Rechnernetze I

Zusammenfassung wichtiger Elemente der Einführung in die Netzwerktechnik

von Alexander Köster

Student der Universität Siegen, RN1 2020

Letzte Aktualisierung: 21. September 2020

Dieser Kurs der Informatik befasst sich mit den Grundlagen, Bausteinen, Modellen und Protokollen der Netzwerktechnik klassischer Rechnernetze.

Einzig für Lernzwecke erstellt.

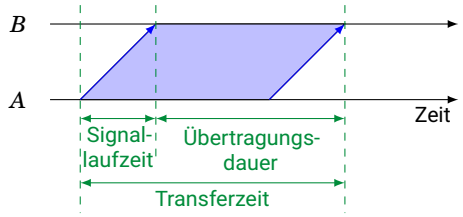
Nicht geeignet als Klausurhilfe.
Das Erstellen einer eigenen Klausurhilfe führt zu einem besonders guten Lerneffekt.
Dieses Dokument sollte nur als Orientierung oder Vergleich dienen.
Jeder sollte seine Klausurhilfe individuell auf seinen Lernstand und seine eigenen Probleme anpassen, gut bekannte Dinge auslassen und schlecht merkbare Dinge hinzufügen.
Dieses Dokument beinhaltet viel mehr, als für eine tatsächliche Klausur durchschnittlich benötigt wird.
Für einen guten Ansatz, welche Inhalte man braucht, sollte sich an der Probeklausur orientiert werden.

1 Einführung

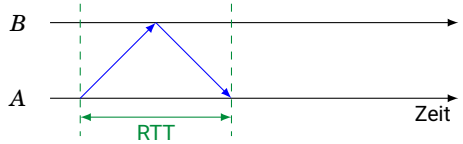
- Klassifikation von Netzwerken nach geographischer Ausdehnung (versch. Technologien):
  - ▷ **SAN**: *System Area Network*
    Hochgeschwindigkeitsnetz innerhalb eines Raumes (Rechenzentrum)
  - ▷ **LAN**: *Local Area Network*
    $\leq 1\ \mathrm{km}$, innerhalb eines Gebäudekomplexes, Bsp.: Ethernet
  - ▷ **MAN**: *Metropolitan Area Network*
    $\leq 10\ \mathrm{km}$, innerhalb einer Stadt
  - ▷ **WAN**: *Wide Area Network*
    länder-/weltumspannend, Bsp.: Internet
- Vermittungsarten
  - ▷ **Leitungsvermittlung** (***circuit switching***):
    Kommunikationspartner sind durch physisch geschaltete Leitung verbunden.
    (vgl.: früheres Telefonnetz)
  - ▷ **Speichervermittlung** (***store-and-forward routing***): Daten von einer Vermittlungsstelle zur nächsten weitergeben, vollständig puffern. (vgl.: Briefe)
  - ▷ **Paketvermittlung** (***packet switching***):
    Daten in Pakete zerteilen, Pakete unabhängig voneinander befördern. (Rechnernetze)
- Anzahl der Empfänger einer Nachricht:
  - ▷ ***Unicast*:** genau einer
  - ▷ ***Broadcast*:** alle (üblicherweise: „alle im lokalen Netzwerk“)
  - ▷ ***Multicast*:** mehrere bestimmte

1.5 Leistungsparameter

- **Bandbreite**
  - ▷ Übertragb. Datenvolumen pro Zeiteinheit
  - ▷ Maßeinheit: Bits pro Sekunde
    (b/s oder bit/s oder bps)
    SI-Einheiten (d. h. $1\ \mathrm{kbit/s} = 1000\ \mathrm{bit/s}$)
    $8\ \mathrm{bit/s} = 1\ \mathrm{B/s}$ (Byte pro Sekunde)
- **Durchsatz** = Transfergröße/Transferzeit
- **Transferzeit**: Zeit von Beginn des Absendens einer Nachricht bis zu ihrem vollständigen Empfang.
  Transferzeit = Signallaufzeit + Übertragungsdauer + Zeit für Pufferung in Zwischenknoten
- **Signallaufzeit** = Entfernung/$c_{\text{im Medium}}$
- **Übertragungsdauer** = Nachrichtengröße/Bandbreite



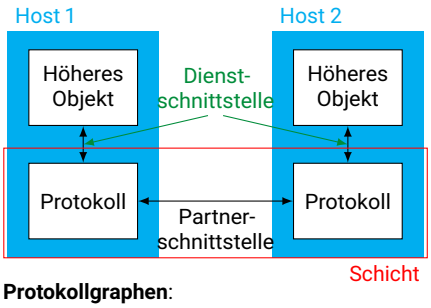
- ***Round-Trip-Time*** **(RTT)**: Zeit, um eine leere Nachricht von $A \to B$ und zurück zu schicken



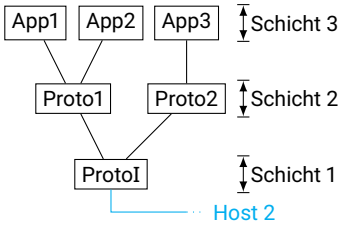
- **Latenz**: (mehrdeutig!)
  - ▷ Var. 1: synonym für Transferzeit
  - ▷ Var. 2: Transferzeit einer leeren Nachricht
    (Signallaufzeit + Pufferung in Zwischenknoten)
- Bandbreite · Signallaufzeit =
  Speicherkapazität einer Leitung
  (vgl.: Wasserschlauch speichert Wasser)

2 Protokolle & Protokollhierarchien

- **Routing:** Bestimmung eines Weges vom Sender zum Empfänger
- Schichtenmodelle:



- **Protokollgraphen**:



- **PDU** (*Protocol Data Unit*): Dateneinheit, die ein Protokoll überträgt. (allgemeiner Begriff)

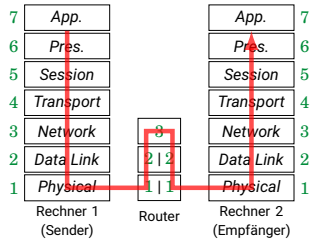
2.3 Die OSI-Architektur

ISO/OSI-Schichtenmodell:

#	…-s-Schicht	… *Layer*	PDU
7	Anwendung	*Application*	-
6	Darstellung	*Presentation*	-
5	Sitzung	*Session*	-
4	Transport	*Transport*	Segment
3	Vermittlung	*Network*	Paket
2	Sicherung	*Data Link*	*Frame*
1	Bitübertragung	*Physical*	-

Details:

1. **Bitübertragungsschicht** (***Physical Layer***):
   - elektrische Spezifikation (Medium, Spannungspegel, Frequenzen, Zeitverhalten, Codierung, Übertragungsrichtung, …)
   - mechanische Spezifikation (Form & Art der Stecker, Kabel, Pins, …)
2. **Sicherungsschicht** (***Data Link Layer***):
   eigentlich heute meist 2 Teilschichten:
   - **Zugriffskontrolle (MAC,** ***Media Access Control*****)**:
     - ▷ physische Adressierung
     - ▷ Zugriff aufs gemeinsame Medium bei Mehrfachzugriffs-Verbindungen (Bsp.: WLAN)
   - **LLC** (***Logical Link Control***):
     - ▷ Fehlerbehandlung, Flusskontrolle
     - ▷ Aufteilung der Daten in *Frames*
3. **Vermittlungsschicht** (***Network Layer***):
   - unterste Schicht der Kommunikation zw. nicht direkt verbundenen Netzwerkknoten (*Host*-zu-*Host*-Kommunikation)
   - oberste Schicht der Netzwerk-Zwischenknoten, Schnittstelle der Subnetze



   - einheitliche logische Adressierung
   - Hauptaufgabe: *Routing*

   Bsp.: IP-Protokoll im Internet
4. **Transportschicht** (***Transport Layer***):
   - Kommunikation zw. Endpunkten (Prozessen) (Ende-zu-Ende-Kommunikation)
   - Prozessadressierung
   - ggf. Verbindungsaufbau (Prozesse erhalten dann den Eindruck einer Leitungsvermittlung, selbst wenn untere Schichten paketorientiert arbeiten)
   - ggf. Sicherung des Datentransports zw. Endpunkten (Fehlerbeh., Flusskontrolle)

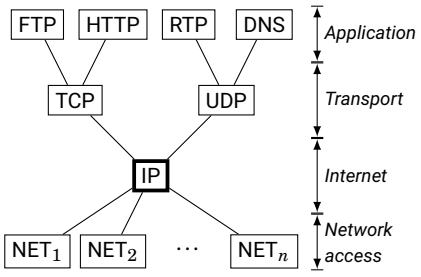
   Bsp.: TCP- & UDP-Protokoll im Internet
5. **Sitzungsschicht** (***Session Layer***):
   - Verwaltung von Sitzungen
     - ▷ Dialogsteuerung
       („wer darf wann senden?“)
     - ▷ atomare Aktionen
       („alles oder gar nichts“)
     - ▷ Synchronisierung, Weiterführung eines unterbrochenen Transfers
6. **Darstellungsschicht** (***Presentation Layer***):
   - konvertiert Datenformate und -darst.
   - Auch: Kompression, Verschlüsselung
   - unterste Schicht, die die Semantik der Daten kennt
7. **Anwendungsschicht** (***Application Layer***):
   Bsp.: HTTP, SMTP, SMB, NFS, SSH, …

2.4 Die Internet-Architektur

Vergleich mit OSI-Architektur:

	OSI	Internet	
7	*Application*	*Application*	
6	*Presentation*		
5	*Session*		
4	*Transport*	*Transport*	TCP, UDP
3	*Network*	*Internet*	IP
2	*Data Link*	*Network access*	
1	*Physical*		

