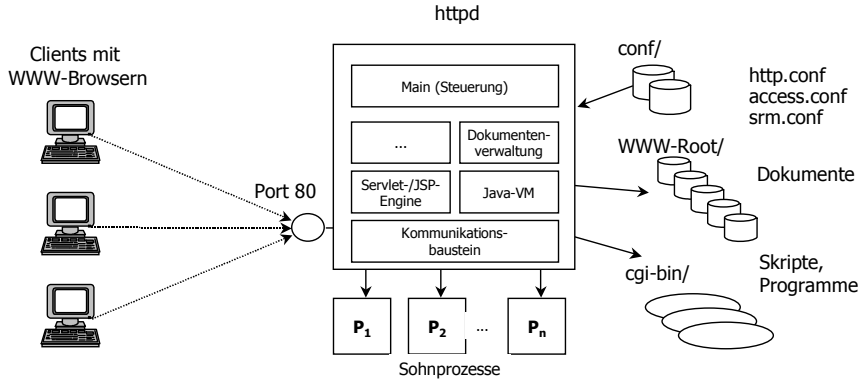


Abbildung 6-13: Grobe Architektur eines Webservers

Web-Server sind heute in Application-Servern integriert (siehe BEA Weblogic oder IBM Websphere) und auch als Standalone-Systeme verfügbar. Der bekannteste und am weitesten verbreitete Web-Server ist der Apache HTTP-Server. Weitere Web-Server sind der NCSA-Server vom National Center of Supercomputing Applications und der CERN-Server vom W3C. Neben dem Apache-HTTP-Server ist der Microsoft Internet Information Server (IIS) vor allem bei Microsoft-Kunden sehr verbreitet.[15]

Ein Web-Server (bei Apache auch als *httpd*-Hintergrundprozess ablaufend) enthält im Allgemeinen neben einer Steuerungskomponente eine Komponente zur Dokumentenverwaltung, evtl. eine Engine für die Servlet-/JSP-Bearbeitung, ASP-Bearbeitung, PHP-Bearbeitung und einiges mehr[16] (vgl. Abbildung 6-13). Bei Java-

[15] Siehe www.bea-systems.com, www.ibm.com, www.apache.org.

[16] Servlets und Java Server Pages (JSP), Active Server Pages, PHP usw. sind serverseitige Programmiertechniken. Hier handelt es sich um Scripting-Sprachen, die im Server ausgeführt werden und daher zu interpretieren sind. Daher werden hierfür Verarbeitungskomponenten im Web-Server benötigt. Es gibt noch einige weitere Webtechniken.

lastigen Web-Servern ist auch eine JVM im Spiel. Die Server erzeugen für die Bearbeitung bestimmter Anfragen auch noch zusätzliche Prozesse. Insbesondere bei der relativ alten serverseitigen Programmiertechnik CGI[17] wird für jede Anfrage ein eigener Sohnprozess zur Bearbeitung erzeugt.

Die HTML-Dokumente liegen im Filesystem unter einem Root-Verzeichnis. Ist das voreingestellte Root-Verzeichnis zum Beispiel */usr/local/httpd/docs*, so setzt sich der volle Pfadname eines adressierten Dokuments mit dem Namen */mandl/beispiel1-.html* wie folgt zusammen:

```
/usr/local/httpd/docs/mandl/beispiel1.html
```

Bei Adressierung eines Verzeichnisses anstatt einer Datei wird im Verzeichnis die sog. Indexdatei *index.html* oder *welcome.html* identifiziert.

Der Web-Server liest seine Konfigurationsdaten von Dateien. Das Verhalten eines Web-Servers kann also durch Konfigurierung verändert werden. Es gibt einige Konfigurationsdateien hierfür (hier am Beispiel des Apache-Web-Servers):

- *httpd.conf* ist die Serverkonfigurationsdatei
- *srm.conf* ist die Ressourcenkonfigurationsdatei
- *access.conf* ermöglicht die Vergabe von Zugriffsrechten
- *mime.types* legt die unterstützten MIME-Typen fest[18]

Beispiele von Konfigurationsparametern in *httpd.conf* sind u.a:

- *AgentLog dateiname*: Datei für Protokollierung
- *Port nummer* : Nummer des TCP-Listen-Ports
- *ServerRoot pfadname:* Wurzelverzeichnis

Die weitere Vertiefung der Funktionsweise und Konfigurierbarkeit von Web-Servern kann den einschlägigen Produktinformationen entnommen werden.

Aus Gründen der Performance-Optimierung, wegen Sicherheitsaspekten bei einem Firewall-Einsatz und aufgrund einer begrenzten Anzahl an vorhandenen IP-Adressen kann man auch einen *Proxy-Cache-Server* einsetzen. Typische Proxy-Cache-Server sind *Squid* (Linux) oder *Apache Cache*. Die Arbeitsweise eines Cache-Proxy ist definiert. Wie Abbildung 6-14 zeigt, wird der Proxy-Cache-Server zwischen die Web-Clients und dem Internet-Zugang gelegt.

---

[17] CGI steht für Common Gateway Interface und ist einer der ersten Standards für den Aufruf externer Funktionen aus dem Web-Server heraus zur Bearbeitung dynamischer Web-Seiten.

[18] MIME-Typen (Multipurpose Internet Mail Extensions) sind standardisierte Dateitypen unterschiedlichsten Inhalts, die ursprünglich aus SMTP stammen.

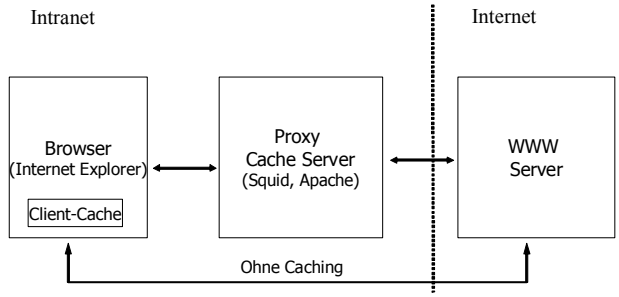


Abbildung 6-14: HTTP-Proxy-Caching

Er wird üblicherweise in einer DMZ positioniert und speichert bereits angefragte Webseiten nach vorgegebenen Strategien. In Mehrbenutzerumgebungen kann damit ein „gecachtes" Dokument mehreren Web-Clients zur Verfügung gestellt werden, was zu einer Erhöhung der Hitrate führt. Die Regeln für das Caching sind in den einzelnen Produkten konfigurierbar.

6.3.3 Web-Browser

Web-Browser stellen die Clients des Web-Kommunikationsmodells dar und bieten dem Anwender eine komfortable Benutzeroberfläche. Die gängigsten Web-Browser sind derzeit der Microsoft Internet Explorer, Netscape Communicator und Mozilla.

Web-Browser sind komplexe Softwareprogramme mit vielen Möglichkeiten der Einstellung. Die Web-Browser der einzelnen Hersteller unterstützen zwar alle HTML, aber sie arbeiten nicht immer gleich. Dies schafft Kompatibilitätsprobleme bei der Entwicklung von Web-Anwendungen.

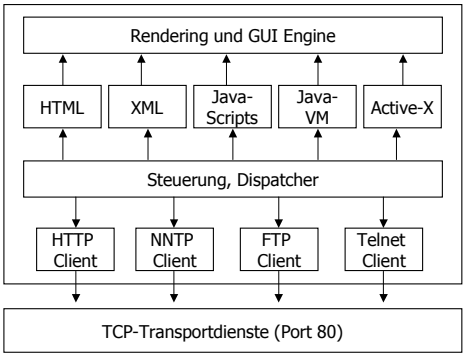


Abbildung 6-15: Grobe Architektur eines Browsers

In Abbildung 6-15 ist eine grobe Architekturskizze für einen Web-Browser dargestellt. Ganz unten ist der TCP-Transportdienst, über dem mehrere Protokolle angesiedelt sind. Ein Web-Browser unterstützt nicht nur das HTTP-Protokoll (Dienst = http://), sondern man kann meist auch noch Filetransfer (FTP, ftp://), News (NNTP) und weitere Protokolle nutzen. In Richtung der Präsentationsschicht ist eine Rendering- und GUI-Engine zur Ausgabe im Browser vorhanden. Es gibt diverse Ausgabekomponenten zur Darstellung von HTML, XML (eXtended Markup Language)[19] usw. Da auch clientseitige Scripts ausgeführt werden können, sind in einem Browser auch Interpreter wie Java und JavaScript sowie im Microsoft Internet Explorer Active-X verfügbar.

Ein Web-Client nimmt normalerweise an der Oberfläche eine URL entgegen und sendet diese nach der Bestätigung durch den Benutzer an den Server. Hierzu benutzt er das HTTP-Protokoll.

6.3.4 HTTP-Protokoll

HTTP (RFC 2616) bildet als Kommunikationsprotokoll zwischen Web-Client- und Server das Rückgrat des Webs. Man muss HTTP verstehen, wenn man gute Web-Anwendungen entwickeln will. HTTP ist ein *verbindungsorientiertes* und *zustandsloses* Request-/Response-Protokoll. Weder der Sender noch der Empfänger merken sich irgendwelche Stati zur Kommunikation. Ein Request ist mit einem Response vollständig abgearbeitet.

In Abbildung 6-16 ist der Protokollstack skizziert. Ein Web-Browser nutzt HTTP, das auf TCP aufsetzt. Dies bedeutet, dass für eine HTTP-Kommunikation ein TCP-Verbindungsaufbau im Sinne eines Drei-Wege-Handshakes durchgeführt werden muss.

HTTP unterstützt standardmäßig keine Komprimierung, alles wird im ASCII-Text unterstützt. Für jede HTML-Seite und jede Grafik, die vom Server gelesen wird, wird in HTTP V1.0 standardmäßig eine TCP-Verbindung zwischen Web-Browser und -Server aufgebaut. Dies hat sich aber ab der HTTP-Version V1.1 verändert, wo man die Wahlmöglichkeit hat, eine Verbindung aufrecht zu erhalten oder zu beenden.

Nebenbei sei noch erwähnt, dass HTTP die Nutzung von MIME-Bezeichnern (Medientypen) für den Inhalt der zu übertragenden Daten unterstützt. Es können also beliebige Dateitypen, die einen MIME-Typ besitzen und vom Web-Browser verarbeitet werden können, über HTTP übertragen werden. Ein Web-Client gibt im Request an, welche Formate er verarbeiten kann. Der Server teilt dem Web-Client

[19] Zu XML siehe www.xml.org.

in der Response mit, welches Format der gesendete Entity-Abschnitt nutzt (z.B. HTML, GIF, JPEG-Bilder, PDF). Medientypen werden mit der Syntax <typ>/-<untertyp> bezeichnet (Beispiele: Accept: */* = alle Medientypen, Accept: text/* = alle Texttypen akzeptieren).

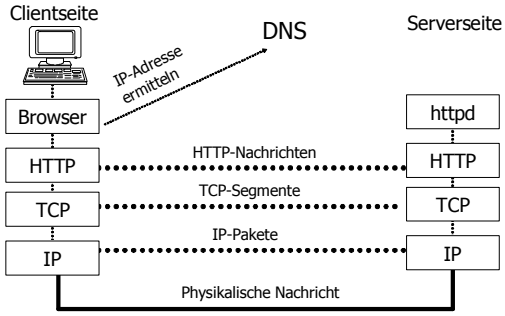


Abbildung 6-16: HTTP-Protokollstack

Für alle Operationen werden im HTTP-Protokoll nur zwei PDU-Typen, die HTTP-Request- und die HTTP-Response-PDU benötigt. Die PDUs sind sehr generisch aufgebaut und können daher mit vielen Spezialinformationen (hier Header genannt) versehen werden. Die wichtigsten *Protokolloperationen* sind GET und POST. Mit GET und POST können Dokumente vom Web-Server angefragt werden. Eine entsprechende HTTP-Request-PDU wird vom Web-Client zum Web-Server gesendet, eine HTTP-Response-PDU mit dem Ergebnis in der umgekehrten Richtung.

Abbildung 6-17 zeigt den grundsätzlichen Ablauf der Kommunikation. Man sieht, dass vor der HTTP-Kommunikation ein Verbindungsaufbau durchgeführt wird und nach dem Empfang der Response-PDU die Verbindung (vom Client) wieder abgebaut wird.

In der Abbildung 6-17 ist der Aufbau der HTTP-PDU dargestellt. Ein HTTP-Request besteht aus folgenden Teilen:

- *Request-Line (Anfragezeile):* Dies ist der „Haupt-Header" mit der Angabe der Operation (hier GET) und der adressierten URL. In diesem Teil werden auch – falls vorhanden – die Eingabeparameter aus den HTML-Formularen als (name, value)-Paare mit übertragen.
- *Liste von Request-Headern*: Hier gibt es eine Fülle von HTTP-Headern. Beispiel *Accept: text/html* mit der Bedeutung, dass der Browser nur HTML-Dateien unterstützt.
- *Entity-Body*: Nutzdatenteil der Nachricht, der bei GET leer ist. Bei POST werden hier, falls vorhanden, Eingabeparameter aus den HTML-Formularen übertragen.

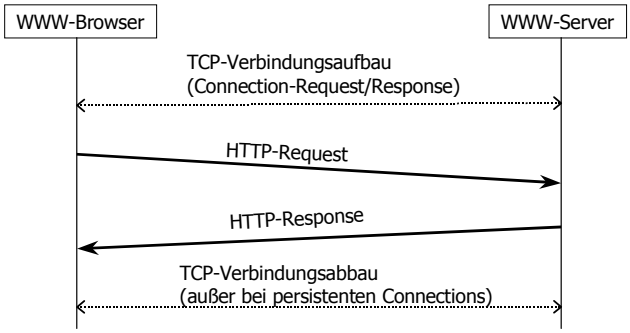


Abbildung 6-17: HTTP-Kommunikation

Eine HTTP-Response besteht aus folgenden Feldern (siehe Abbildung 6-18):

- *Status Line*: Hier wird die HTTP-Version, der Statuscode des Requests sowie ein Text zum Status (im Beispiel: HTTP/1.0 200 OK) angegeben.
- *Liste von Response-Headern*: Auch hier sind viele Header möglich. In der Abbildung ist z.B. der Inhalt der HTTP-Response im Body mit dem Header *Content-type: text/html* angegeben. Mit dem Header *Content-length: 2000* wird die Länge der Antwort im Body angezeigt.
- *Entity-Body:* Dieser Teil enthält die Nutzdaten des Ergebnisses, also in der Regel den HTML-Code.

Da die Nachrichten im Textformat übertragen werden, sind Trennzeichen erforderlich. Zwischen den einzelnen PDU-Feldern werden Leerzeichen eingefügt. Die Header sowie der Entity Body werden mit den ASCII-Zeichen CR (Carriage Return) und LF (Line Feed) abgetrennt. Headernamen und Werte werden durch „:“ voneinander getrennt.

Die HTTP-Spezifikation (V1.1) enthält viele HTTP-Header, die von Browsern, Web-Servern und auch von einem Proxy-Cache-Server in die HTTP-PDU eingetragen werden können. Man unterscheidet:

- Allgemeine Header für Web-Client und -Server
- Request Header für Web-Client
- Response Header für Web-Server
- Entity Header für Web-Client und -Server

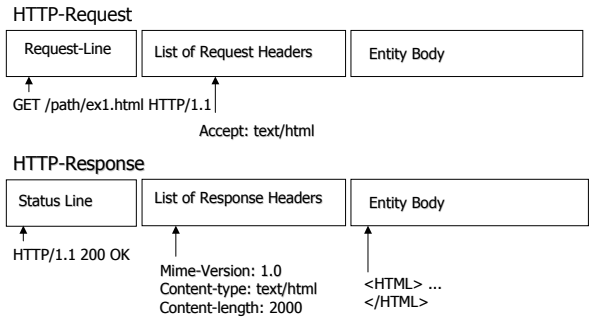


Abbildung 6-18: HTTP-PDUs

Der Aufbau von Headern (Groß-/Kleinschreibung spielt keine Rolle) sieht wie folgt aus:

```
<header-name>: <header-value>
```

Ein Header kann sich über mehrere Zeilen erstrecken, Folgezeilen müssen mit Blank oder Tab beginnen. Im Folgenden werden nochmals einige Beispiele zu den einzelnen Headertypen zusammengefasst:

Beispiele für allgemeine Header: Cache-Control-Direktive: Definiert in einer Liste Caching-Direktiven (Liste mit Komma abgetrennt zulässig):

- Im Request-Header: *Cache-Control: no-cache* bedeutet, dass kein Cache verwendet werden soll.
- Im Response-Header: *Cache-Control: no-cache* bedeutet, dass die Web-Seite nicht in den Cache gelegt werden soll.

Beispiele für Request-Header:

- Accept: <typ>/<untertyp> (Liste ist zulässig), z.B. *Accept: text/** bedeutet, dass alle Textuntertypen erlaubt sind (html, plain, enriched,...).
- Cookie: <name>=<wert>[20] (Liste mit Strichpunkten abgetrennt ist zulässig). Speichert das (name, value)-Paar für die aktuelle URL, z.B. *Cookie: mps=1234123*

Beispiele für Response-Header:

- Server: <zeichenkette>: Name und Versionsnummer des Servers, z.B. Server: *NSCA/2.0*

[20] Cookies sind Dateien, die vom Browser im Filesystem des Web-Clients gespeichert werden. Sie können vom Web-Server angelegt werden, wenn der Browser entsprechend konfiguriert ist. Man kann mit Cookies Session-übergreifend aufbewahren. Bei erneuter Anwahl einer URL kann das Cookie mit dem HTTP-Request gesendet werden.

– Set-Cookie: <name>=<wert> [;option,...]: Setzt ein Cookie für die aktuelle URL, z.B. *Set-Cookie: mandl=1234; expires=datum* setzt ein Cookie mit Namen „mandl" und dem Wert „1234" mit einem Ablaufdatum von „datum"

Beispiel für Entity-Header:

– Content-Length: n: Länge des übertragenen Entity Bodies in Byte, z.B. *Content-length:256*

Betrachten wir nun die Kommunikation etwas detaillierter. Der clientseitige Ablauf einer HTTP-Kommunikation sieht wie folgt aus:

– Der Web-Client ermittelt den Hostnamen des Servers aus der URL und besorgt sich über DNS die IP-Adresse des Web-Servers. Es kann aber auch eine IP-Adresse in der URL stehen, dann entfällt der DNS-Lookup.
– Der Client baut aktiv eine TCP-Transportverbindung zum Socket des Web-Servers mit Portnummer 80 auf.
– Der Web-Server nimmt die Verbindung an.
– Der Web-Client sendet den HTTP-Befehl (z.B. *GET /index/html HTTP/1.1*).
– Der Web-Client sendet weitere optionale HTTP-Header (eigene Konfiguration und akzeptierte Dokument-Formate,...), z.B. *Accept: image/gif.*
– Der Web-Client sendet eine Leerzeile (Ende des Requests anzeigen).
– Der Web-Client sendet evtl. noch zusätzliche Daten als Parameter (Input-Felder aus HTML-Formularen).

Aus Sicht eines Web-Servers sieht der Ablauf der HTTP-Kommunikation wie folgt aus:

– Nach der Bearbeitung des Requests sendet der Server eine Statuszeile (Response Header) mit drei Feldern (Version, Statuscode, lesbarer Text), z.B. *HTTP/1.1 200 OK.*
– Anschließend sendet der Server HTTP-Header wie z.B. *Content-type:text/html* und *Content-length: 2482.*
– Anschließend wird eine Leerzeile (CR LF) gesendet und das angeforderte Dokument.
– Falls vom Client kein Header „*Connection:Keep alive*" gesendet wurde, wird die TCP-Verbindung wieder abgebaut. Ab HTTP 1.1 wird „*Keep alive*" standardmäßig verwendet. Dadurch können Applets, Grafiken, Rahmen und sonstige Objekte über dieselbe Verbindung übertragen werden.

Neben den Protokolloperationen GET und POST gibt es auch noch andere Operationen:

– *LINK*: Fordert an, dass die Header-Informationen zu einem Dokument auf dem Server in Beziehung gesetzt werden.
– *UNLINK*: Inverse Operation zu LINK.
– *PUT*: Gegenstück zu GET, sendet ein Dokument vom Client zum Server im Body-Abschnitt.

- *DELETE*: Fordert die Entfernung eines Dokuments auf dem Server an.
- *OPTIONS*: Fordert vom Server Kommunikationsoptionen an.
- *TRACE*: Fordert vom Server das Zurücksenden des gesendeten Body-Abschnitts.

Bei der GET-Operation ist der Body-Bereich immer leer und zur Übertragung von Input-Parametern für serverseitige Programme wird die URL ergänzt. Nach einem Fragezeichen kommen dann die Parameter aus den Eingabefeldern.

Beispiel: `GET /bin-cgi/my.pl?kdr=12345&name=mandl HTTP 1/1`

Im Beispiel aus Abbildung 6-19 sendet der Server nach Abarbeitung des CGI-Perlscripts *my.pl* die Standardausgabe des Scripts als Response zurück.

Weiterhin soll noch die HEAD-Operation erwähnt werden. Diese Operation funktioniert wie die GET-Operation, nur dass hier vom Web-Server kein Ergebnisdokument gesendet wird. Ein sinnvoller Einsatz für die HEAD-Operation ist z.B. dann gegeben, wenn der Web-Client nur das Datum der letzten Änderung oder die Dokumentgröße wissen will.

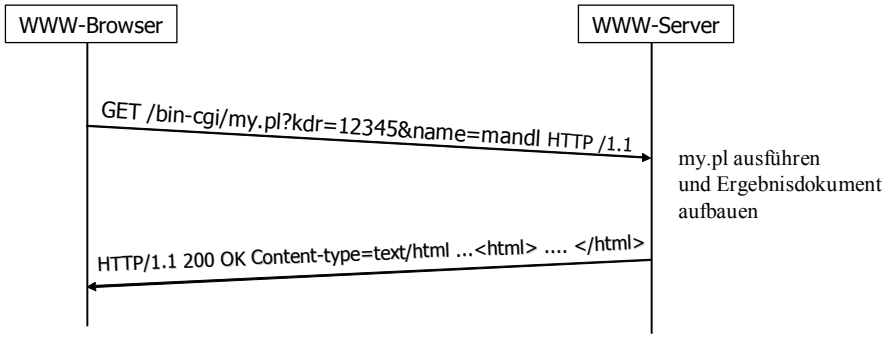


Abbildung 6-19: Ablauf der Get-Operation

Über die POST-Methode können in einem HTTP-Request im Body Daten an den Server zur Weiterverarbeitung über ein Programm gesendet werden. Der Server gibt die Daten an das mit der angegebenen URL adressierte Programm weiter. Am prinzipiellen Aufbau der HTTP-Response-PDU ändert sich im Vergleich zur GET-Operation nichts.

Beispiel zu POST:

```
POST /cgi-bin/my2.pl HTTP/1.1
  Content-type: application/x-www-form-urlencoded
  Content-length: 20
  monat=oktober&tag=24
  <Daten im Body>
```

Der Server sendet in der HTTP-Response-Nachricht einen dreistelligen Statuscode an den Client. Der Statuscode gibt den Erfolg bzw. den Grund des Misserfolgs

eines HTTP-Requests an. Der Statuscode wird vom Web-Server oder vom aufgerufenen, serverseitigen Programm erzeugt.

Es gibt Statuscodes, die in der HTTP-Spezifikation festgelegt sind, sowie auch individuelle Codes der einzelnen Server-Implementierungen. Der Statuscode 100 bedeudet z.B., dass der Einleitungsteil des Requests beim Server angekommen ist und der Rest gesendet werden kann. Der Statuscode 200 zeigt an, dass der Client-Request erfolgreich abgearbeitet wurde. Statuscode 400 deutet auf einen Syntaxfehler im Request hin, und Statuscode 500 signalisiert einen Fehler im WWW-Server. Der Web-Browser oder ein anderer Client, der HTTP beherrscht, kann die Codes analysieren und entsprechende Maßnahmen ausführen, z.B. eine Fehlermeldung auf ein Window schreiben.

6.3.5 HTTPS, SSL und TLS

Für sicherheitskritische Webzugriffe verwendet man heute eine Erweiterung von HTTP, das sog. HTTPS-Protokoll. Dabei erfolgt eine Verschlüsselung der Daten über SSL bzw TLS. HTTPS-Verbindungen nutzen auch TCP als Transportprotokoll. Als TCP-Port ist für HTTPS 443 reserviert.

Bei *Transport Layer Security* (TLS) bzw. dessen Vorgängerversion *Secure Sockets Layer* (SSL) handelt es sich um ein Verschlüsselungsprotokoll für die Datenübertragung, das vorwiegend für das Internet entwickelt wurde. SSL wurde ursprünglich von der Firma Netscape entwickelt und 1996 an die IETF (Internet Engineering Task Force) übergeben. Dort wurde es zu TLS, das im Jahre 1999 in einer ersten Version im RFC 2246 verabschiedet wurde, weiterentwickelt. TLS ist die standardisierte Weiterentwicklung der SSL-Version 3.0. Heute wird SSL/TLS von allen gängigen Web-Browsern und Web-Servern unterstützt.

SSL/TLS setzt auf TCP als Transportprotokoll auf, ist also ein verbindungsorientiertes Protokoll. Es verwendet sowohl symmetrische als auch asysmmetrische Kryptografie-Verfahren[21]. Unter Verwendung eines Schlüsselaustauschverfahrens findet in der Verbindungsaufbauphase zunächst in einem Handshake-Protokoll ein Austausch geheimer Schlüssel statt, um danach in einer weiteren Stufe die jeweiligen öffentlichen Schlüssel in Form digitaler Zertifikate (X.509-Zertifikat[22]) sicher austauschen zu können. Die öffentlichen Schlüssel beider Kommunikationspartner (in diesem Fall der Web-Browser und der Web-Server) werden dann zur Verschlüsselung der Nutzdaten verwendet, um eine möglichst abhörsichere Kommunikation zu gewährleisten.

---

[21] Symmetrische Kryprografie-Verfahren sind z.B. DES (Data Encryption Standard) und IDEA (International Data Encryption Algorithm), ein asymmetrisches Verfahren ist z.B. RSA (nach den Erfindern Ronald L. Rivest, Adi Shamir und Leonard Adleman).

[22] X.509 ist ein ITU-T-Standard für digitale Zertifikate.

In Abbildung 6-20 ist der Austausch der Schlüsselinformation zwischen einem Web-Browser und einem Web-Server bei Nutzung von HTTPS und damit SSL/TLS vereinfacht dargestellt, wobei hier bereits eine TCP-Verbindung besteht. Nach der Berechnung des Session-Schlüssels beginnt erst die Übertragung der Nutzdaten, man sieht also einen nicht unerheblichen Overhead im Vergleich zu HTTP.

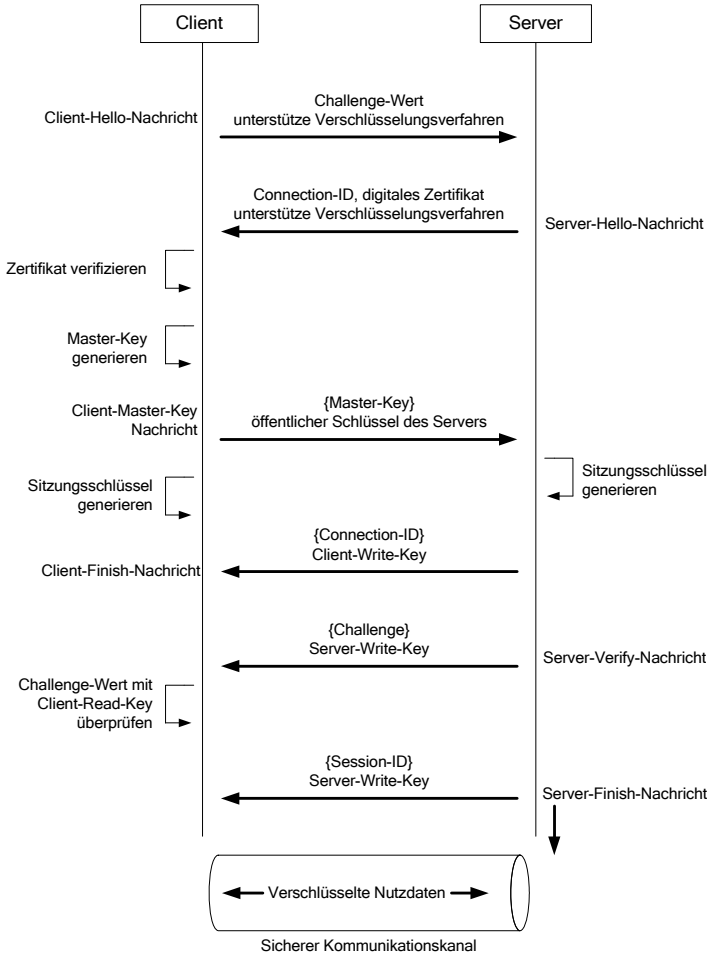


Abbildung 6-20: SSL-Handshake nach (Hansen 2005)

Ein Problem beim Einsatz von HTTPS ist die Notwendigkeit von signierten Zertifikaten. Ohne ein signiertes Zertifikat ist die Sicherheit nur eingeschränkt erreichbar, Angriffe wie etwa die klassische „Man-in-the-middle-Attack“ sind möglich. In der Praxis werden aus Verwaltungsgründen oft nicht signierte Zertifikate benutzt. Dadurch ist die Sicherheit stark eingeschränkt.

In Web-Anwendungen sollte man nicht alle Zugriffe über HTTPS abwickeln, da der Protokoll-Overhead doch beträchtlich ist. Sinnvoll ist der Einsatz von HTTPS bei echt sicherheitsrelevanten Zugriffen wie z.B. bei einem Login, bei dem ein Passwort übertragen wird.

Da Sicherheitsthemen in diesem Buch nur am Rande betrachtet werden, wird bzgl. HTTPS und SSL/TLS auf eine tiefergehende Diskussion verzichtet.

6.3.6 AJAX

Trotz der Fortschritte in der Web-Programmierung waren Web-Oberflächen aufgrund der blockorientierten Kommunikation zwischen Web-Browser und Web-Server nicht vergleichbar mit herkömmlichen GUI-Oberflächen. Erst durch den Einsatz von asynchronen Technologien ist man heute dabei, diese Lücke zu schließen.

Der Name AJAX (Asynchronous JavaScript and XML) wurde von Jesse James Garrett im Artikel „Ajax: A New Approach to Web Applications“ geprägt, in dem er von einem Wandel im bisherigen World Wide Web spricht. Tatsächlich fasste Garrett die schon laufende Entwicklung zum Thema *asynchrone Kommunikation im Internet* in seinem Essay zusammen, da sowohl die nötigen Basistechnologien, als auch schon erste Anwendungen in der Praxis existierten. Der Begriff AJAX steht seitdem für eine neue Generation für die Kommunikation im bisher „blockierenden“ Internet.

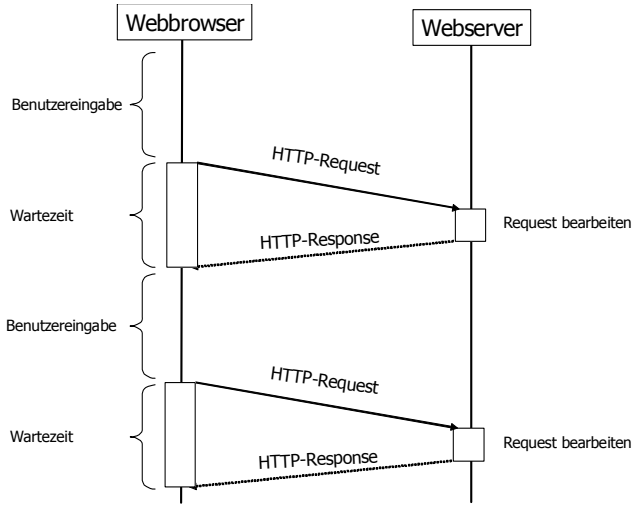


Abbildung 6-21: Herkommliche Webkommunikation

Traditionell kommuniziert ein Browser über Formulardaten mit einem Server, der nach Auswertung dieser Daten mit einer neuen HTML-Seite antwortet. In Abbildung 6-21 sieht man, dass der Benutzer nach einer Aktion warten muss,

bis seine Anfrage am Server bearbeitet, die Antwort empfangen und die neue Seite im Browser aufgebaut ist. Dies kann je nach Netz- und Serverkapazitäten im Bereich von mehreren Sekunden liegen. Die fehlende Kontrolle des Benutzers während dieser Wartezeiten lässt Webseiten oft unhandlich und träge erscheinen.

Grund hierfür ist die synchrone, blockierende Kommunikation zwischen Webbrowser und Webserver. Mit der Möglichkeit eines asynchronen Datenaustauschs innerhalb einer HTML-Seite wird versucht, dieses Problem zu lösen.

Angestoßen durch *JavaScript* (siehe unten) können nun asynchrone Anfragen an einen Webserver gesendet und die Antworten ausgewertet werden ohne, dass die Seite neu gerendert werden muss.

Wie in Abbildung 6-22 dargestellt werden die Anfragen innerhalb der Browsersoftware an die sog. AJAX-Engine übergeben. Diese regelt den Datenaustausch mit dem Server im Hintergrund und stellt das Ergebnis der HTML-Seite wieder zur Verfügung. Über eine Callback-Methode wird dabei die Seite (bzw. das darin befindliche JavaScript) über eine Antwort informiert und kann diese entsprechend verarbeiten. Es werden also nur Nutzdaten ausgetauscht und keine kompletten Seiten mehr übertragen.

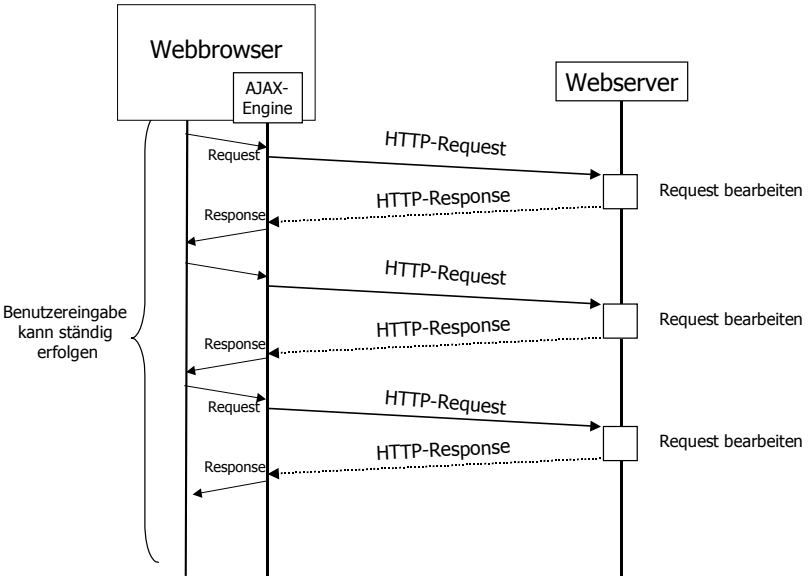


Abbildung 6-22: Webkommunikation mit AJAX

Während der Datenaustausch im Hintergrund abläuft, steht die Seite dem Benutzer weiterhin zur Verfügung. Ankommende Nachrichten werden in die bestehende Webseite, ohne offensichtliche Wartezeiten, integriert. Der Benutzer merkt also nichts vom Datenaustausch im Hintergrund und kann weiterhin mit der Webseite interagieren.

Technologien für AJAX

Bevor wir auf das eigentliche Kernstück, dem *XMLHttpRequest-Objekt*, näher eingehen, werden hier noch einige wichtige Technologien erläutert, die zum Thema AJAX gehören.

DOM. Mit Document Object Model (DOM) werden im Webbrowser alle Elemente einer (X)HTML-Seite strukturiert in einer Baumstruktur hinterlegt. Jedes Element besteht aus einem Knoten (node), welcher einen Elternknoten (parent) und meist ein oder mehrere Kindknoten(child) besitzt. Als Einstiegspunkt in den Objektbaum dient das (Wurzel-)Objekt „document".[23] Über dieses Objektmodell können nahezu alle Informationen einer XHTML-Seite abgerufen und manipuliert werden. So können dynamisch einzelne Elemente, z.B. Artikel eines Warenkorbs, auf der Seite aktualisiert werden (Mintert 2007).

XML. Als Auszeichnungssprache hat XML in der Datenkommunikation einen hohen Stellenwert erlangt. Die Notation hat den Vorteil, dass es auf eine anwendungsspezifische Sprache angepasst werden kann, erweiterbar ist, menschlich lesbar ist und mit einfachen, standardisierten Mitteln auf Wohlgeformtheit und Gültigkeit überprüft werden kann. Mit sog. Parsern, die für alle bekannten Programmiersprachen verfügbar sind, kann programmatisch auf die übertragenen Inhalte zugegriffen werden. Über Transformationssprachen, wie z.B. XSLT, können dieselben Daten mittels sog. Stylesheets direkt in verschiedene Zielformate (XHTML, PDF, ...) umgewandelt werden.[24]

(X)HTML. Als Darstellungssprache wird auf das bereits bekannte HTML bzw. auf das XML-konforme XHTML zurückgegriffen. Wird XHTML als Datenaustauschformat gewählt ist es möglich, die Antwort direkt in den DOM-Baum „einzuspielen".

JavaScript. Die Programmiersprache JavaScript wird verwendet, um clientseitig die verschiedenen Browser-Komponenten miteinander zu verbinden. JavaScript ermöglicht den Zugriff auf Elemente des DOM-Baums, um die Darstellung dynamisch zu ändern oder Informationen aus der Seite auszulesen. Eine wichtige Aufgabe ist die Abwicklung der asynchronen Kommunikation über das *XMLHttpRequest*-Objekt (siehe unten).

JSON. Als Austauschformat ist JSON (JavaScript Object Notation) eine sehr beliebte Alternative zu XML geworden. JSON ist eine literale Objektnotation, die es ermöglicht Objekte (und deren Attribute) in ein für Menschen lesbares Textformat umzuwandeln (und natürlich wieder zu dekodieren). Ein Vorteil von JSON ist, dass es bereits in JavaScript eingebettet und daher clientseitig automatisch verfüg-

---

[23] DOM ist generell für die hierarchische Strukturierung von XML-Dokumenten konzipiert, wird hier aber vor allem im Kontext von XHTML verwendet.

[24] Vgl. http://www.w3.org/XML/, http://www.w3.org/TR/xslt.

bar ist. Über die JavaScript-Funktion *eval()* kann ein Objekt aus einer empfangenen JSON-Nachricht aufgebaut werden. Seine Attribute können danach über die normale JavaScript-Notation ausgelesen werden. JSON-Implementierungen finden sich, trotz JavaScript im Namen, natürlich auch für alle bekannten Programmiersprachen, so dass hier auch serverseitig keine Einschränkungen bestehen. Ein Vorteil von JSON gegenüber XML liegt in der kompakten Form der Kodierung.

CSS. Als Standard für die Formatierung von Websites wird heutzutage CSS (Cascading Style Sheets) eingesetzt. (X)HTML-Elementen einer Seite werden hier verschiedene Styleklassen zugeordnet, in denen dann die eigentliche Formatierung festgelegt wird. Die Darstellung wird also vom Inhalt getrennt. Da CSS-Styles über den DOM-Baum zugewiesen und verändert werden können, können hiermit die angezeigte Seite inkl. aller Elemente hinsichtlich Formatierung und Design dynamisch verändert werden.

Tabelle 6-4: Eigenschaften, Methoden des XMLHttpRequest-Objekts nach (Mintert 2007)

Name	Beschreibung
readyState	Der Status des Request-Objekts
onReadyStateChange()	Event-Handler, der gerufen wird, wenn sich der Status des Request-Objekts (readyState) ändert.
responseText	Die Nutzdaten der Antwort als Text.
responseXML	Die Daten als DOM-Objekt (Typ document). Ist nur gefüllt, wenn der Server die Daten als XML sendet.
status	Der HTTP-Status-Code der Anfrage.
open(method, url, async, user, pwd)	Die Methode bereitet das Request-Objekt auf eine Übertragung an die Adresse URL mit der HTTP-Protokolloperation (GET/POST) vor. Zusätzlich wird angegeben ob die Anfrage asynchron oder synchron gesendet wird (true/false). Optional können ein User und ein Passwort zur Authentifizierung über HTTP angegeben werden.
abort()	Bricht einen laufenden Request ab.
send(requestBody)	Sendet den über *open()* initialisierten Request mit dem übergebenen responseBody(nur bei POST) ab.
setRequestHeader(name,wert)	Setzt ein Request-Header-Attribut.

XMLHttpRequest

Als zentraler Baustein von AJAX muss das *XMLHttpRequest*-Objekt gesehen werden. Ursprünglich wurde die Technologie von Microsoft im Rahmen des Prog-

ramms *Outlook Web Access* eingeführt. Man benötigte eine Möglichkeit, nur „neue" E-Mails zu übertragen und auf der Webseite anzuzeigen. Vorerst integriert als *ActiveX*-Objekt im Internet Explorer, fand es jedoch schon relativ bald den Einstieg in die übrigen Browser (Mintert 2007). Inzwischen gibt es einen Spezifikationsvorschlag des W3C-Konsortiums.[25]

Die Eigenschaften und Methoden eines XMLHttpRequest-Objekts sind in Tabelle 6-4 zusammengefasst. Neben den typischen Kommunikationsprimitiven (*open, send, abort*) stellt das Objekt einen sog. Event-Handler zu Verfügung. Diesem wird der Verweis auf eine selbst zu implementierende sog. Callback-Funktion übergeben, die jedes Mal aufgerufen wird, wenn sich der Status des *XMLHttpRequest*-Objekts bzw. des Datenaustauschs verändert hat. Der Zustand muss also nicht mehr aktiv überwacht und nachgefragt werden, sondern kann bei Aufruf der Callback-Funktion behandelt werden. Über das Attribut *readyState* kann daraufhin der Status abgefragt und entsprechend in der Anwendung darauf reagiert werden (vgl. Tabelle 6-5).

Tabelle 6-5: Statuscodes der Eigenschaft readyState nach (Mintert 2007)

Wert	Bezeichnung	Beschreibung
0	UNSENT	Status nach der Instanziierung des Objekts.
1	OPENED	Das Objekt ist bereit, die Anfrage zu senden.
2	HEADERS_RECEIVED	Die Anfrage wurde gesendet.
3	LOADING	Header und Status sind verfügbar. Der Message-Body wird empfangen.
4	DONE	Die Anfrage ist beendet. Alle Daten liegen nun vor und können abgerufen werden.

Von Interesse ist insbesondere der Statuscode 4, der besagt, dass die asynchrone Anfrage bearbeitet wurde, das Ergebnis eingetroffen ist und abgefragt sowie verarbeitet werden kann. Hierbei ist jedoch noch nicht klar, ob das Ergebnis erfolgreich ist.

Leider konnten sich die Hersteller nicht auf eine einheitliche Integration des *XMLHttpRequest*-Objekts in den Browser einigen. Es bestehen also kleinere Unterschiede in den Implementierungen (ActiveX vs. native Einbindung), welche insbesondere bei der Instanzierung des Objekts beachtet werden müssen. Damit alle Browser bedient werden können, ist also eine Zusatzprogrammierung erforderlich. Folgendes JavaScript-Codestück zeigt eine einfache Browserweiche, die je nach Browser den richtigen Aufruf wählt.

[25] Siehe http://www.w3.org/TR/XMLHttpRequest/.

Beispiel Browserweiche:

```
var xmlhttpRequest;
if(window.XMLHttpRequest) {
// Für Mozilla, Opera, Safari, MS IE 7, etc.
   xmlhttpRequest = new XMLHttpRequest();
}else if(window.ActiveXObject) {
// Für MS IE 5, 6
   try {
      xmlhttpRequest = new ActiveXObject("Msxml2.XMLHTTP");
   } catch (e) {
      try {
         xmlhttpRequest = new ActiveXObject("Microsoft.XMLHTTP");
      }  catch (e) {}
   }
}
```

Eine detaillierte Darstellung weiterer Unterschiede der verschiedenen Browser findet sich u. a. in (Mintert 2007).

Argumente für und gegen AJAX

AJAX bringt einige Vor- und Nachteile mit sich, welche hier auszugsweise besprochen werden sollen. Der größte Vorteil ist wohl in der Benutzerfreundlichkeit zu sehen. Mit asynchronen Anfragen ist es möglich, Felder in großen Formularen einzeln, im Hintergrund zu validieren und dem Benutzer frühzeitig Korrekturhinweise zukommen zu lassen. Ein Neuladen der Seite inkl. langer Wartezeiten entfällt. Es müssen nun also auch keine Formatierungs-/Strukturierungsdaten (HTML-Tags, CSS, etc.) mehr ausgetauscht werden, da diese nach einmaligen Laden im Client vorhanden sind. Das zu übertragende Datenvolumen sinkt also, wenn man auf übermäßige Komfortübertragungen verzichtet. Mit Hilfe Framework-Unterstützung ist es zudem einfach möglich dem Benutzer viele vorgefertigte User Interface-Elemente, wie z.B. *Drag & Drop*, anzubieten.

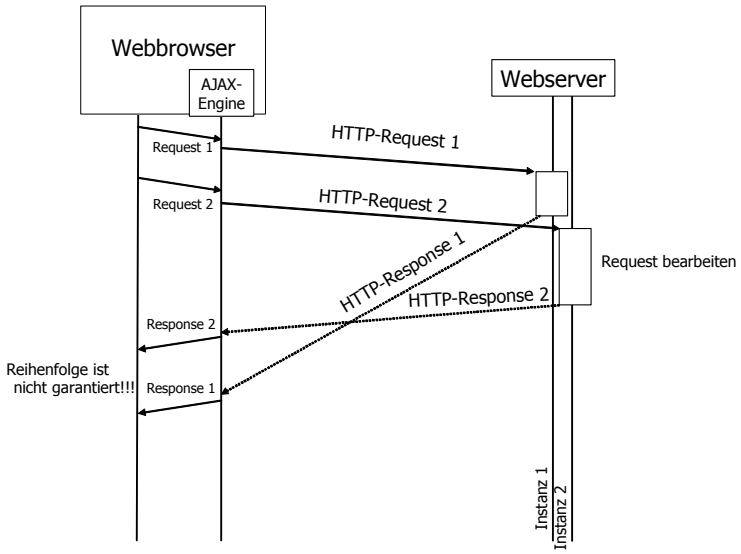


Abbildung 6-23 Konkurrierende Requests

Nachteilig ist, dass die Entwicklung mit *JavaScript* erfolgen muss. Je nach Anwendung muss komplizierte Applikationslogik im Client verankert werden. Dabei muss auch auf Timeouts im Sendemanagement geachtet werden. Die Zustandsverwaltung stellt hier eine besondere Herausforderung dar. Der Server muss jede Anfrage einzeln identifizieren und dem gespeicherten Client-State zuordnen. Dabei ist vor allem zu beachten, dass die Reihenfolge der einzelnen asynchronen Anfragen nicht garantiert werden kann. Man spricht hier auch von konkurrierenden Requests.

Tabelle 6-6 Vor- und Nachteile von AJAX

Vorteile	**Nachteile**
Es werden weniger Daten, nur Nutzdaten, ausgetauscht	Viele kleine Anfragen, zusätzliche Netzlast
Daten werden im Hintergrund ausgetauscht, Anwendung ist nicht blockiert	Kompliziertes State-Management, keine Reihenfolgegarantie.
Einzelne Validierungen von Formularfeldern schon während der Eingabe (Benutzerfreundlichkeit)	Applikationslogik muss im Browser implementiert werden (JavaScript)
Benutzerfreundliche UI-Elemente	Browser-Inkompabilitäten

Abbildung 6-23 zeigt wie Request 2 einen vorher abgesetzten Request 1 „überholt". Dies ist grundsätzlich möglich, da keine Wegegarantie besteht und Request 1 über ein langsameres Netz geleitet werden kann. Durch die verdrehte Reihenfolge kann ein inkonsistenter Zustand zwischen Client und Server entstehen.

