Datenkommunikation und Sicherheit Panikzettel

der Dude*, Luca Oeljeklaus, Philipp Schröer

31. Juli 2020

Inhaltsverzeichnis

*Pseudonyme gehören anonymen Autoren, die anonym bleiben wollen.

1 Grundlagen

1.1 Dienste

- *Dienst*: Dienste sind von Netzen oder Schichten (siehe OSI-Modell) bereitgestellte Kommunikationsfunktionen.
- *Dienstprimitiv*: Dienstprimitive sind die einzelnen Funktionen, welche ein Dienst anbietet.
- *Diensterbringer*: Bereitsteller des Dienstes.
- *Instanz*: Eine Instanz ist eine an der Erbringung des Dienstes aktiv beteiligte Komponente.
- *Protokoll*: Ein Protokoll beschreibt nach welchen Regeln Softwareinstanzen, welche an der Erbringung von Diensten beteiligt sind, miteinander interagieren.
- *Schichten*: Funktionalität des Diensterbringers für Modularität in Schichten aufgeteilt. Dienste werden von einer Schicht in einer Schichtenstruktur an die nächsthöhere (abstraktere) Schicht bereitgestellt. Andersrum greift eine Schicht auf die Dienste der unterliegenden Schicht zu.

Dienstprimitive werden nach folgendem Schema benannt:

`Schichtabkürzung-Dienstleistung.Diensttyp`

Die *Dienstleistung* kann etwa Connect (`Con`), Data (`Dat`), Abort (`Abo`) oder Disconnect (`Dis`) sein. Es gibt vier *Diensttypen:* Request (`Req`), Indication (`Ind`), Response (`Rsp`) und Confirmation (`Cnf`).

Um die Abfolge von Dienstprimitiven bei Benutzung eines Dienstes zu spezifizieren, kann man Zeitablaufsdiagramme (Weg-Zeit-Diagramme) und Zustandsübergangsdiagramme (Automaten) verwenden.

1.2 Vertikale Kommunikation

Nach dem OSI-Modell wird Kommunikation zwischen Schichten so definiert:

- Dienstdateneinheit *SDU* (Service Data Unit): Daten, welche von einer Schicht übertragen werden sollen.
- Schnittstellenkontrollinformationen *ICI* (Interface Control Information): Enthält Informationen über den zu erbringenden Dienst, z.B. Länge der SDU.
- Schnittstellendateneinheit *IDU* (Interface Data Unit): SDU und ICI.
- Protokollkontrollinformationen *PCI* (Protocol Control Information): Kontrollinformationen, welche zwischen den einzelnen Instanzen dieser Schicht ausgetauscht werden sollen.
- Protokolldateneinheit *PDU* (Protcol Data Unit) PCI und SDU.

Die Kommunikation der (N)-Schicht sieht dann wie folgt aus:

1. Die (N)-Instanz erhält von der $(N{+}1)$-Instanz die $(N{+}1)$-IDU und teilt diese in (N)-SDU (Daten) und (N)-ICI (Schnittstelleninfos) auf.
2. Zur Kommunikation mit einer anderen (N)-Instanz wird die (N)-PDU erstellt, die aus (N)-PCI (Protokollinfos) und (N)-SDU (Daten) besteht.
3. Die (N)-PDU wird dann zusammen mit der $(N{-}1)$-ICI (Kontrollinfos für die nächste Schicht) zur $(N{-}1)$-IDU zusammengefasst.

Für jede Schicht gilt also: (N)-PDU = $(N{-}1)$-SDU.

1.3 OSI und TCP/IP

OSI	TCP/IP
1. *Bitübertragungsschicht:* Digitale Datenübertragung, z.B. Ethernet.	1. *Host-to-Network Layer:* Datenübertragung von Punkt zu Punkt, z.B. WLAN, Token Ring, (R)ARP.
2. *Sicherungsschicht:* Gewährleisten von zuverlässiger und fehlerfreier Übertragung, z.B. MAC, ALOHA, Token Ring.	
3. *Vermittlungsschicht:* Vermittlung von Paketen und Routing, z.B. IP, ICMP.	2. *Internet Layer:* Weitervermittlung von Paketen und Routing, z.B. IP, ICMP.
4. *Transportschicht:* Ende-zu-Ende-Kommunikation, z.B. TCP, UDP.	3. *Transport Layer:* Ende-zu-Ende-Kommunikation, z.B. TCP, UDP.
5. *Sitzungsschicht:* Synchronisierter Datenaustausch, z.B. HTTP.	
6. *Darstellungsschicht:* Übersetzer zwischen Datenformen, z.B. HTTP.	4. *Application Layer:* Austausch von anwendungsspezifischen Daten, z.B. HTTP, DNS.
7. *Anwendungsschicht:* Anwendungsspezifische Daten, z.B. HTTP.	

2 Bitübertragungsschicht

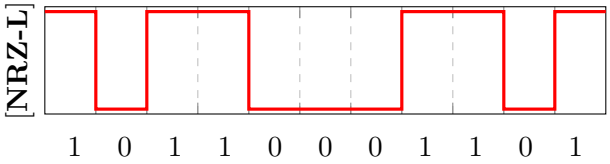
2.1 Definitionen

- *Schritt*: Minimales Zeitintervall für die Änderung des Signals.
- *Schrittgeschwindigkeit*: Schritte pro Zeiteinheit (1 baud = 1/s).
- *Zweiwertiges Signal*: Übertragung eines Bits pro Signal (binär).
- *Mehrwertiges Signal*: Übertragung von mehreren Bits pro Signal (z.B. DIBIT, 2 Bit pro Signal, also quaternär, 4 Werte).
- *Übertragungsgeschwindigkeit*: baud $\cdot \log_2(n)$ Bit, mit n als Anzahl der diskreten Wertstufen pro Signal.