- Schichten 5, 6, 7 werden alle im *Application Layer* (7) behandelt, da die Anwendungszwecke versch. Anwendungen zu verschieden für eine klare Unterteilung sind.
- Schichten 1 & 2 werden im Internet-Modell nicht näher betrachtet, daher nicht unterteilt. („Aufsetzen“ auf bel. Netzwerktechnologien)
- **IP** (***Internet Protocol***): verbindungslos, paketorientiert, unzuverlässig
- Protokollgraph Internet („Sanduhr-Modell“):



- physische Adressen (Sicherungsschicht, OSI 2) im Internet: **MAC-Adresse**
  Jede Netzwerkkarte weltweit hat eindeutige MAC-Adresse. Bsp.: 1a:68:25:f0:a3:d9
- logische Adressen (Vermittlungsschicht, OSI 3) im Internet: **IP-Adresse** (Bsp.: 141.99.179.6)
- Anwendungsschicht (OSI 7): **Hostname** im Internet über DNS (Bsp.: www.example.com)

3 Direktverbindungsnetze

3.1 Hardwarebausteine (OSI 1)

- **NIC**: *Network Interface Controller*
- Richtung des Datenflusses:
  - ▷ **simplex**: nur in eine Richtung
  - ▷ **vollduplex**: beide Richt., gleichzeitig
  - ▷ **halbduplex**: beide Richt., abwechselnd
- **Koaxialkabel**: ein Innenleiter, koaxial umhüllt von einem Außenleiter.
  - ▷ hohe Bandbreite, geringe Dämpf., teuer
  - ▷ **Basisband-Kabel** (direkte elektr. Übertragung, 1 Kanal, $< 500\ \mathrm{m}$) Bsp.: Ethernet 10BASE-5, 10BASE-2
  - ▷ **Breitband-Kabel** (Modulation auf Träger-

welle, mehrere Kanäle, mehrere km) **Bsp.:** Fernsehkabel

- ***Twisted-Pair*-Kabel**: ungeschirmt (UTP) oder geschirmt (S/FTP, F/FTP, S/STP, S/UTP, ...)
  - ▷ gegen äußere Einflüsse: Gesamtkabel abschirmen (1. Buchstabe)
  - ▷ gegen Übersprechen: Adernpaare gegenseitig voneinander absch. (2. Buchstabe)
  - ▷ S: *shield* (Kupfergeflecht), besser F: *foil* (Folie), günstiger

  geringe Kosten, relativ gute Bandbreite. **Bsp.:** Ethernet 100BASE-TX (heute)
- **Glasfaserkabel** (Lichtwellenleiter): Glaskern umhüllt von Glasmantel, versch. Brechungsindex. Trifft Licht entsprechend flach auf die Grenzfläche → Totalreflexion.
  - ▷ Bandbreite: etliche Gbit/s bis Tbit/s
  - ▷ Länge im Bereich km
  - ▷ *Multimode*-Faser vs. *Monomode*-Faser (Laserdioden, hohe Bandbreite, teuer)
- Def.: **Modulation:** Variation von Frequenz (FM), Amplitude (AM) und/oder Phase (PM) einer Welle, um (hier) 0- und 1-Signal darzustellen.

3.3 Codierung (OSI 1)

- einfachste Leitungscodierung: ***Non-Return to Zero* (NRZ)**: $1 \mathrel{\widehat{=}} high$, $0 \mathrel{\widehat{=}} low$ (verschiedene Spannungspegel für *high* & *low*)
  Problem: **Taktwiederherstellung** („Wo ist die Grenze zw. zwei Bits?") bei langen Folgen
- **Manchester-Codierung**: Bits XOR Taktsignal
  - ▷ benötigt doppelte Bandbreite/Abtastrate
  - ▷ **Bsp.:** 10 Mbit/s Ethernet
- ***Non-Return to Zero Inverted* (NRZI)**: Signal bei jedem 1-Bit invertieren
  - ▷ Taktwiederherstellung bei 1-Folgen möglich, aber nicht bei langen 0-Folgen
- **4B/5B-Codierung**: 4 Datenbits werden auf 5-Bit-Codeworte abgebildet, sodass nie mehr als 3 aufeinanderf. Nullen übertragen werden.
  - ▷ max. 1 Null am Anfang, max. 2 am Ende
  - ▷ Übertragung dann z. B. mit NRZI
  - ▷ Overhead 25%

3.4 Framing (OSI 2)

- Gründe für Aufteilen der Daten in *Frames*:
  - ▷ einfaches Multiplexing versch. Kommunikationen
  - ▷ bei Fehler muss nur betroffener *Frame* neu übertragen werden

  Problem: Wann ist ein *Frame* zu Ende?
- ***Byte-Count*-Methode**: *Frame-Header* enthält Länge des Datenteils (*Body*).
  - ▷ Problem: Was passiert, wenn Länge fehlerhaft übertragen wird?
  - ⇒ SYN-Zeichen am Beginn jedes *Frame*s, um Anfang des Folge*frame*s zu finden

  Bsp.: ursprüngliches Ethernet
- ***Sentinel*-Methode**: *Frame*-Ende durch spezielles Zeichen (ETX) markieren
  - ▷ Problem: ETX-Bitmuster kann im Datenteil vorkommen
  - ⇒ ***Byte Stuffing***: ersetze ETX im Datenteil durch DLE ETX, ersetze DLE durch DLE DLE. (DLE = *Data Link Escape*)
  - ⇒ ***Bit Stuffing***: Eindeutigkeit durch Einfügen von Bits in den Bitstrom erreichen

    Bsp.: $\text{ETX} = 01111110_2$. Nach 5 aufeinanderfolgenden 1-Bits wird vom Sender ein 0-Bit eingeschoben; wenn Empfänger 5 aufeinanderf. 1-Bits gelesen hat:
    - nächstes Bit 0: ignorieren, als wäre es nicht da (weil eingeschoben)
    - nächstes Bit 1: wenn danach 0-Bit: ETX erkannt (sonst Fehler).

  ⇒ **Nutzung „ungültiger" Codeworte** (z. B. eines der ungenutzten 16 Codeworte bei 4B/5B-Codierung) für ETX

3.5 Fehlererkennung (OSI 2)

- **Hamming-Distanz** d = max. Anzahl Bits, in denen sich zwei Codeworte unterscheiden
  - ▷ $d \geq f+1 \Rightarrow f$ Bitfehler erkennbar
  - ▷ $d \geq 2f+1 \Rightarrow f$ Bitfehler korrigierbar

  Bsp.: Paritätsbit als Prüfsumme $\Rightarrow d = 2$
- **zweidimensionale Parität**: Erweiterung d. einfachen Parität um Paritätsbyte.
  - ▷ Erkennt alle 1-, 2-, 3- & meiste 4-Bit-Fehler
  - ▷ 1-Bit Fehler korrigierbar (unter Annahme, dass nur 1-Bit-Fehler auftreten):

0	1	0	1	0	0	1	1
1	1	1	1	0	0	1	0
1	0	1	1	1	1	0	1
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1	1	1	1	0	1	1	0

  - ▷ Wird nicht in der Praxis verwendet.
- **CRC (*Cyclic Redundancy Check*)**: siehe study.woalk.de/hapra-ss2018.php
  - ▷ CRC-16-Polynom: $x^{16}+x^{15}+x^2+1$
    Erkennt alle 1-, 2-Bit-Fehler, alle Fehler ungerader Bitzahl, alle Fehlerbündel (aufeinanderf. Fehlerbits) mit Länge ≤ 16 Bit

3.6 Medienzugriffssteuerung (MAC) (OSI 2)

- In vielen LANs: Knoten greifen auf gemeinsames Medium zu; Zugriff muss geregelt werden, um Kollisionen zu vermeiden.

3.6.1 Ethernet

- Früher: **10Base5**: 1 durchgehendes gelbes Koaxialkabel („*The Ether*"), Rechner durch Vampirklemmen alle an dieses angeschlossen.
  - ▷ max. 500 m Koaxialkabelsegmente
  - ▷ Segmente über **Repeater** verbindbar, max. 4 Repeater zw. zwei Knoten erlaubt
  - ▷ Manchester-Codierung
- Heute: **10BaseT/100BaseTx**
  - ▷ *Twisted-Pair*-Kabel, max. 100 m
  - ▷ nur Punkt-zu-Punkt- bzw. sternförmige Verbindungen mit Hubs/Switches:
    - **Hub**: „Verstärker" & Verteiler (OSI 1), sendet ankommendes elektr. Signal ohne Interpretierung auf allen angeschlossenen Leitungen weiter.
    - **Switch**: Vermittlungsknoten (OSI 2), sendet ankommende *Frames* auf die Leitung in Richtung Zieladresse.
  - ▷ 4B/5B-Codierung
- *Frame*-Format:
  1. **Präambel**: feste 56-Bit-Folge zur Taktsynchronisation: $10101010\ldots10101011_2$
  2. **SOF**: *Start-of-Frame*-Byte (*Frame*anfang)
  3. **Zieladresse** (48 Bit)
  4. **Quelladresse** (48 Bit)
  5. **Typ/Länge** (16 Bit):
     - ▷ Wert $< 1536_{10}$ (600_{16}): *Frame*länge (*Legacy* aus altem Xerox-Ethernet)
     - ▷ Wert $\geq 1536_{10}$: *EtherType*, spezifiziert Protokoll im *Body* (bei neuem Ethernet II/DIX Ethernet)

     In diesem Fall muss das *Frame*-Ende auf OSI-Schicht 1 erkannt werden (Fehlen eines Signals bei 10Base5, „ungültiger" 4B/5B-Code bei BaseT).
  6. ***Body* (Nutzdatenteil)**
  7. **FCS** (*Frame Check Seq.*): 32-Bit CRC-Wert
- **Ethernet-Adressen** (**MAC-Adressen**):
  - ▷ 6 Byte (48 Bit) lang, weltweit eindeutig
  - ▷ Schreibweise: byteweise hexadezimal (6 2er-Blöcke) mit Trennzeichen „:" oder „-", niedrigstwertigstes Byte zuerst.
  - ▷ Jeder Netzwerkkartenhersteller erhält 24-Bit-Präfix, vergibt eindeutige Suffixe.
  - ▷ Niedrigstw. Bit = 1: *Multicast*-Adresse
  - ▷ FF:FF:FF:FF:FF:FF *Broadcast*-Adresse
  - ▷ Die Netzwerkkarte bestimmt, welche *Frames* sie empfängt. Speicher von 16 *Multicast*-Adressen, auf die sie hören soll.