Zu beachten ist auch, dass bei AJAX-basierten Anwendungen oft eher kleine, aber dafür viele Anfragen gestellt werden, was eine neue Ausrichtung der Systemkonfiguration des Webservers mit sich zieht. In Tabelle 6-6 sind die Vor- und Nachteile nochmals im Überblick dargestellt.

AJAX-Frameworks

Innerhalb der Internetgemeinde werden zahlreiche sog. AJAX-Frameworks, also Programmier- und Einsatzumgebungen für AJAX-nutzende Anwendungen, angeboten. Man kann zwischen client- und serverseitigen Frameworks unterscheiden.

Clientseitig haben sich hier z.B. *DOJO*, *Scriptaculous*, *JQuery* und als Basisbibliothek *Protoype* bekannt gemacht.[26] Diese Bibliotheken verfügen über innovative und benutzerfreundliche UI-Elemente.

Serverseitig wird AJAX meist in Web-Frameworks und in Komponenten integriert bereitgestellt. Vertreter finden sich zahlreich, wie z.B. *ASP.NET AJAX*, *Apache Struts*, *JSF*, *JBoss Seam*, *Wicket*, *Google Web Toolkit, Eclipse Rich AJAX Platform*[27]

Die Integration von AJAX in Frameworks erleichtert die Entwicklung von benutzerfreundlichen Anwendungen massiv. Hierzu soll noch *Google Web Toolkit* erwähnt werden, welches als einer der ersten Frameworks einen neuen Ansatz gewählt hat. Über eine eigene Java-API wird eine große Auswahl vorgefertigter UI-Komponenten bereitstellt. Der Entwickler arbeitet komplett in Java und kann vorhandene IDEs nutzen. Dabei kann man AJAX-Applikationen entwickeln, ohne mit HTML und JavaScript in Berührung zu kommen. Ein Compiler übernimmt beim Deployment die Übersetzung des Java-Codes in das entsprechende Zielformat (JavaScript und HTML). Dieser Ansatz steigert den Komfort, sowie die Wartbarkeit von Webanwendungen. Erst beim Kompilieren wird entschieden, dass aus dem Code eine Webanwendung gebaut werden soll. Ein entsprechend angepasster Compiler könnte aus dem Java-Code genauso eine lokale, funktional identische GUI-Anwendung generieren. Damit verschmelzen die Grenzen zwischen klassischen GUI- und Webanwendungen sowohl aus Benutzersicht (Bedienbarkeit) als auch aus Entwicklersicht (Programmierung).

---

[26] Siehe dojotoolkit.org, script.aculo.us, jquery.com und www.prototypejs.org.

[27] Siehe asp.net/ajax, struts.apache.org, java.sun.com/javaee/javaserverfaces, www.jboss.com/products/seam, wicket.apache.org, code.google.com/intl/de-DE/webtoolkit/, www.eclipse.org/rap/.

Es bleibt abzuwarten, welche Frameworks sich in Zukunft durchsetzen werden und welche neuen Entwicklungsprodukte auf den Markt kommen.

6.4 Electronic Mail

Ein weiterer, heute im Internet weit verbreiteter Dienst ist der E-Mail-Dienst zum Austausch von elektronischen Nachrichten (Electronic Mails). Auch hier handelt es sich um eine Client-/Serveranwendung. Ein Benutzer verwendet einen E-Mail-Client, *Mail User Agent* (MUA) genannt, um E-Mails zu senden und zu empfangen, und das Weiterreichen der E-Mails wird über E-Mail-Gateways abgewickelt. Jedem Benutzer, der E-Mails senden und empfangen möchte, wird eine elektronische Mailbox mit einer eindeutigen E-Mail-Adresse (z.B. mandl@cs.fhm.edu) in einem E-Mail-Gateway zugeordnet. Die Mailboxen werden üblicherweise in den E-Mail-Gateways von Internet-Providern verwaltet. Die E-Mail-Gateways sind Serverrechner mit speziellen Softwarebausteinen für die Mail-Abwicklung. Sie haben in der Regel eine Doppelrolle. Zum einen nehmen sie Nachrichten von den MUAs entgegen. Diese Aufgabe wird als *Mail Submission Agent* (MSA) bezeichnet. Zum anderen leiten sie Nachrichten (E-Mails) für den Transport bis zum adressierten Empfänger in der Rolle von sog. *Mail Transfer Agents* (MTA) an andere E-Mail-Gateways weiter.

E-Mail-Gateways sind im Internet meist als SMTP-Server implementiert. Ein SMTP-Server übernimmt dabei die beiden Rollen von MSA und MTA. Die SMTP-Server kommunizieren untereinander über das Protokoll SMTP (Simple Mail Transfer Protocol, well-known TCP-Port = 25)[28]. Der lesende Zugang eines Benutzers zu einer Mailbox wird über sog. Mailzugangsprotokolle ermöglicht. Ein Internet-Benutzer kann sich z.B. bei einem Internet-Provider (z.B. T-Online) eine Mailbox einrichten und erhält über das Protokoll POP3 (Post Office Protocol, Version 3, well-known TCP-Port = 110) Zugang zu seiner Mailbox.

Technisch gesehen muss der Internet-Benutzer dann zunächst eine Verbindung zwischen seinem Mail-Client (das könnte *Microsoft Outlook* oder *Mozilla Thunderbird* sein) und dem zugeordneten SMTP-Server (bekannte Serverprogramme sind *sendmail* und *qmail*) bei seinem Internet-Provider herstellen. Dies erfolgt bei privaten Internet-Benutzern meist über eine Wählverbindung auf Basis von Telekom-Diensten wie DSL. Eine entsprechende Konfiguration (Parametereinstellung) im Mail-Client-Programm muss vorgenommen werden, damit die Adressierung auch erfolgen kann. So muss z.B. die Internet-Adresse des Mail-Servers im Mail-Client eingestellt sein, und auch der nächste DNS-Server muss bekannt gemacht werden.

[28] Es gibt noch einen zweiten well-known TCP-Port mit der Nummer 587, der für dedizierte MSA-Implementierungen geschaffen wurde. Heute nutzt ein SMTP-Server meist beide Ports (25 und 587).

Wenn die Verbindung über das POP3-Protokoll aufgebaut ist, kann die Mailbox ausgelesen werden, empfangene Mails können gelesen und zum Clientrechner übertragen werden, und es können auch eigene E-Mails versendet werden. Ein Mail-Client bietet heute meist eine komfortable Unterstützung zur Darstellung und zum Schreiben von Mails. Im POP3-Protokoll werden entsprechende Protokoll-Operationen unterstützt, um die Verbindung aufzubauen und die Nachrichten auszutauschen. Die Nachrichten werden in einem definierten Textformat übertragen.

Ein anderes, etwas komplexeres Zugangsprotokoll ist IMAP4 (Internet Message Access Protocol, well-known TCP-Port = 143). IMAP4 ist ebenfalls ein Internet-Standard. Im Gegensatz zum POP3-Protokoll verbleiben die E-Mails in der Regel auf dem Mailserver, und werden nur bei Bedarf auf den Client-Rechner übertragen. IMAP4 unterstützt aber wesentlich mächtigere Verwaltungsfunktionen für die Mailboxen. IMAP4 wurde ursprünglich mit dem Ziel entwickelt, den Zugriff auf Mailboxen und Nachrichten so bereitzustellen, als wenn diese sich auf dem lokalen Rechner befänden.

Der Nachrichtenaustausch von einem Senderrechner zu einem Empfängerrechner über zwei Mail-Gateways ist in Abbildung 6-24 dargestellt. Die Mail-Clients des Senders und des Empfängers enthalten auch einen POP3- oder einen IMAP4-Client. Über diesen wird mit den entsprechenden IMAP-/POP-Servern auf den zugeordneten Mail-Gateways kommuniziert, um E-Mails abzuholen. Die beiden Mail-Clients unterhalten jeweils Mailboxen für die zugeordneten Mail-Benutzer.

POP3 wird heute meist von privaten Mailnutzern verwendet, während IMAP4 in Unternehmen eingesetzt wird. Meist wird dort dann auch ein eigenes Mail-Gateway unterhalten. POP3 und IMAP4 dienen nur zum Abholen der E-Mails von den zugeordneten Mailboxen in den SMTP-Servern. Wenn eine E-Mail vom Client abgesendet wird, erfolgt dies über das SMTP-Protokoll.

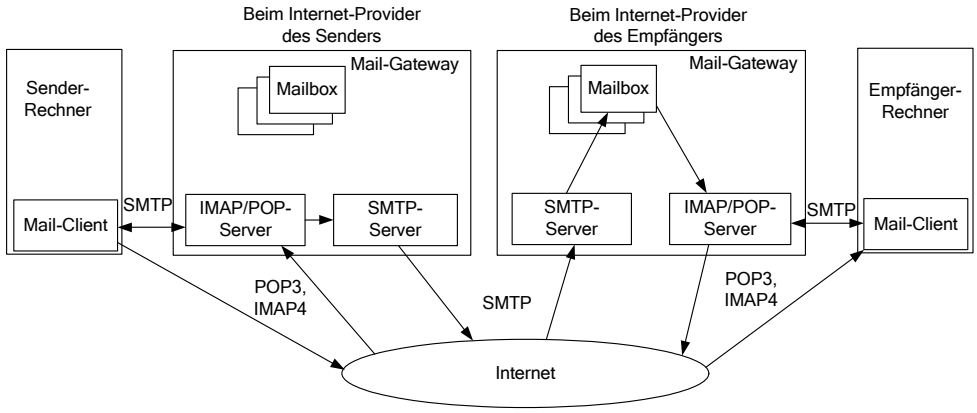


Abbildung 6-24: E-Mail-Nachrichtenaustausch im Internet

Prinzipiell funktioniert der Mail-Austausch nach dem sog. *Store-and-Forward-Prinzip*. Beim Mail-Gateway ankommende E-Mails werden zunächst in den Mailboxen zwischengespeichert, um dann bei Bedarf, wenn die Verbindung zum adressierten Mail-Benutzer aufgebaut ist, an diesen weiterzuleiten.

Das Mail-System im Internet wurde ursprünglich für Nachrichten im ASCII-Text konzipiert. Heute werden alle möglichen Objekte damit übertragen. Dies wird über einen Zusatz, den. sog. *MIME-Standard* (Multipurpose Internet Mail Extensions) ermöglicht. Über MIME können prinzipiell Daten beliebigen Binärtformats (wie Bilder, Sprache und Video) übertragen werden. Sender und Empfänger können sich über die Kodierung der Daten verständigen, wobei ein Nachrichtenfeld namens *Content-Type* verwendet wird. Es sind mehrere Kodierungsmethoden spezifiziert, die die Übertragung im textbasierten E-Mail-System ermöglichen. Die binären Objekte werden beim Sender kodiert und beim Empfänger wieder dekodiert.

6.5 Multimediale Kommunikationsanwendungen

6.5.1 Grundlagen und Anforderungen

Der Begriff *Multimedia* beschreibt in unserem Sinne digitale Informationen, die sich aus verschiedenen Medientypen zusammensetzen. Beispiele hierfür sind Textdaten, Audio- und Videodaten, Bilder, Fotos usw. Man spricht auch von Multimedia-Daten oder multimedialen Informationen. Ein Multimedia-System verarbeitet multimediale Informationen. In (Steinmetz 2000) findet sich für Multimedia-Systeme folgende Definition: *„Ein Multimediasystem ist durch die rechnergestützte, integrierte Erzeugung, Manipulation, Darstellung, Speicherung und Kommunikation von unabhängigen Informationen gekennzeichnet, die in mindestens einem kontinuierlichen und einem diskreten Medium kodiert sind."*

Beispielszenario: Vor allem Anwendungen, die Audio- und Video-Daten bearbeiten, verbreiten sich in der letzten Zeit sowohl in Unternehmen als auch in Privathaushalten immer mehr. In Abbildung 6-25 ist beispielsweise ein Intranet für ein kleines Unternehmen oder einen Haushalt skizziert, das Audio- und Video-Streaming einsetzt. Das dargestellte lokale Netzwerk verfügt über einen eigenen Medien-Server, eine Telefonanlage, ein TV-Gerät als Medien-Player, eine Stereoanlage, eine Spielkonsole und Arbeitsplatz-PCs. Das lokale LAN verfügt typischerweise auch über einen Internet-Zugang und kann auf externe Medien-Server zugreifen. Auch der Arbeitsplatz-PC kann als Medien-Server und Medien-Player agieren. Hierzu wird eine spezielle Software wie etwa der Windows Media Player benötigt. Viele Anwender telefonieren auch bereits über das Internet (IP-Telefonie).

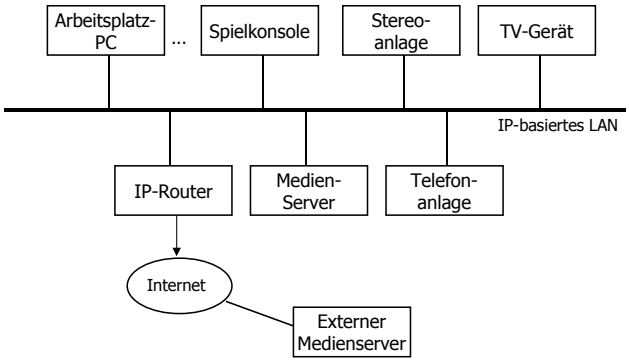


Abbildung 6-25: Typisches multimediales System eines Intranet

Multimedia-Systeme bzw. *Multimedia-Anwendungen* sind oft Anwendungen, die einen kontinuierlichen Datenstrom benötigen. Typischerweise werden kontinuierliche Abfolgen von Daten über einen längeren Zeitraum übertragen. Das bedeutet, dass auch die zeitliche Reihenfolge der übertragenen Daten einen Teil der Information ausmacht. Man spricht auch von zeitabhängigen Medien. Bei vielen Multimedia-Anwendungen besteht sogar eine zeitliche Abhängigkeit zwischen Sender und Empfänger. Eine maximale Verweilzeit der Nachrichten im Kommunikationssystem ist ebenfalls wichtig. Wenn neben einer maximalen Verweilzeit auch eine minimale Verweilzeit wichtig ist, spricht man auch von einer *isochronen* Übertragung. Damit werden Verzögerungsschwankungen (Jitter) eingegrenzt. Bei Audio-Übertragungen sollte der Jitter-Wert beispielsweise unter 20 ms liegen (Schill 2007).

Typische Multimedia-Anwendungen sind Video-Übertragungen, Video-on-Demand, Web-TV, IP-Telefonie, Internet-Radio, Telefonkonferenzen, Videokonferenzen und interaktive Internet-Spiele. In der Regel werden für diese Anwendungen Medienströme verwendet. Man spricht auch von Audio- und Video-Streaming (A/V-Streaming). Medienströme dienen dazu, die aus einem Rechnernetz wie dem Internet empfangenen A/V-Daten gleichzeitig auf einem Ausgabesystem (Medien-Player) wiederzugeben. Medien-Player verfügen über eine spezielle Software, die z.B. als Plugin in einem Web-Browser enthalten sein kann.

Nach (Schill 2007) kann man stromorientierte Anwendungen auch in unidirektionale und bidirektionale Anwendungen unterscheiden. Erstere sind typische Multimedia-Streaming-Anwendungen, die entweder einen vorher gespeicherten Strom oder eine Live-Quelle senden. Eine bidirektionale, strombasierte Kommunikation kann über eine Punkt-zu-Punkt- oder eine Mehrpunkt-Kommunikation gesendet werden. Punkt-zu-Punkt-Verbindungen werden z.B. bei der IP-Telefonie benutzt, Mehrpunktverbindungen bei Video-Konferenzen. Bidirektionale Kommunikation hat noch weit höhere Anforderungen an eine geringe Verzögerungsschwankung als unidirektionale Kommunikation, da durch die Interaktion der beteiligten An-

wender eine Zwischenspeicherung von Strömen nur sehr begrenzt ohne Beeinträchtigung der Nutzer möglich ist. Zudem ist eine genaue Synchronisation paralleler Ströme notwendig (z.B. Audio und Video gemeinsam).

Die von einem Sender verschickten Medienströme müssen möglichst in Echtzeit beim Empfänger ankommen. Man spricht daher auch von *Multimedia*-Anwendungen, die Realzeitanforderungen erfüllen müssen, oder *Realtime-Multimedia-Anwendungen*. Im Gegensatz zu den klassischen Datendiensten sind für diese Art von Anwendung spezielle Kommunikationsdienste erforderlich. Die heute in IP-Netzwerken verfügbaren Mechanismen passen nicht ganz auf die Anforderungen. Zusammengefasst sind folgende Besonderheiten bzw. spezielle Anforderungen zu erwähnen:

- Multimedia-Anwendungen benötigen aufgrund der großen Datenmengen meist auch hohe Übertragungsraten. Die Daten müssen sogar meist komprimiert werden und sind dann trotzdem noch groß. Beispiele hierfür sind Audio- und Videodaten.
- Multimedia-Anwendungen reagieren ungünstig auf Verzögerungen bei der Datenübertragung und weisen meist eine hohe Verzögerungssensibilität auf. Verzögerungsschwankungen (Jitter) sollten daher gering gehalten werden.
- Die Daten müssen möglichst in „Echtzeit“ beim Empfänger ankommen. Datenverluste sind aber weniger kritisch und können bis zu einem bestimmten Grad toleriert werden. Hier unterscheiden sich die Anforderungen an multimediale Anwendungen stark von den Anforderungen an klassische Kommunikationsanwendungen wie Filetransfer, E-Mail, DNS und WWW. Wichtiger ist, dass der Übertragungsstrom nicht abreißt. Eine gewisse Dienstequalität wird vom Kommunikationsdienst erwartet, aber eine vollkommen fehlerfreie Übertragung steht nicht im Vordergrund.

Das Internetprotokoll IPv4 kann hier mit seinem Best-Effort-Ansatz nur wenig Unterstützung leisten. Es wird zwar angestrebt, jedes Paket so schnell wie möglich vom Sender zum Empfänger zu transportieren, aber eine Zusicherung über die benötigte Zeit kann nicht gegeben werden. Auch TCP verhält sich für die multimediale Kommunikation nicht optimal, da es primär dafür sorgt, dass keine Daten verloren gehen. Bei Verlust von Nachrichten sichern Nachrichtenwiederholungen eine zuverlässige Übertragung in der richtigen Reihenfolge ab. Ein erneutes Senden von Daten eines Medienstroms ergibt aber in der Regel keinen Sinn, da die verlorengegangene Information zu einem späteren Zeitpunkt beim Empfänger nicht mehr relevant ist. In einem Film oder einem Musikstück kann eine verspätet ankommende Sequenz nicht mehr sinnvoll abgespielt werden. Daher wird für die Übertragung von multimedialen Daten meist auf UDP als Transportprotokoll zurückgegriffen.

6.5.2 Audio- und Video-Kompression

Audio- und Videodaten, die über ein Netzwerk übertragen werden sollen, müssen zunächst digitalisiert und anschließend komprimiert werden. Die Komprimierung ist wegen der riesigen Datenmengen erforderlich, um die Bandbreitenanforderungen zu reduzieren. Ein einzelnes Bild einer Video-Sequenz benötigt je nach Auflösung schon mehrere Mbyte. Die Notwendigkeit von Kompressionsverfahren kann man am Datenaufkommen eines unkomprimierten Videos erkennen. Ein Fernsehbild besteht beispielsweise in der PAL-Norm[29] aus 768 x 576 Bildpunkten, 25 Bildern pro Sekunde, und pro Bildpunkt werden 3 Byte Farbinformationen (RGB) übermittelt. Daraus ergibt sich ein Datenstrom von

*768 * 576 * 25 * 3 = 31,6 MByte/s bzw. 253 MBit/s.*

Für die zugehörigen Audiodaten in Stereo werden zusätzlich noch 1,4 *MBit/s* benötigt. Ein Film in der Länge von 90 Minuten würde zur Speicherung also 170 GByte Platz benötigen. Analoge Signale werden über das bereits in Kapitel 2 beschriebene PCM-Verfahren digitalisiert. PCM wird auch in der CD-Technik verwendet und benötigt bei einer Abtastrate von 44100 Samples/s bei 16 Bit/Sample eine Übertragungsrate von knapp über 700 kbit/s in Mono und das Doppelte in Stereo. Da dies für eine Übertragung zu viel ist, wird eine Komprimierung vorgenommen.

Man unterscheidet verlustfreie und verlustbehaftete Komprimierungstechniken. Die sog. *Entropiekodierungstechniken* manipulieren nur die Bits des Stroms, betrachten aber nicht deren Bedeutung und sind verlustfrei. *Quellenkodierungstechniken* nutzen die Semantik des zu komprimierenden Bitstroms und sind dagegen üblicherweise verlustbehaftet.

Die bekannteste Audio-Kompressionstechnik ist unter dem Namen *MP3* bekannt. Diese Technik ermöglicht eine Kompression um den Faktor 12, was eine Übertragung mit 128 kbit/s ermöglicht. MP3 wurde im Rahmen der *MPEG*-Standardisierung entwickelt und kann u.a. in (Peterson 2007) nachgelesen werden.

Video-Daten werden heute oft in das *MPEG*-Format[30] komprimiert. Ein Video wird hier als Folge von Standbildern, sog. Frames, dargestellt. Ein Frame wird wiederum in *JPEG* komprimiert. JPEG[31] ist ein Standard für die Komprimierung von Standbildern. MPEG wandelt eine Sequenz von Video-Frames in verschiedene Frame-Typen um und erzeugt einen komprimierten Video-Strom. Dabei wird die Tatsache ausgenutzt, dass sich aufeinanderfolgende Frames einer Sequenz teilweise nur sehr wenig unterscheiden und man nur die Differenzen von aufeinanderfolgenden Frames übertragen muss. Heute gibt es mehrere Varianten bzw. Weiterentwicklungen von MPEG, die als MPEG-1, MPEG-2, MPEG-4 usw. bezeichnet

---

[29] PAL = Phase Alternation Line, ein Verfahren zur analogen Farbfernsehübertragung.

[30] MPEG steht für *Moving Picture Expert Group*.

[31] JPEG wird von der *Joint Photographic Experts Group* definiert.

werden. MPEG-1 wird auch zum Speichern von Filmen auf CD-ROM und MPEG-2 für Filme auf DVDs genutzt.

Kompressionsverfahren für Audio- und Videodaten werden oft auch gemeinsam genutzt, um z.B. A/V-Medienströme für die Übertragung von TV-Daten zu unterstützen. Der MPEG-Standard synchronisiert Audio- und Videosignale über eine gemeinsame Taktung durch eine 90-kHz-Systemuhr. Weitere Informationen zu Kompressionsverfahren wie JPEG und MPEG finden sind in (Tanenbaum 2003a) und (Peterson 2007).

Ein Kompressions-/Dekompressions-Paar wird Codec genannt. Der Begriff setzt sich aus den englischen Begriffen *coder* und *decoder* zusammen. Man kann zwischen Hardware-Codecs und Software-Codecs unterscheiden. Nahezu alle Codecs sind stark asymmetrisch, d.h. der Kompressionsvorgang dauert sehr lange, die Dekompression erfolgt aber dagegen sehr schnell. Dies ist für Video-Streaming in Ordnung, jedoch nicht bei Video-Konferenzen, bei denen nicht viel Zeit für die Kompression verfügbar ist.

Möchte man, dass Kommunikationspartner, die mit unterschiedlichen Codecs arbeiten, miteinander Daten austauschen, ist eine Formatumwandlung erforderlich. Manche Multimedia-Produkte erledigen dies zur Laufzeit. Beim direkten Umwandeln von einem Format in ein anderes spricht man auch von *Transkodierung*. Beispiele für Video-Codecs sind *DivX*[32] und *Xvid* . Beides sind *MPEG-Codecs*, die ein MPEG-4-Videoformat erzeugen. DivX ist eine Weiterentwicklung des Microsoft-MPEG-4-Codec und ermöglicht die Kompression eines DVD-Films von 6 bis 8 Gbyte auf ca. 700 Mbyte bei guter Qualität. Xvid ist wiederum eine Weiterentwicklung von DivX.[33]

In Telefonnetzen verwendet man z.B. den seit 1992 verfügbaren Audio-Codec G.711, der auch im H.323-Standard verwendet wird. G.711 basiert auf dem PCM-Verfahren (8000 Samples) und ist von der ITU-T standardisiert. Die Daten werden bei G.711 nicht komprimiert, und man benötigt eine Datenrate von 64 kBit/s. Hochkomprimierte Audio-Daten, die auch mit 8 Kbit/s übertragbar sind, können dagegen mit dem G.729-Codec erreicht werden (Henning 2007).

Mehrere Codec-Formate werden typischerweise in einem Containerformat zur Speicherung von Multimedia-Daten organisiert und übertragen. In einem Container können verschiedene Codecs zusammenführt werden. Mit AVI (Audio Video Interleave) von Microsoft werden beispielsweise eine MPEG4-Video-Kompression, die mit Hilfe eines Xvid-Codecs erzeugt wurde, ein beliebiges MP3-Format für die Audio-Spur und zusätzliche Text-Untertiteldatenströme miteinander kombiniert.

---

[32] Genauer gesagt wurde DivX durch einen Hacker aus einer Betaversion des Windows Media Players extrahiert.

[33] Siehe www.xvid.org.

AVI basiert wiederum auf dem RIFF (*Resource Interchange Format*), das bereits 1991 von IBM und Microsoft entwickelt wurde. *Interleaved* bedeutet, dass Video und Audio miteinander verschachtelt sind, d.h. Audio- und Videoinformationen werden gemeinsam auf rekonstruierbare Art gespeichert.

6.5.3 Multimedia-Protokolle im Internet

Überblick

Wie die Abbildung 2-26 zeigt, sind für multimediale Anwendungen mehrere Typen von Protokollen wichtig: Protokolle zur Signalisierung, für die Unterstützung von QoS, für das Auffinden von Geräten im Netz und für die eigentliche Übertragung der Medienströme. Die wichtigsten Protokolle sollen im Weiteren vorgestellt werden:

- *RTP* (Real-Time Protocol): Dieses Protokoll wird für den Transport von Medienströmen auf Basis von UDP verwendet.
- *RTCP* (Real-Time Control Protocol): Dieses Protokoll dient dem Austausch von Kontrollinformation zwischen Sender und Empfänger zur Steuerung einer Realtime-Kommunikation. RTCP wird in Verbindung mit RTP eingesetzt.
- *SIP* (Session Initiation Protocol): Dieses Protokoll dient dazu, eine Verbindung bzw. Sitzung für eine Multimedia-Kommunikation über TCP/IP aufzubauen. SIP bezeichnet man auch als Signalisierungsprotokoll.
- *RTSP* (Real-Time Streaming Protocol): Dieses Protokoll ermöglicht die Kontrolle eines Medienstroms über typische Operationen wie *rewind*, *resume*, *pause* usw. RTSP ist ebenfalls ein Signalisierungsprotokoll.
- *RSVP* (Resource Reservation Protocol): Dieses Protokoll dient der Reservierung von Ressourcen für die Kommunikation in IP-Netzen.
- *UPnP* (Universal Plug and Play): ist eine Protokollarchitektur, die der Konnektivität und der automatischen Konfiguration von Geräten in Ad-hoc-Netzen auf TCP/IP-Basis dient.

Die multimedialen Datenströme werden in der Regel über mehr oder weniger standardisierte Verfahren wie MPEG, H.261 und MP3 digitalisiert und komprimiert, dann erst über RTP übertragen und anschließend beim Empfänger wieder dekodiert.

Neben den genannten Protokollen sei auch noch der H.323-Standard mit seiner komplexen Protokollarchitektur für Echtzeitaudio- und Videokonferenzen im Internet genannt. Dieser Standard vereint einige Protokolle und Codecs zu einer Gesamtarchitektur für diese Anwendungstypen. Beispielsweise werden RTP für die Medienströme und RTCP für die Steuerung der Audio-Verbindungen verwendet. Basis dieser Protokollarchitektur bilden die Transportprotokolle TCP und UDP.

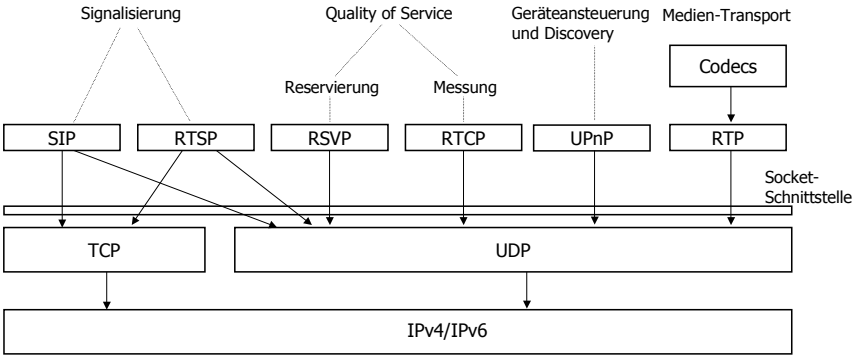


Abbildung 6-26: Überblick über Realtime-Multimedia-Protokolle

Wir werden im Folgenden einen kurzen Überblick über die genannten Protokolle und Protokollarchitekturen geben. Weiterführende Informationen zu den Protokollen (auch zu H.323) findet sich u.a. in (Kurose 2002).

RTP-Protokoll

RTP ist ein „Transportprotokoll" zur Beförderung von Medienströmen und ist im RFC 3550 standardisiert. Es setzt auf UDP auf und verfügt über einige zusätzliche Mechanismen, die den Transport eines Medienstroms erleichtern. RTP nutzt keinen festen UDP-Port. Als Ergänzung zu UDP sei vor allem das Sequenznummernverfahren, ein Zeitstempelverfahren zur Synchronisation zwischen Sender und Empfänger und zur Jitter-Berechnung sowie die Angabe der Nutzlast-Typen genannt. Der RTP-Header sieht wie folgt aus (Abbildung 6-27):

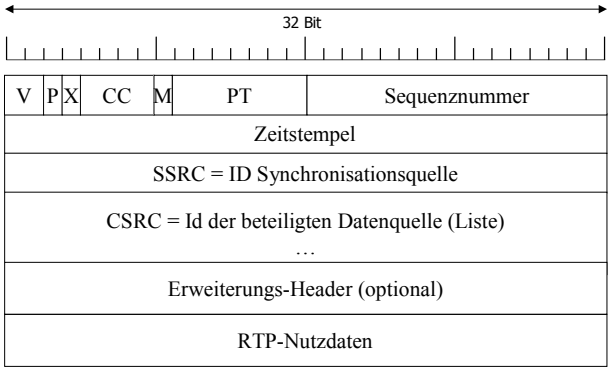


Abbildung 6-27: RTP-Header

Die einzelnen Felder des RTP-Headers sollen nun kurz beschrieben werden:

- *Version (V):* Versionsstand des RTP-Protokolls (Standardwert = 2).
- *Padding (P*): Das Füll-Bit ist gesetzt, wenn ein oder mehrere Füll-Oktets am Ende des Pakets angehängt sind, die nicht zur Payload gehören.
- *Extension (X):* Das Erweiterungs-Bit ist gesetzt, wenn der RTP-Header um einen Erweiterungs-Header ergänzt wird. Erweiterungen sind optional und werden kaum genutzt. Der Aufbau ist dem RFC 3550 zu entnehmen.
- *Marker (M):* Das Marker-Bit ist für Anwendungen reserviert.
- *Sequenznummer*: Die Sequenznummer wird vom Sender für jedes RTP-Datenpaket um eins erhöht, wobei die Startnummer zufällig ausgewählt wird. Ein Empfänger kann mit Hilfe der Sequenznummer die Paketreihenfolge wiederherstellen und ggf. den Verlust von Paketen erkennen.
- *Payload-Typ (PT):* Dieses Feld beschreibt das Format der RTP-Nutzlast. Nutzlast-Typen sind JPEG-, MPEG1/2-Daten und viele mehr.
- *Zeitstempel*: Der Zeitstempel gibt den Zeitpunkt des ersten Oktets innerhalb der RTP-Nutzdaten an. Der Zeitstempel ist taktorientiert, kontinuierlich und linear zunehmend. Damit kann beim Empfänger die Synchronität des Stroms überprüft werden und man kann Laufzeitunterschiede, also Jitter, ermitteln.
- *SSRC*: Dieses Feld identifiziert eine Quelle des RTP-Stroms eindeutig.
- *CSRC Count (CC):* CSRC-Zähler gibt die Anzahl der CSRC-Identifier (Contributing Source) an. Hierunter versteht man Quellen, die an dem RTP-Strom beteiligt sind. Dieses Feld wird benutzt, wenn mehrere RTP-Ströme eine Vermittlungsstelle, die als Mixer bezeichnet wird, durchlaufen. Im Mixer werden die beteiligten Quellen gemischt, also zeitlich synchronisiert. Es sind 0 bis maximal 15 CSRCs in einer Liste möglich.
- *CSRC:* Liste von Ids der Beteiligten Datenquellen.

RTP wird heute in einigen Multimedia-Anwendungen zum Transport von Video und/oder Audio-Streams verwendet. Beispielsweise wird RTP für IP-Telefonie-Anwendungen und auch für Audio- und Videokonferenz-Anwendungen eingesetzt.

RTCP-Protokoll

RTCP (Real Time Transport Control Protocol) ist ein Kontrollprotokoll für Medienströme und gemeinsam mit RTP im RFC 3550 spezifiziert. Es unterstützt die Sicherung der Übertragungsqualität. Der Sender eines Datenstroms weiß in der Regel nicht, mit welcher Bandbreite und Durchsatzrate ein Empfänger auf das Internet zugreifen kann. Zudem können Engpässe in einem Netzwerk nicht vorhergesagt werden. Mit Hilfe von RTCP sind Rückmeldungen über die Leistung von Anwendung und Netzwerk (Bandbreite) sowie die Unterstützung der zeitlichen Synchronisation verschiedener Medienströme möglich.

Die Empfängerseite kann unter Nutzung von RTCP ein Feedback zur Kommunikation an den Sender übermitteln. Der Sender (bzw. die Anwendung auf Senderseite) kann mit Hilfe dieser Daten ihre Verbindungseigenschaften anpassen, um Informationsverluste zu vermeiden und um die Leistung zu steigern. RTCP hat zwar keinen direkten Einfluss auf die Qualität der Verbindung, doch können höhere Protokollschichten mit den Informationen der RTCP-PDUs Rückschlüsse auf die Qualität der Verbindung ziehen.

Die RTCP-PDUs bestehen aus einem RTCP-Header und den eigentlich zu übertragenden Informationen (Nutzlast). Bei RTCP werden fünf PDU-Typen etwa für die Übertragung eines Sendeberichts oder eines Empfängerberichts unterschieden. Der PDU-Aufbau ist je nach PDU-Typ etwas anders gestaltet.

RTCP nutzt keinen festen UDP-Port. Im Zusammenspiel mit RTP wird in der Regel der Folgeport des RTP-Ports verwendet. Der Aufbau eines RTCP-Headers wird grob in der Abbildung 6-28 dargestellt:

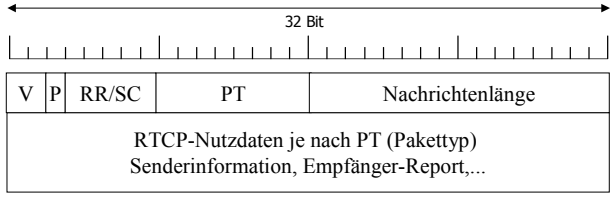


Abbildung 6-28: RTCP-Header

Die Felder enthalten im Einzelnen folgende Informationen:

- *Protokollversion (V):* Gibt die Protokollversion an (Standardwert = 2).
- *Padding (P):* Das Füll-Bit ist gesetzt, wenn ein oder mehrere Füll-Oktets am Ende des Pakets angehängt sind, die nicht zur Payload gehören.
- *Reception Report Count* oder *Sender Report Count (RR/SC):* Gibt an, wie viele Berichtsblöcke (Sender-/Empfängerreport) in der PDU enthalten sind. Von 0 bis 31 Berichtsblöcke sind möglich.
- *Paket Type (PT):* Gibt den PDU-Typen an.

Die Nutzdaten enthalten unterschiedliche Informationen. Bei einem Senderbericht sind dies u.a. die Identifikation der Quelle, der Paket- und der Oktet-Zähler für die empfangenen Daten, Verzögerungsinformationen usw.

RSVP-Protokoll

RSVP ist ein Signalisierungsprotokoll, das dazu dient, Resourcen für Medienströme oder auch andere Ressourcen im Internet zu reservieren. Über RSVP können z.B. für die Übertragung von Medienströmen Bandbreiten reserviert oder Einstellungen für eine maximale Verzögerung vorgenommen werden. Man spricht hier auch von QoS-Parametern und Dienstklassen im Rahmen von sog. integrierten Services oder differenzierten Services. IETF-Arbeitsgruppen wie *IntServ* und *Diff-*

Serv befassen sich mit dieser Thematik. RSVP ist dabei ein Protokoll, das in diesem Zusammenhang in letzter Zeit an Bedeutung gewonnen hat. Allerdings ist die QoS-Problematik im „Best-Effort-Internet" recht kompliziert und sprengt den Rahmen dieses einführenden Buches. Weitere Informationen können dem entsprechenden RFC 2205 oder (Peterson 2007) entnommen werden.

RTSP-Protokoll

RTSP (Real-Time Streaming Protocol) ist im RFC 2326 definiert und ist ein Protokoll zur Steuerung von Datenströmen (z.B. Medienströmen). Es arbeitet im Prinzip wie eine Fernbedienung für einen Videorecorder. Über RTSP können auch mehrere Medienströme gleichzeitig gesteuert werden. Die Übertragung kann über TCP, UDP und auch über Multicast (UDP) erfolgen.

RTSP dient der Steuerung der Übertragung, die eigentliche Übertragung erfolgt über ein Protokoll wie RTP. Heute wird RTSP u.a. für das Abspielen von Audio- und Videostreams aus dem Internet über den Browser benutzt. Dazu wird zunächst eine Verbindung bzw. im Jargon von RTSP eine Session über das Kommando SETUP mit einem Medienserver im Internet aufgebaut und anschließend die Datenübertragung über RTP initiiert. Kommandos (PAUSE, RECORD, TEARDOWN) zur Steuerung des Stroms werden über RTSP übertragen. Das Protokoll ist textbasiert und ähnelt dem HTTP-Protokoll stark. Auch ein Zusammenspiel mit HTTP ist üblich. So kann z.B. eine Beschreibung der Multimedia-Session (Session Description) über HTTP von einem Webserver zum Web-Client transportiert und anschließend ein SETUP-Kommando an den Medienserver, der in der Session-Beschreibung übertragen wurde, gesendet werden. Die Adressierung eines Medienservers erfolgt über eine URL.

In Abbildung 6-29 ist eine typische und sehr vereinfachte Ablaufsequenz für den Einsatz von RTSP im Web skizziert. Die Session-Beschreibung wird von einem Web-Browser über einen Webserver besorgt, und anschließend wird eine Session zu einem Medienserver aufgebaut. Während der Session wird RTP zur Übertragung eines Medienstroms und RTCP zur Kontrolle des Stroms genutzt. Die Kommandos zur Steuerung des Streams (PAUSE,...) werden über RTSP gesendet.

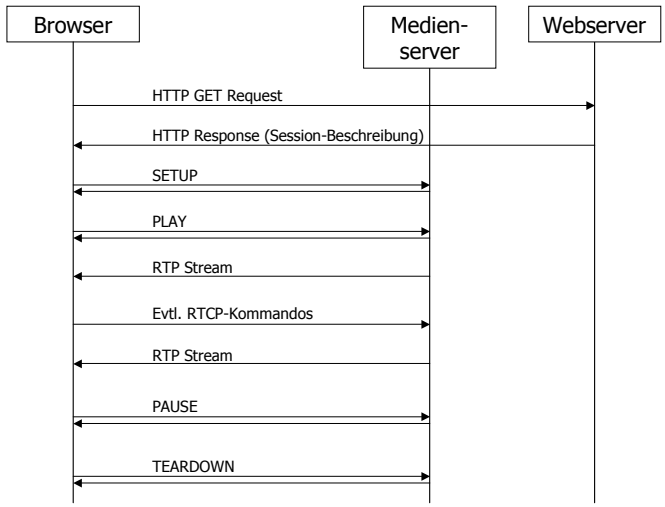


Abbildung 6-29: Beispielhafte Nutzung von RTSP

SIP-Protokoll

Das Session Initiation Protocol (SIP) ist ein Signalisierungsprotokoll zum Aufbau, zur Steuerung und zum Abbau einer Kommunikationssitzung zwischen zwei und mehr Teilnehmern. Das Protokoll wird im RFC 3261 spezifiziert. Vor allem in der IP-Telefonie ist SIP ein häufig angewandtes Protokoll. Es dient nicht zur Übertragung von Medienströmen. Hierfür wird in der Regel RTP verwendet. SIP nutzt sowohl TCP als auch UDP als Transportprotokoll und verwendet standardmäßig die TCP-/UDP-Ports 5060 bzw. bei Einsatz von Verschlüsselungstechniken (TLS) die TCP/UDP-Ports 5061.

SIP unterstützt u.a. Funktionen zum Aufbau einer Kommunikations-Session einschließlich des Einrichtens von Verbindungsparametern, zur Überprüfung der Teilnehmer und zur Auswahl der Medienformate sowie zum Abbau einer Session. Die Adressierung bei SIP ähnelt der von E-Mails. Eine SIP-Adresse sieht beispielsweise wie folgt aus: *sip:username@host*.

Die SIP-Architektur besteht aus den Komponenten *User-Agent-Client (UAC)* und *User-Agent-Server (UAS)*, *SIP-Terminal*, *Proxy-Server*, *Redirect-Server* und *Location-* bzw. *Registrar-Server* (siehe Abbildung 6-30).

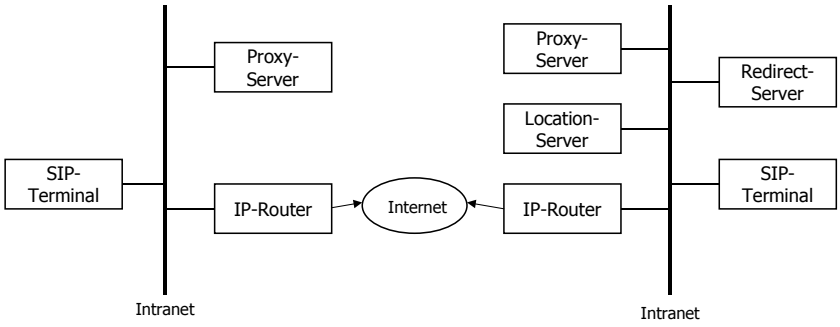


Abbildung 6-30: Einfache SIP-Architektur mit zwei SIP-Terminals in unterschiedlichen Intranets

Die Architekturkomponenten sollen im Folgenden kurz erläutert werden:

SIP-Terminal. Ein SIP-Terminal beinhaltet sowohl einen User-Agent-Client als auch einen User-Agent-Server. Hierbei handelt es sich um Softwarekomponenten. Ein UAC dient zum Initiieren und Senden von SIP-Anfragen. Ein UAS dient dem Empfangen und Beantworten von SIP-Anfragen. Außerdem kann dieser Anrufe akzeptieren, umleiten oder ablehnen. Hardwaretechnisch sind SIP-Terminals entweder IP-Telefone oder PCs.

Proxy-Server. Ein Proxy-Server beinhaltet ebenfalls einen UAC und einen UAS und kann erhaltene Anfragen unverändert oder in veränderter Form an einen anderen Server weiterleiten, oder auch selbstständig auf eine Anfrage antworten. Optional kann ein Proxy-Server Anfragen duplizieren und an mehrere Server weiterleiten. Dieses Vorgehen ist sinnvoll, falls sich eine Person an mehreren Terminals angemeldet hat.

Location- bzw. **Registrar-Server.** Der Location- oder Registrar-Server ist für die Lokalisierung der Anwender zuständig. Seine Aufgabe ist mit der eines DNS-Servers vergleichbar. Er ordnet den abstrakten SIP-Adressen konkrete IP-Adressen zu. Dies ist notwendig, da ein Benutzer auch von verschiedenen Standorten und damit IP-Adressen erreichbar sein kann. Bei Anfragen nach dem Standort einer Person durch den Redirect- oder den Proxy-Server gibt der Location-Server eine Liste der möglichen IP-Adressen des Benutzers zurück.

Redirect-Server. Dieser Server antwortet auf SIP-Anfragen und ist für die Umsetzung einer SIP-Adresse auf IP-Adressen verantwortlich. Eine eingehende SIP-Adresse wird in eine oder mehrere IP-Adressen umgewandelt. Die umgewandelten Adressen erhält der anfragende UAC als Antwort.

Die SIP-Kommandos werden in HTTP-ähnlichen PDUs übertragen. Folgende sechs Kommandos sind für die Kommunikation definiert:

- *INVITE:* Initiieren einer Verbindung
- *BYE:* Beenden einer Verbindung
- *ACK:* Positive Bestätigung
- *OPTIONS:* Erfragen von Medieneigenschaften eines SIP-Teilnehmers
- *CANCEL:* Verbindung vorzeitig abbrechen
- *REGISTER:* SIP-Teilnehmer beim Location-Server mit entsprechenden Parametern (URL, Port, IP-Adresse) registrieren

Für eine Beschreibung der Multimedia-Session wird zudem das *Session Description Protocol* (*SDP*) verwendet, das hier nicht weiter erläutert werden soll.

In Abbildung 6-31 ist eine vereinfachte Kommunikation zwischen zwei SIP-Teilnehmern über Proxy-Server dargestellt. Der SIP-Location-Server und der SIP-Proxy sind im Beispiel zusammengefasst. Nach einer Registrierung der Teilnehmer schickt der initiierende Teilnehmer einen Verbindungsaufbauwunsch an den zweiten SIP-Teilnehmer mittels eines INVITE-Befehls. Die Anfrage wird mit Hilfe eines OK-Befehls bestätigt. Nachdem der zweite Teilnehmer die Verbindung angenommen hat, wird dies nochmals vom Teilnehmer 1 bestätigt. Anschließend werden über RTP Nutzdaten (z.B. Audio- oder Video-Streams) übertragen. Zur Beendigung der Verbindung schickt der Teilnehmer 2 beiden einen BYE-Befehl. Wird dieser dann von seinem Kommunikationspartner mittels einer OK-PDU bestätigt, ist die Verbindung beendet.

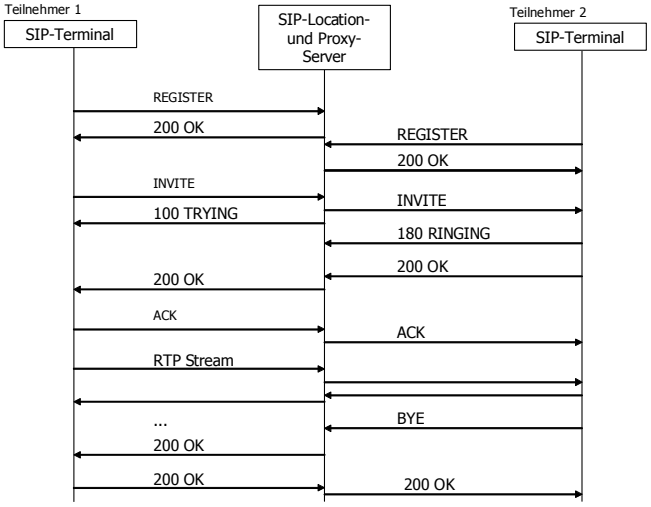


Abbildung 6-31: Typischer Ablauf einer SIP-basierten Kommunikation

UPnP AV

UPnP (Universal Plug and Play) ist eine Technologie und ein Architekturmodell für Netzwerke mit geringem Konfigurationsaufwand (Zero-Configuration). Die Geräte sollen sich dynamisch kennenlernen. Die Architektur wurde von Microsoft entwickelt und setzt sich im Multimedia-Umfeld insbesondere für hausinterne Multimedia-Lösungen (interne Netzwerke) immer mehr durch. UpnP nutzt bestehende Protokolle wie DNS, DHCP, SSDP, HTTP, SOAP (Protokoll für die Übertragung von Webservice-Aufrufen), GENA (Generic Event Notification Architecture) sowie XML zur Nachrichtenbeschreibung. Speziell bei der Suche nach Geräten mit SSDP wird über HTTP eine Punkt-zu-Punkt (HTTPU) oder eine Multicast-Kommunikation (HTTPMU) auf Basis eines UDP-Ports durchgeführt. GENA wird u.a. für die Kommunikation von Zustandsänderungen in den Geräten verwendet. Die Architektur wird auch als *UpnP Device Architecture* bezeichnet. Die Nutzung der einzelnen Protokolle kann (UpnP-Forum 2006) nachgelesen werden.

