Hochschule Bochum
Bochum University
of Applied Sciences

Vertiefung Rechnertechnik und -netzwerke

Sommersemester 2013

Prof. Dr. Peter Gerwinski

Inhaltsverzeichnis

Stand: 22. Juni 2013

Text: Copyright © 2012, 2013 Peter Gerwinski
Lizenz: CC-by-sa (Version 3.0) oder GNU GPL (Version 3 oder höher)

Bild S. 14: Ethernet-Paket
http://de.wikipedia.org/wiki/Datei:Ethernetpaket.svg
Copyright © 2009 Wikipedia-Autor *Bluepoke*
Lizenz: CC-by-sa (Version 3.0) oder GNU FDL (Version 1.2 oder höher)

Bilder S. 23: TP-Kabel
http://de.wikipedia.org/wiki/Datei:TwistedPair_S-FTP.jpg
http://de.wikipedia.org/wiki/Datei:TwistedPair_F-UTP.jpg
Copyright © 2008 Wikipedia-Autor *Hurzelchen*
Lizenz: CC-by-sa (Version 3.0) oder GNU FDL (Version 1.2 oder höher)

Bild S. 25: FDDI-Ring
http://de.wikipedia.org/wiki/Datei:FDDI-Ring.svg
Copyright © 2009 Wikipedia-Autor *Frank Murmann*
Copyright © 2011 Wikipedia-Autor *Hk kng*
Lizenz: vom Autor als gemeinfrei („fehlende Schöpfungshöhe“) freigegeben

Bild S. 26: Hidden Station
http://de.wikipedia.org/wiki/Datei:Hidden_Station.svg
Copyright © 2007 Wikipedia-Autor *Bitbert*
Lizenz: CC-by-sa (Version 2.5)

Bild S. 26: Exposed Station: modifizierte Version von Hidden Station (siehe oben)

Bild S. 34: Tempo vs. Länge
http://de.wikipedia.org/wiki/Datei:Tempo-vs-Länge.svg
Copyright © 2010 Wikipedia-Autor *Biktora*
Lizenz: CC-by-sa (Version 3.0) oder GNU FDL (Version 1.2 oder höher)

Bild S. 18: RS-232: neu erstellt auf Grundlage von:
http://de.wikipedia.org/wiki/Datei:RS-232_timing.png
Copyright © 2005 Wikipedia-Autor *Gerald.deppe*
Lizenz: vom Autor als gemeinfrei („fehlende Schöpfungshöhe“) freigegeben

Bild S. 18: I^2C
http://de.wikipedia.org/wiki/Datei:I2C_data_transfer.svg
Copyright © 2007 Wikipedia-Autor *Mfloryan*
Lizenz: vom Autor als gemeinfrei freigegeben

Bild S. 18: Akustikkoppler
http://commons.wikimedia.org/wiki/File:AcousticCouplerAK2000Tel.jpg
Copyright © 2011 Wikipedia-Autor *Hubert Berberich (HubiB)*
Lizenz: vom Autor als gemeinfrei freigegeben

Bild S. 20: Wellen, Wavelets, Chirps und Chirplets
http://en.wikipedia.org/wiki/File:Wave-chirp-wavelet-chirplet.png
Copyright © 2004 Wikipedia-Autor *Glogger*
Lizenz: CC-by-sa (Version 3.0) oder GNU FDL (Version 1.2 oder höher)

Bild S. 45: TCP-Header: neu erstellt auf Grundlage von:
http://de.wikipedia.org/wiki/Datei:TCP_Header.svg
Copyright © 2007 Wikipedia-Autor *Appaloosa*
Lizenz: CC-by-sa (Version 3.0) oder GNU FDL (Version 1.2 oder höher)

Bild S. 46: Zwei-Armeen-Problem
http://de.wikipedia.org/wiki/Datei:2-generals.svg
Copyright © 2011 Wikipedia-Autor *Jens Erat*
Lizenz: CC-by-sa (Version 1.0, 2.0, 2.5 oder 3.0) oder GNU FDL (Version 1.2 oder höher)

Bild S. 47: TCP-Handshake
http://de.wikipedia.org/wiki/Datei:Tcp-handshake.svg
Copyright © 2010 Wikipedia-Autor *Snubcube*
Copyright © 2005 Wikipedia-Autor *Denniss*
Copyright © 2005 Wikipedia-Autor *Caos*
Lizenz: CC-by-sa (Version 3.0) oder GNU FDL (Version 1.2 oder höher)

Alle anderen Bilder: Copyright © 2012, 2013 Peter Gerwinski
Lizenz: CC-by-sa (Version 3.0) oder GNU GPL (Version 3 oder höher)

Sie können dieses Skript herunterladen unter:
http://www.peter.gerwinski.de/download/net-2013ss.tar.gz

1 Einführung: TCP/IP in der Praxis

Dieser Abschnitt soll in einem schnellen „Rundumschlag“ die Bedeutung der Grundlagen von TCP/IP-Netzwerken für die Praxis vermitteln.

1.1 Verkabelung

Damit Rechner in einem gemeinsamen Netzwerk „miteinander sprechen“ können, müssen sie zunächst korrekt miteinander verbunden sein.

In der kabelgebundenen Variante geschieht dies heutzutage (2013) in der Regel mittels Twisted-Pair-Kabeln, die entweder alle Rechner mit einem Switch (früher: Hub) verbinden oder direkt zwei Netzwerkkarten miteinander.

1.2 Automatismen abschalten

Um das Netzwerk konfigurieren zu können, müssen wir als Benutzer `root` (Systemadministrator) angemeldet sein.

Auf modernen Unix-artigen Betriebssystemen laufen oft Dienste, die sich um die Einrichtung von Netzwerken weitgehend automatisch kümmern. Um das manuelle Einrichten von Netzwerken üben zu können, müssen wir diese Dienste abschalten. Andernfalls werden diese Dienste bestimmungsgemäß die von uns manuell geänderte Netzwerkkonfiguration wieder auf vorher eingestellte Werte zurücksetzen.

- **Network-Manager**

  Dieser Dienst wird über die System-V-Init-Skripte gesteuert
  und kann mit `/etc/init.d/network-manager stop`
  oder `service network-manager stop`
  oder einem ähnlichen Befehl beendet werden.

- **DHCP-Client**

  Ein laufender DHCP-Client kann durch Suche in der Prozeßliste erkannt werden:

  ```
  # ps aux | grep dh
  root     7719  0.0  0.0   3528  1048 ?        S    Apr06   0:05 dhclient
  root    12000  0.0  0.0   3896   836 pts/155  S+   19:33   0:00 grep dh
  #
  ```

  Anschließend können wir ihn über seine Prozeß-Nummer beenden:

  ```
  # kill 7719
  #
  ```

1.3 IP-Adressen vergeben: ifconfig

Mit dem Befehl `ifconfig` können wir die Konfiguration der Netzwerk-Schnittstellen auslesen und verändern.

`ifconfig -a` zeigt sämtliche Netzwerk-Schnittstellen an:

```
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:80:c9:00:2f:31
          inet addr:192.168.1.23  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21a:80ff:fed0:75a8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1167973 errors:0 dropped:2 overruns:0 frame:0
          TX packets:426928 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1644843136 (1.5 GiB)  TX bytes:41314524 (39.4 MiB)
          Interrupt:16
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6914 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6914 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:292448 (285.5 KiB)  TX bytes:292448 (285.5 KiB)

#
```

Ohne die Option `-a` werden nur die momentan aktiven Netzwerkkarten angezeigt.

`eth0` bezeichnet eine physikalische kabelgebundene Netzwerkkarte; `lo` bezeichnet eine – stets vorhandene – virtuelle Netzwerkkarte, über die der Rechner auf ein Netz zugreifen kann, das nur aus ihm selbst besteht.

Auf beiden Netzwerkkarten hat der Rechner eine Nummer, die ihn für das gesamte erreichbare Netz eindeutig kennzeichnet: die *IP-Adresse*. („IP“ steht für „Internet Protocol“.) In diesem Beispiel lautet die IP-Adresse in dem kabelgebundenen Netz, das über `eth0` erreichbar ist, `192.168.1.23`. Die IP-Adresse „für Selbstgespräche“ ist standardisiert und normalerweise immer `127.0.0.1`.

Um ein Interface zu deaktivieren, lautet der Befehl: `ifconfig eth0 down`.
Um es zu aktivieren, lautet er: `ifconfig eth0 up`.

Um einem Interface eine IP-Adresse (hier: `10.1.1.9`) zuzuweisen, lautet der Befehl:

```
ifconfig eth0 10.1.1.9
```

Dieser Befehl kann auch mit dem Aktivieren kombiniert werden: `ifconfig eth0 up 10.1.1.9`.

Die folgenden IP-Adressen stehen für private Zwecke (z. B. internes Firmen-Netz – *Intranet*) zur Verfügung:

- `10.x.y.z`
- `172.16.x.y` bis `172.31.x.y`
- `192.168.x.y`

Die meisten anderen IP-Adressen sind öffentlich und haben einen Besitzer, sollten also – auch nicht zu Testzwecken – ohne Erlaubnis verwendet werden.

1.4 Erreichbarkeit prüfen: ping

Sobald die Rechner korrekt angeschlossen und die Rechner IP-Adressen im gleichen Netz haben, kann man mit dem Befehl `ping` die gegenseitige Erreichbarkeit prüfen. (Was „im gleichen Netz“ bedeutet, wird im Abschnitt 1.8 – Routing – definiert.)

```
# ping 192.168.1.137
PING 192.168.1.137 (192.168.1.137) 56(84) bytes of data.
64 bytes from 192.168.1.137: icmp_seq=1 ttl=64 time=0.368 ms
64 bytes from 192.168.1.137: icmp_seq=2 ttl=64 time=0.397 ms
64 bytes from 192.168.1.137: icmp_seq=3 ttl=64 time=0.424 ms
64 bytes from 192.168.1.137: icmp_seq=4 ttl=64 time=0.420 ms
^C
--- 192.168.1.137 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.368/0.402/0.424/0.026 ms
#
```

Der Befehl `ping` beendet sich normalerweise nicht selbst, sondern man unterbricht ihn mit der Tastenkombination Strg+C.

1.5 Hardware- und IP-Adressen: arp

In der Ausgabe von `ifconfig` wird zu jeder physikalischen Netzwerkkarte eine sog. *Hardware-Adresse* angezeigt. Diese besteht aus sechs hexadezimal notierten Bytes (Zahlen von 0 bis 255) und kennzeichnet eine Netzwerkkarte weltweit eindeutig.

Aus verschiedenen Gründen ist es sinnvoll, die Rechner über eine zweite Nummer, die IP-Adresse, nochmals durchzunumerieren:

- Die Netzwerkkarte eines Rechners kann – z. B. bei einem Defekt – ausgetauscht werden. Man möchte in solche einem Fall nicht sämtlichen Kommunikationspartnern eine neue Hardware-Adresse mitteilen müssen.
- Die Erreichbarkeit einer Hardware-Adresse ist auf das Kabel begrenzt. Um einen Rechner über das Kabel hinaus auffinden zu können, werden IP-Adressen bitweise maskiert – siehe Abschnitt 1.8: Routing. Dies kann mit Hardware-Adressen, die herstellerabhängig vergeben werden, nicht funktionieren.

Der Betriebssystemkern merkt sich eine eindeutige Zuordnung zwischen Hardware- und IP-Adressen in der sog. *ARP-Tabelle*. („ARP“ steht für „Address Resolution Protocol“.) Diese kann mit Hilfe des Befehls `arp` ausgegeben und manipuliert werden:

```
# arp -n
Address          HWtype  HWaddress           Flags Mask    Iface
192.168.1.196    ether   00:1d:ba:bb:39:88   C             eth0
192.168.1.137    ether   f8:d1:11:02:64:e4   C             eth0
#
```

In diesem Fall listet die Tabelle die Hardware-Adressen zweier bekannter Kommunikationspartner mit den IP-Adressen `192.168.1.137` und `192.168.1.196`.

Das `-n` hinter `arp` steht für „numeric“ und bedeutet, daß `arp` keine Rechnernamen (siehe unten) ausgeben soll, sondern IP-Adressen.

1.6 Kommunikation per TCP/IP: Netcat (nc)

Ein Rechner wird durch seine IP-Adresse adressiert, ein konkreter Dienst auf dem Rechner über seine sog. *TCP-Port*-Nummer. („TCP“ steht für „Transmission Control Protocol“.)

Mit „Dienst“ ist ein Programm gemeint, das die Kommunikation aufbaut und entgegennimmt. Im einfachsten Fall nimmt sie die zu sendenden Daten direkt (per Tastatur) von einem Menschen entgegen und leitet die empfangenen Daten (per Bildschirm) an den Menschen weiter. Ein Programm, das dies leistet, ist *Netcat* (als Befehl: `nc`). (Ein älteres Programm mit ähnlichen Eigenschaften ist `telnet`.)

Mit `nc -p 1234 -l` lassen wir Netcat auf dem TCP-Port Nr. 1234 auf eingehende Verbindungen lauschen (engl.: listen). (Je nach Netcat-Version kann der Befehl auch `nc -l 1234` lauten.)

Mit `nc 192.168.1.137 1234` lasse ich meinen Rechner eine Verbindung zu dem Rechner mit der IP-Adresse `192.168.1.137` auf TCP-Port 1234 aufbauen.

Sobald die Verbindung zustandegekommen ist, können beide Kommunikationspartner Daten eingeben. Jeweils nach Betätigen der Eingabetaste werden die Daten an den Zielrechner übertragen und dort auf dem Bildschirm sichtbar. Mit Strg+C können beide Beteiligte die Verbindung abbauen.

Auf 192.168.1.137:

```
# nc -p 1234 -l
Hallo? Hier ist 192.168.1.196.
Hallo, Hier ist 192.168.1.137.
Wie geht's?
http://www.xkcd.com/222/
^C
#
```

Auf 192.168.1.196:

```
# nc 192.168.1.137 1234
Hallo? Hier ist 192.168.1.196.
Hallo, Hier ist 192.168.1.137.
Wie geht's?
http://www.xkcd.com/222/
#
```

Die Port-Nummer 1234 ist hier willkürlich gewählt. Es gibt standardisierte Port-Nummern für wichtige Dienste, z. B. 80 für Webseiten, 25 für E-Mail, 3128 oder 8080 für WWW-Proxies usw. Unter Unix sind die standardisierten Port-Nummern in einer Datei /etc/services aufgelistet.

1.7 Netzwerkanalyse

Die Analyse des in einem gegebenen Netzwerk stattfindenden Verkehrs ist ein unverzichtbares Werkzeug bei Aufbau und Wartung von Computernetzen. Dieselben Werkzeuge können jedoch auch von Angreifern mißbraucht werden, um sich unberechtigt Zugang zu Daten zu verschaffen oder Daten zu manipulieren.

> **Warnung: Das unerlaubte Mitlesen oder Manipulieren von Netzwerkverkehr ist gemäß deutschem Recht eine Straftat, die eine mehrjährige Freiheitsstrafe nach sich ziehen kann.**

Bei der Anwendung der hier vorgestellten Werkzeuge ist daher höchste Sorgfalt geboten, um nicht versehentlich die Grenze zur Kriminalität zu überschreiten. Die Situation ist vergleichbar mit der Steuerung schwerer Maschinen, die bei unsachgemäßer Handhabung erheblichen Sach- und/oder Personenschaden bewirken können.

1.7.1 tcpdump: Pakete beobachten

Das Programm `tcpdump` protokolliert den Netzwerkverkehr eines Netzwerkteilnehmers – entweder den gesamten Netzwerkverkehr oder einen gefilterten Ausschnitt.

Aufruf:

```
# tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
16:56:38.440647 IP foo.local > bar.local: ICMP echo request,
    id 10645, seq 1, length 64
16:56:38.440931 IP bar.local > foo.local: ICMP echo reply,
    id 10645, seq 1 , length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
#
```

Zu sehen ist ein `ping` von einem Rechner foo zu einem Rechner bar. ICMP echo request steht für die gesendete `ping`-Anfrage, ICMP echo reply für die Antwort.

Wenn z. B. das ICMP echo request zu sehen ist, nicht jedoch das zugehörige ICMP echo reply, erkennen wir daran, daß der Rechner foo den Rechner bar erreichen kann, umgekehrt jedoch nicht. Ein sinnvoller Ausgangspunkt für die Fehlersuche wäre dann das Routing auf Rechner bar.

Mit `tcpdump` kann man nur solchen Netzwerkverkehr analysieren, der den Rechner, auf dem tcpdump läuft, auch erreicht. Dies ist in folgenden Situationen der Fall:

- Der Netzwerkverkehr ist für diesen Rechner bestimmt.
- Der Rechner vermittelt zwischen Rechnern, für die der Netzwerkverkehr bestimmt ist.
- Der Rechner ist mit gemeinsam mit denjenigen Rechnern, für die der Netzwerkverkehr bestimmt ist, an einen *Netzwerk-Hub* angeschlossen. (Im Gegensatz zu einem *Netzwerk-Switch*, der Netzwerkverkehr nur an denjenigen Rechner weiterleitet, für den er bestimmt ist, leitet ein Hub den gesamten Netzwerkverkehr an *alle* angeschlossenen Teilnehmer weiter.)
- Der Rechner nimmt die Identität eines anderen Rechners an, für den der Netzwerkverkehr bestimmt ist. (Dies kann mit Hilfe von ARP-, MAC- oder IP-Spoofing erreicht werden – siehe unten.)

1.7.2 wireshark: Pakete analysieren

Das Programm `wireshark` dient dazu, aufgezeichneten Netzwerkverkehr zu filtern und in einer für Menschen lesbaren Weise darzustellen. Die Aufzeichnung erfolgt entweder durch `wireshark` selbst oder durch andere Programme, z. B. `tcpdump`.

Einsatzgebiete von `tcpdump` und `wireshark` sind:

- **Fehlersuche in Netzwerken:** Wenn bestimmte Netzwerkzugriffe nicht wie gewünscht funktionieren, kann man analysieren, bis wohin die Pakete gelangen, und daraus Rückschlüsse ziehen, an welcher Stelle im Netz sich vermutlich eine Störung befindet.
- **Analyse des Verhaltens von Programmen:** Die netzwerktechnischen Anforderungen von Anwenderprogrammen (z. B. Bank-Software) sind häufig nicht oder nur unzureichend dokumentiert.

  Anstatt bei der Benutzung derartiger Programme auf Sicherheit zu verzichten („einfach alle Ports freischalten“), kann man mit Hilfe der o. a. Werkzeuge herausfinden, welche Netzwerkverbindungen die Software tatsächlich benötigt, und nur diese gezielt freigeben.
- **Erkennen von Angriffen:** Eine im Netzwerk befindliche Schad-Software (z. B. ein *trojanisches Pferd*, das Arbeitsplatzrechner von außen steuerbar macht) macht sich häufig durch ungewöhnlichen Netzwerkverkehr bemerkbar. Wer weiß, wie „normaler“ Netzwerkverkehr im eigenen Netz aussieht, kann einen Eindringling sofort erkennen.

  Teilweise ist es möglich, derartige Analysen mit Hilfe eines sog. *Intrusion Detection Systems* zu automatisieren, z. B. mit `snort`.

1.7.3 ARP und ARP-Spoofing

Wie man in `tcpdump` gut erkennen kann, wird die Zuordnung zwischen IP- und MAC-Adressen im Netz dynamisch ausgehandelt. Hierbei kommt das *Address Resolution Protocol (ARP)* zum Einsatz.

Aus Sicht von Ethernet sind ARP-Nachrichten ein spezieller Typ von Nutzdaten, der also an die Stelle von z. B. IP-Paketen tritt.

ARP sieht keinerlei Authentifizierungsmechanismus vor. Es ist daher möglich (z. B. mit Hilfe der Software `ettercap`), durch gezieltes Versenden von ARP-Nachrichten die ARP-Tabellen anderer Netzwerkteilnehmer zu manipulieren und auf diese Weise z. B. den für andere Rechner bestimmten Datenverkehr auf den eigenen Rechner umzuleiten. Diese Vorgehensweise heißt *ARP-Spoofing*.

Einsatzmöglichkeiten von ARP-Spoofing:

- **Man-in-the-Middle-Angriffe**

  Für andere Netzwerkteilnehmer bestimmte Nachrichten mitlesen und/oder manipulieren
- **Hochverfügbarkeit:**

  Mit Hilfe von ARP-Spoofing kann ein Reserve-Server verzögerungsfrei die Aufgaben eines ausgefallenen Servers übernehmen.

Ein verwandtes Verfahren kommt bei *ARP-Ping* zum Einsatz: Man manipuliert die ARP-Tabelle des eigenen Rechners so, daß man einen anderen Rechner, von dem nur die MAC-, aber nicht die IP-Adresse bekannt ist, unter einer selbstgewählten IP-Adresse erreichen kann. Man setzt also gewissermaßen „künstlich von außen“ die IP-Adresse eines anderen Netzwerkteilnehmers und kann diesen anschließend per TCP/IP erreichen – insbesondere mit `ping`, daher die Bezeichnung „ARP-Ping“.

ARP-Ping ist gelegentlich nötig, um eine Netzwerkkomponente einzurichten, deren IP-Adresse man nur über ein Web-Interface setzen kann, dessen Benutzung aber umgekehrt eine IP-Adresse voraussetzt.

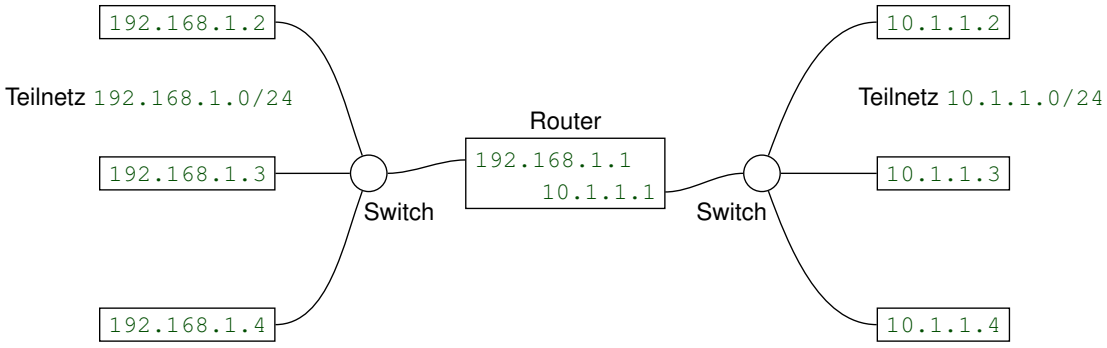
Ein weiteres verwandtes Verfahren ist *MAC-Spoofing*. Die MAC-Adresse eines Ethernet-Adapters ist zwar in der Hardware verankert; letztlich entscheidet jedoch die Treiber-Software darüber, welche MAC-Adresse im Ethernet-Protokoll als Absender verwendet wird.

Durch Manipulation der eigenen MAC-Adresse (mit `ifconfig`) kann daher ein Rechner die Identität eines anderen Rechners annehmen. Die Anwendungen sind ähnlich wie bei ARP-Spoofing.

1.8 Teilnetze miteinander verbinden: IP-Forwarding, Netzmasken, Routing (route)

Nicht alle Rechner, mit denen man kommunizieren möchte, sind direkt, d. h. über denselben Switch, per Kabel mit dem eigenen Rechner verbunden. Um langreichweitige Netzwerke wie das Internet aufzubauen, ist es notwendig, physikalische Teilnetze logisch zu einem größeren Netz zu verbinden.

Hierfür wird ein mit mehreren Netzwerkkarten ausgerüstetes Gerät – z. B. ein PC – verwendet, der sog. *Router*. Jede Netzwerkkarte bekommt eine IP-Adresse aus einem der Teilnetze zugewiesen, die der Router verbinden soll.