3.6.2 CSMA/CD

- Zugangsprotokoll zum gemeinsamen Übertragungsmedium beim Ethernet.
- *Carrier Sense Multiple Access* (CSMA): Jede Netzwerkkarte prüft zunächst, ob die Leitung frei ist, bevor sie ihren *Frame* sendet.
- *Collision Detection* (CD): beim Senden erkennt der Sender Kollisionen mit *Frames*, die andere Netzwerkkarten evtl. gleichzeitig senden. Bei Kollision: Abbruch, JAM-Signal senden, nach einiger Zeit (s. u.) erneuter Sendeversuch.
  Vergleichbar mit menschlicher Unterhaltung!
- ***Exponential-Backoff*-Strategie:** Bestimmung der Wartezeit zw. Kollision und erneutem Sendeversuch. Ziel: erneute Kollision vermeiden.
  - ▷ $c :=$ Anzahl bisheriger Kollisionen
  - ▷ warte $s \cdot 51{,}2\ \mu$s (bei 10 Mbit/s) bzw. $s \cdot 5{,}12\ \mu$s (bei 100 Mbit/s), wobei:
    - falls $c \leq 10$: wähle $s \in \{0, 1, \ldots, 2^c - 1\}$ zufällig
    - falls $c \in [11, 16]$: wähle $s \in \{0, 1, \ldots, 1023\}$ zufällig
    - falls $c = 17$: Abbruch mit Fehler
- max. RTT: 512 Bit-Zeiten ($51{,}2\ \mu$s bzw. $5{,}12\ \mu$s)
  ⇒ max. Ausdehnung: 200 m bei 100BASE-TX mit Hubs
- min. *Frame*länge: 512 Bit (64 Byte), zzgl. Präambel; kleinere *Frames* werden aufgefüllt, damit jede Station min. 1 RTT lang sendet (damit bei Kollision immer beide noch senden).

3.6.3 MAC im Token-Ring

- Jeder Computer hat 2 Netzwerkkarten, alle sind in einem Ring verbunden. *Token* (spezielle Bitfolge) umkreist den Ring. Ein Knoten, der senden will, kann das *Token* „ergreifen", wenn es bei ihm ankommt, und dann senden. *Frame* umkreist den Ring, jeder Knoten reicht ihn weiter, bis er beim Sender wieder ankommt. Dann wird *Token* wieder weitergegeben.
- Mit gegebener *Token*-Haltezeit THT kann jeder Knoten garantiert nach Ablauf der Zeit $TRT \leq \text{Ringlatenz} + (\#\text{Knoten} - 1) \cdot THT$ seinen *Frame* senden.
  ⇒ Eignung für Realzeitanwendungen. (Mit CSMA/CD sind keine derartigen Garantien mgl.)

4 LAN-*Switching*

4.1 Weiterleitungstechniken

- **Datagrammvermittlung:** jeder Vermittlungsknoten besitzt Weiterleitungstabelle (bildet Zieladresse auf Ausgangsport ab).
  - ▷ kein Verbindungsaufbau
  - ▷ Datagramme unabhängig voneinander (können versch. Wege nehmen)
  - ▷ Knoten kann nicht feststellen, ob das Datagramm zugestellt werden kann
  - ▷ Ausfall einer Verbindung kann durch Anpassung der Weiterleitungstabelle theoretisch toleriert werden

  Bsp.: IP (s. Abschnitt 4.3)
- **Virtuelle Leitungsvermittlung** (verbindungsori-

entiert): Aufbau einer virtuellen Verbindung (***Virtual Circuit*, VC**) zw. Quell- & Zielrechner
  - ▷ statisch (*permanent VC*) oder dynamisch (*switched VC*)
  - ▷ Alle Pakete nehmen denselben Weg.
  - ▷ Pakete enthalten Bezeichner des VC (*Virtual Circuit Identifier*, VCI) statt Quell- & Zieladresse, ist nur auf jeweiliger Leitung eindeutig. Weiterleitungstabelle bildet Eingangs-Port & -VCI auf Ausgangs-Port & Ausgangs-VCI ab.

  Bsp.: *Frame-Relay*, MPLS
- ***Switching*/*Forwarding***: Weiterleiten v. *Frames* (Paketen) zum richtigen Ausgangsport
- **LAN-*Switch***: Vermittler im LAN (auf Ebene der Sicherungsschicht/OSI 2)
- **LAN-*Bridge***: LAN-Switch mit nur 2 Ports
- ***Routing***: (dynamischer) Aufbau von Tabellen zum *Forwarding*, Finden von (guten/optimalen) Wegen zu anderen Netzen.
- ***Router***: Vermittler auf Vermittlungsschicht (OSI 3), vereinigt *Routing* & *Forwarding*.

4.2 *Switching*: Einführung (OSI 2)

- *Switch* hat mehrere Ein-/Ausgangsports, sendet ankommende *Frames* auf die Leitung in Richtung Zieladresse im Header.

  ⇒ sternförmige Topologie
- verb. Netze müssen u. a. dasselbe Adressierungsschema haben (heterogene Netze)
- *Switches* bei mehreren Adern (z. B. $100\ \mathrm{Mbit/s}$ Ethernet) erlauben Vollduplexbetrieb.

  Hubs erlauben nur halbduplexbetrieb in der Kollisionsdomäne (Bereich eines Hubs).

4.3 Lernende *Switches* (OSI 2)

- *Switch* untersucht die Quelladresse jedes eingehenden *Frames*, aktualisiert/erzeugt ggf. Eintrag in Weiterleitungstabelle für diese Adresse.

  Wenn längere Zeit kein *Frame* mit dieser Quelladresse kommt, wird dieser Eintrag gelöscht.
- Verhalten beim Eintreffen eines *Frames*:
  - ▷ Quell-*Port* = Ziel-*Port*: *Frame* verwerfen.
  - ▷ Quell-*Port* ≠ Ziel-*Port*: *Frame* an Ziel-*Port* weiterleiten.
  - ▷ Ziel-*Port* unbekannt: *Frame* weiterleiten an alle *Ports* (außer dem Quell-*Port*)

  Broadcast-Frames immer an alle Ports.

4.4 *Spanning-Tree*-Algorithmus (OSI 2)

- Problem bei lernenden Switches: Zyklen!

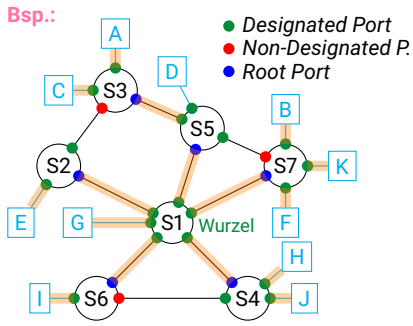
  Frames mit unbekanntem Ziel laufen so ewig im Kreis („*Broadcast*-Sturm“).

  ⇒ Reduktion auf zyklenfreien Graphen durch Deaktivierung einiger *Ports* der *Switches*

  Aber: *Switches* kennen nicht den ganzen Netzwerkgraphen, sondern nur jeweils, was den jeweiligen *Switch* direkt angeschlossen ist!
- Jeder *Switch* hat eindeutige Kennung (z. B. „S1“, „S2“ bzw. MAC-Adresse bei Ethernet-*Switches*)
- **Idee des Algorithmus':**
  - ▷ Wähle den *Switch* mit der kleinsten Kennung als **Wurzel** (*Root Switch*).
  - ▷ Jeder *Switch* bestimmt seinen ***Root Port***:
    - *Port* mit geringster Entf. zur Wurzel
    - bei Gleichheit entsch. *Port*-Prio./-Nr.
  - ▷ Wähle für jedes LAN-Segment den *Port* des *Switches* mit der geringsten Entfernung zur Wurzel als ***Designated Port***.

    Bei Gleichheit entsch. *Switch*-Kennung.
  - ▷ *Ports*, an denen direkt Rechner angeschl. sind, sind immer ***Designated Ports***.
  - ▷ Alle anderen *Ports* sind ***Non-Designated Ports*** und sind blockiert.