UPnP AV (Universal Plug and Play Audio Video) ist im Kontext von UpnP der Standard für Audio- und Video-Übertragung. Insbesondere wegen der *Digital Living Network Alliance (DLNA)*, einem Firmenkonsortium, welches Interoperabilitätsrichtlinien für Multimedia-Geräte festlegt und auch Geräte zertifiziert, erfreut sich UpnP AV zunehmender Beliebtheit. DLNA verwendet nämlich UpnP AV als Standard. Auch immer mehr Fernsehgeräte und Medien-Server unterstützen DLNA.[34]

UPnP wurde ursprünglich von der Microsoft eingeführt. Heute spezifiziert das UPnP-Forum[35] den UPnP-Standard und zertifiziert Geräte, die dem Standard entsprechen.

Das UpnP-Modell kennt drei Architekturbausteine. Dies sind das *Media-Server-Gerät*, das *Media-Renderer-Gerät* und den *Control-Point*. UPnP-fähige Hard- oder Software enthält meist eine oder zwei dieser Komponenten. Jede Komponente bietet auch dedizierte Services an, deren Schnittstellen festgelegt sind. Die Komponenten und Services sollen nachfolgend kurz beschrieben werden:

Media-Server. Der Media-Server ist das Gerät, das die Medienobjekte verwaltet und eine Reihe von Diensten anbietet. Ein wichtiger Service des Media-Servers ist der *Content Directory Service* (*CDS*). Der CDS erlaubt es einem Control-Point, die Medien des Servers in Verzeichnis-Form zu durchsuchen. Genau wie in einem Dateisystem gibt es Ordner (Container) und Dateien (Items). Items sind über Metadaten wie Titel, Autor, Genre usw. beschrieben. Control-Points können Items über das CDS suchen. Ein weiterer Service des Media-Servers wird als *Connection-Manager* bezeichnet. Dieser dient dazu festzustellen, welche Geräte mit dem Me-

[34] Siehe www.dlna.org.

[35] http://www.upnp.org.

dia-Server verbunden sind und welche Formate und Protokolle der Media-Renderer unterstützt. Zudem kann ein Media-Server optional den sog. *AVTransport-Service* zur Verfügung stellen. Dieser Service erlaubt typische HiFi-Kommandos wie *Play*, *Pause* oder das Springen innerhalb eines Titels (ähnlich der Funktionalität von RTSP).

Media-Renderer. Media-Renderer sind für die Anzeige oder Wiedergabe der Medienobjekte zuständig. Beispiele für reine Media-Renderer sind Monitore, Fernseher oder Lautsprecher, die anstatt den üblichen AV-Eingängen eine Netzwerkverbindung (Ethernet oder WLAN) benutzen. Auch der Media-Renderer stellt Services zur Verfügung. Wie der Media-Server liefert er einen *Connection-Manager-Service* zur Überwachung von Verbindungen zwischen Geräten. Zudem verfügt er über einen *Rendering-Control-Service*, mit dem Einstellungen wie Lautstärke und Kontrast möglich sind. Auch ein *AVTransport-Service* ist im Media-Renderer integriert.

Control-Point. Der Control-Point kontrolliert und steuert die anderen Komponenten und kommuniziert dazu mit dem Media-Server und dem Media-Renderer. Um ein Medienobjekt abzuspielen bzw. anzuzeigen, sucht man dieses üblicherweise über den CDS. Für das Abspielen kommunizieren Media-Server und Media-Renderer miteinander. Für den Transfer der multimedialen Daten wird ein Transferprotokoll ausgewählt, das nicht in UpnP festgelegt ist.

UpnP-Geräte sind in einem IP-Netzwerk über eine IP-Adresse erreichbar. Diese kann über den üblichen Weg, etwa über DHCP besorgt werden. Sobald ein UPnP-Gerät über eine IP-Adresse verfügt, muss es seine Existenz im Netzwerk an die Control-Points melden. Dies erfolgt über UDP und zwar konkret über die Multicast-Adresse 239.255.255.250 mit dem UDP-Port 1900. Hierzu wird das *Simple Service Discovery Protocol (SSDP)* verwendet. Auch Control-Points können im Netz nach UPnP-Geräten suchen, indem eine Discovery-PDU gesendet wird. SSDP ist ein HTTP-ähnliches Protokoll. Auf die Beschreibung der entsprechenden PDUs soll hier verzichtet werden.

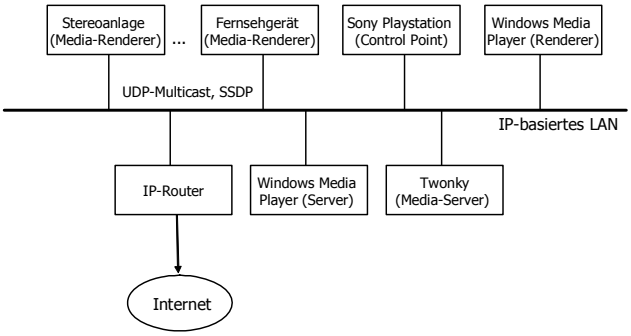


Abbildung 6-32: Typische Konfiguration eines Multimedia-Heim-Netzwerks

Abbildung 6-32 zeigt ein Beispiel-Multimedia-Netzwerk mit zwei Media-Servern, die auch gleichzeitig Control-Points sind. Einer davon ist der bekannte Windows Media-Player, der auch einen Serverdienst bereitstellt. Der zweite ist der Twonky-Media-Server[36], der sich ebenfalls schon einer guten Verbreitung erfreut. Beides sind Softwarekomponenten, die in der Regel auf PCs oder Serversystemen zum Ablauf kommen. Als Media-Renderer sind eine Stereoanlage, ein Fernsehgerät, eine Sony Playstation 3[37] sowie ein PC mit dem Windows Media Player im Netz.

6.6 Übungsaufgaben

1. Wozu dient DNS im globalen Internet?
2. Wie sind die Domains im Internet strukturiert, was sind Zonen im DNS und wer verwaltet diese?
3. Wie findet ein Clientrechner zu einem Rechnernamen eines Servers die zugehörige IP-Adresse?
4. Welche Aufgabe hat der Resolver im DNS und auf welchem Rechner liegt er typischerweise?
5. Wie ist ein DNS-Resolver üblicherweise implementiert?
6. Wozu dient die DNS-Domäne in-addr.arpa?
7. Welche anderen DNS-Name-Server muss ein DNS-Name-Server kennen, der für die Verwaltung einer Zone zuständig ist und drei Subzonen an darunterliegende Name-Server delegiert hat?
8. Was ist der Unterschied zwischen iterativer und rekursiver Behandlung einer DNS-Anfrage?
9. Welche Komponente entscheidet, ob beim Auflösen eines Namens in DNS nach der iterativen oder der rekursiven Abfragemethode vorgegangen werden soll?
10. Wozu werden Resource Records vom Typ „MX“ und wozu solche mit Typ „A“ benötigt?
11. Was versteht man unter einem autoritativen DNS-Server und woran merkt man, dass eine DNS-Response nicht von einem autoritativen DNS-Server kommt?
12. Warum sind die DNS-Root-Name-Server so wichtig und wie findet ein lokaler DNS-Server einen DNS-Root-Name-Server?
13. Beschreiben Sie grob den Aufbau einer HTTP-PDU!

[36] http://www.twonkymedia.com.

[37] http://de.playstation.com.

14. Beschreiben Sie den Ablauf der Kommunikation zwischen einem WWW-Browser und einem WWW-Server zum Lesen eines Dokuments! Erläutern Sie hierbei, was dabei auf der TCP-Ebene passiert, welche und wie viele TCP-Nachrichten gesendet werden. Hinweis: Gehen Sie von einem sehr kleinen Dokument aus (ca. 200 Byte).
15. Was ist der Unterschied zwischen einer GET- und einer POST-Operation im HTTP-Protokoll?
16. HTTP ist ein zustandsloses Protokoll. Was bedeutet dies?
17. Wie wird z.B. in komplexeren WWW-Anwendungen eine Zustandsverwaltung durchgeführt (kurze Erläuterung)?
18. Wie kann ein HTTP-Client angeben, dass er nur die MIME-Typen text/html und text/plain bearbeiten kann?
19. Was bedeutet die Angabe „Connection: Keep alive“ im Header des HTTP-Requests für das Verbindungsmanagement zwischen Web-Client und Web-Server?
20. Was ist POP3 und wozu wird es verwendet?
21. Wozu dient das Protokoll SMTP im Internet?
22. Wie unterscheidet sich eine traditionelle Web-Kommunikation von der Kommunikation mit AJAX?
23. Was versteht man bei AJAX unter konkurrierenden Requests?
24. Welche Anforderungen haben Audio- und Video-Streaming-Anwendungen an die Kommunikation im Netzwerk?
25. Wozu dient das RTP-Protokoll?
26. Was ist ein Codec?
27. Was versteht man unter UPnP?

7 Grundlagen der mobilen Kommunikation

Kommunikationssysteme, die auf der Internet-Technologie basieren, werden heute von vielen Menschen alltäglich benutzt. In den meisten Fällen wird hierfür ein stationäres System wie beispielsweise der PC im Büro oder Zuhause genutzt. Im Bereich der mobilen Kommunikation hat sich die Internet-Technologie noch nicht vergleichbar durchgesetzt. Mobile Endgeräte für den mobilen Zugriff auf das Internet, wie beispielsweise Mobiltelefone oder Notebooks, stehen heute allerdings schon zur Verfügung. Aktuell können moderne Internetdienste noch nicht komfortabel in mobilem Umfeld genutzt werden, da das Kommunikationssystem hierfür noch nicht ausreichend vorbereitet ist.

Die Hinzunahme von mobilen Endsystemen hat Auswirkungen auf das verwendete Kommunikationssystem. Sehr deutlich wird diese Veränderung bei der Betrachtung der Hostanbindung. Eine physikalische Voraussetzung für die mobile Kommunikation ist die drahtlose Anbindung der Endgeräte. Nur wenn die Endgeräte nicht fest über ein Kabel mit dem Netz verbunden sind, ist die gewünschte Mobilität zu erreichen. Um die drahtlose Hostanbindung zu realisieren, wurden spezielle Standards entwickelt, die wir in diesem Kapitel nicht behandeln. Eine kurze Einführung zum weit verbreiteten WLAN-Standard IEEE 802.11 ist im Abschnitt 3.2.3 zu finden. Dieses Kapitel konzentriert sich auf die Anpassungen, die in der Internet- und Transportschicht des TCP/IP-Referenzmodells zur Unterstützung von mobilen Endgeräten vorgenommen werden müssen. Da eine Kommunikation zwischen mobilen und stationären Systemen notwendig ist, ist ein gesonderter Standard für mobile Systeme nicht sinnvoll. In der Internetschicht muss vor allem die Wegewahl an die Situation angepasst werden, sodass mobile Endgeräte während der Kommunikation ihr Zugangsnetz wechseln können. In diesem Kapitel wird gezeigt, wie IPv4 und IPv6 diesem Problem entgegnen. Die angestrebte Mobilität verlangt auch eine Anpassung in der Transportschicht, wenn eine Ende-zu-Ende-Semantik zwischen den Kommunikationspartnern gefordert wird. Am Beispiel von TCP soll gezeigt werden, dass die dort implementierte Fluss- und Staukontrolle bei der Unterstützung von mobiler Kommunikation verändert werden muss.

Zielsetzung des Kapitels

Ziel dieses Kapitels ist es, die besondere Problemstellung der mobilen Kommunikation aufzuzeigen und aktuelle Lösungsstrategien für das Internet vorzustellen. Der Fokus des Kapitels wird auf die nötigen Veränderungen in der Internet- und Transportschicht gelegt. Der Studierende soll die Probleme der Unterstützung mobiler Endgeräte im Internet verstehen.

Wichtige Begriffe

MN, HA, FA, CoA, Care-of-Address, CN, Dreiecks-Routing, Reverse Tunneling, HA-Redirect, Handoff-Roaming, Pre-Registration-Handoff, Post-Registration-Handoff, Performance Enhancing Proxy, PEP, Indirektes TCP, Snooping TCP.

7.1 Mobilitätsunterstützung bei IPv4

In diesem Abschnitt sollen die grundlegenden Probleme der Unterstützung mobiler Endgeräte in einer IPv4-Umgebung erörtert und es sollen Lösungsalternativen diskutiert werden.

7.1.1 Probleme der Mobilität bei IPv4

Zu den wichtigsten Aufgaben von IPv4 zählt die Wegewahl (Routing). Die Daten werden hierzu in Pakete aufgeteilt, die dann einzeln und unabhängig voneinander verschickt werden. Auf dem Weg zum Empfänger wird das Paket von Netzknoten empfangen und entsprechend der Wegewahl an andere Netzknoten weitergeleitet. Die Grundlage für die IP-Wegewahl stellt eine global eindeutige Adressierung dar. Beim Versenden der Pakete wird die eindeutige Empfängeradresse im IP-Protokoll-Header bekannt gegeben. Die Netzknoten, die die Vermittlung übernehmen, können anhand dieser Empfängeradresse eine Routing-Entscheidung vornehmen. Dies erfordert eine Strukturierung der Adressen in eine Netzwerk- und eine Hostnummer.

IP ist primär für eine drahtgebundene und stationäre Kommunikation ausgelegt. Wird ein Knoten aus seinem eigentlichen Teilnetz heraus bewegt und in einem anderen Teilnetz eine Kommunikation auf Basis von IP versucht, scheitert dies zunächst an der Adressierung. IP-Adressen sind nämlich in eine Netzwerk- und Hostnummer aufgeteilt und IP fordert eine topologisch korrekte Adressierung der Knoten. Die Netzwerknummer einer IP-Adresse muss immer mit der Identifikation des aktuellen Zugangsnetzes übereinstimmen. Will man in IP-Netzen mobile Systeme einsetzen, muss die Konfiguration des Systems immer dem aktuellen Teilnetz anpasst werden, um so eine topologisch korrekte Adressierung zu gewährleisten. Eine manuelle Konfiguration kommt nur bei einer sehr eingeschränkten Mobilität in Frage. Werden jedoch die Teilnetze häufig und in kurzen Zeitabständen gewechselt, stellt eine manuelle Konfiguration einen zu hohen Aufwand dar. Abhilfe können hier Mechanismen zur automatischen Konfiguration liefern, die selbstständig aus einem Teilnetz die korrekte IP-Konfiguration beziehen können. Ein beispielhaftes Verfahren zur automatischen Adresskonfiguration ist DHCP, das entwickelt wurde, um die Konfiguration von großen Netzen zu erleichtern. DHCP ermöglicht zwar die automatische Anpassung an ständig wechselnde Teilnetze, jedoch werden dem mobilen System in jedem Teilnetz eine neue Konfiguration und somit auch eine neue IP-Adresse zugewiesen. Dies führt dazu, dass Datagramme, die im IP-Header die Empfängeradresse tragen, nach einem Wechsel

des Teilnetzes nicht mehr zugestellt werden können. Bei einem Wechsel der Teilnetze in kurzen Zeitabständen wird das mobile Endsystem zwar korrekt konfiguriert, ist aber aufgrund der ständig wechselnden IP-Adresse wiederum nicht zu erreichen. Um auch dieses Problem zu lösen, könnte die aktuelle IP-Adresse des mobilen Systems über DNS bekannt gegeben werden. Im Fall eines mobilen Systems müsste dafür gesorgt werden, dass zu dem mobilen System immer die gerade aktuelle IP-Adresse auf den DNS-Servern verteilt wird. Da DNS ein verteiltes System ist, nimmt eine Aktualisierung der Einträge zu einem mobilen System zu viel Zeit in Anspruch und ist somit keine Lösung für die Mobilität. Es verbleibt noch die Möglichkeit, die IP-Adressen von mobilen Systemen in eigenen Einträgen der Routing-Tabellen zu speichern. Auch bei Verwendung dieser Routing-Informationen (es handelt sich hier um eine Host-Route) bleibt jedoch das Problem des enormen Pflegeaufwands. Da im Gegensatz zu den eigentlichen Routing-Tabellen nicht auf Ebene der Subnetze gearbeitet wird, sondern jedes mobile System einzeln berücksichtigt werden muss, stellt allein die Anzahl der zu verwaltenden Knoten ein enormes Problem dar. In der Vermittlungsschicht ist somit nicht die drahtlose Kommunikation an sich problematisch, sondern der durch die Mobilität verursachte Wechsel der Teilnetze. Für das Internet-Protokoll ist somit die Bewegung eines Endsystems solange unerheblich, bis die geografischen Grenzen eines Teilnetzes überschritten werden.

7.1.2 Unterstützung der Mobilität mit IPv4

Um die Mobilität und den damit verbundenen Wechsel von Teilnetzen eines IP-Knotens zu unterstützen, wurde eine Erweiterung ausgearbeitet, die wir als *Mobile IP* bezeichnen. Im RFC 3344 wird diese IP-Anpassung beschrieben. Dieser vorgeschlagene Standard sieht ein Verfahren vor, das den mobilen Knoten automatisch die Adresse an das aktuelle Teilnetz anpassen lässt, aber trotzdem die Erreichbarkeit über eine fixe IP-Adresse sicherstellt.

Zur Beschreibung der Mobilitätsunterstützung wird eine gesonderte Terminologie verwendet, die aus dem RFC 3344 übernommen wurde. Die zu den Begriffen aufgeführten Abkürzungen werden zur Vereinfachung im weiteren Verlauf anstatt der ausgeschriebenen Begriffe verwendet.

- *Mobile Node* (MN): Ein Host (oder Router), der seinen Anschluss an ein Teilnetz wechselt.
- *Home Network*: Ein Teilnetz, welches dem *Mobile Node* als ursprüngliches Netzwerk zugeordnet ist.
- *Home Address*: Eine IP-Adresse, die einem Mobile Node, unabhängig von möglichen Teilnetzwechseln, fest zugeordnet ist. Die Netzwerknummer der *Home Address* muss der Identifikation des Heimnetzes (*Home Network)* entsprechen.
- *Home Agent* (HA): Ein spezieller Router im Heimnetz (*Home Network*), der die Mobilitätsunterstützung ermöglicht.

- *Foreign Network* (FN): Ein Teilnetz, ungleich dem Heimnetz (*Home Network*), in das der *Mobile Node* während der Kommunikation wechselt.
- *Care-of-Address* (**CoA**): Eine topologisch korrekte IP-Adresse, die dem *Mobile Node* im Fremdnetz (*Foreign Network*) zugewiesen wird.
- *Foreign Agent* (**FA**): Ein spezieller Router in einem Fremdnetz (*Foreign Network*), der die Mobilitätsunterstützung ermöglicht.
- *Correspondent Node* (**CN**): Ein beliebiger Host, mit dem die *Mobile Node* über IP kommuniziert.

Automatische Adressierung – Agent-Erkennung

Ein mobiler IP-Knoten muss beim Wechsel des Teilnetzes seine IP-Adresse dem neuen Teilnetz anpassen. Nur mit einer topologisch korrekten IP-Adresse kann ein *Mobile Node* Datenpakete empfangen. Die Mobilitätsunterstützung von IPv4 sieht vor, die Konfiguration der MN automatisch durch den FA vornehmen zu lassen. Voraussetzung ist jedoch, dass der MN erkennt, dass er sich in einem Fremdnetz befindet. Über die Netzadresse des aktuellen Teilnetzes kann eine MN erkennen, ob er sich im Heimnetz befindet oder in einem Fremdnetz. Auch der Wechsel von Teilnetz zu Teilnetz ist über die Veränderung der Netzadresse zu erkennen. Um jedoch zu erfahren, wie die aktuelle Netzadresse lautet, ist ein spezielles Verfahren nötig, welches im Rahmen von Mobile IP *Agent-Erkennung* genannt wird.

Unter Nutzung von ICMP versenden FAs dazu periodisch Meldungen. ICMP-Nachrichten tragen einen normalen IP-Header und ermöglichen es somit dem MN, die aktuelle Netzadresse aus der Quelladresse des IP-Headers auszulesen. Da FAs über diese Meldungen ihre Existenz in ihrem Teilnetz bekannt geben, nennt man diese Meldungen auch *Agent Advertisement*. Für die Mobilitätsunterstützung werden die *Router Advertisement Messages* (siehe RFC 1256) des ICMP um spezielle Felder erweitert. Die Erweiterung wird *Mobility Agent Advertisement Extension* genannt und enthält, wie in Abbildung 7-33 zu sehen, unter anderem für das Teilnetz gültige IP-Adressen. Eine MN erkennt somit den Teilnetzwechsel und erhält auch eine Care-of Adresse. Nach der *Mobility Agent Advertisement Extension* können noch andere Erweiterungen folgen, auf die hier nicht näher eingegangen wird.

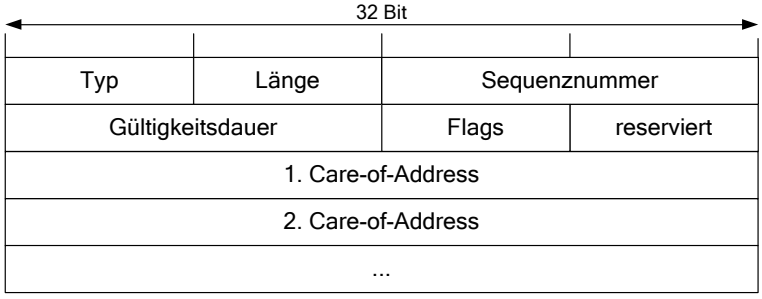


Abbildung 7-33: Mobility Agent Advertisement Extension nach RFC3344

Erhält eine MN nach dem Einschalten oder nach einem Verbindungsabbruch innerhalb kurzer Zeit keine Advertisement-Nachricht, verschickt er eine *Solicitation-Nachricht* (siehe RFC 1256). Diese Solication-Nachrichten sind identisch zu den Solication-Nachrichten von ICMP und veranlassen alle erreichbaren Agenten, eine Advertisment-Nachricht abzuschicken. Meldet sich daraufhin kein Agent, versucht der MN den bekannten Agenten des Heimnetzes zu erreichen.

Befindet sich der MN jedoch in einem Fremdnetz ohne FA, wird versucht über DHCP eine gültige Konfiguration zu erlangen. Eine gültige IP-Adresse, die ohne FA zugewiesen wurde, nennt man *co-located Care-of-Address*. Gelingt es dem MN, nur eine *co-located CoA* zu erlangen, muss der MN auf die Dienste eines FA verzichten. Wie wir sehen werden, muss der MN Aufgaben des FA übernehmen, wenn im aktuellen Teilnetz kein Router mit Mobilitätsunterstützung vorhanden ist.

Registrierung der Care-of-Adresse

Nach der Zuweisung einer Care-of-Adresse (zugewiesen vom FA oder über DHCP) kann der MN Nachrichten über das Internet übertragen und empfangen. Zunächst versucht der MN, die neu zugewiesene IP-Adresse dem Router im Heimnetz mitzuteilen. Um die CoA beim HA zu registrieren, versendet der MN eine spezielle Nachricht, die *Registration Request* genannt wird und in Abbildung 7-34 dargestellt ist.

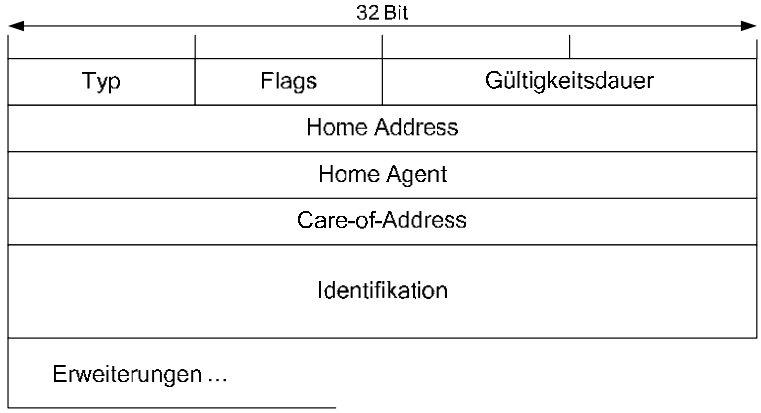


Abbildung 7-34: Registration Request nach RFC 3344

Der HA entnimmt dem Registration Request die aktuelle CoA, unter der der MN zu erreichen ist und speichert diese in einer gesonderten Routing-Tabelle. Die Zuordnung der aktuellen CoA zu einem MN wird *Mobility Binding* genannt und in einem HA für alle ihm zugeordneten MNs gespeichert. Auf Grund der Mobilität sollte ein Mobility Binding nur für einen begrenzten Zeitraum gültig sein. Im Registration Request kann der MN die gewünschte Gültigkeitsdauer mit dem Parameter *Lifetime* angeben. Beendet eine MN die Verbindung zu einem FA oder kehrt

zurück ins Heimnetz, muss die zuvor gültige CoA abgemeldet werden. Ohne eine Abmeldung (Registration Request mit einer Lifetime von 0) ist das aktuelle Mobility Binding bis zum Ablauf der Gültigkeit noch in Verwendung.

Die Registrierung stellt sicher, dass der HA immer über die aktuelle IP-Adresse des MN benachrichtigt wird und somit jederzeit Kontakt zur MN aufnehmen kann. Mobile IP nutzt dies, in dem der HA als erster Ansprechpartner des MN verwendet wird.

Dreiecks-Routing

In der Abbildung 7-35 wird das Standardverfahren zum Routing bei Mobile IP erläutert, wenn ein Router mit Mobilitätsunterstützung (FA) vorhanden ist. Wir gehen von einem MN aus, die sich in einem Fremdnetz befindet und die automatische Adressierung sowie Registrierung der CoA erfolgreich durchgeführt hat. Der MN sendet an eine CN, wobei die Datagramme normal über den FA verschickt werden. Als Quelladresse im IP-Header gibt der MN jedoch nicht die aktuelle CoA, sondern die ihm zugeordnete Adresse aus dem Heimnetz an. Ein CN erwartet, den MN unter der angegebenen Quelladresse erreichen zu können. Da die CoA nur für die Zeit des Aufenthaltes im Teilnetz eines FA gültig ist, wird die feste Adresse aus dem Heimnetz angegeben. Will der CN der MN antworten, schickt er die Datagramme an die ihm bekannte Adresse und somit in das Heimnetz. Da die CoA dem CN verborgen gehalten wird, ist der MN aus Sicht des CN im Heimnetz. Der HA nimmt die eintreffenden Datagramme entgegen und erkennt mit Hilfe der *Mobility Bindings*, dass sich der MN derzeit nicht im Heimnetz befindet. Um die eingehenden Datagramme an das eigentliche Ziel weiterzuleiten, sendet der HA die Daten über einen IP-Tunnel (siehe RFC 2003) an den FA. Der FA entpackt als Endpunkt des IP-Tunnels das ursprüngliche Datagramm und liest die Zieladresse des IP-Headers, die der MN-Adresse aus dem Heimnetz entspricht. Diese Zieladresse vergleicht der FA mit einer Liste aller anwesenden MN und leitet das Datagramm bei einer Übereinstimmung weiter. Über den Umweg des HA kann ein MN immer unter seiner festen Adresse aus dem Heimnetz erreicht werden. Die geforderte Mobilität unter Beibehaltung einer festen IP-Adresse ist somit erreicht, jedoch entstehen durch das *Dreiecks-Routing* neue Probleme.

Im *Dreiecks-Routing* werden Datagramme vom CN zum MN über den Umweg des HA gesendet. Dieser Umweg stellt eine ineffiziente Route dar, die je nach Situation wesentlich länger und damit langsamer als ein direktes Routing zwischen CN und MN sein kann. Im schlechtesten Fall befinden sich MN und CN im gleichen Teilnetz, der HA aber weit entfernt. Ein Datagramm vom CN zum MN wird aus dem gemeinsamen Teilnetz zum HA und über einen IP-Tunnel wieder zurück zum MN gesendet. Eine direkte Route wäre hier deutlich effektiver. Neben der Effizienz geht auf Grund des Dreiecks-Routings noch ein weiterer wichtiger Vorteil des IP-Routings verloren. Da eine Kommunikation vom CN zum MN nur über den HA

möglich ist, wird der HA zum kritischen Punkt. Bei Überlastung stellt der HA einen Engpass und sogar einen *Single-Point-of-Failure* dar.

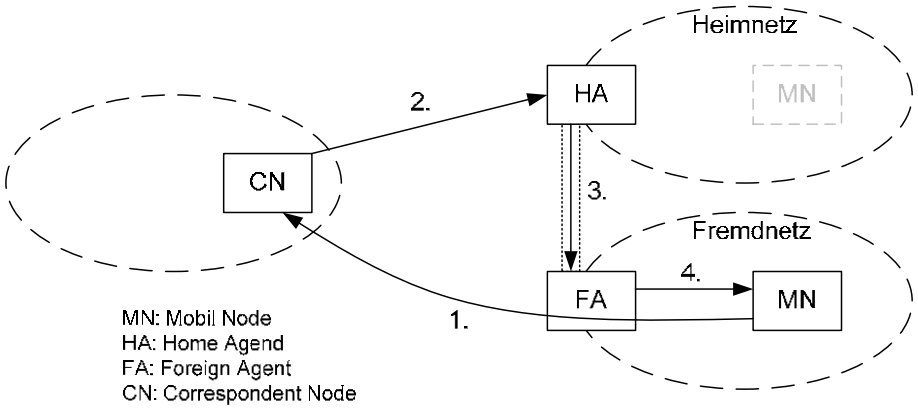


Abbildung 7-35: Architektur für das Dreiecks-Routing

Ein weiteres Problem entsteht durch die Verwendung der MN-Adresse aus dem Heimnetz als Quelladresse für Datagramme an den CN. Da die im IP-Header angegebene Quelladresse topologisch nicht mit dem Teilnetz des FA übereinstimmt, kann es zu Ablehnungen durch Firewalls kommen. Firewalls erkennen topologisch nicht korrekte Quelladressen als *IP-Spoofing* (siehe dazu RFC 2827) und gehen von einer Fälschung der Quelladresse aus. Als Angreifer nutzt man gefälschte IP-Adresse entweder zur Umgehung von IP-basierten Authentifizierungen oder zur Verschleierung der eigenen IP-Adresse. Da IP-Spoofing häufig in Verbindung mit *Denial-of- Service-Attacken* (DoS) auftritt (siehe RFC 2827), fordern Sicherheitsexperten von den Internet Service Providern (ISP), ausgehende Pakete zu verwerfen, die aufgrund der Quelladresse nicht aus dem eigenen Netz stammen können. Das Dreiecks-Routing ermöglicht zwar die Mobilität unter Beibehaltung einer festen IP-Adresse, verursacht aber weitere Probleme, die zu lösen sind.

Reverse Tunneling

Um die beim *Dreiecks-Routing* erkannten Probleme des *IP-Spoofing* bei der Kommunikation vom MN zum CN zu vermeiden, wurde das *Reverse Tunneling* (siehe RFC 3024) entwickelt. Anstatt mit einer nicht korrekten Quelladresse direkt vom MN zum CN zu kommunizieren, werden beim Reverse Tunneling auch diese Datagramme über den HA geroutet. Ein Datagramm vom MN wird vom FA über einen IP-Tunnel zum HA weitergeleitet. Der HA packt das ursprüngliche Datagramm aus und leitet es zum CN weiter. Da die Quelladresse des ursprünglichen Datagramms aus dem Teilnetz des HA stammt, liegt kein *IP-Spoofing* vor. Wie in Abbildung 7-36 zu sehen ist, besteht beim Reverse Tunneling ein bidirektionaler IP-Tunnel zwischen dem HA und dem FA.

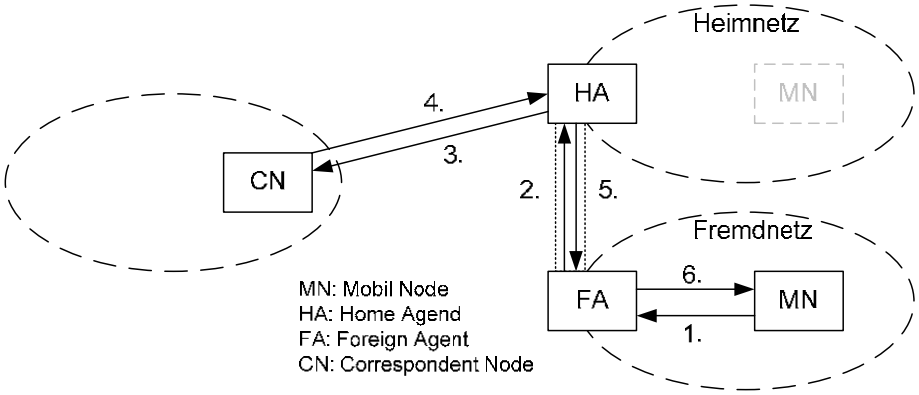


Abbildung 7-36: Reverse Tunneling

Reverse Tunneling löst das Problem der IP-Spoofing-Erkennung beim Dreiecks-Routing, verschärft jedoch die schon erkannten Probleme beim Routing über den HA. Beim Reverse Tunneling wird die Kommunikation in beide Richtungen über den HA abgewickelt, was wie schon gezeigt zu einem ineffizienten Routing führen kann. Auch die Gefahr einer Überlastung des HA steigt, da bei diesem Verfahren die gesamte Kommunikation über den HA geleitet wird. Auch Sicherheitsrisiken werden durch das Reverse-Tunneling verschärft. Wird keine ausreichende Authentifizierung durchgeführt, besteht die Gefahr eines *Reverse-Tunnel Hijackings* (siehe dazu RFC 3024), die hier nicht näher erläutert wird. Grundsätzlich besteht die Gefahr, dass ein Angreifer den Tunnel benutzt, um über ihn in gesicherte Netzwerke einzudringen.

Handoff-Roaming

Neben der Mobilität unter Beibehaltung einer festen IP-Adresse ist die unterbrechungsfreie Kommunikation beim Wechsel von Teilnetzen eine weitere Anforderung an die Mobilitätsunterstützung von IP. Der Übergang von Teilnetz zu Teilnetz wird als *Handoff* oder *Roaming* bezeichnet. Das Problem des Handoffs ist der drohende Datenverlust. Es besteht eine kritische Zeitspanne, beginnend mit dem Abbruch der Verbindung zum alten Teilnetz und endend mit der erfolgreichen Registrierung der CoA des neuen Teilnetzes beim HA. Innerhalb dieser Latenzzeit besteht im HA ein *Mobility-Binding* auf die alte CoA, obwohl der MN unter der alten CoA nicht mehr zu erreichen ist. Treffen innerhalb der Latenzzeit Datagramme für den MN beim FA ein, werden sie an den nicht mehr aktuellen FA weitergeleitet und können den MN nicht erreichen. Die Datagramme gehen somit während eines Handoffs verloren und müssen evtl. von dem Protokoll der Transportschicht neu übertragen werden. In der Anwendungsschicht kann das wiederholte Senden als Verzögerung wahrgenommen werden und muss daher möglichst vermieden werden.

Eine Möglichkeit, den drohenden Verlust der Datagramme abzuwenden, ist das Nachsenden der Daten vom alten FA zum neuen FA. Dieses grundlegende Konzept wird als *Post-Registration-Handoff* (siehe dazu RFC 4881) bezeichnet, da es erst nach der Registrierung beim neuen FA durchgeführt werden kann. Wie in der Abbildung 7-37 zu sehen ist, werden noch eintreffende Datagramme vom alten FA zum neuen FA nachgesendet und erreicht so den MN auch nach dem Teilnetzwechsel. Voraussetzung hierfür ist jedoch die Registrierung der neuen CoA beim alten FA. Nach der Erkennung des Teilnetzwechsels (*Move Detection*) und der erfolgreichen Registrierung beim neuen FA muss die neue CoA wie gewohnt beim HA registriert werden. Zusätzlich muss die neue CoA dem alten FA bekannt gegeben werden.

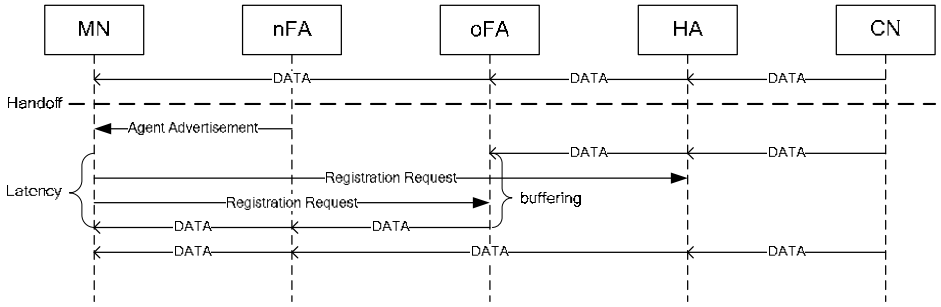


Abbildung 7-37: Konzept des Post-Registration-Handoff[1]

Das Verfahren ist in der Abbildung 7-37 vereinfacht dargestellt, da beispielsweise eine Bestätigungsmeldung nicht abgebildet wurde. Zwischen dem alten und dem neuen FA (nFA, oFA) kann ein IP-Tunnel aufgebaut werden, über den noch eintreffende Pakete nachgesendet werden. Wie in der Abbildung 7-37 zu sehen ist, muss der alte FA bis zur Registrierung der neuen CoA die eintreffenden Datagramme zwischenspeichern (*buffering*), was wiederum zu kurzen Verzögerungen führen kann. Ein guter Vergleich zum eben beschriebenen Verfahren stellt der Nachsendeantrag eines Briefzustelldienstes. Wechselt man seinen Wohnort, muss im Rahmen des Umzugs (Roaming) ein Nachsendeantrag gestellt werden, damit noch eintreffende Briefe zum neuen Wohnort weitergesendet werden. Die Verwaltung eines Nachsendeantrags wird von der lokalen Niederlassung des Dienstleisters übernommen, der mit dem FA verglichen werden kann.

[1] Grundsätzlich kann die Registrierung der neuen CoA auch vom FA übernommen werden.

7.1.3 Optimierung der Mobilität mit IPv4

Die vorgestellte IPv4-Mobilitätsunterstützung ermöglicht es, mit einem mobilen Endsystem eine auf IP basierende Kommunikation aufrecht zu erhalten. Mobile IP ist als Erweiterung zu IPv4 konzipiert und damit kompatibel zu IPv4. Zu den Nachteilen von Mobile IP gehört das schon beschriebene ineffiziente Routing über den HA. Das Routing über den HA macht ihn zu einem möglichen Single-Point-of-Failure und fordert eine ausreichende Leistungsfähigkeit. Ein weiterer Nachteil von Mobile IP ist die Nutzung des Routers im Fremdnetz als FA. Der FA ist beispielsweise der Endpunkt des IP-Tunnels, obwohl er nicht unter der gleichen organisatorischen Kontrolle wie der MN steht. Da auch vom FA Angriffe auf den MN oder auch auf das Netz des HA ausgeübt werden könnten, muss eine ausreichende Vertrauensbasis bestehen. Auch beim Teilnetzwechsel ist, wie bereits erläutert, eine geringe Latenzzeit zu beobachten. Für Anwendungen, wie beispielsweise *Multimedia-Streaming*, können diese von Mobile IP verursachten Verzögerungen störend sein. Im Folgenden werden einige Optimierungen vorgestellt, die zum Teil schon im ursprünglichen RFC 3344 beschrieben wurden.

Encapsulation in der Mobile Node

Übernimmt der MN alle Funktionen eines FA, kann er auch in Netzen ohne FA über das Internet erreicht werden. Der MN muss dazu eine topologisch korrekte Adresse über Mechanismen wie etwa das DHCP beziehen und dann selbst die Registrierung beim HA vornehmen. Die bezogene IP-Adresse wird als *co-located Care-of Address (*siehe dazu RFC 3344) bezeichnet. Auch die Erneuerung der Registrierung und alle anderen Aufgaben des FA sind vom MN zu übernehmen. Da der MN somit auch zum Endpunkt des IP-Tunnels wird, ist das Entpacken (*Encapsulation*) im MN erforderlich. Ein IP-Tunnel bis zum MN ermöglicht eine wesentlich sicherere Kommunikation, da keinem FA Vertrauen entgegengebracht werden muss. Der Verzicht auf einen FA erschwert jedoch die Durchführung der Handoff-Verfahren. Beispielsweise übernimmt der FA beim Post-Registration-Handoff (siehe Abbildung 7-37) die Funktionalität eines Zwischenspeichers. Derartige Handoff-Verfahren können nur mit Hilfe von FAs realisiert werden.

Optimierung des Routings

Mobile IP sieht als Standardverfahren zum Routen das Dreiecks-Routing vor. Wie in Abbildung 7-35 zu sehen ist, werden beim Dreiecks-Routing die Datagramme immer über den HA zur MN übertragen. Um dieses ineffiziente Routing zu verbessern wurde ein Verfahren[2] zur Optimierung des Routings vorgeschlagen. Ziel ist es, ein direktes Routing zwischen CN und MN zu erreichen. Da eine CN aber über den aktuellen Aufenthaltsort der MN nicht informiert ist, muss zum Beginn

[2] Siehe dazu Internet Draft; *Route Optimization in Mobile IP*; C. Perkins, D. B. Johnson; Februar 2000; (draft-ietf-mobileip-optim-09.txt)

der Kommunikation weiterhin der HA kontaktiert werden. Wie in Abbildung 7-38 zu sehen ist, leitet der HA gemäß dem Dreiecks-Routing die Datagramme weiter, schickt aber an der CN eine *Binding-Update Message*[3] mit der aktuellen CoA[4]. Der CN kann diese Informationen ähnlich wie der HA in einer Tabelle verwalten. Ein CN, der Binding-Updates verwaltet, prüft vor jedem Senden, ob die aktuelle CoA den MN bekannt ist. Trifft dies zu, wird direkt zwischen CN und FA ein IP-Tunnel geöffnet und so der Datenverkehr ohne Umweg des HA geroutet. Da der HA den Datenverkehr so umleitet, dass der CN direkt mit der MN kommuniziert, bezeichnet man diese Optimierung als *HA-Redirects*.

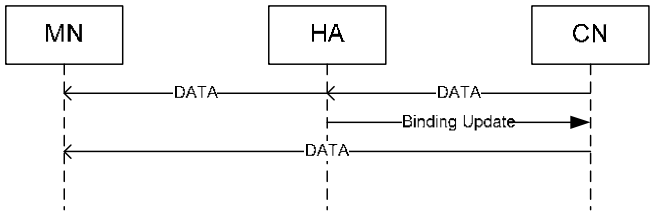


Abbildung 7-38: Mobile IP Routing Optimierung (HA Redirects)[5]

Der Vorteil dieser Routing-Optimierung ist das direkte und somit effizientere Routing. Eine weitere Verbesserung ist die Entlastung des HA, der nun nicht mehr den gesamten Datentransfer weiterleiten muss. Ein Ausfall des HA ist aber weiterhin kritisch, da die *Binding-Updates-Messages* nur über den HA zum CN gelangen. Der Einsatz von *HA-Redirect* ist nur möglich, wenn der CN *Binding-Updates* für alle in Kommunikation stehende MN verwaltet. Auch IP-Tunneling muss ein CN beherrschen, um das direkte Routing zum MN zu unterstützen. Neben den hohen Anforderungen an den CN gibt es aber auch datenschutzrechtliche Bedenken. Führt beispielsweise ein Anwender auf dem MN einen Internet-Dienst aus, bei dem er sich beim CN namentlich registrieren muss, können beim CN zum Nutzer des MN die verwendeten Adressen aus den Fremdnetzen (CoA) gespeichert werden. Anhand einer CoA kann ermittelt werden, welche Organisation das verwendete Fremdnetz verwaltet. Dienste dazu stehen im Internet zur Verfügung[6]. Da ein solches Teilnetz nur einen begrenzten geografischen Bereich abdeckt, kann der Standorte der MN eingeschränkt werden. Zeichnet der CN die Veränderung der CoA über den Zeitraum der Kommunikation auf, können so genannte Bewegungsdaten

---

3 Siehe dazu Internet Draft *Route Optimization in Mobile IP*.

4 Kann auch eine *co-located Care-of-Address* sein.

5 In der Abbildung wird das Routing ohne FA gezeigt (Encapsulation in der MN). Grundsätzlich ist *HA Redirects* auch mit der Unterstützung durch einen FA möglich.

6 Beispielsweise: http://www.ripe.net/whois.

zu einer Person erstellt werden. Die beim *Dreiecks-Routing* vom HA verwalteten Binding-Updates sind hingegen nicht problematisch, da der HA unter der gleichen organisatorischen Kontrolle wie der MN steht. Bei *HA Redirects* ist es jedoch jedem Kommunikationspartner möglich, Bewegungsdaten über den MN aufzuzeichnen.

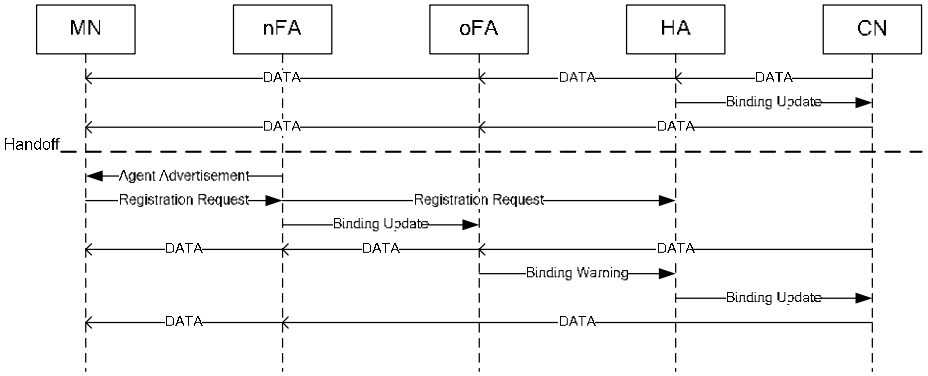


Abbildung 7-39: Handoff-Verfahren mit HA Redirects

Um auch beim Wechsel von Teilnetzen das direkte Routing aufrecht zu erhalten, muss der CN über diese Veränderung informiert werden. In der Abbildung 7-39 ist zunächst das direkte Routing etwas vereinfacht dargestellt. Nach dem Teilnetzwechsel (*Handoff*) registriert sich der MN beim neuen FA (nFA), der die Registrierung der CoA an den HA weiterleitet. Der alte FA (oFA) wird vom neuen FA mit einer *Binding Update Message* über die neue CoA informiert. Der alte FA ist somit in der Lage, noch eintreffende Datagramme in das neue Teilnetz weiterzuleiten. Um den CN über den neuen Aufenthaltsort zu informieren sendet der alte FA eine *Binding Warning Message*[7] an den HA. Der HA schickt darauf hin eine Binding-Update Message an den CN und sorgt somit für die Aktualisierung beim CN. Wie in Abbildung 7-39 zu sehen ist, sendet der CN nach dem Binding-Update wieder direkt zum neuen FA. Der HA hat somit zusätzlich die Aufgabe, sich alle CN zu merken, die aktuell mit der MN kommunizieren. Nur so kann er, nach dem Erhalt einer *Binding-Warning-Message*, die korrespondierenden CNs über den Teilnetzwechsel informieren.