Auf einem Linux-basierten PC kann die Weiterleitung (*IP-Forwarding*) von Netzwerkdaten zwischen mehreren Netzwerkkarten über eine Variable im Betriebssystemkern (*Kernel-Variable*) ein- und ausgeschaltet werden. Diese ist über eine Pseudo-Datei ansprechbar. Der Befehl, um den aktuellen Wert der Variablen auszulesen, lautet:

```
# cat /proc/sys/net/ipv4/ip_forward
0
```

Eine Null steht für „nicht weiterleiten“. Um das Weiterleiten zu aktivieren, wird die Variable wie folgt auf 1 gesetzt (und wieder ausgelesen):

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# cat /proc/sys/net/ipv4/ip_forward
1
#
```

Beide Teilnetze sind für den Router erreichbar, und dieser leitet nun auch bei Bedarf Daten aus dem einen Teilnetz in das andere weiter. Damit dies stattfinden kann, müssen nun noch die Teilnehmer der beiden Teilnetze über diesen Mechanismus informiert werden: Es müssen dort sog. *Routen* eingerichtet werden, die über den Router in das jeweils andere Teilnetz führen. Hierfür stellt der Betriebssystemkern die sog. *Routing-Tabelle* zur Verfügung, die mit dem Befehl `route` ausgegeben und manipuliert werden kann.

Mit `route -n` läßt man sich die Routing-Tabelle ausgeben. Auf einem Rechner im linken Teilnetz erhält man:

```
# route -n
Kernel IP routing table
Destination  Gateway      Genmask         Flags Metric Ref    Use Iface
192.168.1.0  0.0.0.0      255.255.255.0   U     0      0        0 eth0
#
```

Wie bei `arp` steht das `-n` hinter `route` für „numeric“; wir lassen uns also wieder IP-Adressen anstelle von Rechnernamen anzeigen.

Der Befehl, um dem Betriebssystemkern mitzuteilen, daß über den Rechner `192.168.1.1` das Nachbarnetz `10.1.1.1` erreicht werden kann, lautet:

```
# route add -net 10.1.1.0 netmask 255.255.255.0 gw 192.168.1.1
#
```

Anschließend rufen wir mit `route -n` die veränderte Tabelle erneut ab:

```
# route -n
Kernel IP routing table
Destination  Gateway      Genmask         Flags Metric Ref    Use Iface
192.168.1.0  0.0.0.0      255.255.255.0   U     0      0        0 eth0
10.1.1.0     192.168.1.1  255.255.255.0   UG    0      0        0 eth0
#
```

Die Tabelle enthält

- die IP-Adresse des Ziel-Netzes, das über die Route erreicht werden soll,
- die IP-Adresse des Routers („Gateway“), der den Kontakt herstellt, und
- die sog. *Netzmaske*, die die Größe des Ziel-Netzes bestimmt.

Ein Netz (Teilnetz, Subnetz) wird durch die Kombination einer IP-Adresse mit einer Netzmaske spezifiziert.

IP-Adressen und Netzmasken sind 32-Bit-Zahlen, die in Blöcken zu jeweils 8 Bits geschrieben werden. Ein Rechner befindet sich genau dann in einem Netz, wenn die bitweise Und-Verknüpfung der IP-Adresse des Rechners mit der Netzmaske die IP-Adresse des Netzes ergibt. Zum Beispiel:

```
    192.168.   1.137   <-- IP-Adresse des Rechners
&   255.255.255.   0   <-- Maske des Netzes
    ---------------
    192.168.   1.   0   <-- IP-Adresse des Netzes
```

In diesem Beispiel enthält die Netzmaske in Binärschreibweise 24 Einsen und 8 Nullen. Das Netz mit der IP-Adresse `192.168.1.0` und der Netzmaske `255.255.255.0` wird daher auch `192.168.1.0/24` abgekürzt. Es enthält alle IP-Adressen, die mit `192.168.1` beginnen.

Ein wichtiger Spezialfall ist das Netz mit der IP-Adresse `0.0.0.0` und der Netzmaske `0.0.0.0`. Da eine binäre Und-Verknüpfung der IP-Adresse eines beliebigen Rechners mit der Netzmaske dieses Netzes immer `0.0.0.0`, also die IP-Adresse des Netzes ergibt, enthält dieses Netz *alle* möglichen IP-Adressen. Dies macht man sich zu Nutze, um eine Route für „alle anderen“ IP-Adressen anzulegen. Der Router, der ein Netz mit „allen anderen“ Netzen – also der Außenwelt – verbindet, heißt das *Default-Gateway* dieses Netzes.

Der Befehl, um einem Rechner sein Default-Gateway (hier: `192.168.1.1`) mitzuteilen,

```
route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.1
```

darf wie folgt abgekürzt werden:

```
route add default gw 192.168.1.1
```

2 Schichtenmodelle

2.1 Das TCP/IP-Schichtenmodell

Als Vorläufer des Internet entwickelte das amerikanische Verteidigungsministerium (Department of Defense – DoD) in den 1960er und 1970er Jahren das *Arpanet* (Advanced Research Projects Agency Network). Zu den erklärten Zielen gehörte ein möglichst robuster Aufbau.

Zu diesem Zweck unterteilte man die Aufgabe, auf einem entfernten Rechner eine Anwendung zu steuern (z. B. Daten übertragen oder abrufen) in mehrere Teilaufgaben, sog. *Schichten* (engl.: layers).

Jede Schicht hat eine genau definierte Aufgabe zu erfüllen. Jede höhere Schicht betrachtet die Aufgaben der darunterliegenden Schichten als erfüllt. In jeder Schicht kommen *Protokolle* zum Einsatz, die für die darüberliegende Schicht *Dienste* anbieten und gleichzeitig die Dienste der darunterliegenden Schicht nutzen.

Die Schichten im TCP/IP-Modell (oder DoD-Modell) lauten:

Nr.	Name	Beispiel-Protokolle
4	Anwendung	HTTP, SMTP, SSH, OpenVPN, FTP, LDAP, NCP, AppleTalk AFP
3	Transport	UDP, TCP, SPX, AppleTalk ATP
2	Internet	IP, IPX, NetBEUI, AppleTalk DDP
1	Netzzugang	Ethernet, FDDI, Profibus, ARCNET, Token Ring, LocalTalk

Schicht 1 erfüllt die Aufgabe, Daten innerhalb des physikalisch erreichbaren Teilnetzes zu einem Zielrechner zu befördern. Die verschiedenen Wege, in denen dies geschehen kann (Kabel, Glasfasern, Funk, ...), werden in dieser Schicht *gekapselt*, d. h. auf dieser Ebene gelöst und vor höheren Schichten verborgen.

Zu den für die heutige Praxis wichtigsten Beispielen gehören Ethernet und WLAN. Die verschiedenen Netzteilnehmer werden über ihre *Hardware-Adressen* unterschieden.

Schicht 2 baut auf Schicht 1 auf. Ab Schicht 2 werden also die verschiedenen Wege der Datenübertragung nicht mehr unterschieden, weil dieses Problem bereits in Schicht 1 gelöst wurde.

Schicht 2 löst das Problem, Daten von Rechner A zu Rechner B zu transportieren, auch wenn diese nicht direkt über das physikalische Netz, sondern nur über Zwischenstufen (weitere Rechner) erreichbar sind.

Das wichtigste Beispiel ist das Internet Protocol (IP). Die verschiedenen Rechner werden über ihre *IP-Adressen* unterschieden.

Schicht 3 löst die Aufgabe, Daten von einer konkreten Anwendung auf Rechner A zu einer konkreten anderen Anwendung auf Rechner B zu transportieren. Die verschiedenen Anwendungen werden über ihre *Port-Nummern* unterschieden.

Wichtige Beispiele sind TCP (daher: „TCP/IP“) und UDP.

Schicht 4 schließlich beherbergt die gemeinsamen Sprachen (Protokolle) der beteiligten Anwendungen.

Wichtige Beispiele sind das Hypertext Transfer Protokoll (HTTP) zur Übertragung von Webseiten und das Simple Mail Transfer Protokoll (SMTP) zur Übertragung von E-Mails.

2.2 Das OSI-Schichtenmodell

Als Designgrundlage für den Aufbau von Rechnernetzen entwickelte die *International Organization for Standardization (ISO)* von 1979 bis 1983 den Standard *Open Systems Interconnection (OSI)* mit einem verfeinerten Schichtenmodell:

Nr.	Name	Beispiel-Protokolle
7	Anwendung	HTTP, SMTP, SSH, FTP, LDAP, NCP, AppleTalk AFP
6	Darstellung	ASCII, EBCDIC, Kompression, SSL/TLS (Verschlüsselung)
5	Sitzung	RPC, SOCKS, AppleTalk ASP
4	Transport	UDP, TCP, SPX, AppleTalk ATP
3	Vermittlung	IP, ICMP, IPX, NetBEUI, AppleTalk DDP
2	Sicherung	Ethernet, FDDI, Profibus, ARCNET, Token Ring, LocalTalk, Brieftauben
1	Bit-Übertragung	Telefon-, Koaxial-, TP-, Glasfaser- oder sonstige Kabel, Funk, Papier

Gegenüber dem TCP/IP-Schichtenmodell weist das ISO-OSI-Schichtenmodell drei zusätzliche Schichten auf:

- Die unterhalb der Netzzugangs-Schicht liegenden Hardware-Komponenten werden als zusätzliche Schicht eingeführt.
- Zwei zusätzliche Schichten zwischen der Transport- und der Anwendungs-Schicht dienen dem Aufbau eines Sitzungskonzepts (z. B. Warenkorb bei Online-Shopping) sowie Umwandlungen der transportierten Daten (Zeichensatz-Umwandlungen, Kompression, Verschlüsselung).

In der Praxis kommen nicht immer alle Schichten zum Tragen. So wird z. B. der Warenkorb eines Online-Shops i. d. R. oberhalb von HTTP (z. B. durch HTTP-Cookies) realisiert, und Verschlüsselung kann nicht nur in Schicht 6, sondern an beliebigen Stellen erfolgen (z. B. IPsec als verschlüsselte Alternative zu IP in Schicht 3, OpenPGP-Verschlüsselung als Inhalt einer E-Mail oberhalb von Schicht 7). Darüberhinaus können Netzverbindungen (z. B. IP-Pakete, Schicht 4) über höhere Protokolle (z. B. HTTP, Schicht 7) *getunnelt* werden, wodurch nochmals ein ganzer Stapel von Schichten oberhalb von Schicht 7 entsteht. Dies zeigt, daß Schichtenmodelle nicht zu starr angewendet werden dürfen.

Nr.	Name	Beispiel-Protokolle
7	Anwendung	OpenPGP, S/MIME
		HTTP, SMTP, SSH, OpenVPN, FTP, LDAP, NCP, AppleTalk AFP
6	Darstellung	ASCII, EBCDIC, Kompression, SSL/TLS (Verschlüsselung)
5	Sitzung	RPC, SOCKS, AppleTalk ASP
4	Transport	UDP, TCP, SPX, AppleTalk ATP
3	Vermittlung	IP, IPsec, ICMP, IPX, NetBEUI, AppleTalk DDP
2	Sicherung	ARP
		Ethernet, FDDI, Profibus, ARCNET, Token Ring, LocalTalk, Brieftauben
1	Bit-Übertragung	Telefon-, Koaxial-, TP-, Glasfaser- oder sonstige Kabel, Funk

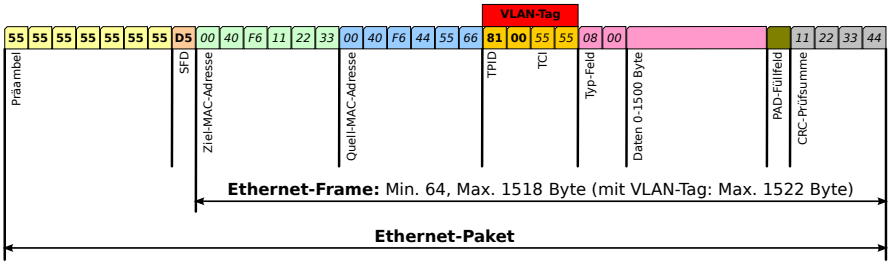
Beispiel: Bei der Realisierung des Internet-Protokolls per Brieftaube gemäß dem Standard *RfC 1149* werden IP-Pakete ausgedruckt, per Brieftaube transportiert und wieder eingescannt. Die Brieftaube ist somit unterhalb von IP, also unterhalb von Schicht 3 einzuordnen. Sie tritt in Konkurrenz zu anderen Möglichkeiten, IP-Pakete zu transportieren, wie z. B. Ethernet, gehört also wie diese der Schicht 2 an. Da die Brieftaube ihre eigene Hardware mitbringt, übernimmt sie die Funktionen von Schicht 1 gleich mit.

Anstelle von IP-Paketen könnten Brieftauben auch Ethernet-Frames transportieren. In diesem Fall würden sie eindeutig in Schicht 1 agieren.

In humorvoller Weise bezeichnet man auch den Benutzer des Computers als *OSI-Schicht 8*.

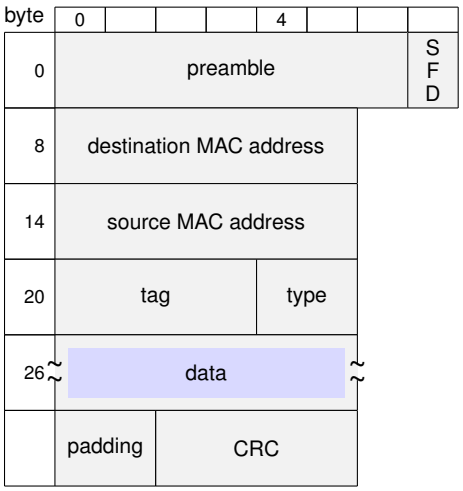
2.3 Protokollstapel

Jede Schicht definiert für den Transport von Daten *Protokolle*. Diese bestehen i. d. R. aus Absender- und Empfängeradresse, den eigentlichen Nutzdaten und eventuellen weiteren Informationen (Beschreibung des Inhalts, Prüfwert). Jedes Protokoll enthält daher einen Vorspann (Header) und/oder Nachspann (Trailer).

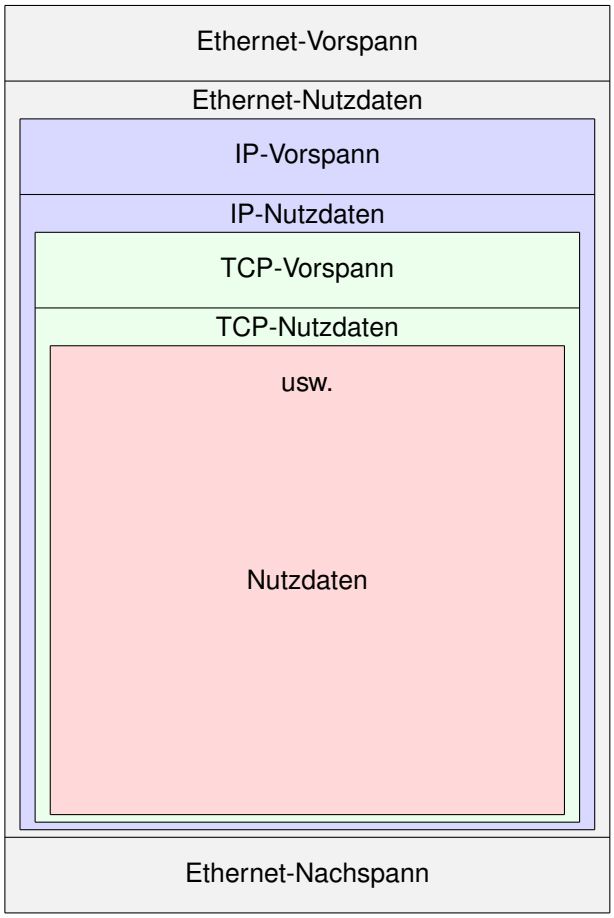


Beispiel: Ein Ethernet-Paket enthält als Vorspann u. a. die Hardware-Adressen des Absenders und des Empfängers und als Nachspann ein Prüfwert.

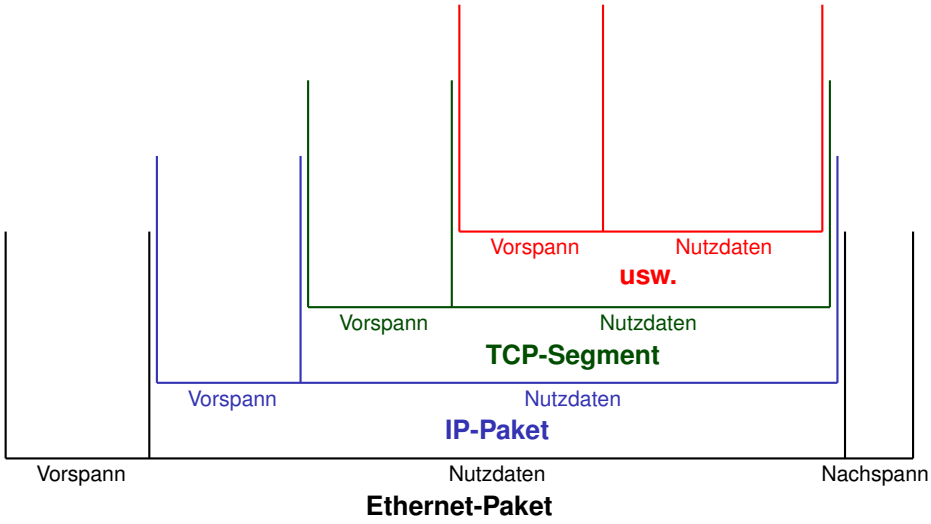
Aus Platzgründen werden wir die nacheinander gesendeten Daten meistens nicht als eine lange Zeile, sondern in mehreren Zeilen notieren:



Bei den Nutzdaten handelt es sich um die Daten der jeweils nächsthöheren Schicht, die wiederum ihren eigenen Vor- und/oder Nachspann haben:



Derartig verschachtelte (in- oder aufeinander gestapelte) Protokolle nennt man einen *Protokollstapel* oder *Protokollturm*:



2.4 Das Internet

- Daten über weite Strecken austauschen – so wie Telefonieren
- Akteure: US-Verteidigungsministerium (Department of Defense – DoD), Hochschulen
- möglichst robust („soll einen Atomkrieg überstehen“), daher möglichst dezentral
- Design-Überlegung: verbindungsorientiert oder verbindungslos?
  $\longrightarrow$ verbindungslos

  verbindungsorientiert: Telefon
  Verbindung wird aufgebaut, besteht (wird reserviert), wird abgebaut

  verbindungslos: Internet
  Nachricht wird in Pakete (Datagramme) unterteilt;
  jedes Paket „sucht sich seinen Weg selbst“
- 2 Schichten: IP vermittelt Datagramme; TCP macht daraus wieder eine Art Verbindung.
- Problem der Heterogenität lösen: SNA, ATM, IPX, NetBEUI, TCP/IP, X.233 (CONS, CLNS), . . .

  Tanenbaum, S. 460:

  > Die Frage, ob der heutige Überfluß an Netzarten vorübergehender Natur ist und verschwindet, sobald alle gemerkt haben, wie wundervoll [tragen Sie hier Ihr bevorzugtes Netz ein!] ist, [. . .], ist Gegenstand heftiger Kontroversen. [. . .]
  >
  > Wir glauben an den Fortbestand verschiedener Netze [. . .] aus folgenden Gründen: [ca. 2/3 einer Buchseite]

  Mit anderen Worten: „TCP/IP wird SNA, ATM, IPX, NetBEUI, Appletalk und X.233 niemals [vollständig] ersetzen.“
- Realität: TCP/IP hat alle anderen Netzwerkprotokolle nahezu vollständig ersetzt.

$\longrightarrow$ Das Problem der Heterogenität wurde gelöst
durch verschiedene Netzwerkzugänge zu TCP/IP
(also eine Schicht tiefer – siehe Abschnitt 3.5).

2.4.1 Warum hat sich TCP/IP durchgesetzt?

- Von Technikern für Techniker, offener Prozeß, RfCs, ...
- IETF = Internet Engineering Task Force: http://tools.ietf.org/html/rfc3935
- Offener Standardisierungsprozeß:
  Experimental, Proposed Standard, Draft Standard, Internet Standard
- Hier sieht man, was Ingenieure leisten können, wenn man sie läßt.

2.4.2 Design-Prinzipien des Internet

Auszug aus RFC 1958 – http://www.ietf.org/rfc/rfc1958.txt

2.1 [T]he goal is connectivity, the tool is the Internet Protocol,
and the intelligence is end to end rather than hidden in the network.

2.2 [T]here should be one, and only one, protocol at the Internet level. [...]

The Internet level protocol must be independent of the hardware medium
and hardware addressing.

2.4 Fortunately, nobody owns the Internet [...].

3.1 Heterogeneity is inevitable and must be supported by design.

3.2 If there are several ways of doing the same thing, choose one.

3.3 All designs must scale readily to very many nodes per site and to many millions of sites.

3.4 Performance and cost must be considered as well as functionality.

3.5 Keep it simple. When in doubt during design, choose the simplest solution.

3.6 Modularity is good. If you can keep things separate, do so.

3.7 In many cases it is better to adopt an almost complete solution now,
rather than to wait until a perfect solution can be found.

3.8 Avoid options and parameters whenever possible.
Any options and parameters should be configured or negotiated dynamically rather than manually.

3.9 Be strict when sending and tolerant when receiving.
Implementations must follow specifications precisely when sending to the network,
and tolerate faulty input from the network.
When in doubt, discard faulty input silently, without returning an error message
unless this is required by the specification.

3.10 Be parsimonious with unsolicited packets, especially multicasts and broadcasts.

3.11 Circular dependencies must be avoided.

For example, routing must not depend on look-ups in the Domain Name System (DNS),
since the updating of DNS servers depends on successful routing.

3.12 Objects should be self decribing (include type and size), within reasonable limits.
Only type codes and other magic numbers assigned by the
Internet Assigned Numbers Authority (IANA) may be used.

3.13 All specifications should use the same terminology and notation,
and the same bit- and byte-order convention.

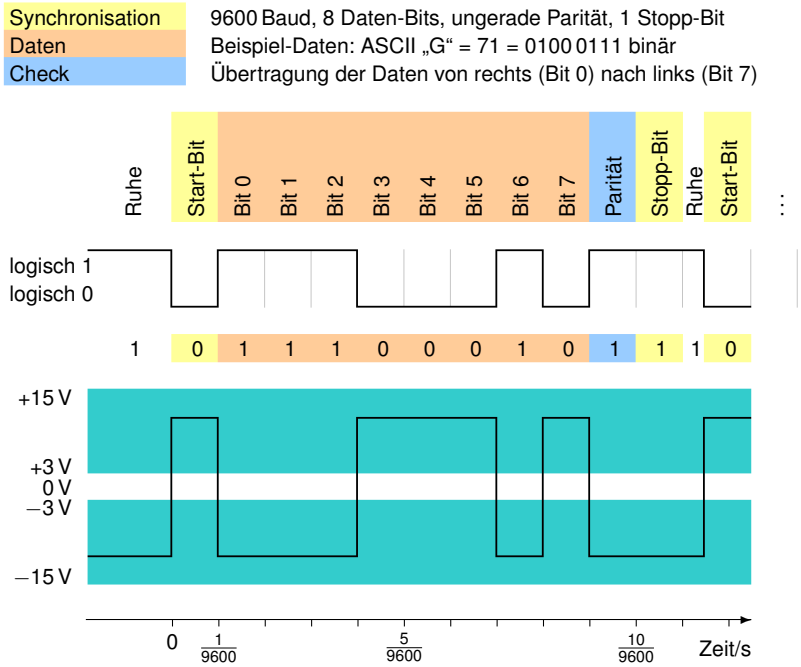
3.14 And perhaps most important:
Nothing gets standardised until there are multiple instances of running code.

3 TCP/IP-Schicht 1: Netzzugang

3.1 Übertragungstechnik

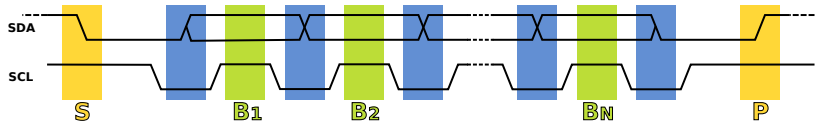
3.1.1 Serielle Datenübertragung über elektrische Leitungen

- Problem: Wie bekommen wir überhaupt ein Bit von A nach B?
- Lösung: definierte Spannungen und Zeiten ⟶ Protokoll
- Beispiel: RS-232
  asynchron: 1 Leitung + Uhr



Ähnliche Protokolle: RS-422, RS-485, LIN-Bus, CAN-Bus, Ethernet

- Beispiel: I^2C
  synchron: 2 Leitungen (statt Uhr)



3.1.2 Datenübertragung über das Telefonnetz

- Problem: Wie bekommen wir überhaupt ein Bit von A nach B?
- Lösung: definierte Frequenzen und Zeiten ⟶ Protokoll
- Beispiel: RS-232 über Modem (Modulator/Demodulator)

noch im Einsatz: Fax

Beispiel: 300 Bit/s

	logisch 1	logisch 0
Anrufer	1270 Hz	1070 Hz
Angerufener	2225 Hz	2025 Hz

3.1.3 DSL

Telefonleitung = 50 Jahre alter Draht

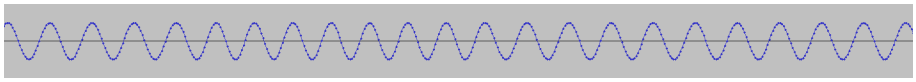
- Problem: Wie bekommen wir möglichst schnell möglichst viele Bits von A nach B?
- Lösung: Zerlegung in Frequenzen Amplituden- und Phasenmodulation
- Ähnliches Prinzip:
  - drahtlose Übertragung
  - PowerLAN

Amplitudenmodulation

Man hört einen Ton lauter bzw. leiser.
Im einfachsten Fall gibt es nur „volle Lautstärke“ und „Ton aus“.

Beispiel: Morsezeichen

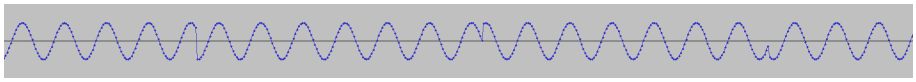
Frequenzmodulation



Ein durchgehender Ton ändert seine Höhe.

Beispiel: Modems ab 110 bis 1200 Bit/s

Phasenmodulation



Im Gegensatz zu Amplituden- und Frequenzmodulation ist Phasenmodulation nicht direkt hörbar.
Man hört einen durchgehenden Ton; Phasenwechsel machen sich als Knackgeräusch bemerkbar.

Beispiel: Modems ab 600 bis 56000 Bit/s

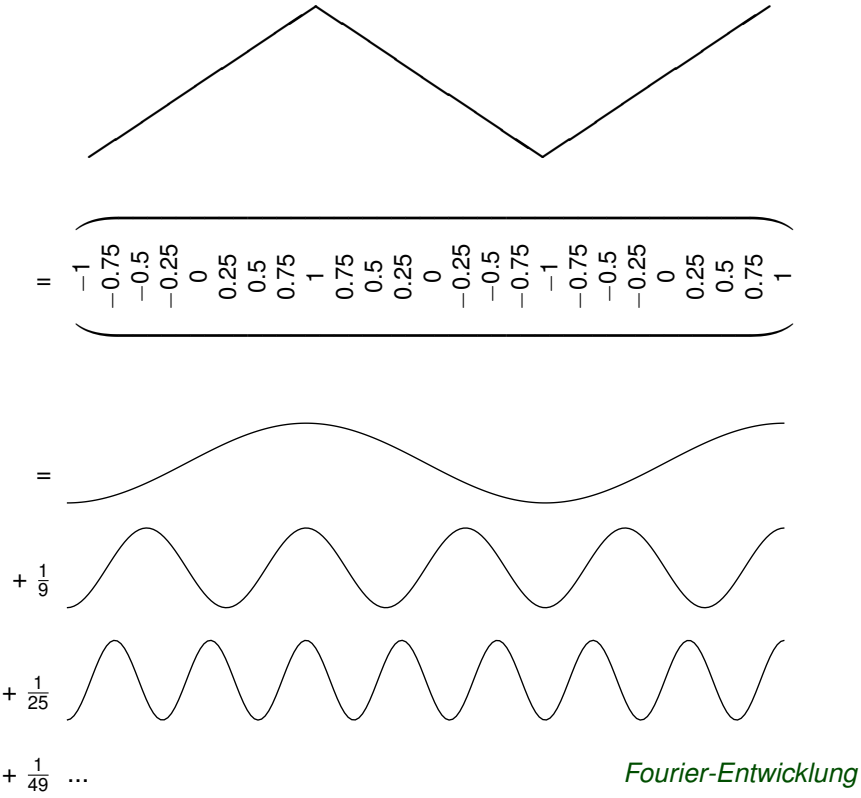
Warum dies so ist und wie man Amplituden-, Frequenz- und Phasenmodulation geschickt nutzen kann, um noch deutlich größere Datenmengen durch Kupferleitungen zu übertragen (DSL), wird im folgenden erläutert.

Verschiedene Basen

Vektoren in $\mathbb{R}^3$:

$$2\begin{pmatrix}1\\0\\1\end{pmatrix}+3\begin{pmatrix}0\\1\\-1\end{pmatrix}+5\begin{pmatrix}-1\\1\\0\end{pmatrix}=\begin{pmatrix}-3\\8\\-1\end{pmatrix}$$
$$=-3\begin{pmatrix}1\\0\\0\end{pmatrix}+8\begin{pmatrix}0\\1\\0\end{pmatrix}-1\begin{pmatrix}0\\0\\1\end{pmatrix}$$

Funktion als Vektor:



Fourier-Entwicklung

Ein Computer nimmt Geräusche (hier: Dreieckschwingung) auf, indem er den Schalldruck als Funktion der Zeit mißt und in Funktionswerte an gewissen *Stützpunkten* zerlegt (hier: quer liegender Vektor). Unser Ohr hingegen zerlegt die Schwingung in *Töne*, die wir als *Grundton* (hier: Sinuskurve ohne Vorfaktor) und *Obertöne* (hier: Sinuskurven mit Vorfaktoren) bezeichnen.

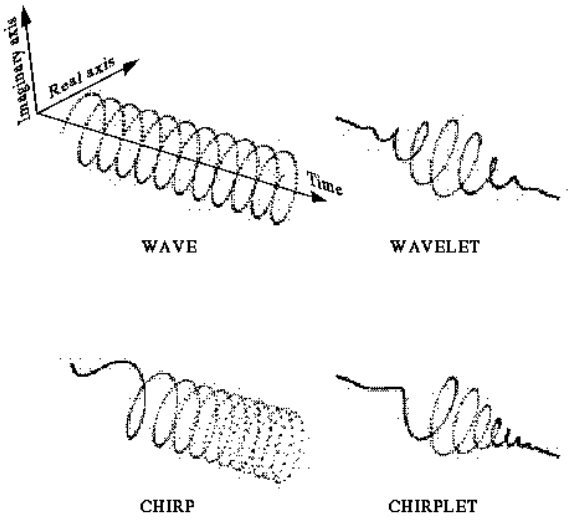
Ein Phasenwechsel stellt einen abrupten Sprung in der Schalldruckkurve dar. Um einen abrupten Sprung in Töne, also Sinus-Schwingungen zu zerlegen, sind sehr viele (theoretisch: unendlich viele) Sinus-Schwingungen erforderlich. Die kurzzeitige gleichzeitige Anregung von sehr vielen Sinus-Schwingungen („alle Töne gleichzeitig“) empfindet unser Ohr als Knackgeräusch.

Insbesondere kann ein Computer die Phase des Signals direkt wahrnehmen, während unser Ohr nur den Phasenwechsel als Knackgeräusch meldet.

Dieses Konzept der *Fourier-Analyse* erklärt auch, weshalb sich Signale mit unterschiedlichen Trägerfrequenzen gegenseitig stören können: Jede Modulation generiert Sinus-„Obertöne“ mit ganz anderen Frequenzen.

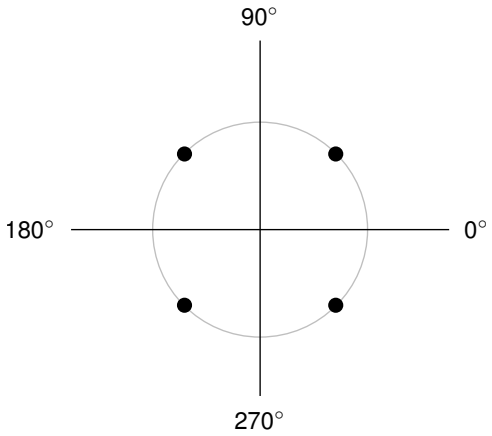
Weitere mögliche Basen:

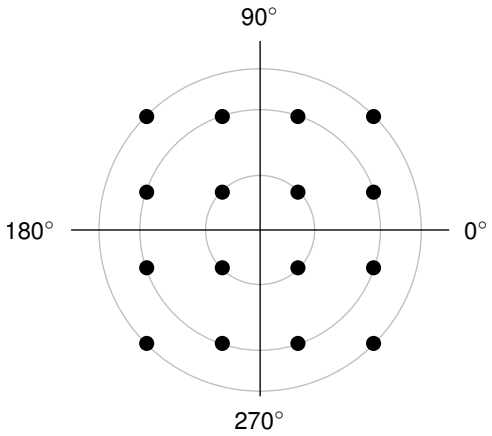
- Chirps
- Wavelets
- Chirplets



Quadraturamplitudenmodulation (QAM)

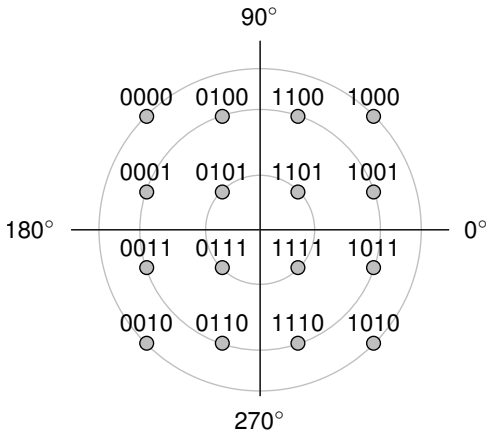
Amplituden- + Phasenmodulation = *Quadraturamplitudenmodulation (QAM)*





Konstellationsdiagramm
. . . z. Zt. bis 64 $\times$ 64

- Abstand vom Ursprung = Amplitude
- Winkel = Phase



Zuordnung mehrerer Bits zu einem *Symbol*

$\longrightarrow$ Bei Verwechselung nur 1 Bit Fehler

$\longrightarrow$ kann mathematisch aufgefangen werden

- 1 Trägerfrequenz, viele Symbole, schnell nacheinander
  ⟶ Störung betrifft komplette Übertragung
- viele Trägerfrequenzen, wenige Symbole, gemächlich nacheinander
  ⟶ Störung betrifft nur einzelne Frequenzen
  ⟶ ***Orthogonales Frequenzmultiplexverfahren (OFDM)***

Trägerfrequenzen beeinflussen sich – eine Symboldauer lang – nicht

Anwendung: DSL, Digitales Radio und Fernsehen, WLAN, ...

Andere Verfahren

- Bluetooth 1.2: Adaptive Frequency Hopping
- In Bluetooth doch nicht umgesetzt: Ultra-Wide-Band
  MB-OFDM (WiMedia-Allianz), DS-UWB (UWB-Forum)

3.2 Störungs- und Kollisionsbehandlung

Bei der Übertragung von Signalen über eine elektrische Leitung („Draht“) gilt es, verschiedene Probleme zu lösen:

- Elektromagnetische Störungen von außen („Störung durch Mikrowellenherd“)
- Elektromagnetische Störungen nach außen („Störung des Handy-Empfangs“)
- Kollisionen gleichzeitiger Datenübertragungsversuche mehrerer Teilnehmer

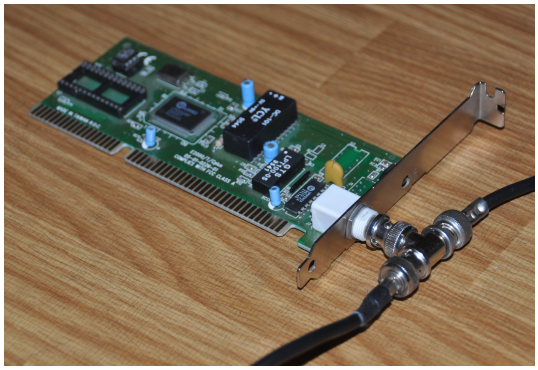
3.2.1 Vermeidung elektromagnetischer Störungen

Die grundsätzlichen Mechanismen zur Vermeidung elektromagnetischer Störungen (in beiden Richtungen) lauten:

- Abschirmung
- (pseudo-) differentielle Signalübertragung

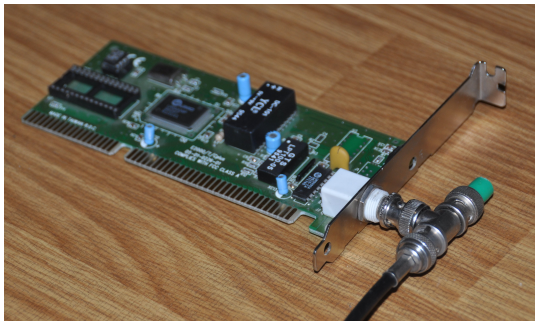
Die besten Abschirmungseigenschaften haben *Koaxialkabel*. Bis in die 1990er Jahre wurden Rechnernetze über Koaxialkabel mit BNC-Steckern und einem Wellenwiderstand von 50 Ω realisiert.

Wie bereits in der Vorlesung *Grundlagen Rechnertechnik* behandelt, werden bei der Verwendung von Koaxialkabeln in Rechnernetzen alle beteiligten Rechner mit Hilfe von T-Stücken an dasselbe Koaxialkabel angeschlossen.

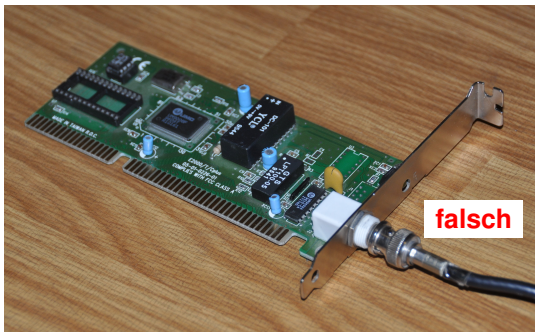


Am Ende des Kabelstrangs erfolgt eine Terminierung über einen ohmschen Widerstand (*Terminator*), der genau dem Wellenwiderstand entspricht.

Ohne Terminator wird das Signal am Ende der Leitung reflektiert und kollidiert – bei hohen Übertragungsgeschwindigkeiten – mit dem nächsten Signal, das sich bereits in der Leitung befindet.



Es ist insbesondere normalerweise falsch, ein Koaxial-Netzwerkkabel direkt an die Netzwerkkarte anzuschließen. (Ausnahme: Manche Netzwerkkarten haben einen eingebauten, per Jumper aktivierbaren Terminator.)



Koaxialkabel sind verhältnismäßig teuer. Eine preisgünstige Alternative („Koaxialkabel für Arme“) stellen verdrillte Adernpaare, sog. *Twisted-Pair*-Kabel (TP-Kabel) dar, wenn sie in Kombination mit differentieller oder pseudo-differentieller Signalübertragung eingesetzt werden.

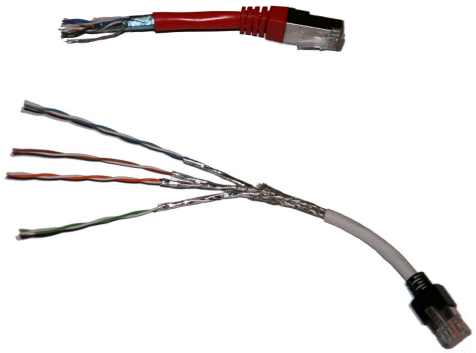
- Bei differentieller Signalübertragung wird das Signal in beide Adern zugleich mit entgegengesetzter Polung eingespeist. Durch die Verdrillung wechselwirken beide Adern im Mittel gleichmäßig mit der Umgebung. Störungen von außen betreffen dann immer beide Adern zugleich, so daß die Differenzspannung erhalten bleibt. Störungen nach außen mitteln sich heraus.
- Bei pseudo-differntieller Signalübertragung wird das Signal in eine Ader eingespeist, während das andere auf Masse gelegt wird. Auch hier betreffen Störungen von außen immer beide Adern zugleich, so daß die Differenzspannung erhalten bleibt.

TP-Kabel verbinden in der Regel immer nur zwei Netzwerkteilnehmer miteinander. Die Terminierung findet im Netzwerkadapter statt. Wenn es mehr als zwei Netzwerkteilnehmer gibt, müssen diese über ein Zusatzgerät zusammengeschaltet werden:

- Ein Netzwerk-*Hub* stellt eine elektrische Verbindung zwischen allen Teilnehmern her. Jedes eintreffende Signal wird an alle Teilnehmer außer dem Absender weitergeleitet.
- Ein Netzwerk-*Switch* analysiert das Ethernet-Protokoll und leitet jedes eintreffende Signal anhand der MAC-Adresse an den Empfänger weiter.

Durch den gegenüber Koaxialkabeln geringeren Platzbedarf ist es möglich, mehr als ein Adernpaar (üblicherweise: vier) in einem Kabel unterzubringen. Dabei wird einem Übersprechen zwischen den Adern durch eine unterschiedliche Schlaghöhe begegnet: Über die Länge eines langen Kabels kommen sich jeweils unterschiedliche Adern nah, so daß sich Übersprechungen herausmitteln.

Bei einfachen TP-Kabeln (UTP für *unshielded twisted pair*, oben) befinden sich die vier Adernpaare in einer gemeinsamen Abschirmung. Bei hochwertigen TP-Kabeln (STP oder FTP für *shielded* oder *foiled*, unten) sind alle vier Adernpaare separat abgeschirmt.



Bezüglich der erreichbaren Datenübertragungsraten werden TP-Kabel in *Kategorien* eingeteilt. CAT-5-TP-Kabel eignen sich bis 100 MBit, CAT-5e-, CAT-6- und CAT-7-TP-Kabel auch für Gigabit-Ethernet. (Der CAT-5-Standard wurde 2002/2003 neu gefaßt und schreibt seitdem Gigabit-Tauglichkeit vor. Dies muß auf ältere CAT-5-Installationen jedoch nicht zutreffen.)

Kabel niedrigerer Kategorien eignen sich für Telefonleitungen einschließlich DSL. Eine Verwendung in Rechnernetzen ist mit Qualitätseinbußen (z. B. Reichweite) möglich.

Um gleichzeitig in beiden Richtungen kommunizieren zu können (*Full Duplex*), verwendet 100-MBit-Ethernet zwei der vier Adernpaare. Telefonverbindungen (einschließlich DSL) und *Half-Duplex*-Verbindungen kommen mit einem Adernpaar aus.

Für Gigabit-Ethernet kommen alle vier Adernpaare zum Einsatz.

3.2.2 Kollisionserkennung: CSMA/CD (Ethernet)

Sobald mehrere Teilnehmer gleichzeitig auf dieselben Leitungen zugreifen, kann es zu Kollisionen kommen. Ethernet begegnet diesem Problem durch *Carrier Sense Multiple Access / Collision Detection (CSMA/CD)*.

Ein Ethernet-Teilnehmer liest während des Sendens, um Kollisionen zu erkennen. Wenn eine Kollision erkannt wird, bricht der Teilnehmer das Senden ab und wartet eine *zufällige* Zahl, bevor er einen neuen Sende-Versuch startet. (Tatsächlich handelt es sich um Pseudo-Zufall.)

Damit während des Sendens eine Kollision erkannt werden kann, muß das Signal mindestens so lang sein, daß es während des Sendens den Hin- und Rückweg zu allen Teilnehmern desselben Netzabschnitts zurücklegt. Bei Ethernet beträgt diese Mindestlänge 64 Bytes; kleinere Pakete werden aufgefüllt.

Die Zufallskomponente ist wichtig, damit sich nicht mehrere Teilnehmer systematisch gegenseitig blockieren.

Dieses System hat sich in der Praxis bewährt. Es hat allerdings auch Nachteile:

- Es sind keine Echtzeit-Anwendungen möglich.
- Die zur Verfügung stehende Bandbreite wird nicht vollständig ausgenutzt.

3.2.3 Kollisionsfreie Datenübertragung: Token Ring, FDDI

Das Verfahren des *Token Ring* vermeidet Kollisionen dadurch, daß ein Signal („Token“) permanent zwischen den Teilnehmern kreist. Um senden zu können, wartet ein Teilnehmer auf den Empfang des Tokens und hängt beim Weitersenden seine eigenen Daten an.

Dieses System hat Vorteile gegenüber CSMA/CD:

- Echtzeit-Anwendungen sind möglich.
- Die zur Verfügung stehende Bandbreite wird vollständig ausgenutzt.

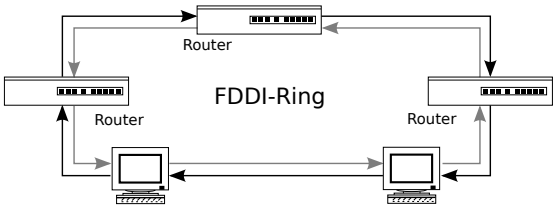
Insbesondere arbeitet ein 10-MBit-Token-Ring-System mehr als doppelt so schnell wie ein 10-MBit-Ethernet.

Ein Nachteil besteht darin, daß der Ausfall einer Verbindung den Ausfall des ganzen Netzes zur Folge hat.

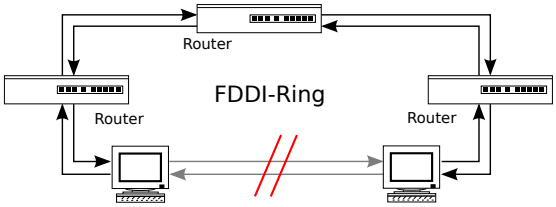
Token Ring war viele Jahre lang Standard bei IBM-Netzwerken, konnte sich jedoch letztlich gegen das erheblich kostengünstigere Ethernet nicht behaupten.

Ein dem Token Ring ähnliches Verfahren kommt nach wie vor in Glasfasernetzen zum Einsatz.

Der Standard *Fiber Distributed Data Interface (FDDI)* löst das Problem der Kollisionen ebenfalls durch ein kreisendes Token. Um dem Problem zu begegnen, daß der Ausfall einer Verbindung den Ausfall des ganzen Netzes nach sich zieht, gibt es FDDI-Netze mit einem zusätzlichen „Reserve-Ring“ in Gegenrichtung:

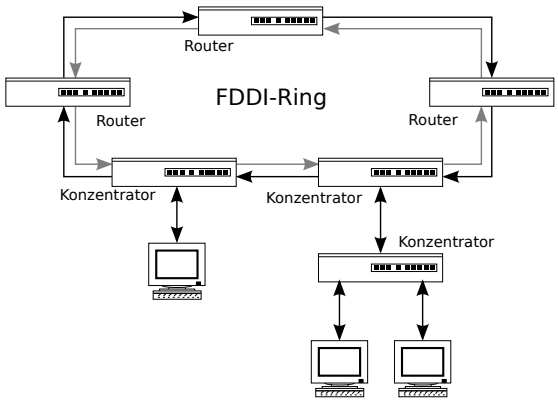


Fällt eine Verbindung aus, kann durch Hinzuziehen des Reserve-Rings wieder eine Ringstruktur aufgebaut werden.



Geräte, die an einem doppelten FDDI-Ring teilnehmen können, heißen *Klasse-A-Geräte*.

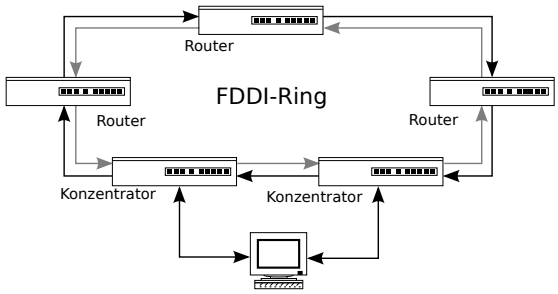
Für weniger wichtige Systeme genügt es oft, sie mit nur einer Verbindung an das FDDI-Netz anzuschließen.



Solche Geräte heißen *Klasse-B-Geräte*.

Ein Gerät, das FDDI-Netze miteinander verbindet (analog zum Hub oder Switch bei Ethernet) heißt *Konzentrator*.

Besonders wichtige Geräte (z. B. Server-Rechner) können mit zwei FDDI-Netzwerkadaptern ausgestattet und an zwei Konzentratoren angeschlossen werden, so daß sie auch bei einem größeren Ausfall noch am Netz teilnehmen.



Dieses Verfahren heißt *Dual Homing*.

3.2.4 Kollisionsvermeidung: CSMA/CA (WLAN)

Auch bei Funk-Netzwerken wie z. B. *Wireless Local Area Network (WLAN)* besteht das Problem von Kollisionen.

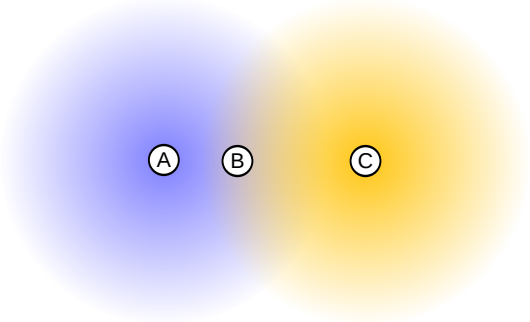
Da es nicht möglich ist, gleichzeitig zu senden und auf Daten anderer Teilnehmer zu lauschen, kann das CSMA/CD-Verfahren (Kollisionserkennung) nicht eingesetzt werden. Stattdessen müssen Kollisionen so weit wie möglich vermieden werden: *Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA)*.

- Ein Teilnehmer, der Daten senden will, muß vorher eine bestimmte Zeit lang (*Distributed Coordination Function Interframe Spacing (DIFS)*) plus eine zufällig Zeitspanne lang warten und prüfen, ob bereits eine Datenübertragung stattfindet.
- Nur wenn das Übertragungsmedium nach dieser Wartezeit immer noch frei ist, wird mit dem Senden begonnen.

Auf diese Weise werden Kollisionen zwar nicht völlig ausgeschlossen, aber doch wesentlich reduziert.

Hidden Station

Ein spezielles Problem von Funknetzwerken ist das Problem der verborgenen Station – *Hidden Station*.



In der links dargestellten Situation sendet Station A an B. C befindet sich außerhalb der Reichweite von A, bekommt also von dieser Datenübertragung nichts mit.

Trotz Anwendung von CSMA/CA wird C daher an B senden, was eine Kollision mit den von A gesendeten Daten zur Folge hat.

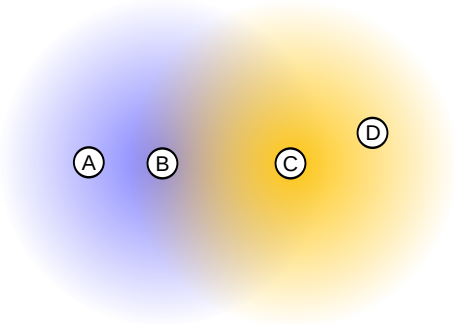
Durch Koordination der Datenübertragung mittelss *RTS/CTS* läßt sich dieses Problem reduzieren: Vor dem Senden der eigentlichen Daten sendet A ein Signal *Ready To Send (RTS)* an die Station B. Diese antwortet mit *Clear to Send (CTS)*. C empfängt das CTS von B, nicht jedoch das RTS von A. Daran kann C erkennen, daß eine Station A außerhalb seiner Reichweite Daten an B senden will, und mit der eigenen Datenübertragung warten.

Zusammen mit den RTS/CTS-Signalen senden A und B die Information, wie lange der Kanal belegt sein wird. Dadurch kann C vermeiden, einen neuen Datenübertragungsversuch zu früh zu starten.

Auch hier werden Kollisionen nicht völlig ausgeschlossen, aber reduziert.

Exposed Station

Ein weiteres spezielles Problem von Funknetzwerken ist das Problem der ausgesetzten Station – *Exposed Station*.



In der links dargestellten Situation sendet Station B an A. Gleichzeitig will Station C an D senden.

Bei Anwendung von CSMA/CA wird C mit der Datenübertragung warten, um die laufende Verbindung nicht zu stören. Tatsächlich wäre dies aber gar nicht nötig, da sich A außerhalb der Reichweite von C befindet, also gar nicht gestört werden kann.

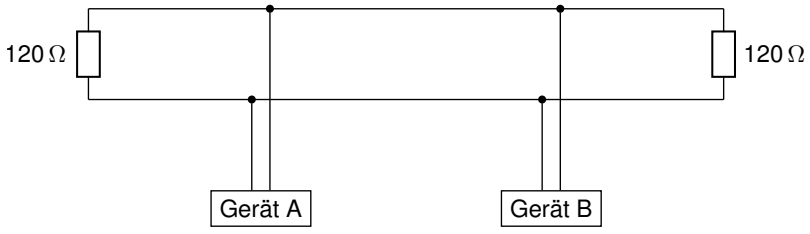
Auch dieses Problem läßt sich durch Koordination der Datenübertragung mittels RTS/CTS reduzieren: C empfängt das RTS von B, nicht jedoch das zugehörige CTS von A. Daran kann C erkennen, daß sich Station A außerhalb seiner Reichweite befindet und seine Datenübertragung an D sofort starten, ohne die Datenübertragung von B nach A zu stören.

3.2.5 Kollisionsauflösung: CSMA/CR (CAN-Bus)

CAN-Bus

Controller Area Network
Anwendungen: Steuergeräte, z. B. in Automobilen

- Problem: Kollisionen
- Lösung: CSMA/CR
  Carrier Sense Multiple Access / Collision Resolution



logisch 1: Ruhezustand – *rezessiv*
logisch 0: Spannungsdifferenz $\longrightarrow$ Vorrang vor 1 – *dominant*

logische Und-Verknüpfung – *Wired And*

Priorität senden, Gesendetes lesen

- stimmt überein $\longrightarrow$ weitersenden
- stimmt nicht überein $\longrightarrow$ warten

$\longrightarrow$ Priorität 0 kann sofort senden

A sendet Priorität 3: 0 0 1 1
B sendet Priorität 5: 0 1 0 1
↑
B hört auf, A sendet weiter

3.3 Fehlererkennung und -korrektur

3.3.1 Fehlererkennung durch Parität

In Zusammenhang mit dem RS-232-Protokoll haben wir bereits das Konzept der *Parität* kennengelernt: Zusätzlich zu den eigentlichen Daten wird ein Paritäts-Bit mitgesendet.

- Bei *gerader Parität* ist eine Nachricht nur dann zulässig, wenn ihre Quersumme – einschließlich Paritäts-Bit – gerade ist.
- Bei *ungerader Parität* ist eine Nachricht nur dann zulässig, wenn ihre Quersumme – einschließlich Paritäts-Bit – ungerade ist.

Beispiel:

Nachricht:	11010110
gerade Parität:	zusätzliches Bit, damit die Quersumme gerade ist: 1
kodierte Nachricht:	110101101
ungerade Parität:	zusätzliches Bit, damit die Quersumme ungerade ist: 0
kodierte Nachricht:	110101100

Wenn während der Übertragung der Nachricht ein einzelnes Bit (genauer: eine ungerade Anzahl von Bits) fehlerhaft übertragen wird, kann der Empfänger dies anhand der Parität erkennen und ein erneutes Senden der Daten anfordern.

3.3.2 Ausfallsicherheit durch Parität: RAID

Ein weiterer Anwendungsfall der Parität lautet: Wenn von einer Nachricht ein Bit verlorengegangen ist, läßt es sich mit Hilfe des Paritäts-Bits rekonstruieren.

Mit anderen Worten: Wenn eine Nachricht der Länge n zuzüglich Parität in insgesamt $n+1$ Bits übertragen wird, genügen dem Empfänger beliebige n Bits, um die Nachricht zu rekonstruieren.

Voraussetzung hierbei ist, daß man von den empfangenen n Bits weiß, daß sie korrekt sind.

Das o. a. Prinzip findet seine Anwendung in der redundanten Datenspeicherung auf Festplatten: *Redundant Array of Independent Disks (RAID)*. (Die Abkürzung stand ursprünglich für „inexpensive disks“, wurde aber aus Marketing-Gründen uminterpretiert.)

Sobald mindestens 3 Festplatten zur Verfügung stehen, kann man diese zu einem sog. *RAID-Verband* zusammenschließen.

Um einen Datenblock A auf drei Festplatten zu speichern, unterteilt man ihn in *zwei* Unterblöcke A1 und A2, also einen Unterblock weniger, als Festplatten zur Verfügung stehen.

Die Daten werden dann wie folgt auf die Festplatten aufgeteilt:

Platte 1	Platte 2	Platte 3
A1	A2	A3 = A1 xor A2

Die xor-Verknüpfung ist äquivalent zur Berechnung des geraden Paritäts-Bits, also zum Bilden der Quersumme modulo 2:

```
      0101   A1
xor   0011   A2
      ----
      0110   A3
```

Die Rekonstruktion eines verlorengegangenen Datenblocks erfolgt ebenfalls mit Hilfe der xor-Operation. Wenn z. B. Festplatte Nr. 2 defekt ist, können die verlorengegangenen Daten A2 wie folgt rekonstruiert werden:

```
      0101   A1
xor   0110   A3
      ----
      0011   A2
```

Wenn die Speicherung der Daten bitweise erfolgt, heißt ein derartiger Festplattenverband *RAID 3*; wenn sie blockweise erfolgt, *RAID 4*.

In der Praxis wurden *RAID 3* und *RAID 4* durch *RAID 5* ersetzt: Wenn die Paritätsinformation immer auf derselben Platte gespeichert wird, nimmt diese an jeder Schreiboperation teil. Sie stellt somit einen „Flaschenhals“ dar und ist höherem Verschleiß ausgesetzt als die anderen Platten.

Um dies zu vermeiden, wird bei RAID 5 die Paritätsinformation blockweise gleichmäßig über alle Platten verteilt:

Platte 1	Platte 2	Platte 3
A1	A2	A3 = A1 xor A2
B1 = B2 xor B3	B2	B3
C1	C2 = C1 xor C3	C3
D1	D2	D3 = D1 xor D2
...	...	...

Durch Verwendung fortgeschrittenerer Techniken anstelle einer einfachen Paritätsbildung ist es auch möglich, RAID-Verbände zu konstruieren, die den Verlust von mehr als einer Festplatte auffangen können.

3.3.3 Fehlerkorrektur durch Hamming-Code

Der *Hamming-Abstand zweier Nachrichten* ist definiert als die Anzahl der Bits, in denen sich die beiden Nachrichten unterscheiden.

Beispiel:

```
Nachricht:            11010110
defekte Nachricht:    11000110   Hamming-Abstand 1
```

Wenn man zur Übertragung von Nachrichten „zu viele“ Symbole wählt, so daß Symbole mit Hamming-Abstand 1 für dieselbe Nachricht stehen, ist es möglich, 1 fehlerhaft übertragenes Bit zu korrigieren.

Ein Code, der Symbole der Länge n Bits verwendet, um Nachrichten der Länge m Bits zu übertragen, heißt *(n, m)-Hamming-Code*.

(3, 1)-Hamming-Code

Jeder 1-Bit-Nachricht wird ein Symbol der Länge 3 Bits zugeordnet, z. B. durch Verdreifachung des Bits: Eine 1 wird durch die Bitfolge 111, eine 0 durch die Bitfolge 000 kodiert.

Eine auf diese Weise übertragene Nachricht, kann durch „Mehrheitsbeschluß“ rekonstruiert werden, sofern nicht mehr als 1 Bit pro Symbol fehlerhaft übertragen wurde. Beispiel:

```
Nachricht:            1  1  0  1  0  1  1  0
kodierte Nachricht:   111111000111000111111000
defekte Nachricht:    111111010111000110111010
Rekonstruktion:       1  1  0  1  0  1  1  0
```

Erst wenn 2 Bits in demselben Symbol fehlerhaft übertragen werden, wird die Nachricht fehlerhaft rekonstruiert.

(5, 1)-Hamming-Code

Jeder 1-Bit-Nachricht wird ein Symbol der Länge 3 Bits zugeordnet, z. B. durch Verfünffachung des Bits: Eine 1 wird durch die Bitfolge 11111, eine 0 durch die Bitfolge 00000 kodiert. Die Rekonstruktion einer fehlerhaft übertragenen Nachricht erfolgt wieder durch „Mehrheitsbeschluß“. Hierbei können zwei fehlerhafte Bits aufgefangen werden.

Der (3, 1)- und der (5, 1)-Hamming-Code verwenden unverhältnismäßig viele zusätzliche Bits. Die folgenden Codes arbeiten effizienter.

(7, 4)-Hamming-Code

Durch geschickte Kombination mehrerer Paritäts-Bits ist es möglich, 4 Bits so durch 7 Bits zu kodieren, daß ein fehlerhaft übertragenes Bit an beliebiger Position korrigiert werden kann.

Die übliche Vorgehensweise lautet:

- Wir wählen eine Zweierpotenz minus 1 als Länge für die kodierte Nachricht, hier $7 = 2^3 - 1$.
- Wir numerieren die Bits der kodierten Nachricht von 1 bis 7 durch.
- Die Bits an den Positionen von Zweierpotenzen, hier 1, 2 und 4, sind Paritäts-Bits, die anderen (hier: 4 Bits) enthalten die Nutzdaten.

  p p d p d d d
- Zur besseren Orientierung ergänzen wir noch ein Bit Nr. 0, das aber nicht benutzt wird.

  x p p d p d d d
- Wir positionieren die Nutzdaten (z. B. 1011) und lassen die Paritäts-Bits zunächst offen.

  x p p 1 p 0 1 1
- Das erste Paritäts-Bit steht auf Position Nr. 1, also der zweiten. (Position Nr. 0 ist, wie gesagt, ungenutzt und dient nur zur Orientierung.) Um es zu setzen, bilden wir eine gerade Parität über *jedes zweite* Bit, d. h., wir setzen das Paritäts-Bit so, daß sich an diesen Positionen eine gerade Anzahl von Einsen befindet.

  x 0 p 1 p 0 1 1
- Das zweite Paritäts-Bit steht auf Position Nr. 2, also der dritten. Um es zu setzen, bilden wir eine gerade Parität über *jedes dritte und vierte von jeweils vier* Bits:

  x 0 1 1 p 0 1 1
- Das dritte Paritäts-Bit steht auf Position Nr. $4 = 2^2$. Um es zu setzen, bilden wir eine gerade Parität über *jedes fünfte bis siebte von jeweils acht* Bits:

  x 0 1 1 0 0 1 1
- Durch Weglassen des ungenutzten Bits Nr. 0 erhalten wir die kodierte Nachricht:

  0 1 1 0 0 1 1

Die Muster, nach denen die Paritäten gebildet werden, entsprechen genau den Positionen der 1-Bits, wenn man binär von 0 bis 7 zählt:

```
0 1 2 3 4 5 6 7
0 1 0 1 0 1 0 1
0 0 1 1 0 0 1 1
0 0 0 0 1 1 1 1
```

Um nun eine Nachricht mit einem Bit-Fehler zu rekonstruieren, bildet der Empfänger alle drei Paritäten. Wenn diese alle in Ordnung sind, ist auch die Nachricht in Ordnung. Wenn eine Parität oder mehrere Paritäten fehlerhaft sind, können wir anhand der betroffenen Paritäten rekonstruieren, welches Bit den Fehler enthält. Da Bits nur zwei Zustände annehmen können, können wir dann den Fehler korrigieren.

Beispiel:

Nachricht:	1 011
kodierte Nachricht (s. o.):	0110011
defekte Nachricht:	0111011

Zur Rekonstruktion prüfen wir alle drei Paritäten:

```
p p d p d d d
0 1 1 1 0 1 1
^   ^   ^   ^   korrekt
  ^ ^       ^ ^ korrekt
      ^ ^ ^ ^   inkorrekt
```

Da nur die dritte Parität inkorrekt ist, muß sich das fehlerhafte Bit an einer Stelle befinden, die an der Bildung der dritten Parität teilnimmt, jedoch nicht an der Bildung der ersten beiden Paritäten. Dies trifft nur auf ein Bit zu, nämlich das vierte. Wir können also das vierte Bit als „fehlerhaft übertragen“ identifizieren und durch Umklappen dieses Bits die kodierte Nachricht 0110011 und damit die ursprüngliche Nachricht 1011 rekonstruieren.

Die Rekonstruktion funktioniert nicht mehr, sobald mehr als ein Bit fehlerhaft übertragen wurde.

Aufgabe

Erstellen und testen Sie einen Hamming-Code mit vier Paritäts-Bits.
Um welchen (?,?)-Hamming-Code handelt es sich?

Lösung

Mit vier Paritäts-Bits lassen sich Binärzahlen bis 15 ausdrücken. Wir können also kodierte Nachrichten der Länge 15 Bits verwenden. Abzüglich der 4 Paritäts-Bits verbleiben 11 Datenbits. Es handelt sich somit um einen $(15, 11)$-Hamming-Code.

Wenn wir, wie üblich, für die Positionen der Paritäts-Bits Zweierpotenzen wählen, lauten die Positionen der Daten- und Paritäts-Bits sowie die Paritäts-Gruppen wie folgt:

```
p p d p d d d p d d d d d d d
^   ^   ^   ^   ^   ^   ^   ^
  ^ ^       ^ ^       ^ ^       ^ ^
      ^ ^ ^ ^           ^ ^ ^ ^
              ^ ^ ^ ^ ^ ^ ^ ^
```

Um den Code zu testen, kodieren wir eine beliebige aus elf Bits bestehende Nachricht,
z. B. 1 0 1 1 0 1 0 0 1 1 1.

```
Nachricht:                1   0 1 1   0 1 0 0 1 1 1

kodierte Nachricht:   0 0 1 1 0 1 1 0 0 1 0 0 1 1 1
                      ^   ^   ^   ^   ^   ^   ^   ^
                        ^ ^       ^ ^       ^ ^       ^ ^
                            ^ ^ ^ ^           ^ ^ ^ ^
                                    ^ ^ ^ ^ ^ ^ ^ ^

defekte Nachricht:    0 0 1 1 0 1 1 0 0 1 0 0 0 1 1
                      ^   ^   ^   ^   ^   ^   ^   ^   1
                        ^ ^       ^ ^       ^ ^       ^ ^   0
                            ^ ^ ^ ^           ^ ^ ^ ^       1
                                    ^ ^ ^ ^ ^ ^ ^ ^ ^       1
                                              !
```

In der defekten Nachricht erkennen wir anhand der Paritäten, welches Bit fehlerhaft übertragen wurde. Daher können wir die kodierte Nachricht und somit auch die ursprüngliche Nachricht rekonstruieren.

3.3.4 Fehlererkennung durch CRC

Der *Cyclic Redundancy Check (CRC)* stellt eine Verallgemeinerung der Parität dar. Obwohl es sich nicht um eine Summe handelt, wird CRC häufig als „Prüfsumme“ bezeichnet.

Bei gerader [ungerader] Parität lautet das Prinzip: „Gültige Nachrichten müssen [dürfen nicht] durch 2 teilbar sein.“ Beides läßt sich zusammenfassen zu: „Gültige Nachrichten müssen bei Division durch 2 einen vorgegebenen Rest haben.“

Wenn wir nicht durch 2, sondern durch „etwas Größeres als 2“ dividieren, erhalten wir einen Prüfwert, der mehr als nur ein einzelnes fehlerhaftes Bit erkennen kann.

Bei CRC lautet nun das Prinzip: „Gültige Nachrichten müssen bei Division durch ein vorgegebenes Polynom einen vorgegebenen Rest haben.“

Die Bits einer Nachricht werden hier als Koeffizienten eines Polynoms aufgefaßt, z. B. steht die Bitfolge 1101 für das Polynom $1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^3 + x^2 + 1$.

Die Variable x ist keine relle Zahl, sondern sie entstammt dem endlichen Körper $GF(2)$, d. h. sie kann nur die Werte 0 und 1 annehmen, und es gilt: $1 + 1 = 0$. Die Polynomdivision hat dann gegenüber einer herkömmlichen Division den Vorteil, daß an die Stelle von Subtraktionen xor-Operationen treten, die sich in elektronischen Schaltungen wesentlich leichter realisieren lassen.

Um eine Nachricht zu kodieren, legen sich Sender und Empfänger zunächst auf ein gemeinsames *CRC-Polynom* (auch: *Generator-Polynom*) fest, z. B. $x^3 + x^2 + 1$, was der Bit-Folge 1101 entspricht.

Nun gilt es, eine kodierte Nachricht zu erzeugen, die durch das CRC-Polynom teilbar ist. Dies erfolgt durch Anhängen von so vielen Bits, wie der Rest höchstens lang werden kann. Dies ist ein Bit weniger, als die Bit-Folge des Polynoms lang ist, gleichbedeutend mit der höchsten im Polynom vorkommenden Potenz, dem sog. *Grad des Polynoms*.

Um nun für eine gegebene Nachricht, z. B. 10110100111, die anzuhängenden Bits zu berechnen, hängen wir eine entsprechende Anzahl von Nullen an und führen die Polynomdivision durch. Der verbleibende Rest sind die anzuhängenden Bits.

Man beachte, daß es sich bei den durch einen horizontalen Strich notierten Operationen nicht um Subtraktionen, sondern um xor-Operationen handelt.

```
1 0 1 1 0 1 0 0 1 1 1   0 0 0   :   1 1 0 1
1 1 0 1
_______
  1 1 0
  1 1 0 1
  _______
  0 0 0 1 1 0 0
        1 1 0 1
        _______
              1 1 1 1
              1 1 0 1
              _______
                  1 0   0 0
                  1 1   0 1
                  _________
                    1   0 1 0
                    1   1 0 1
                    _________
                        1 1 1
```

Die kodierte Nachricht lautet also: 10110100111111. (Es wurden die Bits 111 angehängt.)

Um zu prüfen, daß die Nachricht korrekt übertragen wurde, führen wir dieselbe Polynomdivision mit der kodierten Nachricht aus. Wenn der Rest 0 herauskommt, war die Übertragung korrekt.