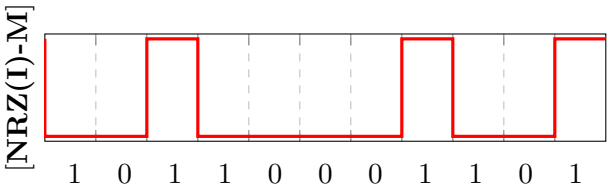
2.2 Leitungscodes

Laut Vorlesung wichtige Codes sind **fett** markiert.

2.2.1 Non-Return-to-Zero Codes

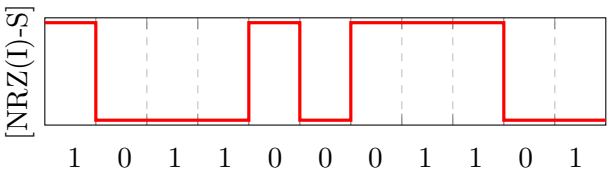


Bei 1: hoher Pegel.
Bei 0: niedriger Pegel.



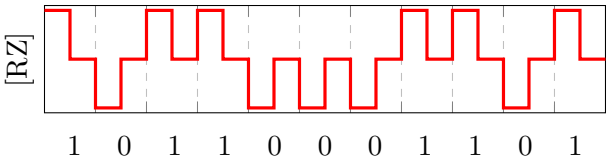
Auch als **Diff. NRZ** bezeichnet.

Bei 1: Pegel wechseln.
Bei 0: Pegel halten.

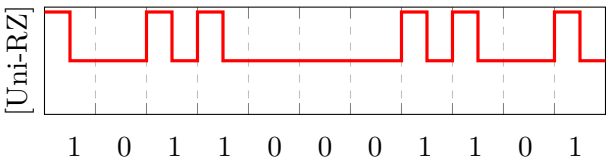


Bei 1: Pegel halten.
Bei 0: Pegel wechseln.

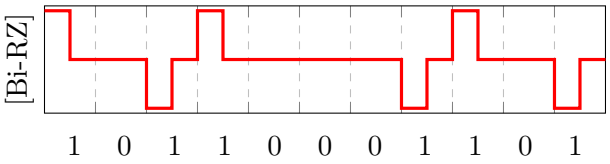
2.2.2 Return-to-Zero Codes



Bei 1: hoher Pegel, halten, dann wieder neutral, halten.
Bei 0: niedriger Pegel, halten, dann wieder neutral, halten.

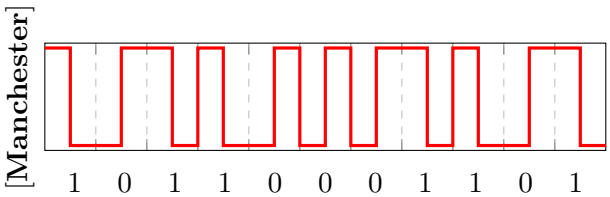


Bei 1: hoher Pegel, halten, dann wieder neutral, halten.
Bei 0: neutral halten.

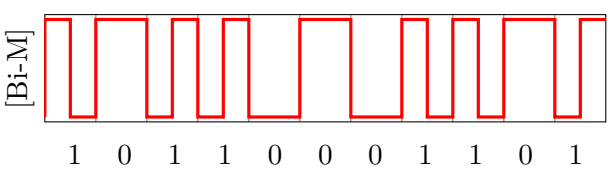


Bei 1: abwechselnd hohen bzw. niedrigen Pegel, halten, dann neutral, halten.
Bei 0: neutral halten.

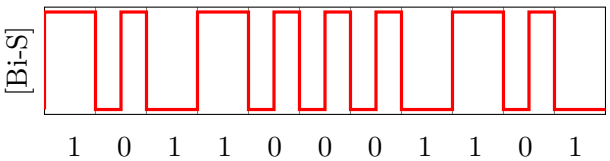
2.2.3 Biphase Codes



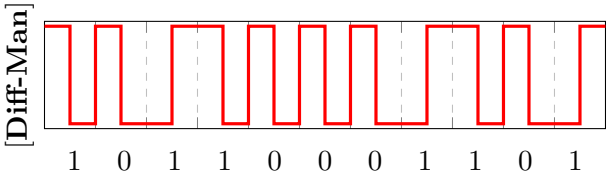
Bei 1: hoher Pegel, halten, dann niedriger Pegel, halten.
Bei 0: niedriger Pegel, halten, dann hoher Pegel, halten.



Bei 1: Pegel wechseln, Pegel halten, dann wechseln, halten.
Bei 0: Pegel wechseln, Pegel halten.



Bei 1: Pegel wechseln, Pegel halten.
Bei 0: Pegel wechseln, Pegel halten, dann wechseln, halten.



Bei 1: Pegel halten, dann wechseln, halten.
Bei 0: Wechseln, halten, dann wieder wechseln und halten.

2.2.4 4B/5B-Kodierung

Verwendet NRZI-M, aber kodiert Daten vorher durch Tabelle. 4B/5B überträgt 4 Bit in 5 Takten und erlaubt außerdem "Steuerwörter", wie `End` oder `Reset`.

