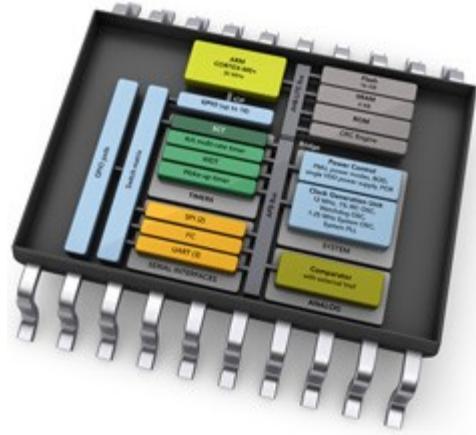


Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Das Folienmaterial basiert auf Unterlagen von Kollege Prof. Dr. Sander

Lernziele der Veranstaltung

Sie werden am Ende der Vorlesung

- Die Grundlagen der Datenkommunikation beherrschen
- Die Funktionsweise der Internet Protokolle beschreiben und spezifische Eigenschaften begründen können
- Die gängigen Techniken zum Aufbau lokaler Netze kennen und anwenden können

Sie werden zum Ende des Praktikums

- Erste Client-Server-Anwendungen entwerfen und implementieren können
- XML verarbeiten und Multithreading anwenden können

Literatur



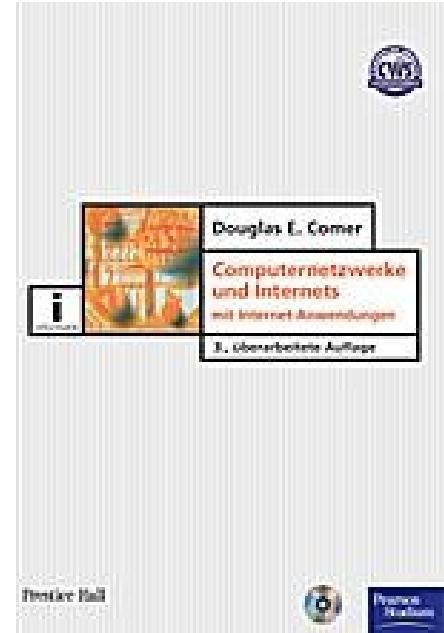
Computernetzwerke
Der Top-Down-Ansatz

6. Auflage
J.F. Kurose / K.W. Ross
Pearson



Computernetzwerke

5. Auflage
A.S. Tanenbaum
D.J. Wetherall
Pearson



Computernetzwerke
und Internets

3-te Auflage
Douglas E. Comer
Pearson Studium
€ 29,95

Weitere Informationsquellen (vor allem zum Internet)

<https://www.ietf.org/>



Home > Internet standards >

RFCs

Memos in the RFC document series contain technical and organizational notes about the Internet.

RFCs cover many aspects of computer networking, including protocols, procedures, programs, and concepts, as well as meeting notes, opinions, and sometimes humor. Below are links to RFCs, as available from ietf.org and from rfc-editor.org. Note that there is a brief time period when the two sites will be out of sync. When in doubt, the RFC Editor site is the authoritative source page.

RFCs associated with an active IETF Working Group can also be accessed from the Working Group's web page via [IETF Working Groups](#).

IETF Repository Retrieval

- Advanced search options are available at [IETF Datatracker](#) and the [RFC Search Page](#).
- A text index of RFCs is available on the IETF web site here: [RFC Index \(Text\)](#).
- To go directly to a text version of an RFC, type <https://www.ietf.org/rfc/rfcNNNN.txt> into

INTERNET
STANDARDS
RFCs
Internet-Drafts
Intellectual property
rights
Standards process

Struktur der Vorlesung

- Klassisch: ‚Bottom up‘ (z.B. Tanenbaum): vom Übertragungsmedium bis zu den Anwendungsprotokollen
- Bei uns: Eher ein Wechsel zwischen ‚unteren‘ und ‚oberen‘ Schichten aufgrund des Praktikums
- Grobe Struktur der Vorlesungsreihe:
 - Grundlagen der Datenkommunikation
 - Einfache Client-Server Anwendungen
 - Internet-Protokolle
 - Netzwerke

Begriffe: Datenkommunikation / Kommunikationssystem

Die **Datenkommunikation** beschäftigt sich mit dem immateriellen Transport digitaler Daten zwischen Endsystemen. Hierbei sind alle hierzu benötigten Verfahren und Regeln Bestandteil der Datenkommunikation.

Im engeren Sinn ist ein **Kommunikationssystem** eine Einrichtung bzw. eine Infrastruktur für die Übermittlung von Informationen. Kommunikationssysteme stellen dazu Nachrichtenverbindungen zwischen mehreren Endstellen her. (Wikipedia)

Bedeutung der Datenkommunikation

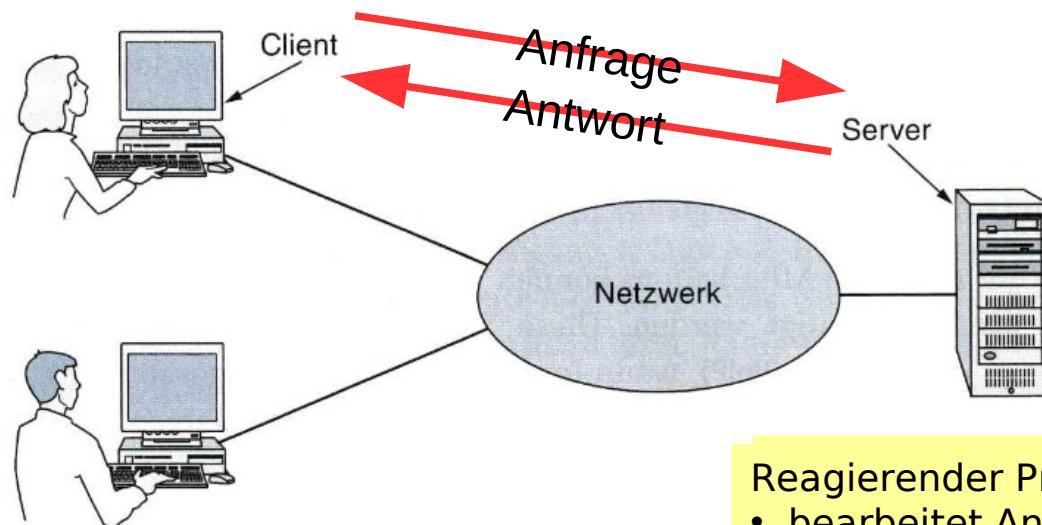
- Durch Datenkommunikation kann man auf fremde/entfernte Ressourcen und Dienste zurückgreifen
- Erforderlich dazu:
 - Effiziente Methoden zum Datenaustausch zwischen Kommunikationspartnern
 - Absprachen/Regeln zur gemeinsamen Nutzung der Infrastruktur
 - → **Kommunikationsdienste** zur Übertragung von Informationen in verteilten Umgebungen
- Zugriff auf lokal nicht verfügbare Ressourcen
- Kostensenkung durch gemeinsame Nutzung von Betriebsmitteln
- Informationsgewinn durch entfernten Zugriff

Kommunikationsmodelle: Client-Server

- Server-Prozess: Langlebige Anwendung, die kontinuierlich auf Anfragen wartet, diese verarbeitet und beantwortet
- Client-Prozess: Zumeist kurzlebige Anwendung die Anfragen an den Server-Prozess stellt und auf die Antwort wartet. Die Rolle ist damit zumeist beendet

Initiierender Prozess

- stellt Anfragen
- Verarbeitet
- optionale Antwort



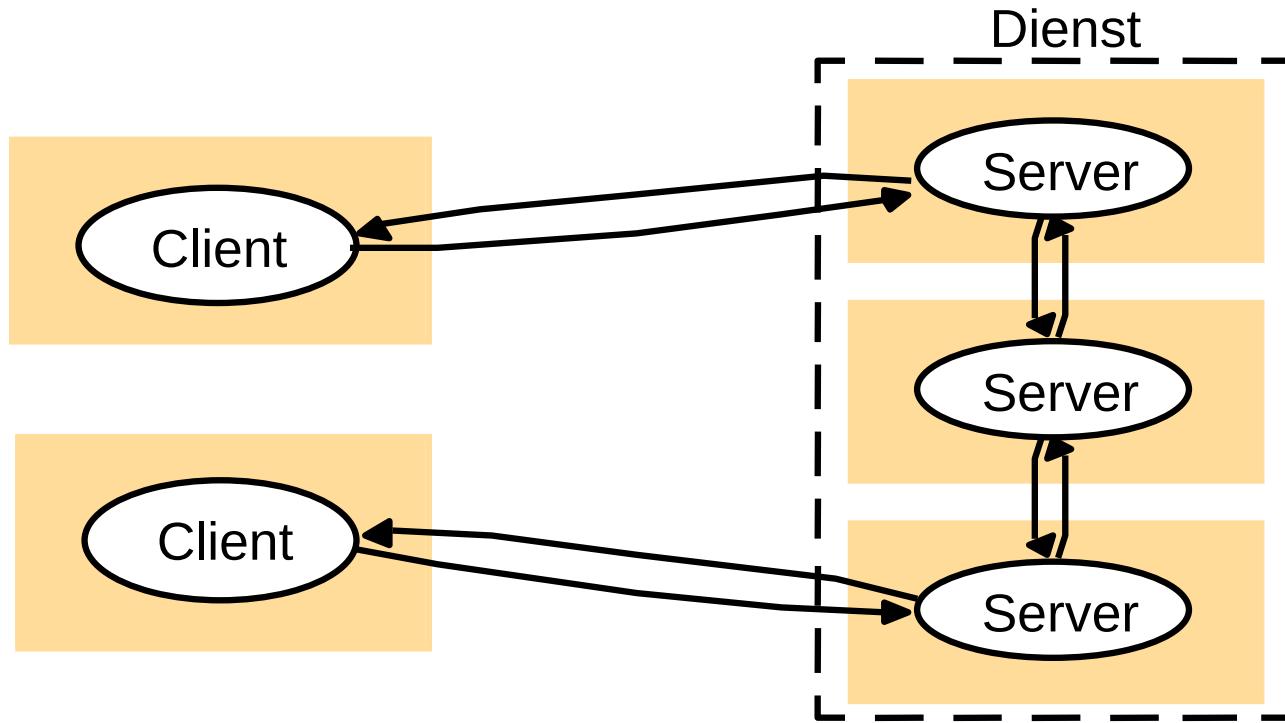
Merke: Ein Server kann auch andere Dienste nutzen und somit kann er auch gleichzeitig Client sein!

Reagierender Prozess

- bearbeitet Anfragen
- erfüllt Aufträge

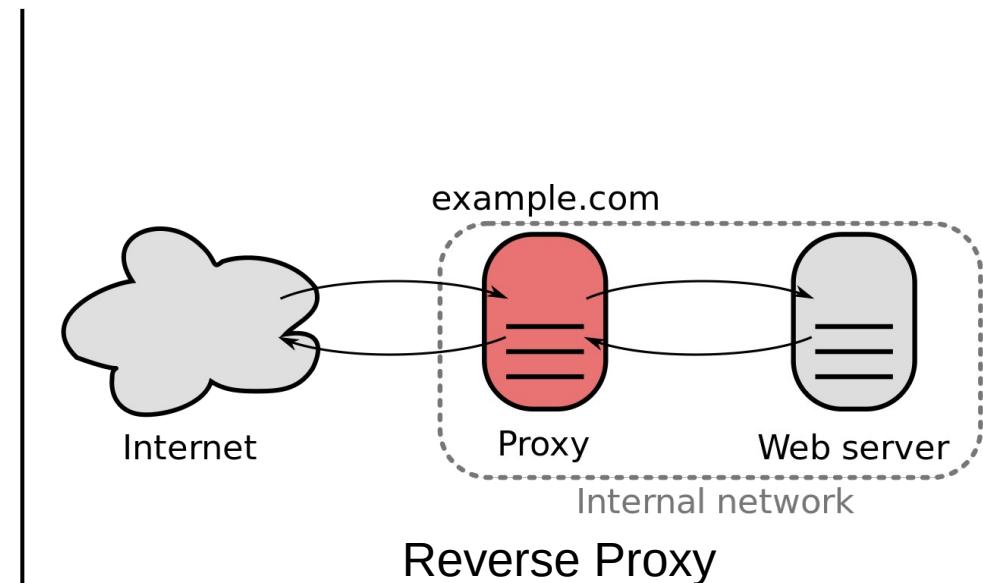
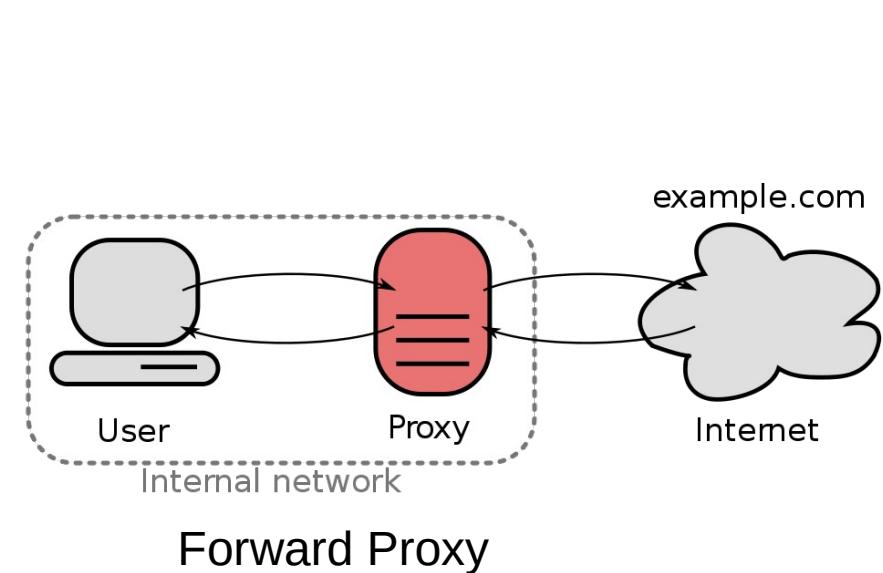
Client-Server Varianten

- Dienst wird von einem Verbund von Servern erbracht
- Erst durch den Verbund ergibt sich die Gesamtsicht
- Ggf. merkt der Client nichts vom Verbund



Client-Server Varianten

- (Forward) Proxy und Reverse-Proxy Modell
- Proxy zum Zwischenspeichern/Anonymisieren
- Proxy zum Lastbalanzieren von Webseiten

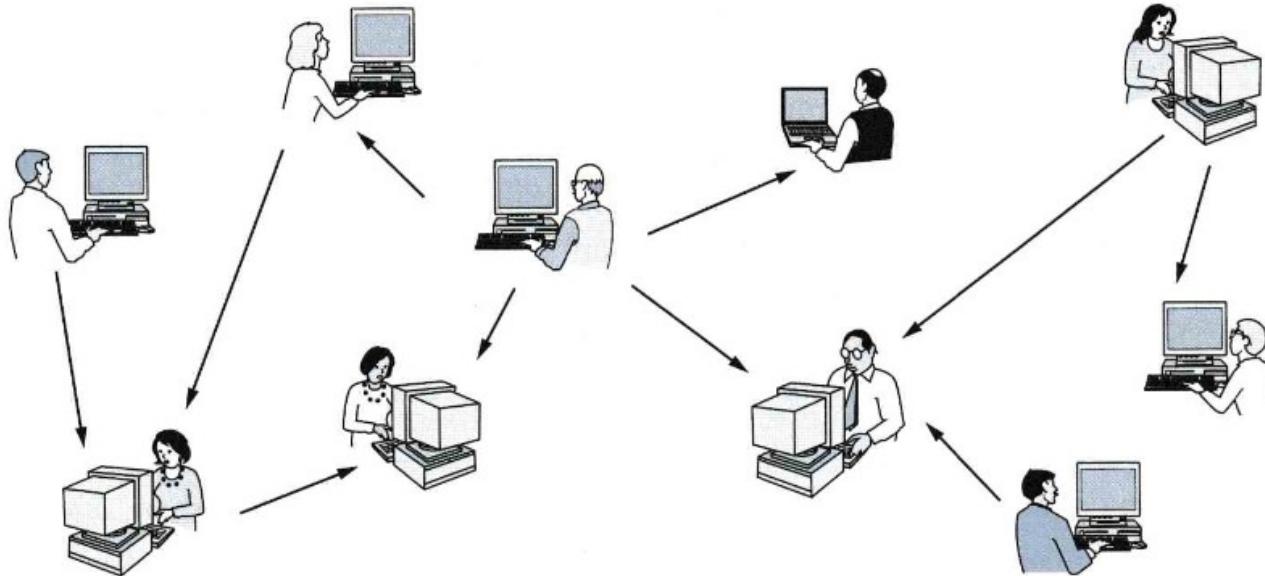


https://upload.wikimedia.org/wikipedia/commons/thumb/1/19/Forward_proxy_h2g2bob.svg/500px-Forward_proxy_h2g2bob.svg.png

https://upload.wikimedia.org/wikipedia/commons/thumb/6/67/Reverse_proxy_h2g2bob.svg/1280px-Reverse_proxy_h2g2bob.svg.png

Kommunikationsmodelle: Peer to Peer

- Gleichrangige Kommunikationspartner
- Oft bessere Leistung als Client-Server
- Übergreifender Datenbestand
- Beispiel: File-Sharing



Kommunikationsmodelle: Peer to Peer

Es ist die Aufgabe der **Datenkommunikation** (des Kommunikationsdienstes), die Information zwischen den beteiligten Systemen gemäß den Anforderungen zu übertragen

Datenkommunikation ist somit ein elementarer Dienst in verteilten Umgebungen

Klassifikation von Kommunikationsnetzen: Nach Übertragungstechnik

Point-to-Point (Punkt-zu-Punkt)

- Ein Paar von Rechnern ist durch eine direkte Leitung verbunden
- kein anderer Rechner kann diese Leitung nutzen
- Full-Duplex: Senden und Empfangen gleichzeitig möglich
- Half-Duplex: Nur eines von beiden gleichzeitig möglich
- Simplex: Daten können nur in eine Richtung fließen

Multi-Access-Netze

- Mehrere angeschlossenen Rechner teilen sich einen Übertragungskanal
- Damit Daten trotzdem an den richtigen Empfänger gesendet werden, müssen sie mit einer Zieladresse versehen werden
- Daten werden in Übertragungseinheiten (Frames) eingeteilt und mit der **Adresse** des Empfängers ausgewiesen
- „Rechner“ prüfen, ob die Nachricht für sie ist (aktiver Vorgang!)
- Sollen alle Stationen gleichzeitig eine Nachricht erhalten, so werden **Broadcast-Adressen** (spezielle Adressen zur Adressierung aller Stationen) verwendet

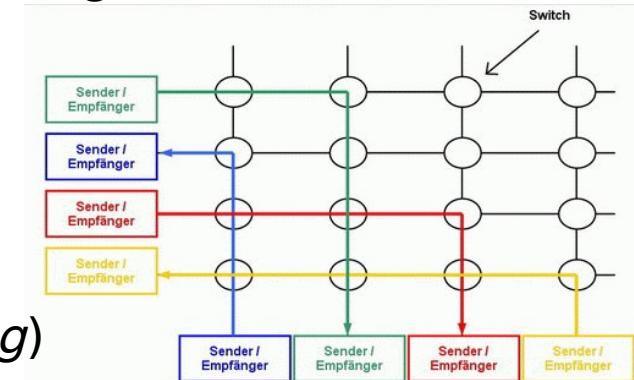
Klassifikation von Kommunikationsnetzen: Nach Verbindungsart

statische Netze

- fest verdrahtete Punkt-zu-Punkt Verbindungen oder Multi-Access-Netze
- jeder Knoten besitzt eine feste Anzahl von Nachbarn oder einen Zugang zu einem Multi-Access-Netze
- besitzen keine inhärent im Netz verankerte Vermittlungsfunktion
- Vermittlung über Netzgrenzen hinweg jedoch durch Weiterleiten möglich (Store-and-Forward, *Paketvermittlung*)

dynamische Netze

- Verbindungen enthalten konfigurierbare Schaltelemente
- diese können dynamisch vermitteln (Weg wird geschaltet, *Leitungsvermittlung*)
- ein- oder mehrstufiger Aufbau möglich
- Mit Aufwand blockadefreie Schaltungen ohne Crossbar



Klassifikation von Kommunikationsnetzen: Nach Topologieeigenschaften

Durchmesser (Diameter)

- Maximaler Abstand zweier Knoten, d.h. die Anzahl von Kanten
- → Ziel: Möglichst klein (Zeitbedarf für Übertragung)

Bisektionsbreite (Connectivity)

- Minimale Anzahl von Kanten die man entfernen muss, um das Netzwerk in zwei Hälften zu teilen
- → Ziel: Möglichst groß zur Verbesserung der Fehlertoleranz

Knotengrad

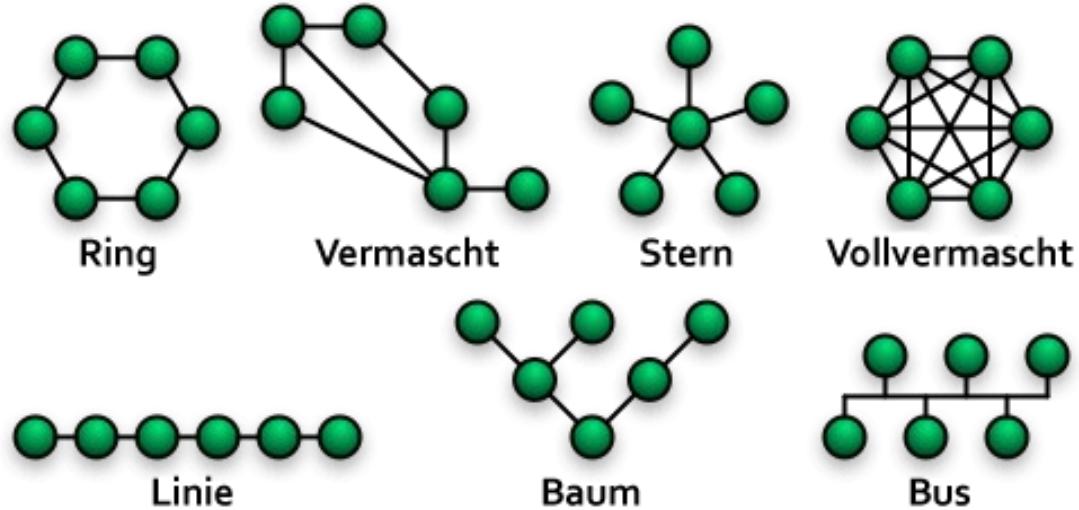
- Anzahl von Verbindungen eines Knotens zu seinen Nachbarn. Ist die Anzahl nicht konstant, so wird der das Maximum aller Knoten genommen
- → Ziel: Möglichst klein, da die Kosten so mit diesem Grad steigen

Topologieeigenschaften von Netzen

Merke:

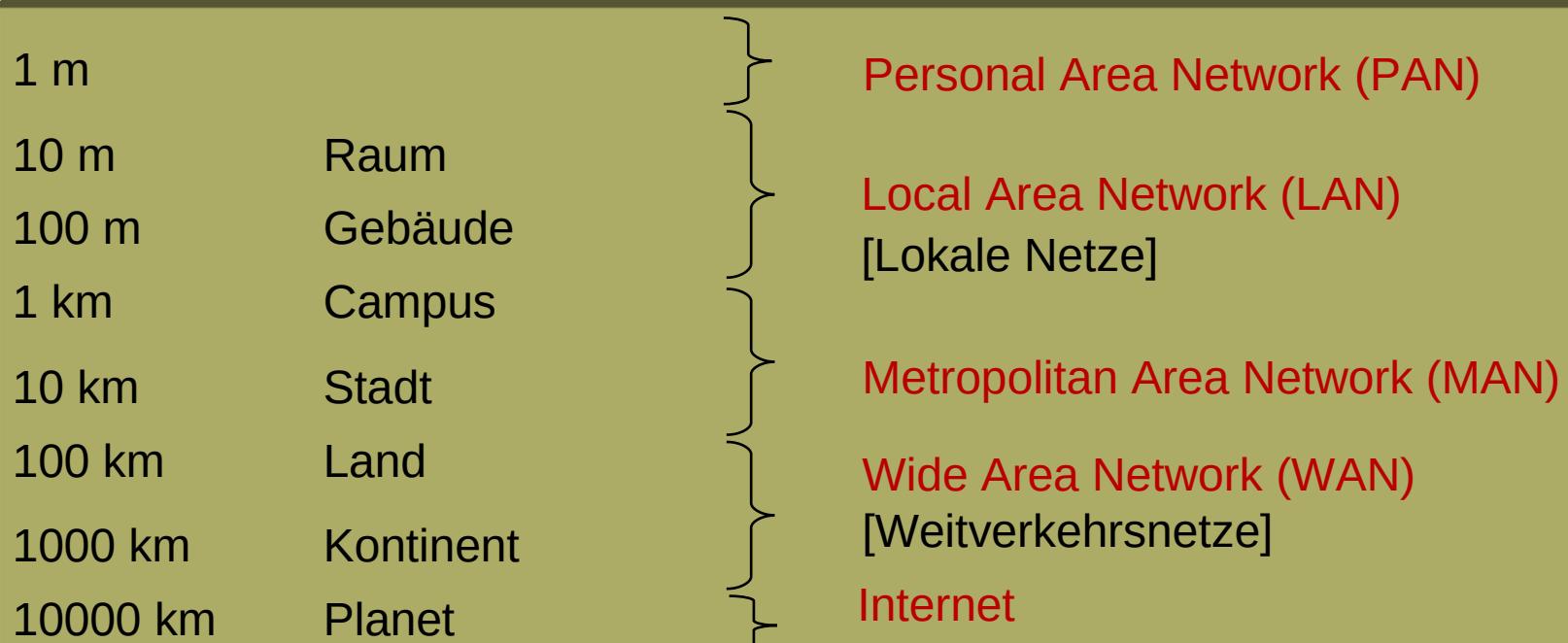
Haben alle Knoten den gleichen Grad, so spricht man von einem **regulären** Netz

Statische Netztopologien

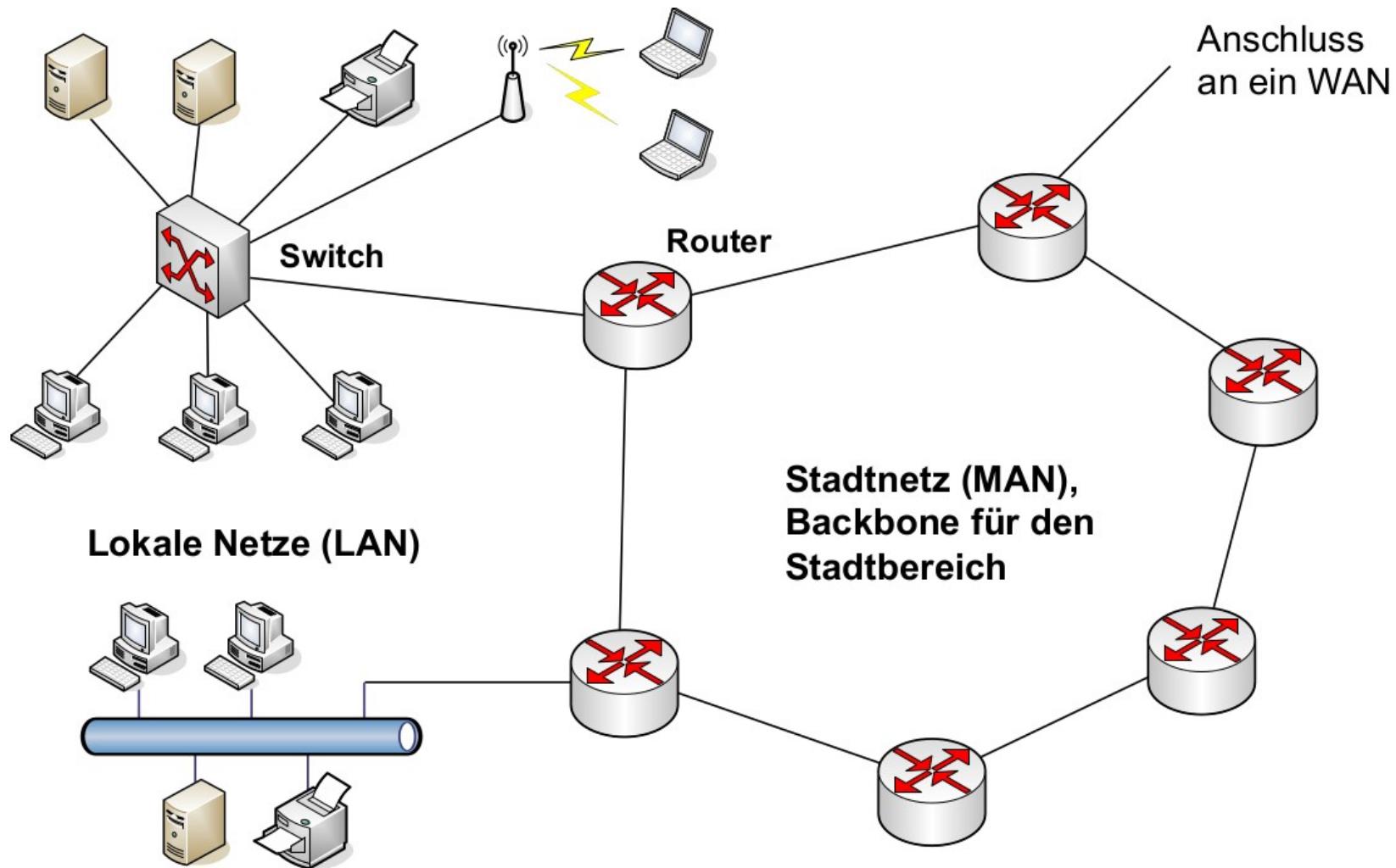


Topologie	Durchmesser	Bisektionsbr.	Knotengrad
Ring	$N/2$	2	2
Stern	2	1	1 bzw. N-1
Linie	$N-1$	1	1 bzw 2
Bus	1	1	1
Vollvermacht	1	$N/2 * N/2$	$N-1$
Baum (binär)	$2\log_2 N$	1	1, 2 oder 3

Klassifikation von Kommunikationsnetzen: Nach Ausdehnung



Klassifikation von Kommunikationsnetzen: Nach Ausdehnung



Klassifikation von Kommunikationsnetzen: Netzkomponenten

- **Switch**

Hat mehrere Anschlüsse, über die Rechner miteinander verbunden werden können. Er merkt sich, welcher Rechner an welchem Anschluss angeschlossen ist (Adresse der Netzwerkkarte) und kann Daten gezielt an einen Anschluss weiterleiten

- **Router**

Der Switch kennt nur Rechner, die direkt an ihn angeschlossen sind; will man Daten an weit entfernte Kommunikationspartner schicken, gibt es meist mehrere mögliche Wege, die man nehmen kann; hier muss also der Weg zu dem entfernten Rechner bestimmt werden. Router verwalten globale Adressinformationen, kennen kürzeste Wege zu allen Rechnern und können Daten gezielt in andere Netze weiterleiten

- **Backbone** (engl.: Rückgrat)

Als Backbone bezeichnet man eine Menge von Rechnern, die miteinander verbunden sind (üblicherweise Punkt-zu-Punkt-Verbindungen, über große Entfernung), um kleinere Netze miteinander zu koppeln und so den Datenaustausch zwischen diesen zu ermöglichen

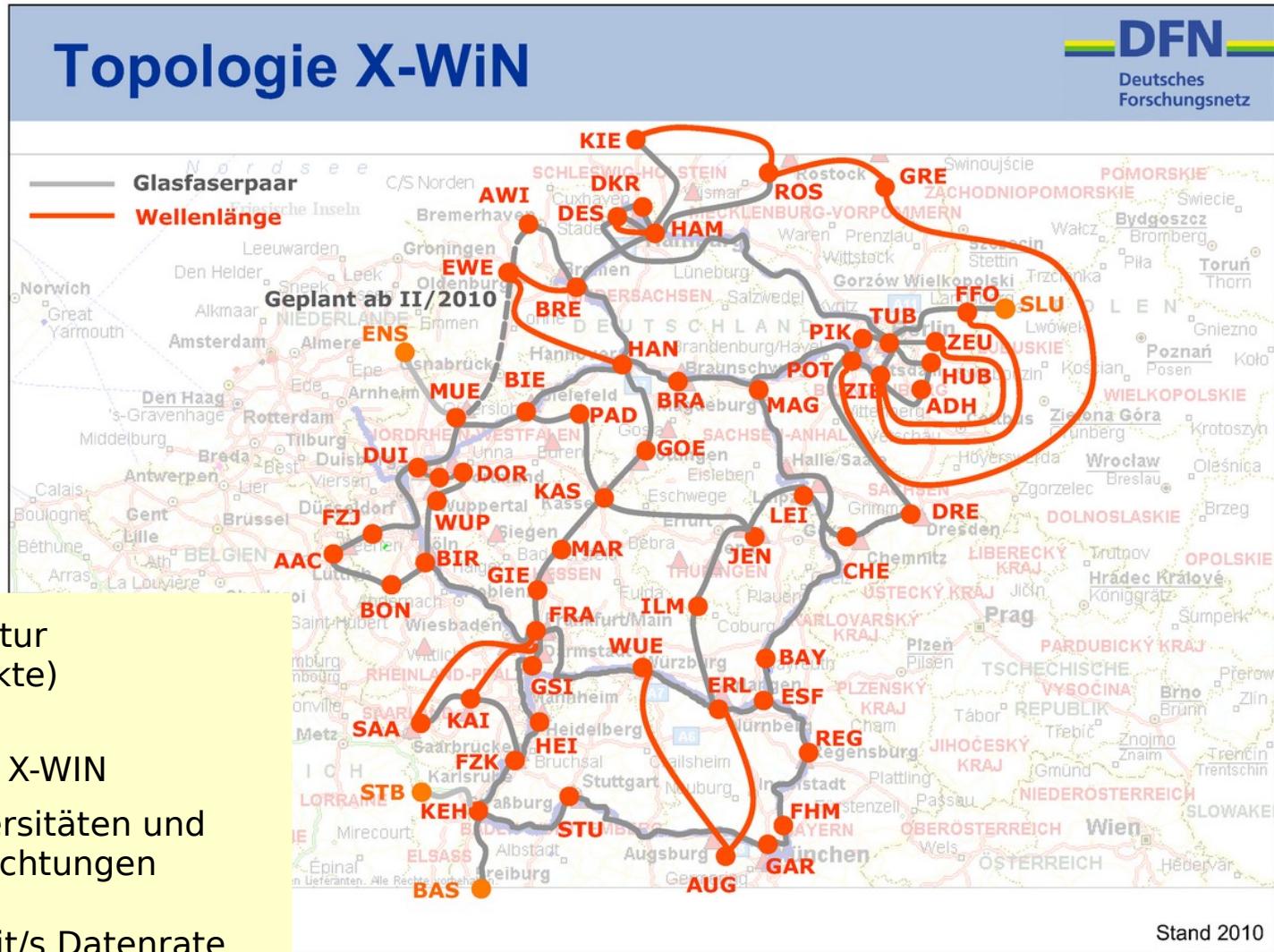
Local Area Network (LAN)

- Kommunikationsinfrastruktur für einen begrenzten geographischen Bereich (10m - wenige km)
- **Üblicherweise im Besitz einer Organisation**
- Übertragungskapazität bis zu 10.000 Mbit/s
- Übertragungsdauer einer Nachricht im unteren Millisekundenbereich (<10 ms)
- Einfache Verbindungsstruktur (“Simple is beautiful”), zumeist mit einheitlicher Technik
- Wichtigstes Beispiel: **(Gigabit-)Ethernet**

Metropolitan Area Network (MAN)

- Überbrücken größere Distanzen als ein LAN, Einsatz z.B. im Städtebereich
- Oftmals Zusammenschaltung mehrerer LANs
- Struktur zumeist regulär
- Wichtiger Unterschied: Wegerechte erforderlich
- Faktisch geht der Trend zur Nutzung von optischer Ethernet-Techniken (Metro-Ethernet)

Wide Area Network (WAN)

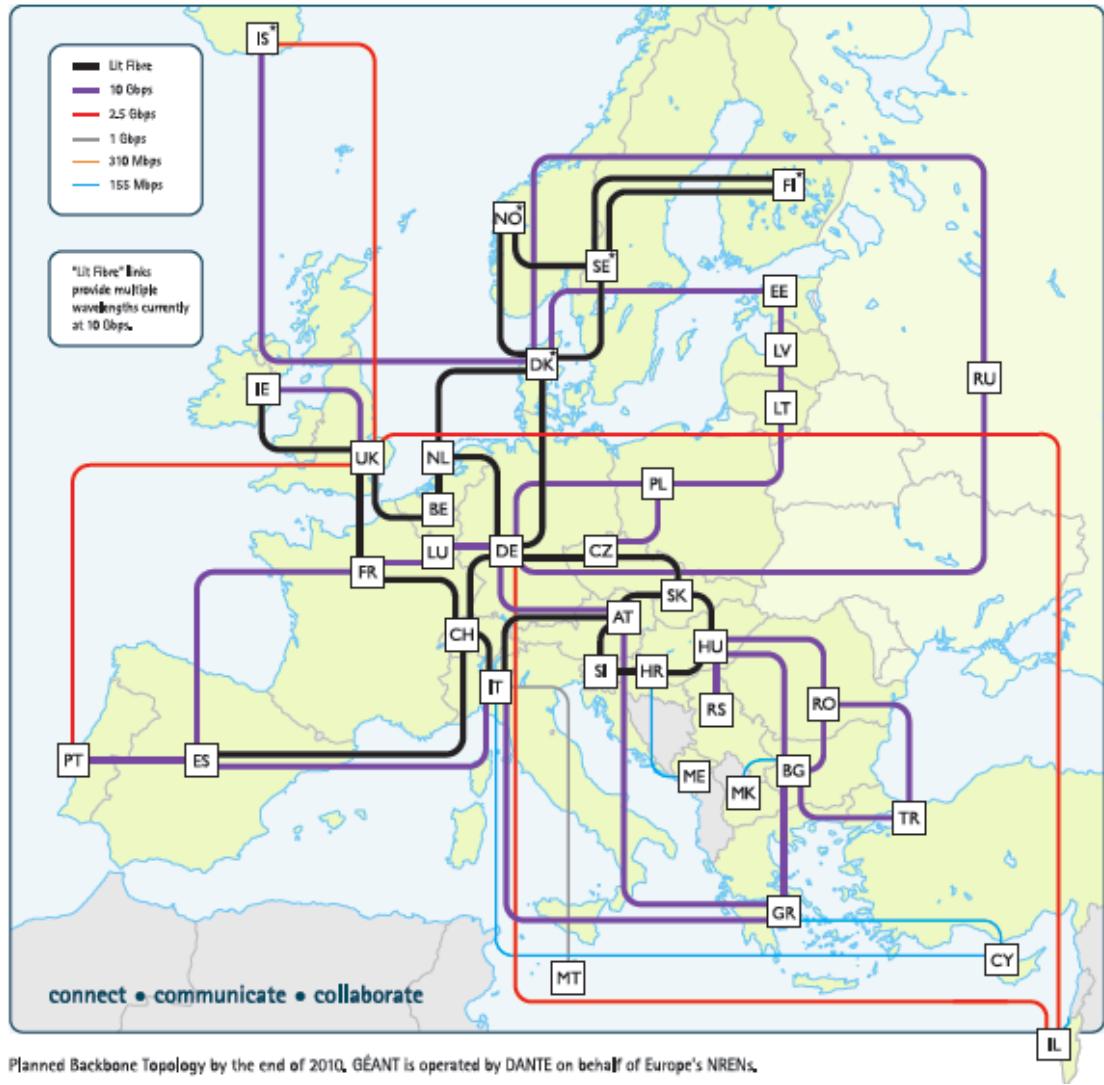


- Irreguläre Struktur
(Übergangspunkte)
 - Hier: deutsches
Forschungsnetz X-WIN
 - verbindet Universitäten und
Forschungseinrichtungen
in Deutschland
Geplant: 1.6 TBit/s Datenrate

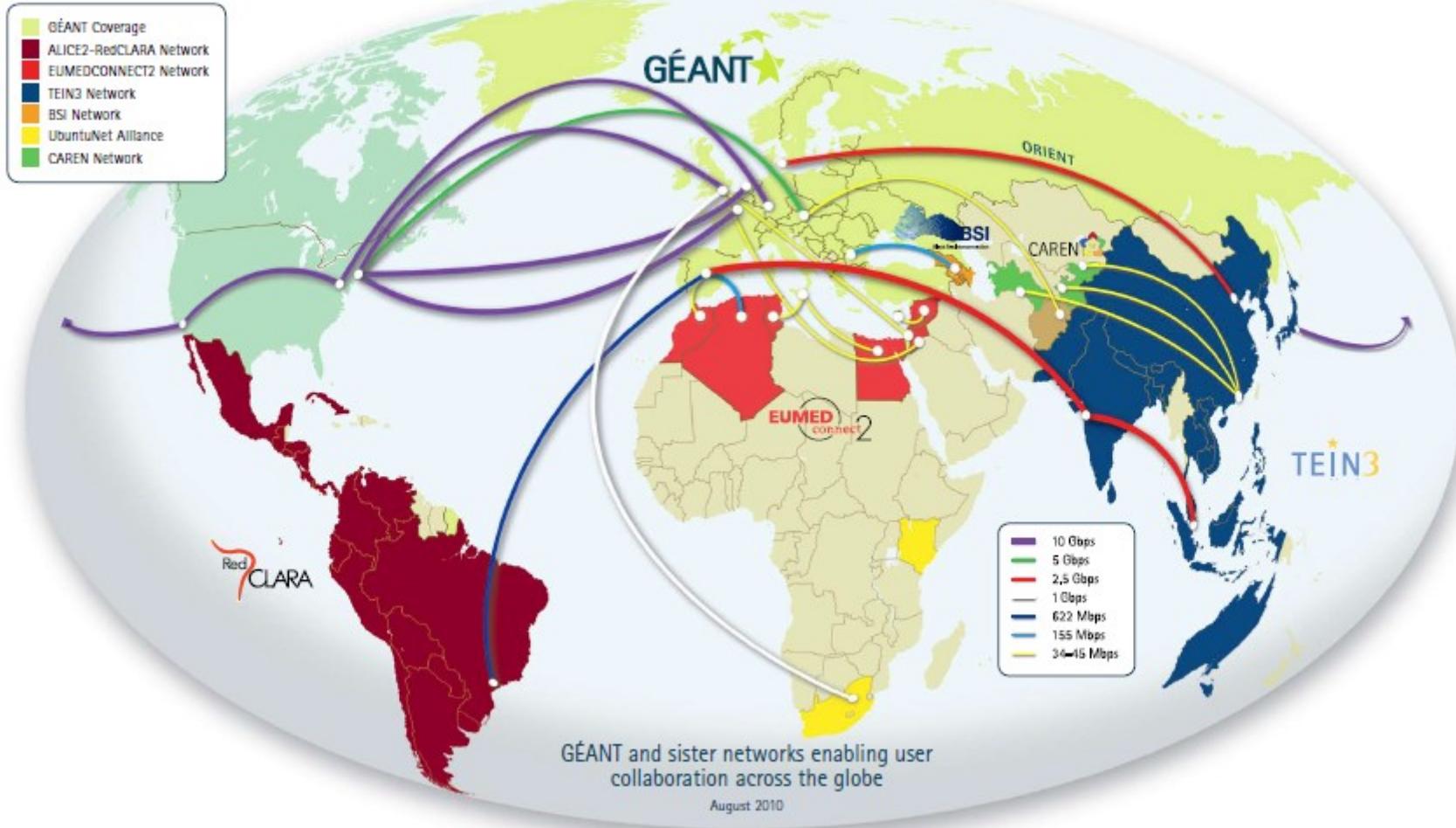
Wide Area Network (WAN)

Zentraler Kontenpunkt
Frankfurt – Anschluss an
das europäische
Wissenschaftsnetz
Geant

Weiterhin in Frankfurt
und Hamburg:
interkontinentale
Anschlüsse.



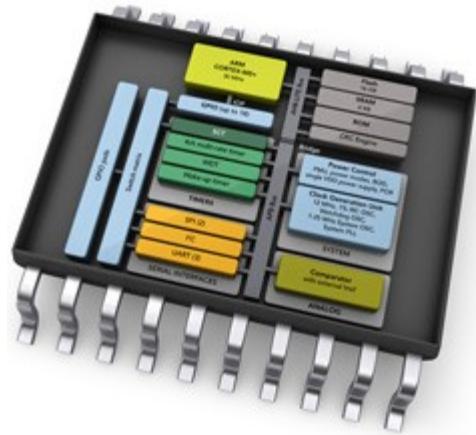
Wide Area Network (WAN)



Kommunikationssysteme

(Modulcode 941306)

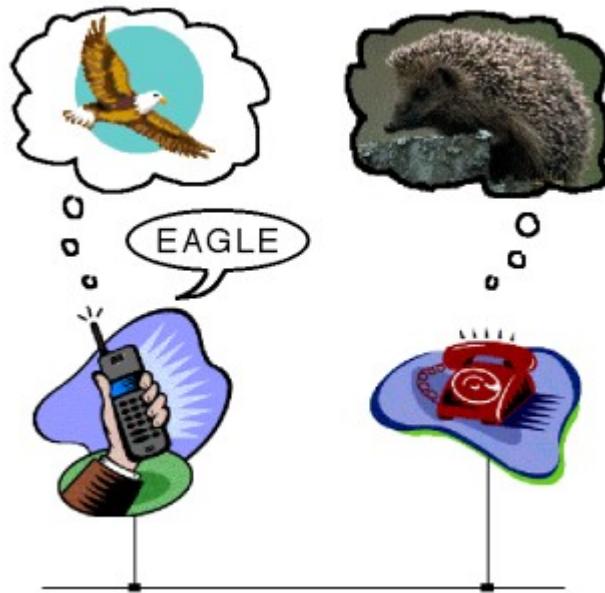
Prof. Dr. Andreas Terstegge



Nun wissen wir, wie die Verbindungsstruktur
aussehen kann, aber **wie** kommunizieren
Rechner und wie Anwendungen miteinander?

Hierzu müssen die Anwendungen

- sich finden
- sich verstehen
- wissen wann man dran ist zu „sprechen“



Netzwerkprotokolle

Um eine Kommunikation durchzuführen, müssen sich die Kommunikationspartner an bestimmte Regeln halten und die gleiche „Sprache“ sprechen

Auch bei Datennetzen ist eine Vereinbarung über den Austausch von Daten zwingend erforderlich:

- Übertragungsrichtung (wer redet, wer hört zu?)
- Datenformat und -kodierung (Sprache)
- Wege-Ermittlung (wie kommen die Daten zum Empfänger?)
- Fehlerbehandlung (was macht man bei Kommunikationsfehlern?)
- ...

Ein Protokoll ist die Gesamtheit aller Vereinbarungen zwischen Computeranwendungen zum Zweck einer gemeinsamen Kommunikation

Unter **Netzwerk** versteht man mehr als nur das verbindende „Kabel“

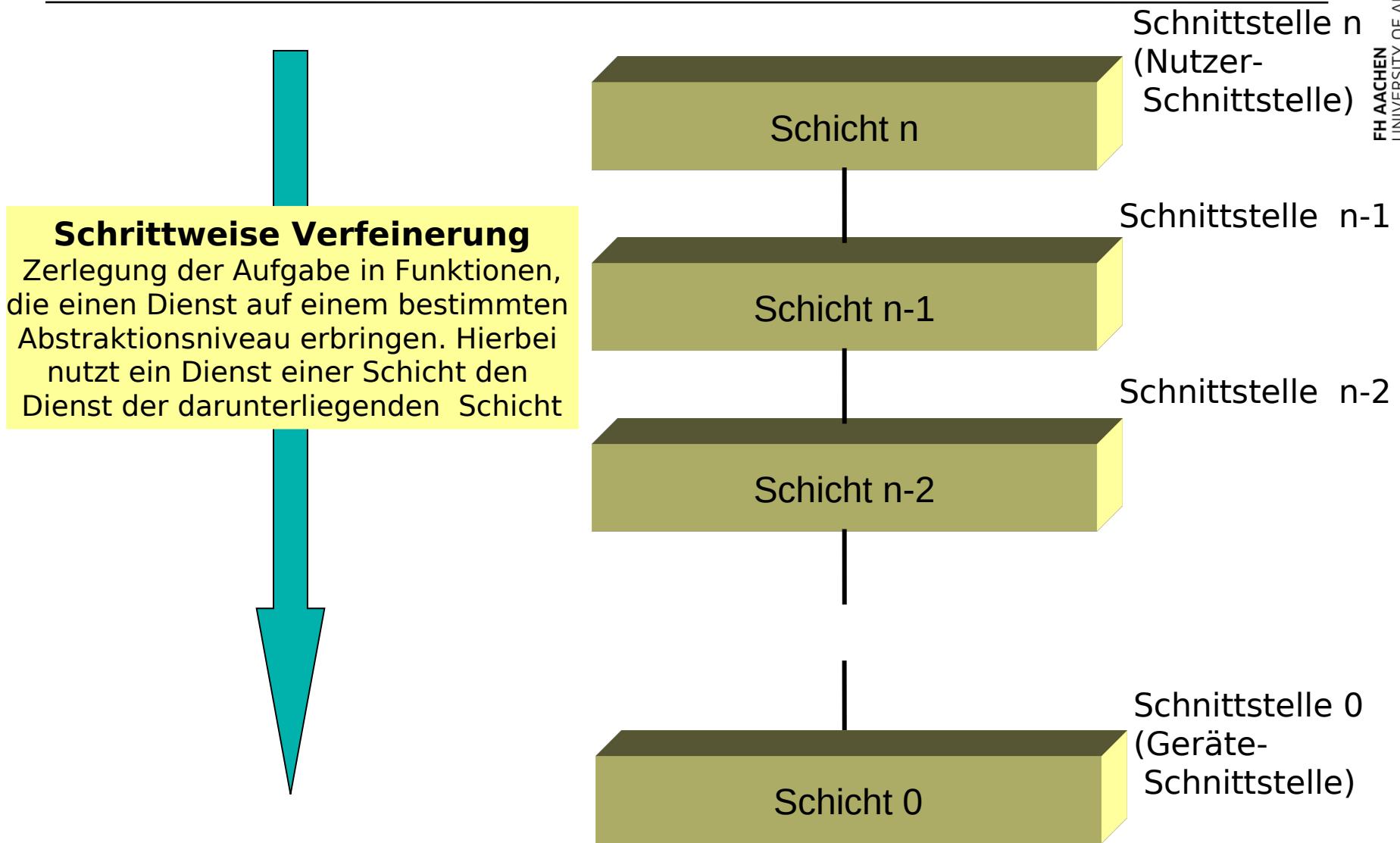
- Wie sind die physikalischen Eigenschaften des Übertragungsmediums?
- Wie werden die Daten als Signal dargestellt?
- Wie sind die Zugriffsregeln der angeschlossenen Systeme?
- Welchen Aufbau (Topologie) unterstützt das System?
- Wie können Übertragungsfehler erkannt werden?
- Wie werden Endpunkte gefunden (adressiert)?
- Wie können entfernte Systeme in anderen Netzen erreicht werden?
- Wie werden die Daten bei den End-Systemen übermittelt?
- Welche Kodierungsregeln und Semantiken gibt es hierbei?
- ...

Protokolle können sehr komplex sein!
Hier ist das Hilfsmittel der Abstraktion
von Bedeutung

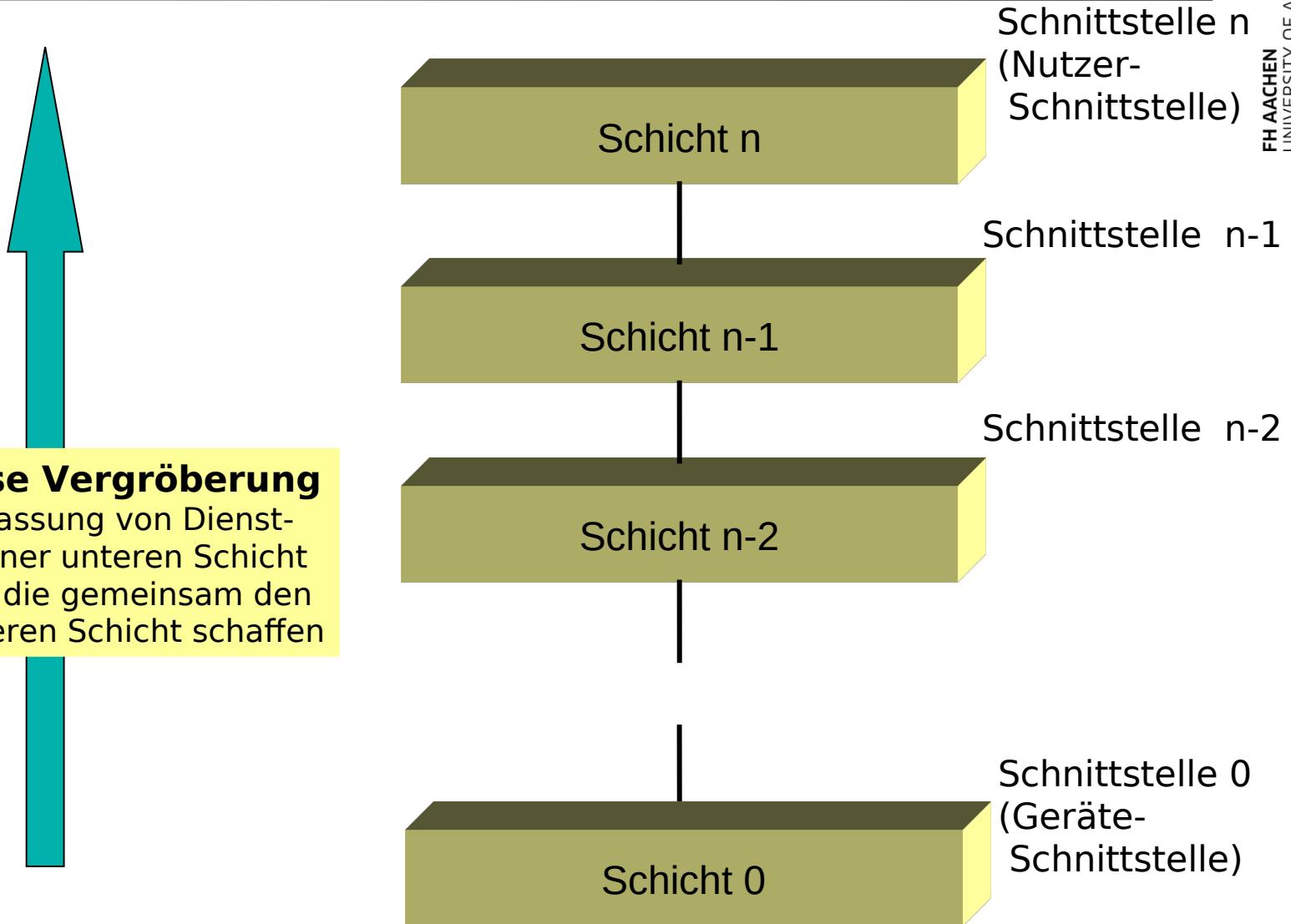
- Lösung 1: „Naives“ Konzept
 - Schreibe ein großes “Kommunikationsprogramm”, welches allen beschriebenen Anforderungen genügt
 - Vorteil: Für eine gegebene Anwendung optimal und effizient
 - Nachteil: Nicht flexibel! Änderungen erfordern hohen Aufwand
- Lösung 2: Modularisierung
 - Schreibe kleinere, für eine Aufgabe spezialisierte Programme, die sich kombinieren lassen
 - Vorteil: Sehr flexibel, da einzelne Komponenten austauschbar
 - Nachteil: Durch vorgegebene Struktur wird vieles umständlich; dadurch nicht so effizient

Realisiert durch **Schichtenmodelle**

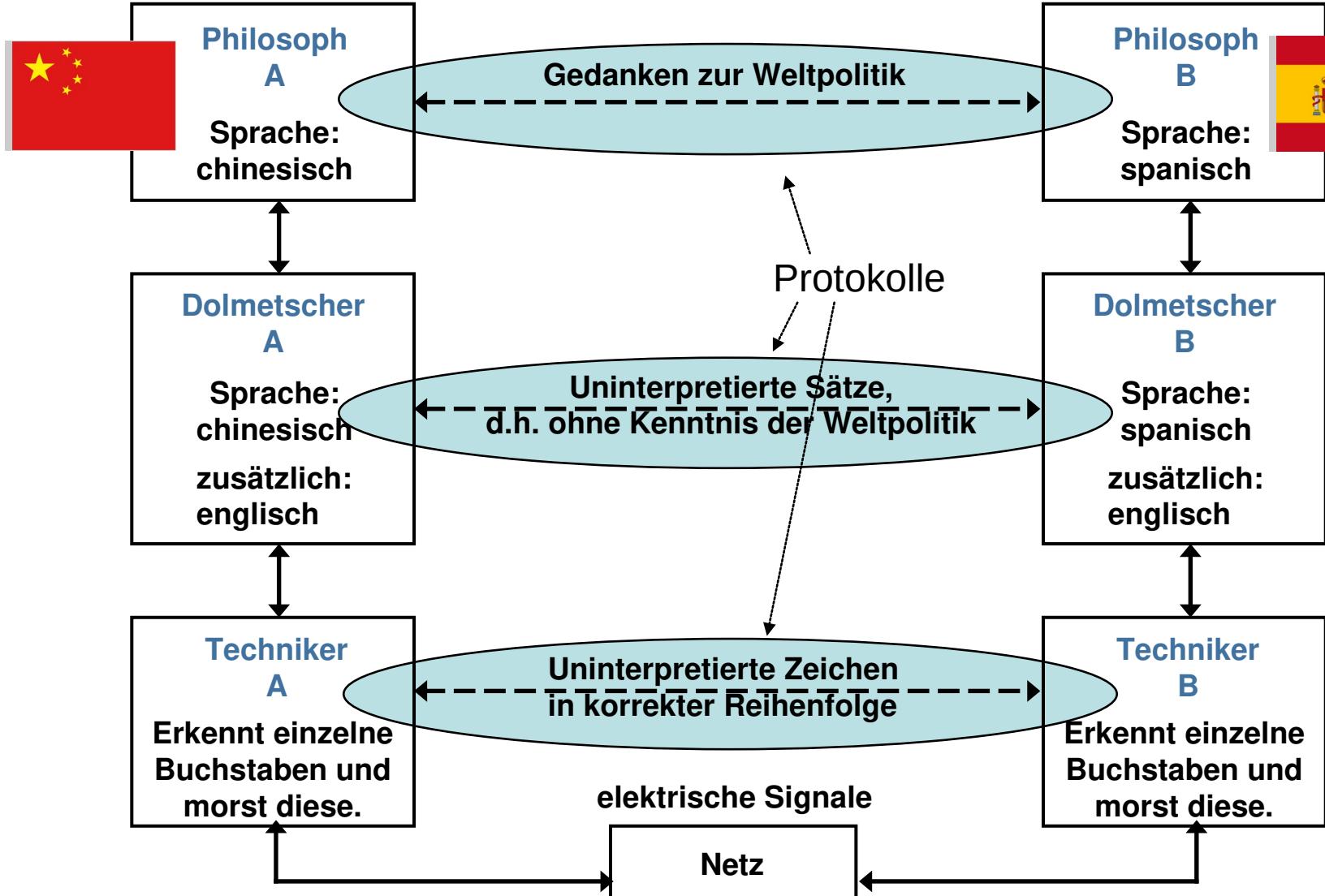
Systemschichtung



Systemschichtung



Beispiel: Gedankenaustausch von Philosophen



Protokolle sollten also unterschiedliche
Abstraktionsebenen adressieren!

Standardisierung der Protokolle

Ziel ist die offene Kommunikation, d.h. die im technischen Sinne unlimitierte Kommunikation zwischen kommunikationswilligen Partnern

Aus offener Kommunikation folgt die Notwendigkeit einer **Standardisierung** aller Netzkomponenten incl. der verwendeten Protokolle

Entstehung und Bedeutung von Standards (teilweise ironisch)

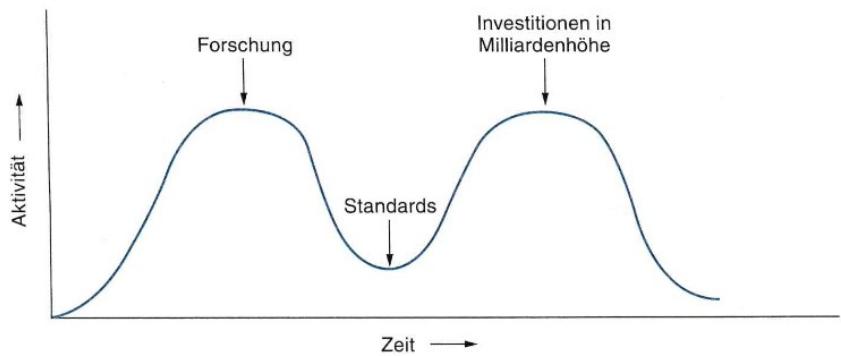
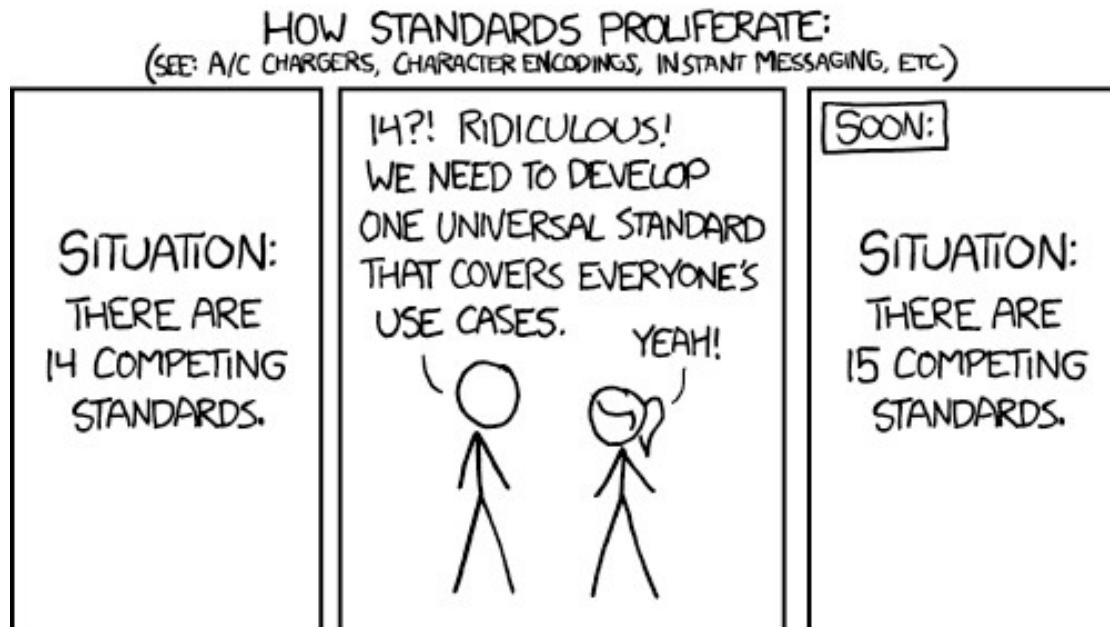


Abbildung 1.24: Die Apokalypse der zwei Elefanten.

Standardisierungsgremien - Auswahl



International Standards Organization - ISO

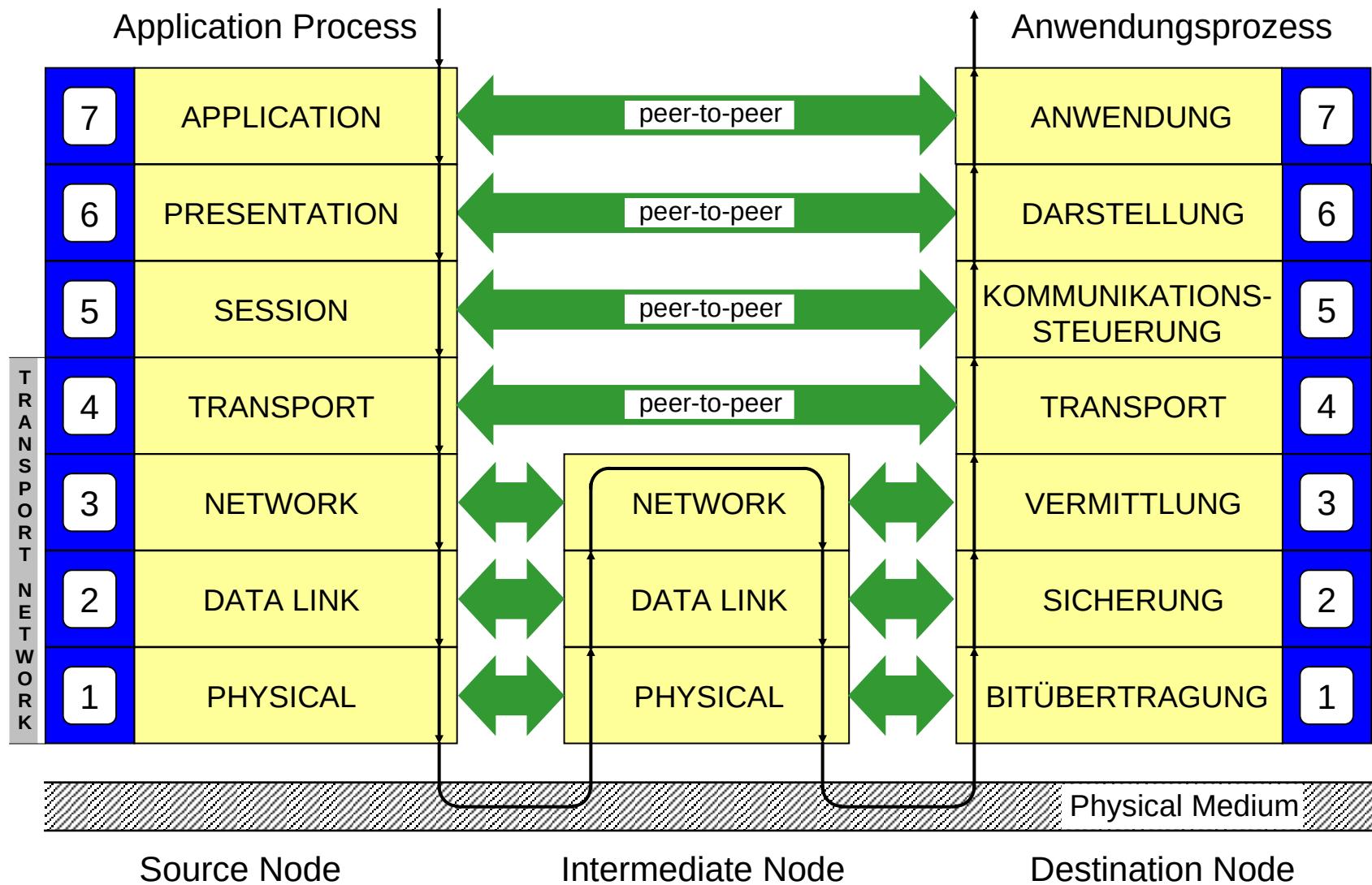
- Organisation, die auf freiwilliger Basis arbeitet (seit 1946).
- Mitglieder: Standardisierungsorganisationen von ca. 90 Ländern.

www.iso.org

- Beschäftigt sich mit einem sehr weiten Spektrum von Standards
- Hat 200 Technical Committees (TC) mit spezifischen Aufgaben (z.B. TC97 für Computer und Informationsverarbeitung)
- TC haben Subkomitees, die wiederum in Arbeitsgruppen unterteilt sind.
- Zusammenarbeit mit ITU-T bzgl. Telekommunikationsstandards, (ISO ist Mitglied von ITU-T)
- **Bahnbrechende Leistung im Bereich der Datenkommunikation: Das ISO/OSI-Referenzmodell.**
- Bahnbrechend bzgl. des Konzepts, nicht wegen der daraus entstandenen Produkte!

(OSI: Open Systems Interconnection)

Das ISO-OSI-Referenzmodell für offene Systeme

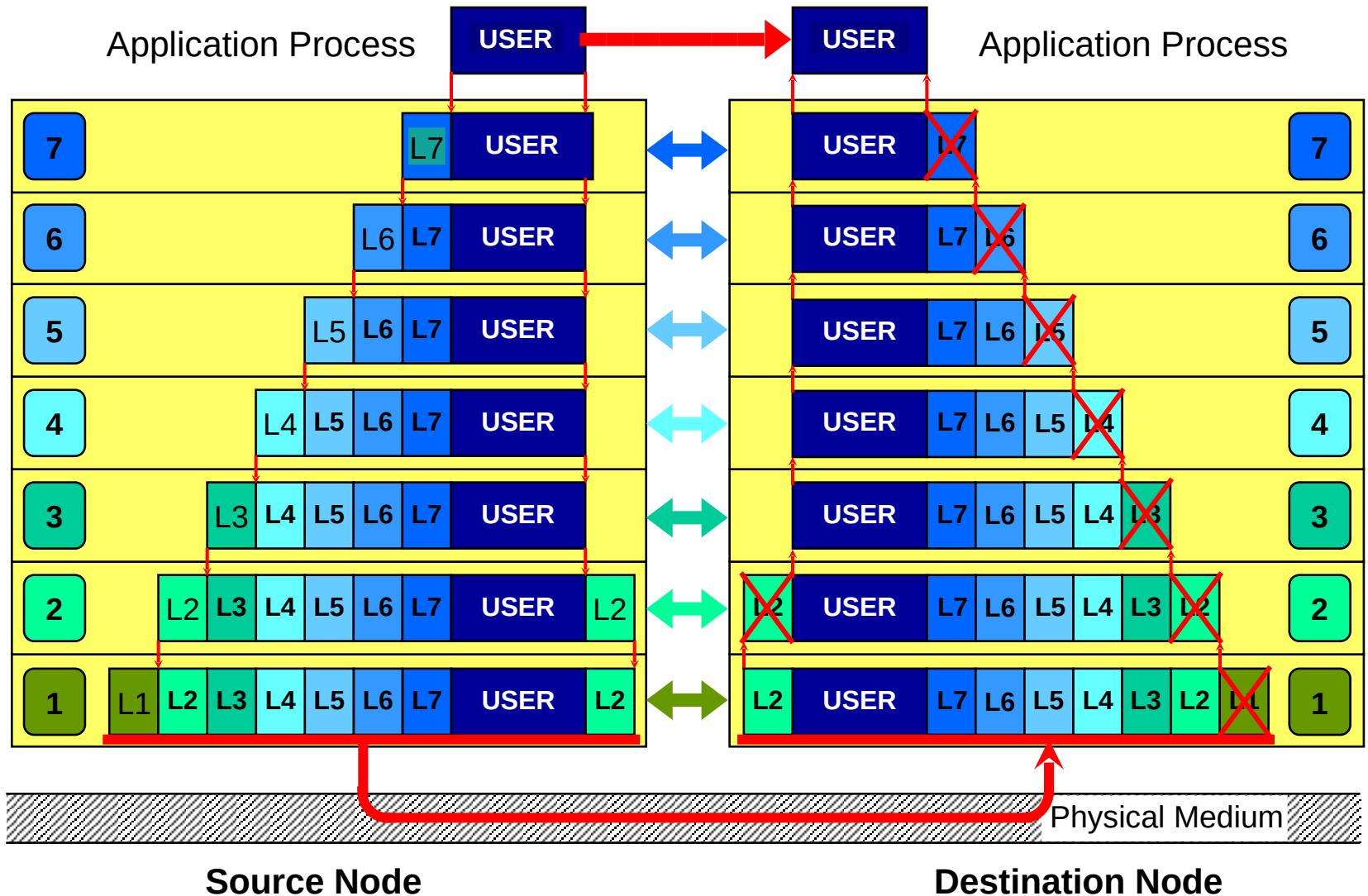


Wechselspiel zwischen den Schichten

- Interaktion der Schichten
 - Eine Schicht ($n-1$) bietet der über ihr liegenden Schicht n Dienste an
 - Schicht n versieht ihre Nachricht mit Kontrollinformationen (**Header**) und versendet alles zusammen (oftmals **Protocol Data Unit** (PDU) genannt)
 - Zwei Kommunikationspartner auf Schicht n tauschen PDUs aus und nutzen dazu die Dienste der nächsttieferen Schicht ($n-1$)
 - Für Schicht ($n-1$) sind diese PDUs die zu übertragenden Daten

Merke: Schicht 2 wird hiervon abweichen. Tatsächlich findet hier ein Framing statt, welches Synchronisation und Fehlererkennung/-korrektur vereinfacht

Das ISO-OSI-Referenzmodell für offene Systeme



Aufgaben der Schichten

- **Schicht 7: Anwendungsschicht (Application Layer)**
 - In dieser Ebene werden **(Standard-)Schnittstellen** zur Verfügung gestellt, die **bestimmten Anwendungstypen** ganze Kommunikationsdienste bereitstellen
 - Ein Beispiel hierfür ein allgemeingültiges Protokoll zur Übertragung von Webseiten samt fest definierter Schnittstelle (GET, POST, DELETE, ...) sein. Wer einen Webbrowser oder einen Webserver implementieren will, könnte dann diese Schnittstelle zur Kommunikation mit den Produkten anderer verwenden

Merke: Das Internet realisiert dies anders

Aufgabe der Schichten

- **Schicht 6: Darstellungsschicht (Presentation Layer)**
 - Beschäftigt sich damit, die zu übertragenden Daten so **darzustellen**, dass sie **von vielen unterschiedlichen Systemen** gehandhabt werden können
 - Beispielsweise codieren manche Rechner einen String mit ASCII-Zeichen, andere benutzen Unicode, manche benutzen bei Integern das 1-, andere das 2-Komplement. Problematisch ist auch die Byteordnung des Prozessors (Big/Little-Endian)
 - Formal wird hier das verwendete Format beschrieben
 - Anstatt für jede Anwendung eine eigene **Übertragungssyntax** und **-semantik** zu definieren, stellt man hier eine allgemeingültige Lösung bereit
 - Die spezifische Daten eines Rechners werden hier eindeutig beschrieben

Aufgabe der Schichten

- **Schicht 5:** Sitzungsschicht (Session Layer)
 - Öffnen, Schließen und Managen einer **Sitzung** zwischen Applikationen, d.h. Aufbau eines semi-permanenten Dialoges
 - **Dialogkontrolle**, d.h. es kann festgelegt werden, welcher Kommunikationspartner wann übertragen darf (wer redet, wer hört zu?). Da wir bei ISO/OSI eigentlich eine Steuerung über einen Header benötigen könnte hierzu ein **Token verwendet werden**. Bei bestimmten Operationen darf dann nur der Kommunikationspartner, der im Besitz des Tokens ist, diese Operation durchführen
 - Wichtiger Ansatz wäre auch die Bereitstellung von **Wiederaufsetzpunkten**. Wurde beispielsweise eine 2-stündige Dateiübertragung mittendrin durch einen Ausfall unterbrochen, so braucht nicht die gesamt Übertragung wiederholt werden, sondern man geht nur bis zum letzten Aufsetzpunkt zurück

Aufgabe der Schichten

- **Schicht 4:** Transportschicht (Transport Layer)
 - Ermöglicht die **Kommunikation zwischen Anwendungen** der Endsysteme
 - Segmentierung von Datenströmen zur Übertragung der Daten in Einheiten: Datagramme (Paketen)
 - Verbergen wesentlicher Charakteristika der Netzinfrastruktur
 - Aufgabe: Transport der Daten zwischen den Kommunikationspartnern mit bestimmten (aushandelbaren) **Dienstmerkmalen**
 - **Adressierung von Anwendungsprozessen**
 - Eventuell Regeln zur *Behandlung von Fehlern*
 - Eventuell **Flusskontrolle/Staukontrolle** zur Anpassung der Datenrate an die Fähigkeiten des Netzes und des Empfängers

Aufgabe der Schichten

- **Schicht 3:** Vermittlungsschicht (Network Layer)
 - **Übertragung der Daten zwischen Rechnern in einem Netz aus Netzen**
 - Hauptaufgabe ist dabei, eine geeignete Wegewahl (*Routing*) zu treffen
 - Eine notwendige Voraussetzung sind dazu u.a. ein gemeinsamer *Adressraum für Rechner* und eine Einigung auf eine *maximale PDU-Größe (Datagramm-Größe)*
 - Statisches Netzkonzept: Zwischenknoten speichern ankommende Nachrichten zwischen und ermitteln (über Tabellen) den Teilnehmer, der die Daten als nächstes erhält. Hierbei muss man mit diesem direkt kommunizieren können.
 - Weiterhin: Multiplexing mehrerer logischer Verbindungen über eine physikalische Verbindung

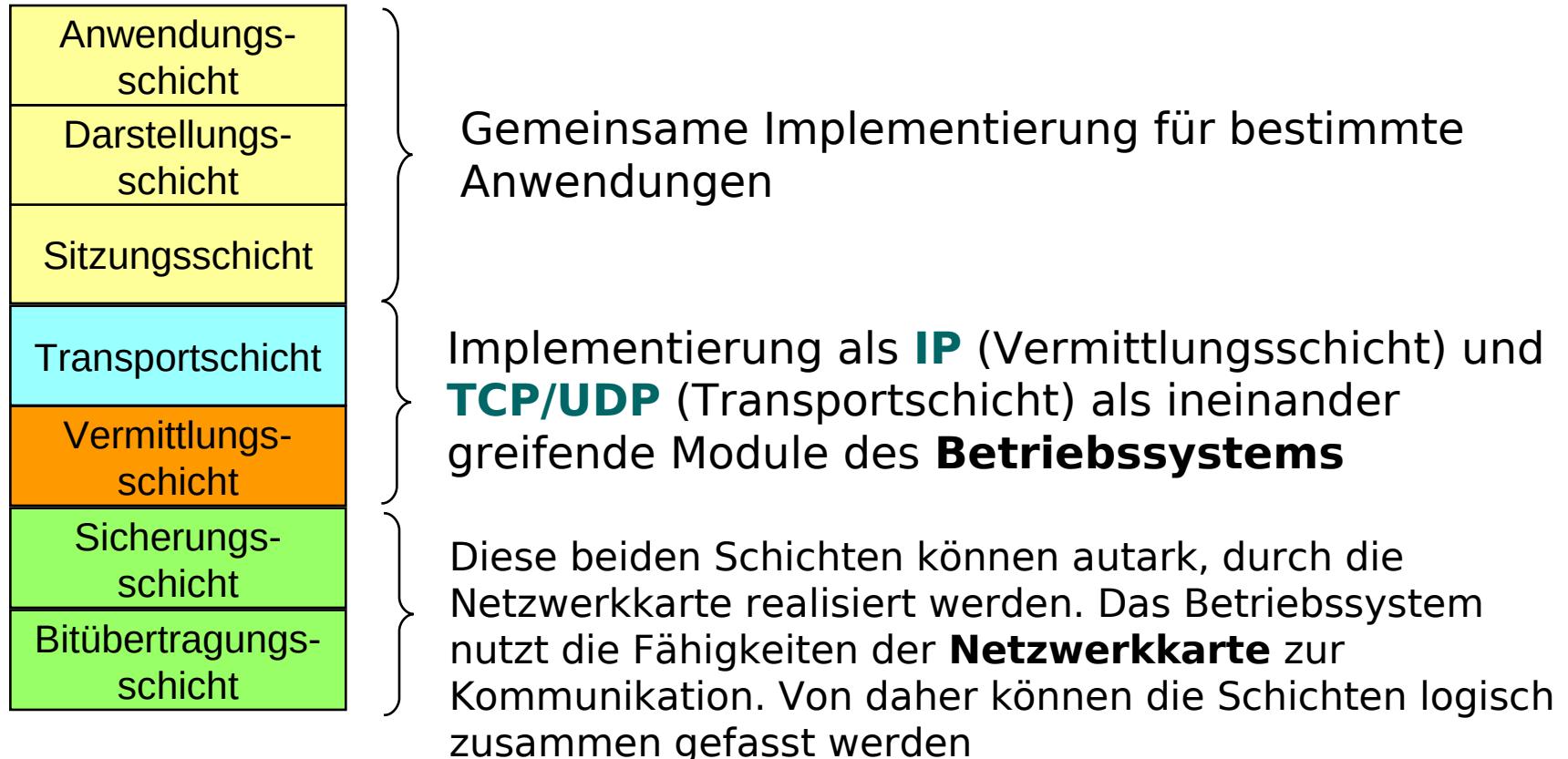
- **Schicht 2: Sicherungsschicht (Data Link Layer)**
 - Kommunikation zwischen Rechnern in einem einzelnen Netz
 - Logical Link Control (LLC):
 - Liefert der Vermittlungsschicht eine *fehlerfreie Übertragung* der Daten zwischen zwei Rechnern (z.B. innerhalb eines lokalen Netzes)
 - Dazu werden die ankommenden Daten in sog. *Rahmen* unterteilt, die einzeln übertragen werden
 - Der Empfänger überprüft, ob die Übertragung korrekt war (z.B. mittels einer *Prüfsumme*). Im Fehlerfall wird der entsprechende Rahmen neu angefordert
 - Weiterhin wird versucht, eventuell auftretende Staus durch *Flusskontrolle* zu vermeiden, z.B. wenn der Empfänger überlastet ist.
 - Medium Access Control:
 - Bei lokalen Netzen wird außerdem der *konfliktfreie Zugriff* auf das Netz geregelt, es können ja ggf. nicht mehrere Teilnehmer gleichzeitig senden

Aufgabe der Schichten

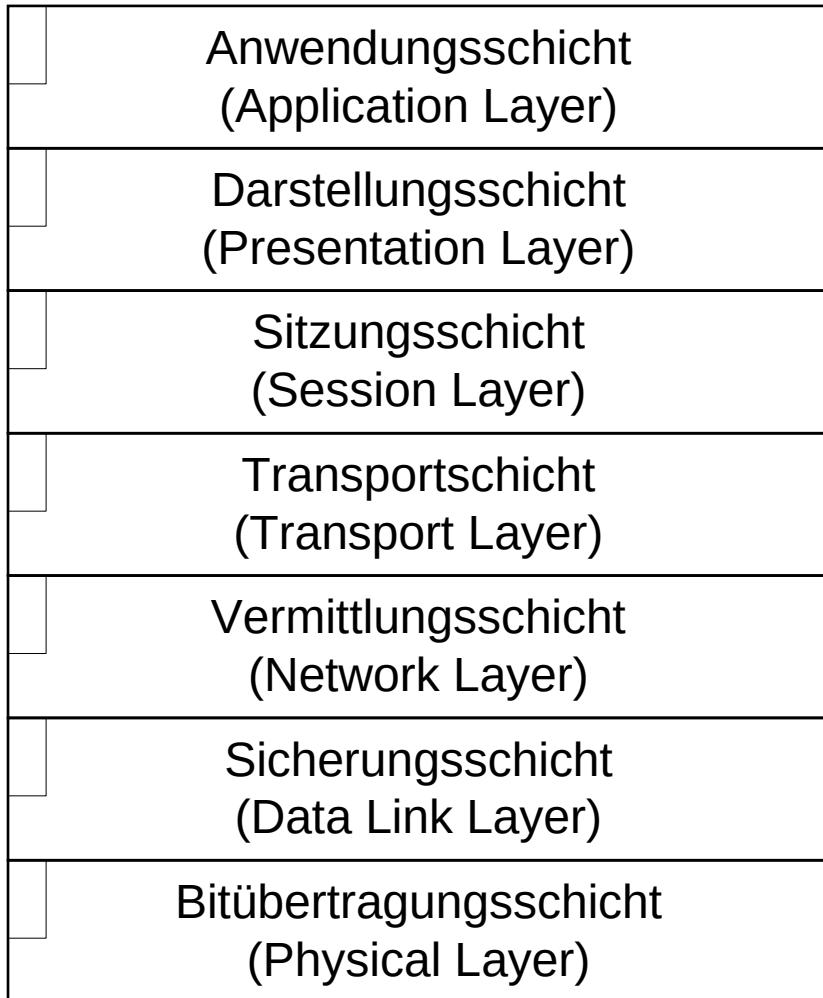
- **Schicht 1:** Bitübertragungsschicht (Physical Layer)
 - Transportiert die einzelnen Bits über eine bestimmte physikalische Leitung (Medium)
 - D.h. es muss festgelegt werden, welchen *Leistungstyp* man benutzt und wie eine “1” bzw. eine “0” auf der Leitung *kodiert* werden
 - Dazu legt man z.B. bei Verwendung von Kupferkabel als Leitung fest, dass Bits als Spannungspulse übertragen werden (z.B. „Übertrage für eine Millisekunde +1 Volt, um eine 1 zu transportieren“)
 - Weiterhin wird definiert:
 - Stecker (*Pinbelegungen*),
 - *Übertragungsrichtung* (uni-/bidirektional) ,
 - ...

Schichtenmodelle in der Praxis

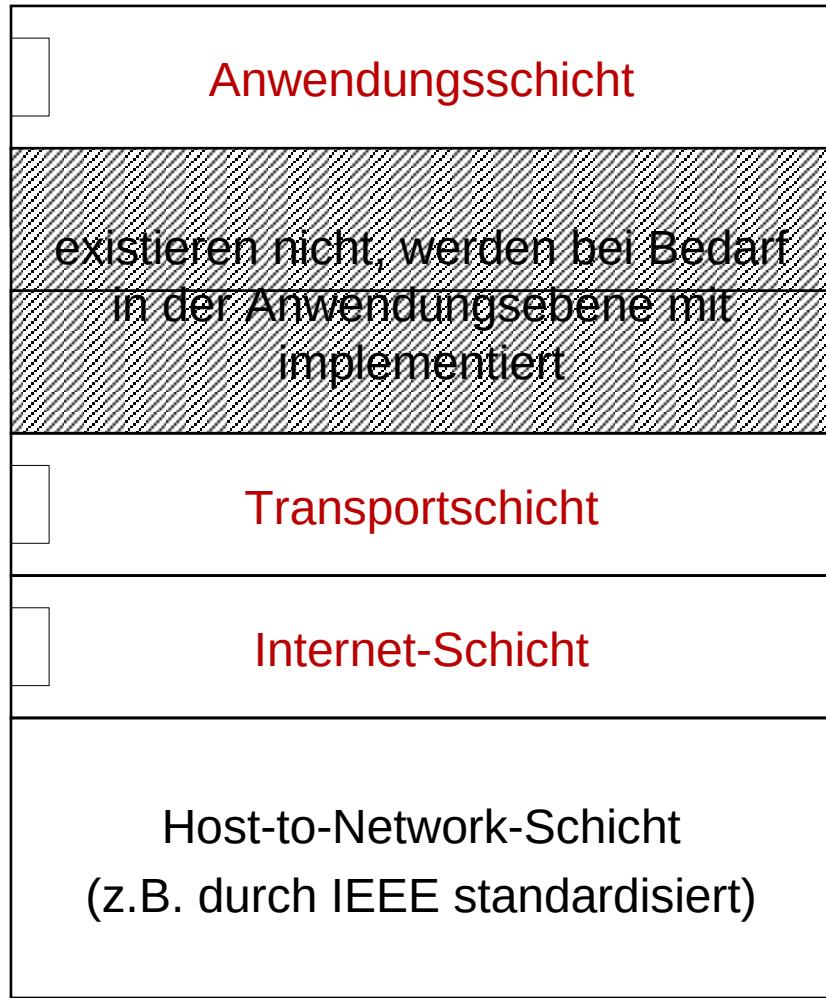
- Orientierung am OSI-Referenzmodell, aber Reduktion des Overheads:



Das Internet-Referenzmodell



ISO/OSI



TCP/IP

Aufgabe der Schichten

- **Host-to-Network-Schicht** (entspricht ISO/OSI 1-2)
 - Idee: die faktische Datenübertragung, wird von der **Netzwerkkarte** des Rechners erledigt
 - Vorteil bei Multi-Access-Netzen: Die Netzwerkkarte kann selber entscheiden, ob die Daten für den eigenen Rechner sind -> Entlastung des Betriebssystems (da kein Interrupt notwendig)
 - Starke Entkopplung vom Betriebssystem: IP teilt dem Treiber der Netzwerkkarte mit, dass diese Daten (zuverlässig) an eine gegeben Zieladresse (MAC-Adresse) übertragen soll
 - Überträgt das IP-Datagramm der Länger 1500 Byte an die MAC-Adresse
 - Direkte Anbindung des Zielrechners erforderlich
 - Beispiele
 - Ethernet in kabelgebundenen, lokalen Netzen
 - WLAN in drahtlosen lokalen Netzen
 - PPP über Punkt-zu-Punkt-Verbindungen über DSL
 - ...