Bsp.:



- Auf jeder Leitung min. 1 *Designated Port* (falls ein Mehrfachzugriffsnetz darunter ist)
- In der Praxis wird neben der Anzahl *Hops* auch die Verbindungsgeschwindigkeit etc. für Bestimmung der „kürzesten Verbindung zur Wurzel“ berücksichtigt.
- ***Spanning-Tree*-Protokoll (STP):**
  - ▷ Knoten tauschen Konfigurationsnachrichten (*Bridge Protocol Data Unit*, BPDU) aus:
    - Kennung des sendenden *Switches*
    - vermutete Wurzel-Kennung
    - eigene Entfernung zu dieser Wurzel
  - ▷ jeder *Switch* behält die beste Nachricht:
    - Wurzel-Kennung kleiner, oder
    - Wurzel-Kennung gleich, aber Entfernung kleiner, oder
    - Wurzel-Kennung & Entfernung gleich, Sender-Kennung kleiner
  - ▷ Jeder *Switch* startet mit sich selbst als Wurzel, bis dies widerlegt ist.
  - ▷ Wenn ein *Switch* erfährt, dass er nicht die Wurzel ist, stellt er das Generieren von Nachrichten ein, leitet Nachrichten aber nach wie vor weiter.

    Am Ende erzeugt nur noch die Wurzel periodisch Nachrichten, automatische Rekonfiguration bei Ausfall der Wurzel.
  - ▷ Topolgie-Änderungen werden von *Switches* zunächst an die Wurzel gesendet, diese gibt sie an alle weiter.

4.5 Virtuelle LANs (VLANs) (OSI 2)

- Virtuelle Aufteilung eines physischen LAN in mehrere „getrennte“ Netze.
- Jedes LAN erhält VLAN-ID. *Switches* werden so konfiguriert, dass bestimmte *Ports* bestimmten VLANs zugeordnet werden und nur im gleichen VLAN untereinander *Frames* ausgetauscht werden dürfen.
- Verbindet man mehrere *Switches* zu VLANs, werden auf den Leitungen zwischen *Switches* (***Trunk*-Leitung**) zusätzliche *Header* mit der VLAN-ID eingefügt. *Frames* werden nur an das LAN mit der korrekten VLAN-ID weitergeleitet.
- Zur Kommunikation zw. VLANs braucht man, wie bei physisch getrennten LANs, *Router*.

5 *Internetworking* (OSI 3)

- Def.: ***Internetwork*:** Zusammenschluss von einzelnen (auch heterogenen) Netzen über *Router* zu einem logischen Netz. („Netz von Netzen“)

  Internetwork ≠ Internet

  (Internet ist ein konkr. Bsp. eines *Internetworks*)

5.1 IP: Grundlagen

- IP-Dienstmodell: Was bietet IP? ⇒ Addressierungsschema & Datagramm-Zustellung

  IP muss für verschiedenste Netzwerktechnologien angepasst werden; je weniger IP selbst „kann“, desto einfacher ist das.
- **IPv4** auch heute noch Standard, aber IPv4-Adressen sind mittlerweile „aufgebraucht“.

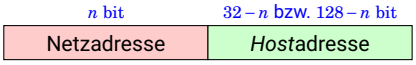
  IPv6 für (u. a.) größeren Adressraum. Aber aufgrund der Notwendigkeit, auf allen IP-Geräten aktualisiert werden zu müssen, bis heute noch nicht weitreichend aktiv.

5.2 IP: Adressierung & Weiterleitung

- *Router* müssen Pakete anhand IP-Adresse nur in das richtige Netz weiterleiten. (vgl.: Weiterleiten von Briefen in das richtige PLZ-Gebiet.)

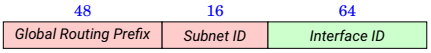
5.2.1 Adressierung in IP

- IP-Adresslänge: IPv4: 32 bit, IPv6: 128 bit
- Schreibweise IPv4: byteweise dezimal, durch Punkt getrennt (Bsp.: 131.159.31.17) (IPv6 s. u.)
- hierarchischer Aufbau:

- in IPv4 ursprünglich: Präfixlänge n geht aus IP-Adresse eindeutig hervor (Adressklassen)
  - ▷ Klasse A: $n = 8$ falls erstes Bit $= 0$
  - ▷ Klasse B: $n = 16$ falls 1. Bit $= 1$, 2. Bit $= 0$
  - ▷ Klasse C: $n = 24$ falls 1. & 2. Bit $= 1$

  heute in IPv4 & IPv6: explizite Angabe von n (klassenlose Adressierung, **CIDR** (*Classless Inter-Domain Routing*)) Bsp.: 192.168.0.0/24 ← n
- **IPv6-Adressen:** Typische Struktur:

  (feste Größen hier nur Konvention; weiterhin durch Angabe einer expliziten Präfixlänge n beeinflussbar wenn nötig)

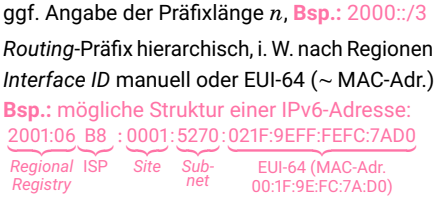
  Schreibweise: 16-Bit-Teile hexadezimal mit Trennzeichen „:“ (d. h. acht 4er-Blöcke) (Bsp.: 47CD:0000:0000:0000:0000:1234:A456:0124)

  Kurzform: ohne führende Nullen, 1 Nullfolge ersetzen durch „::“ (Bsp.: 47CD::1234:A456:124)

  ggf. Angabe der Präfixlänge n, Bsp.: 2000::/3
- *Routing*-Präfix hierarchisch, i. W. nach Regionen

  Interface ID manuell oder EUI-64 (~ MAC-Adr.)

  Bsp.: mögliche Struktur einer IPv6-Adresse:

- Gültigkeitsbereiche von IP-Adressen:
  - ▷ **Global**: weltweit eindeutig, Pakete mit diesen Adressen werden im globalen Internet weitergeleitet
  - ▷ ***Link Local***: nur innerhalb eines physischen LANs eindeutig, Router leiten Pakete mit diesen Adressen nicht weiter
  - ▷ ***Private*** (IPv4) bzw. ***Unique Local*** (IPv6): nur innerhalb eines privaten Netzes eindeutig (z. B. innerhalb einer Organisation), Pakete werden nicht im globalen Internet weitergeleitet
- Arten von IP-Adressen:
  - ▷ ***Unicast*:** für genau eine Netzwerkschnittstelle (in IPv6 hat eine Schnittstelle i. d. R. mehrere Adressen).
  - ▷ ***Multicast*:** für Gruppe von Empfängern.
  - ▷ ***Broadcast*** (nur IPv4)**:** alle Schnittstellen innerhalb eines Netzes
  - ▷ ***Anycast*:** nächstgel. Schnittstelle aus einer Menge
    - mehrere *Hosts* mit identischer Adresse und gleicher Funktion
    - *Router* leiten Pakete zum nächstgelegenen *Host* weiter
    - Anwendung z. B. für DNS-Server

Bedeutung	IPv4	IPv6
Global Unicast	0.0.0.0 bis 223.255.255.255	2000::/3
Link Local Unicast	169.254.0.0/16	FE80::/10
Unique Local Unicast		FC00::/7
Private Adressen	10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	
Multicast	224.0.0.0/4	FF00::/8
Local Broadcast	255.255.255.255	
Loopback	127.0.0.0/8	::1

5.2.2 IP-Weiterleitung

- Hosts mit gleicher Netzadresse kommunizieren direkt über ihr lokales Netz.
- Jedes physische Netz, das Teil des Internets ist, ist mit min. einem *Router* verbunden.
  Die Schnittstelle des *Routers* im lokalen Netz heißt auch ***Gateway***.
- Weiterleitung wird durch ***Routing*-Tabelle** gesteuert (in *Routern* sowie in normalen *Hosts*).

Netzadresse$_1$	Präfixlänge$_1$	*Next Hop*$_1$
Netzadresse$_2$	Präfixlänge$_2$	*Next Hop*$_2$
⋮	⋮	⋮

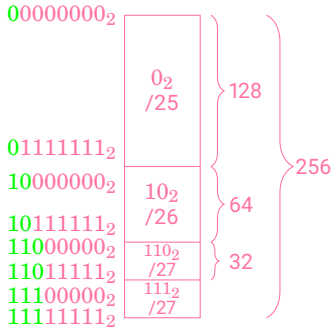
  Next Hop = *Router* bzw. Schnittstelle, an den/die das Paket weitergegeben werden soll, falls das Ziel im angegebenen Netz liegt.
- **Vorgehensweise bei IP-Weiterleitung:**
  Suche Eintrag i mit größter Präfixlänge$_i$, für den gilt: Zieladresse und Netzadresse$_i$ stimmen in den ersten Präfixlänge$_i$ Bits überein, d. h. die Zieladresse liegt in dem durch Netzadresse$_i$ & Präfixlänge$_i$ gegebenen Netz.
  Falls Eintrag gefunden:
  Weiterleiten an *Next Hop*$_i$.
  Sonst: Verwerfen des Pakets.
- Typischerweise zus. Eintrag für *Default-Route*:

0.0.0.0	0	*Next Hop*$_{\text{def}}$

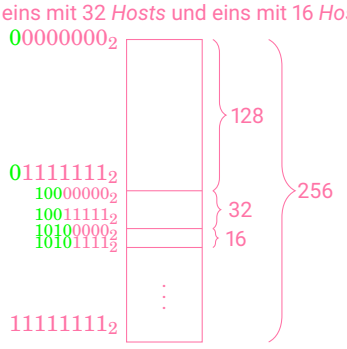
  Falls kein anderer Eintrag passt, passt dieser.

