

CNA Summary

von Eldar Omerovic - 31.06.2023

Dieses Dokument basiert auf der [Zusammenfassung von Cedric Gisler auf StudentBox](#). Die Zusammenfassung von Cedric Gisler ist bereits schon sehr gut und für diejenigen die noch weitere Details aus den Vorlesungen selbst hier vermissen, sollten sich seine Zusammenfassung ansehen.

Ich versuche dieses Dokument so zu gestalten, dass es sich ideal für die Prüfungsvorbereitung eignet. Dazu nutze ich die bestehende Zusammenfassung von Cedric Gisler sowie auch Modulunterlagen CNA (Frühlingssemester 2023) und natürlich das Internet.

Abbildungen wurden oft aus den Vorlesungsfolien der Hochschule Luzern entnommen (Screenshots)! Dies ist jedoch kein offizielles Dokument der Hochschule Luzern und kann durchaus Fehler enthalten!

RA.01 - Einführung

Was ist und kann ein Rechner

Definition eines Rechners

Turingmaschine

Was muss ein Rechner können

Entwicklung der Rechner

Mikroprozessoren

Speicher (Memory)

Die von Neumann Maschine

RA.02 - Zahlensysteme, Digitaltechnik

Verschiedene Basen

Darstellung negativer Zahlen

Nachteile und Vorteile — 1er & 2er Komplement

IEEE-754

Darstellung von Zahlen

Logikgatter

Addierer

Flip-Flop

RA.03 Rechenleistung

Harvard Architektur

Parallelverarbeitung

Superskalare Architekturen

Parallelverarbeitung auf der Prozessorebene (Processor-Level Parallelism)

Context-Switching

Multithreading Implementierungen

Benchmarks

Supercomputing

Quantencomputer

RA.04 - Holzcomputer

Befehlsliste des HOLZI

Befehlsabarbeitung

RA.05 - Befehle & Steuerwerk

Befehlscodierung und Adressierung

Adressbefehle

Adressierungsarten

Das Steuerwerk als Teil der CPU

RA.06 - Erweiterte Rechenarchitektur

Der Stack

Unterprogramme

Interrupts

Spezialprozessoren

SI-Vorsätze

RA07- Technologische Grundlagen

Halbleitertechnologie

Speichertechnologie

Cache-Speicher

Die Speicherhierarchie

Der Endian

Rechnerklassifikationen

Klassifikation nach Flynn

Kodierung

RA.08 - Bestandteile eines PC

Motherboard

Speicher-Busse und -Module

PCI, PCIe & Schnittstellen

Wireless

Grafikkarten

RAID - Redundant Array of Independent Disks

Peripheriegeräte

RA.01-RA.08 Formeln & Berechnungen

Speichergrösse

Boolesche Operationen mit Variablen

BS.01 - Aufgaben eines Betriebssystems

Programmabarbeitung

Interaktion mit I/O Geräten

Initialisierung von Geräten (Rechnerkonfiguration & BIOS)

Shells

Systemaufrufe

Was gehört zum Betriebssystem (OS)

BS.EL - Enterprise Lab

BS.02 - Prozesse

Multitasking

Prozesse

Scheduling

Threads

Prozesssynchronisation

Semaphore

Mutex

Monitor

Message-Passing

Die drei klassischen IPC Probleme

BS.03 - Speichermanagement, I/O, Dateisysteme

Speichermanagement

Speicherzuordnung

Dynamic Memory Allocation

Garbage Collection, Defragmentierung

Adress Relocation

Virtual Memory

Ein- / Ausgabe (I/O)

Dateisysteme

Dateien

Filesystem - Implementierung

Partitionen und Blöcke

Dateisysteme

Zugriffsschutz

Pipes

Konsistenz von Filesystemen

BS.04 - Verteilte Systeme

Das Client-Server Modell

Entfernte Objekte (Remoting)

Peer-to-Peer Netzwerke (P2P)

Cloud Deployment Models

Public Cloud

Private Cloud

Hybrid Cloud

Cloud Service Models

IaaS

PaaS

SaaS

Shared Responsibility

Vorteile und Nachteile der Cloud

Virtualisierung

Containerisierung

Vorteile beim Arbeiten mit Containern

Nachteile beim Arbeiten mit Containern

BS01 - BS04 Berechnungen & Formeln

Zugriffszeiten berechnen

NW.01 - Grundlagen und Modelle

Anforderungen an ein Netzwerk

Datenübertragung

Latenzzeit

Übertragungsfehler

Das Schichtenmodell

Service Access Points

[Service Kategorien](#)
[Referenzmodelle](#)
[OSI-Modell](#)
[Modell-Arten](#)
[Topologien](#)
[Adressräume](#)
[Endliche Automaten](#)
[NW.02 Physical Layer, Network Access](#)
 [Der Physical Layer](#)
 [Datenübertragung](#)
 [Kabel- und Leitertypen](#)
 [Wireless Übertragung](#)
 [Der Data Link Layer](#)
 [Paritätsbit](#)
 [Hamming-Abstand](#)
 [Hamming Code - Fehlererkennung und Fehlerkorrektur](#)
 [Redundanzprüfung Zyklisch mit CRC \(Cyclic Redundancy Check\)](#)
 [Erzeugung von Frames](#)
[NW.03 - Zugriffsverfahren und 802.xx Familie](#)
 [Zugriffsverfahren](#)
 [Statische Kanalzuordnung](#)
 [Dynamische Kanalzuordnung](#)
 [Die 802.xx Familie](#)
 [LLC - Logical Link Control](#)
[NW.04 - Networking & Routing](#)
 [Netzwerk Layer](#)
 [Paketvermittlung](#)
 [Leitungsvermittlung](#)
 [Verbindungslos vs. Verbindungsorientiert](#)
 [Routing-Algorithmen](#)
 [Übersicht der Routing-Algorithmen](#)
[NW.05 - Internetworking](#)
 [Netzwerkgeräte](#)
 [Router vs. Bridges](#)
 [Gateway und Firewall](#)
 [Das Internet Protokoll](#)
 [IP-Header](#)
 [Fragmentierung des IP Pakets](#)
 [IP-Adressen IPv4](#)
 [Spezielle IP-Adressen, Bereiche und Subnetze](#)
 [Arbeitsweise DHCP Protokoll](#)
 [IP-Adressen: IPv6](#)
[NW.06 Transportschicht - UDP & TCP](#)
 [Sender und Empfänger](#)
 [End-zu-End Protokolle](#)
 [User Datagram Protocol \(UDP\)](#)
 [Transmission Control Protocol \(TCP\)](#)
 [Aufbau einer End-zu-End Verbindung mit 3-Way-Handshake](#)
 [Übersicht der TCP Ports](#)
 [TCP Header](#)
 [TCP vs. UDP](#)
[NW.07 - Quality of Service](#)
 [Überlastüberwachung](#)
 [Flusskontrolle \(Datenflussteuerung, Data Flow Control\)](#)
 [Dienstgüte, Quality of Service \(QoS\)](#)
[NW.08 - Sicherheit](#)
 [Bedrohungen](#)
 [Diverse weitere Bedrohungen](#)
 [Sicherheitsmechanismen](#)
 [VPN - Virtuelles Privates Netzwerk](#)
 [Verschlüsselung, Kryptografie, Ciphering](#)
 [Hashalgorithmen](#)

[Das Secure Socket Layer \(SSL\) und Zertifikat](#)

[Security Policy](#)

[Die Firewall](#)

[Steganografie](#)

[NW.09 - Die letzte Meile & Satelitenkommunikation](#)

[Architektur der physischen Kanäle](#)

[Alternativen zu Kupfer & Glasfaser](#)

[Sateliten-Netz](#)

[NW.01 - NW.09 Formeln & Berechnungen](#)

[Fluktuation bestimmen \(Datenübertragung\)](#)

[Latenzzeit bestimmen](#)

[Bitfehlerrate bestimmen](#)

[Paketfehlerrate bestimmen](#)

[Bandbreite bestimmen](#)

[Maximale Datenübertragungsrate bestimmen](#)

[Fehlererkennung und Korrektur mit Hamming-Abstand](#)

[CRC Bestimmung mittels Polynom-Division](#)

[Token Bucket Berechnungen](#)

[Warteschlangen Berechnung](#)

[Weitere Informationen](#)

RA.01 - Einführung

Was ist und kann ein Rechner

Die Art und Weise wie ein Rechner aus den einzelnen Bausteinen aufgebaut ist, beschreibt die Architektur des Rechners.

Physische Bausteine:

- Schalter (grundlegendes Element)
- Elektronenröhren
- Transistoren

Logische Bausteine

- Register, Speicher
- Vergleicher, Addierer
- Steuerwerke, Prozessor

Definition eines Rechners

Rechner können nach festen Regeln (Algorithmen) Daten verarbeiten.



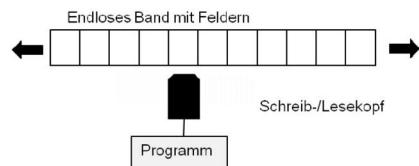
Ein Algorithmus ist eine Handlungsvorschrift zur Lösung eines Problems. Er dient als Grundlage, um Berechnungen auf einer Maschine oder manuell auszuführen. Die Art und Reihenfolge der Anweisung ist abhängig von der sprachlichen Darstellung des Algorithmus.

Turingmaschine

Die Turingmaschine war eines der ersten Rechenmodelle.

Man konnte 3 Operationen auf dieser durchführen:

- Lesen
- Schreiben
- Band bewegen



In diesem Speicherband kann in jeder Zelle jeweils ein Zeichen geschrieben werden. Der Programmablauf in einer Turingmaschine ist folgender:

1. Symbol unter Schreib-/Lesekopf lesen
2. Basierend auf dem gelesenen Symbol und dem Zustand entscheiden, welche Aktion mit dem gelesenen Symbol ausgeführt werden soll
3. Neues Symbol an diese Stelle auf dem Band setzen
4. Kopf links oder rechts bewegen
5. Zustand der Maschine aktualisieren

Dies geschieht in einer Schleife, bis ein Endzustand erkannt wird und ein Rückgabewert vorhanden ist.

Die Turingmaschine bietet eine schlechte Performance und hat keine praktische Anwendung mehr. Sie gilt als hilfreiches Denkmodell.



Turings Beweis:

„Die Maschine ist in der Lage, jedes vorstellbare mathematische Problem zu lösen, sofern dieses auch durch einen Algorithmus gelöst werden kann“

Was muss ein Rechner können

Ein Rechner muss in der Lage sein mithilfe Regeln (einem Algorithmus) anhand von gegebenen Daten ein Resultat zu berechnen.

Ein Rechner muss also folgendes können:

- Steuerwerk: Die Befehle eines Programmes der Reihe nach ausführen
- Speicher: Die Daten eines Programmes speichern
- Rechenwerk: Speicherinhalte interpretieren und verarbeiten, manipulieren, berechnen
- *Dabei muss die Ablaufsteuerung auf den Inhalt des Speichers reagieren können*

Auch wenn alle Rechner nach Definition nach "Rechner" sind, können sie sich in folgenden Aspekten unterscheiden:

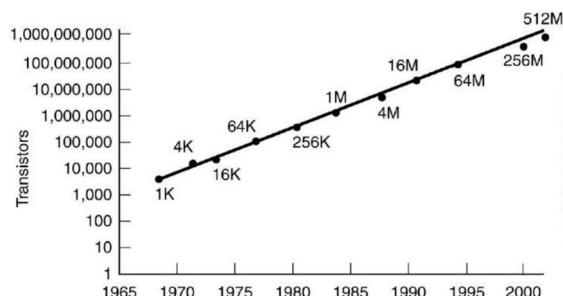
- Rechenleistung
- Ein- / Ausgabedatenrate
- Speichergrösse
- Leistungsverbrauch
- Ein- und Ausgabegeräte

Entwicklung der Rechner

Ein Rechner mit der Leistung eines Rechners 1976, der 8.8 Millionen US-Dollar gekostet hat, würde heute noch 5 Franken kosten.

Das Mooresche Gesetz (Moore's Law)

Das Mooresche Gesetz sagt uns: Es gibt regelmässig eine Verdoppelung der Komplexität von Rechnern. Verdoppelung der Komplexität heisst hierbei, dass sich die Transistoren pro Chip verdoppeln. Ursprünglich haben sich die Transistoren jedes Jahr verdoppelt, mittlerweile spricht man von einer Verdoppelung alle 18-24 Monate.



Mikroprozessoren

Ein Mikroprozessor ist ein Prozessor, der alle Grundfunktionen besitzt, auf einem Mikrochip vereinigt.

Aktuelle Mikroprozessoren besitzen mehr oder weniger folgende Werte, wobei diese von Typ zu Typ auch stark variieren können. Jedoch kann man folgende Aussagen treffen, wobei diese auf die meisten Mikroprozessoren zutreffen:

- ≥ 4 Kerne
- ≥ 1 Milliarde Transistoren
- 32-Bit oder 64-Bit (64-Bit in Home-Computern stärker vertreten)

Speicher (Memory)

Arbeitsspeicher: Viele Speicherzellen, Jede Speicherzelle besitzt eine Adresse als Zahl und einen Inhalt als (meist 8-Bit) Bitmuster.

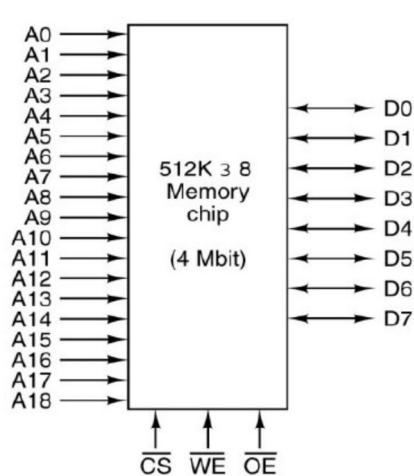
Register: Speicherzellen im Prozessor werden Register genannt. Mehrere Register nennt man einen Registersatz.

Unter Speicherarchitektur versteht man, dass jede Speicherzelle adressiert werden kann. Ein Speicher besteht zumeist aus 2^n Speicherzellen. z. B.

$$2^{16} = 65'536 = 64k \text{ Speicherzellen}$$

$$2^{19} = 524'288 = 512k \text{ Speicherzellen}$$

Die typischen Steuereingänge für Speicher sind folgende:



- CS: Chip-Select
CS ist ein Eingang, der dazu dient, den Speicherchip auszuwählen oder zu aktivieren.
 - Niedrigpegel (0) → Speicher aktiv
 - Hochpegel (1) → Speicher inaktiv
- OE: Output-Enable
OE ist ein Eingang, der dazu dient, die Ausgänge des Speichers zu aktivieren oder zu deaktivieren.
 - Niedrigpegel (0) → Ausgänge aktiv
 - Hochpegel (1) → Ausgänge inaktiv

- WE: Write-Enable
WE ist ein Eingang, der dazu dient, den Schreibzugriff auf den Speicher zu steuern.
 - Niedrigpegel (0) → Schreibzugriff aktiv
 - Hochpegel (1) → Schreibzugriff inaktiv
- R/W: Read-Write
R/W ist ein Eingang, der den Lese- oder Schreibmodus des Speichers auswählt.
 - Niedrigpegel (0) → Schreibmodus
 - Hochpegel (1) → Lesemodus

In einigen Memory-Chips wird eine Kombination von CS, OE und WE verwendet, während andere nur CS und R/W verwenden. R/W ist eine andere Methode um Schreib- und Lesemodus zu steuern. OE und WE können zusammen das gleiche wie R/W alleine. Dies kommt auf das Design des Chipsatzes an.

Ein Speicher funktioniert in grob wie folgt:

Lesen

1. Die CPU gibt eine Adresse auf den Adressbus aus.
2. Die CPU aktiviert die Steuersignale. Damit wird ein oder mehrere Speicherbausteine aktiviert. z.B CS = aktiv (0), R/W = Read (1)
3. Der Speicher legt den Inhalt der adressierten Zelle an seine Ausgänge und somit auf den Datenbus.
4. Die CPU liest den Wert vom Datenbus

Schreiben

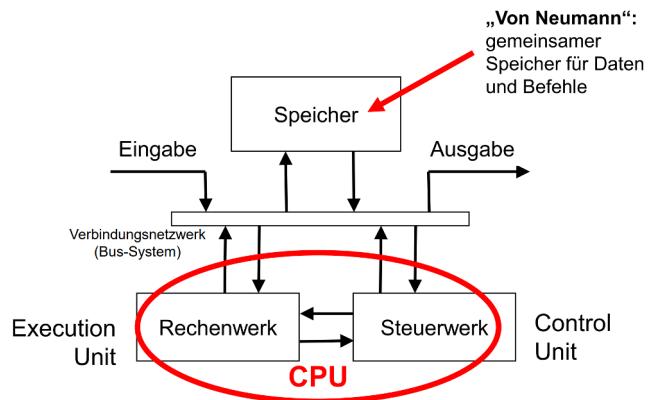
1. Die CPU gibt eine Adresse und die zu speichernden Daten aus.
2. Die CPU aktiviert die Steuersignale damit geschrieben werden kann. z.B CS = aktiv (0), R/W = Write (0)
3. Der Speicher legt die Daten in seine Eingänge und speichert diese in die adressierten Zellen.

Die von Neumann Maschine

Die Von Neumann-Maschine ist eine grundlegende Computerarchitektur, bei der Programme und Daten im selben Speicher abgelegt sind. Sie besteht aus einer CPU, einem Speicher und einem Ein- und Ausgabesystem. Die CPU interpretiert und führt Befehle aus, während der Speicher Programme und Daten enthält. Die Von Neumann-Architektur bildet die Grundlage für moderne Computer.

Folgende Komponenten besitzt eine von Neumann Maschine:

- Rechenwerk
- Steuerwerk
- Speicher
- Ein-Ausgabe



Fetch / Decode / Execute — Lifecycle

Fetch: Nächsten Befehl in das Befehlsregister laden, dann PC++ (Programcounter erhöhen)

Decode: Aktion des Befehls ermitteln, falls Daten aus dem Speicher benötigt werden, diese laden

Execute: Befehl (Aktion) ausführen

Fetch → PC++ → Decode → Execute

RA.02 - Zahlensysteme, Digitaltechnik

Verschiedene Basen

Das 10-er Zahlensystem ist nicht das einzige, das verwendet wird. Zwei andere die auch häufig gebraucht werden sind Binär und Hex. Das Binäre Zahlensystem hat eine Basis von 2 (0,1), das hexadezimale Zahlensystem hat eine Basis von 4 (0-F).

Das oktale ist auch sehr bekannt, wird jedoch praktisch weniger genutzt als Binär und Hexadezimal.

Dezimal	Binär	Oktal	Hex
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Dezimal	Binär	Oktal	Hex
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Dezimal zu Binär, 40 → 00101000

128	64	32	16	8	4	2	1
0	0	1	0	1	0	0	0

Aus dem binären wiederrum, kann man recht einfach die anderen Systeme errechnen.

Oktal, 8 → 3^2 = 3er Blöcke

Hexadezimal, 16 → 4^2 = 4er Blöcke

Oktal

128	64	32	16	8	4	2	1
0	0	1	0	1	0	0	0

Hier unterteilen wir die Zellen von rechts nach links in 3er Blöcke.

1. 000 → dezimal: 0
2. 101 → dezimal: 5
3. 000 → dezimal: 0

Dies lesen wir nun von unten nach oben und wir erhalten: 050

40 = 0o50

Kennzeichnung:

50₈ oder 0o50 oder @50

Hexadezimal

128	64	32	16	8	4	2	1
0	0	1	0	1	0	0	0

Hier unterteilen wir die Zellen von rechts nach links in 4er Blöcke.

1. 1000 → dezimal: 8
2. 0010 → dezimal 2

Auch hier lesen wir das von unten nach oben und wir erhalten: #28

40 = 0x28

Bei den Hexwerten ≥ 10 ist es so, dass wir Buchstaben A-F verwenden.

28₁₆ oder 0x28 oder \$28

Darstellung negativer Zahlen

Es gibt unterschiedliche Möglichkeiten negative Zahlen darzustellen.

Dez	Binär	-N Vorzeichen	-N 1er Komplement	-N 2er Komplement	-N excess127
8	0000 1000	1000 1000	1111 1110	1111 1111	0111 0111
9	0000 1001	1000 1001	1111 0110	1111 0111	0111 0101

- -N Vorzeichen: Das erste Bit von links wird verändert
- -N 1er Komplement: Die Bits werden gekehrt
- -N 2er Komplement: Die Bits werden gekehrt und es wird +1 addiert
- -N excess127: +127 — Nullpunkt verschieben

Nachteile und Vorteile — 1er & 2er Komplement

1er Komplement

Nachteile

- Übertrag muss in zusätzlichem Schritt addiert werden.
- 2 Darstellungen für 0 → 0000 0000 und 1111 1111

Vorteile

- Schneller bei Division und Multiplikation bei doppelt so langen Ergebnissen
- Einfachere Bildung von Prüfsummen

2er Komplement

Nachteile

- Komplementbildung etwas komplexer

Vorteile

- Nur eine Darstellung für 0
- Addition des Übertrags entfällt

IEEE-754

1. Zahl von dezimal zu binär umwandeln

205 → 1100 1101

2. Jetzt müssen wir den Exponenten ermitteln, in dem wir den Binärwert so schieben, dass ganz vorne nur noch die 1 steht.
1100 1101.00 → 1.1001101, hier also um 7 Stellen.

3. Um den Exponent zu berechnen, rechnen wir nun die 7 (Stellen) und addieren diese mit 127 (Excess) → 134. Dies wandeln wir zu Binär um:
Exponent: 1000 0110

4. Da die Zahl positiv ist, bleibt unser Vorzeichen ebenfalls 0

5. Jetzt fügen wir dies in die 32-Bit Tabelle ein

Vorzeichen: 0

Exponent: 10000110

Mantisse: 10011010000000000000000000000000

0	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

Wenn wir dies nun in 4er Blöcke teilen, können wir dies einfach als Hex darstellen

0100 → 4 → 0x4

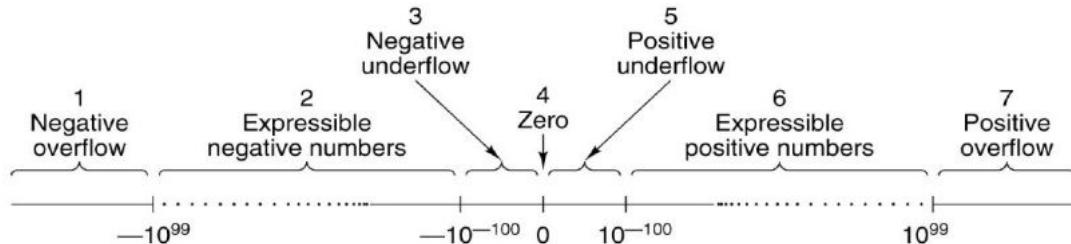
0011 → 3 → 0x3

0100 → 4 → 0x4

1101 → 13 → 0xD

Und so wissen wir das 205 in IEEE-754 = 0x434D0000

Darstellung von Zahlen

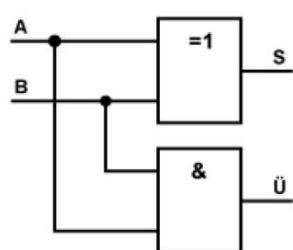


Logikgatter

Name	IEC 60617-12 (ANSI/IEEE Std 91)	ASA (USA) (ANSI/IEEE Std 91/91a)	Beschreibung
AND	A —&— Y B —	A—out B—	Für Output = 1 müssen alle Inputs 1 sein.
NAND	A —&— o—Y B —	A—o—out B—	dito mit Invertierung
OR	A —≥1— Y B —	A—out B—	Für Output = 1 muss mindestens 1 Input 1 sein.
NOR	A —≥1— o—Y B —	A—o—out B—	dito mit Invertierung
XOR	A —=1— Y B —	A—o—out B—	Für Output = 1 muss genau 1 Input 1 sein.
NOT	A —1— o—Y	A—o—out	Invertierung.

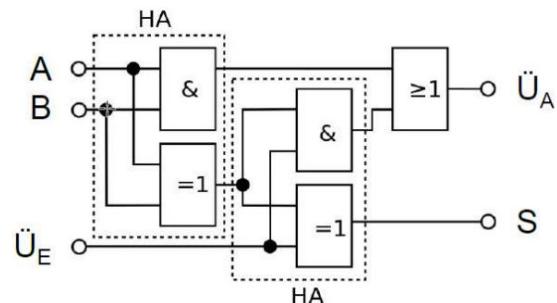
Addierer

Halbaddierer: (HA)



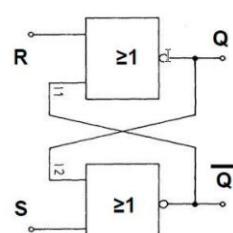
B	A	\bar{U} (2^1)	S (2^0)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Voll addierer: (VA)



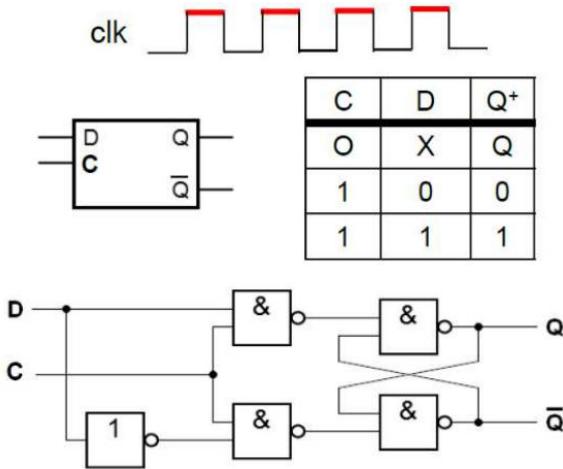
Flip-Flop

Ein Flip-Flop dient dazu, einzelne Bits einer elektronischen Schaltung zu speichern. Wichtig ist, dass die Spannungsversorgung dauernd gewährleistet ist.



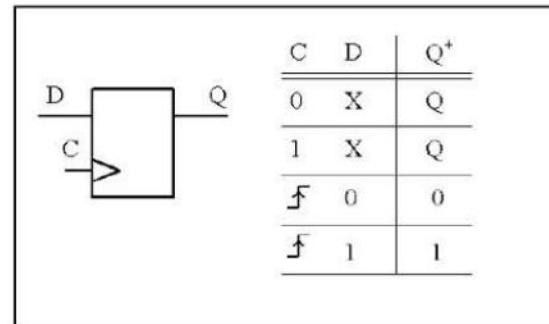
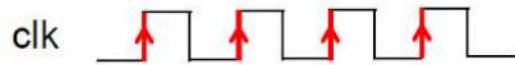
Taktzustandgesteuert

Wechsel des Zustands nur während des positiven Zustands des Taktsignals (clock)



Flankengesteuert

Wechsel des Zustands nur bei der Flanke des Taktsignals (clock, clk)

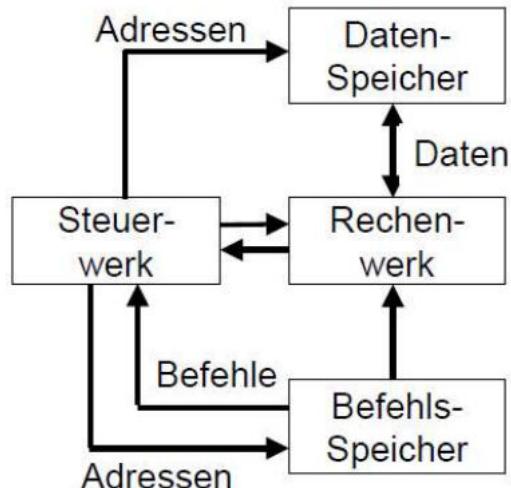


RA.03 Rechenleistung

Harvard Architektur

Der Unterschied zur von Neumann Architektur ist, dass die Daten und Befehle sich nicht im gleichen Speicher befinden. Es gibt je einen Speicher für beide. Deswegen gibt es auch separate Busse zu den beiden Speichern. Der Vorteil dabei ist, dass das Laden eines Befehls gleichzeitig mit dem Laden oder Speichern von Daten geschieht.

Im Idealfall kann man also in einem Taktzyklus die Befehle und die Daten laden. Im Gegensatz dazu sind in der von Neumann Architektur mindestens zwei Taktzyklen nötig.



Weitere Vorteile dieser Architektur sind

- das synchrone Laden auch mit mehreren Rechenwerken (Vektorrechner, SIMD, MIMD)
- Strikte Trennung von Daten und Programmen. Ergibt so einen Speicherschutz und eine einfache Trennung von Zugriffsrechten
- Datenwortbreite und Befehlswortbreite sind unabhängig voneinander festlegbar

Moderne Prozessoren verwenden zumeist eine Mischform aus Harvard- und von-Neumann-Architektur.

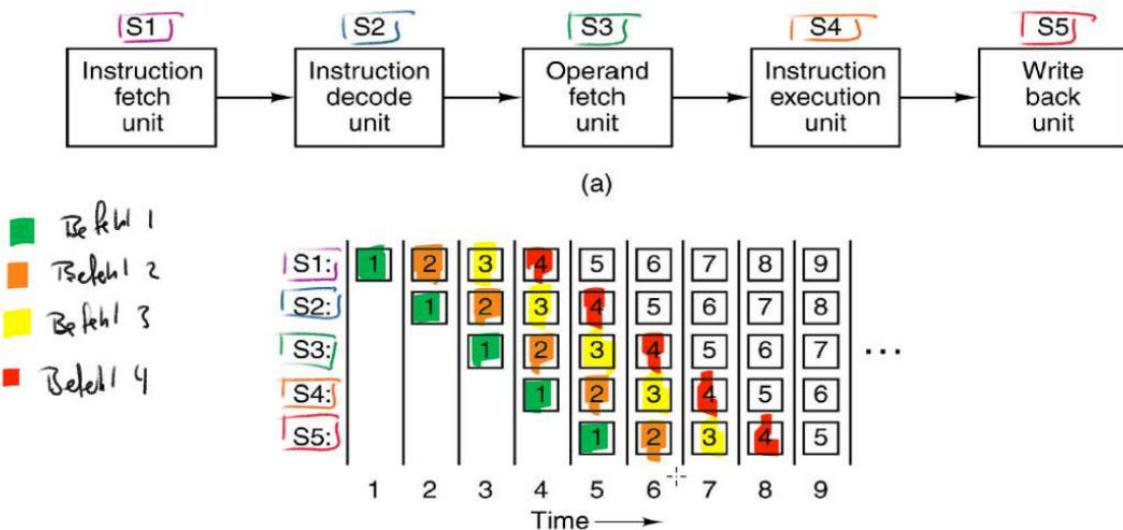
Parallelverarbeitung

Es kann auf verschiedenen Ebenen parallel gearbeitet werden

- Auf der Instruktionsebene (Instruction-Level Parallelism)
- Auf der Prozessorebene (Processor-Level Parallelism)

Für die Parallelverarbeitung gibt es eine Pipeline. In dieser werden Befehle wie am Fliessband überlappend abgearbeitet.

Beispiel einer 5-stufigen Pipeline. Zustand der 5 Stufen im Laufe der Zeit. Hier sind 9 Taktzyklen dargestellt. Während Befehl 1 im 2. Taktzyklus decoded wird, wird Befehl 2 gefetched, usw...



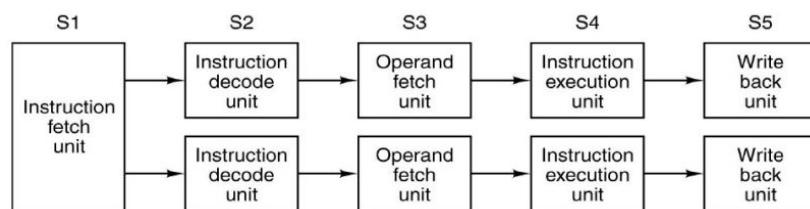
Solche Pipelines haben diverse Features

- Pipeline stall: Bei Konflikten die Verarbeitung anhalten
- Branche-Prediction: Sprungvorhersage
- Flush: Leert die Instruction pipeline in hazard conditions
- Speculative execution: Vorläufige Ausführung beider Programmabläufe

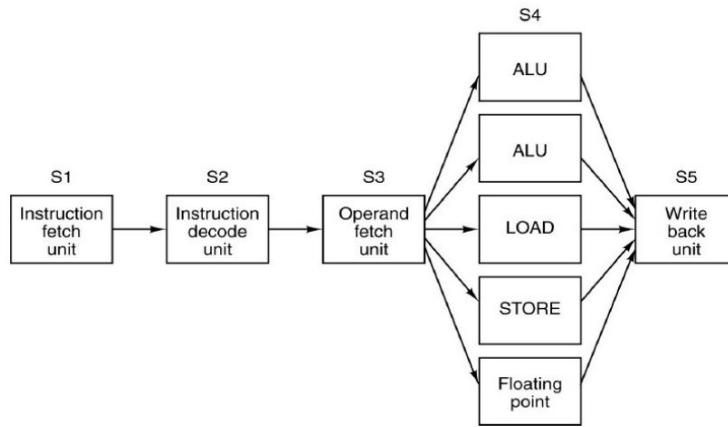
Superskalare Architekturen

Um die Rechenleistung noch mehr zu erhöhen, können Pipelines und Parallelität gemeinsam verwendet werden.

Beispiel einer zweifachen 5-stufigen Pipeline mit gemeinsamer Instruction-Fetch-Einheit.



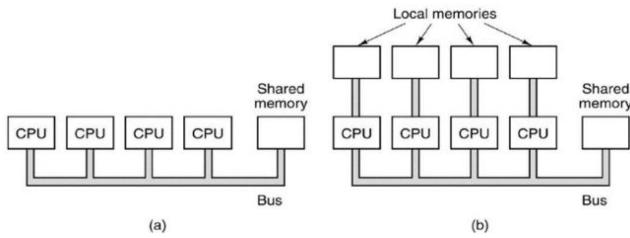
Beispiel eines superskalaren Prozessor mit 5 unterschiedlichen Ausführungseinheiten.



Parallelverarbeitung auf der Prozessorebene (Processor-Level Parallelism)

Sobald mehrere Prozessoren daran arbeiten, kann die Verarbeitungsgeschwindigkeit weiter gesteigert werden.

Ein Prozessorchip kann auch 2 oder mehrere Rechenkerne (Cores) enthalten. Somit können mehrere Kerne gleichzeitig mehrere Programme bearbeiten. Diese Kerne teilen sich alle andere Ressourcen (Zugriff zum Cache, RAM, etc) und können sich daher gegenseitig blockieren.



a) Ein Mehrprozessorsystem mit einem Bus

b) Ein Mehrprozessorsystem mit lokalen Speichern

Context-Switching

Wenn das Betriebssystem einen anderen Thread ausführen möchte, dann wird zu diesem Thread umgeschaltet. Das heisst, es werden die Register Zustände dieses Threads geladen.

Das Context-Switching braucht aber Zeit. Abhilfe schafft da das Multithreading

- Ein Core kann quasi gleichzeitig mehrere (typisch 2) Threads bearbeiten.
- Das Betriebssystem sieht mehrere logische Prozessoren, auf die es die Threads verteilen kann.
- Es kann schnell zwischen diesen Threads umgeschaltet werden.

Multithreading Implementierungen

Bei INTEL nennt man dieses System das Hyper-Threading Technology (HTT) und bei AMD Simultaneous Multithreading (SMT).

HTT wird bei neueren Prozessoren angewendet. Es gibt hierbei nur ein komplexes Steuerwerk und nur Teile der Architektur sind doppelt vorhanden.

- Mehrere vollständige Registersätze
- Pipeline (ALU, FPU, ...) nur einmal vorhanden
- Threads können auf einem Core nicht gleichzeitig bearbeitet werden, nur nacheinander.
- Ein Prozess kann aus mehreren Threads bestehen
- Das Umschalten zwischen Threads geht schneller als das Umschalten zwischen Prozessen
- Beim Hyperthreading kann jeder Core (in der Regel 2) Threads bearbeiten
- Beim Hyperthreading sind in jedem Core Teile mehrfach vorhanden, vor allem Register

Benchmarks

Leistungsmessungen werden gemacht, um den richtigen Rechner für eine Anwendung zu finden. Man versucht so die Zeit zu messen, in der der Rechner gewisse Aufgaben erledigen kann.

Eine Einheit um die Leistung wiederzugeben sind MIPS. Million Instruction Per Second.

Die MIPS sind aber Abhängig vom Instruktionssatz, variiert je nach Programm auf dem selben Rechner, was ein Problem ist.

Die Ausführungszeit für ein Programm (Rechenzeit) hängt ab von

- Technologien
- Instruktionssatz
- Architektur des Systems

Ausführungszeiten und Rechenzeit sind sehr wichtig bei Echtzeitsystemen.

Supercomputing

Ein Supercomputer ist ein Hochleistungsrechner. Diese sind besonders schnell für ihre Zeit. Sie spielen eine essenzielle Rolle im wissenschaftlichen Rechnen und werden dort eingesetzt für Simulationen im Bereich der Quantenmechanik, Wettervorhersage, Klimatologie, etc.