Optimierung des Handoff-Verfahrens

Das Handoff-Verfahren für Mobile IP verursacht Latenzzeiten, die in der Anwendungsschicht zu Verzögerungen führen können. Wie in Abbildung 7-37 zu sehen ist, entsteht die Latenzzeit, da der MN sich zunächst im neuen Teilnetz registrieren und anschließend dem alten FA die neue CoA mitgeteilt werden muss. Der MN

[7] Siehe dazu Internet Draft *Route Optimization in Mobile IP*.

beendet somit zunächst die Verbindung zum alten FA und baut anschließend eine neue IP-Verbindung mit dem FA im aktuellen Teilnetz auf. Betrachtet man den Handoff in der darunterliegenden Schicht, ist festzustellen, dass in der Regel eine Verbindung mit beiden FAs möglich ist. Dies liegt an den meist überlappenden Sendebereichen der drahtlosen Zugangspunkte, die die Aufgabe der FA übernehmen. Abbildung 7-40 zeigt die Bewegung eines MN von einem Teilnetz zum nächsten. Wie zu sehen ist, durchquert der MN beim Teilnetzwechsel einen Bereich, in dem er von beiden FA Daten empfangen kann. Auf das spezielle Problem des ständigen Teilnetzwechsels, bei einer Bewegung entlang der Grenze zwischen zwei Teilnetzen (auch *Ping-Pong* genannt), wird hier nicht näher eingegangen.

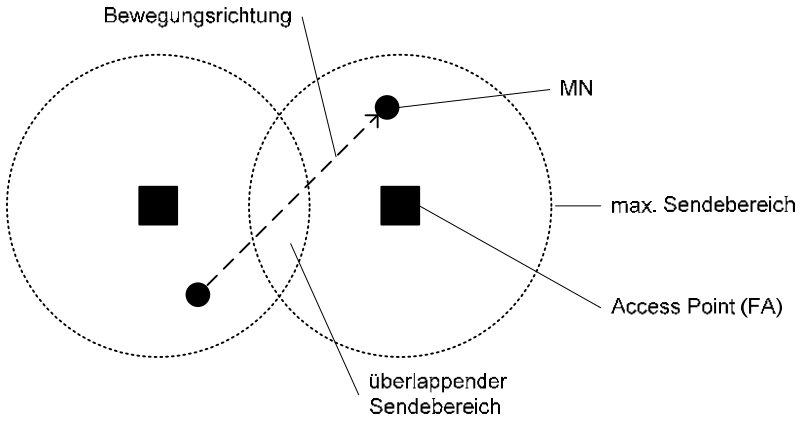


Abbildung 7-40: Überlappender Sendebereich der drahtlosen Zugangspunkte

Die Zeit, in der der MN sich im überlappenden Sendebereich befindet, kann zur Optimierung des Handoff-Verfahrens genutzt werden. Die ursprüngliche Beschreibung von Mobile IP sieht eine klare Trennung zur Schicht 2 vor und kann daher überlappende Sendebereiche nicht ausnutzen. In einem Vorschlag gemäß RFC 4881 die Latenzzeit zu verringern, wird diese klare Trennung aufgehoben und Informationen der Schicht 2 genutzt, um die Handoff-Verfahren zu verbessern. Im RFC 4881 werden dazu zwei Verfahren beschrieben, die auch in Kombination eingesetzt werden können. Die grundlegende Idee ist es Ereignisse, wie beispielsweise den Eintritt eines MN in den überlappenden Sendebereich zweier FAs, die in der Schicht 2 erkannt werden können, in der Schicht 3 (IP) zu nutzen, um den Handoff vorzubereiten. Da in der Schicht 2 verschiedenste Technologien für die drahtlose Kommunikation eingesetzt werden können, werden im RFC nur die grundlegenden Ereignisse beschrieben, ohne auf die speziellen Technologien einzugehen.

Das bereits weiter oben vorgestellte *Post-Registration-Handoff* kann optimiert werden, wenn schon während des Verlassens des aktuellen Teilnetzes ein IP-Tunnel zwischen dem alten FA und dem neuen FA aufgebaut wird. Dazu muss einer der beiden FAs den anstehenden Handoff anhand der Schicht-2-Ereignisse bemerken und eine Verbindung mit dem anderen FA aufbauen. Geht die Verbindung der MN mit dem alten FA verloren, kann diese noch eintreffende Datagramme schneller zum neuen FA weitergeleitet werden. Beim neuen FA müssen die Nachrichten allerdings unter Umständen solange zwischengespeichert werden, bis die Registrierung des MN abgeschlossen ist.

Neben dem *Post-Registration-Handoff* wird im RFC 4881 auch ein *Pre-Registration-Handoff* vorgeschlagen. Ziel des *Pre-Registration-Handoffs* ist es, den Handoff vor dem Verbindungsabbruch zum alten FA durchzuführen. Dazu ist es wichtig, dass die drahtlosen Zugangsknoten ihre direkten Nachbarn kennen. Welche FA als geografische Nachbarn in Frage kommen, muss ein FA beispielsweise über Schicht-2-Ereignisse wie einen schon vollzogenen Teilnetzwechsel erlernen. Ist eine Identifikation (z.B.: IP-Adresse oder Schicht-2-Adresse) eines Nachbarn bekannt, versendet der FA eine spezielle *Agent-Solicitation-Nachricht* (siehe RFC 3344), die nur an die Nachbarn gerichtet ist. Der benachbarte FA muss darauf mit einer *Agent-Advertisement-Nachricht* antworten. Die Antworten der benachbarten FAs werden dann zwischengespeichert. Durch die vorgezogene Abstimmung mit den Nachbarn und der Zwischenspeicherung der Antworten kann das eigentliche Handoff-Verfahren beschleunigt werden.

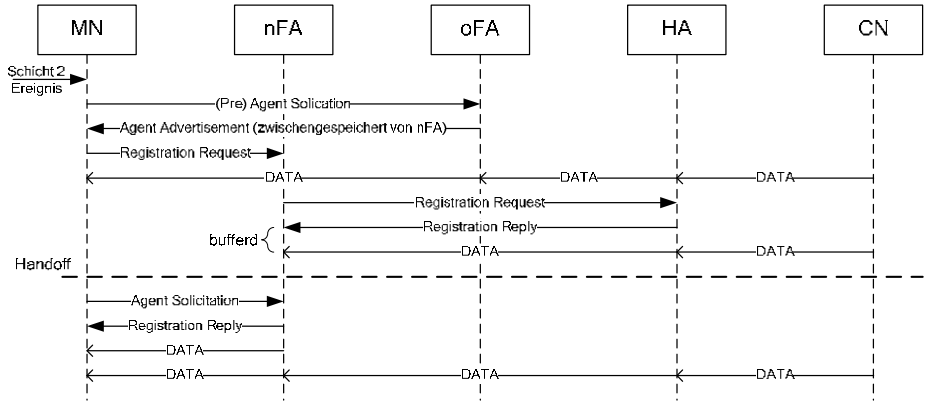


Abbildung 7-41: Pre-Registration-Handoff (Mobil-Initiated) nach RFC 4881

Im Folgenden wird eine spezielle Variante des *Pre-Registration-Handoffs* beispielhaft beschrieben, bei der der MN einen Handoff initiiert. Je nachdem welche Entität ein entsprechendes Schicht-2-Ereignis erkennt, kann der Handoff jedoch auch von einem der FAs gesteuert werden. Wie in Abbildung 7-41 zu sehen ist, gehen

wir bei unserer Betrachtung aber davon aus, dass der MN ein Schicht-2-Ereignis erhält, das auf einen bevorstehenden Handoff hinweist. Beispielsweise kann der MN die *BSSID*[8] eines benachbarten Access Points empfangen und anhand der gemessenen Signalstärke einen möglichen Handoff bemerken. Wenn der IP-Instanz des MN dieses Schicht-2-Ereignis mitgeteilt wird, kann der MN eine spezielle *Agent-Solication-Nachricht* an den alten FA senden, in der die BSSID des benachbarten Access Points enthalten ist. Der alte FA sendet daraufhin die zwischengespeicherte *Agent-Advertisement-Nachricht* des neuen FAs an den MN. Der MN erfährt so die IP-Adresse der neuen FA und kann aus der *Agent-Advertisement-Nachricht* sogar schon eine CoA für das neue Teilnetz beziehen. Bevor die Kommunikation zum alten FA abbricht, kann der MN die ausgewählte CoA, möglicherweise über den alten FA, beim neuen FA registrieren. Der neue FA leitet die Registrierung dann, wie in Abbildung 7-41 zu sehen ist, an den HA weiter. Im besten Fall ist die Bestätigung des HA beim neuen FA eingetroffen, bevor der MN eine IP-Verbindung zum neuen FA aufgebaut hat. Der Handoff ist somit vor der eigentlichen Registrierung beim neuen FA so weit vorbereitet, dass die Latenzzeit stark reduziert werden kann.

Eine weitere Optimierung kann erreicht werden, wenn beide Handoff-Verfahren kombiniert werden. Mit dem *Pre-Registration-Handoff* wird wie eben beschrieben versucht, den Teilnetzwechsel so vorzubereiten, dass vor dem Verbindungsabbruch zum alten FA die Registrierung der neuen CoA beim HA abgeschlossen ist. Parallel dazu wird aus Sicherheitsgründen auch ein *Post-Registration-Handoff* durchgeführt, um einen IP-Tunnel zwischen dem alten und dem neuen FA aufzubauen. Sollte die IP-Verbindung zum alten FA vor der Registrierung der neuen CoA abbrechen, können die noch eintreffenden Datagramme über den IP-Tunnel zum neuen Aufenthaltsort weitergeleitet werden. Ein möglicher Verlust der Datagramme wird so bei einem zu langsamen *Pre-Registration-Handoff* vermieden.

Zusammenfassend kann man festhalten, dass mit dem *Pre-Registration-Handoff* versucht wird, die Latenzzeit bei einem Teilnetzwechsel zu minimieren. Man spricht hier von einem *low-latency handoff*. Das *Post-Registration* Verfahren hat im Gegensatz dazu das Ziel, den Verlust (*low-loss-handoff*) von Datagrammen zu vermeiden. Die Kombination beider Verfahren, um beide Ziele zu erreichen, bezeichnet man auch als *seamless handoff*.

7.2 Mobilitätsunterstützung bei IPv6

Durch das enorme Wachstum des Internets und der neu aufkommenden Anforderungen, wie beispielsweise die Mobilitätsunterstützung, wurden Änderungen am Internet-Protokoll immer wichtiger. Um die schon eingesetzten Erweiterungen und

[8] Die Access Points der WLAN-Technologie können sogenannte Base Station Identifiers versenden um ihre Dienste anzubieten.

vorgeschlagenen Optimierungen zum Internet-Protokoll der Version 4 in einem gemeinsamen Standard zu sammeln, wurde die Version 6 des Internet-Protokolls entworfen. Eine Einführung zu IPv6 bietet der Abschnitt 4.4.

Bei der Entwicklung der Mobilitätsunterstützung von IPv6 (siehe dazu RFC 3775) wurden die als Erweiterung zu IPv4 entwickelten Verfahren genutzt. Die unter Mobile IP bekannte Mobilitätsunterstützung, wie auch die Routing-Optimierungen wurden für IPv6 überarbeitet und in den Protokollstandard integriert. Im Folgenden werden die wichtigsten Unterschiede von Mobile IPv6 zu Mobile IPv4 kurz beschrieben.

- Die Bereitstellung von speziellen FA ist nicht mehr nötig. Über jeden Router, der IPv6 unterstützt, kann ein MN arbeiten.
- Die Routing-Optimierung ist ein fester Bestandteil der Mobilitätsunterstützung bei IPv6 und nicht eine optionale Erweiterung wie bei IPv4.
- Spezielle Erweiterungs-Header zur Mobilitätsunterstützung werden genutzt. (z.B.: Vermeidung von IP-Spoofing)
- Sicherheitsaspekte wie etwa Authentifizierung sind Bestandteil von IPv6 und müssen nicht zusätzlich bereitgestellt werden.

Das grundsätzliche Verfahren der Mobilitätsunterstützung wurde für IPv6 von der Mobilitätsunterstützung der IP-Version 4 übernommen. Auch bei IPv6 ist ein MN immer über eine Adresse aus dem Heimnetzwerk erreichbar, auch wenn er in Fremdnetzen eine CoA zugewiesen bekommen hat. Auch die verwendeten Routing-Verfahren sind vergleichbar mit der Mobilitätsunterstützung von IPv4.

Eine wichtige Erneuerung ist die Verwendung von speziellen Erweiterungs-Headern[9] zur Realisierung der Mobilitätsunterstützung. Sendet beispielsweise der MN an einen CN Daten, wird die *Home-Address-Option*[10] verwendet, um der CN die feste IP-Adresse aus dem Heimnetz mitzuteilen. Der Aufbau einer *Home Address Option* ist der Abbildung 7-42 zu entnehmen. Im IPv6-Header kann daher als IP-Quelladresse die aktuelle und topologisch korrekte CoA verwendet werden. Das Routing von der MN zum CN kann somit normal durchgeführt werden (kein IP-Spoofing). Der CN wird durch die spezielle Erweiterung der IPv6-Header jedoch darüber informiert, dass die Quelladresse nur eine temporär gültige CoA ist. Die feste IP-Adresse aus dem Heimnetzwerk wird von der IP-Instanz in der CN an die höheren Schichten als Quelladresse übergeben. Die höheren Schichten und die darauf aufbauenden Anwendungen können so arbeiten, als gäbe es keine Veränderung der IP-Adresse durch die Mobilität.

---

[9] IPv6 Erweiterungs-Header werden im Abschnitt 4.4.3 beschrieben.

[10] Die *Home Address Option* ist ein spezieller Erweiterungs-Header (*Destination Options Header* nach RFC 2460) mit Optionen, die nur vom Empfänger gelesen werden müssen.

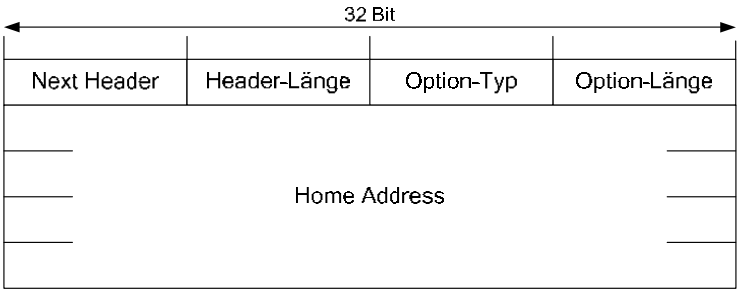


Abbildung 7-42: Home Address Option nach RFC 3775

Auch bei IPv6 hat der MN die Möglichkeit, mit einer *Binding Update Message* den HA über die aktuelle CoA zu informieren. Die Verwendung eines HA mit aktuellen *Mobility Bindings* ist auch bei IPv6 wichtig, um das Dreiecks-Routing zu ermöglichen. Ist einem CN die aktuelle CoA eines MN nicht bekannt, kann er die Nachricht wie bei Mobile IPv4 an die feste IP-Adresse aus dem Heimnetzwerk schicken.

Die Mobilitätsunterstützung von IPv6 sieht das direkte Routing vom CN zum MN als Standardverfahren vor. Im Gegensatz zu IPv4 ist hierfür jedoch kein IP-Tunnel zwischen dem CN und MN notwendig. Als IP-Zieladresse wird die aktuelle CoA des MN verwendet. Die Daten können somit ganz normal vom CN zum aktuellen Aufenthaltsort des MN weitergeleitet werden. Vom CN wird der IPv6-Header um einen speziellen Erweiterungs-Header ergänzt, der die feste IP-Adresse des MN aus dem Heimnetz enthält. Der zu verwendende Erweiterungs-Header ist ein Routing-Header des Typs 2 nach dem RFC 3775 und ist in der Abbildung 7-43 dargestellt. Auch der MN entnimmt die feste IP-Adresse aus dem Erweiterungs-Header und gibt nur diese Adresse an die höheren Schichten weiter. Auch auf der Seite des MN wird somit die Verwendung der CoA den höheren Sichten verborgen.

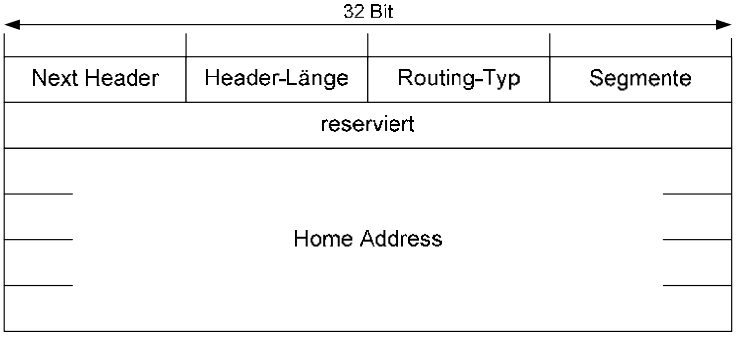


Abbildung 7-43: Typ 2 Routing Header nach RFC 3775

Der HA wird im Standardfall bei IPv6 von der kompletten Kommunikation entlastet. Auch die Weiterleitung von *Binding Updates* an den CN ist nicht mehr nötig. Spezielle FA sind für IPv6 nicht vorgesehen. MN arbeiten nur mit *co-located care-of Adressen* und alle IPv6-Router unterstützen die Mobilität und vor allem die optimierten Handoff-Verfahren. Zu beachten ist jedoch, dass IPv6 die Unterstützung des neuen Standards voraussetzt. Solange jedoch nicht alle beteiligten Knoten IPv6 unterstützen, was heute noch sehr häufig der Fall ist, ist IPv6 nicht in vollem Umfang funktionsfähig. Mobile IP basiert hingegen auf des verbreiteten Standard IPv4 und benötigt nur Erweiterungen in den Routern des Heim- und Fremdnetzen sowie im MN.

Ein von IPv6 nicht gelöstes Problem ist die mögliche Aufzeichnung von Bewegungsdaten. Um ein direktes Routing vom CN zum MN zu ermöglichen, ist die Verwaltung von Binding Updates in der CN auch bei IPv6 nötig.

7.3 Mobilitätsunterstützung bei TCP

TCP ist wie IP auf eine stationäre, drahtgebundene Kommunikation ausgelegt. Der Einsatz von mobilen Systemen führt auch bei TCP zu Problemen. Da eine drahtgebundene Kommunikation sehr stabil ist, geht TCP bei Paketverlust zunächst von einem Stau im Transportsystem aus. Der Verlust von Paketen ist bei drahtgebundener Kommunikation tatsächlich sehr selten. Bei einer drahtlosen Kommunikation ist diese Annahme jedoch nicht immer zutreffend. Die Verbindung zwischen einem mobilen System und einer Basisstation ist nicht annähernd so stabil wie die drahtgebundene Kommunikation. Es treten häufig Schwankungen in der Übertragungsleistung bis hin zu kurzen Verbindungsabbrüchen auf.

Die von TCP eingesetzten Verfahren zur Fluss- und Staukontrolle[11] sind für eine drahtlose Kommunikation nicht geeignet. Verbindungsprobleme zum mobilen System erkennt TCP erst über nicht eintreffende Quittierungen beim Sender. TCP geht dann von einem Stau aus und reduziert die Übertragungsleistung dramatisch. Da Verbindungsprobleme bei drahtloser Kommunikation jedoch meist nur sehr kurz auftreten, wäre die genau gegenteilige Reaktion – eine möglichst schnelle Sendewiederholung – für ein mobiles System nötig.

Für die mobile Datenkommunikation wird eine an die schlechtere Übertragungsqualität angepasste Fluss- und Staukontrolle benötigt. Auf dem Weg zwischen dem mobilen System und der Basisstation kann es leicht zu Paketverlusten oder Fehlern in der Reihenfolge der Pakete kommen. Auch der Wechsel von Teilnetzen führt immer wieder zu einer deutlich eingeschränkten Übertragungsleistung. Eine Anforderung an TCP ist eine Fluss- und Staukontrolle, die auch bei nicht drahtgebundenen Systemen effizient funktioniert. Die dazu neu entwickelten Verfahren

[11] Siehe hierzu Abschnitt 5.2.4 und 5.2.7.

müssen zu dem weit verbreiteten Standard TCP kompatibel sein, um eine Kommunikation zwischen mobilen und stationären Systemen zu ermöglichen. Wie bei IP ist eine vollständige Migration auf eine neue Version des TCP-Standards nicht so einfach möglich.

7.3.1 Performance Enhancing Proxy (PEP)

Die für das Internet verwendeten Übertragungsmedien unterscheiden sich teilweise in ihren Übertragungseigenschaften wesentlich. Neben den drahtgebundenen Standardmedien wie Ethernet sind Übertragungen mit Satellit oder über ein WLAN möglich. Das Übertragungsmedium wechselt jedoch nicht sehr häufig, so dass ein Paket auf dem Weg durch das Internet nur ein paar wenige Medienwechsel erfahren wird. *Performance Enhancing Proxies* (kurz PEP beschrieben im RFC 3135) teilen Netze mit unterschiedlichen Medien auf und schaffen so medieneinheitliche Netze. In diesen durch PEPs aufgeteilten Netzen können speziell für das Medium geeignete Protokolle oder Protokollerweiterungen eingesetzt werden. Der PEP sorgt für die nötige Kompatibilität zwischen den Netzen. Prinzipiell kann man PEPs auf allen Schichten des OSI-Modells aufsetzen.

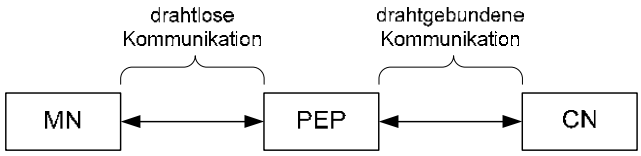


Abbildung 7-44: Einsatz eines Performance Enhancing Proxy

In Abbildung 7-44 ist ein PEP zu sehen, der die drahtlose Kommunikation zum mobilen System von der restlichen Kommunikation im Internet trennt. Durch den Einsatz eines *Performance Enhancing Proxy* kann TCP für den mobilen Einsatz optimiert werden. Zwischen dem MN und dem Zugangspunkt (z.B. ein Access Point) zum drahtgebunden Netzwerk kann ein, an die drahtgebundene Kommunikation angepasstes, Transportprotokoll eingesetzt werden. Der Zugangspunkt muss als PEP arbeiten und für die verbleibende Transportstrecke das unveränderte TCP einsetzen und so den Medienwechsel berücksichtigen. Im Folgenden werden zwei Verfahren vorgestellt, die aufbauend auf der PEP-Technologie eine Mobilitätsunterstützung für TCP ermöglichen. Es ist zu bemerken, dass neben diesen beiden Verfahren noch weitere Möglichkeiten der Mobilitätsunterstützung für TCP vorgeschlagen wurden.

7.3.2 Indirektes TCP

Nach (Bakre 1995) ist das indirekte TCP (I-TCP) eine mögliche Umsetzung der PEP-Technologie in der Transportschicht. I-TCP ermöglicht die Trennung von TCP-Netzen mit unterschiedlichen Medien. Im Beispiel eines mobilen Systems

(MN) würde die TCP-Verbindung vom MN zur Basisstation getrennt von der TCP-Verbindung zwischen der Basisstation und dem CN verwaltet. Da somit keine direkte TCP-Verbindung zwischen dem CN und dem MN besteht, spricht man von getrennten Verbindungen und somit einem indirekten TCP.

Abbildung 7-45 zeigt die Trennung der Kommunikation in die zwei TCP-Verbindungen zwischen dem CN und dem PEP sowie zwischen dem PEP und dem MN. Der PEP empfängt Pakete vom CN, speichert diese in einem Cache und sendet eine positive Quittierung. Dann sendet der PEP die gespeicherten Pakete an den MN. Der PEP stellt sicher, dass die Pakete den MN erreichen. Treten Verbindungsprobleme auf, wird vom PEP eine für die mobile Verbindungsstrecke optimierte Übertragungswiederholung eingesetzt. Da meist eine unmittelbare Verbindung zwischen MN und PEP besteht, ist die Datenübertragung gut zu beherrschen. Der Sender erfährt von Problemen zwischen PEP und MN nichts und kann daher das normale TCP einsetzen. Das Verfahren des I-TCP ist in beide Senderichtungen identisch. In der Abbildung 7-45 ist die Kommunikation vom CN zum MN dargestellt. Das Vorgehen von I-TCP ist für die umgekehrte Kommunikationsrichtung identisch.

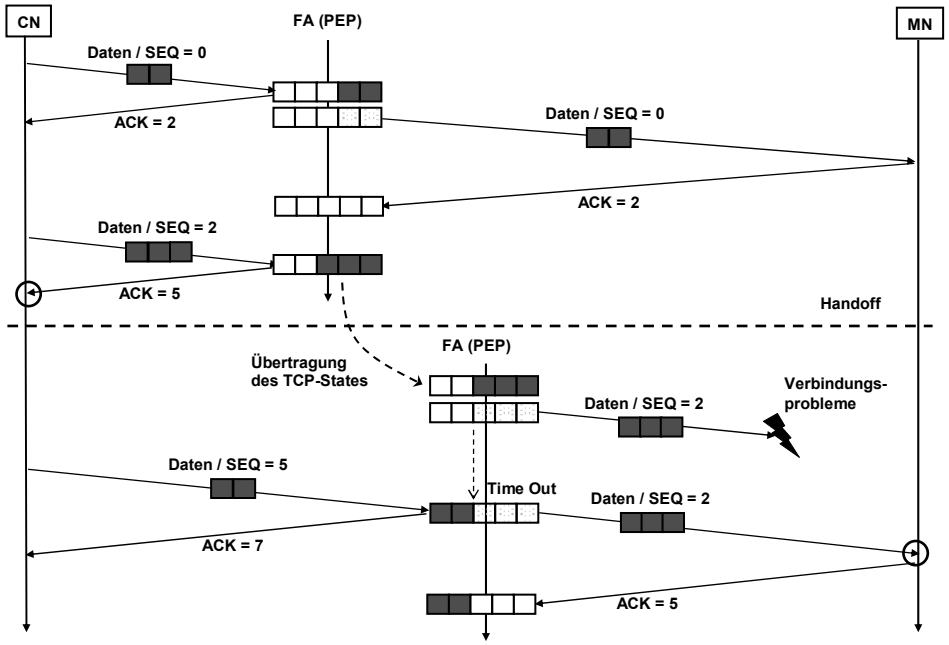


Abbildung 7-45: Datenübertragung vom CN zum MN bei indirektem TCP (I-TCP)

Ein Teilnetzwechsel (Handoff) ist bei I-TCP problematisch. Bei einem Handoff müssen zwischengespeicherte Pakete zum neuen PEP übertragen werden. Der

neue PEP kann diese dann an den MN weiterleiten. Damit dies jedoch möglich ist, kann ein Handoff nur zu PEPs vorgenommen werden, die auch I-TCP unterstützen. Neben dem Cache muss bei einem Handoff auch der TCP-Verbindungszustand von einem PEP zum nächsten übertragen werden. Die Weiterleitung des vorhandenen Caches zu einem neuen PEP kann unter Umständen einen längeren Zeitraum in Anspruch nehmen und so zu Verzögerungen führen. Verlässt der MN den Empfangsbereich eines PEP, dauert es eine gewisse Zeit, bis ein neuer PEP gefunden worden ist und die erforderlichen Konfigurationen und Registrierungen vorgenommen wurden. Erst danach können die Pakete an den neuen PEP übertragen und an den MN gesendet werden. Derart lange Übertragungszeiten sind für TCP unüblich, müssen aber vom PEP gegenüber dem CN verborgen werden. Wie in Abbildung 7-45 zu sehen, kann dies dazu führen, dass kurz vor dem Handoff ein eingehendes Paket quittiert wird, aber der PEP selbst nicht mehr zum Versand kommt. Treten dann beispielsweise noch anfängliche Verbindungsprobleme mit dem neuen PEP auf, kann der Versand des Paketes zusätzlich verzögert werden. Bei I-TCP ist es somit möglich, dass der Sender (CN) eine positive Quittierung für ein Paket erhält, das noch nicht einmal auf dem Weg zum Empfänger (MN) ist. Durch den Einsatz von I-TCP ist somit keine Ende-zu-Ende-Semantik zwischen CN und MN vorhanden. Die Ende-zu-Ende-Semantik von TCP besteht nur noch auf den durch den PEP getrennten Teilstrecken.

7.3.3 Snooping TCP

Da, wie gezeigt, der Verlust einer Ende-zu-Ende-Semantik in der Transportschicht zwischen Sender und Empfänger zu unerwünschten Effekten führen kann, wird mit Snooping TCP (S-TCP) ein PEP ohne Auftrennung der TCP-Verbindung realisiert. Bei S-TCP hört der PEP die Kommunikation zwischen CN und MN ab, um die Pakete ähnlich dem I-TCP zwischen zu speichern. Der PEP sendet jedoch selbst keine Quittierungen und lässt die TCP-Verbindung zwischen CN und MN unberührt.

Das Verfahren von S-TCP unterscheidet sich je nach Senderichtung. Zunächst betrachten wir die Kommunikation vom CN zum MN, die in Abbildung 7-46 gezeigt wird. Es ist zu sehen, dass der PEP die Pakete nur mitliest und zwischenspeichert. Auch die Quittierungen werden vom MN gesendet und vom PEP nur mitgelesen und gespeichert. Treten Verbindungsprobleme zum MN auf, erkennt der PEP dies durch einen eigenen Timeout-Mechanismus und sendet die Pakete erneut. Werden vom MN doppelte Quittierungen gesendet, um verloren gegangene Pakete anzuzeigen, sendet der PEP auch diese Pakete erneut an den MN und unterdrückt die Quittierungen gegenüber dem CN. Lokale Übertragungsprobleme werden also vom PEP behoben und nicht bis zum CN weitergegeben. Eine Ende-zu-Ende-Semantik bleibt jedoch erhalten, da der PEP nicht selbst Pakete quittiert, sondern nur eine Filterung der Bestätigungen vom MN vornimmt. Bei einem Handoff werden bei S-TCP die zwischengespeicherten Pakete nicht an einen neuen PEP über-

tragen. Da bei S-TCP der PEP keine eigene TCP-Verbindung hält, sondern nur die Verbindung zwischen CN und MN abhört, muss kein TCP-Verbindungszustand übertragen werden. Der MN kann somit auch zu Teilnetzen wechseln, deren Router kein S-TCP unterstützen. Pakete, die zum Zeitpunkt des Handoff auf dem Weg zum alten PEP waren, werden gemäß dem verwendeten Handoff-Verfahren von IP an den neuen FA weitergeleitet. Treten bei S-TCP verlängerte Übertragungszeiten bei einem Handoff auf, bemerkt dies der Sender anhand der ausbleibenden Quittierungen und kann so die Sendeleistung anpassen.

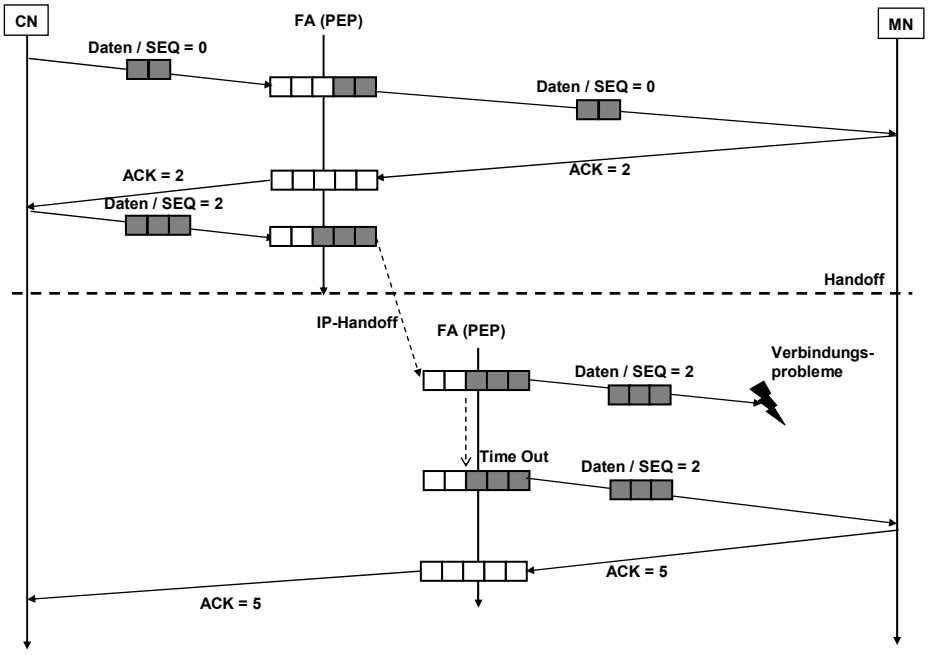


Abbildung 7-46: Datenübertragung vom CN zum MN bei Snooping-TCP (S-TCP)

S-TCP verhält sich bei der Kommunikation von der MN zur CN sehr ähnlich. Die Behandlung von Übertragungsfehlern ist jedoch aufgrund der veränderten Senderichtung unterschiedlich. Der PEP hört auch hier die Übertragung ab und erkennt dabei Übertragungsfehler. Gehen Pakete auf dem Weg zum PEP verloren, sendet der PEP eine doppelte positive Quittierung an den MN. Die frühe Erkennung von Übertragungsfehlern durch den PEP führt zu einer besseren Übertragungsleistung zwischen CN und MN. Doppelte Quittierungen, die vom CN gesendet werden, um einen erkannten Paketverlust anzuzeigen, werden vom PEP unterdrückt, falls sie schon behandelt wurden.

7.4 Resümee

In diesem Kapitel wurden verschiedene mobilitätsunterstützende Erweiterungen von IP und TCP dargestellt. Trotz der Fülle an Vorschlägen ist Mobile IP heute noch recht wenig im Einsatz. Gleiches gilt für mobiles TCP. Dies liegt sicher auch daran, dass die vorgestellten Erweiterungen zu IPv4 und TCP zum Teil recht komplex und nicht einfach zu implementieren und zu betreiben sind. In der Praxis reicht oft auch der Einsatz von DHCP für die Einbindung eines mobilen Rechners in ein fremdes IP-Netz aus. Mobile Computer werden heute zwar an verschiedenen Standorten, aber selten unterwegs, also während der Bewegung eingesetzt. Ein durchgängiger Einsatz von IPv6 brächte sicher auch hier einige Vorteile, da dieses Protokoll bereits standardmäßig eine umfassende Mobilitätsunterstützung bietet. Allerdings wird eine breite IPv6-Abdeckung noch einige Zeit auf sich warten lassen. Mobile Anwendungssysteme nehmen aber immer mehr zu, was dafür spricht, dass sich in Zukunft noch einiges verändern wird.

7.5 Übungsaufgaben

1. Welche Aufgabe übernimmt der HA bei der IPv4-Mobilitätsunterstützung?
2. Warum kann es beim Dreiecks-Routing, im Vergleich zur direkten Route bei einer normalen IP-Kommunikation zu deutlich schlechteren Laufzeiten der IP-Pakete kommen?
3. Erklären Sie kurz, warum es beim Wechsel von Teilnetzen ohne spezielles Handoff-Verfahren zum Verlust von Datenpaketen kommen kann.
4. Beschreiben Sie die unterschiedlichen Zielsetzungen des Pre- und Post-Registration-Handoff-Verfahrens beim Wechsel (Handoff) von Teilnetzen.
5. Erläutern Sie kurz das Optimierungsprinzip eines Performance-Enhancing-Proxy (PEP).

8 Entwicklung von Kommunikationsanwendungen

Die Entwicklung von Kommunikationsanwendungen ist in den letzten Jahren aufgrund neuer Technologien immer komfortabler geworden. Vor nicht allzu langer Zeit musste man sich bei der Programmierung noch mit allen Aspekten (auch der niedrigen Schichten) der Kommunikation befassen. Heute gibt es Programmiermodelle, die dies wesentlich erleichtern und man setzt mindestens auf eine vernünftige Transportzugriffsschicht auf. Für die Entwicklung von verteilten Anwendungen kann man also meistens einen funktionierenden Transportdienst nutzen.

Eine Methode der Programmierung basiert auf der Socket-Schnittstelle. Dies ist eine Transportzugriffsschnittstelle, die in mehreren Sprachen implementiert ist und aus der Unix-Welt kommt. Moderne Sprachen wie Java und C# stellen eine komfortable Nutzungsmöglichkeit über vordefinierte Packages (Java) oder Namespaces (C#) bereit. Sockets sind die Basis für alle im Internet vorhandenen höheren Programmiermodelle. Weiter fortgeschritten sind Konzepte wie RPC (Remote Procedure Call) und noch moderner sind objektorientierte Kommunikationsmechanismen wie RMI (Java-Welt), .NET Remoting (Microsoft) oder CORBA (unabhängiger Standard). Ganz aktuell sind komponentenorientierte Kommunikationsparadigmen wie EJB (Enterprise Java Beans), aber auch sog. Message-orientierte APIs (Application Programming Interface, Programmierschnittstelle) sind heute weit verbreitet. Moderne Kommunikationsmechanismen sind heute vorwiegend in Middleware-Produkten implementiert.

Die Art und Weise, wie man eine verteilte Anwendung programmiert, hängt von den Anforderungen ab. Die meisten verteilten Anwendungen stützen sich heute auf das Client-Server-Modell, in dem ein Serverprozess oder ggf. mehrere Serverprozesse auf Requests von Clients warten und diese beantworten. In diesem Fall geht die Kommunikation immer vom Client aus. Aber auch andere Kommunikationsparadigmen sind häufig anzutreffen. Ein anderes Modell ist die sog. Peer-to-Peer-Kommunikation, in der zwei Anwendungsprozesse gleichberechtigt miteinander kommunizieren. Beispielsweise benötigt man in Automatisierungsanwendungen meist eine gleichberechtigte Kommunikation, in der jeder Partner zu jeder Zeit eine Nachricht (oft Telegramm) senden kann, wenn ein bestimmtes Ereignis (wie z.B. „eine Palette ist an einem bestimmten Meldepunkt zum Einlagern in ein Hochregal bereit“) eintritt. Weiterführende Kommunikationsmechanismen befassen sich mit gesicherten Message-Queues, transaktionsgesicherter Kommunikation mehrerer Objekte usw.

In diesem Kapitel sollen die Grundlagen der Programmierung von verteilten Anwendungen aufgezeigt werden. Nach einer kurzen allgemeinen Einführung in Kommunikationsformen und einer Einführung in die Modellierung von Kommunikationsanwendungen wird auf die (mittlerweile als recht „low-level" eingestufte) Socket-Programmierung eingegangen. Die Socket-Programmierung erscheint uns deswegen so wichtig, weil man auf Basis dieser Technik einige grundlegende Mechanismen aufzeigen kann, die in höheren Kommunikationsmodellen nicht mehr betrachtet werden. Heutige verteilte Anwendungen nutzen zwar oft höhere APIs, und viele CORBA- oder EJB-Programmierer kennen sich mit Sockets gar nicht (mehr) aus. Grundsätzlich ist das in Ordnung, da der Kapselungsgedanke wichtig ist. Ein guter Programmierer verteilter Anwendungen muss aber gelegentlich Probleme lösen, die ihre Ursachen in unteren Kommunikationsprotokollen haben, und deshalb sollte man auch die unteren Schichten verstehen. Sonst passiert das, was in den vergangenen Jahren häufig der Fall war: Man denkt nur noch objektorientiert und glaubt, dass dies auch über ein Netzwerk problemlos angewendet werden kann. So hat man häufig den Fehler gemacht, bei verteilten Objekten viele Methoden übers Netz aufzurufen, um einzelne Attribute zu lesen. Dies führt natürlich auch bei den schnellen Netzen, die man heute hat, immer noch zu einem nicht vertretbaren Protokoll-Overhead.

Zielsetzung des Kapitels

Der Studierende soll verstehen, wie man über TCP- und UDP-Kommunikationsanwendungen auf Basis der Sockets-API programmiert.

Wichtige Begriffe

Synchrone Kommunikation, Asynchrone Kommunikation, Auftragsorientierte und meldungsorientierte Kommunikation, Datagramme, Rendezvous, Fehlersemantiken (Maybe, At-Least-Once, …), Socket, Port.

8.1 Kommunikationsformen

8.1.1 Synchrone und asynchrone Kommunikation

Beim Nachrichtenaustausch unterscheidet man nach (Weber 1998) grundsätzlich zwei Formen der Kommunikation: Synchrone und asynchrone Kommunikation. Da diese Begriffe in unterschiedlichen Zusammenhängen verwendet werden, wollen wir sie hier definieren:

– *Synchron* bedeutet im Zusammenhang mit der Programmierung von verteilten Anwendungen eine blockierende Kommunikation, d.h. der Sender wartet solange, bis eine Methode *send* (oder etwas Ähnliches) mit einem Ergebnis zurückkehrt und kann in dieser Wartezeit nichts tun.[1]

[1] Bei Nutzung von Threads natürlich schon, aber dazu später mehr.

– *Asynchron* bedeutet, dass der Sender nach dem Absenden der Nachricht mit Hilfe einer Methode *send* nicht blockiert und bis zum Eintreffen eines Ergebnisses andere Aufgaben erledigen kann. Die Antworten müssen dann irgendwie eingesammelt werden.

In jedem Fall werden die Puffer für ankommende Nachrichten in den Protokollinstanzen (meist im Betriebssystemkern) verwaltet. Die Instanzen kopieren die Nachrichten in den Adressraum der empfangenden Anwendungsprozesse. Die Pufferspeicher müssen organisiert werden, was Overhead bedeutet. Die Pufferspeicher benötigen Adressraum (Hauptspeicher) und sind begrenzt, was je nach Protokoll evtl. zum Verwerfen von Nachrichten kommt, wenn die Puffer voll sind.

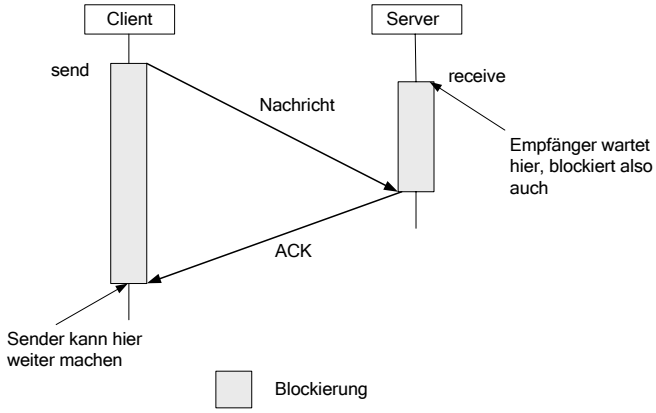


Abbildung 8-1: Synchrone Kommunikation und Blockieren nach (Weber 1998)

Vorteile der synchronen Kommunikation (vgl. Abbildung 8-1) sind die automatische Synchronisation zwischen Sender und Empfänger sowie das Einsparen von Pufferspeicher. Nachteilig sind die eingeschränkte Parallelität und das ggf. lange Blockieren, wenn der Empfänger keine Zeit hat. Aber auch der Empfänger selbst blockiert, wenn er auf Anfragen von Clients wartet.

Bei der asynchronen Kommunikation, wie sie in Abbildung 8-2 skizziert ist, hat man z.B. den Vorteil der zeitlichen Entkoppelung von Sender und Empfänger. Damit sind auch eine bessere Parallelarbeit sowie eine ereignisgesteuerte Kommunikation möglich. Von Nachteil ist, dass eine Zwischenpufferung der Nachrichten notwendig ist. Wenn der Puffer voll wird, dann führt dies trotzdem zum Blockieren, um eine gesicherte Übertragung zu gewährleisten.

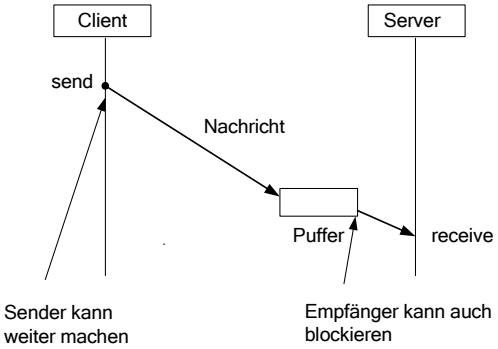


Abbildung 8-2: Asynchrone Kommunikation nach (Weber 1998)

8.1.2 Meldungs- und auftragsorientierte Kommunikation

Weiterhin unterscheidet man grundsätzlich zwischen meldungsorientierter und auftragsorientierter Kommunikation. Während die meldungsorientierte Kommunikation eine Einwegnachricht ohne Antwort ist, spricht man bei der auftragsorientierten Kommunikation von einen Request/Response-Mechanismus, bei dem der Sender ein Ergebnis vom Empfänger erhält (entfernter Dienstaufruf).

Betrachtet man die Tabelle 8-1, so gibt es verschiedene Kombinationsmöglichkeiten zwischen synchroner/asynchroner und meldungs-/auftragsorientierter Kommunikation. Diese sollen im Weiteren kurz diskutiert werden.

Tabelle 8-1: Kommunikationsformen nach (Weber 1998)

	asynchron	synchron
meldungsorientiert	Datagramm	Rendezvous
auftragsorientiert	asynchroner entfernter Dienstaufruf	synchroner entfernter Dienstaufruf

Betrachten wir zunächst die meldungsorientierte Kommunikation wie sie in Abbildung 8-3 dargestellt ist. Man unterscheidet hier synchrone und asynchrone Kommunikation. Die synchrone Kommunikation wird auch unter dem Begriff „Rendezvous“ geführt. Ein Client sendet eine Nachricht an einen in einer *receive*-Primitive blockierten (wartenden) Server, und die Synchronisierung erfolgt dadurch, dass der Sender auf die Quittung, nicht aber auf ein Ergebnis des Servers wartet. Bei der asynchronen Form der meldungsorientierten Kommunikation sendet der Client Datagramme, die nicht bestätigt werden. Der Client wird dadurch auch nicht blockiert und kann sofort andere Dinge erledigen. Der Server ruft eine

receive-Primitive auf, um auf Datagramme zu warten, verarbeitet ankommende Datagramme und ruft dann wieder das blockierende *receive* auf.

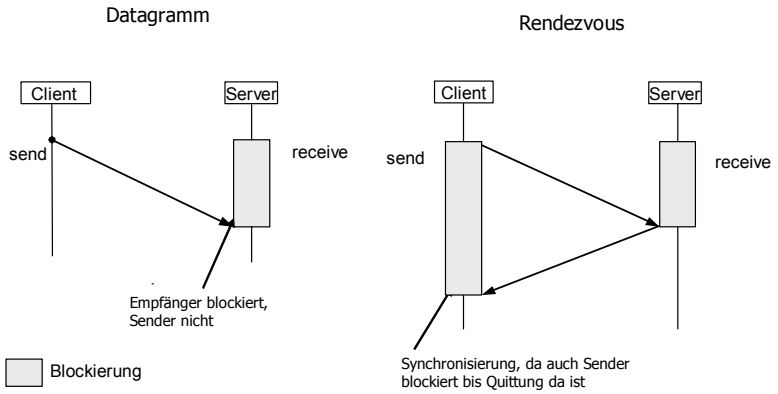


Abbildung 8-3: Meldungsorientierte Kommunikation nach (Weber 1998)

Bei der auftragsorientierten Kommunikation unterscheidet man ebenso synchrone und asynchrone Dienstaufrufe. In Abbildung 8-4 ist die synchrone Variante dargestellt, die gewöhnlich in RPC-Mechanismen (Remote Procedure Call) im Client--Server-Umfeld Anwendung findet. Sie ist auch verwandt mit Rendezvous. Im Unterschied zu Rendezvous wird bei der auftragsorientierten Kommunikation gleich ein Ergebnis geliefert. Der Client blockiert im synchronen Fall, bis die Antwort ankommt. Die Bearbeitung des Auftrags kann sehr komplex sein. Es könnte sich z.B. um eine oder mehrere Datenbankzugriffe handeln, die abgewickelt werden müssen.