IEEE Standards für die Host-to-Network Schicht

- Institute of Electrical and Electronic Engineers (IEEE)

- 802.1 Overview and Architecture of LANs
- 802.2 Logical Link Control (LLC)
- 802.3 CSMA/CD („Ethernet“)
- 802.4 Token Bus
- 802.5 Token Ring
- 802.6 DQDB (Distributed Queue Dual Bus)
- 802.7 Broadband Technical Advisory Group (BBTAG)
- 802.8 Fiber Optic Technical Advisory Group (FOTAG)
- 802.9 Integrated Services LAN (ISLAN) Interface
- 802.10 Standard for Interoperable LAN Security (SILS)
- 802.11 Wireless LAN (WLAN)
- 802.12 Demand Priority (HP's AnyLAN)
- 802.14 Cable modems
- 802.15 Personal Area Networks (Bluetooth)
- 802.16 WirelessMAN (WiMAX)
- 802.17 Resilient Packet Ring
- 802.18 Radio Regulatory Technical Advisory Group (RRTAG)
- 802.19 Coexistence Technical Advisory Group
- 802.20 Mobile Broadband Wireless Access (MBWA)
- 802.21 Media Independent Handover
- 802.22 Wireless Regional Area Networks (WRAN)



Aufgabe der Schichten

- **Internet-Schicht** (entspricht ISO/OSI 3)
 - Aufgabe dieser Schicht: Kommunikation zwischen Rechnern auch über die eigenen Netzwerkgrenzen hinaus. Die dabei gewählte Übertragungstechnik ist paketvermittelnd, d.h. *Datagramme* werden unabhängig voneinander mit *Adressinformation* versehen vom Sender losgeschickt.
 - Im Netze findet ein Store-and-Forward statt. An der Grenze zwischen zwei Netzen sorgen *Router* für die Weiterleitung der Daten. Nach dem Empfang findet die Entscheidung über den nächsten Router konzeptionell anhand der Zieladresse statt.
 - Pakete können im Netz verloren gehen oder beim Empfänger in einer anderen Reihenfolge ankommen als sie abgesendet wurden. Die Behebung solcher Situation ist der Transportebene vorbehalten
 - Festlegung eines einzigen Protokolls zur Kopplung aller Netze im Internet samt Paketformat: das *Internet Protocol (IP)*

Aufgabe der Schichten

- **Transportschicht** (entspricht ISO/OSI 4)
 - Kümmert sich um die Kommunikation zwischen Anwendungen. Definiert sind zwei unterschiedliche Ende-zu-Ende Protokolle:
 - *TCP (Transmission Control Protocol)*, ein *zuverlässiges, verbindungsorientiertes* Protokoll, welches einen Strom von Bytes sicher zwischen zwei Rechnern überträgt. Dazu wird der Strom segmentiert und in IP-Pakete verkapselt. Auf der Gegenseite werden die ankommenden Pakete in der richtigen Reihenfolge zusammengesetzt, so dass der ursprüngliche Datenstrom entsteht. TCP sorgt auch für Flusskontrolle, damit der Sender den möglicherweise langsameren Empfänger nicht überlastet.
 - *UDP (User Datagram Protocol)*, ein *unzuverlässiges* ("best effort") und *verbindungsloses* Protokoll. UDP wird benutzt, wenn nur kurze Nachrichten übermittelt werden sollen oder wenn eine *schnelle* Lieferung der Daten wichtiger ist als eine absolut zuverlässige (Sprache, Video).

- **Anwendungsschicht**

(entspricht ISO/OSI 5-7 + Anwendung)

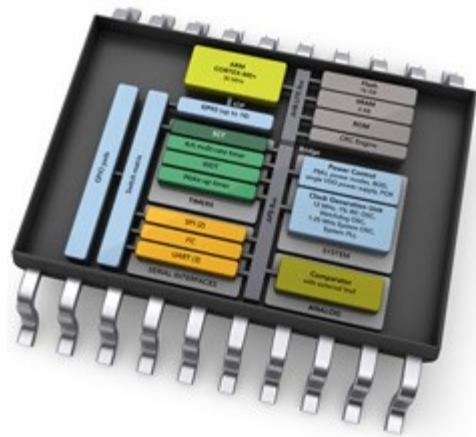
- In dieser Schicht werden wie bei ISO/OSI *Schnittstellen für Anwendungen* implementiert, oder ganze Anwendungen realisiert (Web-Server: httpd)
- Festgelegt werden aber auch *Dialoge* (Schicht 5: üblicherweise Anfrage - Antwort) und *Nachrichtenformate* (Schicht 6: Syntax & Semantik)
- SMTP (für E-Mails), DNS (Abbildung von Rechnernamen auf IP-Adressen), HTTP (Übertragung von Webseiten) usw.

- Datenkommunikation:
 - Sammlung aufeinander aufbauender Protokolle, die gemeinsam die gesamte Funktionalität eines fehlerfreien Datenaustauschs definieren
 - ISO/OSI-Referenzmodell als Versuch, solche Protokolle zu standardisieren; die Terminologie des Modells wird weiter verwendet
 - ISO/OSI-Schichten 1 und 2 werden aus Softwaresicht als *Treiber für Netzwerkkarten* implementiert, und in der Netzwerkkarte selber implementiert. Innerhalb eines lokalen Netzes können mittels dieser Funktionalität Rechner kommunizieren
 - TCP/IP-Referenzmodell baut auf den Netzwerkkartentreibern auf
 - IP und TCP/UDP sind als *Teil eines Betriebssystems* implementiert
 - Anwendungsprotokolle können als *zusätzliche Dienste* auf dem System laufen (sog. Daemon-Prozesse)

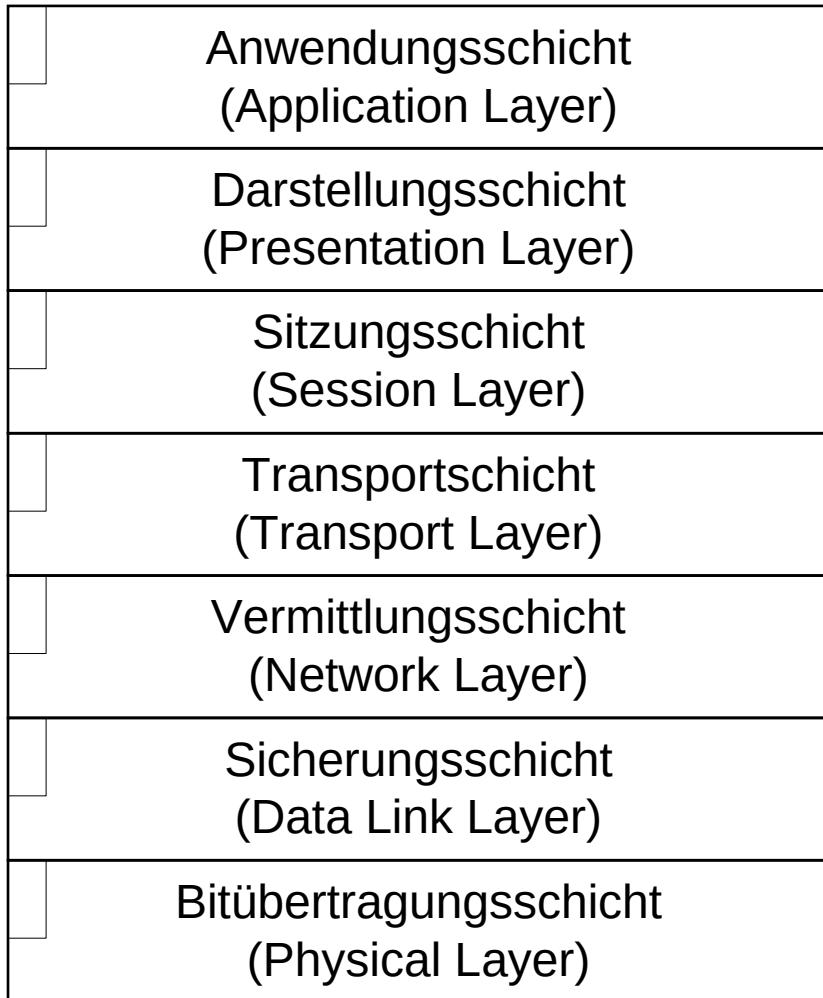
Kommunikationssysteme

(Modulcode 941306)

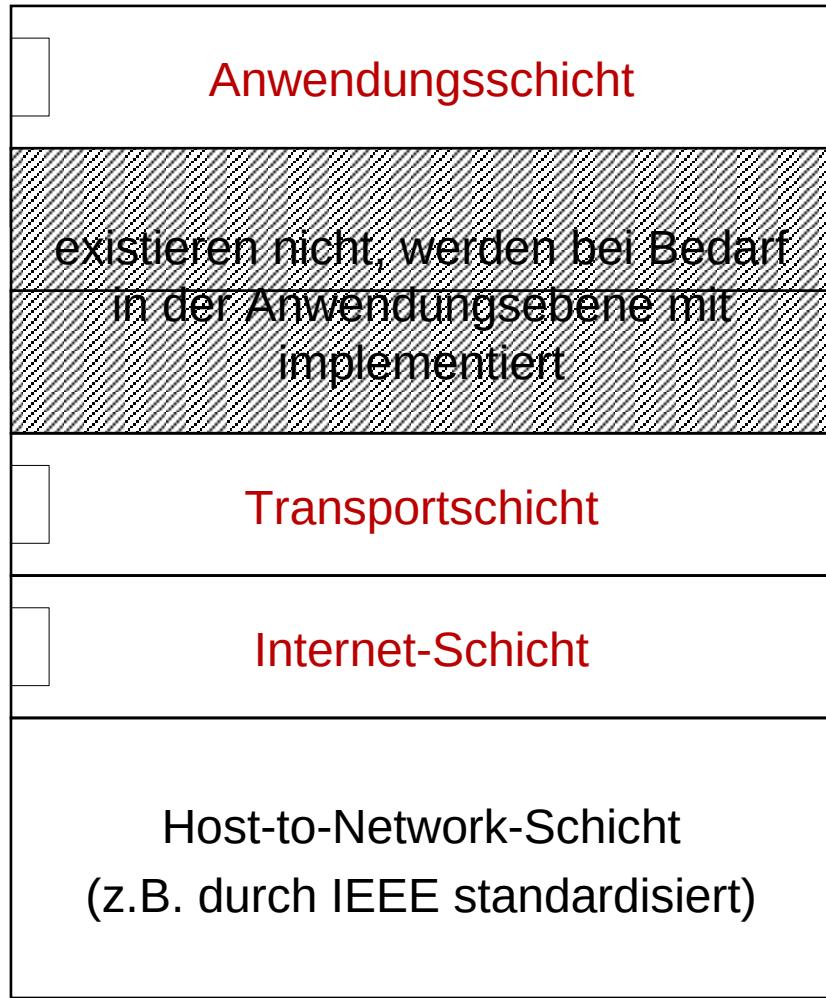
Prof. Dr. Andreas Terstegge



Das Internet-Referenzmodell

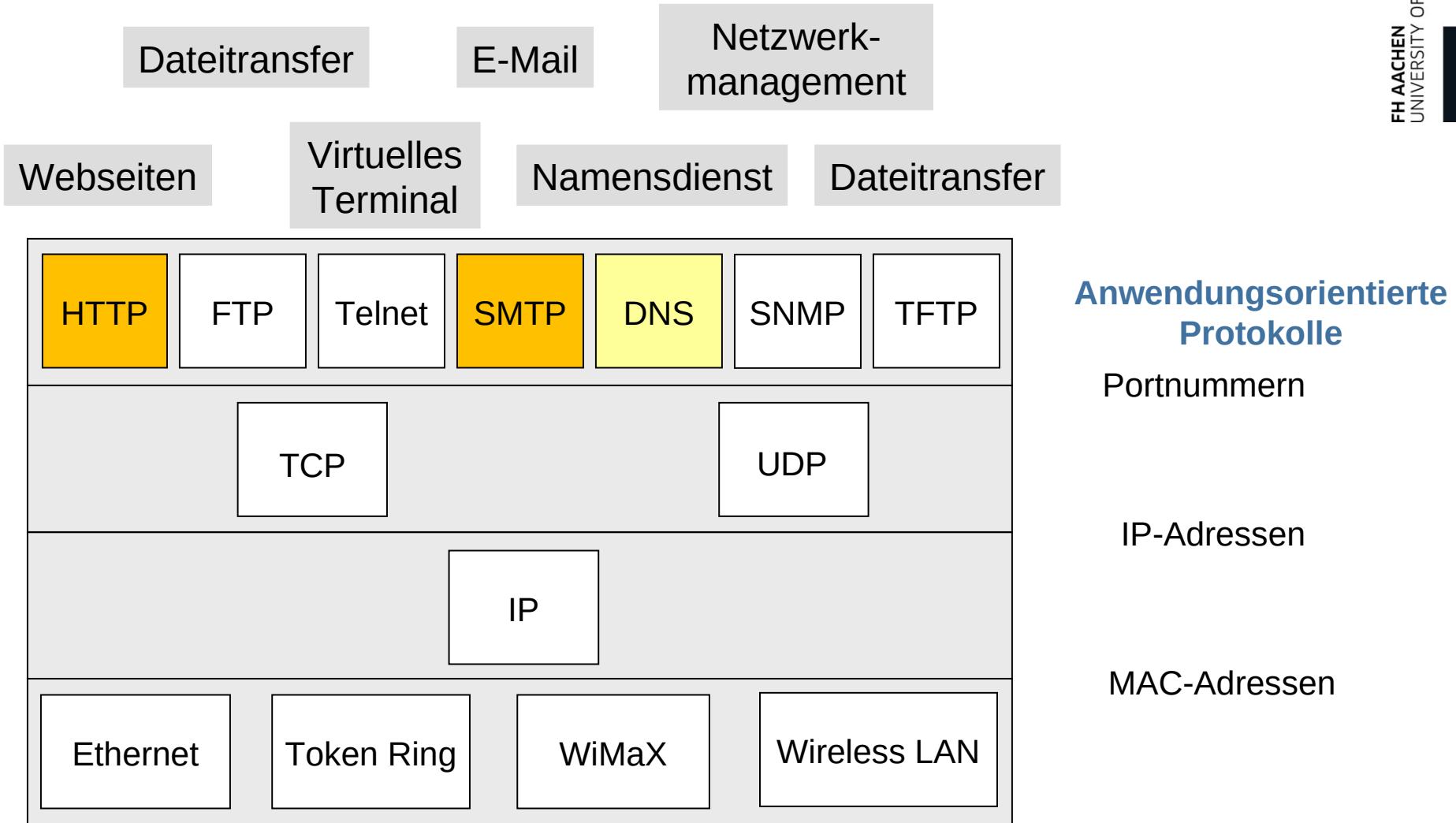


ISO/OSI



TCP/IP

Anwendungsprotokolle im TCP/IP-Referenzmodell



Netzwerkprogrammierung:

- TCP/UDP und IP sind im Betriebssystem integriert
- Wird eine eigene Anwendung entwickelt, hat man kein vorgefertigtes “Anwendungsprotokoll”, sondern muss direkt auf TCP/IP aufbauen
- Port-Nummern identifizieren die Anwendung, IP-Adressen den Zielrechner
- Zugang zu TCP/UDP/IP über **Sockets**

Hinweis: Selbstverständlich gibt es Bibliotheken, die auf Basis von TCP/IP höherwertige Kommunikationsprimitive bieten (Middleware)

Sockets:

- Abstraktion, die vom Betriebssystem in den Programmiersprachen bereitgestellt wird
- Teilt man einer Instanz dieses Typs Adressinformationen des Kommunikationspartners mit, wird ein *Kommunikationskanal* zu diesem Partner aufgebaut (TCP), bzw. vorbereitet (UDP)
- Bei **TCP** kann man einfach durch schreiben auf den Socket bzw. auslesen des Sockets Daten übertragen; In Java wird dies durch **Streams** realisiert
- Bei **UDP** muss man quasi eine Art **Briefkasten** benutzen, über den die Nachricht (mit ausgewiesenen Absender und Empfänger) übertragen wird

Aber: der Programmierer ist verantwortlich dafür, was übertragen wird!

Arbeiten mit TCP/IP: Client (schematisch)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser = new BufferedReader
            (new InputStreamReader(System.in));

        Socket clientSocket = new Socket("zielrechner", 6789);

        DataOutputStream outToServer = new DataOutputStream
            (clientSocket.getOutputStream());
    }
}
```

Erstelle Client-Socket, bau Verbindung auf

Erstelle Datenstrom für den Socket

Angabe der Adresse des Servers in zwei Teilen:
Adresse des Zielrechners (IP-Adresse/Name) und der auf diesem Rechner laufenden Anwendung, die die Daten bekommen soll (Port). Diese Adressinformationen werden mit einem sogenannten Socket verknüpft, der eine Variable zur Kommunikation per TCP/IP darstellt.

Arbeiten mit TCP/IP: Client (schematisch)

```
Erstelle Datenstrom aus dem Socket ]→ BufferedReader inFromServer = new BufferedReader  
new InputStreamReader(clientSocket.getInputStream());  
  
Sende an den Server ]→ sentence = inFromUser.readLine();  
outToServer.writeBytes(sentence + '\n'); SEND  
  
Empfange vom Server ]→ modifiedSentence = inFromServer.readLine(); RECEIVE  
System.out.println("FROM SERVER: " + modifiedSentence);  
  
clientSocket.close();  
  
}
```

Arbeiten mit TCP/IP: Server (schematisch)

```
import java.io.*;
import java.net.*;

class TCPServer {

    public static void main(String argv[]) throws Exception
    {
        String clientSentence;
        String capitalizedSentence;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while(true) {
            Socket connectionSocket = welcomeSocket.accept();

            BufferedReader inFromClient =
                new BufferedReader(new
                    InputStreamReader(connectionSocket.getInputStream()));

            // Verarbeitung der Daten vom Client
            capitalizedSentence = inFromClient.readLine();
            inFromClient.close();
        }
    }
}
```

Wie beim Client: erstelle einen Socket als Variable zur Kommunikation. Aber: lege direkt fest, über welchen Port diese Anwendung Daten empfangen soll.
Warte dann so lange, bis ein Client an diesen Port Daten schickt.

Erstelle Socket → `ServerSocket welcomeSocket = new ServerSocket(6789);`

Warte auf eingehende Verbindungs-wünsche → `while(true) {`
`Socket connectionSocket = welcomeSocket.accept();`

Verknüpfe Buffer mit dem Socket → `BufferedReader inFromClient =`
`new BufferedReader(new`
`InputStreamReader(connectionSocket.getInputStream()));`

Arbeiten mit TCP/IP: Server (schematisch)

```
Verknüpfe  
ausgehenden  
Datenstrom  
mit dem  
Socket } → DataOutputStream outToClient =  
          new DataOutputStream(connectionSocket.getOutputStream());  
  
Lesen vom  
Socket } → clientSentence = inFromClient.readLine(); RECEIVE  
  
Schreiben  
auf den  
Socket } → capitalizedSentence = clientSentence.toUpperCase() + '\n';  
          outToClient.writeBytes(capitalizedSentence); SEND  
      } } ← Gehe zurück, warte auf  
           nächste Anfrage
```

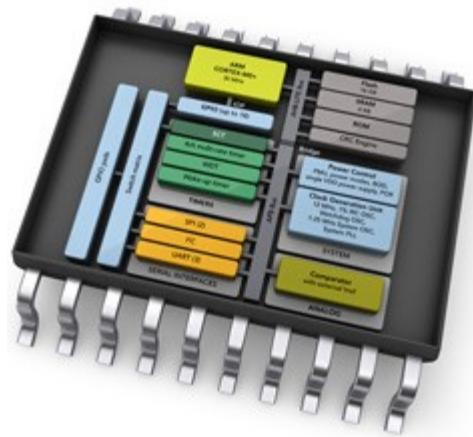
Der eben implementierte Server hat ein Problem!
Welches?

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

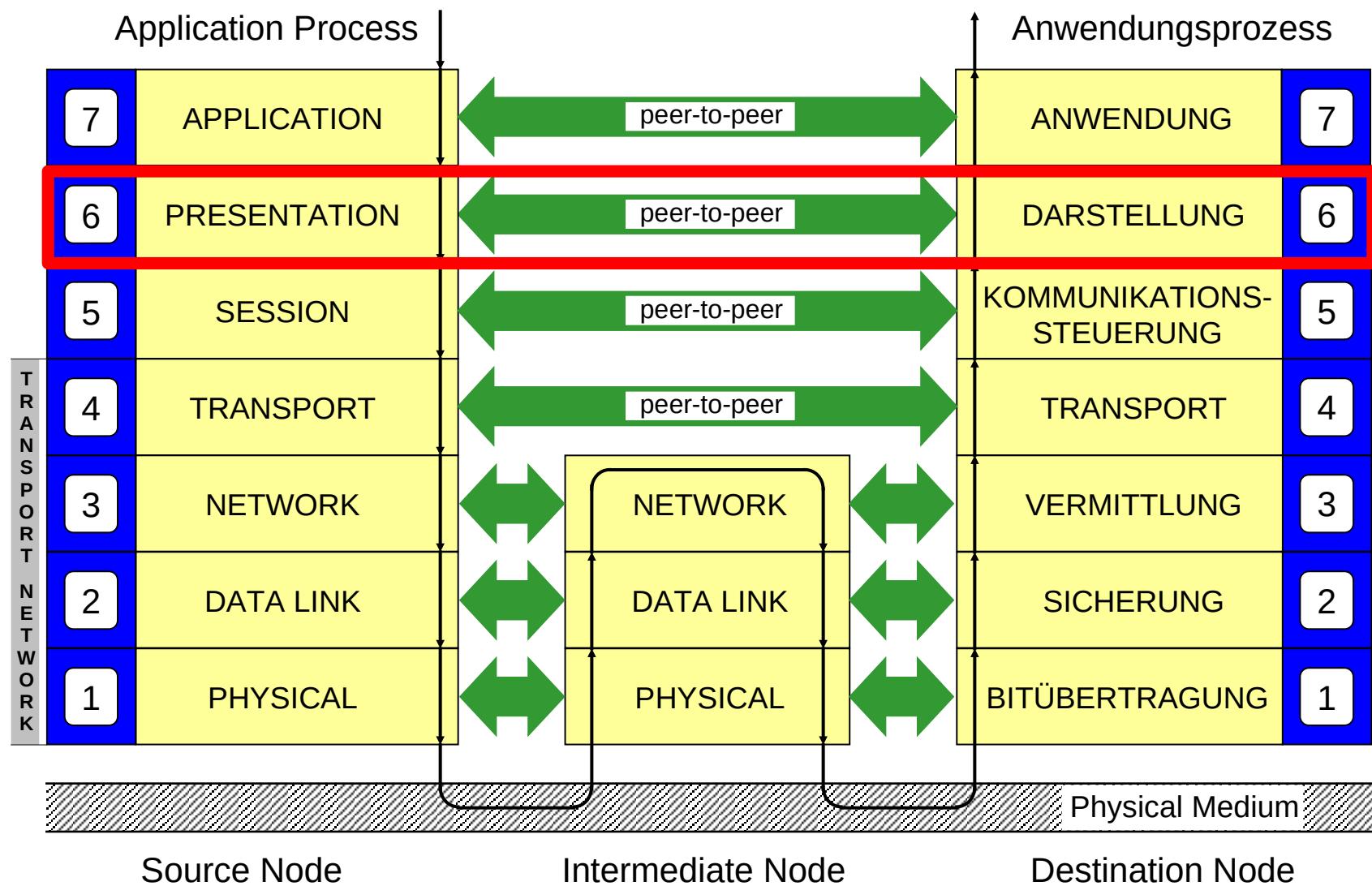
Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Das ISO-OSI-Referenzmodell für offene Systeme



Heterogenität im verteilten System:

- Heterogenität der Plattform (Betriebssystem, Hardware)
- Heterogenität der Programmiersprache/APIs

Heterogene Hardware-Architekturen

- Unterschiedliche Reihenfolge der Speicherung von Bytes:
Little Endian / Big Endian
- Unterschiedliche Zeichencodierung:
 - ASCII auf PCs / EBCDIC auf IBM Mainframes

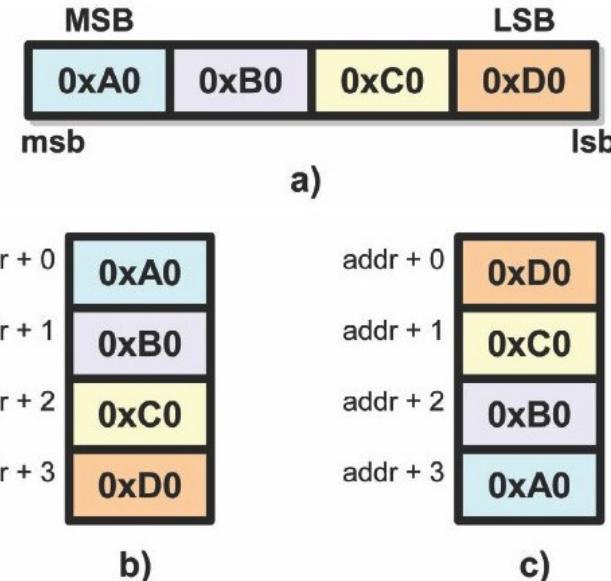
Problem: Uneinheitliche interne Darstellung

Z.B. Unterschiedliche Darstellungen von Integer-Werten im Arbeitsspeicher: Little-Endian / Big-Endian

- a) Ausgangswert
- b) Big-Endian Speicherung
- c) Little-Endian Speicherung

→ Probleme bei Interpretation der Daten nach Versenden ja nach Rechnerarchitektur

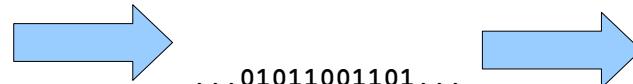
Zeichenketten sind von dieser Problematik nicht betroffen!



Problem: Übertragung von Datenstrukturen

```
struct Student {  
    String Studiengang;  
    int Alter;  
    bool imAuslandssemester;  
}
```

```
Student s;  
s.Studiengang = „AMI“;  
s.Alter = 20;  
...  
send( s );
```



```
Student s = receive();  
if (Alter > 25) ...
```

- Häufig besteht der Bedarf, nicht nur einzelne Werte/Strings zu übertragen, sondern ganze Datenstrukturen
- Daher müssen 2 Probleme gelöst werden:
 - Übertragung der grundsätzlichen Struktur der Daten (**abstrakte Syntax**), *Serialisierung - Deserialisierung*
 - Übertragung des konkreten Datenstroms mit einer vereinbarten Kodierung (**Transfersyntax**), *Marshalling - Unmarshalling*

Datendarstellung sollte eigenständig sein

Idee:

- Definiere eine **Menge abstrakter Datentypen** und eine **Kodierung** (ein genaues Bit-Format) für jeden dieser Typen
- Stelle **Werkzeuge** zur Verfügung, die die abstrakten **Datentypen in** Datentypen der verwendeten **Programmiersprache** übersetzen
- Stelle **Werkzeuge** zur Verfügung, die die Datentypen der verwendeten **Programmiersprache in** die abstrakten **Datentypen** und damit in das kodierte Format übersetzen
- *Senden (Marshalling)*: wenn ein bestimmter Datentyp übertragen werden soll, rufe die Kodierfunktion auf und übertrage das Ergebnis
- *Empfangen (Un-Marshalling)*: dekodierte den Bit-String und erzeuge eine neue lokale Repräsentation des empfangenen Typs

Beispiele verschiedener Datendarstellungs-Standards

ISO: - **ASN.1** (Abstract Syntax Notation)

Sun ONC (Open Network Computing)-RPC:

- **XDR** (eXternal Data Representation)

OSF (*Open System Foundation*)-RPC:

- **IDL** (Interface Definition Language)

Corba: - **IDL** und **CDR** (*Common Data Representation*):

- **CDR** bildet IDL-Datentypen in Bytefolgen ab.

W3C: - **XML/SOAP**

- Darstellung aller Datentypen als (Maschinen-)lesbarer Text.
Zu klären: Zeichenkodierung

Java: - **Objektserialisierung**, d.h. Abflachung eines (oder mehrerer) Objektes zu einem seriellen Format inkl. Informationen über die Klassen. Deserialisierung ist die Wiederherstellung eines Objektes ohne Vorwissen über die Typen der Objekte

JavaScript - **JSON** (JavaScript Object Notation)

ASN.1: Definition

ASN.1 (abstrakte Syntax-Notation eins) ist eine von der ISO genormte Beschreibungssprache zu darstellungsunabhängigen Spezifikation von Datentypen und Werten

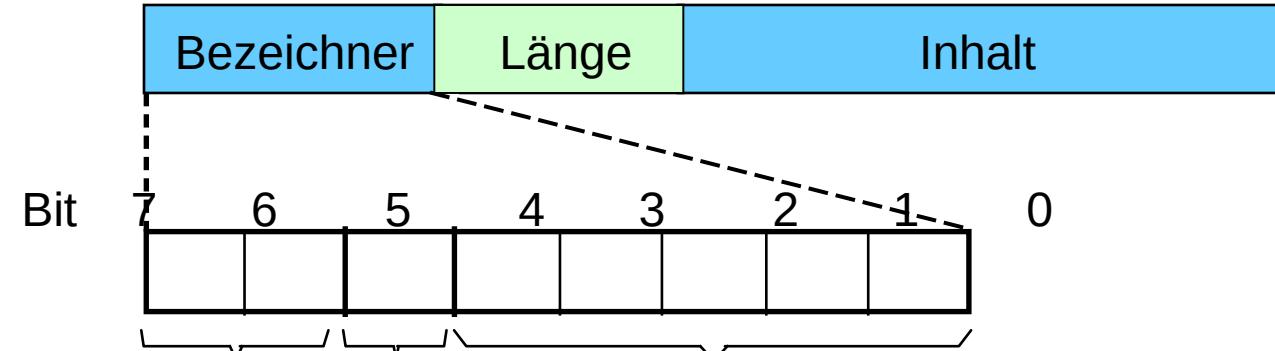
- findet z.B. zur Definition von Managementobjekten bei SNMP Verwendung
- **Elementare Datentypen:**
 - Boolean, Integer, Bitstring, Octetstring, IA5String, ...
- **Strukturierte Datentypen:**
 - Sequence: Geordnete Liste von Datentypen (struct in C)
 - Set: Ungeordnete Menge von Datentypen
 - Sequence OF: Geordnete Liste von Elementen des gleichen Datentyps (Array in C)
 - Set OF: Ungeordnete Menge des gleichen Datentyps
 - Choice: Ungeordnete Menge von Datentypen, aus der einige Datentypen ausgewählt werden können (Union in C)

Beispiel: **Mitarbeiter ::= Set { Name IA5String,
Alter Integer,
Personalnr Integer }**

ASN1: Übertragungssyntax

http://en.wikipedia.org/wiki/Distinguished_Encoding_Rules#BER_encoding

Zusätzlich zur Bereitstellung einer Datenbeschreibungssprache bietet ASN.1 auch sogenannte **Basic Encoding Rules**, also Regeln, die spezifizieren, wie ASN.1-Objekte über das Netzwerk versendet werden sollten.

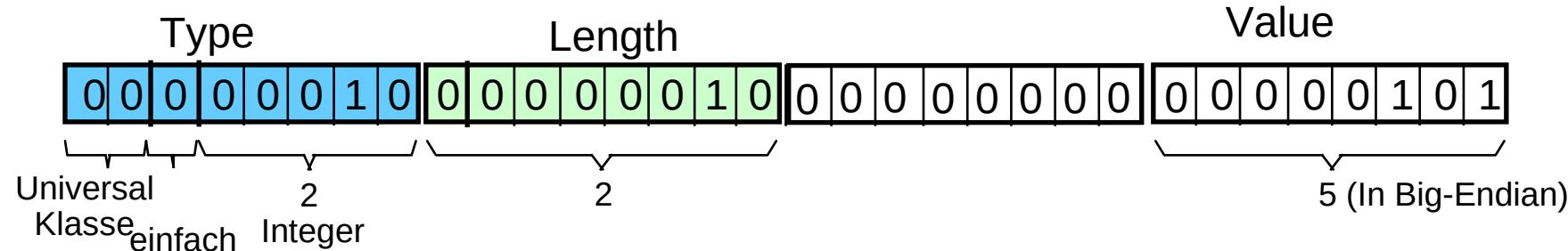


Typklasse
00: Universal
01: Application
10: Context Specific
11: Private

Datentyp
0: einfach
1: strukturiert

Tag-Nummer
0..30
31: nächstes Byte gibt Tag an

Beispiel: Integer Wert 5



Objektserialisierung in Java

- Die Klasse **ObjectInputStream** bietet die Möglichkeit, mit der Methode **readObject()** serialisierte Objekte aus dem darunterliegenden Stream zu lesen
- Serialisierung ist der synonym verwendete Begriff für Objekt-Streams, d.h. das Lesen und Schreiben von Objekten. Serialisierung ist nicht auf singuläre Objekte beschränkt, da diese ja wieder andere Objekte referenzieren können.
 - **writeObject()** wandelt ein Objekt in einen Byte-Stream um
 - **readObject()** wandelt einen Byte-Stream in ein Objekt um
- Im Unterschied zum allgemeinen I/O- bzw. Streaming-Konzept beruht die Objekt-Kommunikation auf Interfaces. Somit ist auch eine komplett eigencodierte Serialisierung möglich

XML: eXtensible Markup Language

HTML wurde zur Darstellung von Information für Menschen entwickelt, und nicht für die Informationsverarbeitung in Programmen

- Die Auszeichnungselemente sind **fest vorgegeben** und sind daher nicht zur Beschreibung beliebiger Daten geeignet
- Problematische **Vermengung** von Auszeichnung und Präsentationssemantik

Dies motivierte die Entwicklung einer offenen, textbasierten Auszeichnungssprache: **eXtensible Markup Language**

XML: eXtensible Markup Language

- XML ist eine Metasprache zur Strukturierung von Dokumenten und Daten
- Basis von XML sind beliebige Auszeichnungselemente, die geringen syntaktischen Regeln zu folgen haben
 - Wohlgeformte XML-Dokumente
- XML macht erst wirklich Sinn, wenn es in einer Anwendung konkretisiert wird. XML-Anwendungen ergeben sich aus
 - **Namensräumen**
Festlegen der semantisch logischen Bedeutung bei Homonymen in den Auszeichnungselementen
 - **Stylesheets**
Definition des Erscheinungsbildes von Auszeichnungselementen
 - **Schemata**
Beschreiben die Dokumentenstruktur eines bestimmten Typs

Beispiel: XSD

XML-Dokument: Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<rechnung kundennummer="k333063143">
    <monatspreis>0,00</monatspreis>
    <einzelverbindungsNachweis>
        <verbindung>
            <datum>26.2.</datum>
            <zeit>19:47</zeit>
            <nummer>200xxxx</nummer>
            <einzelpreis waehrung="Euro">0,66</einzelpreis>
        </verbindung>
        <verbindung>
            <datum>27.2.</datum>
            <zeit>19:06</zeit>
            <nummer>200xxxx</nummer>
            <einzelpreis waehrung="Euro">0.46</einzelpreis>
        </verbindung>
        <verbindungskosten_gesamt
            waehrung="Euro">2.19</verbindungskosten_gesamt>
    </einzelverbindungsNachweis>
</rechnung>
```

XML-Dokument: Eigenschaften

- XML-Dokumente enthalten **Daten und Strukturinformation** über die Daten in einem Dokument (selbstbeschreibend)
- XML-Dokumente haben **Strukturvorgaben** (Wohlgeformtheit)
- Informationen im XML-Dokument haben einen **Datentyp** (typisiert)

Elemente eines XML-Dokuments

- XML-Dokumente: Zeichendaten und Auszeichnungen
 - XML- und Dokumenttyp-Deklarationen

```
<?xml version="1.0" encoding="UTF-8"?>
```
 - Elemente mit möglichem Inhalt

```
<name>Volker Sander</name>
```
 - Attributen in Elementen

```
<name attribute="Wert">
```
 - Entity-Referenzen (< statt <)
 - Kommentare

```
<!-- Das ist ein Kommentar -->
```
 - Processing Instructions – werden an die aufrufende Instanz weitergeleitet

```
<?name pidata?>
```

Elemente eines XML-Dokuments

- Jedes Dokument entspricht einer Baumstruktur (DOM – Document Object Model)
- Die Baumknoten werden Elemente genannt
- Es kann nur ein Element an der Wurzel geben
 - Eine Applikation weiß sofort wann das Dokument zu Ende ist
- Syntaxregeln müssen strikt eingehalten werden
 - Jedes Starttag muss ein Endtag haben

```
<BOOK>...</BOOK>
<BOOK />
```
 - Verschachtelung nur mit vollständigen Tags möglich:

```
<BOOK> <LINE> This is OK </LINE> </BOOK>
<LINE> <BOOK> This is </LINE> definitely NOT </BOOK> OK
```
 - Attributwerte müssen mit ' oder " abgegrenzt werden

Wann ist ein XML-Dokument „korrekt“?

Wohlgeformtheit

Dokument entspricht den syntaktischen Regeln

- Genau ein Dokument-Element
- Jedes öffnende Tag hat ein schließendes Tag.
- Die Verschachtelung ist balanciert.

Gegenbeispiel:

```
<A><B><C> ... </A> ... </C>
```

Validität

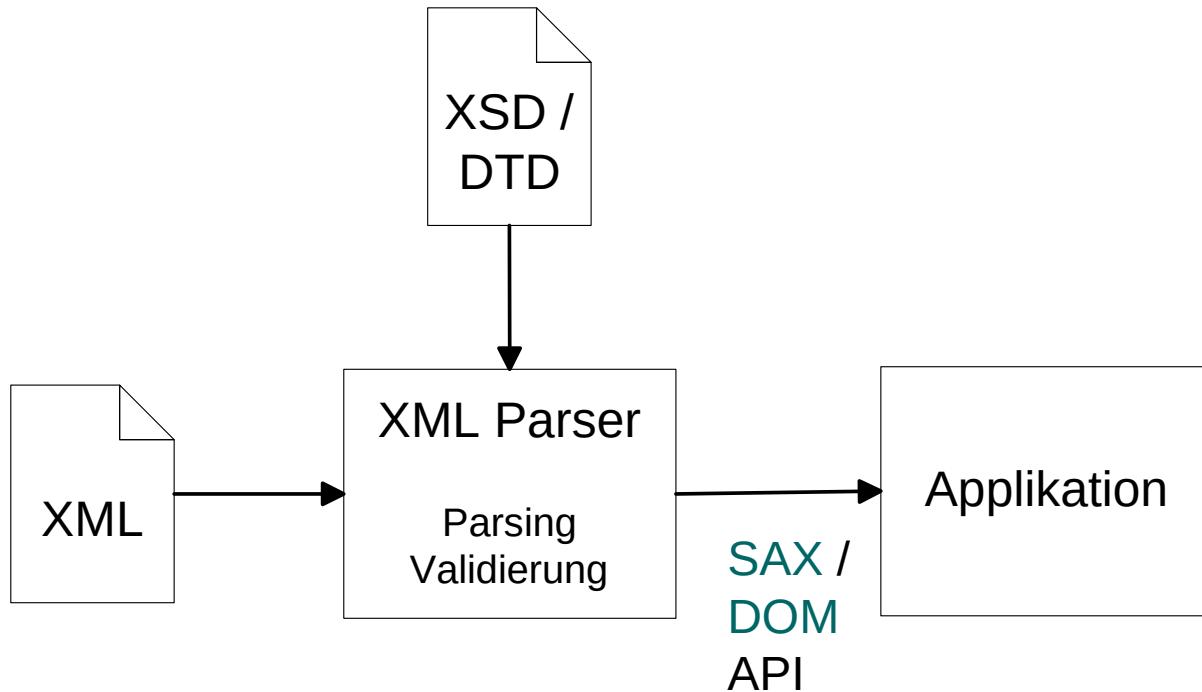
Dokument ist

- wohlgeformt
- konform zu einer fest vorgegebenen Dokumentenstruktur ist

Document Type Description (DTD)

XML Schema Definition (XSD)

XML Parsing: Übersicht



XML Namensräume: Namespaces

Zur Erklärung eines XML-Schemas müssen wir erst ein anderes Problem diskutieren:

- Zum Test eines Dokumentes gegen eine Strukturvorgabe müssen Elementbezeichnungen verarbeitet werden.
- Gleiche Elementbezeichnungen können jedoch unterschiedliche Bedeutung haben:

`<Schloss>Neuschwanstein</Schloss>`

`<Schloss>Sicherheitsschloss Typ X</Schloss>`

- Ähnlich wie Variablen in Programmiersprachen haben Elementbezeichnungen einen Namensraum
- Namespaces dienen der Trennung der Bedeutung der Bezeichner und ermöglichen so, Elementbezeichnungen und Inhalte eindeutig zu referenzieren

XML Namensräume: Namensraumdeklaration

- Namensräume werden weltweit eindeutig durch eine **URI** (Uniform Resource Identifier) identifiziert
- Diese wird über einen Prefix zugänglich gemacht
- Namensraum-Prefix muss ein XML-Name sein
 - darf keine Sonderzeichen usw. enthalten
 - URI's beinhalten aber oftmals Sonderzeichen (<http://test.com/namespace>)
- Namensraumdeklaration ordnet Prefix und URI einander zu:

```
<person xmlns:job="http://www.berufe-online.de/berufe">
    <vorname>Carl Friedrich</vorname>
    <nachname>Gauß</nachname>
    <job:berufsbezeichnung>Mathematiker</job:berufsbezeichnung>
</person>
```

- Ein XML-Schema ist eine XML-Anwendung, mit der die Struktur einer Klasse von XML-Dokumenten beschrieben werden kann
- Der Zweck eines XML-Schemas entspricht dem seines Vorgängers, dem DTD (welches nicht XML war)
- Ein XML Schema enthält in XML notierte Regeln, die erlaubte Elementbezeichner, Reihenfolgen, Inhalte und Attribute mit Wertebereichen aufzählen
- Hierzu existieren vielfältige vordefinierte Datentypen, die Möglichkeit zur Definition eigener Datentypen, und die Möglichkeit zur Darstellung komplexer Integritätsbedingungen

- Um ein Dokument validieren zu können muss es das seine Struktur beschreibende Schema referenzieren (hierzu ist es nicht verpflichtet):

```
<?xml version="1.0" ?>  
  <artikel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:noNamespaceSchemaLocation="artikel_simple.xsd">
```

- Das Schema selber ist immer einem Schema-Namespace zugeordnet

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
  <xS:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

- Somit können in einem Dokument auch Elemente verschiedener Schemata verwendet werden!

XML Schema: Beispiel

Beispiel: <**datum jahr**=“1979”/>

Namensraumdeklaration für
XML-Schema

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">  
  
<xsd:element name = “datum” type= “datumsTyp”/>  
  
<xsd:complexType name= “datumsTyp”>  
  <xsd:attribute name = “jahr”>  
    <xsd:simpleType>  
      <xsd:restriction base = “xsd:integer”>  
        <xsd:maxInclusive value = “2500”>  
        <xsd:minInclusive value = “0”>  
      </xsd:restriction>  
    </xsd:simpleType>  
  </xsd:attribute>  
</xsd:complexType>
```

Attribut-
Deklaration

Element-
Deklaration

Typdefinitionen

XML Schema: Elementdeklarationen

Neben der Benennung eines Tags für ein Element kann auch der Datentyp angegeben werden (per Referenz oder direkter Auflistung)

- Primitive Datentypen

Schema: `<element name="float-element" type="float" />`

Dokument: `<float-element>123.456 </float-element>`

- Deklaration von Datentypen (Elementen)

`<complexType name="studenttype"> ... </complexType>`

- Erweiterung existierender Datentypen mittels Angabe eines Musters (Mit Großbuchstaben beginnend, dann ein oder mehrere Zahlen)

`<simpleType name="studentid" base="string">
 <pattern value="[A-Z][0-9]+">
</simpleType>`

XML Schema: Einfache Datentypen

- Dienen zur Definition der Wertebereiche von Elementen und Attributen

Beispiele für Elementdeklarationen mit einfachen Datentypen:

```
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="schuhgroesse"
              type="xsd:positiveInteger"/>
<xsd:element name="geburtsdatum" type="xsd:date"/>
```

Verwendung des Schemata in einer XML-Datei:

```
<name>Carl Friedrich Gauß</name>
<schuhgroesse>42<schuhgroesse/>
<geburtsdatum>1777-04-30<geburtsdatum/>
```

XML Schema: Komplexe Typen

- Neben diesen eingebauten Datentypen bietet XML-Schema die Möglichkeit, eigene Datentypen abzuleiten
- Komplexe Typen benötigt man immer dann, wenn man Elemente deklarieren möchte, die Attribute besitzen oder andere Elemente beinhalten können, denn folgendes ist nicht möglich:

```
<xs:element name="Kunde">
  <xs:element name="Vorname" type="xs:string">
  <xs:element name="Zweitname" type="xs:string">
  <xs:element name="Nachname" type="xs:string">
</xs:element>
```

XML Schema: Komplexe Typen

- Erfolgt die Definition eines komplexen Typs für ein Element innerhalb des <xs:element>-Elements, so spricht man von einem **anonymen Typen**

```
<xs:element name="Kunde">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Vorname" type="xs:string"/>
      <xs:element name="Zweitname" type="xs:string"/>
      <xs:element name="Nachname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML Schema: Komplexe Typen

- Sollen mehrere Elemente die gleichen Kindelemente und Attribute tragen, so sollten **benannte komplexe Typen** benutzt werden
- Diese werden über das type-Attribut in der Element-Deklaration zugewiesen

```
<xs:complexType name="KundenTyp">
    <xs:sequence>
        <xs:element name="Vorname" type="xs:string"/>
        <xs:element name="Zweitname" type="xs:string"/>
        <xs:element name="Nachname" type="xs:string"/>
    </xs:sequence>
</xs:complexType>

<xs:element name="Kunde" type="KundenTyp">
```

XML Schema: Werkzeuge zur Bildung komplexer Typen

- Kompositoren können mehrere Element-Deklaration zu einer unbenannten Modellgruppe
- Es gibt drei Kompositoren, die kombiniert werden können:

XML Schema: sequence

Sequenzen:

Die Inhalte einer Sequenz müssen in der Reihenfolge erscheinen, in der sie angegeben werden:

```
<xsd:sequence>  
    [Inhalt 1]  
    ...  
    [Inhalt n]  
</xsd:sequence>
```

Die **all**-Gruppe:

- Alle Elemente, die in der **all**-Gruppe deklariert sind, dürfen in beliebiger Reihenfolge instanziert werden, aber höchstens einmal.
- Außerhalb der **all**-Gruppe darf es keine weiteren Element-Deklarationen geben.

```
<xsd:all>  
    [Elementdeklarationen]  
</xsd:all>
```

XML - Schema (choice)

Auswahllisten:

Von den aufgeführten Inhalten muss ein frei wählbarer instanziert werden:

```
<xsd:choice>
```

```
  [Inhalt a]
```

```
  . . .
```

```
  [Inhalt x]
```

```
</xsd:choice>
```

Ergänzungen zu den Kompositoren

Auswahllisten und Sequenzen können beliebig geschachtelt werden.

Neben der Anordnung kann auch die Häufigkeit festgelegt werden, mit der Elemente erscheinen dürfen.

minOccurs gibt an, wie oft ein Element mindestens erscheinen muss (default 1; Angabe optional).

maxOccurs gibt an, wie oft ein Element höchstens auftreten darf (default 1; ∞ = unbounded; Angabe optional).

default gibt den Wert an, der dem Element zugeordnet wird, wenn es vorhanden, aber leer ist.

```
<element name="vorstandsmitglied" minOccurs=4 maxOccurs=6>
```

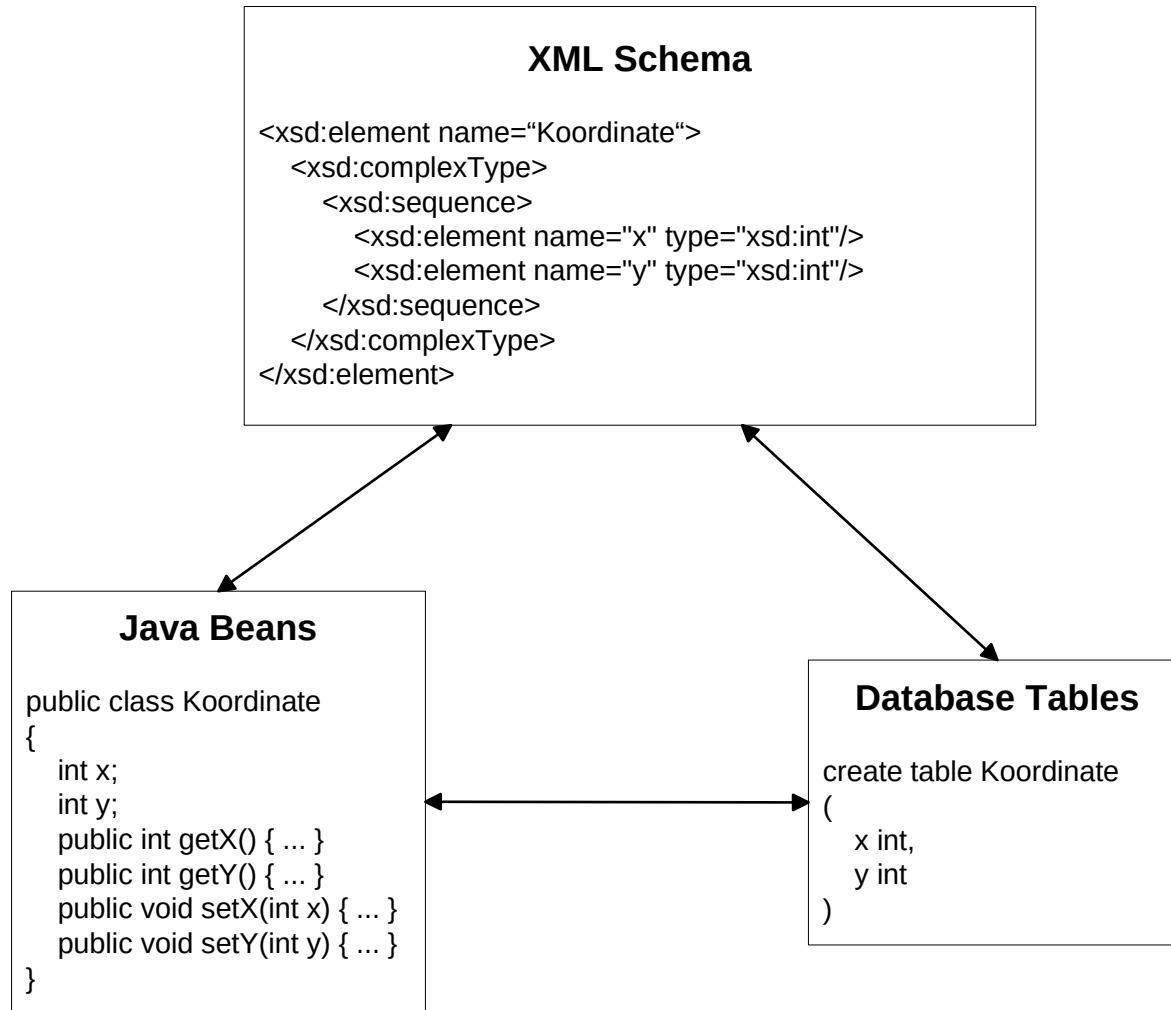
Abgeleitete Typen

- Ein abgeleiteter Datentyp wird auf der Basis eines anderen Datentyps durch Einschränkung oder Erweiterung spezifiziert.
- Auch von abgeleiteten Datentypen können weitere Datentypen abgeleitet werden.
- Datentypen können in verschiedener Weise abgeleitet werden:
 - durch Einschränkung des Wertebereichs (**restriction**)
 - über eine durch Whitespaces notierte Liste verschiedener Werte (**list**)
 - über eine Vereinigung mehrerer Typen (**union**)
 - durch Erweiterung (nur komplexe Typen) (**extension**)

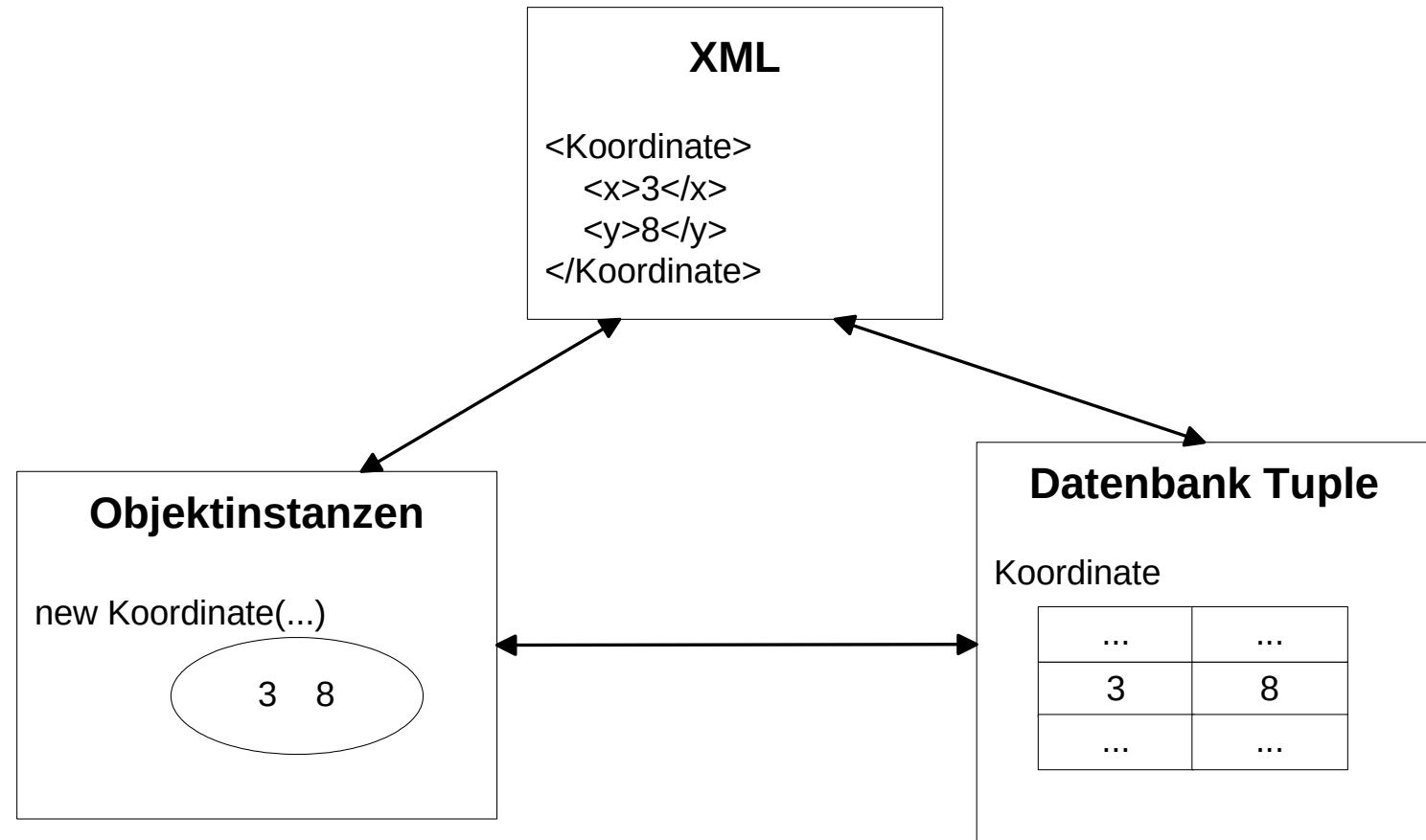
Objekt Mappings

- XML-Schema beschreiben eine Klasse von Dokumenten
- XML-Dokumente sind somit eine Instanz einer Klasse von Dokumenten
- Dies ermöglicht parallelen zu objektorientierten Welt:
 - Der Zustand eines Objektes kann in einem XML-Dokument festgehalten werden
 - Die Struktur der Klasse liefert das Schema
- Die Grenzen zwischen XML, Strukturen in Programmiersprachen und Datenbanken verschwimmen
- Einige Standards nehmen sich dieses Themas an
 - SQL 2003
 - Sun JDO
- Viele Tools sind bereits verfügbar
 - LINQ2XSD .NET
 - Hibernate

Mapping auf der Schema-Ebene



Mapping auf der Instanz-Ebene



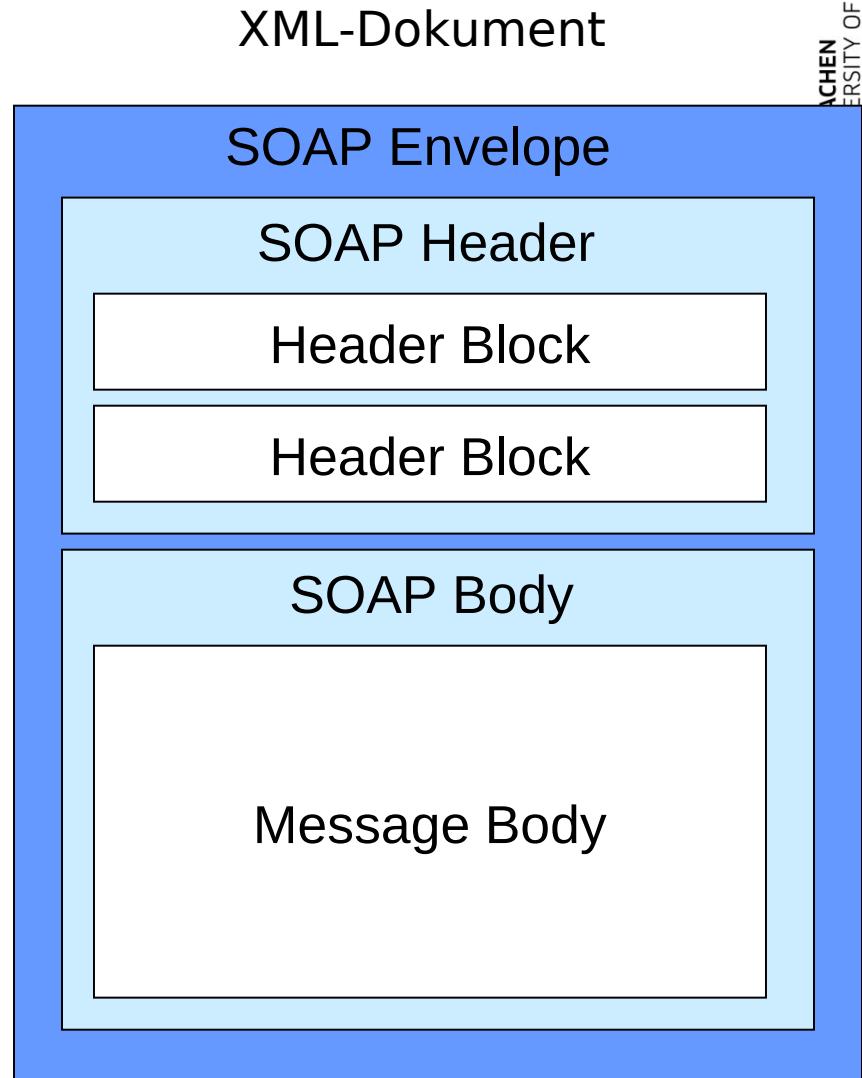
Remote Procedure Calls mittels XML: SOAP

- SOAP (ursprünglich für Simple Object Access Protocol) ist ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können.
- SOAP ist ein industrieller Standard des World Wide Web Consortiums (W3C)
- SOAP ist ein in XML definiertes, anwendungsunabhängiges Nachrichtenformat
- SOAP definiert auch, wie Anwendungsdaten kodiert werden sollen

- SOAP ist genauso wie XHTML eine Anwendung der XML-Spezifikation
- SOAP legt durch Einschränkung der erlaubten Tags und Attribute fest, welche Sprachmittel erlaubt sind
- XML-Schema und XML-Namespaces sind wichtige Komponenten bei der Beschreibung von SOAP
- SOAP regelt, wie Daten in der Nachricht abzubilden und zu interpretieren sind
- SOPA nutzt hierbei existierende Transportprotokolle, ohne diese jedoch festzulegen
- SOAP stellt so eine Konvention für entfernte Prozedur-/Methodenaufrufe dar

SOAP Nachrichten

- Eine SOAP-Nachricht besteht zunächst auf der obersten Ebene aus einem Envelope
- Der Envelope enthält einen optionalen Header sowie die eigentliche Nachricht im SOAP Body
- Aufgabe des Headers: Infos über Transaktionskontakte, Authentifizierung, Autorisierung, Routing- und Auslieferungsinformationen



Request: setHelloMessage

TCPMonitor

Admin Port 9090

Stop Listen Port: 9090 Host: localhost Port: 8080 Proxy

State	Time	Request Host	Target Host	Request...
---	Most Recent	---	---	---
Done	06/24/02 02:18:36 PM	localhost.localdomain	localhost	POST /axis/services/Hello HTTP/1.0
Done	06/24/02 02:18:36 PM	localhost.localdomain	localhost	POST /axis/services/Hello HTTP/1.0

Remove Selected Remove All

Request

```
POST /axis/services/Hello HTTP/1.0
Content-Length: 489
Host: localhost
Content-Type: text/xml; charset=utf-8
SOAPAction: ""

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Body>
<setHelloMessage>
<arg0 xsi:type="xsd:string">ciao, mundi </arg0>
</setHelloMessage>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 368
Date: Mon, 24 Jun 2002 13:18:36 GMT
Server: Apache Tomcat/4.0.4 (HTTP/1.1 Connector)
Set-Cookie: JSESSIONID=8F57C539BD75B6A07BBBEDE307F9DD31;Path=/axis

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Body>
<setHelloMessageResponse SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/envelope">
</setHelloMessageResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

TCPMonitor

Admin Port 9090

Stop Listen Port: 9090 Host: localhost Port: 8080 Proxy

State	Time	Request Host	Target Host	Request...
---	Most Recent	---	---	---
Done	06/24/02 02:18:36 PM	localhost.localdomain	localhost	POST /axis/services/Hello HTTP/1.0
Done	06/24/02 02:18:36 PM	localhost.localdomain	localhost	POST /axis/services/Hello HTTP/1.0

Remove Selected Remove All

Request: getHelloMessage

TCPMonitor

Admin Port 9090

Stop Listen Port: 9090 Host: localhost Port: 8080 Proxy

State	Time	Request Host	Target Host	Request...
---	Most Recent	---	---	---
Done	06/24/02 02:18:36 PM	localhost.localdomain	localhost	POST /axis/services/Hello HTTP/1.0
Done	06/24/02 02:18:36 PM	localhost.localdomain	localhost	POST /axis/services/Hello HTTP/1.0

Remove Selected Remove All

Request

```
POST /axis/services/Hello HTTP/1.0
Content-Length: 419
Host: localhost
Content-Type: text/xml; charset=utf-8
Cookie: JSESSIONID=8F57C539BD75B6A07BBBEDE307F9DD31
SOAPAction: ""

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Body>
<getHelloMessage/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 480
Date: Mon, 24 Jun 2002 13:18:36 GMT
Server: Apache Tomcat/4.0.4 (HTTP/1.1 Connector)

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Body>
<getHelloMessageResponse SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/envelope">
<getHelloMessageReturn xsi:type="xsd:string">ciao, mundi </getHelloMessageReturn>
</getHelloMessageResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XML Format Save Resend Switch Layout Close

JSON: JavaScript Object Notation

- JSON-Dokumente sind gültiges JavaScript
- Einfache Struktur
- Typisierung eingebaut!
- Beispiel:

```
{  
    "id" : 2648,  
    "Name" : "Mustermann",  
    "Vorname" : "Max",  
    "adr" : {  
        "Stadt" : "Aachen",  
        "plz" : 52064  
    },  
    "tel" : [ "0241 1234", "0160123456" ],  
    "partner" : null,  
    "maennlich" : true  
}
```

Weitere Informationen: <http://www.json.org/>

Typen:

- Number (wie in JavaScript)
- String (nur in doppelten Anführungszeichen)
- Boolean (true/false)
- Array (nur in eckigen Klammern)
- Object (in geschweiften Klammern)
- Null

Deckt nicht alle möglichen JavaScript-Werte ab

- NaN, Infinity werden zu null serialisiert
- Function- und RegExp-Objekte werden verworfen

JSON in Java mittels *org.json*

Objektserialisierung:

```
DemoBean demo = new DemoBean();  
demo.setId(1);  
demo.setName("lorem ipsum");  
demo.setActive(true);
```

```
JSONObject jo = new JSONObject(demo);
```

```
{"name":"lorem ipsum","active":true,"id":1}
```

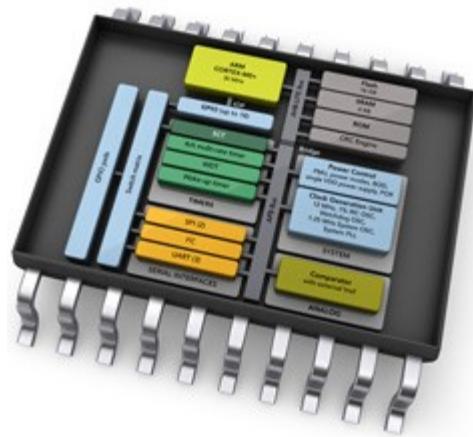
**Meistens nimmt man aber automatische
Marshaller/Unmarshaller wie Jackson**

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Das Internet ist ein Netz aus Netzen, Router als koppelnde Elemente zwischen den Teilnetzen

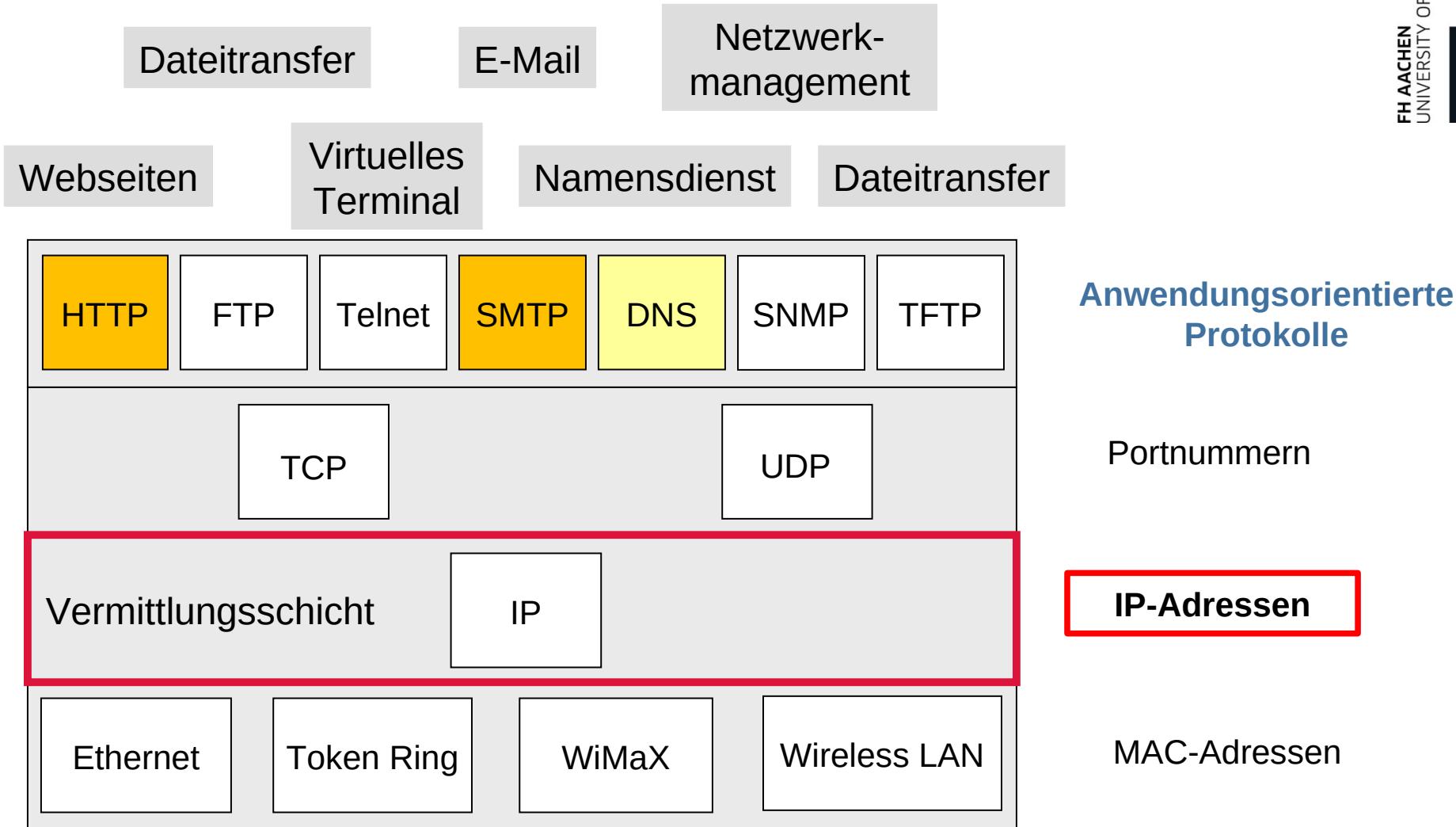
→ **Wie ist das Internet im Detail aufgebaut?**

Auf der Ebene der Vermittlungsschicht werden Pakete in einem Netz aus Netzen von einem Rechner zu einem beliebigen anderen Rechner geschickt

→ **Wie genau funktioniert das Internet auf der Vermittlungsschicht?**

Heute: Adressierung auf der Vermittlungsschicht

Internet-Referenzmodell im Überblick



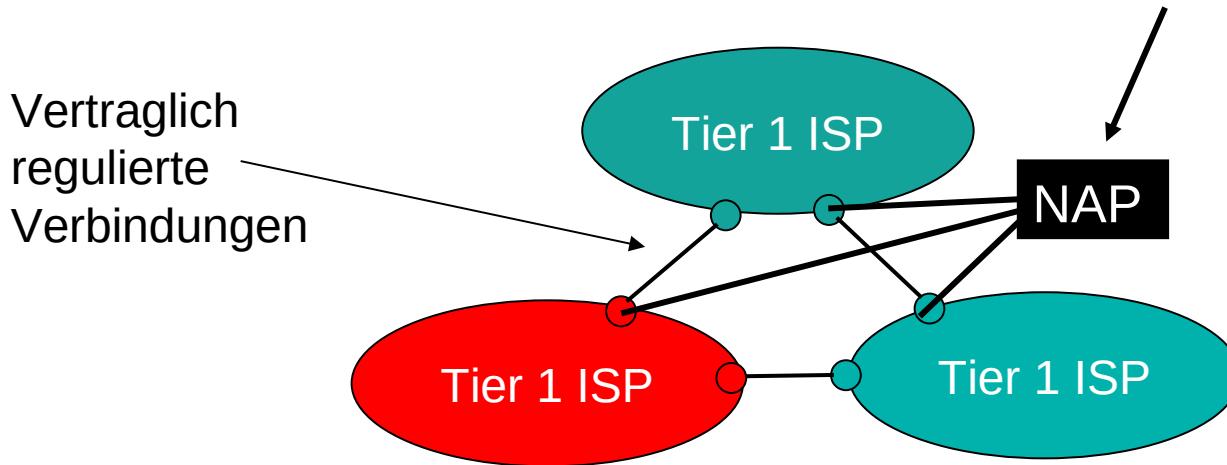
Basis des Internets sind sog. **autonome Systeme** (AS)

In IP-Netzen sind autonome Systeme (AS) ein Verbund von Routern und Netzwerken, die einer einzigen administrativen Instanz unterstehen, einer Organisation oder einem Unternehmen. Das bedeutet, dass sie alle zu einer Organisation oder zu einem Unternehmen gehören.

Aufbau des Internet

- Die Basis des Internets bilden “Tier-1” Internet Service Provider (ISPs), *engl.*: Tier → Schicht, Stufe
- z.B. Deutsche Telekom , BT, Qwest) mit nationaler und internationaler Überdeckung
 - Treten gleichberechtigt auf
 - Anbindungen an bestimmten Orten

Verbindung zwischen Providern:
Network Access Point (NAP) /
Internet Exchange Point (IXP)



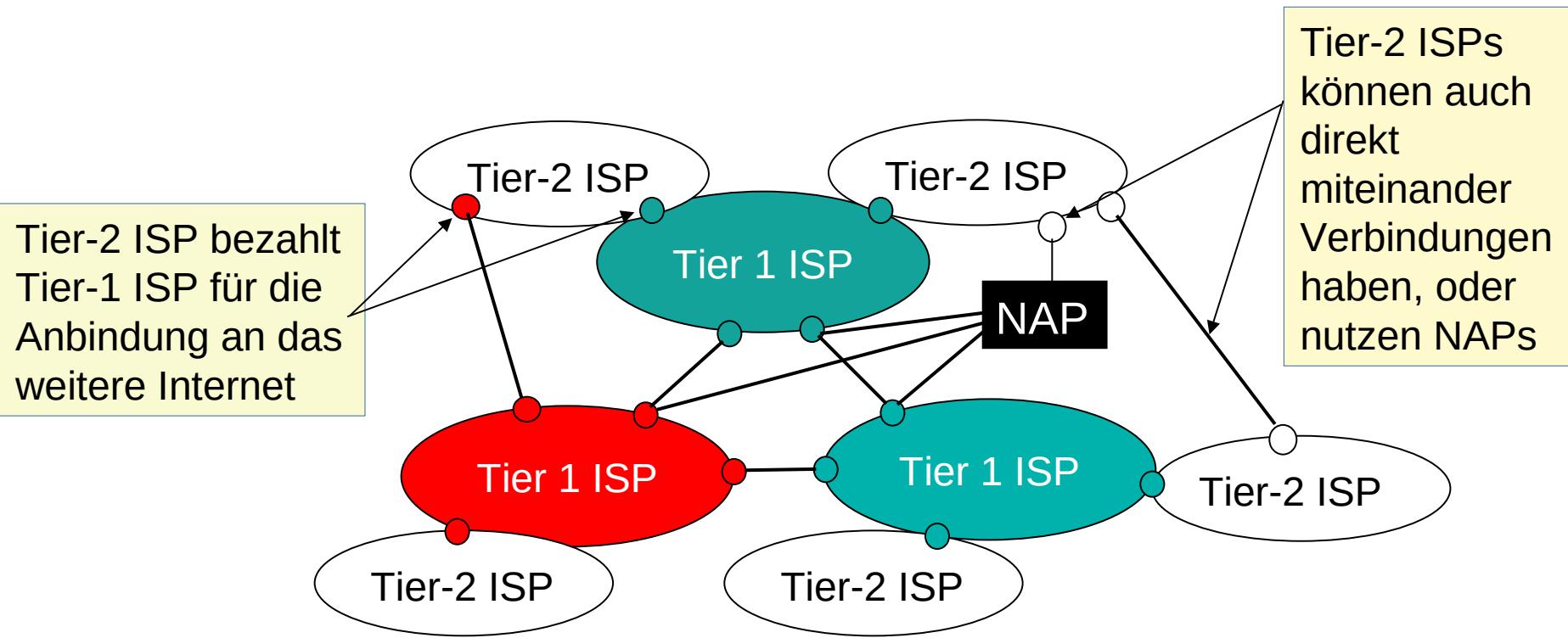
Aufbau des Internet - Tier 1 ISPs (Auswahl)

Name	Headquarters	AS number	CAIDA AS Rank ^[10]	Fiber Route Miles	Fiber Route km	Peering Policy
AT&T ^[11]	United States	7018	23	410,000	660,000 ^[12]	AT&T Peering policy ^[13]
CenturyLink (formerly Level 3) ^{[13][14]}	United States	3356	1	750,000	885,139 ^{[15][16]}	North America ^[17] ; International ^[18] Level 3 Peering Policy ^[19]
CenturyLink (formerly Level 3 formerly Global Crossing) ^{[13][14]}	United States	3549	12	750,000	885,139 ^{[15][16]}	North America ^[17] ; International ^[18] Level 3 Peering Policy ^[19]
Deutsche Telekom Global Carrier ^[17]	Germany	3320	20	155,343	250,000 ^[18]	DTAG Peering Details ^[20]
GTT Communications, Inc.	United States	3257	3	144,738	232,934 ^{[19][20]}	GTT Peering Policy ^[21]
Liberty Global ^{[21][22]}	United Kingdom ^[23]	6830	31	500,000	800,000 ^[24]	Peering Principles ^[25]
NTT Communications (America) (formerly Verio) ^[25]	Japan	2914	5	?	?	North America ^[26]
Orange (OpenTransit) ^[26]	France	5511	18	?	?	OTI peering policy ^[27]
PCCW Global	Hong Kong	3491	9	?	?	Peering policy ^[28]
Sprint (SoftBank Group) ^[27]	Japan	1239	27	26,000	42,000 ^[28]	Peering policy ^[29]
Tata Communications (formerly Teleglobe) ^[29]	India	6453	6	435,000	700,000 ^[30]	Peering Policy ^[31]
Telecom Italia Sparkle (Seabone) ^[31]	Italy	6762	8	347,967	560,000	Peering Policy ^[32]

Aufbau des Internet

“Tier-2” ISPs: kleinerer (oft nur Regional tätiger) ISPs

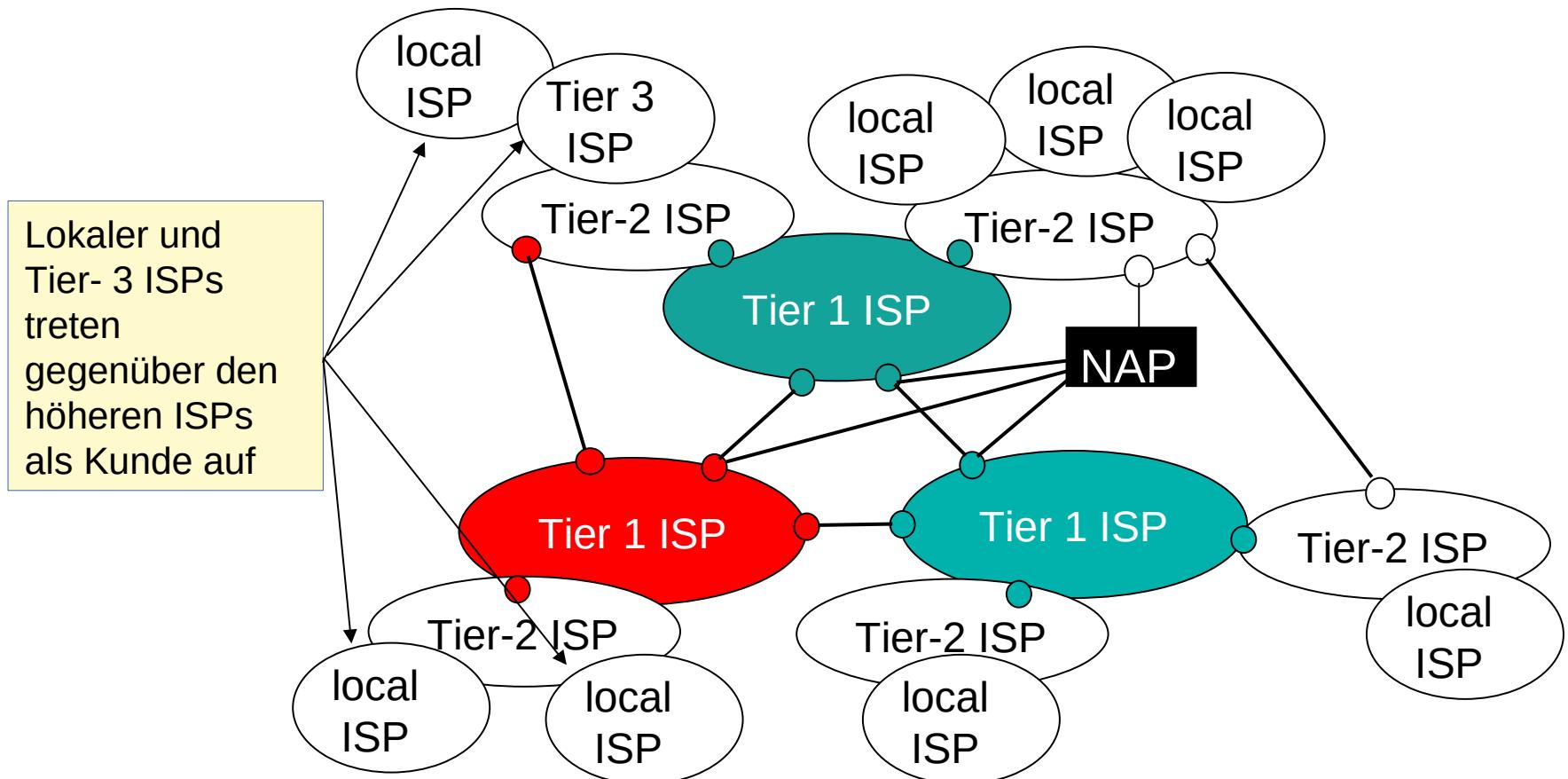
- Anbindung an einen oder mehrerer Tier-1 ISPs
- Ggf. Anbindung an weitere Tier-2 ISPs



Aufbau des Internet

“Tier-3” ISPs und lokale ISPs

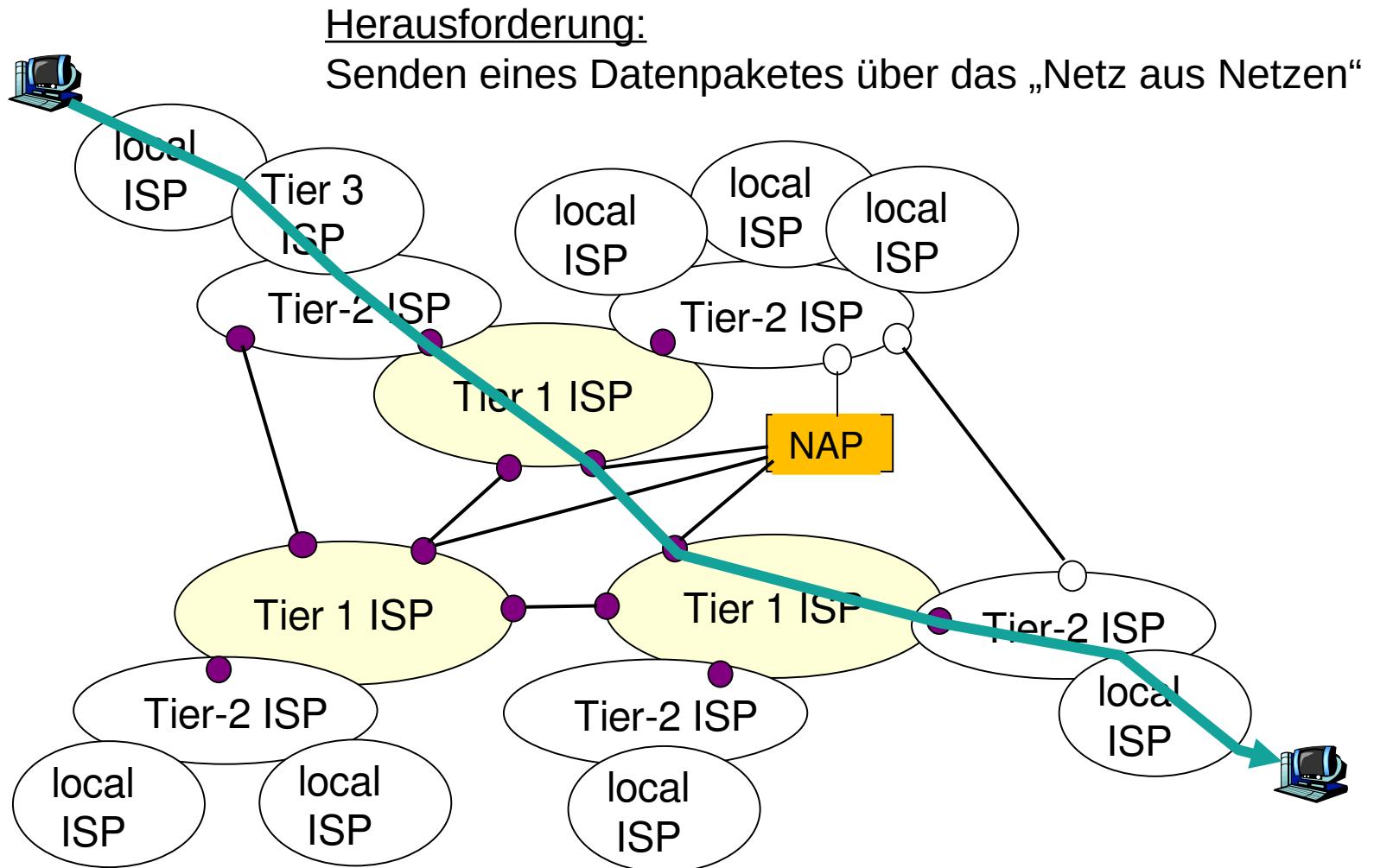
- Binden die Kunden an (“access network”, z.B. über DSL)
- Dienst nahe den Endsystemen



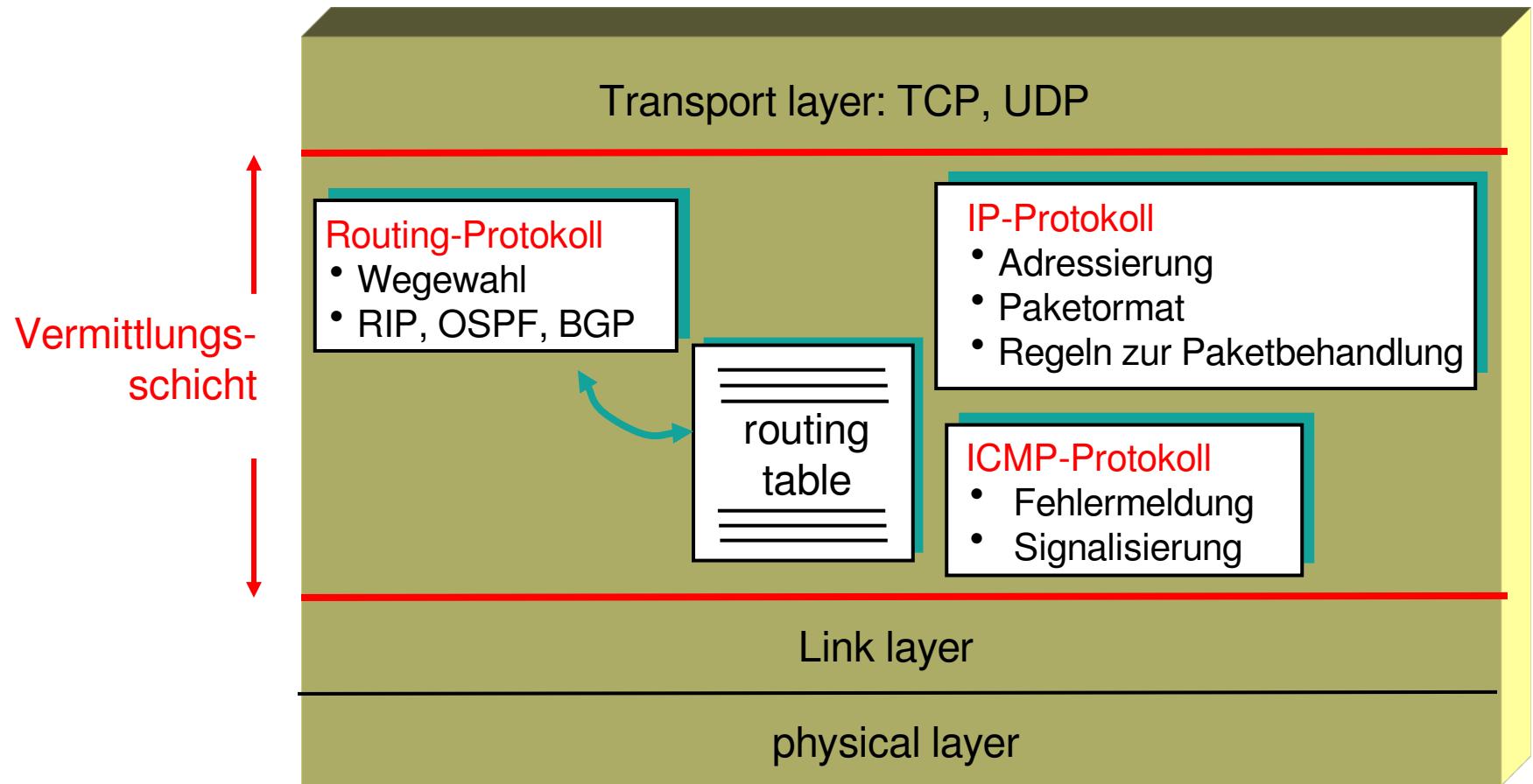
Zusammenfassung

- Das Internet ist grob in drei Schichten (*Tiers*) unterteilt
- Es gibt im Internet viele Eigentümer, so genannte Autonomous Systems (**AS**)
 - Diese können ein eigenes Geschäftsmodell entwickeln
 - Vertragswerke regulieren das Internet
 - Wenn ein AS mehr bei seinem Partner AS einspeist, als die vertragliche Regelung eigentlich erlaubt, dann können Daten verworfen werden...
- Ein Kommunikationsdienst nutzt im Allgemeinen die Infrastruktur mehrerer AS
- Eigentlich regelt das Internet recht wenig, verlangt aber, dass alle die gleiche **Vermittlungsschicht** besitzen

Nachrichtenzustellung



Die Vermittlungsschicht im Internet

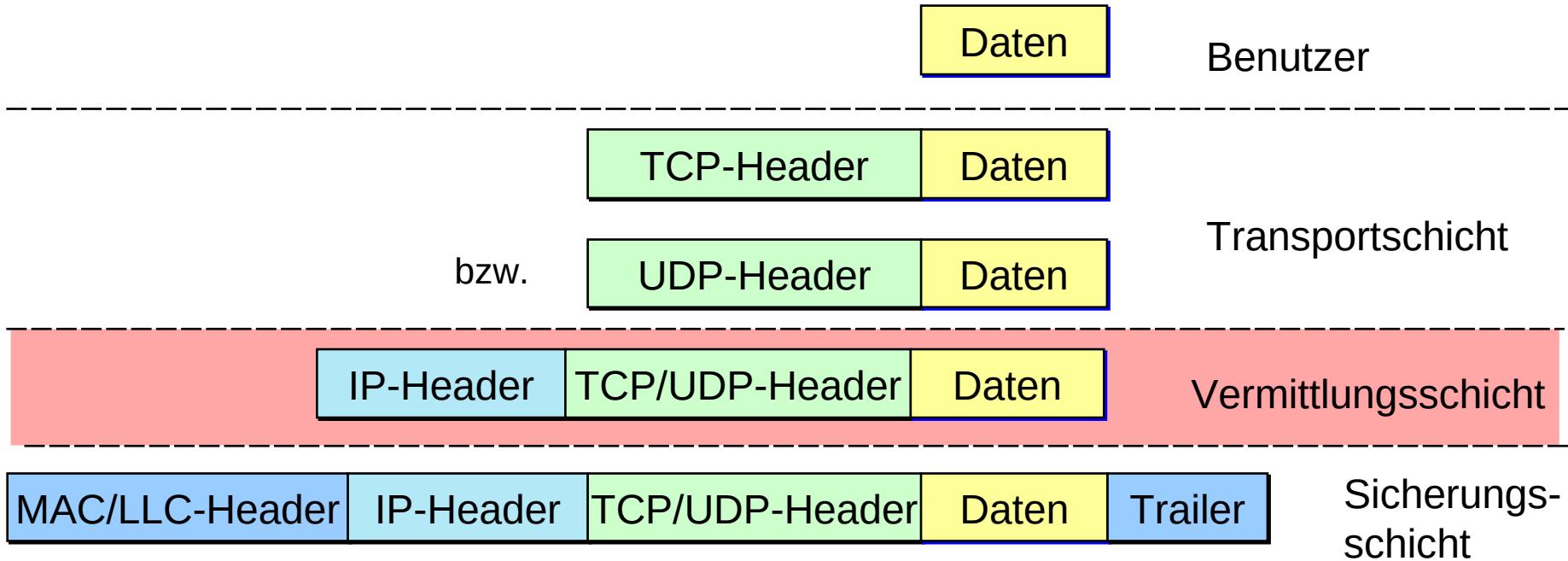


Das Internet-Protokoll

- IP bietet eine Ende-zu-Ende-Kommunikation zwischen Rechnern in einem Netz aus Netzen
- Derzeit flächendeckend eingesetzt werden die Protokolle **IPv4** und **IPv6**
- IP ist
 - paketvermittelnd (Datagramme), statisches Netzwerk
 - verbindungslos (Store-and-Forward) und von daher ungesichert:
 - ➔ Simulation einer ‚Sitzung‘ bzw. eines exklusiven Kommunikationskanals über TCP
 - ➔ Datagramme können verloren gehen
 - ➔ Datagramme können einander überholen
 - ➔ theoretisch können Datagramme auch mehrfach ankommen!