Top 1 Japanischer Supercomputer Fugaku:

7'630'848 Cores | Rmax 442'010.0 TFlop/s | Rpeak 537'212.0 TFlop/s | Power 29'899 kW = 30 MW

Der Schweizer Supercomputer Piz Daint ist auf Top12:

387'872 Cores | Rmax 21'230.0 TFlop/s | Rpeak 27'230.0 TFlop/s | Power 2'384 kW

Quantencomputer

Am 23.10.2019 meldet Google das ihr Quantencomputer eine Berechnung in 200 Sekunden getätigt hat, für die ein Supercomputer ~10'000 Jahre braucht.

Ein Quantencomputer kann gleichzeitig mit den Zuständen 0 und 1 und auch allen möglichen Kombinationen rechnen

RA.04 - Holzcomputer

Befehlsliste des HOLZI

Mnemonic	Code	Worte	Zyklen	Beschreibung
Datentransferbefehle:				
MVI R0	0100	2	8	unmittelbar folgendes Wort in Register 0
MVI R1	0101	2	8	unmittelbar folgendes Wort in Register 1
STO R0	0000	2	10	Register 0 in RAM; 2. Wort RAM-Adresse
STO R1	0001	2	10	Register 1 in RAM, 2. Wort RAM-Adresse
LD R0	0010	2	10	RAM in Register 0; 2. Wort RAM-Adresse
LD R1	0011	2	10	RAM in Register 1; 2. Wort RAM-Adresse
MOV R1,R0	1001	1	5	Register 0 in Register 1 kopieren
MOV R0,R1	1010	1	5	Register 1 in Register 0 kopieren
Input-/Outputbefehle:				
IN	1000	1	7	Input-Port in Register 0
OUT	1011	1	7	Register 0 in Output-Port
Arithmetische Befehle:				
ADD R1	1101	1	5	Register 1 + Register 0 (ohne carry), Resultat in R0
Rotationsbefehle:				
ASL	1110	1	5	Register 0 links schieben, Überlauf in carry, LSB <- 0
RAR	1111	1	5	R0 rechts schieben, CRY <- LSB, MSB <- CRY
Sprungbefehle:				
JMP	0110	2	8	Sprung zu ROM-Adresse in 2. Wort
JC	0111	2	8	Sprung zu ROM-Adresse in 2. Wort wenn CRY = 1
JNC	1100	2	8	Sprung zu ROM-Adresse in 2. Wort wenn CRY = 0

Befehlsabarbeitung

Um einen Befehl abzuarbeiten, benötigt der HOLZI folgende Taktzyklen:

- FETCH-phase: 4 Taktzyklen
- Zweites FETCH, falls es sich um einen Zweiwortbefehl handelt: 3 Taktzyklen
- EXECUTE-phase: 1 - 3 Taktzyklen, abhängig vom Befehl

RA.05 - Befehle & Steuerwerk

Befehlscodierung und Adressierung

Es gibt 3 Gruppen von Befehlen mit den Operationsarten:

- Datentransfer
- Arithmetische und logische Operationen
- Programmablaufsteuerung

Was muss ein Rechner können?

- Die Befehle eines Programms der Reihe nach ausführen
- Werte speichern
- Speicherinhalt interpretieren und manipulieren
- Ablaufsteuerung muss auf den Inhalt des Speichers reagieren

Ein allgemeiner Befehl enthält diverse Informationen, durchzuführende Operationen, Operanden, Adresse des nächsten Befehls.

Mnemonic: ADD #1234, addr1, addr2

- Die durchzuführende Operation steckt in einem Mnemonic, bei ADD = addieren.
- Typ, Länge, Adressierungsart und Adressangabe: implizit im Mnemonic und/oder durch die Angabe der Operanden, z.B. ADD, FADD: addiere Integer, addiere Floating-Point.

Maschienebefehle: \$A6 = 10100110

- Oft vollständig codiert, jeder Code hat eine bestimmte Bedeutung
- Manchmal haben bestimmte Bits eine feste Bedeutung

Adressbefehle

Es gibt Drei-Adressen-Befehl, Zwei-Adressen-Befehl, Ein-Adressen-Befehl und Nulladressbefehle. Die Adressbefehle zeigen auf wie viele Adressen auf einmal übertragen werden können.

Der Ein-Adressen-Befehl beinhaltet nur Operand und Quelle. Das Ergebnis wird in die Quelle geschrieben (z.B. Inkrementieren)

```
INC $800 ; $800 <- $800 + 1
```



In Assembly werden Kommentare mit einem Semikolon ";" von den Befehlszeilen getrennt

Der Zwei-Adressen-Befehl beinhaltet den Operanden mit den zwei Quellen. Das Ergebnis wird in eine der Quelladressen gespeichert

```
ADD $800, $801 ; $800 <- $800 + $801 (1. Option)  
ADD $800, $801 ; $801 <- $800 + $801 (2. Option)
```

Der Drei-Adressen-Befehl beinhaltet den Operanden mit den zwei Quellen sowie als dritte Adresse die Ziel-Speicheradresse.

```
ADD $800, $801, $802 ; $802 <- $800 + $801
```

Beim Null-Adressen-Befehl ist es eine implizite Adressierung. Operand bzw. Register ist im Befehl enthalten.

```
PUSH A  
PUSH B  
ADD ; adds A + B onto the stack
```

Ein Stack basiertes Programm, benötigt keine Speicheradressen. Hier kann das ADD die beiden obersten Elemente des Stacks addieren.

Je kürzer der Befehl, desto weniger Speicher benötigt man, um die Befehle zu speichern. Jedoch nimmt man dadurch die Flexibilität der Befehle weg.



Kürzere Befehle bedeuten also weniger Speicherverbrauch, aber auch weniger Flexibilität

Datenspeicherung

- CPU: In der CPU gibt es einzelne Register, diese zusammen nennt man einen Registersatz
- Befehle und Konstanten werden im Programmspeicher gespeichert
- Daten werden im Datenspeicher gespeichert

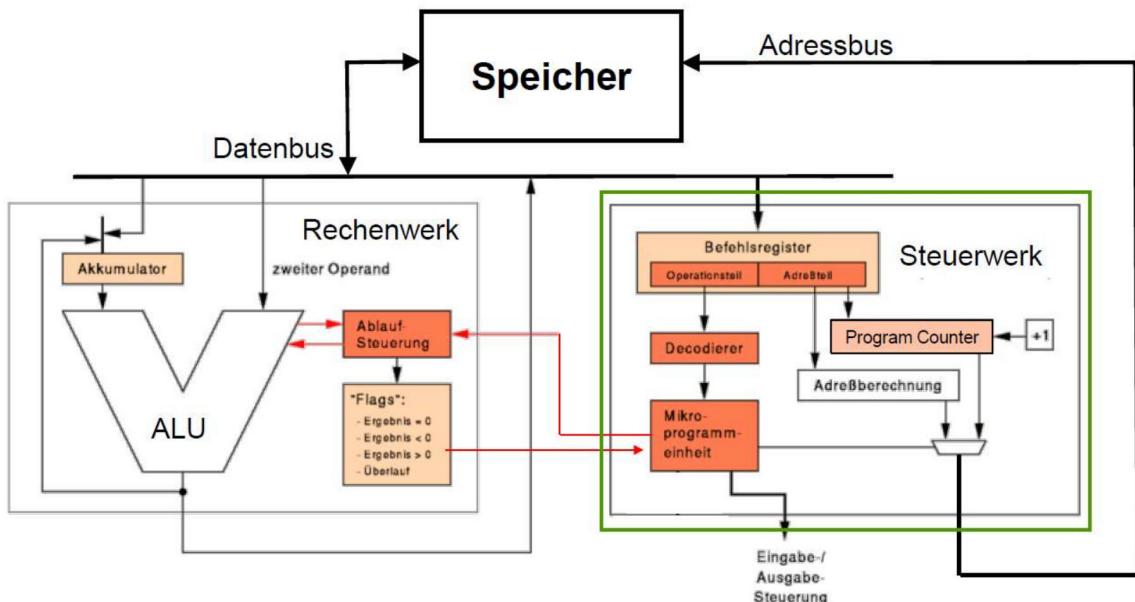
Adressierungsarten

Adressierungsart	Kurzbeschreibung	Beispiel	Erklärung
Absolute, direkte Adressierung	Absolute Adresse	LDA \$0800	Direkt Adresse \$0800 mitgeben
Registeradressierung	Name des Registers	LDA R1	Man gibt den Namen des Registers an
Unmittelbare Adressierung	Wertangabe	LDA #\$F3	Wert F3 wird direkt in die Speicherzelle geladen
Indirekte Adressierung	Relatives Register	LDA (IX)	Adresse ist im Register
Indizierte Adressierung	Absolute Adressierung mit Versatz	LDA \$0700 X	Direkte Adresse mit einem Versetzungs-Index



Die ersten 3 Adressierungsarten (Absolute -, Register- und Unmittelbare Adressierung), sind die wichtigeren.

Das Steuerwerk als Teil der CPU



Das Steuerwerk steuert den Ablauf der Befehlsbearbeitung. Es besteht aus folgenden Elementen:

- Program Counter: Zeigt wo die nächste Instruktion liegt
- Instruktionsregister: Register für die Instruktionen
- Adressregister: Speichert eine Adresse im Speicher
- Stackpointer: Ein Register welches den Zustand des Stack im Speicher angibt

Der Program Counter wird vom Prozess automatisch verändert, somit wird der Inhalt darin neu berechnet.

Das Steuerwerk kann auf zwei verschiedene Arten realisiert werden. Entweder wird diese festverdrahten mit diversen Gattern und FlipFlops realisiert, dies wird vor allem für einfache Befehlssätze (RISC, Reduced Instruction Set Computer) mit hohen Taktraten verwendet. Oder das

Steuerwerk wird mikroprogrammiert, heisst das Steuerwerk ist selber auch nochmal wie ein kleiner Rechner aufgebaut. Dies ist für einen komplexen Befehlssatz (CISC, Complex I.S.C). Somit wird das Mikroprogramm vom Chipentwickler festgelegt ist somit wie eine eigene Firmware des Prozessors

Heutige Prozessoren benutzen beide Arten (RISC & CISC).

RISC

Reduced Instruction Set Computer

- Einfache Instruktionen
- Große Zahl von allgemeinen Zweckregistern
- Feste Befehlsformate
- Harter Befehlsatz
- Pipeline-Architektur
- Load-Store-Architektur
- Hohe Befehlsgeschwindigkeit
- Wenige Adressierungsarten
- Geringer Code-Durchsatz
- Compiler-optimierte Architektur

CISC

Complex Instruction Set Computer

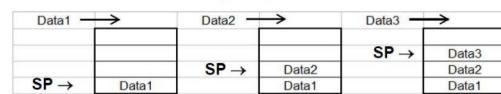
- Komplexe Instruktionen
- Wenige allgemeine Zweckregister
- Variable Befehlsformate
- Weicher Befehlssatz
- Mikroprogrammgesteuerte Architektur
- Memory-to-Memory-Architektur
- Geringe Befehlsgeschwindigkeit
- Vielfältige Adressierungsarten
- Hoher Code-Durchsatz
- Hardware-optimierte Architektur

RA.06 - Erweiterte Rechenarchitektur

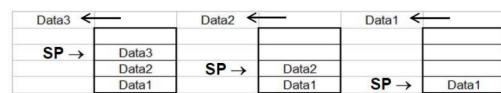
Der Stack

Ein Stack ist ein universeller Zwischenspeicher für Daten, auf ihm kann man Daten abspeichern ohne sich dabei um die Adressierung oder Benennungen von den Daten zu kümmern. Er funktioniert als Stapspeicher, heißt der älteste Eintrag wird auch als letztes entnommen. (LIFO, Last in – First out).

Daten in den Stack schreiben (push):



Daten aus dem Stack auslesen (pop):



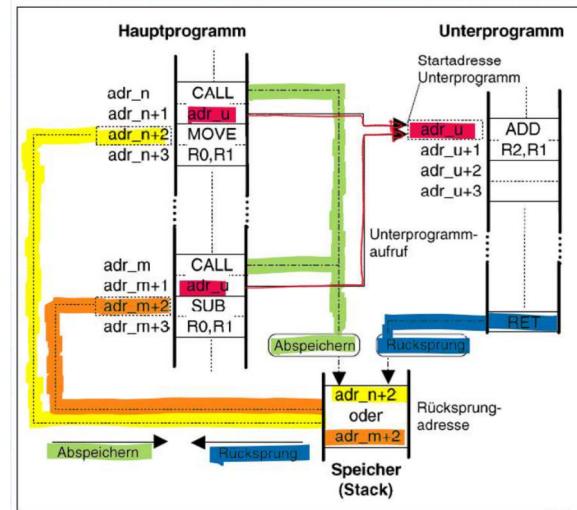
Der Stackpointer (SP, Stapelzeiger) zeigt immer auf die aktuelle Adresse, also auf den obersten Stapeleintrag.

Ein Stack wird für die Ausführung von Unterprogrammen, Parameterübergabe, Resultatrückgabe, Zwischenspeicher und Interrupt-Behandlung gebraucht. Der Stack wird im Arbeitsspeicher realisiert und es gibt mehrere Stacks z.B. für das System und für jeden einzelnen Prozess.

Unterprogramme

Unterprogramme, auch bekannt als Subroutinen oder Funktionen, sind Abschnitte von Code, die in Programmen wiederverwendet werden können. Sie ermöglichen es, bestimmte Aufgaben auszulagern und den Code übersichtlicher und modularer zu gestalten. Hier ist eine kurze Erklärung, wie Unterprogramme in der CPU funktionieren:

1. Aufruf des Unterprogramms: Wenn ein Programm an eine Stelle gelangt, an der ein Unterprogramm aufgerufen werden soll, wird der Befehl zum Aufrufen des Unterprogramms ausgeführt. Der CPU-Steuerkreis speichert die aktuelle Position des Programms (den Program Counter) und springt zum Anfang des Unterprogramms.
2. Parameterübergabe: Wenn das Unterprogramm Parameter erfordert, werden diese anhand der spezifizierten Aufrufkonventionen an das Unterprogramm übergeben. Dies kann über Register, den Stack oder andere Mechanismen erfolgen.



3. Ausführung des Unterprogramms: Die CPU führt die Anweisungen im Unterprogramm aus, um die gewünschte Aufgabe zu erledigen. Dies kann Berechnungen durchführen, Daten verarbeiten oder andere Operationen ausführen.
4. Rückkehr zum Hauptprogramm: Sobald das Unterprogramm abgeschlossen ist, wird die Kontrolle an das Hauptprogramm zurückgegeben. Der Program Counter wird auf den Wert vor dem Unterprogrammaufruf wiederhergestellt, und die Ausführung des Hauptprogramms wird fortgesetzt.
5. Ergebnisrückgabe: Falls das Unterprogramm ein Ergebnis erzeugt, wird dieses möglicherweise über spezifizierte Register oder über den Stapel an das aufrufende Programm zurückgegeben.

Dieser Ablauf ermöglicht die effiziente und strukturierte Verwendung von Unterprogrammen in der CPU, was zur Modularität, Wiederverwendbarkeit und Lesbarkeit von Programmen beiträgt.

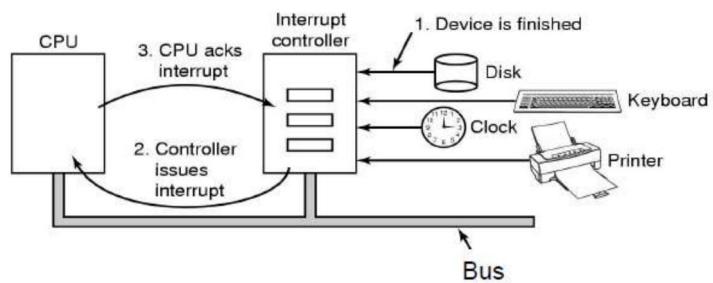
Interrupts

Der Interrupt ist eine Unterbrechungsanforderung für **dringende** Aufgaben durch ein externes Signal. Die Interrupt Service Routine (ISR) funktioniert ähnlich wie ein Unterprogramm (UP) beim UP jedoch ist der Zeitpunkt der Programmierung eingeplant und der ISR wird durch ein externes Ereignis ausgelöst.

Ein Rechner kann mehrere Interrupt Eingänge haben oder es wird ein Interrupt-Controller verwendet. Den verschiedenen Interrupt Eingängen werden Prioritäten zugeordnet. Sie können einzelne gesperrt oder freigegeben werden. (enabled/disabled). Es können auch mehrere Interrupts gleichzeitig aktiv sein, jedoch immer nur eine ISR. Ein Sonderfall hierbei stellt der Interrupt per Software dar. Eine Trap, wird beim Kontext-Switching des Betriebssystems und für Tests und Debugging verwendet.

Der Ablauf eines Interrupts:

1. Interrupt Request (IRQ) vom Gerät
2. Interrupt Controller bearbeitet die Aufforderung und erzeugt ein Interrupt Signal zur CPU.



3. CPU unterbricht, rettet Context auf den Stack und zeigt Acknowledge (INTA) an. Liest dann vom IRC die Interrupt Nummer und verzweigt mit Hilfe der Vektortabelle in die entsprechende ISR.
4. ISR liest das Gerätregister aus und startet entsprechende Aktionen
5. Return from Interrupt (RTI)

Spezialprozessoren

Mikrocontroller

- Mikroprozessor (μ P) + Peripheriefunktion + Speicher + Mikrochip = System on a Chip: SoC
- Anwendung: eingebettete Systeme (embedded systems)
- Bitbreite: 4-, 8-, 16- und 32-Bit Mikrocontroller
- Ein Mikrocontroller ...
 - Kann aus einer ARM core CPU bestehen
 - Wird meist in einem «embedded System» gebraucht
 - Hat meist on-Chip RAM
 - Hat meist ein on-chip programmierbaren ROM (Flash-ROM)
 - Beinhaltet eine CPU
 - Ist eine in sich abgeschlossenes System auf einem Chip

DSP Digitaler Signalprozessor (Digital Signal Processor)

- Zur Bearbeitung digitalen Signale (z.B. Audio- oder Videosignale)
- Mikroprozessor (μ P) + Multiplizier-Addier-Werke (Multiply-Accumulate, MAC)

GPU Grafikprozessor (Graphics Processing Unit)

- Grafikchip, VPU (Visual Processing Unit)
- Massiv parallele Systeme: Streamprozessoren (Shader), Textureinheiten, Render-Einheiten
- Prozessor ist extra auf seine Aufgaben abgestimmt:
 - Rechenintensive Aufgaben der 2D- und 3D- Computergrafik, Berechnen des Bildaufbaus, Raytracing, ...
 - Dekodieren von Video – Datenströmen
 - Vega-GPUs von AMD – für Mining von Kryptowährungen
 - Auch als KI Beschleuniger
- Ansteuerung über Grafik API: DirectX und OpenGL/Vulkan sowie OpenCL (auch für allgemeine Rechenaufgaben).
- Oft auf einer separaten (dedizierten) Grafikkarte oder IGP

SI-Vorsätze

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	m	milli	10^3	k
10^{-6}	0.000001	μ	micro	10^6	M
10^{-9}	0.000000001	n	nano	10^9	G
10^{-12}	0.000000000001	p	pico	10^{12}	T
10^{-15}	0.000000000000001	f	femto	10^{15}	P
10^{-18}	0.000000000000000001	a	atto	10^{18}	E
10^{-21}	0.000000000000000000001	z	zepto	10^{21}	Z
10^{-24}	0.000000000000000000000000000y	y	yocto	10^{24}	Y

Für Datenmengen - Datenübertragungsraten

Bitrate: 1 kbit/s = 1000 bit/s → jeweils *1000: Mbit/s, Gbit/s, englisch auch: bps → bits per second

Byterate: 1 B/s = 8 Bit /s

Für Datenmengen - Speichergrößen

1 Kilobyte (kB) = 1000 Byte, 1 MB = 1000 Kilobyte

Normgerecht mit Binärpräfixen mit dem Zusatz i

1 Kibibyte (KiB) = 1024 Byte, (grosses K hier wichtig)

1 Mebibyte (MiB) = $1024 * 1024$ Byte

RA07- Technologische Grundlagen

Halbleitertechnologie

Für den Bau eines Rechners, benötigt man Schalter. Denn darauf baut ein Rechner — An und Aus, 1 und 0. Dazu gab es im Verlauf der Jahre unterschiedliche Methoden diese An und Aus Zustände zu verwalten.

Elektronen Röhre

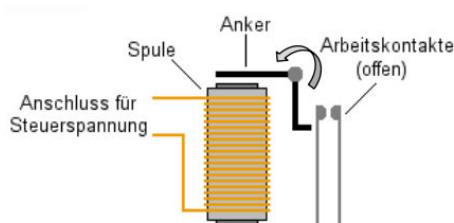


Vor allem von 1940 bis 1960 eingesetzt.

Schon ab 1920 für auch andere Anwendungen eingesetzt.

Hoher Stromverbrauch, Geringe Lebensdauer

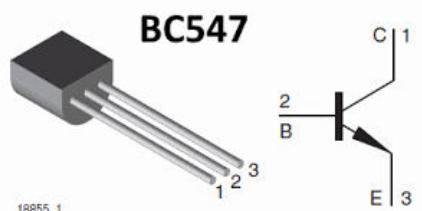
Elektrisches Relais



1950er bis 1960er Jahren eingesetzt

Gross, langsam

Transistor



1950 in tragbaren Radios eingesetzt, ab 1960 in ICs verbaut

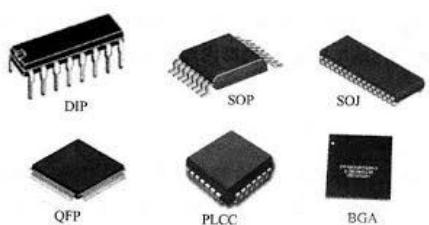
Wurden mit der Zeit immer kleiner

Aktuell bis unter 5nm gross

Vergleich: Menschliches Haar: 90'000nm

Ein Virus: 20nm

Integrated Circuit: IC



Ab den 1960 Jahren gebaut

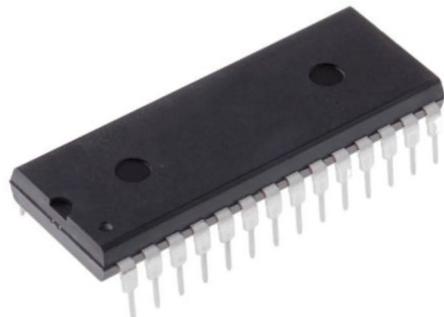
Viele Transistoren auf einem Chip

Speichertechnologie

Es gibt diverse Speichertechnologien.

SRAM - (Static RAM)

- 6 Transistoren pro Speicherzelle
- Für Cache und Register im Mikroprozessor
- Hohe Geschwindigkeit
- Kleinere Speicherkapazität pro Chip
- Datenhaltung nur während Spannungsversorgung



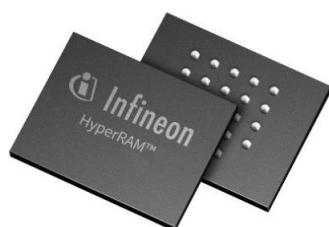
DRAM - (Dynamic RAM)

- 1 Transistor pro Speicherzelle
- Hohe Datendichte auf einer kleinen Chipfläche
- Langsamer, aber billiger pro Bit
- Muss alle 32 oder 64ms aufgefrischt werden
- Für Hauptspeicher



PSRAM - (Pseudo-static RAM)

- DRAM mit eingebauter Auffrischung
- Auch 1T-SRAM und cellular RAM (In Mobiltelefonen verwendet)



Flash Speicher

- Information auf Floating Gate gespeichert, nicht Kondensator
- BIOS, Firmware, USB, SSD
- Persistent (nicht flüchtig)
- Beschreiben langsamer

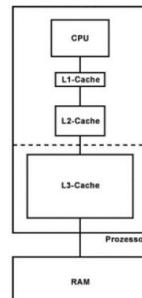


Beim DRAM muss regelmäßig der Zustand aufgefrischt werden, da der Kondensator langsam die elektrische Ladung verliert (und damit auch die gespeicherte Information). Deswegen ist ein Refresh der Zellen nötig, welcher vom Refresh Controller durchgeführt wird. Viele Mikroprozessoren und Mikrochips machen dies selbstständig oder es ist integriert im Memory Controller Chipsatz.

Es gibt noch folgende andere Speichertypen: Magnetspeicher (Festplatten wie HD), Optische Speicher (CD, DVD, Blue-ray Disc)

Cache-Speicher

Cache Speicher ist ein kleiner und schneller zwischenspeicher, der die Zugriffszeiten reduzieren soll. Dieser Speicher ist oft direkt in der CPU verbaut und es gibt meist 3 Cache Levels. Level 1 bis Level 3 Cache. Der Level 1 Cache ist dabei der schnellste und kleinste, während Level 3 Cache der grösste jedoch langsamste ist. Trotzdem ist der L3-Cache noch um einiges schneller als Daten aus dem RAM zu laden.



Neuere CPU Designs haben auch jeweils einen L1 Cache und L2 Cache pro Kern und einen geteilten gemeinsamen L3 Cache. Ältere CPU haben einen L1-, L2- und L3-Cache für alle Kerne.

Die Speicherhierarchie

Typische Zugriffszeit	Anwendung (Technologie)	Typische Kapazität (in Byte)
< 1 ns	Register (SRAM)	einige Byte bis kByte
2 ns	Cache (SRAM)	0,5 – 256 MB (Server: bis 768MB)
5-10 ns	Arbeitsspeicher (DRAM)	4 – 384 GB (Server: bis 2 TB)
HD: 3-10 ms SSD: 30µs	Dateisystem (HD, SSD)	HD: 73 GB – 22 TB (bis 30TB angekündigt) SSD: 120 GB – 30 TB
Band: 1 - 110 s Optische: 100ms	Archiv, Backup, ext. Datenträger (HD, Magnetband-Cartridge, opt. Datenträger)	HD: bis 22 TB oder Array Cartridge (z.B. LTO-9 bis 18 TB) Optische: 650 MB – 100 GB

Der Endian

Beschreibt die Art und Weise wie Informationen im Speicher abgelegt und gelesen werden. Der Endian ist die Reihenfolge, in der Bytes gespeichert werden.

Zum Beispiel: 123 (So im Speicher)

Als Big Endian kommen die höheren Stellen zuerst: 1 Hunderter - 2 Zehner - 3 Einer → 123

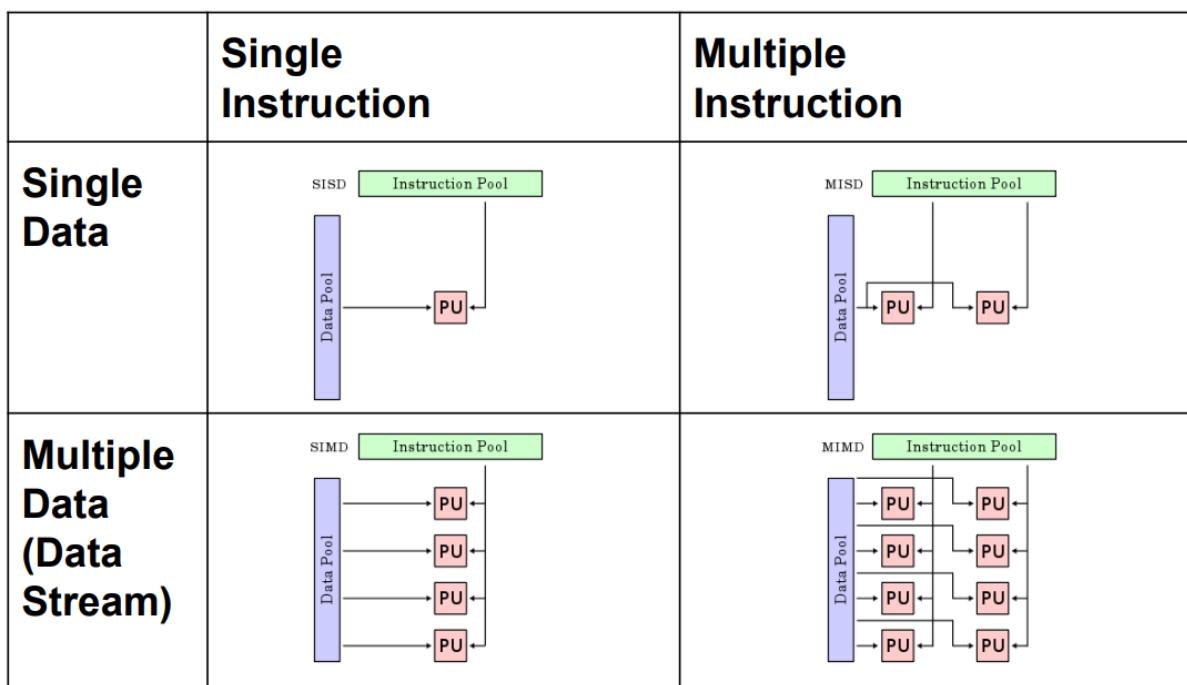
Beim Little Endian kommen die niederen Stellen zuerst: 1 Einer - 2 Zehner - 3 Hunderter → 321

Rechnerklassifikationen

Kategorien, in die Rechner eingeteilt werden können:

Typ, Bauart	Rechenleistung	Standort
Supercomputer	Supercomputer, Mainframe	Rechenzentrum: Server, Supercomputer, Mainframe
Mainframes	Workstation	
Minicomputer, Desktop	Server, PC, Notebook	Arbeitsplatz
Mobilgerät mit Akku	Tablet, Netbook	Auf dem Schoss: Notebook
	Pad, Smartphone	Handheld: Smartphone

Klassifikation nach Flynn



SISD

- Klassische Einkernprozessor Rechner

MISD

- Eine Architektur von Grossrechner bzw. Supercomputern. Die Zuordnung von Systemen zu dieser Klasse ist schwierig, sie ist deshalb umstritten.

SIMD

- Auch bekannt als Array Prozessor oder Vektorprozessor, dienen der schnellen Ausführung gleichartiger Rechenoperationen auf mehrere gleichzeitig eintreffende oder zur Verfügung stehende Datenströme

MIMD

- Eine Architektur von Grossrechner bzw. Supercomputer. Sie führen gleichzeitig verschiedene Operationen auf verschiedenen Eingangsdatenströmen durch.

Kodierung

Wichtige 8-Bit und 16-Bit Kodierungen:

- ASCII (American Standard Code for Information Interchange)
- EBCDIC (Extended Binary Coded Decimal Interchange Code)
- ANSI (ISO 8859) 8 Bit-Zeichensätze, aufbauend auf ASCII
- Unicode (ISO 10646) v13.0 March 2020
Code für Schriftzeichen und Symbole sowie Textelemente aller bekannter Schriftkulturen und Zeichensysteme

RA.08 - Bestandteile eines PC

Motherboard

Motherboard (Hauptplatine)

- CPU mit Cache, Arbeitsspeicher (RAM), BIOS-Flash (Boot-Programm)
- Diverse Anschlüsse, Busse und Einschübe, EIDE, USB

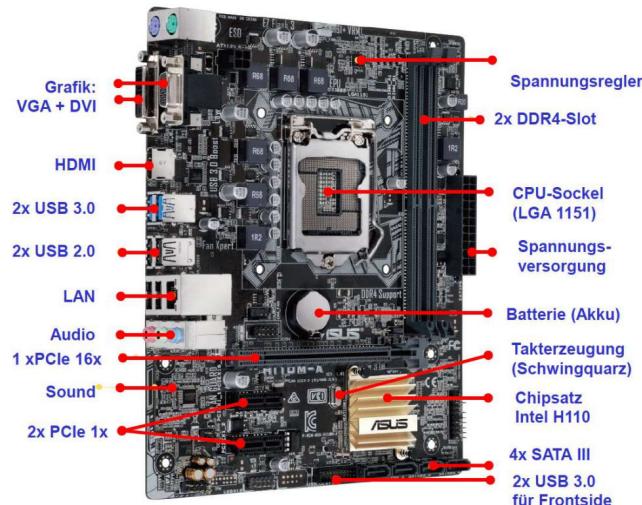
Laufwerke

- Festplatten (Harddisk, HD), Flash-Laufwerke (Solid State Drive, SSD)
- DVD Laufwerke / Brenner, Flash-Speicher (SD-Karten, Memory Stick, ...), Tape, etc.

Plug-in Karten

- Grafikkarten, Netzwerkkarten, SCSI-Kontroller, TV-Karten, Hardwarebeschleuniger

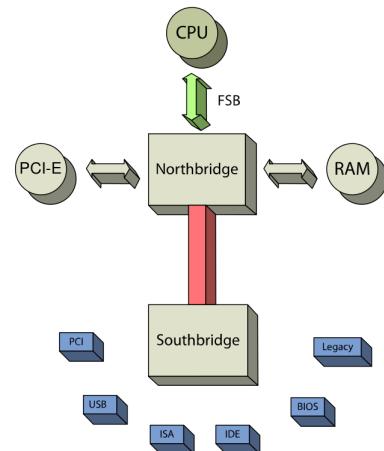
Netzteil und Gehäuse



Der PC-Chipsatz unterstützt den Prozessor bei seinen Aufgaben. Die klassische Aufteilung auf dem Mainboard ist in eine North-Bridge (neu: Memory Controller Hub, MCH) und eine South-Bridge (neu: I/O Controller Hub, IOCH). Wobei die North-Bridge für den Datenfluss zwischen CPU, Speicher und der South-Bridge regelt und die South-Bridge für die Ansteuerung der Peripherie, PCI, Festplatten und die externen Schnittstellen verantwortlich ist.

Aktuell geht der Trend immer mehr Richtung einem Chip → PCH (Plattform Controller Hub) oder (fast) alles direkt im Mikroprozessor.

*FSB = Front Side Bus



Das BIOS und UEFI-BIOS initialisieren die zum Booten des Betriebssystems notwendigen HW-Komponenten: CPU, RAM, Chipsatz mit PCIe-, SATA- und USB-Controller, Massenspeicher (HD oder SSD). Es ermittelt dann Bootdevices mit Bootloader oder Bootmanager. Dann kann das Betriebssystem geladen und gestartet werden.

Speicher-Busse und -Module

DRAM-Modul Typen

- EDO-RAM: Extended Data Output RAM
- SD-RAM: Sychnrones DRAM
- DDR-SD-RAM: Double data rate (typ: 0.5 - 2GB)
- DDR2: 4-fach Fetch (typ: 1-8 GB)
- DDR3: 8-fach Fetch (typ: 1-8GB)
- DDR4: 8-fach Fetch mit höherer Datenrate (Typ 4 - 16GB / Modul) bis zu 256GB / Modul
- DDR5: 16 - 32 fach Fetch, bis 512 GB / Modul

RAM-Modul Bauformen

- SIMM: Single Inline Memory Module
- DIMM: Dual Inline Memory Module
- SO-DIMM: Small Outline DIMM

PCI, PCIe & Schnittstellen

Der Peripheral Component Interconnect Bus wird für diverse Anschlüsse gebraucht.

Sonderformen für PCIe Busse sind die PEG = PCI Express for Graphic oder auch M2 Steckplätze, als kompakte Steckerfassung für SSD Speicher, welche bis zu 4 PCIe-Lanes haben.

PCI:



PCIe:



Es gibt einige serielle Anschlüsse:

- USB in diversen Versionen
 - USB 2.0 bis 480MBit/s
 - USB 3.0 bis 5 Gbit/s → USB A mit mehr Kontakten
 - USB 3.1 bis 10 Gbit/s → und dem neuen USB C Stecker
 - USB 3.2 bis 20 Gbit/s
 - USB 4 bis 40 Gbit/s (Ende 2020)
- Thunderbolt
 - Ähnlich wie DisplayPort + PCIe von Intel und Apple entwickelt
- Serial ATA (S-ATA)
 - Rev 2 / SATA II 2.4 Gbit/s
 - Rev 3 / SATA II 4.8 Gbit/S oft mit den blauen Buchsen
- SAS (Serial Attached SCSI)
 - 6 Gbit/s
 - 12 Gbit/s

Wireless

WLAN (IEEE 802.11 a/b/g/n/ac/ax)

2.45 und 5 GHz mit 11/54/150 Mbit/s (ac bis 3Gbit/s; ax: bis 11Gbit/s)

Bluetooth

723.2 kbit/s

IrDA

z.B Very Fast Infrared - 4 Mbit/s

Grafikkarten

Grafikkarten werden über den PCI-Express angeschlossen. Die gängigsten Hersteller sind AMD, Nvidia und Intel.

RAID - Redundant Array of Independent Disks

Bei einem Raid System werden Daten auf mehrere Festplatten aufgeteilt. Der Hauptvorteil von RAIDs ist die Redundanz und die damit verbundene Sicherheit der Daten. Da die Daten doppelt gespeichert werden, können diese bei einem Ausfall wiederhergestellt werden. Auch sind mithilfe von RAIDs grosse logische Laufwerke möglich.