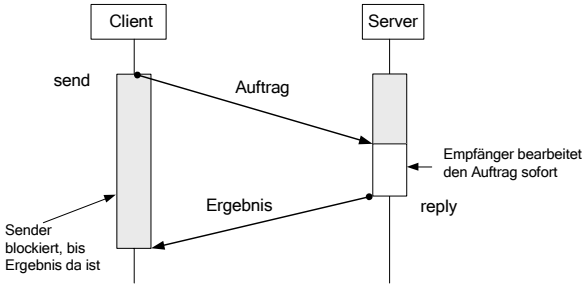


Abbildung 8-4: Synchroner entfernter Dienstaufruf nach (Weber 1998)

In der asynchronen Variante (vgl. Abbildung 8-5) der auftragsbezogenen Kommunikation kann der Client nach dem Versenden der Anfrage andere Aufgaben erledigen, muss sich allerdings irgendwann mit dem Server synchronisieren, was üblicherweise durch Aufruf einer *receive*-Primitive erfolgt.

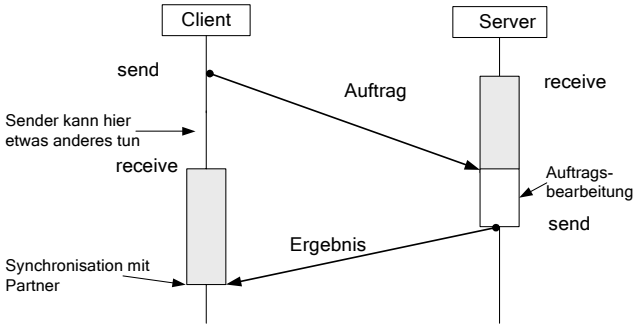


Abbildung 8-5: Asynchroner entfernter Dienstaufruf nach (Weber 1998)

Erwähnt sei noch, dass die angesprochenen Funktionsaufrufe bzw. Primitive (*send, receive*) durch eine Kommunikations-Middleware oder eine Transportzugriffsschicht bereitgestellt werden müssen. Welche Kommunikationsform man nun verwendet, hängt von der Anforderung der Anwendung ab. Wenn eine Client-Anwendung ohnehin erst nach dem Empfang eines Ergebnisses weiterarbeiten kann, ist eine blockierende Anfrage (synchron) sinnvoll. Kann der Client bis zum Empfang eines Ergebnisses etwas anderes tun oder erwartet gar kein Ergebnis, ist eine asynchrone Variante zu nutzen.

Natürlich reden wir immer noch von Protokollvarianten höherer Kommunikationsschichten, und es sollte klar sein, dass die Ergebniszuteilung grundsätzlich nichts mit einer Bestätigung (ACK-PDU) eines zuverlässigen Transportdienstes zu tun hat. ACK-PDUs werden zusätzlich gesendet, ohne dass es die höheren Schichten bemerken. Man kann auch ein Datagramm eines Anwendungsprotokolls über einen gesicherten Transportdienst senden. Hier hat man dann die Gewähr, dass das Datagramm richtig angekommen ist; ob es verarbeitet wird, weiß man dann nicht. Was man in der Transportschicht nicht gewährleisten kann, ist die Sicherstellung der Bearbeitung einer Anfrage. Diesem Thema widmet sich der nächste Abschnitt.

Abschließend soll noch darauf hingewiesen werden, dass man heutzutage eine Blockierung eines Anwendungsprogramms auch durch die Ausgliederung des Codings für die Kommunikation in einen eigenen, nebenläufigen Thread erreichen kann. Dies ist insbesondere bei Serveranwendungen wichtig, die viele Requests bearbeiten müssen. Jeder Request kann z.B. von einem eigenen Thread abgearbeitet werden.

8.1.3 Fehlersemantiken

Bei der Kommunikation in höheren Schichten gibt es trotz gesicherter Transportprotokolle viele Fehlerursachen. In manchen Situationen ist es schwierig festzustellen, ob eine Anfrage schon ausgeführt wurde oder nicht. Neben den klassischen

Fehlern im Netzwerk kann z.B. ein Sender vor dem Empfang des Ergebnisses ausfallen, was zu *verwaisten Aufträgen* (sog. *Orphans*) führt. Der Empfänger kann während der Bearbeitung eines Requests ausfallen, wenn z.B. der Anwendungsprozess aufgrund eines Programmfehlers abstürzt. Dies geht weit über die Sicherstellung der Kommunikation hinaus. Es geht vielmehr darum, ob aufgrund einer Nachricht (Auftrag, Request) schon eine Abarbeitungslogik durchlaufen wurde.

Ausfälle sind zu jeder Zeit möglich. Verschiedene sog. Fehlersemantiken sind möglich, und das Kommunikationssystem kann sich hier je nach Realisierung unterschiedlich verhalten. Man unterscheidet bei einem verteilt ausgeführten Request die Fehlersemantiken *Maybe*, *At-Least-Once*, *At-Most-Once* und *Exactly-Once*. *Maybe* ist die schwächste Form der Fehlersemantik, in der keine Fehlerbehandlung stattfindet. Im Fehlerfall kann man nicht feststellen, ob der Auftrag ausgeführt wurde oder nicht. *At-Least-Once* stellt sicher, dass der Auftrag wenigstens einmal ausgeführt wird, garantiert aber nicht, dass er genau einmal abgearbeitet wird.

At-Most-Once ist ähnlich wie *At-Least-Once*, filtert aber auch noch zusätzlich Duplikate aus. *At-Most-Once* wird heute von den meisten RPC-Mechanismen erfüllt. *At-Most-Once* hat gegenüber *At-Least-Once* Vorteile, aber ist auch aufwändiger in der Implementierung. Der Server muss eine Requestliste verwalten und bei jedem ankommenden Request prüfen, ob für ihn ein Eintrag in der Liste steht. Je nachdem sind dann die entsprechenden Aktionen auszuführen. Ist der Request z.B. schon in der Liste und schon ausgeführt, ist nur noch die Antwort zu übertragen. Weiterhin benötigt man eine zusätzliche ACK-Nachricht des Clients, die bestätigt, dass eine Antwort angekommen ist. Bei Systemausfällen gibt es auch bei *At-Most-Once* keine Garantie, also *At-Most-Once* kann dann ggf. nicht eingehalten werden.

Exactly-Once kann nur durch komplexe Transaktionsmechanismen gewährleistet werden. *Exactly-Once* bedeutet nämlich, dass ein Request, komme was wolle (auch bei einem Systemausfall), genau einmal ausgeführt wird. Dies erfordert für Systemausfälle Recovery-Mechanismen, konsistentes Zurücksetzen und damit die Verwaltung von Wiederaufsetzinformationen über Logging-Mechanismen. Diese Mechanismen sind in verteilten Transaktionssystemen und Datenbankmanagementsystemen zu finden.

Bei der Fehlersemantik *Maybe* wird im fehlerfreien Ablauf ein Request genau einmal im Server ausgeführt, und es wird genau einmal ein Ergebnis an den Client geliefert. Bei Nachrichtenverlust ist die Ausführung möglich, jedoch kommt es darauf an, ob die Request- oder die Response-PDU verloren wurde. In jedem Fall wird hier kein Ergebnis an den Client gesendet. Bei Ausfall des Servers vor dem Beenden der Requestbearbeitung ist eine Ausführung möglich, aber nicht sicher. Das Ergebnis wird in jedem Fall nicht an den Client geliefert.

Zusammenfassend kann man festhalten, dass nur bei der Fehlersemantik *Exactly-Once* bei allen Fehlersituationen eine korrekte Ausführung und Ergebnisauslieferung gegeben ist. Die Nutzung von At-*Least-Once* oder *At-Most-Once* führt bei Feh-

lersituationen zu Unsicherheiten über die Ausführung eines Requests. Verschiedene Beispielsituationen sollen die Problemstellung nochmals verdeutlichen. Es kann u.a. passieren, dass

- ein Request verloren geht,
- das Ergebnis des Servers verloren geht,
- der Server während der Ausführung des Requests abstürzt,
- der Server für die Bearbeitung des Requests zu lange braucht.

In Abbildung 8-6 ist an dem Fehlerszenario „Ergebnis des Servers geht verloren" dargestellt, was diese Fehlersituation nach sich zieht. Der Server hat die Auftragsbearbeitung schon ausgeführt und muss sich das merken, bei *Exactly-Once* sogar über einen Systemausfall hinweg. Der Client muss mit einer erneuten Anfrage reagieren, der Server darf aber die Anfrage bei *At-Most-Once* und *Exactly-Once* nicht noch einmal bearbeiten, sondern muss das temporär gespeicherte Ergebnis an den Client senden.

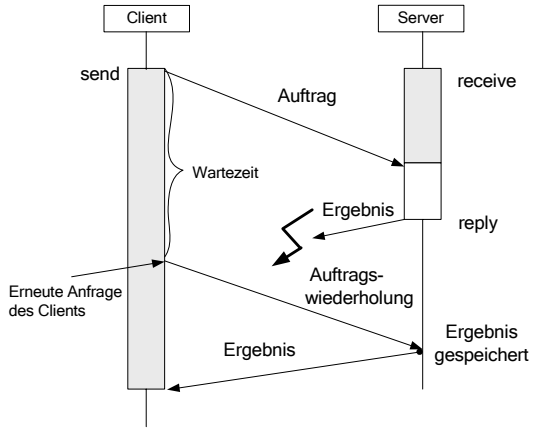


Abbildung 8-6: Beispielsituation: Ergebnis geht verloren nach (Weber 1998)

Transaktionsprotokolle, die *Exactly-Once* garantieren sollen, verwenden für die Ergebniskoordination sog. Commit- oder Koordinationsprotokolle (1-Phase-Commit. 2-Phase-Commit, 3-Phase-Commit), was aber nicht Gegenstand unserer Betrachtungen sein soll.

8.2 Entwicklung verteilter Anwendungen

8.2.1 Überblick über Modellierungstechniken

Verteilte Anwendungen und Kommunikationsprotokolle müssen ebenso wie lokale Anwendungen entwickelt werden. Das bedeutet, dass nach einer Definition der Anforderungen ein Entwurf (Design) erfolgt und danach erst die Programmierung und der Test. Eine iterative und inkrementelle Vorgehensweise ist empfehlenswert.

Verteilte Anwendungen können als endliche Automaten (Finite State Machine) beschrieben werden. Sie verhalten sich wie ereignisgesteuerte Systeme. Ereignisse, wie z.B. ankommende Nachrichten, werden aus der Umwelt entgegengenommen und über definierte Aktionen bearbeitet. Dabei durchlaufen die kommunizierenden Instanzen jeweils Zustandsautomaten. Nach der Ausführung einer Aktion senden sie in der Regel eine Nachricht nach außen.

Für die Modellierung verteilter Systeme bieten sich eine Reihe von Beschreibungsmethoden an, die von formalen Methoden wie LOTUS, ESTELLE und SDL[2], bis hin zu rein grafischen Möglichkeiten reichen.

Rein formale Methoden haben sich in der Praxis als zu kompliziert erwiesen. Hier wird noch viel geforscht. Man wendet heute eher visuelle Beschreibungsmöglichkeiten an, und da gibt es wiederum viele Möglichkeiten. Auch bei verteilten Anwendungen muss man das zu entwickelnde System aus verschiedenen Perspektiven betrachten, die sowohl statische Aspekte als auch laufzeitabhängige (dynamische) Aspekte abdecken.

Folgende Beschreibungsmöglichkeiten bieten sich (auch in Kombination) an:

- Mit den verfügbaren Notationen aus der UML (Unified Modelling Language) kann man einige (statische und dynamische) Sichten abdecken (StevensP 2000).
- Klassische Time-Sequence-Diagramme und Zustandsautomaten, wie sie in diesem Dokument mehrfach verwendet werden, ermöglichen die Modellierung des dynamischen Verhaltens eines Systems. Sequenzdiagramme (oder auch das Pendant aus der UML, Interaktionsdiagramme) eignen sich besonders, um ein bestimmtes Szenario mit seinem Nachrichtenfluss darzustellen.
- Eine verbale Beschreibung ist dort sinnvoll, wo grafische Notationen nicht ausreichen, und dies ist nach Meinung des Autors in der Regel immer der Fall.

---

[2] Formale Beschreibungsmethoden wie LOTUS (Language Of Temporal Ordering Specification), ESTELLE (Extended Finite State Machine Language) und SDL (Specification and Description Language) können in (Turner 1993) nachgelesen werden. Diese Methoden basieren auf Automaten.

– Die grafische Variante der SDL-Methode ist sowohl für den statischen, als auch für den dynamischen Teil nützlich. SDL (Standard SDL-2000) ist in der neuesten UML-Version ebenfalls enthalten. Die UML-Aktionsdiagramme bieten ähnliche Möglichkeiten an.

Die Anforderungsanalyse sollte unabhängig vom Design und das Design zunächst unabhängig vom konkreten Kommunikationsmechanismus (Sockets, RPC, RMI,…) erfolgen. Auf einem entsprechend hohen Abstraktionsniveau spielt es keine Rolle, ob die Kommunikation auf Basis von Sockets, RPC, RMI oder einer anderen Methode erfolgt. Das Feindesign muss sicherlich auf die Belange der konkreten Kommunikationsplattform eingehen und natürlich auch die Programmierung.

Wir wollen hier einige Empfehlungen aus eigener Erfahrung geben, ohne zu stark ins Detail zu gehen:

– *Anforderungsdefinition*: Die Anforderungen an eine verteilte Anwendung können mit Use-Cases (Text+Diagramme) beschrieben werden. Der Kommunikationsfluss kann dabei mit wichtigen Szenarien (vereinfachte UML-Interaktionsdiagramme ohne Objektbetrachtung) untermauert werden. Die Zustandsautomaten der einzelnen Kommunikationsinstanzen können mit SDL beschrieben werden, wobei anfänglich ein hohes Abstraktionsniveau zu empfehlen ist, man also nicht zu stark ins Detail gehen sollte.
– *Design:* Im Design muss eine geeignete Architektur gefunden werden. Subsysteme (Pakete) sind zu identifizieren. Die Zustandsautomaten sind zu verfeinern. Für die Subsystembildung eignen sich z.B. UML-Pakete oder die SDL-Systemspezifikation mit Blöcken.
– *Feindesign*: Hier wird das Design verfeinert und der aktuellen Umgebung angepasst, wobei auch die speziellen Anforderungen der einzusetzenden Kommunikationsmechanismen zu betrachten sind. Die Kommunikationsprozesse sind zu verfeinern. UML-Aktionsdiagramme oder SDL-Prozessspezifikation eignen sich gut. Es sollte hier deutlich gemacht werden, was eine Kommunikationsinstanz macht, wenn ein Ereignis eintrifft, also die Aktionen sollten beschrieben werden. Die Subsysteme sind zu verfeinern, wofür sich UML-Klassenmodelle[3] gut eignen.
– *Programmierung:* Die Klassenmodelle sind weiter zu verfeinern und die Zustandsautomaten in Programmen zu realisieren.

Wenn man es schafft, Implementierungsdetails so lange wie möglich zu ignorieren, kann man ein Design erreichen, das mit verschiedenen Kommunikationsparadigmen realisiert werden kann. Dies erhöht natürlich den Wiederverwendungsgrad des Designs, wenn man aus irgendeinem Grund die Plattform wechseln muss.

Weiterhin ist anzumerken, dass der Test von verteilten Anwendungen etwas komplexer ist als der Test lokaler Anwendungen. Insbesondere stark verteilte Sys-

[3] Wir gehen von einer objektorientierten Umsetzung aus.

teme sind schwer zu debuggen. Hier empfiehlt sich in jedem Fall der Einsatz eines Logging-Mechanismus, um Trace-Ausgaben auf Datei zu protokollieren.

Zudem sollte man sich bei der Konzeption und Implementierung immer mit dem Thema „Performance" befassen. Verteilte Anwendungen können auch heute (noch) nicht über beliebig viel Bandbreite verfügen. Meist gibt es hier Obergrenzen, die beachtet werden müssen. Performancetests sind – bei Unsicherheit am besten schon in frühen Entwicklungsphasen – unbedingt notwendig.

Bei dieser kurzen Einführung in die Modellierung und Realisierung von verteilten Anwendungen wollen wir es belassen. Modellierungstechniken gehören in den Bereich des Software Engineering, und hierfür gibt es eine Reihe von guten Büchern. Im nächsten Abschnitt wollen wir ein kurzes Beispiel zur Modellierung geben.

8.2.2 Fallbeispiel: Chat-Anwendung

Die Chat-Anwendung aus (Haase 2001) dient als Anregung für dieses Fallbeispiel. Im Folgenden sollen die Anforderungen an das Chat-System angerissen und anschließend ein erstes Design skizziert werden. Wir gehen also aus Sicht des Beispiels von Haase noch mal einen Schritt zurück und überlegen uns die Anforderungen und ein mögliches Design für die Chat-Anwendung. Es gibt viele Wege, die zum Ziel führen können, und man kann lange über eine vernünftige Lösung diskutieren. Deshalb finden wir diese Anwendung sehr geeignet für unsere Betrachtungen.

Anforderungsdefinition:

Beliebig viele Anwender sollen über eine einfach zu bedienende Zugangsmöglichkeit eine „Chat-Kommunikation" derart ausführen können, dass eine Nachricht, die ein Chat-Anwender sendet, an alle angeschlossenen Chat-Anwender übertragen wird.

Folgende vier Use-Cases soll das Chat-System unterstützen (vgl. Abbildung 8-7):

- Login durchführen: Ein Anwender registriert sich beim Chat-System mit einem Namen.
- Nachricht einstellen: Ein Anwender erfasst eine Nachricht im Dialog und stellt sie in das Chat-System ein, die allen registrierten Anwendern zugestellt wird.
- Nachricht empfangen: Ein Anwender empfängt die Nachrichten aller Chat-Teilnehmer an einer Oberfläche automatisch.
- Logout durchführen: Ein Anwender meldet sich vom Chat-System ab und erhält fortan keine Nachrichten mehr zugestellt.

Eine detailliertere Beschreibung der Anforderungen evtl. mit entsprechenden Use-Case-Templates ist dem Leser überlassen.

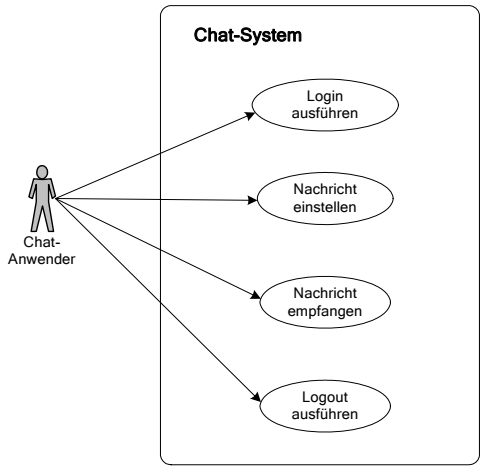


Abbildung 8-7: Use-Case-Modell des Chat-Systems

Ein paar Einschränkungen grenzen die Funktionalität des Chat-Systems ein, um das Beispiel möglichst einfach zu gestalten:

- Es gibt keine speziellen Chat-Gruppen, sondern nur eine Chat-Gruppe für alle registrierten Anwender.
- Nur Anwender, die gerade im Chat-System eingeloggt sind, erhalten Nachrichten. Ältere Nachrichten (vor einem Login-Zeitpunkt) werden nicht aufgehoben und können nicht eingesehen werden.
- Nach dem Ausschalten des Chat-Systems sind alle Daten verloren. Bei einem Neustart beginnt das System von vorne.
- Die Anwendung muss nur im Intranet funktionieren. Sicherheitstechnische Aspekte für eine Anwendung im globalen Internet sind nicht vorgesehen.
- Administrative Aspekte werden vernachlässigt. Das System muss man nur starten und stoppen können.
- Ausfallsicherheit wird nicht berücksichtigt.
- Wenn ein Anwender sich abmeldet, verschwindet er vollständig aus dem System und es bleibt keine Information im System erhalten.
- Wir sehen auch, dass zunächst nicht die Rede vom Design oder gar von Realisierungsaspekten ist. Das Chat-System ist rein fachlich beschrieben. Die Realisierung kann auf Basis beliebiger Kommunikationsparadigmen erfolgen. Aus Sicht des Anwenders spielt dies keine Rolle. Der Anwender sieht nur die Benutzeroberfläche, das zugrunde liegende Basissystem stellt Dienste bereit, es ist aber nicht sichtbar. Natürlich müsste man noch wesentlich detaillierter spezifizieren. Beispielsweise wird die Oberfläche gar nicht betrachtet. Aber für unser Beispiel soll das auch nicht geschehen, da es nur unnötig ablenkt.

Design:

Eine Chat-Gruppe kommuniziert von der Logik her so, dass eine Nachricht, die ein Chat-Client absendet, an alle anderen registrierten Chat-Clients gesendet werden muss. Um sicher zu sein, dass die Nachricht auch angekommen ist, muss von jedem Empfänger eine Bestätigung gesendet werden. Es handelt sich also um eine typische Anwendung für eine Multicast-Kommunikation gleichberechtigter Partner, wie dies in Abbildung 8-8 dargestellt ist.

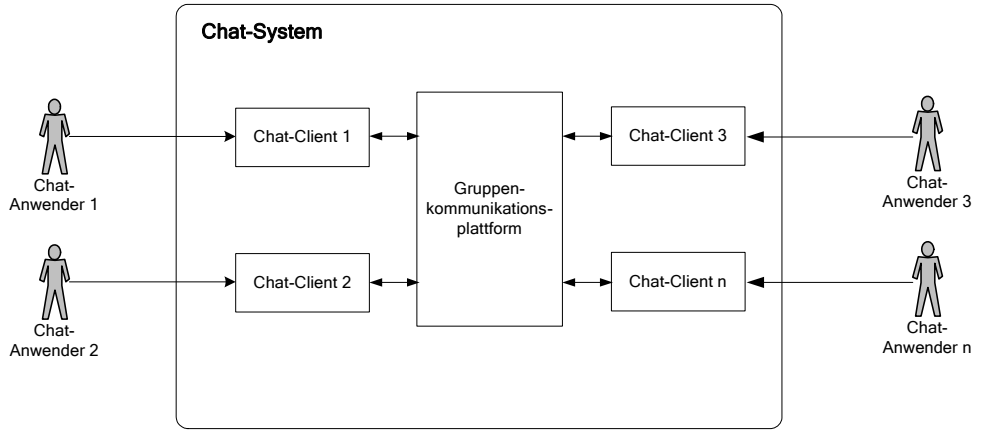


Abbildung 8-8: Gruppenkommunikation im Chat-System

Stellt der Chat-Anwender 1 eine Nachricht ein, so wird diese über eine Gruppenkommunikationsplattform an alle anderen Chat-Anwender übertragen. Ein evtl. Verlust einer Nachricht bei einem Systemausfall kann bei dieser Art von Anwendung toleriert werden, da die Daten nicht so kritisch sind.

Wir treffen aus didaktischen Gründen im Design eine erste wichtige Entscheidung, die sich gegen eine Gruppenkommunikation mit gleichberechtigten Partnern ausspricht. Wir entscheiden uns – wie es auch bei (Haase 2001) der Fall ist – für einen zentralen Chat-Server, der die Kontexte aller Chat-Clients verwaltet, also für eine Client-Server-Variante. Ein Grund dafür ist die Möglichkeit der zentralen Administrierbarkeit in einer zukünftigen Weiterentwicklung. Ein weiterer Grund ist die Möglichkeit, das System zukünftig auch über eine konventionelle Web-Kommunikation auf Basis von HTTP abwickeln zu können.

Diese Entscheidung hat eine schwerwiegende Folge. Der Chat-Server muss nämlich alle ankommenden Nachrichten entgegennehmen und an alle registrierten Chat-Clients verteilen. Die Chat-Clients müssen diese Ereignisse abfragen. Hierfür würde sich ein Publish/Subscribe- oder Eventing-Mechanismus anbieten, der aber nicht in jedem Kommunikationsmechanismus verfügbar ist.

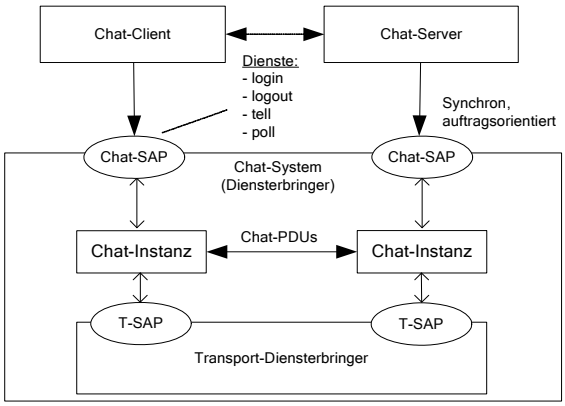


Abbildung 8-9: Grobarchitektur eines Chat-Systems

Daraus leiten wir eine erste Architektur für das Chat-System ab (siehe Abbildung 8-9):

- Das Chat-System bietet für jeden Chat-Client und den Chat-Server einen Chat-SAP an, in dem genau die Dienste angeboten werden, die ein Chat-Client benötigt. Die Dienste sind alle synchron (also blockierend) und heißen *login*, *logout*, *tell* und *poll*.
- Die Chat-Instanzen der Clients kommunizieren mit der Chat-Instanz des Servers über ein zu spezifizierendes Chat-Protokoll und tauschen dabei Chat-PDUs (oder bei höheren Kommunikationsmechanismen Methodenaufrufe) aus.
- Eine Kommunikation der Chat-Clients untereinander ist nicht vorgesehen.
- Die benötigte Transportschicht (bzw. höhere Diensteschicht) stellt eine gesicherte oder ungesicherte Kommunikationsverbindung (zuverlässiger Kommunikationsdienst ohne Duplikate und Nachrichtenverlust,...) bereit. Bei ungesicherter Transportschicht (wie UDP) muss die entsprechende Übertragungssicherheit ergänzt werden.

Die am Chat-SAP angebotenen Dienste können in die Kategorie „auftragsorientiert und synchron“ eingeordnet werden. Folgende Chat-PDUs sind erforderlich:

- Login-Request- und Login-Response-PDU
- Logout-Request- und Logout-Response-PDU
- Tell-Request- und Tell-Response-PDU (Nachricht einstellen)
- Poll-Request- und Poll-Response-PDU

Wie man sieht, sind für alle Request-PDUs auch Response-PDUs vorgesehen. Jeder Request wird also vom Partner bestätigt. Die Message-Request/Response-PDUs sind sowohl für das Einstellen einer Nachricht in das Chat-System als auch für die Zustellung einer Nachricht an den Chat-Client nutzbar.

Wir wollen zunächst eine Zerlegung in Subsysteme (Pakete) vornehmen. Drei Pakete kristallisieren sich bereits heraus: *ChatClientInstance*, *ChatServerInstance* und *ChatClient*. Weiterhin erscheint es zweckmäßig, die Anwendungslogik der Server-Instanz von der Kommunikationslogik zu trennen. Wir nennen das Paket *ChatServerManager*. Die *ChatClientInstance* stellt dem *ChatClient* eine Schnittstelle zur Verfügung und die *ChatServerInstance* stellt dem *ChatServerManager* eine Schnittstelle für die Kommunikation bereit. Diese Schnittstellen sollen so beschrieben werden, dass sie auf unterschiedliche Kommunikationsmechanismen abbildbar sind.

Die Zerlegung und Identifikation der erforderlichen Schnittstellen ist in Abbildung 8-10 skizziert. Neben den genannten Paketen haben wir ein Paket mit Basiskomponenten für das Logging und das Timermanagement ergänzt, die von allen anderen Subsystemen verwendet werden können. Eine weitere Zerlegung wird dem Feindesign überlassen.

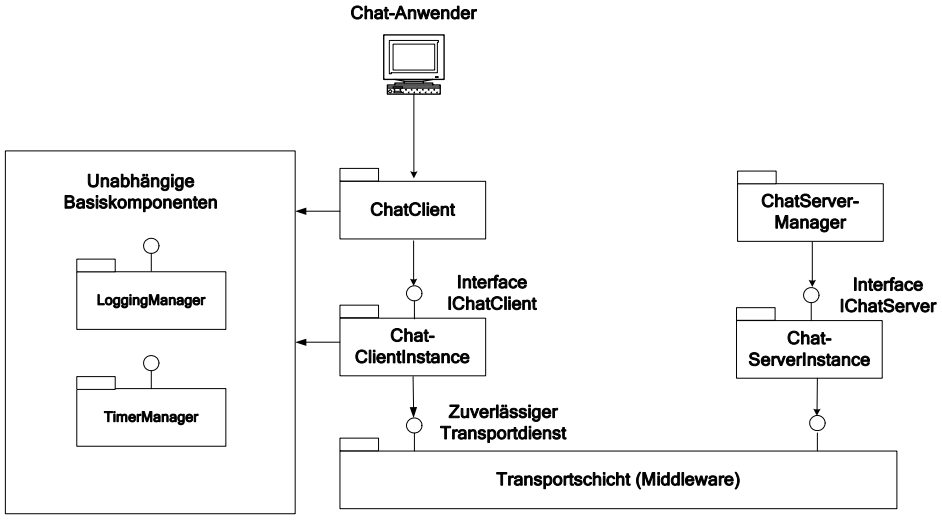


Abbildung 8-10: Subsysteme und Schnittstellen des Chat-Systems

Das Design sollte nun um Sequenz- oder Interaktionsdiagramme sowie Zustandsautomaten (SDL oder UML-Zustandsautomaten/-Aktivitätsdiagramme) für die Kommunikationsinstanzen (*ChatClient*- und *ChatServerInstanz*) und die Anwendungs-Subsysteme (*ChatClient* und *ChatServerManager*) ergänzt werden. Im anschließenden Feindesign ist dann ein spezieller Kommunikationsmechanismus mit einzubeziehen. Diese Aufgaben werden allerdings den Lesern als Übung zu diesem Kapitel überlassen.

8.3 Programmierung mit Sockets

8.3.1 Einführung und Programmiermodell

Die Socket-Schnittstelle ist eine klassische Transportzugriffsschnittstelle. Sie stellt ein API bereit, mit der man Kommunikationsanwendungen entwickeln kann. Die Socket-Schnittstelle ist heute der De-facto-Standard für ein API zur Kommunikation über TCP/IP und UDP/IP.

Sockets wurden in der Universität von Berkeley entwickelt (Unix BSD), und zwar in der ersten Version 4.1 des BSD-Systems für die VAX im Jahr 1982. Sie werden daher auch als Berkeley-Sockets bezeichnet. Die Originalversion der Socket-Schnittstelle stammt von Mitarbeitern der Firma BBN und wurde während eines ARPA-Projekts (1981) entwickelt (Hafner 2000).

Die Unterschiede verbindungsloser und verbindungsorientierter Kommunikationsabläufe spiegeln sich auch in den Aufrufen der Socket-Schnittstelle wieder. Es werden daher grundsätzlich TCP- und Datagramm-Sockets unterschieden.

TCP-Sockets. Die Socket-API für TCP-Sockets unterstützt vor allem Client-Server-Anwendungen[4], was aus dem Programmiermodell hervorgeht: Es gibt einen aktiven und einen passiven Partner. Als Socket bezeichnet man die Kommunikationsendpunkte innerhalb der Applikationen, die in der Initialisierungsphase miteinander verbunden werden. Dabei spielt es keine Rolle, auf welchen Rechnern die miteinander kommunizierenden Prozesse laufen.

In Abbildung 8-11 ist die Einbettung der Socket-Implementierung in ein Betriebssystem skizziert. Meistens ist sie Bestandteil des Betriebssystems.

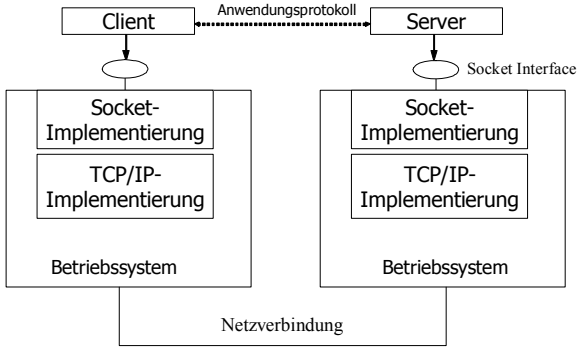


Abbildung 8-11: Einbettung der Socket-Implementierung

[4] Selbstverständlich kann man auch Peer-to-Peer-Anwendungen mit Sockets realisieren.

Die TCP-Socket-Schnittstelle ist streamorientiert. Beim Verbindungsaufbau wird ein Datenstrom zwischen den Kommunikationsendpunkten eingerichtet. Die UDP-Socket-Schnittstelle ist dagegen nachrichtenorientiert. Es ist demnach verbindungsorientierte (TCP-basierte) und verbindungslose (UDP-basierte) Kommunikation möglich. Die Adressierung der Kommunikationspartner erfolgt über die Kommunikationsendpunkte mit dem Tupel (IP-Adresse, Portnummer), das auch als Socket-Adresse bezeichnet wird. Der Aufbau der Kommunikationsbeziehung bei verbindungsorientierter Kommunikation läuft wie folgt ab, wobei man gerne die Begriffe Client für den aktiven Partner und Server für den passiven Partner verwendet:

- Beide Seiten rufen zunächst die *socket*-Primitive auf, um Kommunikationsendpunkte einzurichten. Im Server wird zusätzlich der (well-known oder ein anderer) Port an den Socket gebunden. Hierzu wird die Primitive *bind* verwendet.
- Die Serveranwendung wartet auf ankommende Verbindungsaufbauwünsche (Aufruf einer *listen*-Primitive) an einem TCP-Port.
- Die Clientanwendung ruft eine *connect*-Primitive auf, die T-Instanz erzeugt daraufhin eine Connect-Request-PDU und sendet sie zum Server.

Die Serveranwendung nimmt den Verbindungswunsch über einen *accept*-Aufruf entgegen und beantwortet ihn mit einer Connect-Response-PDU. Implizit wird ein Dreiwege-Handshake ausgeführt. Anschließend kann mit *send*- und *receive*-Primitiven (*recv*) ein bidirektionaler und vollduplex-fähiger Datenaustausch erfolgen. Die aufeinanderfolgenden Aufrufe von Socket-Primitiven auf der Client- und auf der Serverseite sind in Abbildung 8-12 skizziert.

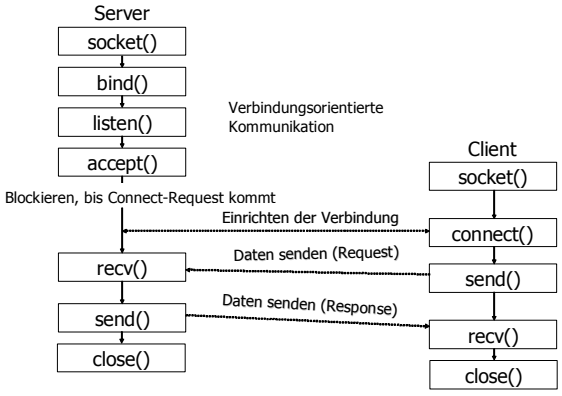


Abbildung 8-12: Socket-Primitive für die TCP-basierte Client-Server-Kommunikation

Aus der Abbildung wird auch deutlich, dass der Server auf ankommende Verbindungswünsche wartet und dann eine Verbindung eingerichtet wird. Komfortable

Serveranwendungen richten die Verbindung ein und gehen dann sofort wieder an den Wartepunkt, der über die *accept*-Primitive realisiert ist.

Für den Verbindungsabbau wird von einer beliebigen Seite eine *close*-Primitive abgesetzt, die ein TCP-Drei-Wege-Handshake einleitet.

Datagramm-Sockets. Im Falle einer UDP-Sockets-Kommunikation, wie sie in Abbildung 8-13 dargestellt ist, fällt der Verbindungsaufbau entsprechend weg. Beide Partner erzeugen ein Socket, binden jeweils eine Adresse und dann kann die Kommunikation beginnen. Hierfür stehen ähnliche Primitive wie bei TCP-Sockets bereit (*recvfrom*, *sendto*).

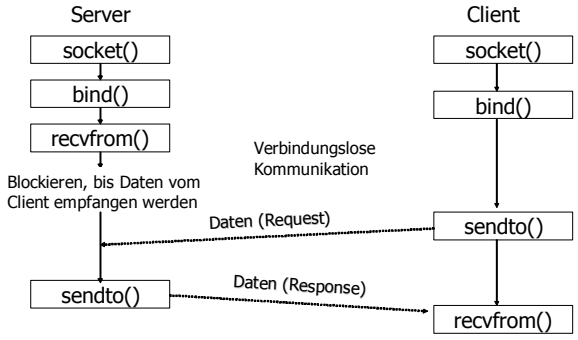


Abbildung 8-13: Socket-Primitive für die UDP-basierte Kommunikation

Im Unterschied zu TCP-Sockets gibt es keinen Verbindungsaufbau, beim Senden muss daher immer die Adresse des Partners mit angegeben werden.

Nach dieser Einführung in das Programmiermodell soll nun ein Überblick über die wichtigsten Socket-Funktionen den Einstieg in die darauf folgenden Beispiele erleichtern.

8.3.2 Die wichtigsten Socket-Funktionen im Überblick

Die C-Sockets-Schnittstelle wird in nahezu jedem Betriebssystem in einer Funktionsbibliothek bereitgestellt. Auf Basis der Funktionsbibliothek lässt es sich, verglichen mit anderen Sprachen wie Java, relativ aufwändig programmieren. Die meisten Socket-basierten Kommunikationsanwendungen sind heute aber (noch) in C/C++ programmiert.

Der folgende Auszug aus der API der Berkeley Software Distribution (BSD) zeigt elementare Funktionen für die Entwicklung von Kommunikationsanwendungen auf Basis von TCP und UDP (Stevens 2000). Da für die Initialisierung von UDP und TCP zum Teil die selben Socket-Funktionen mit verschiedenen Parametern verwendet werden, wird hier nach der Beschreibung zusätzlich aufgeführt, auf

welcher Seite (Server, Client) und bei welchem Protokoll (TCP/UDP) die Funktionen typischerweise verwendet werden.

Diese Schnittstellenspezifikation soll auch als Basis für die anderen Sockets-Implementierungen dienen. In Java und C# ist die Nutzung der Socket-API zwar komfortabler, aber im Prinzip läuft das Gleiche ab.

socket()

```
#include <sys/socket.h>

int socket(int family, int type, int protocol);
```

Initialisiert ein Socket zur Kommunikation und liefert bei Erfolg eine entsprechen Socket-Id (Socket-Deskriptor) zurück.

Verwendet von: Client und Server, TCP und UDP

Parameter:

family: Die verwendete Adressfamilie (hier wird auch oft das Prefix PF_ vorgefunden[5])

protocol: Das zu verwendende Protokoll der Adressfamilie. Oft 0, da die ersten beiden Angaben das Protokoll meist schon eindeutig spezifizieren

type: Der Socket-Typ (bei TCP: SOCK_STREAM)

Tabelle 8-3: Belegung des Parameters family bei *socket*-Aufruf

family	Beschreibung
AF_INET	IPv4-Protokolle
AF_INET6	IPv6-Protokolle
AF_LOCALE	Unix Domain Protokolle
AF_ROUTE	Routing–Sockets
AF_KEY	Key Socket

Tabelle 8-4: Belegung des Parameters *type* bei *socket*-Aufruf

type	Beschreibung
SOCK_STREAM	Stream-Socket

[5] Mit der Trennung von Adressfamilie (AF_*) und Protokollfamilie (PF_*) wollte man sicherstellen, dass Protokollfamilien mit mehreren Adressstrukturen unterstützt werden können. Da dies bisher nicht vorkommt, werden die jeweiligen Konstanten gleichgesetzt. Hier wird nur ein Auszug der möglichen Werte gezeigt, vgl. auch (Stevens 2000).

SOCK_DGRAM	Datagramm-Socket
SOCK_RAW	Raw-Socket

bind()

```
#include <sys/socket.h>

int bind(int sockfd, const struct sockaddr *myaddr, socklen_t addrlen);
```

Ordnet dem Socket eine lokale Adresse (myaddr) zu.

Verwendet von: Server (und Client[6]), TCP und UDP

Parameter:

sockfd: Der Socket-Deskriptor (siehe *socket())*

sockaddr: Zeiger auf eine Socket-Adressstruktur (bei TCP wird hier IP-Adresse und Port angegeben)

addrlen: Länge der Server-Adressstruktur

connect()

```
#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);
```

Baut eine Verbindung vom Client zum Server mit der im Parameter *servaddr* angegebenen Adresse auf. Bei TCP wird hiermit der Drei-Wege-Handshake initiiert. Findet zuvor kein *bind()* statt, wird hier eine zufällige lokale Adresse zugeordnet.

Verwendet von: Client, TCP

Parameter:

sockfd: Der Socket-Deskriptor (siehe socket())

serveraddr: Zeiger auf eine Socket-Adressstruktur (bei TCP wird hier die IP-Adresse und der Port angegeben)

addrlen: Länge der Server-Adressstruktur

listen()

```
#include <sys/socket.h>

int listen(int sockfd, int backlog);
```

Setzt das Socket in einen passiven, d.h. auf ankommende Verbindungswünsche wartenden Zustand.

Verwendet von: Server, TCP

---

[6] Beim TCP-Client wird bei Aufruf der *connect*-Funktion automatisch eine IP-Adresse und ein Port zugewiesen. Ein *bind*-Aufruf muss daher nicht verwendet werden.

Parameter:

sockfd: Der Socket-Deskriptor (siehe *socket()*)

backlog: Maximale Anzahl der ankommenden Verbindungen, die im Hintergrund (im Betriebssystemkernel) in einer Warteschlange eingereiht werden können

accept()

```
#include <sys/socket.h>

int accept(int sockfd, struct sockaddr *cliaddr, int *addrlen);
```

Wird bei TCP-Verbindungen verwendet und gibt die nächste ankommende und aufgebaute Verbindung aus der Warteschlange zurück.

Verwendet von: Server, TCP

Parameter:

sockfd: Der Socket-Deskriptor (siehe *socket()*)

cliaddr: Zeiger auf die Datenstruktur, in welches die Client-Adresse der Verbindung gespeichert wird

addrlen: Größe der Socket-Adressstruktur des Clients

close()

```
#include <unistd.h>

int close(int sockfd);
```

Schließt eine Verbindung. Bei TCP wird dabei versucht, alle noch nicht gesendeten Nachrichten zu senden und anschließend den Verbindungsabbau zu initialisieren. Nach Aufruf der Methode steht der Socket-Deskriptor nicht mehr zur Verfügung.

Verwendet von: Client und Server, TCP und UDP

Parameter:

sockfd: Der Socket-Deskriptor (siehe *socket()*)

recv()

```
#include <sys/types.h>

#include <sys/socket.h>

ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```

Liest Daten aus dem spezifizierten Socket und gibt die Anzahl der tatsächlich gelesenen Byte zurück.

Verwendet von: Client und Server, TCP

Parameter:

sockfd: Der Socket-Deskriptor (siehe *socket()*)

buf: Zeiger auf den Puffer, in den die ankommenden Daten geschrieben werden sollen

len: Die Anzahl der zu lesenden Byte

flags: Weitere Optionen zum Leseverhalten (vgl. Stevens 2000). Beispiel: Option MSG_WAITALL blockiert, bis alle erwarteten Daten (siehe *len*) im Stream zum Lesen bereit sind.

send()

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t send(int sockfd, const void *msg, size_t len, int flags);
```

Sendet Daten über das spezifizierte Socket und gibt die Anzahl der tatsächlich gesendeten Byte zurück.

Verwendet von: Client und Server, TCP

Parameter:

sockfd: Der Socket-Deskriptor (siehe *socket()*)

msg: Zeiger auf den Puffer, in dem die zu sendenden Daten vorliegen

len: Die Anzahl der zu sendenden Byte

flags: Weitere Optionen zum Sendeverhalten (vgl. Stevens 2000)

recvfrom()

```
#include <sys/socket.h>
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct
sockaddr *from, socklen_t *fromlen);
```

Empfängt eine Nachricht an dem angegebenen Socket, speichert sie im Puffer *buf* und gibt die Anzahl der gelesenen Byte zurück. Zusätzlich werden hier die Adressdaten des Senders in der übergebenen Adressstruktur *from* abgelegt.

Verwendet von: Client und Server, UDP

Parameter:

sockfd: Der Socket-Deskriptor (siehe *socket())*

buf: Zeiger auf den Puffer, in dem die zu sendenden Daten vorliegen

len: Die Anzahl der zu sendenden Byte

flags: Weitere Optionen zum Leseverhalten (vgl. Stevens 2000)

from: Zeiger auf die Adressstruktur in der die Client-Adressdaten gespeichert werden sollen.

fromlen: Zeiger auf einen Speicherbereich vom Typ *Integer*, in dem die Länge

von *from* abgelegt wird

sendto()

```
#include <sys/socket.h>

ssize_t sendto(int sockfd, const void *msg, size_t len, int flags, struct
sockaddr *to, socklen_t tolen);
```

Sendet eine Nachricht aus dem übergebenen Puffer *msg* an die Adresse *to* und liefert die Anzahl der gesendeten Byte zurück.

Verwendet von: Client und Server, UDP

Parameter:

sockfd:	Der Socket-Deskriptor (siehe *socket())*
msg:	Zeiger auf den Puffer, in dem die zu sendenden Daten vorliegen
len:	Die Anzahl der zu sendenden Byte
flags:	Weitere Optionen zum Sendeverhalten (vgl. Stevens 2000)
to:	Zeiger auf die Adresse des Empfängers
tolen:	Länge der Adresstruktur

Die Funktion *fork()* gehört zwar nicht zur Sockets-API, ist aber für die Verwaltung von mehreren TCP-Verbindungen wichtig und wird daher hier zusätzlich aufgeführt. Mit *fork()* können unter Unix und Unix-ähnlichen Betriebssystemen neue Prozesse erzeugt werden.

fork()

```
#include <unistd.h>

pid_t fork(void);
```

Erstellt eine Kopie des aktuellen Prozesses. Der neue erzeugte Prozess läuft als Kindprozess parallel zum erzeugenden Prozess (Elternprozess). Der Rückgabewert der Funktion ist im Elternprozess die Prozess-Id (pid) des Kindprozesses. Im Kindprozess wird 0 zurückgeliefert, da dieser immer über die Methode *getppid()* auf die pid des Elternprozesses zugreifen kann.

Diese Funktion wird bei Socket-Anwendungen genutzt, um die parallele Bearbeitung mehrerer Verbindungen sowie das gleichzeitige Entgegennehmen neuer Verbindungswünsche durch nebenläufige Prozesse zu ermöglichen.

8.3.3 Socket-Programmierung in C

Im Folgenden sind einige Ausschnitte aus einfachen Beispielanwendungen nach (Stevens 2000) skizziert, in denen jedoch solche wichtigen Aspekte wie die Fehlerbehandlung sehr stark vereinfacht wurden.

Beispielprogramm für TCP-Sockets: Im folgenden Beispielcode wird der Verbindungsaufbau zwischen einem TCP-Client und einem TCP-Server skizziert. Die eigentliche Verarbeitungslogik wird zur Vereinfachung weggelassen, und die Adressen sind im Programm als Konstanten angegeben[7]. Im Server wird zunächst ein Socket erzeugt und an eine Adresse gebunden (*socket*- und *bind*-Primitive). Das Zusammenstellen der Adresse ist in C etwas umständlich, da man eine Struktur vom Typ *sockaddr_in* befüllen muss. Da es verschiedene Socket-Typen gibt, muss beim *socket*-Aufruf der passende Typ (hier SOCK_STREAM für TCP-Sockets) angegeben werden. Es wird ein Socket-Descriptor zurückgegeben, der im Weiteren bei jedem Aufruf genutzt wird.