2.2.5 Modulationsverfahren

Die obigen Codes waren Basisbandübertragungsverfahren, d.h. sie verwenden das gesamte Frequenzspektrum des Mediums. Für etwa Funkübertragung kann nur ein bestimmtes Frequenzspektrum verwendet werden. Im Allgemeinen nennt man die Beinflussung der Parameter des Trägersignals *Modulation*. Dabei gibt es drei Arten:

- *Amplitudenmodulation:* Übertragung durch Veränderung der Amplitude. Störanfällig.
- *Frequenzmodulation:* Veränderung der Übertragungsfrequenz.
- *Phasenmodulation:* Übertragung durch Phasensprünge. Bestes Verfahren.

Eine Kombination der Verfahren ermöglicht etwa die *Quadraturamplitudenmodulation* (QAM). 2^n-QAM kodiert n Symbole in einem Taktschritt und hat dann 2^n Zustände.

2.3 Theoreme von Nyquist und Shannon

Beide Theoreme beschreiben die maximale Datenrate bei einem Kanal mit eingeschränkter Bandbreite. Für einen Kanal muss das Minimum der Ergebnisse beider Theoreme für die maximale Bandbreite verwendet werden.

Nyquist sagt, für einen *störungsfreien* Kanal ist die maximale Datenrate in Bit pro Sekunde:

$$2 \cdot B \cdot \log_2(n),$$

wobei B die Bandbreite und n die Anzahl diskreter Signalstufen ist. Falls n-QAM verwendet wird, so ist n die Anzahl der diskreten Signalwerte.

Nach Shannon ist die maximale Datenrate bei einem Kanal mit *zufälligem Rauschen*:

$$B \cdot \log_2(1 + S/N)$$

S/N ist der Signal-Rausch-Abstand. Dieser wird oft in Dezibel angegeben, dann ist $S/N = 10^{x/10}$.

2.4 Konvertierung zwischen analogen und digitalen Werten

- *Abtastung* des analogen Signals in regemäßigen Intervallen (Zeitdiskretisierung).
- *Quantisierung* der abgetasteten Werte (Signalwertdiskretisierung).
- *Kodierung* ist Zurdnung eines Codes zu einem diskreten Wert.

Nach dem *Abtasttheorem* muss die Abtastrate mindestens doppelt so hoch sein wie die maximal relevante abzutastende Frequenz.

2.5 Betriebsarten eines Kanals

Bei Kanälen wird zwischen drei Betriebsarten unterschieden:

- *simplex:* Senden nur in eine Richtung möglich.
- *halbduplex:* Senden ist nur in eine Richtung erlaubt und Wechsel der Senderichtung ist möglich.
- *duplex:* Senden in beide Richtungen gleichzeitig möglich.

2.6 Multiplexing

Wenn das Problem auftritt, das mehrere Sender das selbe Medium zur gleichen Zeit benutzen wollen, so kann man dies mithilfe eines Multiplexverfahrens lösen. Man unterscheidet hierbei:

- **Raum**multiplex: Zusammenfassung mehrerer physikalischer Leitungen. Z.B. Bündelung von Adernpaaren in Kupferkabeln.
- **Zeit**multiplex: Der Zugriff auf das Medium erfolgt zeitlich versetzt, sodass zu jedem Zeitpunkt nur ein Sender über das Medium senden kann.
- **Frequenz**multiplex und **Wellenlängen**multiplex: Man teilt die Bandbreite auf die Sender auf.
- **Code**multiplex: Beim Codemultiplex können alle Sender gleichzeitig im gleichen Frequenzband senden. Die zu sendenden Nutzdaten werden mit einer für den Sender eindeutigen Pseudozufallszahl (der Chipping-Sequenz) XOR-verknüpft. So kann der Empfänger das Original rekonstruieren. Man beachte dass jedes Bit der Nutzdaten immer mit der kompletten Chipping-Sequenz verknüpft wird, und dass die Chipping-Sequenz auch bzgl. einer Invertierung eindeutig sein muss.

3 Sicherungsschicht

Die Sicherungsschicht soll garantieren, dass Daten vollständig und fehlerfrei übertragen werden. Dies teilt sich in verschiedene Anforderungen auf: *Synchronisation*, *Codetransparenz*, *Fehlererkennung- und Behandlung*, *Flusskontrolle*, *Koordinierter Medienzugriff* und *Steuerung.*

3.1 Bit Stuffing

Wenn eine bestimmte Bitfolge etwa zur Rahmenbegrenzung verwendet wird, muss sie natürlich in den Nutzdaten, falls sie vorkommt, ersetzt werden.

3.2 Übertragungsfehler

Die *Bitfehlerrate* ist: $\frac{\text{Summe gestörte Bits}}{\text{Summe übertragene Bits}}$.

3.2.1 Parität

Zur Fehlererkennung kann ein *Paritätsbit* an Daten angehangen werden. Bei *gerader Parität* ist die Anzahl der Einsen (einschließlich Parity-Bit) gerade. Äquivalent: Das Paritätsbit ist die verXORung der geprüften Bits[1]. *Ungerade Parität* analog.

Hat man mehrere Blöcke von Daten, kann zudem noch zwischen *Blockparität/Längsparität* (Parität der Bits eines Blocks) und *Zeichenparität/Querparität* (Parität von Bits gleicher Position in aufeinanderfolgenden Blöcken) unterschieden werden. Bei *Kreuzsicherung* werden beide Verfahren verwendet.