RAID 0 Stripping - Beschleunigung ohne Redundanz

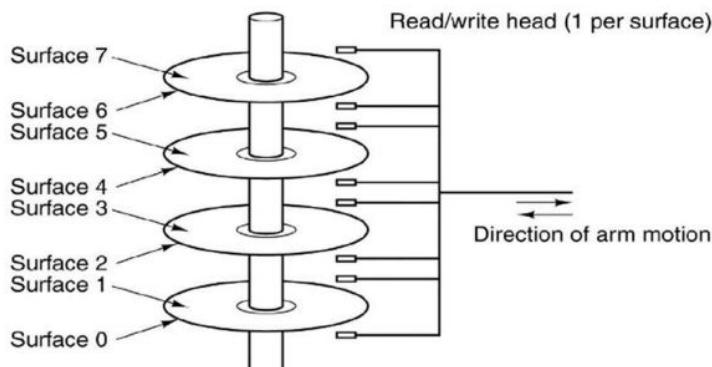
RAID 1 Mirroring - Spiegelung

RAID 5 - Leistung + Parität, Block Level Stripping mit verteilten Paritätsformen (Standard für Server mit hoher Fehlertoleranz)

RAID 10 - Kombination aus RAID 1 und RAID 0 für Beschleunigung + Sicherheit

Peripheriegeräte

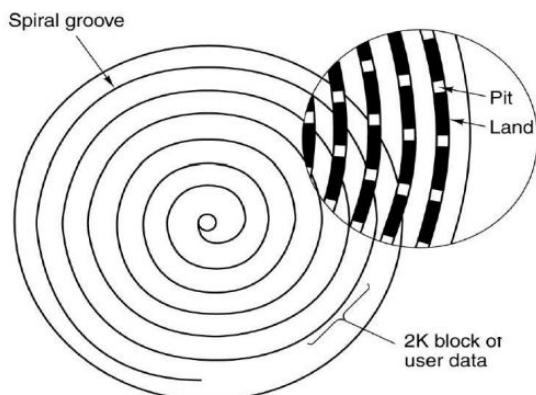
Festplatten



Typische Daten:

- Baugröße: 5.25 / 3.5 / 2.5 und 1.8 Zoll
- Scheiben sind 1 bis 12 verbaut typischerweise 1 - 4
- Luftgefüllt oder mit Helium
- Kapazität 250 GB bis 18 TB
- Cache: 2 ... 64 ... 256 MB

CD-ROM, DVD-ROM



Der Hauptunterschied zur HD ist das die Informationen optisch und nicht magnetisch auf die Disk gespeichert werden.

Die CD besteht aus einem Layer und die DVD aus einem double Layer.

Blue Rays werden mit blauen Lasern beschrieben, daher auch der Name.

Display Technologie

- LCD: passiv
- Twisted TFT (Thin Film Transistor) aktiv, 1 Transistor pro Farbpixel
- OLED, aktives Matrix Display aus organischen LED

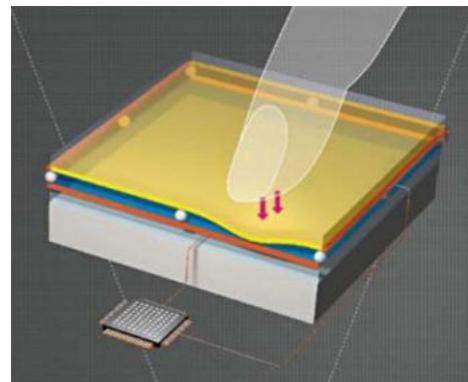
Anschlüsse:

- VGA
- DVI
- HDMI
- DisplayPort
- MiniDP (Apple)

Touchscreens

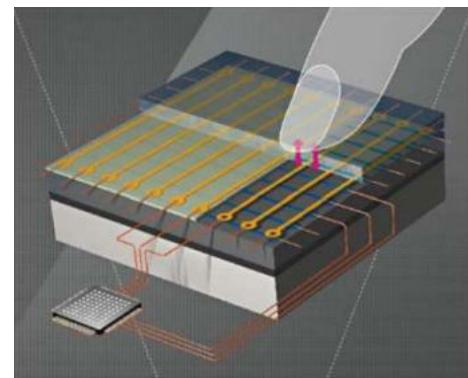
Resistive Touchscreen

- Zwei parallele Leiterschichten, durch Abstandhalter getrennt
- Durch Drücken kommen sie an einem Punkt in Berührung
- Elektronik errechnet Fingerposition
- Vorteil: mit Handschuhen bedienbar
- Lassen sich nicht mit dickem Glas schützen, da die Oberfläche nachgeben muss. Oft eine beschränkte Haltbarkeit



Kapazitiver Touchscreen

- Finger verändert das elektronische Feld
- Elektronik errechnet Fingerposition
- Können mehrere gleichzeitige Berührungen lokalisieren
- Lassen sich durch ein dicker Deckglas schützen



RA.01-RA.08 Formeln & Berechnungen

Speichergrösse

Speichergrösse = Speicherbausteine · Anz. Adressen pro Baustein · Baustein-Datenbreite

$$\text{Anz. Adressen pro Baustein} = 2^{\text{Adressbus-Breite}}$$

Erstes Beispiel

Ein Speichermodul besteht aus 8 Speicherbausteinen. Der Adressbus hat eine Breite von 16 bit und ist an jeden Baustein geführt. Zum Steuerbus gehören unter anderem 8 Chip-Select-Signale, die jeweils an einen der Bausteine geführt sind. Der Datenbus hat eine Breite von 8 bit. Wie gross ist die Speicherkapazität des Moduls?

$$K = 8 \cdot 2^{16} \cdot 8 = 4'194'304 = 4 \text{ Mbit}$$

Zweites Beispiel

Ein äusserlich ähnliches Modul wie in der Aufgabe oberhalb hat keine Chip-Select-Signale, der Adressbus ist identisch. Die 8 Speicherbausteine haben jeweils 4 bit als Datenschnittstelle. Wie breit ist der Datenbus und wie gross ist die Speicherkapazität?

$$K = 8 \cdot 2^{16} \cdot 4 = 2 \text{ Mbit}$$

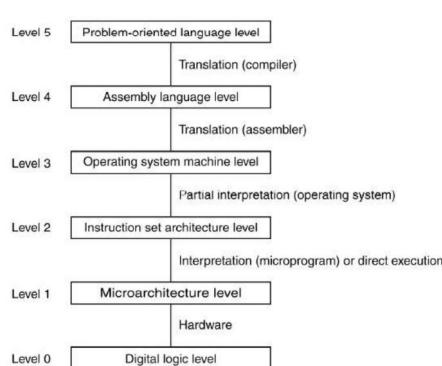
Boolesche Operationen mit Variablen

$$\text{Anz. Funktionen} = 2^{2^n}$$

BS.01 - Aufgaben eines Betriebssystems

Das Betriebssystem ist eine Software-Schicht zwischen den Anwendungen und der Hardware. Es verwaltet die Hardwarekomponenten und stellt einfachere Schnittstellen bereit. Das Betriebssystem läuft im Kernelmode, wodurch dieses mehr Rechte auf die Hardware hat. Anwendungen laufen im Usermode und haben somit beschränkt Kontrolle über die Hardware. Dies ist zum Schutz des Computers.

Ein Betriebssystem ist in mehrere Levels aufgeteilt, wobei jedes Level gewisse Aufgaben erfüllt.



Wenn wir dies von oben anhand einer C++ Anwendung anschauen, wird diese zuerst (level 5) compiled, damit sie dann (level 4) in Maschinensprache umgewandelt werden kann. Dann (level 3) trifft das Betriebssystem Vorkehrungen die nötig sind, um diesen Maschinencode (auf level 2) auszuführen. Nun wird dies (level 1) Hardware zugewiesen und die logischen Operationen (level 0) setzen die Befehle schlussendlich um.

Das Betriebssystem kontrolliert wie viele Ressourcen welche Anwendungen zur Verfügung stehen. Dieses Zuteilen von Ressourcen nennt man Multiplexing.

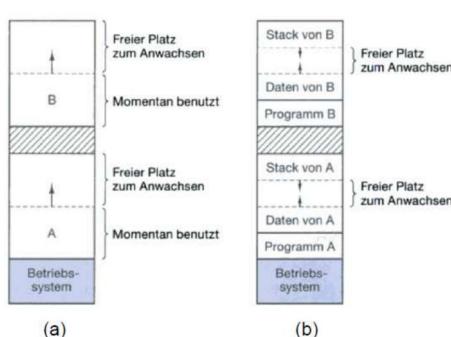
- Zeitlich (z.B.: CPU, Schnittstellen, Drucker, ...) Anhand von Prioritäten und Fairness
- Räumlich (z.B.: Hauptspeicher, Festplatte, ...) Anhand von Effizienz und Zugriffsschutz

Auch die Benutzer und Rechteverwaltung macht das Betriebssystem. Somit ist es ein Ressourcen-Manger

Programmabarbeitung

Das Betriebssystem entscheidet, wann und wie lange welche Anwendung auf der CPU ausgeführt wird. Das OS lädt das Programm in den Arbeitsspeicher und jedes Programm erhält zusätzlich einen Datenbereich im Arbeitsspeicher.

Bei Suspendierung eines Programms wird der gesamte Kontext (= Registerinhalt + Zeiger auf den Datenbereich des Programms) auf den Stack abgelegt und bei Wiederaufnahme frisch geladen.



Speicherreservierung für Programme A und B.

- a) für wachsende Datensegmente
b) für Stack- und Datensegmente

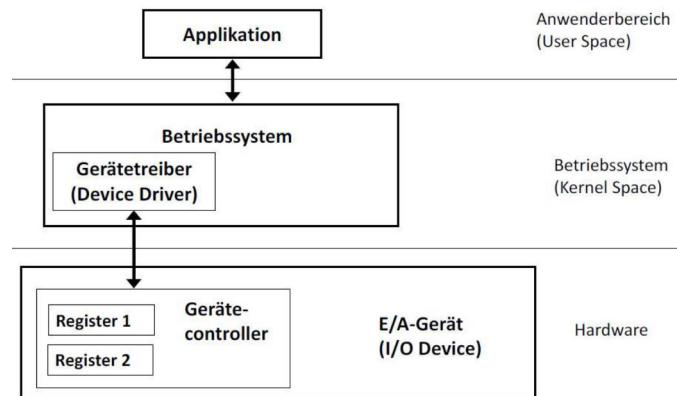
Stack und Daten von Programm A wachsen aufeinander zu. Somit hat man eine möglichst flexible Nutzung des freien Platzes. Stack und Daten dürfen sich maximal berühren, da es sonst zu Datenverlust kommt

Grafik aus "Moderne Betriebssysteme" von A. Tanenbaum

Interaktion mit I/O Geräten

Damit das Betriebssystem auf I/O Geräte zugreifen kann, werden Treiber (Driver) benötigt.

- Gerätetreiber sind betriebssystemabhängig und somit Software.
 - Statisch integriert im OS Kern
 - Werden über Tabellen eingebunden
 - Bei Plug & Play automatisch geladen
- Der Gerätecontroller (Hardware) verfügt über spezielle Register und stellt vereinfachte Schnittstellen zu den eigentlichen Geräten bereit, die durch den Gerätetreiber angesprochen werden.



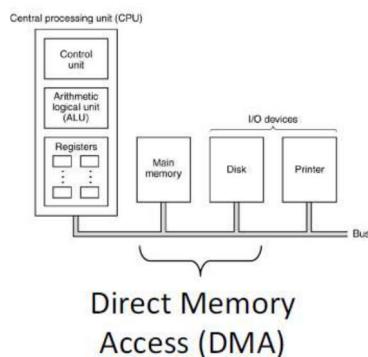
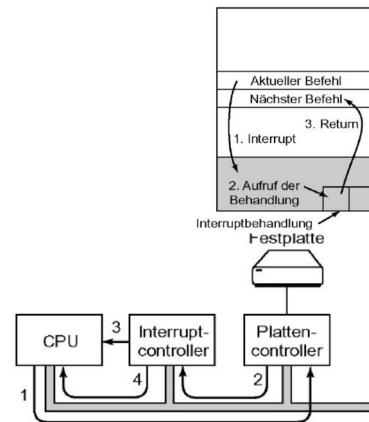
Es gibt allgemein zwei Varianten wie Interaktion mit den I/O-Geräten stattfinden kann.

- Polling (Busy Waiting)
 - Der System-Call einer Anwendung wird vom OS durch einen Prozedurauftrag direkt an den Treiber weitergeleitet.
 - Der Treiber initiiert dann die I/O Operation, liefert und empfängt Daten und wartet auf die Rückmeldung des Gerätes. Hierzu wird laufend der Gerätecontroller angefragt (Polling), dies solange bis die Geräte fertig ist.
 - Während dieser Wartezeit ist die CPU belegt.
 - Erst wenn der Prozess fertig ist geht die Kontrolle an die Anwendung zurück.

⇒ Diese Methode ist zu vermeiden

- Interrupts

- Der System-Call einer Anwendung wird vom OS durch eine Prozedurauftrag an den Treiber weitergeleitet.
- Der Treiber initiiert dann die I/O Operation, liefert und empfängt Daten und programmiert den Gerätecontroller so, dass dieser einen Interrupt auslöst sobald das Gerät wieder **Ready** ist.
- Der Treiber gibt die Kontrolle jetzt wieder zurück
- Die aufrufende Anwendung wird vom OS blockiert, denn diese muss ja auf das I/O Gerät warten.
- Andere Aufgaben (Prozesse) können bis zum Interrupt des Gerätecontrollers ausgeführt werden.
- Beim Interrupt suspendiert die CPU ihre momentan ausgeführte Aktivität und ruft den (im Gerätetreiber enthaltenen) Interrupthandler auf der dann z.B. Daten vom Gerätecontroller übernimmt.



Ein spezieller Controller ist der Direct Memory Access Controller (DMA). Dieser übernimmt und liefert Daten direkt zwischen dem Gerätecontroller und dem Hauptspeicher. Die CPU teilt vorher dem DMA-Controller die erforderlichen Informationen mit (Wieviel von Wo nach Wo). Dies ist eine partielle Entlastung der CPU und ein Interrupt gibt es erst nach Abschluss des Transfers. So ein DMAController ist z.B. im Chipsatz integriert.

Initialisierung von Geräten (Rechnerkonfiguration & BIOS)

Das Betriebssystem benötigt Informationen zu Interrupt-Nummer, Interruptlevel (Priorität) und I/O Adresse. Früher wurde dies statisch an den Gerätecontrollern festgelegt (mit DIP-Switches). Heute wird dies dynamisch «Plug and Play» über den PCIe-Bus zugewiesen. Hilfsmittel dazu sind BIOS und das UEFI-BIOS.

Das BIOS (Basic Input Output System) besitzt ein Programm auf sehr niedriger Stufe, üblicherweise im Flash-Speicher, welches beim Einschalten des Rechners gestartet wird. Es werden damit alle angeschlossenen Geräte und statischen Adressbereiche abgefragt. Auch die dynamischen zugewiesenen Adressbereiche werden an den Gerätecontroller übermittelt.

Shells

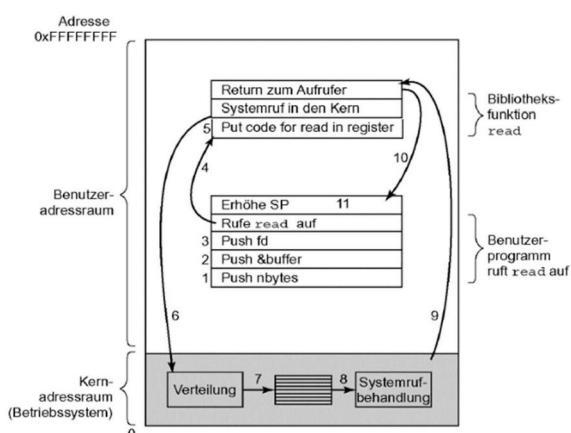
Betriebssysteme haben meist einen Kommandointerpreter → CMD, Terminal. Dieser erlaubt es uns OS-Befehle auszuführen. Dies nennt man Shell. Bei Betriebssystemen mit einem GUI werden diese Kommandos durch Mouse-Clicks auf GUI Elemente ausgeführt.

Systemaufrufe

Anwendungen verwenden Betriebssystemfunktionen über Systemaufrufe (System-Calls). Diese Systemaufrufe sind meistens in Bibliotheksfunktionen eingebettet. Die Parameterübergabe erfolgt über den Stack.

Beispiel: Das Benutzerprogramm möchte Daten aus einer Datei lesen und ruft dazu die OS-Funktion

`«read» auf. Count = read(fd, $buffer, nbytes)`



1 - 4: Dass Benutzerprogramm legt die Parameter nbytes, &buffer und fd auf den Stack und ruft «read» auf.

5 - 6: Im User-Space befindet sich eine Bibliothek mit Kopffunktionen. Diese legt die Argumente in Register ab und führt einen Trap (Interrupt) aus. Damit geht die Ausführung an das OS.

7 - 8: Im OS wird in einer Tabelle zur gewünschten Funktion verzweigt.

9 - 1: Rücksprung via Kopffunktion zum Benutzerprogramm, das dann noch den Stack aufräumt.

Was gehört zum Betriebssystem (OS)

Die Grenzen zwischen Anwendungen und dem Betriebssystem verfließen und sind nicht immer ganz deutlich.

Bekannte Betriebssysteme:

- Windows
 - (alte) Windows 9x (95, 98, ME), MS-DOS 3.1
 - Windows NT (New Technology, alle aktuellen Windows Systeme bauen darauf auf)
- macOS
 - früher selbstständiges OS
 - ab 2001 Mac OS X auf Unix basierend
- Unix
 - kommerzielles Produkt, Mehrplatz-, Multitasking-, Netzwerk-fähig
 - modularer Aufbau und hohe Portabilität
- Linux
 - Freie UNIX Implementierung
 - Open Source
 - Unterschiedlichste Distributionen (Debian, Ubuntu, CentOS, ...)

Es gibt auch noch andere Betriebssysteme spezifischer für gewisse Geräte entwickelt:

- Android, iOS → Smartphones
- Raspberry Pi OS (kleines Linux) → Raspberry Pi
- Contiki, RIOT → Internet of Things

BS.EL - Enterprise Lab

Das Enterprise Lab der HSLU ist ein eigenständiges Team der HSLU – Informatik, welches ein kleines Rechenzentrum betreut, wartet und erweitert. Es stellt Computing Ressourcen für die Lehre und Forschung bereit. Alles wird von Stundeten betrieben.

Login

<https://eportal.enterpriselab.ch/#/home>

BS.02 - Prozesse

Multitasking

Ein Rechner macht vieles gleichzeitig und soll nie durch einen Task blockiert werden.

Die Standardlösung für Multitasking ist es, dass das Betriebssystem mehrere Tasks gleichzeitig ausführt. Dies kann jedoch nicht wirklich gleichzeitig ausgeführt werden, da die CPU Kerne ja nur jeweils einen Task erledigen können. So werden die Tasks also oft immer kurz nacheinander abgespielt. So entsteht eine Pseudo-Parallelität.

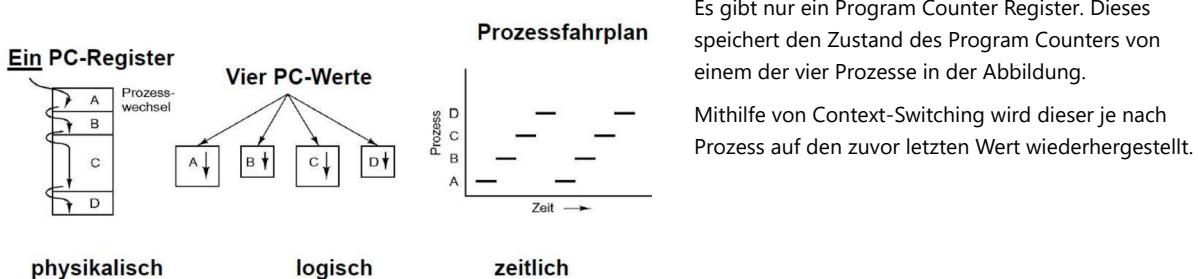
Multitasking bedeutet jedoch nicht Multiuser (Mehrbenutzersystem, ein OS kann mehrerer Benutzer verwalten) oder Multisession (ein OS kann gleichzeitig verschiedenen Benutzer eigenen Arbeitsumgebungen zur Verfügung stellen).

Prozesse

Ein Prozess ist ein laufendes Programm mit Umgebung. Somit ist ein Prozess eine Abstraktion eines laufenden Programms. Durch Context-Switching kann zwischen Prozessen hin- und hergewechselt werden. Der Schedulingalgorithmus entscheidet, wann welcher Prozess wie lange bearbeitet wird. Sobald auf einem Rechner mehr als ein Prozess ausgeführt wird, wissen wir nicht mit Bestimmtheit wann welcher Prozess ausgeführt wird. Bei jedem Durchlauf könnten die Unterbrechungen an anderen Stellen geschehen.

→ Prozesse sind Träger der Aktivitäten im Programmsystem

→ Sie dienen der dynamischen Strukturierung der zu lösenden Aufgaben



Im Prozessfahrplan wird veranschaulicht, wie die Prozesse zeitlich bearbeitet werden könnten.

Prozesse können ganz unterschiedlich entstehen:

- Bei der Systeminitialisierung (Foreground: Benutzerdienste, Background: Systemdienste)
- Beim Ablauf bereits bestehender Prozesse
- Können explizit durch den Benutzer generiert werden (Starten eines Programms)
- Als Batch-Job zu bestimmten Zeiten (Scheduled Tasks)

Prozesse können auch unterschiedlich beendet werden:

- Beabsichtigte Beendigung durch Abarbeitung der Aufgabe (Normal)
- (Beabsichtigte) Beendigung durch Applikationsfehler
- Unfreiwillige Beendigung aufgrund eines (Ausführungs-)Fehler
- Unfreiwillige Beendigung aufgrund eines anderen Prozess

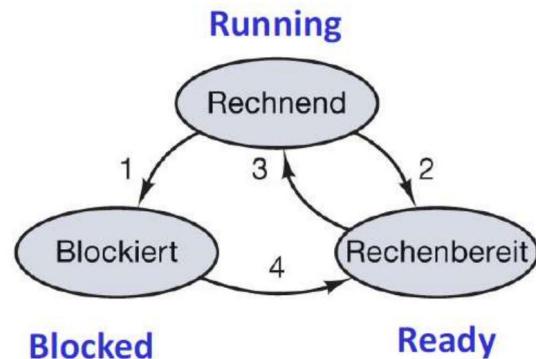
Prozesse haben Zustände, die sich in Phasen ändern, da sie ja nur phasenweise von der CPU bearbeitet werden. Prozesse tauschen Daten aus und kommunizieren miteinander, wodurch es durchaus sein kann, dass Prozesse aufeinander warten müssen.

Die 3 Prozesszustände

- **Rechnend (running):** In Ausführung auf der CPU
- **Rechenbereit (ready):** Temporär suspendiert
- **Blockiert (blocked):** Wartend auf externes Ereignis

Zwischen den Zuständen kann wie folgt gewechselt werden:

1. Prozess blockiert, da er z.B. auf eine Eingabe warten muss
2. Prozess wird suspendiert, da Scheduler einen anderen Prozess gewählt hat. Sein Context (PCB) wird gespeichert
3. Prozess wird vom Scheduler aktiviert, PCB → Prozescontrolblock (Context), wird geladen
4. Der Blockierungsgrund ist beendet



Scheduling

Ein Rechnersystem arbeitet normalerweise in einem der drei Modi:

▼ Stapelverarbeitung (Batch-Job)

Der Rechner erledigt fest definierte Aufgaben ohne Benutzerinteraktionen. Die Aufgaben und Daten liegen vor dem Start bereit. Die Stapelverarbeitung kann durch ein Kommando manuell gestartet werden oder auch auf eine bestimmte Zeit festgelegt werden.

▼ Interaktives System (Dialogbetrieb)

Der Anwender greift über eine Schnittstelle (Shell, GUI) auf den Rechner zu und erwartet kurze Antwortzeiten

▼ Echtzeitssystem (Realtime-System)

Es müssen Echtzeitanforderungen erfüllt werden. Heißt es müssen in bestimmten Zeiten antworten vorliegen.

Je nach Betriebsmodi muss das Betriebssystem das Scheduling auf eine andere Art gestalten. Bewerben sich mehrere Prozesse für die CPU so muss der Scheduler entscheiden, welchen er zuerst nimmt.

▼ Nonpreemptive (kooperatives) Scheduling:

Prozesse werden vom Betriebssystem nicht vorzeitig unterbrochen. Die Prozesse laufen so lange bis sie den Aktivzustand freiwillig verlassen und auf das nächste Ereignis warten oder sich selbst beenden.

▼ Preemptive Scheduling: (zeitscheibenverfahren)

Diese Strategie erlaubt es einer zentralen Instanz, einem laufenden Prozess zu (nahezu) jedem Zeitpunkt den Prozessor zu entziehen und einem anderen Prozess zuzuteilen. Die temporäre Unterbrechung des Prozesses ist für diesen transparent.

Nur im Batch-Betrieb macht das Nonpreemptive Scheduling Sinn. Bei interaktiven Systemen wird immer mit dem Preemptive Scheduling gearbeitet. Dazu braucht es einen Interrupt, der von einem Timer gesteuert wird.

Scheduling-Algorithmen:

- SJF: Shortest Job First
- FCFS: First Come First Serve → FIFO
- Round Robin: Prozesse werden nach der Reihe jeweils eine gewisse Zeit ausgeführt
- Prioritäts-Verfahren: Windows kennt z.B 32 Prioritätsstufen
- Earliest Due Date: Bei Echtzeitanforderungen bei RTS eingesetzt
- Gemischtes Verfahren → Bei Mehrkern Prozessoren entsprechend aufwändiger

Threads

Traditionell gibt es für Prozesse einen Adressraum, einen Kontrollfluss und eine Menge von Ressourcen, die von Prozessschritten zusammen benutzt werden. Oft ist es jedoch wünschenswert mehrere Ausführungsstränge "parallel" auf dem gleichen Adressraum mit den gleichen Ressourcen auszuführen. Diese Ausführungsstränge nennt man Threads.

Ein Thread hat einen eigenen Program Counter, Register und Stack. Threads innerhalb Prozessen generieren zusammen Prozessschritte, die dann zusammenhängend sequentiell pro Thread auf der CPU abgearbeitet werden. Sie können nacheinander oder auch verzahnt (Round Robin Prinzip) ablaufen.

In vielen Anwendungen finden mehrere Aktivitäten quasi gleichzeitig statt und können blockieren.

Die Bündelung zusammenhängender, sequentieller Prozessschritte vereinfacht die Handhabung bei der Programmierung. Durch die Handhabung von Threads kann das Kontextswitching gespart werden und ist somit viel weniger aufwendig als bei den Prozessen. Das Context-Switching muss weniger gemacht werden, da man mithilfe der gruppierten Threads schon angibt, welche Ausführungsstränge miteinander erledigt werden müssen.

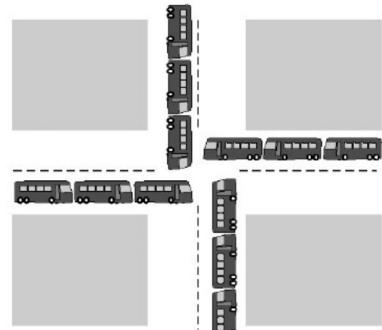
Beim Hyper-Threading wird ein Core so erweitert, dass er 2 Threads bearbeiten kann (sogar 2 Threads von 2 unterschiedlichen Prozessen). Es ist aber zu jedem Zeitpunkt maximal 1 Thread in Ausführung. Dafür wird ein komplexes Steuerwerk benötigt und Teile der Architektur sind doppelt vorhanden (mehrere vollständige Registersätze). Pipeline (ALU, FPU, ..) ist nur einmal vorhanden.

Prozesssynchronisation

Wenn zwei oder mehrere Prozesse gleichzeitig um die gleichen Ressourcen konkurrieren, kann es zu Deadlocks (Vorklemmungen) kommen. Das Betriebssystem soll Deadlocks verhindern, bzw. auflösen.

Die Kommunikation zwischen den Prozessen heißt IPC (Interprocess Communication). Das Ziel dieser Kommunikation ist es, einen strukturierten und konfliktfreien Ablauf der Prozesse zu ermöglichen (Prozesssynchronisation).

Der Hauptgrund dafür ist es die Deadlocks zu verhindern. Dies kann gemacht werden indem sich die Prozesse untereinander Nachrichten schicken. So verhindert man, dass sie sich gegenseitig in die Quere kommen. Prozesse die voneinander abhängig sind können so auch in der richtigen Reihenfolge abgearbeitet werden.



Sobald ein Prozess eine gemeinsam benutzte Ressource verwendet, wird diese Ressource auch kritische Region oder kritischer Abschnitt genannt.

Da die Abarbeitung von Prozessen nicht vorhersehbar ist, müssen absolut robuste Lösungen gefunden werden, die immer funktionieren. Eine scheinbare oder tatsächliche Reduzierung des Risikos reicht hier nicht aus. Daher gibt es diese Anforderungen:

- Keine Annahme über Geschwindigkeiten oder CPU-Anzahl
- Kein Prozess kann ausserhalb seiner kritischen Regionen andere Prozesse blockieren
- Kein Prozess sollte ewig warten müssen, bevor er in die kritische Region darf

Semaphore

Um zu verhindern, dass genau so ein Deadlock entsteht, gibt es Kontrollelemente. Das Semaphore ist eine Integer Variable auf die über 2 spezielle Operationen zugegriffen wird. Diese sind unteilbar, atomar und nicht unterbrechbar:

- Down-Operation: wait, acquire, P(s)
 - Falls Semaphore = 0, dann sleep
 - Erniedrige Semaphore falls > 0
- Up-Operation: release, V(s)
 - Erhöhe den Semaphore. OS weckt einen auf das Semaphore wartenenden Prozess wieder auf.

Das Semaphore ist somit wie eine Zählstation. Wenn das Semaphore auf 0 ist, können keine weiteren Prozesse mehr gestartet werden. Wenn das Semaphore über 0 ist, können je nach Wert des Semaphores Prozesse gestartet werden (Wert: 3 → 3 Prozesse können gestartet werden). Semaphore funktionieren also wie Ampeln, die eine Region kontrollieren. Erst wenn es wieder einen freien Platz gibt (Prozess verlässt kritische Region), kann ein weiterer Prozess den kritischen Abschnitt betreten.

Mutex

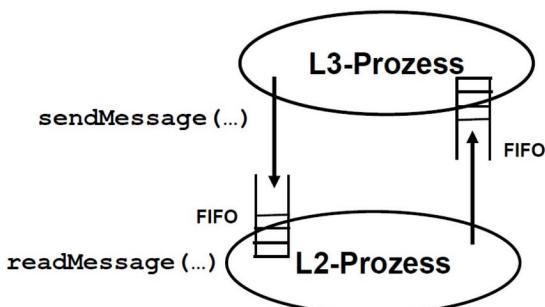
Mutex ist eine einfachere Version der Semaphore. Anstelle eines Integer Werts, gibt es hier nur **locked** oder **unlocked**. Wenn ein Prozess auf eine gemeinsame Ressource zugreift führt er die Operation **mutex_lock** aus. Wenn er die Ressource wieder freigibt führt er **mutex_unlock** aus. Der Prozess der den Zugriff nicht erhalten konnte, suspendiert sich selber.

Monitor

Der Monitor ist ein weiteres Kontrollelement. Da Semaphoren fehleranfällig sind, gibt es Lösungen für das Koordinierungsproblem auf höheren Ebenen. Der Monitor kontrolliert den Zugriff auf Prozeduren, Variablen und Datenstrukturen. Der Zugriff darauf ist nur über die von den programmierten definierten Programmteilen möglich. Zu jeder Zeit kann sich nur ein Prozess im Monitor aufhalten.

Message-Passing

Damit die verschiedenen Prozesse miteinander kommunizieren können, gibt es das Message Passing. Die Prozesse haben gleichzeitig keinen Zugriff auf die gemeinsamen Ressourcen, sondern übermitteln die Informationen direkt mit Message Passing.



Die drei klassischen IPC Probleme

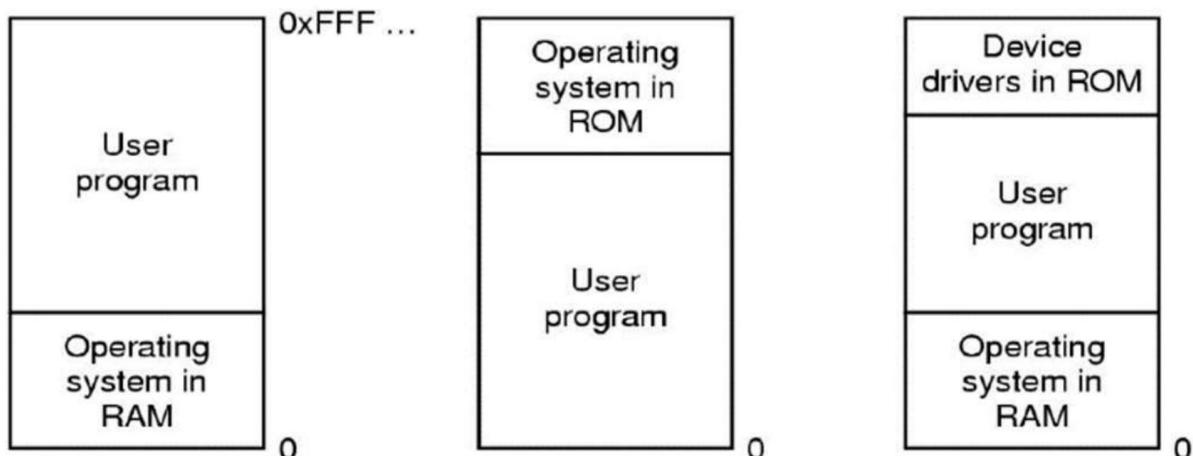
- Produzenten-Konsumenten-Problem (PCP)
 - Mehrere Prozesse greifen auf gemeinsame Daten zu, erzeugen bzw. verbrauchen Daten
- Dining Philosophers
 - Mehrere Prozesse benötigen explizit die gleichen Ressourcen
 - Wie kann man Fortschritt und Fairness sicherstellen
- Readers and Writers
 - Schreib-/Leseproblem auf einer Datenbank
 - Schrei-/Lesekonflikt verhindern

BS.03 - Speichermanagement, I/O, Dateisysteme

Speichermanagement

Es gibt unterschiedliche Wege um das memory zu organisieren.

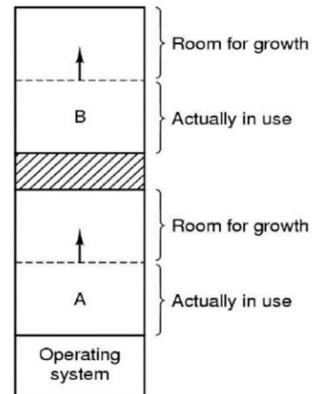
Für einen normalen Home-Rechner, wird das OS in das RAM geladen, um es für den Benutzer veränderbar zu machen. Für einen Rechner im Kioskmodus → Betriebssystem kann nicht verändert werden, wird das OS in das ROM geladen, da man ins ROM nicht schreiben, sondern nur lesen kann. Bei CNC-Maschinen, Druckmaschinen, oder sonstigen Maschinen bei der die Hardware fest hinterlegt ist, lädt man das OS ins RAM, jedoch die Device Drivers ins ROM.



Speicherzuordnung

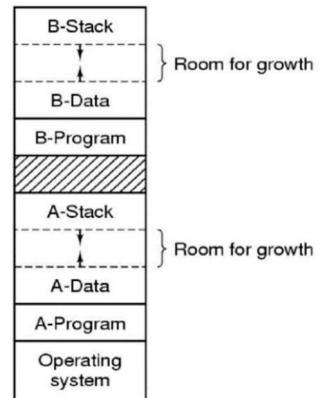
Allocation space for growing data segment

Dies bezieht sich auf die Zuordnung von Speicherplatz für den wachsenden Datenbereich. Der Datenbereich umfasst Variablen, Arrays, Strukturen und andere Datenelemente, die während der Laufzeit eines Programms erstellt und verwendet werden. Wenn das Betriebssystem Platz für den wachsenden Datenbereich zuweist, ermöglicht es dem Programm, zusätzlichen Speicherplatz für neue Datenobjekte bereitzustellen, wenn sie benötigt werden. Dies ist besonders wichtig, wenn Programme dynamisch Speicherplatz anfordern oder wenn sie während der Laufzeit Datenstrukturen erstellen, die größer werden können.



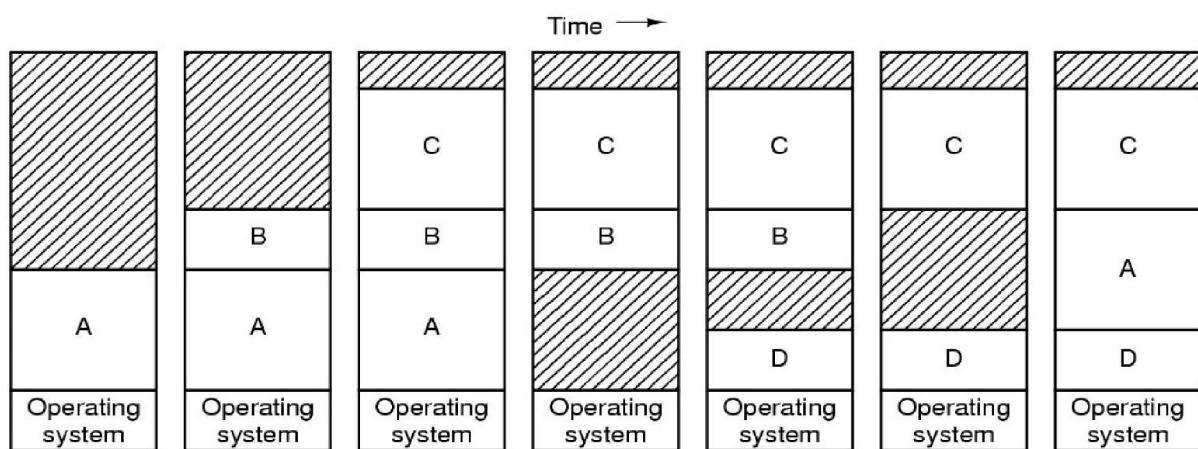
Allocation space for growing stack & data segment

Diese Art der Speicherzuordnung bezieht sich sowohl auf den wachsenden Stapel (Stack) als auch auf den wachsenden Datenbereich. Der Stapel ist ein Speicherbereich, der für die Verwaltung von Funktionsaufrufen und lokalen Variablen verwendet wird. Jedes Mal, wenn eine Funktion aufgerufen wird, werden Parameter, Rückgabeadressen und lokale Variablen auf den Stapel gelegt. Wenn das Betriebssystem Platz für den wachsenden Stapel und Datenbereich zuweist, ermöglicht es dem Programm, zusätzlichen Speicherplatz für sowohl den Stapel als auch den Datenbereich bereitzustellen, wenn sie wachsen.