5.2.3 *Subnetting*

- Unterteilung eines großen Netzes in mehrere kleine Netze. Nach außen hin ist nur das Gesamtnetz sichtbar.
- IPv6: eigenes 16-Bit-Feld für Subnetz-ID (Subnetze einfach durchnummerieren).
- IPv4: ein Teil der *Host*-Bits wird für die Subnetz-ID „geborgt“ ⇒ neue (größere) Präfixlänge.
- Aufteilung eines IPv4-Netz in unterschiedlich große Subnetze:
  - **hierarchisches *Subnetting*:**
    Bsp.: Klasse-C-Adresse (/24):



  - ***Variable Length Subnet Mask* (VLSM):**
    Bsp.: Klasse-C-Adresse (/24). Angenommen, wir brauchen ein Netz mit 128 *Hosts*, eins mit 32 *Hosts* und eins mit 16 *Hosts*:



- Bestimmung der (Sub-)Netzzugehörigkeit:
  - Gegeben:
    (Sub-)Netzadresse und Präfixlänge n
    ⇒ Stimmen Zieladresse und Netzadresse in den ersten n Bits überein?
  - Gegeben:
    (Sub-)Netzadresse und (Sub-)Netzmaske
    ⇒ Gilt
    Zieladresse AND Netzmaske = Netzadresse?

5.3 Aufbau eines IP-Pakets

- erstes Feld im *Header* ist IP-Version (4 oder 6)
- **TTL** (v4)/***HopLimit*** (v6): zur Erkennung endlos kreisender Pakete. Wird von jedem *Router* heruntergezählt, bei 0 wird das Paket verworfen.
- ***Protocol*** (v4)/***NextHeader*** (v6): kennzeichnet Protokoll im Datenteil (*Multiplexing*); bei IPv6 ggf. Typ des folgenden Erweiterungs*headers*

5.4 IP: Fragmentierung/*Reassembly*

- **MTU (*Maximum Transmission Unit*):** maximale *Frame*größe der Netzwerktechnologie (**Bsp.:** Ethernet hat MTU von 1500 Byte.)
- Ist MTU auf Wegstück eines IP-Pakets zu klein, muss dieses fragmentiert (geteilt) werden
- zusätzliche *Header*-Felder:
  - ***Ident*:** kennzeichnet zusammengehörige Fragmente
  - **M**-Bit (bei IPv4 im ***Flags***-Feld): “*more fragments*” (Paket ist nicht das letzte Fragm.)
  - ***Offset*:** enthält Anfangsadresse des Fragments, zählt in 8-Byte-Schritten

  bei IPv6 sind diese Felder bei Bedarf in einem eigenen Erweiterungs*header*.
- Fragmentierung geschieht beim Sender, bei IPv4 auch in *Routern*, wenn **DF**-*Flag* nicht gesetzt (“*don't fragment*”), sonst Fehlerm. (ICMP)
  - Sender muss min. MTU kennen (ICMP)

5.5 ICMP (*Internet Control Message Protocol*)

Datagramme für Fehler- & Verwaltungsmeldungen

- Ziel nicht erreichbar
- *Reassembly* fehlgeschlagen
- Fragmentierung nicht erlaubt, aber erforderlich
- TTL wurde 0
- *Redirect*: besserer *Router* für das Ziel (wird heute aus Netzwerksicherheitsgründen ignoriert)
- *Echo Request/Reply*: **Bsp.:** `ping`, `traceroute`
- IPv6: *Router Solicitation/Advertisement*: Suche nach/Bekanntgabe von lokalen *Routern*
- IPv6: *Neighbor Solicitation/Advertisement*: Adressübersetzung (s. nächster Abschnitt)

⋮

5.6 Adressübersetzung

- Umsetzung IP-Adresse ↔ MAC-Adresse
- IPv4: **ARP (*Address Resolution Protocol*):**
  - Suche nach der zu gegebener IP-Adresse gehörender MAC-Adr. im ARP-*Cache*
  - Falls nicht gefunden, sende ARP-*Request* per *Broadcast* ins Zielnetz; betroffener Rechner sendet ARP-*Reply* per *Broadcast* mit seiner IP- & MAC-Adresse zurück
  - Setzt auf OSI-Schicht 2 auf. Ändert man Schicht-2-Technologie, muss neben IP auch ARP angepasst werden!
- IPv6: **NDP (*Neighbor Discovery Protocol*):**
  - Nutzung von ICMPv6-Paketen statt ARP-Protokoll (*Neighbor Solicitation* statt ARP-*Request*, *Neighbor Advertisement* statt ARP-*Reply*), ICMP setzt auf IP auf
  - Anfrage per *Multicast* statt *Broadcast* an zugehörige *Solicited-Nodes-Multicast*-Gruppe (bestimmt aus letzten 24 Bit der Ziel-IP-Adresse)

5.7 Automatische IP-Konfiguration

- **DHCP (*Dynamic Host Configuration Protocol*):**
  1. Rechner sendet per *Broadcast* Anfrage ins Netz (`DHCPDISCOVER`)
  2. DHCP-Server sendet `DHCPOFFER`: Angebot für IP-Adresse (dynamisch oder anhand von MAC-Adr.) und ggf. weitere Optionen
  3. Rechner fordert angebotene Adresse an (`DHCPREQUEST`), um die Adr. zu „mieten“ (falls es mehrere Server mit mehreren Angeboten gibt); regelmäßig wiederholen, um Adresse weiter zu belegen
  4. DHCP-Server bestätigt (`DHCPACK`)
- **Zustandslose IPv6-Autokonfiguration:**
  (ohne DHCP-Server)
  1. bilde *link-local*-Adresse (Präfix FE80::/10 und EUI-64/Zufallszahl)
  2. prüfe Adresse mit NDP (*Neighbor Solicitation*); falls keine Antwort: Adresse im LAN eindeutig
  3. warte auf *Router Advertisement*, sende ggf. *Router Solicitation* wegen Wartezeit
  4. *Router Advertisement* teilt Adresse, globales *Routing*-Präfix & Präfixlänge; ggf. Befehl, doch einen DHCP-Server zu fragen
  5. bilde Adresse aus Präfix und z. B. EUI-64
  6. prüfe Adresse mit NDP

5.8 *Network Address Translation* (NAT)

speziell: ***Network Address Port Translation* (NAPT)**

- Ziel: Einsparen von IP(v4)-Adressen
- jeder Rechner bekommt private IP-Adresse (s. Abschnitt 5.2.1), das gesamte Netz erhält eine „echte“ IP-Adresse, ausgehende Pakete durchqueren NAT-Box (üblicherweise im *Router*/*Modem*), ersetzt IP-Adressen im Paket
- Woher weiß die NAT-Box, an wen das Paket gehen soll? Lösung: (schmutzig! ↯ OSI 4)
  - UDP-/TCP-*Header* enth. Felder zur Adressierung von Quell- & Zielprozess (*Port*)
  - NAT-Box verwendet Quell-*Port* zum Speichern eines Demultiplex-Schlüssels
    - bei ausgehendem Paket:
      speichere Quell-IP-Adresse & Quell-*Port* (⇒ Schlüssel), ersetze IP-Adr., ersetze Quell-Port durch Schlüssel
    - Antwortpaket erhält Schlüssel als Ziel-*Port*, ersetze Ziel-Adresse/*Port* durch gespeicherte Adresse/*Port*
- mit NAT wird Internet verbindungsorientiert; Ausfall der NAT-Box = Zerstörung aller Kommunikationsbeziehungen
- es gibt Protokolle, die eigene IP-Adresse enthalten (**Bsp.:** FTP, H.323) ⇒ auch diese müssen durch NAT-Box ausgetauscht werden (↯ OSI 7)

6 *Routing* (OSI 3)

- **Autonomes System (AS):** unabhängige Netze
  - ***Transit*-AS:** z. B. *Backbone Service Provider*, leitet Pakete weiter
  - ***Multihome*-AS:** an mehrere *Backbones* angeschlossen, leitet keine Pakete weiter
- hierarch. Ansatz: ***Routing Domains*** (Bereiche)
  - *Routing* innerh. eines administrativen Bereichs (**Bsp.:** Campus, Unternehmen, ISP)
    → *Interior Gateway Protocols* (IGP)
    (**Bsp.:** RIP, OSPF, EIGRP)
  - *Inter-Domain Routing*
    → *Exterior Gateway Protocols* (EGP)
    (**Bsp.:** BGP (*Border Gateway Protocol*)
- Wir betrachten nur Netze (*Router*, keine *Hosts*).

6.2 *Routing* innerhalb eines Bereichs

- *Routing* als Graph-Problem: Knoten: Router, Kanten: Verbindungen mit Kosten (Metrik) (symmetrisch oder asymmetrisch).
  Aufgabe: Finde Pfade mit geringsten Kosten zwischen allen Paaren von Knoten.

- **Statisches *Routing*:** Pfade manuell bestimmen und in *Routing*-Tabelle eintragen. Probleme:
  - ▷ Ausfall von Knoten/Verbindungen
  - ▷ neue Knoten/Verbindungen
  - ▷ dynamische Änderung der Verbindungskosten (Last)

  Macht man üblw. nur bei sehr kleinen Netzen.
- **Dynamisches *Routing*:**
  - ▷ verteilte Algorithmen in den *Routern* zum automatischen Aufbau der Tabellen
  - ▷ Anforderungen: schnelle Konvergenz, Skalierbarkeit, einfache Administration

6.2.1 *Distance Vector Routing*

- *Router* kennt nur Distanz & „Richtung“ (*Next-Hop*) zum Ziel
- verteilter Algorithm. zur Erstellung der *Routing*-Tabellen (typisch: Bellman-Ford-Algorithmus)
- Zu Beginn kennt jeder *Router* nur sich selbst (Kosten 0) und seine direkten Nachbarn.