Zusammenspiel der Protokollinstanzen

- Prinzip der Kapselung
- TCP/UDP fügen Prozessadressierung (Ports) zu IP hinzu
- TCP sichert darüber hinaus die Datenübertragung
- IP leitet Datenpakete durch das Netzwerk zum Empfänger



IP – Internet Protocol: Aufgaben

Ende-zu-Ende-Kommunikation zwischen Rechnern
auch über Netzgrenzen hinweg

- Bereitstellung **weltweit eindeutiger Adressen**
 - > IPv4: 32 Bit
 - > Topologische Struktur der Adressen
- Definition eines **Paketformats**
 - > Header mit Kontrollinformationen
 - > Maximale Paketgröße: 64 KByte (in der Praxis: 1500 Byte)
- Wegewahl mit **Routing-Tabellen**
(gepflegt durch Routing-Protokolle → später)
- Zusammenspiel mit den tieferen Schichten
(ARP, Fragmentierung)

IP – Internet Protocol: Wegefindung

Ein IP-Paket wird an einen entfernten Host, der an einem entfernten Netzwerk angeschlossen ist, übertragen.

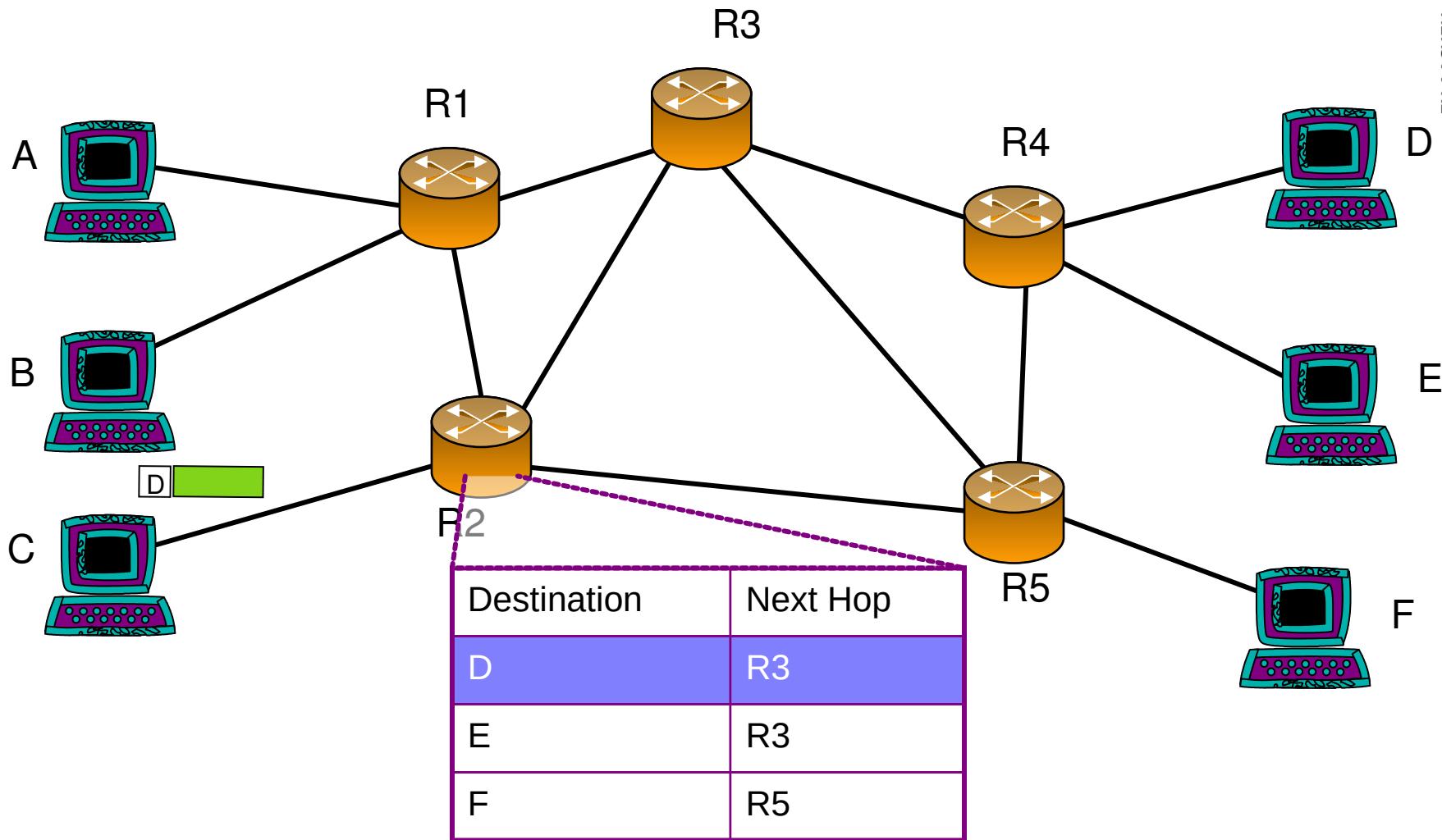
- Zwei Probleme sind daher zu lösen:
 - An welches **Netz** muss ich das Paket weiterleiten, wenn ich noch nicht im Zielnetz bin?
 - An welchen **Rechner** muss ich das Paket weiterleiten, wenn ich im Zielnetz bin?

Die Wegewahl im Internet-Protokoll

- Die IP-Implementierung eines Rechners oder Router-Systems entscheidet, wohin ein Datagramm übertragen wird
- Basis für diese Entscheidung sind die sogenannten **Routing/Forwarding-Tabellen**
- Diese Tabellen werden mittels der Informationen aus dem **IP-Header** durchsucht
- Normalerweise wird hierbei ausschließlich die Zieladresse des Pakets verwendet

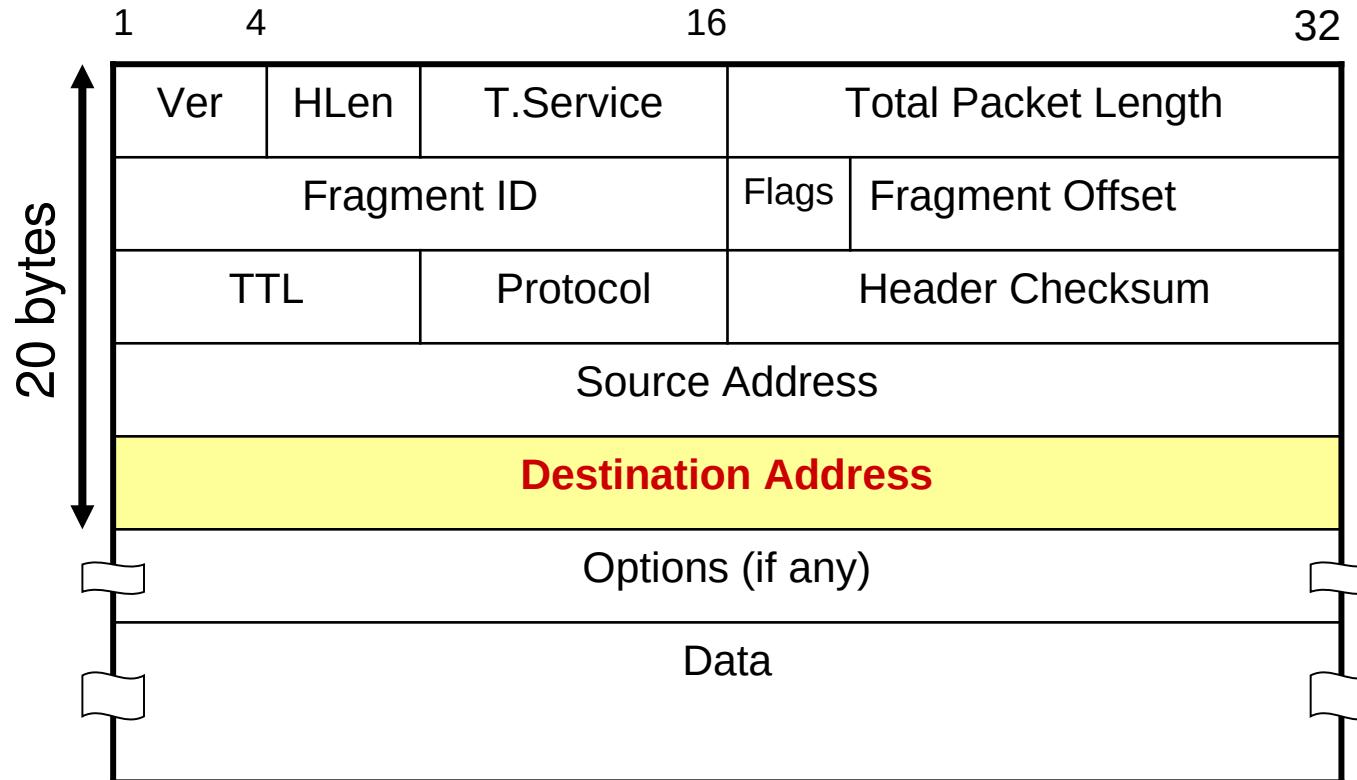
Merke: Jeder Router trifft diese Entscheidung für **jedes Paket** und zwar **unabhängig** von den anderen Routern

Der IP Routing



Der IP Header

- Den Header schauen wir uns noch zu einem späteren Zeitpunkt noch genauer an!
- Hier interessiert uns zunächst die **Ziel-IP-Adresse**, anhand derer der Weg ermittelt wird



Fallunterscheidung innerhalb eines Routers

- Kann das Datagramm **direkt** über Verfahren der Sicherungsschicht **an das Zielsystem** ausgeliefert werden, so wird das Paket zur Übertragung an die Sicherungsschicht weiter geleitet
 - Beide Systeme befinden sich im „gleichen Netz“ sein
 - Router muss irgendwie die HW-Adresse der Schicht 2 in Erfahrung bringen, um der Netzwerkkarte mitzuteilen, an wen sie übertragen soll
- Kann man **nicht direkt** mit dem Zielsystem kommunizieren, so muss die Nachricht **an einen weiteren “Router”** weiter geleitet werden, der sich in einem gleichen Netz befindet
 - Das Zielsystem befindet sich in einem anderen Netz
 - Mittels der Routing-Tabelle ermittelt das System, welcher direkt erreichbare Router das Paket zur Weiterleitung erhält (analog zum ersten Fall, der Router ist aus dieser Sicht einfach nur ein anderer Rechner).
 - Das Paket wird an die Sicherungsschicht zur Übertragung an den selektierten Router geleitet (indirekte Übertragung), die HW-Adresse des anderen Routers muss in Erfahrung gebracht werden.

Fallunterscheidung innerhalb eines Routers

Merke:

Router haben grundsätzlich Anschlüsse in mehr als in einem Netz!

Damit muss die IP-Vermittlungsschicht entscheiden, ob sich ein Zielrechner in einem der Netze befindet, an den der Router angeschlossen ist!

Der Router muss aus der Zieladresse also 2 Informationen extrahieren:

- In welches **Netz** muss das Paket ausgeliefert werden?
- Falls Router das Zielsystem direkt erreichen kann (also eine direkte Verbindung zum Zielnetz besitzt):
An welchen **Rechner** muss das Paket ausgeliefert werden?

Die Adresse muss also Informationen über das
Ziel-Netz und den Ziel-Rechner enthalten!
(vgl.: Vorwahl/Durchwahl beim Telefonieren)

IP-Adressierung (v4)

- Am Internet angeschlossene Systeme werden über eine weltweit eindeutige IP-Adresse identifiziert (Ausnahme: private Netze)
- IP-Adressen sind 32 Bit lang
- IP-Adressen haben eine bestimmte Struktur:
 - Der **vordere Teil bestimmt die Netzwerk-Adresse** für das **Netz** (z.B. **140.201**.10 .100), in dem sich der Rechner befindet. Innerhalb dieses Netzes kann eine direkte Kommunikation mittels der Sicherungsschicht geschehen
 - Der **hintere Teile bestimmt die Rechner-Adresse im gegebenen Netz** für einen Kommunikationsendpunkt, einen **Host** (z.B. 140.201.**10** .100)

Um unterschiedlich große Netze zu unterstützen wurden mehrere Adressklassen definiert

(Heute wird zwar noch die Terminologie verwendet, aber nicht mehr diese recht statische Klassenaufteilung!)

IP-Adressklassen: Der ursprüngliche Ansatz

1. Class A für Netze mit bis zu 16 Mio. Knoten (0-127)



2. Class B für Netze mit bis zu 65.536 Knoten (128-191)



3. Class C für Netze mit bis zu 256 Knoten (192-223)



4. Class D für Gruppenkommunikation (Multicast) (224)



5. Class E, noch reserviert für zukünftige Anwendungen



Aufteilung der IP Adresse in eine

- Netz-ID
- Knoten-ID

Bei der Interpretation der IP-Adressen sind für den Rechnerteil (Knoten-ID im Netz) zwei Adressen reserviert:

Knoten-ID = 0...0 bezeichnet das Netz selber

Knoten-ID = 1...1 bezeichnet eine Broadcast im Netz

IP-Adressen / Adressklassen

Class	Start	Ende	Anzahl Netze	Rechner / Netz
A	1.0.0.0	127.255.255.255	127	16.777.214
B	128.0.0.0	191.255.255.255	16384	65534
C	192.0.0.0	223.255.255.255	2.097.152	254
D	224.0.0.0	239.255.255.255	<i>Multicast</i>	
E	240.0.0.0	255.255.255.254	<i>Reserved</i>	

IP-Adressen / Private Netzwerke

Netzadressbereich	CIDR-Notation	Verkürzte CIDR-Notation	Anzahl Adressen	Anzahl Netze gemäß Netzklasse (historisch)
10.0.0.0 bis 10.255.255.255	10.0.0.0/8	10/8	$2^{24} =$ 16.777.216	Klasse A: 1 privates Netz mit 16.777.216 Adressen; 10.0.0.0/8
172.16.0.0 bis 172.31.255.255	172.16.0.0/12	172.16/12	$2^{20} =$ 1.048.576	Klasse B: 16 private Netze mit jeweils 65.536 Adressen; 172.16.0.0/16 bis 172.31.0.0/16
192.168.0.0 bis 192.168.255.255	192.168.0.0/16	192.168/16	$2^{16} =$ 65.536	Klasse C: 256 private Netze mit jeweils 256 Adressen; 192.168.0.0/24 bis 192.168.255.0/24

Die Loopback-Adresse

Eine Loopback-Adresse kann immer dann verwendet werden, wenn eine Kommunikation mit einer Zielanwendung statt finden soll, die auf dem gleichen Rechner läuft

Während IP-Adresse faktisch sonst immer mit einer Netzwerkkarte assoziiert sind ist ein Loopback nicht mit einer physikalisch vorhandenen Karte assoziiert

Damit beschreitet die Kommunikation die Anwendungs-Transport und Vermittlungsschicht und geht über das Loopback direkt über Vermittlungs-, Transport- zur Anwendungsschicht

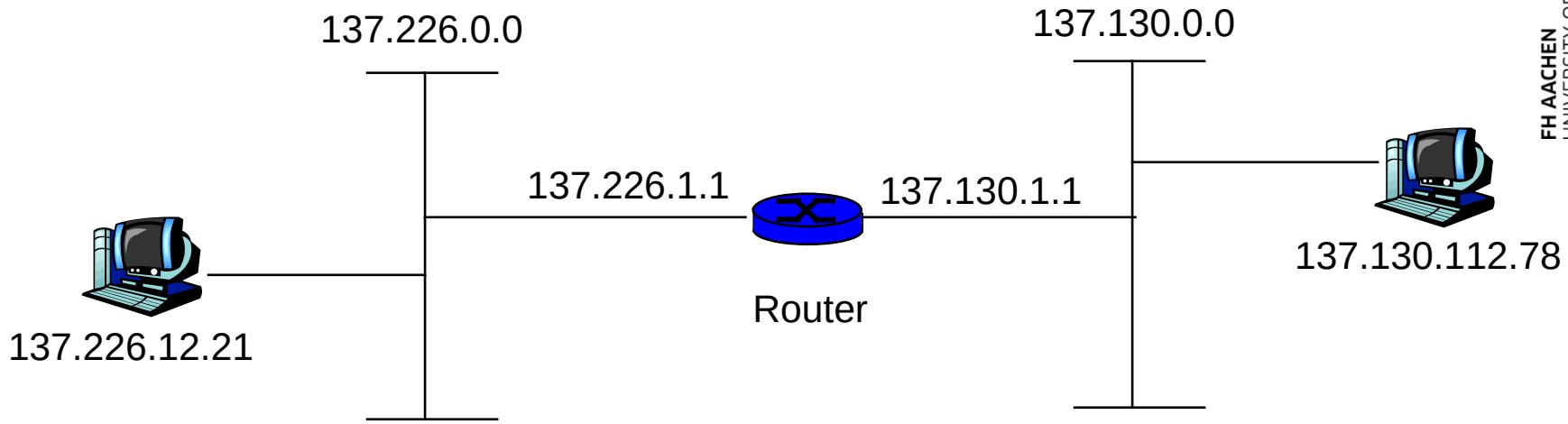
Eine zu konfigurierende Netzwerkkarte wird nicht angesprochen

127.x.x.x ist reserviert (loopback = 127.0.0.1)

Die Loopback-Adresse

Merke: Auch wenn man verschwenderisch für das Loopback eine Class-A-Adresse vorgehen hat, so schreibt der Standard RFC3330 doch vor, dass die Adresse 127.0.0.1 das (virtuelle) Loopback-Interface bezeichnet

IP-Adresse

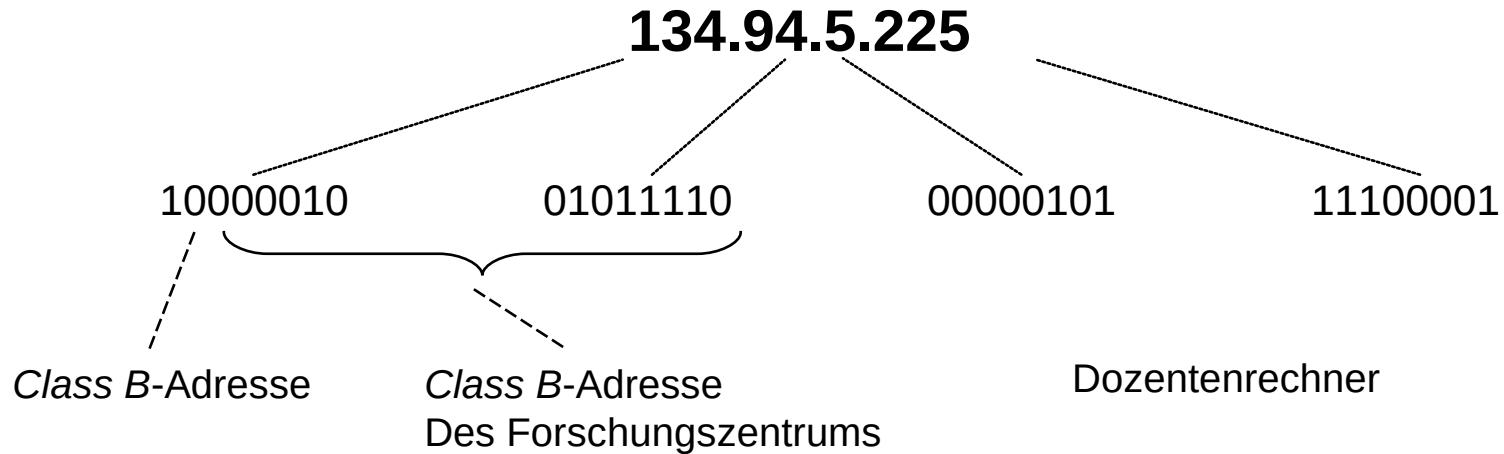


Binärformat	10001001 11100010 00001100 00010101
Dotted Decimal Notation	137.226.12.21

- jeder Rechner hat (wenigstens) eine weltweit eindeutige IP-Adresse
 - **Ausnahme: private Adressen mit Adressumsetzung** im Router: 10.x.x.x, 172.16.0.0 - 172.31.255.255 192.168.0.0 - 192.168.255.255
- Router oder Gateways, die mehrere Netze miteinander verknüpfen, haben für jedes angeschlossene Netz eine IP-Adresse

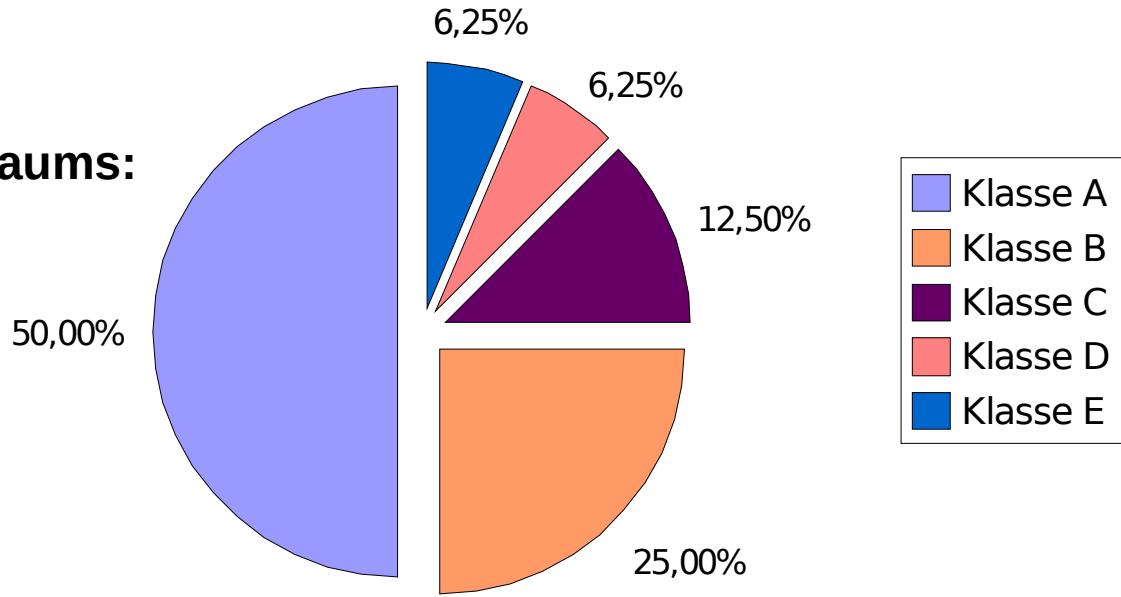
IP-Adressierung - Beispiele

Die Darstellung der 32-Bit-Adresse erfolgt in 4 Teilstücken zu je 8 Bit:



IP-Adressen

Aufteilung des Adressraums:



IP-Adressen werden knapp...

Problem

- Niemand hatte damals mit einem derart starken Wachstum des Internet gerechnet (sonst hätte man von Anfang an längere Adressen definiert)
- Allzu viele Class A-Adressen wurden in den ersten Internetjahren vergeben
- Ineffiziente Nutzung des Adressraums

Beispiel: wenn 500 Geräte in einem Unternehmen angeschlossen werden sollen, braucht man eine Class B-Adresse, die unnötigerweise mehr als 65.000 Rechneradressen blockiert.

Lösungsversuch

Erweiterung des Adressraums bei IPv6 gegenüber der aktuellen Version IPv4

- ⇒ IP Version 6 hat 128 Bit-Adressen
- ⇒ 7×10^{15} IP-Adressen pro Quadratmilimeter der Erdoberfläche (incl. der Ozeane!)
- ⇒ Vergleich IPv4: 8,5 IP-Adressen pro Quadratkilometer!

Ein weiteres Problem...

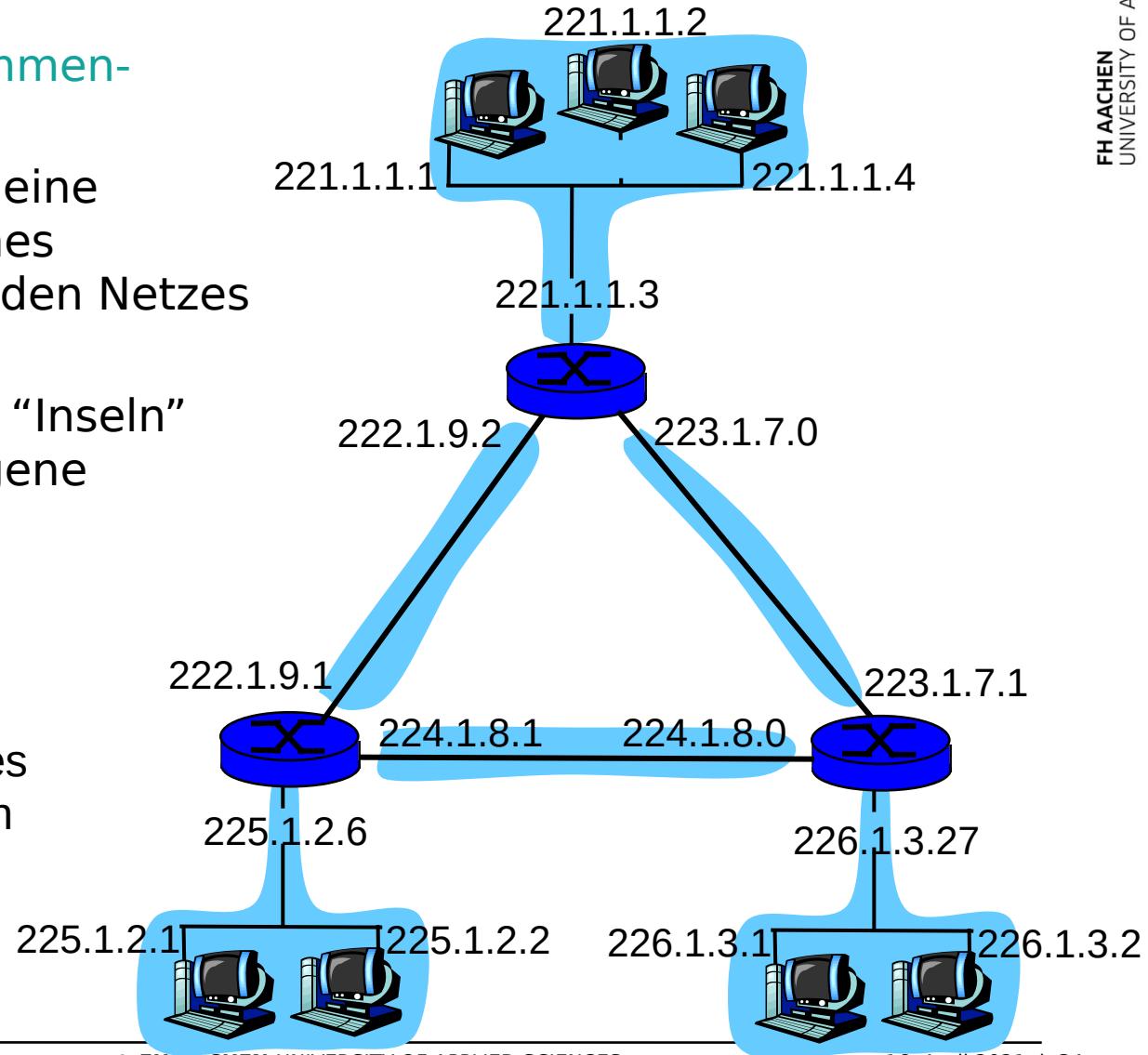
- Router verbinden immer Netze, wodurch beispielsweise unterschiedliche Netztechniken in den einzelnen Netzen eingesetzt werden können
- Damit **erfordern** sie aber auch eine **Änderung der Netzadresse**, wenn sie in mehreren Netzen sein sollen
- Router trennen damit Netze, was diesen eine verbesserte Struktur gibt
- Institutionen mit größeren Netzen möchten auch intern Router verwenden (strukturierte Netzplanung)
- Noch haben diese aber immer nur „eine“ an sie vergebene Netzadresse

IP-Adressierung - Bedeutung des Netzadressteils

Wie findet man zusammenhängende Netze?

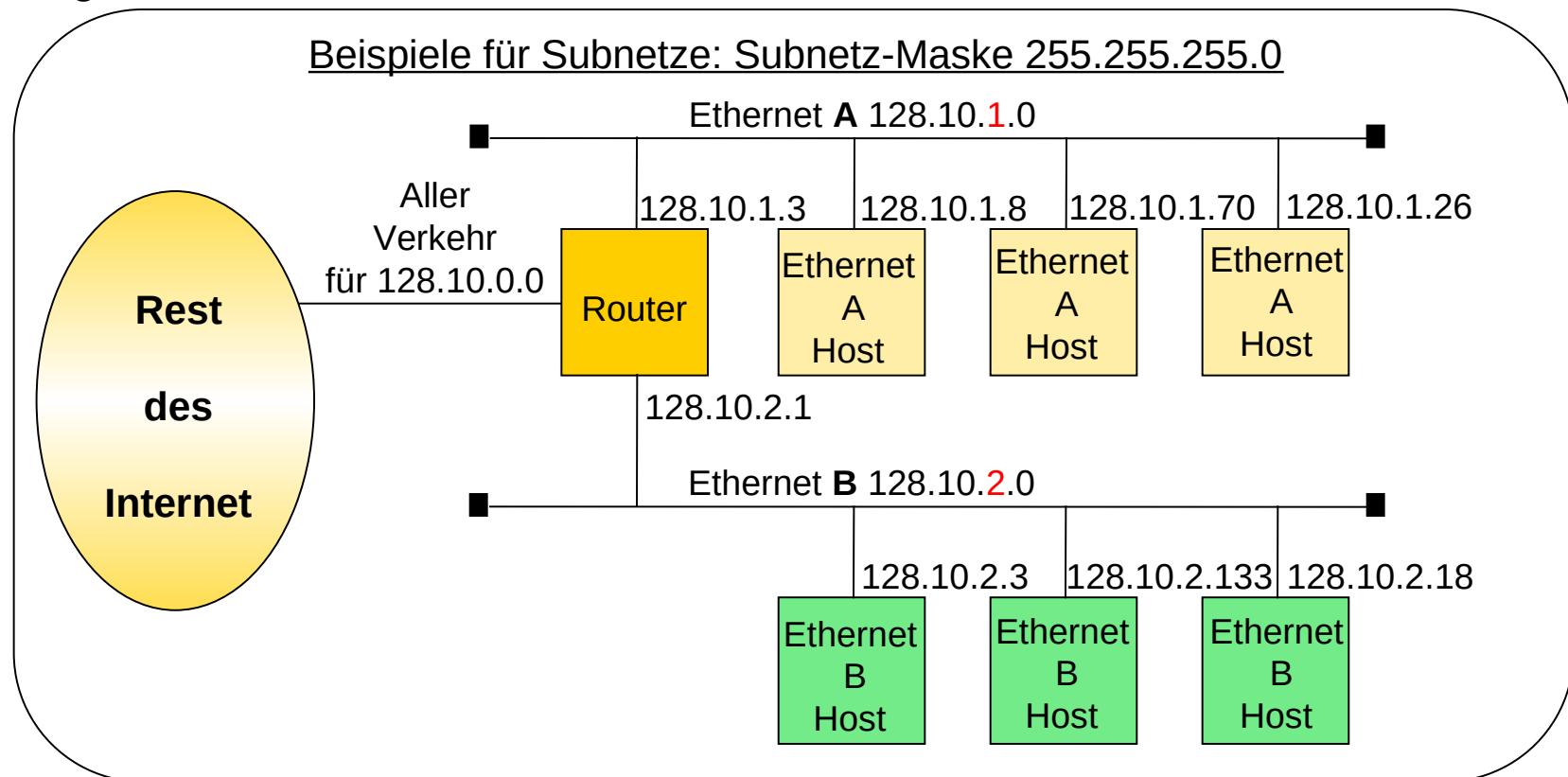
- Jeder Router stellt eine Unterbrechung eines zusammenhängenden Netzes dar
- Die verbleibenden "Inseln" benötigen eine eigene Netzadresse
→ Subnetze!!!

Netzwerk, welches aus sechs Netzen besteht



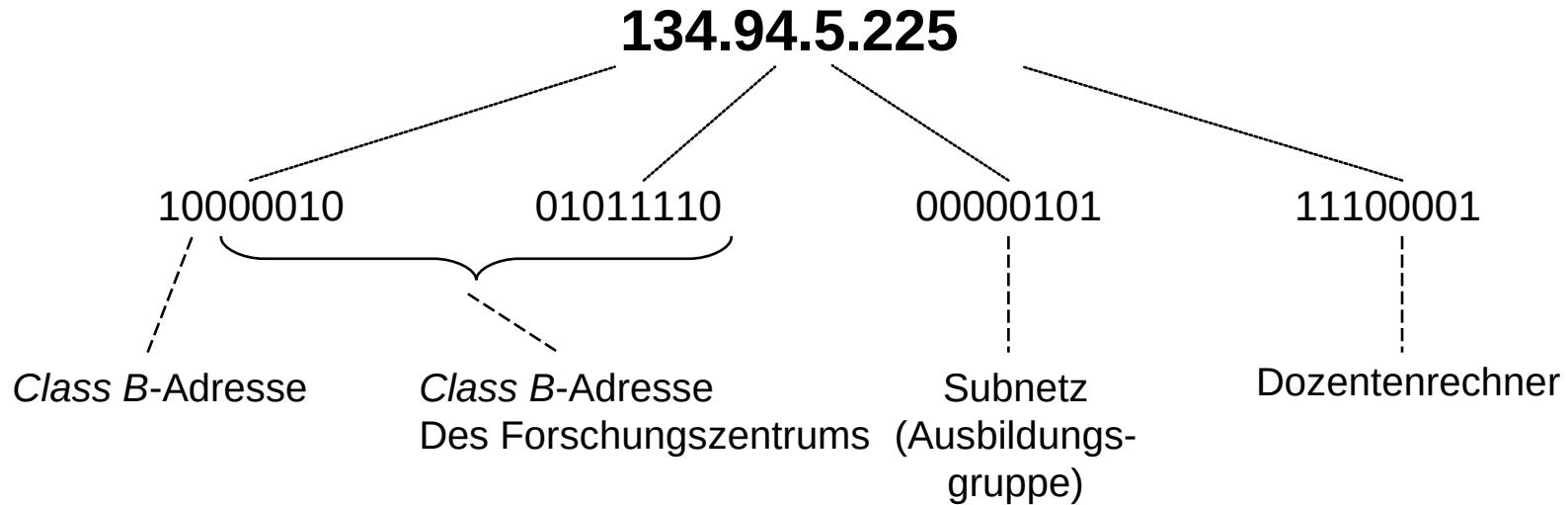
IP-Subnetze

Problem: Class C-Netze sind sehr klein, Class B-Netze oft aber schon wieder zu groß, um sie ohne Router zu konzipieren. Daher gibt es die Möglichkeit, ein durch die IP-Adresse identifiziertes Netz in so genannte **Subnetze** zu zerlegen.



IP-Adressierung - Subnetze

Die Darstellung der 32-Bit-Adresse erfolgt in 4 Teilstücken zu je 8 Bit:

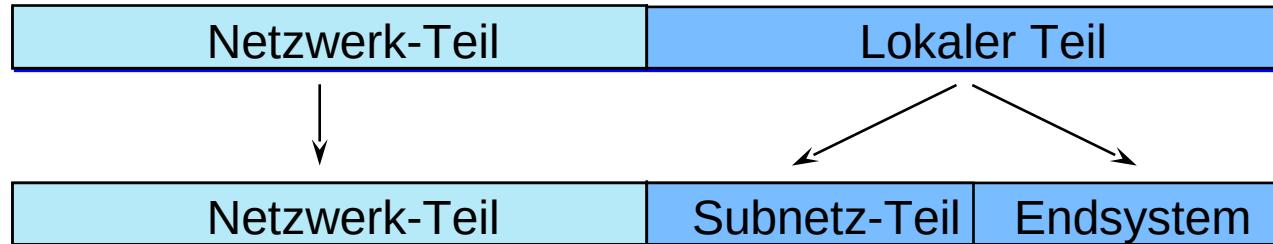


Mittels der Subnetzmaske wird eine Folge **zusammenhängender** Bits der Adresse angegeben, die den Netzwerkadressteil bestimmt.

1...10...0

IP-Subnetz-Adressen

- **IP-Adresse** (hier Klasse B):



- **Subnetzmasken** kennzeichnen den Bereich der IP-Adresse, der das Netzwerk und das Subnetzwerk beschreibt. Dieser Bereich wird dabei durch Einsen („1“) in der binären Form der Subnetzmaske festgestellt.

- Beispiel:

140.	201.	10.	100
255.	255.	255.	0

Netzwerk:

140.

Subnetz:

201.

Endsystem:

10.

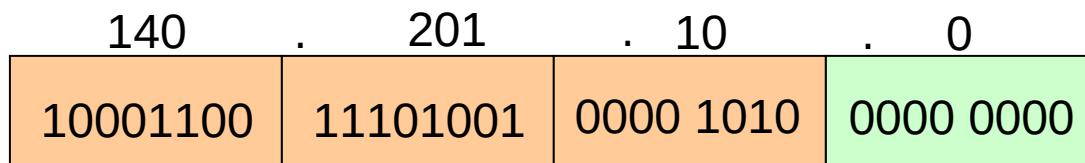
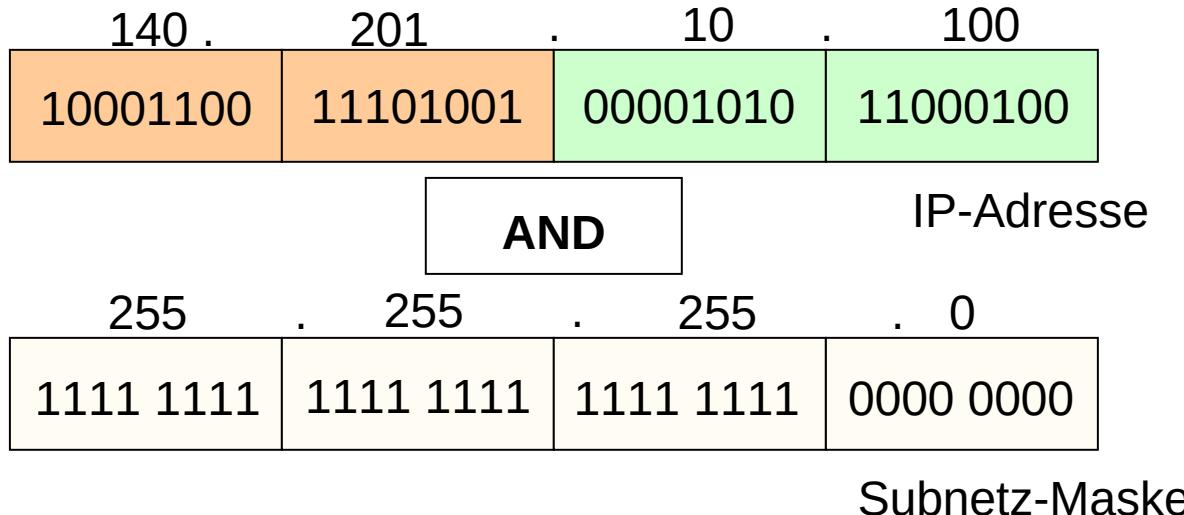
100

- Der Netzwerk-Teil kann aus der Adressklasse abgeleitet werden
- Überdeckt die Subnetzmaske nur den Netzwerk-Teil, dann gibt es keinen Subnetz-Teil (z.B. 255.255.0.0)

Beachten Sie...

- Die Subnetzmaske ist faktisch eine binäre Maske
- Sie beginnt mit einer 1 und es gibt nur einen Wechsel auf 0
- Sie wird über die Binärdarstellung der Zieladresse gelegt
- Die UND-Verknüpfung liefert die Netzwerkadresse
(Knoten-ID = 0...0 bezeichnet ja das Netzwerk!)

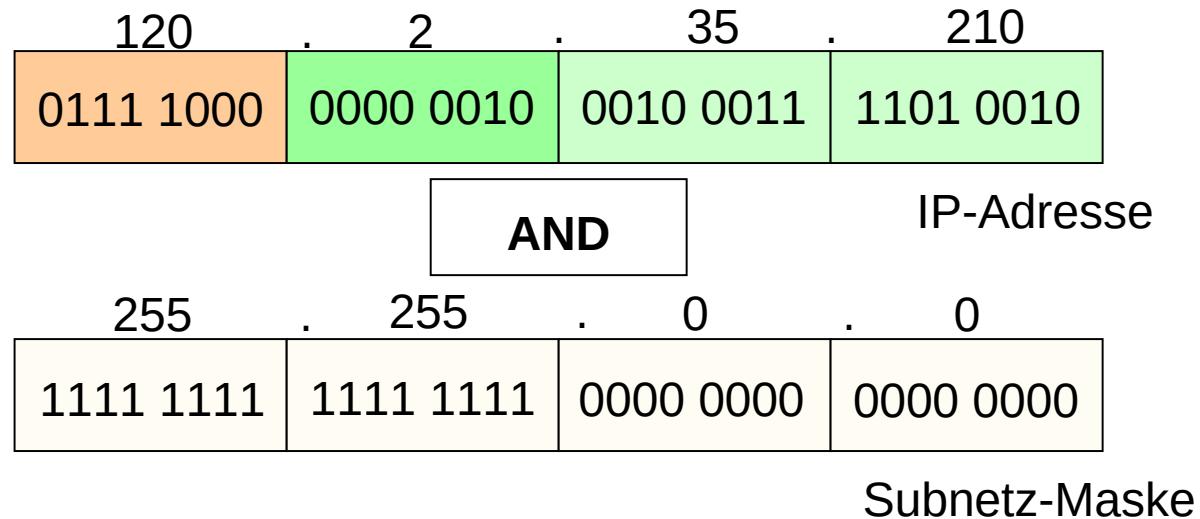
IP-Subnetze - Berechnung des Zielhosts



Netzwerk des bezeichneten Hosts
nach Anwendung des Subnetz-Regel

IP-Subnetze - Berechnung des Zielhosts

Keine Aufteilung in Subnetze, es liegt ein virtuelles Class-B-Netz vor:



120	.	2	.	0	.	0
0111 1000		0000 0010		0000 0000		0000 0000

Netzwerk des bezeichneten Hosts

Beachten Sie...

Subnetzmasken...

- können Netze unterteilen
- erschweren das Erkennen der Netzadressteile
(nicht für den Rechner)
- haben lokale Bedeutung
(auf dem System auf dem sie konfiguriert sind)
- gehören zur systematischen Netzplanung hinzu
- Innerhalb eines Subnetzes sollte die Rechneradresse
(binär) „0...0“ und die Rechneradresse
(binär) „1....1“ nicht verwendet werden

Warum nicht?

Aufgabe:

Sie haben die Class-B-Adresse 134.94.0.0 erhalten

Bestimmen Sie eine Subnetzmaske, mit der Sie dieses Netz in 16 Subnetze untergliedern

Geben Sie für die ersten beiden Netze den Adressbereich für Rechner in diesem Netz an

Sie haben die Class-B-Adresse 134.94.0.0 erhalten

Welche Subnetzmaske müssen Sie nehmen, wenn Sie das Netz in 12 Unternetze aufteilen wollen?

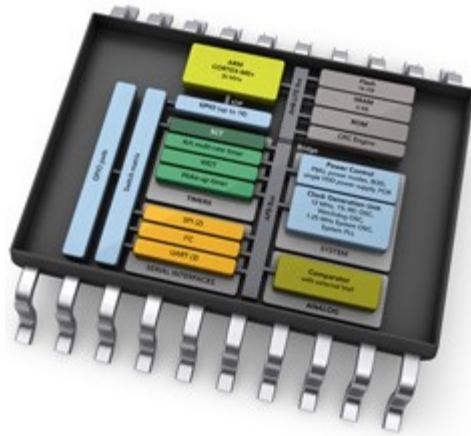
FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

(Modulcode 941306)

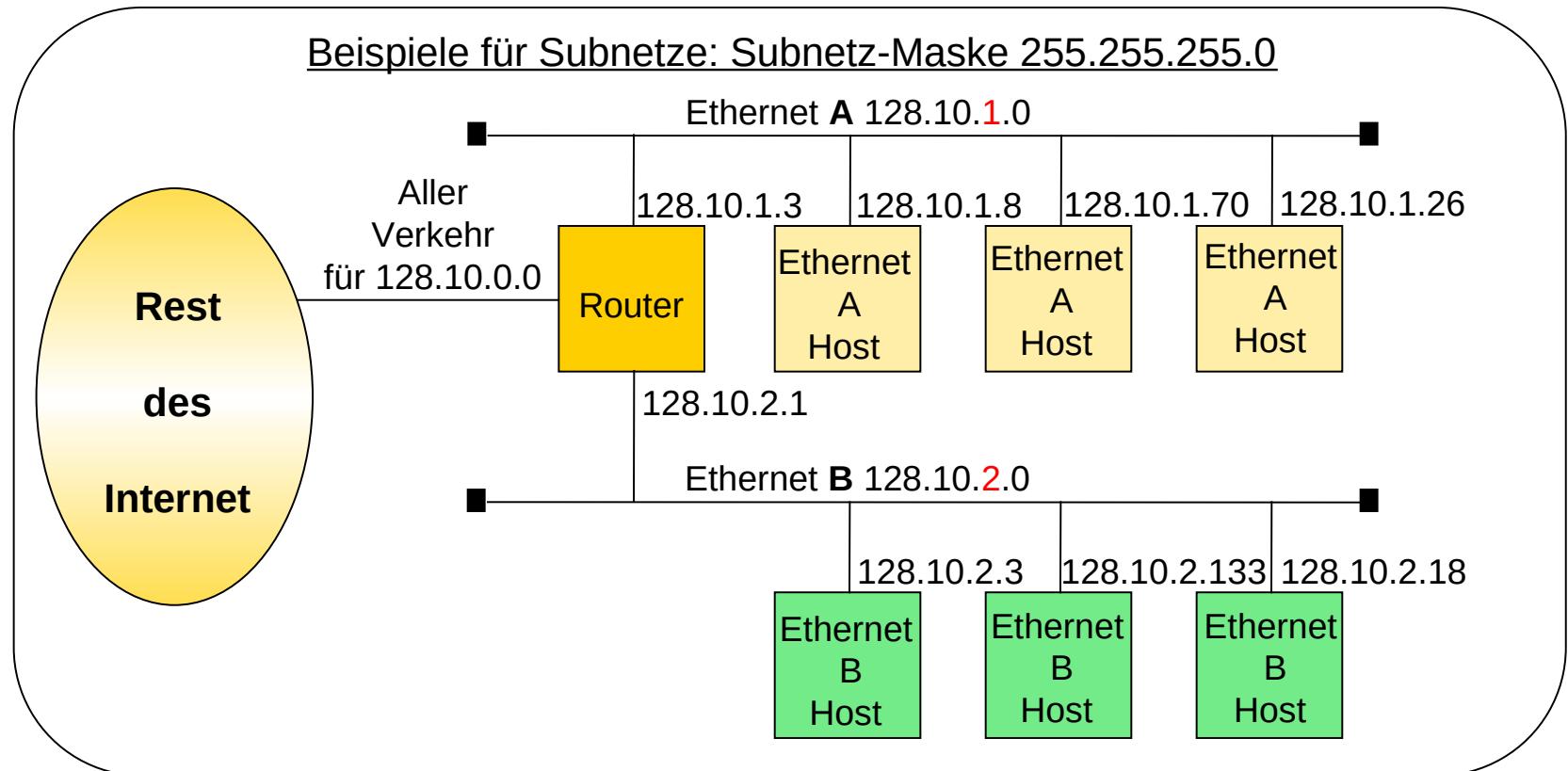
Prof. Dr. Andreas Terstegge

SS 2020



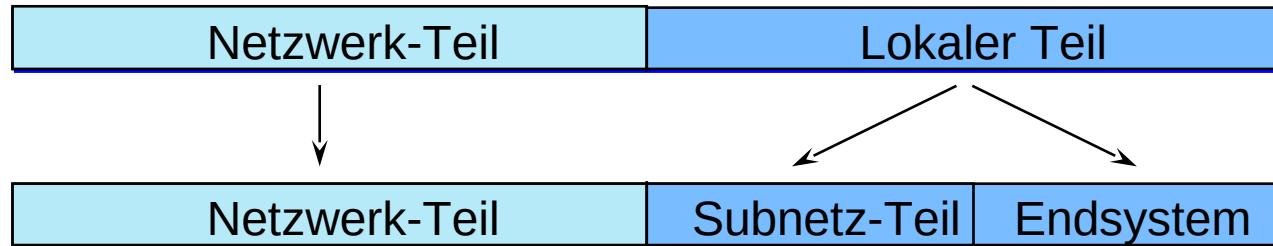
IP-Subnetze

Problem: Class C-Netze sind sehr klein, Class B-Netze oft aber schon wieder zu groß, um sie ohne Router zu konzipieren. Daher gibt es die Möglichkeit, ein durch die IP-Adresse identifiziertes Netz in so genannte **Subnetze** zu zerlegen.



IP-Subnetz-Adressen

- **IP-Adresse** (hier Klasse B):



- **Subnetzmasken** kennzeichnen den Bereich der IP-Adresse, der das Netzwerk und das Subnetzwerk beschreibt. Dieser Bereich wird dabei durch Einsen („1“) in der binären Form der Subnetzmaske festgestellt.

- Beispiel:

140.	201.	10.	100
255.	255.	255.	0

Netzwerk:	140.	201.	10.	100
Subnetz:				
Endsystem:				

- Der Netzwerk-Teil kann aus der Adressklasse abgeleitet werden
- Überdeckt die Subnetzmaske nur den Netzwerk-Teil, dann gibt es keinen Subnetz-Teil (z.B. 255.255.0.0)

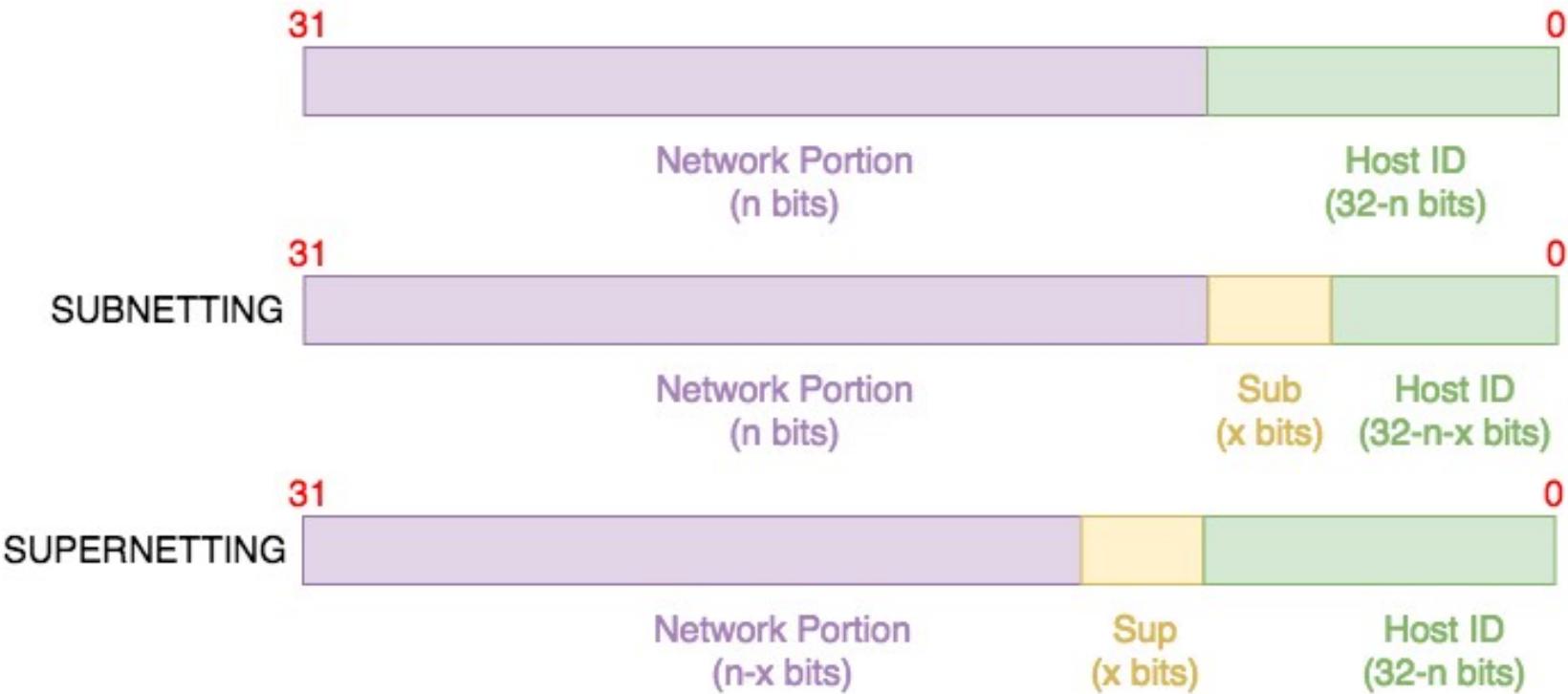
Flexiblere Adressierung

„Durch das Vorgehen, den Adressraum in nur drei Klassen zu organisieren, werden Millionen von Adressen durch Fragmentierung verloren. Insbesondere finden sich fast nur im Bereich der Class-C-Netze noch Freiräume.

Lösung: Classless Inter-Domain Routing (CIDR)

- Trennung von starrer Klasseneinteilung durch Ersetzen der festen Klassen durch Netzwerk-Präfixe variabler Länge
- Die Längenangabe sagt aus, wie viele Bit als Netzteil der Adresse verwendet werden sollen (Länge der 1-Folge)
- Router merken sich in ihrer Routing-Tabelle zusätzlich zu den IP-Adressen die Präfixlänge,
z.B. 194.142.0.x/17 = betrachte die ersten 17 Bit als Netzadresse
- Sehr flexible Gestaltung von Routing-Tabellen möglich

Subnetting - Supernetting



- **Subnetting:** Der Verfügbare Host-Bereich wird in weitere Unternetze (Subnets) unterteilt
- **Supernetting:** Zusammenfassen mehrere Netze mit teilweise übereinstimmendem Network-Bereich zu einem gemeinsamen Netz (aus Sicht des Routings)

CIDR – Classless InterDomain Routing

Einige Beispieladressen

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Bei der Anzahl der möglichen Knoten muss die

Netzwerk-Adresse (Host-Teil nur 0-Bits) und
Broadcast-Adresse (Host-Teil nur 1-Bits)

abgezogen werden (also immer 2 Adressen weniger)!

CIDR-Adressblöcke

CIDR Block Prefix	# of Host Addresses
/27	32
/26	64
/25	128
/24	256
/23	512
/22	1,024
/21	2,048
/20	4,096
/19	8,192
/18	16,384
/17	32,768
/16	65,536
/15	131,072
/14	262,144
/13	524,288

Bei der Anzahl der möglichen Knoten muss die

- Netzwerk-Adresse (Host-Teil nur 0-Bits) und
- Broadcast-Adresse (Host-Teil nur 1-Bits)

abgezogen werden (also immer 2 Adressen weniger)!

Problem

Während wir vorher die Routing-Tabelle sehr einfach durchsuchen konnten verursacht CIDR, dass es mehrere gültige Einträge geben kann

Beispiel:

Die Zieladresse 134.94.80.2 ist

für 134.94.0.0/16 als auch für 134.94.80.0/24 zutreffend

Wegewahl: Longest Prefix Match

- Suche nach dem Routing-Eintrag mit der größten Überdeckung der Zieladresse
- Es befinden sich auch Einträge für einzelne Rechner (Host route, loopback entry)
→ 32-bit prefix match
- Default route wird als 0.0.0.0/0 repräsentiert
→ 0-bit prefix match

Destination address	Next hop
10.0.0.0/8	R1
128.143.0.0/16	R2
128.143.64.0/20	R3
128.143.192.0/20	R3
128.143.71.0/24	R4
128.143.71.55/32	R3
default	R5

Der longest prefix match für **128.143.71.21** wird für 24 bits mit dem Eintrag **128.143.71.0/24** erreicht. Datengramm wird nach R4 verschickt

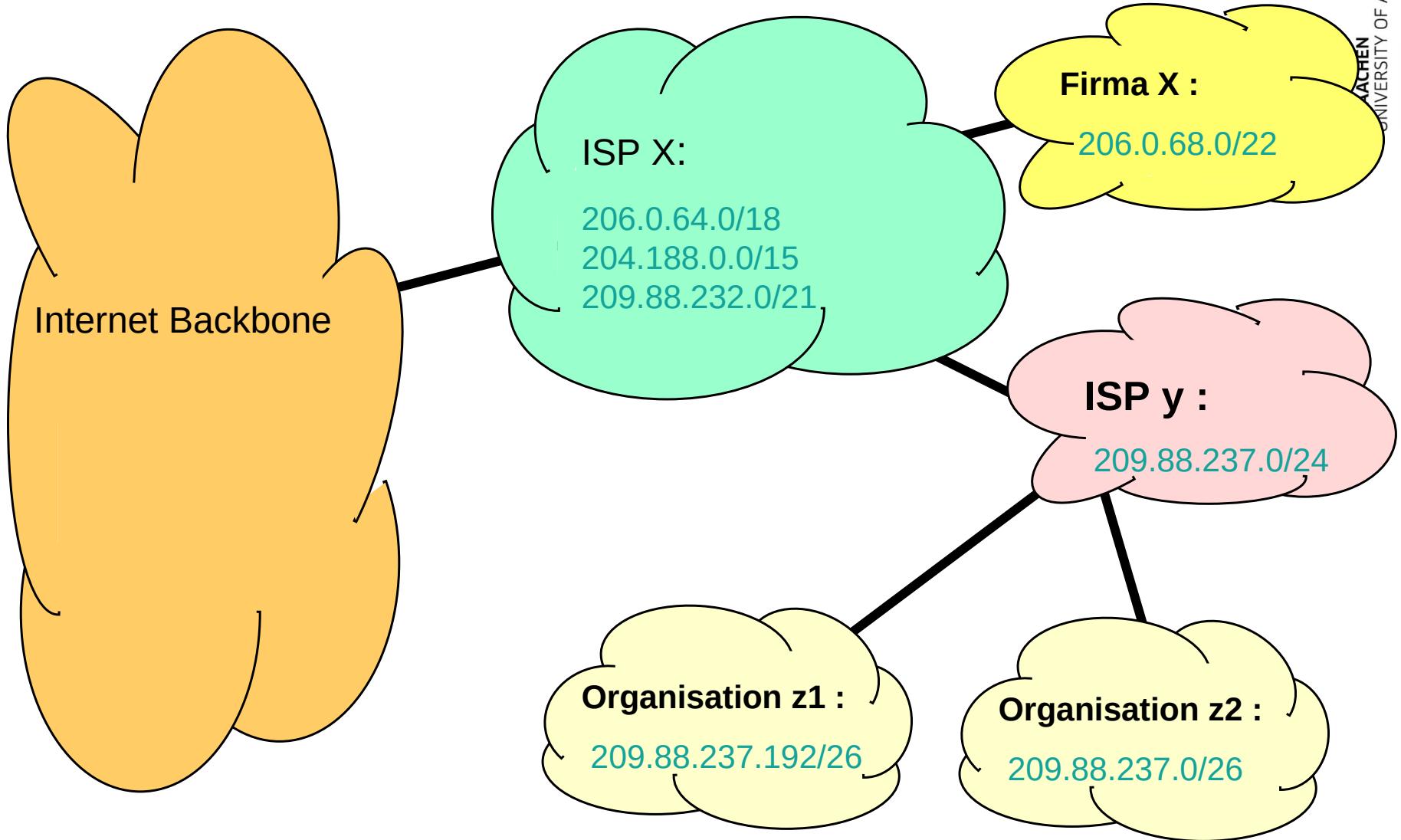
Longest Prefix Match: Ein Beispiel

Ziel	11.1.2.5	= 00001011.0000001.00000010.00000101
Route #1	11.1.2.0/24	= 00001011.0000001.00000010.00000000
Route #2	11.1.0.0/16	= 00001011.0000001.00000000.00000000
Route #3	11.0.0.0/8	= 00001011.0000000.00000000.00000000

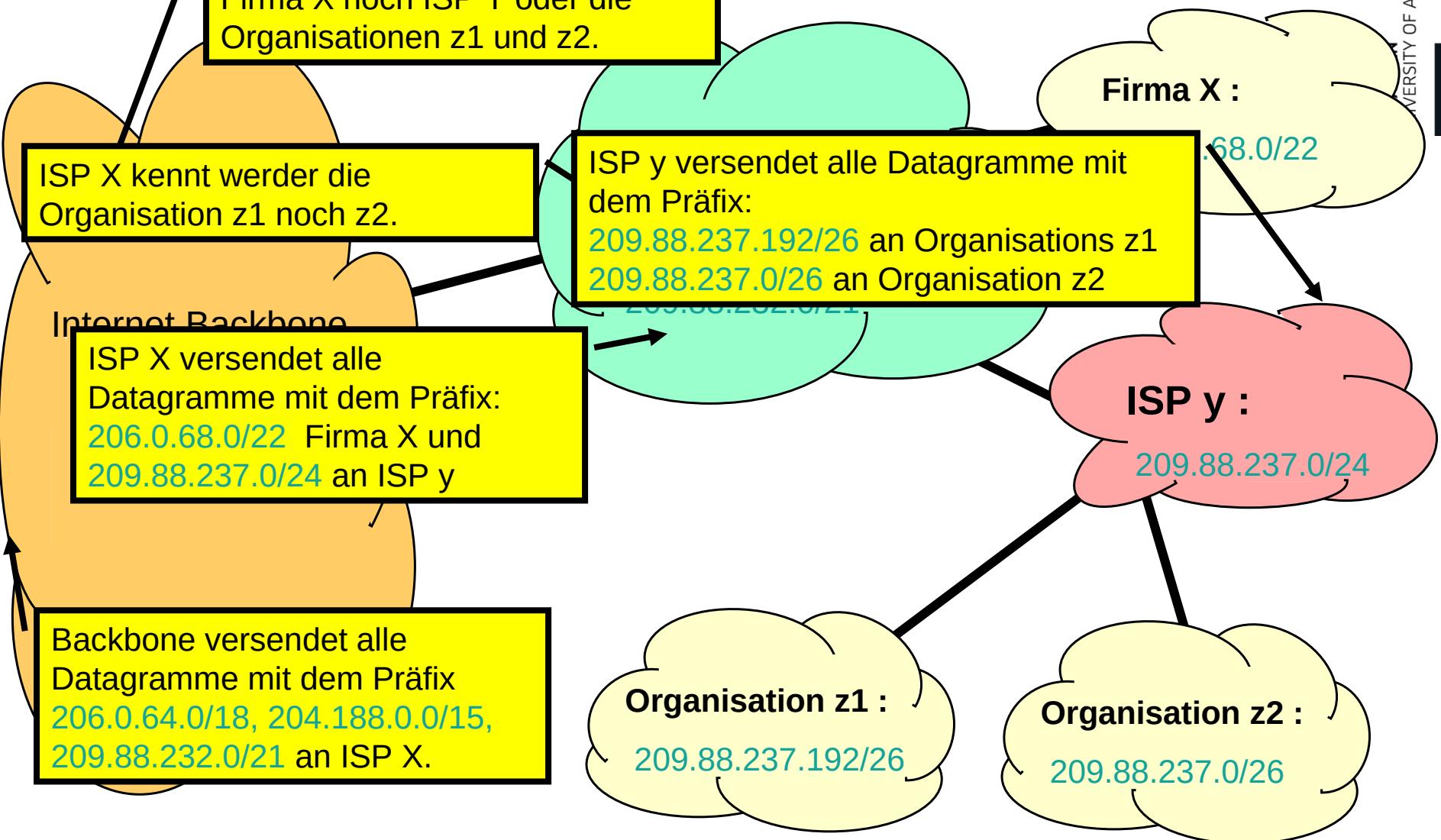
Es wird der Weg ermittelt, welcher am genauesten spezifiziert wurd (most specific)

CIDR hilft dabei, die für das Routing notwendige Anzahl der bekannten Netze durch „verstecken“ zu reduzieren

CIDR und Routing



CIDR und Routing



Verkleinerung der Routing-Tabelle durch CIDR

- Der Longest Prefix Match Algorithmus erlaubt das Zusammenfassen von Routen
 - Es reicht, wenn die genaue Adresse erst nahe am Ziel bekannt ist
 - Signifikante Beitrag zur Reduktion der Größe der Routing-Tabellen im Internet

Destination	Next Hop
10.1.0.0/24	R3
10.1.2.0/24	direct
10.2.1.0/24	direct
10.3.1.0/24	R3
20.2.0.0/16	R2
20.1.1.0/28	R2

Destination	Next Hop
10.1.0.0/24	R3
10.1.2.0/24	direct
10.2.1.0/24	direct
10.3.1.0/24	R3
20.0.0.0/14	R2

Vorsicht: Durch das Zusammenfassen von Routing-Einträgen dürfen keine fehlerhaften Regeln entstehen

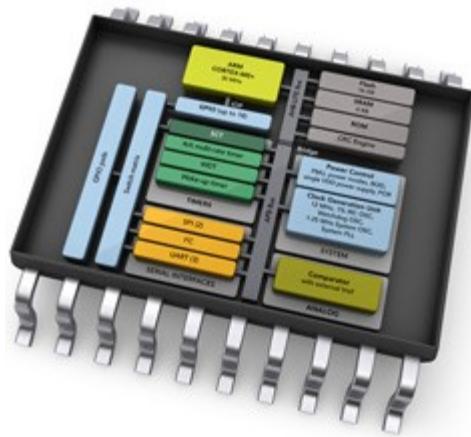
1. Ein Zusammenfassen ist nur möglich, wenn **gleiche Ziele** (Next Hop) verwendet werden
2. Die relaxierte Interpretation der Netzwerkadresse kann ggf. andere, nicht gewollte Einträge umfassen. Hier hilft aber ggf. die Longest Match Regel. Hierzu muss aber der Präfix der überschriebenen Regel länger sein.
3. 0.0.0.0/0 ist ein Default-Route

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

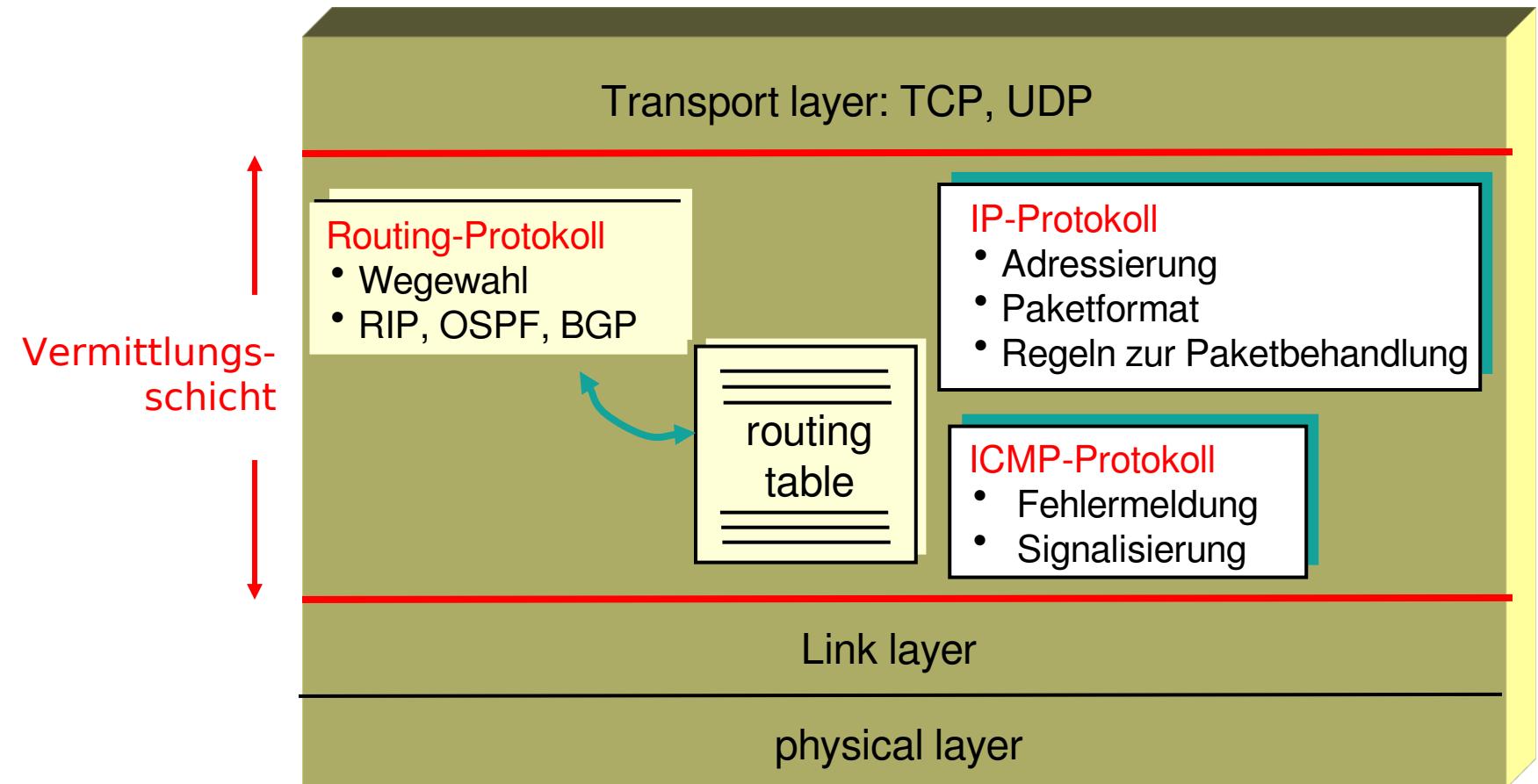
Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Die Vermittlungsschicht im Internet



Routing im Internet

„Das Weiterleiten (Routing) erfüllt die wichtige Aufgabe, einzelne Teilstrecken des Kommunikationsnetzes so zu verbinden, dass eine Ende-zu-Ende-Kommunikation zwischen den angeschlossenen Teilnehmern möglich wird.“

Das Routing kann in zwei Teilprozesse zerlegt werden:

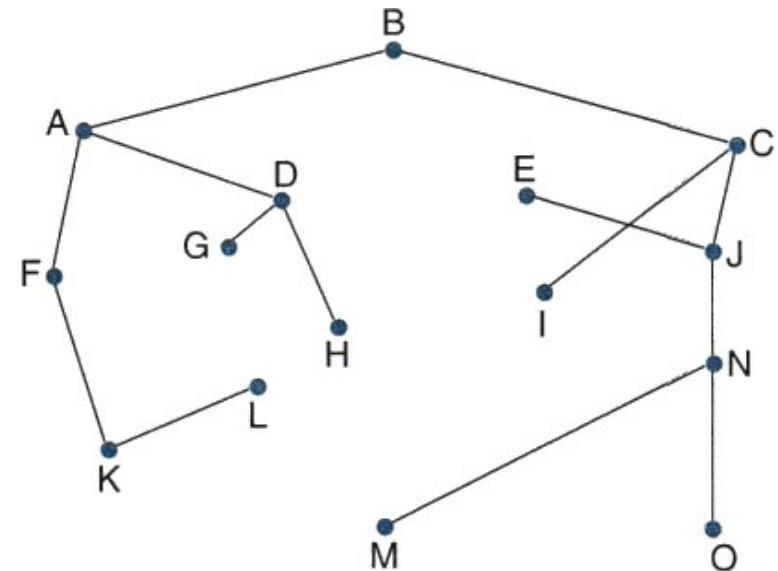
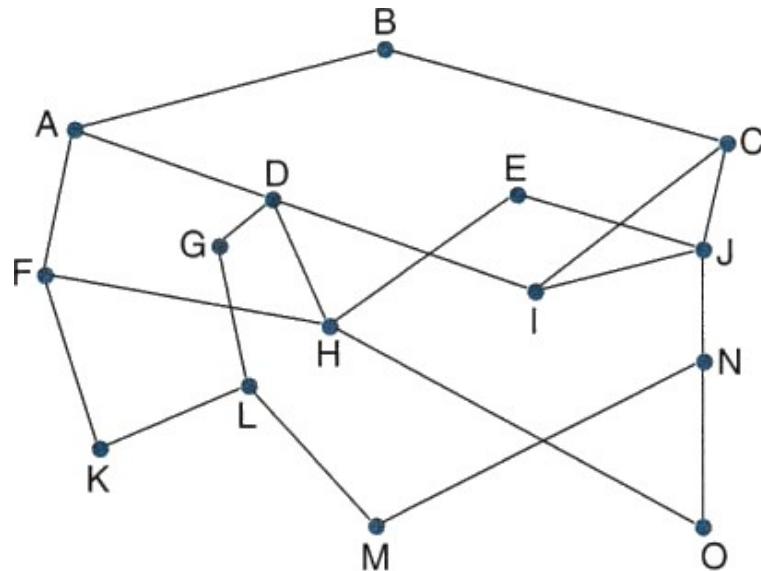
- Das eigentliche **Routing** (die Control-Plane) ist verantwortlich für Wegewahl, mit der die Ende-zu-Ende-Kommunikation erreicht wird
- Das **Weiterleiten** (engl. *forwarding*) der Pakete (die Data-Plane) gemäß den Vorgaben der Control-Plane

Die Forwarding-Entscheidung wird für jedes Paket durchgeführt

- Unabhängig von vorherigen Entscheidungen
- Entscheidung auf Basis der Zieladresse

Das Optimalitätsprinzip

- Das Internet ist im weitesten Sinne ein Graph
Knoten: Netzwerk-Elemente (Rechner/Router etc.)
Kanten: Existierende physikalische Verbindungen
- Gesucht wird der (eindeutige) **Quelle-Senke-Baum** mit den ‚kürzesten‘ Verbindungen von beliebigen Quellen (Blätter) zum Ziel (Wurzel)



Routing - Kostenfunktion eines Weges

Was bedeutet eigentlich ‚kürzester Weg‘?

- Geringste Anzahl ‚hops‘ ?
- Schnellste Verbindung ?
- Physisch kürzeste Verbindung?
- Maximierung des Datendurchsatzes ?

Es gibt noch weitere Qualitätsmaße für ein ‚gutes‘ Routing, die sich ggf. widersprechen:

- korrekt, einfach, robust, stabil, fair, effizient

Für die weiteren Folien nehmen wir ein quantifizierbares **Gewicht** für jede Kante an, das sich i.d.R. nicht kurzfristig ändert.

Lösung des Optimierungsproblems: Dijkstra Algorithmus

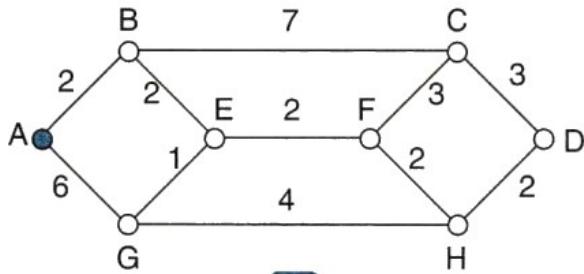
Ansatz:

- Ausgehend von der Quelle wird mit jedem Schritt ein neuer Knoten hinzugenommen
- Die Auswahl des Knotens geschieht so, dass der Knoten selektiert wird, der mit die günstigste Anbindung erlaubt
- Damit teilt sich der Graph in zwei Teile auf:
 - Eine Menge N' von Knoten für die der günstigste Weg schon ermittelt wurde
 - Eine Menge von Knoten, für die der Weg noch zu ermitteln ist
- Ordnung innerhalb der direkten Verbindungen zwischen diesen zwei Mengen bringt die Lösung
 - Auswahl des kleinsten Wertes
 - Es kann keinen günstigeren Weg geben

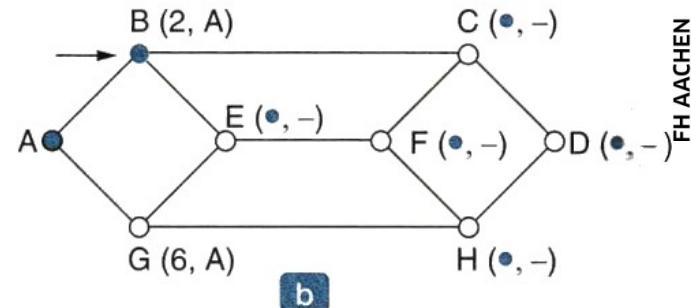
Lösung des Optimierungsproblems: Dijkstra Algorithmus (‘Shortest Path’)

Der ‚kürzeste‘ Weg von A nach D soll berechnet werden.

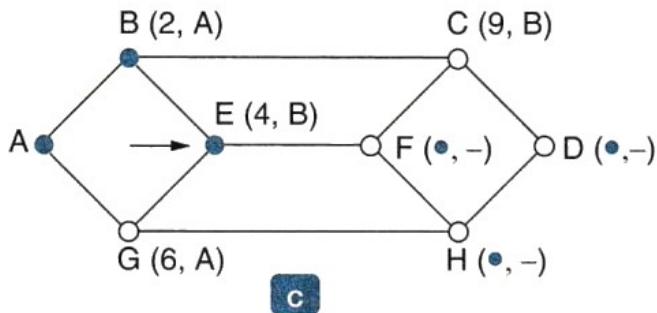
Hier: Die ersten 6 Iterationen



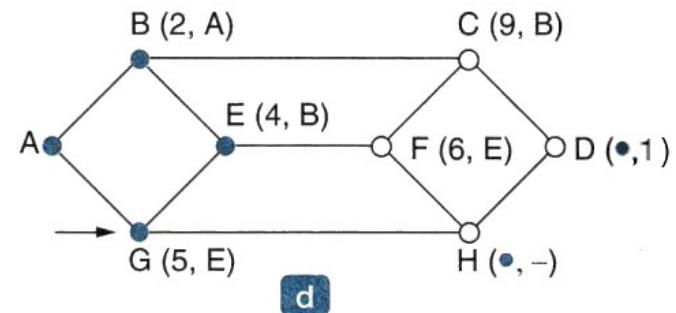
a



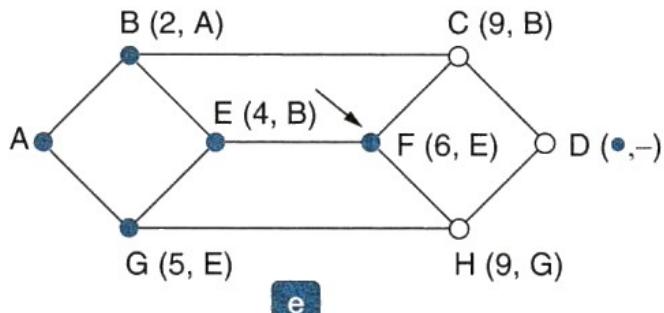
b



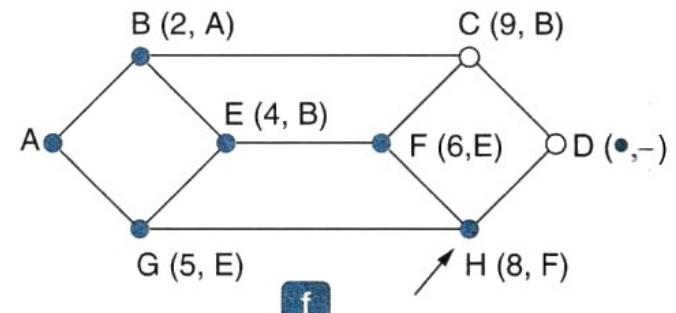
c



d



e



f

Routing im realen Kontext

Verschicken von Paketen

- Jeder Rechner und jeder Router hat eigene Weiterleitungstabelle
 - > Wird bei der Versendung jedes einzelnen Paketes konsultiert
 - > Zieladresse im IP-Paket bestimmt, welcher Weg gewählt wird
- Kann statisch konfiguriert werden (z.B. im eigenen lokalen Netz via DHCP oder mittels Kommandos)
- Router im Backbone sollen auf Änderungen im Netz reagieren können
 - > Statische Konfiguration unmöglich
 - > **Wie kommen die Router dynamisch zu ihren Tabellen?**

→ Routing-Protokolle

- Router unterhalten sich und tauschen Informationen über Verbindungen im Netz aus!

Routing im realen Kontext

Zwei große Klassen von Routing-Verfahren:

- ohne Kenntnis der gesamten Netztopologie
→ **Distanzvektoralgorithmen**
- mit Kenntnis der gesamten (zumindest im AS) vorhandenen Netztopologie
→ **Link-State Routing**

- Innerhalb der AS werden heute Routing-Protokolle eingesetzt, die Link-State-Routing realisieren.
- Bei den Inter-AS Protokollen kommen auch noch Distanzvektoralgorithmen zum Einsatz (Verbergen der Struktur des AS).

Distanzvektoralgorithmen

Grundidee:

- Jeder Router schickt regelmäßig (z.B. alle 30s) seinen Nachbarn einen **Distanzvektor**, der die (geschätzten) Kosten zu allen anderen Knoten im Netz enthält.
- Jeder Router aktualisiert daraufhin seine eigene Liste mit den Kosten zu den anderen Knoten und wählt dabei den Weg mit den geringsten Kosten

→ **Bellman-Ford-Algorithmus**

- Grundlage des Routing im ARPANET und im Internet im Protokol RIP (Routing Information Protocol) verwendet

Bellman-Ford Algorithmus

Definiere

$c_x(v) :=$ Kosten zum Nachbarn v von x aus

$d_x(y) :=$ Kosten des günstigsten Weges von x nach y

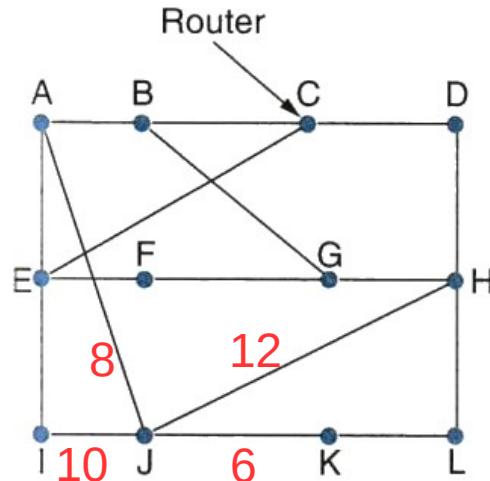
Gibt es einen Weg zwischen x und y und sind x und y nicht direkt verbunden, dann muss es einen Nachbarn v von x geben für den gilt:

$$d_x(y) = \min \{c_x(v) + d_v(y)\}$$

hierbei wird das Minimum über alle Nachbarn von x genommen.

Die Kosten der direkten Verbindung wird als bekannt angenommen

Distanzvektoralgorithmen



- Betrachtet wird Router J.
- Er kennt die Kosten (z.B. ms) zu den Routern A, I, H und K
- Er erhält Übertragungsvektoren von A, I, H und K

neu geschätzte Verzögerung von J ↓ Leitung

	A	I	H	K	
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 -
K	24	22	22	0	6 K
L	29	33	9	9	15 K

JA-Verzögerung ist 8 JI-Verzögerung ist 10 JH-Verzögerung ist 12 JK-Verzögerung ist 6

von den vier Nachbarn von J erhaltene Vektoren

neue Routing-Tabelle für J

Distanzvektoralgorithmen

Vor- und Nachteile des Distanzvektoralgorithmus

Vorteile:

- Leicht zu implementieren, einfache Berechnung
- Neue, bessere Routen werden im Netz schnell propagiert

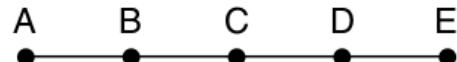
Nachteile:

- Der Ausfall von Routen/Routern führt zum „*Count-to-Infinity*“ Problem:

A - **B** - C

Betrachte B: Router A fällt aus, bekommt aber von C gesagt, dass er einen Weg nach A kennt (der allerdings über B führt, was B nicht weiß...) → Die Kosten zum Weg nach A schaukeln sich langsam auf...

Distanzvektoralgorithmen: Count to Infinitiy



					Initially
1	•	•	•	•	After 1 exchange
1	2	•	•	•	After 2 exchanges
1	2	3	•	•	After 3 exchanges
1	2	3	4	•	After 4 exchanges



A	B	C	D	E	Initially
1	2	3	4	•	After 1 exchange
3	2	3	4	•	After 2 exchanges
3	4	3	4	•	After 3 exchanges
5	4	5	4	•	After 4 exchanges
5	6	5	6	•	After 5 exchanges
7	6	7	6	•	After 6 exchanges
7	8	7	8	•	
⋮	⋮	⋮	⋮	⋮	
•	•	•	•	•	

Router A ist initial ‚down‘,
und dann ‚up‘

→ Neue Routen werden schnell
gelernt und propagiert.

Router A ist initial ‚up‘,
und dann ‚down‘

→ Count to Infinity-Problem

Link-State Routing

Grundidee:

- Jeder Router führt die folgenden Schritte durch:
 - 1) Die Nachbarn und deren Netzadressen ermitteln
 - 2) Die Kosten zu jedem seiner Nachbarn festlegen
 - 3) Ein Paket zusammenstellen, in dem alles steht was bisher gelernt wurde
 - 4) Dieses Paket an alle anderen Router senden und von allen anderen Routern derartige Pakete empfangen
 - 5) Den kürzesten Pfad zu allen anderen Routern berechnen
- Dadurch wird die gesamte Topologie in jedem Router abgebildet. Anwendung des Dijkstra-Algorithmus möglich!