```
1 0 1 1 0 1 0 0 1 1 1   1 1 1   :   1 1 0 1
1 1 0 1
-------
  1 1 0
  1 1 0 1
  -------
        1 1 0 0
        1 1 0 1
        -------
              1 1 1 1
              1 1 0 1
              -------
                  1 0   1 1
                  1 1   0 1
                  -------
                    1   1 0 1
                    1   1 0 1
                    ---------
                            0
```

Das CRC-Verfahren ist ein sehr leistungsfähiges Fehlererkennungsverfahren. Nur dann, wenn eine fehlerhafte Nachricht zufällig durch dasselbe Polynom teilbar ist wie die korrekte Nachricht, bleibt ein Fehler unerkannt. Wenn das CRC-Polynom in Hinblick auf „typische“ Fehler geschickt gewählt wird, ist dies äußerst unwahrscheinlich.

CRC eignet sich hingegen *nicht* als Hash-Wert, für z. B. kryptographische Anwendungen (digitale Signatur). Wer *mit Absicht* eine Nachricht konstruieren will, die denselben CRC-Rest hat wie eine andere, gegebene Nachricht, kann dies ohne nennenswerte Hindernisse tun.

Das CRC-Verfahren *CRC-32* mit dem standardisierten Polynom $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0$ wird zur Fehlererkennung im Ethernet-Protokoll verwendet.

Weitere Anwendung findet CRC u. a. bei USB, Bluetooth, ISDN, GSM, RFID, SD-Karten sowie bei verschiedenen komprimierten und/oder verschlüsselten Datenformaten.

3.3.5 Ausfall- und Fehlerkorrektur durch Reed-Solomon-Code

Um in einer Situation, in der mit Übertragungsausfällen gerechnet werden muß, n Zahlen zu übertragen, eignet sich die folgende Vorgehensweise:

- Man verwendet die n Zahlen als Koeffizienten eines Polynoms (vom Grad $n-1$).
- Sender und Empfänger einigen sich auf $m > n$ Stützstellen.
- Man überträgt die m Funktionswerte des Polynoms. Wenn mindestens n dieser Werte durchkommen, kann man ein Polynom entsprechenden Grades hindurchlegen und auf diese Weise die Koeffizienten, also die zu übertragenden n Zahlen rekonstruieren.

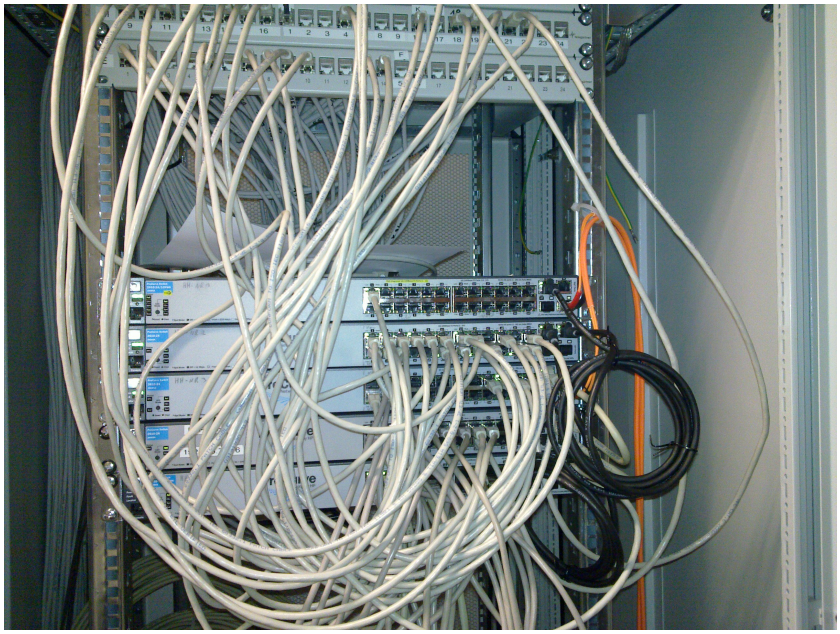
(Beispiel: Für $n = 3$ ist das Polynom eine Parabel. Eine Parabel ist durch 3 Punkte eindeutig bestimmt.)

Mit größerem mathematischen Aufwand ist auch die Korrektur fehlerhaft übertragener Werte möglich. (Zur Illustration: Wenn zu $n+1$ korrekten Punkten ein fehlerhafter Punkt hinzukommt, legen diese $n+2$ Punkte eindeutig ein Polynom fest. Dieses hat aber mit hoher Wahrscheinlichkeit einen höheren Grad als $n-1$. Da aber alle korrekten Punkte dasselbe Polynom vom Grad $n-1$ beschreiben, läßt sich der fehlerhafte Punkt im Prinzip per „Mehrheitsentscheid“ herausfiltern – bzw. in der Praxis durch eine Rechnung ermitteln.)

Wie bei CRC, rechnet man auch bei der Reed-Solomon-Kodierung nicht mit rellen Zahlen, sondern mit endlichen Körpern.

Der Reed-Solomon-Code kommt u. a. bei Audio-CDs, Mobilfunk, digitalem Rundfunk und bei der Kommunikation mit Raumsonden zum Einsatz.

3.4 Strukturierte Verkabelung



Universelle Gebäudeverkabelung (UGV) gemäß EN 50173-1

Teil der Gebäudeinfrastruktur

- sicher
- zuverlässig
- erweiterbar

} redundant auslegen, Leistungsreserve

- anwendungsneutral

Primärbereich: zwischen Gebäuden, bis 2 km

- Glasfaserkabel (Potentialtrennung!)
- DSL über eigene Kupferkabel

Sekundärbereich: zwischen Stockwerken, bis 500 m

- Glasfaser- oder Kupferkabel

Tertiärbereich: innerhalb eines Stockwerks, bis 100 m

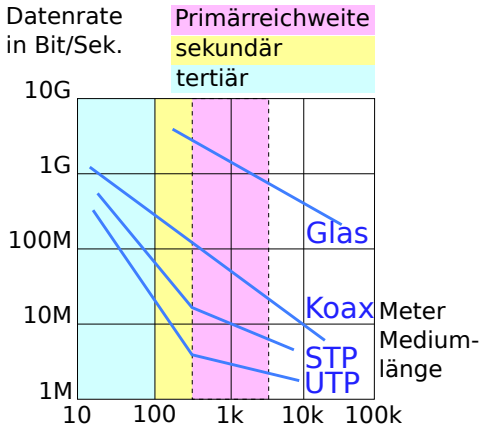
- Twisted-Pair-Kabel

Endpunkte: Standort-, Gebäude-, Etagenverteiler

- Patch-Felder, Switches, Router

Warum Glasfaserkabel?

- Kein Potentialausgleich erforderlich
- Auch im km-Bereich noch GBit-Übertragung



Verkabelung einer Anlage

Beispiel: Flugzeugkabinensimulator

- Haupt-PC ⟷ Feldbus:
  Ethernet (TP)
  (früher: CAN-Bus)
- Feldbus ⟷ Verbraucher:
  mehradrige Leitungen
- Feldbus ⟷ Beleuchtung:
  DALI-Bus
- Haupt-PC ⟷ Tür-PC:
  Ethernet (TP)
- Tür-PC ⟷ Tür-Steuerung:
  CAN-Bus
- Tür-Steuerung ⟷ Motoren:
  spezielle Leitungen

3.5 Zugänge zum Internet

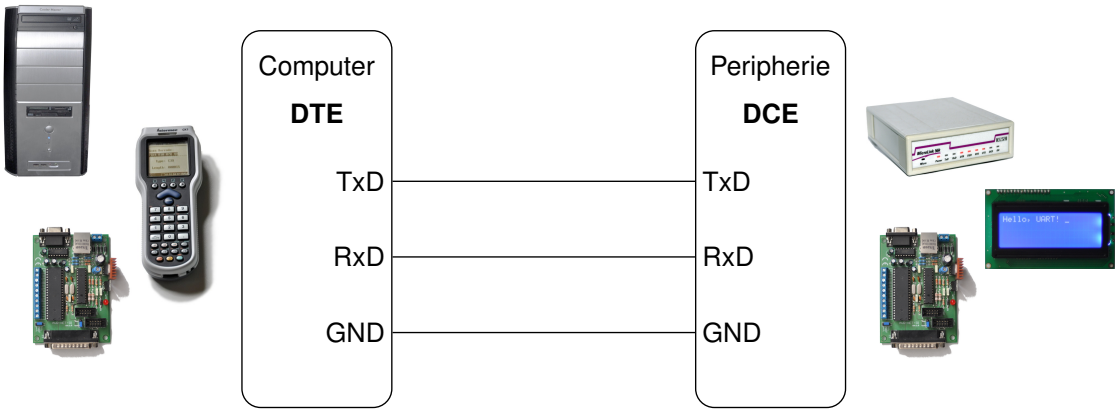
- Ethernet
- WLAN
- RS-232 mit PPP
- Bluetooth mit RS-232-Emulation und PPP
  oder mit PAN
- PPP über ein Programm, speziell: PPPoE
- USB (spezielles Gerät oder spezielles Kabel)
- FireWire

3.5.1 RS-232

Über die serielle Schnittstelle (RS-232) können beliebige Geräte miteinander kommunizieren. Mit Hilfe des Point-to-Point-Protocols (PPP, siehe Abschnitt 3.5.2) kann eine RS-233-Verbindung als Internet-Zugang genutzt werden.

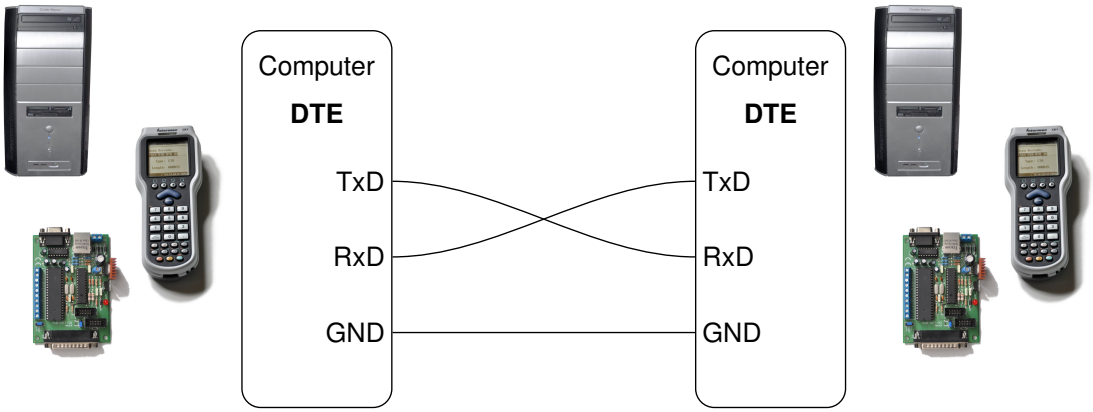
In der Praxis ist es wichtig, daß beide Geräte dieselben Kommunikationsparameter verwenden. Dies betrifft die Übertragungsgeschwindigkeit und Parität (siehe Abschnitt 3.1.1), aber auch die Verkabelungskonvention.

Im „klassischen“ Fall verbindet RS-232 einen Computer (PC, eingebettetes System, ...) mit einem Peripherie-Gerät (Modem, Display, ...). In dieser „asymmetrischen“ Situation verwendet man Kabel, die die Anschlüsse beider Geräte 1-zu-1 miteinander verbinden. Die Anschlüsse des Peripherie-Geräts sind dabei so beschaltet, daß durch die 1-zu-1-Verbindung der Datenausgang des Computers mit dem Dateneingang des Peripherie-Geräts verbunden wird und der Datenausgang des Peripherie-Geräts mit dem Dateneingang des Computers.



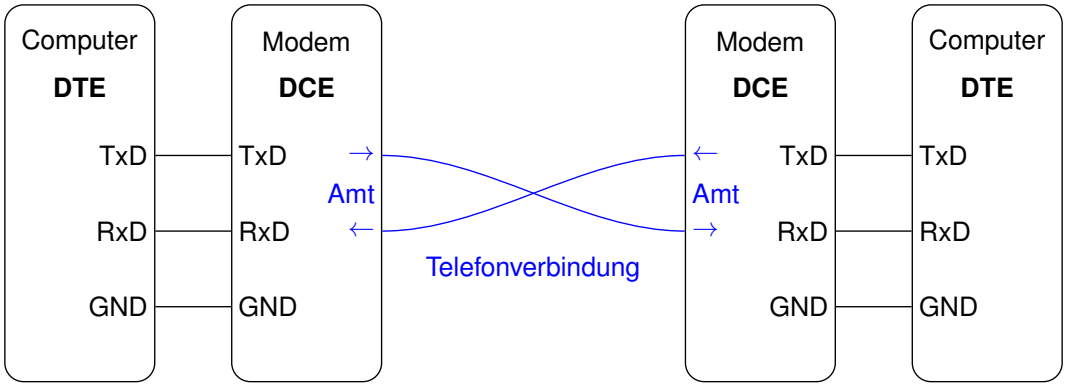
„TxD“ steht für „transmit data“, „RxD“ für „receive data“.

Wenn man stattdessen zwei gleichberechtigte Geräte direkt per RS-232 miteinander verbindet, müssen die Leitungen überkreuz angeschlossen werden.



Ein derartiges „gekreuztes“ heißt *Null-Modem-Kabel*.

Diese Bezeichnung kommt daher, daß ein gekreuztes Kabel gewissermaßen zwei Modems „ersetzt“: Bei einer Verbindung zweier Rechner über zwei Modems sorgen die Modems dafür, daß sich die Datenströme überkreuzen, d. h. sie sorgen dafür, daß ausgehende Daten des einen Rechners beim anderen Rechner als eingehende Daten eintreffen.



3.5.2 Point-to-Point Protocol (PPP)

PPP über Schnittstelle, z. B. /dev/ttyS0 auf Rechner A, /dev/ttyS3 auf Rechner B

Auf Rechner A:

```
# pppd /dev/ttyS0 192.168.42.1:192.168.42.2 debug nodetach noauth
```

Auf Rechner B:

```
# pppd /dev/ttyS3 passive debug nodetach noauth
```

Erklärung der Optionen:

IP:*IP*	IP-Adressen zuweisen (erste: eigener Rechner, zweite: für Partner)
`passive`	auf IP-Adressen-Zuweisung warten
`noauth`	auf Passwörter o. ä. verzichten
`debug`	Debug-Informationen ausgeben
`nodetach`	nicht in den Hintergrund gehen

Siehe auch: `man pppd`

3.5.3 Bluetooth

`hciconfig`	Bluetooth-Schnittstellen anzeigen
`hciconfig hci0 up`	Bluetooth-Schnittstelle hci0 in Betrieb nehmen
`hciconfig hci0 piscan`	Bluetooth-Schnittstelle hci0 suchbar machen
`hcitool scan`	nach möglichen Partnern suchen
`rfcomm -r listen 0 1`	RS-232 emulieren (Server)
`-r`	„raw“ – Rohdaten übertragen
`listen`	auf Verbindung der Gegenstelle warten
`0`	Gerätedatei /dev/rfcomm0 nennen
`1`	Kanal 1 benutzen
`rfcomm -r connect 0` *Adresse* `1`	RS-232 emulieren (Client)
`-r`	„raw“ – Rohdaten übertragen
`connect`	mit Gegenstelle verbinden
`0`	Gerätedatei /dev/rfcomm0 nennen
Adresse	Bluetooth-„MAC“-Adresse der Gegenstelle
`1`	Kanal 1 benutzen

Siehe auch: `man hciconfig`, `man hcitool`, `man rfcomm`

Durch die emulierte RS-232-Schnittstelle kann man PPP über Bluetooth betreiben.

Alternative: PAN-Daemon
TCP/IP direkt über Bluetooth (ohne PPP)

`pand --listen --role NAP`	Server
`pand --connect` *Adresse*	Client
`ifconfig bnep0`	Schnittstelle

Mit den o. a. Adressen sind Bluetooth-Adressen gemeint, insbesondere keine IP-Adressen. Hardware- oder MAC-Adressen sehen genauso aus und werden durch `pand` mit Bluetooth-Adressen gemischt, werden aber technisch auf andere Weise verwendet.

3.5.4 PPP über Software, PPP over Ethernet (PPPoE)

Anstelle einer Schnittstelle (z. B. /dev/ttyS1) kann man `pppd` auch anweisen, ein anderes Programm aufzurufen und dessen Standard-Ein- und -Ausgabe für die Datenübertragung zu nutzen.

Die Option lautet: `pty "`*Befehl*`"`

Beispiel 1: PPP über TCP/IP (Netcat – `nc`)

Auf Rechner A:

```
# pppd pty "nc -l -p 1234" 192.168.44.17:192.168.44.18 noauth nodetach debug
```

Auf Rechner B:

```
# pppd pty "nc Rechner_A 1234" passive nodetach noauth debug
```

Beispiel 2: PPP over Ethernet – PPPoE

(Wird von Internet-Providern anstelle von direktem TCP/IP über Ethernet genutzt, um mehr Kontrolle über die Verbindung zu haben)

Auf Rechner A:

```
# echo "noauth" > /etc/ppp/pppoe-server-options
# ifconfig eth1 up
# pppoe-server -I eth1 -F -L 192.168.42.1 -R 192.168.42.2
```

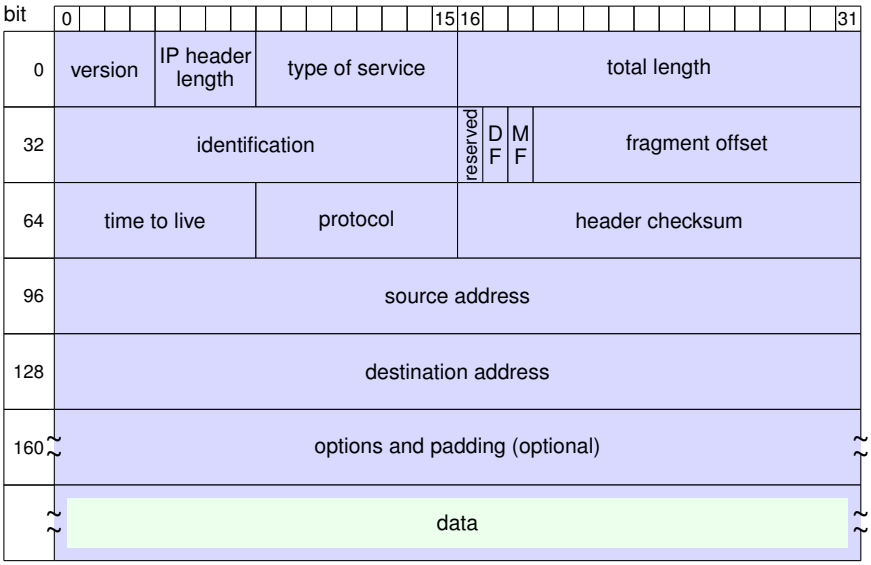
Auf Rechner B:

```
# ifconfig eth1 up
# pppd pty 'pppoe -I eth1' passive debug nodetach noauth
```

4 TCP/IP-Schicht 2: Internet-Protokolle (Vermittlung)

4.1 Internet Protocol, Version 4 (IPv4)

Das derzeit im Internet gebräuchlichste Protokoll ist das *Internet Protocol (IP)* in der Version 4 – *IPv4*. Es setzt auf den in Abschnitt 3.5 genannten Zugängen auf und vermittelt Datenpakete, hauptsächlich für die Transport-Protokolle TCP und UDP sowie ICMP.



- **version:** Der Wert 4 steht für IPv4.
- **IP header length:** Größe des IP-Headers, gemessen in 32-Bit-Worten
- **type of service:** Priorität
- **total length:** Länge des IP-Pakets (Header + Nutzdaten) in Bytes
- **identification:** Diese eindeutige Kennung dient dazu, fragmentierte Datagramme wieder zusammensetzen zu können.
- **DF**-Bit: „don't fragment". Die Nutzlast dieses IP-Pakets darf nicht fragmentiert werden, selbst wenn das IP-Paket deswegen einen ungünstigern Weg durch das Internet nehmen muß oder u. U. überhaupt nicht ankommt. Durch gezieltes Senden von IP-Paketen unterschiedlicher Länge mit gesetztem DF-Bit läßt sich ausloten, Pakete bis zu welcher Größe zwischen zwei Zielen unfragmentiert übertragen werden können.
- **MF**-Bit – „more fragments": Wenn dieses Bit gesetzt ist, sind die Nutzdaten mit dem vorliegenden IP-Paket nicht komplett, sondern es folgen weitere Fragmente.
- **fragment offset:** Speicheradresse in Einheiten von 8 Bytes, an der die Nutzdaten des vorliegenden IP-Pakets innerhalb der Gesamt-Nutzdaten abzulegen sind. Bei nicht fragmentierten Nutzdaten oder wenn es sich um das erste Fragment handelt, ist dieser Wert 0.
- **time to live:** Jeder Netzteilnehmer, der dieses IP-Paket weiterleitet, verringert diesen Wert um 1. Für jede Sekunde, die das Paket auf einem Gerät verweilt (was heutzutage praktisch nicht mehr vorkommt), wird nochmals 1 abgezogen. Sobald 0 erreicht wird, wird das IP-Paket gelöscht und der Absender per ICMP benarchrichtigt. Auf diese Weise können fehlgeleitete, endlos kreisende IP-Pakete aus dem Verkehr gezogen werden.

  Das Programm `traceroute` (unter Microsoft Windows: `tracert`) setzt absichtlich niedrige Time-to-live-Werte, um den Weg, den ein IP-Paket durch das Netz nimmt, zu verfolgen. (Für Rechner, die nicht antworten, werden Sterne angezeigt.)
- **protocol:** Art der Nutzdaten (z. B. TCP, UDP, ICMP)
- **header checksum:** Prüfsumme für den IP-Header

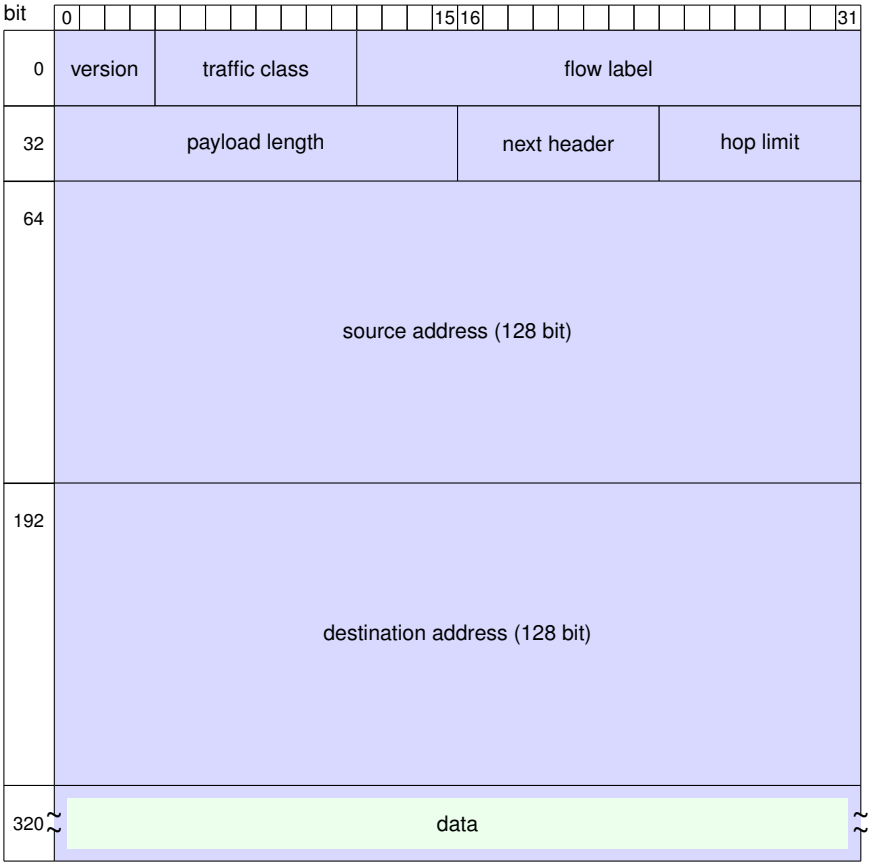
  Es handelt sich nicht um CRC oder ein ähnliches Verfahren, sondern tatsächlich um eine Summe. Dies hat den Vorteil, daß die Prüfsumme nicht komplett (in Hardware) neu berechnet werden muß, wenn sich die *time to live* beim Weiterleiten ändert.
- **source address, destination address:** IP-Adressen des Absenders und Empfängers, jeweils 32 Bit breit
- **options and padding:** Dieses optionale Feld kann zusätzliche Informationen enthalten, u. a. ein Bit, das das IP-Paket als „geheim" ausweist. Dies sollte ursprünglich dazu dienen, Weiterleitung durch „unsichere" Staaten zu vermeiden, bewirkt aber in der Praxis lediglich, daß Angreifer interessante Nachrichten schneller finden. Aus diesem Grunde wird das „geheim"-Bit nicht mehr verwendet.

4.2 Internet Protocol, Version 6 (IPv6)

Die Version 6 des Internet-Protokolls, *IPv6*, ist der Nachfolger von IPv4.

Wichtige Unterschiede:

- Der Vorrat von IPv4-Adressen ist ausgeschöpft. Die Größe einer IP-Adresse wurde daher von 32 auf 128 Bits erweitert.
- Der IPv6-Header hat eine konstante Länge. Dies vereinfacht die Verarbeitung in Hardware.
- IP wird zunehmend auch für eigentlich verbindungsorientierte Dienste (Multimedia-Streaming, VoIP-Telefonie) genutzt. Ein zusätzliches Header-Feld *flow label* ermöglicht daher ein Verbindungskonzept innerhalb von IPv6.



- **version:** Der Wert 6 steht für IPv6.
- **traffic class:** Priorität
- **flow label:** Dieser eindeutige Wert kennzeichnet eine „Verbindung“: Pakete mit gleichem *flow label* werden von Routern usw. auf gleiche Weise behandelt, nehmen also denselben Weg durch das Internet. Dies entspricht *ungefähr* dem Verhalten von Telefonverbindungen, die ja ebenfalls das ganze Gespräch über denselben Weg durch die Vermittlungsstellen nehmen. Man erhofft sich von IPv4, mittels *traffic class* und *flow label* verbesserte Qualität bei VoIP-Telefonie und Multimedia-Streaming anbieten zu können.
- **payload length:** Da der IPv6-Header im Gegensatz zum IPv4-Header eine konstante Länge hat, ersetzt dieses eine Feld die beiden Felder *IP header length* und *total length*.
- **next header:** Dieses Feld spezifiziert entweder das Nutzdaten-Protokoll (TCP, UDP, ICMP, . . .) oder einen zusätzlichen IPv6-Header (gleicher Größe), der einige weniger gebräuchliche Optionen enthält.
- **hop limit:** Dieses Feld ersetzt die *time to live*. Da Laufzeiten von über 1 Sekunde heutzutage keine Rolle mehr spielen, zählt dieses Feld nur noch die Anzahl der Weiterleitungen. Die Verweilzeit wird nicht mehr berücksichtigt.

Eine Prüfsumme über den IPv6-Header wird nicht mehr berechnet. Hier verläßt man sich jetzt ganz auf die Prüfmechanismen der darunterliegenden Protokolle, insbesondere TCP.

Spezielle IPv6-Adressen:

- **Link Local Unicast:** `fe80::/10` (`fe80`... bis `febf`...)
  Wird aus MAC-Adresse berechnet, gilt nur an einem Netzwerkadapter
- **Site Local Unicast (veraltet):** `fec0::/10` (`fec0`... bis `feff`...)
  Selbst wählbare private Adressen (analog zu `192.168.x.y`, `172.16.x.y` bis `172.31.x.y`, `10.x.y.z`)
  Nicht weltweit eindeutig, daher wieder verworfen
- **Unique Local Unicast:** `fc00::/7` (`fc`... und `fd`...)
  Ähnlich *Site Local Unicast*, aber weltweit eindeutig durch vorgeschriebenen Zufallsanteil

Siehe auch: http://de.wikipedia.org/wiki/IPv6#Adressbereiche

4.3 Netzmasken und Routing

Wie wir bereits auf Seite 12 gesehen haben, verwendet man *Netzmasken*, um die Größe von Subnetzen festzulegen.

Ein Rechner befindet sich genau dann in einem Netz, wenn die bitweise Und-Verknüpfung der IP-Adresse des Rechners mit der Netzmaske die IP-Adresse des Netzes ergibt. Zum Beispiel:

```
    192.168.   1.137
&   255.255.255.   0
    ---------------
    192.168.   1.   0
```

Die Netzmaske darf nicht nur die Zahlen 0 und 255 enthalten. Durch Verwendung von Zahlen mit weniger als acht 1-Bits kann man Sub-Netze mit weniger als 256 IP-Adressen definieren:

```
    192.168. 13. 37    <-- IP-Adresse des Rechners    37_10 = 00100101_2
&   255.255.255.248    <-- Maske des Netzes           248_10 = 11111000_2
    ---------------
    192.168. 13. 32    <-- IP-Adresse des Netzes      32_10 = 00100000_2
```

Die Netzmaske 255.255.255.248 hat in Binärschreibweise 29 Einsen und 3 Nullen. Das o. a. Subnetz kann daher auch 192.168.13.32/29 notiert werden.

Damit physikalisch getrennte Subnetze miteinander kommunizieren können, sind *IP-Forwarding* und *Routing* erforderlich. Dies wurde bereits ausführlich in Abschnitt 1.8 vorgestellt.

Dieses sog. *statische Routing* hat u. a. den Nachteil, daß jeder Netzteilnehmer die Route zu jedem anderen Netzteilnehmer kennen muß. Ein Mechanismus, der dies vereinfacht, ist *Network Address Translation (NAT)* – siehe Abschnitt 4.4.1.

Mit Hilfe des Programms `traceroute` können wir in einem Netz, in dem Routing zwischen Subnetzen stattfindet, feststellen, welche Wege die IP-Pakete zwischen zwei gegebenen Rechnern nehmen.

4.4 Paketfilter (Firewall)

Eine Software, die die Weiterleitung von IP-Paketen gezielt steuert, heißt *Paketfilter* oder *Firewall*. Sie besteht aus einem Satz von *Regeln*, die in Abhängigkeit von den Verbindungsparametern (Start- und Ziel-IP-Adresse, Start- und Ziel-Port, verwendetes Protokoll plus eventuell weitere Daten) die Weiterleitung eines IP-Pakets erlauben oder verbieten oder das IP-Paket in irgendeiner Weise modifizieren.

Eine Firewall arbeitet eng mit dem Betriebssystem-Kern zusammen und ist daher immer auf diesen abgestimmt. Unter Linux heißt die Firewall `iptables`, unter den freien Unix-Systemen der BSD-Familie `pf` (für „packet filter“). Für Microsoft Windows gibt es verschiedene Firewall-Implementationen verschiedener Hersteller. So verschieden diese Programme sein mögen: Das Grundprinzip ist stets dasselbe.

(Speziell unter Microsoft Windows werden auch Programme mit anderen Funktionen als „Firewall“ bezeichnet. Wenn technische Präzision gefragt ist, ist es daher besser, von „Paketfiltern“ zu sprechen.)

Beispiele für `iptables`:

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT DROP
```

Mit diesen Befehlen setzt man die sog. *Policy* für die folgenden Aktionen:

- Annahme eines IP-Pakets durch diesen Rechner (`INPUT`)
- Weiterleitung zwischen zwei Netzwerkschnittstellen (`FORWARD`)
- Weiterleitung zu einem anderen Rechner (`OUTPUT`)

Die Policy besagt, was zu tun ist, wenn keine ausdrückliche Regel zur Anwendung kommt. Die Standard-Policy lautet `ACCEPT`, d. h., die IP-Pakete werden angenommen und intern und nach außen weitergeleitet. Die o. a. Befehle ändern die Policy auf `DROP`, d. h., kein IP-Paket wird verarbeitet, sofern es nicht ausdrücklich durch eine Regel spezifiziert ist.

```
# iptables -A FORWARD -i eth0 -o ppp0 -j ACCEPT
```

Dieser Befehl hängt an die *Forward-Chain*, also die Regel-Kette für das Weiterleiten zwischen Netzwerkschnittstellen, eine Regel an. Die Regel besagt, daß Netzwerkverkehr, der über die Schnittstelle `eth0` empfangen wird und deren Empfänger über die Schnittstelle `ppp0` erreichbar ist, durchgereicht werden soll. Mit einer derartigen Regel kann man z. B. einem Netz von Büro-Rechnern (`eth0`) unbegrenzten Zugriff auf das Internet (`ppp0`) gewähren.

```
# iptables -A FORWARD -p tcp --dport 25 -i eth0 -o ppp0 -j ACCEPT
```

Diese Variante der o. a. Regel beschränkt den Internet-Zugriff auf das Versenden von E-Mails (TCP-Port 25).

```
# iptables -A FORWARD -p tcp -d Rechner --dport 25 -i eth0 -o ppp0 -j ACCEPT
```

Diese Variante der o. a. Regel beschränkt das Versenden von E-Mails weiter auf einen bestimmten Rechner, z. B. den Server des eigenen Providers für ausgehende E-Mail.

Bei den Regeln kommt es auf die Reihenfolge an. Mit `-A` wird eine Regel hinten angefügt, mit `-I` vorne.

Für Detail-Informationen zu `iptables` siehe `man iptables` sowie eine reichhaltige Auswahl an Literatur, z. B. http://de.wikibooks.org/wiki/Linux-Kompendium:_Linux-Firewall_mit_IP-Tables.

Ein anonymisiertes Beispiel für eine reale Paketfilter-Konfiguration finden Sie in der Datei netfilter. Dieses Beispiel hat die folgenden Besonderheiten:

- Die Rechner im LAN-Subnetz („Büro-Rechner“) sind voll vertrauenswürdig und erhalten uneingeschränkten Zugriff auf das Internet. Dies ist nur in den wenigsten Büros sinnvoll.
- Die Rechner im DMZ-Subnetz („Server“) stellen E-Mail an die Büro-Rechner direkt per SMTP (TCP-Port 25) zu. Im Normalfall würden sich die Büro-Rechner die E-Mail per IMAP (Port 143) oder POP3 (Port 110) von den Servern holen.

4.4.1 Network Address Translation (NAT)

Paketfilter beherrschen eine spezielle Regel: Weiterleitung von IP-Paketen mit *Network Address Translation (NAT)*, auch *IP Masquerading* genannt. Dieser Mechanismus dient dazu, mehrere Rechner über eine einzige IP-Adresse mit der Außenwelt zu verbinden – eine Situation, die typischerweise auftritt, sobald man ein Heim- oder Firmen-Netzwerk mit dem Internet verbindet.

Mit `iptables` wird die Regel wie folgt aktiviert:

```
# iptables -t nat -A POSTROUTING -o WAN-Interface -j MASQUERADE
```

(Im Falle eines Internet-Zugangs per PPPoE kann *WAN-Interface* z. B. für `ppp0` stehen.)

Wenn ein Rechner aus dem internen Netz – mit einer nur lokal gültigen IPv4-Adresse – eine Anfrage an einen Rechner im Internet stellt, ersetzt der Paketfilter die IP durch seine eigene – öffentlich gültige – IP-Adresse. Wenn nun der Server auf die Anfrage antwortet, hat der Paketfilter die Aufgabe, die Antwort wieder an den richtigen lokalen Rechner zuzustellen. Dies ist dann ein Problem, wenn mehrere lokale Rechner gleichzeitig mit demselben externen Server auf demselben Port kommunizieren.

Das Problem wird gelöst durch die Zuteilung eines separaten *ausgehenden* Ports für jede ausgehende Verbindung eines lokalen Rechners. Anhand des Ports kann der Paketfilter zuordnen, für welchen lokalen Rechner die Antwort des Servers bestimmt war.

NAT verletzt das Schichtenmodell: Es verwendet Informationen aus der Transportschicht (Port-Nummern) zur Lösung eines Problems innerhalb der Vermittlungsschicht (Zustellung an lokale IP-Adresse). Dies verursacht Probleme, sobald NAT zusammen mit anderen Protokollen als TCP und UDP eingesetzt wird, z. B. ICMP, das insbesondere von `ping` verwendet wird.

Für `ping` läßt sich das Problem noch dadurch lösen, daß man anstelle einer Portnummer einen Teil des Datenfeldes verwendet. Nichtsdestoweniger bedeutet NAT eine Verwässerung des Schichtenprinzips, auf dem die Robustheit des Internet beruht, und sollte daher stets nur als Notlösung verstanden werden.

4.5 Zuweisung von IP-Adressen

- *Reverse ARP (RARP):*
  Zuweisung der IP-Adresse anhand der MAC-Adresse durch einen Server
  Auf der gleichen Ebene wie ARP, also als Ethernet-Nutzdaten
- *Bootstrap Protocol (BOOTP):*
  Anfrage per UDP-Broadcast auf Port 67 (mit MAC-Adresse und Cookie)
  Antwort per UDP-Broadcast auf Port 68 (mit IP-Adresse und demselben Cookie)
  Zusätzlich zur IP-Adresse: Netzwerkmaske, Standard-Gateway, Boot-Server
- *Dynamic Host Configuration Protocol (DHCP):* Erweiterung von BOOTP
  Weitere Zusatzdaten: Rechnername, DNS-Server (siehe 4.6), ...
  Ablaufzeit für „gemietete“ IP-Adresse (*DHCP-Lease*)

4.6 Namensauflösung

4.6.1 Die Datei /etc/hosts

IP-Adressen sind für Menschen unintuitiv zu merken, daher gibt es zusätzlich Rechnernamen, um Rechner im Netz identifizieren zu können.

Beispiel: Anstelle der IP-Adresse 193.175.85.36 kann man sich für den Web-Server der Hochschule Bochum auch den Namen www.hs-bochum.de merken.

In kleineren Rechnernetzen genügt für die Zuordnung zwischen Namen und IP-Adressen eine Datei auf jedem Rechner. Unter Unix, für das TCP/IP ursprünglich entwickelt wurde, heißt diese Datei /etc/hosts; unter Microsoft Windows heißt sie %SystemRoot%\system32\drivers\etc\hosts.

Die Datei enthält eine Tabelle mit IP-Adressen und Namen. Es sind auch mehrere Namen pro IP-Adresse zulässig.

Die Tabelle wird in beiden Richtungen verwendet: Man kann daraus sowohl die zu einem Namen gehörige IP-Adresse entnehmen als auch den oder die zu einer IP-Adresse gehörigen Namen.

Beispiel:

```
127.0.0.1   localhost
10.1.1.1    munimentum muni
10.1.1.191  voyager1
10.1.1.192  sojourner
10.1.1.193  voyager2
10.1.1.195  near
10.1.1.196  beagle2
```

Durch einen Eintrag wie

```
10.1.1.193  voyager2 www.sparkasse-essen.de
```

kann man den für den Rechner www.sparkasse-essen.de bestimmten Netzwerkverkehr auf eine selbstgewählte IP-Adresse (hier: `10.1.1.193`) umleiten. Dies kann z. B. zu Testzwecken geschehen; es stellt aber auch eine Möglichkeit für Angreifer (z. B. trojanische Pferde) dar, Man-in-The-Middle-Angriffe gegen Webseiten Dritter zu führen.

4.6.2 Domain Name System (DNS)

Die Namen im Internet sind hierarchisch in *Domänen (Domains)* angeordnet:

- *Top-Level-Domains*: begrenzte Liste (z. B. .de, .com, .edu, .net, .org, ...)
- *First-Level-Domains*: müssen registriert werden (z. B. hs-bochum.de, gerwinski.de, ...)
- Namen innerhalb der Domains und tiefer liegende Domains (*Subdomains*):
  können selbst definiert werden (z. B. peter.gerwinski.de, www.peter.gerwinski.de, ...)

Ein spezielles Programm (z. B. BIND) stellt per Netzwerk (UDP-Port 53) Zugriff auf eine Datenbank zur Verfügung.

Dieses muß für beide Richtungen separat konfiguriert werden.

Beispiele:

- Datei db.intern: Konfigurationsdatei für ein privates Netz:
  Zuordnung Name $\longrightarrow$ IP-Adresse
- Datei db.1.10.in-addr.arpa: Konfigurationsdatei für ein privates Netz:
  Zuordnung IP-Adresse $\longrightarrow$ Name
- Datei gerwinski.de: Konfigurationsdatei für eine öffentliche Domain:
  Zuordnung Name $\longrightarrow$ IP-Adresse

Innerhalb der Zonen-Dateien müssen Domains mit einem Punkt abgeschlossen werden, andernfalls wird die Standard-Domain, für die der DNS-Server zuständig ist, angehängt.

Die verschiedenen Datensätze stehen für:

- **IN**: Internet – optionale *Klasse* der Informationen
  (Heute ausschließlich `IN`, früher auch andere)
- **NS**: Nameserver – für diese Zone (*Subdomain*) zuständiger DNS-Server
- **A**: Address – zu einem Namen gehörende IPv4-Adresse
- **AAAA**: Address – zu einem Namen gehörende IPv6-Adresse
- **MX**: Mail Exchange – zu einem Namen gehörender E-Mail-Server
- **CNAME**: Canonical Name – Verweis auf einen anderen (den „kanonischen“) Namen
- **PTR**: Pointer – Verweis von einer IP-Adresse auf einen Namen

Um IP-Adressen als „Namen“ zu behandeln und auflösen zu können, werden diese rückwärts notiert und in der Domain in-addr.arpa verortet. Beispiel: aus `192.168.0.1` wird 1.0.168.192.in-addr.arpa.

Mit DNS lassen sich auch andere Informationen effizient bereitstellen.

Beispiel: Eine *DNS-based Blackhole List (DNSBL)* oder *Real-time Blackhole List* ist eine „Schwarze Liste“ von Rechnern, die für den Versand unerwünschter E-Mail („Spam“) bekannt sind. Durch eine „Abfrage der IP-Adresse“ eines Rechners innerhalb einer für diesen Zweck bereitgestellten Domain läßt sich innerhalb von Millisekunden über das Internet abfragen, ob ein Rechner auf einer derartigen Schwarzen Liste steht.

4.7 Überlastkontrolle

- *Stop-and-Wait-Protokolle*:

  Serielle Schnittstelle: RTS/CTS-Leitungen (oder RTR/CTS oder DTR/DSR)
  oder XON/XOFF-Steuerzeichen (XOFF = Ctrl+S, XON = Ctrl+Q)

  Parallele Schnittstelle: $\overline{\text{STROBE}}$, $\overline{\text{ACK}}$, BUSY
- TCP:

  window-Feld: „mehrere RTR auf einmal“

  Bei erkanntem Stau: Verringerung der Senderate
  $\longrightarrow$ sinnvoll bei kabelgebundenen Netzen, problematisch bei WLAN

4.8 Dienstgüte – Quality of Service (QoS)

- Verschiedene Anforderungen: *Latenz* vs. *Jitter* vs. *Verluste* vs. *Durchsatz*
- Ressourcen reservieren: *IntServ* mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung: *DiffServ* mit Type-of-Service-Bits (IPv4)
  bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste): *Asynchronous Transfer Mode (ATM)*

4.9 Dynamisches Routing

4.9.1 Routing Information Protocol (RIP)

- Jeder Router teilt periodisch (alle 30 s) per UDP
  seine eigene Routing-Tabelle seinen Nachbarn mit. (*Advertisement*)
- Auswahl der besten Route: anhand des *Hop Count*.
  (16 Hops steht für „nicht erreichbar“.)
- Veraltete Informationen werden noch weitergegeben.
  $\longrightarrow$ Neue Informationen verbreiten sich nur langsam.
- Speziell: Ein ausgefallener Router gilt noch über Schleifen als erreichbar,
  bis sich sein Hop Count bis 16 „aufgeschaukelt“ hat.
- veraltet, aber einfach,
  daher weiterhin verbreitet
- Programme: routed, gated

4.9.2 Open Shortest Path First (OSPF)

- Jeder Router teilt per *IP-Multicast*
  (spezielle IP-Adressen für eine Gruppe von Empfängern)
  seine eigenen direkten Verbindungen dem gesamten Netz mit. (*Link State Advertisement*)
- Jedem Weg werden „Kosten“ zugeordnet,
  die vom *Type of Service* (siehe 4.8) abhängen können.
  Auswahl der besten Route: Dijkstra-Algorithmus
- Optional: Unterteilung des Netzes (*Autonomes System*) in Bereiche (*Areas*),
  darunter eine zentrale *Backbone Area*
- Optional: Wahl eines *Designated Router*s
- leistungsfähig, aber kompliziert,
  daher nur in großen bis sehr großen Netzen im Einsatz
- Programme: z. B. quagga, OpenOSPFD

4.9.3 Border Gateway Protocol (BGP)

Während RIP und OSPF für das Routing innerhalb eines geschlossenen Netzes (z. B. einer Firma) zuständig sind, dient das *Border Gateway Protocol (BGP)* dazu, das Routing *zwischen* diesen Netzen, also das *Routing im Internet* zu steuern.

- Unterteilung des Internet in *Autonome Systeme*, denen Nummern zugeordnet sind

  Beispiele:

BelWü	553
DTAG	3320
freenet.de	5430
KPN Eurorings	286
Universität Frankfurt	20633
Vodafone	3209
Wikimedia Foundation	14907

- Jeder Router speichert eine „Entfernungstabelle“ des Internet:
  - Welches Netz gehört zu welchem autonomen System?
  - Über welche anderen autonomen Systeme kann ich dieses Netz erreichen?
  - An welchen Router muß ich mich wenden?
- Informationsaustausch per TCP (Port 179):
  Open, Keep-Alive, Update, Notification

- Auswahl der besten Route:
  - präziseste Netzmaske
  - über möglichst wenige autonome Systeme
  - lokale Präferenzen: *Policies* und *Filter*:
    Lastverteilung, ökonomische Gründe, Größenbeschränkung der Routing-Tabelle, Verteidigung gegen Denial-of-Service-Angriffe
- Den ausgewählten Router erreichen: „normales“ Routing
  („BGP ersetzt das Standard-Gateway.“)
- leistungsfähig, aber gefährlich:
  Jede Änderung betrifft das gesamte Internet.
- Programme: z. B. quagga, OpenBGPD

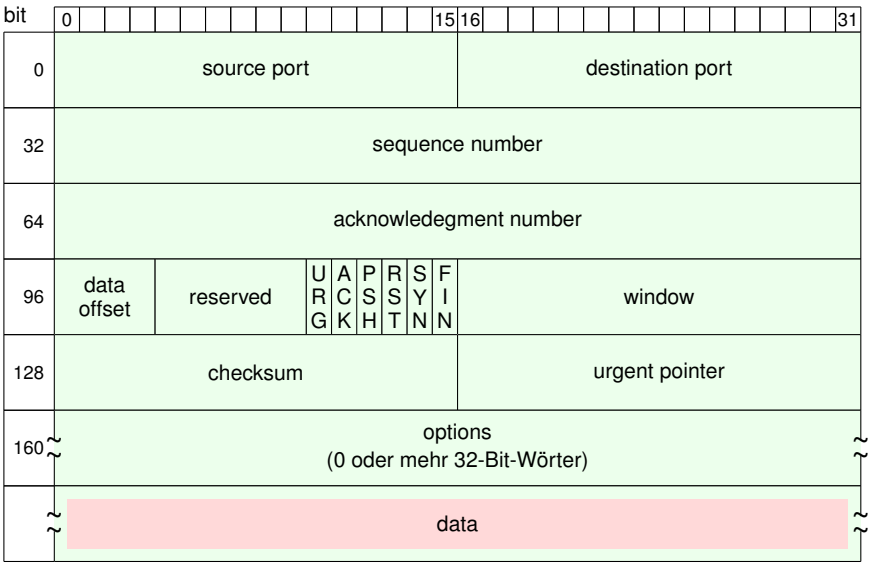
Durch IPv6 werden die Bezeichnungen für Subnetze größer; jede Zeile in der o. a. „Entfernungstabelle“ belegt also mehr Speicherplatz. Durch die bei weitem größere Anzahl verfügbarer IP-Adressen wird es aber durch IPv6 möglich, die Subnetze den autonomen Systemen systematischer zuzuordnen, als dies z. Zt. mit IPv4 der Fall ist. Die o. a. „Entfernungstabellen“ werden durch IPv6 also deutlich kürzer.

5 TCP/IP-Schicht 3: Transport-Protokolle

5.1 Transmission Control Protocol (TCP)

In der *TCP/IP-Protokollfamilie* sorgt das *Internet Protocol (IP)* für die Zustellung einzelner Datenpakete (Datagramme) von Rechner zu Rechner, identifiziert durch ihre IP-Adressen. Darauf aufsetzend stellt das *Transmission Control Protocol (TCP)* eine Verbindung zwischen zwei Endpunkten her. Seine Aufgaben lauten also:

- Umwandlung einer Sequenz von Datenpaketen in einen Datenstrom
- Herstellung von Zuverlässigkeit
- Aufbau und Abbau von Verbindungen



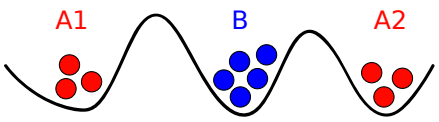
- **source port, destination port:** TCP erlaubt mehrere Ziele (Prozesse, Programme) pro Rechner und unterscheidet diese durch sog. *Port*-Nummern.

- **sequence number:** TCP zerlegt einen Datenstrom in Datenpakete (*Datagramme*), versieht sie mit fortlaufenden *Sequenz-Nummern* und versendet sie per IP. Anhand der Sequenz-Nummern kann der Empfänger die Datagramme wieder zu einem Datenstrom zusammensetzen, auch wenn unterwegs die Reihenfolge verändert wurde. TCP realisiert also eine verbindungsorientierte Datenübertragung über eine paketorientierte (verbindungslose).
- **acknowledgement number:** Bei jeder Antwort wird die um 1 erhöhte Sequenz-Nummer des zuletzt erhaltenen Datagramms zurückgesendet. Dadurch erkennt TCP den Verlust von Datagrammen und kann sie erneut senden.
- **data offset:** Da der Header Optionen enthalten kann, ist seine Länge nicht konstant. Das *Daten-Offset* zeigt an, wo der Header aufhört und die Nutzdaten beginnen.
- **SYN, ACK, RST:** Diese Bits dienen dem Verbindungsaufbau per *Drei-Wege-Handschlag (Three-Way Handshaking)* – siehe unten.
- Das **FIN**-Bit dient dem Verbindungsabbau.
- Das **URG**-Bit dient zur Markierung besonders wichtiger Steuernachrichten. Da es sich dabei in der Regel um einen Verbindungsabbau handelt, wird das URG-Bit selten verwendet.
- Das **PSH**-Bit signalisiert dem Gegenüber, daß die bisher empfangenen Daten nicht weiter zwischengepuffert, sondern der Anwendung übergeben werden sollen („Zeilenende“).
- **window:** Anzahl der Bytes, die der Absender des Pakets bereit ist, zu empfangen
- **checksum:** Prüfsumme für das TCP-Paket sowie Teile des umgebenden IP-Headers (IP-Adressen, Protokollkennung, Länge – zusammen: *TCP-Pseudo-Header*)
- **urgent pointer:** Zeiger auf das Ende der URG-Daten (s. o.), relativ zur Sequenz-Nummer
- **options:** optionale Zusatzinformationen (z. B. maximale Größe für Nutzdaten)

5.1.1 Verbindungsaufbau

Das Aufrechterhalten einer Verbindung bindet Ressourcen. Um zu prüfen, ob sich der Aufwand lohnt, wird jede Verbindung beim Aufbau in beiden Richtungen geprüft. Diese Prüfung heißt *Handschlag (Handshaking)*.

Ein zu 100 % zuverlässiger Handschlag über eine unzuverlässige Verbindung ist prinzipiell nicht möglich. Dies läßt sich durch das *Zwei-Armeen-Problem* veranschaulichen: Man denke sich eine rote und eine blaue Armee. Die rote Armee ist in zwei Teile geteilt, die einzeln der blauen Armee unterlegen, gemeinsam ihr aber überlegen sind.

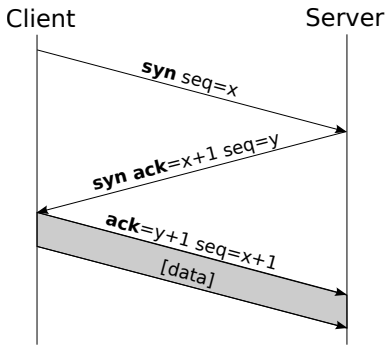


Die Befehlshaber der beiden roten Teilarmeen versuchen nun, über eine unzuverlässige Botenverbindung einen gemeinsamen Angriffszeitpunkt zu verabreden. Jeder Befehlshaber wird nur dann angreifen, wenn er absolut sicher ist, daß der andere Armeeteil dies ebenfalls tut. Die ausgetauschten Botschaften sehen dann wie folgt aus:

> A1: „Wir greifen morgen früh um 3:00 Uhr zeitgleich an. Bitte bestätigen, sonst kein Angriff.“
> A2: „Bestätige Angriffstermin. Bitte Erhalt der Bestätigung bestätigen, sonst kein Angriff.“
> A1: „Bestätige Bestätigung. Bitte bestätigen, sonst kein Angriff.“
> A2: „Bestätige Bestätigung der Bestätigung. Bitte bestätigen, sonst kein Angriff.“
> usw.

Um *absolut sicher* zu sein, daß der andere Armeeteil ebenfalls angreifen wird, muß *jede* Botschaft noch einmal bestätigt werden. Wir benötigen also unendlich viele Bestätigungen; das Problem ist unlösbar.