  Kosten zu unbekannten Netzen seien ∞.
- *Router* senden ihren Distanzvektor an alle direkten Nachbarn. Bessere *Routen* werden in den eigenen Distanzvektor (und *Routing*-Tabelle) übernommen, sowie schlechtere vom *NextHop* (damit sich Verbindungen auch ändern dürfen).
- Nachrichtenaustausch periodisch (alle 30 s), sowie bei Änderung eines Distanzvektors oder auf Anfrage (z. B. *Router*-Neustart).
- Problem: ***Count-to-Infinity*:** Wird ein Ausfall eines Knotens erkannt, aber ein anderer Knoten sendet in dem Moment seinen Distanzvektor, so kreist dieser umher und die gespeicherte Distanz zum ausgefallenen Knoten wird immer wieder zueinander addiert, statt direkt auf ∞ gesetzt.

  Lösung: beschränke ∞ auf 16 (im Internet)
- Alternative Lösung: *Holddown-Timer:* wenn eine *Route* ausfällt, wird für die nächsten 180 s keine neue *Route* mit gleicher oder schlechterer Distanz für den ausgefallenen *Router* gespeichert.
- Anderes Problem bei *Count-to-Infinity*-Zustand: *Routing*-Schleifen. Lösung: TTL
- **Bsp.: RIP** (*Routing Information Protocol*):
  - ▷ einfaches *Distance-Vector-R.*-Protokoll
  - ▷ Internet-Standard
  - ▷ Alle *Link*-Kosten sind 1, d. h. Distanz = *Hop Count*.
  - ▷ RIPv1: leitet keine Präfixlänge weiter, RIPv2: ermöglicht klassenloses *Routing*, RIPng: unterstützt IPv6
- **Bsp.: EIGRP** (*Enhanced Interior Gateway Routing Protocol*):
  - ▷ Cisco-proprietär, seit 2013 offener Internet-Standard
  - ▷ *Link*-Kosten berücksichtigen Bandbreite & Latenz
  - ▷ untertützt IPv4, IPv6 und andere OSI-Schicht-3-Protokolle
  - ▷ *Updates* nur bei Änderungen, kein *Count-to-Infinity*
  - ▷ behält alle *Routen*, nicht nur die beste

6.2.2 *Link State Routing*

- *Router* erhalten Informationen über die Struktur des gesamten Netzwerks (ganzer Graph)
- Vorgehensweise: Kennenlernen der direkten Nachbarn, Versenden von *Link-State*-Paketen an alle anderen *Router* (***Reliable Flooding***).
- Berechnung der kürzesten Wege mit Dijkstra-Algorithmus (s. u.).
- ***Link-State*-Pakete**: Inhalt:
  - ▷ ID des erzeugenden *Routers*
  - ▷ Liste der dir. Nachbarn mit *Link*-Kosten
  - ▷ Sequenznummer (Paket wird nur weitergeleitet, wenn die Sequenznummer größer als die des letzten Pakets ist)
  - ▷ TTL
- Versenden der Pakete periodisch (~ Stunden) oder bei Topologie-Änderungen
- **Dijkstra-Algorithmus:**

  EINGABE: Knotenmenge N, *Link*-Kosten $l(i,j)$ von Knoten i zu Knoten j, Startknoten s

  AUSGABE: Pfad-Kosten $C(n)$ von s zu $n \in N$

  ALGORITHMUS:

  $M := \{s\}$
  FORALL $n \in N \setminus \{s\}$ DO
  $C(n) := l(s,n)$ (∞ falls nicht von s erreichbar)
  ENDFOR
  WHILE $M \neq N$ DO
  Wähle $w \in N \setminus M$ so, dass $C(w)$ minimal.
  $M := M \cup \{w\}$
  FORALL $n \in N \setminus M$ DO
  $C(n) := \min\big(C(n),\ C(w) + l(w,n)\big)$
  ENDFOR
  ENDWHILE

  SCHREIBWEISE: Tabelle **Bsp.:** (Startknoten A)

Schritt	w	M	$C(B)$	$C(C)$	…
Initial		$\{A\}$	5	∞	…
1	B	$\{A,B\}$	5	16	…
⋮	⋮	⋮	⋮	⋮	⋱

- **Bsp.: OSPF** (*Open Shortest Path First*):
  - ▷ weit verbreitetes *Link-State-Routing*-Protokoll
  - ▷ Internet-Standard
  - ▷ *Link*-Metrik nicht spezifiziert; in der Praxis: Bandbreite
  - ▷ OSPFv2 für IPv4, OSPFv3 für IPv6
  - ▷ Besonderheit in Mehrfachzugriffsnetzen: *Router* wählen einen „designierten“ *Router* (DR) und einen *Backup*-DR (BDR). Bei Ausfall von DR: BDR wird neuer DR, Neuwahl BDR.

    Link-State-Pakete werden nur an diese gesendet und vom DR an alle weiterverteilt. (Verhindert exzessives *Flooding* im Netz.)
  - ▷ ***Multi-Area*-OSPF:** Unterteilung eines sehr großen *Routing*-Bereichs in mehrere kleinere Bereiche (***Areas***)
    - *Area Border Router* (ABR) verbinden *Areas*
    - *Area 0*: *Backbone*-Bereich, der mit allen anderen *Areas* verbunden ist. *Areas* dürfen nicht untereinander verbunden sein, nur über *Backbone*!
    - innerhalb eines Bereichs: *Flooding* von *Link-State*-Paketen. ABR geben diese nicht weiter, stattdessen senden diese zusammengefasste Information (gibt vor, dass alle *Router* in *Area* direkt mit ihm verbunden sind).

6.3 *Inter-Domain Routing*

- *Routing* zwischen autonomen Systemen (/*Domains*) ⇒ keine gemeinsame Metrik
- ***Border Gateway Protocol* (BGP):**
  - ▷ *Routen* werden „politisch“ bestimmt (z. B.: benutze bestimmten *Provider* für bestimmte Adressen, weil man mit dem *Provider* einen Vertrag hat)
  - ▷ wichtigstes Ziel: Erreichbarkeit („gute“ *Routen* sind sekundär)
  - ▷ Jedes AS hat
    - ≥ 1 *Border Router* (Verbindung zu anderen AS)
    - einen BGP-Sprecher, der lokales Netzwerk & über dieses AS erreichbare Netzwerke angibt (vollständige Pfade zur Vermeidung von zyklischen *Routen*)

7 Ende-zu-Ende-Protokolle (OSI 4)

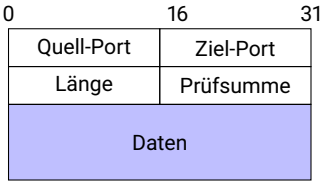
(d. h. genauer: Prozess-zu-Prozess-Kommunikation)

- Protokolle der Transportschicht:
  - ▷ Adressierung von Prozessen auf Rechner
  - ▷ ggf. Garantien: Zustellung, Reihenfolge, …
- indirekte Adressierung durch ***Ports*** (16 bit-Nr.) statt z. B. die systemabhängige, zufällige Prozess-ID des OS zu nutzen
- *“well-known ports”* für Systemdienste (i. d. R. *Port* 0 bis 1023) **Bsp.:** 80: *Webserver*, 25: SMTP
- *Server* kennt *Port*-Nummer des *Clients* aus UDP- bzw. TCP-*Header* der Anfrage
- *Ports* sind typischerweise durch Warteschlangen realisiert. (*Demultiplexing*: verschiedene Warteschlangen pro *Port*)

  Bei voller Warteschlange: Paket verwerfen

7.2 UDP (*User Datagram Protocol*)

- unzuverlässiger Dienst
- „Mehrwert“ im Vergleich zu IP: Kommunikation zwischen Prozessen, UDP übernimmt das Demultiplexen, Prüfsumme über *Header*
- Aufbau eines UDP-Pakets:



7.3 TCP (*Transmission Control Protocol*)

- zuverlässige Übertragung von Datenströmen zwischen Prozessen, verbindungsorientiert
- meist-verwendetes Internetprotokoll, befreit Anwendungen von Sicherung der Übertragung
- Vollduplexverbindungen
- Flusskontrolle, Überlastkontrolle
- überträgt Daten (Byteströme) segmentweise, Daten auf Schicht 7 nicht abgegrenzt

  Ein Segment wird gesendet, wenn…
  - ▷ *Maximum Segment Size* (MSS) erreicht

    MSS = MTU − Size(TCP-*Header*) − Size(IP-*Header*)

    (verhindert, dass Segment von IP sofort wieder fragmentiert werden muss)
  - ▷ der Sender es ausdrücklich fordert (genannt *Push*- oder *Flush*-Operation)
  - ▷ nach Ablauf eines periodischen *Timers*
- Aufbau TCP-*Header*:
  - ▷ Quell- & Ziel-*Port* analog UDP
  - ▷ Felder für Übertragungssicherung (Sequenznr., *Acknowledgement* (nächste erwartete Sequenznr.), *AdvertisedWindow*)
  - ▷ optionale/variable Optionen (mit *HdrLen*)
  - ▷ 6 bit *Flags*:
    - `SYN`: Verbindungsaufbau
    - `FIN`: Verbindungsabbau
    - `ACK`: *Acknowledgement*-Feld gültig
    - `URG`: dringende Daten (*out-of-band data*), ***UrgPtr***-Feld zeigt Länge der dringenden Daten an, welche die Puffer „überholen“ darf
    - `PSH`: Anwendung hat *Push*-Operation ausgeführt
    - `RST`: Abbruch der Verbindung (Fehler)
- Verbindungsaufbau (***3-Way-Handshake***):
  - ▷ asymmetrisch
  - ▷ *Client* sendet Verbindungswunsch zum *Server* (`SYN`, *SequenceNum* = x ← zuf. Startw.)