3.2.2 Cyclic Redudancy Check (CRC)

Es wird ein *Generatorpolynom* festgelegt. Die zu prüfende Bitfolge wird als Polynom aufgefasst (über $\mathbb{F}_2$). Dann ist die *Frame Check Sequence*, die Prüfzahl, der Rest der Division des Datenpolynoms durch das Generatorpolynom. Die Prüfzahl wird wieder als Bitfolge an die Daten angehangen.

Zur Validierung wird dann alles zusammen als Polynom interpretiert. Dieses Polynom muss nun ein Vielfaches des Generatorpolynoms sein; das heißt der Rest der Divison des Datenpolynoms durch das Generatorpolynoms muss 0 sein.

Illustrativ dazu ein Beispiel. Wir wollen 110011 senden und haben das Generatorpolynom als 11001 festgelegt.

Senden

```
110011 0000 % 11001 = 100001
11001
000001 0000
     1 1001
     0 1001 = Rest
```

Also ist zu übertragen: 110011 <u>1001</u>.

Empfangen

```
110011 1001 % 11001 = 100001
11001
000001 1001
     1 1001
     0 0000 = Rest
```

Also war die Übertragung wahrscheinlich fehlerfrei.

3.2.3 Vorwärtsfehlerkorrektur

Wenn zu übertragene Daten in mit Prüfsumme versehene Blöcke eingeteilt werden, kann ein zusätzlicher Block als XOR der anderen Blöcke zur Redundanz genutzt werden. Wenn bei (maximal) einem Block ein Fehler erkannt wird, kann mithilfe des Redundanzblocks und der anderen Blöcke der fehlerhafte Block rekonstruiert werden.

3.2.4 Hamming-Code

Der Hamming-Code erlaubt Fehlerkorrektur von einem Bit mit (bewiesen) minimalem Overhead.

Eine Bitfolge wird kodiert, indem an allen Positionen, die eine Zweierpotenz sind, ein Paritätsbit eingefügt wird. Ein Paritätsbit an der Position 2^{n-1} prüft dann alle Bits an Positionen mit Bit n auf 1 (dies beinhaltet dann auch die Paritätsbits selber).

[1] $p \oplus d_1 \oplus \ldots \oplus d_n = 0 \iff p = d_1 \oplus \ldots \oplus d_n$

Ein Empfänger prüft die Paritätsbits und kann durch Aufsummierung der falschen Paritätsbits die Position des falschen Datenbits errechnen.

3.3 Fehlerbehebung

- *Stop-and-Wait:* Sende einen Rahmen und warte auf eine Bestätigung (`ACK` für "Acknowledgement"), bevor den nächsten Rahmen gesendet werden darf. Ein Empfänger kann auch `NAK`s ("No Acknowledgement") senden, wenn er fehlerhafte Rahmen empfangen hat. Ebenfalls kann der Sender nach einem Timeout den Rahmen erneut senden.
- *Go-Back-N:* Der Empfänger kann verlangen, ab Rahmen n alle folgenden Rahmen neu senden zu lassen.
- *Selective Repeat:* Der Sender wiederholt nur einzelne Rahmen.

Stop-and-Wait ist schlecht. Go-Back-N sollte nur bei Leitungen mit niedriger Latenz, niedriger Datenrate, oder keinem Empfangspuffer verwendet werden. Selective Repeat ist gut.

3.4 Flusskontrolle

Ein *Sliding Window* begrenzt die Rahmen, die auf einmal ohne Bestätigung gesendet werden dürfen. Ist das Sendefenster voll, so muss auf eine Bestätigung gewartet werden, bevor ein Rahmen gesendet werden darf.

3.5 Medienzugriff

Verfahren:

- Frequenzmultiplex
- Zeitmultiplex
  - synchron
  - asynchron
- Codemultiplex

Asynchrone Synchronisierung:

- Konkurrierender Zugriff: Eine Station prüft, ob sie senden kann, und sendet dann. Kollisionen sind dann möglich.
- Geregelter Zugriff: Sendeberechtigung wird vorher aufgeteilt.
  - Zentrale Zuteilung (etwa HDLC)
  - Zyklische Zuteilung (etwa Token Bus, Token Ring, FDDI)
  - Resilient Packet Ring

3.5.1 Zentrale Protokolle

- *Polling:* Ein zentraler Rechner fragt alle angeschlossenen Stationen regelmäßig, ob sie senden wollen und entscheidet dann, wer senden darf.
- *Abfragen mit gemeinsamer Busleitung:* Wenn ein gesonderter Kanal verfügbar ist, kann dieser genutzt werden, um einen Sendewunsch mitzuteilen.

3.5.2 Dezentrale Protokolle

- *Konkurrierender Zugriff:* Jeder sendet, wann er will (siehe ALOHA, CSMA/CD).
- *Token Passing:* Das Senderecht wird zyklisch unter den Stationen durchgereicht (siehe Token Bus, Token Ring, FDDI)
- *Daisy-Chaining:* Gesonderter Kanal, der einen zukünftigen Zeitslot reserviert.

3.5.3 HDLC

Das *High-Level Data Link Control* Protokoll wurde für den Anschluss von Datennetzen über Telefonnetze entworfen. Es unterstüzt Framing, Fehlerkorrektur (mit CRC), Flusskontrolle (mit Sliding Window) und Zeitmultiplexing.