Link-State Routing

1) Die Nachbarn und deren Netzadressen ermitteln

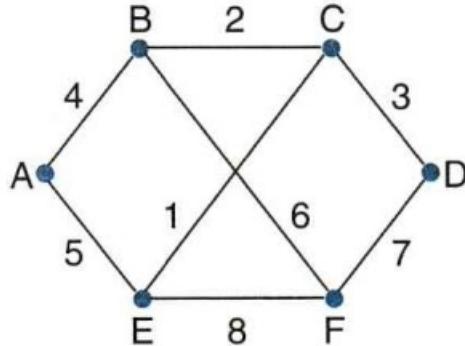
- Versenden von HELLO-Paketen

2) Die Kosten zu jedem seiner Nachbarn festlegen

- Z.B. über die reziproke Bandbreite der Verbindung
- Z.B. über spezielle ECHO-Pakete, die die Übertragungsdauer messen

Link-State Routing

3) Ein Paket zusammenstellen, in dem alles steht was bisher gelernt wurde



Link-State-Pakete					
A	B	C	D	E	F
Seq.-Nr.	Seq.-Nr.	Seq.-Nr.	Seq.-Nr.	Seq.-Nr.	Seq.-Nr.
Alter	Alter	Alter	Alter	Alter	Alter
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

4) Dieses Paket an alle anderen Router senden und von allen anderen Routern derartige Pakete empfangen

- regelmäßig oder bei ‚Ereignissen‘
- Realisierung über Fluten (Broadcast)
- Versionierung durch Sequenznummern
- Altern der Pakete → keine unbegrenzte Lebensdauer

Link-State Routing

5) Den kürzesten Pfad zu allen anderen Routern berechnene

- z.B. über Dijkstra-Algorithmus

Vorteile:

- Optimale Lösung des Routings kann berechnet werden
- Gute Performance bei Änderungen des Netzwerkes

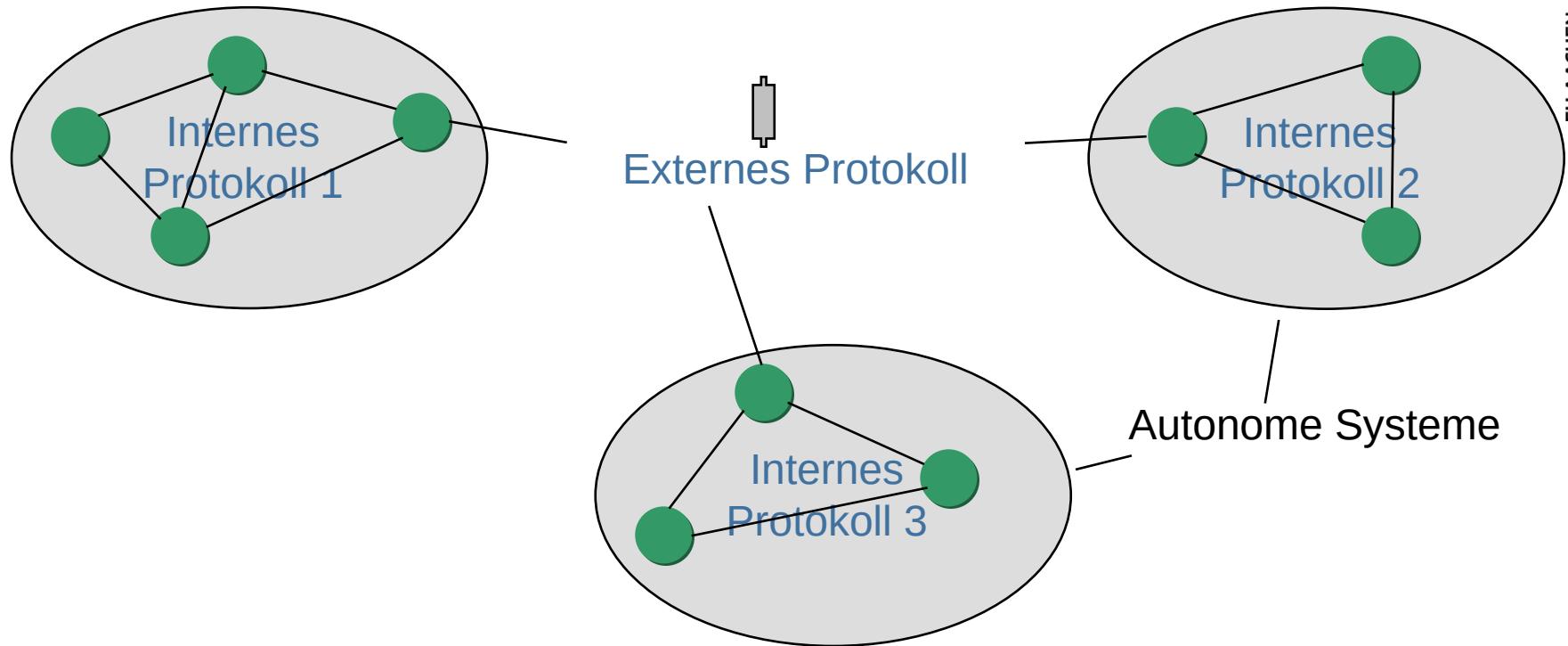
Nachteile:

- Höherer Rechenaufwand im Router
- Höhere Komplexität bei der Versendung der Link-State Pakete an alle andern Router und der Analyse der empfangenen Pakete
- Höhere Netzlast durch Routing Protokoll

Routing im Internet

- Routing im Internet muss höchst autonom und extrem dynamisch realisiert werden
- Das kann nur durch die bisher skizzierten Routing-Protokolle realisiert werden: Router unterhalten sich und tauschen Informationen aus
- Abgestuftes System:
 - Innerhalb eines Autonomen Systems kann man die gesamte Topologie wissen
 - Zwischen Autonomen Systemen sollte dies nicht Voraussetzung sein

Routing im Internet - Regionen



- Interne Protokolle – Intra-AS (OSPF, RIP, IS-IS)
- Externe Protokolle – Inter-AS (BGP)

Übersicht Routing-Protokolle

Internet Routing-Protokolle:

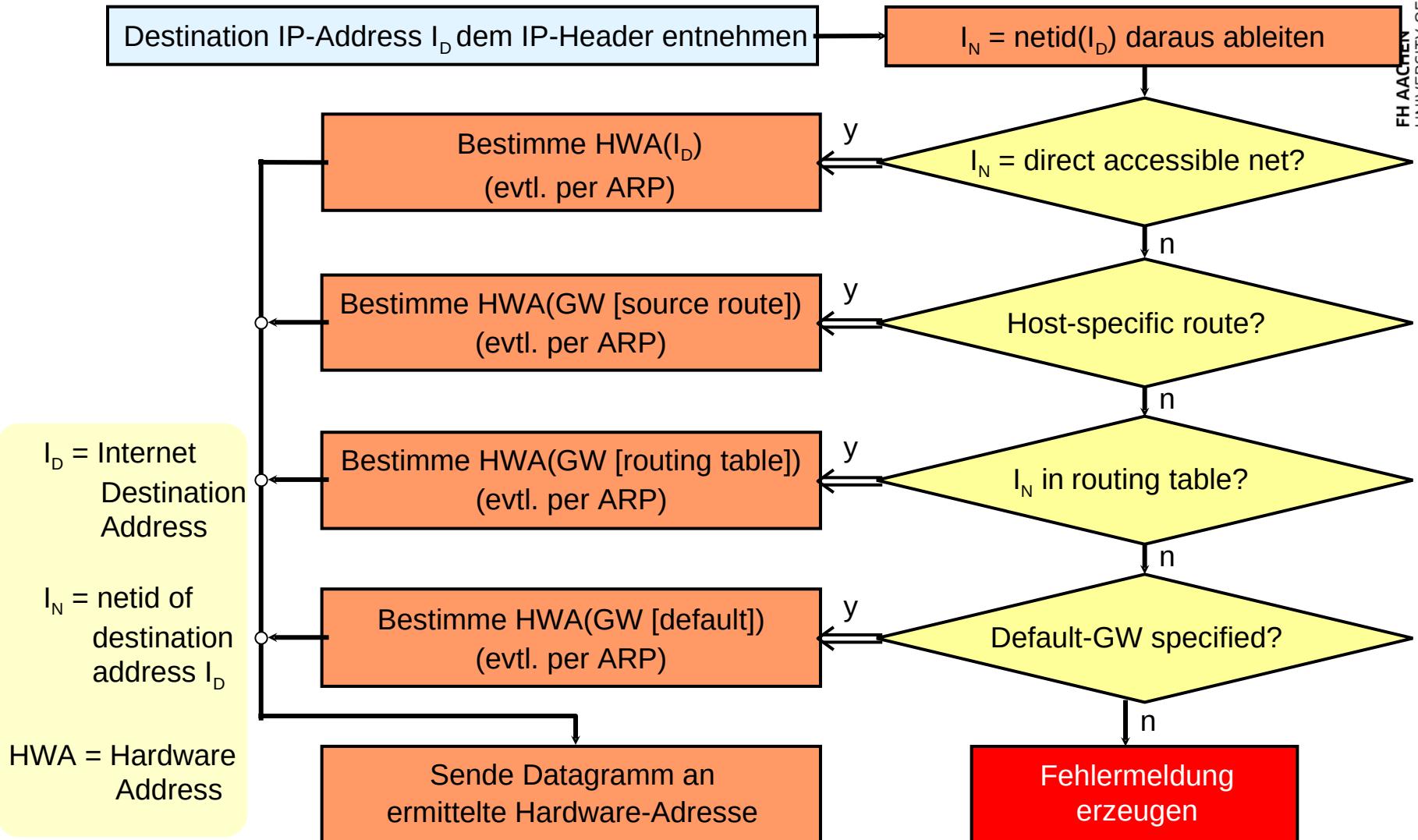
- **RIP, RIP2:** Intra-AS, Distance-Vector, veraltet
- **IS-IS:** Intra-AS, ursprünglich für DECnet, danach ISO Standard und Grundlage für OSPF, (Shortest Path First)
- **OSPF:** Intra-AS, Link-State, Unicast (Shortest Path First)
- **BGP:** Border Gateway Protokol Inter-AS, Distance-Vector

Routing: “lokal” vs. “global”

Jeder Rechner braucht eine Routing-Tabelle

- **Router im Internet**
 - > Meist Link-State-Verfahren (Dijkstra) wie OSPF **innerhalb** autonomer Systeme (Backbones)
 - > Distance-Vector-Verfahren (Bellmann-Ford) zur Kopplung autonomer Systeme
- **Router in Firmen / Stadtnetzen**
 - > Können Link-State- oder Distance-Vector-Verfahren nutzen
 - > Meist sinnvoller: statische Konfiguration
 - o Wenn es sowieso nur einen Pfad in den Backbone gibt, braucht man auch keine Informationen über mögliche Pfade auszutauschen
 - > Endrechner
 - o Statische Konfiguration oder per DHCP bezogene Informationen

Die Forwarding-Entscheidung im Detail

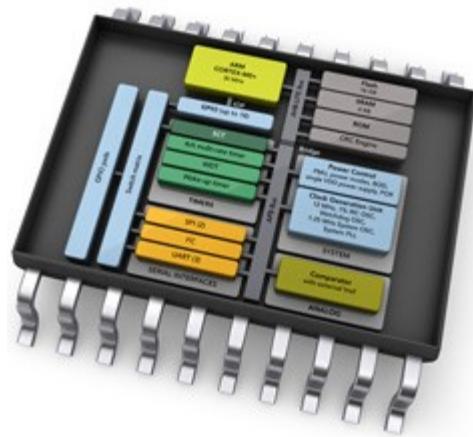


FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

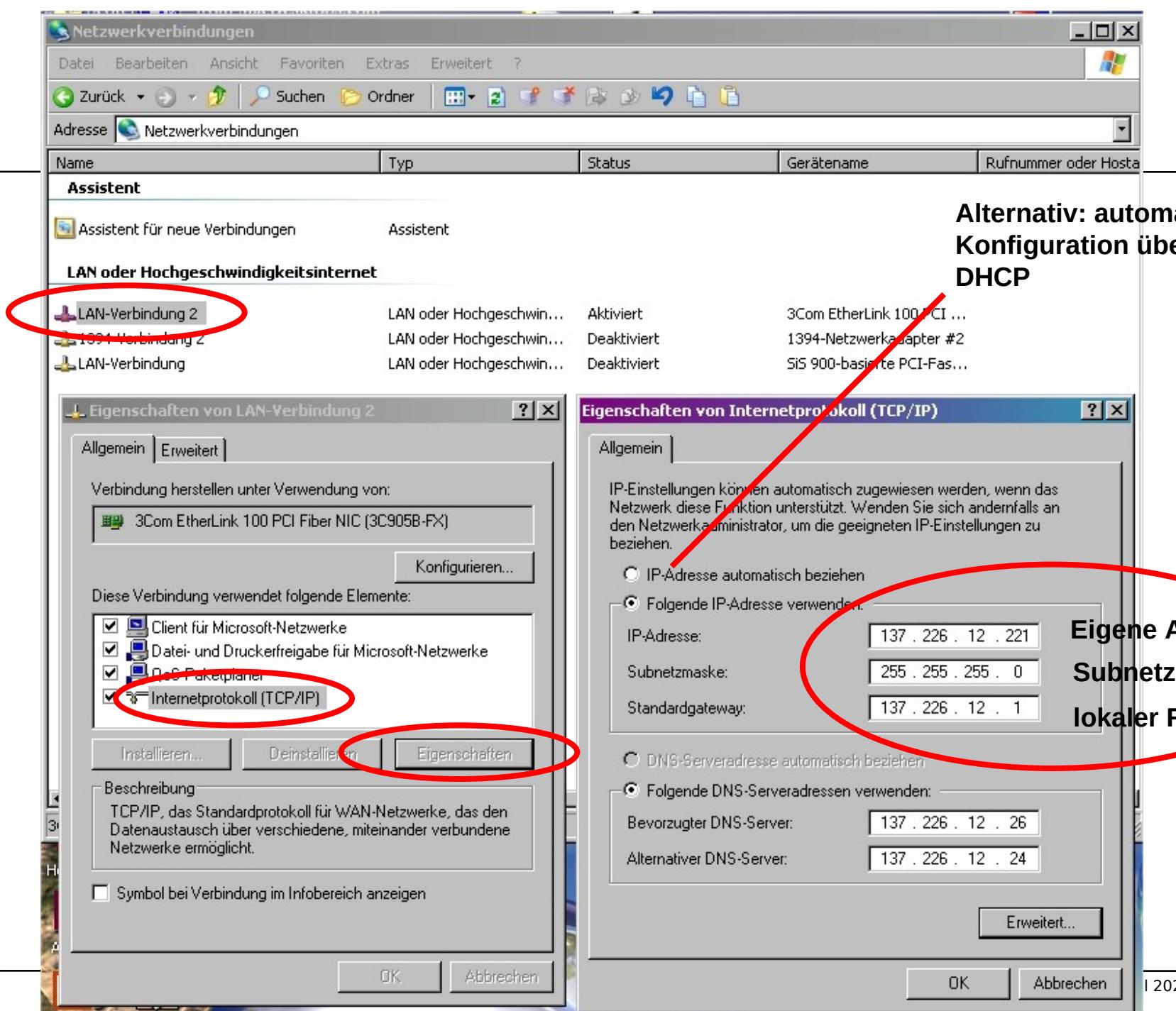
(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Nachtrag zu IP Adressen: Konfiguration

- IP-Adressen müssen in jedem Endgerät (Rechner) und auch in Routern und weiteren Netzkomponenten konfiguriert werden.
- Ein Knoten muss folgende Daten besitzen:
 - IP und Netzmaske
 - Standard-Gateway
 - DNS-Server
- Grundsätzlich existieren 2 Möglichkeiten:
 - Manuelle Konfiguration
 - Konfiguration über ein Protokoll (RARP, BOOTP, DHCP)



Beispiel: Netzwerk-Konfiguration

Routing-Tabelle:

```
andreas@notebook-fh:~$ netstat -r
Kernel-IP-Routentabelle
Ziel      Router      Genmask      Flags   MSS Fenster irtt Iface
default   fritz.box  0.0.0.0      UG        0 0          0 enx482ae3a901b8
default   fritz.box  0.0.0.0      UG        0 0          0 wlp0s20f3
link-local 0.0.0.0    255.255.0.0  U         0 0          0 enx482ae3a901b8
192.168.178.0 0.0.0.0    255.255.255.0 U         0 0          0 enx482ae3a901b8
192.168.178.0 0.0.0.0    255.255.255.0 U         0 0          0 wlp0s20f3
andreas@notebook-fh:~$
```