Dynamic Memory Allocation

Die Memory allocation, wechselt wenn Prozesse ins Memory kommen oder das Memory verlassen



Garbage Collection, Defragmentierung

Mit der Zeit füllt sich der Speicher recht schnell. Die Garbage Collection räumt den Speicher automatisch auf. Es ist eine Funktion der Programmiersprache oder des OS. So wird während der Laufzeit der Speicherbedarf minimiert. Nicht mehr benötigte Bereiche werden identifiziert und freigegeben.

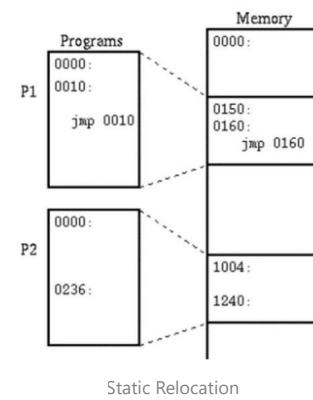
Mit der Zeit kann es vorkommen, dass der Speicher viele kleine freien Lücken hat. Um aus diesen wieder grössere Blöcke zu machen, gibt es die Defragmentierung. Diese "schiebt" die bestehenden Blöcke zusammen um grössere freie Blöcke zu erhalten.

Adress Relocation

Adressrelokation bezieht sich auf den Prozess, bei dem die Adressen von Objekten im Speicher angepasst werden, um ihre tatsächlichen Speicherpositionen widerzuspiegeln. Dadurch können Programme korrekt auf den Speicher zugreifen, unabhängig davon, wo sie im Speicher geladen wurden.

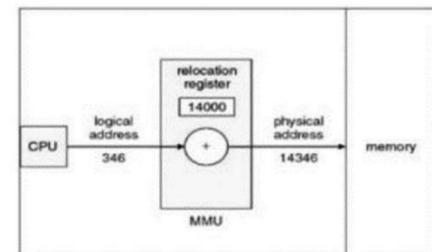
Statische Relocation (Verlagerung)

Statische Relokation erfolgt während des Kompilierungs- oder Linking-Prozesses. Bei der statischen Relokation werden die Adressen von Objekten im Programmcode anhand einer vordefinierten Basisadresse angepasst. Diese Basisadresse ist normalerweise die Adresse, an der das Programm im Speicher geladen wird. Durch diese Anpassungen werden die relativen Adressen in absolute Adressen umgewandelt, sodass das Programm ordnungsgemäß funktioniert, wenn es ausgeführt wird. Der Vorteil der statischen Relokation besteht darin, dass die Adressanpassungen einmalig während des Kompilierungs- oder Linking-Prozesses erfolgen und das Programm bei der Ausführung keine zusätzlichen Anpassungen benötigt. Der Nachteil ist jedoch, dass das Programm an eine feste Speicheradresse gebunden ist und möglicherweise nicht an verschiedenen Speicherorten ausgeführt werden kann.



Dynamische Relocation

Dynamische Relokation erfolgt zur Laufzeit des Programms, wenn es geladen und im Speicher platziert wird. Bei der dynamischen Relokation werden die Adressen von Objekten angepasst, nachdem das Programm im Speicher platziert wurde. Dies ermöglicht eine größere Flexibilität, da das Programm an verschiedenen Speicherorten ausgeführt werden kann, ohne dass Änderungen am Code erforderlich sind. Ein typisches Szenario für dynamische Relokation ist, wenn ein Betriebssystem ein ausführbares Programm in den Speicher lädt und dabei die Adressen der im Programm verwendeten Objekte an die tatsächlichen Speicherpositionen anpasst. Dynamische Relokation erfordert jedoch zusätzliche Zeit und Ressourcen, da die Adressanpassungen zur Laufzeit erfolgen müssen.



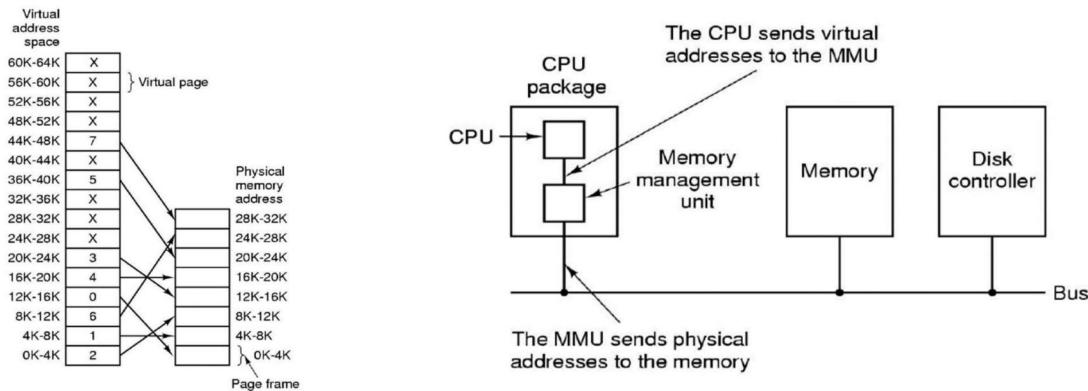
Dynamic Relocation

Zusammenfassend lässt sich sagen, dass statische Relokation während des Kompilierungs- oder Linking-Prozesses durchgeführt wird und das Programm an eine feste Speicheradresse bindet, während dynamische Relokation zur Laufzeit erfolgt und das Programm an verschiedene Speicherorte angepasst werden kann.

Virtual Memory

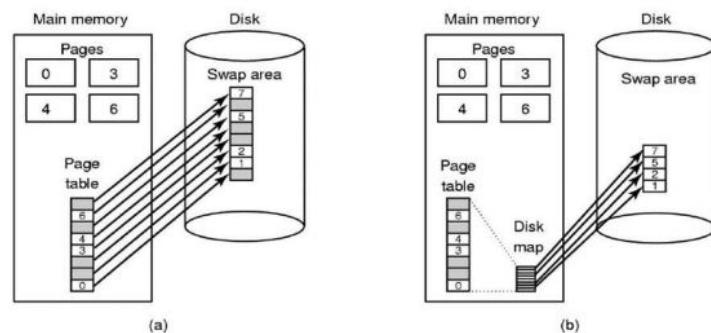
Das virtuelle Memory wird verwendet, wenn das physische Memory voll ist. Dabei wird auf der Festplatte Platz für Memory gemacht.

Über die Paging Table wird auf das virtuelle Memory verwiesen. Die MMU (Memory Management Unit) übernimmt das Umrechnen der virtuellen Adressen.



Auf der Harddisk wird das Ganze dann mittels Swapping angezeigt

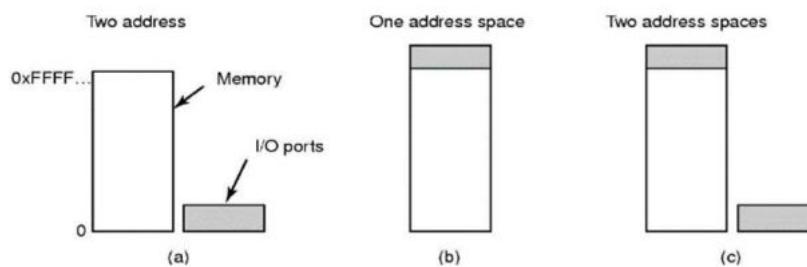
- a) Paging to static swap area
- b) Backing up pages dynamically



Ein- / Ausgabe (I/O)

Es gibt mehrere Möglichkeiten mit I/O Geräten zu kommunizieren. Dazu gibt es einen I/O Speicher, aus dem Daten gelesen werden können. Folgende drei Optionen gibt es:

1. Unabhängige I/O und Memoryspeicher
2. Das I/O ist im Memory gemappt
3. Eine Hybride Methode



Dateisysteme

Es gibt diverse Anforderungen an Dateisystem:

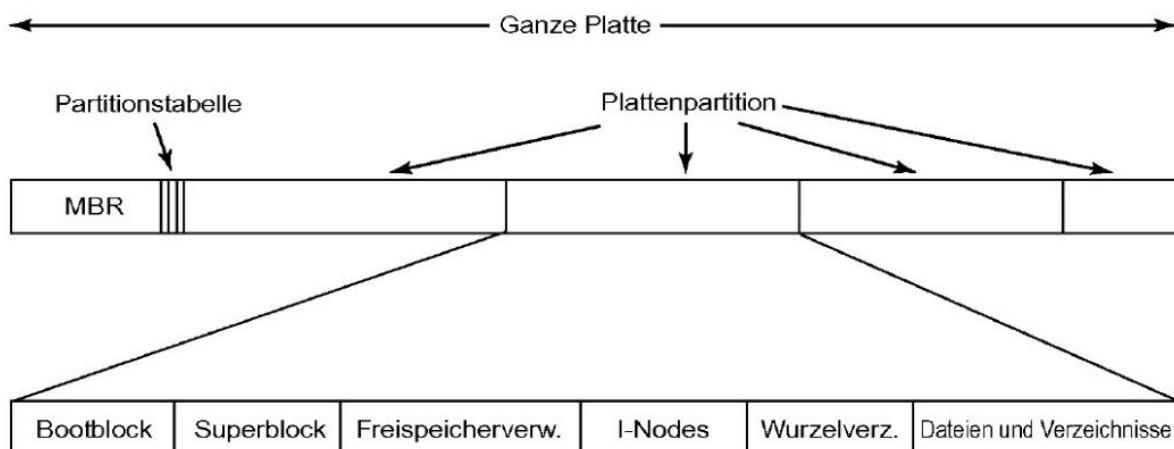
- Verwaltung grosser Datenmengen
 - Während ein Prozess läuft, kann er Informationen innerhalb seines Adressraumes speichern. Die Kapazität ist durch die Grösse des virtuellen Speichers begrenzt. Für viele Anwendungen reicht das aus.
- Persistenz
 - Die im Adressraum abgelegten Informationen gehen verloren, wenn der Prozess endet oder aus irgendwelchen Gründen abbricht. → Nicht persistent gespeichert.
- Paralleler Zugriff
 - Informationen die im Adressraum eines Prozesses abgelegt werden, sind nur für diesen Prozess verfügbar. Es gibt aber auch Informationen, auf die von mehreren Prozessen aus zum Teil auch gleichzeitig zugegriffen werden kann.

Dateien

- Daten auf Festplatten werden in persistenten (dauerhaften) Dateien (Files) verwaltet.
- Das Betriebssystem stellt über Systemaufrufe Dienste für Bannung, Strukturierung, Typisierung, Zugriff, Schutz usw. von Dateien im Dateisystem bereit.
- Zur Verwaltungsvereinfachung verwendet das Betriebssystem eine Struktur: Das Dateisystem (File System) mit Verzeichnissen
- Festlegungen für ein Dateisystem:
 - Länge und erlaubte Zeichen der Namen
 - Interner Aufbau, z.B. Sequenz von Bytes, Records

Filesystem - Implementierung

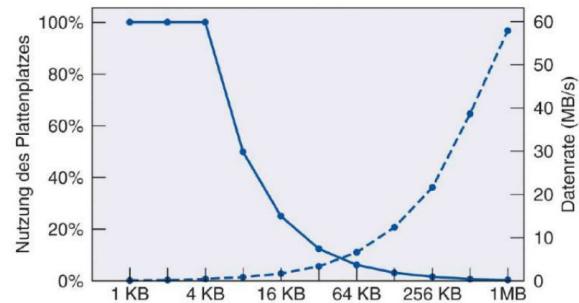
- Eine Festplatte kann in eine oder mehrere Partitionen aufgeteilt werden
- Im Master Boot Record (MBR, Sektor 0) der Festplatte befindet sich Code, der vom BIOS angestossen wird und der auf eine bestimmte Partition zugreift.
- Im Bootblock der Partition befindet sich der Code zum Starten des OS
- Im nachfolgenden Superblock befindet sich grundlegende Informationen zum Dateisystem.



Partitionen und Blöcke

Jede Partition wird in Blöcke (Cluster) aufgeteilt. Die Blockgrösse (2er Potenz) wird beim formatieren der Platte festgelegt.

- Kleine Blockgrösse: Gute Ausnutzung der Plattenkapazität, Längere Lese- und Schreibzeiten
- Große Blockgrösse: Schlechtere Ausnutzung der Plattenkapazität, Kürzere Lese- und Schreibzeiten



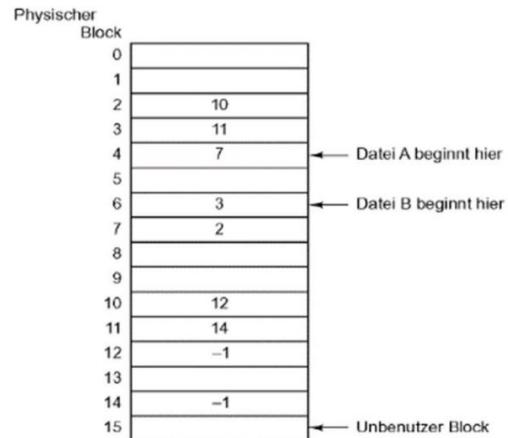
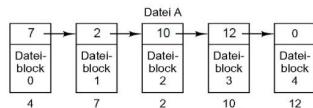
In den Festplattencache werden immer ganze Blöcke geladen / geschrieben. Das Ziel ist es daher die Informationen möglichst zusammenhängend abzuspeichern. Diese Blöcke werden durch das Dateisystem des OS, mithilfe von FAT oder I-Nodes, verwaltet.

Dateisysteme

FAT = File Allocation Table

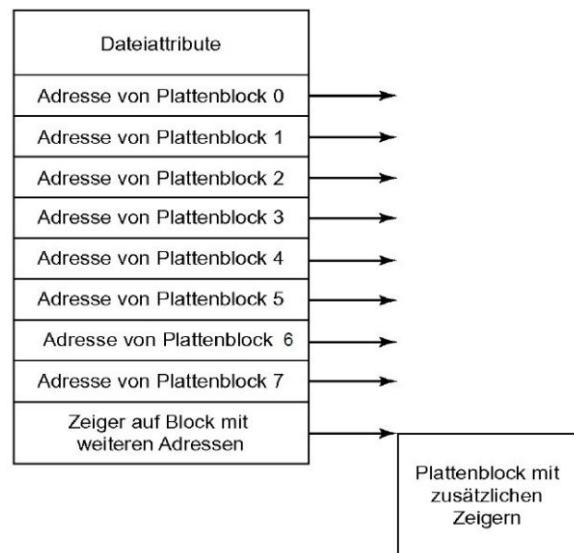
Hier gibt es eine Tabelle mit fester Grösse in der Dateisystemstruktur pro FAT-Partition. Ein Eintrag für jeden Clusterzustand.

Die einzelnen Datei-Teile sind dann als LinkedList abgelegt. So benötigt man zur Identifizierung nur den 1. Block und die Dateien können dynamisch wachsen.



I-Node Verwaltung (Unix, Linux, macOS)

Für jede Datei eine eigene I-Node-Struktur. Diese enthält Dateinamen und die Attribute. Nur die I-Nodes geöffneter Dateien sind im Hauptspeicher geladen. Diese Datenstruktur hat eine feste Länge und Platz für bis zu 12 Blöcken. Weiter Blockverweise sind dann auf einem Block auf der Festplatte.



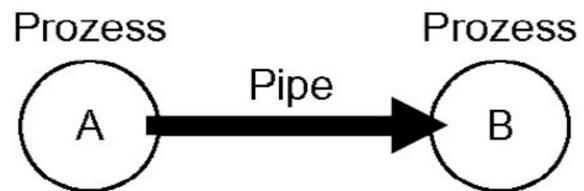
Zugriffsschutz

Es gibt einen Zugriffsschutz zwischen den einzelnen Benutzern, der Superuser (Root, Admin) darf mehr/ alles. Die einzelnen Prozesse haben in Mehrbenutzersystemen immer einen Besitzer. Dies wird für die Kreierung, Terminierung und die Sichtbarkeit benötigt. Dieser Schutz gilt auch für die Dateien des Filesystems:

- Protection Code mit Rechten für User/Groups/Other (jeweils 3Bits: rwx = Read/Write/Execute)
- Oder mit ACL (Access Control List) in der beliebige Rechtedefinitionen möglich sind.

Pipes

Eine Pipe ist ein Pseudofile, über das zwei Prozesse miteinander kommunizieren und entspricht einer Queue, auf die gleichzeitig lesend und schreibend zugegriffen werden kann (Ein Ringbuffer für beide Prozesse).



Konsistenz von Filesystemen

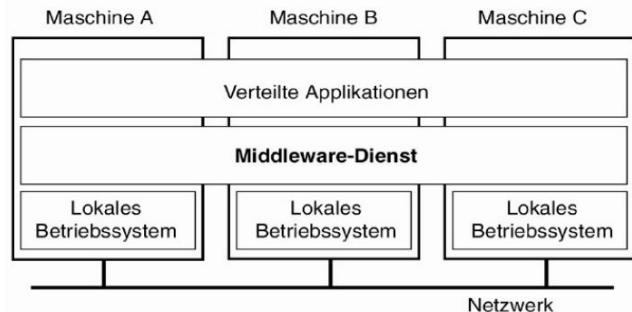
Nach einem Systemzusammenbruch kann das Filesystem in einem inkonsistenten Zustand sein, insbesondere, wenn Blocks für die Dateiverwaltung nicht korrekt auf die Festplatte geschrieben wurde. Der Konsistenztest erfolgt auf File und Blockebene:

- Bestimmte Blöcke sind weder Files zugeordnet noch frei.
- Ein Block tritt zweifach als frei auf
- Ein Block ist zwei verschiedenen Files zugeordnet

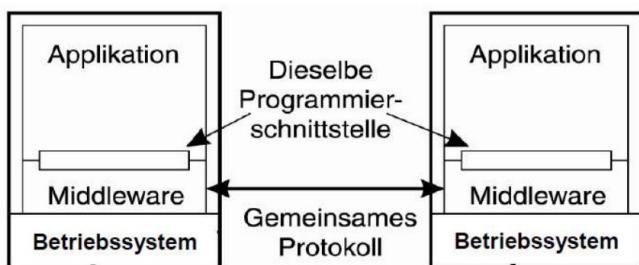
Um das Filesystem zu überprüfen gibt es diverse Tools scandisk, fsck, ...

BS.04 - Verteilte Systeme

Ein verteiltes System ist eine Verbindung von unabhängigen Rechnern, zur gemeinsamen Bearbeitung von Aufgaben. Die Rechner erscheinen dem Benutzer als einzelnes zusammenhängendes System. Für das wird eine zusätzliche Schicht auf die Betriebssysteme gelegt. Diese Schicht wird Middleware genannte.



Das Bereitstellen von IT-Leistungen über das Netzwerk kennen wir unter Cloud-Computing.

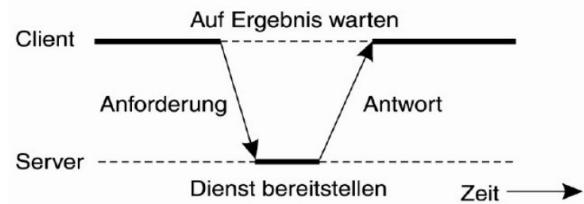


Diese Middleware muss alle notwendigen Schnittstellen vollständig anbieten damit es für alle Anwendungszwecke ausreicht. Darum muss auch auf dieser Ebenen ein standardisiertes Austauschprotokoll existieren.

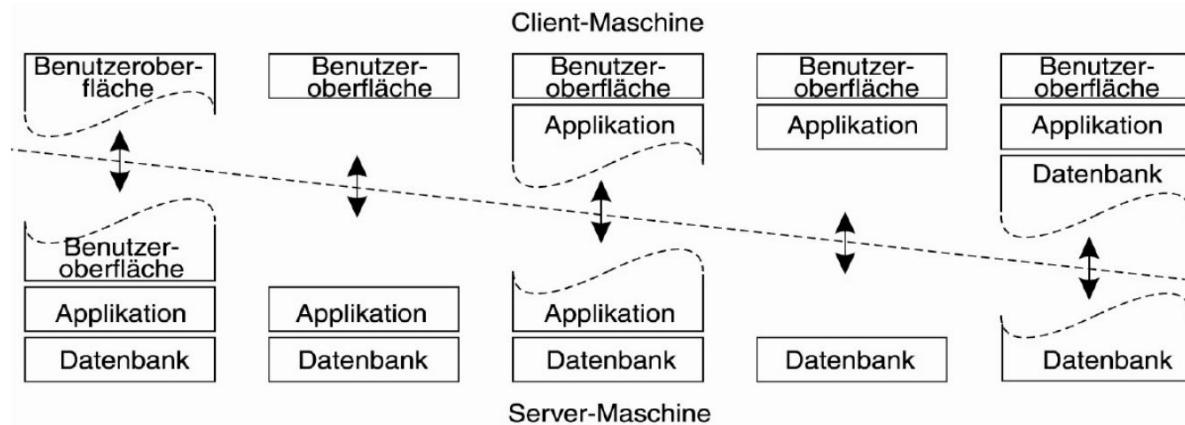
Das Client-Server Modell

In diesem Modell arbeiten zwei Rechner zusammen. Der Server bietet Dienstleistungen an, die vom Client in Anspruch genommen werden.

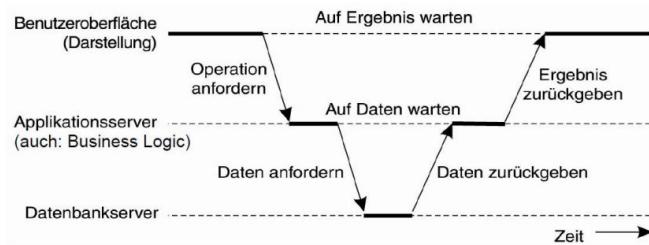
Das Ganze funktioniert in der Zwei-Stufen Architektur. Heisst das die Prozesse immer aufgeilt werden auf den Server und den Client:



- Links z.B ein Client welcher eine Webapplikation ausführt
- Rechts z.B ein Client welcher sogar die Datenbank ausführt



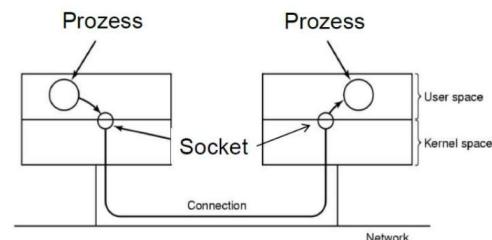
Bei der Drei-stufen Architektur funktioniert das System ganz ähnlich wie bei der Zweistufen Architektur ausser das noch ein weiterer Server dazwischengeschaltet ist. Möglicherweise ein Applikationsserver und ein Datenbankserver.



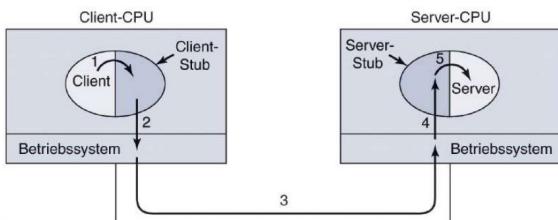
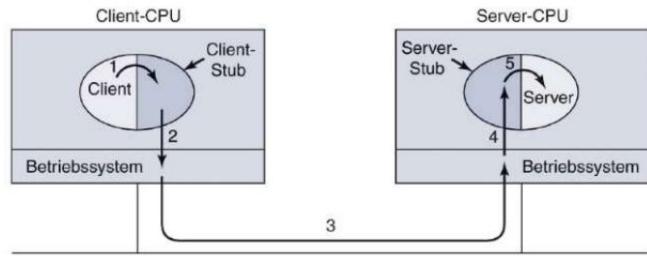
Entfernte Objekte (Remoting)

Ein Socket ist ein Kommunikationsendpunkt und somit eine Netzwerk-Schnittstelle von Anwendung zum OS. Dieser Endpunkt ist definiert durch die IP-Adresse und den Port. Die Prozesse greifen über den Socket auf das Netzwerk und darüber auf entfernte Prozesse zu. Darüber können dann die entfernten Prozesse kommunizieren.

Im Vergleich zu einem einfachen Request → Response Prinzip, funktioniert ein Socket beidseitig. Es ist wie ein offener Tunntel für Kommunikation zwischen zwei Systemen.



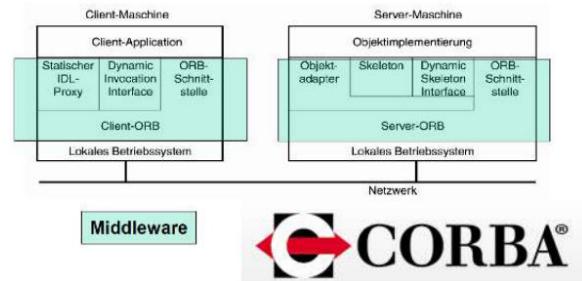
RPC Remote Procedure Call ist ein Aufruf auf eine Procedure auf entfernte Rechner. Ein Stub (Stumpf, Stummel) auf jeder Seite «regelt» die Übertragung auf dem Netzwerk. Funktioniert aber sonst wie ein lokaler Prozederaufruf. Transfer ist via UDP, falls die Datenmenge aber zu gross ist wird auch das TCP Protokoll verwendet.



Der **Object Request Broker (ORB)** ist eine Middleware-Komponente, die in verteilten Systemen verwendet wird, um die Kommunikation zwischen verschiedenen Objekten oder Komponenten zu ermöglichen.

Der **ORB** fungiert als Vermittler zwischen den Objekten, indem er die Anfragen, die von einem Objekt gesendet werden, empfängt, sie an das entsprechende Zielobjekt weiterleitet und die Ergebnisse zurück zum aufrufenden Objekt sendet. Er ermöglicht die Kommunikation zwischen Objekten, die in verschiedenen Programmiersprachen geschrieben wurden oder auf verschiedenen Betriebssystemen laufen. Er bietet eine transparente Schnittstelle für den Aufruf von Methoden und den Austausch von Daten zwischen den Objekten, unabhängig von ihrer tatsächlichen Implementierungsdetails.

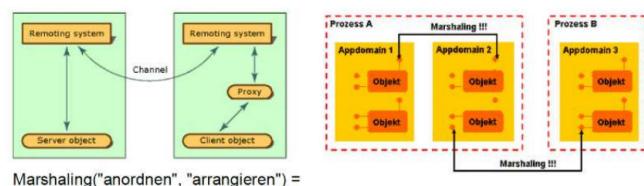
Der **ORB** basiert normalerweise auf dem **CORBA (Common Object Request Broker Architecture)**-Standard, der eine spezifizierte Schnittstelle und Kommunikationsprotokolle definiert. CORBA wurde entwickelt, um die Interoperabilität* zwischen verschiedenen Systemen und Plattformen zu ermöglichen und ist in verschiedenen Programmiersprachen verfügbar.



* Interoperabilität bezieht sich auf die Fähigkeit verschiedener Systeme oder Komponenten, zusammenzuarbeiten und effektiv miteinander zu kommunizieren.

Aktuellere Middleware-Lösungen sind oft schon in den Programmiersprachen integriert. Es gibt einzelne Bibliotheken etc. und gehören heute schon zum Standard.

- ◆ **Java RMI: Remote Method Invocation**
- ◆ **Java EJB: Enterprise Java Beans**
- ◆ **Microsoft: DCOM / .NET-Remoting / WCF**
- ◆ **ZeroC-Ice**
- ◆ **Service-Oriented Architecture (SOA)**
- ◆ **WEB Services**



Peer-to-Peer Netzwerke (P2P)

Ein verteiltes System aus (gleichberechtigen) Rechnern.

In einem P2P-Netzwerk sind alle Peer(Knoten) gleichberechtigt. Es kann daher ohne oder mit einem Master organisiert sein. Meistens ist aber kein Master vorhanden. Somit hat jeder Peer eine hohe Autonomie und organisiert sich selber.

Die Einfachste Form eines P2P-Netzwerk ist das File Sharing.

Zusätzliche Information (ein Beispiel für P2P):

In den Anfangszeiten, nutzte Spotify auch ein P2P-Netzwerk, um Songs schneller laden zu können. So war jeder Spotify Nutzer selbst ein Peer, der die Songdateien an nahe-liegende Peers weitergeben konnte. So konnte die Serverlast gedrosselt werden und Songs, die von nahen Peers bereits cached sind, konnten so schneller übertragen werden.

Cloud Deployment Models

Es gibt unterschiedliche Cloud Deployment Models.

Public Cloud

Die weitaus häufigste Form. Alle Nutzer teilen sich eine gemeinsame Cloud-Infrastruktur eines Anbieters. Das gilt für praktisch alle SaaS-, PaaS- und IaaS-Angebote (siehe weiter unten), Online-Speicher und App-Daten. Eine Public Cloud oder öffentliche Cloud ist über das Internet zugänglich.

Private Cloud

Ein Unternehmen betreibt eine eigene Cloud, entweder im eigenen Rechenzentrum am Standort (on-premis) oder bei einem spezialisierten Anbieter. Der Zugriff ist auf Benutzer aus dem Unternehmen beschränkt und erfolgt über abgesicherte Verbindungen. Private Clouds sind nicht allgemein über das Internet zugänglich. Diese Variante eignet sich für grössere Unternehmen, die ihre eigene, spezialisierte IT-Infrastruktur in die Cloud auslagern möchten und hohe Ansprüche an Datenschutz und -Sicherheit stellen.

Hybrid Cloud

Die Mischform aus obigen beiden Varianten ist in Grossunternehmen häufig anzutreffen. Dabei wird für Standardanwendungen wie beispielsweise E-Mail auf das Public-Cloud-Angebot eines Anbieters zurückgegriffen, während sensible Firmendaten und -Anwendungen in der privaten Cloud betrieben werden.

Cloud Service Models

IaaS

Infrastructure as a service.

Die grundlegende Ebene des Cloud-Computings ist IaaS, denn hier werden Hardware-Ressourcen in virtualisierter Form bereitgestellt. Ob Speicherplatz, Prozessoren oder Netzwerk – alle Recheninstanzen können in beliebiger Menge hinzugefügt und auch wieder entfernt werden. Manchmal wird deswegen auch von einem virtuellen Rechenzentrum (Virtual Data-Center) gesprochen.

PaaS

Plattform as a Service

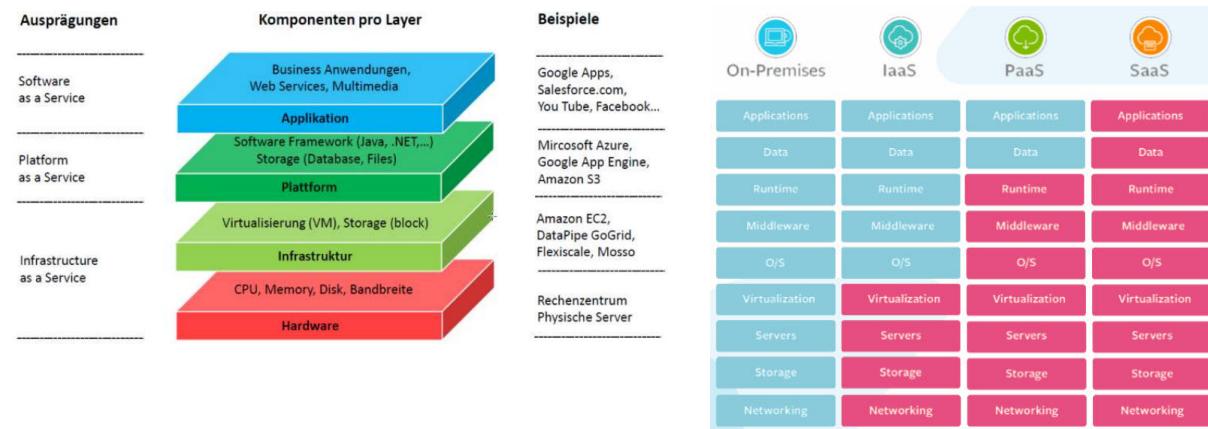
PaaS ist das Bindeglied zwischen IaaS und SaaS und ermöglicht erst das Zusammenspiel der beiden anderen Ebenen. Denn auf der Plattformebene werden die Entwicklungs- und Laufzeitumgebungen für Software bereitgestellt, aufbauend auf IaaS-Ressourcen wie etwa Betriebssystemen. Die anderen beiden Ebenen IaaS und SaaS werden in der Regel durch APIs angesprochen. Für PaaS interessieren sich demzufolge vor allem Software-Entwickler.

SaaS

Software as a Service

Bei SaaS werden Programme bedarfsabhängig bereitgestellt – und zwar meist direkt dem Endanwender. Die Nutzung erfolgt in der Regel über das Internet bzw. einen Webbrowser. Durch SaaS können die Anwender sich meist Lizenzgebühren sparen und müssen zudem nicht für Installation und Administration aufkommen. Beispiele dafür sind DropBox, Microsoft Office 365 oder Gmail.

Shared Responsibility



Vorteile und Nachteile der Cloud

Vorteile

- Niedrige IT-Investitionskosten
- Skalierbarkeit der Dienste
- Keine Wartung oder Upgrades nötig
- Keine Räumlichkeiten, Klima, Backup
- Standortunabhängig von Diensten
- Einfache Pilotprojekte / Testphase

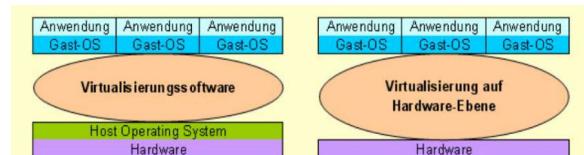
Nachteile

- Internet Anbindung i.d.R. muss immer Verfügbar sein
- Umgang mit sensiblen Daten
- Abhängig von Provider
- Verfügbarkeit der Dienste, Garantien und Ausfälle <99.95%
- Lange Verträge beeinflussen Kosten

Virtualisierung

Das Betriebssystem selber ist eine erweiterte oder virtuelle Maschine, die mächtiger ist als die eigentliche durch Hardware realisierte Maschine. In einem weiteren Schritt also, ist es auch möglich, eine weitere virtuelle Maschine auf unserer "virtuellen Maschine (hier Host)" zu betreiben.

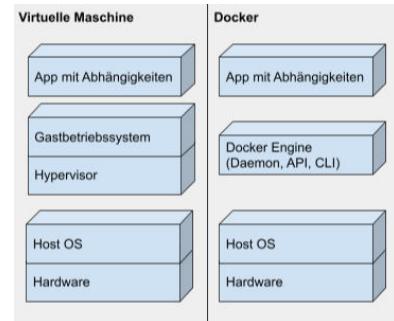
Der Hypervisor setzt entweder auf ein Host-OS auf oder direkt auf die Hardware.



Containerisierung

In dem Kontext von IT bezieht sich Containerisierung auf den Prozess des Verpackens einer Anwendung zusammen mit all ihren Abhängigkeiten und Bibliotheken in einen einzigen Container. Dieser Container kann dann auf jedem Computer oder Server ausgeführt werden, der die erforderliche Software zur Unterstützung hat, ohne dass weitere Konfigurationen oder Abhängigkeiten erforderlich sind.

Die Containerisierung ist ein wichtiges Werkzeug in der IT-Branche, das die effiziente Bereitstellung und Verwaltung von Anwendungen in einer Vielzahl von Umgebungen ermöglicht. Der Unterschied zwischen diesen zwei Virtualisierungsmöglichkeiten ist, dass bei einer Virtuellen Maschine der Hypervisor ein komplettes Betriebssystem als Gastsystem auf dem Host erlaubt. Dies bedeutet, dass bei jeder VM ein eigener Kernel mit eigenen Grundfunktionalitäten vorhanden ist, während bei einem Container der Kernel des Host Systems verwendet wird.



So also nutzen Container die Grundfunktionalitäten des Kernels des Host Systems. Das Betreiben von Applikationen in Containern benötigt weniger Ressourcen als das Betreiben von Applikationen in virtuellen Maschinen. So spart man also Ressourcen, wenn man Applikationen in Containern laufen lässt.

Vorteile beim Arbeiten mit Containern

1. Portabilität

Da alle erforderlichen Abhängigkeiten im Container enthalten sind, kann die Anwendung auf jeder kompatiblen Plattform ausgeführt werden, ohne dass zusätzliche einrichten oder konfigurieren erforderlich ist. Dadurch wird die Bereitstellung und Verwaltung von Anwendungen in Cloud-Umgebungen oder anderen verteilten Systemen erheblich erleichtert.