Der Medienzugriff hat mehrere Betriebsarten:

- *Aufforderungsbetrieb* ("NRM"), d.h. Polling.
- *Spontanbetrieb* ("ARM"): Eine Folgestation darf jederzeit an die Leitstation senden.
- *Gleichberechtigter Spontanbetrieb* ("ABM"): Stationen sind gleichberechtigt ("Hybridstationen") und dürfen jederzeit senden.

3.5.4 LLC

Das *Logical Link Control* Protokoll liegt auf IEEE-Schicht 2b und ergänzt damit die 2a-Schicht um Flusskontrolle, zuverlässige Übertragungen durch Bestätigungen und ein einheitliches Interface auf die Netzwerkschicht.

3.5.5 MAC-Protokolle

Die MAC-Protokolle sind hardware-abhängig und stellen Medienzugriff und Fehlererkennung sowie Fehlerkorrektur bereit.

Token Ring (IEEE 802.5):
Physikalisch durch einen Ring verbundene Stationen reichen zyklisch einen Token herum, der dem aktuellen Inhaber das Senden erlaubt. Daten werden durch den Ring weitergeleitet, bis sie wieder beim Sender (der Empfänger leitet auch weiter!) ankommen. Danach gibt der Sender das Senderecht weiter.
Es gibt einen sogenannten *Monitor*, der Fehler wie Tokenverlust oder doppelte Token erkennt und korrigiert. Es gibt eine maximale Sendedauer ("Token Holding Time"), standardmäßig 10 ms.

+ Hohe Effizienz unter hoher Last
+ Erkennung von Kabelbruch
+ Echtzeitbetrieb möglich
− Komplexe Fehlerfälle
− Unnötige Verzögerungen bei niedriger Last

CSMA/CD (IEEE 802.3):
Wer senden will, hört vorher das Medium ab und prüft, ob es frei ist ("Listen before Talk"). Kollisionen sind so insbesondere auf (räumlich) kleinen Netzen unwahrscheinlich.
Außerdem wird während des Sendens geprüft, ob jemand anderes sendet. Wenn ja, dann wird die Leitung durch ein Jamming-Signal kurz blockiert und alle Sender müssen aufhören, zu senden ("Listen while Talk").

+ Einfaches Protokoll
+ Geringe Verzögerung bei geringer Last
− Kein Echtzeitbetrieb möglich
− Viele Kollisionen bei höherer Last

Ethernet ist eine Implementierung von CSMA/CD.

CSMA/CD ist aus (Slotted) ALOHA hervorgegangen:

- *ALOHA:* Wer senden möchte, sendet. Kollisionen können da mal schnell passieren. Angeblich ist der maximale Datendurchsatz bei 18%.
- *Slotted ALOHA:* Das Senden wird nur in bestimmten Zeitslots erlaubt. Durchsatz: 36%.

3.6 Ethernet

Ethernet ist eine Protokollfamilie, die den Physical-Layer und den MAC-Layer umfasst.

Interessant sind für uns in diesem Kontext *Bridges:* Diese trennen Kollisiondomänen auf unterschiedlichen Segmenten. Solche Bridges sind *Repeater*, die nur das Signal wiederauffrischen. *Hubs* haben mehrere Ports und leiten Daten von einem Port auf alle anderen weiter.

Die spannendsten Bridges sind aber *Switches:* Diese lernen MAC-Adressen von angeschlossenen Stationen und stellen Punkt-zu-Punkt-Verbindungen bereit. Wenn ein Rahmen auf Port p ankommt, macht der Switch folgendes:

1. Empfangen: Speichere (und überschreibe ggf.) für die Absenderadresse den Port p in der Weiterleitungstabelle.
2. Senden: Es wird geprüft, ob schon ein Eintrag für die Empfängeradresse in der Weiterleitungstabelle existiert. Gibt es diesen und ist dieser nicht Port p, so wird der Rahmen dorthin weitergeleitet. Ist der Eintrag Port p, wird der Rahmen verworfen. Gibt es keinen Eintrag, so wird der Rahmen an alle Ports außer p gebroadcastet.

4 Vermittlungsschicht

Die Vermittlungsschicht ist dafür zuständig, Daten zwischen zwei Kommunikationspartnern über Netzgrenzen zu vermitteln.

- *Circuit Switching:* Eine Leitung wird zwischen zwei Stationen geschaltet.
- *Store & Forward:* Vermittlungsstellen puffern und leiten weiter, wenn die Leitung frei ist. Datenverlust und falsche Zustellungsreihenfolge sind so möglich.
- *Virtuelle Verbindung:* Eine Verbindung wird aufgebaut, sodass den beiden Kommunikationspartnern es so vorkommt, als würde eine feste Verbindung existieren.

Das *ATM*-Protokoll kombiniert Leitungs- und Paketvermittlung durch Verbindungs-IDs, die in Weiterleitungstabellen auf Routern (für eine Verbindung) immer die gleiche Route festlegen.

4.1 IP

Das *Internet Protocol* ist unzuverlässig und verbindungslos. Pakete (Datagramme) können verloren, dupliziert, in falscher Reihenfolge oder endlos wiederholt werden. Außerdem existiert keine Fluss- oder Staukontrolle. Dies wird darüber liegenden Protokollen überlassen.

IP-Adressen sind eindeutig und bei IPv4 32 bit lang, bei IPv6 128 bit. Ein Rechner besitzt eine IP-Adresse für jedes Interface.

Adressen werden hierarchisch in *Subnetze* aufgeteilt. Dazu verendet man "Netzwerk-IDs", gefolgt von der Länge der Einsen in der Subnetzmaske, die angeben, welcher Teil der Adresse Netzwerk-ID ist.