▷ *Server* wartet auf eingehende Verbindungswünsche, akzeptiert ggf. (`SYN+ACK`, *SequenceNum* $= y$, *Ack.* $= x + 1$)

▷ *Client* bestätigt (`ACK`, *Ack.* $= y + 1$)

vgl. Telefonanruf

- Verbindungsabbau: symmetrisch, beide Seiten müssen Verbindung schließen

  Danach *Timeout* ($2\times$Segmentlebensdauer), bevor eine Verbindung mit gleichem Ziel & gleichem Quell-*Port* aufgebaut werden kann, damit keine Verwechslung zwischen verschiedenen Inkarnationen derselben Verbindung entsteht

7.4 Sicherung der Übertragung (OSI 2 & 4)

Hier wird fortan von *Frames* (d. h. OSI 2) gesprochen, gilt aber genauso für Segmente auf OSI-Schicht 4.

- verlorene/mit Fehler(n) übertragene *Frames* müssen neu übertragen werden
- Basismechanismen zur Lösung:
  - ▷ Bestätigungen (*Acknowledgemens*, `ACK`)

    Bei Duplex-Verfahren (**Bsp.:** TCP) auch **Huckepack-Verfahren** (*Piggyback*): Bestätigungen im *Header* eines normalen *Frames* mitübertragen
  - ▷ *Timeouts*: wenn nach einer bestimmten Zeit kein `ACK` eintrifft, erneut senden

7.4.1 *Stop-and-Wait*-Algorithmus

- Sender wartet nach Übertragung eines *Frames*, bis `ACK` eintrifft, dann erst wird der nächste *Frame* gesendet.
- Problem: Falls `ACK` verloren geht: Wiederholung des gesendeten *Frames*.

  Daher: *Frames* erhalten Sequenznummer. Bei *Stop-and-Wait* reicht 1 bit (abwechselnd $0, 1$).
- Problem: Bandbreite kann nicht ausgenutzt werden. (Nur ein *Frame* pro RTT.)

7.4.2 *Sliding-Window*-Algorithmus

- Sender sollte (RTT $\cdot$ Bandbreite)-viele Daten senden, bevor auf das erste `ACK` gewartet wird. Dann mit jedem `ACK` 1 neuen *Frame* senden.
- Jeder *Frame* erhält eine Sequenznummer, nach *Timout* ohne `ACK` wird *Frame* erneut gesendet (analog zum *Stop-and-Wait*-Algorithmus).
- Der Sender besitzt ein „Schiebefenster“:
  - ▷ LAR: *Last Acknowledgement Received*

    Bis zu diesem *Frame* wurden alle quittiert.
  - ▷ LFS: *Last Frame Sent*
  - ▷ SWS: *Sender Window Size*

  max. SWS-viele *Frames* werden ohne `ACK` abgeschickt.
- Der Empfänger hat ebenf. ein „Schiebefenster“:
  - ▷ LFR: *Last Frame Received*

    Alle *Frames* n mit $n \leq$ LFR wurden korrekt empfangen & quittiert.
  - ▷ LAF: *Largest Acceptable Frame*

    Frame n wird nur akzeptiert, wenn LFR $< n \leq$ LAF.
  - ▷ RWS: *Receiver Window Size*

    Anzahl der Pufferplätze beim Empfänger.
- Quittierung von *Frames*:
  - ▷ **Akkumulatives *Acknowledgement*:** `ACK` für *Frame* n gilt auch für alle *Frames* $\leq n$.
  - ▷ Zusätzlich **negative *Acknowledgements*:** Wenn *Frame* n empfangen wird, aber *Frame* m mit $m < n$ noch aussteht, wird für *Frame* m ein `NACK` geschickt.
  - ▷ Alternativ: **selektives *Acknowledgement*:** `ACK` für *Frame* n gilt nur für diesen *Frame*.
- Reicht endlicher Bereich für Sequenznr.? Ja. #Sequenznummern $\geq$ SWS + RWS

  Aber nur, wenn die Reihenfolge der *Frames* bei der Übertragung nicht verändert werden kann! (d. h. im Internet reicht ein endl. Bereich nicht! ⇒ s. nächster Abschnitt)

7.5 Übertragungssicherung in TCP (OSI 4)

- TCP benutzt *Sliding-Window*-Algorithmus.
  - ▷ Sequenznr. zählt Bytes, nicht Segmente.
  - ▷ TCP benötigt Verbindungsaufbau ⇒ Austausch über *Sliding-Window*-Parameter (Start-Sequenznr., nicht bei 0 anfangen)
  - ▷ TCP toleriert bis zu 120 s alte Pakete
- Sicherstellung der richtigen Reihenfolge: TCP gibt Segmente nur an obere Schicht weiter, wenn alle vorherigen Segm. bestätigt wurden.
- **Flusskontrolle:** Empfänger teilt dem Sender den freien Pufferplatz mit (*AdvertisedWindow*, gesendet in jedem `ACK`). Sender passt Senderfenstergröße automatisch an.
- **Überlastkontrolle:** verhindern, dass mehrere Sender einen Teil des Netzwerks überlasten (Pufferüberlauf in *Routern*)
  - ▷ TCP beobachtet das Verhalten des Netzes (Lastsituation) anhand Paketverlust und ggf. RTT & Durchsatz.
  - ▷ neues Feld *CongestionWindow*, wird verkleinert anhand von Anzeichen hoher Last (üblicherweise: halbiert)
- Sequenznummern-Überlauf:
  - ▷ TCP-*Header*: 32-bit-Feld für Sequenznr.
  - ▷ bei Netzen von 1 Gbit/s und schneller läuft dieses in deutlich unter 120 s über ⇒ TCP-Erweiterung: Zeitstempel
- *AdvertisedWindow* ist 16 bit, d. h. max. Puffer 64 KiB ⇒ TCP-Erweiterung: Skalierungsfaktor
- RTT im Internet nicht konstant ⇒ *Sliding-Window-Timeout* muss adaptiv sein
  - ▷ Ursprünglich: Messung der durchschnittlichen RTT (Zeit zwischen Senden eines Segments & Ankunft des `ACK`), *Timeout* $= 2 \cdot$ Durchschnitts-RTT

    Problem: Varianz nicht berücksichtigt
  - ▷ heute: **Jacobson/Karels-Algorithmus**: Berücksichtige die Standardabweichung ($\sqrt{\text{Varianz}}$) der RTT mit Gewichtung δ

8 Datendarstellung (OSI 6)

- Rechner haben unterschiedliche Datendarstellung. Häufigster Unterschied: Byte-Reihenfolge

  Bsp.: `int` $34\,677\,374 = 0211227E_{16}$
  - ▷ ***Big Endian*:** $(02_{16})(11_{16})(22_{16})(7E_{16})$
  - ▷ ***Little Endian*:** $(7E_{16})(22_{16})(11_{16})(02_{16})$

  Andere Untersch.: z. B. Größe (16, 32, 64 bit)
- ***Marshalling* (Darstellungsformatierung):**

  1. **Basistypen:** `int`, `double`, `bool`, `char`, …

     ⇒ Formate & Byte-Reihenf. konvertieren
  2. **Flache Datentypen:** `struct`, Arrays

     ⇒ Füllbytes für *Alignment* (falls Struktur z. B. nur bei geraden Adressen beginnen darf)
  3. **Komplexe Datentypen:** Listen, Bäume, … mit Zeigern ⇒ Serialisierung

- Konvertierungsstrategien:
  - ▷ **Kanonisches Netzwerk-Datenformat:**
    - Sender konvertiert in Netzwerk-Datenf.
    - Empfänger konv. dann in sein Format

    Bsp.: Internet-Protokolle
  - ▷ ***Receiver-Makes-Right*:**
    - Sender sendet in seinem Format, serialisiert lediglich; fügt **Datentyp-Kennzeichnung (*Tags*)** an (z. B. Typ, Array-Länge, Architektur/*Endianness*, …)
    - Empfänger konvertiert, falls nötig, in sein Format

9 Anwendungsprotokolle (OSI 7)

9.1 E-Mails/SMTP

- ASCII-Text-basiert (d. h. lesbar)
- Mails werden gesendet über SMTP zu Mail-Servern
- *Clients* ohne eigenen Mail-Server können Mails über POP oder IMAP herunterladen
- Format einer Mail: *Header* und Rumpf. Mehrere Teile einer Mail werden durch im *Header* spezifizierten `boundary`-String getrennt.
- Medien werden oft als `base64` kodiert.
- Serverantworten beginnen immer mit 3 Ziffern:
  - ▷ erste Ziffer: Status
    - 2: OK
    - 3: weitere Aktion notwendig
    - 5: Fehler

9.2 HTTP

- wie SMTP ebenfalls Text-basiert
- Kommandos:
  - ▷ `GET <URL>`: Anfrage nach Dokument
  - ▷ `HEAD <URL>`: Anfrage nach Metainfo
  - ▷ `POST <URL>`: Senden von Information **Bsp.:** Online-Formular
  - ▷ `PUT <URL>`: Speichern eines Dokuments
  - ▷ `DELETE <URL>`: Löschen eines Dok.
- Ergebniscodes:

$1xx$	Informativ	Anfrage erhalten
$2xx$	Erfolg	Anfrage empfangen & akzeptiert
$3xx$	Weiterleitung	weitere Aktion notwendig
$4xx$	*Client*-Fehler	Syntaxfehler, nicht erfüllbar, …
$5xx$	*Server*-Fehler	Anfrage OK, Problem im Server

- HTTP 1.0: neue TCP-Verbindung für jedes Seiten-Element

  ab HTTP 1.1: persistente Verbindung möglich (mehrere Anfragen über dieselbe TCP-Verbindung) ⇒ *Timeout*-Mechanismus beim *Server* notwendig (viele offene Verbindungen)

9.3 DNS

- IP arbeitet mit numerischen Adressen.