2. Effizienz

Container sind sehr effizient, da sie den Kernel des Host Systems verwenden. Durch diese Gabe, gibt es kein Betriebssystem, welches gebootet werden muss. Auch Lizenzierungskosten sinken, da man ja kein komplettes Betriebssystem auf dem Server braucht. Dadurch, dass die Container so viel leichter als VMs sind, können auch mehr Applikationen auf dem selben Server gehostet werden, da jede einzelne Applikation nicht mehr so viel Ressourcen benötigt.

3. Fehler Isolierung

Jeder Container agiert unabhängig von den Anderen. Wenn nun also ein Fehler in einer der Applikationen in einem Container auftaucht, beeinflusst dies die anderen Container (Applikationen) nicht.

4. Sicherheit

Durch die Trennung der Applikationen in einzelne Container, ist es für Angreifer viel schwerer, auf den Host selbst Angriffe auszuführen. Container kann man ausserdem auch zusätzlich noch Rechte geben oder Rechte wegnehmen, so dass aus einem Container gar nicht erst auf gewisse Ressourcen zugegriffen werden kann.

5. Einfaches Management

Bei der Verwendung von Containern, kann man die Container Orchestration verwenden, um sich das Management zu vereinfachen. Container Orchestration bezeichnet die Automatisierung eines Großteils des operativen Aufwands, der für die Ausführung von containerisierten Workloads und Diensten erforderlich ist. Damit kann ein Container Orchestration-Tool die Ressourcen von Containern dynamisch vergrössern und verkleinern.

6. Wiederholbarkeit und Automatisierung

Sie erstellen Code mit wiederholbarer Infrastruktur und Konfiguration. Dadurch wird der Entwicklungsprozess enorm beschleunigt. Ausserdem sind die Container-Images oft klein. Folglich können Sie neue Anwendungscontainer schnell ausliefern und die Bereitstellung verkürzen.

7. Testen, Zurückrollen und Bereitstellen

Container Umgebungen bleiben von Anfang bis Ende konsistent. Container-Images lassen sich leicht versionieren, so dass sie bei Bedarf leicht zurückgesetzt werden können. Wenn es ein Problem mit der aktuellen Iteration des Images gibt, können Sie einfach auf die ältere Version zurücksetzen.

8. Modularität

Containerisierung ermöglicht es Ihnen, eine Anwendung zu segmentieren, so dass Sie sie aktualisieren, bereinigen und reparieren können, ohne die gesamte Anwendung herunterfahren zu müssen.

Nachteile beim Arbeiten mit Containern

1. Container Ökosystem

Obwohl zum Beispiel der Kern der Docker-Plattform Open Source ist, funktionieren einige Container-Produkte nicht mit anderen - in der Regel aufgrund des Wettbewerbs zwischen den Unternehmen, die sie unterstützen.

2. Persistente Datenspeicherung

Alle Daten innerhalb eines Containers verschwinden für immer, wenn der Container heruntergefahren wird, es sei denn, man speichert sie vorher an einem anderen Ort. Es gibt zwar Möglichkeiten, Daten persistent zu speichern, z.B. Docker Data Volumes, aber dies ist wohl eine Herausforderung, die noch nicht nahtlos gelöst wurde.

3. Geschwindigkeit

Container verbrauchen Ressourcen effizienter als virtuelle Maschinen. Aber Container unterliegen immer noch einem Leistungs-Overhead aufgrund von Overlay-Netzwerken, Schnittstellen zwischen Containern und dem Host-System usw. Wenn Sie eine 100-prozentige Bare-Metal-Leistung wünschen, müssen Sie Bare-Metal verwenden, nicht Container.

4. Grafische Anwendungen bereiten noch Probleme

Containers wurde als Lösung für die Bereitstellung von Server Anwendungen entwickelt, die keine grafische Oberfläche benötigen. Es gibt zwar einige kreative Strategien (z. B. X11-Videowiederleitung), mit denen Sie eine GUI-Anwendung in einem Container ausführen können, aber diese Lösungen sind bestenfalls klobig.

5. Nutzen von Containern

Im Allgemeinen profitieren nur solche Anwendungen am meisten von Containern, die für die Ausführung als diskrete Microservices konzipiert sind. Ansonsten besteht der einzige wirkliche Vorteil von Containern darin, dass es die Bereitstellung von Anwendungen durch die Bereitstellung eines einfachen Paketierungsmechanismus vereinfachen kann.

BS01 - BS04 Berechnungen & Formeln

Zugriffszeiten berechnen

Der Anteil der Hits im Cache beträgt 95% wobei die Lesezeit 2ns ist.

Es bleiben 5% übrig, wobei wiederum 99% dieser, im DRAM landen, welches eine Zugriffszeit von 10ns hat. Die restlichen 1% gehen auf die HD, wobei die Zugriffszeit 10ms ist.

Zuerst vereinheitlichen wir die Einheiten zu Mikrosekunden, da dies in der Mitte aller Massen liegt.

$$95\% \rightarrow 0,002 \mu s$$

$$5\% * 99\% \rightarrow 0,010 \mu s$$

$$5\% * 1\% \rightarrow 10000 \mu s$$

Berechnung

$$t = 95\% \cdot 0.002 \mu s + 5\% \cdot 99\% \cdot 0.010 \mu s + 5\% \cdot 1\% \cdot 10000 \mu s = 5.002395 \mu s$$

$$t = 0.95 \cdot 0.002 + 0.05 \cdot 0.99 \cdot 0.010 + 0.05 \cdot 0.01 \cdot 10000 = 5.002395 \mu s$$

NW.01 - Grundlagen und Modelle

Anforderungen an ein Netzwerk

Ein Netzwerk soll einzelne Systeme so verbinden, dass zuverlässig, schnell und sicher Daten irgendwelcher Art ausgetauscht werden können. Das Netzwerk soll transparent funktionieren, so dass auch unterschiedliche Systeme (andere OS, andere HW) problemslos miteinander kommunizieren können.

Ein Netzwerk hat folgende wichtige Eigenschaften:

- Datenübertragungsrate
 - Wir wollen als Benutzer immer höhere Werte erreichen. Grenzen der Physik
 - Gewisse Kapazität Bit/s oder Byte/s
- Antwortzeiten
 - Spielt bei Echtzeit Informationen eine Rolle oder wenn sehr rasches reagieren gefragt ist (Online-Gaming)
 - Ping: je schneller desto besser
- Übertragungsfehler
 - Mit diesen Fehlern müssen wir leben
 - Störungen und Ausfälle lassen sich vermindern und zum Teil korrigieren
 - Können aber nie ganz vermieden werden
- Zuverlässigkeit
 - Kann mit Parallelität und Redundanz verbessert werden
- QoS (Quality of Services)
 - Ist eine grosse Sammlung von Parametern
 - Für die Qualität und Bandbreite einer Verbindung

Datenübertragung

Die Datenübertragungsrate muss genau definiert werden, da es sonst zu Missverständnissen kommen kann. Sie wird in bit/s (bps), kbit/s, Mbit/s oder Gbit/s gemessen.

- Datenübertragungsrate, Kanalkapazität, Bandbreite, Durchsatz
 - Übertragungsrate die das Netzwerk liefert
- Bitrate, natürliche Bitrate, Datenrate
 - Datenrate die der Dienst benötigt

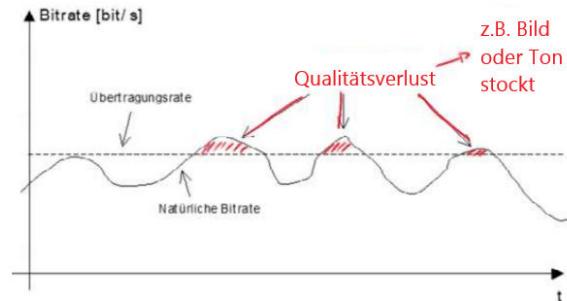
Typische Datenübertragungsraten

Technologie	Rate
Modem / Dialup	bis 56 kbit/s
Mobilfunk GSM-GPRS (typ. Konfiguration)	bis 53,6 kbit/s
ADSL (down)	8 - 24 Mbit/s
Ethernet	10 Mbit/s
Mobilfunk 3G: UMTS / mit HSPA	384 kbit/s / bis 42 Mbit/s
WLAN 802.11g	54 Mbit/s
Fast Ethernet	100 Mbit/s
USB 2.0 / 3.0	480 Mbit/s / 2,4 Gbit/s
Gigabit Ethernet	1-100 Gbit/s
HDMI 1.3	8,1 bis 10,2 Gbit/s
HDMI 2.1	38,4 Gbit/s
WLAN 802.11n	bis 600 Mbit/s
Mobilfunk 4G: LTE / LTE-Advanced	300 Mbit/s / 1000 Mbit/s
Mobilfunk 5G	Bis 10 Gbit/s

Entscheidend für die Datenrate (Bitrate), ist hauptsächlich die Qualität und der Frequenzbereich des Dienstes.

Bei konstanter Bitrate wird an heiklen Stellen ein Qualitätsverlust in Kauf genommen. Da die natürliche Bitrate die Übertragungsrate übersteigen kann, kann es an diesen Stellen zu Qualitätsverlusten kommen.

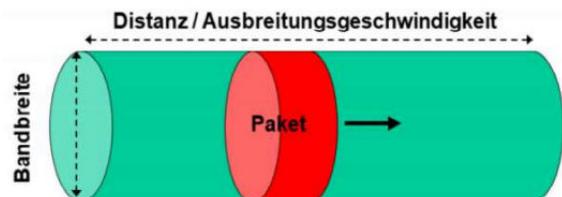
Somit kommt es zu Fluktuationen der natürlichen Bitrate.



Fluktuation hier: Anomalien in der Übertragungsgeschwindigkeit

Latenzzeit

Kein Signal kann Daten sofort übertragen. Die Daten müssen von der Quelle zuerst ans Ziel kommen. Die Zeit die dafür benötigt wird, nennen wir Latenzzeit.



In Glasfasern bewegen sich die Signale mit Lichtgeschwindigkeit. Die leitstrahlischen Signale in den Kupferkabeln bewegen sich mit ca. 2/3 der Lichtgeschwindigkeit. Bei längeren Distanzen spielen diese 1/3 Lichtgeschwindigkeit definitiv eine Rolle.

i Glasfaserkabel Übertragungsgeschwindigkeit ca. 300'000 km/s
Kupferkabel Übertragungsgeschwindigkeit ca. 200'000 km/s

Übertragungsfehler

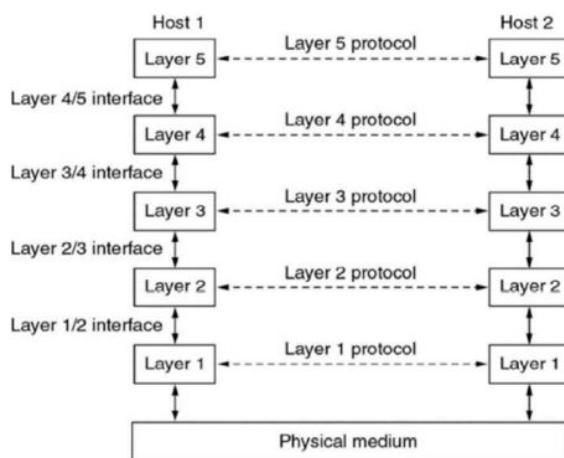
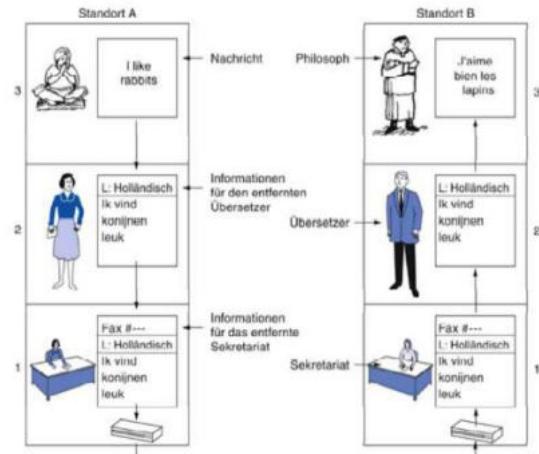
Es kann auf der Leitung immer wieder mal zu Übertragungsfehlern kommen. Diese Bitfehlerrate sollte man kennen, um den Aufwand abzuschätzen der benötigt wird, um diese Fehler korrigieren zu können. Dazu gibt es 2 Berechnungen die man machen kann. Einmal die Bitfehlerrate, um die Rate der fehlhaften Bits auszurechnen und die Paketfehlerrate, um die Rate der fehlerhaften Pakete zu berechnen.

Das Schichtenmodell

Es ist kaum möglich, die gesamte Datenübertragung in einem Schritt zu definieren und abzuwickeln. Deshalb wurde dies in mehrere Schichten unterteilt, welche jede für sich einen in sich abgeschlossenes Set von Definitionen umfasst.

Das bedeutet, dass die Übertragung von Daten auf mehrere Schichten geschieht. Wichtig ist es, dass Empfänger und Sender die selben Protokolle kennen.

Der Sender auf dem Layer 1 muss wissen, wie er seine Signale übermitteln soll, damit der Empfänger diese auch versteht.



Damit ein Paket dann auch wirklich am richtigen Ort ankommt, muss jeder Layer dieses Paket anders handhaben.

Eine Message M wird z.B. an einen anderen Empfänger übermittelt. Diese Message M wird auf dem Layer 5 erstellt und wird dann durch die verschiedenen Layer bis zum Layer 1 durchgereicht, zu einem Paket verschnürt und mit den jeweiligen Layer Header versehen. Teilweise wird die Message auch geteilt und verkleinert.

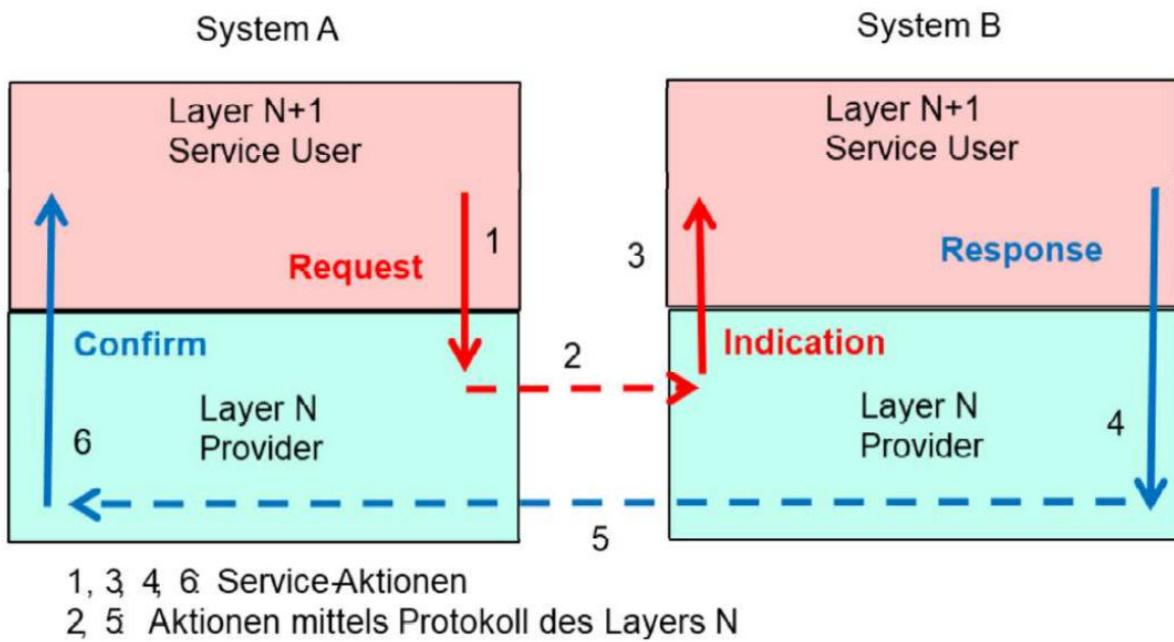
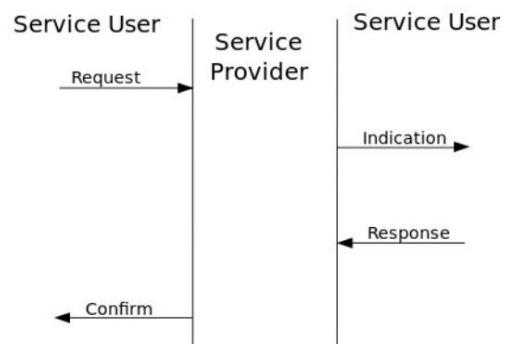
Sobald das Paket beim Empfänger auf dem Layer 1 ankommt, wird es wieder auseinander genommen. Die Layer kennen ihre eigenen Paketinformationen. So können sie das Paket nun wieder von Layer zu Layer übergeben um es in ihre Ursprungsform (Information) auf Layer 5 zurückzubringen.

Service Access Points

Service Access Points sind Dienstzugangspunkte (Schnittstellen) zwischen den Schichten. Es ermöglicht den Schichten untereinander zu kommunizieren.

An diesen Service Access Points wird eine ganz einfache Sprache und einfache Regeln definiert. Dabei teilt sich dies in vier Gruppen auf:

- Request (REQ): Instanz der Schicht N fordert Dienst der darunterliegenden Schicht N-1 an
- Indication (IND): N-1 signalisiert den Erhalt von Daten von Layer N
- Response (RESP): Layer N reagiert auf die Indication und bestätigt den Daten erhalt von Layer N-1
- Confirm (Conf): Service-Provider N-1 bestätigt den Erhalt der Daten von Layer N



Die **Schnittstelle (Interface)** zeigt, welche Dienste angeboten werden und wie auf diese Dienste zugegriffen wird (Befehle, Parameter, Ergebnisse).

Ein **Dienst** gibt an, welche Dienstleistungen (Service Primitives) sie der übergeordneten Schicht über den Service Access Point anbietet. Die übergeordnete Schicht ruft die Dienstleistung als Operation auf.

Das **Protokoll** regelt den Datenaustausch zwischen den gleichen Schichten auf zwei Rechnern. Das Protokoll ist eine Art Verpackungs- und Entpackungsanleitung.

Service Kategorien

Eine weitere Eigenschaft einer Verbindung ist die Verbindlichkeit der Übertragung. Dabei wird in zwei Kategorien unterschieden.

Connection-Oriented Services

Es wird zuerst eine Verbindung aufgebaut, für diesen Verbindungsaufbau wird eine Adresse benötigt. Sobald die Verbindung dann aber steht werden die Nachrichten ohne Adresse hin und her geschickt. Dabei gibt es drei Phasen:

Verbindungsaufbau → Datenübertragung → Verbindungsabbau

Connectionless Services

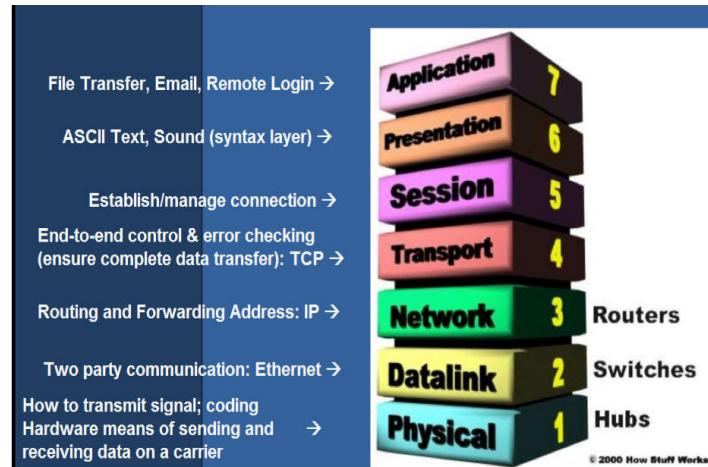
Hier wird jede einzelne Nachricht an den Adressaten verschickt. Es wird sich selber nicht um die Verbindung gekümmert und ob die Nachricht richtig ankommt ist unbekannt.

Referenzmodelle

Eines der möglichen Referenzmodelle ist das OSI-Modell.

Auf Deutsch:

- 7: Anwendungsschicht
- 6: Darstellungsschicht
- 5: Sitzungsschicht oder Kommunikationssteuerungsschicht
- 4: Transportschicht
- 3: Vermittlungsschicht
- 2: Sicherungsschicht
- 1: Bitübertragungsschicht



OSI-Modell

Layer 7 - Application Layer

Layer 7, auch als Anwendungsschicht bezeichnet, ist die oberste Schicht im OSI-Modell. Sie ist verantwortlich für die Interaktion zwischen Anwendungen und dem Netzwerk. Hier werden Protokolle und Dienste wie HTTP, FTP und DNS verwendet, um die Kommunikation zwischen Anwendungen auf verschiedenen Geräten zu ermöglichen.

Bekannte Protokolle:

1. HTTP (Hypertext Transfer Protocol): Ein Protokoll zur Übertragung von Webseiten und Daten im World Wide Web.
2. HTTPS (Hypertext Transfer Protocol Secure): Eine sichere Version von HTTP, die Verschlüsselung für sichere Datenübertragung verwendet.
3. SMTP (Simple Mail Transfer Protocol): Ein Protokoll zum Versenden von E-Mails über das Internet.
4. POP3 (Post Office Protocol version 3): Ein Protokoll zum Abrufen von E-Mails von einem Mailserver.
5. IMAP (Internet Message Access Protocol): Ein Protokoll zum Zugriff auf und Verwalten von E-Mails auf einem Mailserver.
6. DNS (Domain Name System): Ein Protokoll zur Auflösung von Domainnamen in IP-Adressen.
7. SSH (Secure Shell): Ein Protokoll zur sicheren Remote-Verbindung zu einem entfernten System. Es ermöglicht die sichere Übertragung von Daten und die Fernsteuerung von Systemen über eine verschlüsselte Verbindung.
8. FTP (File Transfer Protocol): Ein Protokoll zum Übertragen von Dateien zwischen einem Client und einem Server über ein Netzwerk.

Layer 6 - Presentation Layer

Der Presentation Layer, auch als Darstellungsschicht bezeichnet, ist die sechste Schicht im OSI-Modell. Sie kümmert sich um die Datenformatierung, Datenkompression und Verschlüsselung, um sicherzustellen, dass verschiedene Systeme miteinander kompatibel sind.

Bekannte Protokolle:

1. JPEG (Joint Photographic Experts Group): Ein Kompressionsprotokoll für die effiziente Übertragung von Bildern.
2. MPEG (Moving Picture Experts Group): Ein Protokoll für die Komprimierung und Übertragung von Audio- und Videodaten.
3. SSL (Secure Sockets Layer): Ein Verschlüsselungsprotokoll für die sichere Übertragung von Daten über das Internet.
4. TLS (Transport Layer Security): Ein Nachfolger von SSL, der ebenfalls zur sicheren Datenübertragung verwendet wird.

Layer 5 - Session Layer

Der Session Layer, auch als Sitzungsschicht bezeichnet, ist die fünfte Schicht im OSI-Modell. Sie etabliert, verwaltet und beendet Kommunikationssitzungen zwischen Anwendungen.

Bekannte Protokolle:

1. NetBIOS (Network Basic Input/Output System): Ein Protokoll für die Namensauflösung und Kommunikation zwischen Computern in einem lokalen Netzwerk.
2. RPC (Remote Procedure Call): Ein Protokoll, das es ermöglicht, eine Prozedur oder Funktion auf einem entfernten System auszuführen, als ob sie lokal aufgerufen würde. Es erleichtert die Kommunikation und Interaktion zwischen Client- und Serveranwendungen über ein Netzwerk.
3. SDP (Session Description Protocol): Ein Protokoll zur Beschreibung von Sitzungen und deren Eigenschaften. Es wird verwendet, um Informationen über multimediale Sitzungen wie Audio- und Videostreams, Teilnehmer, Codecs und andere Parameter zu übermitteln. SDP ermöglicht es Anwendungen, die benötigten Ressourcen für eine Sitzung zu verhandeln und zu koordinieren.

Layer 4 - Transport Layer

Der Transport Layer, auch als Transportschicht bezeichnet, ist die vierte Schicht im OSI-Modell. Sie bietet zuverlässige Datenübertragung zwischen Endpunkten und sorgt für Flusskontrolle und Fehlerbehebung.

Bekannte Protokolle:

1. TCP (Transmission Control Protocol): Ein verbindungsorientiertes Protokoll für die zuverlässige Übertragung von Datenströmen.
2. UDP (User Datagram Protocol): Ein verbindungsloses Protokoll für die schnelle Übertragung von Daten ohne Überprüfung der Zustellung.
3. SCTP (Stream Control Transmission Protocol): Ein Protokoll zur zuverlässigen Übertragung von Datenströmen mit Unterstützung für Multihoming und Multi-Streaming.

Layer 3 - Network Layer

Der Network Layer, auch als Vermittlungsschicht bezeichnet, ist die dritte Schicht im OSI-Modell. Sie ermöglicht die Verbindung von unterschiedlichen Netzwerken und die Weiterleitung von Datenpaketen über Routern.

Bekannte Protokolle:

1. IP (Internet Protocol): Ein Protokoll zur Adressierung und Weiterleitung von Datenpaketen im Internet.
2. ICMP (Internet Control Message Protocol): Ein Protokoll zur Fehlerdiagnose und Nachrichtenübermittlung im Netzwerk.
3. ARP (Address Resolution Protocol): Ein Protokoll zur Auflösung von IP-Adressen in physikalische MAC-Adressen.
4. OSPF (Open Shortest Path First): Ein Routing-Protokoll für das effiziente Weiterleiten von Datenpaketen in IP-Netzwerken.

Layer 2 - Data Link Layer

Der Data Link Layer, auch als Sicherungsschicht bezeichnet, ist die zweite Schicht im OSI-Modell. Sie stellt eine zuverlässige Verbindung zwischen benachbarten Netzwerknoten her und behandelt Fehlererkennung und -korrektur.

Bekannte Protokolle:

1. Ethernet: Ein weit verbreitetes kabelgebundenes Netzwerkprotokoll.
2. Wi-Fi (IEEE 802.11): Ein drahtloses Netzwerkprotokoll für die kabellose Datenübertragung.
3. HDLC (High-Level Data Link Control): Ein Protokoll zur Übertragung von Daten über Synchronleitungen.
4. PPP (Point-to-Point Protocol): Ein Protokoll zur Punkt-zu-Punkt-Datenübertragung über serielle Verbindungen.
5. PPTP (Point-to-Point Tunneling Protocol): Ein Protokoll zur Erstellung virtueller privater Netzwerke (VPNs) über öffentliche Netzwerke.

Layer 1 - Physical Layer

Der Physical Layer, auch als physikalische Schicht bezeichnet, ist die unterste Schicht im OSI-Modell. Sie ist für die Übertragung von Rohdaten über physikalische Medien wie Kabel oder Funkwellen verantwortlich.

Bekannte Protokolle:

1. Ethernet (physical layer): Ein weit verbreitetes kabelgebundenes Netzwerkprotokoll.
2. RS-232: Ein serieller Kommunikationsstandard für die Übertragung von Daten zwischen Geräten.
3. USB (Universal Serial Bus): Ein Standardprotokoll zur Verbindung von Geräten mit Computern und anderen elektronischen Geräten.

Modell-Arten

Strukturmodell

Sie dienen in allererster Linie zum Aufzeigen von Zusammenhängen und für das Verständnis des gesamten und groben Zusammenspiels von Komponenten.

- Physikalische Relationen
- Logische Relationen
- zeigt die Komponenten
- zeigt die Funktionsweise

z.B ein Modellauto

Referenzmodell

Es werden alle Einzelheiten genau aufgezeigt, so dass sich z.B. verschiedene Hersteller daran orientieren können (komplett und sehr detailliert).

- Umfassende Darstellung
- dient als Bezug für reale Implementierung
- dient als Basis für Normen

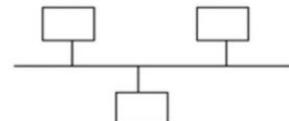
z.B OSI-Modell

Topologien

Mit den Topologien, kann das Zusammenschalten von Geräten in einem Netzwerk veranschaulicht werden.

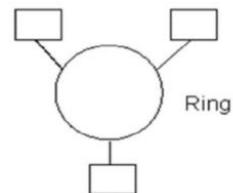
BUS-Topologie

- Sie stammt ursprünglich noch aus der Ethernet-Welt, als noch Koaxialkabel verwendet wurden.
- Heute wird dies vor allem noch im Steuerungsumfeld von Maschinen eingesetzt, da der Verkabelungsaufwand deutlich reduziert werden kann.



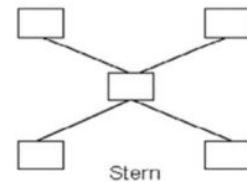
RING-Topologie

- Ringsysteme stammen aus der Tokenring Zeit (IBM), werden heute aber aus Redundanzgründen immer noch verwendet.
- Sollte der Ring aufgeschnitten werden so funktioniert die Verbindung immer noch in umgekehrter Reihenfolge.



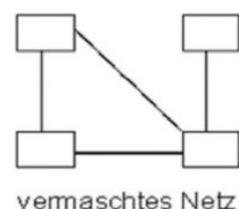
STERN-Topologie

- Der Stern ist das gebräuchlichste Verfahren für kleinere Netwerke. In der Mitte befindet sich dann z.B. der Switch
- Es kann ein einzelner Zugangspunkt ausfallen ohne dass das Netz zusammenbricht. Wichtig dabei ist das die Mitte stabil läuft.



VERMASCHTES-Netzwerk

- Dieses Netzwerk wird im globalen Netzwerk verwendet, da damit viele Redundanzen und Lastverteilungen erreicht werden. Die einzelnen Blöcke können hier ganze Netzwerke in beliebigen Topologien darstellen.

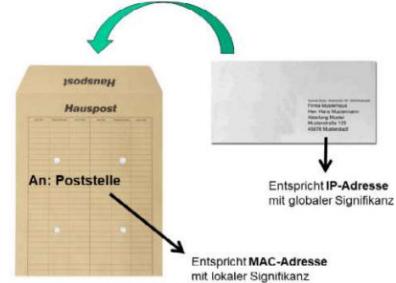


vermaschtes Netz

Adressräume

Damit die einzelnen Systeme im Netzwerk miteinander kommunizieren können, müssen sie sich eindeutig identifizieren können. Sie brauchen also eine Adresse. Jeder Ethernet-Chip bekommt bereits bei der Herstellung eine eindeutige **MAC** (Media Access Control) Adresse.

In einem kleinen lokalen Netz funktioniert die Kommunikation über MAC Adressen. Wenn aber ein weltweites Netzwerk aufgebaut werden soll, müsste jedes Ethernet-System die MAC-Adressen aller anderen Systeme kennen. Das wären $281 \cdot 10^{12}$ Adressen. Dies wäre schlicht unmöglich. Dazu kommt dass wenn sich die Topologie ändert, müssten diese riesigen Tabllen weltweit abgeglichen werden.



Daher wurde die **Internet-Paket-Adressierung (IP-Adressen)** entwickelt. Diese Adressierung ist hierarchisch aufgebaut und besteht aus 4 Bytes (IPv4). Dabei kann grob gesehen jede Stelle als Verfeinerung der vorangegangenen interpretiert werden.

Die **MAC-Adresse** hat somit eine lokale und die **IP-Adresse** eine globale Bedeutung.

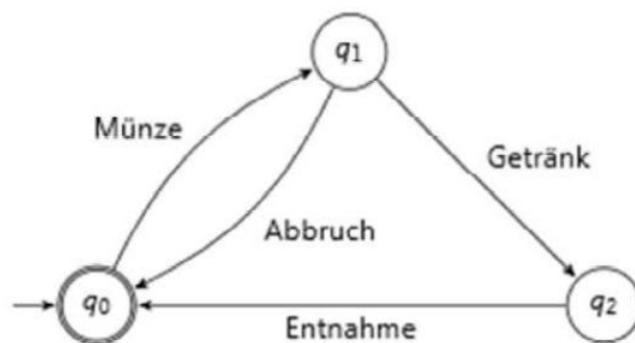
Endliche Automaten

Ein endlicher Automat beschreibt ein System mit verschiedenen Zuständen. Er besteht aus Zuständen / Übergängen / Eingängen / Ausgaben. Die Darstellung erfolgt mit Blasen für Zustände (Bubble, Knoten) und Pfeile für Übergänge (Arrow, gerichtete Kante).

Wird oft für den Entwurf von Hardware wie auch für Software verwendet.

Beispiel eines endlichen Automaten — Der Getränkeautomat

1. Münze einwerfen → Q1
2. Getränk wählen → Q2 oder abbrechen → Q0.
3. Getränk wird ausgeworfen, Getränk entnehmen
→ Q0 (Bereit für



Mehr zum Thema Automaten und Graphen steht in meiner Zusammenfassung für das Modul **Algorithmen und Datenstrukturen** auf **Github** <https://github.com/omeldar/hslu>

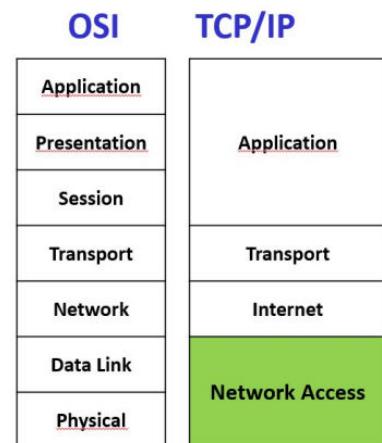
NW.02 Physical Layer, Network Access

Der Physical Layer

Der Physical Layer ist für die Übertragung der Bits (0, 1) Informationen verantwortlich.

- Elektrisch: Spannung, Frequenz, Phase, ...
- Optisch: Lichtwellen (Infrarot und sichtbar)
- Drahtlos: elektromagnetische Wellen (Funk, Radio), Licht, Infrarot, Ultraschall

Alle Daten werden mit einer Frequenz übertragen, wobei die Frequenz die Anzahl Schwingungen pro Sekunde steht (in Hz, kHz, MHz, GHz).



Datenübertragung

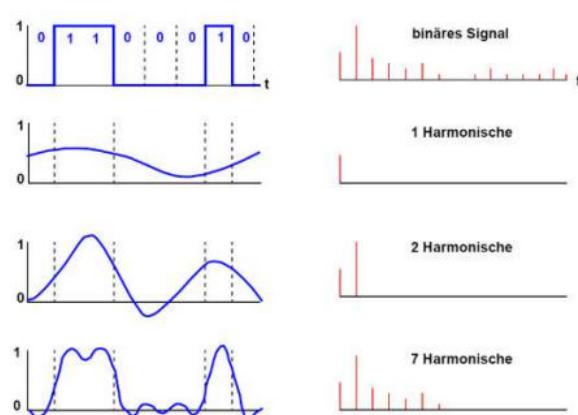
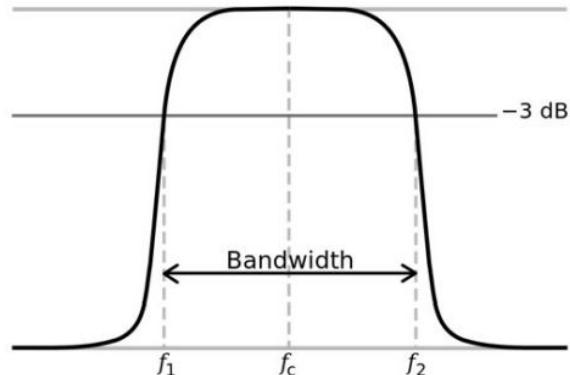
Durch die minimale und maximale Grenzfrequenz kann die Bandbreite (hier B) bestimmt werden.

$$B = f_{max} - f_{min}$$

Die Grenzfrequenz ist bei 50% der Leistungsübertragung definiert.

$$-3\text{dB} = 10 \cdot \log_{10}(0.5)$$

Bei Glasfasern z.B. gibt es Lichtwellen, welche durch das Glas gesendet werden können und es gibt Lichtwellen, die nicht durch das Glas gesendet werden können.



Die Fourierreihe beschreibt ein Signal $s(t)$ als Summe von Sinus und Cosinus Schwingungen in verschiedenen Frequenzen.

Ein Binärsignal enthält Schwingungen der Grundfrequenz, aber auch Anteile mit verschiedenen Vielfachen der Grundfrequenz (Harmonische).

Die maximale Datenübertragungsrate ist durch Shannon/Hartely so definiert:

Allgemein:

$$R = 2 \cdot B \cdot \log_2 N$$

R: Übertragungsrate, Datenrate in Bit/s

B: Bandbreite der Leitung in Hz

Mit Rauschen:

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

N: Anzahl diskrete Signalstufen

S/N: Rauschabstand (Signal-to-Noise)

Übertragung der Daten Asymmetrisch

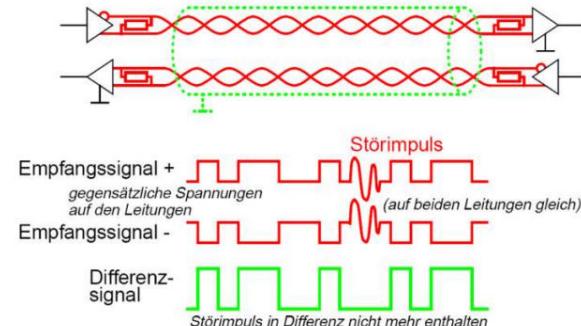
Die Datenleitungen laufen parallel zueinander und haben somit eine gemeinsame Masse für die Hin- und Rückleitung.