4.1.1 Weiterleitungstabelle

Für die Übertragung eines Paketes wird eine Weiterleitungstabelle verwendet, die angibt, "in welche Richtung" ein Paket gesendet werden muss, um beim Empfänger anzukommen. Diese speichert die folgenden Daten für jeden Eintrag:

- Die Zieladresse.
- Einen *Gateway*, die Adresse des nächsten Routers, wenn das `G`-Flag gesetzt ist. Sonst ist dieser Eintrag die Adresse des lokalen Netzadapters.
- Eine *Netmask*, die Subnetzmaske für die Zieladresse.
- *Flags:* `G` wenn der Knoten ein Router (Gateway) ist, sonst `H` (Host). `U` (UP) für Verfügbarkeit. `S` für statische Route.
- Das Interface, auf dem abgesendet wird.

Wenn mehere Einträge der Weiterleitungstabelle zu einer Adresse passen, wird der *Longest Prefix Match* durchgeführt, d.h. es wird der Eintrag mit der längsten passenden Subnetzadresse verwendet. Wird keiner gefunden, wird 0.0.0.0/0 ("default") verwendet.

4.1.2 NAT

Rechner in privaten Netzen wollen mit dem Internet kommunizieren, aber haben nur private Adressen. Also muss ein Router eine lokale Adresse einer globalen (äußeren) Adresse zuordnen, mit der der Rechner nach außen kommunizieren kann.

Router mit *NAT-Overloading* speichern Tupel (Protokoll[2], lokale IP, lokaler Port, globale IP, globaler Port) und können so auch wieder ankommende Pakete an den lokalen Rechner weiterleiten.

4.1.3 IP-Header (IPv4)

Ein IPv4-Paket kann in mehrere Datagramme ("Fragmente") aufgeteilt werden. Ein IPv4-Datagramm beginnt immer mit einem Header. Dieser enthält unter anderem die folgenden Felder:

- *IHL:* Die Länge des Headers als Vielfaches von 4 Byte.
- *Total Length:* Die Gesamtlänge des Datagramms.
- Quell- und Zieladressen. (Nicht aber etwa Ports, die sind Aufgabe von TCP/UDP).
- *Time-to-Live:* Lebenszeit des Datagramms, die (in der Praxis) von jedem Router um 1 reduziert wird. Erreicht sie 0, wird das Paket verworfen.
- Eine Prüfsumme des Headers.
- *Identification:* Eine eindeutige Kennzeichnung des Pakets.
- *Don't Fragment (DF):* Ob das Paket nicht fragmentiert werden darf.
- *More Fragments (MF):* Ob noch weitere Fragmente eines Pakets folgen oder ob dieses das letzte ist.
- *Fragment Offset:* Vielfaches von 8 Byte, das angibt, zu welcher Stelle im Datenteil des Pakets dieses Fragment gehört.

Die Fragmentierung gibt es, weil für Netze eine Maximallänge von Datagrammen existieren kann, die *MTU* (Maximum Transfer Unit), etwa 1500 B bei Ethernet. Ein fragmentiertes Paket kann so vom Empfänger anhand der Identification und der Offsets wieder zusammengesetzt werden.

4.2 ARP

Das *Address Resolution Protocol* kümmert sich darum, IP-Adressen zu Schicht-2-Adressen (MAC-Adressen) zu übersetzen. Wenn eine IP-Adresse a übersetzt werden soll, findet bei ARP eine Auflösung wie folgt statt:

1. Broadcast im LAN, das nach a fragt.
2. Alle Empfänger des Broadcasts speichern sich MAC- und IP-Adressen des Senders. Nur der Besitzer von a antwortet mit seiner MAC-Adresse.
3. Der Anfragende speichert a in seinem Cache. Der Cache muss irgendwann gereinigt werden.

[2] TCP oder UDP

4.3 Routing

Beim Routing geht es darum zu entscheiden, an wen Pakete gesendet werden müssen, damit sie am Ziel ankommen. Eine Routingtabelle hält Zuordnungen von Zielen und zugehörigen Links "in der Richtung" des Ziels vor.

Routing-Verfahren kann man in zwei Dimensionen einteilen: Zentralisation und Dynamik. Weniger *Zentralität* bürgt den einzelnen Knoten mehr Aufgaben beim Routing auf. Mehr *Dynamik* macht öftere Anpassungen an sich ändernde Netzzustände wie Auslastung möglich.

- *Statisches Routing:* Routing-Tabellen haben einen Eintrag für jeden möglichen Zielknoten. Es gibt mehrere Routingmöglichkeiten, die gewichtet sind. Eine Möglichkeit wird durch das Ziehen einer Zufallszahl mithilfe der Gewichtungen gewählt.
- *Zentralisiertes Routing:* Ein "Routing Control Center" gibt den anderen Knoten Tabellen vor.
- *Isoliertes Routing:* Routing ausschließlich anhand eigener Informationen.
  - *Flooding:* Jedes Paket wird auf jeder Leitung (außer der Herkunftsleitung) weitergeleitet.
  - *Backward Learning:* Pakete müssen einen Hop-Count enthalten (etwa TTL bei IP). Bei Empfang eines Paketes weiß dann der Empfänger, wie viele Hops die Route zum Sender brauchte.
- *Dynamisches Routing:* Entscheidungen mithilfe von Informationsaustausch zwischen Routern.
  - *Distance Vector Routing:* Router tauschen Kosten einer Übertragung von sich zu einem Ziel untereinander aus. Dann versucht jeder Router, die selbst eingetragenen Kosten zu verringern, indem über einen (anderen) Hop gesendet wird. Probleme hier sind langsame Konvergenz sowie Count-to-Infinity.
  - *Link State Routing:* Nachbarn messen Kosten einer Übertragung untereinander und senden diese Informationen dann an alle anderen Router. Dann kann jeder Router etwa mit dem Dijkstra-Algorithmus die beste Route im so entstandenen gewichteten Graph aller Verbindungen berechnen.