Netzwerk-Interfaces:

```
enx482ae3a901b8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.178.34 netmask 255.255.255.0 broadcast 192.168.178.255
    inet6 2001:16b8:fad:6700:c55d:1dba:e09a:9b26 prefixlen 64 scopeid 0x0<global>
    inet6 fe80::5e92:f028:42e5:6a28 prefixlen 64 scopeid 0x20<link>
    inet6 2001:16b8:fad:6700:266c:e42f:be8:96e2 prefixlen 64 scopeid 0x0<global>
    ether 48:2a:e3:a9:01:b8 txqueuelen 1000 (Ethernet)
    RX packets 470938 bytes 467529854 (467.5 MB)
    RX errors 0 dropped 22921 overruns 0 frame 0
    TX packets 293772 bytes 31287369 (31.2 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
wlp0s20f3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.178.33 netmask 255.255.255.0 broadcast 192.168.178.255
    inet6 2001:16b8:fad:6700:5db9:dd93:9199:b384 prefixlen 64 scopeid 0x0<global>
    inet6 2001:16b8:fad:6700:244d:c84:b68d:718f prefixlen 64 scopeid 0x0<global>
    inet6 fe80::df01:cf99:4e73:c7f2 prefixlen 64 scopeid 0x20<link>
    ether b4:0e:de:52:74:6b txqueuelen 1000 (Ethernet)
    RX packets 26325 bytes 1818515 (1.8 MB)
    RX errors 0 dropped 22503 overruns 0 frame 0
    TX packets 2805 bytes 368587 (368.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Das Interface hat
eine **private IP** !

IPv4 und Adressknappheit

- IPv4 hat ‚relativ‘ wenig IP Adressen (2^{32})
- Verschwenderischer Umgang mit Class-A Netzen in der Vergangenheit
- Man ging davon aus, dass bereits vor Jahren (ca. 2012) die letzte freie IP-Adresse (bzw. das letzte frei Subnetz) vergeben wäre.
- IPv6 (128 Bit Adressen) ist schon seit Ende der 90-er Jahre spezifiziert, setzt sich aber nur sehr langsam durch.
- Einer der Gründe dafür ist eine technische ‚Krücke‘, die die Adressknappheit bis heute aufgehalten hat:
NAT (Network Address Translation)

IPv4 und Adressknappheit

Idee:

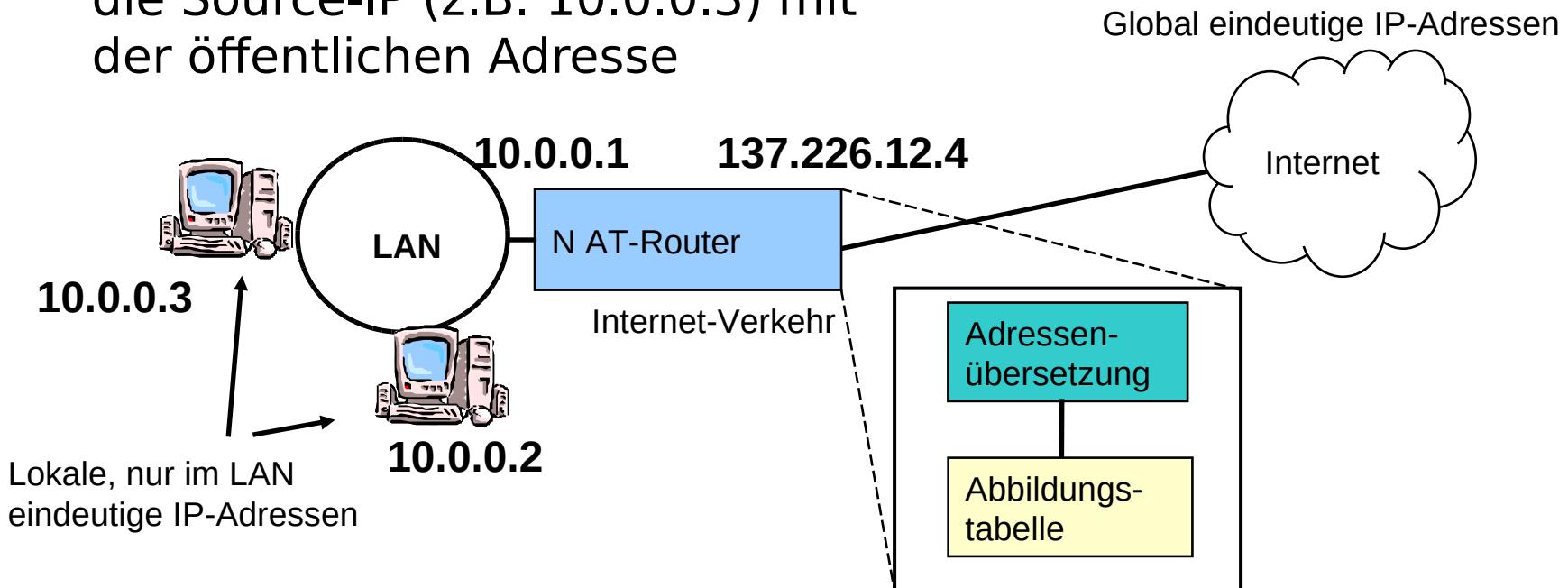
- Jeder Endkunde (jede kleinere Firma etc.) bekommt lediglich nur eine eindeutige öffentliche IPv4 Adresse vom ISP, d.h. es existiert auch nur ein Zugangspunkt (Gateway) zu diesem Netz
- Innerhalb des Hauses/der Firma wird ein ‚privates‘ Netz betrieben, was die spezifizierten privaten IP-Adressblöcke benutzt:
 - 10.0.0.0 – 10.255.255.255
 - 172.16.0.0 – 172.31.255.255
 - 192.168.0.0 – 192.168.255.255
- Problem: Die privaten IPs dürfen nicht ins öffentliche Internet gelangen, bzw. sind im öffentlichen Internet nicht eindeutig. Wie ist nun eine Kommunikation ‚nach außen‘ möglich?

Network Address Translation (NAT)

Senderichtung:

- Ein Rechner im lokalen Netz möchte ein IP Paket zu einem externen Knoten senden
- Lösung: Der NAT-Router ersetzt die Source-IP (z.B. 10.0.0.3) mit der öffentlichen Adresse

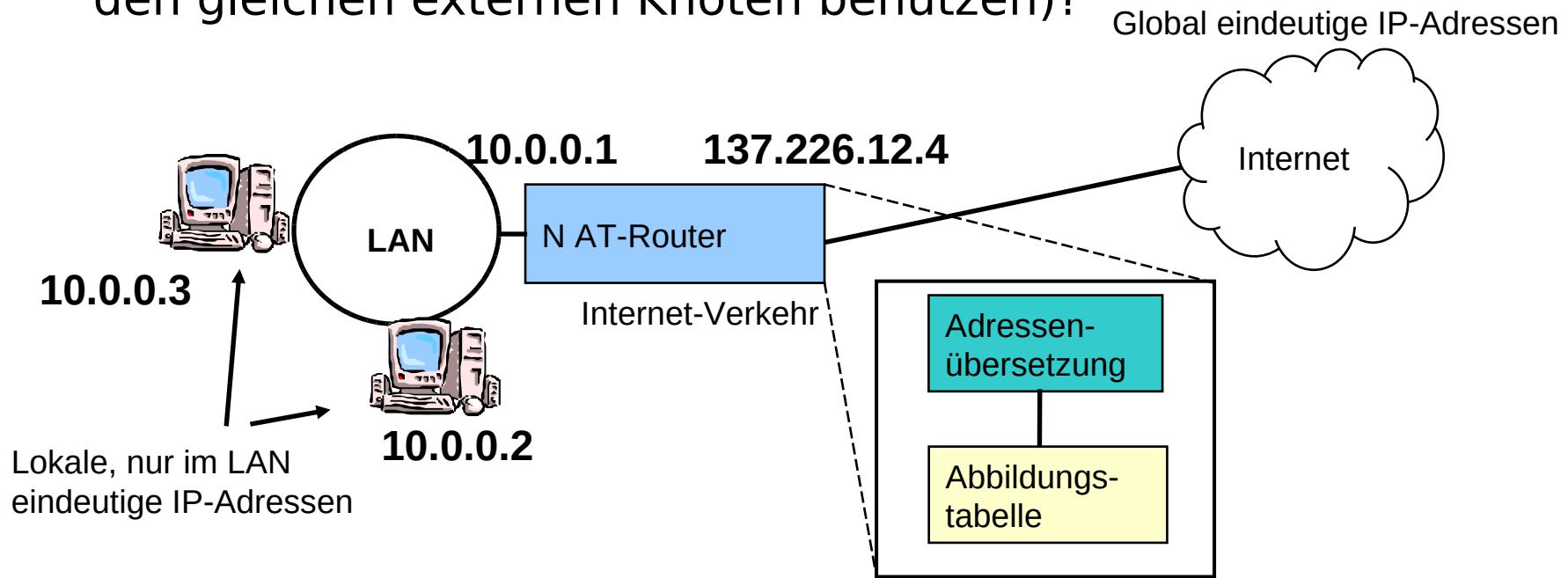
Version	IHL	Type of Service	Total Length			
Identification			Flags	Fragment Offset		
Time to Live	Protocol		Header Checksum			
Source Address						
Destination Address						
Options			Padding			



Network Address Translation (NAT)

Empfangsrichtung:

- Die Antwort des externen Knotens wird an 137.226.12.4 zurück geschickt.
- Woher weiß der NAT-Router, von welchem seiner ‚internen‘ Rechner das Paket kam (mehrere interne Rechner können den gleichen externen Knoten benutzen)?



Network Address Translation (NAT)

- Erste Idee: Suche im IP-Header nach freien Feldern, die zur Identifikation des Absenders genutzt werden können. → Geht nicht, nur noch 1 Bit frei...
- Zweite Idee: Gehe davon aus dass die Kommunikation über TCP bzw UDP erfolgt. Nutze die Portnummern der Transportschicht zur Identifikation des Absenders
- Verletzung der Schichten-Architektur: Layer 3 kennt Details des Layer 4, was eigentlich nicht sein darf.

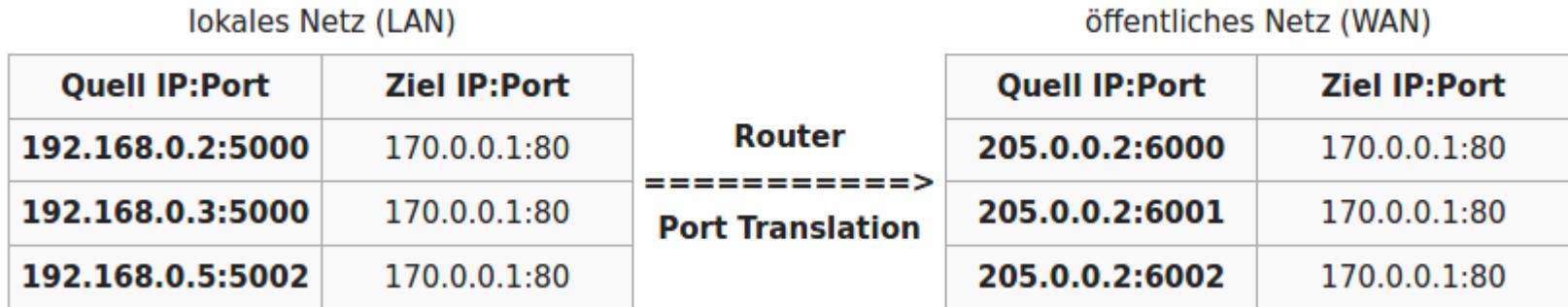
TCP Segment Header Format										
Bit #	0	7	8	15	16	23	24			
0	Source Port				Destination Port					
32	Sequence Number									
64	Acknowledgment Number									
96	Data Offset	Res	Flags		Window Size					
128	Header and Data Checksum				Urgent Pointer					
160...	Options									

UDP Datagram Header Format								
Bit #	0	7	8	15	16	23	24	
0	Source Port				Destination Port			
32	Length				Header and Data Checksum			

Network Adress Translation (NAT)

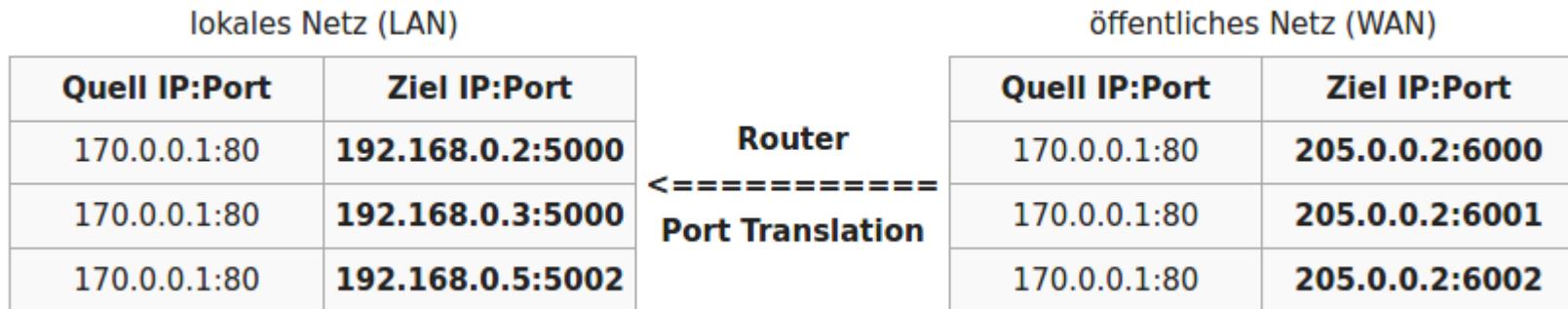
- Bei TCP und UDP wird sowohl das Destination Port als auch das Source Port mitgeschickt.
- Das **Destination Port** darf nicht verändert werden, da es definiert auf welchen Service zugegriffen werden soll.
- Das **Source Port** wird vom Betriebssystem vergeben, ist aber ggf. nicht eindeutig (2 Rechner können zufällig das selbe Source Port nutzen).
- Daher: NAT-Router ersetzt das Source Port durch eine ID, die den privaten Rechner bzw. die Verbindung identifiziert.
- Im Antwortpaket muss das Destination Port extrahiert werden, und der NAT Router ersetzt die Ziel-IP mit der entsprechenden privaten IP und das Zielpot mit dem ursprünglichen Port

Network Address Translation (NAT)



192.168.0.2:5000	↔	6000
192.168.0.3:5000	↔	6001
192.168.0.5:5002	↔	6002

Übersetzungstabelle
im NAT Router



Network Adress Translation (NAT)

Nachteile:

- Nicht jeder Rechner ist eindeutig per IP identifizierbar (Nachverfolgbarkeit...)
- Verletzung des Schichtenmodells
- Übersetzungstabelle kann i.d.R. nur aufgebaut werden, wenn interner Knoten die Kommunikation initiiert.
- Was ist, wenn nicht TCP/UPD verwendet wird?
- Was passiert, wenn übergeordnete Protokolle die IP als Payload übertragen (z.B. ftp)?
- Was passiert bei verschlüsselten Verbindungen?

- Einige Nachteile sind durch technische Lösungen behoben

Network Adress Translation (NAT)

NAT ist auch bekannt als

- Network Address Port Translation (NAPT)
- Hiding NAT, NAT-Overloading
- Masquerading

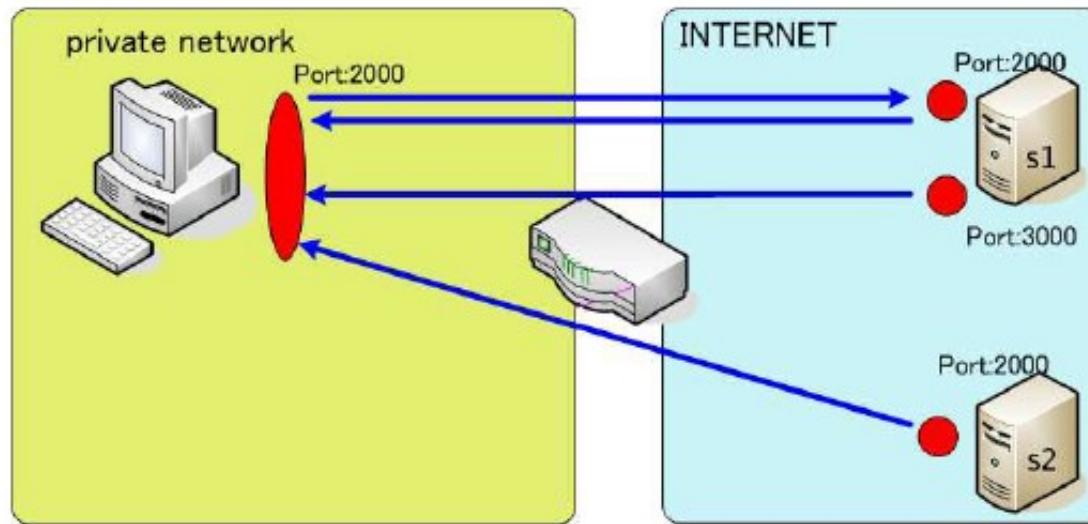
NAT kennt unterschiedliche Sicherheitsstufen

- Ein Port wird unbegrenzt nach außen freigegeben (dadurch kann auch von außen ein Rechner die Kommunikation initiieren)
- Ein Port wird nur nach dem Start der internen Kommunikation freigegeben (Quell-IP in den Rückpaketen wird überprüft).
- Die Rückpakte werden zusätzlich auf das Source Port hin überprüft

Network Address Translation (NAT)

Full Cone NAT

Sobald die Kombination einer interner IP, interner Port zu einer extern erreichbaren Adresse, Port abgebildet wurden findet die Abbildung automatisch statt. Alle externen Rechner können dann über das extern erreichbare Tupel mit dem internen Rechner kommunizieren. Der Router merkt sich Nichts über die jeweiligen Ziel-Rechner.

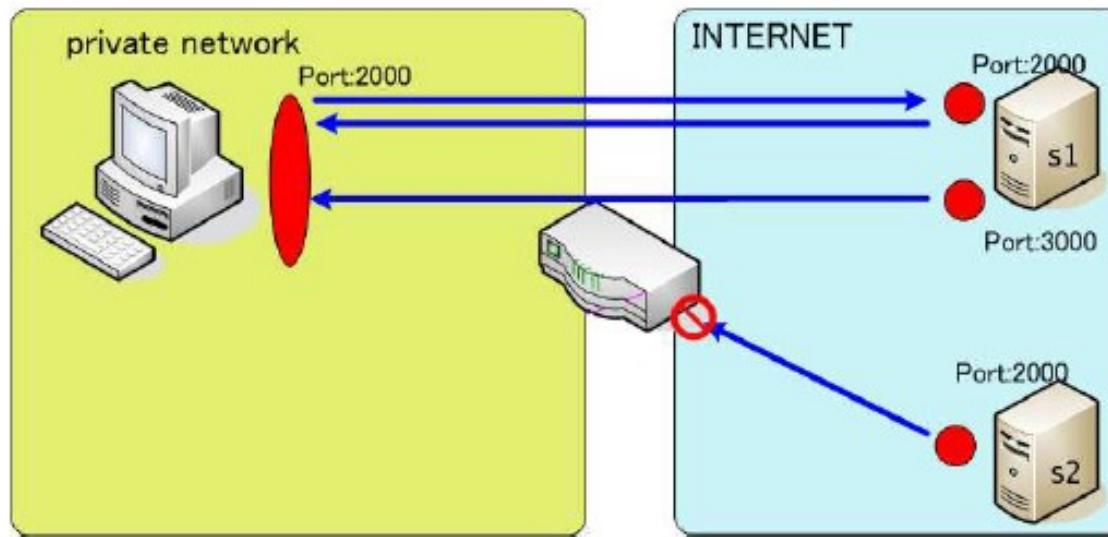


Aus: A New Method for Symmetric NAT Traversal in UDP and TCP
[Daisuke Yamada, Suguru Yoshida, Shigeki Goto](#)

Network Address Translation (NAT)

Restricted Cone NAT

Dieser Fall ist ähnlich dem Full Cone Fall, allerdings muss hier die Kommunikation vom internen Rechner vorab einmal durchgeführt werden. Damit merkt sich das NAT die **Ziel-IP**. Somit kann nur der externe Rechner mit dem internen Rechner kommunizieren, der vorher adressiert wurde

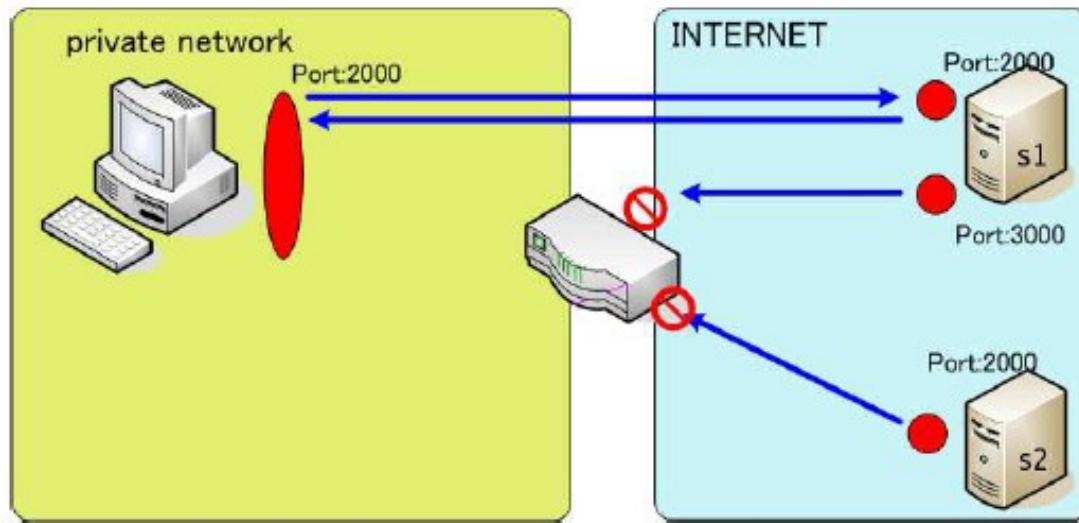


Aus: A New Method for Symmetric NAT Traversal in UDP and TCP
Daisuke Yamada, Suguru Yoshida, Shigeki Goto

Network Address Translation (NAT)

Port Restricted Cone NAT

Dieser Fall erweitert den restricted cone NAT in der Art, dass auch nur der entfernte Port mit dem internen Rechner kommunizieren kann. Der Router merkt sich **Ziel-IP** und **Ziel-Port**.

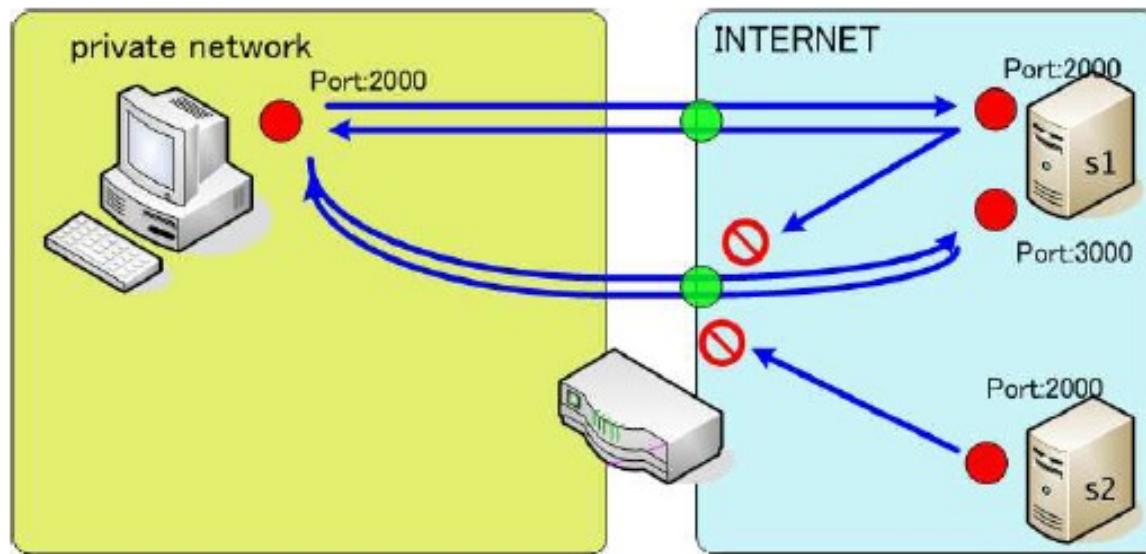


Aus: A New Method for Symmetric NAT Traversal in UDP and TCP
Daisuke Yamada, Suguru Yoshida, Shigeki Goto

Network Address Translation (NAT)

Symmetric NAT

Hier wird für jeden Datenstrom aus dem internen Netz eine eigene Abbildung durchgeführt. Wenn der Quell-Rechner z.B. vom gleichen Quell-Port aus mit zwei unterschiedlichen Ziel-Rechner kommuniziert, wird für jede Verbindung eine eigene NAT-Tabelle angelegt. Nur der jeweilige Zielrechner darf jeweils mit der eingerichteten Portnummer antworten.



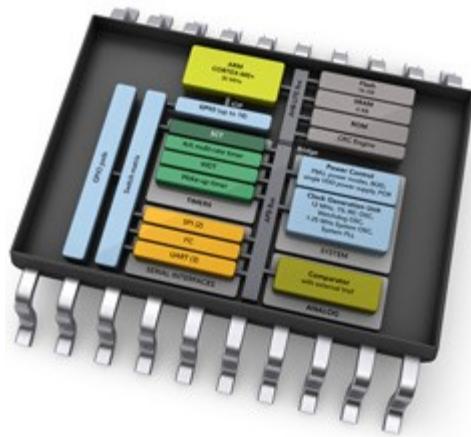
Aus: A New Method for Symmetric NAT Traversal in UDP and TCP
Daisuke Yamada, Suguru Yoshida, Shigeki Goto

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

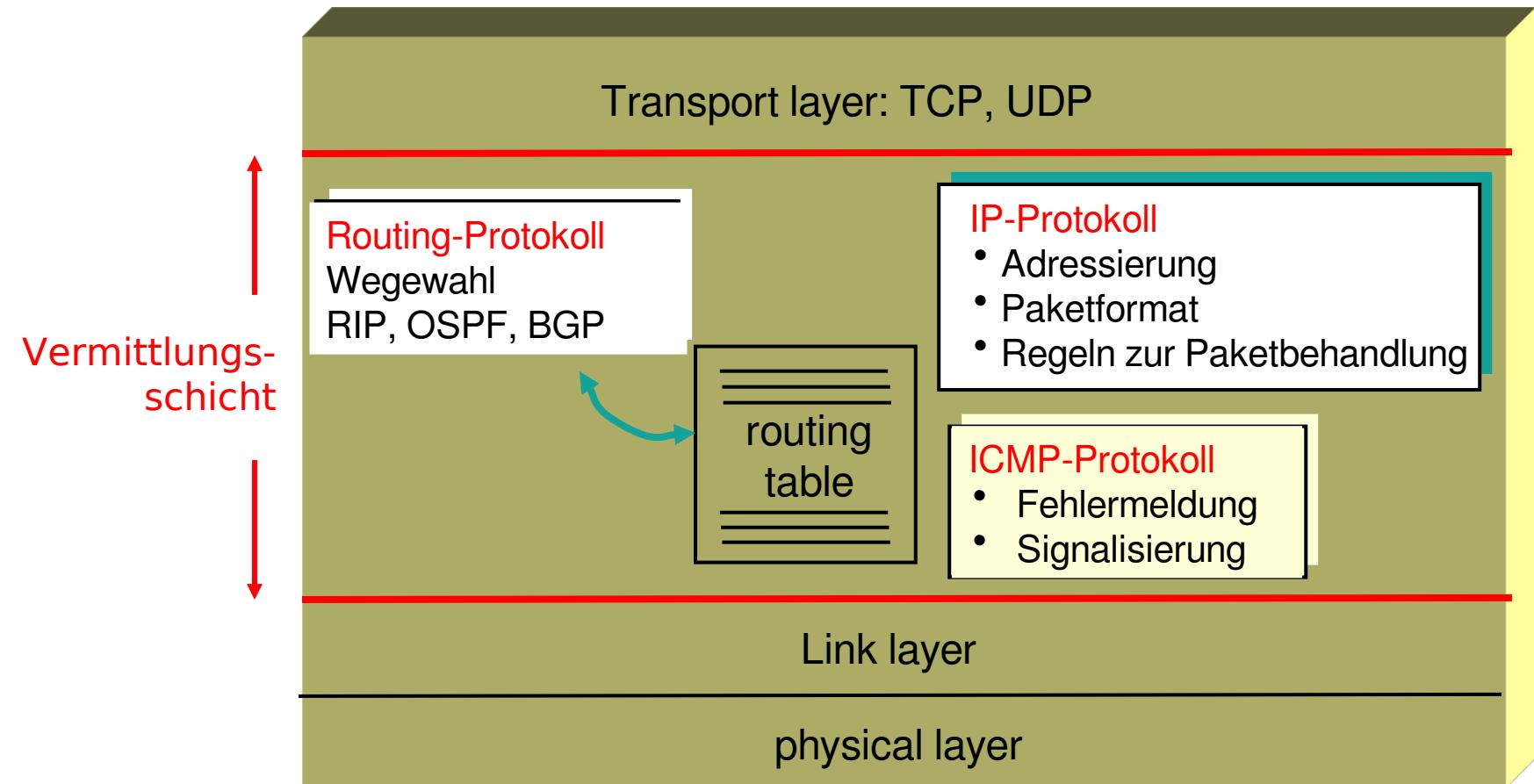
Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Die Vermittlungsschicht im Internet



Inhalt

- IP- und MAC Adressen, ARP Protokoll
- RARP, BOOTP, DHCP
- IP Header
- Fragmentierung
- Path MTU Discovery
- ICMP
- IPv6

Adressierung im Internet: IP Adresse

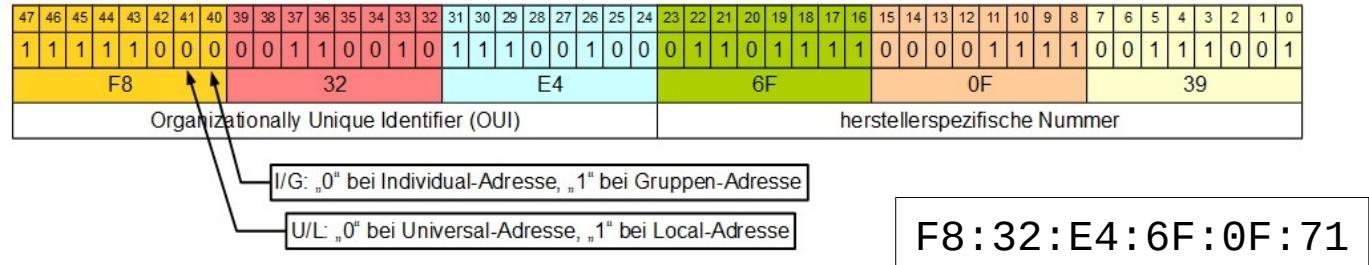
- 32-Bit
- Ist eine **logische Adresse**, die Topologische Informationen enthält (Netzwerk/Host)
→ Wird zur Wege-Findung über Netze hinweg verwendet
- ggf. nicht eindeutig (siehe private Netze)
- Soll ggf. mit anderer Hardware **wiederverwendet** werden können
- Wäre schon lange zu klein um jeden einzelnen Rechner zu adressieren

Adressierung im Internet: IP und MAC Adresse

- Schon kennengelernt: **IP-(Ziel)Adresse** als zentrale Information zur Weiterleitung von IP Datagrammen
- Auf Layer-2 Ebene existiert aber die sog. **MAC-Adresse**
- Warum?
- IP-Adresse: „Logische“ Adressierung (Layer 3)
- MAC-Adresse: „Physikalische Adressierung“ (Layer 2)
- Sinnvoll z.B. beim Szenario: „Austausch eines Servers“
Neue MAC-Adresse, alte IP !

Adressierung im Internet: MAC Adresse

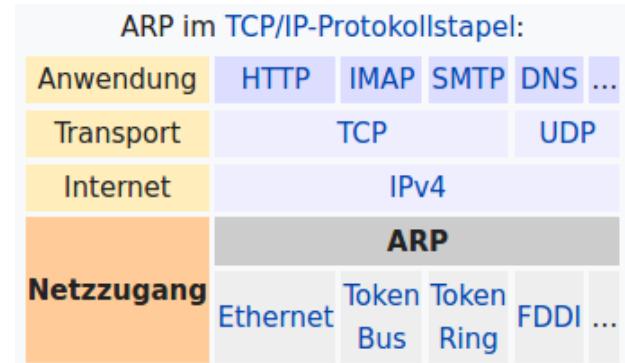
- 48-Bit



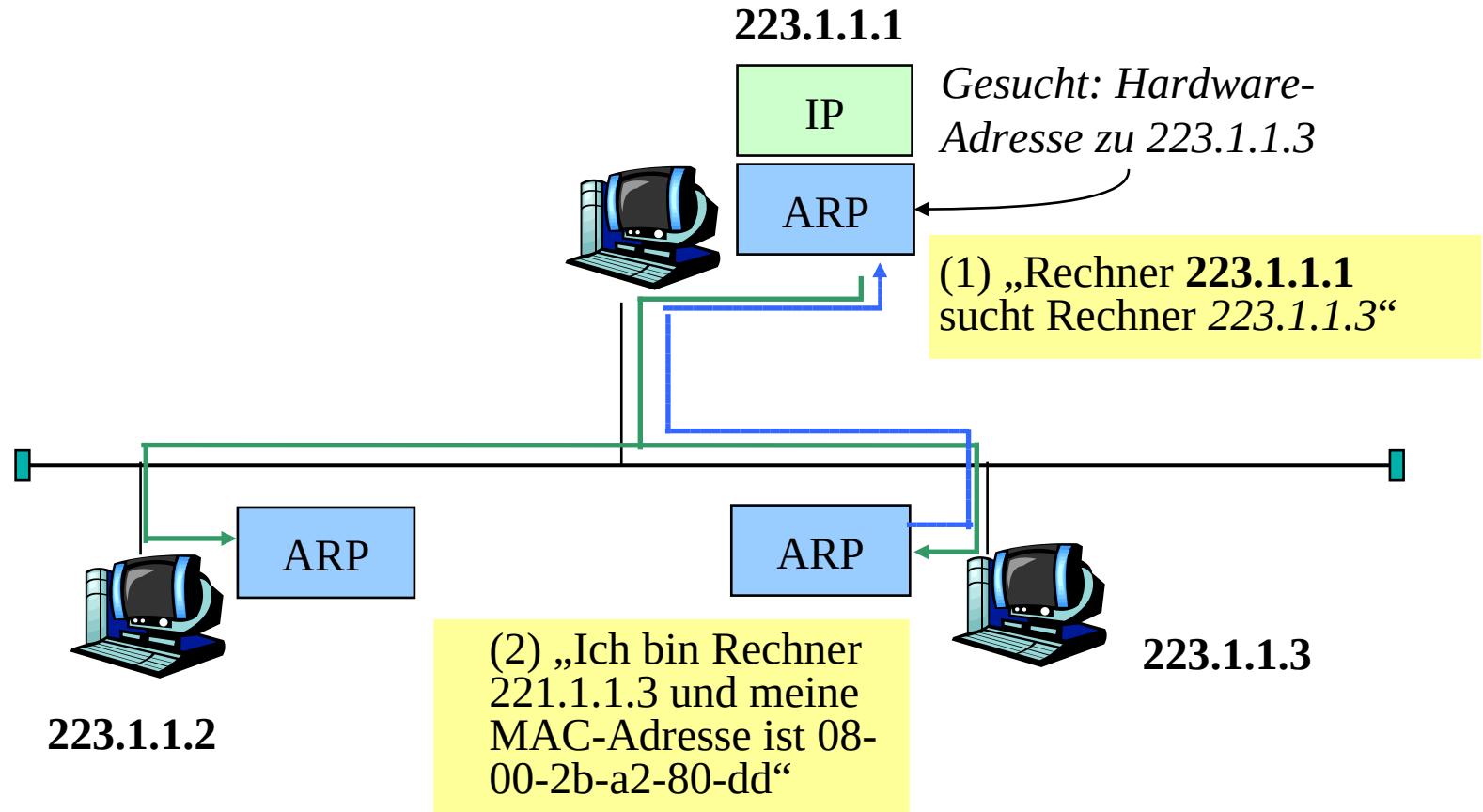
- Ist eine **pysikalische Adresse**, die neben einer ID lediglich Informationen zum Hersteller enthält → kein Routing möglich
 - 22 Bit Herstellernummer, 24 ,Bit Seriennummer'
 - **Broadcast**-Adresse: FF:FF:FF:FF:FF:FF
 - Ist für jede Netzwerk-Schnittstelle **eindeutig**
 - Dient der Addressierung von Paketen **innerhalb des gleichen Subnetzes**
 - Ermöglicht die Idenzifizierung eines neuen Knotens auch **ohne vorherige Konfiguration**
 - Ist auf Layer-2 ebene einfach zu vergleichen

Adressierung im Internet: IP & MAC Adresse

- Wenn ein IP Datagram über Schicht 2 versendet werden soll, muss die MAC-Adresse bekannt sein!
- Layer-2 Protokoll:
ARP (Address **R**esolution **P**rotocol)
- Ermöglicht das Herausfinden einer MAC-Adresse auf Basis einer versendeten IP-Adresse
- Funktioniert über Broadcast-Paket an MAC FF:FF:FF:FF:FF:FF
- Der Host mit der angefragten IP-Adresse antwortet mit seiner MAC-Adresse
- Der empfangende Knoten „cached“ i.d.R. die MAC-Adressen

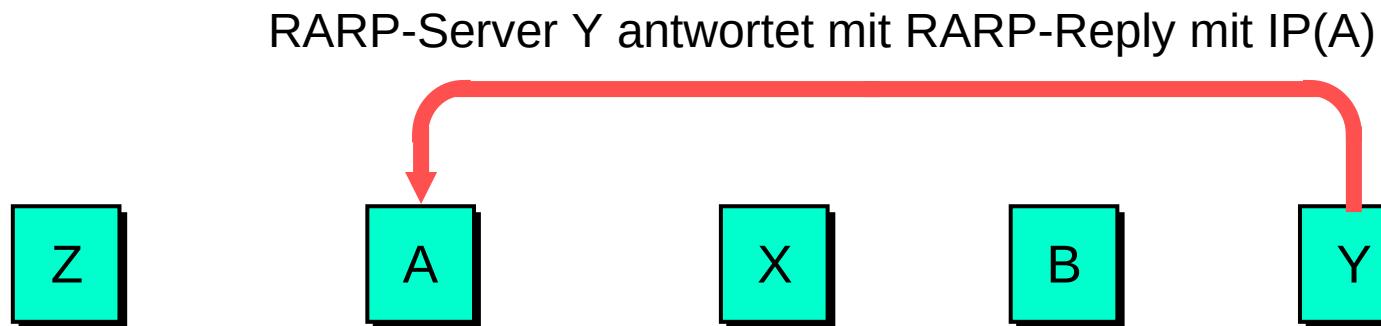
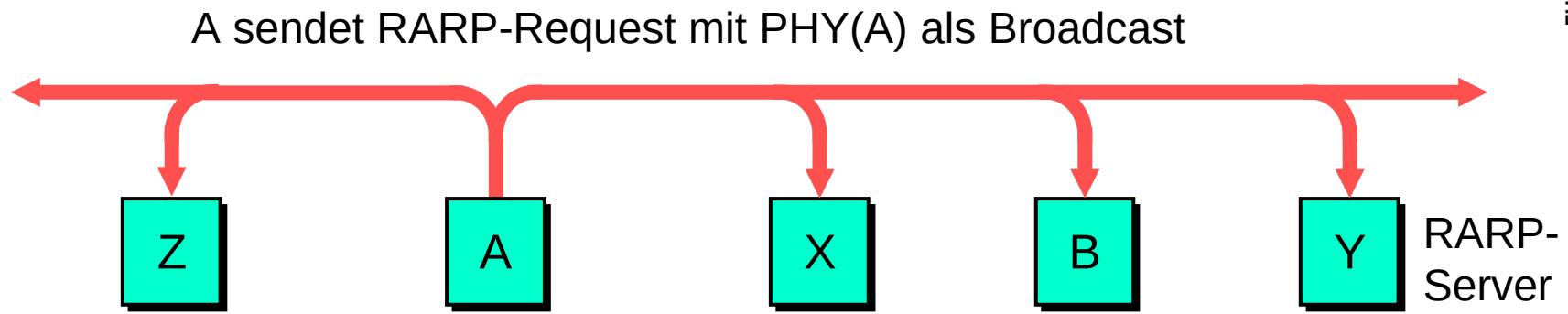


Address Resolution Protocol (ARP)



- Beim Neuanschluss eines Knotens an das Netz entsteht das umgekehrte Problem:
- Ich bin ein Knoten mit MAC-Adresse X, und brauche zur Kommunikation im Internet eine **IP**
- Einfache Lösung:
Manuelle Konfiguration
- Benutzerfreundliche Lösung:
Automatisch per Protkoll
- Veraltetes Protokoll:
RARP (Reverse Address Resolution Protokoll)

RARP (Reverse Address Resolution Protocol)



Probleme von RARP

- Ethernet-Broadcasts sind auf Subnetze beschränkt.
In einem LAN mit Subnetze braucht man mehrere RARP-Server
- Durch RARP erfährt ein Rechner nur seine IP-Adresse!
Zu einer vollständigen Konfiguration einer Netzwerkschnittstelle gehören noch mindestens Netzmaske und Default-Gateway.
- DHCP hat RARP heute komplett abgelöst!

DHCP

- **DHCP (Dynamic Host Configuration Protokoll)**

- Basiert auf dem älteren BOOTP-Protokoll und ist zu diesem (eingeschränkt) kompatibel

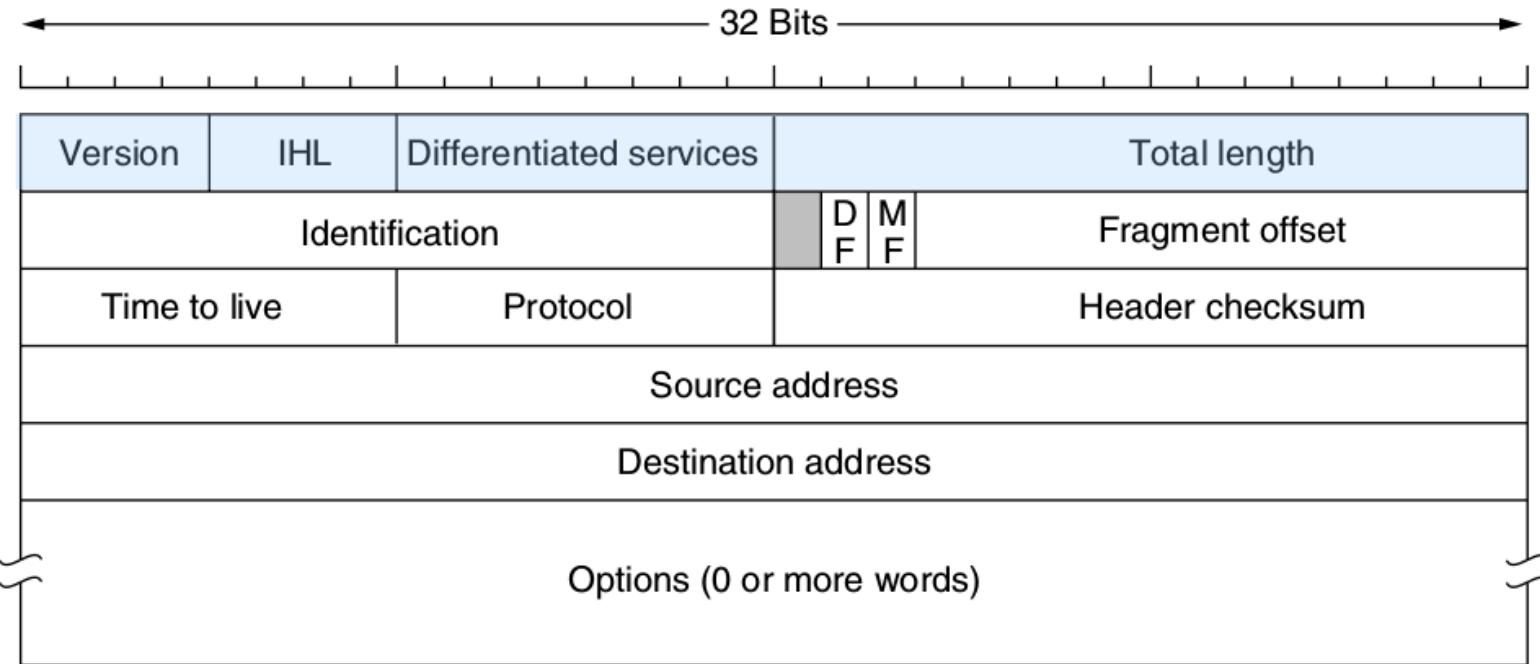
Anwendung	DHCP				
Transport	UDP				
Internet	IP (IPv4, IPv6)				
Netzzugang	Ethernet	Token Bus	Token Ring	FDDI	...

- Realisiert als Application Protokoll über UDP Port 67
- Ermöglicht die Konfiguration über Subsystemgrenzen (DHCP Relay Agents) und mit Szenarien mit mehreren DHCP-Servern
- Erlaubt das Konfigurieren aller wichtigen Parameter (IP/Mask/default Gateway/DNS Server)
- Erlaubt das zeitlich u.A. beschränkte Vergeben von IP-Konfigurationen (leases) und das Steuern der Vergabe
- Sicherheitsprobleme: MAC-Spoofing, DHCP Starvation etc.

Wie wird IP datentechnisch repräsentiert?

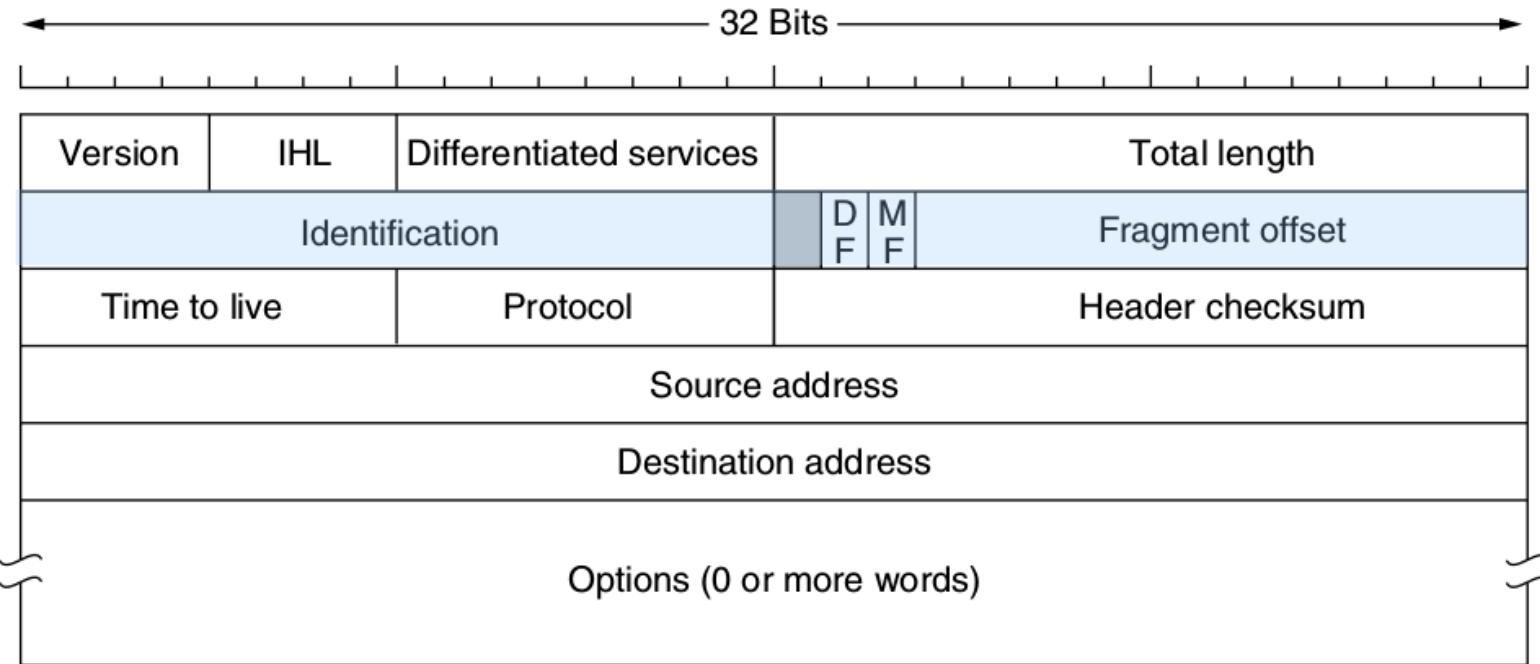
→ **IP Header**

Der IP-Header



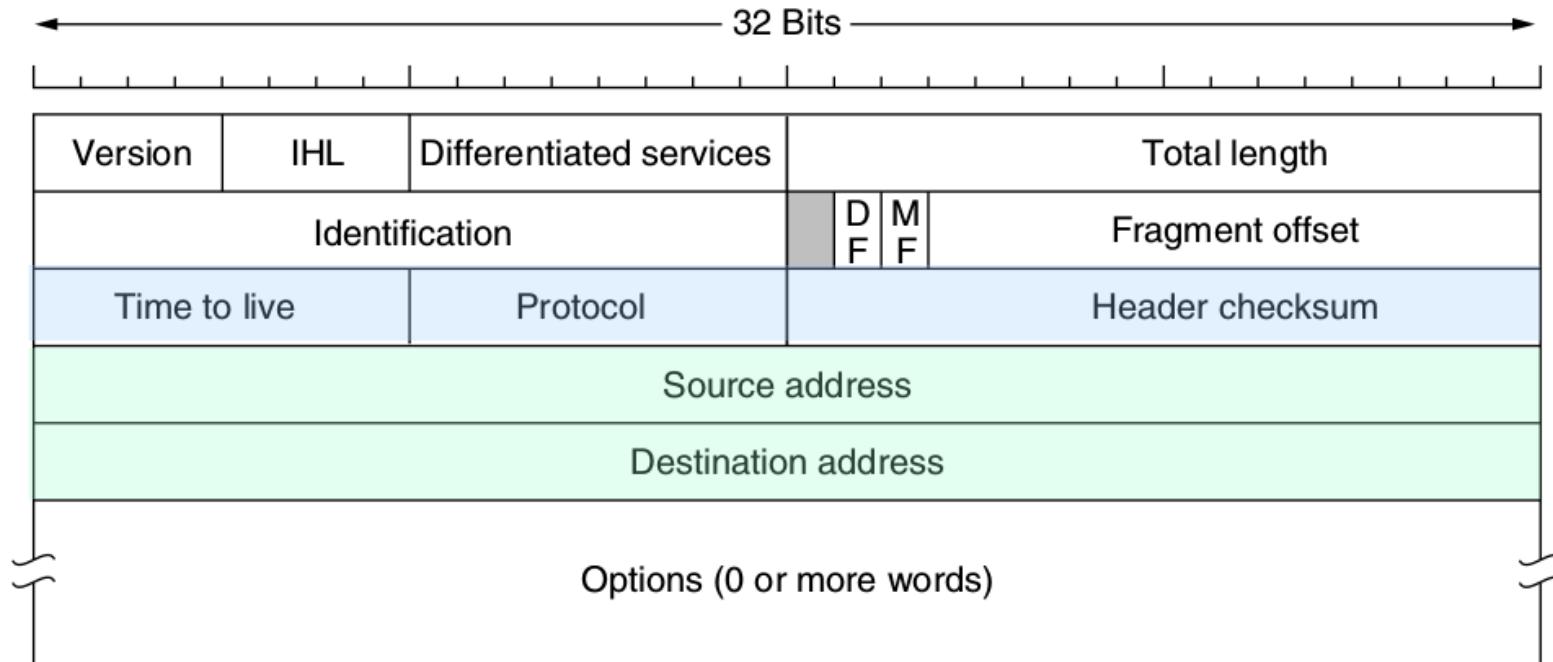
- Version: 4 / 6
- IHL: Länge des Headers mit $N * 32\text{Bit}$
- Diff. Services: Delay / Vorrang etc.
- Total length: Länge des **gesamten** Paketes mit Kopf

Der IP-Header



- Identification: Eindeutige **Kennung** des Datagramms
- DF: **Don't Fragment**
- MF: **More Fragments**
- Fragment offset: Offset im Payload **N * 8 Byte**

Der IP-Header



- Time to live (TTL): max. 255, Dekrementiert, 0→delete
- Protocol: Transportiertes Protokoll (TCP/UDP/ICMP/BGP ...)
- Header checksum: Einfache Prüfsumme
- Source address: Quell-**IP**
- Destination address: Ziel-**IP**

Probleme mit der Paketgröße

- Ein IP-Datagram ist offensichtlich bis zu 64kB groß
- Auf Layer 2 gibt es in der Regel eine

MTU (Maximum Transfer Unit)

z.B.

Ethernet: 1500 Bytes

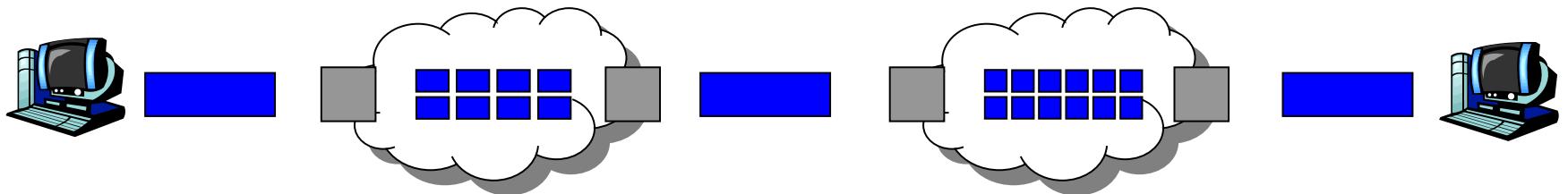
IEEE 802.11 (WLAN): 2272 Bytes

Mögliche Lösungen:

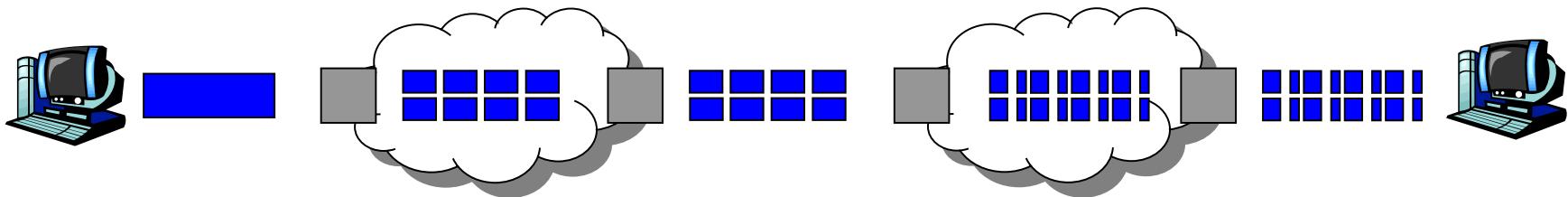
- **Fragmentierung**
- **Path MTU discovery**
- **Jumbo frames**

Fragmentierung

- „Zerhacken“ eines großen IP-Datagramms in mehrere Teile
- Im Internet wird nur eine Vorgabe über die Mindestrahmenlänge von 576 Bytes gemacht.
- Zerhacken ist einfach, aber wo wird das Datagramm wieder zusammengesetzt?



Transparente Fragmentierung

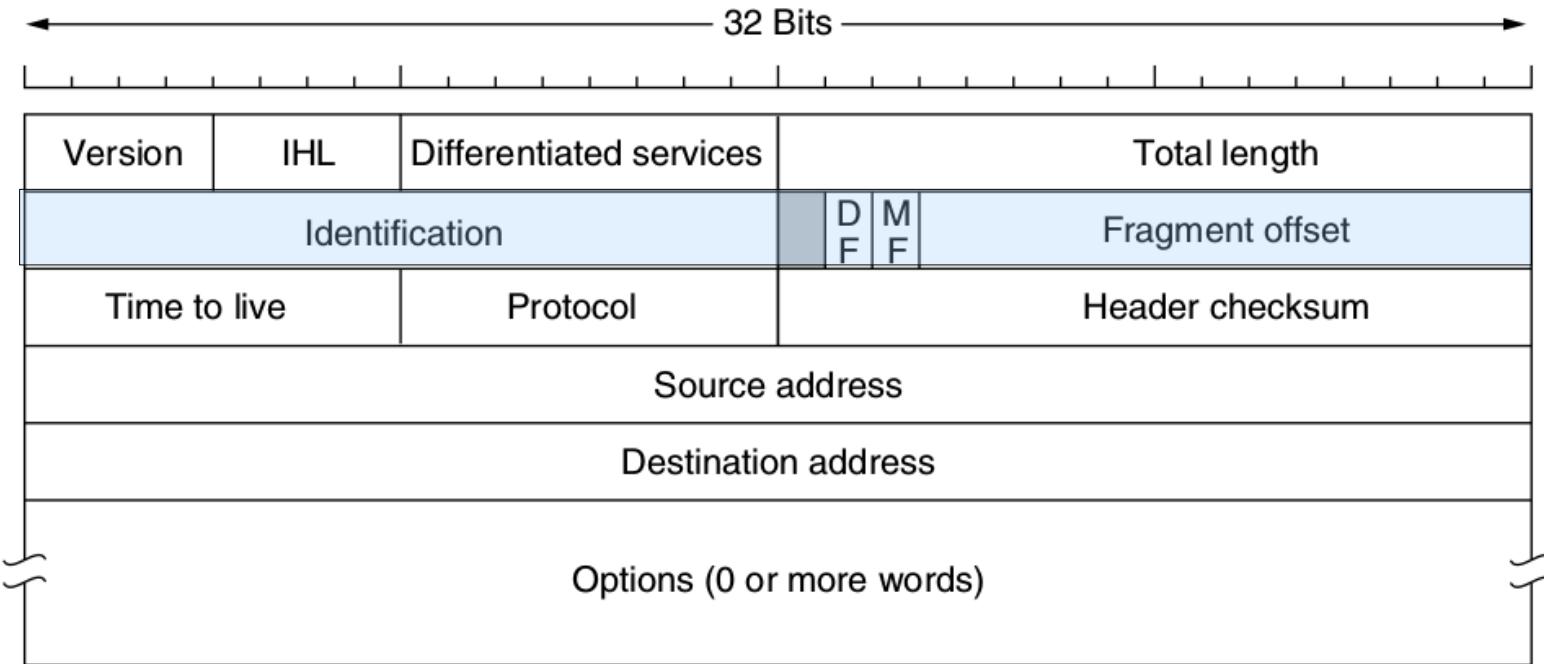


Nicht transparente Fragmentierung (Internet)

Fragmentierung

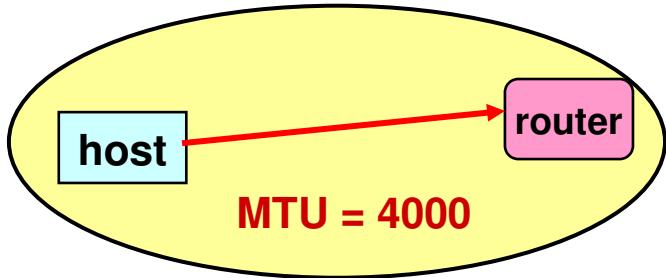
- Im Zielsystem (oder bei den Zwischenstationen) muss aus den Fragmenten wieder die ursprüngliche Dateneinheit hergestellt werden (**reassembly**).
 - Wenn nicht alle Fragmente eines Datagramms das Zielsystem erreichen, muss das gesamte Datagramm von der Quellstation aus wiederholt werden.
 - Fragmente können in unterschiedlicher Reihenfolge beim Zielsystem ankommen
- Empfänger braucht einen 64kB Buffer, in den er die ankommenden Fragmente einsortiert ...

Der IP-Header



- Identification: Eindeutige **Kennung** des Datagramms
- DF: **Don't Fragment**
- MF: **More Fragments**
- Fragment offset: Offset im Payload **N * 8 Byte**

IP-Fragmentierung Beispiel #1



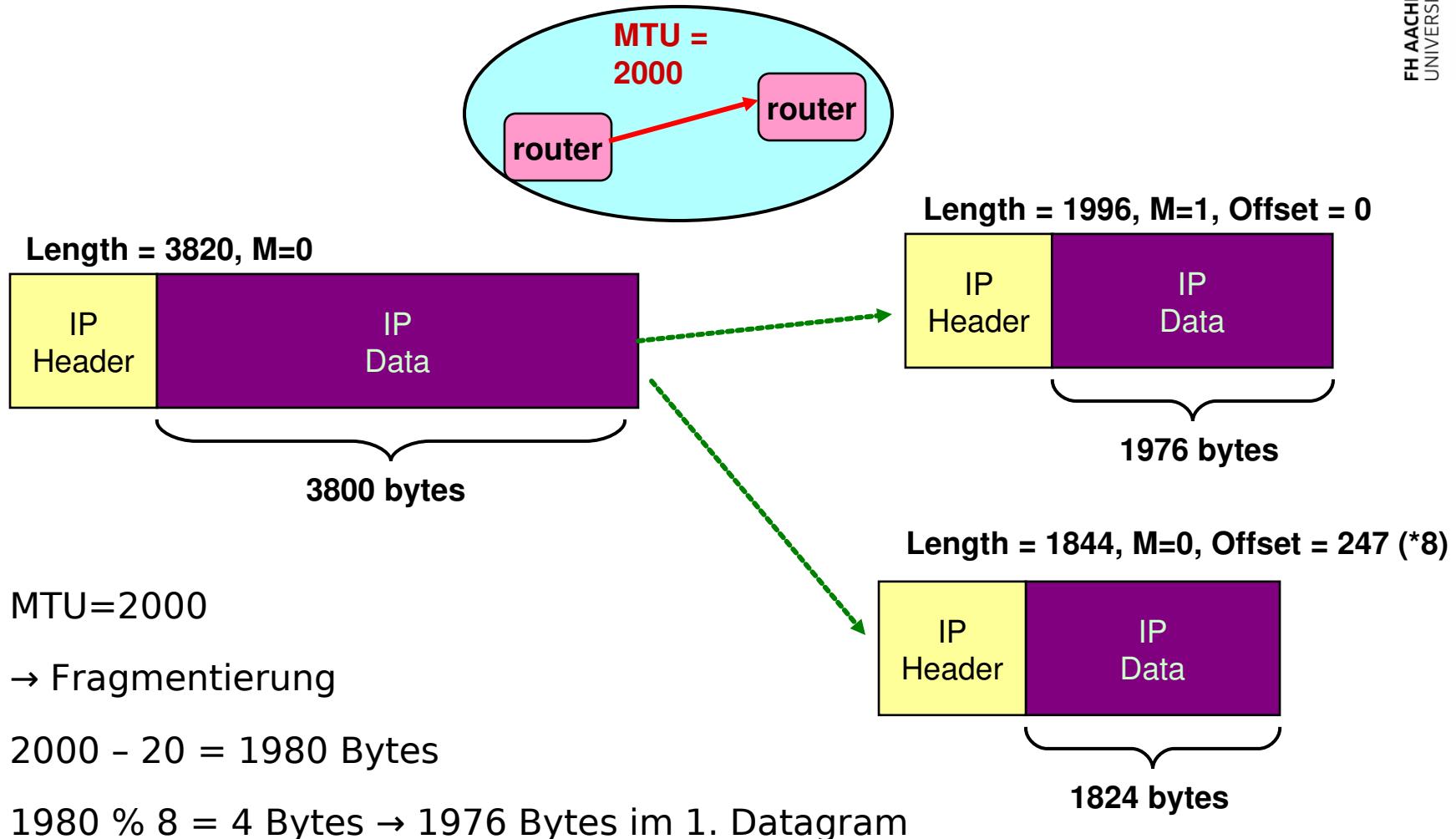
- Es sollen 3800 Bytes übertragen werden
- MTU=4000
- → keine Fragmentierung

Length = 3820, M=0

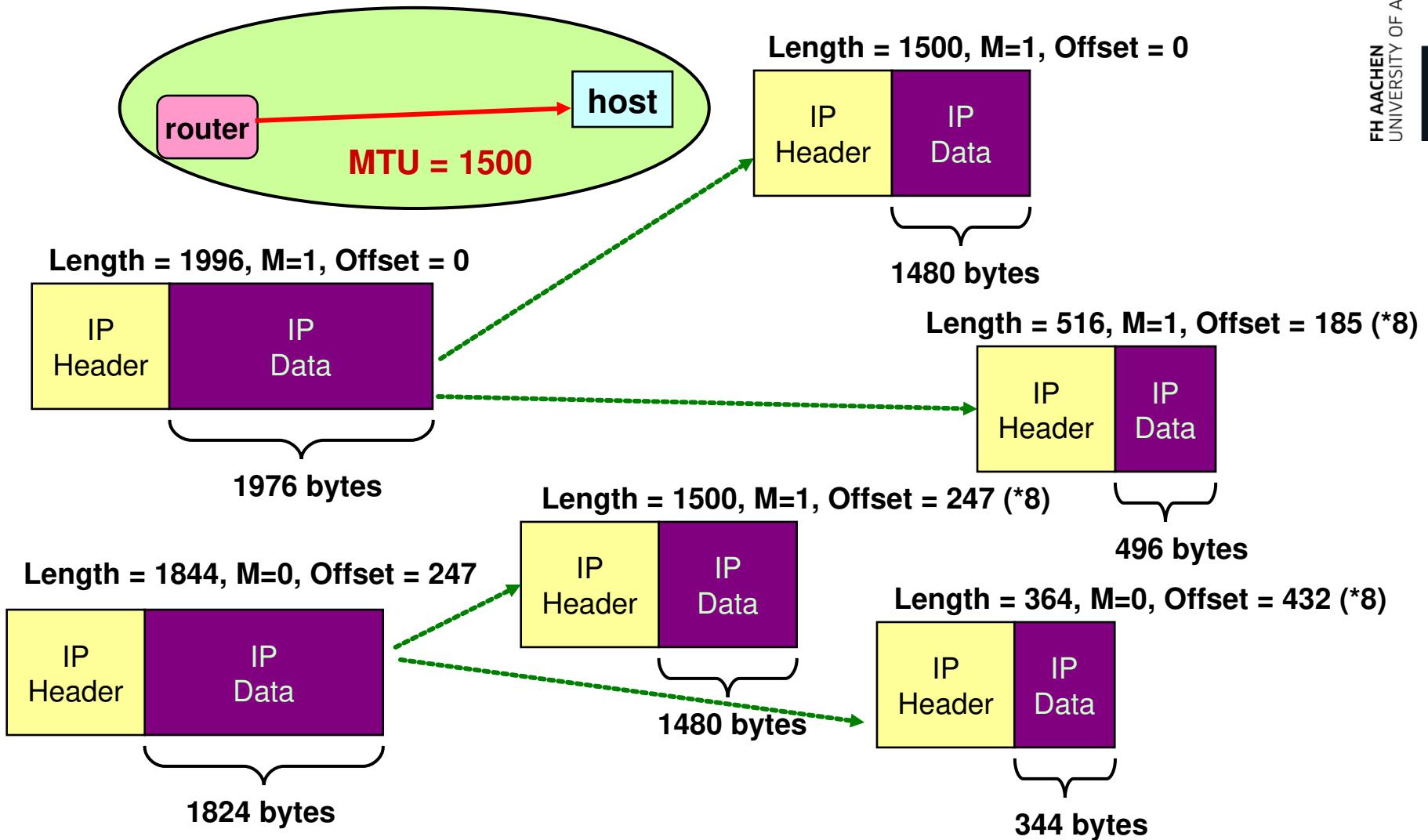


M : „More Fragments“ - Flagge

IP-Fragmentierung Beispiel #2



IP-Fragmentierung Beispiel #3



Zusammensetzung der Fragmente

Length = 1500, M=1, Offset = 0



Length = 520, M=1, Offset = 185



Length = 1500, M=1, Offset = 247



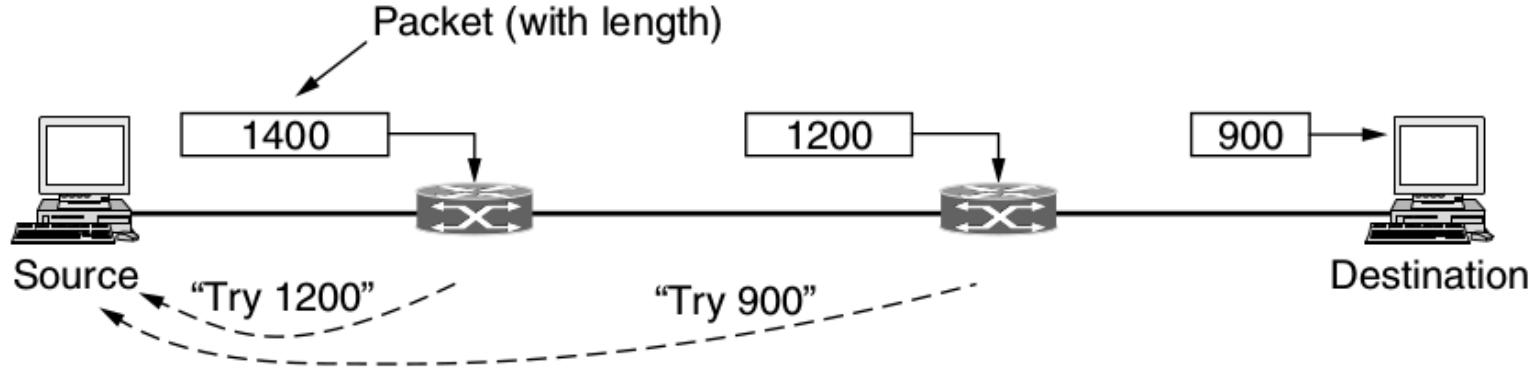
Length = 360, M=0, Offset = 432



- Zusammensetzen möglich durch Fragment Offset
- Fragmente können out-of-order ankommen
- Es ist unklar wieviel Speicher verwendet werden muss (max 64 kB)
- MF = 0 kann auch durch Misordered Packets vorab kommen
- Fragmente können dupliziert werden
- Identifikation und Filtern
- Einige Fragmente können nie ankommen
→ irgendwann muss aufgegeben werden



Path MTU Discovery



- Das Quell-System schickt die Pakete mit der Flagge DF (don't fragment)
- Wenn ein Router eine zu kleine MTU erkennt, wird das Paket verworfen und eine **ICMP**-Nachricht zurück gesendet
- Das Quell-System kann jetzt kleinere Pakete erzeugen, die weiter geleitet werden können
- Der Prozess muss ggf. mehrmals wiederholt werden!

Jumbo Frames

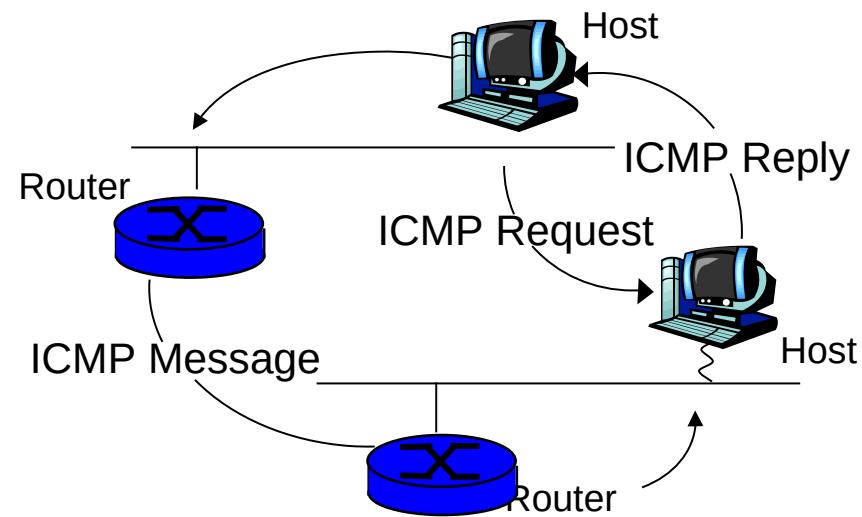
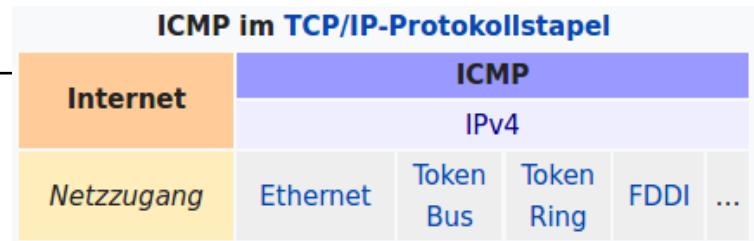
- Auf Layer 2 können größere MTUs verwendet werden
- Für standard Ethernet existiert ein (eher selten unterstützter) Standard für eine MTU von 9000 Byte
- Problem für IP Datagramme > 9000 Byte besteht weiter...

Frame-level bandwidth efficiency										
Frame type	MTU	Layer 1 overhead		Layer 2 overhead		Layer 3 overhead	Layer 4 overhead	Payload size	Total transmitted ^[A]	Efficiency ^[B]
Standard	1500	preamble 8 byte	IPG 12 byte	frame header 14 byte	FCS 4 byte	IPv4 header 20 byte	TCP header 20 byte	1460 byte	1538 byte	94.93%
Jumbo	9000	preamble 8 byte	IPG 12 byte	frame header 14 byte	FCS 4 byte	IPv4 header 20 byte	TCP header 20 byte	8960 byte	9038 byte	99.14%
Other frame sizes for reference										
IEEE 802.11 ^{[14][15]}	7935	PLCP preamble & header 24 byte	IPG varies	frame header & security ovhd 52 byte	FCS 4 byte	IPv4 header 20 byte	TCP header 20 byte	7895 byte	8015 + IPG size byte	< 98.5%
IEEE 802.11 bridged to Ethernet	1500	PLCP preamble & header 24 byte	IPG varies	frame header & security ovhd 52 byte	FCS 4 byte	IPv4 header 20 byte	TCP header 20 byte	1460 byte	1580 + IPG size byte	< 92.4%

1500 Bytes ohne Layer1/2 header

ICMP

- **Internet Control Message Protocol**
- ICMP ist ein Steuerprotokoll der Schicht 3, welches auf IP **aufbaut!** Dieses Protokoll wird z.B. von Routern verwendet, wenn etwas Unerwartetes passiert.
- Aufgaben:
 - Mitteilung von Problemen beim Paketversand
 - Echo-Anfragen (existiert der Zielknoten?)
 - Unterstützung von höheren Protokollen und Anwendungen (z.B. Path MTU discovery, traceroute, ...)



ICMP - Header

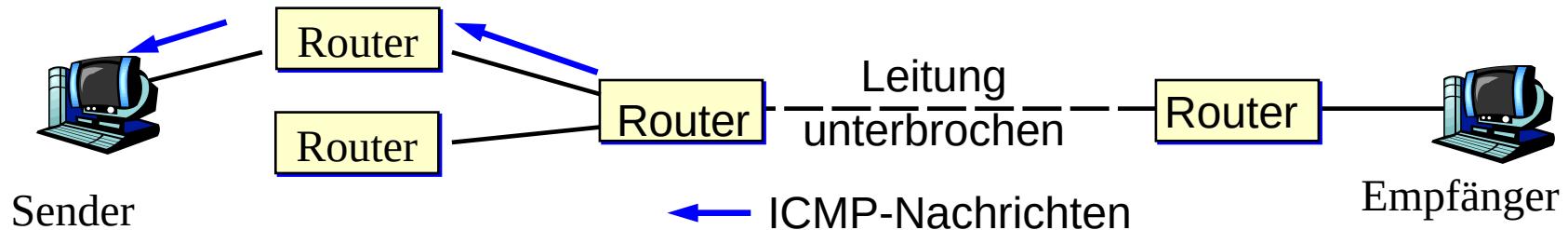
- ICMP versendet Fehler- und Kontrollnachrichten auf Netzebene. Diese Nachrichten werden in ein IP-Paket verpackt
- ICMP-Nachrichtenformat (Auszug!):

IP Header (<i>Protocol = 1</i>)		
Type	Code	Checksum
Identifier	Sequence Number	
Optional Data		

Typ	Typname	Code	Bedeutung
0	Echo-Antwort	0	Echo-Antwort
3	Ziel nicht erreichbar	0	Netzwerk nicht erreichbar
		1	Host (Zielstation) nicht erreichbar
		2	Protokoll nicht erreichbar
		3	Port nicht erreichbar
		4	Fragmentierung nötig, Don't Fragment aber gesetzt
		5	Route nicht möglich (die Richtung in IP-Header-Feld Option falsch angegeben)
		13	Communication administratively prohibited (Paket wird von der Firewall des Empfängers geblockt)
4	Entlasten der Quelle	0	Datagramm verworfen, da Warteschlange voll
8	Echo-Anfrage	0	Echo-Anfrage (besser bekannt als „Ping“)
11	Zeitlimit überschritten	0	TTL (Time To Live, Lebensdauer) abgelaufen
		1	Zeitlimit während der Defragmentierung überschritten

Steuerung von IP: ICMP

- IP ist nur für den (unzuverlässigen) Datenaustausch zuständig.



- **Nachrichtentypen, Beispiele:**

- **Destination Unreachable**: Ziel nicht erreichbar.
- **Time Exceeded**: Time-to-Live-Feld eines Pakets ist abgelaufen.
- **Echo Request / Reply**: Echo Reply wird angefordert ("ping").
- **Timestamp Request / Reply**: Ähnlich Echo Request. Zusätzlich Zeitstempel mit Ankunftszeit der Anfrage/Sendezeit der Antwort.

IPv6

IPv6 - Übersicht

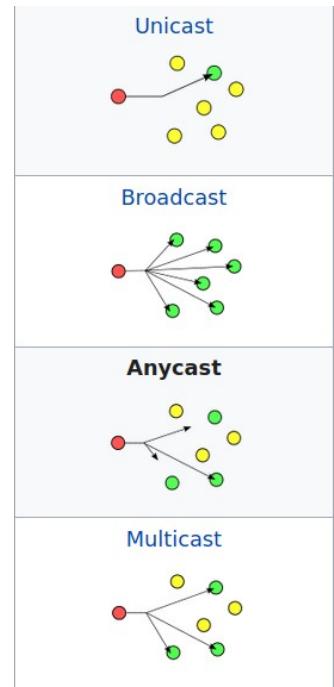
IPv4 (September 1981)  IPv6 (seit Dezember 1995)

Warum ein Wechsel, wenn IPv4 gut funktioniert?

- Dramatisch anwachsender Bedarf für neue IP-Adressen
- Vereinfachung des Protokolls, um eine schnellere Abarbeitung zu gewährleisten
- Sicherheitsmechanismen (Authentifikation und Datenschutz)
- Mehr Gewicht für Dienststarts, insbesondere für Echtzeitanwendungen
- Unterstützung von Mobilität (Hosts können ohne Adressänderung auf Reisen gehen)
- Möglichkeiten zur Fortentwicklung des Protokolls

IPv6 - Auswahl der Eigenschaften

- Adressgröße
 - 128-Bit-Adressen (8 Gruppen zu je 4 Hexadezimal-Zahlen)
- Verbesserter Optionsmechanismus / Einfacher Header
 - Vereinfacht und beschleunigt die Verarbeitung von IPv6-Paketen für Router
 - IHL: überflüssig, keine Optionen mehr
 - Protocol, Fragmentierung: überflüssig, wird durch Optionen mit abgedeckt
 - Checksum: Handhabung durch Schicht 2 und 4
- Verbesserung der Adressflexibilität
 - Anycast Address: Erreiche irgendeinen von mehreren (alle haben die gleiche Adresse)
- Unterstützung der Reservierung von Ressourcen
 - Erkennen von Datenströmen in IP (FlowLabel)
- Sicherheitsmaßnahmen
 - Authentifizierung und Privacy



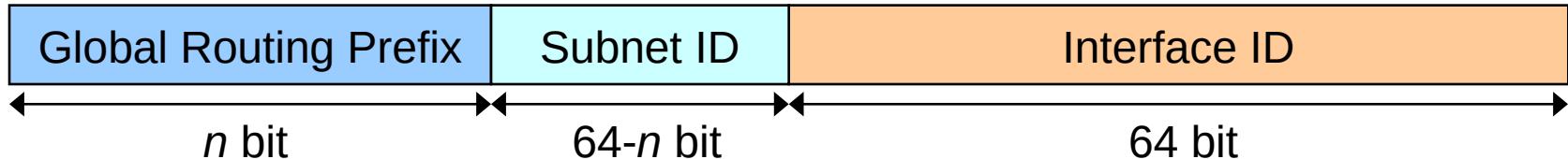
IPv6: Adressierung

IPv6-Adressen umfassen 128 Bit

- IPv6-Adressen werden in **hexadezimaler** Notation mit Doppelpunkten geschrieben: Format x:x:x:x:x:x:x:x, wobei jede Stelle 16 Bit hexadezimal kodiert
- Beispiele:
 - 3FFE:400:20::A00:2BFF:FEA3:ADCB
 - FF01:0:0:0:0:0:0:101 oder FF01::101
 - FEDC:BA98:7600::/40 40 Bit langes Präfix für das Routing
- Unterscheidung von **Adressklassen**:
 - > Unicast-, Anycast- (one-to-nearest), Multicast-Adressen
- Typ einer Adresse wird durch *Präfix* (führende Bits) festgelegt:
 - > Loopback-Adresse: 00...01 (128 Bit) = ::1/128
 - > Multicast-Adresse: 11111111 = ff00::/8
 - > Link-local Unicast-Adresse (DHCP-Ersatz) 1111111010 = fe80::/10
 - > Site local Unicast-Adresse (private) 1111110 = fc00::/7
 - > IPv4-Adresse: 0....01....1 = 0:0:0:0:ffff:0:0/96
 - > Global Unicast-Adressen alles andere

Adressbeispiel

Global Unicast: dreigeteilte Hierarchie



- Globales Routing-Präfix zur Reduktion des Umfangs von Routing-Tabellen (z.B. geographischer Identifier/Identifier pro Provider)
 - Subnet-ID als Adresse eines bestimmten Netzes
 - Interface-ID als eindeutige Adresse eines Rechners, automatisch generiert, z.B. aus der MAC-Adresse:
 - > MAC-Adresse 00:1D:72:8E:74:6C (48 bit)
- Interface ID: **00:1D:72:FF:FF:8E:74:6C** (64 bit)

IPv4-Mapped IPv6 Adressen

- IPv4-Mapped-Adressen ermöglicht die Kommunikation mit End-Systemen, die nur IPv4 können
- Die IPv6 basiert vollständig auf der IPv4 Adresse

0000 . . . 0000

80 bits

FFFF

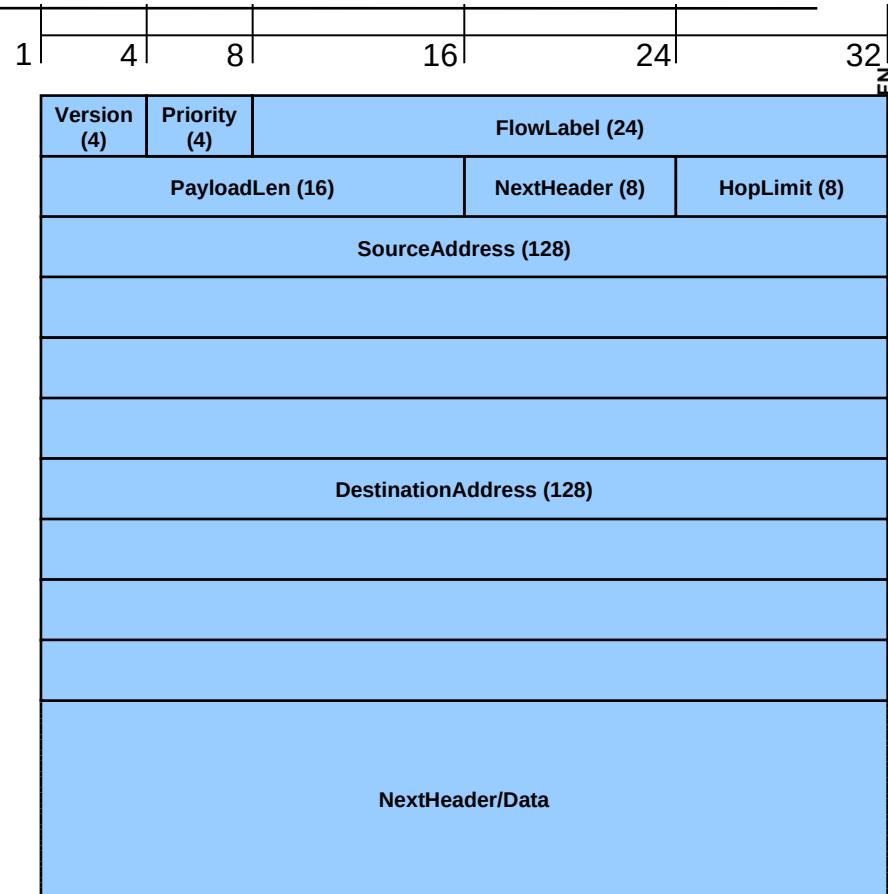
16 bits

IPv4 Address

32 bits

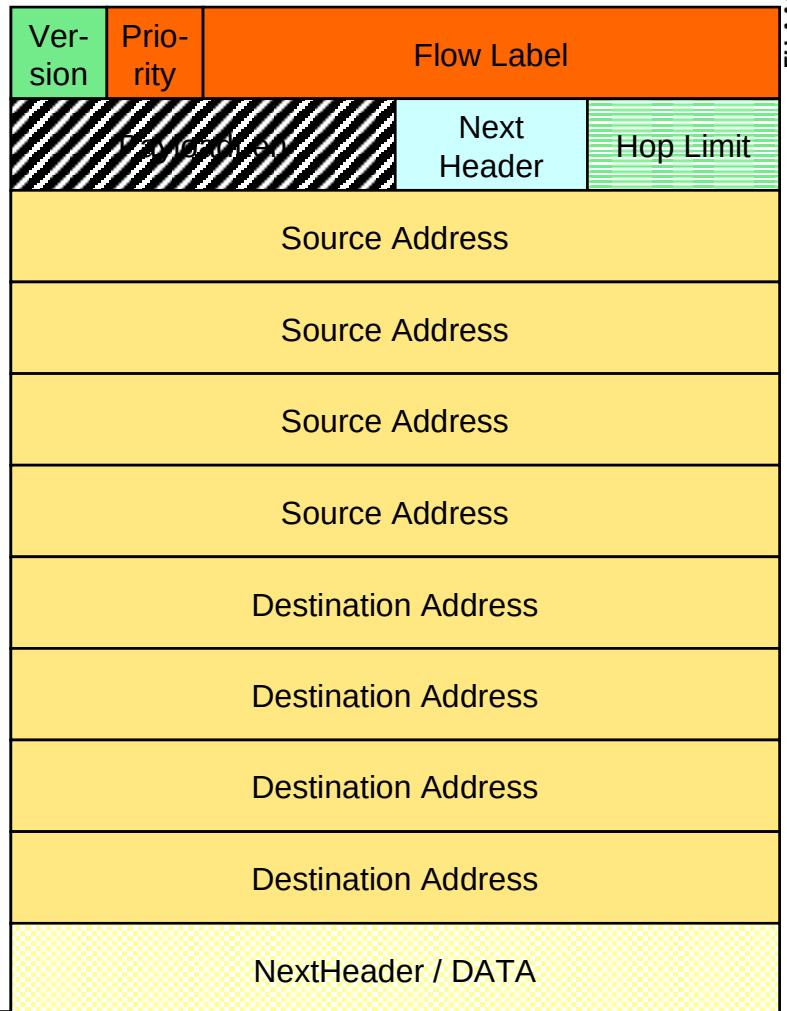
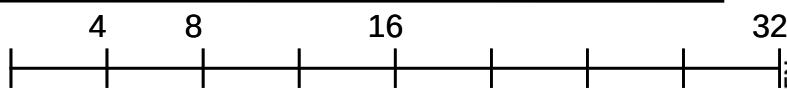
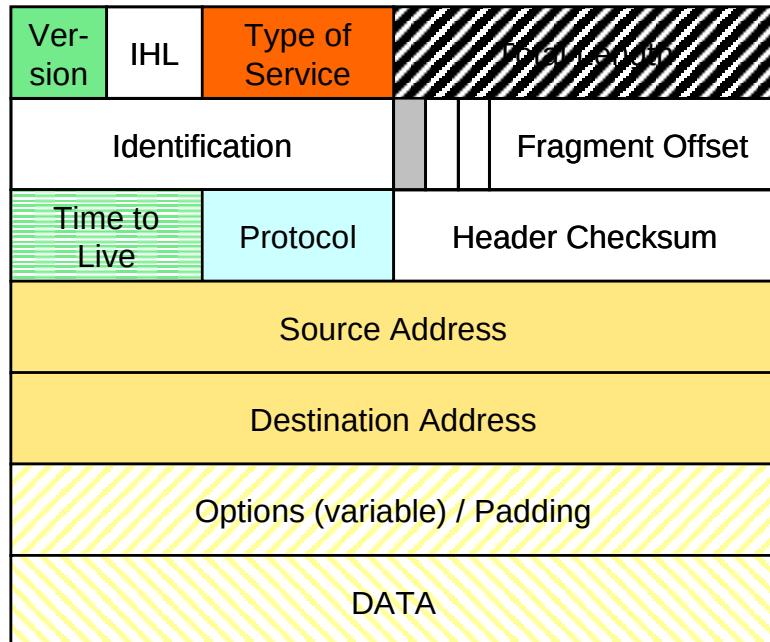
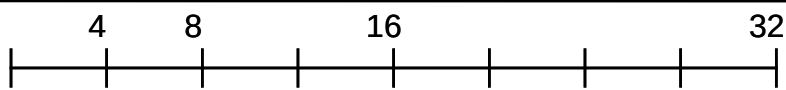
IPv6 Haupt-Header

- **Version:** IP Version Nummer.
- **Priority:** 4 Bit für Priorität. 1 - News, 4 - FTP, 6 - Telnet, 8 bis 15 - Echtzeitverkehr.
- **FlowLabel:** virtuelle Verbindung mit bestimmten Merkmalen/Anforderungen
- **PayloadLen:** Paketlänge nach dem 40-Byte-Header (also ohne Header)
- **NextHeader:** 8-Bit-Selektor. Gibt den Typ des folgenden Erweiterungs-Headers an (oder den Transport-Header)
- **HopLimit:** Wird bei jedem Knoten dekrementiert. Bei Null wird das Paket verworfen
- **SourceAddress:** Die Adresse des ursprünglichen Senders des Pakets
- **DestinationAddress:** Die Adresse des Empfängers (nicht unbedingt das endgültige Ziel, wenn es einen Optional Routing Header gibt)



Das Präfix einer Adresse charakterisiert geographische Bereiche, Provider, lokale interne Bereiche, ...

IPv4 vs. IPv6: Header



Der IPv6-Header ist zwar länger, doch dies liegt nur an den längeren Adressen. Ansonsten ist er „besser sortiert“ und im Router einfacher abzuarbeiten (8 statt 13 Felder)

IPv6 Erweiterungs-Header

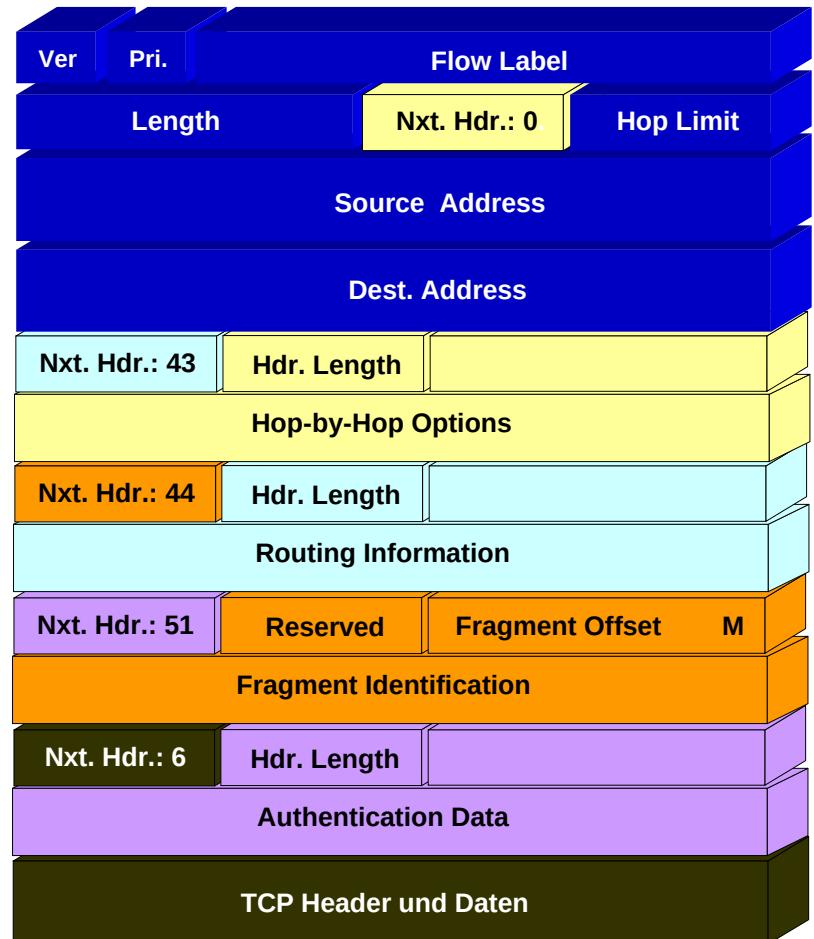
Optionale Angaben folgen in *Erweiterungs-Headern*. Davon sind 6 definiert:

- **Hop-by-Hop** (Informationen für Teilstrecken)
Alle Router müssen dieses Feld prüfen. Momentan definiert ist nur die Unterstützung von Jumbogrammen, d.h. Paketen mit Überlänge (Hierbei wird eine Längenangabe eingetragen).
- **Routing** (Definition einer vollen oder teilweise festgelegten Route)
- **Fragmentierung** (Verwaltung von Fragmenten)
Unterschied zu IPv4: Nur die Quelle kann eine Fragmentierung vornehmen. Router, für die ein Paket zu groß ist, schicken eine Fehlermeldung an die Quelle.
- **Authentifikation** (des Senders)
- **Verschlüsselte Sicherheitsdaten** (Informationen zur Verschlüsselung der Daten)
- **Zieloptionen** (Zusatzinformationen für das Ziel)

Der IPv6-Header ist erweiterbar

Nutzung der Erweiterungs-Header

- Per Hop ausgewertete Header
 - Hop-by-Hop Options (z.B. Jumbogramm Notifier)
 - Routing Information Header
- Nur im Endsystem ausgewertete Header
 - Fragmentation Header
 - Authentication Header
- Header-Extensions u.U. auf Applikationsniveau direkt nutzbar

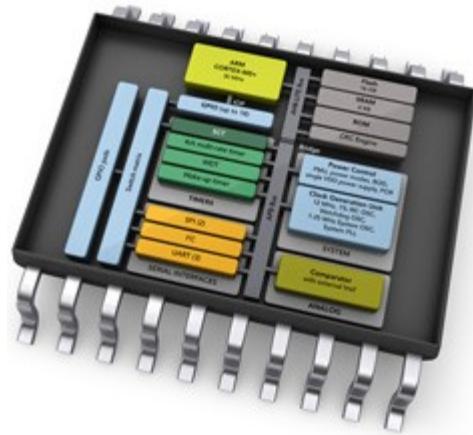


FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge

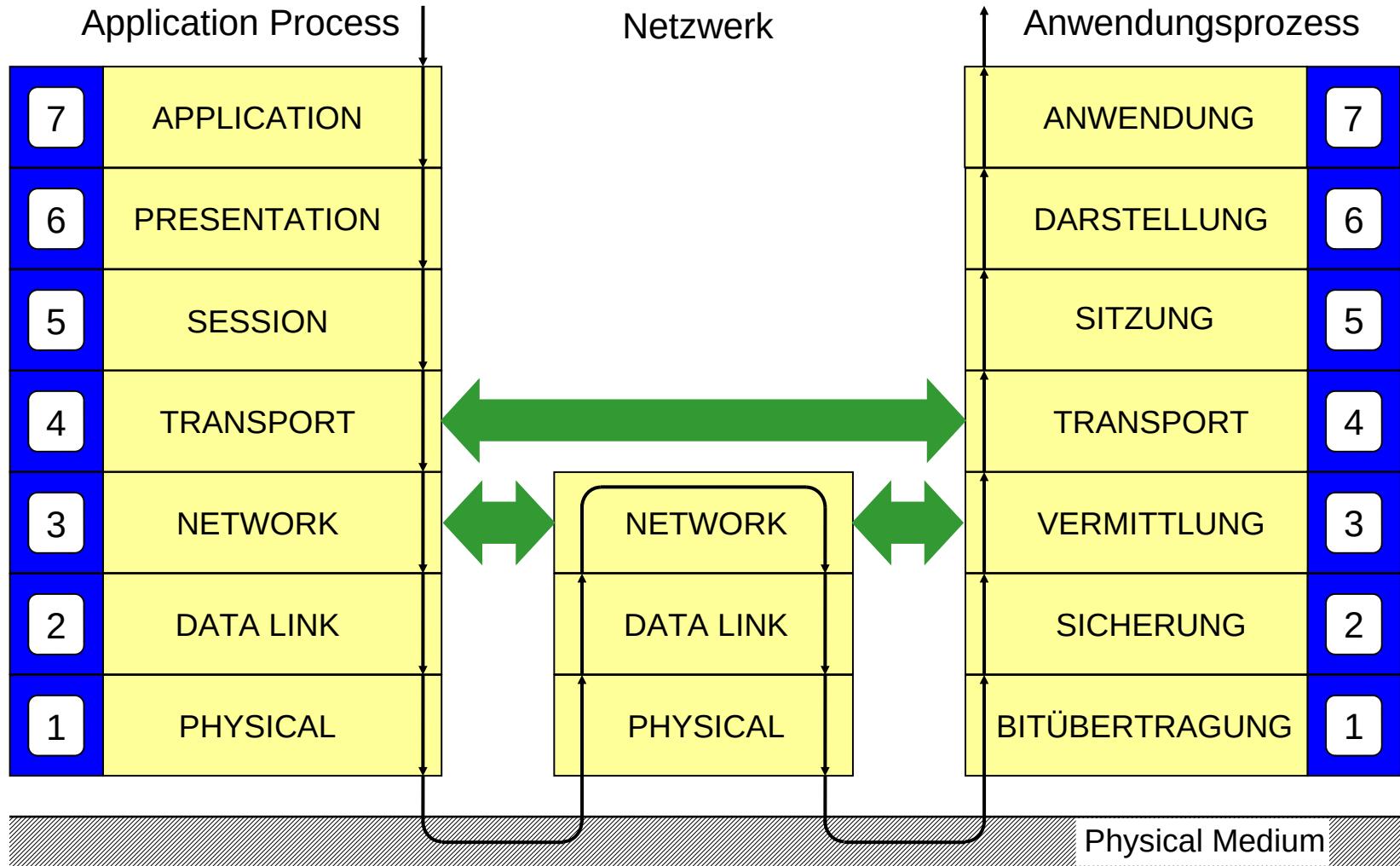


Arbeitsweise der Transportschicht

Transport- und Vermittlungsschicht

- Vermittlungsschicht: logische Kommunikation zwischen *Rechnern in einem Netz aus Netzen*
 - Anbindung von Rechnern in die Netze
 - Regelung des netzübergreifenden Verkehrs:
 - > Kopplung von lokalen Netzen
 - > Globale Adressierung
 - > Routing (Weiterleitung über Zwischenstationen) von Datenpaketen
- Transportschicht: logische Kommunikation zwischen *Prozessen*
 - Benutzt und erweitert die Dienste der Vermittlungsschicht
 - Das erzielbare Dienstangebot wird daher von den Fähigkeiten der Vermittlungsschicht beeinflusst!

Transport- und Vermittlungsschicht



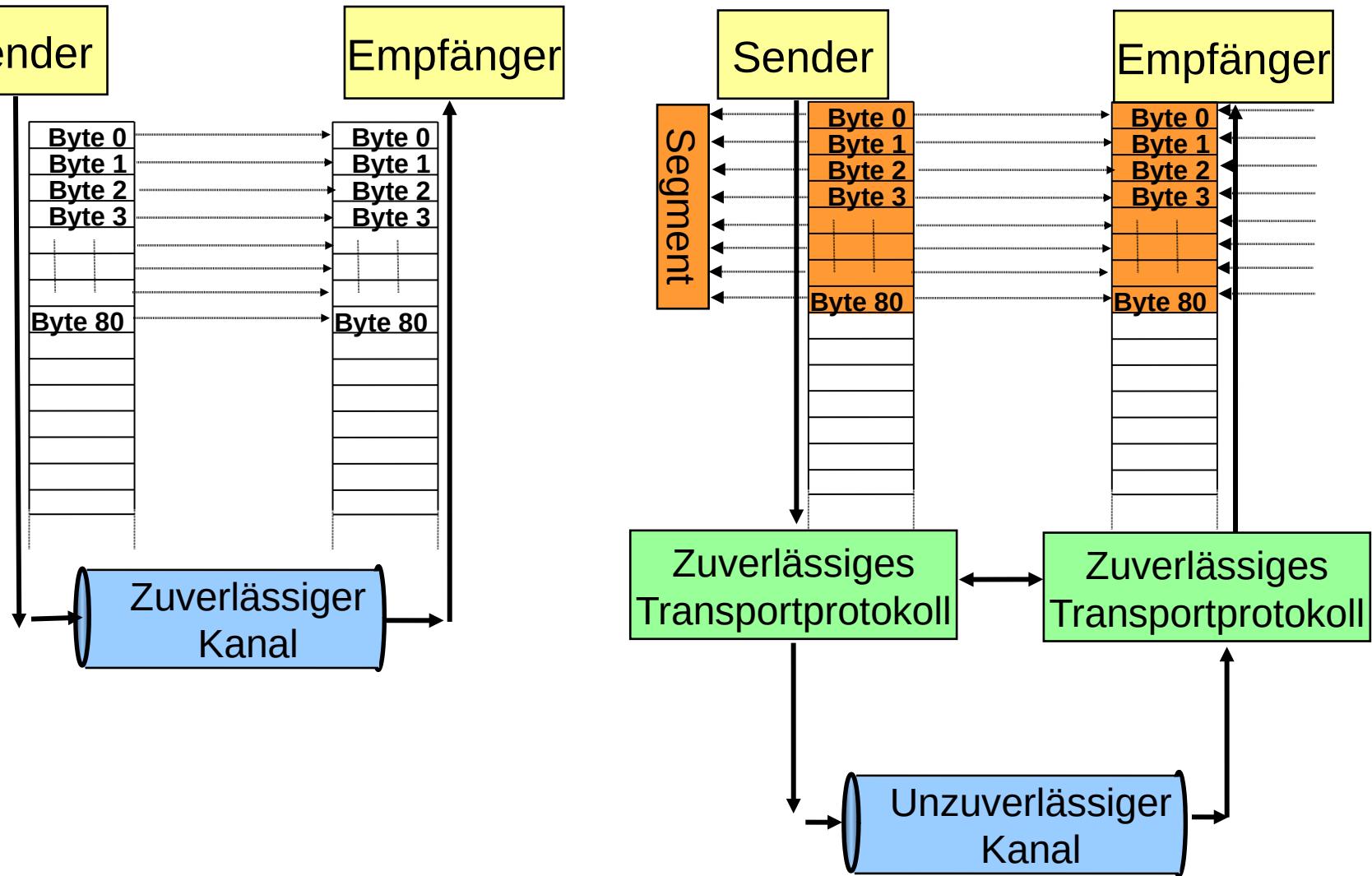
Im **Internet** wird eine zuverlässige Kommunikation zwischen Prozessen entweder auf Transport- oder auf Anwendungsebene realisiert

Viele heute besprochene Sicherungsmechanismen können Sie in anderen Netzen aber auch in der **Sicherungsschicht** finden!

Internet-Protokolle - TCP und UDP

- Transport- und Vermittlungsschicht im Internet
 - Verbindungsorientierte Kommunikation
 - Verbindungslose Kommunikation
- Transmission Control Protocol (TCP) -> zuverlässig
 - Kommunikationsprimitive, Sockets, Ports
 - Virtuelle Verbindungsorientierung: Modell zweier Streams die die Prozesse miteinander verbinden; „Open“-“Close“ der Streams sind die virtuellen Verbindungen
 - Flusskontrolle, Staukontrolle
- User Datagram Protocol (UDP) -> unzuverlässig
 - Kommunikationsprimitive, Sockets, Ports
 - Postkartenmodell: Ein Empfangspunkt (Port) für alle Nachrichten, versenden ohne Absprache mit dem Empfänger

Zuverlässiges Transportprotokoll



Ausgangspunkt

- Wie können eigentlich zuverlässige Protokolle über dem unsicheren IP implementiert werden?
 - Pakete können verloren gehen
 - Die Reihenfolge der Pakete kann sich ändern
 - Pakete können fehlerhaft empfangen werden
- Anmerkung
 - Im folgenden gehen wir von einer **existierenden „Verbindung“** zwischen Sender und Empfänger aus
 - Sender und Empfänger haben also bereits Nachrichten ausgetauscht und **wissen voneinander**
 - Es gibt **keine Verzögerung** bei der Beschaffung oder dem Zustellen neuer Nachrichten an höhere Schichten

Zuverlässige Transportprotokolle

- Ausgangspunkt:
 - Übertragung eines kontinuierlichen Datenstroms in Nachrichteneinheiten (Segmente), die identifizierbar sind
 - Sender und Empfänger wissen voneinander (verbindungsorientiert)
- Erforderliche Mechanismen
 1. Fehlererkennung und ggf. -behebung
 - > Der Empfänger muss Bitfehler in der Übertragung erkennen
 - > Der Empfänger muss das Fehlen (kann aber ggf. später noch ankommen) eines Segmentes erkennen und darauf geeignet reagieren
 2. Empfänger-Feedback
 - > Quittierungsnachrichten zwischen Empfänger und Sender
 - > Positive **ACK**nowledgment (**ACK**)
 - Empfänger bestätigt den korrekten Erhalt eines Segments mit einem ACK
 - > **Optional:** Negative **Ac****K**nowledgment (**NAK**)
 - Empfänger informiert Sender über „Lücken“ oder Fehler
 3. Neuübertragung
 - > Sender überträgt fehlerhafte Segmente erneut
 - > Timer- und/oder NAK-gesteuert

Das “Send and Wait”-Protokoll

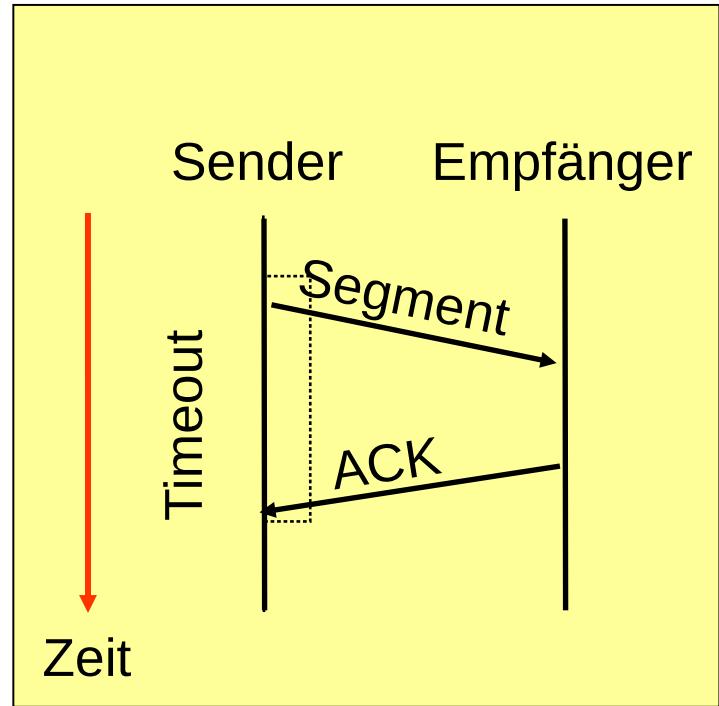
(gelegentlich auch als “Stop-and-Wait” bezeichnet)

■ Automatic Repeat-reQuest (ARQ)

- Empfänger verschickt Bestätigungen (**ACKs**) bei korrektem Empfang
- Fehlerhafte Übertragungen werden durch ausbleiben der Bestätigung erkannt und vom Sender durch Wiederholung behoben

■ Einfachstes ARQ-Protokoll

- Segmentweises Vorgehen
- Timer zur Fehleridentifikation
 - Verschicke ein Segment und warte auf den Empfang des ACKs
 - Bei Empfang eines ACKs wird das nächste Segment verschickt, andernfalls wird die Übertragung wiederholt



Ist dieses Protokoll sicher?

Schlechter Nutzungsgrad der Verbindungskapazität

- Jeder Datenstrom wird in Segmente unterteilt, z.B. 1460 Bytes Nutzdaten
- Nach jedem gesendeten Segment wird auf eine Bestätigung gewartet; während der Wartezeit werden keine weiteren Segmente übertragen
 - ➡ Begrenzung auf ein Paket pro Zyklus (Round-Trip)!!!!
- **Nutzungsgrad:** Verhältnis von genutzter Übertragungszeit zu gesamter Übertragungsdauer (inkl. der Wartezeit)

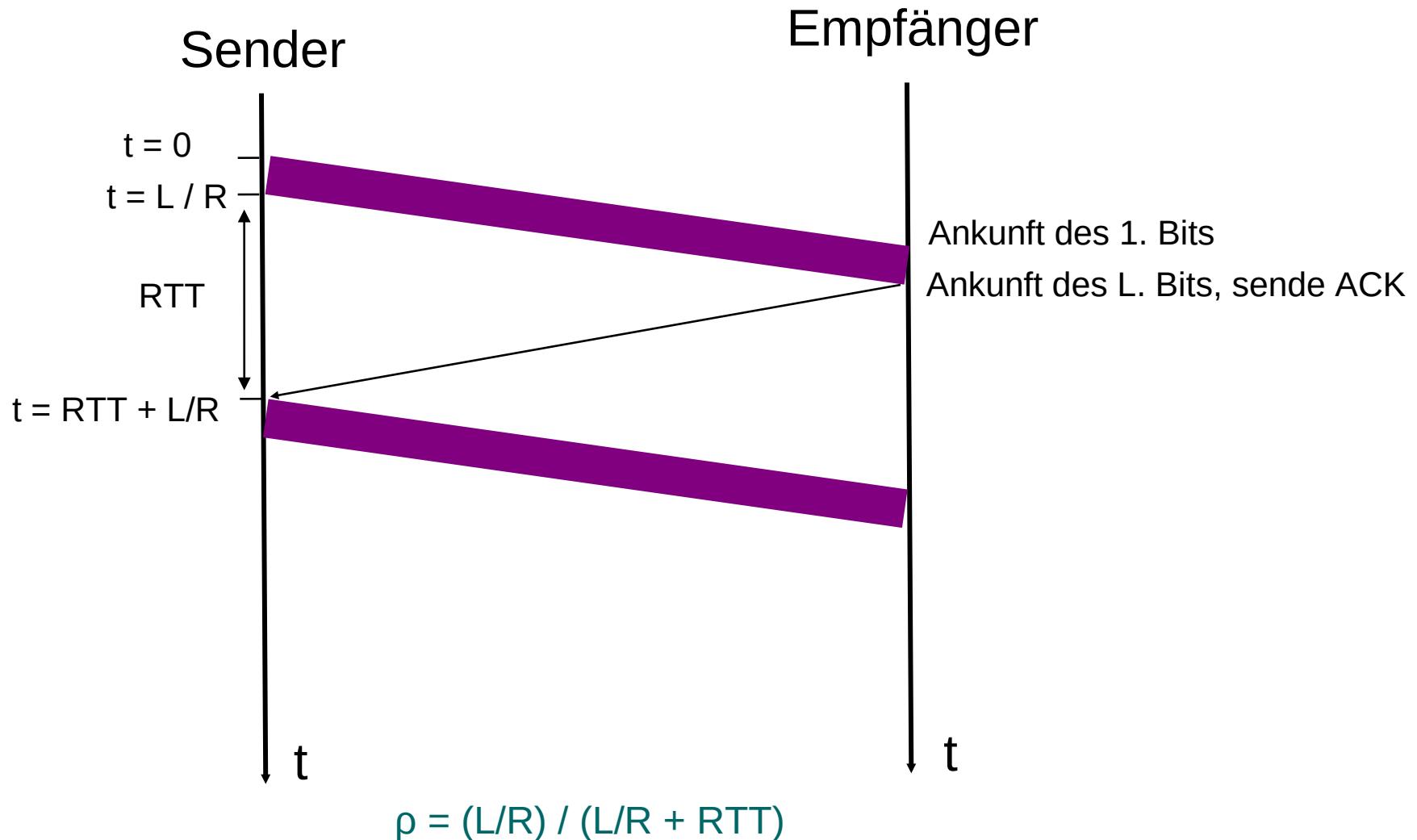
$$\rho = (\text{genutzte Übertragungszeit}) / (\text{genutzte Übertragungszeit} + \text{Wartezeit})$$

Sei L die Länge eines Segmentes [bits]

Sei R die Übertragungsrate des Kanals [bits/s]

→ L / R Zeit zur Übertragung des Segments [s]

Schlechter Nutzungsgrad der Verbindungskapazität



Schlechter Nutzungsgrad der Verbindungskapazität

Beispiel: FastEthernet im LAN

$$R = 100 \text{ Mbit/s}$$

$$L = 1460 \text{ Byte} \quad L/R = 116,8 \mu\text{s}$$

$$RTT = 3 \text{ ms}$$

$$\rho = (L/R) / (L/R + RTT) = 0,03747 = \textcolor{red}{3,74\%}$$

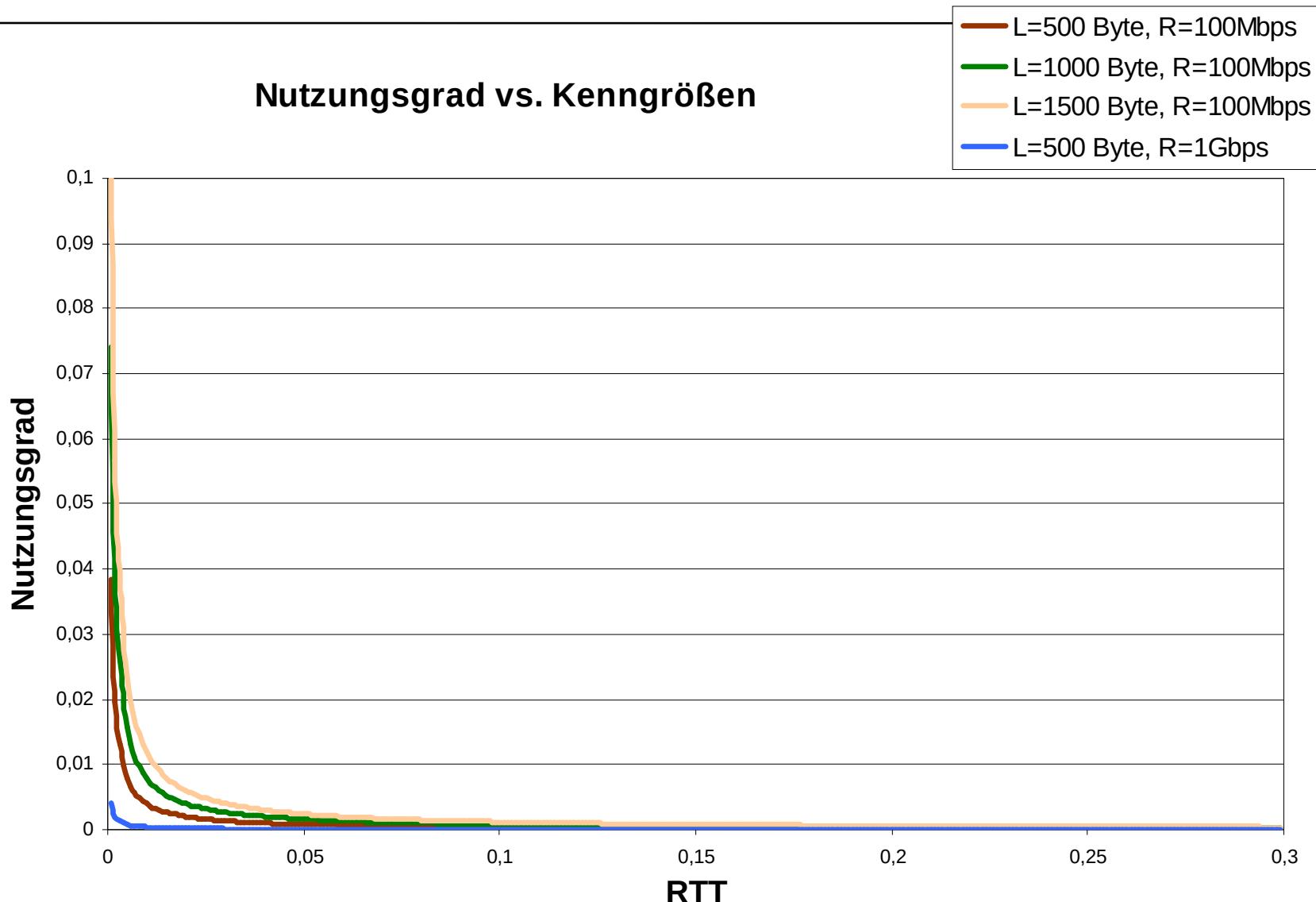
Beispiel: Kommunikation mit Kalifornien

$$R = 100 \text{ Mbit/s}$$

$$L = 1460 \text{ Byte} \quad L/R = 116,8 \mu\text{s}$$

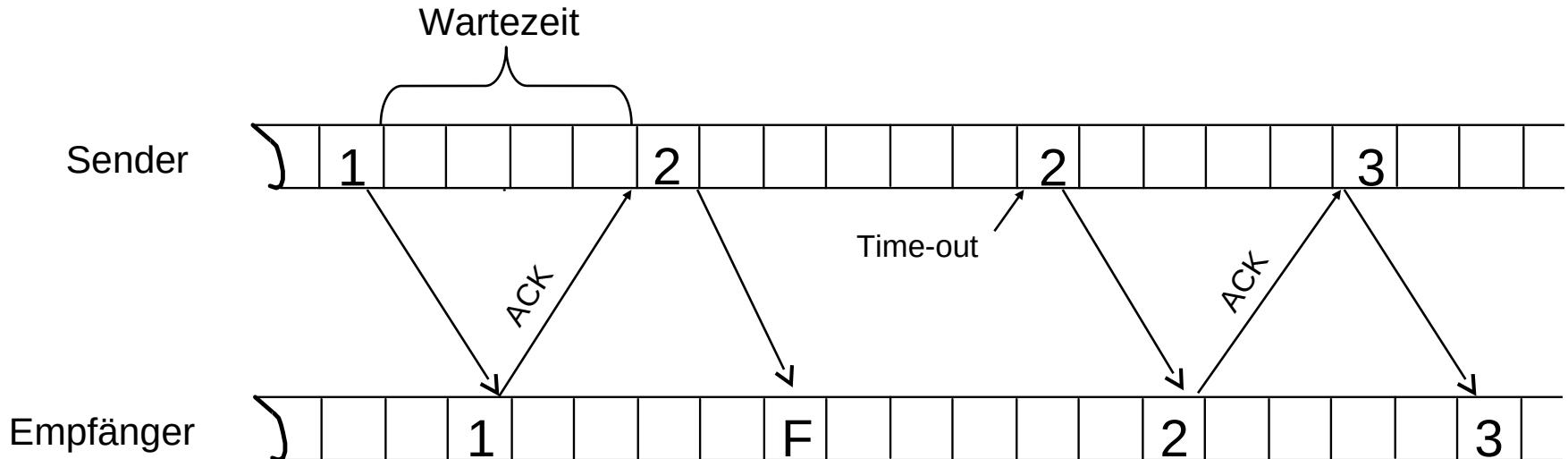
$$RTT = 200 \text{ ms} \quad \rho = (L/R) / (L/R + RTT) = \textcolor{red}{0,05837\%}$$

Schlechter Nutzungsgrad der Verbindungskapazität



Eins nach dem anderen: Send-and-Wait

- Sender überträgt ein Segment und startet einen Timer
 - Trifft eine Quittung ein: sende das nächste Segment
 - Tritt ein Timeout ein (d.h. läuft der Timer ab): wiederhole das vorherige Segment
- Lange Wartezeiten zwischen den Segmenten! Dadurch Verschwendungen von Übertragungskapazität!

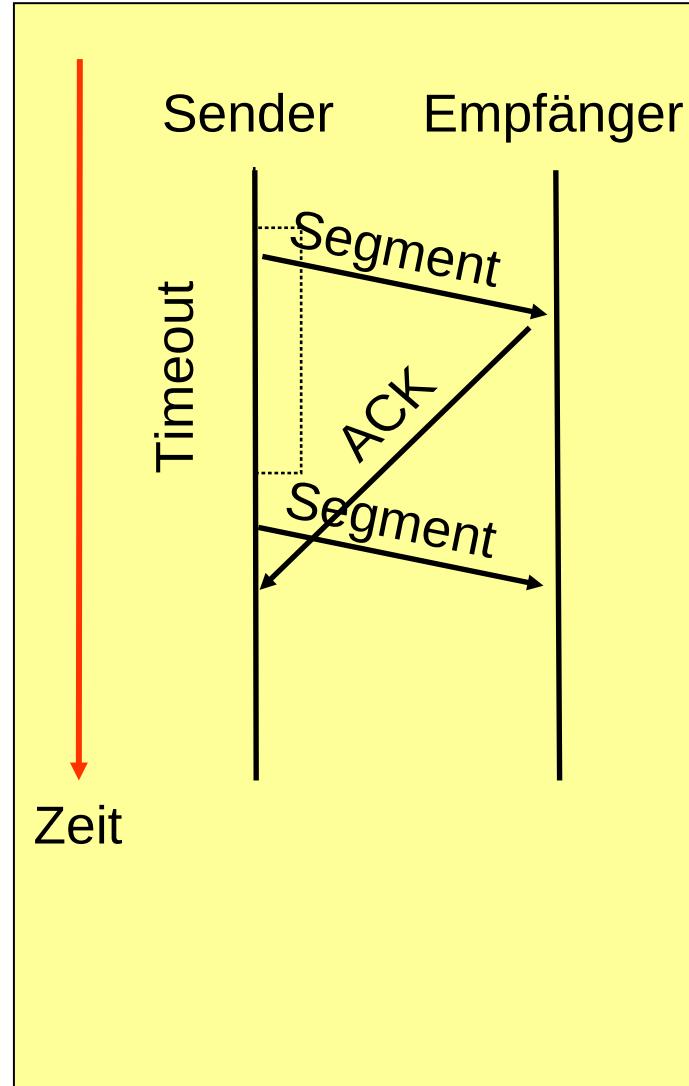


Verfahren bei Fehler/Sonderfällen

Welche Ereignisse können den geschilderten Ablauf stören?

Verfahren bei Fehler/Sonderfällen

Welche Bestätigung
haben wir hier
empfangen?



Welches Segment
haben wir hier
empfangen?

Konsequenz

- Wir benötigen eine eindeutige Identifikation der Nachricht
- Damit müssen wir die Segmente und die dazugehörigen Bestätigungen mit einer eindeutigen Nummer versehen

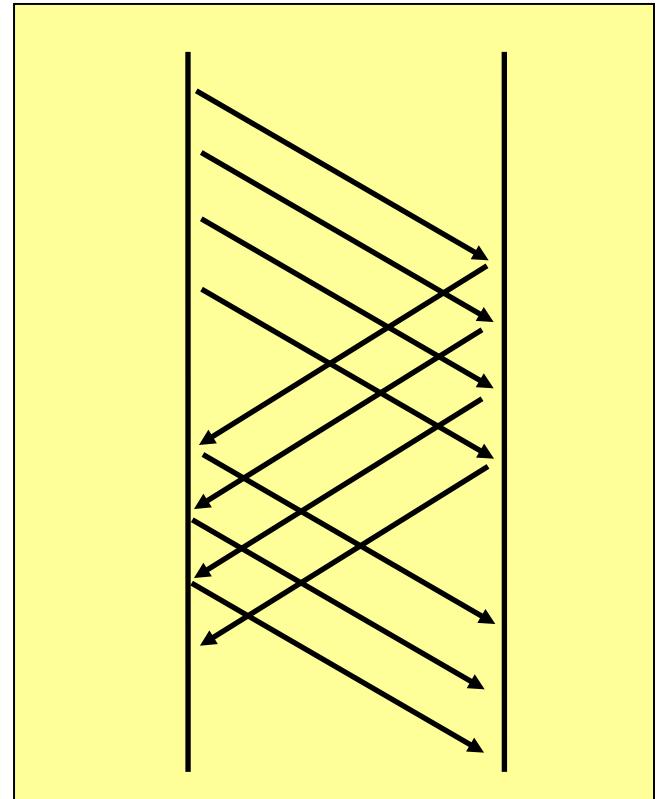
Lösungsansätze:

- Wir verwenden den Byte-Offset im Stream in der Art, dass für das aktuelle Segment angezeigt wird, an welcher Position das 1. Byte liegt, oder wir nummerieren die Nachrichten durch
- Zyklisch, da Zahlenraum begrenzt
- Bestätigungen verweisen entweder auf das **nächste zu erwartende** Byte oder auf die **empfangene** oder als nächste zu erwartende Nummer

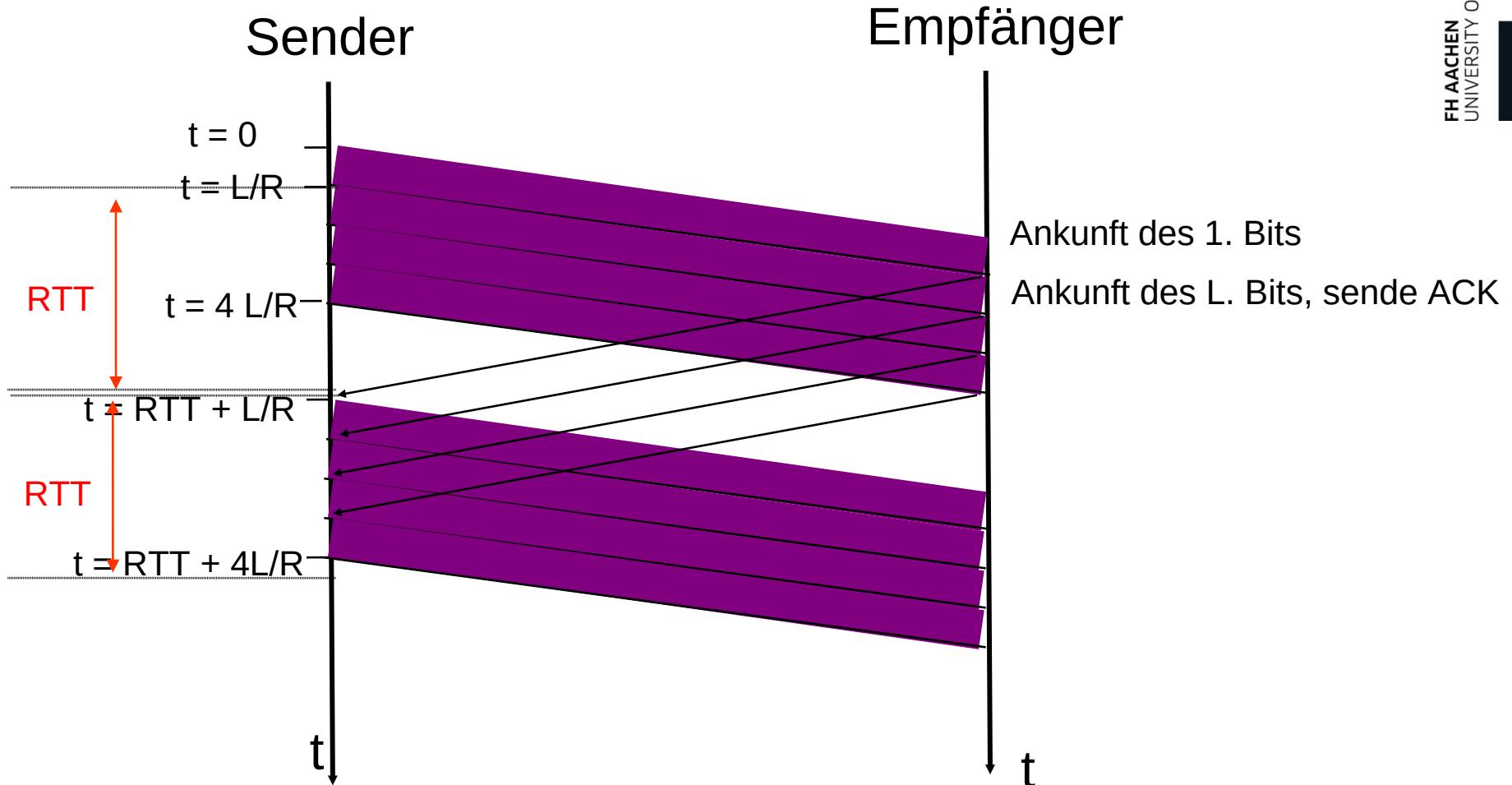
Wie kann man das Netz besser auslasten?

Pipelining:

- Senden ohne vorheriges ACK
- **Fensterbreite** beim Sender:
Anzahl der Segmente, die ohne Bestätigung gesendet werden
- **Fensterbreite** beim Empfänger:
Anzahl der Segmente, die auch bei Lücken oder Fehlern zwischengespeichert werden

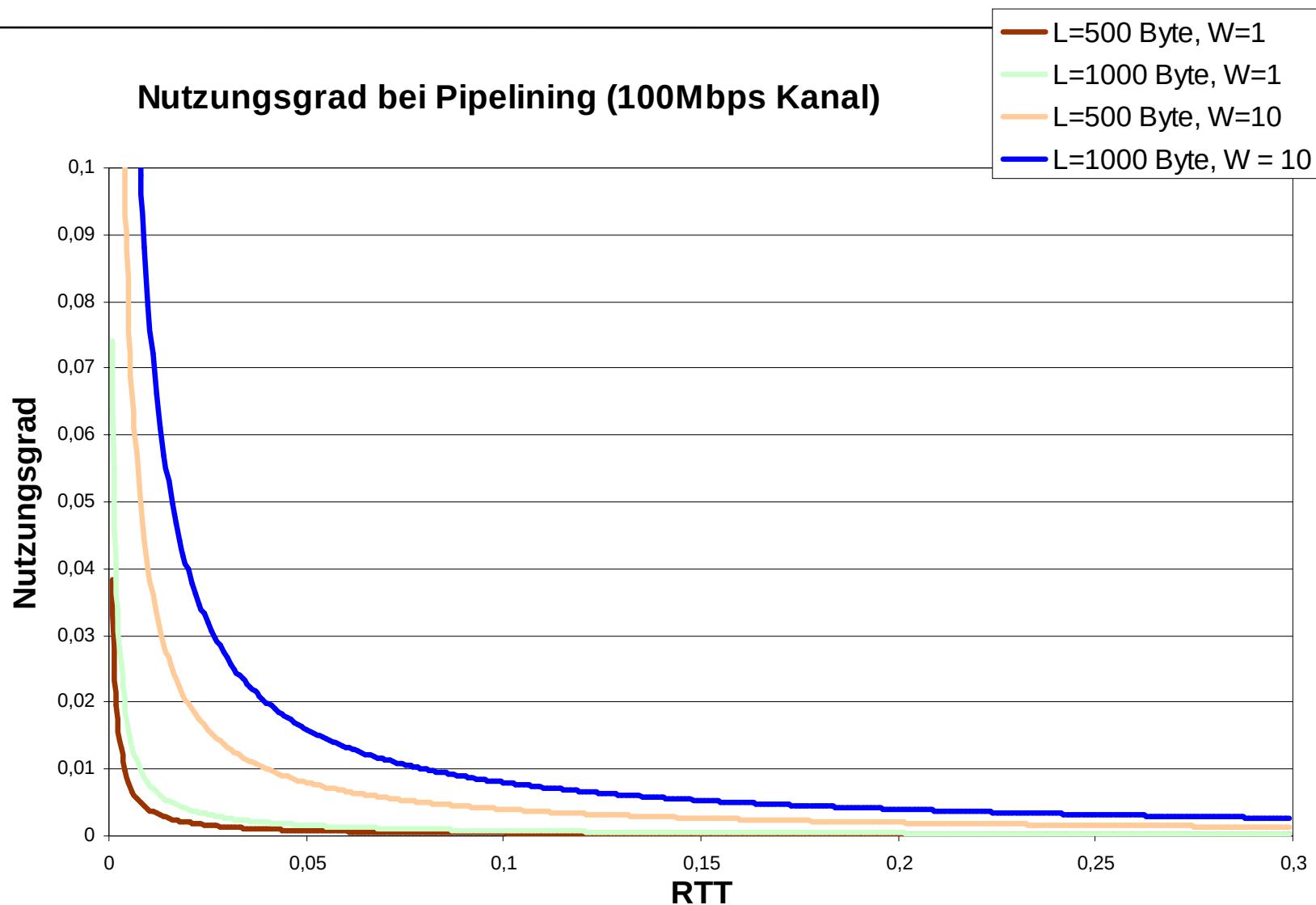


Flusskontrolle folgt dem Pipelining-Prinzip



$$\rho = (4L/R) / (4L/R + (RTT + L/R - 4L/R)) = (4 L/R) / (L/R + RTT)$$

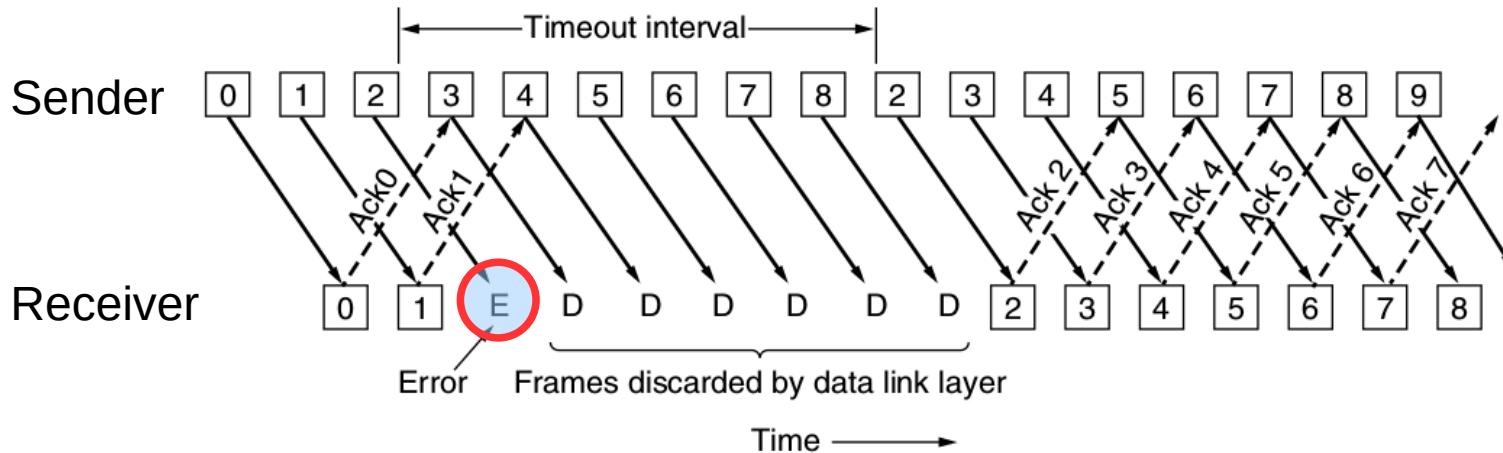
Schlechter Nutzungsgrad der Verbindungskapazität



Wir erhöhen hier aber die Komplexität im Fehlerfall

Einfachste Lösung: **Go-Back-N**

Go-Back-N



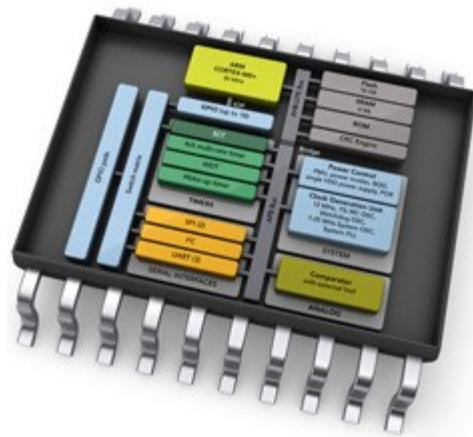
- Der Empfänger sendet nach dem Fehler kein ACK mehr
- Der Sender läuft dadurch in ein Timeout und fängt wieder ein Segment nach dem letzten bestätigten Segment wieder an
- Alle direkt nach dem Fehler versendeten Segmente werden verworfen
- → Der Sender hat eine „Fensterbreite“ > 1
- → Der Empfänger hat eine „Fensterbreite“ von 1

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Sliding Window Verfahren - Schiebefensterverfahren

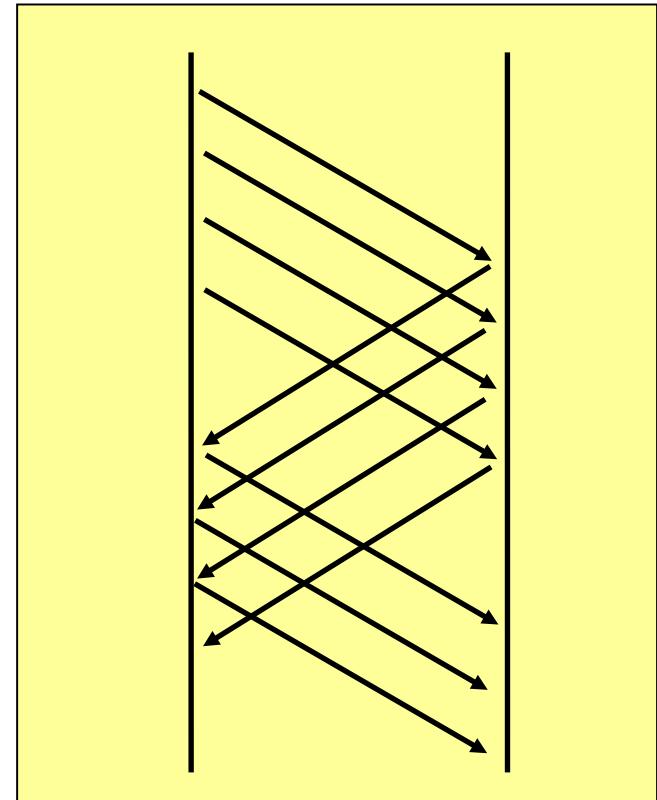
Alle auf einmal

- Segmente müssen auf Sender- und Empfängerseite gepuffert werden
 - Sender muss ein Segment speichern, bis es bestätigt wurde
 - Empfänger muss ein Segment speichern, bis die Anwendung es ausliest und verarbeitet
 - Senden aller Segmente auf einmal:
 - > Große Puffer im Betriebssystem nötig
 - > Netz wird spontan sehr stark belastet
- **Prinzip der Flusskontrolle**
 - Erlaube das Senden mehrerer (n) Segmente ohne Quittierung
 - Der Empfänger kann den Wert von n an seinen Puffer anpassen
 - Die aktuelle Netzauslastung wird auch mit berücksichtigt

Die Idee des Sliding-Window-Protokolls

Pipelining:

- Senden ohne vorheriges ACK
- Fensterbreite beim Sender:
Anzahl der Segmente, die ohne Bestätigung gesendet werden
- Fensterbreite beim Empfänger:
Anzahl der Segmente, die auch bei Lücken oder Fehlern zwischengespeichert werden



Fensterbreite und Segment- bzw. Sequenznummern

- Verwendung eines „Fensters“ der Größe W (in Segmenten oder in Bytes)
 - Sender und Empfänger vereinbaren ein **Übertragungsfenster** (= Größe des freien Pufferspeichers)
 - Sender nummeriert die zu versendenden Bytes/Segmente des Datenstroms z.B. mit $0, 1, \dots, \text{MODULUS}-1, 0, \dots$ durch, wobei $W \leq \text{MODULUS}$ gelten muss
 - > Fenstergröße W bedeutet: der Sender darf maximal W fortlaufend nummerierte Bytes/Segmente verschicken, ohne eine Bestätigung zu bekommen
 - > Empfänger bestätigt empfangene Bytes durch Quittungen (ACK)
 - > Sender „rückt“ Fenster um die Anzahl der Segmente/Bytes vor, sobald ein ACK eintrifft, und darf neue Segmente verschicken

Das Prinzip der Sliding-Window-Protokolle

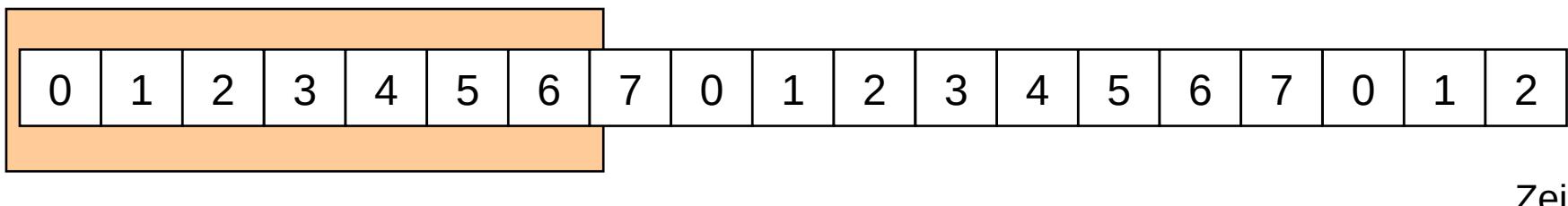
- Jedes Segment wird durch eine im Transport-Header gespeicherte Sequenznummer durchlaufend gekennzeichnet
 - Zyklische Nummerierung der Segmente
(z.B. 0, 1, 2, ..., $MODULUS-1$, 0, ...; wobei $W < MODULUS$)
 - Wir können alternativ auch die Position des ersten Bytes des Segments im Stream nehmen, besser bei variabel langen Segmenten
 - Damit gibt es die Fenstergrenzen MIN, MAX mit
 - $W = MAX - MIN + 1$ für $MIN < MAX$ und
 - $W = MODULUS - MIN + MAX + 1$ für $MIN > MAX$
- Empfänger bestätigt den korrekten Empfang durch Quittungen (ACK)
- ACKs beziehen sich auf eine Sequenznummer
 - Nummer des empfangenen Segments
 - Nummer des als nächstes erwartetes Segment
 - Offset des als nächstes erwartetes Segment im Stream
- Übertragungsfenster des Senders wird bei Empfang eines ACK angepasst (MIN und MAX werden angepasst), was faktisch ein Verschieben des Fensters bedeutet

Sliding-Window-Protokoll

Beispiel (3-Bit Sende-/Empfangsnummer – MODULUS = $W+1$)

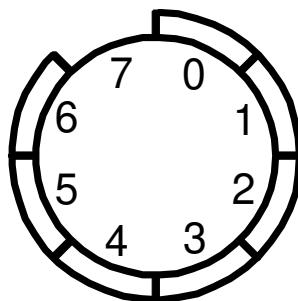
- Der Einfachheit halber nummerieren wir Segmente durch Sequenznummern mit 3 Bits: $m = 8$ mögliche Kombinationen
- Übertragungsfenster muss **kleiner** sein, z.B. können die Stationen eine Fenstergröße $W=7$ mit $1 \leq W < m$ vereinbaren
- Das Fenster beschränkt die Anzahl der unbestätigten Byte, die zu einem Zeitpunkt unterwegs sein dürfen (hier max. 7 wegen $W = 7$)
- Bei Empfang einer Quittung wird das Fenster entsprechend verschoben
- Bytes werden fortlaufend modulo m nummeriert (für $m = 8$ also Nummern von 0 bis 7)

ACK 0 ACK 1 ACK 2

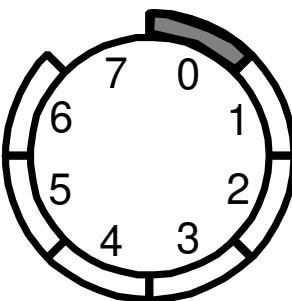


Sliding Window - andere Darstellung

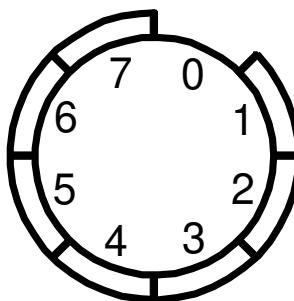
- Beispiel (für 3-Bit Sende-/Quittungsnummer)
 - 3 Bit Sende-/Quittungsnummer $\rightarrow m = 8$ mögliche Kombinationen
 - Bytes werden fortlaufend modulo m nummeriert
 - Stationen vereinbaren Fenstergröße W mit $1 \leq W < m$, z.B. $W = 7$
 - W beschränkt die Anzahl der unbestätigten Bytes, die zu einem Zeitpunkt unterwegs sein dürfen (hier max. 7)
 - Bei Empfang einer Quittung wird Fenster entsprechend verschoben



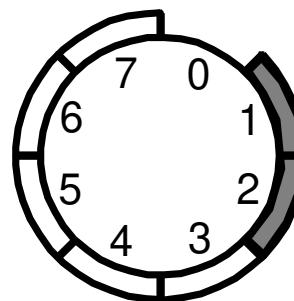
Station sendet
Byte 0 - 6



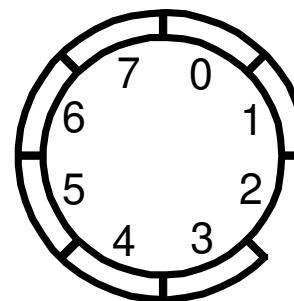
Station erhält
Quittung für
Byte 0 (d.h.
ACK 1)



Station verschiebt
Fenster um 1,
sendet Byte 7



Station erhält
Quittung für 1,
2



Station verschiebt
Fenster um 2,
sendet Byte 0, 1

Wie groß sollte das Übertragungsfenster sein?

Basis: Berechnung des Nutzungsgrades aus letzter Vorlesung

$$\rho = \#seg * (L/R) / (L/R + RTT)$$

$$R\rho = \#seg * L / (L/R + RTT)$$

$$R\rho = \text{erzielbare Bandbreite} = \mathbf{B}$$

$$\#seg * L = \text{Fensterbreite in bit} = \mathbf{W}$$

Es gilt $L/R \ll RTT$ (siehe vorherige Beispiele)

$$\mathbf{B} \leq \mathbf{W} / RTT \quad \text{bzw.} \quad \mathbf{W} \geq \mathbf{B} * RTT$$

Wie groß sollte das Übertragungsfenster sein?

Das Bandwidth-Delay-Produkt als Grundgesetz!

- Sei w die Größe des Übertragungsfenster in **Bits**:
 - Wir übertragen maximal ein Fenster pro RoundTrip
 - Erreichbarer Durchsatz $\leq w / \text{RTT}$
 - Bandwidth-Delay-Produkt: $w \geq \text{Bandwidth} * \text{Delay}$

Beispiel 1:

- Die Anwendung möchte eine Rate von 10Mbps erreichen.
Wir kennen die Round-Trip-Zeit, die bei 8ms liegt. Die Segmente seien 500 Byte groß. Wie groß muss w sein?
 - > Antwort: $w \geq 10 \text{ Mb/s} * 0,008\text{s} = 80.000 \text{ Bit} = 10.000 \text{ Byte}$
 - > In Segmenten ausgedrückt: 20 Segmente a 500 Byte

Wie groß sollte das Übertragungsfenster sein?

Das Bandwidth-Delay-Produkt als Grundgesetz!

- Sei w die Größe des Übertragungsfenster in **Bits**:
 - Wir übertragen maximal ein Fenster pro RoundTrip
 - Erreichbarer Durchsatz $\leq w / \text{RTT}$
 - Bandwidth-Delay-Produkt: $w \geq \text{Bandwidth} * \text{Delay}$

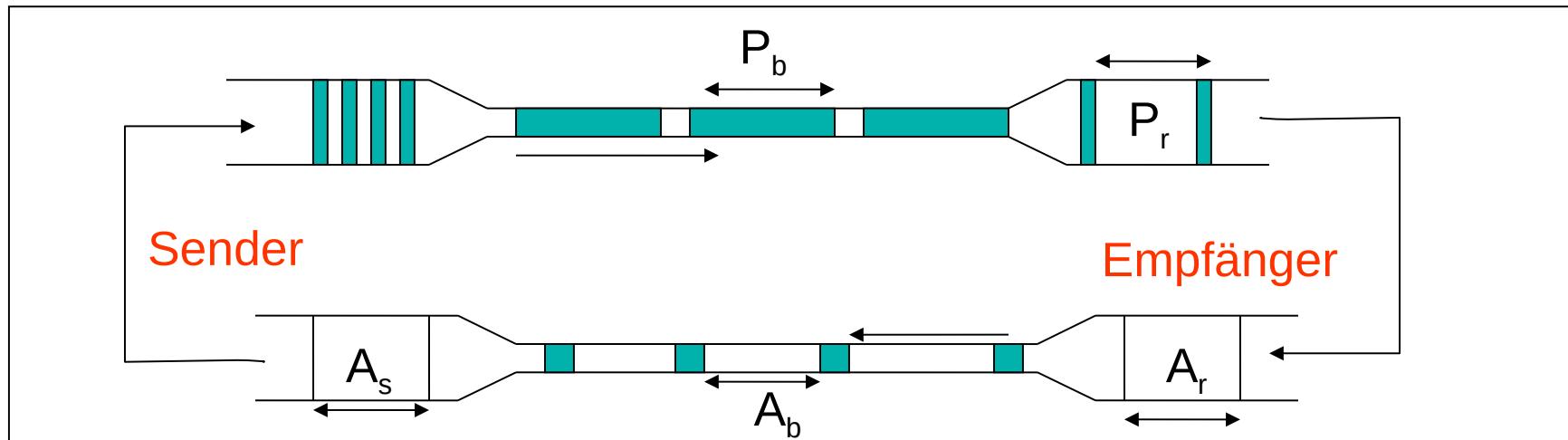
Beispiel 2:

- Die Anwendung möchte eine Rate von 1Gbps erreichen. Wir kennen die Round-Trip-Zeit, die bei 240ms liegt. Die Segmente seien 1000 Byte groß.
Wie groß muss w sein?

- > Antwort: $w \geq 1 \text{ Gb/s} * 0,24\text{s} = 240.000.000 \text{ Bit} = 30.000.000 \text{ Byte}$
 - > In Segmenten ausgedrückt: 30.000 Segmente a 1.000 Byte

Sliding Window liefert Flusskontrolle

- Feedback-Informationen zwischen Empfänger und Sender reguliert die Senderate der Quelle
- Flusskontrolle
- Der Fortschritt des Sliding-Windows passt sich den Fähigkeiten des Netzes an
- Im „*steady state*“ wird immer dann ein Paket verschickt, wenn ein ACK empfangen wird
 - Gleichgewicht stellt sich ein



Verfahren bei Fehler/Sonderfällen

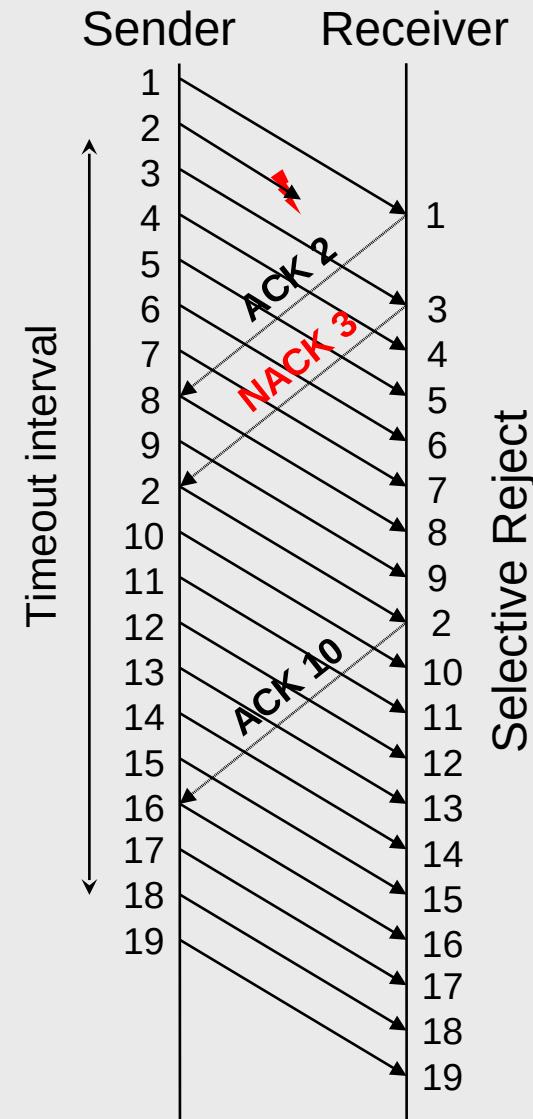
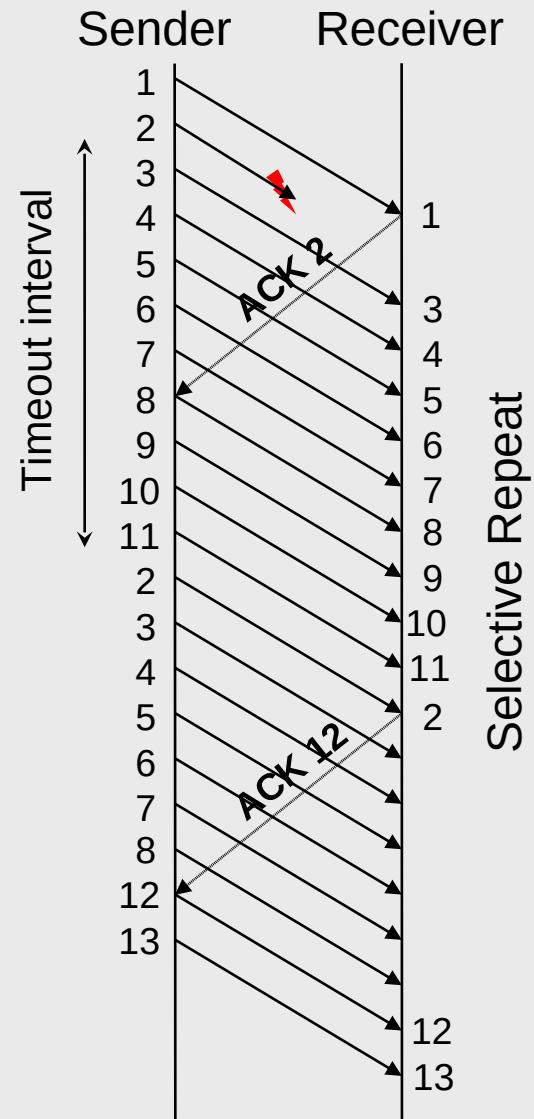
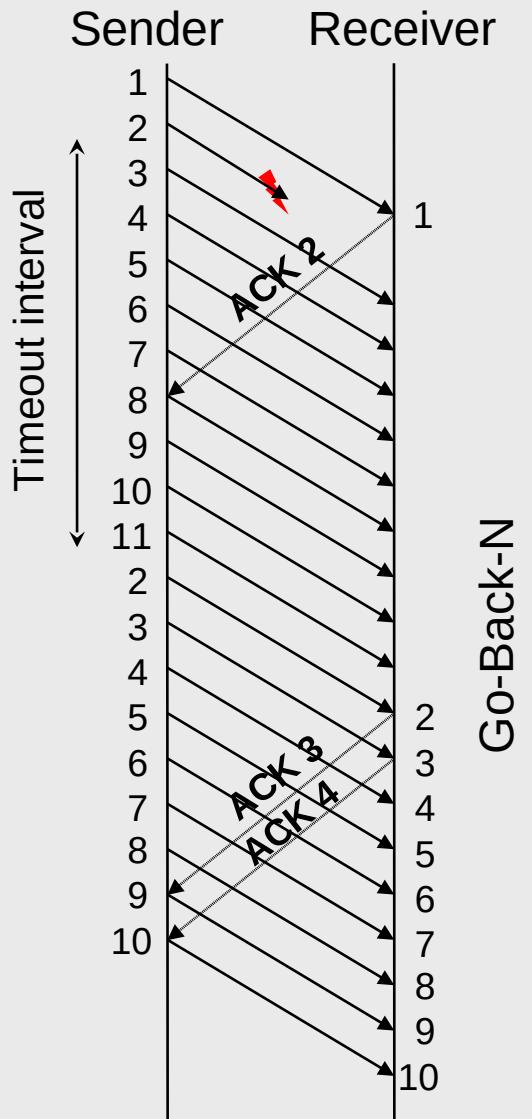
Welche Ereignisse können den geschilderten Ablauf stören?

- beim Empfänger:
 - Empfang eines out-of-order-Segments
 - > Eigentlich wurde ein anderes Segment erwartet
 - > Je nach Implementierung wird das nicht erwartete Segment einfach ignoriert, es wird dann verworfen
 - Empfang eines fehlerbehafteten Segments
 - > Keine Reaktion (Der Sender wird irgendwann eine erneute Übertragung vornehmen, da ja keine Bestätigung verschickt wurde)