Leider sind sie daher anfällig für Störungen und Rauschen. Eine Abschirmung des Kabels hilft hierbei. Es verhindert, die Einstrahlung von Störungen und verhindert das Abstrahlen von Störungen (z.B. Störungen von anderen Funkdiensten).

Übertragung der Daten Symmetrisch (differentiell)

Jedes Signal wird über zwei Leitungen mit gegensätzlichen Spannungspegeln übertragen. Damit können Störimpulse erkannt und werden direkt herausgefiltert.



Kabel- und Leitytypen

Twisted-Pair-Kabel

Twisted Pair Kabel sind verdrillte Kupferkabel welche für die symmetrische Übertragung gebraucht werden.



Bsp. CAT-3



Bsp. CAT-5

- CAT-3: Telefonnetz, Ethernet 10Base-T mit ~ 10Mbit/s
- CAT-5: Ethernet 100Base-TX mit ~100Mbit/s
- Cat-5e/6: Gigabit-Ethernet GbE mit ~ 1Gbit/s
- Cat-6e/7: 10-Gbit-Ethernet 10GbE mit 10Gbit/s und mehr

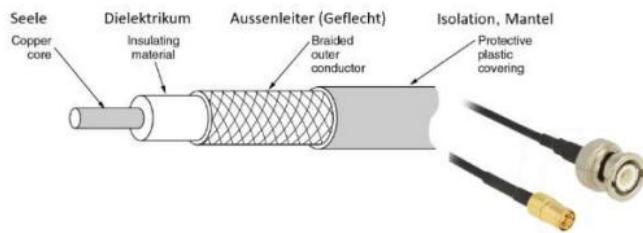
Bei der Anschluss-Schnittstelle handelt es sich um einen RJ45 (Modulstecker 8P8C)



Koaxialkabel

Das Koaxialkabel wird nur noch in wenigen Spezialfällen für die reine Datenübertragung verwendet. Damit ist eine Übertragung in beide Richtungen auf vielen Kanälen gleichzeitig möglich.

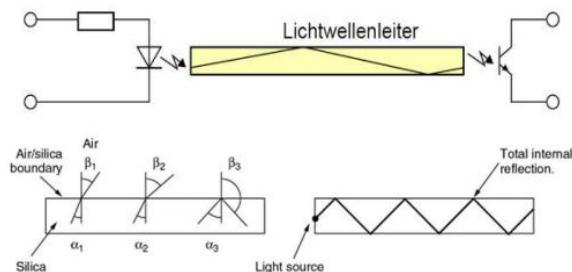
- 50 Ohm für Messgeräte in der Elektronik als Prüfleitung
- 75 Ohm für (Kabel-) Fernsehen



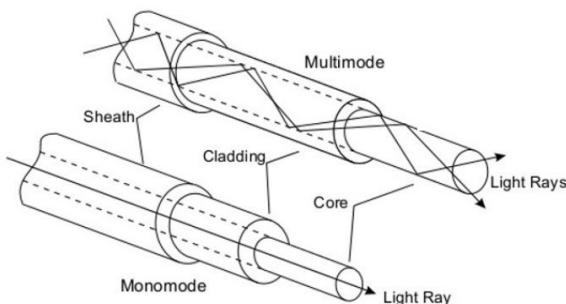
Glasfaserleiter

Die Glasfaserleiter macht sich die Eigenschaften und die Geschwindigkeiten des Lichts zu nutzen. Es arbeitet mit Totalen Reflexionen innerhalb der Glasfaser.

Die Glasfaser selber ist sehr anfällig auf ihr Umfeld und lange nicht so robust wie eine Kupferleitung. Deswegen ist diese gut geschützt und darf nicht zu stark gekrümmmt werden.



Sie wird in zwei Modes eingeteilt.



Multimode

Im Multimode breiten sich viele einzelne Lichtwellen in der Glasfaser aus. Sie hat einen Core-Durchmesser von $65,5\mu\text{m}$ oder $50\mu\text{m}$ und ist somit relativ dick. Die Lichtwellen darin haben einen Frequenzbereich von 850nm (rot). Multimode wird für kurze Strecken eingesetzt.

Monomode

Im Monomode breitet sich nur eine einzelne Lichtwelle in gerader Linie aus. Sie ist mit einem Core-Durchmesser von nur 3 bis $9\mu\text{m}$ einiges dünner. Die Lichtwelle darin hat einen Frequenzbereich von $1'300$ / $1'550\text{nm}$ (Infrarot). Monomode wird für lange Strecken verwendet.

Die technisch nutzbare Bandbreite ist bei beiden jeweils 25 bis 50 GHz. Im Labor ist sie aber deutlich höher.

Vergleich der Leitungen

	Frequenzen	Repeater-Abstand	Bandbreite
Freileitung	0 Hz - 100 kHz	2 - 20 km	< 10 kHz
Verdrillte Kupferleitung	4kHz - 1 MHz	2 - 20 km	100 kHz - 600 kHz
Koaxialkabel	bis 500 MHz	1 - 10 km	900 MHz
Hohlleiter (Mikrowellen, Radar)	3 - 30 GHz		
Lichtwellenleiter	30 GHz	10 - 100 km	40 Gbit/s (Labor 160 Gbit/s)

Wireless Übertragung

Signale (Daten) können auch drahtlos übertragen werden. Diese sind somit nicht leitungsgebundene Frequenzen.

- GSM: 900MHz, 1800MHz
- UMTS: 1885 - 2025 MHz, 2110 - 2200 MHz
- WLAN: 2.4GHz bis 2.4835 GHz, 5GHz
- Bluetooth: 2.4GHz
- 5G-Netz: 700MHz - 3.8GHz

Der Data Link Layer

Der Data Link Layer ist die Sicherungsschicht. Folgende Aufgabe hat die Sicherungsschicht:

- Übertragung von Datenframes (Datenrahmen)
- Fehlersicherung (sicherstellen dass alle Datenframes fehlerfrei empfangen werden)
- Adressierung der Pakete auf dem Medium

Bei der **Übertragung von Daten werden Fehler auftreten**. Sei das durch thermisches Rauschen in (Halb-) Leitern, durch elektromagnetische Einstrahlungen (Übersprechen, Funkanlagen, Motoren) oder durch radioaktive Einstrahlungen (Höhenstrahlungen, Alpha-Teilchen).

Die **Datenübertragung soll aber fehlerfrei sein**. Fehler müssen erkannt oder sogar behoben werden. Dazu braucht es eine Lösung. Diese kommt in der Form von Prüfbits oder Prüfsummen, die ebenfalls übertragen werden. Der Sender setzt diese, der Empfänger liest diese ein und kann so die Daten überprüfen.

Paritätsbit

Das **Paritätsbit** wird an den übertragenen Daten mitangehängt. Es wird dann die Summe aller 1-er insgesamt ausgerechnet und je nachdem eine zusätzliche 1 oder 0 angehängt.

01010101	-->	010101011	ungerade Parität
11010011	-->	110100111	gerade Parität
10010011	-->	100100110	gerade Parität

- So dass die Summe gerade ist für eine gerade Parität, even parity
Das bedeutet, dass alle Pakete eine gerade Anzahl von 1-er Bits haben (mit dem Parity-Bit).
- So dass die Summe ungerade ist für eine ungerade Parität, odd parity
Das bedeutet, dass alle Pakete eine ungerade Anzahl von 1-er Bits haben (mit dem Parity-Bit).

Hamming-Abstand

Der Hamming-Abstand (Hamming-Distanz) ist eine weitere Möglichkeit der Fehlerüberprüfung. Dabei werden den Daten noch zusätzliche Codeworte mit angefügt. Diese Codeworte basieren auf Paritycodes über einem Datenblock fixer Länge. Der Hamming-Abstand ist also die Anzahl Bits in denen sich zwei Codewörter unterscheiden.

Ein Datenblock von m bits hat somit 2^m mögliche Codewörter. Sollten alle 2^m Codewörter benutzt werden ist der Hamming-Abstand $h = 1$ und es ist somit keine Redundanz vorhanden.

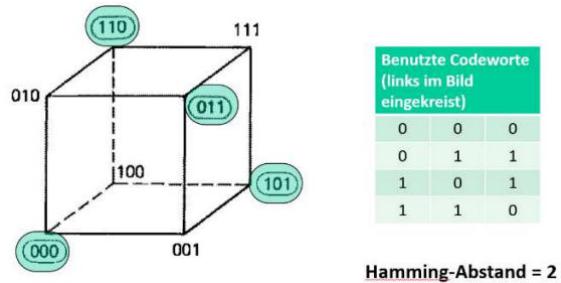
Beispiel:

10110011

11010010

Hier wäre der Hamming-Abstand: $h = 3$

Der Hamming-Code dient zur Erkennung von Fehlern. Bei diesem Beispiel ist der Coderaum $n = 3$ Bits. Und ein Hamming Abstand von 2. Die Codeworte werden so ausgewählt und zusammengestellt, dass sich immer zwei Bits unterscheiden, wenn man die einzelnen Zeilen vergleicht.



Hamming Code - Fehlererkennung und Fehlerkorrektur

Um e gleichzeitig auftretende Einzelbitfehler erkennen zu können, braucht man eine Kodierung mit einem Hamming-Abstand von $h = e + 1$

- Wenn e Bits (oder weniger) fehlerhaft sind, entsteht ein ungültiges Codewort
- Der Empfänger erkennt so den Fehler

Um e Bitfehler korrigieren zu können, benötigt man eine Kodierung mit Hamming-Abstand von $h = 2 \cdot e + 1$

Redundanzprüfung Zyklisch mit CRC (Cyclic Redundancy Check)

Dies ist ein mathematisches Modell zur Erzeugung einer effizienten Prüfsumme. Es wird mithilfe einer Bitfolge als Koeffizient ausgerechnet. Beispiel anhand von 10011001

$$\begin{aligned} 1x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 0x^1 + 1x^0 \\ = x^7 + x^4 + x^3 + 1 \end{aligned}$$

Mit einer einfachen Rechnung kann die zu sendende Information mit dem Generatorpolynom ausgerechnet werden. Der überbleibende Rest wird der zu sendenden Information angehängt und mitgeschickt.

CRC Bestimmung mittels Polynom Division: Die Rechnung wird nach den Regeln der algebraischen Körpertheorie durchgeführt. Ohne Borgen und Übertragen. Das heißt anstatt Subtraktion sind es einfache EXOR Verknüpfungen an jeder Stelle und die erste führende Null pro Resultatzeile wird weggelassen.

Siehe Kapitel NW.01 - NW.09 Formeln & Berechnungen

Erzeugung von Frames

Bei der Übertragung von grossen Datenmengen werden diese kleine Stücke zerlegt und durchnummieriert, damit beim Empfang und zusammensetzen der Frames verlorene Frames erkannt werden. Diese Frames werden dann schlussendlich auf dem Layer 1 als binäre Informationskette übertragen.

FLAG	Header	Payload field	Trailer	FLAG
------	--------	---------------	---------	------

- Flagbyte: eindeutige Kennung für Start und Ende
- Header: Adressierung und Zusatzinfo / Codierung
- Payload: Die Nutzlast Daten aus iener höheren Schicht
- Trailer: angehängte CRC-Daten und Zusatzinfos

NW.03 - Zugriffsverfahren und 802.xx Familie

Zugriffsverfahren

Das Zugriffsverfahren kann je nach Dienst und Service unterschiedlich sein.

Direktverbindung (Punkt zu Punkt)

- Ein Sender
- Ein Empfänger
- Ein Verbindungskanal
- z.B Ethernet Kabel 100BaseT von Switch zum Rechner

Broadcast (Rundruf)

- Mehrere (potentielle) Sender
 - Ein Sender aktiv
- Mehrere (potentielle) Empfänger
- Ein Verbindungskanal
- z.B. WLAN

Ein Broadcast ist im Prinzip das Senden von Informationen an mehrere Systeme. Dies geschieht in dem man die Information auf einen Übertragungskanal legt, auf den alle Empfänger Zugriff haben. Es gibt 4 unterschiedliche Möglichkeiten für ein Broadcast:

- Broadcasting: Versand an alle
- Multicasting: Versand an eine Gruppe
- Unicasting: Versand an einzelne Empfänger
- Anycast: Versand an einen (beliebigen) Rechner aus einer Gruppe

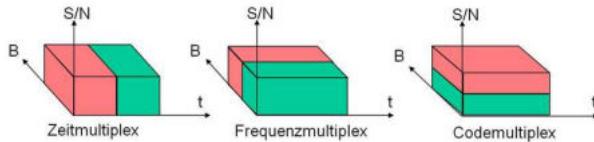
So ein Broadcast-Kanal wird auch als Mehrfachzugriffskanal (Multiaccess Channel) oder Random Access Channel (Wahlfreier Zugriffskanal) bezeichnet. Das heisst, jeder Sender hat (zunächst) die freie Wahl, wann er den Kanal benutzen und ein Frame senden möchte.



IMPORTANT:

Da nicht alle gleichzeitig auf einer Leitung sprechen können, muss dies geregelt werden. Das Medium Access Control Protokoll (MAC) regelt die Vergabe des Zugriffsrechts auf den Übertragungskanal (Zugriffsverfahren).

Statische Kanalzuordnung



FDM

Frequency Division Multiplexing, Frequenzmultiplex

- Die Bandbreite wird in Frequenzbereiche aufgeteilt
- Jedes System erhält ein eigenes Frequenzband

TDM

Time Division Multiplexing, Zeitmultiplex

- Wie bei FDM, jedoch Aufteilung des Kanals in Zeitschlüsse (nummeriert, wiederholt)

CDMA

Code Division Multiple Access, Codemultiplex

- Wie bei FDM und TDM, jedoch Aufteilung der Codierung → Sprache

SDMA

Space Division Multipl eAccess, Raummultiplex

- Aufteilung des Raumes z.B in Funkzellen beim Mobiltelefon

Dynamische Kanalzuordnung

Für die dynamische Kanalzuordnung kommen diverse MAC Protokolle zum Einsatz. Diese werden vor allem gebraucht, wenn viele unabhängige Stationen über einen Einzelkanal senden und empfangen möchten. Und es dabei zu Kollisionen kommen kann.

Einige der Mehrfachzugriffsprotokolle sind:

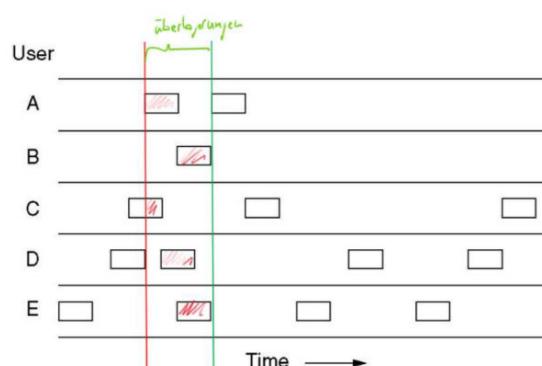
- ALOHA
- CSMA (Carrier Sense Multiple Access)

Das ALOHA Protokoll

ALOHA ist ein sehr einfache und kostengünstiges Protokoll. Es wurde auf Hawaii entwickelt.

Beim reinem ALOHA Protokoll darf jede Station zu einem beliebigen Zeitpunkt senden und es wird regelmässig überprüft ob die Frames angekommen sind. Sollte das nicht der Fall sein so wird nach einer zufälligen Wartezeit das Frame erneut gesendet.

Eine verbesserte Version ist das Slotted-ALOHA (S-ALOHA). Dabei werden die Übertragungen mit definierten Zeitschlitten synchronisiert.

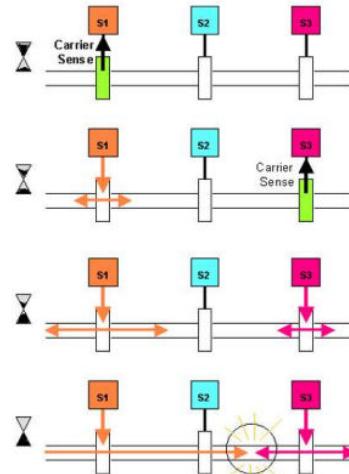


CSMA (Carrier Sense Multiple Access) Protokoll

Der Sender hört den Datenverkehr auf der Leitung ab (= carrier sense) und wartet bis die Leitung frei ist. Sobald dann die Leitung frei ist, darf dann gesendet werden. Falls aber mehrere Sender (fast) gleichzeitig anfangen zu senden, gibt es eine Kollision und der Vorgang wird nach einer zufälligen Zeitspanne wiederholt.

Bei **CSMA/CA (CA = Collision Avoidance)** wird zur Kollisionsvermeidung nach dem erkennen eines freien Kanals eine zufällige Wartezeit abgewartet. Wird z.B bei WLAN 802.11 — DCF angewandt.

Bei **CSMA/CD (CD = Collision Detection)** wird die Übertragung abgebrochen sobald eine Kollision erkannt wird. z.B. bei Ethernet.



Die **Durchsatzraten** hängen von den entstehenden Kollisionen ab.

- Die kleinste Durchsatzrate hat ALOHA und es kommt ständig zu Kollisionen
- Slotted ALOHA hat eine höhere Rate, da die Kommunikation zu definierten und synchronisierten Zeitschlitten stattfindet
- Beim CSMA werden die meisten Kollisionen vermieden, da bereits laufende Übertragungen erkannt werden. Es gibt nur noch durch die Laufzeitverzögerungen bedingte Kollisionen
- Beim CSMA/CD werden dann diese wenigen Kollisionen noch frühzeitig erkannt und der Kanal sofort freigegeben, also eine Verbesserung zu CSMA
- Beim CSMA/CA werden Kollisionen von Grund auf vermieden, was die höchstmögliche Rate ergibt

Es gibt auch noch **komplett Kollisionsfreie Protokolle**

- Token-Ring: Kommunikation ist synchronisiert durch ein umlaufendes Token. Dieser Token-Ring wird heute aber nur noch selten verwendet
- Bitmusterprotokolle, Bit-Arbitierung, Binärer-Countdown: Jeder Teilnehmer hat eine eigene Kennung (ID) oder Zahl die seiner Priorität entspricht
 - Die Teilnehmer senden z.B gleichzeitig, beginnend mit dem höchst-wertigsten Bit ihrer Kennung
 - Sobald ein Teilnehmer einen anderen Sender mit einer höheren Kennung erkennt, zieht er sich zurück
 - Typische Beispiele sind der CAN-Bus (Controller Area Network), welches auch als CSMA/CR → Collision Resolution bezeichnet wird.

Die 802.xx Familie

Der Layer 2 (Data Link Layer) wird oft weiter in zwei Teilschichten unterteilt:

- LLC (Logical Link Control): In diesem werden die Datenframes von einer Station zu einer anderen gesichert transportiert.
- MAC (Medium Access Control): Vergibt Zugriffsrechte auf den Kanal

3	Network Layer	Netzwerk-verwaltung	802.1 Internetworking (High Level Interface)			
2	Data Link Layer	Logische Verbindungs-steuerung	802.2 (LLC: Logical Link Control)			
		Medium-zugriffs-steuerung	MAC: Medium Access Control		802.3	802.5
1	Physical Layer	elektronischer und mechanischer Anschluss	Ethernet	Tokenring	802.11	802.15
			Zugriff auf physikalisches Medium			
				WLAN	WPAN (Bluetooth)	

LLC - Logical Link Control

Die 802 - Protokolle bieten bis zur MAC-Schicht keinen gegen Frameverluste gesicherten Dienst. Das LLC-Protokoll fügt in jedem Frame einen LLC Header zu. Dieser Header enthält Folge und Bestätigungsnummern.

Mit LLC sind folgende Dienste möglich:

- Unzuverlässiger Datagrammdienst: Wie eine Werbesendung, als normaler Brief
- Bestätigter Datagrammdienst: Wie ein eingeschriebener Brief mit Rückantwort
- Zuverlässiger, verbindungsorientierter Dienst: Wie eine dedizierter Verbindungskanal, Rohrpost, Botenübermittlung

Ethernet IEEE 802.3 Framestruktur (vereinfacht):

56	48	48	16	variabel	variabel	32
Präambel	Ziel-adresse	Quell-adresse	Tag	LLC-Header	Datenbereich	CRC

- Präambel (Flag): Zur Regenerierung des Sendetakts (64 Bit), Vorspann
- MAC-Zieladresse (48 Bit)
 - 0xxx ... xxx - Unicast: Individuelle Adresse. In Hardware fest vorgegeben
 - 1xxx ... xxx - Multicast
 - 1111 ... xxx - Broadcast
- MAC-Quelladresse (48 Bit)
- Datenfeld, 0 bis 1500 Bytes gross
- CRC Prüfsumme über alle Felder ausser der Präambel
- MAC (Medium Access Control) wird CSMA/CD eingesetzt, jedoch meist Vollduplex von Punkt-zu-Punkt Verbindung zu einem Switch

NW.04 - Networking & Routing

Netzwerk Layer

Der Data Link Layer vermittelt Pakete von Punkt zu Punkt, also immer nur eine Teilstrecke. Der Netzwerk Layer hingegen transportiert Pakete von der Quelle bis zum Ziel durch das gesamte Netzwerk. Bzw. alle Teilnetze (End-to-End).

Der Netzwerk Layer stellt dem darüber liegenden Transport Layer seine Dienste zur Verfügung. Dieser Dienst muss unabhängig sein von der Anzahl der Router / der Router Technologie / der Netztopologie. Weiter muss die Netzadresse für den Network Layer global eindeutig sein.

Ein Netzwerk kann aufgeteilt werden in:

- PAN: Personal Area Network (Vernetzung der von einer Person benutzten Geräte)
- LAN: Local Area Network (Vernetzung in einem Gebäude)
- MAN: Metropolitan Area Network (Vernetzung einer Stadt, kaum noch gebräuchlich)
- WAN: Vernetzung von Städten, Ländern, Interkontinental. (auch: Weltverkehrsnetz)

Der Network Layer kann auf 2 Arten arbeiten:

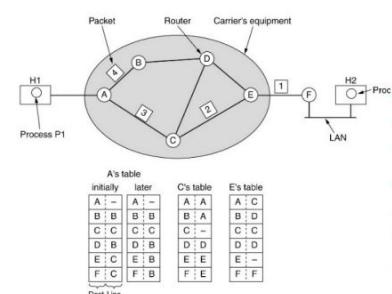
- Verbindungslos (Datagrammdienst)
 - Vermittlung einzelner Pakete durch das Netzwerk
 - Jedes Paket trägt volle Zieladresse mit sich
 - Jedes Paket kann unterschiedliche Wege nehmen
 - Die Fehlerüberwachung machen die Hosts
 - Internet Protokolle (reines IP-Paket, ICMP oder auch UDP-Paket)
- Verbindungsorientiert (virtuelle Verbindung, VC = virtual circuit)
 - Verbindung wird aufgebaut
 - Qualität der Verbindung kann gewährleistet werden
 - Verbindung wird nach der Übertragung auch wieder abgebaut
 - ATM, FrameRelay, X.25, Telefonsignalisierung (ISDN), GPRS, MPLS

Bei der Paketvermittlung durch ein Netzwerk wird das Store-and-Forward-Verfahren angewendet. Das heisst, die Pakete werden jeweils an den nächsten Router / Knoten geschickt. Dort werden sie gespeichert, verarbeitet und dann schon zum nächsten Router weitergeschickt.

Die Steuerprotokolle der Vermittlungsschicht sind: ICMP, ARP, DHCP

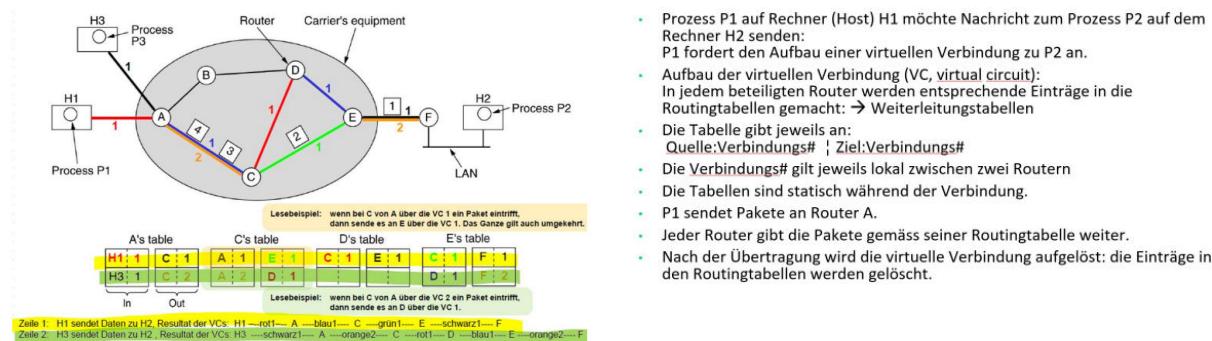
Paketvermittlung

Bei der Paketvermittlung (Verbindungslosen Dienst) werden die Pakete einzeln an das Netzwerk übergeben. Die einzelnen Pakete nehmen somit unabhängig voneinander ihren eigenen Weg durch das Netzwerk. Solche Pakete werden auch Datagramme genannt. Somit ist ein solches Netzwerk ein Datagramm-Netzwerk.



Leitungsvermittlung

Bei der Leitungsvermittlung (verbundungsorientierter Dienst) muss zuerst eine eigene Verbindung aufgebaut werden. Dafür wird ein Sondierungspaket vorausgeschickt und hinterlässt auf jedem Router spuren, so dass dann darüber eine virtuelle Verbindung (virtual Circuit, kein VPN) eingerichtet wird. Sobald diese Verbindung steht, können darüber alle Pakete, welche einen Kennzeichner für diese Verbindung haben, darüber versendet werden. So brauchen diese Pakete keine IP-Adresse und die Übermittlung ist teilweise sogar performanter.



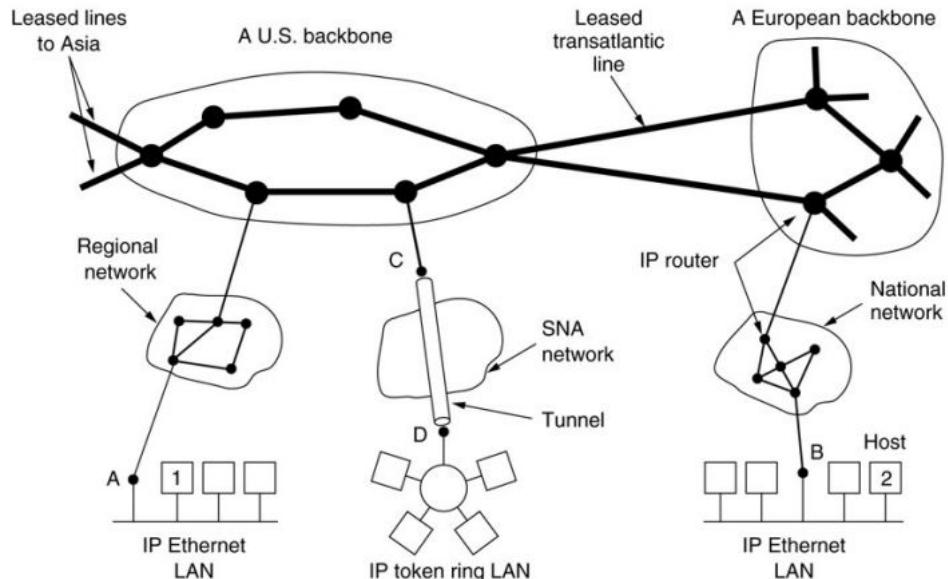
Verbindungslos vs. Verbindungsorientiert

	Teilnetz mit Datagrammen	Teilnetz mit virtuellen Verbindungen
Verbindungsaufbau	Nicht erforderlich	Erforderlich
Adressierung	Jedes Paket enthält volle Quell- und Zieladresse	Für jede virtuelle Verbindung ist ein Tabelleneintrag erforderlich
Statusinformationen	Router führen keine Statusinformationen über die Verbindung	Für jede virtuelle Verbindung ist ein Tabelleneintrag erforderlich
Routing	Jedes Paket wird unabhängig befördert	Die Route wird beim Aufbau der virtuellen Verbindung gewählt. Alle Pakete folgen dieser Route.
Wirkung von Fehlern	Keine, ausser das Pakete verloren gehen	Alle virtuellen Verbindungen über den ausgefallenen Router werden beendet.
Dienstgüte	Schwierig zu gewährleisten	Einfach, wenn im Voraus für jede virtuelle Verbindung ausreichend Ressourcen bereitgestellt werden
Überlastungsüberwachung	Schwierig	Einfach, wenn im Voraus für jede virtuelle Verbindung ausreichend Ressourcen bereitgestellt werden

Routing-Algorithmen

Diese Algorithmen sind Vorschriften / Verfahren / Prozeduren, mit welchen der Weg durch ein Netzwerk gesucht / optimiert / gefunden werden kann.

Das Internet ist eine riesige Ansammlung von einzelnen Teilnetzen.



Die Darstellung von einem Netzwerk, wobei hier die Buchstaben (A,B,C,D,E,F) die Router sind und die Zahlen 1-5 die verschiedenen Netzwerke.

Damit das Routing funktioniert, benötigt jeder Router für die Weiterleitung von Informationen eine Routingtabelle. So eine Routingtabelle kann statisch oder dynamisch erstellt werden. Bei einer statischen Routingtabelle müssen die Einträge manuell erstellt werden. Das heißt auch Änderungen muss man manuell eintragen.

Bei einer dynamischen Routingtabelle wird diese selbstständig immer erweitert und angepasst. Dafür braucht es während des Betriebs zusätzliche Protokollmeldungen und es kann mit Netzwerkfehlern viel besser umgehen. Die Routing-Algorithmen sind adaptive Algorithmen und passen sich der Topologie an. Routing Entscheidungen werden aufgrund von Messungen oder Schätzungen (Lastregulierung) gemacht. Das Ergebnis ist eine ausgefüllte Routing-Tabelle.

Wenn ein Router neu das Netzwerk betritt, meldet er sich oft per Broadcast bei allen, damit andere Router diesen in die Tabelle aufnehmen können.

Übersicht der Routing-Algorithmen

i In diesem Abschnitt wird oft von Algorithmen und Graphen (Knoten, Kanten, Gewichten) gesprochen. Ich gehe nicht gross darauf ein hier.

Details zu SPF und Dijkstra sowie Graphentheorie steht in meiner Zusammenfassung für das Modul **Algorithmen und Datenstrukturen** auf **Github** <https://github.com/omeldar/hslu>

Zum Thema Graphentheorie wird auch noch etwas in meiner Zusammenfassung für das Modul Diskrete Mathematik Ende Februar 2024 stehen. → Gleicher Github Repo!

Shortest Path Routing (SPF, Dijkstra)

Der Shortest Path Routing, Dijksta Algorithmus, sucht den kürzesten Pfad. Daher wird der kürzeste Pfad (mit dem geringsten Gewicht → Graphentheorie) ermittelt und rechnet dann systematisch und schrittweise den Weg von Startknoten zum Zielknoten. Diese Route ist dann statisch eingetragen.

Flooding

Beim Flooding wird ein Paket über alle Ausgangsleitungen gesendet. Das Netzwerk wird regelrecht geflutet. Sobald das erste Paket angekommen ist, ist der schnellste Pfad gefunden. Dieses package-flooding muss mittels Teilstreckenzählern kontrolliert werden, da sonst Pakete ewig im Netzwerk im Kreis drehen könnten (Package-Lifetime setzen). Das Flooding ist ein robustes Verfahren, allerdings aufwändig und nur eine statische *Jetzt-Aufnahme* des Netzwerks. Oft wird es auch als Benchmark für andere Algorithmen verwendet.

Distance Vector Routing

Dieser Algorithmus ist sehr dynamisch. Dieser berücksichtigt auch die aktuelle Netzlast. Die einzelnen Router haben somit eine eigene Tabelle mit den bestmöglichen Verbindungen. Der Nachteil hierbei ist, dass sich Informationen über schlechte Kanäle nur sehr schlecht verbreiten.

Link State Routing

Beim Link State Routing ermittelt jeder Router seine Nachbarrouter und seine Entfernung zu diesen (Gewicht der Kanten im Graphen → Graphentheorie). Dieser sendet dann diese Information an alle Router weiter und so wird die gesamte Topologie erforscht. Danach wird der kürzeste Pfad berechnet, z.B mit SPF. Hierbei wird sehr viel Datenverkehr (Traffic) erzeugt.

Hierarchisches Routing

Beim hierarchischen Routing wird das gesamte Netzwerk in Regionen unterteilt. Jeder Router kennt nur die anderen Router in seiner Region. Ein überregionales Paket wird dann über Verbindungsrouter in andere Regionen geschickt. Somit gibt es eine Mehrstufige Hierarchie, z.B Regionen, Cluster, Zonen, Gruppen

Broadcast-Routing

Der Router versendet einen Broadcast an alle Teilnehmer seines Netzwerks und gibt bekannt welche anderen Router er kennt. Sollte hierbei ein anderer Router zuhören kann er dies bei sich eintragen und für später verwenden. Dafür wird mit Multidestination-Routing und dem Spanning-Tree Protocol gearbeitet.

Multicast-Routing

Ein Spezialfall des Broadcast Routing wobei nur eine Gruppe von Hosts angesprochen wird.

Routing für mobile Hosts

Das Routing für mobile Hosts ist sehr aufwändig. Da sich die Stationen ständig bewegen und aufgepasst werden muss, dass die Verbindung nicht abreisst. Mit IPv4 relativ aufwändig, mit IPv6 viel einfacher.

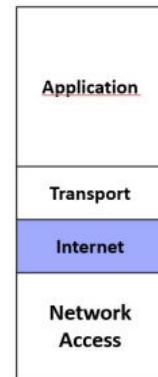
Routing in Ad-hoc-Netz

In einem Ad-hoc-Netz gibt es keine feste Topologie. Daher ist das Routing hier auch speziell. Die Hosts würden sich spontan untereinander verbinden (ohne Access-Point) und Informationen zu Verbindungen zu anderen Hosts austauschen. Sie würden dann selbstständig ein kleines Miniurnetzwerk erstellen. Wenn sich zum Beispiel Autos untereinander über Informationen wie bevorstehendes bremsen, etc. austauschen könnten, um die Sicherheit zu erhöhen.

NW.05 - Internetworking

Das gesamte Netzwerk ist kein homogenes Netzwerk. Es besteht aus vielen kleinen einzelnen Teilnetzwerken. Es gibt verschiedenste Protokolle, Frameformate, Hardware, Leitungen und Modulierungen.

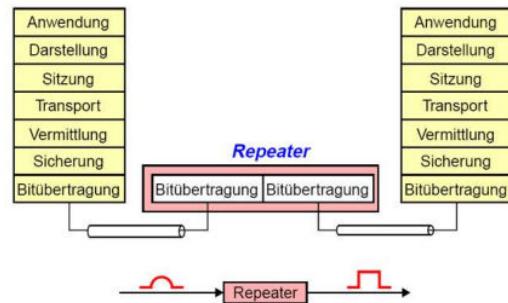
Das Ziel der Verbindungen der Netze ist das Benutzer in den unterschiedlichen Netzen miteinander kommunizieren und Daten über Netzgrenzen hinweg austauschen können. Das Problem hierbei ist gerade die Unterschiedlichkeit der Netze und die verschiedenen Adressierungsarten.