5 Transportschicht

Das *Transport Control Protocol* (TCP) ist ein verbindungsorientiertes Protokoll, welches zuverlässige Übertragung garantiert. Das *User Datagram Protocol* (UDP) ist einfach nur eine Anwendungsschnittstelle über IP, garantiert also weder korrekte Reihenfolge, noch Zuverlässigkeit, noch dass keine Duplikate auftreten - genau wie IP. TCP und UDP erlauben aber beide Multiplexing durch Ports.

Ports sind 16-Bit Zahlen, wobei die ersten 1023 für Standarddienste reserviert sind. *Sockets* sind dann eine IP-Adresse und ein Port.

5.1 TCP

- Passt sich an das unterliegende Netz dynamisch an (etwa schwankende Bandbreiten).
- Fehlerkontrolle: Sequenznummern zum sortieren beim Empfänger, Timeouts bei ausbleibender Quittung beim Sender.
- Flusskontrolle verhindert Überlastung des Empfängers.
- Staukontrolle verhindert Überlastung des Netzes.

Verbindungen können bei TCP entweder aktiv oder passiv aufgebaut werden. *Clients* bauen ihre Seite der Verbindung *aktiv* auf, während der *Server passiv* auf eine eingehende Verbindung wartet.

TCP zerlegt einen Bytestrom in *Segmente*, die über IP übertragen werden. Mithilfe von Sequenznummern wird ein Segment vom Empfänger durch Quittungen bestätigt und einsortiert. Empfängt der Sender nach einer bestimmten Zeit keine Quittung, wird das Segment neu versendet (oder die Verbindung wird als abgebrochen betrachtet).

5.1.1 Verbindungsmanagement

Die meisten übertragenen Segmente enthalten eine Sequenznummer (`SEQ`) für die eigenen gesendeten Segmente und eine Bestätigungsnummer (`ACK`), die dem Empfänger den Empfang eines seiner Segmente mit der angebenen Sequenznummer signalisiert.

Ein normaler Verbindungsaufbau funktioniert wie folgt, hier sind die Segmenttypen `SYN` und `ACK` relevant.

1. Client $\overset{\texttt{SYN}}{\rightarrow}$ Server. $\texttt{SEQ} = x$.
2. Client $\overset{\texttt{SYN}}{\leftarrow}$ Server. $\texttt{SEQ} = y$, $\texttt{ACK} = x + 1$.
3. Client $\overset{\texttt{ACK}}{\rightarrow}$ Server. $\texttt{ACK} = y + 1$, $\texttt{SEQ} = x + 1$.

Es ist auch ein "ungeregelter" Verbindungsaufbau möglich, wobei die ersten zwei Schritte von beiden Endpunkten initiiert werden. Nach dem `ACK` ist dann die TCP-Verbindung aufgebaut.

Datenaustausch passiert dann ähnlich; es werden `Data` Segmente ausgetauscht, die dann `SEQ` und `ACK` enthalten. Es ist zu bemerken, dass in Serverrichtung das `ACK`-Feld nicht streng nötig ist, aber in der Praxis immer gesendet wird.

Ein Verbindungsende wird durch das Senden von `FIN`-Segmenten signalisiert. Daraufhin antwortet der Empfänger ebenfalls mit einem `FIN`, bestätigt aber auch das empfangene `FIN` durch ein `ACK`-Segment, welches dann noch einmal bestätigt wird. Dann ist die Verbindung geschlossen.

Für das Senden und Empfangen wird das bekannte Sliding-Window-Verfahren benutzt. Dabei kann bei einfacher Implementation das *Silly-Window-Syndrom* auftreten, wobei wegen eines fast vollen Empfängerbuffers nur wenig Daten gesendet werden, sodass der TCP-Overhead die eigentlichen Daten überwiegt.

5.1.2 Fluss- und Staukontrolle

Die *Flusskontrolle* bei TCP regelt den Datenfluss zwischen Instanzen, d.h. limitiert die gesendeten Daten, wenn der Empfänger nicht mehr empfangen kann. Dies kann der Sender anhand der vereinbarten Fenstergröße und der bestätigten Pakete schätzen. Diese Begrenzung heißt *Receiver Window* (`rwnd`).

Wenn Daten verloren gehen, nimmt die *Staukontrolle* eine Überlastung des Netzes an. Die geschätzte Datenrate gibt das *Überlastfenster* vor, *Congestion Window* (`cwnd`) im TCP-Lingo.

Die Datenrate der gesendeten Pakete wird dann durch das *Sendefenster* (`swnd`) limitiert. `swnd` = min(`rwnd`, `cwnd`).