 - > Falls unterstützt: Verschicken einer „negativen“ Bestätigung oder einer Nachricht, die dies kommuniziert (als Indiz für einen möglichen Paketverlust)

Ereignisse

- beim Sender:
 - Timeout -> Worauf bezieht sich dieser?
 - Empfang einer nicht erwarteten Bestätigung (out-of-order) ACKs
 - Empfang einer negativen Bestätigung(falls es die gibt)
 - Reaktion
 - Identifikation und erneutes Verschicken des Segments
 - (**Go-Back-N**): Keine Kenntnis über etwaig bereits empfangene Segmente, wir übertragen ab dem Fehlerfall alles neu; Empfänger ignoriert nicht erwartete Segmente
 - (**Selective Repeat**): Mittelbare Kenntnis über etwaig bereits empfangene Segmente, Empfänger puffert nicht erwartete Segmente, so dass sich bei Neuübertragung der „Lücke“ das Fenster ruckartig verschiebt
 - (**Selective Reject**): Unmittelbare Kenntnis über unerwartete Ereignisse und sofortige Reaktion zur Behebung
 - Anmerkung: Alle Segmente müssen bis zum Empfang einer Bestätigung beim Sender im Betriebssystem gespeichert bleiben

Flusskontrolle: Varianten



Kumulative Bestätigungen als Kompromis

Idee der kumulativen Bestätigungen:

Eine Bestätigung zeigt an, dass alle Daten bis zu der verwendeten Nummer/Byte Position korrekt empfangen wurden

Kommt ein Out-of-Order-Segment an, so wird demnach eine Bestätigung verschickt, die schon einmal verschickt wurde ("ich warte immer noch auf x")

Speichert der Empfänger außerhalb der Reihenfolge empfangene Segmente zwischen, dann bedeutet ein „Lückenschluss“, dass dieser einen „Sprung“ bei den Bestätigungen bedingt

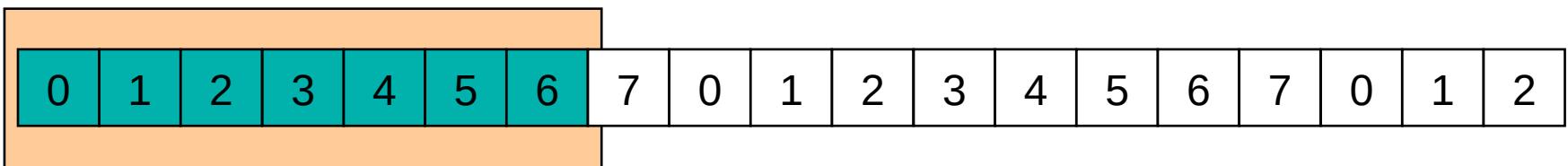
Sliding-Window-Protokoll mit kumulativen ACKs

Beispiel (3-Bit Sende-/Empfangsnummer mit MODULUS = W+1)

- Bestätigungen werden kumuliert verschickt werden
- Bei Empfang einer Quittung wird das Fenster entsprechend verschoben
- Bytes werden fortlaufend modulo m nummeriert (für $m = 8$ also Nummern von 0 bis 7)

ACK 0 ACK 2

ACK 6



Kumulative Bestätigungen als Ersatz für NACKs

Wir wissen:

Kommt ein Out-of-Order-Segment an, so wird demnach eine Bestätigung verschickt, die schon einmal verschickt wurde ("ich warte immer noch auf x")

Wenn ein Segment verloren gegangen ist, dann werden alle weiteren empfangenen Segmente die gleiche Position anzeigen

Duplicate Acknowledgments: Empfang einer Bestätigung, die eine schon versendetes Feedback erneut gibt: ("ich warte immer noch auf x")

Segmente können sich überholen, deshalb ist das nicht sofort als NACK aufzufassen

Aber: **Triple Duplicate Acknowledgement wird als NACK aufgefasst!**

Selective Repeat und Fensterbreite

Beispiel:

Sequenznummern: 0, 1, 2, 3

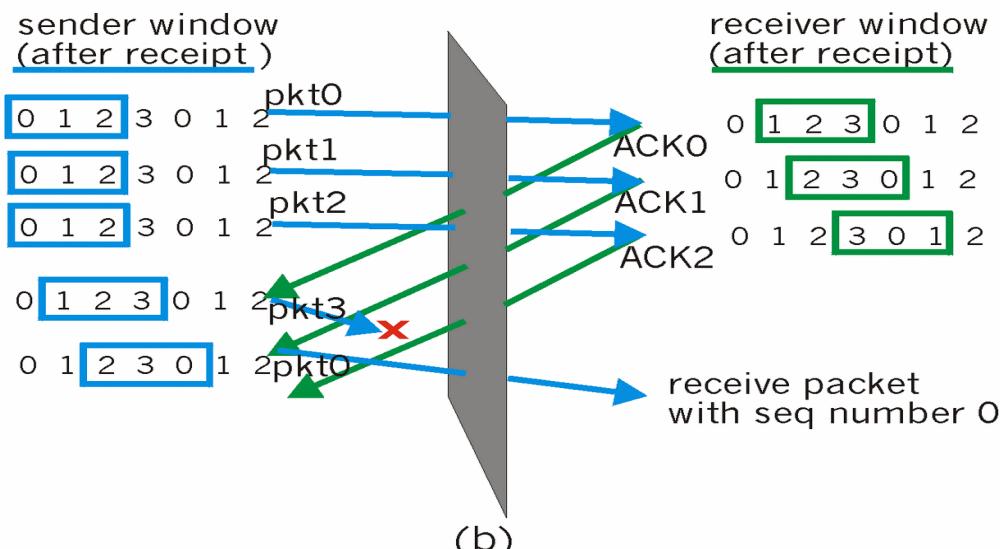
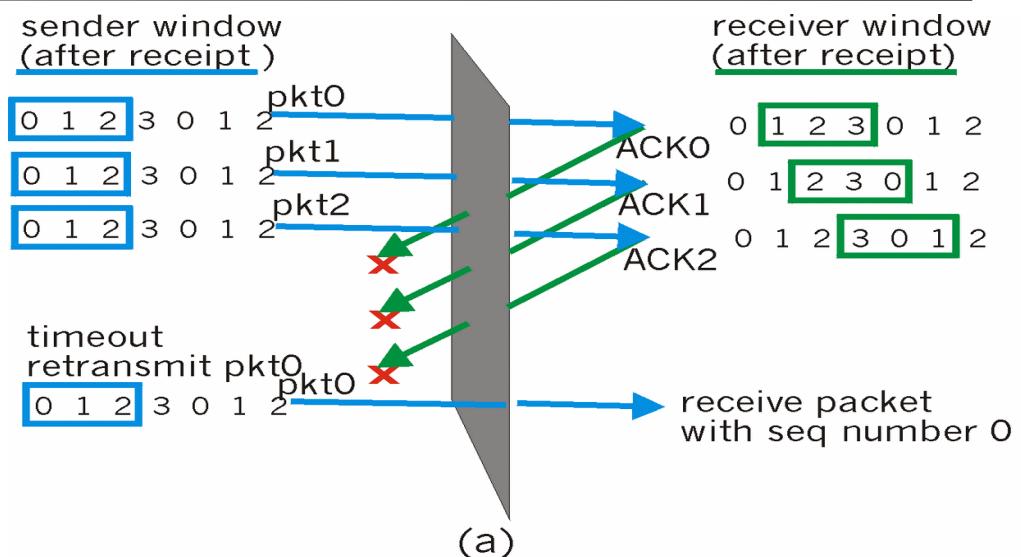
Fenstergröße = 3

Empfänger sieht keinen Unterschied zwischen den beiden Szenarios!

Fehlerhaftes Versenden neuer Daten in a

→ Bei selective Repeat darf die Fensterbreite w maximal MODULUS/2 sein!

In diesem Beispiel also 2

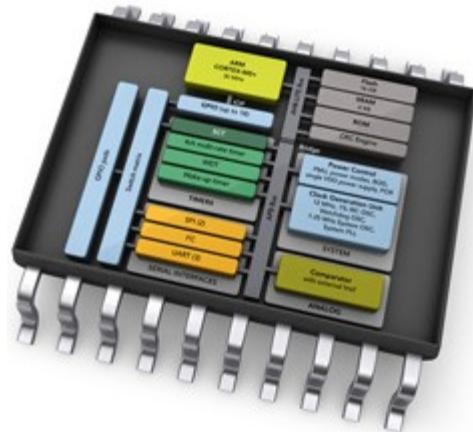


FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge

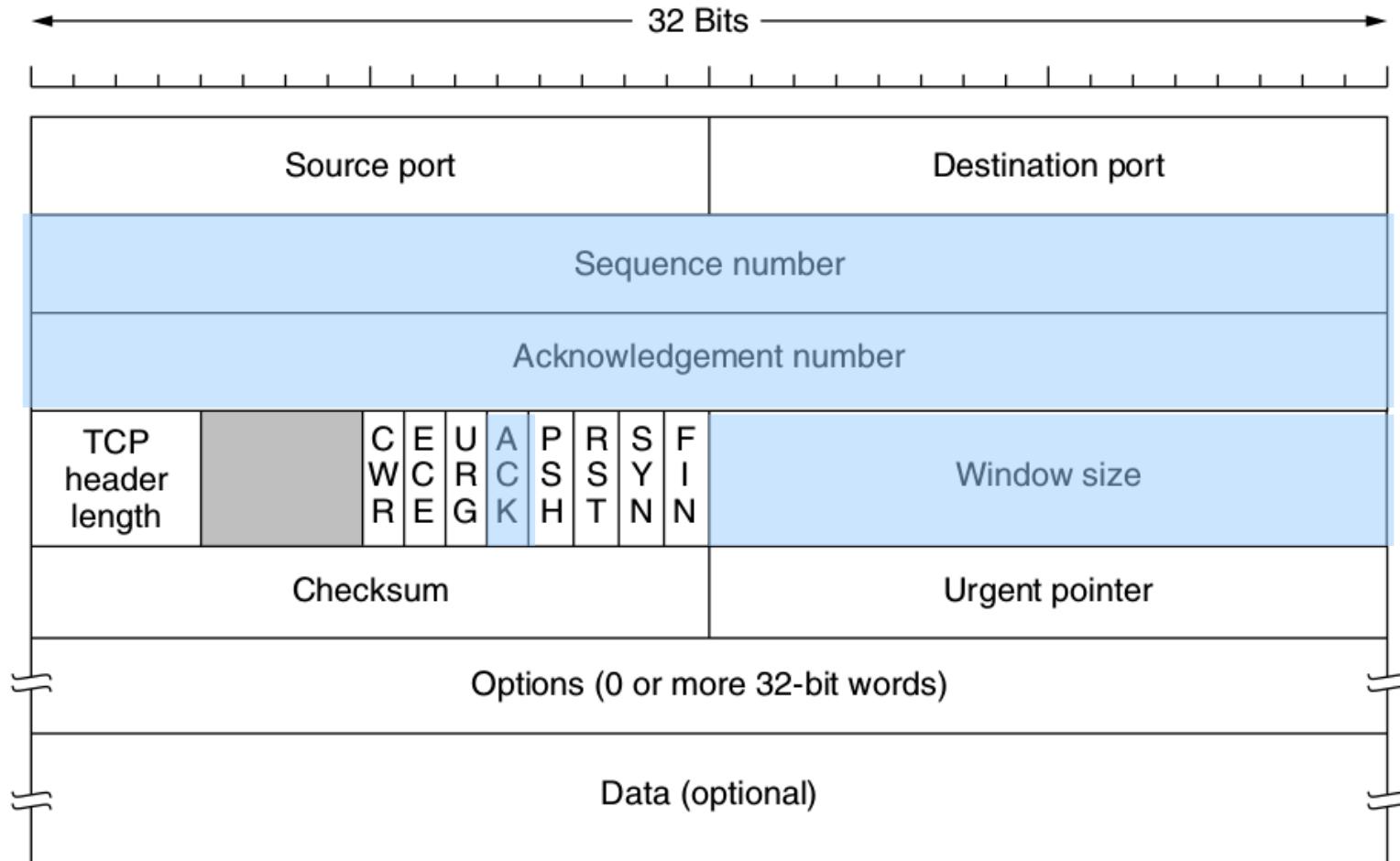


TCP: Ein Transportprotokoll mit einem dynamischen ‚Sliding-Window‘

Sliding-Window Protokoll als Grundlage

- Übertragene Segmente und ACKs müssen sich identifizieren lassen!
- Effiziente Übertragung kann nur erfolgen, wenn die Fenstergröße auf Sender- und Empfängerseite >1 ist
- Sliding-Window Protokolle können nur sinnvoll eingesetzt werden, wenn Sender und Empfänger ihre Fensterbreiten abstimmen: Situationen wie beim Go-Back-N sind zu vermeiden → es werden viele Daten auf Empfängerseite verworfen
- Fensterbreiten sind keine statischen Größen:
Das ‚Fenster‘ auf Empfängerseite wird z.B. vom ‚Abholen‘ der Daten durch die Anwendung beeinflusst!
- Mit der Fenstergröße des Senders kann die Bandbreite gesteuert werden → Flusskontrolle

TCP Header



TCP Header

- **Sequence Number**: Byte-Offset im Datenstrom
- **Acknowledgement Number**: Nächstes erwartetes Byte (Offset)
→ kumulative Bestätigung
- **CWR** : Congestion Window reduced
- **ECE** : Explicit Congestion Notification
- **URG** : Urgent Pointer in use → Event-Erzeugung auf Gegenseite
- **ACK** : Acknowledgement Number gültig
- **PSH** : PUSH → Daten auf Empfängerseite zustellen (kein Puffern)
- **RST** : RESET
- **SYN** : Aufbau der Verbindung
- **FIN** : Abbau der Verbindung
- **Window Size** : Größe des möglichen Fensters (incl. 0)
- **Urgent Pointer** : Offset zu „urgent“-Daten

Flusskontrolle in TCP

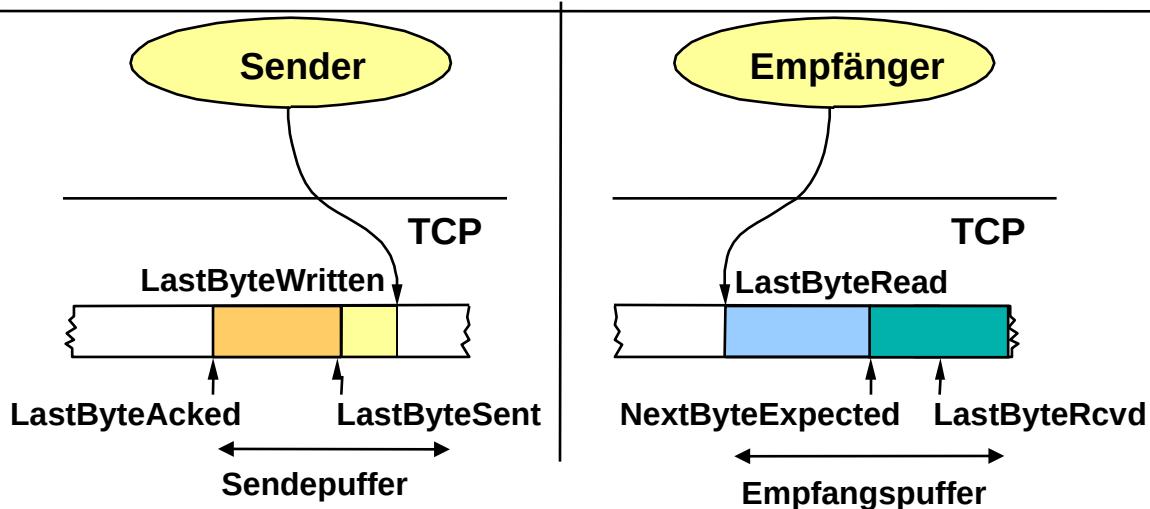
- TCP implementiert das **Sliding-Window-Protokoll**
 - Bei einer Fenstergröße von n können n Bytes verschickt werden, ohne dass ein ACK empfangen werden muss
 - Wenn der Empfang der Daten vom Empfänger bestätigt wurde, so verschiebt sich das Fenster
- Nummerierung (zyklisch mit 32-Bit)
 - Segmente werden durch ihren Byte-Offset im Stream identifiziert (Sequence Number), wobei die Startposition beim Verbindungsauftbau zufällig festgelegt wird
 - TCP verwendet kumulative ACKs: ACK $n+1$ sagt aus, dass alle Daten von der vorigen logischen Position bis zur Position n korrekt empfangen wurden und nun das Segment $n+1$ erwartet wird

Besonderheiten

- Der Empfänger puffert außerhalb der Reihenfolge empfangene Segmente, so dass die Möglichkeit besteht, Selective Repeat/Reject durchzuführen
- TCP verwendet hierbei eine **variable Fenstergröße**
 - Jede Bestätigung spezifiziert den aktuell freien Platz im Empfangspuffer (Advertised Window/Receiver Window)
 - Das Sliding Window des Senders wird somit durch die noch freie Pufferkapazität beim Empfänger beeinflusst

Wieso ist das wichtig?

Flusskontrolle in TCP: Aus dem Code



- Bedingungen für den Sender

$$\text{LastByteAcked} \leq \text{LastByteSent}$$

$$\text{LastByteSent} \leq \text{LastByteWritten}$$

Zwischenspeichern der Daten zwischen
LastByteAcked und **LastByteWritten**

$$\text{LastByteSent} - \text{LastByteAcked} < \text{AdvertisedWindow}$$

$$\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})$$

- Bedingungen für den Empfänger

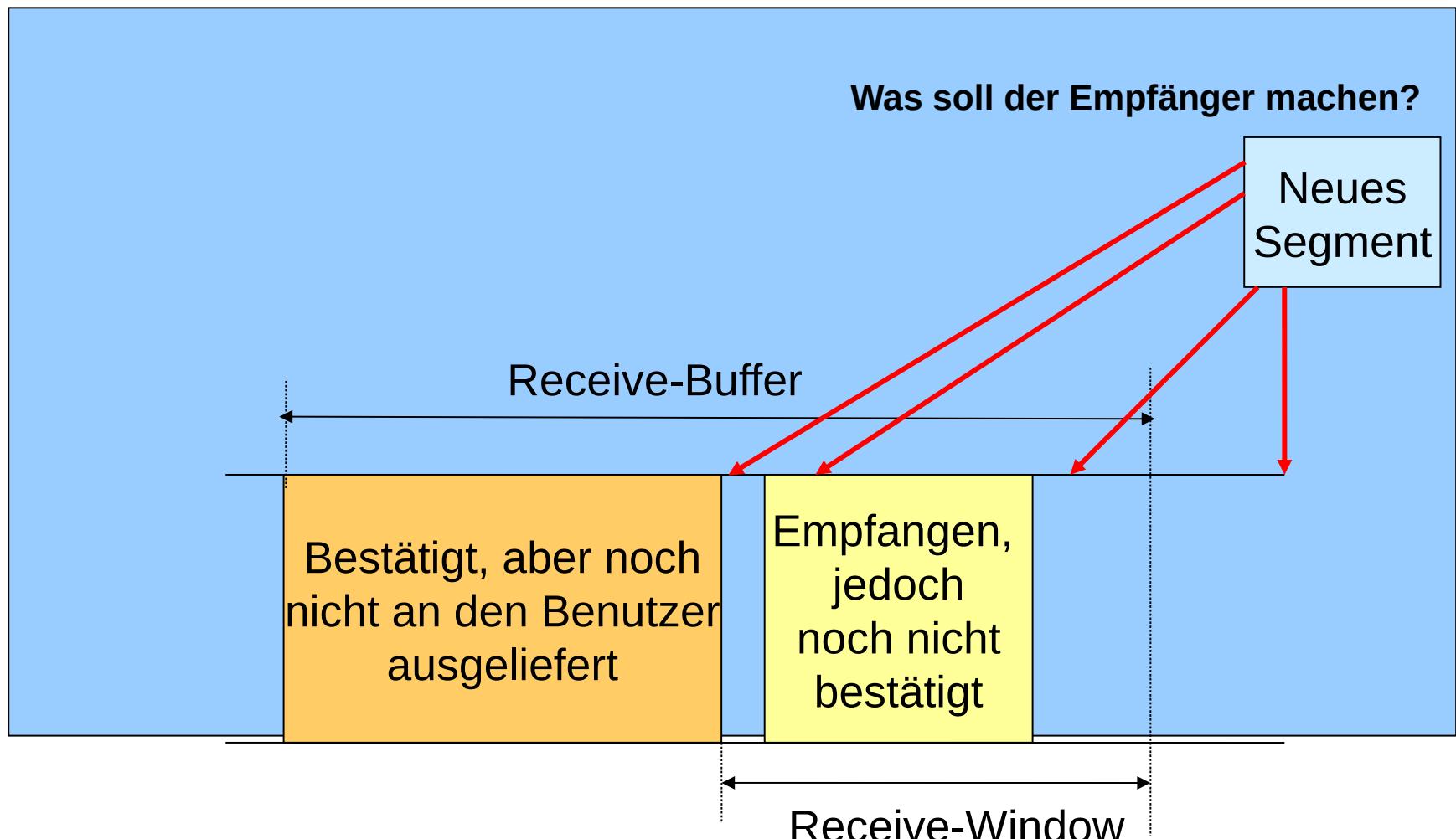
$$\text{LastByteRead} < \text{NextByteExpected}$$

$$\text{NextByteExpected} \leq \text{LastByteRcvd} + 1$$

Zwischenspeichern der Daten zwischen
LastByteRead and **LastByteRcvd**

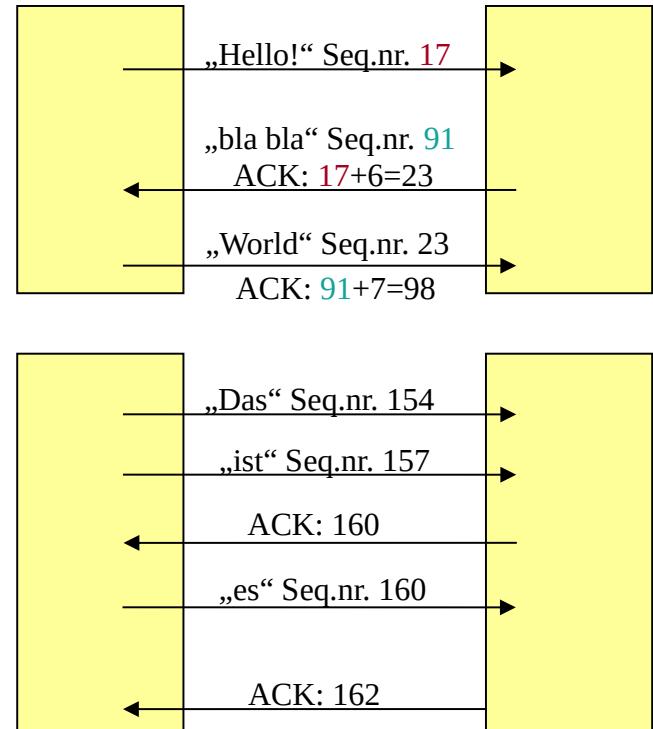
$$\text{AdvertisedWindow} = \text{Empfangspuffer} - ((\text{NextByteExpected} - 1) - \text{LastByteRead})$$

Flusskontrolle: Empfänger-Seite



Keine eigenständigen Bestätigungen

- Huckepack-Technik
 - Bestätigungen können auf dem Datenpaket der Gegenrichtung „reiten“
- Eine Bestätigungs Nachricht kann viele Segmente bestätigen
 - Kumulative Bestätigung
- Liegen keine Daten an, werden Acks verzögert – irgendwann aber doch verschickt
 - Annahme: Es kommen viele Segmente hintereinander
 - Delayed Acknowledgments
 - Nach 500ms **muss** ein ACK gesendet werden
 - Nach zwei vollständigen Segmenten **sollte** ein ACK gesendet werden



Reaktion des Empfängers (Delayed Ack)

Ereignis beim Empfänger	Aktion beim Empfänger
Ankunft eines Segments mit der erwarteten Sequenznummer. Alle Daten bis dahin sind schon bestätigt worden.	Delayed ACK. Warte bis zu 500ms auf das nächste Segment. Wenn dieses nicht empfangen wird, verschicke die Bestätigung.
Ankunft eines Segments mit der erwarteten Sequenznummer. Allerdings wurde noch keine Bestätigung des vorigen Segments verschickt.	Sofortiges Verschicken einer Kumulativen Bestätigung, die beide Segmente bestätigt.
Ankunft eines Segments hinter der erwarteten Segmentnummer. Es wird eine Lücke entdeckt.	Sofortiges Verschicken eines Duplicate ACK (DupACK). Dieses gibt die erwartete Segmentnummer an.
Ankunft eines Segments, das eine Lücke teilweise oder vollständig füllt.	Sofortiges Verschicken einer Bestätigung.

Transmission Control Protocol: Eine hybride Lösung

Go-Back-N/Selective Repeat Protokoll: TCP-Sender verwaltet lediglich einen Timer für das Segment, welches als nächstes bestätigt werden muss. Kommt es zu einem Time-Out, so wird, bedingt durch das Zwischenspeichern auf Empfängerseite, hier zu einem **Selective Repeat** durchgeführt.

Selective Reject: Empfänger speichert nicht nur out-of-Order Segmente im Empfangspuffer. Die meisten Versionen von TCP **emulieren** NAK-Mechanismen mittels dreifachem ACK mit gleicher Sequenznummer:

Hierdurch kann ein **erneutes Übertragen eines Segments VOR Timeout** initiiert werden (beim 3. Duplicate Acknowledgement) Wir haben somit eine Art **Selective Reject**. Die kumulative Bestätigungen ermöglichen das **Nutzen zwischengepufferter Daten** (Datenlücken können geschlossen werden)

Transmission Control Protocol: Timer

- **Retransmission Timer**

Timer wird beim Senden eines Segmentes gestartet.
Bei Ablauf: Retransmission!

- **Persistence Timer**

Timer wird nach Empfang einer Fenstergröße von 0
gestartet. Nach Ablauf Anfrage nach neuem Fenster (>0)

- **Keepalive Timer**

Rel. lange Wartezeit. Wir bei jeder Transaktion rückgesetzt.
Nach Ablauf → Abbau der Verbindung

TCP Round Trip Time und Timeout

Q: Wann sollte es ein Timeout geben?

- RTT ist die Zeit, die eine Antwort aktuell (mindestens) braucht
- Timeout sollte nicht vor dem RTT kommen
- Problem: RTT variiert
- **Zu kurz:** vorzeitiger Timeout und damit unnötige Neuübertragungen
- **Zu groß:** langsame Reaktion auf Paketverluste → geringer Datendurchsatz

Q: Wie kann die RTT geschätzt werden?

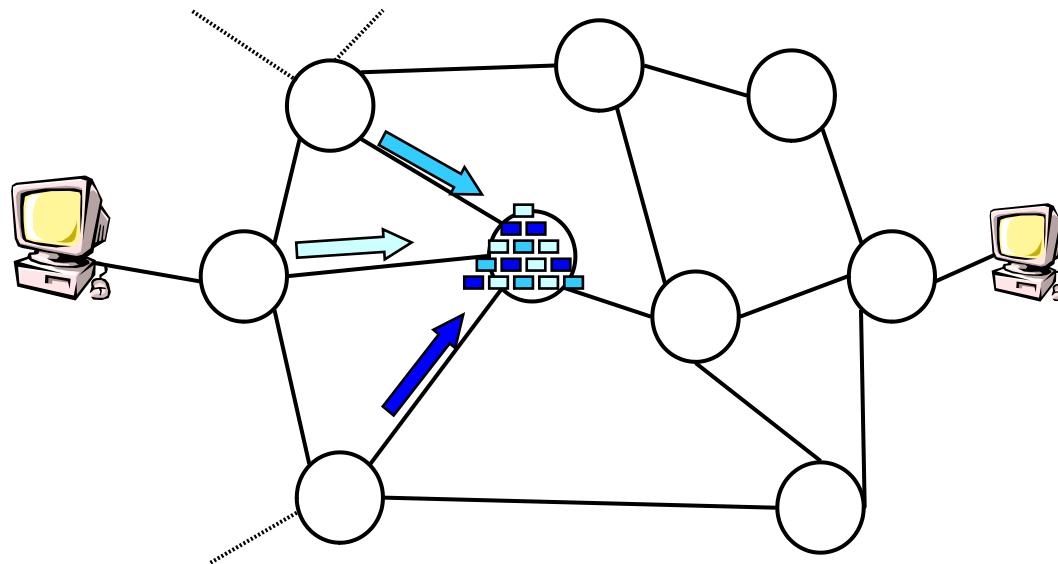
- TCP stoppt die Zeit zwischen dem Versenden eines Segmentes und dem Empfang des ACKs
- Neuübertragungen gehen hier nicht ein, Delayed Acknowledgments müssen ausgeblendet werden
- **Smoothed RTT** wird zur Schätzung der RTT geglättet
$$SRTT = \alpha SRTT + (1-\alpha) R$$
- Hierdurch ergibt sich eine Mittelung über die Vergangenheit, und nicht nur die aktuelle **Smoothed RTT**

TCP ist eigentlich noch viel mehr...

- Der Verlust von Segmenten führt ggf. zum Selective Repeat und damit zu erheblichen Neuübertragungen!
 - Einigen sich Sender und Empfänger (durch ein großes Fenster bei der Flusskontrolle) auf eine hohe Datenrate, wird das Netz ggf. stark belastet
 - Existieren viele solche Übertragungen im Netz, können Router überlastet werden. Als Resultat verwerfen sie Pakete, so dass auf TCP-Ebene keine Quittungen mehr eingehen
 - TCP wiederholt die Daten und belastet das Netz damit noch stärker
- Die Staukontrolle bei TCP berücksichtigt auch noch den **Netzzustand**
 - Slow Start, Congestion Avoidance, Fast Retransmit, Fast Recovery, Selective ACK, ...

TCP - Überlastvermeidung

- TCP ist vom zugrundeliegenden Netz getrennt
 - Großes Fenster der Flusskontrolle: hohe Datenrate, Netz stark belastet
 - Durch viele Verbindungen können Router **überlastet** werden
 - > Überlast/Verstopfung: engl.: **Congestion**
 - > Pakete werden verworfen, auf TCP-Ebene gehen keine Quittungen ein
 - > TCP wiederholt die Daten und belastet das Netz damit noch stärker



TCP: Flusskontrolle/Staukontrolle

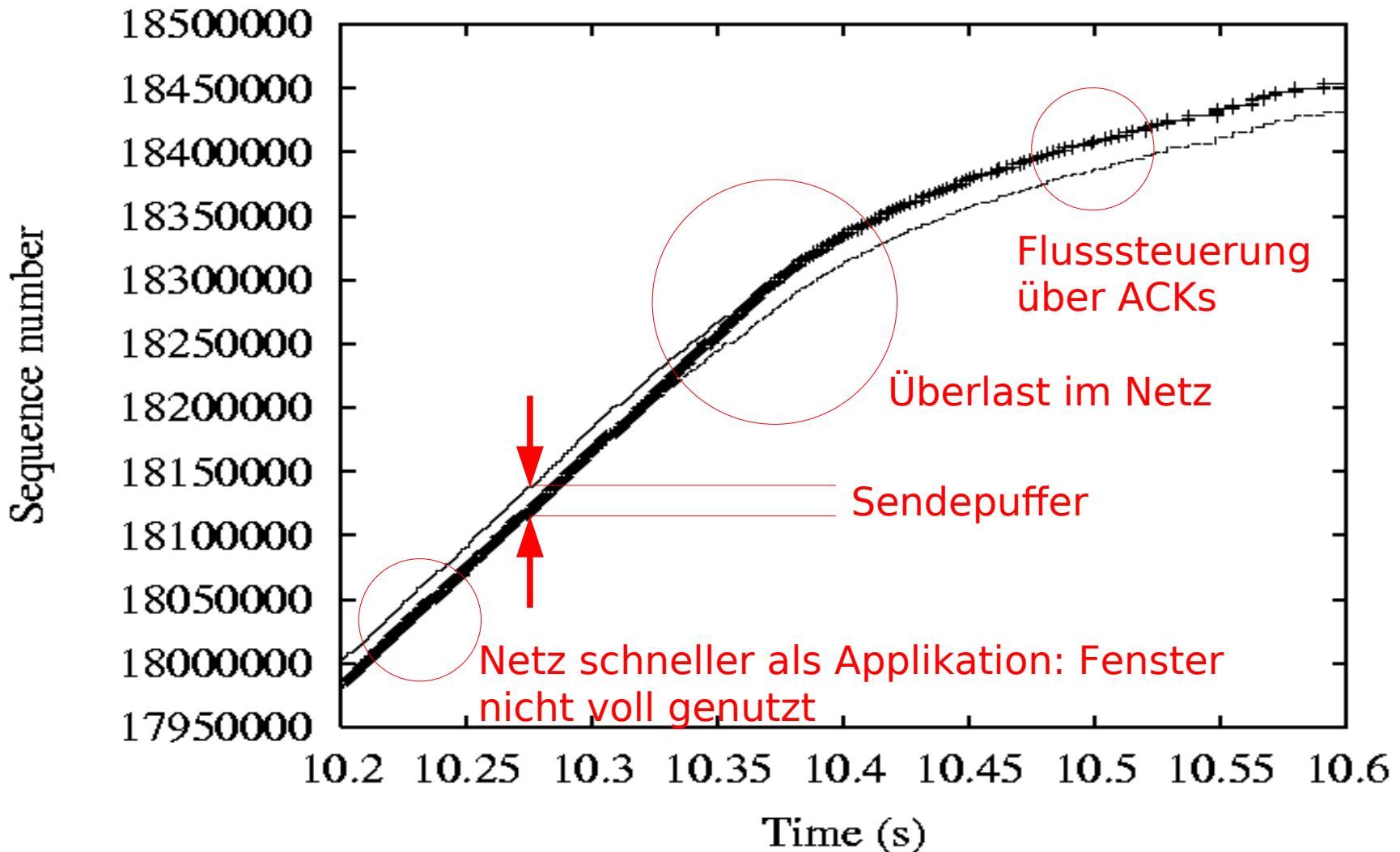
Flusskontrolle:

- basiert auf dem Sliding-Window-Protokoll
- regelt den Datenfluss zwischen den Transportdienst-Nutzern
- Ack-getriebenes injizieren von Daten

Staukontrolle:

- befasst sich mit Überlastsituationen in den Zwischensystemen (Routern)
- Überlastsituationen in Zwischensystemen führen zu Paketverlusten, so dass die entsprechenden Segmente nach einer gewissen Zeit erneut übertragen werden, bei Time-Out kommt es gar zu einem „Go-Back-N“/Selective Repeat
 - Verstärkung der Überlastsituation durch unnütze Übertragung (congestion collapse))!

Flusskontrolle in TCP: Self-Clocking



TCP: Staukontrolle

- **Überlastfenster (Congestion-Window):**

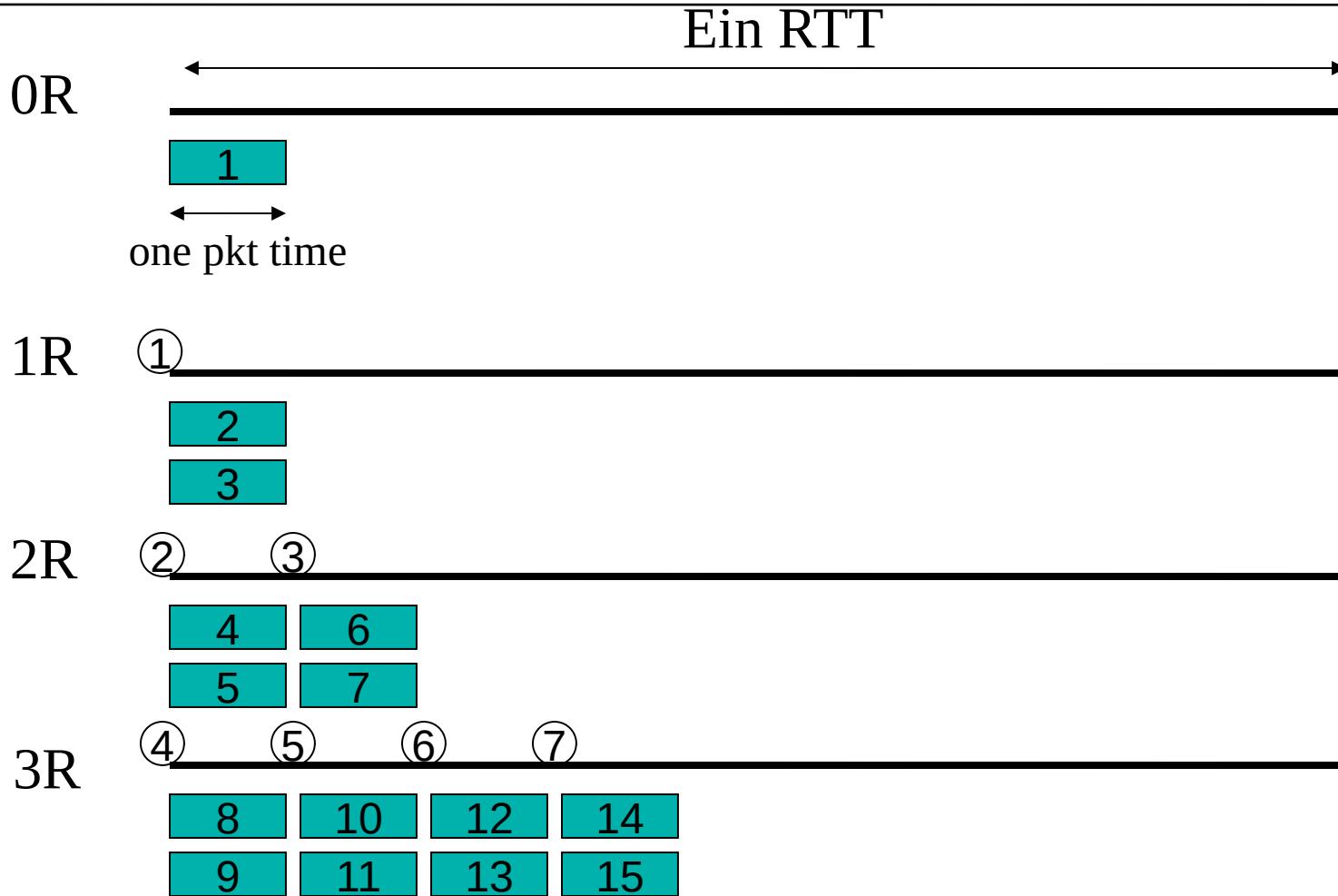
- Zusätzliche Beschränkung des verfügbaren Fensters durch ein **Congestion Window**

- Pflege eines internen Thresholds, der das Vorgehen beeinflusst (**slow-start threshold**):

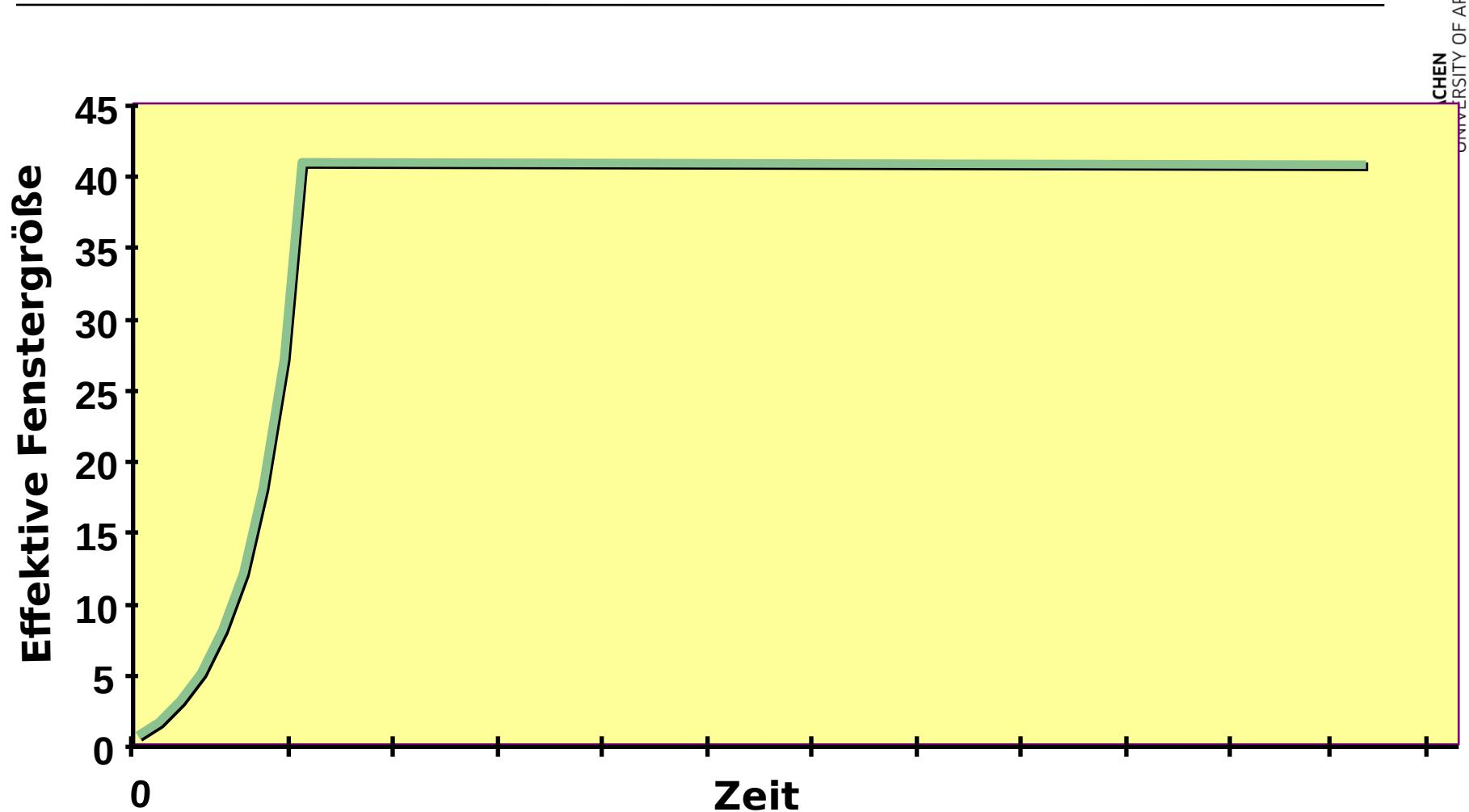
ssthresh = min (Receiver Window, Congestion Window) [Byte]

- Congestion Window bestimmt insbesondere zu Beginn die Übertragungsrate (Slow-Start)
 - Initiale Größe = 1-4 Segmente maximaler Größe
 - Größe des Überlastfensters wird in der Slow-Start-Phase bei jeder erfolgreichen Bestätigung um die Größe eines Segments erhöht
 - > erfolgreiche Übertragung eines vollen Fensters, d.h. von n Segmenten (Burst), verdoppelt somit die Größe → exponentielles Wachstum
 - > maximal bis Größe des Receiver-Windows erreicht

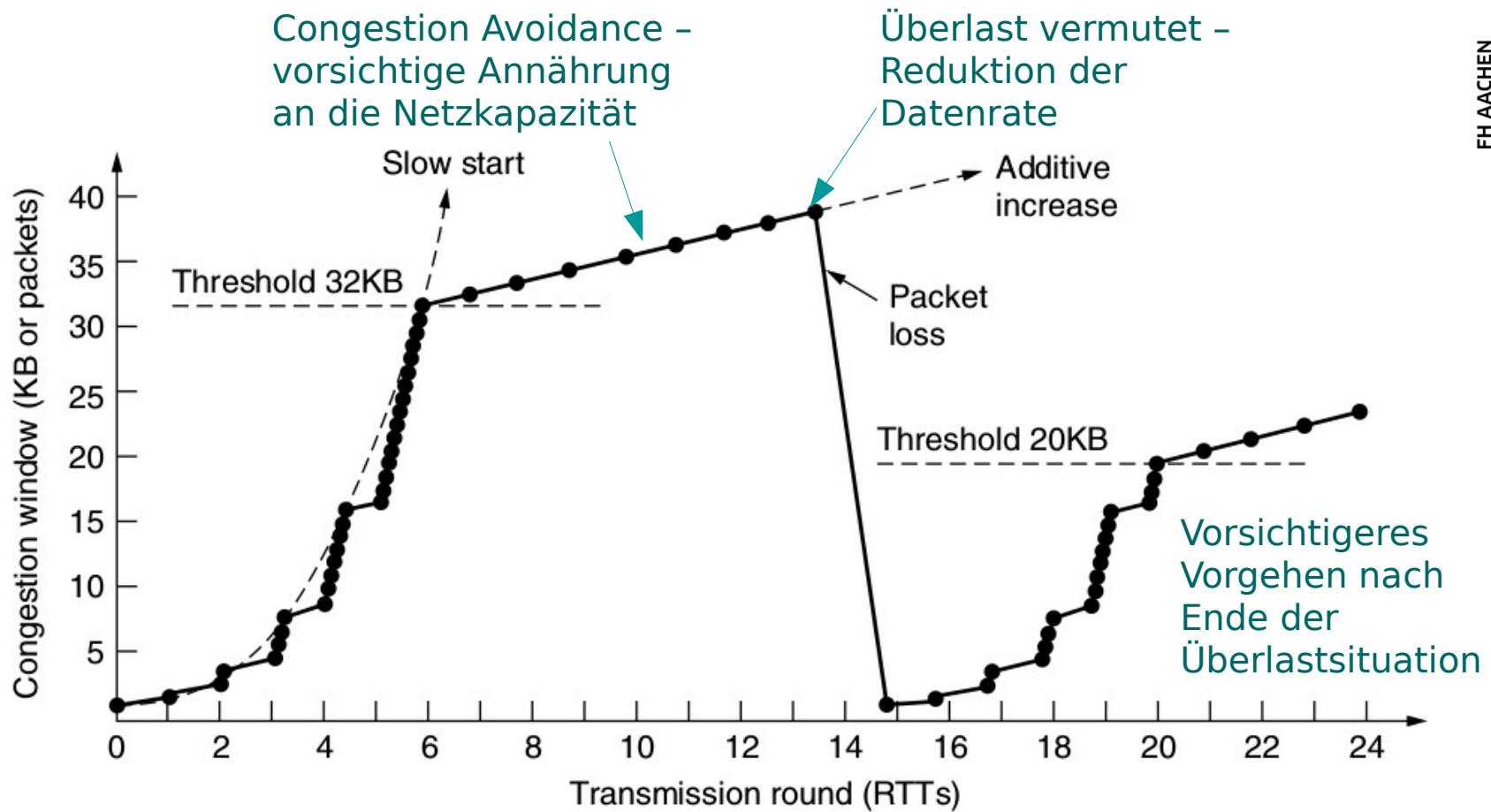
Slow-Start-Beispiel



Staukontrolle - Slow-Start

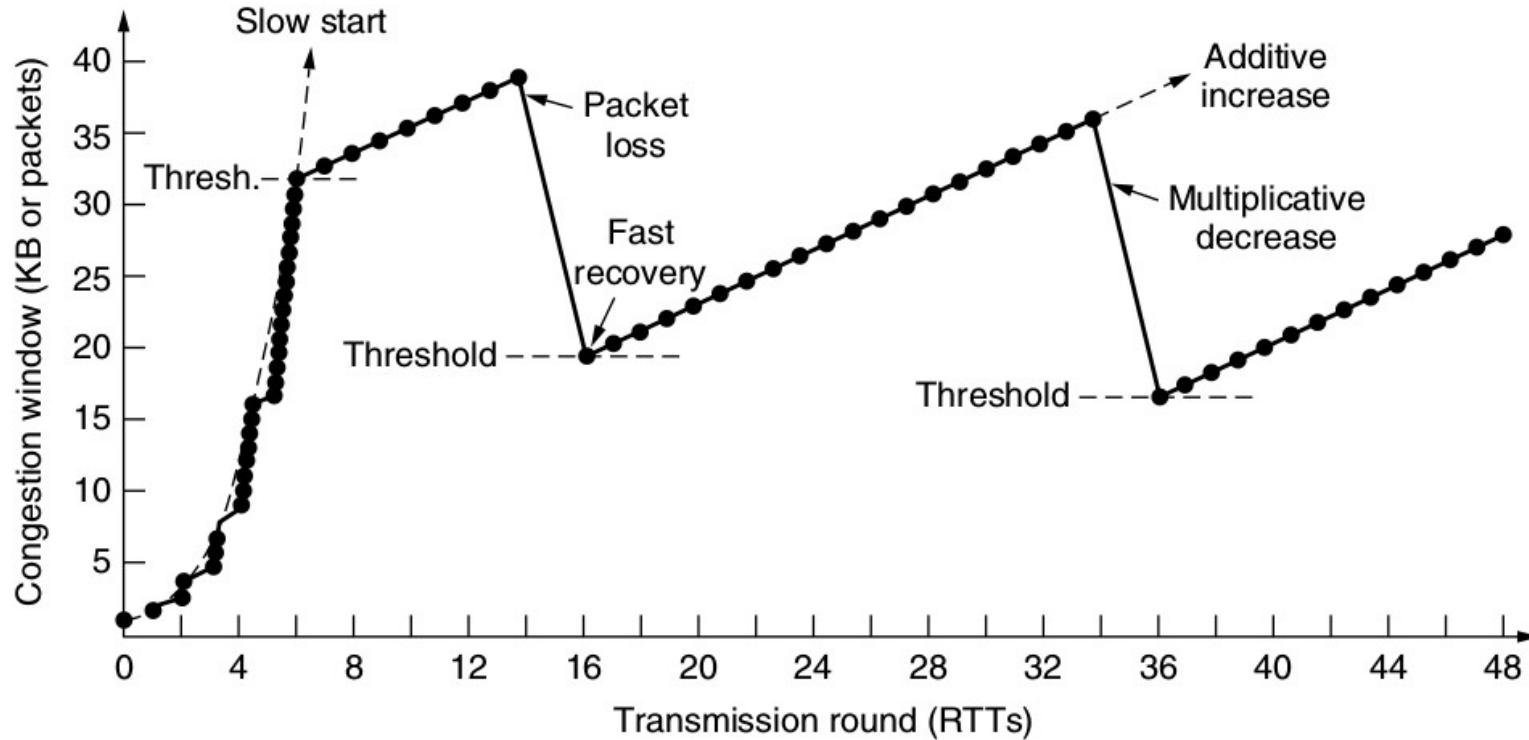


Beispiellauf von Slow Start / Congestion Avoidance (TCP Thaoe)



- Vorsichtiger Beginn (Slow-Start), aber nur bis zu einem Schwellwert
- Keine Unterscheidung zwischen Timeout und duplicate ACKs

Fast Retransmit und Fast Recovery (TCP Reno)



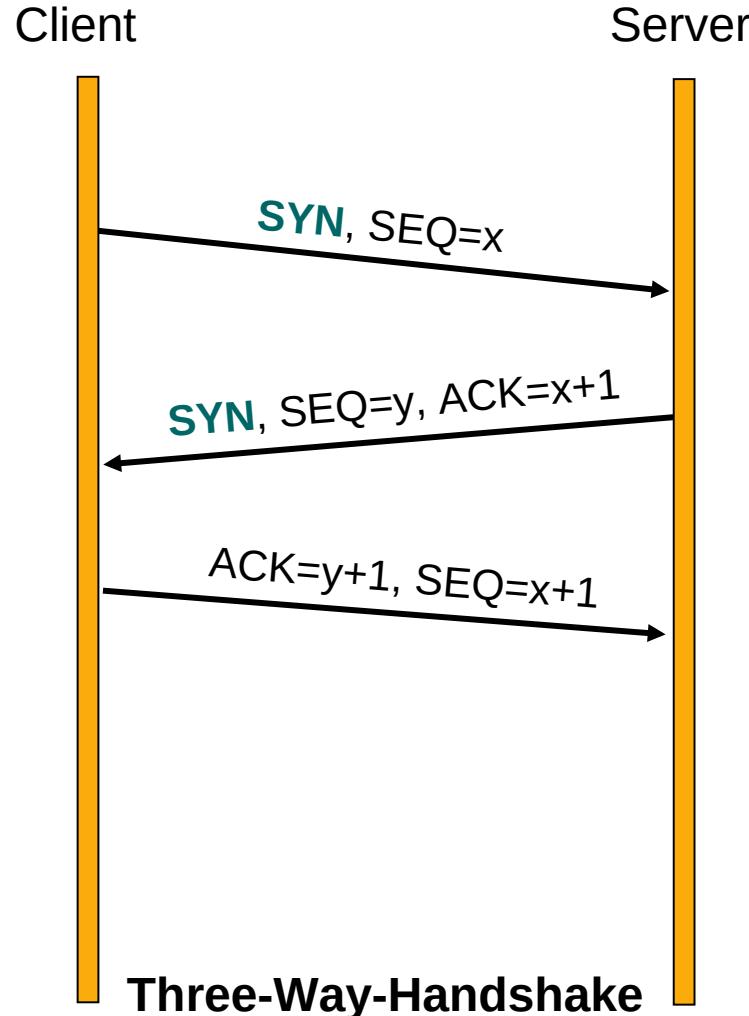
- Slow Start wie bei TCP Thaoe
- Unterscheidung zwischen Timeout und duplicate ACKs:
 - Bei Timeout gleiches Verhalten wie TCP Thaoe
 - Bei dup-ACKs (selective Reject) → Halbierung des Congestion Window (fast recovery)

Staukontrolle bei TCP

- Congestion Control ist deutlich komplexer als hier dargestellt
 - Viele Erweiterungen, um Einbruch der Datenrate zu vermeiden
 - ...
- Staukontrolle schafft Fairness! Wird eine Leitung von N-TCP-Verbindungen genutzt, so erhält jede $1/N$ -Anteile der verfügbaren Bandbreite

TCP-Verbindungsmanagement:

1. Verbindungsauftbau



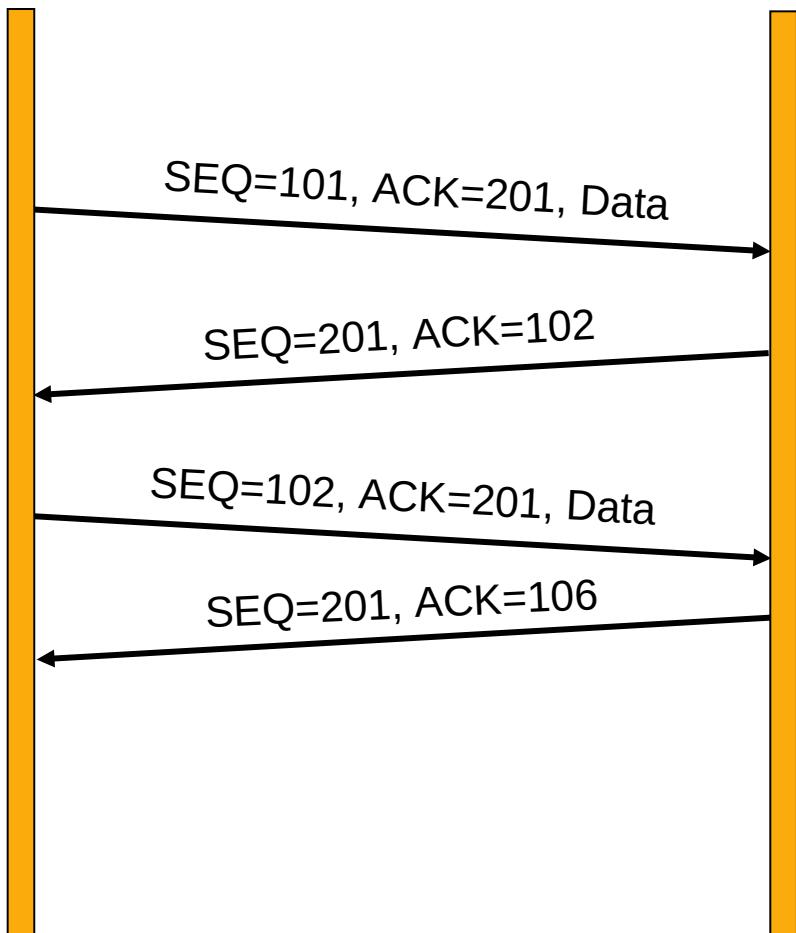
- Der Server wartet auf eingehende Verbindungswünsche.
- Der Client führt unter Angabe von IP-Adresse, Portnummer und maximal akzeptabler Segment-Größe eine CONNECT-Operation aus
- CONNECT sendet ein SYN
- Ist der Destination Port der CONNECT-Anfrage identisch zu der Port-Nummer, auf der der Server wartet, wird die Verbindung akzeptiert, andernfalls mit RST abgelehnt
- Der Server schickt seinerseits das SYN zum Client und bestätigt zugleich den Erhalt des ersten SYN-Segments
- Der Client schickt eine Bestätigung des SYN-Segments des Servers. Damit ist die Verbindung aufgebaut

TCP-Verbindungsmanagement:

2. Datenübertragung

Client

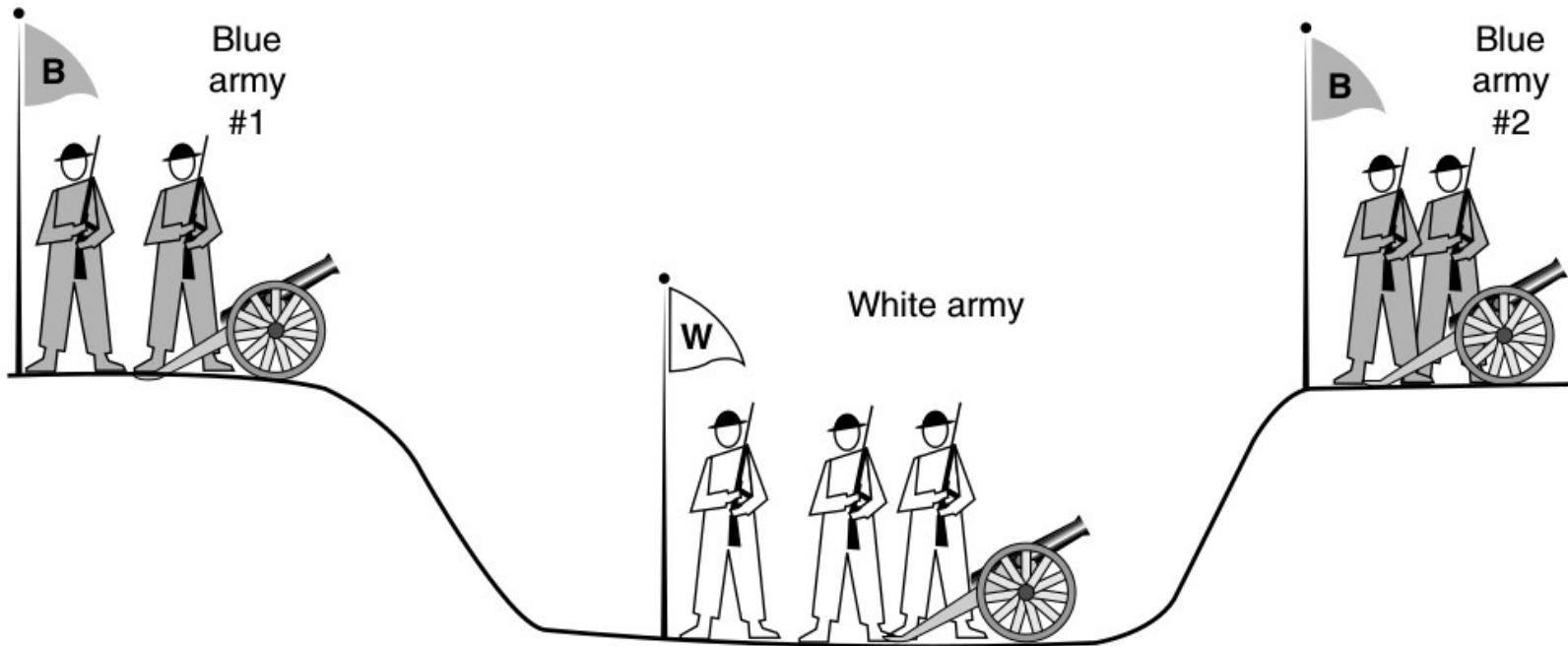
Server



- Vollduplex-Betrieb
- Aufteilung eines Bytestroms in Segmente. Übliche Größen sind 1460, 536 oder 512 Byte; dadurch wird IP-Fragmentierung vermieden.
- Üblicher Quittungsmechanismus: Alle Segmente bis ACK-1 sind bestätigt. Hat der Sender vor dem Empfang eines ACKs einen Timeout, überträgt er erneut.

TCP-Verbindungsmanagement:

3. Verbindungsende



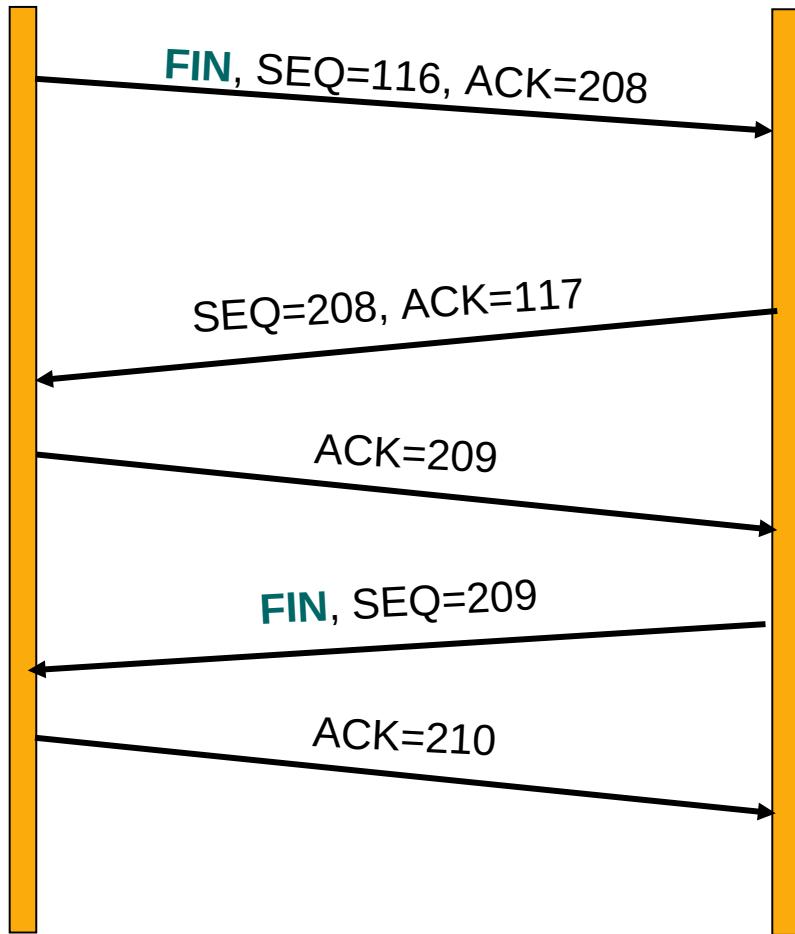
- Wann kann die blaue Armee angreifen? ($2 * \text{Blue} > \text{White}$, $\text{Blue} < \text{White}$)
- Benachrichtigung über Boten...
- Ist das sicher?
- Analogie zum Verbindungsabbau bei TCP: Wann sind sich beide Seiten sicher, dass die Verbindung abgebaut wurde?

TCP-Verbindungsmanagement:

3. Verbindungsende

Client

Server



- Abbau als zwei Simplex-Verbindungen
- Senden eines FIN-Segments
- Wird das FIN-Segment bestätigt, wird die Richtung 'abgeschaltet'. Die Gegenrichtung bleibt aber noch offen, hier kann noch weiter gesendet werden.
- Verwendung von Timern zum Schutz vor Paketverlust.

Transmission Control Protocol: Timer

- **Retransmission Timer**

Timer wird beim Senden eines Segmentes gestartet.
Bei Ablauf: Retransmission!

- **Persistence Timer**

Timer wird nach Empfang einer Fenstergröße von 0 gestartet. Nach Ablauf Anfrage nach neuem Fenster (>0)

- **Keepalive Timer**

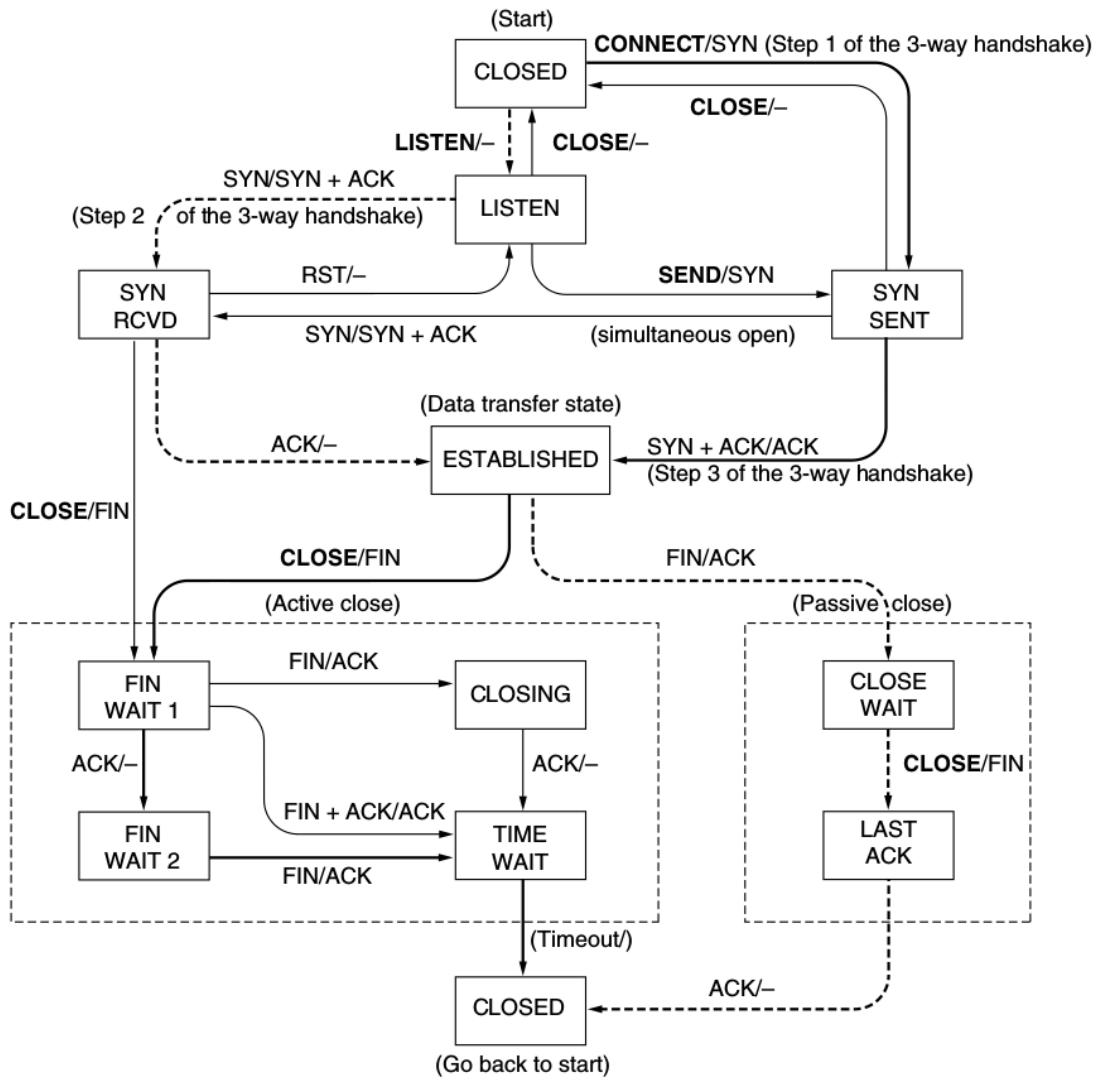
Rel. lange Wartezeit. Wir bei jeder Transaktion rückgesetzt.
Nach Ablauf → Abbau der Verbindung

- **Timer in TIME_WAIT**

Folgt auf ein FIN innerhalb von 2 RTTs kein ACK, wird die Verbindung abgebaut!

TCP-Verbindungsmanagement:

Übersicht Zustandsautomat



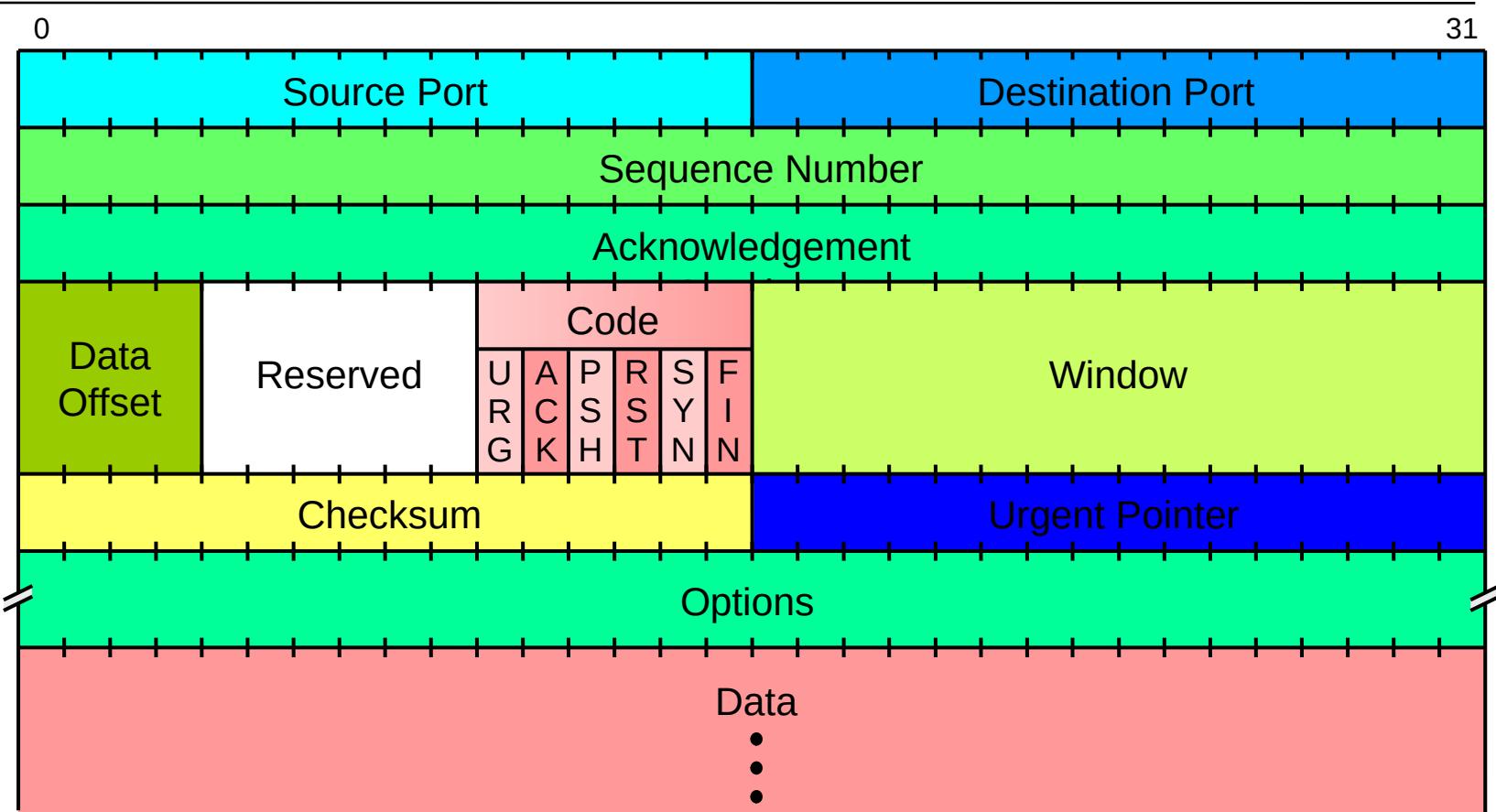
Syntax der Übergänge:
Ereignis/Aktion

Fette Linie:
Normaler Pfad des Clients

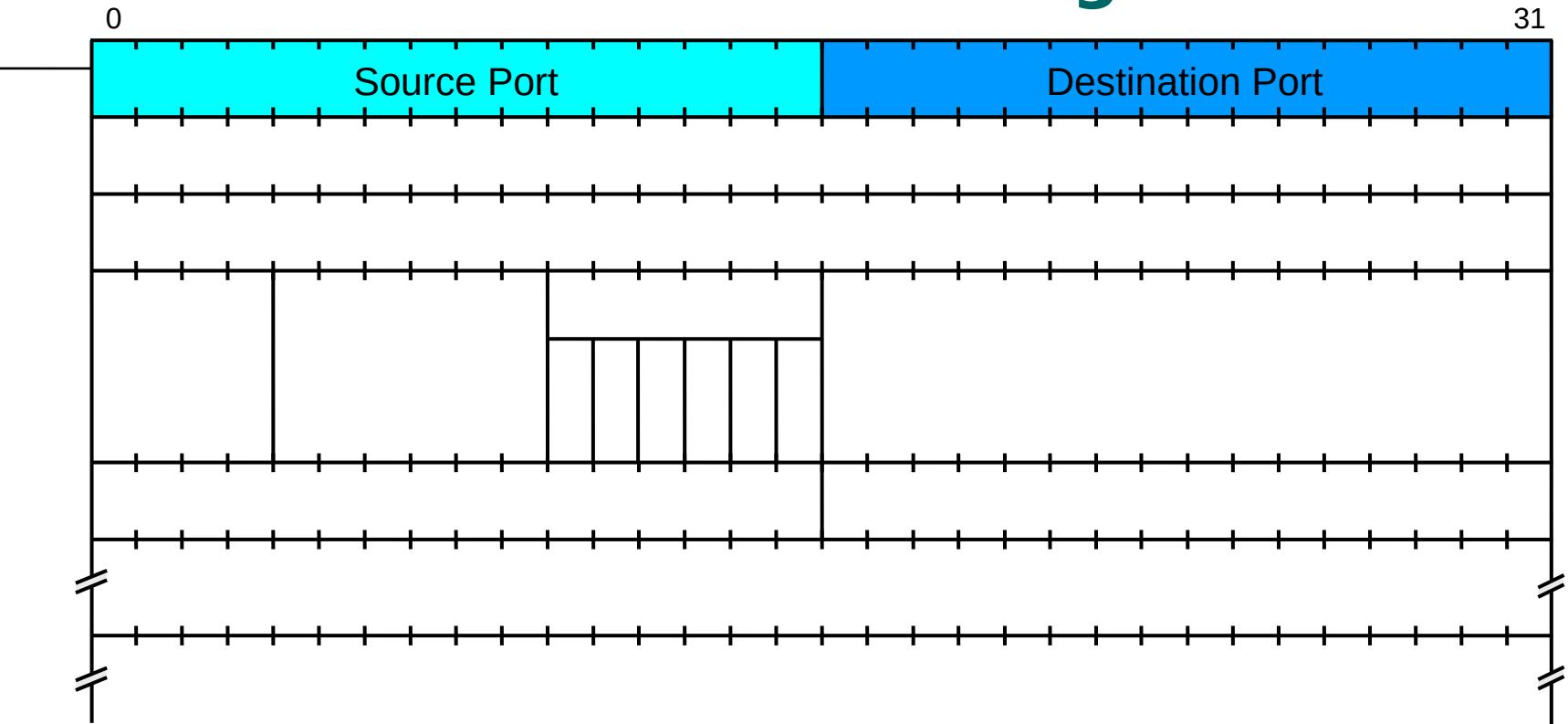
Fette gestrichelte Linie:
Normaler Pfad des Servers

Dünne Linien:
Ungewöhnliche Ereignisse

Format eines TCP-Segments



Format eines TCP-Segments



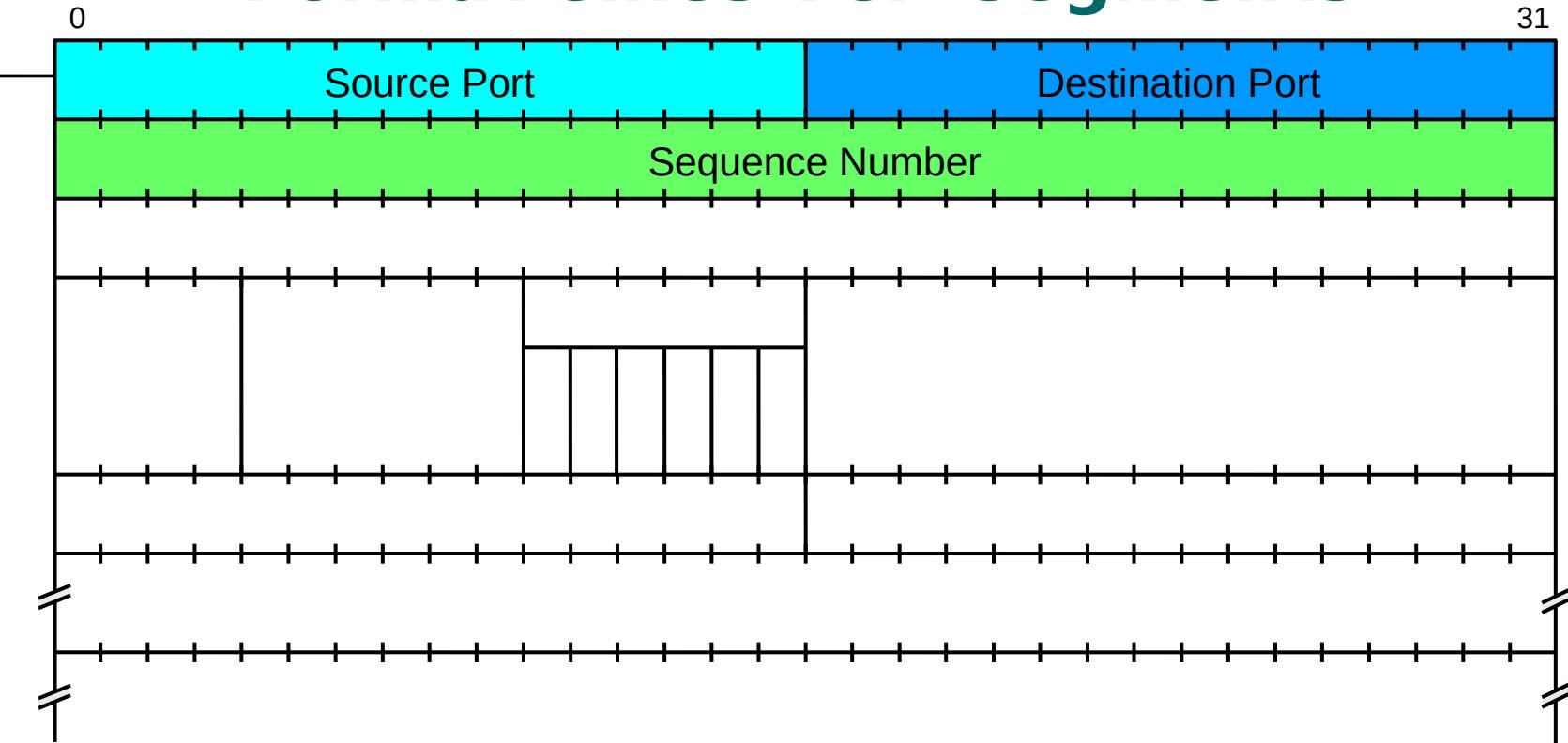
Source Port

Identifiziert die Anwendung auf der Senderseite.

Destination Port

Identifiziert die Anwendung auf der Empfängerseite.

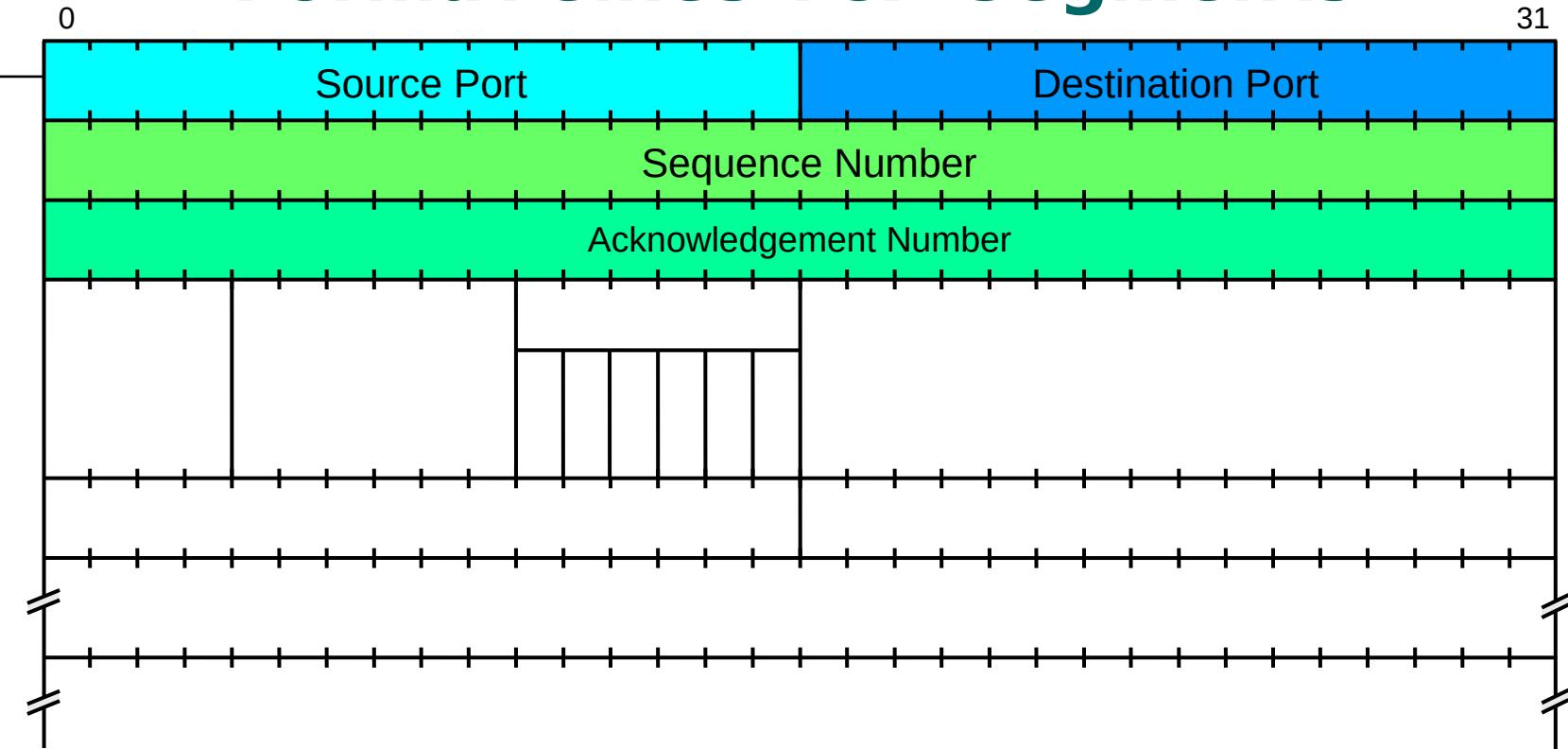
Format eines TCP-Segments



Sequence Number

TCP betrachtet die zu übertragenden Daten als nummerierten Byte-Strom.
Die *Sequence Number* ist die Nummer des ersten im Segment enthaltenen Datenbytes.

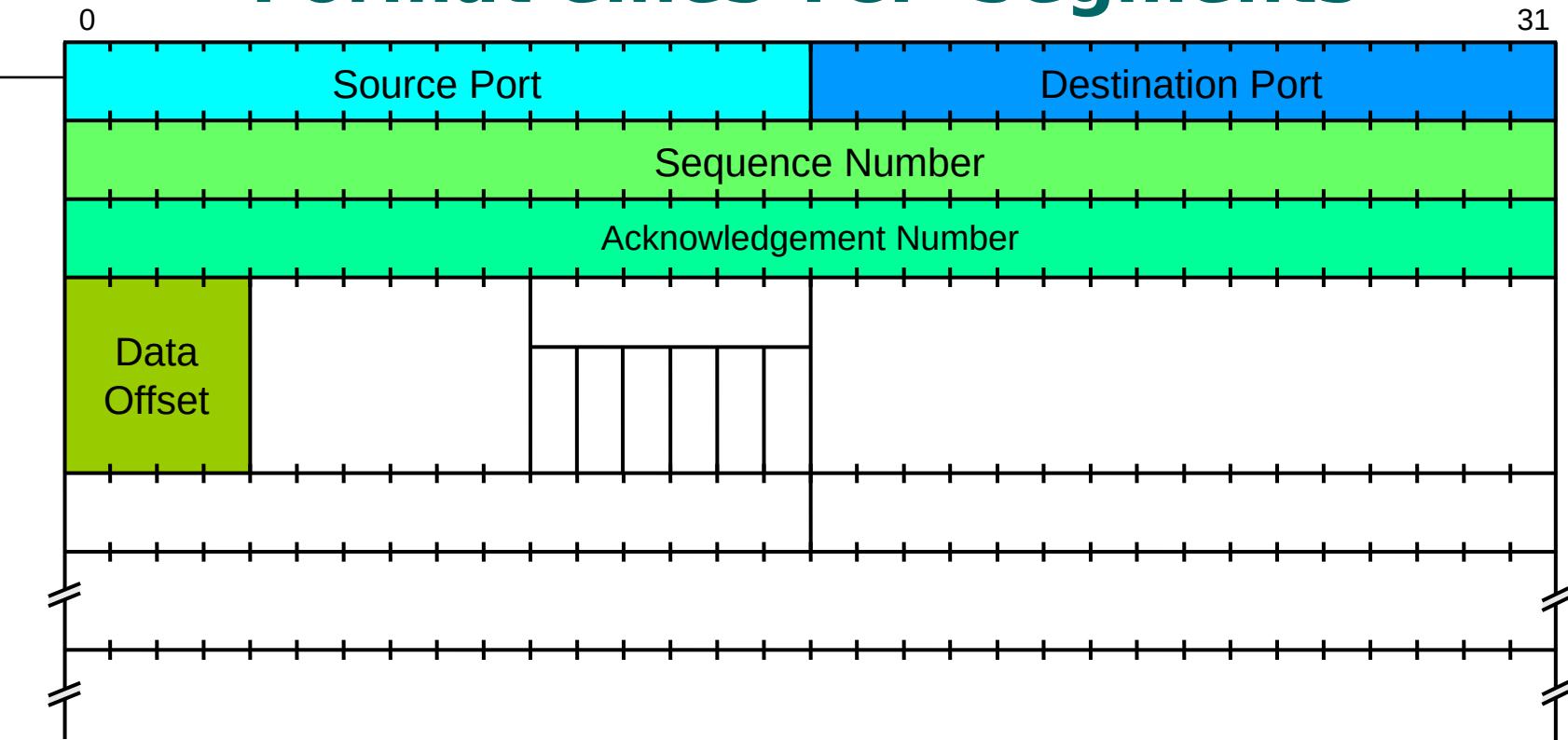
Format eines TCP-Segments



Acknowledgement Number

Dieses Feld bezieht sich auf einen Datenfluss in Gegenrichtung, d.h. hiermit werden Daten bestätigt, die die Station, die das Segment absendet, zuvor von der Zielstation empfangen hat.

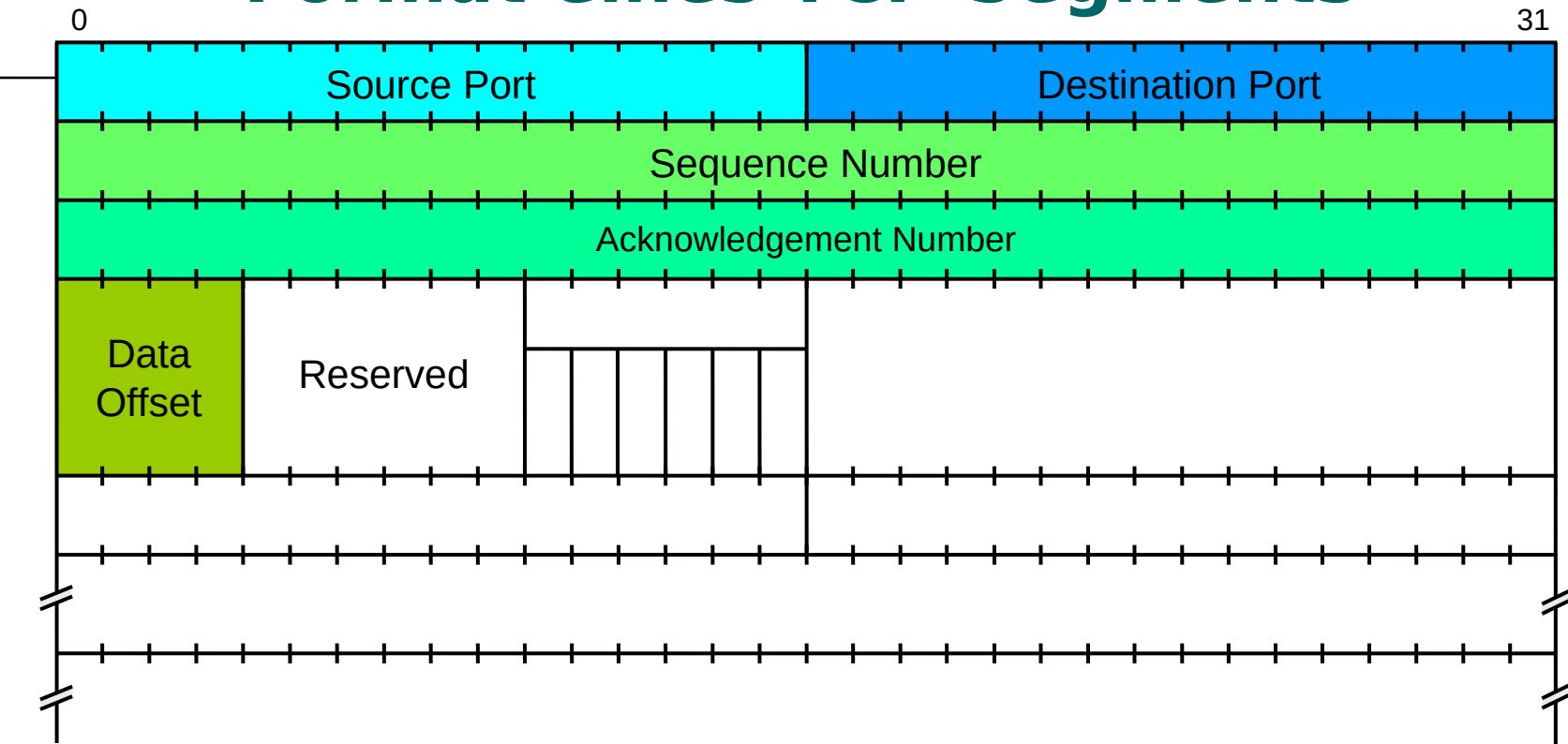
Format eines TCP-Segments



Data Offset

Da der Segment-Header Optionen enthalten kann, ist seine Länge nicht fix.
Im *Data Offset*-Feld wird die Länge (d.h. der Beginn des Datenteils) in 32-Bit-Einheiten angegeben.

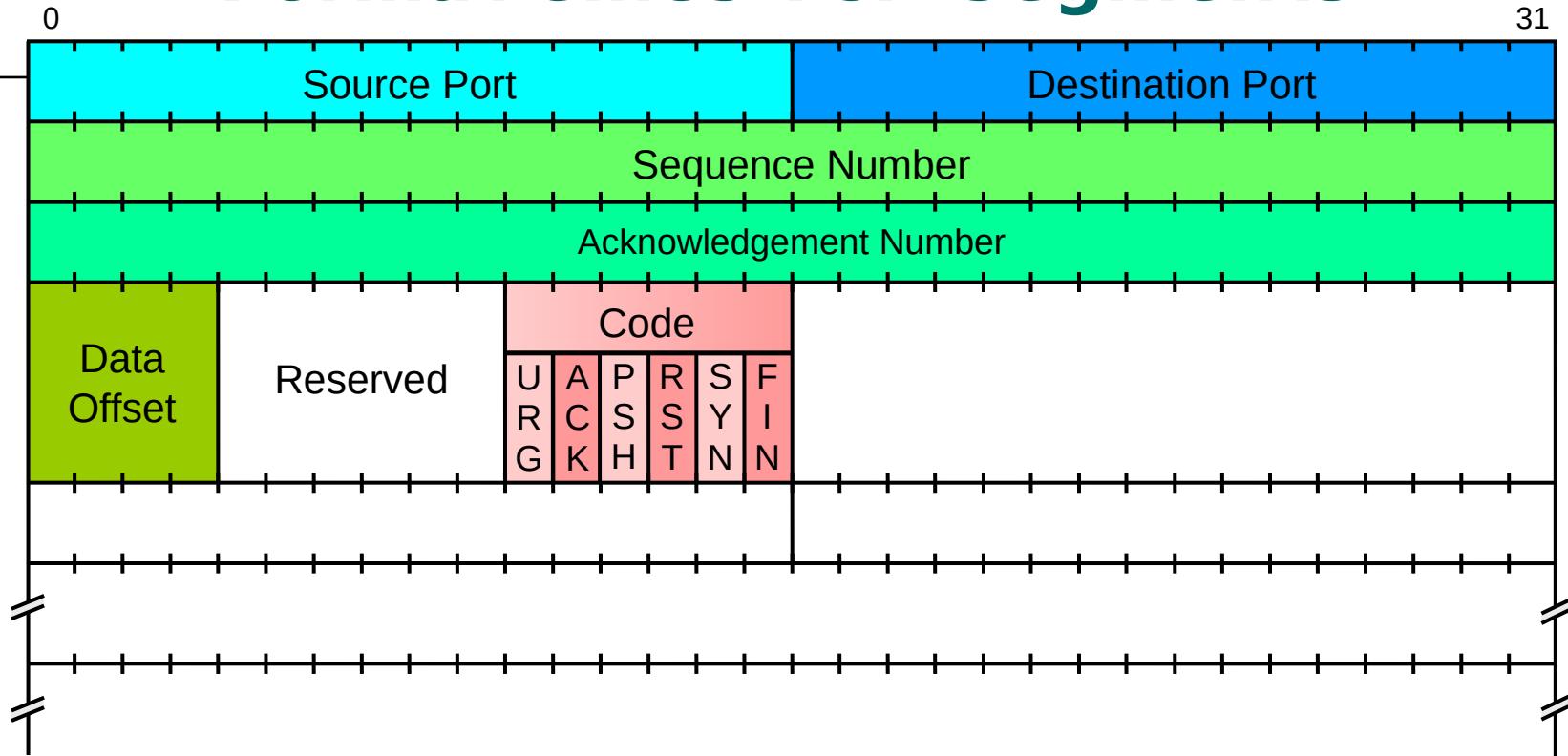
Format eines TCP-Segments



Res.

Reserviert für zukünftige Nutzung.

Format eines TCP-Segments



Code Die Bits des Code-Feldes steuern besondere Funktionen des Segments.

URG *Urgent Pointer Field is valid*

PSH *This segment requests a push*

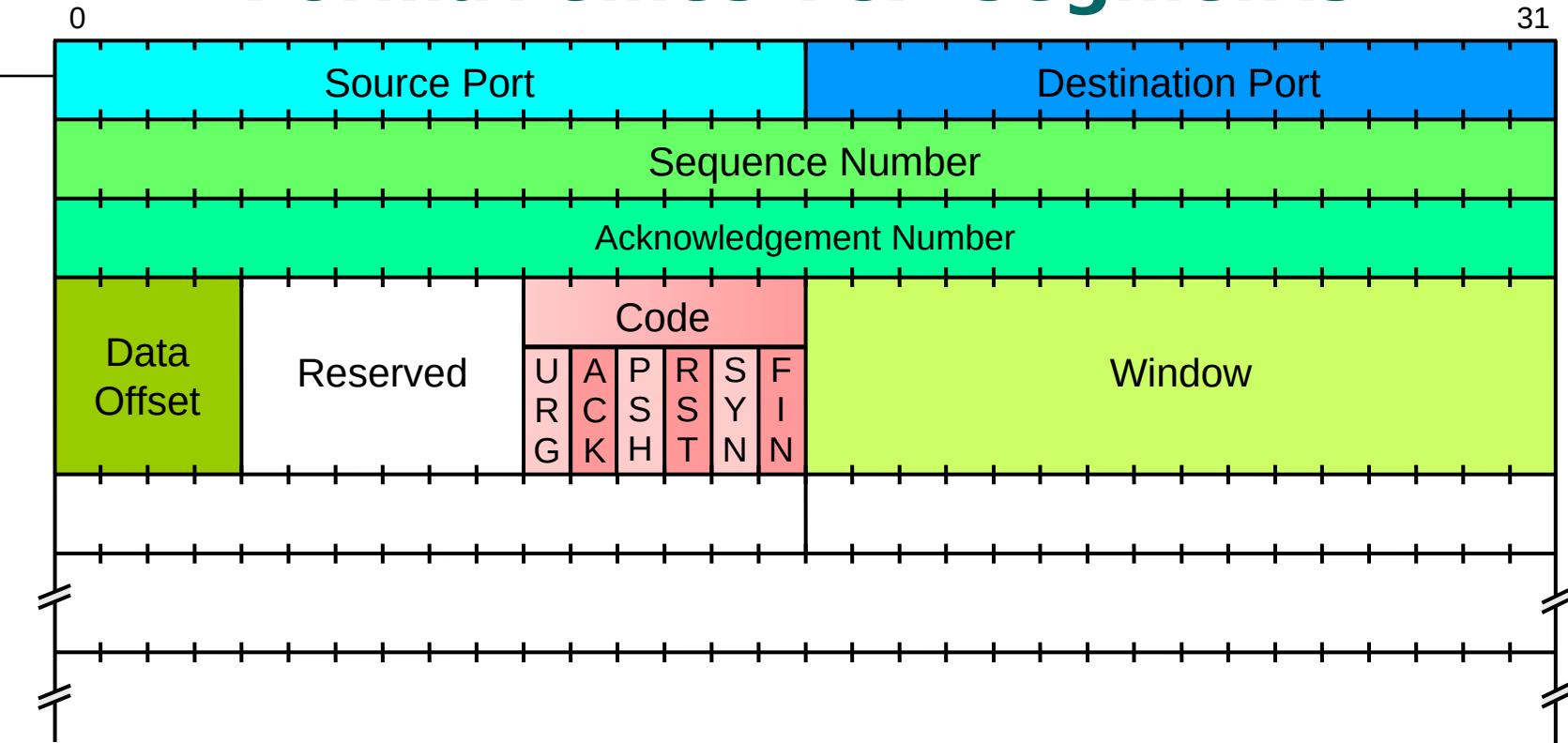
SYN *Synchronize sequence numbers*

ACK *Acknowledgement Field is valid*

RST *Reset the Connection*

FIN *Sender has reached end of his byte stream*

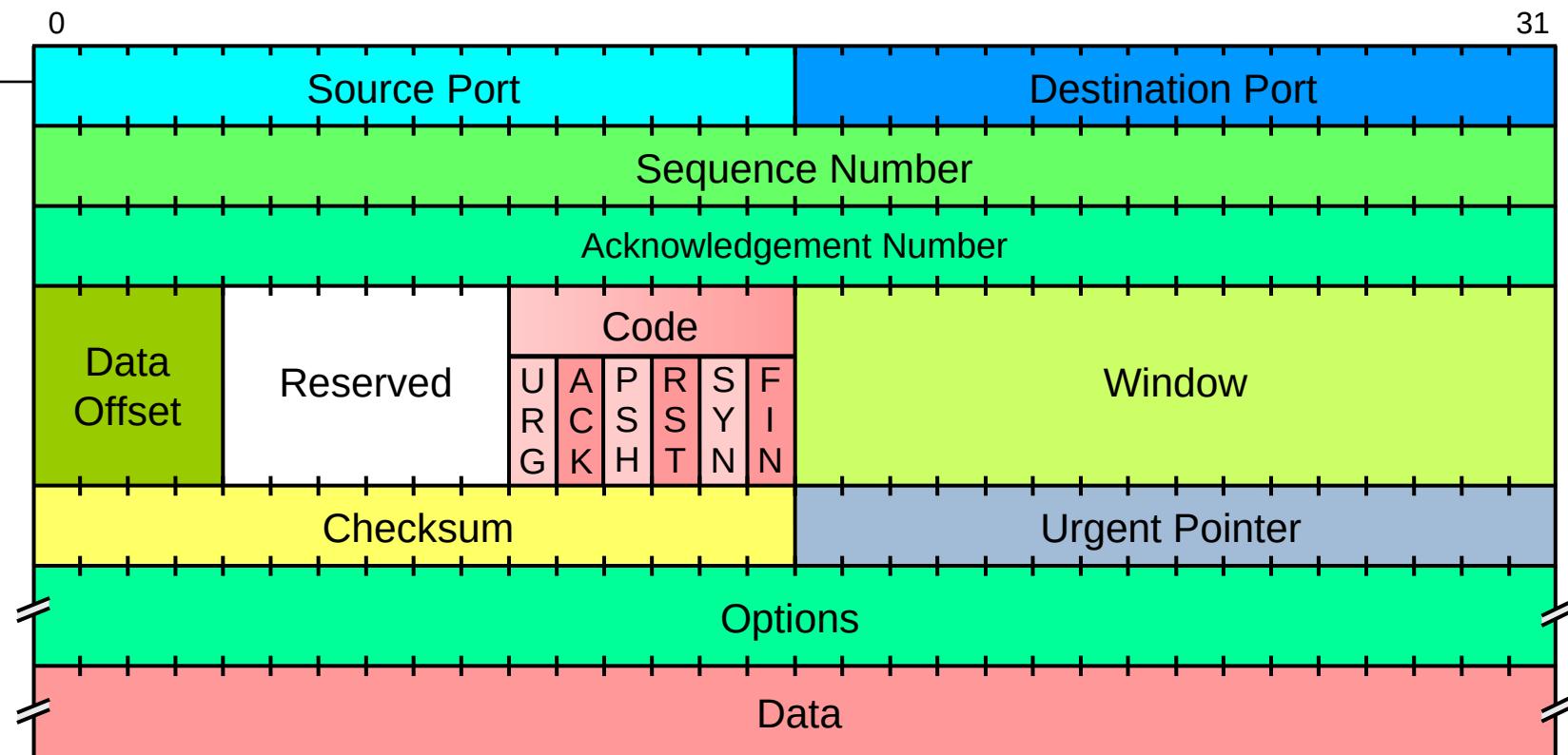
Format eines TCP-Segments



Window

Spezifiziert die Anzahl der Datenbytes (beginnend mit der im *Acknowledgement*-Feld angegebenen Byte-Nummer), die der Sender des Segments als Empfänger eines Datenstromes in Gegenrichtung akzeptieren wird.

Format eines TCP-Segments



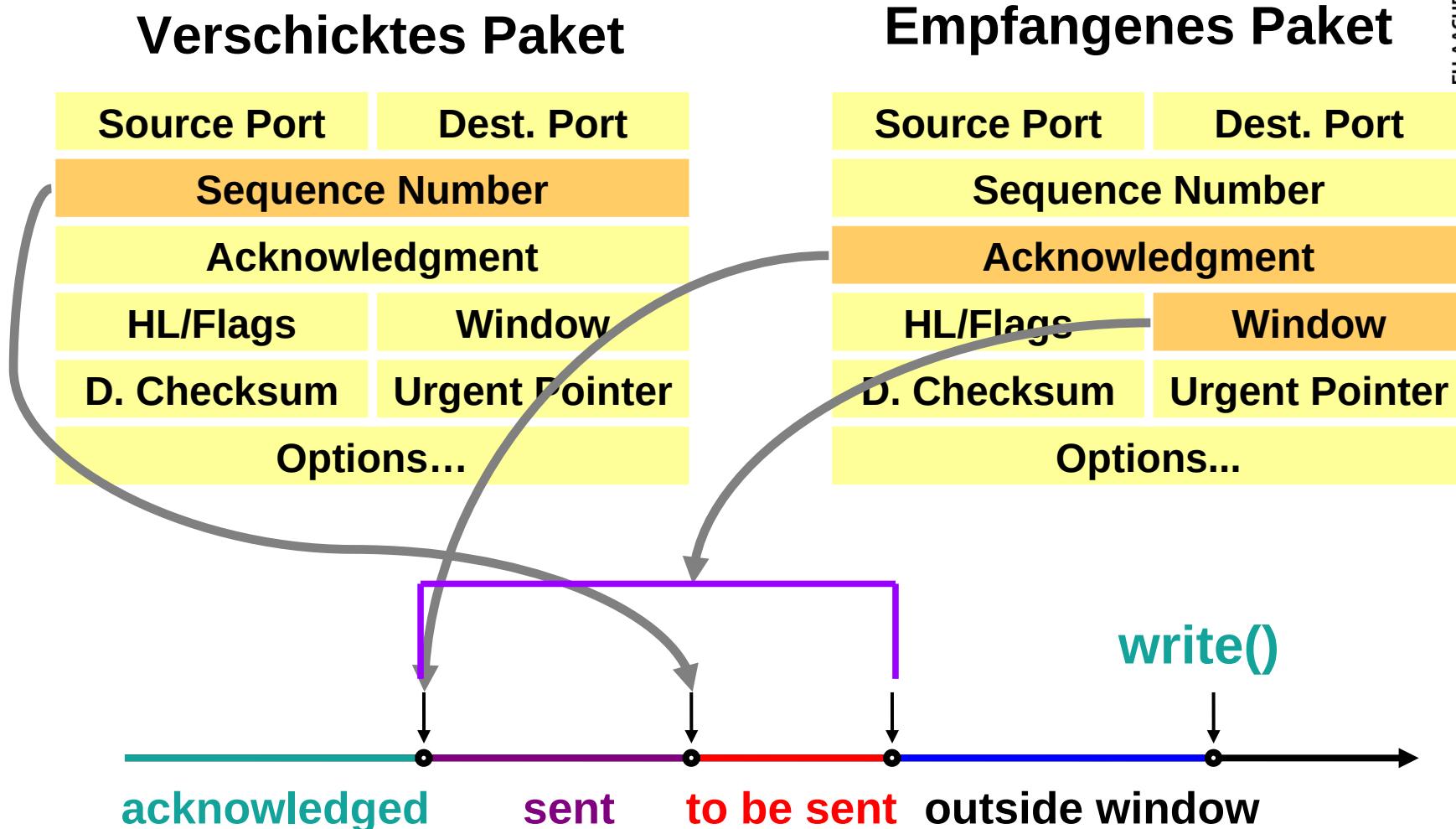
Checksum

16-Bit Längsparity über das gesamte Segment (*Header + Daten*).

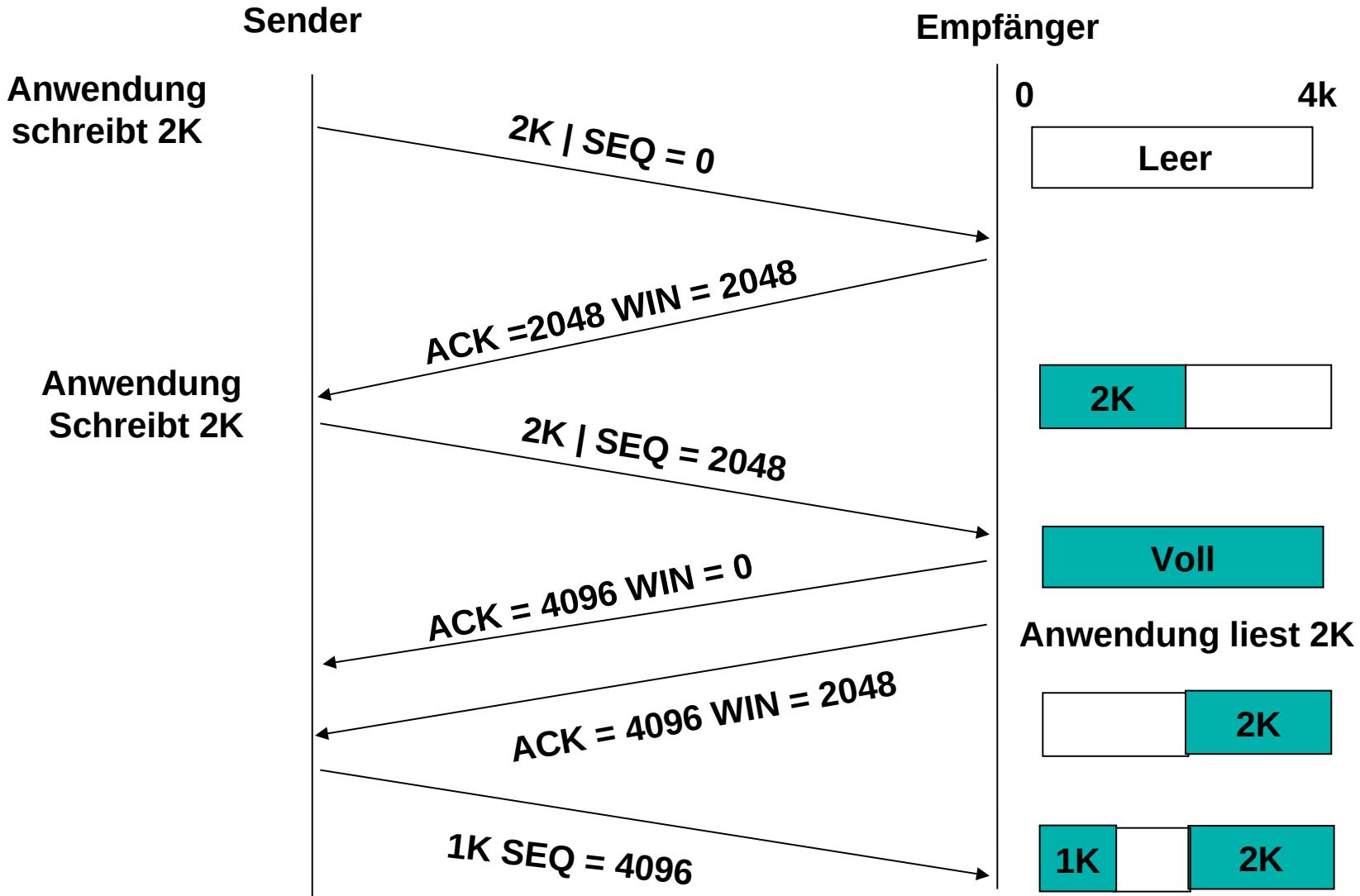
Urgent Pointer

Damit können Teile des zu übertragenden Byte-Stroms als dringend markiert werden.

Window-Flußkontrolle: Sender-Seite



Window-Management in TCP



Problemszenario 1: Server antwortet nicht nach WIN=0

- Nach WIN=0 könnte das folgende ACK vom Empfänger der Daten verloren gehen...
- Auf Sender-Seite gibt es für diesen Fall einen weiteren Timer:

Transmission Control Protocol: Timer

- **Retransmission Timer**

Timer wird beim Senden eines Segmentes gestartet.
Bei Ablauf: Retransmission!

- **Persistence Timer**

Timer wird nach Empfang einer Fenstergröße von 0 gestartet. Nach Ablauf Anfrage nach neuem Fenster (>0)

- **Keepalive Timer**

Rel. lange Wartezeit. Wir bei jeder Transaktion rückgesetzt.
Nach Ablauf → Abbau der Verbindung

- **Timer in TIME_WAIT**

Folgt auf ein FIN innerhalb von 2 RTTs kein ACK, wird die Verbindung abgebaut!

Problemszenario 2: Bandwidth-Delay-Produkt >> 64kB

- Der TCP-Header stellt nur ein 16-bit-Feld für das Advertised Window zur Verfügung
- Damit kann zunächst keine Übertragungsfenster größer 64kB genutzt werden
- Das Bandwidth-Delay-Produkt gibt an, wie groß die Größe des Übertragungsfensters bei gegebener Bandbreite und Round-Trip-Zeit sein könnte:

$$100\text{ms} * 1\text{Gbit} = 12,5 \text{ MB} >> 64 \text{ kB}$$

- Wir brauchen größere ‚Fenster‘!
- **Lösung:** Window-Scale Option:

TCP Window Scale Option

- Einführen einer **Window-Scale-Erweiterung** die ein Skalierungsfaktor für das 16-Bit Window-Feld darstellt
- Der Skalierungsfaktor wird als neue TCP-Option eingeführt: Window Scale ist 3 Byte lang – Das letzte Byte gibt die Skalierung LOGARITHMISCH an (shift count: Das Fenster wird shift count bits nach links verschieben – hierbei ist der maximale Wert 14!)
- Die Option wird nur in einem SYN-Segment **beim Verbindungsauftbau übertragen**. Danach wird der Wert für die Ganze Verbindung angenommen
- Beide Seiten müssen eine Window-Scale-Option verschicken Null bedeutet "keine Skalierung"
- Die neue maximale Fenstergröße in Bytes errechnet sich wie folgt:

$$65536 * 2^{14} = 65535 * 16384 = 1.073.725.440$$

- TCP-intern wird die Fenstergröße jetzt als 32-Bit-Zahl verwaltet

TCP Window Scale Option: RFC 1323

[RFC 1323](#)

TCP Extensions for High Performance

May 1992

but can be overridden by a user program before a TCP connection is opened. This determines the scale factor, and therefore no new user interface is needed for window scaling.

2.2 Window Scale Option

The three-byte Window Scale option may be sent in a SYN segment by a TCP. It has two purposes: (1) indicate that the TCP is prepared to do both send and receive window scaling, and (2) communicate a scale factor to be applied to its receive window. Thus, a TCP that is prepared to scale windows should send the option, even if its own scale factor is 1. The scale factor is limited to a power of two and encoded logarithmically, so it may be implemented by binary shift operations.

TCP Window Scale Option (WSopt):

Kind: 3 Length: 3 bytes

+-----+	+-----+	+-----+
Kind=3	Length=3	shift.cnt
+-----+	+-----+	+-----+

TCP Window Scale Option

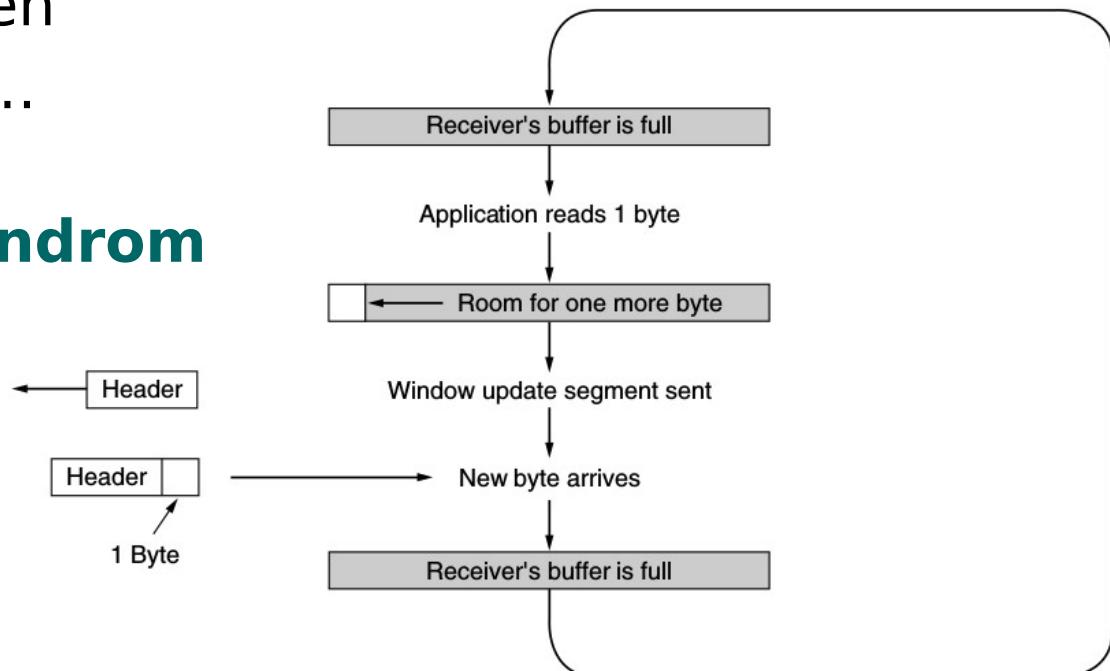
- Ohne Window-Scale-Option ist TCP nicht geeignet für sogenannte Long-Fat-Pipes
- Die Window-Scale Option wird beim Aufbau der Verbindung **in beiden Richtungen** separat mitgeteilt
- Anschaulich handelt es sich um die Anzahl der binären Nullen, die hinter dem Advertised/Receiver Window zusätzlich stehen
- Damit können aber auch nur noch größere Änderungen im Puffer des Empfängers angezeigt werden
- Verpasst eine Anwendung das Setzen der Option vor einer Verbindung, so kann sie ggf. die gewünschte Rate nicht erzielen

Problemszenario 3: Pakete und einzelne Zeichen

- Das Versenden und Empfangen von Daten geschieht in Paketen von z.B. typischerweise 1460 Bytes:
- Der **Sender** wartet erst, bis er die Daten für solch ein Paket gesammelt hat, und sendet dann das ganze Paket
- Der **Empfänger** würde erst die Daten zwischenspeichern, bevor er sie der Applikation zur Verfügung stellt.

Problemszenario 3: Pakete und einzelne Zeichen

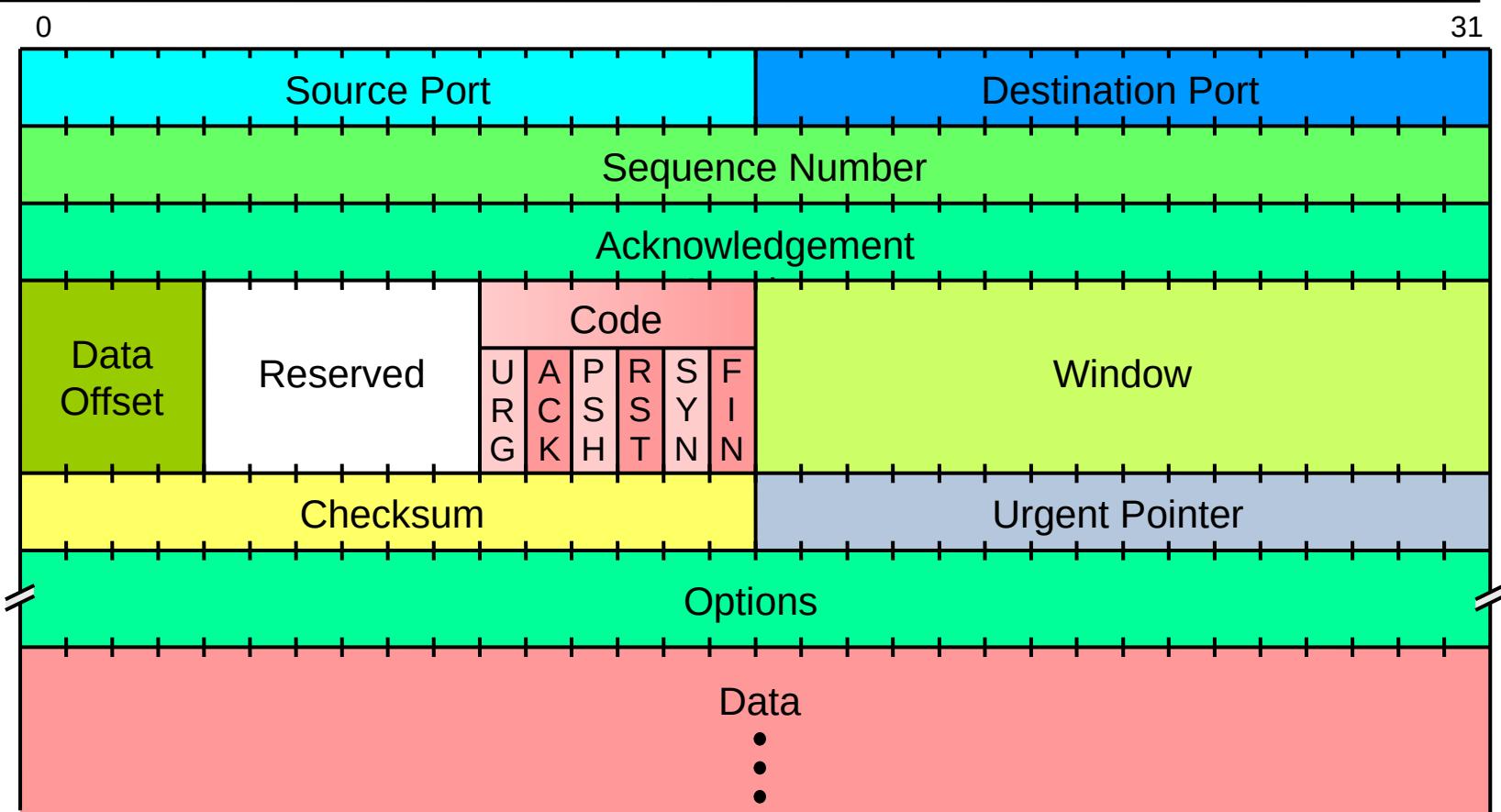
- Was passiert, wenn das Empfangsfenster ‚voll‘ ist, und die empfangende Applikation die Zeichen aber einzeln ausliest?
- Der Sender würde für jedes einzelne Zeichen ein neues Paket senden
- Sehr ineffizient...
- Name dafür:
Silly Window Syndrom
- Lösung (Clark):
Empfänger darf neue Fenstergröße erst senden, wenn in seinem Buffer mehr Platz ist (z.B. ein Segment)



Problemszenario 3: Pakete und einzelne Zeichen

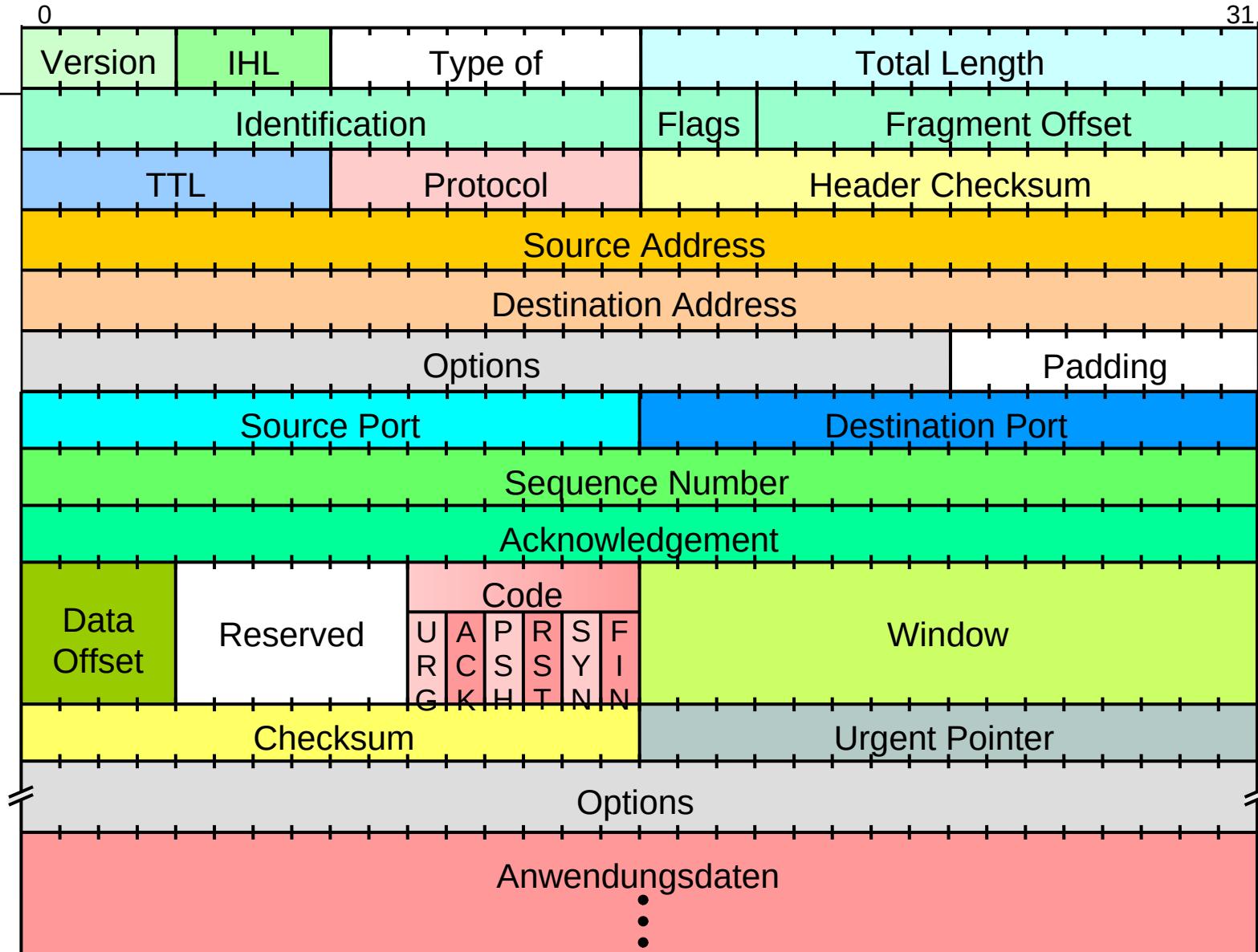
- Was passiert, wenn der Sender wichtige Daten hat, die er ohne Pufferung an die empfangende Applikation weiter geben will?
- Beispiel: Ctrl-C zum Abbruch einer Ausgabe
- Lösung:
 - **PSH** – Flagge im TCP Header:
Daten auf Empfängerseite direkt zustellen (kein Puffern)
- Alternative:
 - **URG** – Flagge im TCP Header:
,Event' auf Empfängerseite auslösen

Format eines TCP-Segments



Frage: Wie wird die Länge eines Datagramms ermittelt?

Was kommt am Rechner an?

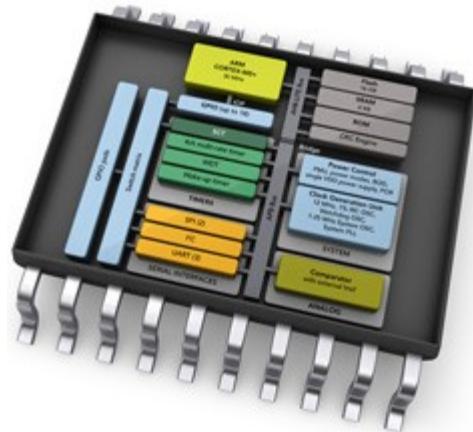


FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

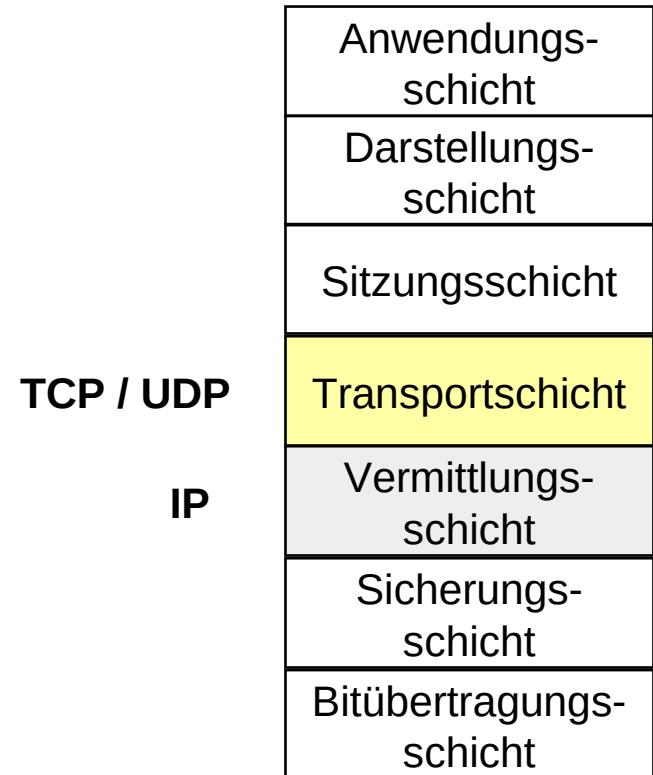
(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Einführung in das User Datagram Protocol (UDP)

- IP kann potentiell Pakete verwerfen (Store-and-Forward Prinzip)
- TCP ist sicher, aber komplex
- Häufig wird das nicht benötigt:
 - Kommunikation ist nur lokal
 - Geringer Datenverlust ist okay (Audio/Video Daten)
 - Feste Datenraten (z.B. beim Streaming ↔ TCP Slow-Start ???)
 - Implementierung eigener („leichter“) Sicherungsmechanismen
 - ...



UDP: User Datagram Protocol

- UDP stellt eine „direkte“ Schnittstelle zur Nutzung von IP dar: Anwendungen können Nachrichten direkt verschicken, ohne Verbindungsauftbau
- Unzuverlässig, verbindungslos
- einfacher und schneller als TCP
- Optionale Prüfsumme
- Sehr viele Multimedia-Anwendungen verwenden UDP, da dort keine zuverlässige Verbindung benötigt wird

UDP Datagram Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Length				Header and Data Checksum			

Der UDP-Header

Source Port

Identifiziert den sendenden Prozess, also den Prozess, an den gegebenenfalls Rückmeldungen zu senden sind. Die Angabe ist optional; das Feld soll den Wert null haben, wenn die Option nicht genutzt wird.

Destination Port

Identifiziert den Prozess im Zielsystem, an den die Daten abzuliefern sind.

Length

Im Längenfeld wird die Gesamtlänge des UDP-Datagramms in Bytes angegeben; die Mindestlänge beträgt somit 8 (= *Header*-Länge)

Checksum

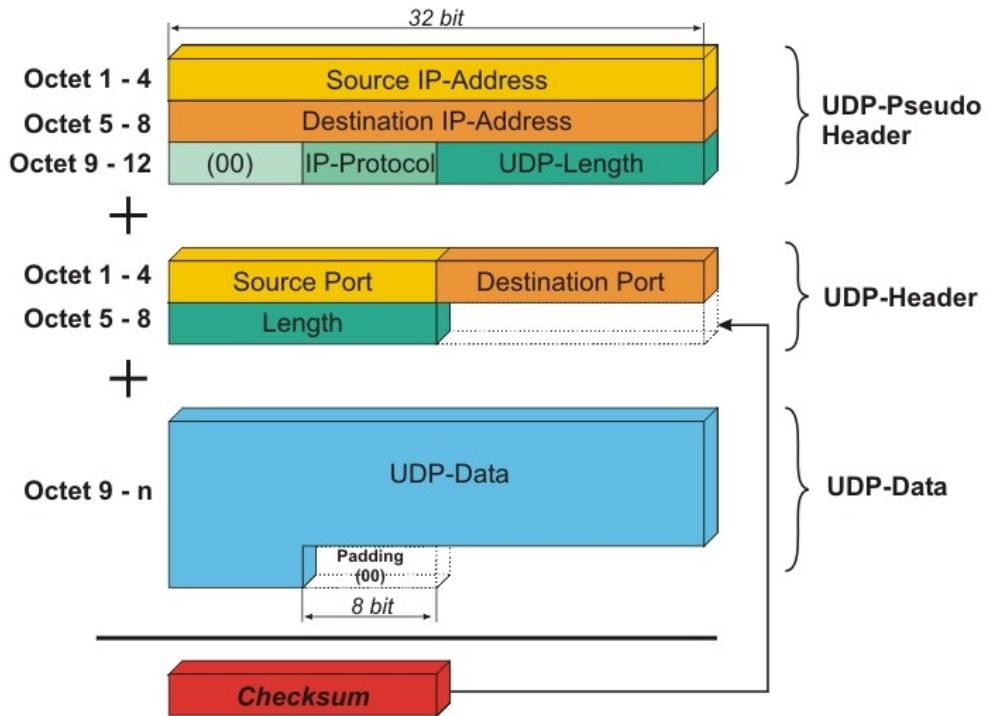
Die Angabe ist optional (0 bedeutet: keine Angabe). Für die Berechnung der Längsparität wird dem UDP-Datagramm ein (nicht mitübertragener) Pseudo-Header von 12 Bytes Länge vorangestellt, der im wesentlichen IP-*Source Address*, IP-*Destination Address* und die im IP-Datagramm angegebene Protokoll-Nr. für UDP (17) enthält.

Da der Datenteil eines IP-Datagramms nicht durch die IP *Header Checksum* geschützt ist, bedeutet ein Verzicht auf die UDP-*Checksum*, dass der Inhalt des UDP-Datagramms (*Header* und Daten) nicht durch eine Prüfsumme gesichert ist.

UDP Checksum

Ziel: Erkennen von Fehlern (z.B. flipped bits) im übertragenen Segment – optionale Nutzung!

- Formal über eine Einer-komplementsumme über ...
 - Pseudo-IP-Header: (Verletzung der Schichtgrenze!)
 - UDP Header
 - Daten
- Details zur Berechnung:
 - Optionales Auffüllen der Daten (wenn ungerade Byte-Anzahl → ‚Padding‘)
 - Berechnung der Prüfsumme durch Interpretation der Daten als 16-Bit Werte. Aufaddieren der 16-Bit-Werte im Einer-Komplement.
 - Am Ende wird das 1-er Komplement der Summe berechnet
- Einfügen der Prüfsumme in den UDP-Header



Wozu UDP? Anwendungen

- **Multimedia:**
Die digitale Übertragung von Audio- und Videodaten besitzt spezifische Anforderungen:
 - Geringe Verlustraten stören nicht
 - Isochrones Abspielen → schwierig mit TCP ...
 - Latenzzeiten müssen insbesondere bei interaktiven Anwendungen gering sein (Telefonie erfordern eine maximale Latenz von 150ms)
 - Jitter: Die Variation der Laufzeit sollte ebenso beschränkt sein
- **RPC:** Remote Procedure Calls
- **NFS:** Network File System
- **RTP:** Real-Time Transport Protocol
- **DNS:** Domain Name System
- ...

Weitere wichtige UDP-basierte Anwendung

Das Laden des Betriebssystems (Boot-Vorgang) über das Netzwerk benötigt entsprechende Protokolle

- TCP ist aufwendig, Fähigkeiten der im BIOS verankerten Mechanismen ist begrenzt
- Ethernet als Sicherungsschicht implementiert bereits Mechanismen zur Zuverlässigkeit, wenn der Server im gleichen Netz ist braucht man viele Mechanismen von TCP nicht
- Übertragung von Dateien z.B. über das Trivial File Transfer Protocol (**TFTP**)

BIOS verfügen zumeist bereits über eine UDP-Implementierung, daher wurde mit **BOOTP** und **DHCP** entsprechendes auf UDP-Basis entwickelt Broadcast an alle durch Zieladresse 255.255.255.255.

Continuous Media: Digitalisierung der Daten

Kleiner Exkurs

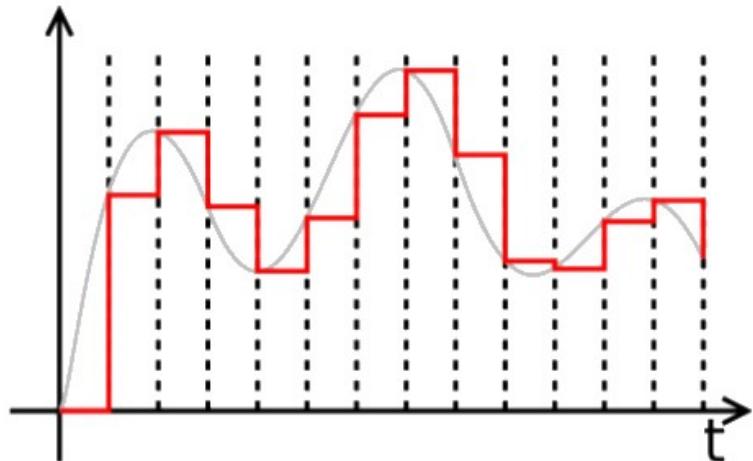
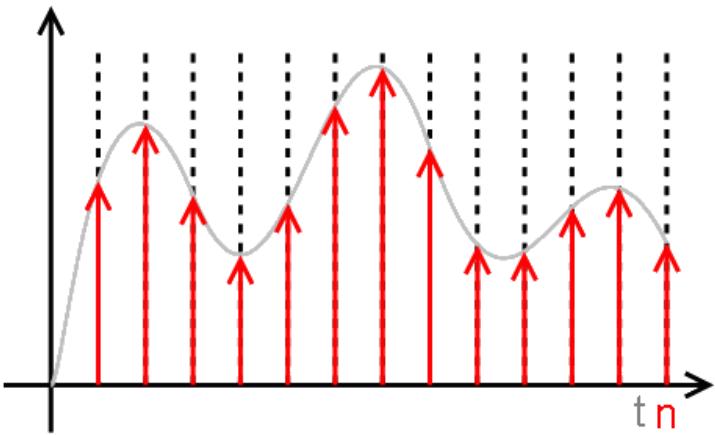
Übertragen analoger (streaming)-DATEN

- Abtasten
- Halten
- Quantisieren
- Kodieren



Abtastung

- Signalstärke wird regelmäßig gemessen



- **Abtastung:**

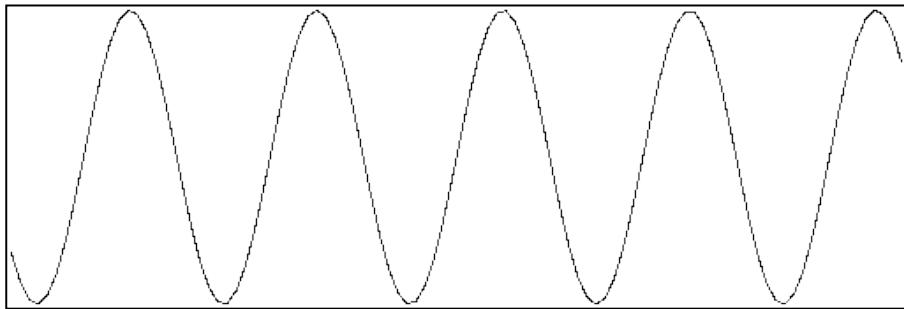
Messen des analogen Wertes zu den Zeitpunkten

$$t_n = n * \Delta t$$

- **Halten des Wertes:**

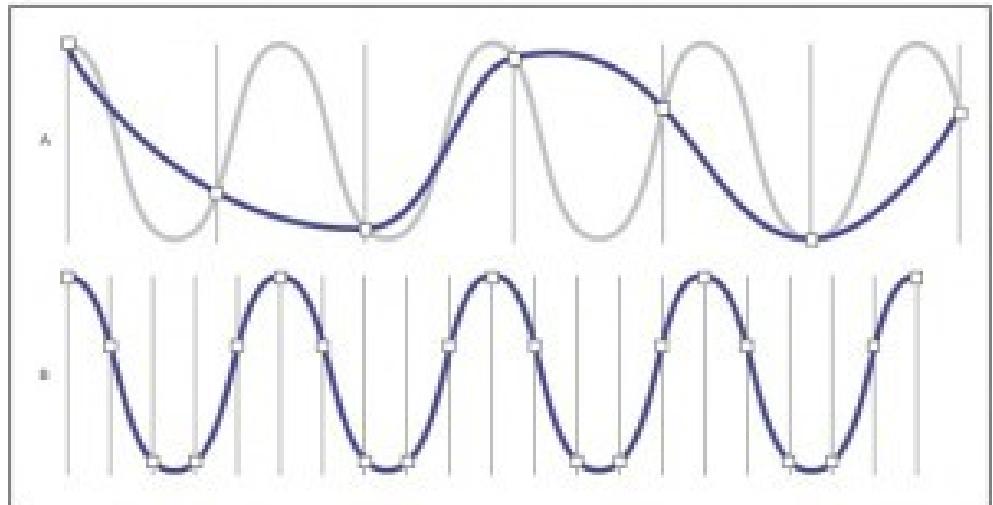
Festhalten des Messwertes, damit er in der Zeit Δt digitalisiert werden kann.

Sampling Rate



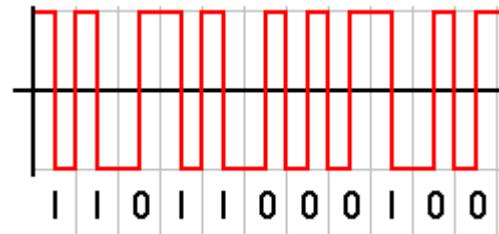
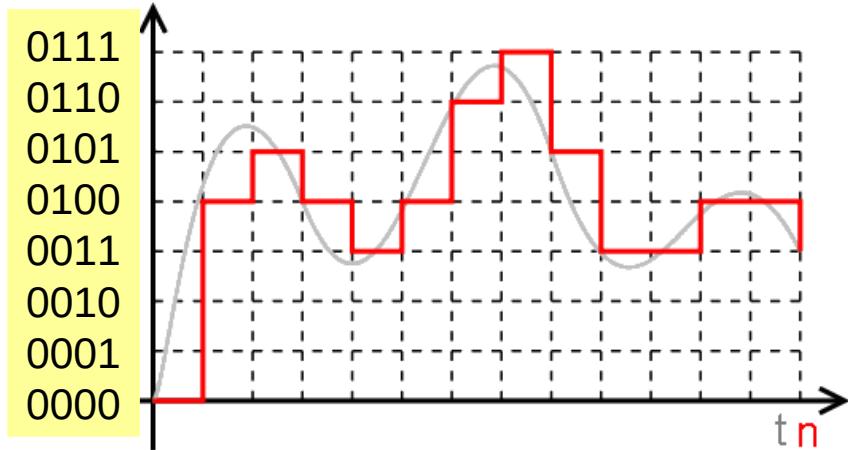
Nyquist Theorem:

Sampling-Frequenz \geq
 $2 * \text{maximale Frequenz des}$
Signals



Quantisierung und Kodierung

- Umwandlung Spannungswerte → Zahlenwerte



- Quantisierung:**
Umwandlung des Messwertes in den bestmöglichen Wert des digitalen Wertevorrats (Binärzahl)

- Kodierung:**
Umwandlung der digitalen Werte in z.B. einen Leitungs-Code
 - ggf. Fehlerkorrektur
 - Taktrückgewinnung
 - ...

Qualität des resultierenden Digitalsignals

- Abhängig von
 - Anzahl der Quantisierungsstufen (Auflösung der einzelnen Niveaus / Lautstärken)
→ Quantisierungsrauschen
 - Abtastrate (liefert Bandbreite):
- Bitrate/s = Abtastrate/Hz * Bits/Abtastwert
- **Beispiel Audio-CD:**
 $44,1\text{kHz} * 2 \text{ Kanäle} * 16\text{bit} = 1,4 \text{ MBit/s}$
Speicherbedarf bei 60 min : 630 MB
- **Beispiel Telefonie:**
 $8\text{kHz} * 1 \text{ Kanal} * 8\text{bit} = 64\text{kBit/s}$

Multimediale Netzanwendungen

Kategorien:

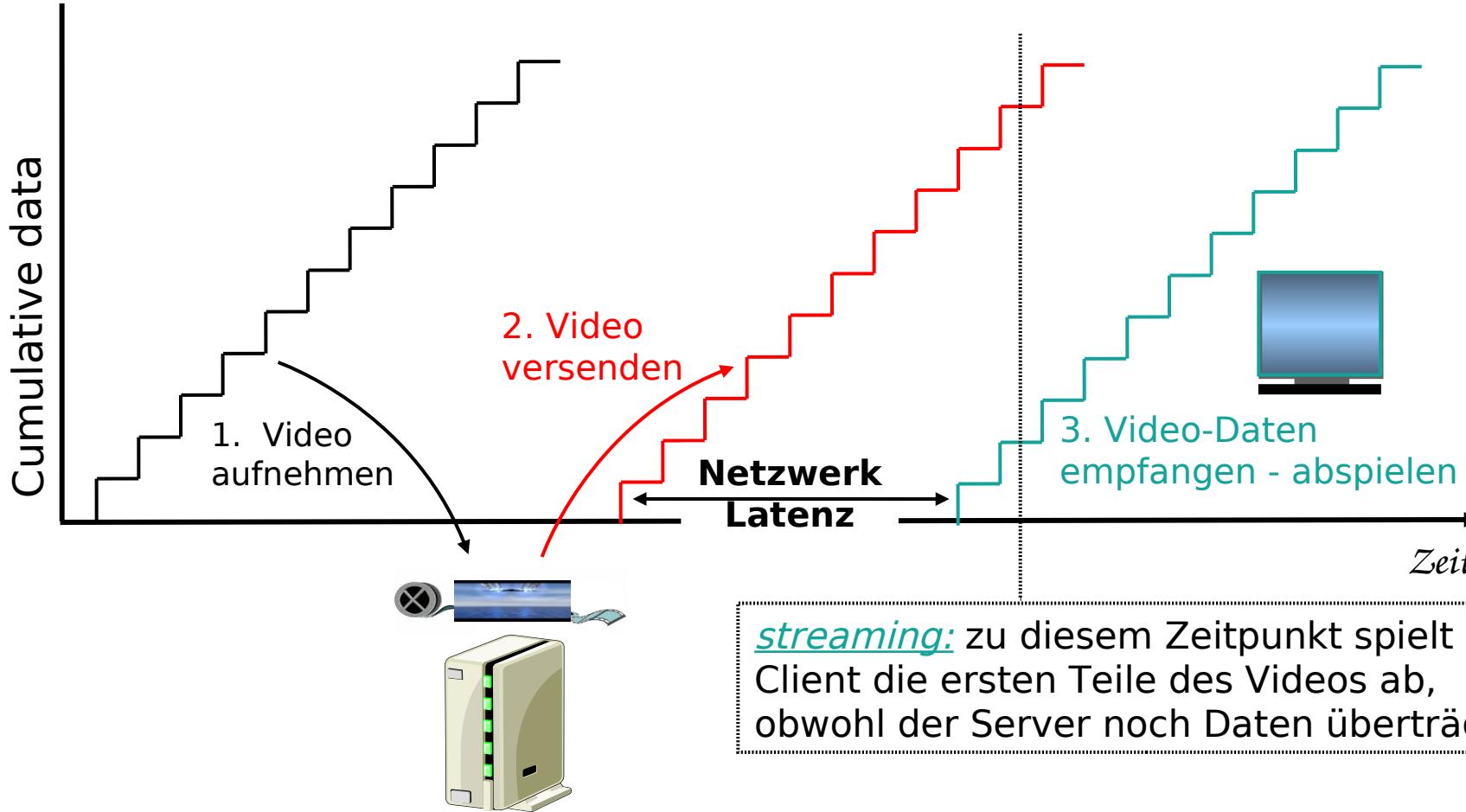
- 1) Streaming gespeicherter Audio- und Video-Daten
- 2) Streaming von live Audio und Video
- 3) Interaktive Audio und Video Nutzung

Streaming:

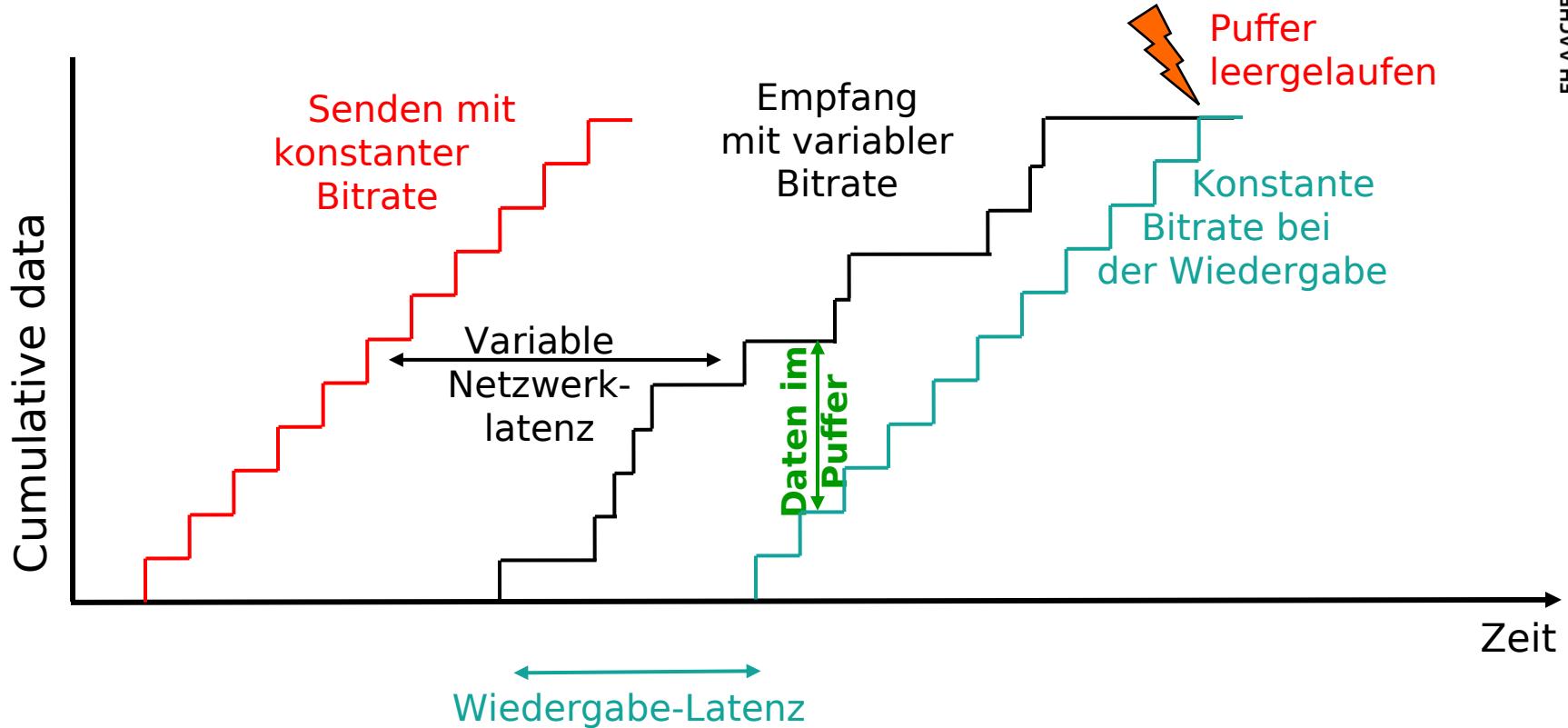
Das Verwenden (ausliefern) von Daten bevor diese vollständig übertragen wurden

→ Interpretation der Daten als (ggf. sehr lange dauernder) Datenstrom

Streaming gespeicherter multimedialer Daten



Streaming gespeicherter multimedialer Daten



Live-Streaming multimedialer Daten

Beispiel:

- Internet-Radio
- IPTV

Streaming:

- Playback-Puffer
- Zeitverschiebung zwischen Wiedergabe und originaler Zeit kann 10 Sekunden sein
- Zeitkritisch: Playback-Puffer darf nicht leerlaufen

Interaktivität

- Vorspulen kann nicht funktionieren
- Rückspulen und pausieren ist möglich

Wie kann das mit UDP realisiert werden?



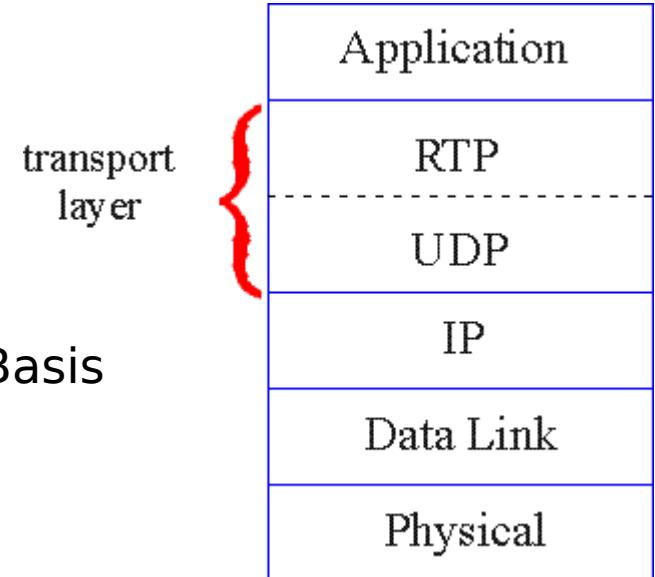
Real-Time Protocols

RTP/RTCP/RTSP

RTP: Eine UDP-Anwendung

Das **RTP**-Protokoll liefert Transportschnittstellen, die UDP erweitern:

- UDP liefert Port-Nummern
- IP die Adressen der Endpunkte
- RTP liefert u.a.
 - Kodierungskennung
 - Sequenznummern
 - Zeitstempel
- **RTCP** liefert Statistiken (auf UDP)-Basis
- **RTSP** ist unsere Fernbedienung



RTP Header:

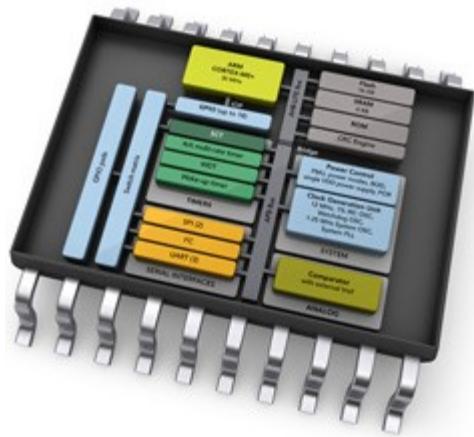
Byte 0								Byte 1								Byte 2								Byte 3																										
Bit 0	1	2	3	4	5	6	7	Bit 0	1	2	3	4	5	6	7	Bit 0	1	2	3	4	5	6	7	Bit 0	1	2	3	4	5	6	7																			
V=2	P	X	CC		M	PT				Sequence Number																																								
Timestamp (in sample rate units)																																																		
Synchronization Source (SSRC) identifier																																																		
Contributing Source (CSRC) identifiers (optional)																																																		
Header Extension (optional)																																																		

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge

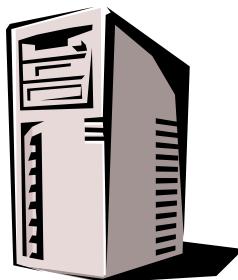


Domain Name System

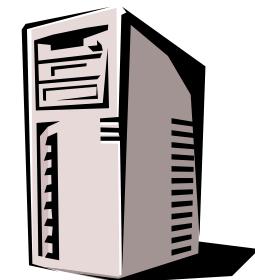
Lesbare Adressen im Internet

Semantische/mnemonische Namen (DNS)

- Socket-basierte Kommunikation nutzt IP-Adressen
- End-Anwender wollen sprechende Bezeichnungen mnemonisch
Eigenschaften von Bezeichnungen und Namen, wenn sie leicht zu merken sind und Rückschlüsse auf ihre Bedeutung haben.
- End-Anwender wollen stabile Namen für einen Service, und keine neuen IP nach einem Umzug des Servers
- **DNS** = Domain Name System
→ Namen müssen übersetzt und aufgelöst werden

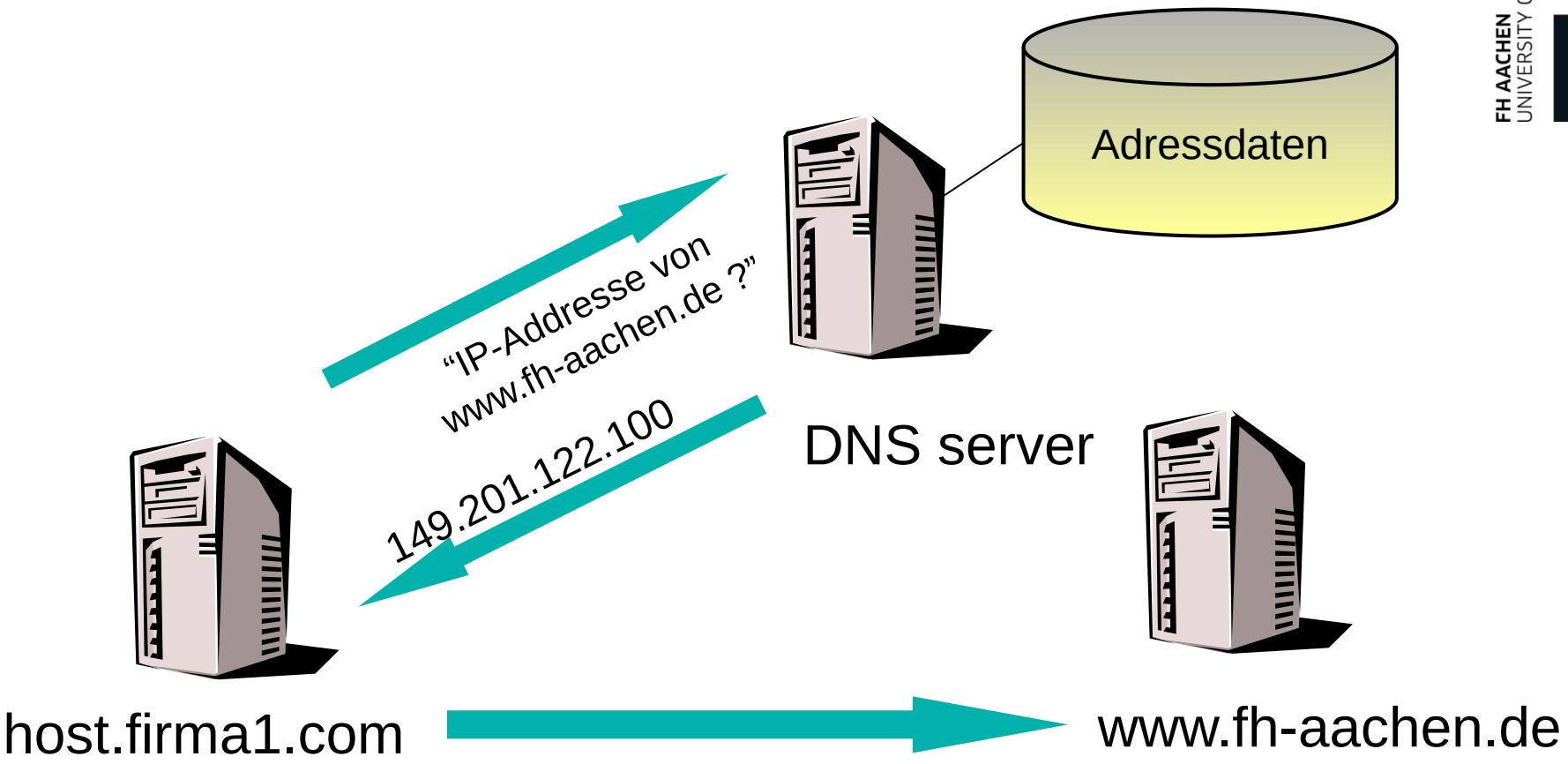


host.firma1.com

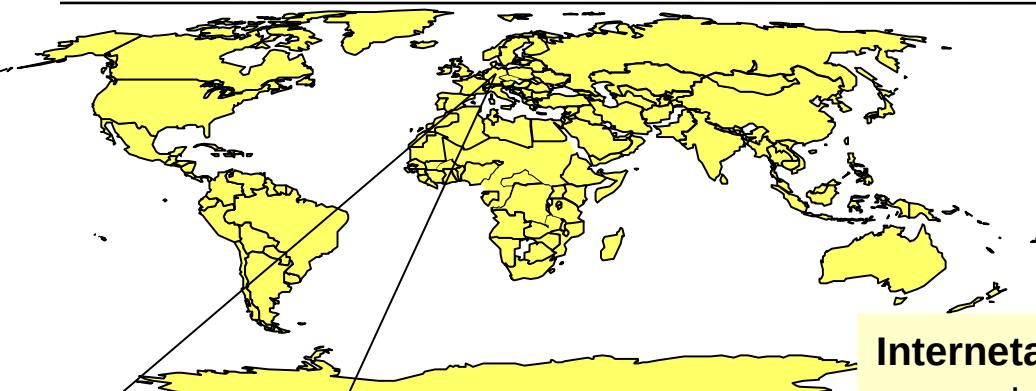


www.fh-aachen.de

Im Programmcode verankerter “Lookup”



DNS - Domain Name System



Top Level
Domain **de**

fh-aachen

agnm05

www.agnm05.fh-aachen.de



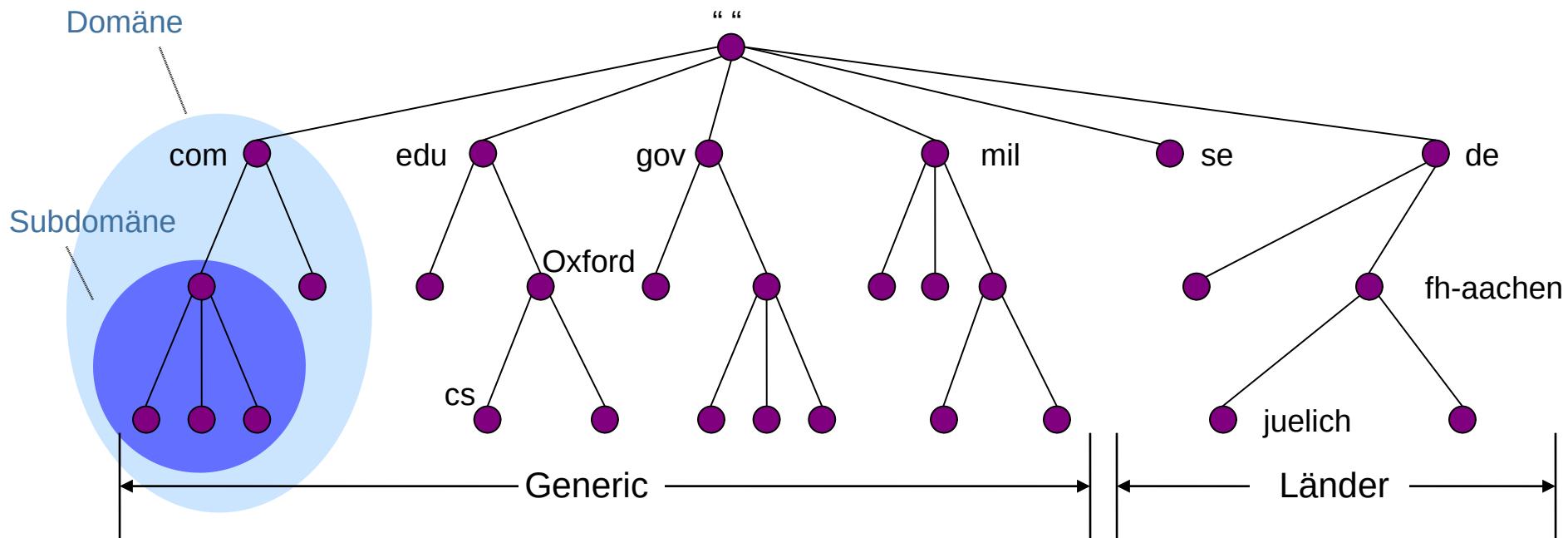
149.201.10.30

Internetadressen sind für Menschen schlecht zu merken, aber Rechner können damit perfekt umgehen.

Symbolische Name sind für Menschen einfacher zu handhaben, aber Maschinen können leider damit nichts anfangen.

Struktur der Datenbank

- Zur Strukturierung aller Informationen: Datenbank lässt sich als **Baum** darstellen
- jeder Knoten des Baums ist mit einem Label beschriftet, das ihn relativ zum Vaterknoten identifiziert
- jeder (innere) Knoten ist wiederum selber Wurzel eines Teilbaums
- jeder dieser Teilbäume repräsentiert eine **Domäne**
- jede Domäne kann wiederum weiter in **Subdomänen** unterteilt werden

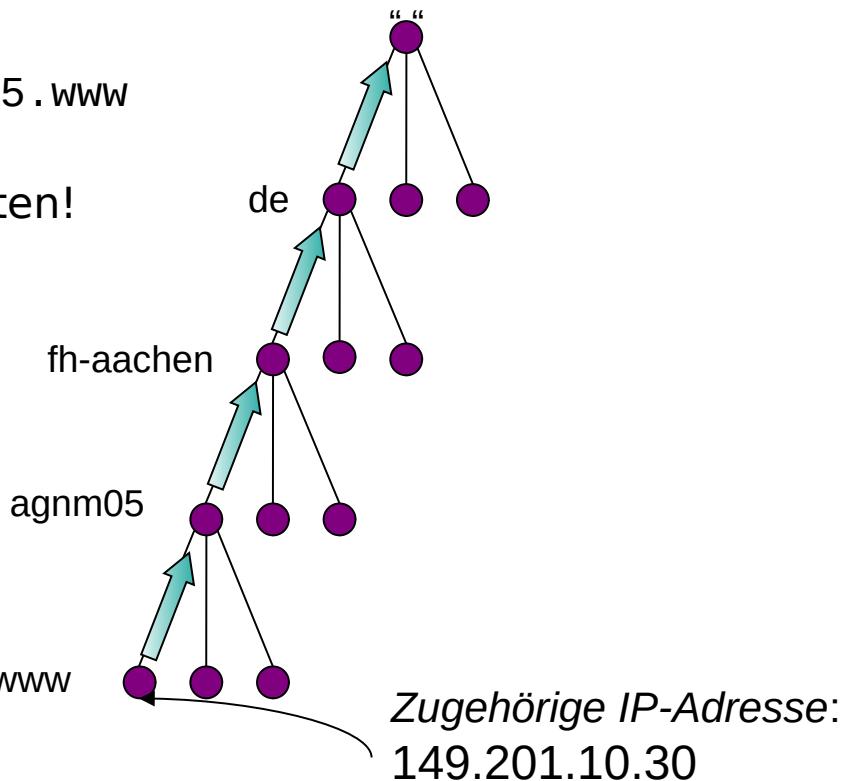


Struktur der Datenbank: Generische Top level Domains

Domain	Intended use	Start date	Restricted?
com	Commercial	1985	No
edu	Educational institutions	1985	Yes
gov	Government	1985	Yes
int	International organizations	1988	Yes
mil	Military	1985	Yes
net	Network providers	1985	No
org	Non-profit organizations	1985	No
aero	Air transport	2001	Yes
biz	Businesses	2001	No
coop	Cooperatives	2001	Yes
info	Informational	2002	No
museum	Museums	2002	Yes
name	People	2002	No
pro	Professionals	2002	Yes
cat	Catalan	2005	Yes

Domänennamen

- der Name einer Domäne besteht aus der Folge von Labeln (getrennt durch „.“) beginnend beim ‚Blatt‘ der Domäne und aufsteigend bis zur Wurzel des Gesamtbaums
- In den Blattknoten sind die IP-Adressen der durch die Labelsequenz gegebenen Namen gespeichert
- Eine Darstellung de.fh-aachen.agnm05.www wäre logischer, aber weniger lesbar
- Daher: Auflösung des Namens von hinten!

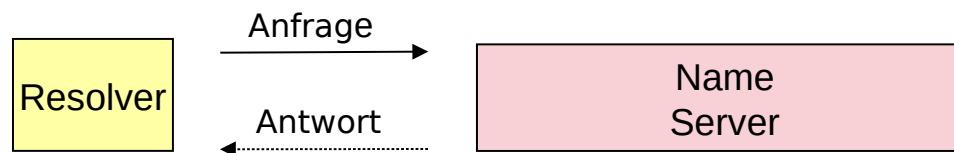


logischer Name:
www.agnm05.fh-aachen.de

Zugehörige IP-Adresse:
149.201.10.30

DNS - Konzept

1. DNS handhabt die Abbildung von Rechnernamen auf Adressen (**und weitere Dienste**)
2. DNS ist eine **verteilte Datenbank**, d.h. die einzelnen Segmente unterliegen einer *lokalen Kontrolle*
3. Die Struktur des verwendeten **Namensraums** der Datenbank **gibt die administrative Einteilung des Internets wider**
4. Daten jedes lokalen Bereichs sind mittels einer Client/Server-Architektur im gesamten Netzwerk verfügbar
5. Robustheit und Geschwindigkeit des Systems werden durch Replikation der Daten und Zwischenspeicherung (engl. *Caching*) erreicht
6. Hauptkomponenten:
 - **Name Server:** Server, die Informationen über einen Bereich der Datenbank verwalten
 - **Resolver:** Clients, die Anfragen an die Server stellen

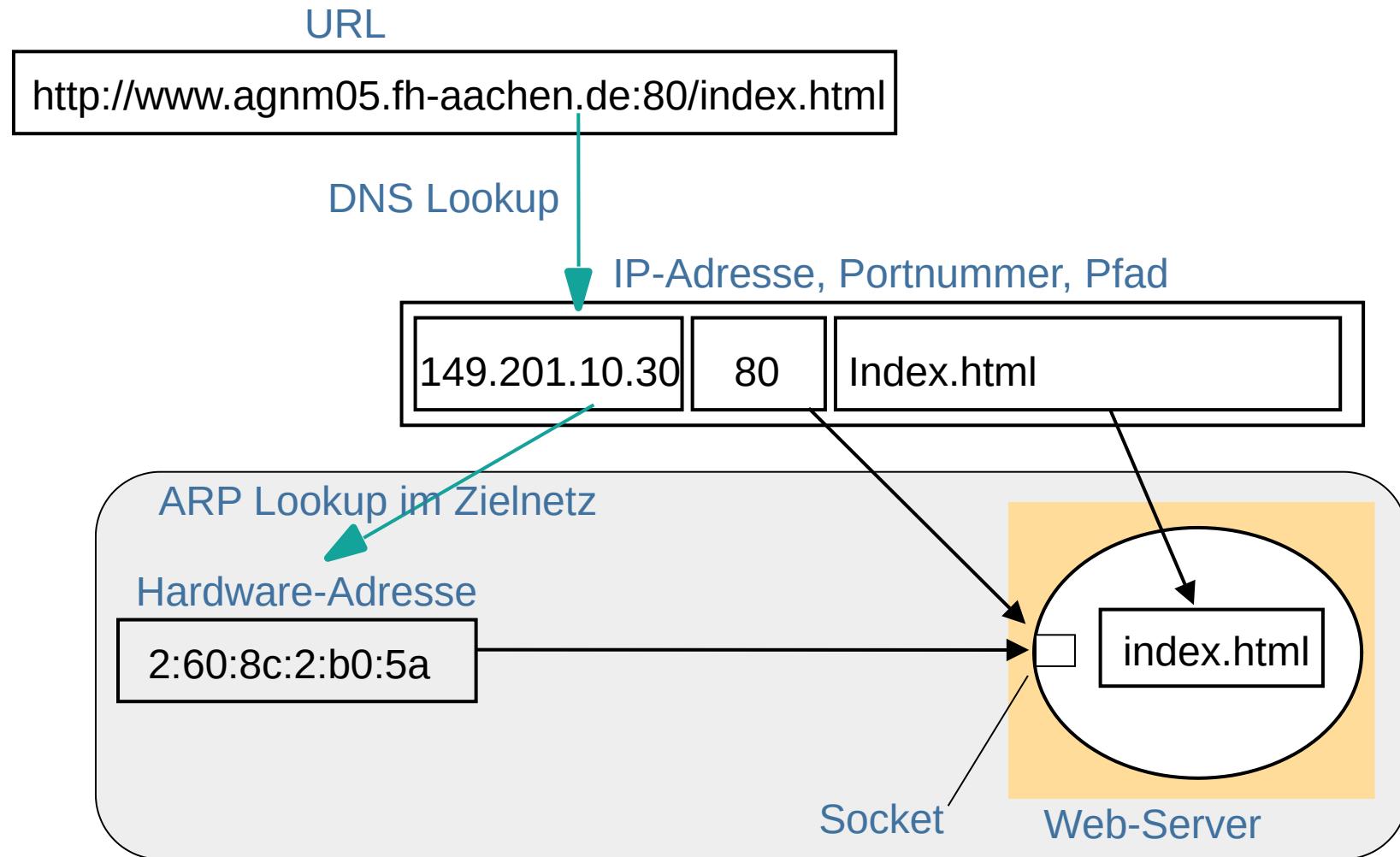


DNS-Server

- Server-Prozess im Netz (z.B. im DSL-Router)
- Wartet auf Anfragen (UDP!)
- Enthält eine **Tabelle** mit **Namen** von lokalen Hosts und zugehörigen **IP-Adressen** (nicht injektiv)
- Wird von den Klienten über Konfigurationsdatei oder DHCP gefunden
- Zusammenspiel mit lokaler Datei auf Resolver-Seite
 /etc/hosts (UNIX) c:\WINDOWS\system32\drivers\etc (Windows)
- Reihenfolge bei manchen Systemen einstellbar:
 /etc/resolv.conf /etc/nsswitch.conf (UNIX)

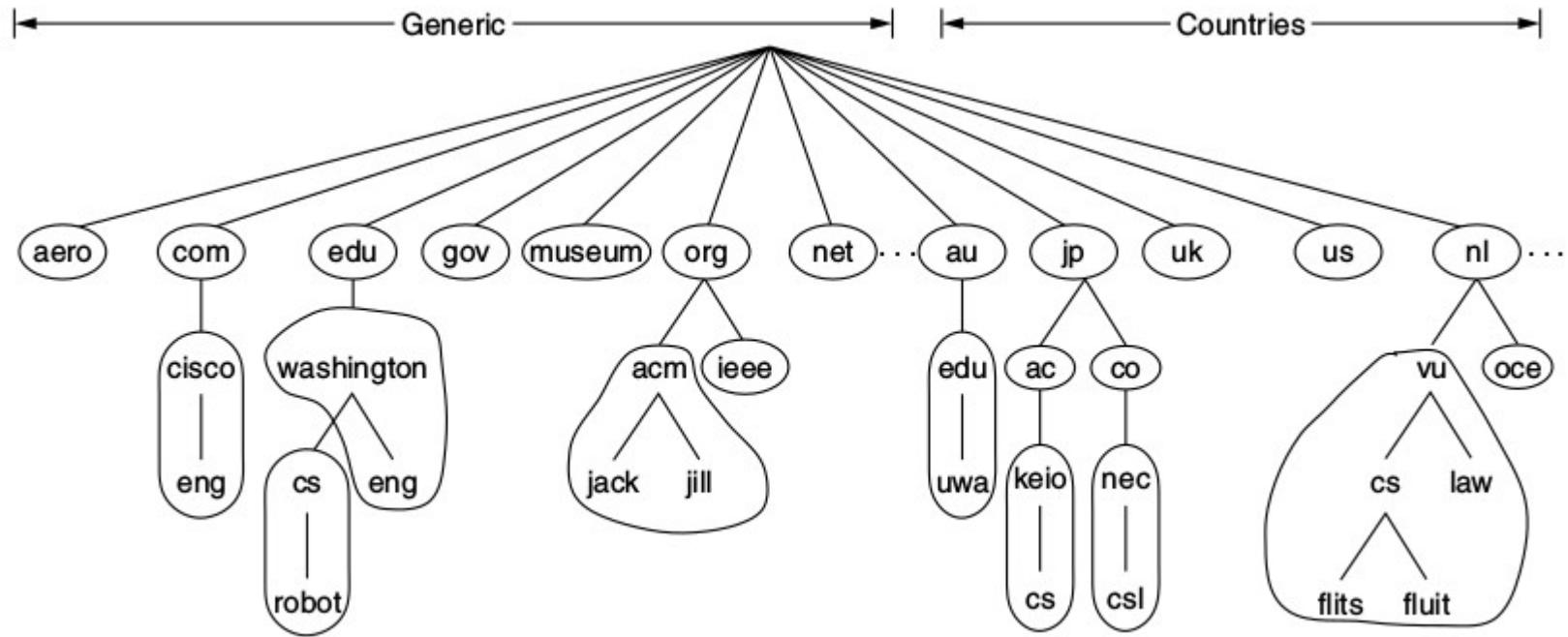


Zugriff auf entfernte Rechner



Domänen und Zonen

- Jeder Name Server verwaltet eine **Zone**, das ist ein Teil des Domänenbaumes
- Domäne und Zone sind unterschiedliche Konzepte:



- **Zonen** sind (außer in den ‚tieferen‘ Bereichen des Baumes) meistens nur für ein Namenselement einer Domänen zuständig (dann müssen vom Name Server weniger Informationen verwaltet werden)

DNS-Zonen

- Eine Zone ist ein autarker und gemeinsam administrierter Bereich im DNS-Namensraum
- Jede Zone hat einen **primären** und beliebig viele **sekundäre Nameserver (NS)**
 - Jeder NS kennt nur einen Ausschnitt des gesamten Namensraums
 - Jeder NS kennt **alle IP-Adressen** seiner **direkt** untergeordneten Sub-Domains
 - Sekundäre NS führen ein periodisches Update („Zonentransfer“) ihrer Datenbasis durch (vollständige Datenbestand des primären NS wird transferiert) (Master-Slave-Prinzip)
- Zur Einrichtung einer Zone muss der übergeordnete Knoten davon überzeugt werden, die Verwaltung zu delegieren
- Eine Zone ist ein Namensraum mit eigener Datenbank
 - Knoten im Baum (Bezeichnung darf maximal 63 Zeichen lang sein)
 - Bis zu 127 Ebenen (ohne Bezeichnung)

DNS: Root-Name-Server

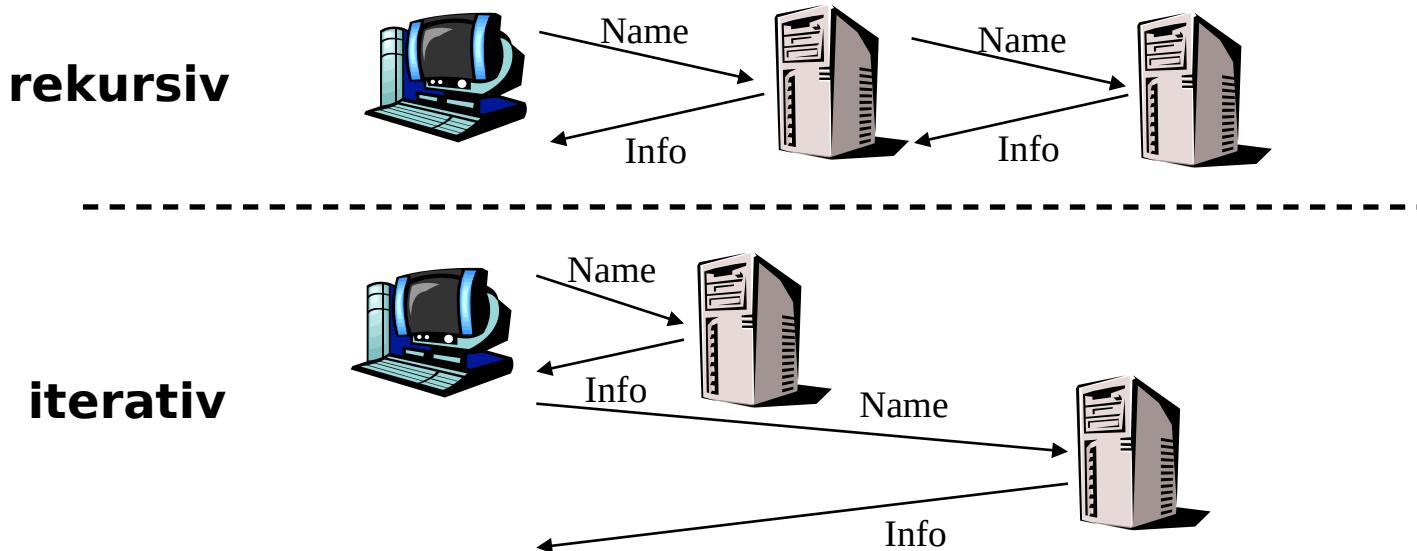
- Bilden die Wurzel der hierarchischen DNS-Struktur
- Es gibt weltweit 13 Root-Name-Server
 - z.Zt. {a-m}.root-servers.net
 - Die tatsächliche Anzahl der Server ist deutlich größer, da mit IPv6 hier Anycast-Adressen greifen können, so dass mehrere physikalische Server einen DNS-Root-Server realisieren können. Stichwort Anycast !
- Root-Server müssen hohe Anforderungen an die bearbeitbaren Lasten erfüllen
- Name-Server der tieferen Hierarchien werden mit festen Root-Servern konfiguriert
(über bekannte IP-Adressen, meistens IPv6).

DNS: Root-Name-Server Bekannte IPs

Letter	IPv4 address	IPv6 address	AS-number ^[9]	Old name	Operator
A	198.41.0.4	2001:503:ba3e::2:30	AS19836 ^{[9][note 1]} AS36619, AS36620, AS36622, AS36625, AS36631, AS64820 ^{[note 2][11]}	ns.internic.net	Verisign
B	199.9.14.201 ^{[note 3][12][13]}	2001:500:200::b ^[14]	AS394353 ^[15]	ns1.isi.edu	USC-ISI
C	192.33.4.12	2001:500:2::c	AS2149 ^{[9][17]}	c.psi.net	Cogent Communications
D	199.7.91.13 ^{[note 4][18]}	2001:500:2d::d	AS27 ^{[9][19]}	terp.umd.edu	University of Maryland
E	192.203.230.10	2001:500:a8::e	AS21556 ^{[9][21]}	ns.nasa.gov	NASA Ames Research Center
F	192.5.5.241	2001:500:2f::f	AS3557 ^{[9][22]}	ns.isc.org	Internet Systems Consortium
G ^[note 5]	192.112.36.4 ^[note 6]	2001:500:12::d0d ^[note 6]	AS5927 ^{[9][24]}	ns.nic.ddn.mil	Defense Information Systems Agency
H	198.97.190.53 ^{[note 7][25]}	2001:500:1::53 ^{[note 8][25]}	AS1508 ^{[25][note 9][26]}	aos.arl.army.mil	U.S. Army Research Lab
I	192.36.148.17	2001:7fe::53	AS29216 ^{[9][27]}	nic.nordu.net	Netnod

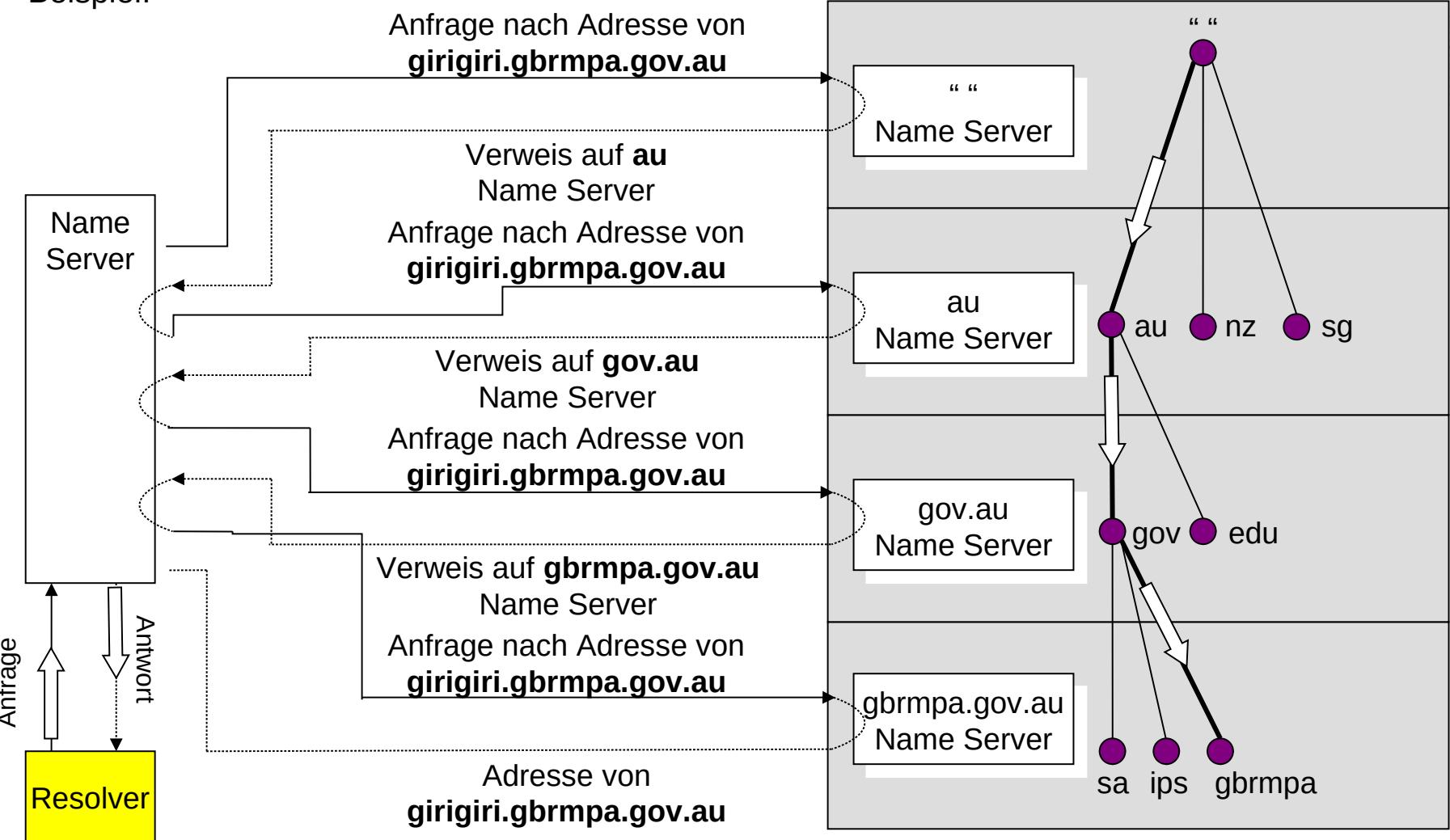
Die Auflösung: Rekursive und Iterative Anfragen

- Rekursive Anfragen -
Server schickt Anfrage zum nächsten Server weiter (oder Fehlermeldung). → Client-Server-Anfragen
- Iterative Anfragen -
Server antwortet dem Fragenden direkt mit IP-Adresse des nächsten Servers. → Server-Server-Anfragen



Namensauflösung: Iterativ

Beispiel:



DNS-Dienste

- DNS leistet mehr als nur die Auflösung von Namen
- Der **Mail-Exchange-Record** liefert zu einer Zone den Mail-Server (so erhält man das wissen, wer bei der FH-Aachen die Mails empfängt)
- Weitere Informationen können z.B. im Zusammenhang mit der IP-Telefonie geliefert werden
- Jeder DNS-Server besitzt auch einen **Cache**. Dieser hält vor kurzem aufgelöste Anfragen vor. Den Zeitraum liefert der Server, der diese Adresse ursprünglich einmal aufgelöst hat

DNS-Datenbank: : Domain Resource Records