Bei TCP ist es also nicht möglich, aber andererseits auch nicht erforderlich, beim Aufbau einer Verbindung *absolute Sicherheit* für das Zustandekommen der Verbindung zu fordern. Stattdessen wartet jeder der beiden Kommunikationspartner *jeweils eine* Bestätigung ab:



Dieses Verfahren heißt *Drei-Wege-Handschlag (Three-Way-Handshaking)*. Die Verbindung wird in beiden Richtungen *jeweils einmal* geprüft und danach freigegeben.

Wenn ein Kommunikationspartner auf eine SYN-Anfrage nicht mit SYN-ACK, sondern mit RST antwortet, signalisiert er dadurch, daß er nicht zum Verbindungsaufbau bereit ist. Man spricht in diesem Fall von einem *geschlossenen Port*, andernfalls (also bei Antwort mit SYN-ACK) von einem *offenen Port*.

Analog zum Verbindungsaufbau erfolgt der *Verbindungsabbau* mit FIN, FIN-ACK und ACK.

5.1.2 SYN-Flooding

Der Drei-Wege-Handschlag dient dazu, die Ressourcen eines Servers ökonomisch einsetzen zu können: Erst nach erfolgreichem Verbindungsaufbau beginnt die Server-Software mit der eigentlichen Arbeit.

Umgekehrt können Angreifer den Drei-Wege-Handschlag ausnutzen, um die Ressourcen eines Servers von außen mit geringem eigenen Aufwand zu blockieren (*Denial-of-Service-Angriff*). Zu diesem Zweck generiert der Angreifer eine Flut von TCP-Paketen mit gesetztem SYN-Bit (*SYN-Flooding*), antwortet aber nicht auf die zurückkommenden SYN-ACK-Pakete. In diesem Fall muß der Server für jedes gesendete SYN-ACK-Paket die Sequenz-Nummer des erwarteten ACK-Pakets speichern, um die Verbindung im Falle eines legitimen Verbindungsaufbaus zuordnen zu können. Der Angreifer versendet also lediglich (kleine) SYN-Pakete, von denen jedes auf dem Server Speicherplatz belegt.

Eine Verteidigung gegen SYN-Flooding stellen sog. *SYN-Cookies* dar: Anstatt sich die Sequenznummern der angefragten Verbindungen in einer Tabelle zu merken, berechnet man diese durch Bilden eines *Hash-Wertes* über die Verbindungsdaten (IP-Adressen, Ports, Zeitstempel) sowie einen geheimen Schlüssel. Auf diese Weise kann der Server einem anschließend eintreffenden ACK-Paket ohne Tabelle direkt „ansehen“, ob es sich um einen legitimen Verbindungsaufbau handelt, der zu einem vorher empfangenen SYN-Paket paßt.

5.1.3 Software-Werkzeuge

Auf Unix-artigen Systemen ist es möglich, beliebige Programme mit ihrer Standardeingabe und -ausgabe an TCP-Ports anzuschließen, sie also als Internet-Server zu verwenden. Zu diesem Zweck konfiguriert man das Dienstprogramm (*Daemon*) `inetd` wie folgt (Datei: /etc/inetd.conf):

```
1234 stream tcp nowait peter /usr/bin/bc
```

Die erste Zahl ist die Port-Nummer, unter der der Dienst angeboten werden soll, der letzte Eintrag das Programm, das aufgerufen werden soll (hier: ein „Taschenrechner“), der vorletzte Eintrag der Benutzer, mit dessen Rechten das Programm laufen soll. (Weitere Informationen: `man inetd`)

Beispiele für Internet-Dienste:

- Webseiten: Port 80
- E-Mail: Port 25
- Krieg der Sterne: Rechner towel.blinkenlights.nl, Port 23

Mit Hilfe des Programms `nmap` kann man prüfen, auf welchen Port-Nummern ein Rechner Dienste anbietet:

```
# nmap -p 0-65535 Rechner

Starting Nmap 5.00 ( http://nmap.org ) at 2012-06-11 16:38 CEST
Interesting ports on Rechner (192.168.42.123):
Not shown: 65535 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:1A:80:D0:75:A8 (Sony)

Nmap done: 1 IP address (1 host up) scanned in 1.19 seconds
```

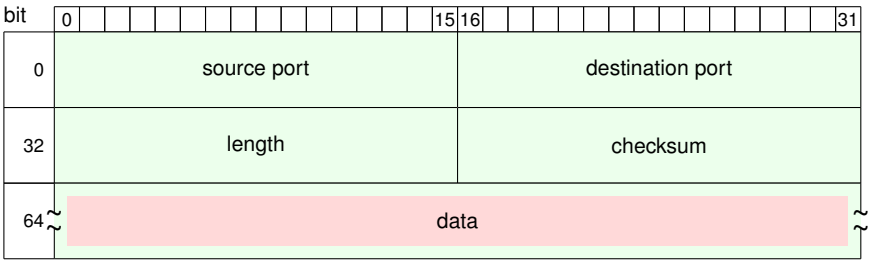
(Weitere Informationen: `man nmap`)

Derartige *Port-Scans* sind ein wichtiges Werkzeug, um zu überprüfen, ob ein Rechner auch tatsächlich genau diejenigen Dienste anbietet, die er anbieten soll. Zusätzliche Dienste unbekannter Herkunft können u. U. von einem Eindringling installiert worden sein, der über diesen Dienst den Rechner kontrollieren und für eigene Zwecke nutzen will.

Warnung: Manche Unternehmen und Institutionen werten einen Port-Scan gegen ihre Rechner als Angriff und reagieren darauf, teilweise automatisiert, mit einer Strafanzeige.

5.2 User Datagram Protocol (UDP)

Zusätzlich zum zuverlässigen, verbindungsorientierten Protokoll TCP enthält die TCP/IP-Protokollfamilie auch ein „leichtgewichtiges“ Protokoll: das *User Datagram Protocol (UDP)*.



Wie TCP verwaltet auch UDP mehrere Ziele pro Rechner und verwaltet sie durch Port-Nummern. TCP- und UDP-Portnummern sind voneinander unabhängig; man kann also insbesondere unter derselben Port-Nummer einen TCP- und einen UDP-Dienst unabhängig voneinander laufen lassen.

UDP ist unzuverlässig und verbindungslos: Im Gegensatz zu TCP kennt es keinen Verbindungsaufbau (Handschlag), und es gibt keine Mechanismen, um verlorengegangene Datenpakete erneut zu senden oder in der falschen Reihenfolge eingetroffene Pakete zu sortieren. Wenn derartige Mechanismen gewünscht werden, muß die Anwendung sie selbst implementieren (daher: *User* Datagram Protocol) – oder man verwendet stattdessen TCP.

Der Prüfwert wird bei UDP genau wie bei TCP gebildet, ist jedoch optional.

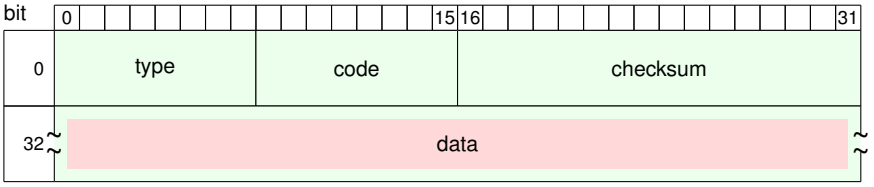
Das Programm `inetd` zum Bereitstellen von Internet-Diensten und das Programm `nmap` zum Prüfen auf offene Ports (Port-Scan) unterstützen auch UDP. Für nähere Informationen siehe `man inetd` bzw. `man nmap`.

5.3 Internet Control Message Protocol (ICMP)

Ein weiteres Protokoll der TCP/IP-Familie ist das *Internet Control Message Protocol (ICMP)*. Es dient nicht dem Datenaustausch, sondern dazu, das Netzwerk selbst zu steuern:

- Testen der Erreichbarkeit (`ping`)
- Routing-Informationen

- Antwort auf vergebliche UDP-Anfragen (Ersatz für das bei UDP nicht vorhandene RST-Bit)



Da bei ICMP der Rechner (und nicht ein Programm auf dem Rechner) das eigentliche Ziel darstellt, verwendet ICMP keine Port-Nummern.

Obwohl ICMP eigentlich nicht dafür vorgesehen ist, kann es durch Verwendung des Daten-Feldes auch zur Datenübertragung zwischen Rechnern genutzt werden (*ICMP-Tunnel*).

6 TCP/IP-Schicht 4: Anwendung

6.1 „OSI-Schicht 6“: Darstellung

Tatsächlich: oberhalb der Anwendungsschicht (TCP/IP-Schicht 4 = OSI-Schicht 7)

Beispiel:
Ethernet-Frame ⟶ IP-Paket ⟶ TCP-Segment ⟶ SMTP-Body ⟶ Base64 ⟶ UTF-8 ⟶ Inhalt

6.1.1 Zeichensätze

- **ASCII**

  nur 7 Bits, keine Umlaute

  Variante: Umlaute anstelle von eckigen/geschweiften Klammern

- **8-Bit-Zeichensätze**

  Beispiel: Latin-9 (ISO-8859-15)
  enthält Umlaute und das Euro-Symbol

```
G  l  ü  c  k  w  ü  n  s  c  h  e  \n
47 6c fc 63 6b 77 fc 6e 73 63 68 65 0a
```

- **Unicode**

  32-Bit-Zeichensatz
  enthält alle bekannten Zeichen einschließlich asiatische Zeichensätze

6.1.2 Kodierungen

- **UTF-8**

  8-Bit-Kodierung von Unicode

```
G  l  Ã  ¼  c  k  w  Ã  ¼  n  s  c  h  e  \n
47 6c c3 bc 63 6b 77 c3 bc 6e 73 63 68 65 0a
      ü              ü
```

- **UTF-16**

  16-Bit-Kodierung von Unicode

```
UTF-8 G    l    ü    c    k    w    ü    n    s    c    h    e    \n
 feff 0047 006c 00fc 0063 006b 0077 00fc 006e 0073 0063 0068 0065 000a
```

- **Base64**

  Kodierung von Binärdaten durch druckbare Zeichen:
  26 Großbuchstaben + 26 Kleinbuchstaben + 10 Ziffern = 62
  $\longrightarrow$ 2 weitere Zeichen (+ und /)

  6 Bit pro Zeichen

  ```
  Glückwünsche
  R2z8Y2t3/G5zY2hlCg
  ```

  19 Zeichen statt 12

- **Quoted-Printable**

  Kodierung von Binärdaten durch druckbare Zeichen:
  Was über 7 Bits hinausgeht, wird durch ein Gleichheitszeichen und eine zweistellige Hexadezimalzahl dargestellt

  ```
  Gl=FCckw=FCnsche
  ```

- **Punycode**

  ```
  xn--Glckwnsche-bebd
  ```

  Verwendung in Internet-Domain-Namen

6.2 Dateiübertragungsprotokolle

6.2.1 Hypertext Transfer Protocol (HTTP)

- Zustandslos: Übertragung einzelner Dateien, kein Sitzungskonzept
- HTTP 1.0: 1 TCP-Verbindung pro Datei

```
$ nc 213.239.212.49 80
GET /
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
 <head>
  <title>ngc224.gerwinski.de</title>
 </head>
 <body background="ngc224.jpg" bgcolor="#000000" text="#CFCFCF"
       link="#7F7FFF" alink="#CFCFFF" vlink="#FF5FFF">
  <h1>ngc224.gerwinski.de - der Server</h1>
  <p>Benannt nach der Andromeda-Galaxie - M31 - NGC224
  <p align="center">
  <img src="null.png" alt="" width="1" height="1" vspace="360">
  <p><font size="-2">
  <a href="http://www.peter.gerwinski.de">Peter Gerwinski</a><br>
  <a href="http://www.noao.edu/image_gallery/html/im0424.html">Bild:
  Bill Schoening, Vanessa Harvey/REU program/NOAO/AURA/NSF</a></font>
 </body>
</html>
```

- HTTP 1.1: Mehrere Dateien pro TCP-Verbindung möglich,
  mehrere „virtuelle“ Web-Server auf derselben IP-Adresse möglich

```
$ nc 213.239.212.49 80
GET /net-2013ss.html HTTP/1.1
Host: www.peter.gerwinski.de

HTTP/1.1 200 OK
Date: Sat, 30 Mar 2013 11:30:23 GMT
Server: Apache/2.2.9 (Debian) mod_ssl/2.2.9 OpenSSL/0.9.8g
Last-Modified: Mon, 02 Apr 2012 19:05:59 GMT
```

```
ETag: "7e400d-d5-4bcb6e16e13c0"
Accept-Ranges: bytes
Content-Length: 213
Content-Type: text/html; charset=ISO-8859-15

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
 <body>
  <h1>
   Rechnertechnik und -netzwerke
  </h1>
  <p align="right">
   Sommersemester 2013<br>
   Prof. Dr. Peter Gerwinski
  </p>
 </body>
</html>
```

- Daten zum Server schicken: POST
- WebDAV: Netzwerklaufwerke über HTTP
- Schema://Benutzer:Passwort@Rechner:Port/Verzeichnis/Datei?Abfrage#Fragment
  Zeichenkodierung mit „%“

  Beispiel: http://www.microsoft.com:80@172.16.221.180:1234
  bezeichnet eine Ressource auf dem Rechner 172.16.221.180, Port 1234
  bei Anmeldung als Benutzer www.microsoft.com mit Passwort 80
  (und *eben nicht* eine Ressource auf dem Rechner www.microsoft.com, Port 80).
- Nachträgliches Sitzungskonzept („OSI-Schicht 5“) durch Inhalt der Daten
  (Cookies, spezielle URLs, spezielle Abfragen),
  also insbesondere oberhalb der Anwendungsschicht (TCP/IP-Schicht 4 = OSI-Schicht 7)
- Server-Programme: Apache HTTP Server, Apache Tomcat, Roxen,
  Microsoft Internet Information Server
- Client-Programme: lynx, w3m, Mozilla Firefox, Konqueror, Opera, Microsoft Internet Explorer

6.2.2 File Transfer Protocol (FTP)

Beim *File Transfer Protocol (FTP)* handelt es sich um ein älteres Protokoll. Als es entwickelt wurde, gab es noch kein NAT. Man konnte also davon ausgehen, daß jeder Rechner, der mit dem Internet verbunden ist, über eine öffentliche IP-Adresse erreicht werden kann.

Eine FTP-Dateiübertragung arbeitet nicht mit nur einer, sondern mit zwei TCP-Verbindungen: Die direkt aufgebaute TCP-Verbindung (Server-Port: 21) dient nur zur Steuerung; für die eigentliche Dateiübertragung wird eine zweite TCP-Verbindung aufgebaut. Dabei kann man zwischen zwei grundsätzlich verschiedenen Modi wählen:

- **Aktives FTP:** Der Client öffnet einen zufälligen Port und teilt dem Server seine IP-Adresse und die Port-Nummer mit. Der Server baut daraufhin die zweite TCP-Verbindung zum Client auf.
- **Passives FTP:** Der Server öffnet einen zufälligen zweiten Port und teilt dem Client seine IP-Adresse und die zweite Port-Nummer mit. Der Client baut die zweite TCP-Verbindung zum Server auf.

Dies stellt eine Verletzung des Schichtenprinzips dar: Ein Protokoll der Anwendungsebene (OSI-Schicht 7) arbeitet mit Informationen, die eigentlich in den darunterliegenden Ebenen verborgen (*gekapselt*) sein sollten.

Dies führt zunächst dazu, daß FTP grundsätzlich nur zusammen mit TCP/IP verwendet werden kann. Es hat aber auch zur Folge, daß aktives FTP nicht mit NAT zusammenarbeitet. (Auch NAT verletzt das Schichtenprinzip, indem es IP-Informationen durch Port-Informationen kodiert.) Da NAT heute weit verbreitet ist, sollte aktives FTP nicht mehr verwendet werden.

Eine weitere Folge ist, daß FTP nicht über eine SSH-Port-Weiterleitung (siehe Abschnitt 8.8) getunnelt werden kann.

6.2.3 Simple Asynchronous File Transfer (SAFT) und Frams' Fast File Exchange (F*EX)

Problem: Asynchrone Übertragung größerer Datenmengen (zu groß für E-Mail) – http://xkcd.com/949/

Lösung: Spezielles Protokoll – http://fex.rus.uni-stuttgart.de

- E-Mail: ASCII-Kodierung von Binärdaten $\longrightarrow$ Datenmenge wird größer
  SAFT, F*EX: automatische Kompression
- automatische Wiederaufnahme bei Abbruch der Verbindung
- automatisches Löschen nach dem Herunterladen,
  automatische Benachrichtigung des Empfängers,
  effizient auch bei mehreren Empfängern,
  ...
- skriptfähig

6.2.4 Unix-to-Unix Copy (UUCP)

Problem: Asynchrone Datenübertragung bei nicht-permanenter, nicht-direkter Verbindung

Lösung: Pfad angeben; sobald Verbindung besteht, zum nächsten Rechner senden

Auch möglich: Programme ausführen (Vorgänger von SSH)

6.3 E-Mail-Protokolle

6.3.1 Simple Mail Transport Protocol (SMTP)

- Situation: Ein Rechner ist permanent über das Internet erreichbar.
  $\longrightarrow$ E-Mail-Server
- Der Rechner lauscht auf TCP-Port 25.
- `HELO`: Sitzung eröffnen, eigene Identität angeben
- `MAIL FROM`: Absender angeben
- `RCPT TO`: Empfänger angeben
- `DATA`: zuerst die Kopfzeilen, dann eine Leerzeile, dann die eigentliche E-Mail
- E-Mail beenden: ein Punkt in einer Zeile
- `QUIT`: Sitzung beenden

Binärdaten werden gemäß MIME-Standard als ASCII kodiert.

6.3.2 Post Office Protocol (POP)

- Abholung von E-Mails durch den Client
- Keine Verschlüsselung (auch nicht für Passwörter)
- Server-Programme: qpopper, Cyrus, Courier, Microsoft Exchange
- Client-Programme: Alle gängigen E-Mail-Programme

6.3.3 Internet Message Access Protocol (IMAP)

- Erweiterung der Funktionalität von POP
- Mailbox wahlweise auf dem Server:
  Befehle zum Verwalten, hierarchische Ordnung
  $\longrightarrow$ zentrales Backup möglich
- Server-Programme: UW IMAP, Cyrus, Courier, Microsoft Exchange
- Client-Programme: Alle gängigen E-Mail-Programme
  bis auf Microsoft Outlook bis einschließlich Version 2007 (IMAP fehlerhaft)

6.4 X11

Grafik über das Netz
Standard für Unix

X-Server: Treiber für Ein- und Ausgabegeräte

X-Client: Anwenderprogramm

Kommunikation mittels Interprozeßkommunikation *oder per TCP/IP*

- `DISPLAY`-Variable: *Rechner*`:`*Display−Nr.*, z. B. `localhost:0`, abgekürzt `:0`
- Verbindungsaufbau erlauben: `xhost +`*Rechner*
- X-Server starten: `X :`*Display−Nr.*,
  z. B. `X :1`
- X-Server starten, TCP-Verbindungen sperren: `X -nolisten tcp :1`
  (heutzutage üblich)

VNC ist ein „virtueller“ X-Server im Speicher.
Anzeigen des Inhalts eines VNC-X-Servers: über das Netz per VNC-Client

Ähnliches Konzept: NoMachine (NX), mit starker Kompression

Variante: `x11vnc`, `WinVNC`
Einen „realen“ X-Server bzw. den MS-Windows-Bildschirminhalt als VNC-Server exportieren

Thin Clients: Ein *Display Manager* lauscht auf TCP-Verbindungen (Protokoll: XDMCP)
Aufruf des Clients: `X -query` *Rechner*

6.5 Proxies

Proxy = Stellvertreter

Ein Programm kennt das Protokoll
und übernimmt auf der einen Seite die Rolle des Clients,
auf der anderen die des Servers

$\longrightarrow$ Datenströme kontrolliert durchlassen und filtern

6.6 Tunnel

HTTP-, ICMP-, DNS-Tunnel

SSH-Tunnel: siehe 8.8

$x^4 + x + 1$ Der Herr der Ringe

oder: Rechnen mit merkwürdigen Objekten

(oder: Mathematische Grundlagen der Kodierung und Verschlüsselung)

$(x^4 + x + 1).1$ Motivation

Natürliche Zahlen (1, 2, 3, ... – Anzahl von Gegenständen) lassen sich ohne jegliche Einschränkung addieren. Bei der Subtraktion hingegen stößt man auf Grenzen:

- Sobald man eine Zahl von sich selbst subtrahiert, ist das Ergebnis keine natürliche Zahl mehr.
- Ebensowenig ist es bei natürlichen Zahlen möglich, von einer kleineren Zahl eine größere zu subtrahieren.

Durch die Einführung der Zahl 0 läßt sich die erste Grenze aufheben, durch die Einführung negativer Zahlen die zweite.

Analoge Überlegungen bezüglich der Multiplikation und der Division führen zur Einführung von Brüchen, wobei die Zahl 1 bezüglich der Multiplikation den Platz einnimmt, den bei der Addition die Zahl 0 innehat.

Dei mathematische Disziplin der *Algebra* abstrahiert diese Überlegungen und ermöglicht den Zugang zu einer Vielzahl von mathematischen Objekten mit sehr merkwürdigen Eigenschaften, mit denen man trotzdem „ganz normal“ rechnen kann:

- Objekte, die man „wie ganze Zahlen“ addieren und subtrahieren kann, bilden eine *Gruppe*.
- Objekte, die man „wie Brüche“ multiplizieren und dividieren kann, bilden ebenfalls eine *Gruppe*. Dies läßt erkennen, daß die Addition und die Multiplikation keineswegs grundverschieden sind, sondern einige grundsätzliche Gemeinsamkeiten aufweisen.
- Objekte, die man „wie ganze Zahlen“ addieren, subtrahieren und multiplizieren kann, bilden einen *Ring*.
- Objekte, die man „wie positive und negative Brüche“ addieren, subtrahieren, multiplizieren und dividieren kann, bilden einen *Körper*.

$(x^4 + x + 1).2$ Gruppen

Definition: Sei G eine Menge, $*$ eine Verknüpfung auf G, d. h. für $a, b \in G$ gibt es stets ein $c = a * b$, das ebenfalls in G liegt. Wenn

- für alle $a, b, c \in G$ gilt: $(a * b) * c = a * (b * c)$ (Assoziativgesetz),
- ein $e \in G$ existiert, so daß für alle $a \in G$ gilt: $a * e = e * a = a$ (neutrales Element) und
- für jedes $a \in G$ ein $a^{-1} \in G$ existiert, so daß gilt: $a * a^{-1} = a^{-1} * a = e$ (inverses Element),

dann heißt $(G, *)$ eine *Gruppe*.

Definition: Sei $(G, *)$ eine Gruppe. Wenn zusätzlich

- für alle $a, b \in G$ gilt: $a * b = b * a$ (Kommutativgesetz),

dann heißt $(G, *)$ eine *kommutative Gruppe*.

Beispiele:

- Die Menge der ganzen Zahlen mit der Verknüpfung der Addition ist eine kommutative Gruppe.
- Die Menge der positiven Brüche (ohne Null) mit der Verknüpfung der Multiplikation ist eine kommutative Gruppe.
- Die Menge $\{0\}$ mit der Verknüpfung der Addition ist eine kommutative Gruppe.
- Die Menge $\{1\}$ mit der Verknüpfung der Multiplikation ist eine kommutative Gruppe.
- Die Menge der Zahlen von 0 bis 6 mit der Verknüpfung der Addition modulo 7 ist eine kommutative Gruppe.
- Die Menge der Zahlen von 1 bis 6 mit der Verknüpfung der Multiplikation modulo 7 ist eine kommutative Gruppe. Hierbei lauten die inversen Elemente
  $1^{-1} = 1$, denn $1 \cdot 1 = 1$,
  $2^{-1} = 4$, denn $2 \cdot 4 = 8$ und daher $(2 \cdot 4) \,\%\, 7 = 1$,
  $3^{-1} = 5$, denn $3 \cdot 5 = 15$ und daher $(3 \cdot 5) \,\%\, 7 = 1$,
  $4^{-1} = 2$, denn $2^{-1} = 4$,
  $5^{-1} = 3$, denn $3^{-1} = 5$ und
  $6^{-1} = 6$, denn $6 \cdot 6 = 36$ und daher $(6 \cdot 6) \,\%\, 7 = 1$.
- Die Menge der Polynome mit ganzzahligen Koeffizienten mit der Verknüpfung der Addition ist eine kommutative Gruppe.

Nicht-kommutative Gruppen sind beispielsweise bei den Betrachtungen von Symmetrien wichtig (z. B. Drehungen und Spiegelungen eines regelmäßigen n-Ecks). Solche Gruppen werden häufig durch Matrizen beschrieben.