Netzwerkgeräte

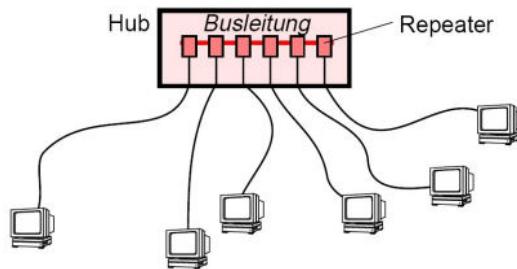
Repeater → Layer 1

- Hat keine Auswertung der Protokolle
- Einfachste Verbindungskomponente
- Zugeschnitten auf das Transportmedium
- Regeneriert lediglich das Signal auf der Leitung
- Trennt Übertragungsmedium in zwei Segmente
- Für Datenfluss transparent
- Nur gleichartige Netze verbindbar



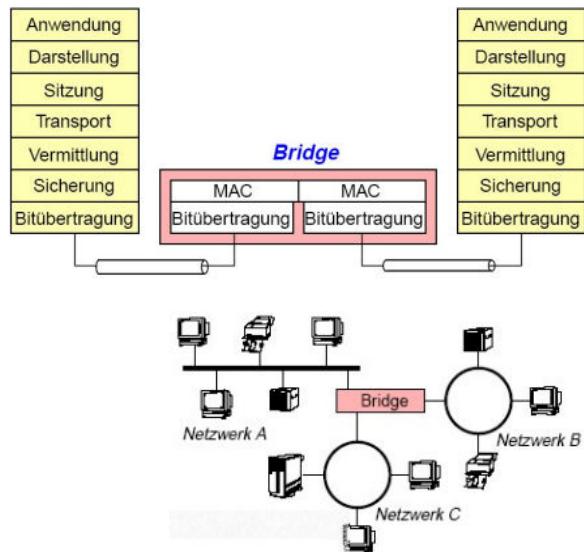
Hub → Layer 1

- Sternförmige Verkabelung
- Hub = Konzentrator welcher Ethernet Kabel simuliert
- Intern prinzipiell gleiche Funktion wie Repeater
- Kann auch als *Multiportrepeater* beschrieben werden



Bridge → Layer 2

- Intelligenter als ein Repeater
- Können gleichartige und verschiedenartige Netzwerke verbinden
- Werden nicht adressiert (angesprochen)
- Lesen alle Frames eines angeschlossenen Netzwerks und bauen aus den MAC-Adressen eine eigene Tabelle mit allen Stationen
- Sobald die MAC-Adresse nicht in der Tabelle ist leitet die Bridge das Frame auf die andere Seite
- Ist dynamisch und lernt während des Betriebs dazu



Transparente Bridge

Verbindung zweier lokaler Netze mit dem selben Data-Link Protokoll. z.B. Ethernet → Ethernet

Übersetzende Brdige

Verbindung zweier lokaler Netze mit verschiedenen Data-Link Protokollen, können mit Rahmenanpassungen umgehen, meist auch unterschiedliche Anschlüsse. z.B. Ethernet → Tokenring

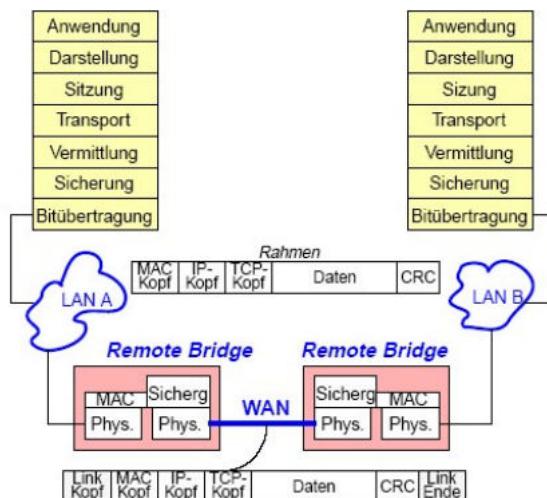
Virtuelle Bridge → VM's, Container

Erweitert reelle Netzwerkkarte um virtuelle Netzwerkkarte, die dann von einem virtualisierten OS benutzt werden kann.

Remote-Bridge (Tunnel) → Layer 2

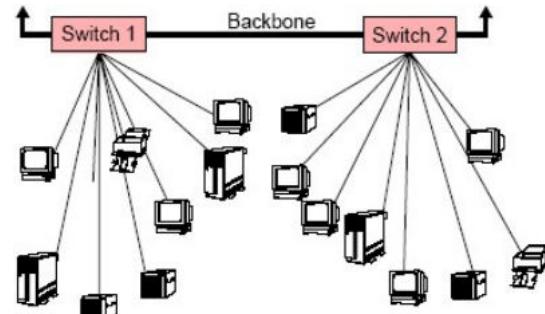
- Unterstützung der Verbindung weit entfernter LANs über ein WAN
- Datenübertragung mittels WAN-Protokoll über WAN
- Mehrere LAN-Frames werden in ein WAN-Paket gepackt und auf der Gegenseite wieder entpackt
- LAN-Frames tunneln durch das WAN

Ähnlich wie ein VPN allerdings auf Layer 2, VPN ist verschlüsselt auf Layer 3.



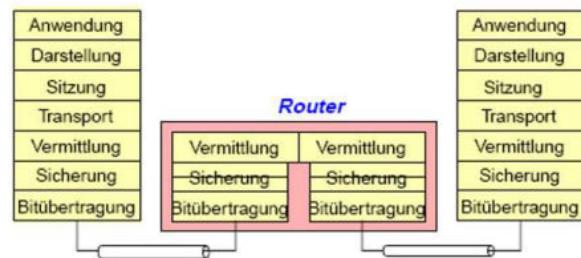
Switch → Layer 2

- Physisch wie ein Hub aufgebaut an die jeweils ein Computer angeschlossen wird. Funktioniert aber wie eine Ansammlung von Bridges
- Simuliert Netzsegmente, die jeweils mit einer Bridge verbunden sind
- Der Switch schaltet jeweils Rechner zusammen
- Jedes Netzsegment arbeitet unabhängig, d.h. viele Pakete können gleichzeitig transportiert werden
- Erhöhung der Lokalität und Geschwindigkeit auf den einzelnen Segmenten



Router → Layer 3

- Arbeitet im Internet Layer
- Im Gegensatz zu Bridges werden hier Schicht 3 Adressen (z.B. IP-Adressen) betrachtet
- Aktiver Teilnehmer im Netz (mit eigener Adresse)
- Bestimmt den günstigsten Pfad über eine Kette von Verbindungen zwischen Quelle und Ziel

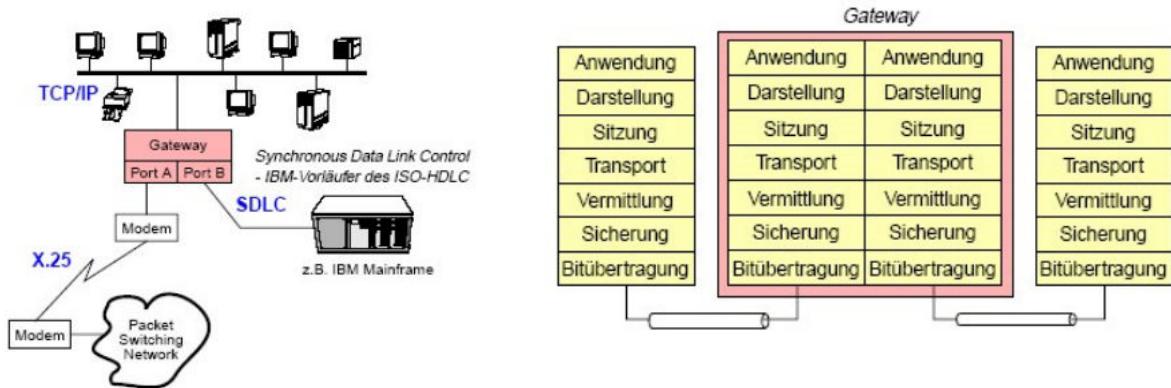


Router vs. Bridges

Router	Bridge
Router sind effizienter bei komplexen und dynamischen Netzwerken	Bridges sind effizienter bei Verbindungen mit wenigen Pfaden
Router machen eine aufwändige Wegesuche und verwenden aufwändige Algorithmen um den günstigsten Weg zu finden	Bridges machen nur binäre Entscheidungen, anhand den physikalischen Adressen
Router oder Layer-3 Switch: öffnet das Paket, entfernt MAC-Frame, wertet IP-Adresse aus und schickt das Paket weiter in ein Teilnetz in dem es in ein neues MAC-Frage verpackt wird	Bridge oder Layer-2-Switch: schaltet das Paket einfach direkt durch

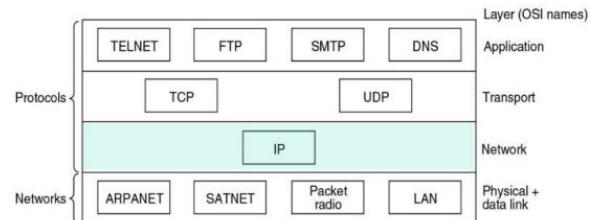
Gateway und Firewall

Gateways sind durch alle Schichten hinweg einsetzbar. Sie verbinden Netzwerke transparent miteinander und können auch mit den unterschiedlichsten Protokollen arbeiten. Sobald dann noch Regeln zur Weiterleitung von Paketen hinzukommen, die Pakete auch vernichten können, spricht man von einer Firewall.



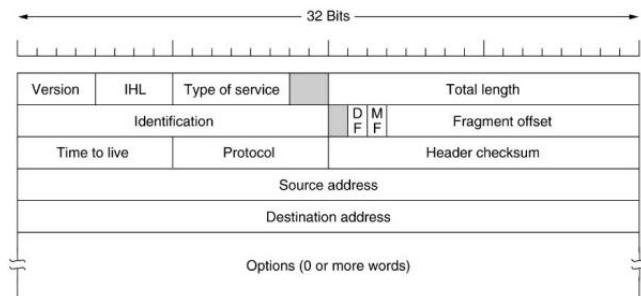
Das Internet Protokoll

IP ist das Protokoll der Internet-Schicht, also dem Network Layer im TCP/IP -Protokollstapel. TCP/IP ist die Bezeichnung für die gesamte Protokoll-Suite. Es ist eine Zusammensetzung aus den meistverwendeten Layer 4 Protokoll TCP und der IP-Adressierung → IP-Protokolls. Es gibt also kein TCP/IP Protokoll und auch keine TCP/IP Adressen.



IP-Header

- Version
- IHL (IP Header Length)
- Diensttyp: Vorrang (Delay, Durchsatz, Zuverlässigkeit)
- Gesamtlänge des IP Pakets (max 64'535 Bytes)
- Identifikation: kennzeichnet Fragmente eines Datagramms



- Fragment-Offset, Lebensspanne (TTL), Protokoll, Prüfsumme
- Quelladresse, Zieladresse

Länge 20 Byte + optionaler Abschnitt (Total max. 60 Byte) — kann somit variieren in der Länge.

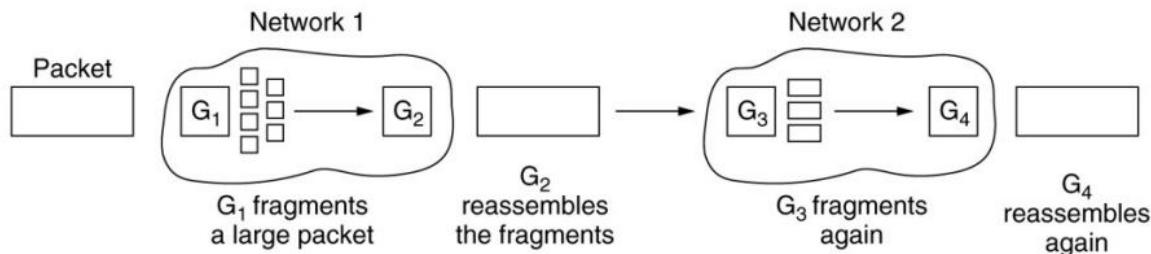
Fragmentierung des IP Pakets

Bei der Fragmentierung werden grosse Pakete in kleine Pakete aufgeteilt. Der Grund dafür ist die maximale Nutzdatengröße (engl. MTU, max transfer unit) einzelner Netzwerksegmente. Das Problem dabei: Diese fragmentierten kleinen Pakete wieder zusammenzusetzen. IP-Pakete sind nicht immer alle gleichberechtigt.

Dafür gibt es zwei Ansätze:

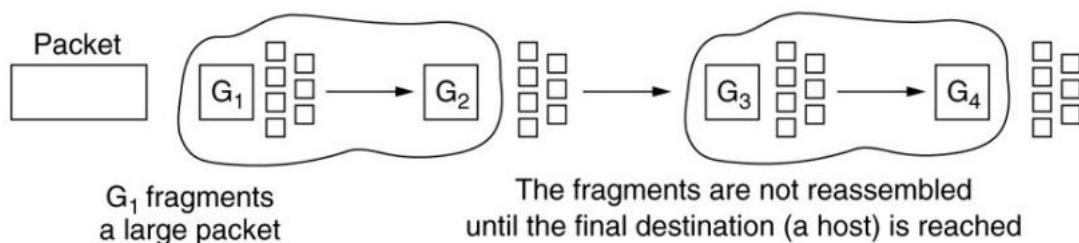
Transparente Fragmentierung

⇒ Alle Fragmente gehen an das gleiche Ausgangs-Gateway und werden dort wieder zusammengesetzt.



Nichttransparente Fragmentierung

⇒ Das Zusammensetzen der einzelnen Fragmente geschieht erst beim Zielhost. Die Pakete können irgendeinen Weg durch das Netzwerk nehmen.



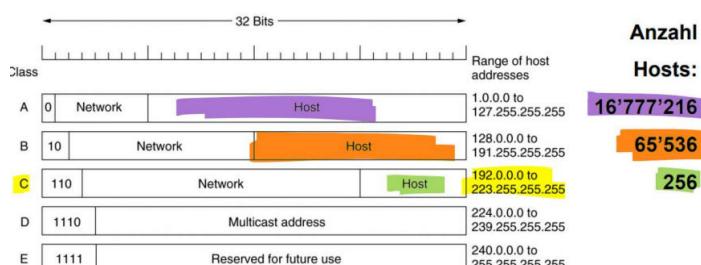
IP-Adressen IPv4

Die global gültigen IP-Adressen werden an einer zentralen Stelle verwaltet. ICANN und IANA

- <https://www.icann.org/> → Internet Corporation for Assigned Names and Numbers
- <https://www.iana.org/> → Internet Assigned Numbers Authority

Das Ziel war es, dass jeder Rechner eine eindeutige 32-Bit-Adresse erhält. Jedes Byte wird als Dezimalzahl geschrieben:

1000 0000 . 0000 1110 . 0000 0001 . 0000 1101 = 128.14.1.13



Spezielle IP-Adressen, Bereiche und Subnetze

IPv4	Use-Case
0.0.0.0	This host (localhost)
1.1.1.1	Broadcast on the local network
127.x.y.z	Loopback

Es gibt auch IP Adressen, welche nur im lokalen Netzwerk funktionieren:

IPv4	Use-Case
10.x.x.x	10.0.0.0 /8 (255.0.0.0)
172.16.x.x bis 172.31.x.x	172.16.0.0 /12 resp. 255.240.0.0
192.168.x.x	192.168.0.0 /16 resp. 255.255.0.0
Network.1.x.1	Broadcast on a distant network
169.254.x.x	169.254.0.0 /16 (APIPA: Automatic Private IP Addressing, kein DHCP)

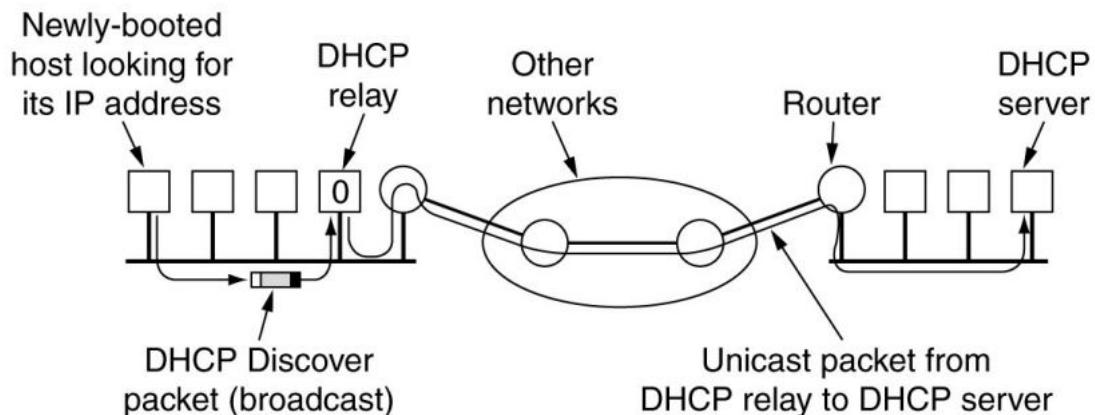
Da es immer mehr Netzwerkgeräte gibt und die 2^{32} Adressen nicht ausreichen, muss das Grosse Netz in Teilnetze aufgeteilt werden. Damit das Problem mit den Klassen überwindet werden kann, wurden die Adressen in Netz-Adressen und Host-Adressen aufgeteilt. Dazu hat jeder Router in seiner Routing-Tabelle einen zusätzlichen Eintrag mit der Netzmase (ebenfalls 32Bit). Die Auswahl des richtigen Weiterleitungsweg ist dadurch entsprechend aufwändiger.

Schreibweise:

- CIDR-Notation: z.B. 192.168.1.0 / 24 = 24 bit für Netzwerk, 8 bit für 254 Hostadressen
- Netzmase: 255.255.255.0 = 1111'1111.1111'1111.1111'1111.0000'0000

Arbeitsweise DHCP Protokoll

DHCP arbeitet auf dem Layer 7, kommuniziert jedoch mittels UDP auf Layer 4 und gibt einem neuen Host im Netz so die dynamische IP-Adresse, die Subnetzmaske, den Standardgateway und den DNS-Server mit.



IP-Adressen: IPv6

Im IPv6 Protokoll gibt es ein neues Adressformat:

- 128 Bit, 8 Blöcke a 16 Bit, in Hex
- 2001:0db8:0000:0000:0000:8a2e:0070:7344

Mit dieser Änderung vergrössert sich der Adressraum von 2^{32} (IPv4) auf 2^{128} (IPv6) Adressen. Im Verhältnis umgerechnet auf die Erdoberfläche: Mit IPv4 gäbe es 8.4 Adressen pro km^2 , mit IPv6 gäbe es 667 Billiarden Adressen pro mm^2 .

IPv4 Adressen: 4,294,967,296

IPv6 Adressen: 340,282,366,920,938,463,463,374,607,431,768,211,456

Auch bei IPv6 gibt es spezielle IP-Adressen:

- ::/128 (128 0-Bits) = IP Adressen nicht spezifiziert
- ::1/128 (127 0-Bits, 1 1-Bit) = loopback, localhost
- fe::/10 = Link-Lokal-Unicast-Adr. (^= IPv4-APIPA)
- fec0::/10 = Site-Lokal-Unicast-Adr. (^= IPv4 priv. Adr)
- Global Unicast
 - 0:0:0:0:ffff:/96 IPv4 mapped (letzte 32 Bit=IPv4)
 - 2000::/3 (2000... bis 3fff...): von IANA eindeutig vergeben

Weitere neue Features von IPv6 ist die Unterstützung für Authentifizierung, Security und Verschlüsselung (neuer Dienst IPSec). Es gibt einen vereinfachten und verbesserten Protokollheader (für Router vereinfacht). Die Mobile IP werden besser unterstützt und das umnummernieren ist einfacher geworden. Auch wird DHCP nicht mehr notwendig sein.

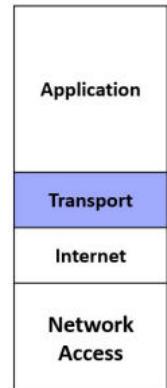
IPv6 Header

Feld	Länge	Inhalt
Version	4 Bit	IP-Versionsnummer (6)
Traffic Class	8 Bit	Quality of Service
Flow Label	20 Bit	Ebenfalls für QoS oder Echtzeitanwendungen
Payload Length	16 Bit	Länge des IPv6-Paketinhaltes (ohne Kopfdatenbereich, aber inklusive der Erweiterungs-Kopfdaten) in Byte
Next Header	8 Bit	Typ des nächsten Kopfdatenbereiches
Hop Limit	8 Bit	Entspricht Time to Live (TTL) bei IPv4.
Source Address	128 Bit	Adresse des Senders
Destination Address	128 Bit	Adresse des Empfängers

NW.06 Transportschicht - UDP & TCP

End-zu-End Protokolle bieten im Gegensatz zum Internetprotokol (IP) einen zuverlässigeren Dienst. Es benutzt das Internet Protocol (IP) und baut darauf einen zuverlässigen Datendienst. Damit bietet es einen End-zu-End Lieferdienst für die darüberliegenden Anwendungen.

- Verlorene, gestörte Pakete werden erkennt und durch Wiederholung der Übertragung korrigiert
- Verstecken der Vermittlungsschicht für obere Schichten
- Transportschicht verbessert die Dienstgüte
- Bildet einen End-zu-End Lieferservice



Sender und Empfänger

Die Transportschicht verbindet direkt Sender und Empfänger bzw. Client und Server. Die beiden Protokolle TCP (Transport Control Protocol) und UDP (Unsigned / User Datagram Protocol) sind beide auf dem Layer 4 zu finden. Dabei werden die Mängel der Vermittlungsschicht (Layer 3) verborgen, da sie nicht weitergegeben werden. Auch erkennt die Transportschicht verlorener und gestörte Pakete und kann diese durch Wiederholung der Übertragung korrigieren. Ähnlich wie in der Sicherungsschicht mit CSMA/CD & CSMA/CA.

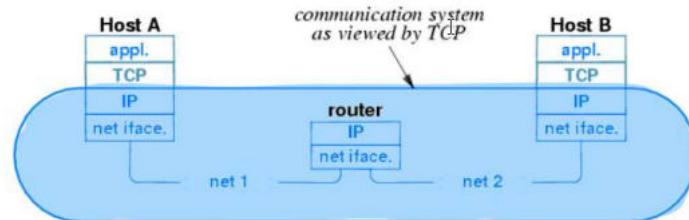
Service Primitives der Transportschicht:

- Listen: Block until some process tries to connect
- Connect: Actively attempt to establish a connection
- Send: Send information
- Receive: Block until a DATA packet arrives
- Disconnect: This side wants to release the connection

End-zu-End Protokolle

Bei den Protokollen in der Transportschicht (TCP/UDP) handelt es sich um End-zu-End Protokolle. Die Verbindung geht dabei direkt von einer Anwendung eines Computers zur Anwendung des anderen Computers. Aus der Sicht der Transportschicht ist das gesamte Internet ein einziger «einfacher» Übertragungskanal. Die Transportschicht muss sich mit folgenden Problemen beschäftigen:

- Verlorene oder defekte Pakete
- Pakete kommen verspätet an
- Verzögerungszeiten sind variabel
- Doppelte Pakete
- Datenstau, Anhäufung von empfangenen Paketen
- Neubooten von Rechner und Wiederaufnahme einer laufenden Übertragung



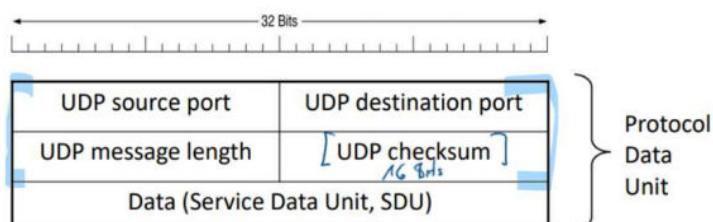
Dafür gibt es mehrere Lösungsansätze. Pakete werden mit Folgenummern durchnummeriert, die Verbindung wird angefordert und bestätigt oder Pakete haben eine Lebensdauer, welche mit einem Timer überwacht wird. Aber auch der 3-Way-Handshake ist so ein Lösungsansatz.

Die beiden Transportprotokolle von Layer 4, sind ziemlich unterschiedlich.

User Datagram Protocol (UDP)

UDP liefert unabhängige Meldungen, sogenannte Datagramme. Diese Datagramme können bei der Übertragung verloren gehen oder sie werden in falscher Reihenfolge geliefert. Daher garantieren optionale Checksummen Datenintegrität. Die Endpunkte einer solchen Übertragung werden Ports genannt, wobei der Prozess die Zuordnung zu einem Port festlegt. Bei jeder UDP Datenübertragung wird die IP-Adresse und die Port-Nummer des Senders und Empfängers festgelegt (Port-Nummer können bei Sender und Empfänger unterschiedlich sein).

Das UDP Protokoll ist sehr einfach, im Wesentlichen gibt es nur einen Header mit Quell- und Zielport, Paketlänge und Checksumme. Und danach kommt die Nutzlast (SDU) der Schichten darüber dazu.



Typische Anwendungen des UDP Protokolls ist die Kommunikation im Client-Server-Betrieb. Der Client sendet kurze Anforderungen an den Server und der Server führt diese Anfrage aus und sendet darauf eine kurze Antwort. Beispiele sind Remote Procedure Call (RPC), Real-Time Transport Protocol (RTP) und der Domain Name Service (DNS).

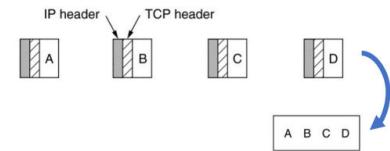
RPC (Remote Procedure Call) ruft Prozeduren auf einem entfernten Rechner aus. Wobei ein Stub (Stumpf, Stummel) auf jeder Seite die Übertragung auf dem Netzwerk regelt. Es simuliert sozusagen die Prozedur lokal, sendet aber die Anfrage ans Netzwerk.

Falls die Datenmenge zu gross wird, kommt das TCP-Protokoll zum Einsatz.

Transmission Control Protocol (TCP)

TCP gewährleistet eine zuverlässige Datenübertragung, was UDP nicht bieten kann. Es ist verbindungsorientiert und es wird ein fehlerfreier Datenstrom von einem Rechner zum anderen durch Timeout's und Paketwiederholungen garantiert. Es bietet Flusskontrolle zur Geschwindigkeitsanpassung. Die Endpunkte nennt man Sockets. Jeder Socket hat eine Portnummer von 16 Bit.

Damit sind TCP-Verbindungen Endpunkt zu Endpunkt Verbindungen, sie bieten daher keine Unterstützung für Multicast und Broadcast. Die Datenübertragung läuft im Duplexbetrieb (in beide Richtungen) und es werden keine Datenstrukturen gesendet, sondern es ist ein stetiger Bytestrom. Zusätzlich wird beim Sender die Übertragungsrate angepasst dank einer Überlastungsüberwachung.



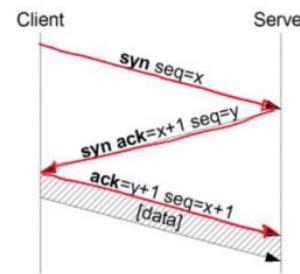
Ein TCP Bytestrom sind Daten die in kleinere Pakete aufgeteilt wurden. Dieser kann aus den Header Informationen auslesen wie viele Pakete (Datenmenge) bei dieser Übertragung zusammen gehören. Da die Pakete nummeriert sind, kann er diese egal in welcher Reihenfolge er dieser erhält richtig zusammensetzen.

Aufbau einer End-zu-End Verbindung mit 3-Way-Handshake

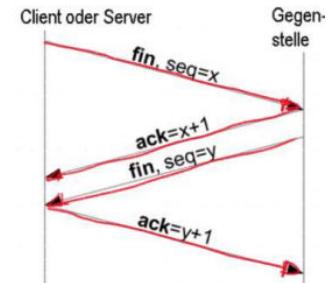
Der 3 Wege Handschlag ist dazu da eine saubere TCP Verbindung aufzubauen, dafür werden folgenden Bits verwendet:

- Syn: synchronize = Anfrage
- Ack: acknowledge = Bestätigung
- Fin: finish = Abschluss

Aufbau:

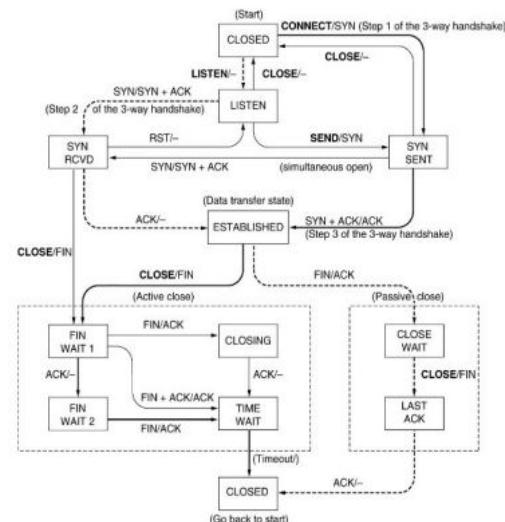


Abbau:



Sobald die Verbindung nicht mehr benötigt wird, muss diese Verbindung wieder abgebaut werden. Dazu gibt es noch die Bits:

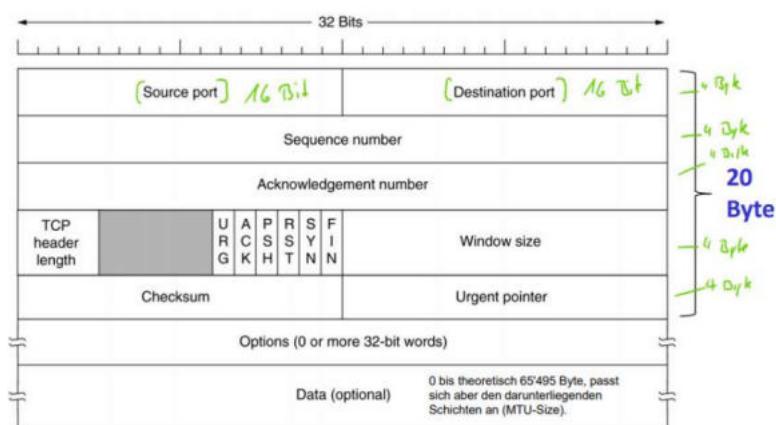
- CR: Connect Request
- ACK: Acknowledge
- DR: Disconnect Request



Übersicht der TCP Ports

Port	Use	Protocol
20/21	FTP	File Transfer
23	Telnet	Remote login
25	SMTP	E-Mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about user
80 / 443	HTTP / HTTPS	Hypertext Transfer Protocol (WWW)
110	POP-3	Remote E-Mail Access
119	NNTP	USENET news

TCP Header



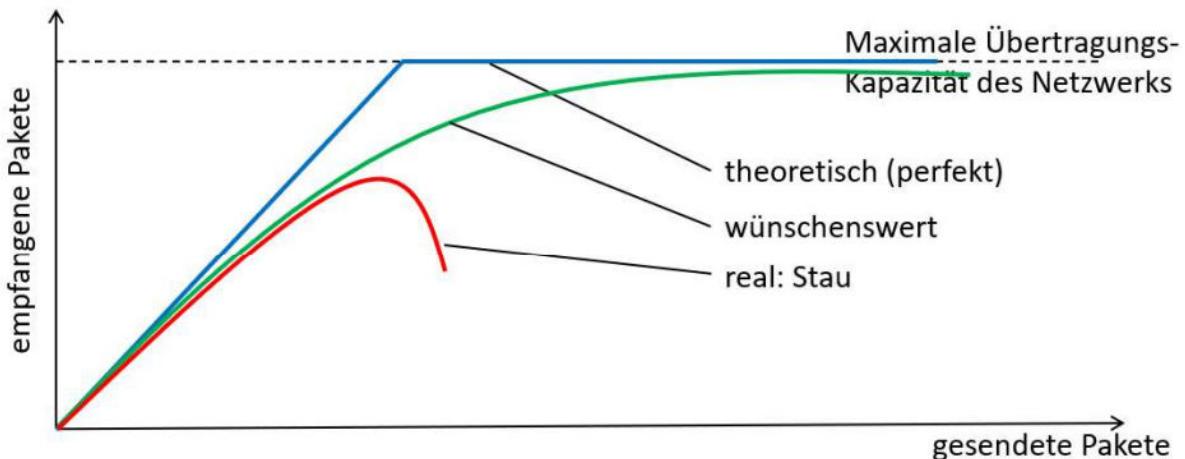
TCP vs. UDP

Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Gewährleistet eine zuverlässige Datenübertragung	Datagramm-Dienst
Verbindungsorientiert	Verbindungslos
Fehlerfreier Datenstrom von einem Rechner zum anderen	Unzuverlässig
Flusskontrolle zur Geschwindigkeitsanpassung	Für einmalige, schnelle Übertragung

NW.07 - Quality of Service

Überlastüberwachung

Wenn die Last grösser ist als die verfügbaren Ressourcen kann es zu Überlastungen im Netzwerk kommen. Dabei sinkt die Leistung rapide ab. Weitere Parameter sind Störungen von aussen, Ausfälle, Rerouting oder wenn Paketverlust. Wenn der Sender die gestörten Pakete nochmal sendet, kann dies ganze Segmente blockieren.



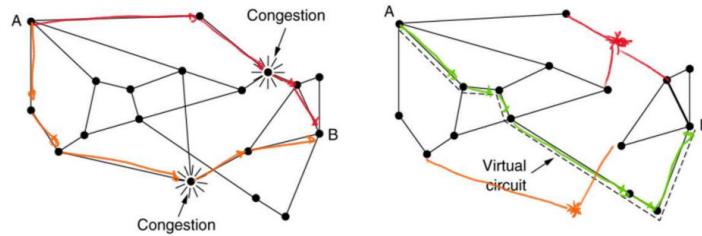
Um eine Überlastung zu verhindern muss das gesamte System überwacht werden. Informationen im Netzwerk müssen weitergegeben und anhand diesen muss der Systembetrieb angepasst werden.

1. Überwachung des Systems, Erkennen von Überlastungen: wann und wo
 2. Weitergabe dieser Information an die Stellen, die Gegenmassnahmen ergreifen können
 3. Anpassung des Systembetriebs, um das Problem zu korrigieren:
 - Reduzierung der Last, oder:
 - Erhöhung der Ressourcen
- 1-2-3-1-2-3- ... → Regelschleife (Feedback-Loop)

Wenn eine Überlastung auftritt können diverse Massnahmen auf den verschiedenen Schichten getroffen werden. Man sieht anhand der Tabelle das im Layer 2 und im Layer 4 die selben Massnahmen getroffen werden können. Weiter sind einige Massnahmen nur bei einer verbindungsorientierter Übertragung möglich.

Transport-Layer (Layer 4)	- Timeout für erneute Übertragung - Zwischenspeichern von Out-of-Order Paketen - Bestätigungen (einzeln oder gesammelt) - Flusskontrolle (straff oder locker)
Vermittlungs-Layer (Layer 3)	- Warteschlangen für Pakete - Verwerfen von Paketen - Routing-Algorithmen - Management der Paket-Lebensdauer
Sicherungs-Layer (Layer 2)	- Timeout für erneute Übertragung - Zwischenspeichern von Out-of-Order Paketen - Bestätigungen (einzeln oder gesammelt) - Flusskontrolle (straff oder locker)

Bei virtuellen Verbindungen können überlastete Verbindungen einfach frisch ausgelegt werden und somit wird der überlastetet Bereich umgangen (Congestion – Stau).



Kann die Überlastung nicht eingedämmt werden, können betroffene Router die Pakete einfach verwerfen. (Lastabwurf). Dieser Abwurf passiert entweder nach dem Zufallsprinzip, es werden die ältesten oder neusten Pakete abgeworfen. Eine andere Option ist es, wenn diese Information vorhanden ist, weniger wichtige Pakete zu verwerfen.

Flusskontrolle (Datenflussteuerung, Data Flow Control)

Dies ist eine Methode der Sicherungsschicht und der Transportschicht. Hierbei ist das Problem das der Sender schneller Daten verschickt als der Empfänger verarbeiten kann. Es gibt hierbei zwei Lösungsansätze:

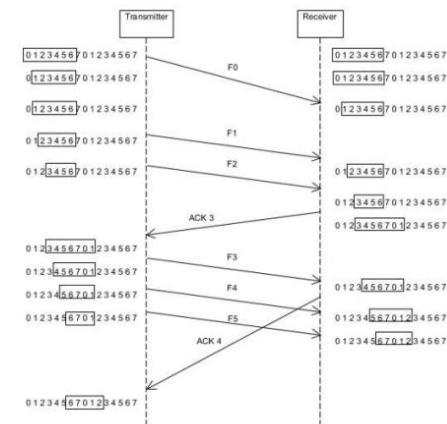
- Die Feedback-basierte Flusskontrolle in dem der Empfänger eine Information dem Sender zurückschickt.
- Die Flusskontrolle basiert auf der Übertragungsrate (Bestätigung, Sliding-Window)

Die Flusskontrolle kann Hardware basierend oder Software basierend sein. Hardware Flow Control nutzt dabei die RTS / CTS Protokolle während Software Flow Control die XON / XOFF Protokolle verwendet.

Hardware Flow Control RTS / CTS Protokoll	Software Flow Control XON / XOFF Protokoll
<p>RS232 Pinout (9 Pin Male)</p> <ul style="list-style-type: none"> • RTS (Request To Send) und CTS (Clear To Send), verwendet bei RTS flow control • DTR (Data Terminal Ready) und DSR (Data Set Ready), verwendet bei DTR flow control 	<p>Im ASCII-Zeichensatz sind die ersten 32 Zeichen für Steuerungsaufgaben reserviert.</p> <p>Vier davon, DC1 bis DC4 (Device Control), sind Gerätesteuerzeichen.</p> <p>Die Software-Flussteuerung verwendet folgende Zeichen:</p> <ul style="list-style-type: none"> • DC1 (oft als XON bezeichnet, engl. für <i>Transmission ON</i>, Zeichencodierung 11_{hex} bzw. 17_{dec}, Tastatur: Strg-Q) • DC3 (oft als XOFF bezeichnet, engl. für <i>Transmission OFF</i>, Zeichencodierung 13_{hex} bzw. 19_{dec}, Tastatur: Strg-S). <p>Diese Zeichen sind sowohl in Richtung Endeinrichtung zum Übertragungsgerät als auch umgekehrt nutzbar.</p>

Die Flow Control mit dem Sliding Window Protokoll basiert auf der Übertragungsrate, sie einigen sich im Voraus auf die Übertragungsrate. Es wird nur im TCP – Protokoll verwendet

- Es wird zu Beginn eine begrenzte Anzahl Meldungen gesendet.
- Dann muss mit einer Quittung der Sender wieder freigegeben werden
- Alternativ kann der Sender auch mit einer negativen Quittung gebremst werden, wenn der Empfänger die gesendeten Daten nicht verarbeiten kann
- Wird bei TCP und HDLC (High-Level Data Link Control → OSI L2) eingesetzt.