  Benutzer verwenden (*Host*-/Rechner-)Namen (benutzerfreundl., aber nicht f. *Routing* nutzbar)

  ⇒ Umsetzung von Namen auf IP-Adressen durch **DNS (*Domain Name Service*)**
- *Client* stellt Anfrage an *Nameserver* mit *Host*name, dieser antwortet mit dessen IP-Adresse.
- *Nameserver* sind hierarchisch (in Zonen) (TLD – Domains – Subdomains – …)
- jeder Rechner kennt nur seinen lokalen *Nameserver*
- lokale *Nameserver* lösen lokale Namen auf, kennen *Root-Nameserver* (über Konfig.-datei)
- Server führen *Cache* mit bereits aufgelösten Namens-Adress-Paaren mit begrenzter TTL.

  Falls Zuordnung nicht im *Cache*:
  - ▷ *recursive query*: Nachfrage bei anderem *Server* (in der Hierarchie höher, ggf.)
  - ▷ *non-recursive query*: Antworte mit der Adresse eines anderen *Nameservers*, erneute Anfrage des *Clients* (heute üblich)

10 Netzwerksicherheit

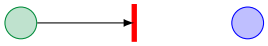
- Allgemeine Sicherheitsanforderungen:
  - ▷ **(Informations-)Vertraulichkeit (*confidentiality*)**: Schutz vor unautorisierter Informationsgewinnung (Das Bitmuster (Daten) ist ggf. unproblematisch, solange daraus keine Information extrahiert

werden kann ⇒ Verschlüsselung)

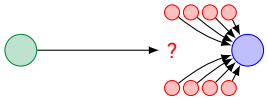
- ▷ **(Daten-)Integrität (*integrity*)**: Schutz vor unautorisierter Veränderung von Daten
- ▷ **(Nachrichten-)Authentizität (*message authenticity*)**: Urheber der Daten soll identifizierbar sein (**Bsp.:** Spam-Mails)
- ▷ **Verbindlichkeit (*nenrepudiation*)**: „Nicht-Abstreitbarkeit“ (**Bsp.:** Unterschrift)
- ▷ **Verfügbarkeit (*availability*)** (**Bsp.:** DoS)
- ▷ **Anonymität** der Kommunikationspartner

- Angriffe auf die Netzwerksicherheit:
  - ▷ Normaler Informationsfluss:

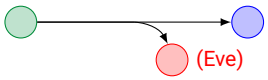
  - ▷ Unterbrechung (***Denial of Service*, DoS**): Angriff gegen Verfügbarkeit:



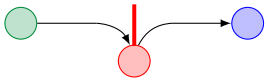
Distributed Denial of Service (DDoS):



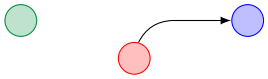
  - ▷ Abhören (***Sniffing***): Angriff gegen Vertraulichkeit:



  - ▷ Modifikation (***Man-in-the-Middle***): Angriff gegen Integrität:



  - ▷ Erzeugung (***Spoofing***): Angriff gegen Authentizität:



  - ▷ Wiedereinsp. abgehörter Daten (***Replay***): Kombination aus *Sniffing* & *Spoofing*; abgehörte Daten werden gespeichert und vom Angreifer 1:1 erneut zu *Bob* gesendet.
- Internet-Standard-Protokolle (u. a. IP, TCP, DNS, ARP, NFS, HTTP, SMTP, …) erfüllen in sich keine der genannten Sicherheitsanforderungen.

  Lösung: sichere Protokolle in der Anwendungsschicht, **Bsp.:** SSL/TLS (HTTPS, SFTP, …), S/MIME, PGP, SSH, …, oder: IPsec

10.3 Kryptographische Grundlagen

- **Symmetrische Verschlüsselungsverfahren:** gemeinsamer geheimer Schlüssel
- **Asymmetrische Verschlüsselungsverfahren:** öffentlicher Schlüssel zum Verschlüsseln, privater Schlüssel zum Entschlüsseln. Probleme:
  - ▷ f. Sicherheit deutl. längere Schlüssel nötig
  - ▷ Authentizität des öffentlichen Schlüssels
- Anforderung: Nur der Besitzer des geheimen bzw. privaten Schlüssels kann den Chiffretext entschlüsseln.

  Sicherheit basiert nicht auf Geheimhaltung der Algorithmen.
- Mögliche Angriffe:
  - ▷ Klartext-Angriff: Abhören von verschlüsselten Nachrichten, deren Inhalt bekannt ist, um den Schlüssel zu rekonstruieren
  - ▷ alle Schlüssel durchprobieren
  - ▷ Berechn. v. asymmetrischen Schlüsseln
- **Kryptographische *Hashes* (*message digest*)**: kollisionsresistente *Hash*-Funktion zur Sicherung der Integrität („kryptograph. Prüfsumme“)

  Bsp.: MD5, SHA-1 (unsicher); SHA-2, SHA-3
- Authentizität & Integrität sind nur zusammen sinnvoll! Wenn man sicher sein kann, dass eine Nachricht ursprünglich von *Alice* kommt, aber sich nicht sicher sein kann, dass diese Nachricht nicht auf dem Weg verändert wurde, ist das vollkommen witzlos.

10.4 Sicherheitsmechanismen

- **Partner-Authentifizierung:** „Wer ist mein Gegenüber?“ **Bsp.:** *Fileserver*, HTTPS-*Webserver*
  - ▷ **Drei-Wege-*Handshake*:** (symm.)
    1. *Client* sendet *ClientID* und verschlüsselte Zufallszahl x.
    2. *Server* sucht den zu *ClientID* gehörigen Schlüssel, sendet damit verschlüsselt $x+1$ und weitere Zufallszahl y.
    3. *Server* ist authentifiziert, wenn $x+1$ korrekt beim *Client* ankommt.
    4. *Client* sendet entsprechend $y+1$ verschlüsselt zurück. *Client* ist authentifiziert, wenn $y+1$ korrekt beim *Server* ankommt.
    5. *Server* sendet *Session Key* für weitere Kommunikation.
  - ▷ **Asymmetrische Chiffre:**
    1. Sende mit öffentl. Schlüssel vom Ziel verschlüsselte Zufallszahl x.
    2. Ziel sendet entschlüsselte Zufallszahl zurück. Wenn x korrekt ankommt, ist das Ziel authentifiziert.

    Der Schlüssel darf dann nicht mehr zur Verschlüsselung verwendet werden!
- Sicherung der Integrität:
  - ▷ bei symmetrischer Verschlüsselung:

    Füge kryptographischen *Hash*wert $H(M)$ an Klartext M an, dann verschlüsseln.
  - ▷ *Replay*-Schutz: Transaktionszähler oder Zeitstempel in M
- ***Hash*wert mit geheimen Schlüssel:** Einbeziehen eines geheimen Schlüssels K in den *Hash*wert: Füge $H(M+K)$ an Nachricht M an.
  - ▷ Sichert, dass Angreifer den *Hash*wert selbst nicht neu berechnen können.

    (Nachrichten-Integrität & Authentizität)

  Bsp.: HMAC-SHA-256
- **Digitale Signatur** mit asymmetrischer Chiffre:
  1. Sender A sendet Nachricht M und mit dem privaten Schlüssel $\mathrm{Private}_A$ verschlüsselte Signatur (d. h. kryptographischer *Hash*wert) $H(M)$ an Empfänger B.
  2. B entschlüsselt mit Public_A und prüft, ob das Ergebnis $= H(M)$ ist.
- Verteilung öffentlicher Schlüssel: Woher weiß B, dass Public_A tatsächlich der Schlüssel von A ist (und nicht Schlüssel eines Angreifers)?

  ⇒ **Digitales Zertifikat**
  - ▷ Zertifizierungsstelle (CA, *Certification Authority*) beglaubigt die Zuordnung zwischen einem öffentlichen Schlüssel und seinem Besitzer durch digitale Signatur.
  - ▷ Nur noch der öffentliche Schlüssel der CA muss separat veröffentlicht werden.

    ⇒ **Zertifizierungshierarchie**

    Überprüfe immer wieder ausstellende CA, bis zur Wurzel-CA, welche fest im Betriebssystem/Browser als vertrauensvoll hinterlegt sind.
  - ▷ **Bsp.: X.509:** Standard für Zertifikate
  - ▷ Zertifikate haben i. A. Ablaufdatum, damit kompromittierte Zertifikate/private Schlüssel nicht beliebig lang gültig sind.

    Es gibt *Certificate Revocation List* (CRL), welche widerrufene Zertifikate, signiert von CA, enthält. Ablaufdatum der Zertifikate begrenzt Länge dieser Liste.

10.6 *Firewalls*

- d. h. *Router* mit Paket-Filterfunktion
- *Firewall* arbeitet üblw. auf Transportschicht (OSI 4), kann Pakete basierend auf *Port* filtern.
- ***Proxy*-basierte *Firewalls*:** Mittler zwischen *Client* & *Server*. Für *Client* ist *Proxy* der *Server*, für *Server* ist *Proxy* der *Client*.

  Proxy arbeitet auf Anwendungsschicht (OSI 7), kann basierend auf Inhalt der Nachricht filtern.
- Filter über Sender-IP & -Port unsicher!