Server:

```
#include "inet.h"
#define SERV_TCP_PORT 5999
char *pname;...
main(int argc, char argv[]){
  int sockfd, newsockfd, clilen, childpid;
  struct sockaddr_in cli_addr, serv_addr;
  pname = argv[0];
  // Socket öffnen und binden
  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0) ) < 0) {
    // Fehlermeldung ausgeben und Fehlerbehandlung durchführen ...
    exit(1);
  }
  // Lokale Adresse binden, vorher sockaddr mit IP-Adresse und Port belegen
  bzero((char *) &serv_addr, sizeof(serv_addr));
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
  serv_addr.sin_port = htons(SERV_TCP_PORT);
  if (bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr) < 0) {
    // Fehlermeldung ausgeben und Fehlerbehandlung durchführen ...
    exit(1);
  } else {
    listen(sockfd, 5);
    for (;;) {// Auf Verbindungsaufbauwunsch warten
      clilen = sizeof(cli_addr);
      newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
      if ((childpid = fork()) < 0) {
      // Fehlermeldung ausgeben
      } else if (childpid == 0) { // Sohnprozess
```

[7] In guten Socket-Programmen wird man keine verdrahteten IP-Adressen vorfinden, sondern Hostnamen, die über DNS aufgelöst werden.

```
        close(sockfd);
        // Client-Request bearbeiten...
        exit(0);
      } //else
      close (newsockfd); // Elternprozess
    } // for
  } // if
}
```

Anschließend geht der Server in den Listen-Zustand, in dem er auf ankommende Verbindungswünsche wartet. Die Länge der Anfragewarteschlange besagt, dass gleichzeitig fünf Clients einen Verbindungsaufbauwunsch absetzen dürfen.

In der folgenden Endlosschleife wird für einen Verbindungsaufbauwunsch ein blockierender *accept* ausgeführt. Mit *accept* wird jeweils der erste in der Warteschlange stehende Verbindungsaufbauwunsch angenommen und für die neue Verbindung ein weiteres Socket mit den gleichen Eigenschaften wie das angegebene Socket generiert. Steht keine Anforderung an, blockiert der *accept*-Aufruf.[8] Nach Aufbau der Verbindung wird ein Kindprozess erzeugt und diesem die Verbindung zur weiteren Verarbeitung übergeben. Der Elternprozess geht sofort wieder zum *accept* und wartet auf den nächsten Verbindungsaufbauwunsch. Wenn der Kindprozess den Request abgearbeitet hat, terminiert er.

Diese Art der Programmierung ist typisch für viele heute existierende Anwendungen und ermöglicht durch die Einschaltung von Sohnprozessen die Parallelbearbeitung mehrerer Clients. Anstelle von mehreren Worker-Prozessen kann man hier auch Threads verwenden.

Der zugehörige Client ist sehr einfach gestaltet. Er kennt die IP-Adresse des Servers sowie die Portnummer des gewünschten Serverdienstes (hier 5999) und initiiert nach dem Anlegen eines Sockets den Verbindungsaufbau über einen *connect*-Aufruf. Anschließend sendet er einen Request, verarbeitet das Ergebnis und schließt das Socket. Dies ist eine sehr vereinfachte Socket-Anwendung. In der Regel wird eine bestehende Verbindung für mehrere Requests benutzt, da der Verbindungsaufbau doch relativ aufwändig ist.

Client:

```
#include ...
#include "inet.h"
#define SERV_TCP_PORT 5999

// Serveradresse
#define SERV_HOST_ADDR "192.43.235.6"
char *pname;
```

[8] Ein nicht blockierender Aufruf ist ebenfalls möglich.

```
main(int argc, char argv[]) {
  int sockfd;
  struct sockaddr_in serv_addr;
  pname = argv[0];
  // Serveradresse belegen...
  bzero((char *) &serv_addr, sizeof(serv_addr));
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
  serv_addr.sin_port = htons(SERV_TCP_PORT);
  // Socket öffnen
  if ((sockfd =socket(AF_INET, SOCK_STREAM, 0) < 0) {
    // Fehlermeldung ausgeben und Fehlerbehandlung durchführen ...
    exit(1);
  }
  // Verbindung zum Server aufbauen
  if (connect(sockfd,
    (struct sockaddr *) &serv_addr,  sizeof(serv_addr)) < 0) {
    // Fehlermeldung ausgeben...
  }
    // Verarbeitung: Request senden ...
  close(sockfd);
  exit(0);
}
```

Beispielprogramm für UDP-Sockets: Die Nutzung von Datagramm-Sockets ist im folgenden Beispiel dargestellt. Es handelt sich hier um einen einfachen Echo-Service. Der Client sendet eine Anfrage mit einem Text an den Server, der diesen einfach zurücksendet.

Der Server öffnet ein Datagramm-Socket (SOCK_DGRAM), bindet seine Adresse an das Socket und springt dann in eine Echo-Funktion. Diese Funktion macht nichts anderes als auf einen Request zu warten, ihn mit einem *recvfrom*-Aufruf zu lesen und die ganze Nachricht mit der *sendto*-Primitive an den Client zurückzusenden. Die Maximallänge der empfangenen Nachricht wird im *recvfrom*-Aufruf (MAXMESG) begrenzt.

Server:

```
#include „inet.h“;
#include <sys/types.h>
#include <sys/socket.h>
#define MAXMESG 2048
#define SERV_UDP_PORT 5999
...
main(int argc, char argv[])
{
```

```
  int sockfd;
  struct sockaddr_in serv_addr, cli_addr;
  pname = argv[0];
  // UDP-Socket oeffnen
  if (sockfd = socket(AF_INET, SOCK_DGRAM,0)) < 0) {
    // Fehlerbehandlung durchführen
  }
  /* Adresse aufbauen */
  bzero((char *) &serv_addr, sizeof(serv_addr));
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = inet_addr(INADDR_ANY);
  serv_addr.sin_port = htons(SERV_UDP_PORT);
  // Binde lokale Adresse
  if (bind(socketfd, &serv_addr,...) {
    // Fehlerbehandlung durchführen
  }
  else {
    wait_and_send (sockfd, (struct sockaddr *) &cli_addr, sizeof(cli_addr));
  }
}
/* Jede Nachricht wird zum Client zurück gesendet, echo */
wait_and_send(int sockfd, sockaddr pcli_addr, int maxclilen)
{
  int n, clilen;
  char mesg[MAXMESG];
  for (;;) {
    clilen = maxclilen;
    n = recvfrom(sockfd, mesg, MAXMESG, 0, pcli_addr, &clilen);
    if (n < 0) {
      // Fehlerbehandlung durchführen
    }
    if (sendto(sockfd, mesg, n, 0, pcli_addr, clilen) != n) {
      // Fehlerbehandlung durchführen
    }
  }
}
```

Der Client erzeugt zunächst ebenfalls ein Datagramm-Socket. Danach liest er einen Text aus der Standardeingabe ein, legt diesen in ein Datagramm und sendet dieses an den bekannten Server. Nach dem Empfang des Echos wird dieses auf die Standardausgabe ausgegeben, anschließend wird das Socket geschlossen und die Clientanwendung beendet.

Client:

```
#include „inet.h“
#include <stdio.h>
#include <sys/socket.h>
#define MAXLINE 512
#define SERV_UDP_PORT 5999
#define SERV_HOST_ADDR “192.43.235.6“
main(int argc, char argv[])
{
  int sockfd;
  struct sockaddr_in serv_addr, cli_addr;
  pname = argv[0];
  /* Adresse aufbauen */
  bzero((char *) &serv_addr, sizeof(serv_addr());
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
  serv_addr.sin_port = htons(SERV_UDP_PORT);
  // UDP-Socket öffnen und lokale Adresse binden
  if (sockfd = socket(AF_INET, SOCK_DGRAM,0)) < 0) {
    // Fehlerbehandlung durchführen
  }
  if (bind(socketfd, (struct sockaddr *) &cli_addr,...) {
    // Fehlerbehandlung durchführen
  }
  else {
    send_and_wait (stdin,
      sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
  }
  close(sockfd); exit(0);
}

/* Nachricht über stdin einlesen, zum Server senden, Echo wieder */
/* empfangen und auf stdout ausgeben */
send_and_wait (FILE fd, int sockfd, struct sockaddr *pserv_addr, int
servlen)
{
  int n;
  char sendline[MAXMESG]; recvline[MAXLINE+1];
  /* Nachricht von stdin einlesen … */
  if (sendto(sockfd, sendline, n, 0, pserv_addr, servlen) != n) {
    // Fehlerbehandlung durchführen …
  }
  n = recvfrom(sockfd, recvlen,
```

```
    MAXLINE, 0, (struct sockaddr *) 0, (int *) 0);
  if (n < 0)
  {
    // Fehlerbehandlung durchführen
  }
  /* Nachricht auf Standardausgabe (stdout) ausgeben … */
}
```

Darstellung. Kommunikationsprotokolle sollten unabhängig von der lokalen Syntax sein. Generell muss man sich bei der Socket-Programmierung selbst um die Darstellung der Daten in den Nachrichten kümmern. Zwei kommunizierende Prozesse wissen ja nicht, ob ihre lokalen Syntaxen gleich sind oder nicht, da ihnen die Computerarchitekturen der Rechner, auf denen sie laufen, nicht bekannt sind. Beispielsweise kann es sein, dass die beiden Rechner unterschiedliche Integer-Formate nutzen. Aus diesem Grund stellt die Socket-Schnittstelle Konvertierungs-Routinen bereit. Beispiele für Konvertierungs-Routinen sind *htonl* und *htons* zur Umwandlung von lokalen Long- und Short-Integerwerten in eine unabhängige Transportsyntax sowie die Umkehrfunktionen *ntohl* und *ntohs*.

In höheren RPC-basierten Protokollen werden für die Serialisierung von Nachrichten gewisse Automatismen bereitgestellt. Bei ONC-PRC, einer RPC-Implementierung von Sun Microsystems, wird z.B. XDR (eXternal Data Representation) zur Umwandlung beliebiger Datenstrukturen in eine Transportsyntax genutzt. Aus der ISO/OSI-Welt ist hier die *Abstract Syntax Notation* 1 (*ASN.1*) mit den zugehörigen *Basic Encoding Rules* (*BER*) bekannt, die über sog. ASN.1-Compiler unterstützt werden.

8.3.4 Java-Socket-Programmierung

Die Entwicklung von Java-basierten Socket-Programmen ist komfortabler als in C, da einfach zu nutzende Objektklassen bereitgestellt werden. Die Java-API stellt das Package *java.net* für die Netzwerkprogrammierung zur Verfügung, und hierin sind auch die Objektklassen zur Socket-Programmierung enthalten. Das Package *java.net* stellt eine höherwertige Schnittstelle für Sockets zur Verfügung, die als objektorientiert bezeichnet werden kann und die Socket-Details gut kapselt. Die Programmierung wird durch Input- und Output-Streams, die den Sockets zugeordnet werden, vereinfacht.

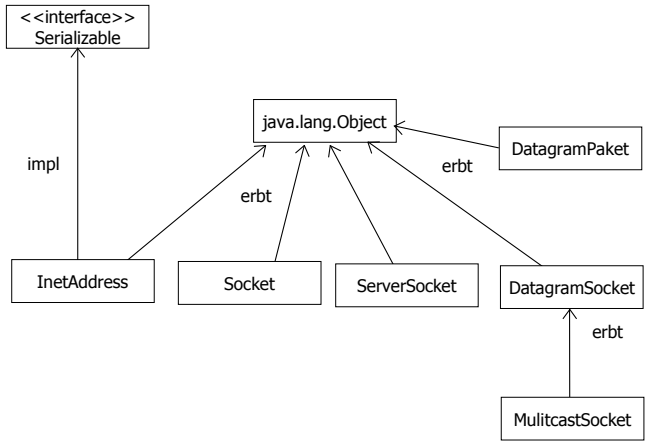


Abbildung 8-14: Objektklassen für Sockets aus java.net

In Abbildung 8-14 ist ein Ausschnitt aus der Java-Dokumentation des Packages *java.net* in einem Klassenmodell skizziert. Wie in Java üblich, sind Klassen entweder direkt oder indirekt von der Basisklasse *java.lang.Object* abgeleitet. Die Klasse *MulticastSocket* ist eine Unterklasse von *DatagramSocket* und dient zur Multicast-Kommunikation über UDP.

Beispielprogramm für Java-TCP-Sockets: In Java ist das Streamkonzept für TCP-Klassen sehr komfortabel. Daten, die über einen TCP-Stream gesendet werden, können über einen sog. Objektstrom serialisiert, also in eine Java-Transfersyntax gebracht werden. Man muss nur einen in Java vordefinierten Objektstrom über den eigentlichen Input- bzw. OutputStream legen. Bei Datagramm-Sockets ist dies aber nicht ohne weiteres möglich.

Wichtige Klassen für die Nutzung von TCP-Sockets sind:

- *InetAddress* für die Nutzung von Internet-Adressen
- *Socket* zur clientseitigen Socket-Nutzung
- *ServerSocket* zur serverseitigen Socket-Nutzung

Das Package *java.net* enthält auch Klassen zur Bearbeitung von UDP-Sockets. Wichtige Klassen sind hier:

- *DatagramSocket* als Socket-Interface
- *DatagramPaket* zur Aufbereitung und Bearbeitung von Datagrammen

Das Package *java.net* stellt auch vordefinierte Exceptions für die Ausnahmebehandlung bei Netzwerkproblemen bereit. Die Exception-Hierarchie für die Ausnahmebehandlung bei Java-Sockets sieht wie folgt aus, wobei die Bezeichnungen der Ausnahmen für sich sprechen:

```
java.lang.Object
```

```
java.lang.Throwable
  java.lang.Exception
    java.io.IOException
      java.net.ProtocolException
      java.net.UnknownHostException
      java.net.UnknownServiceException
      java.net.SocketException
        java.net.ConnectException
        java.net.BindException
              java.net.NoRouteToHostException
    ...
```

Die Ausnahmen werden ggf. nach einem Methodenaufruf erzeugt (geworfen) und müssen entsprechend behandelt werden.

In Abbildung 8-15 ist der Ablauf einer TCP-basierten Kommunikation mit Java-Mitteln schematisch skizziert. Der Server legt ein Server-Socket an, wobei ein Port angegeben wird, und ruft die Methode *accept* auf. Der Client erzeugt ein Socket, wobei implizit ein Verbindungswunsch abgesetzt wird, den der Server akzeptiert. Ein TCP-Drei-Wege-Handshake-Verbindungsaufbau wird ausgeführt. Anschließend stehen an den Sockets jeweils Input- und Output-Streams zur Verfügung, über die beide Seiten unabhängig voneinander Nachrichten senden und empfangen können. Es handelt sich also um einen bidirektionalen und vollduplexfähigen Kommunikationskanal. Beide Seiten dürfen also unabhängig voneinander senden.

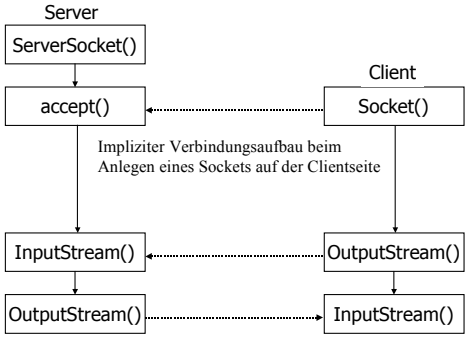


Abbildung 8-15: Ablauf einer TCP-Socket-Kommunikation mit Java

Der folgende Programmrahmen zeigt einen (stark vereinfachten) Server, der Client-Requests nur sequentiell abarbeiten kann. Der Server übernimmt die Verbindung und arbeitet den Request ab. Erst nach der Bearbeitung kann er wieder neue Requests empfangen.

Server:

```
ServerSocket server = new ServerSocket(Portnummer);
...
// Beispiel mit einem Thread, der auf einen Verbindungsaufbauwunsch wartet,
// den Client bedient, die Verbindung anschließend wieder beendet und
// auf die nächste Anfrage wartet.
while (true) {
    Socket incoming = server.accept();
    ObjectInputStream in;
    ObjectOutputStream out;
    try {
      out = new ObjectOutputStream(incoming.getOutputStream());
      in = new ObjectInputStream(incoming.getInputStream());
      // Empfangen über Inputstream
      MyPDU pdu = (MyPDU) in.readObject();
      // Empfangene PDU verarbeiten
      // ...
      // Senden über OutputStream
      // MyPDU ist eine eigene Objektklasse
      out.writeObject(new MyPDU(...));
      // Stream und Verbindung schließen
      incoming.close();
    }
    catch (Exception e) { ... }
}
...
```

Client:

```
// Client sendet nur eine Anfrage
...
  try {
    Socket client = new Socket (Host, Portnummer);
    ObjectInputStream in =
      new ObjectInputStream(client.getInputStream();
    ObjectOutputStream out =
      new ObjectOutputStream(client.getOutputStream());
    // Senden über OutputStream
    // MyPDU ist eine eigene Objektklasse
    out.writeObject(new MyPDU(...));

    // Empfangen über Inputstream
    MyPDU pdu = (MyPDU) in.readObject();
    // Response verabeiten
```

```
    // Stream und Verbindung schließen
    incoming.close();
  }
...
```

Die Streams sind quasi Bestandteil eines Socket-Objekts und können dort direkt angesprochen werden. Um serialisierte Daten zu übertragen, muss man den Input- und Output-Stream jeweils um einen Objektstrom-Filter ergänzen. Weiterhin muss man darauf achten, dass alle Objekte, die über die Objektströme ausgetauscht werden, das Java-Interface *Serializable* implementieren. Alternativ zur hier im Beispielcode angedeuteten seriellen Abarbeitung von Requests könnte man für die Bearbeitung nebenläufiger Requests Java-Threads nutzen. Verbindungen würden dann über eigene Threads angenommen und bearbeitet werden.

Beispielprogramm für Java-UDP-Sockets: UDP-basierte Sockets nutzen die Klasse *DatagramSocket* und wie der Name schon sagt, kann man damit eine datagrammorientierte Kommunikation realisieren. Portnummern können im Konstruktor angegeben oder im anderen Fall automatisch aus einem Portnummern-Pool vergeben werden. Methoden der Klasse *DatagramSocket* für die nachrichtenorientierte Kommunikation sind:

- void *send*(DatagramPacket p) zum Senden und
- void *receive*(DatagramPacket p) zum Empfangen von Datagrammen

Die Methode *receive* blockiert den Aufrufer, bis ein Datagramm eingeht. Datagramme werden über Instanzen der Klasse *DatagramPacket* erzeugt.

Für den Empfang von Datagrammen gibt man dem Konstruktor ein Byte-Array mit, in das die zu empfangenden Daten eingetragen werden sollen. Zum Versenden, gibt man dem Konstruktor neben den Daten (Byte-Array) die IP-Adresse und den Port des Empfängers (IP-Adresse und Portnummer) als Parameter mit. Es gibt mehrere Konstruktoren in dieser Klasse, die der Java-Dokumentation zu entnehmen sind.

Die Datagramme werden mit der Methode *receive* des Datagramm-Sockets empfangen. Die eigentlichen Daten und die Senderadresse werden mit Hilfe der Methoden *getData* bzw. *getAddress* des Objekts vom Typ *DatagramPacket* ausgelesen. Ein Datagramm-Socket wird durch den Aufruf der Methode *close* geschlossen.

Im Folgenden ist ein Beispielprogramm für eine einfache Echo-Anwendung auf Basis von UDP-Datagramm-Sockets dargestellt. Es sind zwei Klassen definiert, die die eigentliche Arbeit ausführen: UDPEchoServer und UDPEchoClient. Beide Klassen sind im Package *UDPEchoExample* angelegt. In den Konstruktoren werden die Adressen (Portnummer und beim Client auch der Hostname) angegeben. Die Parameter werden in der main-Methode jeweils über die Startzeile an die Programme übergeben. Die weitere Interpretation der Programme ist dem Leser überlassen.

Server:

```
package UDPEchoExample;
import java.net.*;
import java.io.*;

public class UDPEchoServer {
  protected DatagramSocket socket;

  public UDPEchoServer (int port) throws IOException
  {
    socket = new DatagramSocket (port);
  }

  public void execute() throws IOException
  {
    while (true)
    {
      DatagramPacket packet = receive();
      sendEcho (packet.getAddress (), packet.getPort (),
        packet.getData(), packet.getLength());
    }
  }

  protected DatagramPacket receive () throws IOException
  {
    byte buffer[] = new byte[65535];
    DatagramPacket packet = new DatagramPacket (buffer, buffer.length);
    socket.receive(packet);
    System.out.println ("Received " + packet.getLength () + " bytes.");
    return packet;
  }

  protected void sendEcho (InetAddress address,
    int port, byte data[], int length) throws IOException
  {
    DatagramPacket packet =
      new DatagramPacket (data, length, address, port);
    socket.send (packet);
    System.out.println („Response sent");
  }
  public static void main (String args[]) throws IOException
  {
    if (args.length != 1) {
```

```
      throw new RuntimeException ("Syntax: UDPEchoServer <port>");
      UDPEchoServer echo = new UDPEchoServer (Integer.parseInt (args[0]));
      echo.execute();
    }
  }
}
```

Client:

```
package UDPEchoExample;
import java.net.*;
import java.io.*;
public class UDPEchoClient
{
  protected DatagramSocket socket;
  protected DatagramPacket packet;
  public UDPEchoClient (String message,
    String host, int port) throws IOException
  {
    socket = new DatagramSocket ();
    // Datagramm aus message, host und port aufbauen
    // …
    try {
      sendPacket ();
      receivePacket ();
    }
    finally {
      socket.close ();
    }
  }

   protected void sendPacket () throws IOException
  {
    socket.send (packet); ...
  }

   protected void receivePacket () throws IOException
  {
    byte buffer[] = new byte[65535];
    DatagramPacket packet = new DatagramPacket (buffer, buffer.length);
    socket.receive (packet);
    ByteArrayInputStream byteI =
      new ByteArrayInputStream (packet.getData (), 0, packet.getLength ());
    DataInputStream dataI = new DataInputStream (byteI);
```

```
    String result = dataI.readUTF();
  }
  public static void main (String args[]) throws IOException
  {
    if (args.length != 3) {
      throw new RuntimeException („EchoClient <host> <port> <message>");
    }
    while (true) {
      new UDPEchoClient (args[2], args[0], Integer.parseInt (args[1]);
      try {
        Thread.sleep (1000);
      }
      catch (InterruptedException ex)
      {
        //...
      }
    }
  }
}
```

Abschließend soll noch die Klasse *MulticastSocket,* die zum Senden und Empfangen von IP-Multicast-Datagrammen dient, erwähnt werden. *MulticastSocket* ist von der Klasse *DatagramSocket* abgeleitet und kann zur UDP-basierten Gruppenkommunikation verwendet werden. Für das Multicasting werden im Internet die IP-Adressen 224.* - 239.* verwendet.

Die Klasse MulticastSocket stellt u.a. folgende Methoden zum Beitritt bzw. Austritt aus einer Multicast-Gruppe bereit:

- *joinGroup*(InetAddress groupAddr) zum Anbinden an eine Gruppe
- *leaveGroup*(InetAddress mcastaddr) zum Verlassen der Gruppe

Ein Datagramm, das mit Hilfe der oben beschriebenen DatagramSocket-Methode *send* gesendet wird, wird durch alle Gruppenmitglieder empfangen.

8.3.5 C#-Socket-Programmierung

Auch in der Sprache C# unter .NET ist eine Socket-API ähnlich wie in Java verfügbar. Die wichtigsten Klassen sollen kurz eingeführt werden. C# stellt hierfür einen eigenen Namespace *System.Net.Sockets* zur Verfügung. Der Namespace *System.Net.Sockets* stellt eine Implementierung der Windows Sockets-Schnittstelle (Winsock) bereit. Man unterscheidet einen „low-level" Mechanismus, in dem Client und Server zur Kommunikation die Klasse *Socket* verwenden und einen komfortableren Mechanismus. Einige Basisklassen werden ebenfalls benötigt:

- Die *IPAddress*-Klasse wird benötigt, um IP-Adressen zu bilden.
- Die *IPEndPoint*-Klasse enthält die Information eines Connection End Point (CEP), also die IP-Adresse und den Port.

Die Basisklasse *Sockets* stellt die Methoden zum Senden und Empfangen von Daten über einen *Sockets*-Stream bereit. Sockets können sowohl im synchronen (blockierenden) als auch im nicht-blockierenden Modus programmiert werden. Nicht-blockierende Programmierung ist insbesondere dann interessant, wenn eine Anwendung verschiedene Ereignisquellen bedienen muss. Hierfür verwendet man in der Regel nebenläufige Threads. Zur nicht-blockierenden Sockets-Programmierung mit C# sei auf die asynchronen Client-Sockets verwiesen.

Die Klassen *TcpClient* und *TcpListener* stellen eine komfortablere, aber nicht so mächtige Schnittstelle wie die *Socket*-Klasse bereit. Die Klasse *UdpClient* dient schließlich der Programmierung von UDP-Clients und UDP-Servern.

In Abbildung 8-16 sind die wesentlichen Klassen der C#-Socket-Programmierung dargestellt. Sie erben meistens direkt von der Klasse *System.Object*. Betrachtet wird zunächst die Klasse *Socket* etwas genauer. Sie stellt eine Anzahl von Methoden und Eigenschaften für die Programmierung von TCP- oder UDP-Anwendungen bereit.

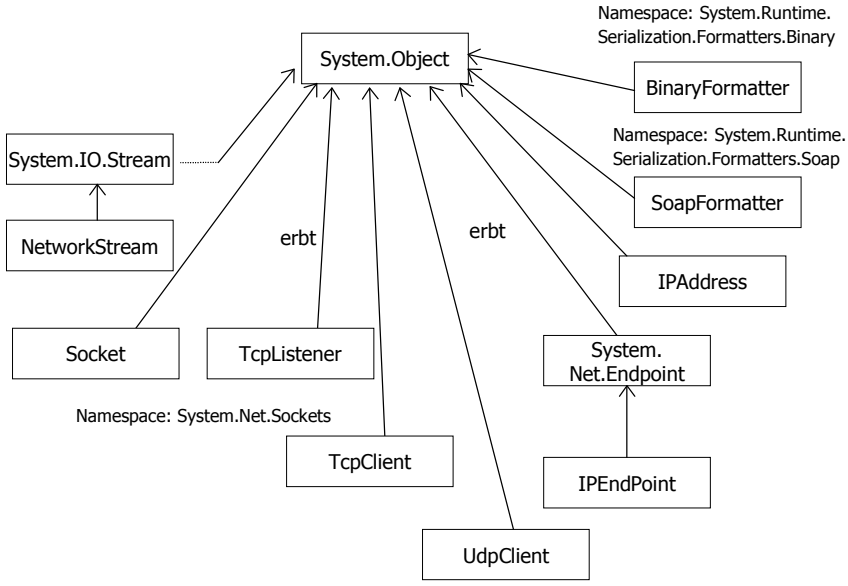


Abbildung 8-16: Die wichtigsten Klassen für die Socket-Programmierung in C#

In dieser Klasse sind die aus der C-Programmierung bekannten klassischen Socket-Methoden *Bind*, *Connect* für die Clientseite sowie *Listen* und *Accept* für die Serverseite verfügbar. Darüber hinaus gibt es Methoden zum Senden und Empfangen

von Nachrichten (*Send* und *Receive*) sowie weitere Methoden für die Programmierung von verbindungslosen UDP-Anwendungen. Die Kommunikation kann sowohl mit TCP als auch mit UDP blockierend und nicht-blockierend erfolgen. Die *Socket*-Klasse setzt hierzu das .NET-Framework-Benennungsmuster für synchrone und asynchrone Methodennamen um. Beispielsweise entspricht die synchrone *Receive*-Methode den asynchronen *BeginReceive*- und *EndReceive*-Methoden.

Mit der Klasse *Socket* hat man alle Möglichkeiten, allerdings ist die Programmierung etwas aufwändiger. Einfacher gestaltet sich die Programmierung über die vordefinierten C#-Klassen *TcpClient* und *TcpListener*.

Die Klasse *TcpClient* stellt einfache Methoden für die Verbindung, das Senden und Empfangen von Daten in einen Socket-Stream im synchronen Blockiermodus bereit. Damit ein TCP-Client eine Verbindung herstellen und Daten austauschen kann, muss auf der Serverseite die Klasse *TCPListener* (oder *Socket*) verwendet werden. Man kann eine TCP-Verbindung wie folgt herstellen:

- Der Server verwendet die Klasse *TCPListener*. Über den Aufruf der Methode *Start* wird eine Warteschlange eingerichtet, und danach werden eingehende Verbindungsanforderungen überwacht. Nach dem Aufruf der *Listen*-Methode „horcht" der Server auf ankommende Verbindungsaufbauwünsche von Clients. Der Port des Servers muss den Clients (wie üblich) bekannt sein.
- Der Client nutzt die Klasse *TcpClient* und ruft die Methode *Connect* auf, wobei er die Zieladresse des Servers angibt. Nun wird sofort eine TCP-Verbindung über den Konstruktor aufgebaut.
- Der Server nimmt den Verbindungaufbauwunsch mit Aufruf der Methode *Accept* an. Diese Methode ist blockierend. Will man das Blockieren vermeiden, kann man mit der Methode *Pending* ermitteln, ob ein Verbindungsaufbauwunsch in der Warteschlange steht. Die Methode *Accept* gibt einen weiteren Socket zurück, über den dann mit dem *Client* kommuniziert werden kann.

Beide Seiten können sich über die Verbindung evtl. einen Socket-Stream besorgen und die Daten serialisiert (siehe unten) senden und empfangen. Der Netzwerk-Stream kann mit der Methode *GetStream* oder über den Kontruktor der Klasse *NetworkStream* besorgt werden. Am Ende der Verbindung wird sowohl der Socket-Stream als auch die Verbindung auf beiden Seiten geschlossen. Die Methode *TcpListener.Stop* dient dazu, die Warteschlange freizugeben. Sie schließt keine Verbindung, beendet aber den *TcpListener*.

In C# ist ähnlich wie in Java auch ein Serialisierungsmechanismus realisiert, und zwar im *Namespace System.Runtime.Serialization*. Zur Serialisierung von Objekten gibt es verschiedene Ansätze, die als *Formatter* bezeichnet werden. Die Klasse *Formatter* ist eine abstrakte Basisklasse für alle Formatierungsprogramme zur Serialisierung. Man unterscheidet die *SoapFormatter* und die *BinaryFormatter*-Klasse. Erstere dient der Serialisierung über XML-Mechanismen, letztere dient der binären

Serialisierung. C#-Formatter sind nicht kompatibel mit Java-Objektströmen. Wenn man eigene Objektklassen in Nachrichten übertragen möchte und diese über einen Formatter serialisiert, kennzeichnet man diese über das Attribut *[Serializable].*

Für die Sockets-Programmierung ist auch eine Exception definiert. Die spezielle Socket-Exception ist in die folgende Vererbungshierarchie eingebettet:

```
System.Object
  System.Exception
    System.SystemException
      System.Runtime.InteropServices.ExternalException
        System.ComponentModel.Win32Exception
          System.Net.Sockets.SocketException
```

Beispielprogramm C#-TCP-Sockets: Anhand eines einfachen Beispiels soll im Folgenden die Programmierung mit den Klassen *TcpClient* und *TcpListener* unter Nutzung eines Socket-Streams und eines binären Formatters beschrieben werden. Das Beispiel ist eine einfache Client-Server-Anwendung, die nichts anderes macht, als einfache Strings auszutauschen. Ein Client sendet in einer Schleife mehrere Nachrichten mit Strings, die ein Server entgegennimmt und auf der Konsole ausgibt. Als letzte Nachricht sendet er den String „Ende“, was den Server veranlasst, die Verbindung abzubauen. Es gibt verschiedene Möglichkeiten der Programmierung, dieses Beispielprogramm stellt natürlich nur einen Lösungsansatz dar.

Das Programm ist wie folgt strukturiert:

- Eine Basisklasse *MyTCPSocketStream* für das Socket-Streams-Handling. Diese Klasse wird sowohl vom Client als auch vom Server verwendet.
- Eine Basisklasse *MyTcpSocketClient*, welche die clientseitigen TCP-Mechanismen einschalt (kapselt).
- Eine Basisklasse *MyTcpSocketServer*, welche der Kapselung der serverseitigen TCP-Mechanismen dient.
- Jeweils ein Hauptprogramm für den Client und den Server.

Das ganze Programm ist, insbesondere was das Exception-Handling angeht, stark vereinfacht. Socket-Streams können über die individuell entwickelte Basisklasse *MyTcpSocketStream* bedient werden. Diese Klasse kapselt lediglich die Streams-Nutzung und die Serialisierung über einen *BinaryFormatter*. Über die Konstruktoren wird der Stream einschließlich des Formatters erzeugt und kann im Weiteren zum Senden und Empfangen von beliebigen Objekten, die serialisierbar sein müssen, verwendet werden. Es sind zwei Konstruktoren definiert. Der erste Konstruktor ist für Server gedacht. Das Socket für die Verbindung zu einem Client wird beim Aufruf übergeben. Der zweite Konstruktor eignet sich für Clients, da ein Objekt vom Typ *TcpClient* übergeben wird.

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Runtime.Serialization.Formatters.Binary;
namespace TCPSocketBase {
  public class MyTcpSocketStream
  {
    private NetworkStream networkStream;
    private StreamWriter streamWriter;
    private StreamReader streamReader;
    private BinaryFormatter bf;
    private Socket incoming;
    private TcpClient clientSocket;
    public MyTcpSocketStream(Socket incoming)
    {
      this.incoming = incoming;
      networkStream = new NetworkStream(incoming);
      streamWriter = new StreamWriter(networkStream);
      streamReader = new StreamReader(networkStream);
      bf = new BinaryFormatter();
    }
    public MyTcpSocketStream(TcpClient clientSocket)
    {
      this.clientSocket = clientSocket;
      networkStream = clientSocket.GetStream();
      streamWriter = new  StreamWriter(networkStream);
      streamReader = new StreamReader(networkStream);
      bf = new BinaryFormatter();
    }
    public void Close()
    {
      try {
        streamWriter.Close();
        streamReader.Close();
        networkStream.Close();
        incoming.Close();
      }
      catch(Exception e) {
        Console.WriteLine("Exception: " + e.Message);
        throw;
      }
    }
```

```
    public object ReceiveObject()
    {
      object receivedData = null;
      try {
        receivedData = bf.Deserialize(networkStream);
      }
      catch(Exception e) {
        throw;
      }
      return receivedData;
    }
    public void SendObject(object messageObject)
    {
      try {
        bf.Serialize(networkStream, messageObject);
      }
      catch(Exception e) {
        throw;
      }
    }
  }
}
```

Die Basisklasse für die serverseitige Nutzung eines Sockets ist im Folgenden notiert. Im Konstruktor wird der Listener erzeugt und der Socket-Stream angelegt. Die Methode *Listen* liefert nach einem synchronen Warten auf einen Verbindungsaufbauwunsch einen Netzwerk-Stream, der dann für die weitere Kommunikation benutzt werden kann. In der Methode *Close* wird sowohl der Strom als auch die Verbindung abgebaut.

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Runtime.Serialization.Formatters.Binary;
namespace TCPSocketBase {
  public class MyTcpSocketServer
  {
    private TcpListener tcpListener; // Listen-Port
    private Socket incoming; // Socket
    private MyTcpSocketStream myStream; // TCP-Stream für Listen-Port
    public MyTcpSocketServer(int port)
    {
        tcpListener = new TcpListener(port);
```

```
        tcpListener.Start();
        // Erst jetzt kann auf TCP.Connect-Request gewartet werden
      }
      public MyTcpSocketStream Listen()
      {
      try {
        Console.WriteLine("Server wartet auf Verbindung ");
        incoming = tcpListener.AcceptSocket();
        myStream = new MyTcpSocketStream(incoming);
      }
      catch (Exception e) {
        Console.WriteLine("Exception: " + e.Message);
        throw;
      }
      return myStream;
    }
    public void Close()
    {
      myStream.Close();
      tcpListener.Stop();
    }
  }
}
```

Der TCP-Server (Klasse TcpServerTest) ist sehr einfach gestaltet. Es wird ein Socket erzeugt, über den auch eine Verbindung angenommen wird. Anschließend werden alle Nachrichten des Clients empfangen, bis dieser eine Nachricht mit einem String „Ende“ sendet. Die eigentliche Arbeit wird in der Methode Work ausgeführt. Der Port 9000 wird als Serverport genutzt.

Server:

```
using System;
using TCPSocketBase;
namespace TestServerSocket {
  public class TcpServerTest
  {
    // Serversocket
    private MyTcpSocketServer server;
    // TCP-Stream
    private MyTcpSocketStream myServerSocketStream;
    public TcpServerTest()
    {
      // Nichts zu konstruieren
    }
```

```
    public void Work()
    {
      bool finished = false;

      try {
        server = new MyTcpSocketServer(9000);
        // Auf TCP.Connect-Request warten
        myServerSocketStream = server.Listen();
      }
      catch (Exception e)
      {
        Console.WriteLine("Exception: " + e.Message);
        finished = true;
      }
      while (! finished) {
        try {
          string recMessage =
          (string) myServerSocketStream.ReceiveObject();
          Console.WriteLine("Empfangene Nachricht: " + recMessage);
          if (recMessage.Substring(0) == "Ende") {
                finished = true;
          }
        }
        catch (Exception e){
          Console.WriteLine
          ("Exception: " + e.Message);
          finished = true;
        }
      }
      // Stream und Socket schließen
      server.Close();
      Console.Write("Weiter>");
      Console.ReadLine();
    }
    public static void Main(string[] args)
    {
      TcpServerTest t = new TcpServerTest();
      t.Work();
    }
  }
}
```

Die Basisklasse des Beispiels mit dem Namen *MyTcpSocketClient* für die clientseitige Programmierung kapselt die Nutzung eines clientseitigen Sockets und erzeugt im Konstruktor einen Socket-Stream sowie eine Instanz der Klasse *BinaryFormatter* zur Serialisierung. Über die Methode *Open* kann ein „Stream-Handle“ besorgt werden, über den Nachrichten empfangen und gesendet werden können.

Client:

```
using System;
using System.IO;
using System.Net.Sockets;
using System.Net;
using System.Runtime.Serialization.Formatters.Binary;
namespace TCPSocketBase {
  public class MyTcpSocketClient
  {
    private TcpClient myClient;
    private MyTcpSocketStream myStream;  // TCP-Stream

    public MyTcpSocketClient (string host, int port)
    {
      try {
        myClient = new TcpClient(host, port);
        myStream = new MyTcpSocketStream(myClient);
        Console.WriteLine
        ("Socket erzeugt und verbunden mit " + host + "/" + port);
      }
      catch (Exception e) {
        Console.WriteLine("Exception: " + e.Message);
        throw;
      }
      finally {
        Console.Write("Weiter>");
        Console.ReadLine();
      }
    }
    public MyTcpSocketStream Open()
    {
      return myStream;
    }
    public void Close()
    {
      try {
        myClient.Close();
        Console.WriteLine("Socket geschlossen");
```

```
      }
      catch(Exception e) {
        Console.WriteLine"Exception in Close: " + e.Message);
        throw;
      }
    }
  }
}
```

Das Hauptprogramm des Clients erzeugt sich einen TCP-Socket (hier wird auch gleich die Verbindung zum Server aufgebaut, was bedeutet, dass der Server vorher gestartet werden muss), holt sich den Socket-Stream dazu und überträgt dann 20 mal die gleiche Nachricht. In einer weiteren Nachricht wird der String „Ende“ gesendet. Die Nutzung eines *BinaryFormatters* wäre hier nicht unbedingt nötig, da nur Strings übertragen werden. Er wird nur zu Demonstration eingesetzt.

```
using System;
using System.Threading;
using TCPSocketBase;
namespace TestClientSocket
{
  class TcpClientTest
  {
    static void Main(string[] args)
    {
      MyTcpSocketClient myClient = null;
      MyTcpSocketStream myStream = null;
      try {
        myClient = new MyTcpSocketClient("localhost", 9000);
        myStream = myClient.Open();
      }
      catch (Exception e) {
        Console.WriteLine("Exception: " + e.Message);
        Environment.Exit(1);
      }
      try {
        string sendObject;
        for (int i = 0; i < 20; i++)
        {
          sendObject = "Message " + (i+1);
          myStream.SendObject(sendObject);
          Thread.Sleep(100);
          Console.WriteLine(i+1 + ". Nachricht gesendet");
        }
```

```
        // Ende-Nachricht senden
        sendObject = "Ende";
        myStream.SendObject(sendObject);
        Console.WriteLine("Ende-Nachricht gesendet");
      }
      catch (Exception e) {
        Console.WriteLine("Exception: " + e.Message);
      }
      finally {
        myClient.Close();
        Console.WriteLine("Socket geschlossen");
        Console.Write("Ende>");
        Console.ReadLine();
      }
    }
  }
}
```

Schließlich soll noch auf die UDP-Kommunikation über C#-Sockets eingegangen werden. Hierzu dient die Klasse *UdpClient*, mit der sowohl Server als auch Clients programmiert werden können. Die Klasse stellt Methoden zum Senden und Empfangen von UDP-Datagrammen im synchronen Modus (Blockierung) bereit. Da UDP ein verbindungsloses Übertragungsprotokoll ist, muss vor dem Senden und Empfangen von Daten keine Verbindung aufgebaut werden. Die Kommunikation stellt sich also recht einfach dar. Zu beachten ist allerdings, dass bei UDP Nachrichten verloren gehen können.

Beispielprogramm C#-UDP-Sockets: Das folgende Beispielprogramm ist eine einfache Client-Server-Anwendung auf UDP-Basis. Beide Partner erzeugen zunächst eine Instanz der Objektklasse *UdpClient*. Der Server gibt dabei auch seine bekannte Portnummer (11000) mit[9].

Der Client sendet 10 UDP-Datagramme mit einer einfachen Nachricht und beendet sich dann selbst. Der Server geht in eine Endlosschleife und kann von beliebigen Clients, die Nachrichten an den gebundenen Port 11000 senden, Nachrichten empfangen. Der Server gibt bei Aufruf der Methode *Receive* eine Referenz auf ein Objekt vom Typ *IPEndPoint* an (*udpServer.Receive(ref RemoteIpEndPoint).* Über die angegebene Referenz kann er sich die IP-Adresse und den Port des Absenders ermitteln. Die Kodierung der Zeichen erfolgt in UTF-8-Zeichenkodierung.[10]

[9] Es gibt verschiedene Konstruktoren in der Klasse *UdpClient*.

[10] Die UTF-8-Zeichenkodierung ist in ISO 10646-1 und im RFC 3629 standardisiert.