$(x^4 + x + 1).3$ Ringe

Definition: Sei R eine Menge; seien $+$ und $\cdot$ Verknüpfungen auf R. Wenn

- $(R, +)$ eine kommutative Gruppe ist,
- für alle $a, b, c \in R$ gilt: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ (Assoziativgesetz),
- für alle $a, b, c \in R$ gilt: $(a + b) \cdot c = a \cdot c + b \cdot c$ und $a \cdot (b + c) = a \cdot b + a \cdot c$ (Distributivgesetze),

dann heißt $(R, +, \cdot)$ ein *Ring*.

Definition: Sei $(R, +, \cdot)$ ein Ring. Wenn zusätzlich

- für alle $a, b \in R$ gilt: $a \cdot b = b \cdot a$ (Kommutativgesetz),

dann heißt $(R, +, \cdot)$ ein *kommutativer Ring*.

Definition: Sei $(R, +, \cdot)$ ein (kommutativer) Ring. Wenn zusätzlich

- ein $e \in G$ existiert, so daß für alle $a \in G$ gilt: $a \cdot e = e \cdot a = a$ (neutrales Element),

dann heißt $(R, +, \cdot)$ ein *(kommutativer) Ring mit 1*.

Beispiele:

- Die Menge der ganzen Zahlen mit den Verknüpfungen der Addition und der Multiplikation ist ein kommutativer Ring mit 1.
- Die Menge der Brüche mit den Verknüpfungen der Addition und der Multiplikation ist ein kommutativer Ring mit 1.
- Die Menge $\{0\}$ mit den Verknüpfungen der Addition und der Multiplikation ist ein kommutativer Ring (ohne 1).
- Die Menge der Zahlen von 0 bis 6 mit den Verknüpfungen der Addition und der Multiplikation modulo 7 ist ein kommutativer Ring mit 1.
- Die Menge der Polynome mit ganzzahligen Koeffizienten mit den Verknüpfungen der Addition und der Multiplikation ist ein kommutativer Ring mit 1, der sog. *Polynomring* über den ganzen Zahlen.

Insbesondere kann man mit Polynomen wie mit ganzen Zahlen „ganz normal“ rechnen.

$(x^4 + x + 1).4$ Körper

Definition: Sei K eine Menge; seien $+$ und $\cdot$ Verknüpfungen auf K. Wenn

- $(K, +, \cdot)$ ein Ring mit 1 ist und
- $(K\backslash\{0\}, \cdot)$ eine kommutative Gruppe ist,

dann heißt $(K, +, \cdot)$ ein *Körper*.

Beispiele:

- Die Mengen der gebrochenen, reellen und komplexen Zahlen, jeweils mit den Verknüpfungen der Addition und der Multiplikation, sind Körper.
- Die Menge der ganzen Zahlen mit den Verknüpfungen der Addition und der Multiplikation ist *kein* Körper.
- Die Menge $\{0, 1\}$ mit den Verknüpfungen der Addition und der Multiplikation ist ein Körper.
- Die Menge der Zahlen von 0 bis 6 mit den Verknüpfungen der Addition und der Multiplikation modulo 7 ist ein Körper.
- Allgemein gilt: Sei p eine Primzahl, dann ist die Menge der Zahlen von 0 bis $p - 1$ mit den Verknüpfungen der Addition und der Multiplikation modulo p ein Körper.
- Wenn p keine Primzahl ist, dann ist die Menge der Zahlen von 0 bis $p - 1$ mit den Verknüpfungen der Addition und der Multiplikation modulo p zumindest noch ein Ring.
- Ein Polynomring ist i. a. *kein* Körper.

Bemerkung: Ein Polynomring über einem Körper (z. B. alle Polynome, deren Koeffizienten Brüche, reelle oder Komplexe Zahlen sind) erlaubt *Division mit Rest*, die sog. *Polynomdivision*.

Bemerkung: Wenn man für die Verknüpfunt „·“ auf die Forderung nach Kommutativität verzichtet, definiert man einen *Schiefkörper*. Das wichtigste Beispiel eines Schiefkörpers ist der *Quaternionenschiefkörper*: Während die komplexen Zahlen aus den reellen Zahlen durch Hinzunehmen einer Zahl i mit der Eigenschaft $i^2 = -1$ hervorgehen, gelangt man von den reellen Zahlen zu den Quarternionen durch das Hinzunehmen dreier Zahlen i, j, k mit den Eigenschaften

$$i^2 = j^2 = k^2 = -1 \qquad \text{und} \qquad i \cdot j = k = -j \cdot i.$$

Anwendungen finden Quaternionen

- in der Zahlentheorie z. B. beim Beweis des Vier-Quadrate-Satzes
  („Jede natürliche Zahl kann als Summe von vier Quadratzahlen geschrieben werden.“),
- in der Geometrie und Computer-Grafik z. B. zur Beschreibung von Drehungen im Raum
  (Realteil = Drehwinkel; i-j-k-Teil = Drehachse),
- in der theoretischen Physik z. B. bei der Betrachtung der elektroschwachen Wechselwirkung
  und des Higgs-Feldes.

8 Verschlüsselung

8.8 Einführung

Geheime Botschaft: Apfelkuchen

Verschlüsselung: Jeden Buchstaben durch seinen Nachfolger ersetzen

Verschlüsselte Botschaft: Bqgfmlvdifo

Schlüssel: 1

Zum Entschlüsseln: -1

Verbesserung: Buchstaben wild permutieren

Knacken: Buchstabenhäufigkeit

Verbesserung:
1. Buchstaben um 2 verschieben
2. Buchstaben um 7 verschieben
3. Buchstaben um 18 verschieben
4. Buchstaben um 22 verschieben
und wieder von vorne

Passwort der Länge 4

Knacken: Autokorrelationsfunktion
Text um n Buchstaben verschieben, Differenz bilden, Buchstabenhäufigkeit berechnen
$\longrightarrow$ Länge des Passworts berechnen
$\longrightarrow$ Rest mit modernen Rechnern machbar

Verbesserung:
Passwort als Startwert für Pseudozufallszahlengenerator nehmen
(In C: srand (42); random () % 26)
Passwort = Startwert

Knacken: Zahlenbereich durchprobieren
typischerweise: $0 \ldots 2^{32} - 1$, ca 4 Milliarden Schlüssel
alle durchprobieren $\longrightarrow$ fertig in Sekundenbruchteilen

Verbesserung:
Echter Zufall (z .B. Würfel)
Passwort = alle erwürfelten Zufallszahlen

Knacken nicht möglich („One Time Pad“ – mathematisch beweisbar)

Für die Praxis nahezu nutzlos, weil Passwort genau so lang wie der Klartext

Kompromiß:
Passwort als Startwert für *einen speziellen* Pseudozufallszahlengenerator nehmen
Passwort = Startwert
ausgeklügelte Verfahren: DES, IDEA, 3DES, CAST5, AES, Blowfish, Twofish, ...
alle Schlüssel durchprobieren: nicht möglich, wenn Zahlenbereich groß genug

DES: 56 Bits
3DES: 112 Bits
IDEA: 128 Bits
AES: 128–256 Bits
...

sicher ab ca. 80 Bits

Symmetrische Verfahren: Derselbe Schlüssel zum Ver- und Entschlüsseln

Problem: sichere Übertragung des Schlüssels

Lösung: *Asymmetrische Verfahren*

- Ein Schlüssel dient zum Verschlüsseln,
- ein anderer zum Entschlüsseln.

8.8 RSA

Bei diesem asymmetischen Verschlüsselungsverfahren wählt man zufällig drei Primzahlen p, q und r.

Das Produkt $p \cdot q$ zusammen mit r bilden zusammen den öffentlichen Schlüssel, die einzelnen Primzahlen p und q den geheimen Schlüssel.

Bei hinreichend großen Zahlen p und q kann man deren Produkt veröffentlichen, ohne befürchten zu müssen, daß irgendjemand daraus die einzelnen Zahlen p und q berechnen könnte.

Beispiel: $p = 7$, $q = 11$, $r = 13$

Verschlüsselung mit Hilfe des öffentlichen Schlüssels:

- Wir rechnen modulo $p \cdot q$, also im Ring der Zahlen von 0 bis $p \cdot q - 1$.
- Für eine unverschlüsselte Nachricht x berechnet man die verschlüsselte Nachricht c wie folgt:

$$c = x^r \,\%\, (p \cdot q)$$

- Beispiel: $x = 2$

$$c = 2^{13} \,\%\, 77 = 30$$

- Beispiel: $x = 75$

$$c = 75^{13} \,\%\, 77 = 47$$

Entschlüsselung mit Hilfe des öffentlichen Schlüssels:

- Wir berechnen das multiplikative inverse Element r^{-1} von r modulo $(p-1) \cdot (q-1)$, also im Ring der Zahlen von 0 bis $(p-1) \cdot (q-1) - 1$. Hierfür ist die Kenntnis der individuellen Zahlen p und q notwendig.

  (Da die Menge der Zahlen von 0 bis $(p-1) \cdot (q-1) - 1$ nur ein Ring ist und kein Körper, ist die Existenz von r^{-1} nicht selbstverständlich. In dieser speziellen Situation jedoch kann man beweisen, daß es existiert.)

- Für eine verschlüsselte Nachricht c berechnet man die entschlüsselte Nachricht x wie folgt:

$$x = c^{(r^{-1})} \,\%\, (p \cdot q)$$

  Als Andeutung des Beweises sei darauf hingewiesen, daß für reelle Zahlen gilt:

$$c^{(r^{-1})} = (x^r)^{(r^{-1})} = x^{r \cdot (r^{-1})} = x^1 = x$$

  Bei RSA haben wir es nicht mit rellen Zahlen zu tun; stattdessen rechnen wir in zwei Ringen, nämlich modulo $p \cdot q$ und modulo $\varphi(p \cdot q) = (p-1) \cdot (q-1)$. ($\varphi$ bezeichnet die sog. *Eulersche φ-Funktion*.) In dieser speziellen Situation läßt sich der Satz von Euler-Fermat anwenden (bewiesen von Euler, 1763), so daß die o. a. Gleichung ebenfalls gilt.

- Beispiel: $p = 7$, $q = 11$, $r = 13$

$$(p-1) \cdot (q-1) = 60, \qquad r^{-1} \,\%\, 60 = 37, \text{ denn } 13 \cdot 37 \,\%\, 60 = 1$$

- Beispiel: $c = 30$

$$x = 30^{37} \,\%\, 77 = 2$$

- Beispiel: $c = 47$

$$x = 47^{37} \,\%\, 77 = 75$$

Siehe auch: durchblicker1.1.pdf, S. 49–63

Wenn man in den o. a. Formeln die Rollen von r (öffentlich) und r^{-1} (geheim) vertauscht, wird es möglich, einen Text mit Hilfe des geheimen Schlüssels zu verschlüsseln und mit Hilfe des öffentlichen Schlüssels zu entschlüsseln. In diesem Fall kann der Text von jedermann entschlüsselt werden; der verschlüsselte Text kann aber nur von derjenigen Person stammen, die im Besitz des geheimen Schlüssels ist. Das Verfahren eignet sich somit dazu, die Authentizität eines Textes sicherzustellen – den Text *digital zu signieren* (zu „unterschreiben“).

In der Praxis wendet man digitale Signaturverfahren nicht auf einen ganzen Text an, sondern nur auf einen Prüfwert („Checksumme“), die sog. *kryptographische Hash-Funktion*. Solche Funktionen werden speziell daraufhin entworfen, daß es einfach ist, für einen gegebenen Text den Prüfwert zu ermitteln, aber schwierig bis unmöglich, zu einem gegebenen Prüfwert einen anderen passenden Text als den ursprünglichen zu finden.

Beispiele für kryptographische Hash-Funktionen:

- *Secure Hash Algorithm (SHA-2)*: Für diese Familie von Hash-Funktionen mit verschiedenen Schlüssellängen (224 bis 512 Bit – SHA-224, SHA-256, SHA-384 und SHA-512) sind derzeit (2013) keine Schwachstellen bekannt. Die Algorithmen gelten als sicher; ihre Verwendung wird empfohlen.
- *Message-Digest Algorithm 5 (MD5)*: Seit 2004 ist bekannt, daß sich Kollisionen (Texte mit gleichem MD5-Hash-Wert) erzeugen lassen. MD5 kann somit nicht mehr als sicher bezeichnet werden.
- *Data Encryption Standard (DES)*: Dieser symmetrische Verschlüsselungsaltorithmus war lange Zeit der etablierte Standard zum Speichern von Hash-Werten von Passwörtern. Mit einer maximalen Schlüssellänge von 56 Bits ist DES bei weitem nicht sicher genug; dennoch findet man noch gelegentlich Software und Betriebssysteme, die Passwörter in Gestalt von (knackbaren) DES-Hash-Werten abspeichern.

Andere Prüfwerte wie z. B. der Cyclic Redundancy Check (CRC – siehe S. 31) sind *nicht* dafür entwickelt worden, Kollisionen möglichst schwer zu machen (sondern z. B. daraufhin, typische Übertragungsfehler möglichst zuverlässig zu erkennen). Sie dürfen daher *nicht* als Hash-Funktionen für digitale Signaturen verwendet werden.

Siehe auch: Übungsaufgabe Nr. 1 vom 26. April 2013

8.8 Diffie-Hellman-Schlüsselaustausch

Problem: über eine unsichere Verbindung einen gemeinsamen geheimen Schlüssel vereinbaren

Lösung: *Diffie-Hellman-Schlüsselaustausch*

Wie in der Kryptographie üblich, bezeichnen wir die beiden Parteien, die verschlüsselt kommunizieren wollen, als „Alice“ (A) und „Bob“ (B).

- Alice und Bob einigen sich auf einen gemeinsamen Modulus p und übertragen ihn über die unsichere Verbindung. Alle folgenden mathematischen Operationen erfolgen modulo p.
- Alice und Bob einigen sich auf einen gemeinsame Basiszahl g und übertragen diese über die unsichere Verbindung.
- Alice erzeugt einen *geheimen Schlüssel* a und überträgt $g^a \pmod p$ über die unsichere Verbindung. $g^a \pmod p$ ist in diesem Fall der zu a gehörende *öffentliche Schlüssel*.

  Ein Angreifer, der $A = g^a \pmod p$ abhört, müßte $\log_g A \pmod p$ berechnen, um a zu ermitteln. Wenn p, g und a groß genug gewählt sind (mehrere 100 Dezimalstellen), ist dies nicht möglich. (Die nötige Rechenzeit betrüge länger als das Alter des Universums; die nötige Energie wäre größer als die Gesamtenergie des Universums.)
- Bob erzeugt einen geheimen Schlüssel b und überträgt $g^b \pmod p$ über die unsichere Verbindung.
- Alice und Bob verwenden nun $g^{ab} = (g^a)^b = (g^b)^a \pmod p$ als gemeinsamen geheimen Schlüssel.

Die Sicherheit des Verfahrens beruht auf der Annahme, daß sich $a = \log_g A \pmod p$ nur durch mehr oder weniger intelligentes Ausprobieren ermitteln läßt. Trotz intensiver Forschung ist seit der Entdeckung des Verfahrens im Jahr 1976 kein Algorithmus gefunden worden, um $\log_g A \pmod p$ auf effizientere Weise zu berechnen; das Verfahren kann daher nach aktuellem Kenntnisstand als *sicher* angenommen werden.

8.8 Secure Shell (SSH)

- Aufruf einer Kommandozeile auf einem entfernten Rechner über eine verschlüsselte Verbindung
- Ersetzt unverschlüsselte Protokolle: *Remote Shell (RSH)*, *Telnet*

Verschlüsselter Verbindungsaufbau:

- Gemeinsamer geheimer Schlüssel durch Diffie-Hellman-Schlüsselaustausch
- Verschlüsselte Abfrage eines Passworts
  $\longrightarrow$ ohne weitere Maßnahmen: anfällig gegen Man-in-the-Middle-Angriffe
- Daher zusätzlich: Authentifizierung des Kommunikationspartners durch öffentliche Schlüssel („Zertifikate“): RSA (s. o.) oder DSA

  Es wird ein *Fingerprint* des öffentlichen Schlüssels abgefragt, den man mit einem ausgeschriebenen „`yes`“ bestätigen muß. Ob der Fingerprint tatsächlich der richtige ist, muß auf anderem Wege (persönlicher Datenaustausch, Telefon, Einschreiben) überprüft werden.
- Schwachstelle: Durchprobieren von Paßwörtern
  Daher optional: Authentifizierung des Benutzers durch öffentliche Schlüssel anstelle von Paßwörtern

  Schlüsselpaar erzeugen: `ssh-keygen`
  Öffentlichen Schlüssel hinterlegen: an Datei $HOME/.ssh/authorized_keys anhängen
- Schwachstelle: Diebstahl des geheimen Schlüssels
  Daher optional: Freigabe des geheimen Schlüssels durch ein Paßwort

  Wenn bekannt ist, daß ein geheimer Schlüssel gestohlen wurde,
  kann man den zugehörigen öffentlichen Schlüssel (das Zertifikat) sperren.

SSH setzt als Anwendung auf TCP/IP auf, arbeitet also in der TCP/IP-Schicht 4 = OSI-Schicht 7.

Port-Forwarding: Optionen `-L` und `-R`

SSH stellt einen komfortablen Mechanismus zur Verfügung, um auf entfernten Rechnern mit graphischen Programmen arbeiten zu können. Wenn beim Aufruf die Option `-X` angegeben wird, wird SSH für entfernte Programme selbst zum X-Server. Es setzt die `DISPLAY`-Variable für entfernte Programme so, daß diese ihre Grafikbefehle an SSH senden. Dieses leitet sie an den lokalen X-Server weiter, so daß die Grafikbefehle auf dem Bildschirm des Aufrufers ausgeführt werden.

Eine weitere nützliche Option für SSH ist Kompression: `-C`

Um Dateien über SSH zu kopieren, enthält die SSH-Software das Programm `scp`.

8.8 Virtual Private Network (VPN)

8.8.1 PPP über SSH

Dies ist ein Spezialfall von *PPP über Software* (Abschnitt 3.5.4) Wie dort weisen wir `pppd` an, anstelle einer Schnittstelle ein anderes Programm aufzurufen und dessen Standard-Ein- und -Ausgabe für die Datenübertragung zu nutzen.

Damit dies funktioniert, muß auf beiden Rechner jeweils ein Server und ein Client für die Datenübertragung laufen (`nc` bzw. `ssh`) sowie jeweils ein `pppd`.

Bei `nc` erreichen wir dies dadurch, daß wir auf jedem Rechner jeweils einen `pppd` starten, der dann seinerseits einen `nc`-Server (`nc -l -p 1234`) bzw. `nc`-Client (`nc` *Rechner* `1234`) startet.

Bei `ssh` läuft der SSH-Server bereits. Der Server-`pppd` kann die Standard-Ein- und Ausgabe nur vom SSH-Server ausgehändigt bekommen. Dieser händigt sie jedoch nur an einen SSH-Client aus. Im Gegensatz zu `nc` ist der SSH-Client jedoch in der Lage, weitere Programme aufzurufen und ihnen so die Standard-Ein- und Ausgabe zu übertragen. Der Aufruf des Server-`pppd` muß also innerhalb des SSH-Client erfolgen, der seinerseits vom Client-`pppd` aufgerufen wird.

Mit anderen Worten: Anstatt auf jedem der Rechner jeweils einen `pppd` zu starten, der jeweils ein `nc` aufruft, rufen wir auf *einem* Rechner einen `pppd` auf, der einen SSH-Client startet, der sich mit dem bereits laufenden SSH-Server verbindet und einen zweiten `pppd` startet und mit seiner Standard-Ein- und Ausgabe verbindet.

Der Aufruf lautet somit:

```
# pppd pty "ssh Rechner pppd 115200 nodetach noauth notty \
            10.1.42.1:10.1.42.2" passive debug nodetach noauth
```

Während PPP über `nc` eher von akademischem Interesse ist, ermöglicht es PPP über SSH, zwei Netze, die nur durch ein unsicheres Netz miteinander verbunden sind, über eine verschlüsselte Verbindung miteinander *sicher* zu verbinden. Dadurch wird Teilnehmern beider Netze ermöglicht, Dienste des jeweils anderen Netzes zu verwenden, ohne daß Teilnehmer in dem dazwischenliegenden unsicheren Netz Zugriff auf diese Dienste bekämen.

Eine derartige gesicherte Verbindung zweier privater Netze heißt *Virtual Private Network (VPN)*.

8.8.2 OpenVPN

Ähnlich wie der PAN-Daemon den Verbindungsaufbau über Bluetooth mit den Aufgaben des PPP-Daemons vereinigt, kombiniert OpenVPN die Funktion von PPP über SSH zum Aufbau eines VPN in einer einzigen Software.

Ein OpenVPN-Server lauscht – ähnlich wie ein SSH-Server – auf einem TCP-Port. Nach dem Verbindungsaufbau stellt OpenVPN – ähnlich wie ein PPP-Daemon – eine Punkt-zu-Punkt-Verbindung zur Verfügung.

Ähnlich SSH stellt OpenVPN mehrere Authentifizierungsmechanismen zur Verfügung:

- **Pre-Shared-Key:** Auf Server und Client wird im Vorfeld ein gemeinsamer geheimer Schlüssel fest installiert.

- **Passwort-Authentifizierung:** Beim Verbindungsaufbau muß ein Passwort eingegeben werden.
- **Öffentliche Schlüssel:** Auf dem Server wird ein öffentlicher Schlüssel hinterlegt; der Client authentifiziert sich mit Hilfe des zugehörigen geheimen Schlüssels. Dieser kann sich in einer Datei auf der Festplatte befinden oder auf spezieller Hardware (Chipkarte, USB-Token, ...).

OpenVPN setzt als Anwendung auf TCP/IP auf, arbeitet also in der TCP/IP-Schicht 4 = OSI-Schicht 7. Das von OpenVPN bereitgestellte VPN transportiert IP-Pakete (TCP/IP-Schicht 2 = OSI-Schicht 3); OpenVPN ersetzt somit Protokolle der TCP/IP-Schicht 1 = OSI-Schicht 2 (z. B. Ethernet).

(Alternativ kann OpenVPN auch Ethernet-Frames transportieren. Dies ist jedoch weniger effizient und erfordert mehr Sorgfalt bei der Konfiguration.)

8.8.3 Internet Protocol Security (IPsec)

Ein weiteres weitverbreitetes Verfahren zum Aufbau eines VPN ist *IPsec*. Dieses setzt nicht auf TCP/IP auf, sondern stellt selbst eine Alternative zu IP dar (TCP/IP-Schicht 2 statt 4 bzw. OSI-Schicht 3 statt 7). Für Verwaltungsaufgaben (Authentifizierung, Schlüsselaustausch, Prüfen der Verbindung, ...) kommt zusätzlich UDP zum Einsatz.

IPsec stellt daher nicht zusätzliche IP-Schnittstellen zur Verfügung (wie etwa PPP oder OpenVPN), sondern eigene Schnittstellen und Routing-Tabellen. Die Verwaltung erfolgt nicht über die Standardwerkzeuge (z. B. `ifconfig` und `route`), sondern über neue Kommandos (z. B. `ipsec eroute` anstelle von `route`).

IPsec verfügt über eine Vielzahl konfigurierbarer Optionen, die an beiden Endpunkten des VPN-Tunnels zueinander passend eingestellt werden müssen.

Obwohl IPsec als herstellerunabhängiges Protokoll definiert wurde, führen die o. a. Komplexitäten in der Praxis häufig dazu, daß nur IPsec-Produkte desselben Herstellers miteinander zusammenarbeiten. Alternativen wie z. B. OpenVPN erweisen sich in dieser Hinsicht oft als pflegeleichter.

> "IPsec was a great disappointment to us. Given the quality of the people that worked on it and the time that was spent on it, we expected a much better result."
>
> „IPsec war eine große Enttäuschung für uns. In Anbetracht der Qualifikation der Leute, die daran gearbeitet haben, und der Zeit, die dafür aufgebracht wurde, haben wir ein viel besseres Ergebnis erwartet."
>
> – Bruce Schneier, Niels Ferguson: A Cryptographic Evaluation of IPsec (S. 1, Abs. 2), übersetzt durch und zitiert nach: http://de.wikipedia.org/wiki/IPsec#Konzeptionelles

In IPv6 ist die Funktionalität von IPsec über zusätzliche Header-Felder bereits integriert.

9 Programmierung

Das Beispiel-Programm server.c öffnet Port 1234 und liefert dem Client die Botschaft: „Hello, world!"

Das Beispiel-Programm client.c nimmt Verbindung mit dem Rechner localhost auf Port 1234 auf und sendet dorthin die Botschaft: „Hello, world!"