```
; Authoritative data for cs.vu.nl
cs.vu.nl.      86400  IN  SOA   star boss (9527,7200,7200,241920,86400)
cs.vu.nl.      86400  IN  MX    1 zephyr
cs.vu.nl.      86400  IN  MX    2 top
cs.vu.nl.      86400  IN  NS    star

star           86400  IN  A     130.37.56.205
zephyr         86400  IN  A     130.37.20.10
top            86400  IN  A     130.37.20.11
www            86400  IN  CNAME star.cs.vu.nl
ftp             86400  IN  CNAME zephyr.cs.vu.nl

flits          86400  IN  A     130.37.16.112
flits          86400  IN  A     192.31.231.165
flits          86400  IN  MX   1 flits
flits          86400  IN  MX   2 zephyr
flits          86400  IN  MX   3 top

rowboat        IN  A     130.37.56.201
                IN  MX   1 rowboat
                IN  MX   2 zephyr

little-sister   IN  A     130.37.62.23

laserjet       IN  A     192.31.231.216
```

Beispiel: Einträge des Nameservers für cs.vu.nl

$24*60*60\text{s} = 86400\text{s}$

Typen von Einträgen

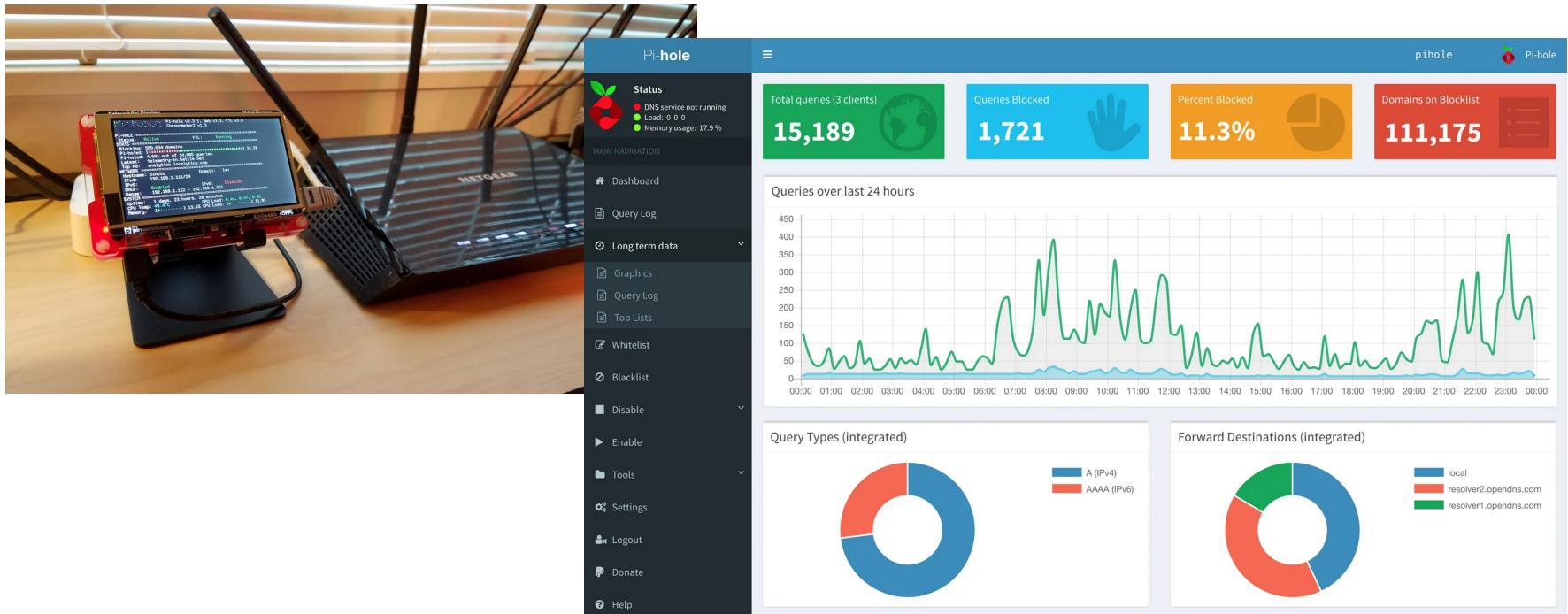
Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text

DNS-Attacken

- DNS ist ein durchaus kritischer Dienst
- Da er nur **UDP** verwendet, ist er häufig Angriffsziel:
 - DNS ID Hacking: Anfragen werden über IDs geschützt, d.h. man der Client erwartet nicht nur die Auflösung, sondern auch noch eine spezielle ID. Wenn diese nicht vom Netz abgegriffen werden kann, so muss man sie erraten
 - DNS spoofing: Hier beantwortet ein falscher DNS-Server die Anfrage. Hier muss die ID verwendet werden. Auch wird die falsche IP, also die des eigentlich richtigen Servers angegeben (was von den Providern zu unterbinden ist)
 - DNS Cache Poisoning: Hier wird versucht, einen eigentlich korrekten DNS-Server zu „verseuchen“. Idee ist, den Cache falsch zu füllen.

DNS-Filter

- DNS-Anfragen können mit einem Proxy gefiltert werden, um z.B. unerwünschte Anfragen zu unterdrücken
- Projekt Pi-Hole (Raspberry Pi wird zum DNS-Proxy, und leitet nur gewünschte DNS Anfragen weiter)

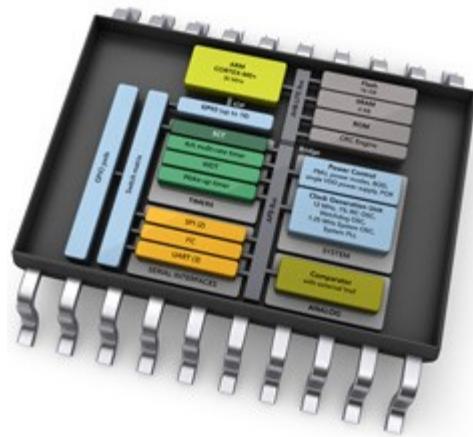


FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Anwendungsprotokolle

Kurzer Überblick über:

- HTTP
- SSL / TLS

- FTP / SFTP
- Email: SMTP / POP3 / IMAP
- Telnet / ssh
- SNMP

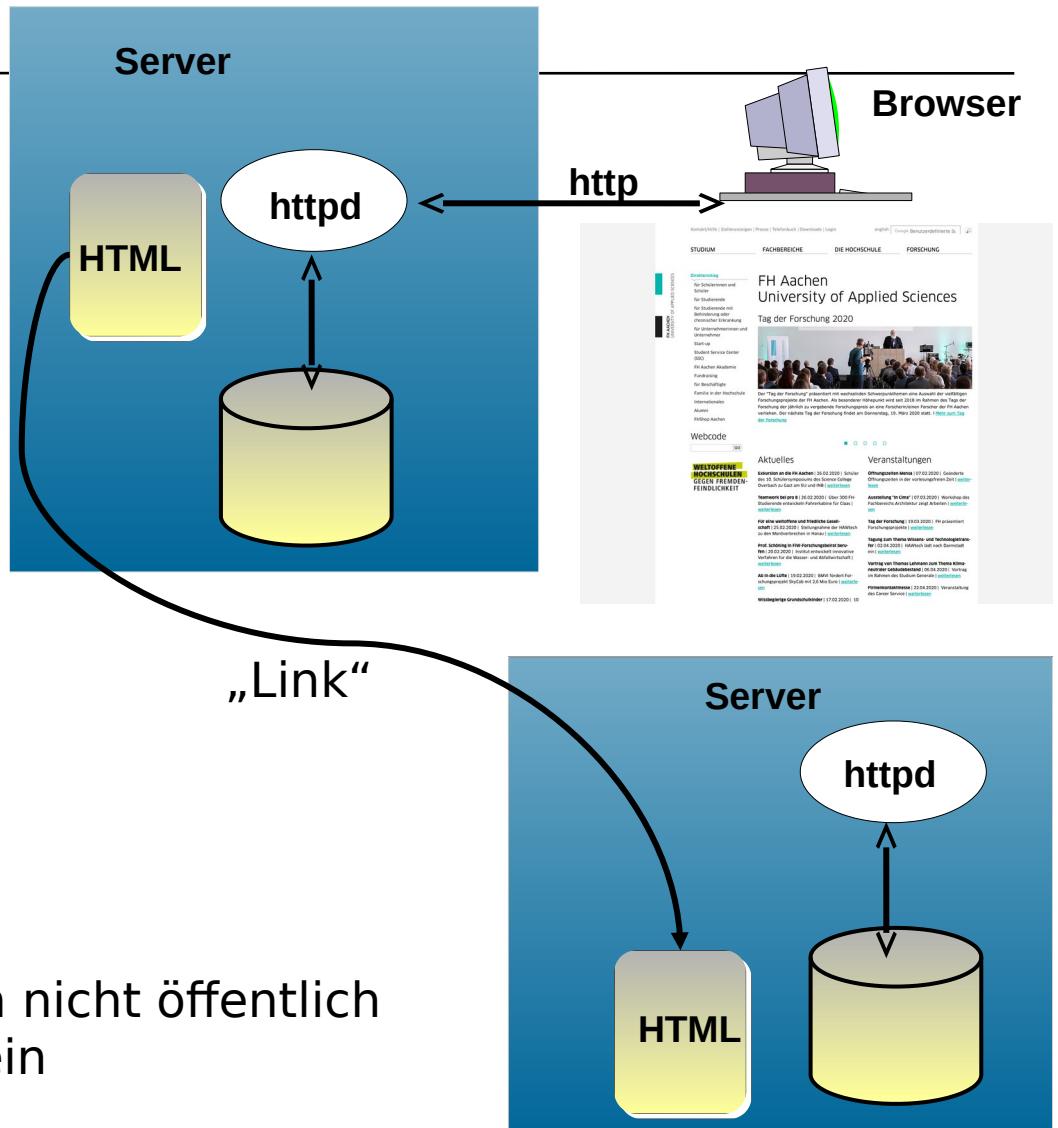
Anwendungsprotokolle