Client:

```
using System;
using System.Net.Sockets;
using System.Threading;
using System.Text;
namespace UDPTestClient
{
  class UDPTestClient1
  {
    static void Main(string[] args)
    {
      UDPTestClient1 client = new UDPTestClient1();
      Console.WriteLine
      ("UPTestClient gestartet, Log in Datei udpclientlog.log");
      client.Communicate();
    }
    public void Communicate()
    {
      UdpClient udpClient = new UdpClient();
      Byte[] sendBytes =
       Encoding.UTF8.GetBytes("Ist da jemand?");
      string sentData = Encoding.UTF8.GetString(sendBytes);
      for (int i = 0; i < 10; i++) {
        try {
          // Senden an den Port 11000, wo der Server wartet!
          udpClient.Send((Byte[])sendBytes, sendBytes.Length,
            "localhost", 11000);
          Console.WriteLine(i+1 + ". " + "Nachricht der Länge "
            + sendBytes.Length + " gesendet: " + sentData);
        }
        catch ( Exception e ) {
          Console.WriteLine("Exception: " + e.Message);
        }
        finally {
          Thread.Sleep(1000);
        }
      }
      udpClient.Close();
      Console.WriteLine("UDPTestClient beendet");
    }
  }
}
```

Server:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Text;
namespace UDPTestServer
{
  class UdpTestServer1
  {
    static void Main(string[] args)
    {
      UdpTestServer1 server = new UdpTestServer1();
      Console.WriteLine
      ("UPPTestServer gestartet, Log in Datei udpserverlog.log");
      server.Communicate();
    }
    public void Communicate()
    {
      // "well-known" Port 11000, jeder Partner ist ein UdpClient!
      UdpClient udpServer = new UdpClient(11000);
      IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
      bool finished = false;
      while (! finished) {
        try {
          Byte[] receiveBytes =
            udpServer.Receive(ref RemoteIpEndPoint);
          // Nachricht umwandeln in String
          string receiveData =
            Encoding.UTF8.GetString(receiveBytes);
          Console.WriteLine
            ("Nachricht der Länge " + receiveBytes.Length
               + " erhalten von " + RemoteIpEndPoint.Address.ToString()
               + " über UDP-Port " + RemoteIpEndPoint.Port.ToString()
               + ": " + receiveData);
      }
      catch ( Exception e ) {
        Console.WriteLine(e.ToString());
        Console.Write("Abbruch des Servers mit Enter bestätigen> ");
        Console.ReadLine();
        finished = true;
      }
      finally {
```