Slow-Start-Algorithmus `cwnd` wird auf 1 initialisiert. Nach jeder Bestätigung wird `cwnd` um die Größe eines Segments erhöht. Damit wächst `cwnd` bei einer erfolgreichen Übertragung eines vollen Fensters exponentiell. Wird ein Schwellwert `ssthresh` erreicht, wird `cwnd` nur noch inkrementiert (lineares Wachstum). Bei Paketverlust wird `ssthresh` auf $1/2 \cdot$ `swnd` gesetzt und `cwnd` = 1MSS.

Fast Retransmit Der Slow-Start-Algorithmus reagiert bei einzelnen Paketverlusten über und reduziert die Datenrate zu stark. Bei Fast Retransmit wird jedes erhaltene Segment sofort quittiert; ein Segment mit falscher Sequenznummer löst das wiederholte Senden eines `ACK`s aus (`DUP-ACK`). Wenn drei `DUP-ACK`s empfangen werden, wird das Segment neu gesendet.

Wenn das wiederholte Segment erfolgreich empfangen wurde, werden wieder reguläre Quittungen gesendet. Tritt aber ein Timeout ein, so wird auf Slow Start zurückgefallen.

Fast Recovery Verbesserung von Fast Retransmit. Wenn das dritte `DUP-ACK` empfangen wird:

- `ssthresh` = max(`swnd`$/2, 2 \cdot$ MSS).
- Das fehlende Segment wird wieder gesendet.
- `cwnd` = `sstresh` $+ 3 \cdot$ MSS.
- Für jedes weitere `DUP-ACK`: `cwnd` += MSS.
- Wird ein normales `ACK` empfangen: `cwnd` = `sstresh` und mache normal weiter.

5.2 UDP

UDP-Pakete haben nur einen 8 B großen Header: Sie enthalten Quell- und Zielports, die Nachrichtenlänge und eine (optionale) Checksumme. Mehr gibt es zu UDP eigentlich nicht zu sagen.

6 Sicherheit

Um Sicherheit von Daten und Infrastruktur zu gewährleisten, bedient man sich einer Menge an Werkzeugen: *Firewalls*, *Intrusion Detection*, *Authentifizierung*, *Verschlüsselung* und *Integritätsprüfung*,

6.1 Verschlüsselung

- *Symmetrische Verschlüsselung:* Gleicher Schlüssel zum Ver- und Entschlüsseln.
- *Asymmetrische Verschlüsselung:* Zwei verschiedene Schlüssel.

Wir werden in diesem Kapitel die meisten wirklichen Algorithmen auslassen und uns auf wichtige Konzepte konzentrieren.

6.1.1 Symmetrische Verschlüsselung

Eine *Blockchiffre* verschlüsselt Blöcke einer festen Länge gleichzeitig. Jeder der Blöcke wird mit dem gleichen Algorithmus verschlüsselt.

Mit *Cipher Block Chaining* wird jeder Block mit dem Ciphertext des vorherigen Blocks XOR-verknüpft. Für den ersten Vektor muss dann ein Zufallswert, ein sogenannter *Initialization Vector* (IV) gewählt werden. Für jede neue Nachricht sollte ein neuer IV gewählt werden.

Ein *Message Authentication Code* (MAC) ist eine digitale Signatur, die garantiert, dass bei verifizierter MAC der Ciphertext nicht verändert wurde.

6.1.2 Schlüsselaustausch

Schlüssel auszutauschen ist natürlich unsicher, weil man bei dem Schlüsselaustausch keine gesicherte Verbindung hat.

Diffie-Hellman

1. Wähle Primzahl p und Generator $g < p$. Diese sind öffentlich.
2. A wählt zufällig S_a und berechnet $T_a = g^{S_a} \mod p$.
   B wählt zufällig S_b und berechnet $T_b = g^{S_b} \mod p$.
3. A und B tauschen T_a und T_b aus.
4. A berechnet $k = T_b^{S_a} \mod p$. Analog für B.

Dann ist k ein geheimer Schlüssel, den man praktisch nicht aus den öffentlichen Werten p, g, T_a und T_b berechnen kann.

Natürlich kann man aber Diffie-Hellman durch einen Man-in-the-Middle-Angriff kompromittieren, wenn A und B nicht authentifiziert sind.

6.1.3 Asymmetrische Verschlüsselung

RSA

1. Wähle zwei große Primzahlen p und q.
2. Berechne $n = p \cdot q$.
3. Berechne $\phi(n) = (p-1) \cdot (q-1)$.
4. Wähle eine Zahl e teilerfremd zu $\phi(n)$.
5. Finde ein d mit $d \cdot e = 1 \mod \phi(n)$.

Dann ist (e, n) der öffentliche Schlüssel und (d, n) der private Schlüssel.

Ver- und Entschlüsselung	**Digitale Signatur**
Nachricht m verschlüsseln: $c = m^e \mod n$	Signieren: $s = m^d \mod n$
Nachricht c entschlüsseln: $m = c^d \mod n$	Verifikation: $m = s^e \mod n$

6.2 Authentifizerung

Das *Challenge/Response*-Verfahren authentifiziert etwa mit RSA, indem die sich zu authentifizierende Person zufällig vom Partner gewählte Daten signiert. Die Daten müssen also noch nie verwendet sein, deshalb nennt man sie auch *Nonce* (Number used once).

Ein *Key Distribution Center* (KDC) verteilt auf Anfrage für eine Sitzung zwei Schlüssel an die Partner. Dieses KDC ist dann ein Single-Point-of-Failure.

Certificate Authorities sind vertrauenswürdige Stellen, die *Zertifikate* signieren, sodass zu einem Zertifikat ein Inhaber sicher zugeordnet werden kann.