Dienstgüte, Quality of Service (QoS)

Bandwidth = Bandbreite, Delay = Verzögerung, Jitter = Schwankungen, Loss = Verlust

Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

Es gibt verschiedene Wege die Dienstgüte zu gewährleisten. Man kann Ressourcen im Überfluss bereitstellen oder mit Zwischenpuffern arbeiten (was auch Ressourcenintensiv ist).

Auch kann man die Ressourcen zeitlich einplanen und reservieren oder das Senden der Pakete planen (Paket-Scheduling). Eine wichtige Option ist das Traffic Shaping mit dem Leaky Bucket und dem Token Bucket Algorithmus.

Zwischenpuffer

Bei Zwischenpuffern werden die Pakete beim Empfänger zwischengespeichert und erst dann wiedergegeben, wenn man einen sauberen Stream hat.

Traffic Shaping

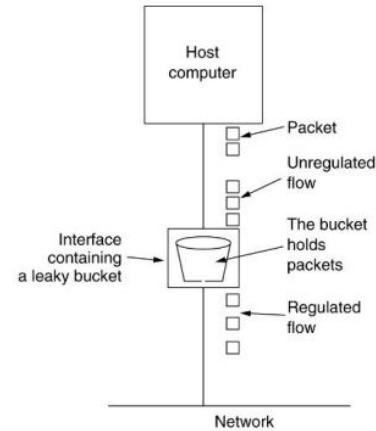
Beim Traffic Shaping versucht der Host sich dem Empfänger anzupassen. Dabei werden bestimmte Regeln und Richtlinien angewendet, um den Datenverkehr innerhalb eines Netzwerks entsprechend festgelegter Kriterien zu formen. Traffic Shaping ermöglicht es, den Datenverkehr zu priorisieren, Bandbreite zu begrenzen oder zu reservieren und die Übertragungsgeschwindigkeit für verschiedene Datenströme zu steuern.

Leaky Bucket Algorithmus

Beim Leaky Bucket Verfahren werden ausgehende Pakete in den Bucket befördert. Dieser Bucket hat eine konstante Ausgabe von Paketen. So kann der Datenstrom reguliert werden und es entsteht ein konstanter Datenstrom.

- Alle Pakete werden in eine FIFO-Schlange (First In, First Out) eingesortiert → Queue
- Dieser Bucket hat eine gewisse definierte Grösse (Kapazität). Sobald er voll ist werden alle folgenden Pakete verworfen.
- Es gibt eine feste Rate welche beschreibt wie viele Pakete pro Sekunde den Bucket verlassen.

Der Empfänger der Daten kann die Ausgangs-Rate aus dem Bucket festlegen. Dadurch kann verhindert werden, dass das Netzwerk unnötig ausgelastet wird. Es wird nur so schnell gesendet wie empfangen werden kann.

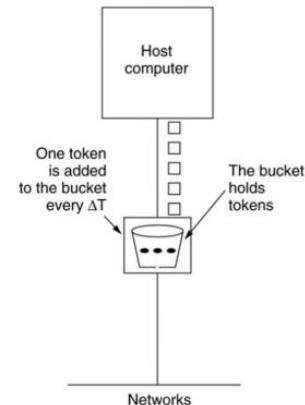


Token Bucket Algorithmus

Beim Token Bucket ist es ziemlich ähnlich wie beim Leaky Bucket. Hier gibt es jetzt einen Bucket mit Tokens darin. Diese Tokens sind an das Senden von Paketen gebunden. Es gibt eine maximale Anzahl von Tokens, die der Bucket fassen kann. Es werden regelmässig neue Tokens generiert.

Jedesmal wenn ein Paket verwendet wird, wird eines der Tokens verwendet. Wenn der Bucket leer ist, wird entweder gewartet oder die Pakete werden verworfen. Die Tokens im Bucket werden konstant erneuert, allerdings nur in einer gewissen Rate.

Der Unterschied zum Leaky Bucket ist hier, dass Bursts von Paketen möglich sind, jedoch die Durchschnittsraten geregelt wird.



Ressourcen reservieren

Bei diesem Verfahren werden gewisse Ressourcen für die Verbindung zwischen Sender und Receiver reserviert. Diese fixierten Ressourcen (Bandbreite, Pufferspeicher, CPU) werden zum Senden verwendet. So kann man die Datenrate kontrollieren.

NW.08 - Sicherheit

Die Hauptaspekte der Sicherheit sind:

- Sicherheit der persönlichen Daten vor Einsicht und Missbrauch (Datenschutz)
- Sicherheit vor Datenverlust
- Sicherheit vor Datendiebstahl (Industriespionage, Internetkriminalität)
- Sicherheit und Zuverlässigkeit der Informationsinfrastruktur
 - Sabotage, Blockierung, Überlastung
 - Diebstahl oder unerlaubte Benutzung von Ressourcen
 - Technische Defekte, Software Bugs

Die daraus resultierenden Bereiche der Sicherheit sind:

- Geheimhaltung, Vertraulichkeit
- Authentifizierung, Autorisierung
- Nichtabstrebbarkeit, Verbindlichkeit
- Integrität

Bedrohungen

Es gibt verschiedenste Bedrohungen welche die Sicherheit und die Bereiche der Sicherheit beeinflussen können.

DoS (Denial of Service) oder DDoS (Distributed Denial of Service)

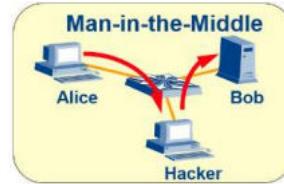
DoS steht für "Denial of Service" und bezieht sich auf einen Angriff, bei dem ein Angreifer versucht, einen Dienst, eine Website oder ein Netzwerk zu überlasten, um ihn für berechtigte Benutzer unzugänglich zu machen. Dabei wird meistens eine einzelne Quelle verwendet, um den Angriff durchzuführen.



DDoS steht für "Distributed Denial of Service" und ist eine weiterentwickelte Form des DoS-Angriffs. Hierbei greift der Angreifer nicht nur von einer einzigen Quelle aus an, sondern nutzt ein Botnetzwerk, das aus vielen kompromittierten Computern besteht. Diese Computer, auch "Zombies" genannt, werden ferngesteuert und senden gleichzeitig eine große Anzahl von Anfragen an das Ziel, um es zu überlasten und den Dienst zu beeinträchtigen.

MITM - Man in the Middle

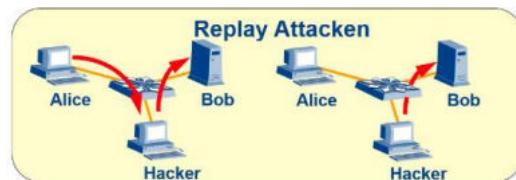
Beim Man in the Middle Angriff versucht der Angreifer zwischen zwei Kommunikationspartnern die Kommunikation abzuhören. Der Angreifer versucht also komplett die Kontrolle über den Datenverkehr zu verhälten. So kann er Informationen nicht nur einsehen sondern auch kontrollieren.



- Vortäuschen eines falschen WLAN Access Point bei öffentlichen WLAN. Mit Tools wie Ettercap, Cain & Abel
- Keylogger: Hardware, Software (Spyware), Akustisch (Richtmikrofon), elektromagnetisch, optisch (Kamera)
- Phishing: Eine Website wird fast identisch nachgebaut. Nur werden dann die Login oder auch weitere Informationen für den Angreifer ersichtlich gespeichert.

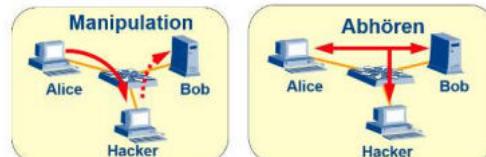
Replay Attacken

Der Angreifer zeichnet die laufenden Verbindungen auf, um diese später dann abzuspielen, inklusive Passwörter. So wird die Authentizität des Datenursprungs angegriffen. Eine Gegenmassnahme dagegen sind Sitzungs-Token oder Laufvariablen in den IP-Headern (Beispiel IPsec).



Manipulationen, Abhören, Sniffer

Das Trojanische Pferd ist ein typisches Beispiel für die Manipulation. Auch könnte eine Spionagesoftware eingespielt werden, um gezielt nach Daten zu suchen.



Diverse weitere Bedrohungen

Trojaner sind Programme welche als nützliche Anwendungen getarnt sind, im Hintergrund läuft aber ohne Wissen des Anwenders ein Programm welches eine andere Funktion erfüllt und gewährt zum Beispiel Eindringlingen eine Hintertür (Backdoor).

Der **Virus** ist ein sich selbst verbreitendes Programm, es infiziert ausführbare Dateien und verbreitet sich durch die Weitergabe dieser Dateien.

Der **Wurm** verbreitet sich über das Netzwerk, dafür wird aber eine Wirtsapplikation benötigt. Weiter ist auch ein Netzwerkdienst oder eine Benutzerinteraktion benötigt.

SPAM sind unerwünschte und in Massen versandte E-Mails. Oft werden durch diese Viren, Würmer, Trojaner, etc. verteilt.

Der **Dialer** ist veraltet, er war ein Programm eine betrügerische Telefonnummer verwendet um sich über das analoge Telefon oder ISDN mit dem Internet zu verbinden.

Bei der **Brute-Force Attacke** wird das System brutal ausgetestet. Es ist ein erschöpfendes Ausprobieren aller (oder vieler) mögliche Passwörter. Dabei können Wörterbücher zum Einsatz kommen in dem alle möglichen Wortkombinationen ausprobiert werden. Oder die Hash-Funktion wird direkt berechnet, dazu kann eine Rainbow-Table zu Hilfe gezogen werden in der eine Hash Funktion abgespeichert wird sobald das Passwort bekannt ist und so kann dann das Passwort nachgeschlagen werden.



IMPORTANT!

- Ein Virus ist ein Stück Programmcode innerhalb eines anderen Programms
- Trojaner spionieren Menschen aus; sie geben vor, eine nützliche Funktion zu haben
- Spam ist Werbung, die keinen direkten, sondern nur ökonomischen Schaden anrichtet
- Ein Wurm breitet sich unabhängig aus. z.B. dadurch, dass er sich an alle Kontakte versendet
- Eine Virus-Signatur ist ein Strukturmuster, an dem man Schadsoftware erkennen kann

Sicherheitsmechanismen

Um die Sicherheit bei der Übertragung zu gewährleisten müssen Sicherheitsmaßnahmen ergriffen werden. Der simpelste Schutz ist der Zugriff von aussen z.B. über eine Firewall. Auch wichtig ist es die richtigen Protokolle und Verschlüsselungstechniken anzuwenden. Um Daten-Integrität und die Authentifizierung zu gewährleisten gibt es diverse Tools und Hilfen: Biometrischer Scanner, spezielle Authentifizierungsprotokolle und Zweifaktor-Authentifizierung (wie SecureID: Ein Zufallszahlengenerator, der synchron zu einem ReferenzServer Key-Tokens erzeugt, die zur Authentifizierung eingegeben werden).

Die beiden wichtigsten Authentifizierungsprotokolle sind **RADIUS (Remote Authentication Dial-In User Service)**, welches für Einwahlverbindungen über Modem, ISDN, VPN, WLAN und DSL Standard ist und **KERBEROS**, dass ein Authentifizierungsprotokoll für offene und unsichere Computernetzwerke ist. z.B MS Active Directory. Die Authentifizierung findet beidseitig statt. Also von Client zu Server und von Server zu Client.

Weiter kann mit Hilfe von Kryptografie die Verbindung verschlüsselt und somit sicherer gemacht werden. Eingesetzte Algorithmen und Protokolle sind:

- IPSec (IP Security), welches das IP-Protokoll um Vertraulichkeit, Authentizität und Integrität erweitert
- SSL (Secure Sockets Layer), was ein Verschlüsselungsprotokoll für Datenübertragungen ist (HTTPS, POP3, SMTPS, FTPS)

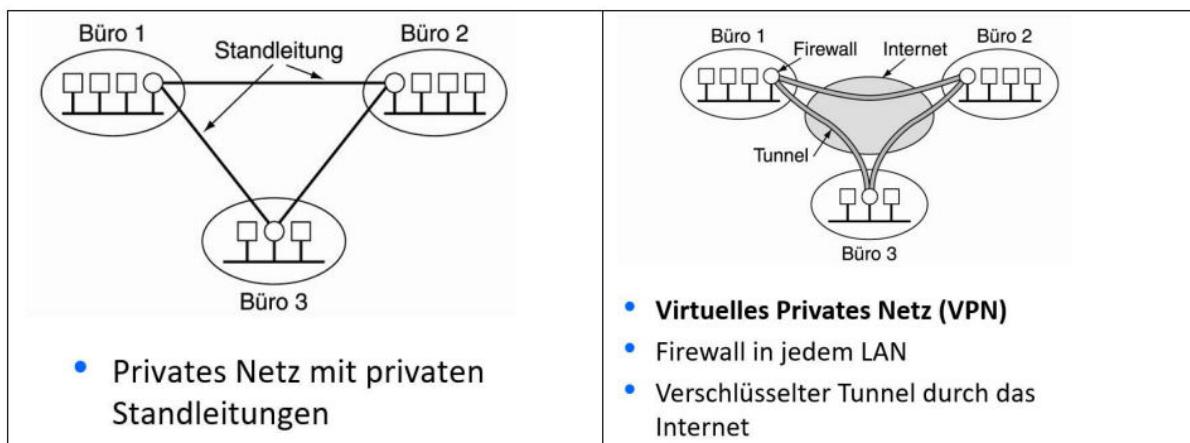
Man kann jedoch schon mit einfache Mitteln sein Netzwerk sicherer machen.

- Nur sichere Software einsetzen (offizielle Tools von Unternehmen in diesem Bereich)
- Regelmässige Software-Updates um Sicherheitslücken zu schliessen
- Physischen Zugang zum Netzwerk beschränken mittels Zugangskontrolle
- Aktuelle Anti-Viren Software einsetzen und Virensignaturen aktuell halten
- Datensicherung gegen Datenverlust durch fehlerhafte Software etc.

VPN - Virtuelles Privates Netzwerk

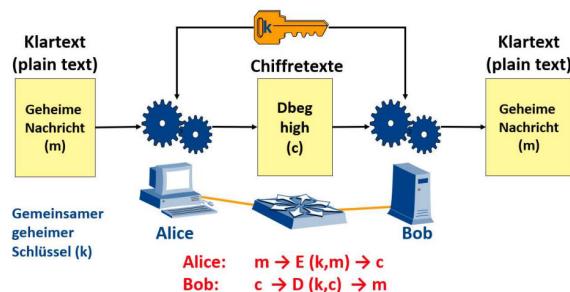
Bei einem Virtual Private Network, kurz VPN, handelt es sich um ein virtuelles Netzwerk: Anders als etwa bei Ihrem Heimnetzwerk sind die verschiedenen Endgeräte bei einem VPN nicht direkt physisch miteinander oder mit einem zentralen Router verbunden – etwa über Netzwerkkabel oder eine WLAN-Anbindung.

Beim Arbeiten mit einem VPN wird intern eine neue Quell-Adresse zugewiesen. So wird die Orginale IP-Adresse der Quelle verschleiert auf dem Zielsystem.



Verschlüsselung, Kryptografie, Ciphering

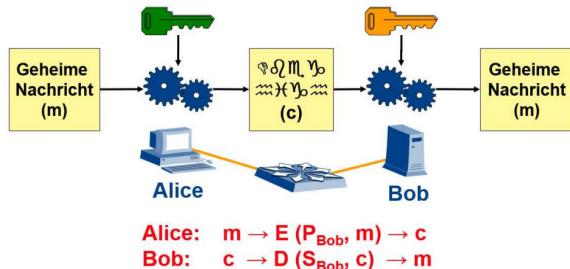
Das **Symmetrische Verschlüsselungsverfahren** beruht auf einem einzelnen privaten Schlüssel, beide Stationen haben diesen Schlüssel und können somit die verschlüsselte Nachricht entziffern. Das Hauptproblem dabei ist das dieser private Schlüssel beide Stationen benötigen und so beim Austauschen in die falschen Hände geraten könnte. Somit könnte dann jeder welche diesen Schlüssel hat auch die Daten auslesen.



Die meist verwendeten symmetrischen Verschlüsselungsverfahren sind:

- Data Encryption Standard (DES)
- Triple DES (3DES)
- RC5 (Rivest Cipher) von RSA
- Advanced Encryption Standard (AES)

Beim **Asymmetrischen Verschlüsselungsverfahren** gibt es immer zwei unterschiedliche Schlüssel, ein sogenanntes Schlüsselpaar aus dem öffentlichen Schlüssel (Public Key) und dem privaten Schlüssel (Private, Secret Key). Dabei wird der Public Key, veröffentlicht und jeder kann seine Nachricht damit verschlüsseln. Allerdings kann dann diese Nachricht nur gelesen werden, wenn man den dazu passenden privaten Key hat.



Beispiel

1. Primzahlen werden gewählt, $p = 11$, $q = 17$
 $n = p * q = 187 \rightarrow (n) = (p-1) * (q-1) = 10 * 16 = 160$
2. $e = 7$ (klein gewählt), so dass $\text{ggT}(e, (n)) = \text{ggT}(7, 160) = 1$
3. Suche d mit $(7 * d) \% 160 = 1 \rightarrow d = 23$ (geheim)
4. Public Key von Alice: $n=187$, $e=7$
5. Bob verwendet public key von Alice
6. Chiffriert ASCII "X" = 88 → $c = (88 ^ 7 \% 187) = 11$
7. $p = (c ^ d) \% n = (11 ^ 23 \% 187) = 88$

Hashalgorithmen

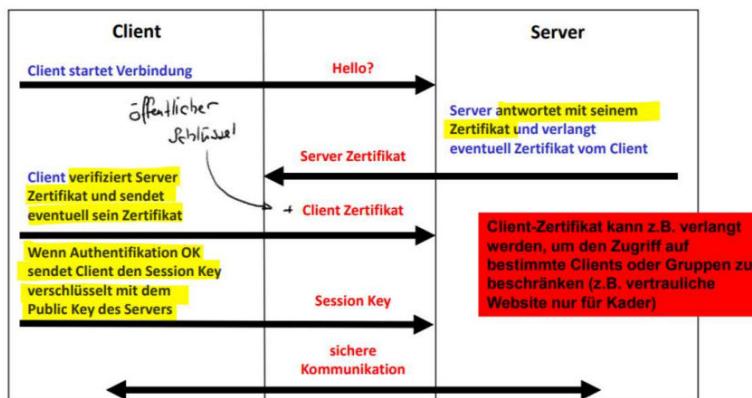
Hashalgorithmen sind mathematische Funktionen die für den Rechner einfach zu rechnen sind, jedoch nur schwer umzukehren. Passwörter zum Beispiel werden in Datenbanken gehashed gespeichert. So stehen keine Clear-Passwörter in einer Datenbank. Auch wenn jemand Zugriff auf die Datenbank erhalten würde, könnte er die Passwörter nicht herauslesen.

Dies wird auch bei der Datenübertragung angewendet, damit überprüft werden kann, ob eine Sequenz von Daten korrekt angekommen ist. Dabei gibt der Sender den Hashalgorithmus an und den errechneten Hashwert. Der Empfänger kann diesen Hashwert nun selber errechnen und schauen ob die Daten so wie vom Sender angegeben empfangen wurden. Dies kann uns helfen Manipulationen durch eine MITM Attacke zu identifizieren.

Im WWW Bereich werden Keyed Hash Funktionen für die Sicherheit verwendet. Diese sind in den Zertifikaten gespeichert.

Das Secure Socket Layer (SSL) und Zertifikat

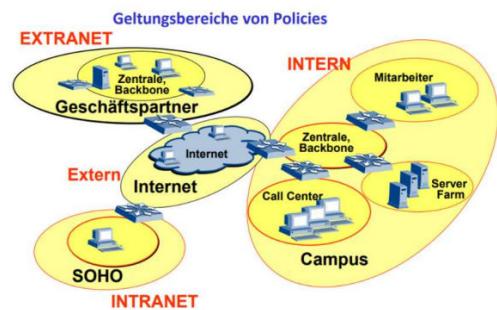
Hier handelt es sich um eine sichere Verbindung von einer z.B. einer HTTPS Site



Security Policy

Die Security Policy ist ein Regelwerk welches den Umgang mit Sicherheit unter ganzheitlichen Gesichtspunkten beschreibt. Es muss regelmässig überprüft und angepasst werden.

Eine Security Policy definiert zur Verfügung gestellte Dienste (Internet, Intranet, Mail, Datenbanken), es definiert die erlaubten Nutzer dieser Dienste (Intern, Extern, Gruppen) und die dazugehörigen Technischen Massnahmen (verwendete Technologie, Regelmässige Funktionsprüfung).



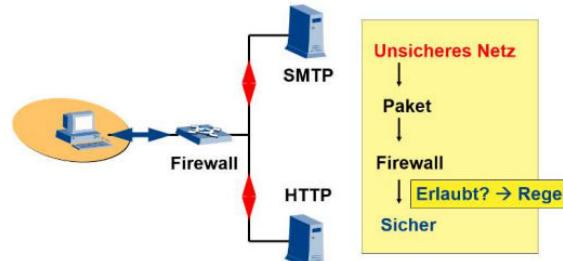
Policies werden oft genutzt um in unter Benutzern einer Rolle weiter Berechtigungen zu spezifizieren.

Policies legen fest auf welche Ressource welche Berechtigungen vergeben werden. Die Personen (User) die diese Policy besitzen, können dann die von der Policy definierten Aktionen auf diesem System ausführen.

Die Firewall

Firewalls arbeiten auf allen Layern. Die typischen Firewalls, welche Netzwerkregeln und Policies verwalten, arbeiten auf Layer 3 und 4. Sie steuern:

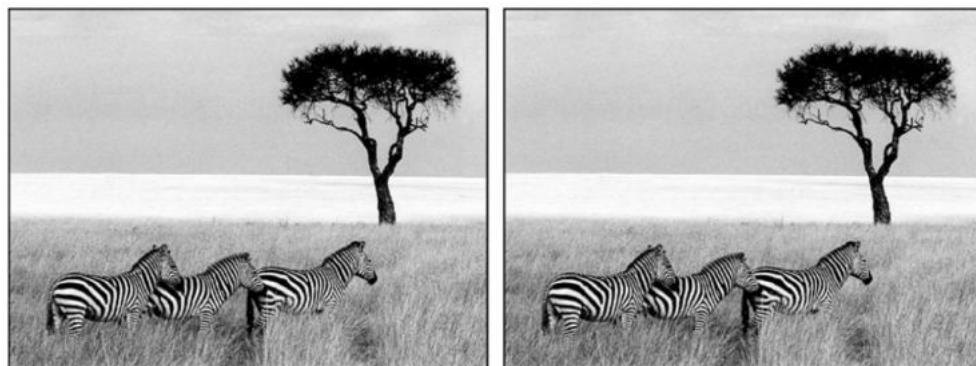
- Welche Pakete passieren dürfen
- Von welchem Absender
- An welchen Empfänger
- An welchen Port
- Mit welchem Protokoll



Die Firewall als Eingangstor von einem Netzwerk überwacht das gesamte Netzwerk und kontrolliert und analysiert die eingehenden Pakete und leitet diese weiter sollten dies erlaubt sein. Eine Firewall kann auch als Triage verwendet werden in dem sie nur die Daten an die dafür vorgesehenen stellen leitet.

Steganografie

Als Steganografie bezeichnet man das verstecken von Informationen in einer Datei.



Zum Beispiel sehen wir auf dem linken Bild 3 Zebras und einen Baum. Das glebe gilt für das rechte Bild. Was wir nicht sehen, ist dass im rechten Bild der vollständige Text von fünf William Shakespeare Stücken steckt.

NW.09 - Die letzte Meile & Satelitenkommunikation

Als letzte Meile bezeichnet man den Anschluss eines Teilnehmers an die Telekom Infrastruktur. Es ist die Brücke zwischen dem zahlenden Kunden und dem Dienstanbieter. Diese "letzte Meile" bietet die Transportkanäle für die verschiedensten Informationsdienste und für die Signalisierung. Nach der OSI Referenzarchitektur ist sie somit beim Layer 1 und 2 einzuordnen.

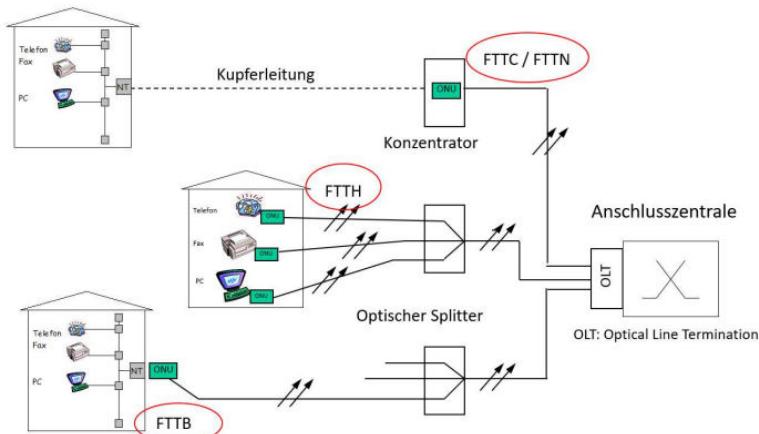
Neue Verfahren und Technologien ermöglichen heute der Kupferleitung den Einsatz als Breitbandschlusstechnologie (xDSL).

Technologie	Bitrate [Mbit/s]	Distanz [km]
Analoges Telefon (POTS)	300 Hz – 3,4 kHz	> 6 km
DSL (ISDN)	0.16, symmetrisch	5.4
HDSL	2, symmetrisch	4
ADSL	2 / 0.6 6 / 1.2	5.4 3.6
VDSL / VDSL2	bis 14 bis 28 50 bis 100	1.5 1 0.3 - 1

Architektur der physischen Kanäle

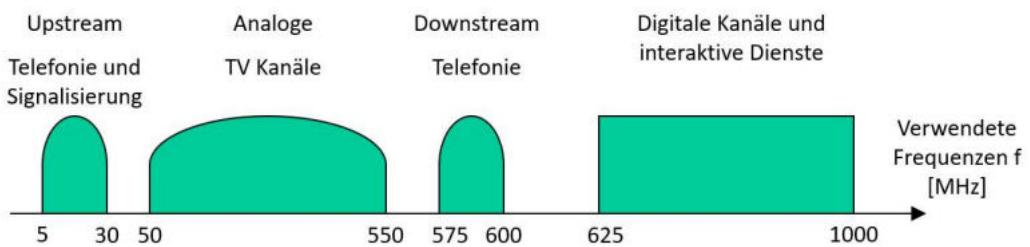
Je nach Platzierung der optischen Netzabschlusseinheiten ONU werden unterschiedliche Begriffe für die Architektur benutzt.

- FFTC / FTTN: Fiber to the Cabinet / Curb / Neighborhood
- FFTH: Fiber to the Home (Desk)
- FTTB: Fiber to the Building

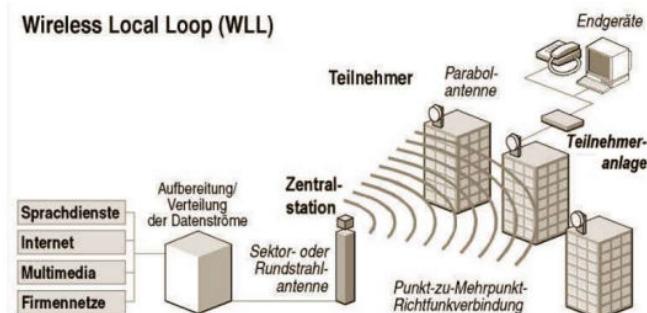


Alternativen zu Kupfer & Glasfaser

Das Kabelfernseh Anschlussnetz der TV Anbieter funktioniert über Koaxialkabel welches mehrere hundert Mbit/s bis Gbit/s Bandbreite bietet. Damit sind verschiedenen Netzwerkarchitekturen denkbar, reine Koaxialkabelnetze oder eines welches mit Glasfaser kombiniert wird (Hybrid Fiber Coaxial-Cable HFC). Das Coaxialkabel bietet die Möglichkeit, mehrere Frequenzen / Kanäle gleichzeitig zu nutzen. Das ganze verwendete Spektrum wird dabei in verschiedene Kanäle oder Frequenzbänder aufgeteilt.



Mit Wireless Local Loop (WLL), werden drahtlose Verbindungen zwischen einem Teilnehmer und einer zentralen Stelle ermöglicht. Es wird somit keine leitergebundenen Infrastrukturen der Swisscom benutzt werden. Grund-Idee wird wieder mit den 5G-Netzen wieder aufgenommen, stationär und auch mobil. Wird auch als Backupleitung eingesetzt sollten die Kabelleitung versagen.

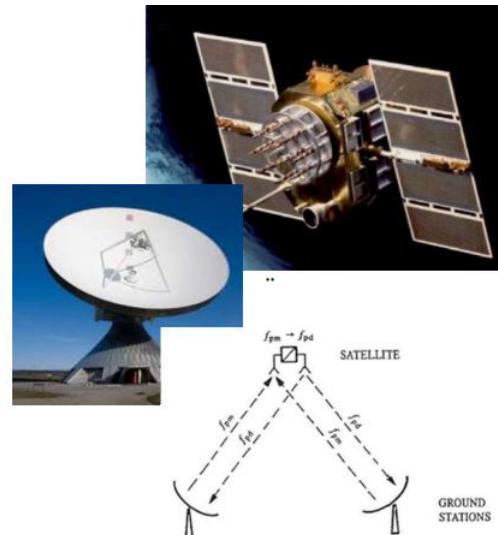


WiMAX (Worldwide Interoperability for Wireless Access) ist ein Standard (IEEE 802.16) für ein Stadtweites drahtloses Netzwerk. Es spezifiziert die Schnittstelle für Punkt-zu-Mehrpunkt Systemen und ist für das IP-Protokoll optimiert. Um diese Schnittstelle weltweit einsetzen zu können, umfasst der Standard verschiedene Optionen. Aktuell wird dieses Protokoll vor allem an Flughäfen eingesetzt für die Kommunikation zwischen und mit den einzelnen Flugzeugen. (AeroMACS Aeronautical Mobile Airport Communication System) – wird zunehmend von LTE verdrängt.

Die Power Line Communication (PLC) dient als Access Technologie für eine Inhouse Vernetzung. Sie wird in die «Schmalband»-PCL-Technik mit einem Frequenzband von 9kHz-148.5 kHz und einer Datenrate von ca. 150kbit/s. Die «Breitband»-PCL-Technik hingegen nutzt ein Frequenzband von 1.6MHz – 30MHz und soll eine wesentlich höhere Datenrate erlauben. Diese Technik nutzt für die Datenübertragung das Erdkabel, es wird vor allem im Mittel- und Niederspannung Stromnetz eingesetzt.

Satelliten-Netz

Auch die Satelliten in der Umlaufbahn der Erde müssen kommunizieren und Daten austauschen können. Das Einsatzgebiet von solchen Satelliten ist umso Umfangreicher. Erdbeobachtungssatelliten, Nachrichtensatelliten, Fernsehsatelliten, Astronomiesatelliten, Forschungssatelliten, Raumstationen und Navigationssatelliten.



Die verschiedenen Satelliten haben auch dementsprechend verschiedene Umlaufbahnen

- Geostationary Earth Orbiter (GEO)
 - Ca 36'000 km von der Erde entfernt
 - «aufgehängt» an einer festen Position über dem Äquator
 - Der Sender und Empfänger auf der Erde kann dauerhaft auf den Satelliten ausgerichtet werden
- Medium Earth Orbiter (MEO)
 - Erdumlaufende Systeme
 - Flughöhe zwischen 10'000 km und 15'000km
- Low Earth Orbiter (LEO)
 - Erdumlaufende Systeme
 - Flughöhe zwischen 700km und 1'500km

NW.01 - NW.09 Formeln & Berechnungen

Fluktuation bestimmen (Datenübertragung)

Fluktuationen sind einfach gesagt "Anomalien" oder "Schwankungen" in der Datenübertragungsrate.

$$\text{Fluktuation (B)} = \frac{\text{Spitzenwert}(S)}{\text{Durchschnittswert (E)}} \rightarrow B = \frac{S}{E}$$

Der minimale Wert, den die Fluktuation B erreichen kann ist 1. Dann ist die Datenübertragung konstant und es gibt keine Fluktuationen.

Latenzzeit bestimmen

$$\text{Latenz} = \frac{\text{Distanz}}{\text{Ausbreitungsgeschwindigkeit}} + \frac{\text{Paketgrösse}}{\text{Durchsatz}} + \text{Wartezeit}$$

Beispiel: Latenzzeit eines 20km Glasfaserkabel

Länge: 20km,

Ausbreitungsgeschwindigkeit: 200'000 km/s,

Paketgrösse: 100 Bit,

Durchsatz: 100'000 Bit/s

$$\text{Latenz} = \frac{20 \text{ km}}{200'000 \frac{\text{km}}{\text{s}}} + \frac{100 \text{ Bit}}{100'000 \frac{\text{Bit}}{\text{s}}} + 0 = 0.1 \text{ ms}$$

Bitfehlerrate bestimmen

Wird verwendet, um die Rate der fehlerhaften Bits einer Übertragung auszurechnen.

$$\text{BER} = \frac{\text{Anz. fehlerhafter Bits}}{\text{Gesamtzahl der gesendeten Bits}}$$

Paketfehlerrate bestimmen

Wird verwendet, um die Rate der fehlerhaften Pakete zu berechnen.

$$\text{PER} = \frac{\text{Anz. der fehlerhaften Pakete}}{\text{Gesamtzahl gesendeter Pakete}}$$

Bandbreite bestimmen

B = Bandbreite, f = Frequenz

$$B = f_{max} - f_{min}$$

Maximale Datenübertragungsrate bestimmen

Allgemein:

Mit Rauschen:

$$R = 2 \cdot B \cdot \log_2 N$$

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

R: Übertragungsrate, Datenrate in Bit/s

B: Bandbreite der Leitung in Hz

N: Anzahl diskrete Signalstufen

S/N: Rauschabstand (Signal-to-Noise)

Fehlererkennung und Korrektur mit Hamming-Abstand

Fehlererkennung, für Erkennung von e Bitfehlern

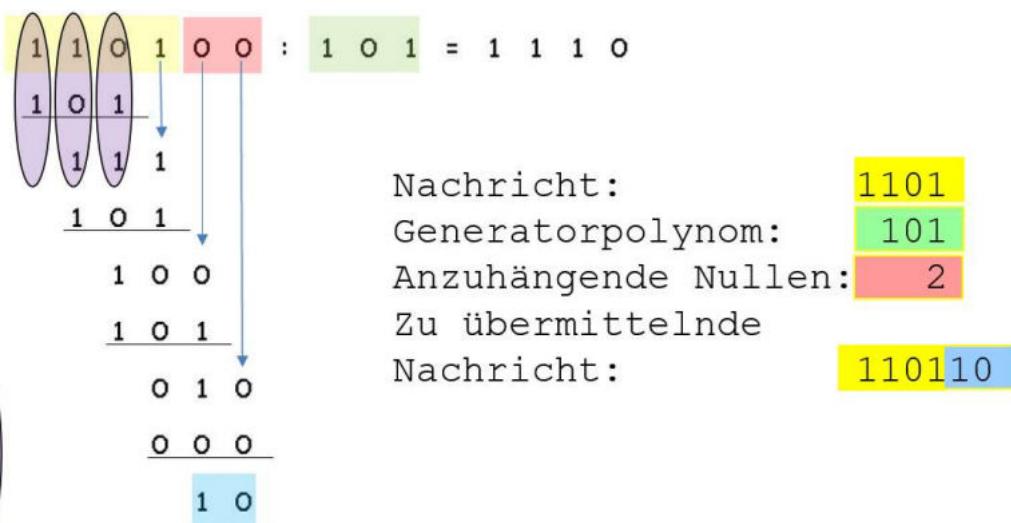
$$h = e + 1$$

Fehlerkorrektur, für Korrektur von e Bitfehlern

$$h = 2 \cdot e + 1$$

CRC Bestimmung mittels Polynom-Division

CRC Bestimmung mittels Polynom Division: Die Rechnung wird nach den Regeln der algebraischen Körpertheorie durchgeführt. Ohne Borgen und Übertragen. Das heisst anstatt Subtraktion sind es einfache EXOR Verknüpfungen an jeder Stelle und die erste führende Null pro Resultatzeile wird weggelassen.



Token Bucket Berechnungen

$$T_{Senden} = \frac{\text{Initial Tokens}}{\text{Abfluss} - \text{Zufluss}}$$

Warteschlangen Berechnung

$$T = \frac{1}{\mu} \cdot \frac{1}{1 - \frac{\lambda}{\mu}} = \frac{1}{\mu} \cdot \frac{1}{1 - \rho}$$

$\rho = \frac{\lambda}{\mu}$ = die CPU Auslastung

λ = Anfragerate, Ankunftsrate

μ = Abarbeitungsrate, Verarbeitungsrate

Weitere Informationen

Mehr Zusammenfassungen, Übungslösungen, etc sind auf Github.

<https://github.com/omeldar/hslu>

Wenn ihr einige der Dokumente nützlich gefunden habt dann markiert doch das Repository mit einem Stern ;) So wird es vielleicht anderen Informatikstudenten ebenfalls vorgeschlagen.