```
          Thread.Sleep(1000);
        }
      }
      udpServer.Close();
      Console.WriteLine("UDPTestServer beendet");
    }
  }
}
```

Mit speziellen Methoden der Klasse *UdpClient* kann man auch Multicast-Datagramme senden und empfangen. Über die Methode *JoinMulticastGroup* kann sich ein UDP-Client an einer Multicast-Gruppe beteiligen. Über die *DropMulticast-Group*-Methode kann man das „Abonnement" für eine Multicast-Gruppe aufheben. Das Senden und Empfangen erfolgt mit den bekannten Methoden *Send* und *Receive*.

8.4 Übungsaufgaben

1. Beschreiben Sie den Unterschied zwischen synchroner und asynchroner Kommunikation!
2. Was versteht man unter einem synchronen, entfernten Dienstaufruf im Vergleich zu einem asynchronen entfernten Dienstaufruf?
3. Was versteht man unter einem Objektstrom in Java und wozu wird der Mechanismus bei der TCP-Socket-Programmierung verwendet?
4. Kann man Java-Objektströme auch für Datagramm-Sockets verwenden? Erläutern Sie Ihre Entscheidung!
5. Warum erzeugt ein TCP-basierter Server bei einem ankommenden Verbindungsaufbauwunsch in der Regel für die neue Verbindung einen neuen Prozess bzw. Thread für die neue Verbindung?
6. Was muss in einem Client- und was in einem Serverprogramm programmiert werden, damit diese für die Kommunikation über UDP-Datagramm-Sockets vorbereitet sind?
7. Was macht die Socket-Programmierung mit TCP-Sockets in Java so einfach im Vergleich zur Nutzung der Standard-Socket-Bibliothek der Sprache C?
8. Was hat ein Java-Objektstrom mit dem TCP-Stream zu tun? Erläutern Sie Ihre Antwort kurz!
9. Erläutern Sie die Fehlersemantik Exactly-Once im Vergleich zu At-Most-Once bei auftragsorientierter Kommunikation! Wie kann man Exactly-Once realisieren (kurze Erläuterung)?
10. Warum müssen Objekte, die in Java-Programmen mit TCP-Sockets gesendet werden, das Interface Serializable implementieren und wie macht man das in Java?
11. Was ist der prinzipielle Unterschied zwischen den beiden Fehlersemantiken At-Most-Once und At-Least-Once? Welche der beiden ist einfacher zu implementieren und warum?

12. Funktioniert das Zusammenspiel zwischen einem in Java entwickelten TCP-Client und einem in C# entwickelten TCP-Server reibungslos, wenn beide zur Nachrichtenübertragung die in der jeweiligen Sprache angebotenen Serialisierungsmechanismen (Java-Objektstrom und C#-Formatter) verwenden? Begründen Sie Ihre Entscheidung!
13. Erläutern Sie die Aufgaben der C#-Klassen TcpClient und TcpListener!

9 Schlussbemerkung

Dieses Buch sollte vor allem einen Überblick über grundlegende Aspekte der Datenkommunikation und der Rechnernetze vermitteln und bei einigen wichtigen Themen exemplarisch in die Tiefe gehen. Nach einer Einführung in untere Kommunikationsschichten wurden insbesondere die Schichten 3 und 4 des TCP/IP-Referenzmodells ausführlich diskutiert. Ein kleiner Überblick über die Probleme der mobilen Kommunikation in TCP/IP-Netzen wurde ebenfalls gegeben. Weiterhin wurden grundlegende Konzepte zur Entwicklung von Kommunikationsanwendungen insbesondere auf Basis von TCP/UDP-Sockets eingeführt. Einige Anwendungsprotokolle wie DNS, HTTP, SMTP usw. rundeten die Thematik ab.

Zu dem Themenbereich *Datenkommunikation und Rechnernetze* gibt es fast täglich Neuigkeiten. Protokolle entwickeln sich weiter, neue Standards oder auch nur Ideen werden produziert und neue Systemarchitekturen und -modelle werden erfunden. Einige Themen konnten auch in dieser Auflage aus Platzgründen nicht untergebracht werden. Hierzu gehört z.B. das Thema *Netzwerksicherheit*, welches nur am Rande gestreift werden konnte, sowie der Themenbereich *Netzwerkmanagement* mit seinem Standardprotokoll *SNMP*.

In Zukunft ist zu erwarten, dass sich Themenkomplexe wie *Mobile Computing* und *Multimediale Systeme* stark weiterentwickeln und auch Einfluss auf die Datenkommunikation nehmen werden. Spannend bleibt auch die Migration der vorhandenen IPv4-Infrastruktur nach IPv6, die nun langsam Fahrt aufnimmt.

10 Lösungen zu den Übungsaufgaben

10.1 Einführung in Referenzmodelle und Protokolle

1. *Erläutern Sie das TCP/IP-Referenzmodell und vergleichen Sie es hinsichtlich seiner Schichten und Dienste mit dem ISO/OSI-Referenzmodell.*

   Das TCP/IP-Referenzmodell beschreibt nur vier Schichten im Gegensatz zum detaillierteren ISO/OSI-Referenzmodell mit sieben Schichten. Vergleicht man die Schichten der beiden Referenzmodelle, erkennt man Gemeinsamkeiten in der Definition der Schichten. So beschreibt die Anwendungsschicht des TCP/ IP-Referenzmodells alle Funktionalitäten (Verwaltung von Sitzungen, Zeichendarstellung, usw.) zum Austausch von Daten basierend auf einem Transportsystem. Im ISO/OSI-Referenzmodell entspricht dies den Schichten fünf bis sieben.

   Unterhalb der Anwendungsschicht wird im TCP/IP-Referenzmodell analog zum ISO/OSI-Refernzmodell ein Transport- und darunter eine Vermittlungsschicht beschrieben. Diese beiden Schichten sind die tragenden Schichten des TCP/IP-Referenzmodells. In der Transportschicht (z.B. TCP oder UDP) wird eine Ende-zu-Ende-Verbindung bereitgestellt. Zu den Funktionalitäten der Transportschicht gehört beispielsweise die Fluss- und Staukontrolle. Hauptaufgabe der Vermittlungsschicht (IP) ist die Weiterleitung der Datenpakete von der Quelle über die Transitsysteme (Knoten) des Netzwerks bis hin zum Zielsystem, wobei eine geeignete Wegewahl (Routing) durchzuführen ist.

   Die unteren beiden Schichten des ISO/OSI-Referenzmodells werden beim TCP/IP-Referenzmodell zur Netzzugangsschicht zusammengefasst. Das TCP/ IP-Referenzmodell legt sich hier nicht fest. Möglich sind daher unterschiedliche Lösungen zur Anbindung an ein Netzwerk.

2. *Nennen Sie Mechanismen zur Fehlerbehandlung in Protokollen.*

   Bei der Übertragung von Nachrichten zwischen Kommunikationspartnern kann es zu Fehlern und somit Verfälschungen der Daten kommen. Die Protokolle der Datenkommunikation versuchen mit verschiedenen Mechanismen, diese Fehler zu erkennen und zu behandeln:

   - Die Übertragung von redundanten Informationen ermöglicht es, verfälschte Information zu erkennen oder sogar zu korrigieren (z.B. Prüfsummen).
   - Durch die Nummerierung der Nachrichten durch Sequenznummern kann der Empfänger die Reihenfolge der Nachrichten überprüfen. Der Empfänger kann so feststellen, ob eine Nachricht mehrmals empfangen wurde oder ob ein Teil einer Nachricht nicht angekommen ist.

- Auf Basis der Sequenznummern können so genannte Quittierungen zur Bestätigung empfangener Nachrichten versendet werden. Eine solche Bestätigung wird vom Empfänger der Daten an die Quelle gesendet. Es existieren sehr unterschiedliche Quittierungsverfahren wie zum Beispiel das kumulative Bestätigen von mehreren Datenpaketen oder das Versenden von negativen Quittierungen zur Meldung eines erkannten Fehlers.
- Durch Zeitüberwachungsmechanismen können Fehler erkannt werden. So kann beispielsweise nach dem Versenden einer Nachricht eine festgelegte Zeitspanne auf eine Quittierung gewartet werden. Trifft innerhalb dieses Zeitraums keine Empfangsbestätigung ein, wird von einem Übertragungsfehler ausgegangen (z.B. „implicit not acknowledge“ bei TCP). Bei einem Zeitüberwachungsmechanismus wird immer eine festgelegte Zeit lang auf ein bestimmtes Ereignis gewartet. Tritt das Ereignis vor Ablauf der Zeitspanne (timeout) nicht auf, wird mit einer definierten Aktion reagiert.

3. *Was ist eine A-PDU?*

   Der Begriff A-PDU entstammt dem ISO/OSI-Referenzmodell. Das Referenzmodell unterteilt die Komplexität eines Datenkommunikationssystems in sieben Schichten. Die von einer Schicht entgegengenommenen Daten werden als Nutzdaten oder als SDU (Service Data Unit) bezeichnet. Aus der SDU und den jeweiligen Kontrollinformationen einer Schicht, die als PCI (Protocol Control Information) bezeichnet werden, wird die PDU (Protocol Data Unit) gebildet. Als A-PDU wird eine PDU der Anwendungsschicht bezeichnet.

4. *Was ist eine Instanz im ISO/OSI-Referenzmodell?*

   Das ISO/OSI-Referenzmodell beschreibt sieben Schichten. Eine konkrete Implementierung einer Schicht wird als Instanz bezeichnet. Zu einer Schicht kann es verschiedene Implementierungen geben.

10.2 Technische Grundlagen von Rechnernetzen

1. *Unterscheiden Sie die Begriffe Transferzeit, Laufzeit und Übertragungszeit..*

   Die Übertragungszeit bestimmt die Zeit, die der Sender benötigt, um die zu sendenden Daten an das Medium zu übergeben. Neben der Bitlänge der Daten ist die Übertragungszeit abhängig von der Übertragungsgeschwindigkeit $v_ü$ (in Bit/s) die angibt, wie schnell die Daten an das Medium übergeben werden.

   Mit der Laufzeit wird die Zeit beschrieben, die das Signal zum Zurücklegen der Strecke vom Sender zum Empfänger benötigt. Die Laufzeit ist abhängig von der Entfernung d (in m) und der Ausbreitungsgeschwindigkeit v_d (in m/s).

   Die Transferzeit wird durch das Summieren der Übertragungszeit und der Laufzeit ermittelt. Sie beschreibt somit die Zeit vom Übertragen des ersten Bits beim Sender bis zur Ankunft des letzen Bits beim Empfänger.

2. *Was beschreibt die Schrittgeschwindigkeit und welche Einheit wird verwendet?*

   Mit der Schrittgeschwindigkeit wird angegeben, wie oft pro Zeiteinheit ein Signal seinen Zustand ändern kann. Die Schrittgeschwindigkeit (auch Taktfrequenz) wird in baud (1/s) angegeben.

3. *Was versteht man bei der Verkabelung unter S/STP?*

   Die zur Verkabelung oft verwendeten Twisted-Pair-Kabel (z.B. Telefonkabel) werden in verschiedenen Bauformen angeboten, die sich hauptsächlich durch die Qualität der Abschirmung unterscheiden. Ein Screened/Shielded Twisted Pair-Kabel (S/STP) bietet die beste Abschirmung, da die einzelnen Adernpaare durch einen Adernschirm (Shield) und das gesamte Kabel durch einen Kabelschirm (Screen) gegen Störungen geschützt sind.

4. *Erklären Sie die Grundprinzipien der strukturierten Verkabelung.*

   Die strukturierte Verkabelung unterteilt die Verkabelung eines Gebäudes in drei Bereiche. Der Primärbereich ist der gebäudeübergreifende Bereich, der möglichst redundant auf Basis von Lichtwellenleitern verkabelt wird. Im Sekundärbereich installiert man ein Gebäude-internes Backbone oft auf Basis von Kupferkabeln oder Lichtwellenleitern. Den Tertiärbereich bildet eine sternförmige Verkabelung auf den einzelnen Etagen. Die Endgeräte werden mit Etagenverteilern verbunden.

5. *Erläutern Sie den Unterschied zwischen p-persistent und 1-persistent CSMA?*

   Wird das gemeinsame Medium als frei erkannt, wird beim p-persistent CSMA mit einer Wahrscheinlichkeit von p sofort begonnen zu senden. Mit der Wahrscheinlichkeit von 1-p wird vor dem Senden eine bestimmte Zeit gewartet. Das 1-persistent CSMA ist eine Spezialvariante, bei der sofort gesendet wird, wenn das Medium als frei erkannt wird. Die Wahrscheinlichkeit p, mit der sofort gesendet wird, ist bei dieser Spezialvariante auf 1 gesetzt, was den Namen erklärt.

6. *Was versteht man unter gesteuerten Buszugriffsverfahren und welche Varianten gibt es hierfür?*

   Bei einem gesteuerten Buszugriffsverfahren erfolgt die Zuteilung des Mediums nach einem klar definierten Steuerungsmechanismus. Bei einem zentralen Verfahren gibt es einen Masterknoten, der die Zugriffssteuerung übernimmt. Erfolgt die Zugriffssteuerung durch alle Knoten, spricht man von einem dezentralen Verfahren.

7. *Was versteht man unter ungesteuerten Buszugriffsverfahren und welche Varianten gibt es hierfür?*

   Basiert das Buszugriffsverfahren nicht auf einem definierten Steuerungsmechanismus (z.B. Wettkampfverfahren) spricht man von einem ungesteuerten Verfahren. Kann jeder Knoten zu jeder Zeit auf das Medium zugreifen, kann es zu Kollisionen kommen (nicht kollisionsfreies Verfahren). Wird dies verhindert, handelt es sich um ein kollisionsfreies Verfahren.

8. *Wozu benötigt man in der Schicht 2 ein XON/XOFF-Protokoll?*

   Das XON/XOFF-Protokoll dient der Flusssteuerung in der Schicht 2. Ist der Puffer beim Empfänger fast voll, wird dem Sender mit dem XOFF-Zeichen (ASCII) das weitere Senden verboten. Erst nach dem Empfang des XON-Zeichens darf der Sender mit den Senden beginnen.

10.3 Ausgewählte Technologien und Protokolle unterer Schichten

1. *Welche HDLC-Betriebsmodi kennen Sie? Erläutern Sie einen davon.*

   Das HDLC-Protokoll ermöglicht durch die Kombination der Link-Konfigurationen mit dem Einsatz der vorhandenen Stationstypen drei Betriebsmodi:

   – NRM: Normal Mode
   – ARM: Asynchronous Response Mode
   – ABM: Asynchronous Balanced Mode

   Wird das HDLC-Protokoll im Normal Mode betrieben, wird die Kommunikation von einer primären Station gesteuert. Die sekundären Stationen können nur auf Anfragen der primären Station antworten, nicht jedoch selbstständig die Kommunikation beginnen.

2. *Skizzieren Sie einen Ablauf einer HDLC-Kommunikation zwischen zwei Stationen im ABM-Mode*

   Eine Kommunikation im ABM-Mode kann von einer kombinierten Station durch das Senden eines entsprechenden Überwachungsrahmens (S, SAMB) initiiert werden. Der Verbindungswunsch muss durch den Kommunikationspartner mit einem unnummerierten Rahmen (UA) bestätigt werden. Nach dem erfolgreichen Verbindungsaufbau können beide Stationen Daten mit Informationsrahmen übertragen. Empfangene Informationsrahmen werden mit gesonderten Steuerrahmen oder im Huckepackverfahren mit Informationsrahmen bestätigt. Um eine bestehende Verbindung abzubauen, muss eine Station dies mit einem unnummerierten Rahmen ankündigen. Auch der Verbindungsabbau muss betätigt werden.

3. *Was bedeutet LAN-Switching?*

   Im Gegensatz zu einem Hub besitzt ein Switch (auch LAN-Switch genannt) keine gemeinsame Anschlussleitung. Zwischen Stationen, die über einen Switch zusammengeschlossen sind, besteht somit keine dauerhafte physikalische Verbindung. Nur für die Dauer der Kommunikation werden die Kommunikationspartner zusammengeschaltet.

4. *Wozu dient das Protokoll PPP?*

   Im Internet wird das Point-to-Point-Protocol (PPP) als Standardprotokoll für die Punkt-zu-Punkt-Verbindung zwischen dem Dienstnutzer und dem Internet-Provider eingesetzt. Das PPP wird der Schicht zwei nach dem ISO/OSI-Referenz-

modell zugeordnet. Mit dem PPP kann eine Vollduplex-Verbindung zwischen genau zwei Knoten aufgebaut werden. In der Regel werden über PPP Verbindungsparameter (z.B. IP-Adresse) der darüberliegenden Schicht ausgehandelt.

5. *Können bei Ethernet, speziell bei 100Base-T, Kollisionen auftreten und was macht man in heutigen Ethernet-Netzwerken gegen Kollisionen?*

   Ethernet ist eine LAN-Technologie, die auf der Bus-Topologie aufbaut. Da sich die Stationen den Bus als gemeinsames Medium teilen müssen, sind Kollisionen in der Regel möglich. Bei 100Base-T wird die Kopplung der Stationen über einen Switch, im Gegensatz zu einem Hub, realisiert. Kollisionen sind dann nicht mehr möglich.

6. *Was ist im WLAN nach IEEE 802.11 ein Infrastruktur-Modus?*

   Die WLAN-Technologie ermöglicht zwei Betriebsarten, die sich in der Organisation des Netzwerkes unterscheiden. Beim Infrastruktur-Modus gibt es eine zentrale Station (Access-Point), über die alle beteiligten Stationen ihre Kommunikation abwickeln.

7. *Was bedeutet ADSL im Vergleich zu SDSL?*

   ADSL steht für Asymmetrical DSL und bietet unterschiedliche Bitraten für den Down- und Upstream. Beim Symmetrical DSL (SDSL) werden für Down- und Upstream gleiche Bitraten erzielt.

10.4 Konzepte und Protokolle der Vermittlungsschicht

1. *Nennen Sie zwei Aufgaben der Vermittlungsschicht und beschreiben Sie diese kurz!*

   Die wesentliche Aufgabe der Vermittlungsschicht ist die Wegewahl, auch Routing genannt. Eine weitere Aufgabe ist die Fragmentierung und Defragmentierung von Datenpaketen.

   - Wegewahl (Routing): Ein Kommunikationsnetzwerk besteht aus Knoten (z.B. ein Rechner oder ein Router), die über Teilstrecken verbunden sind. Besitzt ein Knoten mehr als zwei Verbindungen zu anderen Knoten, muss beim Weiterleiten eines Datenpaketes eine Entscheidung getroffen werden. In der Vermittlungsschicht wird im Rahmen der Wegwahl versucht eine optimale Route zu berechnen.
   - Fragmentierung/Defragmentierung: Für die Übertragung eines Datenpaketes kann eine Vielzahl unterschiedlicher Teilstrecken genutzt werden, die unterschiedliche Rahmenbedingungen bieten. Wichtig für die Datenübertragung ist die maximale Größe eines Datenpaketes (als MTU bezeichnet = Maximum Transmission Unit). Ist die MTU einer Teilstrecke kleiner als die aktuelle Paketgröße, muss der weiterleitende Knoten das Datenpaket entsprechend der MTU-Größe aufteilen (Fragmentierung) und die Bestandteile einzeln weiterleiten.

2. *Was unterscheidet die Vermittlungsverfahren „Leitungsvermittlung“ und „Paketvermittlung“?*

   Die Leitungsvermittlung reserviert beim Verbindungsaufbau alle benötigten Ressourcen wie beispielsweise die Bandbreite und Pufferspeicher über die gesamte Übertragungsstrecke hinweg. Für die Verbindung kann somit eine feste Bandbreite garantiert werden.

   Bei der Paketvermittlung werden zwischen dem Sender und Empfänger keine Ressourcen für die Übertragung reserviert. Die Nachricht wird immer als Ganzes von einem Knoten zum nächsten übertragen. Speziell bei der Paketvermittlung wird die Nachricht hierzu in kleine Datenpakete aufgeteilt. Da die Wegewahl in den Knoten für jedes Paket gesondert entschieden wird, kann die Bandbreite bei der Paketvermittlung schwanken.

3. *Warum werden virtuelle Verbindungen in der Schicht 3 auch scheinbare Verbindungen genannt?*

   Man spricht von virtuellen Verbindungen (engl. Virtual Circuits), wenn bei einer Nachrichtenübermittlung beim Verbindungsaufbau eine Route festgelegt und Ressourcen reserviert werden. Da im Gegensatz zur Leitungsvermittlung jedoch keine physikalische Leitung durchgeschaltet wird, spricht man von scheinbaren Verbindungen.

4. *Erläutern Sie den Unterschied zwischen statischen und dynamischen Routing-Mechanismen! Nennen Sie je ein konkretes Verfahren hierzu!*

   Bei einem statischen Routing-Mechanismus (z.B. Flooding) werden die optimalen Routen einmal (z.B. durch manuelle Konfiguration) ermittelt und diese dann unverändert verwendet. Ein dynamischer Routing-Mechanismus (z.B. Link-State-Verfahren) besitzt einen Algorithmus, mit dem die optimalen Routen während des Betriebs aktualisiert werden. Eine wiederkehrende Neuberechnung der optimalen Routen ermöglicht eine Reaktion auf Veränderungen im Netzwerk wie beispielsweise den Ausfall einer Teilstrecke.

5. *Was versteht man unter einem zentralen Routing-Verfahren? Handelt es sich hier um ein statisches oder um ein adaptives Routing-Verfahren?*

   Bei einem zentralen Routing-Verfahren werden die Routing-Informationen von einem zentralen Knoten (Routing-Kontroll-Zentrum) ermittelt und an die einzelnen Knoten übertragen. Das Routing der einzelnen Knoten basiert somit ausschließlich auf den Routing-Informationen des zentralen Knotens.

   Zentrale Routing-Verfahren werden der Gruppe der adaptiven Routing-Verfahren zugeordnet, da die laufend neu berechneten Routing-Informationen vom zentralen Knoten an die einzelnen Knoten verteilt werden.

6. *Welche Vorteile bringt ein hierarchisches Routing-Verfahren?*

   Ein hierarchisches Routing-Verfahren reduziert den Aufwand beim Ermitteln der optimalen Route, da weniger Informationen verarbeitet werden müssen. Knoten, die über einen gemeinsamen Pfad erreicht werde können, werden hierfür zu Gruppen (Region) zusammengefasst. In den einzelnen Knoten muss daher nur noch die Routing-Information zu den Regionen und nicht für jeden einzelnen Knoten verwaltet werden.

7. *Erläutern Sie kurz das Optimierungsprinzip beim Routing!*

   Liegt ein Knoten B auf der optimalen Route zwischen den Knoten A und C, besagt das Optimierungsprinzip, dass auch der optimale Pfad von B nach C auf dieser Route liegt. Die optimalen Routen von einer Menge an Quellknoten zu einem Zielknoten können daher mit einem Baum (Graphentheorie) den sogenannten „Sink Tree“ dargestellt werden.

8. *Was versteht man im Distance-Vector-Routing-Verfahren unter dem Count-to-Infinity-Problem und wie verhält sich das Verfahren im Hinblick auf Konvergenz? Begründen Sie Ihre Entscheidung!*

   Bei Distance-Vector-Routing-Verfahren (DVR-Verfahren) tauschen die Knoten ihre Routing-Informationen nur mit den direkten Nachbarn aus. Fällt beispielsweise der Knoten A (oder die Teilstrecke zu diesem Knoten) aus, können dies nur die direkten Nachbarn erkennen. Die anderen Knoten im Netzwerk propagieren jedoch ungeachtet des Ausfalls ihre Routing-Information zum Knoten A an ihre direkten Nachbarn. Die vermeintliche „Entfernung“ zum ausgefallen Knoten wird so immer weiter inkrementiert bis ein Schwellwert erreicht wird. Dieses Verhalten wird als „Count-to-Infinity-Problem“ bezeichnet. Die Konvergenz bei schlechten Nachrichten ist somit schlecht, da es lange dauert, bis der Schwellwert erreicht ist und somit der Knoten als unerreichbar erkannt wird.

9. *Welche Art von Routing-Verfahren ist das Distance-Vector-Verfahren und wann wird es in IP-Netzen auch heute noch eingesetzt?*

   Das Distance-Vector-Verfahren ist ein verteiltes und adaptives (dynamisches) Verfahren, welches heute vor allem in kleinen autonomen Systemen eingesetzt wird.

10. *Nennen Sie drei mögliche Metriken, die ein Routing-Verfahren zur Ermittlung der optimalen Routen nutzen kann!*

    Metriken können sein:

    - Hop Count (Anzahl der Knoten bis zum Ziel)
    - Durchschnittliche Übertragungszeit
    - Kosten für die Leitungsnutzung

11. *Wie sieht ein einzelner Router im Link-State-Routing-Verfahren die aktuelle Netzwerktopologie?*

    Beim Link-State-Routing-Verfahren besitzt jeder Router die Kosteninformationen aller Teilstrecken in einem Netzwerk. Jeder einzelne Router kennt somit die vollständige Topologie und nicht nur seine direkte Nachbarschaft.

12. *Sind im Link-State-Routing-Verfahren Schleifen möglich? Begründen Sie Ihre Entscheidung!*

    Da die einzelnen Router im Link-State-Routing-Verfahren die gesamte Topologie kennen, können Schleifen erkannt und so vermieden werden.

13. *Was versteht man unter einem Leaky-Bucket-Verfahren und wozu dient es?*

    Das Leaky-Bucket-Verfahren versucht beim Sender die Menge der Nachrichten pro Zeiteinheit möglichst konstant zu halten. Analog zu einem „rinnenden Eimer" werden die Nachrichten vom Sender zwischengespeichert und dann in einer konstanten Folge weitergeleitet. Durch dieses Verfahren werden Belastungsspitzen des Netzwerks vermieden und somit wird Stausituationen vorgebeugt.

14. *Was ist ein autonomes System im Internet und welche Arten von autonomen Systemen kennen Sie?*

    Ein autonomes System (AS) ist ein eigenständiges Teilnetz des Internets, das von einer Organisation verwaltet wird. Neben den AS von Unternehmen, Institutionen (z.B. Universitäten) und Providern (ISP) gibt es Transit-AS die als Transportnetze arbeiten und die unterschiedlichen autonomen Systeme verbinden.

15. *Was ist im globalen Internet ein Transit-AS?*

    Die Aufgabe eines Transit-AS ist die Verbindung der autonomen Systeme untereinander. Da nur mit den Transportnetzen der Transit-AS von einem AS zum anderen kommuniziert werden kann, stellen diese das Rückrat (backbone) des Internets dar.

16. *Beschreiben Sie kurz den Dienst, den IP für die darüberliegenden Schichten zur Verfügung stellt im Hinblick auf die Übertragungssicherheit!*

    IP bietet eine nicht zuverlässige und verbindungslose Nachrichtenübertragung als Dienst an. Die Übertragung wird als nicht zuverlässig eingestuft, da weder eine Empfangskontrolle noch eine Überprüfung der Nachrichtenintegrität vorgenommen wird. IP ist ein verbindungsloses Protokoll, weil zwischen Sender und Empfänger keine Steuerung der Kommunikation vorgenommen wird.

17. *Welches Vermittlungsverfahren verwendet die Internet-Schicht?*

    Die Internet-Schicht nutzt die Paketvermittlung als Vermittlungsverfahren.

18. *Was versteht man unter einem Sink Tree im Sinne der Wegewahl?*

    Ein Sink Tree stellt die optimalen Routen von allen Knoten in einem Netzwerk zu einem Ziel (Senke) als einen Baum (Graphentheorie) dar.

19. *Was versteht man unter NAT (Network Address Translation) und welche Vorteile bietet das Verfahren?*

    Das NAT-Verfahren wird in der Regel an der Grenze zwischen einem privaten und dem öffentlichen Netzwerk eingesetzt. Wird aus dem privaten Netzwerk ein IP-Paket in das öffentliche Netzwerk gesendet, wird die Quelladresse durch das NAT-Verfahren ersetzt. Durch das Ersetzen der eigentlichen Quelladressen durch meist nur eine öffentliche Adresse werden öffentliche IP-Adressen eingespart. Darüber hinaus wird die eigentliche Topologie des privaten Netzwerks verdeckt und so ein Angriff erschwert.

20. *Welche Aufgabe verrichtet ein NAT-Router im Rahmen der Adressierung für ankommende und abgehende IP-Pakete?*

    Bei abgehenden IP-Paketen muss der NAT-Router die Quelladresse ersetzen und sich diese Adressumsetzung merken. Bei eingehenden IP-Paketen wertet der NAT-Router die Zieladresse aus und ersetzt diese durch die entsprechende Zieladresse des privaten Netzwerks.

21. *Warum muss ein NAT-Router vor dem Weiterleiten eines vom globalen Internet ankommenden oder vom Intranet abgehenden IP-Paketes die Checksumme im IP-Header jedesmal neu berechnen?*

    Der NAT-Router ändert die Quell- oder Zieladresse der IP-Pakete. Da die Checksumme auch über die Quell- und Zieladresse berechnet wird, muss diese Berechnung bei einer Änderung erneut durchgeführt werden. Der Empfänger des Paketes würde ansonsten das Paket als fehlerhaft verwerfen, da die berechnete Checksumme nicht mit der angegebenen übereinstimmt.

22. *Nennen Sie den Unterschied zwischen der klassischen Subnetz-Adressierung und dem Classless Interdomain Routing (CIDR)! Welche Vorteile bringt CIDR für die Adressenknappheit im Internet (Begründung)?*

    Eine klassische IP-Adresse unterteilt sich in vier Gruppen zu je einem Byte. Je nach gewählter Adressklasse stehen entweder ein, zwei oder drei Byte für die Adressierung der Knoten eines Netzwerks zur Verfügung. Beim CIDR/VLSM wird die starre Aufteilung in Adressklassen aufgehoben. Die Anzahl der Bits für den Host-Anteil kann somit so gewählt werden, dass die für eine Organisation benötigte Anzahl von öffentlichen Adressen gerade erreicht wird. Es werden somit nicht so viele öffentliche IP-Adressen verschwendet.

23. *Wozu wird in IPv4-Netzen im Router das Wissen über eine Netzwerkmaske für jedes angeschlossene Netz benötigt?*

    Für die Wegewahl im Router ist die Netzwerknummer der angeschlossenen Netzwerke interessant. Um aus der vollständigen Adresse eines Netzwerkes den Netzteil zu ermitteln, reicht es aus, eine logische UND-Verknüpfung der Adresse mit der Netzwerkmaske durchzuführen.

24. *Was bedeutet in CIDR die Darstellung 132.10.1.8/24?*

    Mit der 24 hinter dem Slash wird angegeben, dass die ersten 24 Bit der Adresse das Netzwerk beschreiben. Die Netzwerknummer ist somit die 132.10.1.0.

25. *Wie findet ein Host innerhalb eines LANs (IPv4-Netzwerks) die MAC-Adresse eines Partner-Hosts, wenn er das erste Mal ein IP-Paket an diesen senden will?*

    Kennt der Host die IP-Adresse des Partner-Hosts, kann er mit Hilfe des ARP (Address Resolution Protocol) die MAC-Adresse zur bekannten IP-Adresse ermitteln. Über ARP wird eine Anfrage nach der MAC-Adresse an das gesamte Netzwerk gestellt. Kennt ein Knoten im Netzwerk die MAC-Adresse zur angegebenen IP-Adresse, liefert er diese als Antwort zurück.

26. *Wie findet ein Host die MAC-Adresse eines Partner-Hosts, der nicht im eigenen LAN, sondern irgendwo in einem entfernten LAN, das aber über einen Router erreichbar ist, liegt?*

    Ist der gesuchte Host nicht im gleichen Netzwerk, erkennt dies der verantwortliche IP-Router im Netzwerk anhand der angegeben IP-Adresse und leitet die Anfrage entsprechend weiter. Der IP-Router fungiert also als ARP-Proxy.

27. *Über welches Steuerprotokoll wird eine Ping-Nachricht abgesetzt? Verwendet das besagte Steuerprotokoll TCP, UDP oder direkt IP zur Nachrichtenübertragung?*

    Ping-Nachrichten werden über das ICMP (Internet Control Message Protocol) abgewickelt. ICMP wird der Vermittlungsschicht zugeordnet und nutzt direkt IP zur Nachrichtenübertragung.

28. *Erläutern Sie den Unterschied zwischen limited Broadcast und directed Broadcast in IP-Netzen! Wann benötigt man z.B. diese Broadcast-Varianten (Nennen Sie je ein Beispiel)?*

    Eine normale Broadcast-Nachricht (limited Broadcast) wird vom zuständigen IP-Router nicht in andere Netzwerke übertragen und erreicht somit nur die Knoten im gleichen Netzwerk. Die Beschränkung auf das lokale Netzwerk ist notwendig, um eine Überlastung des Internets zu vermeiden. Limited Broadcast wird beispielsweise beim Booten eines Rechners verwendet, um Anfragen an das gesamte lokale Netz zu richten (z.B.: DHCP).

    Mit einem direkten (directed) Broadcast werden alle Hosts in einem entfernten Netz angesprochen. Ein direkter Broadcast wird bis zum Router des Zielnetzes weitergeleitet, der diesen dann an alle Hosts des Zielnetzes verteilt. Das Weiterlei-

ten von direkten Broadcast-Nachrichten ist in vielen Routern deaktiviert, da es für Angriffe (z.B. Smurf-Attacken) genutzt werden kann.

29. *Was kann die Vermittlungsschicht zur Vermeidung bzw. zum Abbau von Stausituationen im Netz beitragen?*

    In der Vermittlungsschicht kann der Datenfluss so reguliert werden, dass ein möglichst konstanter Datenfluss erreicht wird. Beispielsweise können mit dem Leaky-Bucket-Verfahren Lastspitzen verhindert und somit Stausituationen vorgebeugt werden.

    Verursacht ein Knoten eine zu hohe Netzlast, kann dieser mittels eines Choke-Paketes (engl.: choke = Drossel) direkt zur Reduzierung der Sendeleistung aufgefordert werden. Reagiert der Sender auf diese Aufforderung, wird die Stausituation wieder abgebaut.

30. *Kann man innerhalb eines autonomen Systems im globalen Internet unterschiedliche Routing-Verfahren verwenden? Begründen Sie Ihre Entscheidung!*

    Innerhalb eines autonomen Systems muss ein einheitliches Routing-Verfahren verwendet werden. Beim Einsatz von verteilten Verfahren ist die Kommunikation zwischen den Routern nur möglich, wenn das gleiche Verfahren eingesetzt wird. Aber auch bei lokalen oder statischen Verfahren ist eine Abstimmung aller Router eines autonomen Systems wichtig.

31. *Warum werden in IPv4 IP-Adressen „verschenkt" und wie werden im derzeitigen globalen Internet IP-Adressen eingespart? Nennen Sie zwei Einsparvarianten!*

    IP-Adressen der Version 4 (IPv4) sind fest in Klassen aufgeteilt. Je nach Klasse stehen entweder ein, zwei oder drei Byte zur Adressierung der Hosts in einem Netzwerk bereit. Bei der Vergabe einer Netzadresse muss sich ein Unternehmen beispielsweise entscheiden, ob ein Klasse-C-Netz mit max. 254 (2^8-2) oder ein Klasse-B-Netz mit max. 65534 (2^{16}-2) benötigt wird. Will das Unternehmen 300 Host mit einer öffentlichen IP-Adresse ausstatten, muss ein Klasse-B-Netz genutzt werden. In diesem Fall werden 65234 – also der größte Teil – nicht verwendet und daher „verschenkt".

    Um diesem Problem entgegenzuwirken, wurde die feste Aufteilung in Klassen aufgehoben. Mit Hilfe von VLSM (Variable Length of Subnet Mask) und CIDR (Classless Inter-Domain Routing) ist es bei IPv4 möglich, die Länge des Host-Anteils variabel zu wählen. So kann beispielsweise ein Host-Anteil von 9 Bit (9^2 – 2 = 510) gewählt werden, um die 300 Hosts des vorherigen Beispiels zu adressieren.

    Eine weitere Möglichkeit ist die Verwendung des NAT-Verfahrens (Network Address Translation). Mit NAT kann man ein Intranet mit privaten IP-Adressen betreiben und mit einer einzigen öffentlichen IP-Adresse den Zugang zum globalen Internet ermöglichen. Dies reduziert die Anzahl der benötigten öffentlichen IP-Adressen und somit die Adressknappheit.

32. *Welche Bedeutung hat das TTL-Feld im IP-Header und wie wird es in einem Router im Rahmen der Bearbeitung eines ankommenden IP-Pakets bearbeitet?*

    Mit dem TTL-Feld (Time To Live) wird die Verweilzeit eines Pakets im Netzwerk beschränkt. Somit wird verhindert, dass ein Paket unendlich lange im Netz zirkuliert. Der Sender setzt den TTL-Wert auf einen festen Start-Wert (z.B. 128). Die meisten Router verringern diesen Wert beim Weiterleiten einfach um 1. Das TTL-Feld entspricht somit mehr einem Hop-Zähler. Erreicht das TTL-Feld den Wert 0 wird es von den Routern nicht weitergeleitet und verworfen. Das Routing in IP-Netzen terminiert somit entweder bei Erreichen des Ziels oder nach einer festen Anzahl von Hops.

33. *Wozu benötigt eine IP-Instanz das Feld Protokoll aus dem IP-Header?*

    Es zeigt dem Empfänger an, mit welchem Protokoll das Paket weiter verarbeitet werden soll. Ist im Protokoll-Feld beispielsweise eine 6 angegeben, muss die IP-Instanz das Paket an eine TCP-Instanz übergeben.

34. *Beschreiben Sie kurz den Protokollmechanismus der Fragmentierung am Beispiel von IPv4 und gehen Sie dabei auf die genutzten Felder Identifikation, Fragment Offset und Flags ein!*

    Mit der Fragmentierung wird die Größe eines Paketes an die MTU (Maximum Transmission Unit) einer Teilstrecke angepasst. Ermöglicht eine Teilstrecke nur die Übertragung von maximal 512 Byte großen Datenpaketen, muss z.B. ein Paket mit 2000 Byte in vier Teilstücke aufgeteilt werden.

    IPv4 teilt bei der Fragmentierung die Nutzdaten in entsprechende Teilstücke auf und erzeugt für jedes Teilstück einen neuen IP-Header. Das Identifikation-Feld wird bei allen Teilstücken mit der gleichen Identifikationsnummer beschrieben. Der Empfänger kann so alle Teilstücke einem ursprünglichen Paket zuordnen. Mit dem Fragment Offset wird die relative Position zum Anfang der ursprünglichen Nutzdaten angegeben. Der Abstand wird beim Fragment Offset in 8-Byte-Schritten angegeben. Mit dem MF-Flag wird gekennzeichnet, ob noch weitere Fragmente folgen oder ob dieses Fragment das letzte ist. Ist das MF-Flag auf 0 gesetzt, wurde das letzte Fragment empfangen. Steht es auf 1, ist die Übertragung noch nicht abgeschlossen. Das DF-Flag macht schließlich eine Aussage darüber, ob ein Paket grundsätzlich zerlegt werden darf oder nicht.

35. *Welches Problem im Routing-Protokoll RIP versucht die Split-Horizon-Technik zu lösen und wie funktioniert diese Technik?*

    Bei der Verbreitung von „schlechten“ Nachrichten (z.B.: Ausfall eines Knotens) besitzt RIP eine langsame Konvergenz. Aufgrund des Count-To-Infinity-Problems benötigt es eine lange Zeit, bis alle Knoten im Netz den Ausfall erkannt haben.

    Die Split-Horizon-Technik versucht, dieses Problem zu lösen, indem es Informationen nicht zu dem Router zurücksendet, von dem die Route auch empfangen

wurde. Mit dieser Technik kann in einfachen Topologien das Count-To-Infinity-Problem beherrscht werden.

36. *Im globalen Internet setzt man prinzipiell zwei verschiedene Routing-Verfahren ein (EGP und IGP). Erläutern Sie den Unterschied zwischen EGP und IGP und stellen Sie dar, wo beide Routing-Verfahren Verwendung finden? Nennen Sie je ein konkretes Routing-Protokoll für die beiden Verfahren!*

    Innerhalb eines autonomen Systems werden Interior-Gateway-Protokolle (IGP) verwendet. Für die Kommunikation zwischen autonomen Systemen werden Exterior-Gateway-Protokolle (EGP) eingesetzt. Ein IGP wie zum Beispiel das RIP konzentriert sich stark auf die Berechnung der schnellsten Routen innerhalb des Autonomen Systems. Bei einem EGP (z.B.: BGP) müssen bei der Ermittlung der Routen zusätzliche Aspekte wie beispielsweise Sicherheit oder Kosten berücksichtigt werden.

37. *Nennen Sie drei Ziele der IPv6-Entwicklung!*

    Folgende Ziele können u.a. genannt werden:

    - Umfassende und langfristige Lösung der Adressproblematik (Adressknappheit).
    - Flussmarken sollen unterstützt werden.
    - Unterstützung mobiler Kommunikation.

38. *Welchen Sinn haben im IPv6-Protokoll die sog. Erweiterungsheader? Nennen Sie zwei Erweiterungsheader und beschreiben Sie kurz deren Aufgabe!*

    Mit den sog. Erweiterungsheadern können zusätzliche Informationen zwischen den IP-Instanzen ausgetauscht werden. Da ein IPv6-Paket keinen oder mehrere Erweiterungsheader besitzen kann, ist dies eine sehr flexible Möglichkeit, spezielle Funktionalitäten zu unterstützen, den Protokoll-Overhead im Standardfall jedoch gering zu halten. Zwei Beispiele für Erweiterungsheader:

    - Fragment-Header: Ermöglicht die aus IPv4 bekannte Fragmentierung von IP-Paketen.
    - Routing-Header: Enthält eine Liste von Routern, die auf dem Weg zum Ziel angesteuert werden müssen.

39. *Wozu sollen im IPv6-Protokoll Flussmarken dienen?*

    Mit Flussmarken können qualitative Anforderungen wie beispielsweise die benötigte Bandbreite oder die maximale Verzögerung einer IP-Verbindung zwischen einer Quelle und einem Ziel definiert werden. Router können diese Informationen nutzen und entsprechend Ressourcen reservieren, um die benötigte Qualität einer Anwendung sicherstellen zu können.

40. *Host A sendet in einem IPv4-Netzwerk seinem Partnerhost B ein IP-Paket der Länge 5000 Byte. 20 Byte davon enthält der IP-Header des Pakets (minimaler IP-Header ohne Optionen). Es gelten folgende Bedingungen:*
    - *Das IP-Paket muss von A nach B drei IP-Netze durchlaufen, Host A liegt im Netz 1, Netz 2 ist ein Transitnetz und Host B liegt in Netz 3*
    - *Netz 1 und Netz 2 werden durch Router R1 verbunden*
    - *Netz 2 und Netz 3 werden durch Router R2 verbunden*
    - *Netz 1 hat eine MTU von 2048 Byte*
    - *Netz 2 hat eine MTU von 1024 Byte*
    - *Netz 3 hat eine MTU von 576 Byte*
    - *Skizzieren Sie die gesamte Netzwerktopologie!*

    - *Wie viele IP-Fragmente verlassen R1 für das besagte IP-Paket in Richtung Netzwerk mit der Nummer 2?*

      Da die MTU des Netzes 1 mit 2048 Byte schon zu klein für das IP-Paket mit einer Gesamtlänge von 5000 Byte ist, muss zunächst im Host A eine IP-Fragmentierung vorgenommen werden. Für die maximale Nutzlast ergeben sich 2024 Byte. Die Länge der Nutzlast muss entsprechend der Fragment-Offset-Angabe ein Vielfaches von 8 Byte sein. Mit dem IP-Header von 20 Byte ergibt sich eine Gesamtlänge von 2044 Byte. Es müssen somit 3 IP-Fragmente erstellt werden. Die ersten beiden IP- Fragmente besitzen jeweils eine Nutzdatenlänge von 2024 Byte, während das letzte Fragment eine Nutzdatenlänge von 932 Byte aufweist.

      Der Router R1 versucht die drei ankommenden IP-Fragmente wie gewöhnlich als IP-Pakete an den nächsten Knoten zu übertragen. Die MTU von Netz 2 beträgt 1024 Byte. Im Router R1 wird keine Defragmentierung des ursprünglichen Paketes unternommen. Für die maximale Nutzlast ergeben sich 1000 Byte (abzüglich IP-Header und Vielfaches von 8 Byte). Zwei ankommende IP-Fragmente mit einer Nutzdatenlänge von 2024 Byte müssen somit jeweils nochmals in drei IP-Fragmente aufgeteilt werden (1000 + 1000 + 24). Das Fragment mit einer Nutzdatenlänge von 932 Byte muss nicht weiter aufgeteilt werden.

      Den Router R1 verlassen somit 7 IP-Fragmente in Richtung Netzwerk mit der Nummer 2.

- *Wie viele IP-Fragmente verlassen R2 für das besagte IP-Paket in Richtung Netzwerk mit der Nummer 3?*

  Berechnung wie bei b)

  MTU = 576 Byte

  Nutzlast = 552 Byte (572 Byte Gesamtlänge)

  Im Router 2 müssen 12 IP-Fragmente mit folgenden Nutzdatenlängen gebildet werden. Notation: (Laufende Nummer, Nutzdatenlänge im Byte):

  (1, 552); (2, 448); (3, 552); (4, 448); (5, 24); (6, 552); (7, 448); (8, 552); (9, 448); (10, 24); (11, 552); (12, 380)

- *In welchem System (Host oder Router) werden die IP-Fragmente wieder zusammengebaut? Wie wird in diesem System erkannt, welche IP-Fragmente zum ursprünglichen IP-Paket gehören?*

  Die Defragmentierung findet erst im Zielhost (B) statt. Mit Hilfe des Identifikation-Felds können die IP-Fragmente einem ursprünglichen IP-Paket zugeordnet werden.

41. *Eine Organisation hat von seinem ISP den IPv4-Adressblock 131.42.0.0/16 (classless) zugewiesen bekommen. Die Organisation möchte gerne ihr Netzwerk intern wie folgt aufteilen:*

    – *1 Subnetz mit bis zu 32000 Rechnern*
    – *15 Subnetze mit bis zu 2000 Rechnern*
    – *8 Subnetze mit bis zu 250 Rechnern*

    Der Adressblock wird zunächst in die zwei Adressblöcke 131.42.0.0./17 und 131.42.128.0/17 aufgeteilt. Zeigen Sie auf, wie die Organisation intern die Adressen weiter aufteilen könnte, um obiges Ziel zu erreichen. Hinweis: Alle beteiligten Router beherrschen CIDR.

131.42.0.0/17	1. Subnetz mit bis zu 32000 Rechnern (max. 32766)
131.42.128.0/21	1. Subnetz mit bis zu 2000 Rechnern (max. 2046)
131.42.136.0/21	2. Subnetz mit bis zu 2000 Rechnern (max. 2046)
…	
131.42.240.0/21	15. Subnetz mit bis zu 2000 Rechnern (max. 2046)
131.42.248.0/24	1. Subnetz mit bis zu 250 Rechnern (max. 254)
131.42.249.0/24	2. Subnetz mit bis zu 250 Rechnern (max. 254)
…	
131.42.255.0/24	8. Subnetz mit bis zu 250 Rechnern (max. 254)

42. *Was passiert, wenn ein IPv4-Fragment, also ein Teil eines IPv4-Pakets, in einem Netzwerk landet, dessen MTU kleiner ist als die Länge des Fragments?*

    Das IPv4-Fragment wird vom ersten Router dieses Netzwerkes weiter fragmentiert.

43. *Wo (auf welchem Rechner) werden IPv4-Fragmente wieder zum ursprünglich abgesendeten IPv4-Datagramm reassembliert?*

    Das Defragmentieren (reassembliert) wird bei IPv4 in Zielknoten vorgenommen.

44. *Was versteht man im Sinne der IP-Adressvergabe unter einem multihomed Host?*

    Als multihomed Host wird ein Host bezeichnet, der über mehrere IP-Adressen erreicht werden kann. Ein Host kann dazu entweder mehrere Netzwerkkarten nutzen oder eine Netzwerkkarte, die mehrere IP-Adressen verwalten kann.

45. *Wie lauten die entsprechenden Netzwerkmasken für die CIDR-Präfixnotationen /16, /20 und /24?*

    Die CIDR-Präfixnotation gibt an, wie viel Bit für den Netzwerkteil der IP-Adresse verwendet werden. Es ergeben sich somit folgende Netzwerkmasken:

    /16: 11111111.11111111.00000000.00000000 oder 255.255.0.0

    /20: 11111111.11111111.11110000.00000000 oder 255.255.240.0

    /24: 11111111.11111111.11111111.00000000 oder 255.255.255.0

46. *Aus dem Adressbereich 11.1.253/24 eines VLSM-Teilnetzes sollen /27-Teilnetze herausgeschnitten werden. Wie lauten die Teilnetzwerkadressen? Wie viele Adressen bleiben pro /27-Teilnetz?*

    1. VLSM-Teilnetz: 11.1.253.0/27
    2. VLSM-Teilnetz: 11.1.253.32/27
    3. VLSM-Teilnetz: 11.1.253.64/27

    …

    8. VLSM-Teilnetz: 11.1.253.224/27

    Es bleiben je Teilnetz 30 ($2^5 - 2$) Adressen für Hosts verfügbar.

47. *Erläutern Sie, wie ein neu in ein Netzwerk hinzukommender OSPF-Router seine Routing-Information aufbaut und verwaltet. Gehen Sie dabei auf den Begriff des Spanning-Tree und auf die nachbarschaftliche Beziehung der OSPF-Router ein.*

    Ein OSPF-Router bezieht die benötigten Routing-Informationen von den direkten Nachbarn oder speziellen „designierten" Routern. OSPF-Router bauen untereinander Verbindungen auf, um Routing-Informationen auszutauschen und die gegenseitige Erreichbarkeit fortlaufend zu prüfen.

Beim Austausch der Routing-Informationen verteilt ein OSPF-Router seine gesamten – nicht nur die eigenen – Verbindungsinformationen an seine direkten Nachbarn. Ein OSPF-Router, der Informationen enthält, gleicht diese mit seiner lokalen Datenbank ab und nimmt entsprechende Aktualisierungen vor. Diese Aktualisierungen werden dann an alle OSPF-Router weitergeleitet, zu denen eine Verbindung aufgebaut wurde.

Aus den Informationen der lokalen Datenbank berechnet jeder OSPF-Router eigenständig die günstigste Route zu allen bekannten Routern. Mit diesen Informationen wird ein Baum (Graphentheorie) aufgebaut, in dem der aktuelle Router die Wurzel darstellt. Als Kanten werden nur die günstigsten Routen zu den bekannten Zielen (weitere Knoten) berücksichtigt. Mit Hilfe dieses sogenannten „Spanning-Tree“ kann jeder OSPF-Router die optimalen Router für ein Ziel bestimmen.

48. *Ein Problem bei Routing-Protokollen ist das Konvergenzverhalten bzw. die Konvergenzdauer bei Änderungen der Netzwerktopologie oder bei Änderungen von Routen. Wie ist das Konvergenzverhalten bei den Routing-Protokollen RIP-2 und OSPFv2? Welche Mechanismen nutzt RIP-2 zur Verbesserung der Konvergenz? Sind in beiden Routing-Protokollen Endlosschleifen (Count-to-Infinity-Problem) möglich?*

Die Protokolle RIP-2 und OSPFv2 kommunizieren nur mit ihren direkten Nachbarn, um Änderungen der Netzwerktopologie auszutauschen. Im Gegensatz zu RIP-1 bietet RIP-2 eine sogenannte „Triggered-Updates-Methode“, die neue Routing-Informationen sofort weiterleitet. Eine neue Information verbreitet sich somit bei RIP-2 und OSPFv2 ähnlich schnell. Da RIP-2 auch ein Distance-Vector-Verfahren verwendet, sind Endlosschleifen prinzipiell möglich. Mit verschiedenen Techniken wie beispielsweise Split-Horizon-Technik wird versucht, das Count-to-Infinity-Problem zu beherrschen. OSPFv2 basiert auf den Link-State-Verfahren und kennt somit die gesamte Netzwerktopologie. Endlosschleifen können daher bei OPSFv2 vermieden werden.

49. *Wie funktioniert bei IPv6 prinzipiell die automatische Adresskonfiguration?*

In IPv6 ist die automatische Adresskonfiguration analog zu IPv4 mittels DHCPv6 (Dynamic Host Configuration Protocol) möglich. Die automatische Adresskonfiguration über DHCP setzt einen DHCP-Server voraus und wird „Stateful Autoconfiguration“ genannt.

IPv6 bietet auch eine automatische Adresskonfiguration ohne einen speziellen DHCP-Server. Diese Variante wird als „Stateless Autoconfiguration“ bezeichnet. Grundlage dieser automatischen Adresskonfiguration ist die Aufteilung der IPv6-Adresse mit lokaler Bedeutung in einen Link-Präfix (entspricht der Netzwerkadresse) und einen Link-Token (z.B. die MAC-Adresse). In einem ersten Schritt überprüft ein Host, ob sein Link-Token im Netz eindeutig ist. Dazu versendet er „Neighbor Solicitation“ Nachrichten an alle Host im lokalen Netz. Ist die Eindeutigkeit des Link-Token gegeben, wird in einem zweiten Schritt von einem lokalen

Router das Link-Präfix erfragt. Der Host kann aus dem Link-Präfix und dem als eindeutig bestätigten Link-Token eine vollständige IPv6-Adresse zusammenstellen.

10.5 Konzepte und Protokolle der Transportschicht

1. *Nennen Sie vier typische Protokollfunktionen, die in der Transportschicht implementiert sein sollten!*
   - Verbindungsmanagement und Adressierung
   - Zuverlässiger Datentransfer
   - Flusskontrolle
   - Staukontrolle
   - Multiplexierung und Demultiplexierung
   - Fragmentierung (bzw. Segmentierung) und Defragmentierung
2. *Beschreiben Sie einen Drei-Wege-Verbindungsaufbau und erläutern Sie kurz, wie man durch diese Art des Verbindungsaufbaus Duplikate erkennen kann!*
   - Host A initiiert einen Verbindungsaufbau mit einer Connect-Request-PDU und sendet dabei eine vorher ermittelte Folgenummer mit. Mit der Connect-Request-PDU wird auch die Transportadresse gesendet, um den Empfänger zu identifizieren.
   - Host B bestätigt den Verbindungsaufbauwunsch mit einer ACK-PDU, bestätigt dabei auch die Folgenummer und sendet seine eigene Folgenummer mit der ACK-PDU an Host A.
   - Host A bestätigt die Folgenummer von Host B mit der ersten Data-PDU im Pickypacking-Verfahren, und damit ist die Verbindung aufgebaut.

   Duplikate können aufgrund der vereinbarten und jeweils bestätigten Folgenummern (fortlaufende Zähler der abgesendeten Nachrichten) in Kombination mit einer maximalen Paketlebensdauer erkannt werden.
3. *Was ist eine Ende-zu-Ende-Verbindung im Sinne der Transportschicht?*

   Hierunter versteht man eine Verbindung zwischen zwei kommunizierenden Prozessen auf unterschiedlichen Rechnersystemen oder auch auf dem gleichen Rechnersystem.
4. *Nennen Sie je einen Vorteil für ein verbindungsloses und ein verbindungsorientiertes Protokoll der Transportschicht!*

   Vorteile eines verbindungslosen Protokolls:
   - Schnellere Kommunikation durch geringen Overhead
   - Einfachere Implementierung

Vorteile eines verbindungsorientierten Protokolls:

- Sichere Zustellung der Daten (kein Verlust von einzelnen Paketen)
- Fehlerfreie Auslieferung der Daten
- Reihenfolgerichtige Auslieferung der Daten

5. *Was will man mit einer Timerüberwachung beim Verbindungsabbau einer Transportverbindung erreichen?*

   Beim Verbindungsabbau sollen keine Nachrichten verloren gehen. Durch Störungen im Netz kann jedoch jederzeit eine Disconnect-Request-PDU oder eine Bestätigung verloren gehen. Über eine Timerüberwachung mit begrenzter Anzahl von Nachrichtenwiederholungen wird versucht, dieses Problem zu lösen. Falls alle Disconnect-Requests verloren gehen, baut der Sender die Verbindung ab. Der Partner weiß davon nichts (halboffene Verbindung), baut jedoch auch nach Ablauf eines Timers die Verbindung automatisch ab.

6. *Bei der Fehlerbehandlung nutzt man in einer gesicherten Transportschicht u.a. das positiv selektive und das negativ selektive Quittierungsverfahren. Erläutern Sie die beiden Verfahren!*

   Positiv selektives Verfahren: Es wird jede einzelne erfolgreich übermittelte Nachricht quittiert (hoher zusätzlicher Nachrichtenverkehr).

   Negativ selektives Verfahren: Es werden nur selektiv nicht empfangene, also verloren gegangene Nachrichten erneut angefordert.

7. *Zur Fehlerbehebung nutzt man in der Transportschicht das Verfahren der Übertragungswiederholung. Nennen Sie hierzu zwei Verfahren und erläutern sie diese kurz! Was muss der Sender tun, damit eine Übertragungswiederholung möglich ist?*

   Selektive Wiederholung: Nur negativ quittierte Nachrichten werden wiederholt. Der Empfänger puffert Nachrichten, bis die fehlende da ist.

   Go-Back-N-Verfahren: Es werden die fehlerhafte, sowie alle darauf folgenden Nachrichten erneut übertragen. Der Empfänger benötigt eine geringere Speicherkapazität.

   Bei beiden Verfahren muss der Sender die Nachrichten über einen gewissen Zeitraum bereithalten. Er kann sie nur beim Empfang einer ACK-PDU verwerfen. Beim Stop-and-Wait-Verfahren ist dies einfach. Der Sender muss nur eine Nachricht speichern und kann sie bei Empfang der ACK-PDU verwerfen. Schwieriger ist es beim positiv-kumulativen und beim negativ-selektiven Verfahren. Beim negativ-selektiven Verfahren ist es für den Sender problematisch festzustellen, wann eine gesendete Nachricht aus dem Puffer eliminiert werden kann. Er muss Nachrichten puffern, bis er sicher ist, dass sie übertragen wurden.

8. *Was ist bei TCP ein Port und was ist bei TCP ein well-known Port?*

   Anwendungsprozesse sind über sog. Sockets (IP-Adresse + Portnummer) adressierbar. Ein Serverprozess stellt einen Dienst bereit und bietet ihn über eine wohlbekannte Portnummer, einen sog. well-known Port an. Wenn ein Clientprozess die IP-Adresse und die Portnummer des Dienstes sowie das zugehörige Protokoll kennt, kann er mit dem Server kommunizieren und dessen Dienste nutzen. Well-known Ports sind innerhalb der TCP/IP-Gemeinde bekannt und werden immer gleich benutzt. Wichtige Well-known Ports und dazugehörige Dienste sind z.B.:

   - Port 23, Telnet (Remote Login)
   - Ports 20 und 21, ftp (File Transfer Protocol)
   - Port 25, SMTP (Simple Mail Transfer Protocol)
   - Port 80, HTTP (Hyper Text Transfer Protocol)

9. *TCP ist datenstromorientiert (stream-ortientiert). Was bedeutet das?*

   Ein Anwendungsprozess schreibt seine Daten Byte für Byte in einen Bytestrom. Im Gegensatz zur blockorientierten Übertragung kümmert sich hier die TCP-Instanz um die Segmentierung der Daten.

10. *Was sind TCP-Segmente, wie groß sind diese mindestens und warum?*

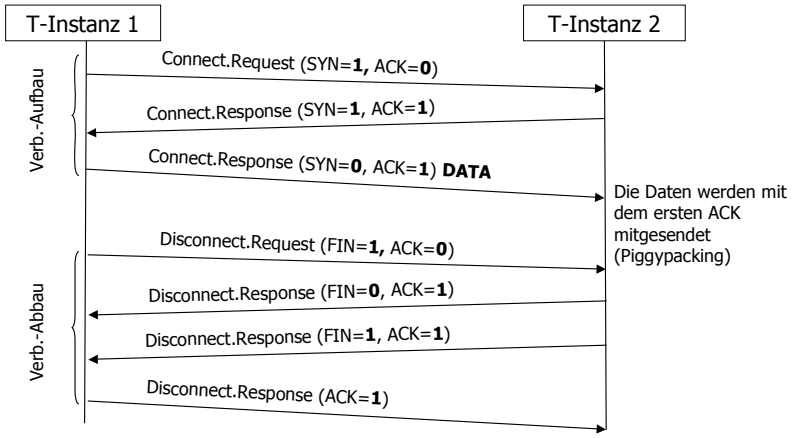
    TCP sieht den Datenstrom (Stream) als eine Sequenz von Octets (Byte) und unterteilt diese zur Übertragung in Segmente. Ein Segment besteht aus dem mindestens 20 Byte langen TCP-Header.

11. *Wie wird in TCP eine Verbindung eindeutig identifiziert?*

    Eine Verbindung wird durch ein Paar von Endpunkten eindeutig identifiziert (Socket Pair). Dies entspricht einem Quadrupel, das die IP-Adresse von Host 1, die Portnummer, die in Host 1 genutzt wird, die IP-Adresse von Host 2 und die Portnummer auf der Seite von Host 2 enthält.

    Beispiel: `((195.214.80.76, 80) (196.210.80.10,6000))`

12. *Wie viele TCP-Segmente werden für die Übertragung von 100 Byte Nutzdaten durchs Netz gesendet? Erläutern Sie dies anhand eines Time-Sequence-Diagramms!*



T-Instanz 1 baut mittels Three-Way-Handshake eine Verbindung zu T-Instanz 2 auf (3 TCP-Segmente) Mit dem letzten ACK des Three-Way-Handshake wird hier das zu sendende Datenpaket übertragen (Piggy-Packing). Somit kann ein weiteres Daten-TCP-Segment eingespart werden. Anschließend wird die Verbindung gemäß dem Protokoll abgebaut (3 bis 4 Segmente). Das Nutzdatenpaket wird hier im Rahmen des Verbindungsabbaus bestätigt.

Es werden also mindestens 7 Segmente benötigt, um 100 Byte Nutzdaten zu übertragen.

13. *Welche Quittierungsvariante wird bei TCP eingesetzt, damit der Empfänger den ordnungsgemäßen Empfang einer Nachricht bestätigen kann? Was macht der Sender eines Pakets, wenn er keine Quittung vom Empfänger erhält?*

    Es wird ein positives, kumulatives Bestätigungsverfahren mit Timerüberwachung bei jeder Nachricht verwendet. Bei Ablauf des Timers ohne vorige Bestätigung wird die Nachricht erneut gesendet.

14. *Erläutern Sie das Silly-Window-Syndrom bei TCP!*

    Wenn der Anwendungsprozess im Empfänger die empfangenen Daten byteweise ausliest, wird nach jedem gelesenen Byte eine ACK-PDU mit dem Hinweis, dass wieder ein Byte gesendet werden kann, übertragen. Das Netz wird also durch viele 1-Byte-Nachrichten (Nutzlast) belastet.

15. *Wozu dient das Slow-Start-Verfahren bei TCP? Beschreiben Sie das Verfahren kurz!*

    Das Slow-Start-Verfahren wird im Rahmen der Staukontrolle eingesetzt. Es wird versucht, sich an die maximale Menge an Daten, die über das Netz gesendet werden kann, heranzutasten. Zusätzlich zum Empfangsfenster der Flusskontrolle wird ein Überlastfenster eingeführt. Dabei gilt, dass der Sendekredit für eine TCP-Verbindung aus dem Minimum aus {Überlastungsfenster, Empfangsfenster} berechnet wird.

Das Slow-Start-Verfahren läuft in zwei Phasen ab:

- Slow-Start-Phase: Sender und Empfänger einigen sich beim Verbindungsaufbau auf eine erste zu sendende Anzahl an Segmenten mit der vereinbarten TCP-Segmentlänge. Der Sender sendet zunächst ein Segment dieser Länge. Kommt eine ACK-PDU hierfür rechzeitig, wird die Anzahl der Segmente, die gesendet werden dürfen, also das Überlastungsfenster, verdoppelt. Das geht bis zu einem Schwellwert so weiter (exponentielle Steigerung).
- Nach dem Erreichen des Schwellwerts geht das Verfahren in die sog. Probing- oder Überlastvermeidungsphase über. In dieser Phase erhöht sich bei jeder empfangenen Quittung die Anzahl der zulässigen Segmente nur noch linear um 1.

16. *Wozu dienen die beiden Zustände TIMED_WAIT und CLOSE_WAIT im TCP-Zustandsautomaten? Gehen Sie dabei kurz auf den Verbindungsabbau ein!*

    Der Client initiiert den Verbindungsabbau mit einem Socket-Close-Aufruf. Er sendet ein FIN-Segment und wechselt in den Zustand FIN_WAIT_1. Kommt ein ACK vom Server, wird die Verbindung in einer Richtung geschlossen und der Zustand wechselt zu FIN_WAIT_2. Schließt auch die andere Seite die Verbindung, wird ein FIN vom Client empfangen, das wiederum bestätigt wird.

    Der Client geht in den Zustand TIMED_WAIT. Um sicherzustellen, dass beim Verbindungsabbau keine Nachrichten verloren gehen, wird ein Timer aufgezogen, der als Wartezeit die doppelte Paketlebensdauer verwendet. Erst nach Ablauf des Timers wird in den Zustand CLOSED gewechselt und somit die Verbindung beendet.

    Auf Serverseite wird nach Bestätigung des Abbauwunsches des Clients in den Zustand CLOSE_WAIT gegangen. Nachdem alle noch zu sendenden Pakete abgearbeitet wurden, wird auch vom Server der Verbindungsabbau mit dem Senden eines FIN-Segments signalisiert. Der Server befindet sich danach im Zustand LAST_ACK, wo er noch auf die letzte Bestätigung wartet und anschließend in den Zustand CLOSED geht.

17. *Nennen Sie zwei Timer, die bei TCP verwendet werden, und beschreiben Sie kurz deren Aufgabe!*

    TCP verwendet folgende Timer:

    - Retransmission-Timer: Überwachung jeder einzelnen Übertragung der TCP-Segmente und Übertragungswiederholung nach Timerablauf.
    - Keepalive-Timer: Überprüfung, ob der Partner noch lebt.
    - Timed-Wait-Timer: Sicherstellen, dass alle Pakete bei einem Verbindungsabbau noch übertragen werden (doppelte Paketlebensdauer).

18. *Was unternimmt die empfangende UDP-Instanz, wenn eine UDP-PDU ankommt? Gehen Sie hier auf den Sinn des UDP-Pseudo-Headers ein!*

    Nach Empfang einer UDP-PDU wird anhand einer Prüfsumme das Gesamtpaket (-Daten+Header) überprüft. Dazu wird ein virtueller Header (Pseudo-Header) aufgebaut, der zusätzlich Informationen aus dem IP-Header enthält.

    Der Empfänger muss bei Empfang einer UDP-Nachricht, die eine Prüfsumme enthält, folgendes unternehmen:

    - Die IP-Adressen aus dem ankommenden IP-Paket lesen.
    - Der Pseudo-Header muss zusammengebaut werden.
    - Die Prüfsumme muss ebenfalls berechnet werden.
    - Die im Header gesendete Prüfsumme mit der berechneten vergleichen.

19. *Nennen Sie zwei Vorteile von UDP im Vergleich zu TCP!*

    Folgende Vorteile hat eine UDP-Nutzung:

    - Schneller, da (fast) keine Sicherungsmechanismen vorhanden sind
    - Einfacher zu implementieren

20. *Wozu dienen die Stati SYN_RECVD und SYN_SENT des TCP-Zustandsautomaten?*

    - SYN_RCVD: Ankunft einer Verbindungsanfrage und Warten auf Bestätigung.
    - SYN_SENT: Die Anwendung hat begonnen, eine Verbindung zu öffnen.

21. *Welchen Sinn hat der UDP-Pseudo-Header?*

    Der UDP-Pseudo-Header ist ein virtueller Header der u.a. die IP-Adresse des Senders und Empfängers beinhaltet. Über ihn wird eine Prüfsumme berechnet, die mit der übertragenen Prüfsumme im UDP-Header verglichen wird. Stimmen die Prüfsummen überein ist sichergestellt, dass das Datagramm den richtigen Empfänger (IP und Port) erreicht hat.

22. *Übernimmt eine UDP-Instanz die Aufgabe der Segmentierung eines langen Datagramms oder muss diese Aufgabe das Anwendungsprotokoll erledigen?*

    Eine UDP-Instanz segmentiert die Pakete. Dabei ist es sinnvoll, UDP-Nachrichten nicht länger als in der maximal möglichen IP-Paketlänge zu versenden, da sonst in der IP-Schicht fragmentiert wird.

23. *Erläutern Sie kurz, was bei einer negativen Quittierung für eine Transport-PDU passiert, wenn in einem Netzwerk mit hoher Pfadkapazität eine fensterbasierte Flusskontrolle mit einem großen Fenster angewendet wird und die Übertragungswiederholung im Go-Back-N-Verfahren erfolgt.*

    Bei entsprechender Fenstergröße können viele PDUs gesendet werden, bevor die erste Negativ-Quittung beim Sender eintrifft. Dies führt dann ggf. zur unnötigen Übertragungswiederholung vieler PDUs.

10.6 Ausgewählte Anwendungsprotokolle

1. *Wozu dient DNS im globalen Internet?*

   Das Domain Name System (DNS) ist ein sog. Internet-Directory-Service also eine Art Adressbuch des Internets. Es ist für die Abbildung von symbolischen Namen (Domainnamen usw.) auf Adressen (IP-Adressen usw.) zuständig.

2. *Wie sind die Domains im Internet strukturiert, was sind Zonen im DNS und wer verwaltet diese?*

   DNS ist ein hierarchisches Namensverzeichnis für IP-Adressen, das in einer Baumstruktur organisiert ist. Das Internet ist dabei in mehrere hundert *Domänen* aufgeteilt. Die Domänen sind wiederum in *Teildomänen* (Subdomains) untergliedert usw. Dabei ist zu beachten, dass dies einer rein organisatorischen und keiner physikalischen Einteilung entspricht.

   Ein DNS-Server verwaltet jeweils sog. Zonen des DNS-Baums, wobei eine Zone an einem Baumknoten beginnt und die darunterliegenden Zweige beinhaltet. Die Verantwortung darunterliegender Subzonen kann an weitere DNS-Server delegiert werden.

3. *Wie findet ein Clientrechner zu einem Rechnernamen eines Servers die zugehörige IP-Adresse?*

   Der Clientrechner, bzw. der entsprechende Anwenderprozess, wendet sich lokal an einen *Resolver*. Der Resolver kann die Anfrage entweder lokal befriedigen, wenn er die IP-Adresse in seinem Cache gespeichert hat, oder er setzt einen Request an den ihm zugeordneten lokalen DNS-Server ab. Der DNS-Server prüft seinerseits, ob er die Adresse in seinem Cache hat. Falls ja, dann gibt er diese in einer Response-Nachricht an den Resolver zurück. Falls er die Adresse nicht hat, sendet er seinerseits einen Request an den nächsten DNS-Server. Dabei wird der Hostname je nach Implementierung mit einer iterativen bzw. rekursiven Anfrage aufgelöst. Auch Mischformen sind denkbar.

4. *Welche Aufgabe hat der Resolver im DNS und auf welchem Rechner liegt er typischerweise?*

   Der DNS-Resolver liegt typischerweise auf dem Client und stellt über eine API den Anwendungsprozessen die entsprechende Funktionalität zur Namensauflösung zur Verfügung. Dieser übernimmt beim Aufruf die Delegation der Anfrage an den zugeordneten DNS-Server.

5. *Wie ist ein DNS-Resolver üblicherweise implementiert?*

   Der DNS-Resolver ist üblicherweise in Systembibliotheken implementiert.

6. *Wozu dient die DNS-Domäne in-addr.arpa?*

   Die DNS-Domäne in-addr.arpa dient zur umgekehrten Namensauflösung, also der Umwandlung von IP-Adressen in die zugehörigen Hostnamen.

7. *Welche anderen DNS-Name-Server muss ein DNS-Name-Server kennen, der für die Verwaltung einer Zone zuständig ist und drei Subzonen an darunterliegende Name-Server delegiert hat?*

   Der DNS-Server muss die Server der „delegierten" Subzonen, sowie den DNS-Server der ihm übergeordneten Zone kennen. Zusätzlich hat jeder Server eine Liste mit den Adressen der DNS-Root-Name-Server.

8. *Was ist der Unterschied zwischen iterativer und rekursiver Behandlung einer DNS-Anfrage?*

   Iterativ: Die kontaktierten DNS-Server geben jeweils nur die als nächstes zuständigen DNS-Server bekannt. Der lokale Server muss demnach alle „auf dem Weg liegenden" DNS-Server kontaktieren.

   Rekursiv: Jeder angefragte DNS-Server gibt die Anfrage an den nächsten DNS-Server weiter und erhält irgendwann das Ergebnis zurück, das er dann seinerseits an den anfragenden DNS-Server bzw. Host weiterreicht.

9. *Welche Komponente entscheidet, ob beim Auflösen eines Namens in DNS nach der iterativen oder der rekursiven Abfragemethode vorgegangen werden soll?*

   Im DNS-Server ist implementiert, wie die Abfrage behandelt werden soll. Daher sind bei mehreren DNS-Servern auch Mischformen möglich.

10. *Wozu werden Resource Records vom Typ „MX" und wozu solche mit Typ „A" benötigt?*

    MX: Resource Record für die Angabe eines Mailservers

    A: Resource Record für eine "normale" IP-Adresse

11. *Was versteht man unter einem autoritativen DNS-Server und woran merkt man, dass eine DNS-Response nicht von einem autoritativen DNS-Server kommt?*

    Der einem Host direkt zugeordnete DNS-Server wird als autoritativer DNS-Server bezeichnet. Er verfügt immer über die Adressen der ihm direkt zugeordneten Hosts und ist verantwortlich für eine Zone.

    Im DNS-Header befindet sich ein Feld mit Parametern, von denen ein Flag angibt, ob die Antwort von einem autoritativen Server stammt.

12. *Warum sind die DNS-Root-Name-Server so wichtig und wie findet ein lokaler DNS-Server einen DNS-Root-Name-Server?*

    Ein DNS-Root-Name-Server verfügt über Informationen zu allen freigegebenen Top-Level-Domains. Eine Suchanfrage findet hier immer als Ergebnis den nächsten zuständigen DNS-Server. Auf jedem DNS-Server-System sind in statischen Tabellen die Adressen der DNS-Root-Name-Server hinterlegt. Kann kein DNS-Server erreicht werden, wird direkt ein DNS-Root-Name-Server kontaktiert.

13. *Beschreiben Sie grob den Aufbau einer HTTP-PDU!*

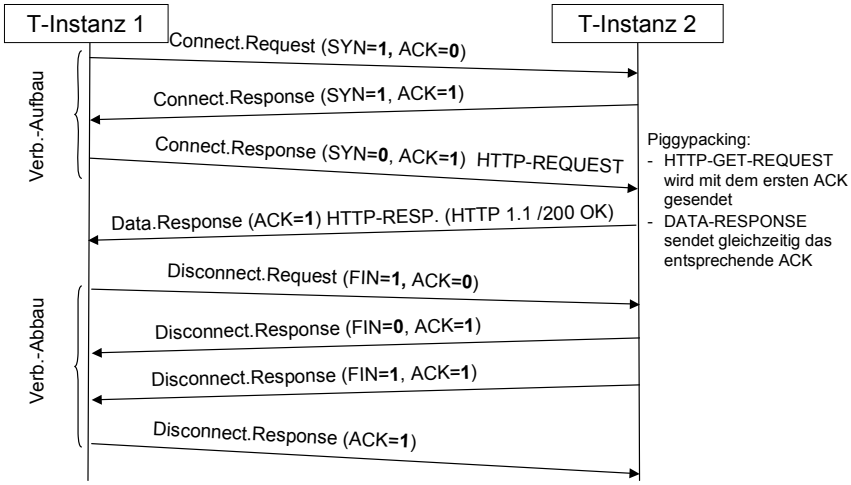
    Folgende PDU-Bestandteile sind definiert:

    – Request-Line (Anfragezeile): „Haupt-Header“ mit der Angabe der Operation (GET/POST) und der URL. Optional: Eingabeparameter aus den HTML-Formularen als (name, value)-Paare mit übertragen.

    – Liste von Request-Headern

    – Entity-Body: Nutzdatenteil der Nachricht

14. *Beschreiben Sie den Ablauf der Kommunikation zwischen einem WWW-Browser und einem WWW-Server zum Lesen eines Dokuments! Erläutern Sie hierbei, was dabei auf der TCP-Ebene passiert, welche und wie viele TCP-Nachrichten gesendet werden. Hinweis: Gehen Sie von einem sehr kleinen Dokument aus (ca. 200 Byte).*



Nach einem erfolgreichen TCP-Verbindungsaufbau wird per Piggypacking der HTTP-GET-Request gesendet. Die erfolgreiche Antwort (HTTP 1.1 / 200 OK) wird mit dem nächsten TCP-Segment gesendet, welches gleichzeitig das vorige TCP-Segment bestätigt. Anschließend wird der Verbindungsabbau durchgeführt.

Es müssen also mind. 8 TCP-Segmente versandt werden.

15. *Was ist der Unterschied zwischen einer GET- und einer POST-Operation im HTTP-Protokoll?*

    GET: Der Nutzdatenteil ist leer. Parameter aus Formularen werden, falls vorhanden, über die Request-Line übertragen.

    POST: Parameter aus Formularen werden im Nutzdatenteil übertragen.

16. *HTTP ist ein zustandsloses Protokoll. Was bedeutet dies?*

    Im Kommunikationsverlauf werden keine Zustandsinformationen gespeichert. Der Server beantwortet also jede Anfrage, als wäre sie die erste.

17. *Wie wird z.B. in komplexeren WWW-Anwendungen eine Zustandsverwaltung durchgeführt (kurze Erläuterung)?*

    Über entsprechende Parameter (Sitzungs-Ids) werden HTTP-Requests erkannt und gespeicherten Sitzungen auf dem Server zugeordnet.

18. *Wie kann ein HTTP-Client angeben, dass er nur die MIME-Typen text/html und text/plain bearbeiten kann?*

    Im HTTP-Request-Header können alle Formate angegeben werden, die der Client verarbeiten kann, Abgabe z.B. accept: text/html, text/plain.

19. *Was bedeutet die Angabe „Connection: Keep alive" im Header des HTTP-Requests für das Verbindungsmanagement zwischen Web-Client und Web-Server?*

    Die verwendete TCP-Verbindung zwischen Client und Server wird für weitere HTTP-Requests aufrechterhalten.

20. *Was ist POP3 und wozu wird es verwendet?*

    Das Post Office Protocol Version 3 (POP3) ist ein Mailzugangsprotokoll. Ein Internetnutzer erhält über POP3 lesenden Zugang zu seiner Mailbox.

21. *Wozu dient das Protokoll SMTP im Internet?*

    Das Simple Mail Transfer Protocol (SMTP) wird von SMTP-Servern zur Kommunikation untereinander verwendet. Das Protokoll spezifiziert den Empfang und Versand von elektronischen Nachrichten. Zudem sendet der Mail-Client über SMPT seine Mail-Nachrichten an den Server.

22. *Wie unterscheidet sich eine traditionelle Web-Kommunikation von der Kommunikation mit AJAX?*

    Traditionell kommuniziert ein Browser über Formulardaten mit einem Server, der nach Auswertung dieser Daten mit einer neuen HTML-Seite antwortet. Bei AJAX können, angestoßen durch *JavaScript*, auch asynchrone Anfragen an einen Webserver gesendet und die Antworten ausgewertet werden, ohne dass die Seite neu gerendert werden muss. Der Datenaustausch mit dem Webserver wird durch eine AJAX-Engine aus Sicht des Browser-Anwenders im Hintergrund abgewickelt.

23. *Was versteht man bei AJAX unter konkurrierenden Requests?*

    Wenn bei Einsatz von AJAX ein HTTP-Request einen vorher abgesetzten HTTP-Request überholt und beide Requests dasselbe Auslöseereignis haben, spricht man von einem konkurrierenden Request. Hier kann es zu Inkonsistenzen kommen.

24. *Welche Anforderungen haben Audio- und Video-Streaming-Anwendungen an die Kommunikation im Netzwerk?*

    Derartige Anwendungen fordern einen kontinuierlichen Datenstrom. Typischerweise werden kontinuierliche Abfolgen von Daten über einen längeren Zeitraum übertragen. Das bedeutet, dass auch die zeitliche Reihenfolge der übertragenen Daten einen Teil der Information ausmacht. Man spricht auch von zeitabhängigen Medien. Die Übertragung erfolgt im Gegensatz zu den klassischen Anwendungen synchron, wobei eine maximale Verweilzeit der Nachrichten wichtig ist. Verzögerungsschwankungen (Jitter) sollen möglichst minimal sein.

25. *Wozu dient das RTP-Protokoll?*

    RTP ist ein „Transportprotokoll" zur Beförderung von Medienströmen (Video und/oder Audio-Streams). Beispielsweise wird RTP für IP-Telefonie-Anwendungen und auch für Audio- und Videokonferenz-Anwendungen eingesetzt.

26. *Was ist ein Codec?*

    Als Codec bezeichnet man ein Kompressions-/Dekompressions-Paar. Der Begriff setzt sich aus den englischen Begriffen *coder* und *decoder* zusammen. Codecs werden zur Kompression von Multimedia-Daten (z.B. Audio und Video) verwendet.

27. *Was versteht man unter UPnP?*

    UPnP (Universal Plug and Play) ist eine Technologie und ein Architekturmodell für Netzwerke mit geringem Konfigurationsaufwand (Zero-Configuration). Die Geräte sollen sich dynamisch kennenlernen. Es wird hauptsächlich für Multimedia-Systeme im Home-Office benutzt. UPnP nutzt vorhandene Protokolle wie DNS, DHCP, SSDP, HTTP, SOAP.

10.7 Grundlagen der mobilen Kommunikation

1. *Welche Aufgabe übernimmt der HA bei der IPv4-Mobilitätsunterstützung?*

    Der HA (Home Agent) stellt sicher, dass der MN (Mobile Node) unter einer festen IP-Adresse erreicht wird. Diese feste IP-Adresse gehört topologisch zum Netzwerk des HA (das Heimnetzwerk). Der HA erhält somit alle Pakete, die an diese feste IP-Adresse gesendet werden, und kann evtl. die Weiterleitung an den MN veranlassen.

2. *Warum kann es beim Dreiecks-Routing im Vergleich zur direkten Route bei einer normalen IP-Kommunikation zu deutlich schlechteren Laufzeiten der IP-Pakete kommen?*

    Beim Dreiecks-Routing werden die Pakete vom CN (Correspondent Node) zum MN (Mobile Node) immer über den HA (Home Agent) versendet. Befinden sich CN und MN im gleichen Teilnetz, der HA jedoch in einem weit entfernten Netzwerk, werden die Pakete aus dem gemeinsamen Teilnetz zunächst an den HA gesendet. Dieser empfängt die Pakete für den MN und sendet diese über einen IP-

Tunnel zurück in das gemeinsame Teilnetz der beiden Kommunikationspartner. Ein direktes Routing vom CN zum MN würde in diesem Fall nur das gemeinsame Teilnetz betreffen und somit viel effizienter sein.

3. *Erklären Sie kurz, warum es beim Wechsel von Teilnetzen ohne spezielles Handoff-Verfahren zum Verlust von Datenpaketen kommen kann.*

   Wechselt ein MN (Mobil Node) von einem Teilnetz zum nächsten, erfährt der HA (Home Agent) davon erst nach der erfolgreichen Registrierung der neuen CoA (Care-of-Address). Bis diese Registrierung der neuen CoA beim HA eintrifft, werden eintreffende Pakete für den MN noch an die alte CoA und somit in das alte Teilnetz weitergeleitet. Der FA (Foreign Agent) des alten Teilnetzes kann diese Pakete aber möglicherweise nicht mehr an den MN ausliefern, wenn die Verbindung zum MN schon abgebaut wurde. Der FA des alten Teilnetzes muss ohne spezielles Handoff-Verfahren noch eintreffende Pakete für die MN verwerfen.

4. *Beschreiben Sie die unterschiedliche Zielsetzung des Pre- und Post-Registration-Handoff-Verfahrens beim Wechsel (Handoff) von Teilnetzen.*

   Beim Post-Registration-Handoff wird versucht, nach der Durchführung des Handoffs die Pakete, die noch im alten Teilnetz eintreffen, zur neuen CoA weiterzuleiten. Ziel dieses Verfahrens ist es somit, den Verlust von Datenpaketen möglichst zu vermeiden.

   Im Gegensatz dazu wird beim Pre-Registration-Handoff versucht, vor der Durchführung des Handoffs schon Vorbereitungen für den anstehenden Teilnetzwechsel zu treffen. Ziel ist es, Verzögerungen in der Auslieferung der Datenpakete, die durch den Teilnetzwechsel auftreten können, zu minimieren.

5. *Erläutern Sie kurz das Optimierungsprinzip eines Performance-Enhancing-Proxy (PEP).*

   Der Übertragungsweg zwischen zwei Kommunikationspartnern besteht in der Regel aus mehren Teilstrecken, die unterschiedliche Technologien in den unteren Schichten (Schicht 1 und 2 der ISO/OSI-Referenzmodells) nutzen können. Wird, wie beispielsweise bei TCP, eine Ende-zu-Ende-Semantik über den gesamten Übertragungsweg realisiert, können Besonderheiten einzelner Teilstrecken nicht berücksichtigt werden. Mit einem PEP wird versucht, den gesamten Übertragungsweg in zwei Abschnitte aufzuteilen, um Optimierungen für einen abgegrenzten Teil des Übertragungsweges zu realisieren. Beim Einsatz der PEP wird jedoch versucht, die eigentliche Ende-zu-Ende-Semantik über den gesamten Übertragungsweg (mindestens in Teilen) aufrecht zu erhalten.

10.8 Entwicklung von Kommunikationsanwendungen

1. *Beschreiben Sie den Unterschied zwischen synchroner und asynchroner Kommunikation!*

   Synchrone Kommunikation bedeutet blockierende Kommunikation. Der Sender wartet auf ein Ergebnis und kann in der Zwischenzeit nichts anderes tun. Asynchron bedeutet nicht blockierend. Der Sender schickt die Anfrage und kann bis zum Eintreffen des Ergebnisses weiterarbeiten.

2. *Was versteht man unter einem synchronen, entfernten Dienstaufruf im Vergleich zu einem asynchronen entfernten Dienstaufruf?*

   Beim synchronen, entfernten Dienstaufruf blockiert der Sender, bis ein Ergebnis da ist. Beim asynchronen, entfernten Dienstaufruf kann der Sender nach dem Senden der Anfrage weiterarbeiten und muss sich nach einer gewissen Zeit wieder mit dem Empfänger synchronisieren, um das Ergebnis abzuholen.

3. *Was versteht man unter einem Objektstrom in Java und wozu wird der Mechanismus bei der TCP-Socket-Programmierung verwendet?*

   Die TCP-Socket-Schnittstelle ist streamorientiert. Es wird also auf dieser Ebene zwischen Sender und Empfänger über einen Datenstrom kommuniziert. Java bietet einen vordefinierten Objektstrom an, der die Serialisierung der Datenobjekte übernimmt und über den eigentlichen Input-/Output-Stream gelegt werden kann.

4. *Kann man Java-Objektströme auch für Datagramm-Sockets verwenden? Erläutern Sie Ihre Entscheidung!*

   Java-Objektströme sind nicht geeignet für eine Kommunikation über UDP. Da UDP nachrichtenorientiert ist, müssen die zu übertragenden Daten als Byte-Array-Pakete einzeln übertragen werden.

5. *Warum erzeugt ein TCP-basierter Server bei einem ankommenden Verbindungsaufbauwunsch in der Regel für die neue Verbindung einen neuen Prozess bzw. Thread für die neue Verbindung?*

   Die aufgebaute Verbindung wird in einen eigenen neuen Prozess überführt, der die weitere Bearbeitung im Sinne der Datenübertragung übernimmt. Der Hauptprozess ist dadurch entlastet und kann weiterhin auf neue Verbindungsanfragen warten.

6. *Was muss in einem Client- und was in einem Serverprogramm programmiert werden, damit diese für die Kommunikation über UDP-Datagramm-Sockets vorbereitet sind?*

   Da UDP verbindungslos ist, findet auch kein Verbindungsaufbau bzw. -abbau statt. Daher muss sowohl auf Server- als auch auf Clientseite nur ein UDP-Socket erzeugt werden, bevor mit Sende- und Empfangsmethoden Daten ausgetauscht werden können. Zum Versand der Daten müssen die IP- und die Portadresse des Partners bekannt sein.

7. *Was macht die Socket-Programmierung mit TCP-Sockets in Java so einfach im Vergleich zur Nutzung der Standard-Socket-Bibliothek der Sprache C?*

   Java stellt dem Netzwerkentwickler das Package java.net zur Verfügung, welches Objektklassen beinhaltet, mit denen die Kommunikation einfach implementiert werden kann.

8. *Was hat ein Java-Objektstrom mit dem TCP-Stream zu tun? Erläutern Sie Ihre Antwort kurz!*

   Java Objektströme serialisieren Java-Objekte und können über einen Java-Input-/Outputstream gelegt werden, sodass die in den Strom eingestellten Datenobjekte ohne weitere Behandlung übertragen werden.

9. *Erläutern Sie die Fehlersemantik Exactly-Once im Vergleich zu At-Most-Once bei auftragsorientierter Kommunikation! Wie kann man Exactly-Once realisieren (kurze Erläuterung)?*

   Bei Exactly-Once findet die Bearbeitung einer Anfrage genau einmal statt, auch bei beliebigen Fehlersituationen. Bei At-Most-Once findet die Bearbeitung einer Anfrage höchstens einmal statt, kann aber im Fehlerfall auch gar nicht ausgeführt werden.

   Die Realisierung der Exactly-Once-Fehlersemantik ist sehr aufwändig. Der Server muss auch über einen Systemausfall hinaus speichern, welche Anfragen schon bearbeitet wurden. Wird eine Anfrage erneut, also doppelt gestellt, darf der Server diese nicht bearbeiten und muss das zuvor gespeicherte Ergebnis an den Client senden.

10. *Warum müssen Objekte, die in Java-Programmen mit TCP-Sockets gesendet werden, das Interface Serializable implementieren und wie macht man das in Java?*

    Die Implementierung des Interface wird in der Klassendeklaration wie folgt angegeben:

    ```
    public class DataObject implements Serializable
    ```

    Dadurch können Java-Objekte über einen Objektstrom serialisiert und versendet werden. Die JVM stellt dabei sicher, dass das Datenobjekte korrekt zerlegt (serialisiert) und wieder zusammengebaut (deserialisiert) wird.

11. *Was ist der prinzipielle Unterschied zwischen den beiden Fehlersemantiken At-Most-Once und At-Least-Once? Welche der beiden ist einfacher zu implementieren und warum?*

    Bei At-Most-Once wird eine Anfrage höchstens einmal ausgeführt. Im Fehlerfall kann es jedoch vorkommen, dass sie nicht bearbeitet wird. Bei At-Least-Once wird dagegen eine Anfrage mindestens einmal ausgeführt. Im Fehlerfall kann es jedoch vorkommen, dass sie öfters ausgeführt wird.

At-Least-Once ist einfacher zu implementieren, da keine Überprüfungen auf Duplikate notwendig ist. Bei At-Most-Once muss eine Request-Liste im Server verwaltet werden.

12. *Funktioniert das Zusammenspiel zwischen einem in Java entwickelten TCP-Client und einem in C# entwickelten TCP-Server reibungslos, wenn beide zur Nachrichtenübertragung die in der jeweiligen Sprache angebotenen Serialisierungsmechanismen (Java-Objektstrom und C#-Formatter) verwenden? Begründen Sie Ihre Entscheidung!*

    Java-Objektstrom und C#-Formatter sind nicht miteinander kompatibel, da sie verschiedene Regeln zur Serialisierung verwenden.

13. *Erläutern Sie die Aufgaben der C#-Klassen TcpClient und TcpListener!*

    TcpClient: Die Klasse stellt einfache Methoden für die Verbindung, das Senden und Empfangen von Daten in einen Socket-Stream im synchronen Blockiermodus bereit.

    TcpListener: Die Klasse richtet eine Warteschlange ein und überwacht eingehende Verbindungsanforderungen.

Literaturhinweise

[Abts 2007] *Abts, D.:* Masterkurs Client/Server-Programmierung mit Java, 2. Auflage, Vieweg Verlag, 2007

[Badach 2001] *Badach, A.; Hoffmann, E.:* Technik der IP-Netze, Hanser-Verlag, 2001

[Bengel 2001] *Bengel, G.:* Verteilte Systeme, 2. Auflage, Vieweg-Verlag, 2001

[Bakre 1995] Bakre, A.; Badrinath, B.R.: I-TCP: Indirect TCP for Mobile Hosts; Proceedings - International Conference on Distributed Computing Systems; 1995

[Comer 2002] *Comer, Douglas, E.:* Computernetzwerke und Internets, 3. überarbeitete Auflage, Pearson Studium, 2002

[Gerdsen 1994a] *Gerdsen, P.; Kröger, P.:* Kommunikationssysteme 1 Theorie Entwurf Meßtechnik, Springer Verlag, 1994

[Gerdsen 1994b] *Gerdsen, P.; Kröger, P.:* Kommunikationssysteme 2 Anleitung zum praktischen Entwurf (SDL), Springer Verlag, 1994

[Coularis 2002] *Coulouris, G.; Dollimore, J.; Kindberg, T.:* Verteilte Systeme Konzepte und Desgin, Pearson Studium, 2002

[Haase 2001] *Haase, O.:* Kommunikation in verteilten Anwendungen, Einführung in Sockets, Java RMI, CORBA und Jini, Oldenbourg Verlag, 2001

[Hafner 2000] *Hafner, K; Lyon, M.:* ARPA KADABRA oder die Geschichte des Internet, dpunkt.verlag, 2000

[Hansen 2005] *Hansen H. R.; Neumann, G.:* Wirtschaftsinformatik 2 Informationstechnik, Lucius & Lucius, 2005

[Henning 2007] Henning, P. A.: Taschenbuch Multimedia, Hanser Verlag 2007

[Kurose 2002] *Kurose, J. F.; Ross, K. W.:* Computernetze, Pearson Studium, 2002

[Kurose 2008] *Kurose, J. F.; Ross, K. W.:* Computernetze, 4. aktualisierte Auflage, Pearson Studium, 2008

[Lockemann 1993] *Lockemann u.a.:* Telekommunikation und Datenhaltung, Hanser Verlag, 1993

[Meyer 2002] *Meyer, M.:* Kommunikationstechnik, Konzepte der modernen Nachrichtenübertragung, 2. Auflage, Vieweg Verlag, 2002

[Mintert 2007] *Mintert, S.; Leisegang, Ch.*: Ajax: Grundlagen, Frameworks und Praxislösungen, dpunkt.verlag, 2007

[Peterson 2007] *Peterson, L. L., Davie, B. S.:* Computernetze, Eine systemorientierte Einführung, dpunkt.verlag, 2007

[Popp 2004] *Popp, M.; Weber, K.:* Der Schnelleinstieg in PROFINET, PROFIBUS Nutzerorganisation e.V., 2004

[Rech 2008] *Rech, J.:* Ethernet Technologien und Protokolle für die Computervernetzung, 2. Auflage, Heise Verlag, 2008

[Riggert 2001] *Riggert, W.:* Rechnernetze, München, Wien 2001

[Roth 2002] *Roth, J.:* Mobile Computing, Grundlagen, Technik, Konzepte, dpunkt.verlag, 2002

[Sauter 2004] *Sauter, Martin:* Grundkurs Mobile Kommunikationssysteme, Vieweg Verlag, 2004

[Schill 2007] Schill, A., Springer, T.: Verteilte Systeme, Springer-Verlag, 2007

[Schiller 2000] *Schiller, J.:* Mobilkommunikation, Techniken für das allgegenwärtige Internet, Addison-Wesley, 2000

[Sikora 2003] *Sikora, A.:* Technische Grundlagen der Rechnerkommunikation, Internet-Protokolle und Anwendungen, Hanser Verlag, 2003

[Stein 2004] *Stein, E.:* Taschenbuch Rechnernetze und Internet, Fachbuchverlag Leipzig, 2. Auflage, 2004

[Steinmetz 2000] *Steinmetz, R.*: Multimedia-Technologie Grundlagen, Komponenten und Systeme, 3. Auflage, Springer-Verlag, 2000

[Stevens 2000] *Stevens, R. W.:* Programmieren von UNIX-Netzen, Hanser Verlag, 2000

[StevensP 2000] *Stevens, P.; Pooley, R.:* UML Softwareentwicklung mit Objekten und Komponenten, Pearson Studium, 2000

[Sun 2005] *Sun Microsystems:* Java2 Platform Standard Edition 1.5

[Tanenbaum 2003a] *Tanenbaum, A. S.:* Computernetze, 4. Auflage, Prentice Hall, 2003

[Tanenbaum 2003b] *Tanenbaum, A.S., van Stehen, M.:* Verteilte Systeme Grundlagen und Paradigmen, Pearson Studium, 2003

[Turner 1993] *Turner, K .J.:* Using Formal Description Techniques An Introduction to ESTELLE, LOTUS und SDL, John Wiley & Sons, 1993

[UpnP-Forum 2006] *UpnP-Forum*: http://www.upnp.org (Letzter Zugriff am 02.07.2009)

[Weber 1998] *Weber, M.:* Verteilte Systeme, Spektum Akademischer Verlag, 1998

[Wiese 2002] *Wiese, H.*: Das neue Internetprotokoll IPv6, Hanser Verlag, 2002

[Wöhr 2004] *Wöhr, H.:* Webtechnologien, dpunkt.verlag, 2004

[Zitterbart 1995] *Zitterbart, M.:* Hochleistungskommunikation Band 1: Technologie und Netze, Oldenbourg Verlag, 1995

[Zitterbart 1996] *Zitterbart, M.; Braun, T.:* Hochleistungskommunikation Band 2: Transportdienste und -protokolle, Oldenbourg Verlag, 1996

Sachwortverzeichnis