```
tcpmux          1/tcp          # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp          sink null
discard         9/udp          sink null
systat          11/tcp         users
daytime         13/tcp
daytime         13/udp
netstat          15/tcp
qotd            17/tcp          quote
chargen          19/tcp          ttyst source
chargen          19/udp          ttyst source
ftp-data        20/tcp
ftp              21/tcp
fsp              21/udp          fspd
ssh              22/tcp          # SSH Remote Login Protocol
telnet           23/tcp
smtp             25/tcp          mail
time             37/tcp          timserver
time             37/udp          timserver
whois            43/tcp          nickname
tacacs           49/tcp          # Login Host Protocol (TACACS)
tacacs           49/udp
domain           53/tcp          # Domain Name Server
domain           53/udp
bootps           67/udp
bootpc           68/udp
tftp              69/udp
gopher            70/tcp          # Internet Gopher
finger            79/tcp
http              80/tcp          www          # WorldWideWeb HTTP
kerberos         88/tcp          kerberos5  krb5  kerberos-sec   # Kerberos v5
...
...
```

Linux: Zuordnung
der Portnummern in
/etc/services

Web-Anwendungen: HTTP

- Internet-basierte Client/Server-Architektur
 - Browser (Client) zur graphischen Darstellung
 - HTTP-Server zur Übertragung der Daten und Dokumente
- **HTML:** Sprache zur Beschreibung der Seiten
- **HTTP:** Protokoll zur Übertragung der Seiten
- **TCP:** Von HTTP verwendetes Transportprotokoll
- **URL:** Spezifikation von Ort und Zugriffsmodalitäten



URI (Uniform Resource Identifier)

URI dient der eindeutigen Adressierung von abstrakten und physikalischen Ressourcen im Internet (Spezifikation in RFC2396)

URI: URLs u URNs

URI

URL (Uniform Resource Locator)

Adressierung von Informationsobjekten mit Festlegung des Zugangs-Protokolls (Ort der Ressource). RFC2141

URN (Uniform Resource Name)

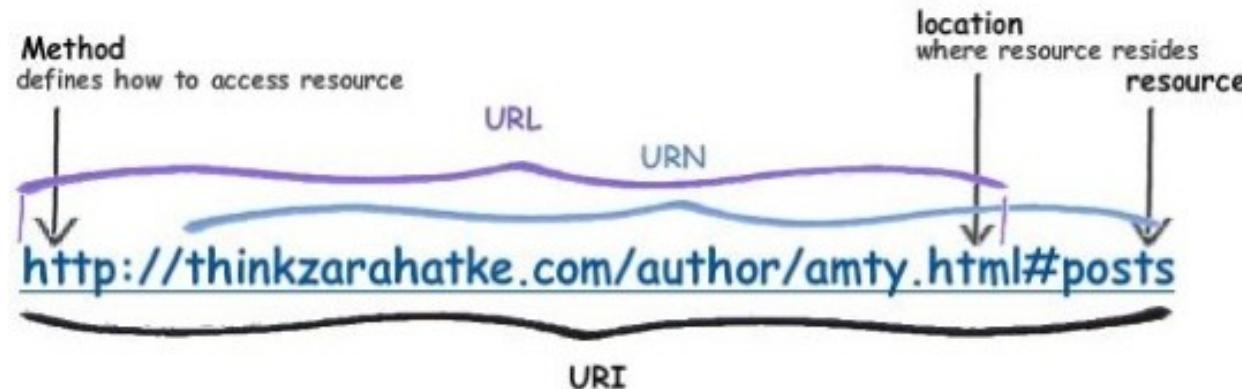
Adressierung von Objekten ohne ein Protokoll festzulegen (Eindeutige und gleichbleibende Referenz – Name der Ressource). RFC1738

URI (Uniform Resource Identifier)

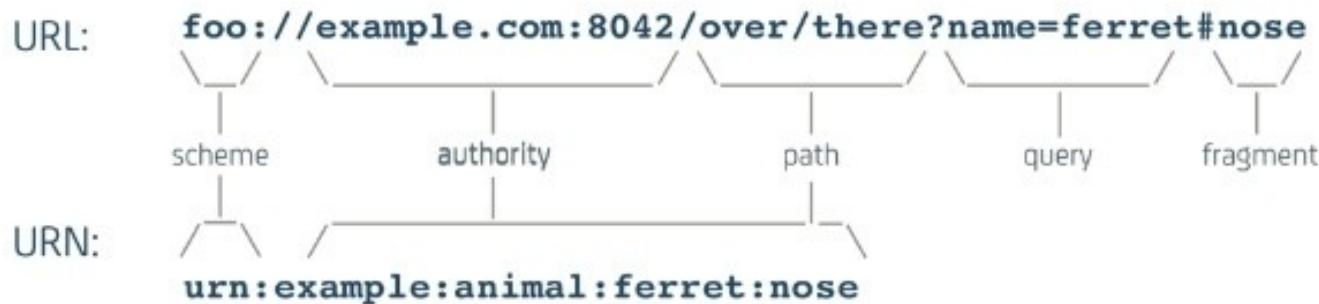
URI

URL

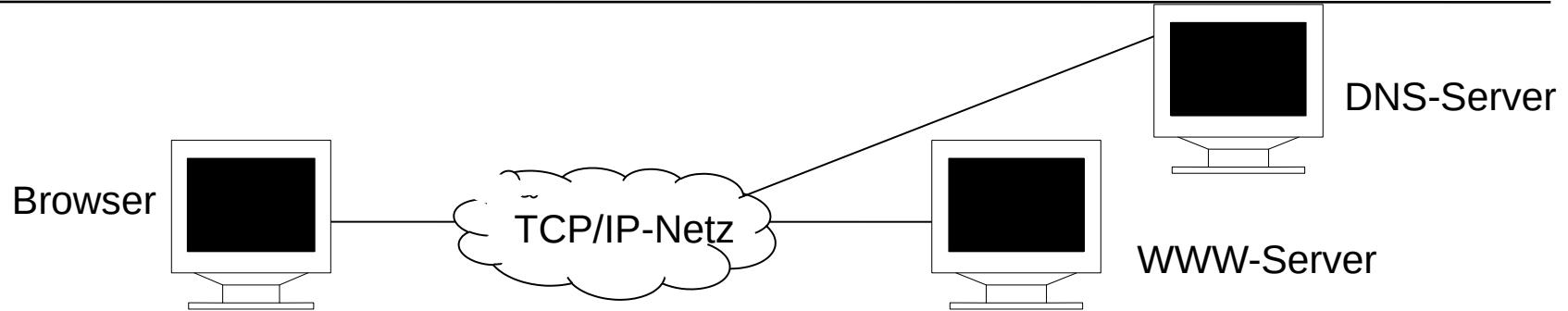
URN



The structure of URIs



Abruf von Webseiten



Browser fragt DNS nach der IP-Adresse des Servers



DNS antwortet



Browser öffnet eine TCP-Verbindung zu Port 80 des Rechners



Browser sendet das Kommando GET /material/allgemein.html



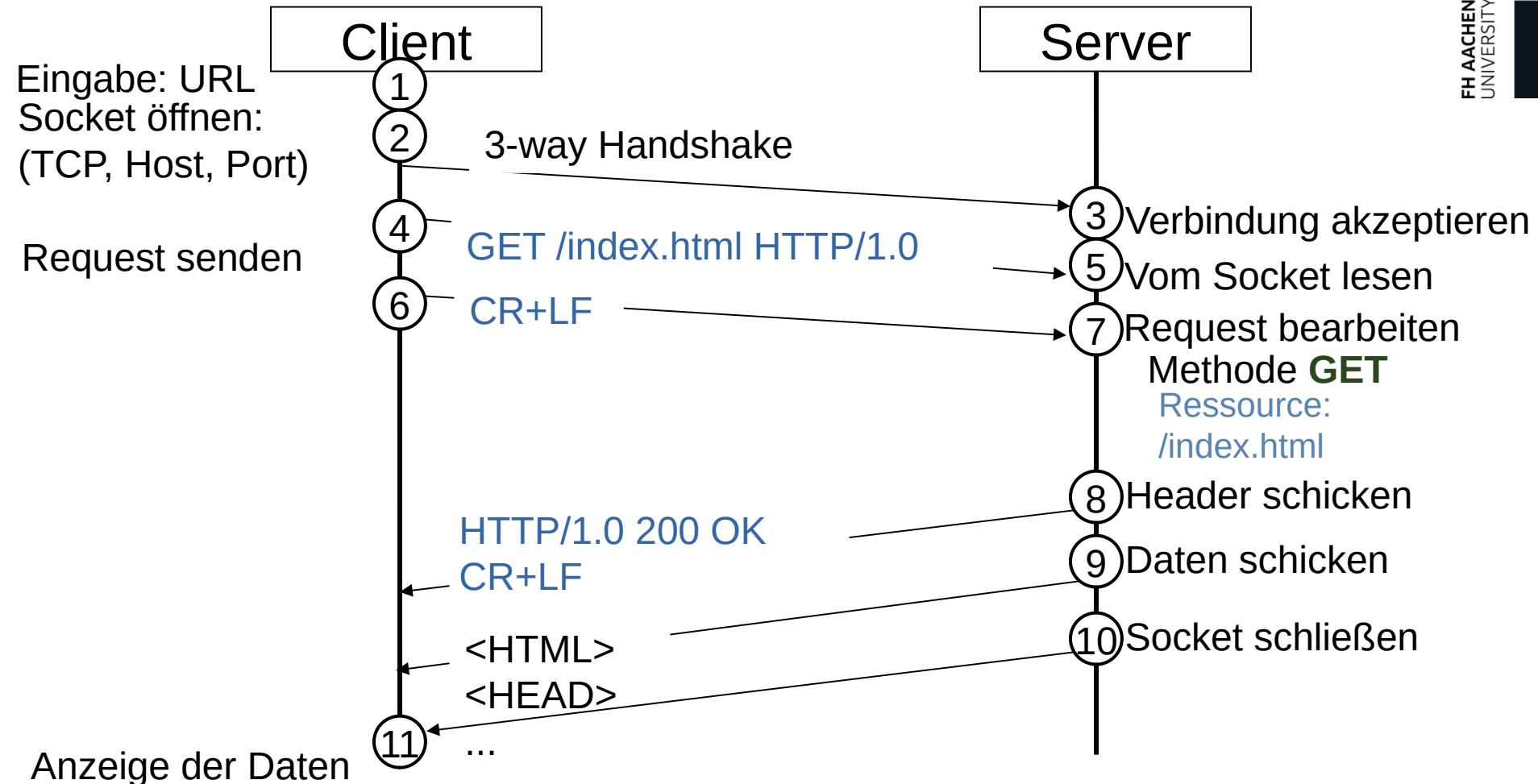
WWW-Server schickt die Datei allgemein.html zurück



Verbindung wird wieder abgebaut.



Der Ablauf einer HTTP-GET-Anfrage



HTTP Request Header

method	sp	URL	sp	version	cr	If
header field name	:	value	cr	If		
header field name	:	value	cr	If		
:						
header field name	:	value	cr	If		
cr	If					
data						

Request line: notwendiger Teil, z.B.

GET server.name/path/file.type

Header lines: optional, weitere Angaben zum Host/Dokument, z.B.

Accept-language: fr

Entity Body: optional. Weitere Angaben, falls der Client Daten überträgt (*POST-Method*)

HTTP Request

Request-Beispiel (GET):

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
           (KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-DE, de;q=0.8, en-US;q=0.6, en;q=0.4
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: gzip, deflate, sdch
Connection: keep-alive
```

HTTP Response Header

version	sp	status code	sp	phrase	cr	If
header field name	:	value	cr	If		
header field name	:	value	cr	If		
:						
header field name	:	value	cr	If		
cr	If					
data						

Entity Body: erfragte Daten

Status line: *status code* und *phrase* übertragen das Ergebnis einer Anfrage und eine zugehörige Meldung, z.B.

200 OK

400 Bad Request

404 Not Found

Gruppen von Status-Meldungen:

1xx: Nur zur Information

2xx: Erfolgreiche Anfrage

3xx: Redirection, es müssen weitere erforderliche Aktivitäten durchgeführt werden

4xx: Client-Fehler (Syntax)

5xx: Server-Fehler

HTTP Response

Response-Beispiel:

HTTP/1.1 200 OK

Date: Wed, 26 Jun 2013 16:36:27 GMT

Server: Apache

Content-Type: text/html; charset=UTF-8

Content-Length: 12313

Last-Modified: Mon, 16 Apr 2013 20:27:06 UTC

Connection: keep-alive

<!DOCTYPE html>

<html>

...

HTTP Anfragen/Methoden

GET *retrieve* information
HEAD *retrieve* resource headers

POST *submit* data to the server.
PUT *save* an object at the location
DELETE *delete* the object at the location

HTTP

Binärdaten und das textbasiert

HTTP ist textbasiert (eMail auch!)

Wie können binäre Daten übertragen werden?

- Die Antworten des Servers auf eine vollständige GET-Request beinhaltet MIME-Informationen
- **MIME = Multipurpose Internet Mail Extensions**
- Definiert die Kodierungsregeln für Nicht-ASCII-Nachrichten
- MIME ermöglicht die Nutzung verschiedener Kodierungen (media types) in einer Nachricht

Die “Content-Type:”-Zeile im MIME-Header legt den Datentyp (type/subtype) einer Nachricht fest

Content-Transfer-Encoding: definiert die Transfersyntax, in der die Daten des Hauptteils übertragen werden, wird aber bei HTTP nicht benutzt

- Content-Encoding und Transfer-Encoding Felder

Beispiele:

- Content-Type: text/html
- Content-Type: image/GIF

HTTP

Binärdaten und das textbasiert

```
MIME-Version: 1.0
Content-Type: MULTIPART/MIXED;
  BOUNDARY= "8323328-2120168431-824156555=:325"
--8323328-2120168431-824156555=:325
Content-Type: TEXT/PLAIN; charset=US-ASCII
A picture is in the appendix
--8323328-2120168431-824156555=:325
Content-Type: IMAGE/JPEG; name="picture.jpg"
Content-Transfer-Encoding: BASE64
Content-ID: <PINE.LNX.3.91.960212212235.325B@localhost>

/9j/4AAQSkZJRgABAQEAlgCWAAD/
2wBDAAEBAQEBAQEBAQEBAQIBAQIBAQECAGICAgICAgIDAwQDAwMDA
wICAwQDAwQEBAQEAgMFBQQEBQQEBAT/2wBDAQEBAQIBAQIEAwIDBAQEBA
[...]
KKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAoooo
AKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAo
oooAD//Z
---8323328-2120168431-824156555=:325 --
```

Virtual Hosts

- Auf einem Rechner sollen verschiedene Domains und Web-Server zur Verfügung stehen
 - Jeder Server hat die gleiche IP, aber ggf. unterschiedliche DNS-Namen!
- Ein oder mehrere Webserver (Software) sollen die Anfragen, für die auf dem Rechner vorhandenen Domains, beantworten
- Typische Anwendung: Web-Hosting (Provider)

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
            (KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-DE, de;q=0.8, en-US;q=0.6, en;q=0.4
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: gzip, deflate, sdch
Connection: keep-alive
```

HTTP Evolution

Standard:	RFC 1945 ↗ HTTP/1.0 (1996) RFC 2616 ↗ HTTP/1.1 (1999) RFC 7540 ↗ HTTP/2 (2015) RFC 7541 ↗ Header Compression (2, 2015) RFC 7230 ↗ Message Syntax and Routing (1.1, 2014) RFC 7231 ↗ Semantics and Content (1.1, 2014) RFC 7232 ↗ Conditional Requests (1.1, 2014) RFC 7233 ↗ Range Requests (1.1, 2014) RFC 7234 ↗ Caching (1.1, 2014) RFC 7235 ↗ Authentication (1.1, 2014)
-----------	---

HTML Evolution

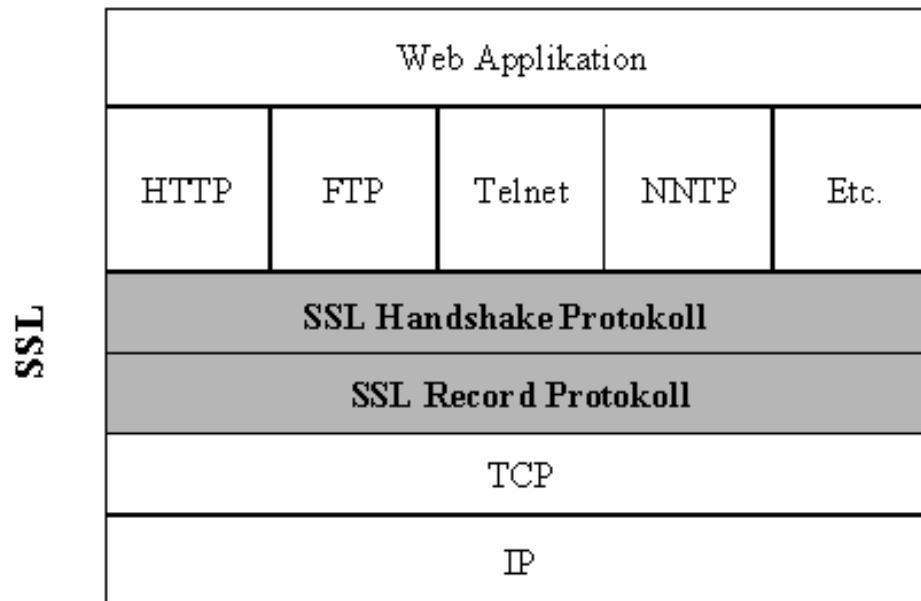
Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	X	X	X	X	X
Images	X	X	X	X	X
Lists	X	X	X	X	X
Active maps & images		X	X	X	X
Forms		X	X	X	X
Equations			X	X	X
Toolbars			X	X	X
Tables			X	X	X
Accessibility features				X	X
Object embedding				X	X
Style sheets				X	X
Scripting				X	X
Video and audio					X
Inline vector graphics					X
XML representation					X
Background threads					X
Browser storage					X
Drawing canvas					X

Absicherung der Kommunikation: https

- Kommunikation kann durch Einsatz des Secure-Socket-Layer-Protokolls (SSL) abgesichert werden
 - SSL wurde von der Firma Netscape entwickelt und ist unter dem Namen Transport Layer Security (TLS) standardisiert (RFC2246)
 - SSL sichert die Transportschicht ab - Basis ist das RSA-Verfahren
 - SSL ist verbindungsorientiert
- Das Secure Socket Layer-Protokoll hat drei grundlegende Eigenschaften
 1. Eine Verbindung ist privat. Beim Verbindungsaufbau wird ein Sitzungsschlüssel bestimmt, der dann zur Verschlüsselung benutzt werden kann
 2. Die beiderseitige Authentifikation (mutual authentication) wird ermöglicht
 3. Eine Verbindung wird immer durch eine Signatur abgesichert

SSL im ISO-OSI - Stack

- Transparente Schicht zwischen TCP/IP und Application Layer
- Universell einsetzbar



SSL - Sichere Datenübertragung

- Zur Verschlüsselung gibt es mehrere Verfahren:
 - symmetrische Verschlüsselung:
Ein Schlüssel zum Ver- und Entschlüsseln
 - asymmetrische Verschlüsselung:
Zwei Schlüssel zum Ver- und Entschlüsseln. Davon ist typischerweise einer öffentlich, und einer privat.
- Das SSL Protokoll muss mehrere Teilaufgaben lösen:
- **Authentifizierung:** WER redet mit mir?
- **Tauschen eines gemeinsamen Schlüssels zur symmetrischen Verschlüsselung**

SSL - Handshake Protokoll

- Das SSL Handshake Protokoll erledigt genau diese Aufgaben:
 - Den stärksten gemeinsam unterstützten Algorithmus ermitteln
 - Authentifikation der Kommunikationspartner (Client optional)
 - Ermitteln eines Session Keys zur symmetrischen Verschlüsselung (optional)

X509 Zertifikate: Wem gehört der public Key?



Idee:

$\{ \text{Klartext} \text{private_key} \text{public_key} =$
 $\{ \text{Klartext} \text{public_key} \text{private_key} =$
Klartext

X509 Zertifikate

**Aussteller der
die Identität
validiert hat**

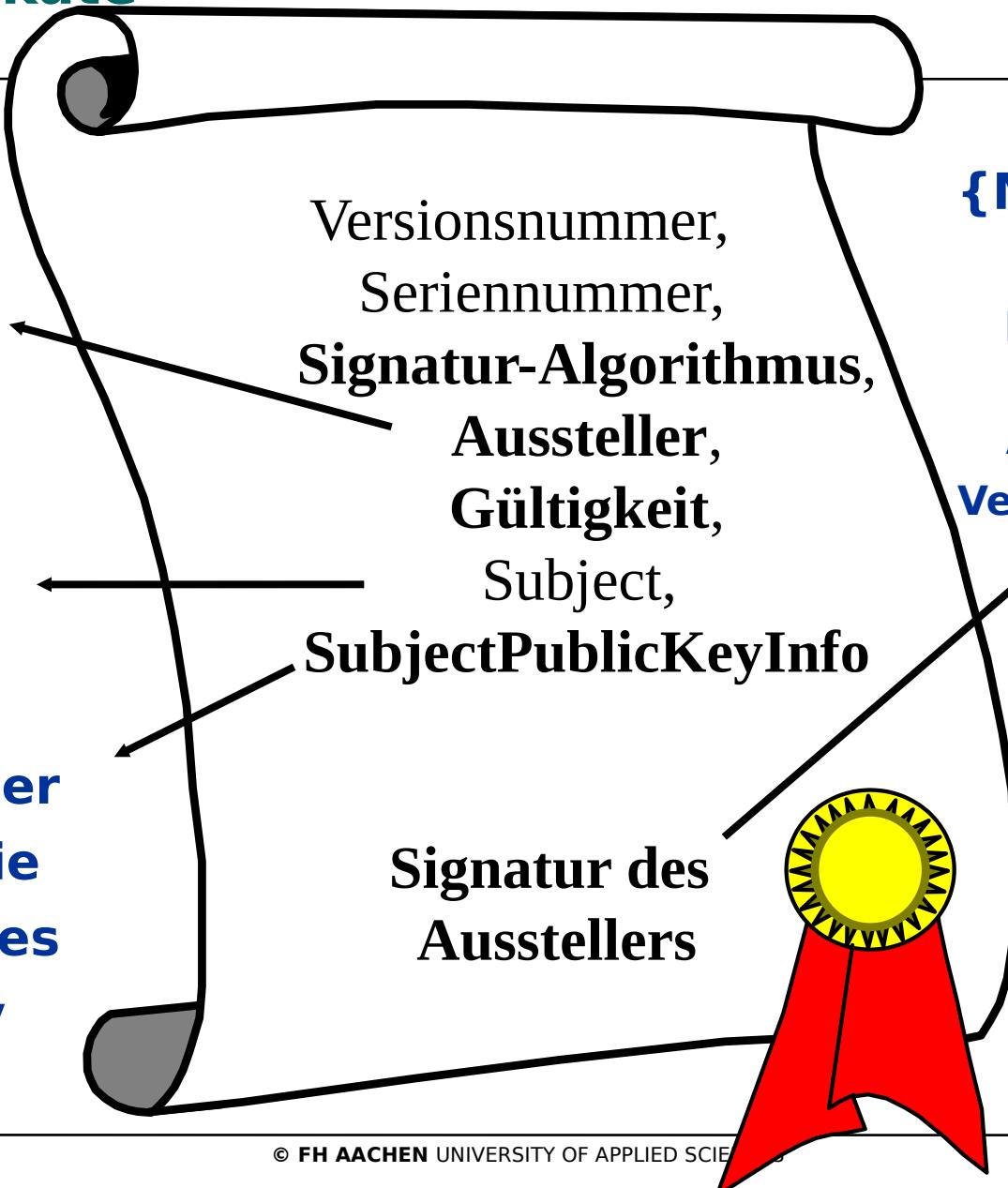
**Die zu
prüfende
Identität**

**Der Aussteller
bestätigt die
zuordnung des
Public Key**

Versionsnummer,
Seriennummer,
Signatur-Algorithmus,
Aussteller,
Gültigkeit,
Subject,
SubjectPublicKeyInfo

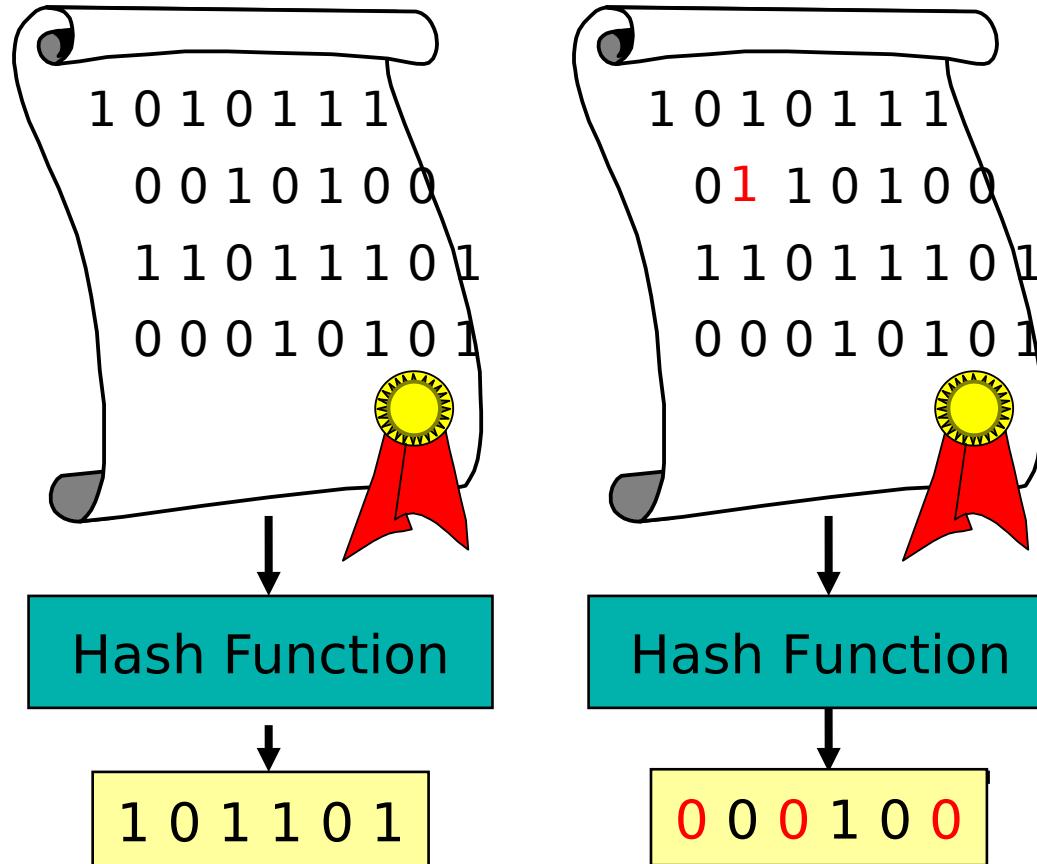
**Signatur des
Ausstellers**

{**MD5 hash**}
Mit dem
private key
des
Ausstellers
Verschlüsselter
Hash



Signaturen: Einweg-Hash-Funktionen

Dokument
oder Nachricht
beliebiger
größe

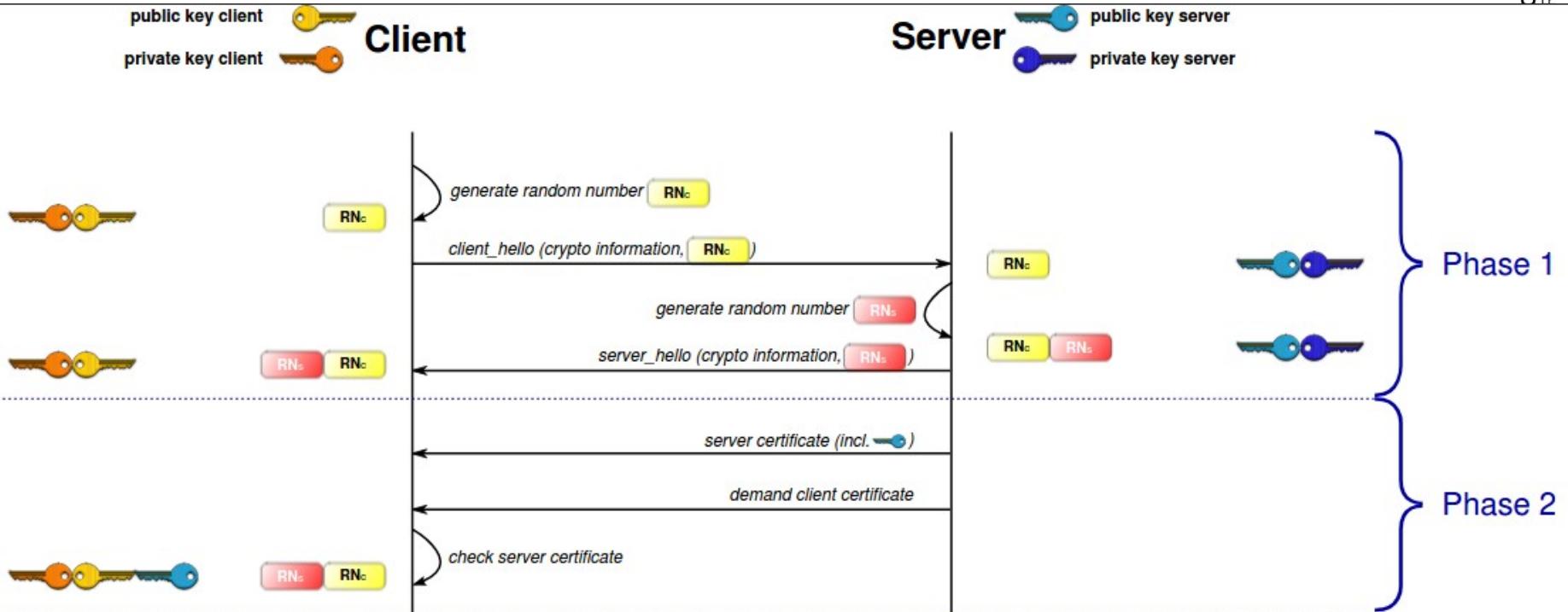


- Die Änderung eines einzelnen Bits im Dokument sollte ungefähr 50% der Hash-bits verändern!

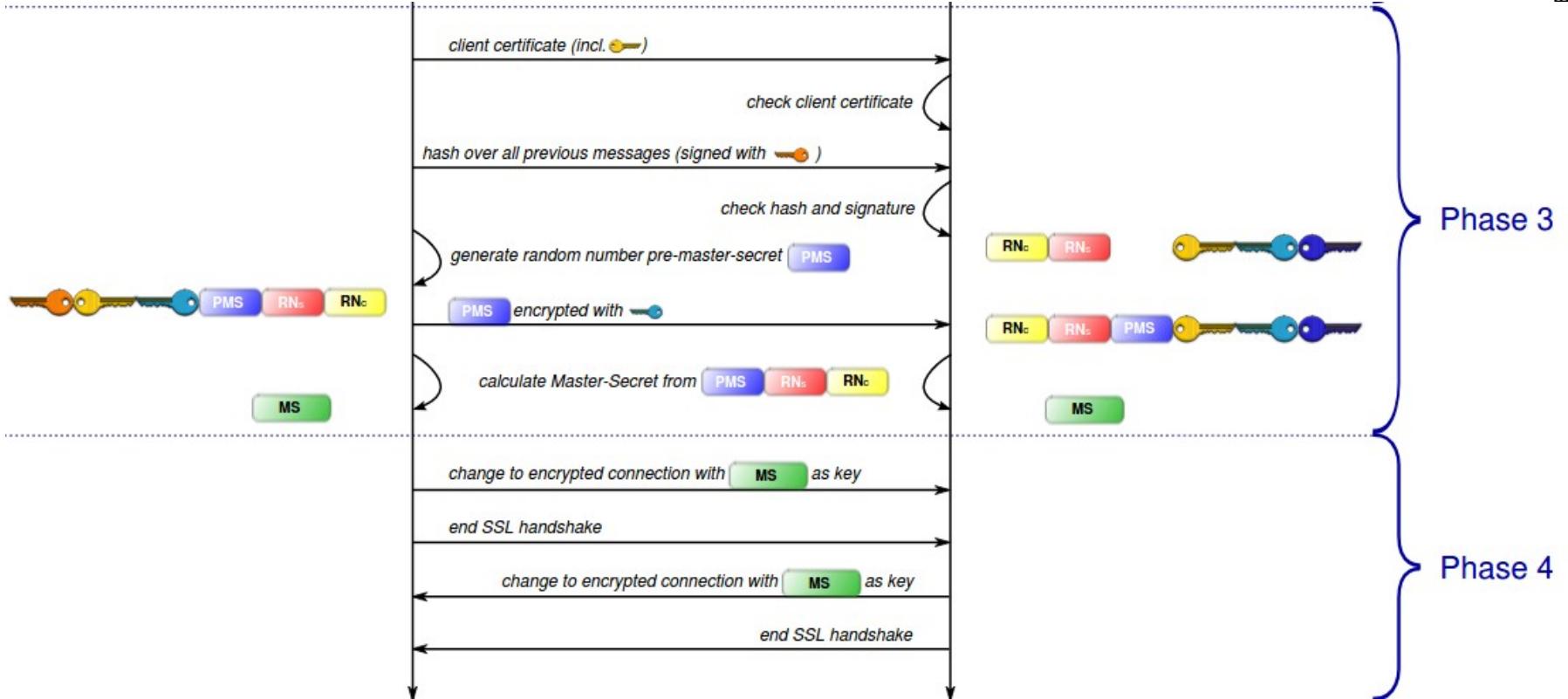
SSL Handshake (vereinfacht)

- Client-Hello-Nachricht:
 - Unterstütze Algorithmen, Zufallsnummer des Clients
 - Session ID (Abkürzung möglich)
- Server-Hello:
 - Ausgewählter Algorithmus, Zufallsnummer des Servers
 - Session ID
- Server Certificates
- Server Hello Done
- Client Key Exchange
 - Nach Prüfung des Zertifikats wird mit dem Public Key des Servers die Basis des Sessions Keys verschlüsselt
- Optional Client Zertifikat + mit Private Key verschlüsselte Zufallsnummer
- Beiderseitiges Finished

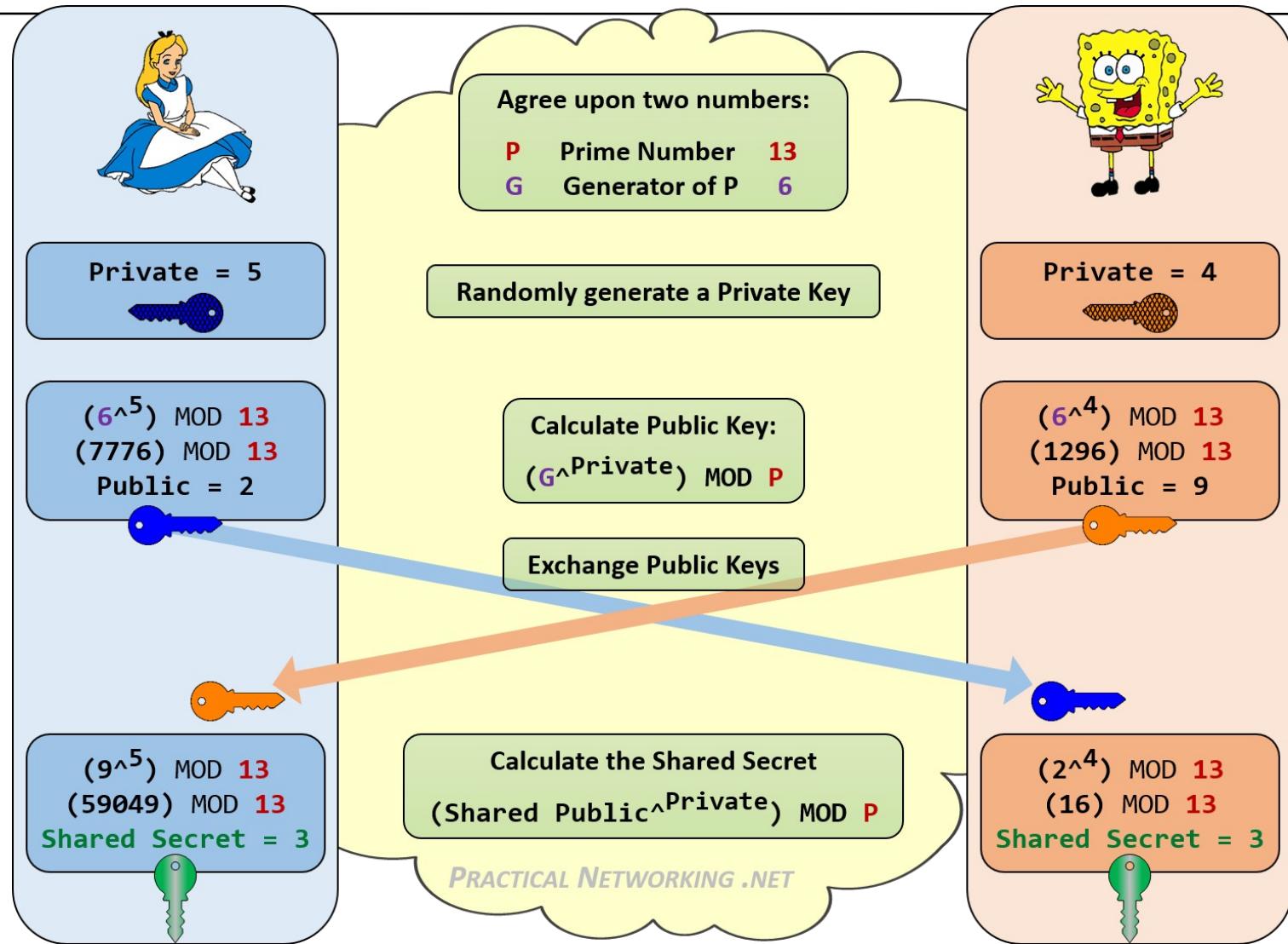
SSL Handshake 1 (vereinfacht)



SSL Handshake 2 (vereinfacht)



Diffie-Hellman-Schlüsselaustausch



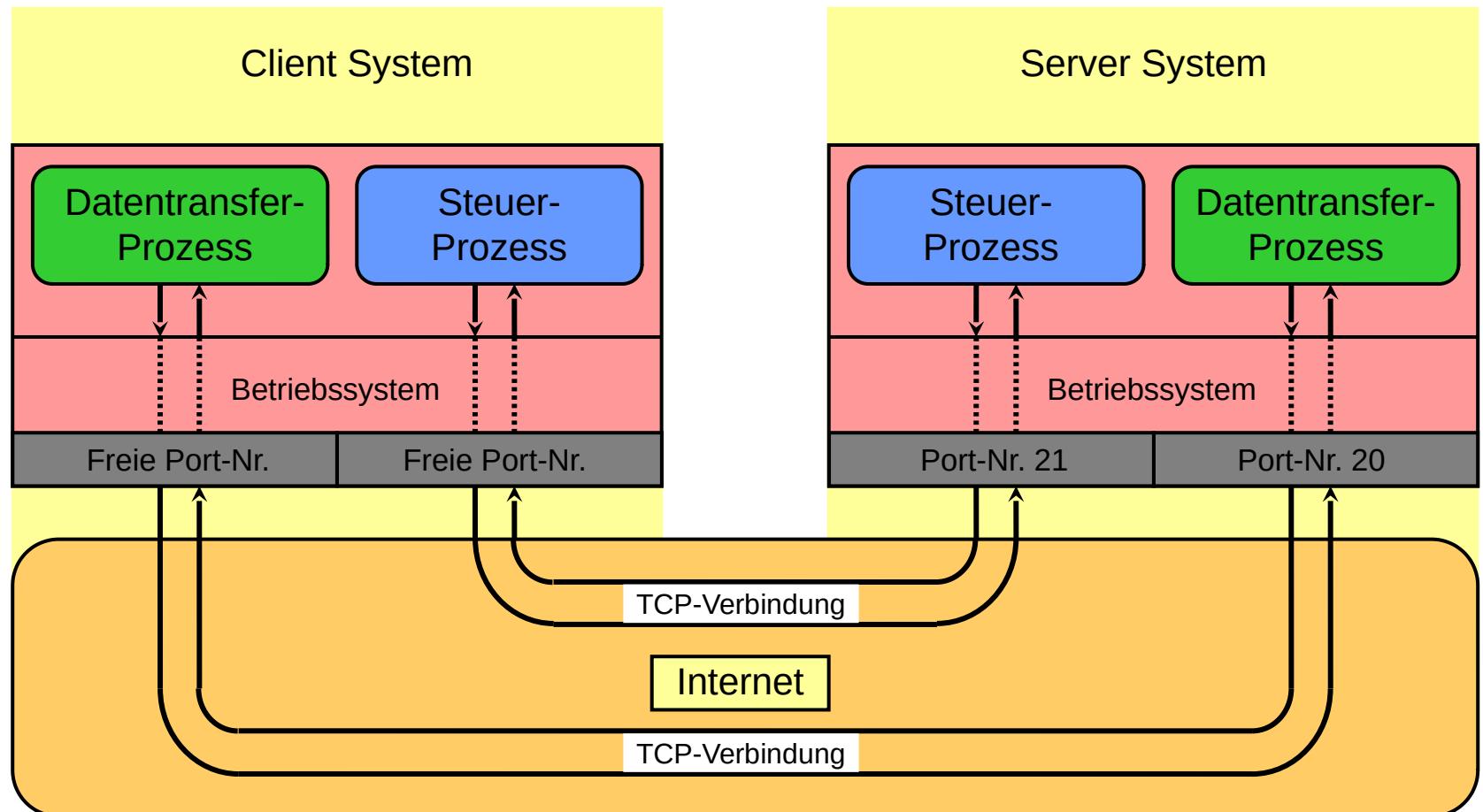
SSL Record Layer

- Vollständig getrennt vom Handshake Protokoll
- Verschickt Daten symmetrisch mit dem im Handshake ausgehandelten Verschlüsselungsalgorithmen und Session Keys
- Bildet zu jedem Datenblock einen Message Digest zur Sicherung der Integrität

FTP - File Transfer Protocol

- FTP ist der *Internet-Standard für die Übertragung von Dateien*
- FTP wird benutzt, um eine **komplette Datei** von einem Rechner auf einen anderen zu **kopieren**
- FTP bietet neben dem reinen File-Transfer noch andere Möglichkeiten:
 - **Interaktiver Zugriff** durch den Nutzer (z.B. Wechsel von Verzeichnissen)
 - **Format-Spezifikation** (Binär- oder Textdateien, ASCII- oder EBCDIC-Code)
 - **Authentifizierung** urspr. nicht empfehlenswert : (login-Name und Passwort)
 - Es gibt Varianten, die eine verbesserte Authentifikation vornehmen

FTP Client/Server-Beziehung



FTP: Passive und Active Mode

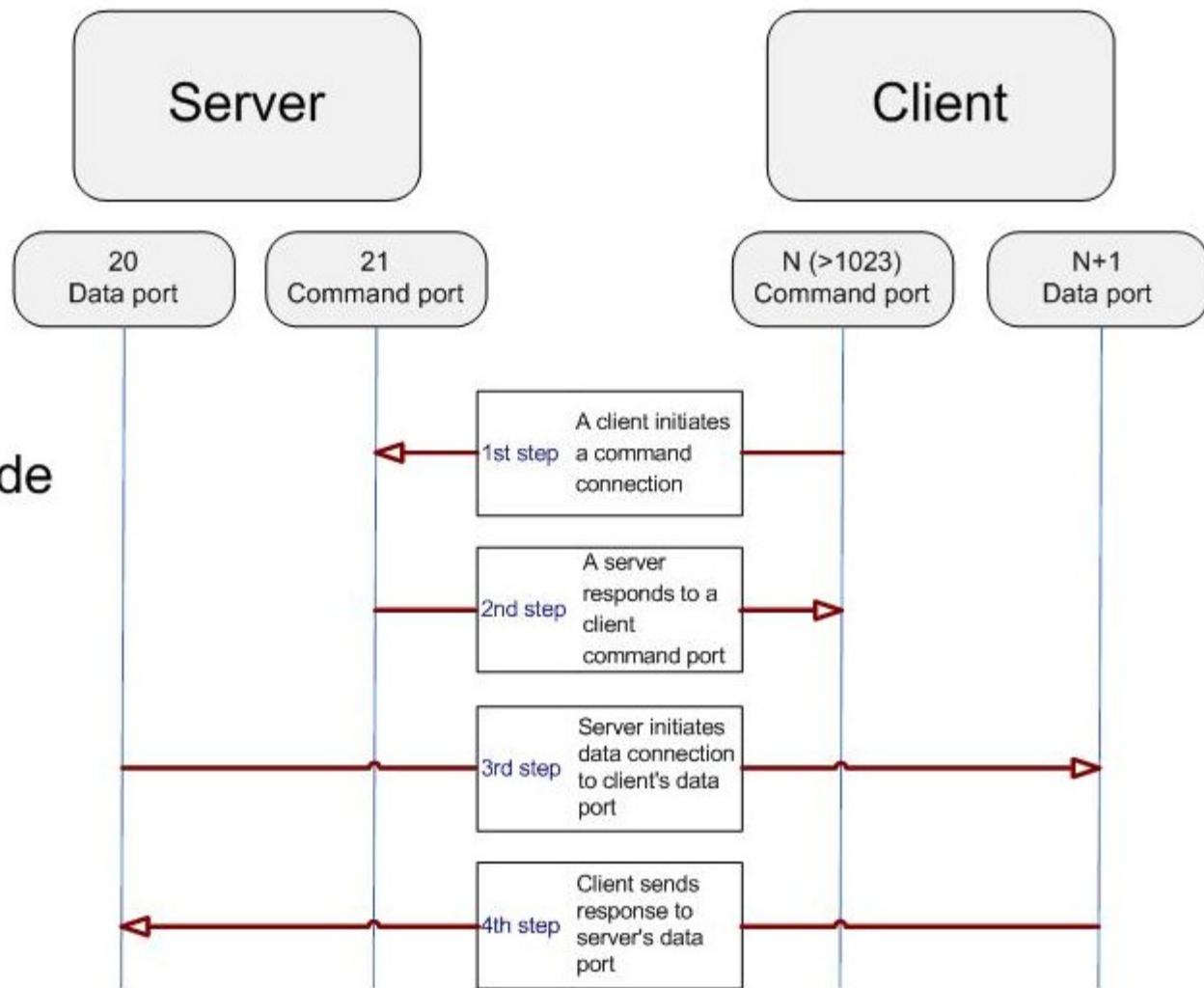
FTP ist deshalb besonders, weil es für die Datenübertragung eine separate TCP-Verbindung verwendet. Diese kann somit über verbesserte Parameter (WSCALE-Option verfügen).

Es gibt 2 verschiedene Arten, die Datenverbindung zu öffnen:

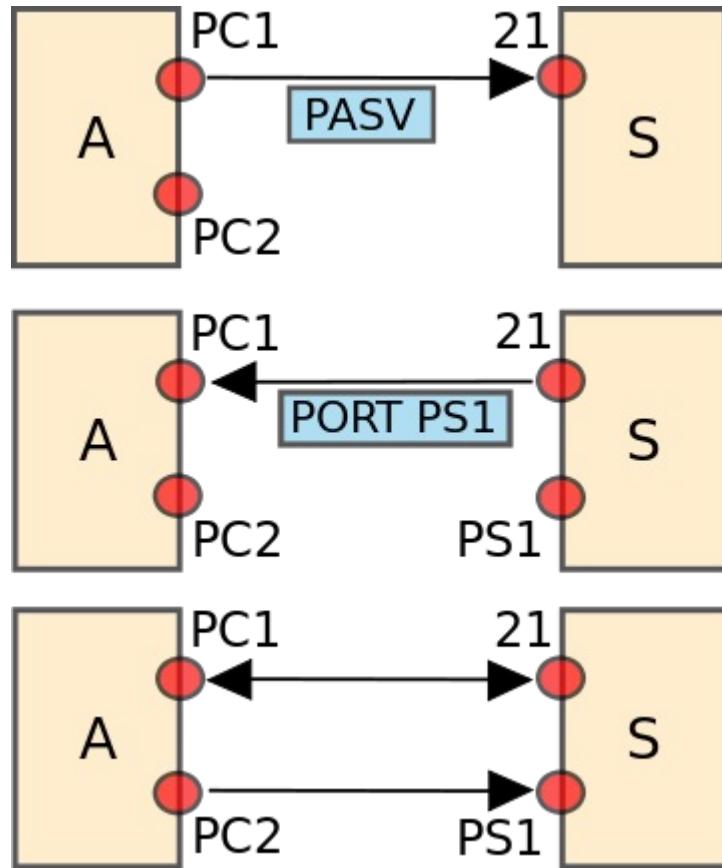
- 1. Active Mode:** Hier horcht der Client für die Datenverbindung auf einem zufälligen Port und teilt diesen dem Server über die Kontrollverbindung mit. Hierzu dient das PORT-Kommando. Der Server (Port 20) verbindet sich dann mit dem Client und wickelt den Transfer hierüber ab
- 2. Passive Mode:** Öffnet der Server einen Port und teilt diesen dem Client mit. Der Client verbindet sich nun durch mit dem Server. Das PASV-Kommando regelt dies. Vorteil: Dieses Verfahren funktioniert auch bei Adressumsetzungen (NAT) und Firewalls.

FTP: Active Mode

Active FTP Mode



FTP: Passive Mode

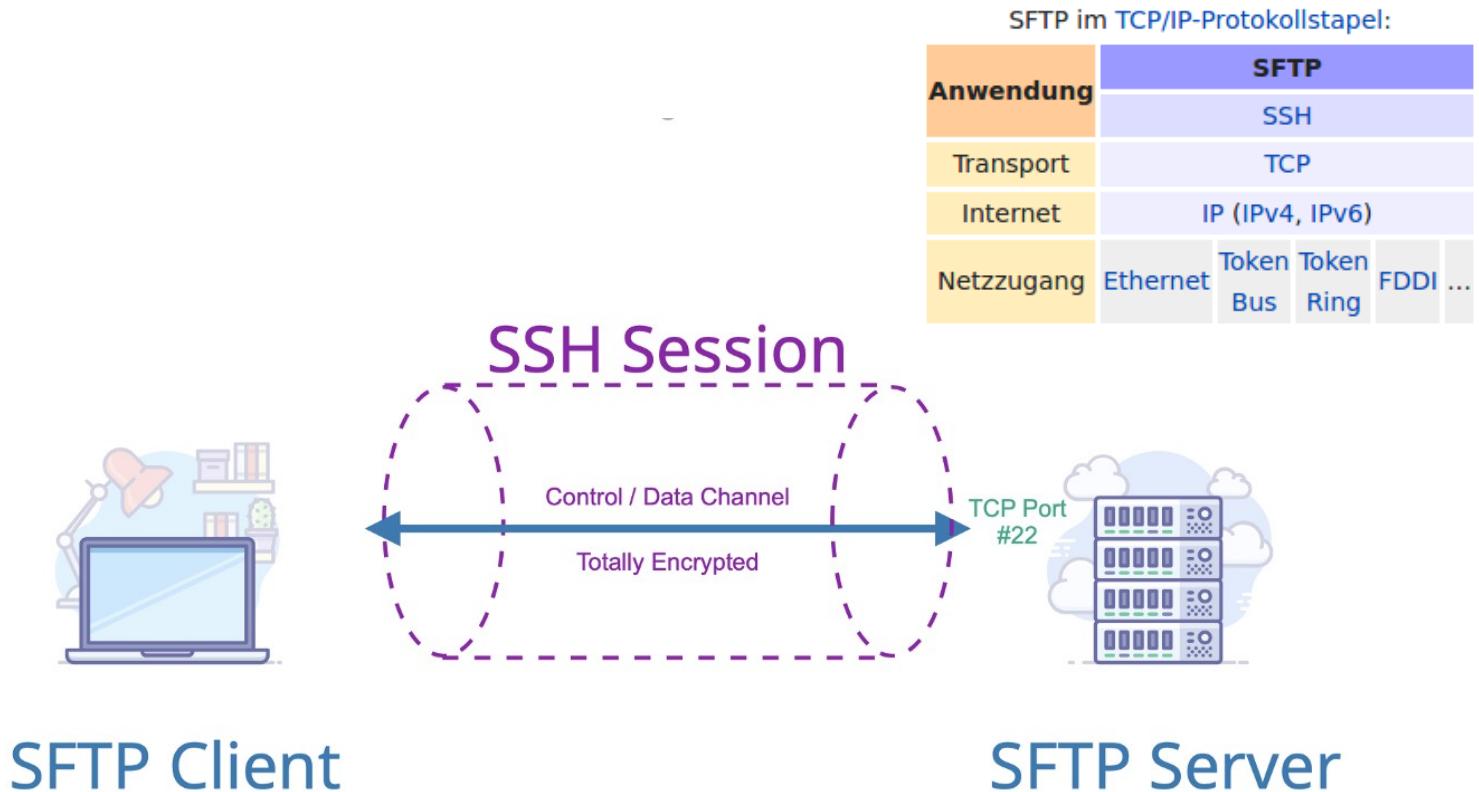


https://commons.wikimedia.org/wiki/File:Passive_FTP_Verbindung.svg

FTP - Befehle

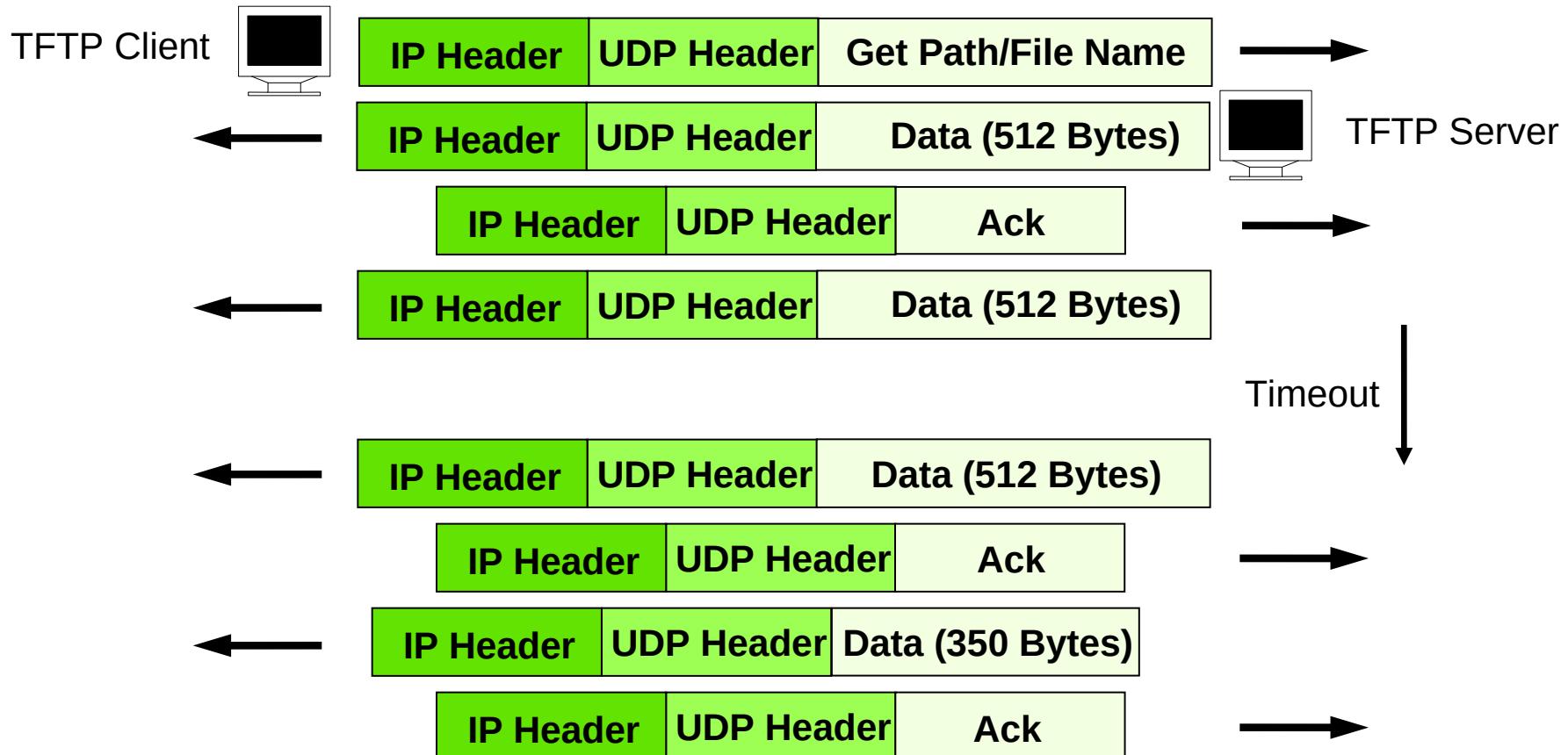
Kommando	Wirkung
open	Verbinden zum FTP-Server
disconnect	Beende die FTP-Sitzung
user	Sende Benutzerinformationen nach dem Verbinden
cd	change directory auf dem entfernten Rechner
lcd	change directory auf dem eigenen Rechner
pwd	Drucke das Arbeitsverzeichnis des entfernten Rechners
get/mget	Der Client empfängt ein (bzw. mehrere) Dokument
put/mput	Der Client sendet ein (bzw. mehrere) Dokument
binary	Setze den Übertragungsmodus auf binary
ascii	Setze den Übertragungsmodus auf ASCII
dir/ls	Liste den Inhalt des entfernten Verzeichnisses auf
help	Hilfe
delete	Lösche eine entfernte Datei
bye	Beende die FTP-Sitzung, Abbruch

Sicheres FTP: SFTP - FTP über SSH



TFTP - Trivial File Transfer Protocol

- TFTP ist ein sehr **einfaches** Protokoll für den File-Transfer
- die Kommunikation läuft über Port 69 und benutzt **UDP**, nicht TCP
- TFTP hat **keine Authentifizierung**
- TFTP benutzt immer 512-Byte-Blöcke

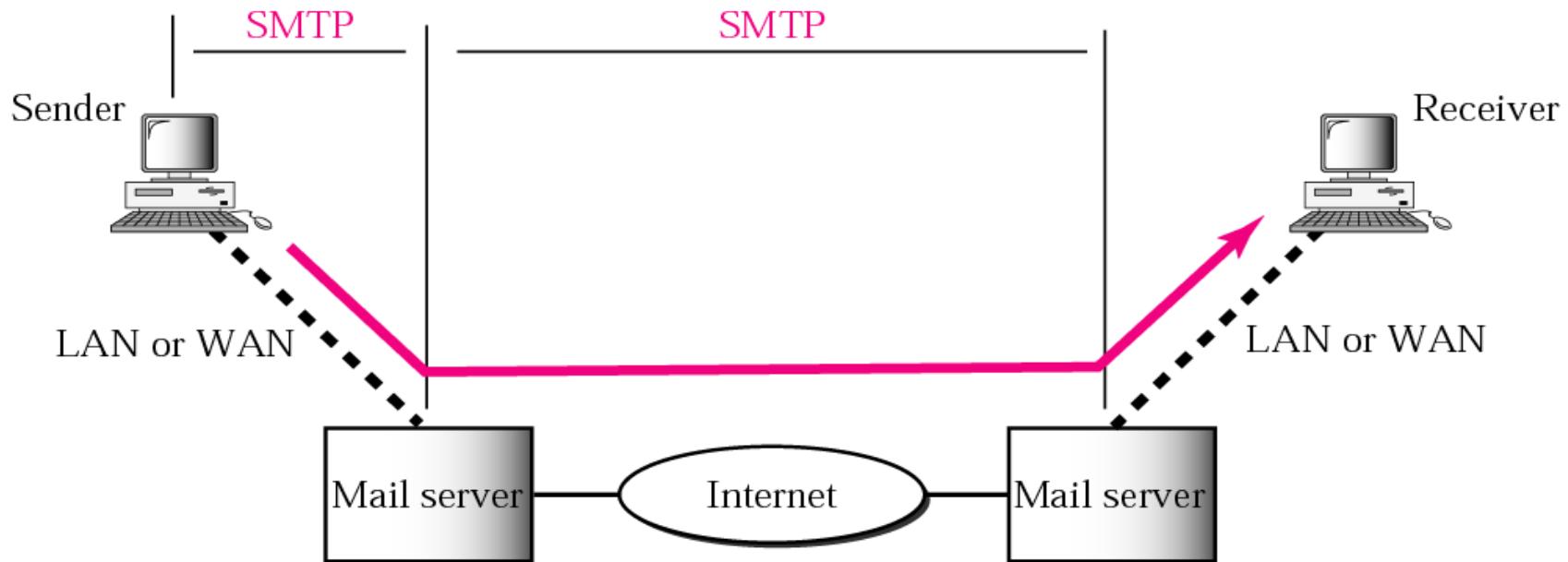


Elektronische Post: E-Mail

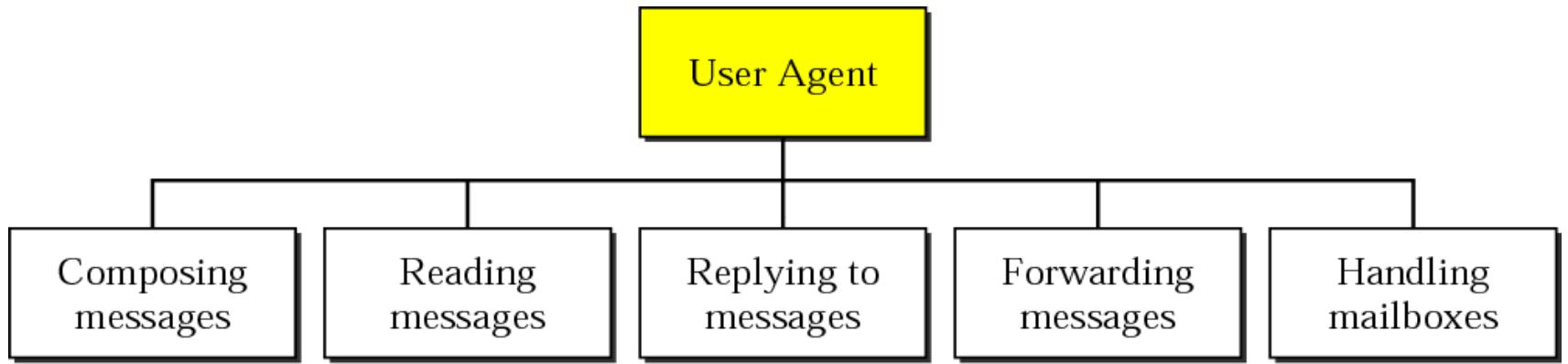
Ein E-Mail-System besteht im allgemeinen aus zwei Subsystemen:

- **User Agent** (UA, normales Email-Programm)
 - läuft meist auf dem Rechner des Benutzers und hilft bei der Bearbeitung von E-Mails
 - Erstellung neuer und Beantwortung alter E-Mail
 - Empfang und Anzeigen von E-Mail
 - Verwaltung von erhaltenen E-Mail
- **Message Transfer Agent** (MTA, Mailserver, mailrelay)
 - läuft meist im Hintergrund (rund um die Uhr)
 - Zustellung von E-Mails, die von User Agents losgeschickt wird
 - Zwischenspeicherung von Nachrichten für User oder andere Message Transfer Agents
- **Simple Mail Transfer Protokoll (SMTP)** zum Verschicken von E-Mails

Elektronische Post: E-Mail



Der User Agent



Das Senden von E-Mails

Zum Verschicken einer eMail muss der Benutzer folgende Angaben machen:

- *Nachricht* (meist normaler Text + Attachements, z.B. Word-Datei, GIF...)
- *Zieladresse* (i.a. in der Form mailbox@location, z.B. v.sander@fh-aachen.de)
- evtl. *zusätzliche Parameter* bzgl. Priorität oder Sicherheit

eMail-Formate

Zwei verbreitete Standards:

- **RFC 822** (ARPA Internet Text Messages)
- **MIME** (Multipurpose Internet Mail Extensions)

Bei **RFC 822** besteht die E-Mail aus

- einem einfachen “Umschlag” (erstellt durch den Message Transfer Agent anhand der Daten im Header),
- einer Reihe von Header-Feldern (je eine Zeile ASCII-Text),
- einer Leerzeile und
- der eigentlichen Nachricht (Message Body).

Header einer eMail

Header	Bedeutung
To:	eMail-Adresse des Hauptempfängers (evtl. mehrere oder auch Verteilerliste)
Cc:	Carbon Copy (Durchschrift), Email-Adressen von weniger wichtigen Empfängern
Bcc:	Blind Carbon Copy, Empfänger, die anderen Empfängern <i>nicht</i> angezeigt werden
From:	Person, die die Nachricht generiert hat
Sender:	Adresse des eigentlichen Senders der Nachricht (evtl. ungleich der “From-Person”)
Received:	Je ein Eintrag pro Message Transfer Agent auf dem Weg zum Ziel
Return-Path:	Pfad zurück zum Sender (meist nur eMail-Adresse des Senders)
Date:	Senden-Datum und -Uhrzeit
Reply-To:	eMail-Adresse, an die Antworten gerichtet werden sollen
Message-Id:	Eindeutige Nummer der Email (für spätere Referenzen)
In-Reply-To:	Message-Id der Nachricht, auf die geantwortet wurde
References:	Andere relevante Message-Ids
Subject:	Einzelige Angabe des Inhalts der Nachricht (wird beim Empfänger angezeigt)

Anmerkung: neben dieser Liste existieren noch weitere mögliche Header-Felder

Header einer E-Mail: MIME

RFC 2822 nur geeignet für Nachrichten aus reinem ASCII-Text ohne Sonderzeichen

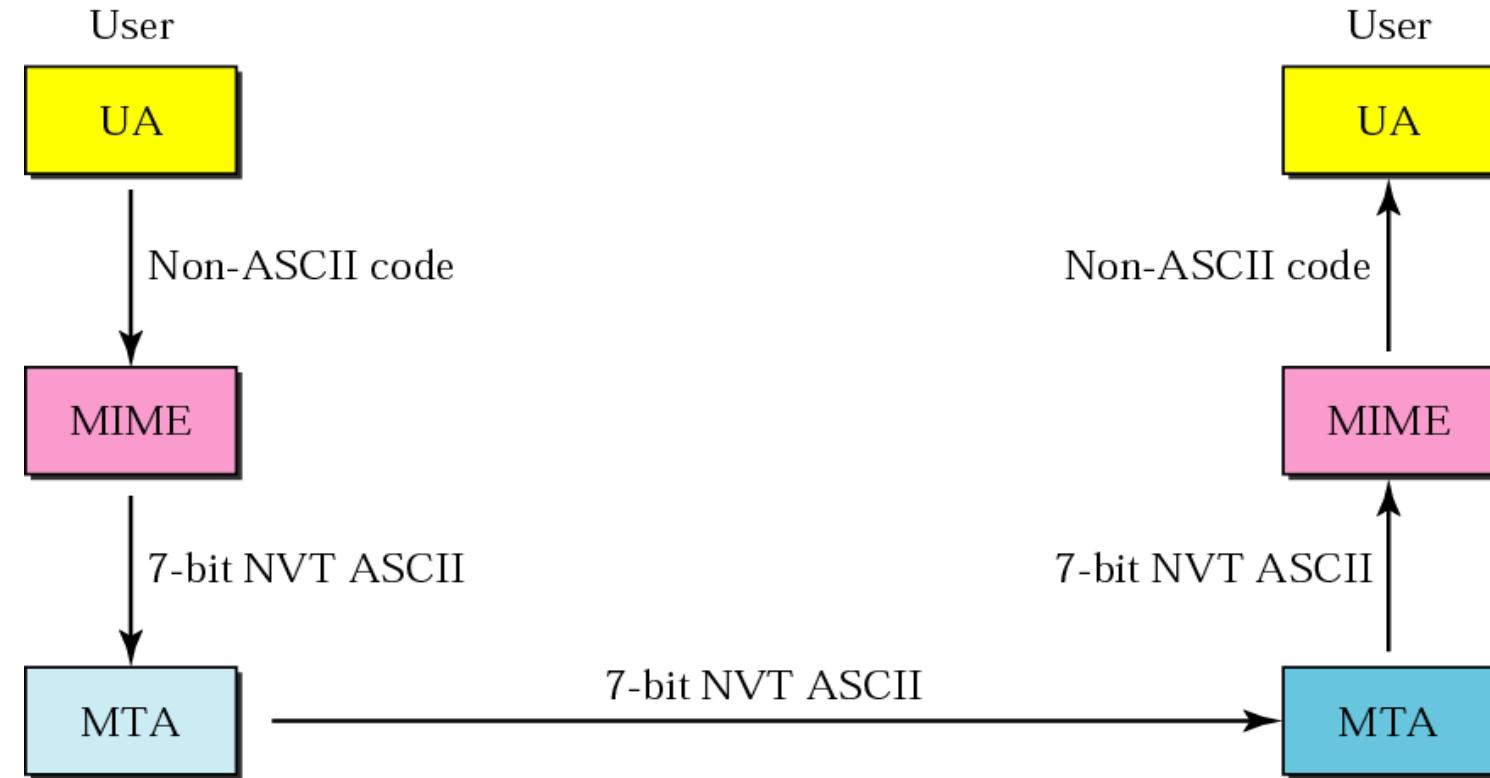
Heutzutage zusätzlich gefordert:

- eMail in Sprachen mit Sonderzeichen (z.B. französisch oder deutsch)
- eMail in Sprachen, die nicht das lateinische Alphabet benutzen (z.B. russisch)
- eMail in Sprachen, die überhaupt kein Alphabet benutzen (z.B. japanisch)
- eMail, die teilweise überhaupt keinen Text enthält (z.B. Audio oder Video)

MIME behält das RFC 2822 Format bei, definiert dabei aber eine Struktur im Message Body (durch zusätzliche Header) und Kodierungsregeln für Nicht-ASCII-Zeichen

Header	Bedeutung
MIME-Version:	Kennzeichnet die benutzt Version von MIME
Content-Description:	Für Menschen lesbarer String, der den Inhalt der Nachricht beschreibt
Content-Id:	Eindeutiger Bezeichner des Inhalts
Content-Transfer-Encoding:	Verpackung, die für den Inhalt der eMail gewählt wurde (manche Netze verstehen z.B. nur ASCII-Zeichen). Beispiele: base64, quoted-printable
Content-Type:	Typ/Subtyp gemäß RFC 1521, z.B. text/plain, image/jpeg, multipart/mixed

MIME-Ablauf

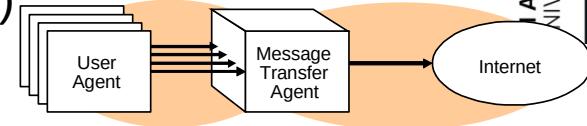


Der konkrete Ablauf wird nunmehr in den RFCs 2045 (MIME) und 2821 (SMTP) beschrieben

eMail Abruf über POP3 oder IMAP

Simple Mail Transfer Protocol (SMTP)

- Versenden von eMails über TCP-Verbindung (Port 25)
- SMTP ist ein einfaches ASCII-Protokoll
- Ohne Prüfsummen, ohne Verschlüsselung
- Ist der Server zum Empfangen bereit, signalisiert er dies dem Client.
Dieser sendet die Information, von wem die eMail kommt und wer der Empfänger ist. Ist der Empfänger dem Server bekannt, sendet der Client die Nachricht, der Server bestätigt den Empfang.



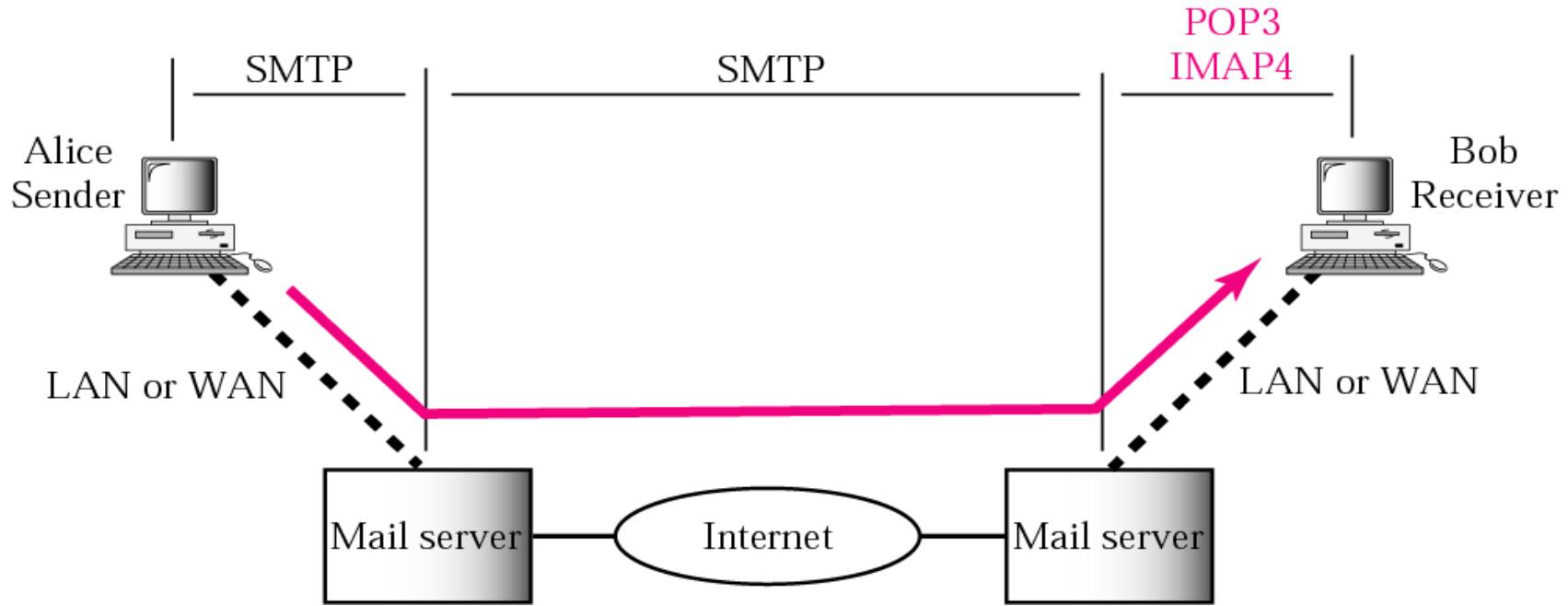
Post Office Protocol Version 3 (POP3)

- Abholen der eMails beim Server über eine TCP-Verbindung, Port 110
- Befehle zum An- und Abmelden, Nachrichten herunterladen, Nachrichten auf dem Server löschen oder liegen lassen, Nachrichten ohne vorherige Übertragung vom Server direkt löschen



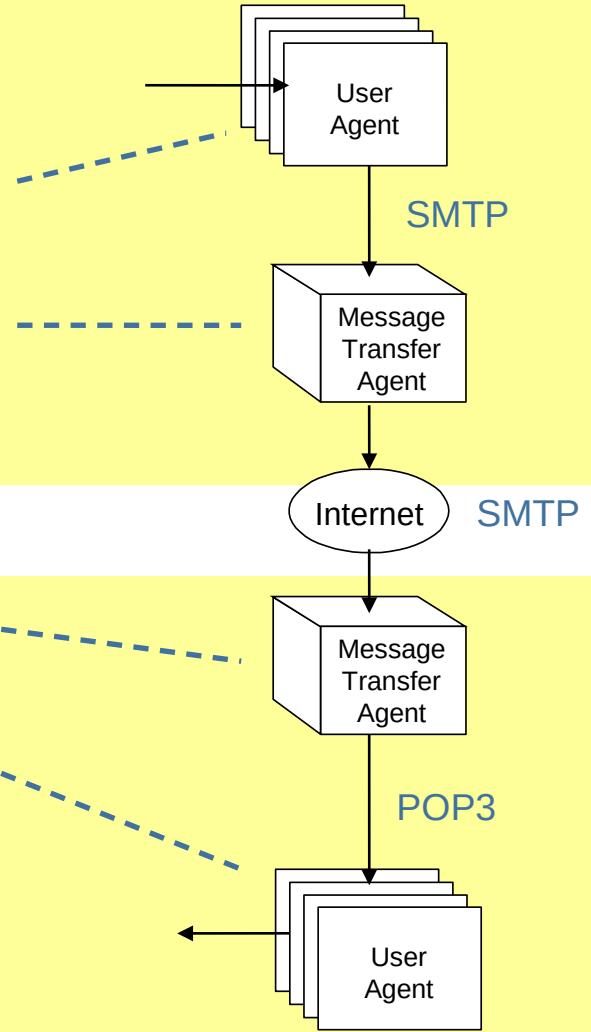
IMAP (Interactive Mail Access Protocol). Hier werden die eMails nicht abgerufen und lokal gespeichert, sondern bleiben auf dem Server liegen!

eMail Abruf über POP3 oder IMAP



Beispiel: eMail über SMTP und POP3

- *Benutzer 1*: schreibt eine eMail
- *Mailprogramm 1 (UA 1)*: Formatiert die eMail, erzeugt die Empfängerliste und schickt die eMail an seinen Mailserver (MTA 1)
- *Client 1 (MTA 1)*: Baut die Verbindung zum SMTP-Server (MTA 2) auf und schickt eine Kopie der eMail dorthin
- *Server (MTA 2)*: Erzeugt den Header der eMail und platziert die eMail in die passende Mailbox
- *Client 2 (UA 2)*: baut die Verbindung zum POP3-Server auf, authentifiziert sich mit Username und Passwort (unverschlüsselt!)
- *Server (MTA 2)*: schickt die eMail an den Client
- *Mailprogramm 2 (UA 2)*: formatiert die eMail
- *Benutzer 2*: liest die eMail



SMTP - Befehlsabfolge

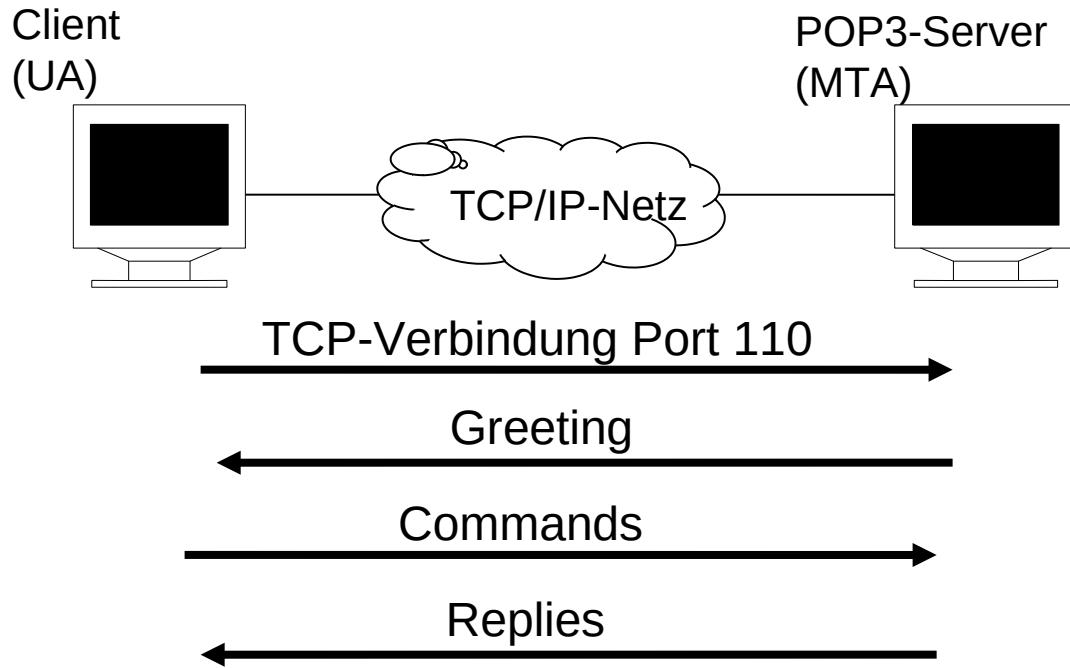
Kommunikation zwischen Partnern (von abc.com nach beta.edu) in Textform der Art:

S: 220 <beta.edu> Service Ready	/* Empfänger ist bereit */
C: HELO <abc.com>	/* Identifikation des Senders */
S: 250 <beta.edu> OK	/* Server meldet sich */
C: MAIL FROM:<Krogull@abc.com>	/* Sender der eMail */
S: 250 OK	/* Senden ist erlaubt */
C: RCPT TO:<Bolke@beta.edu>	/* Empfänger der eMail */
S: 250 OK	/* Empfänger bekannt */
C: DATA	/* Jetzt kommen die Daten */
S: 354 Start mail input; end with "<crlf>.<crlf>" on a line by itself	
C: From: Krogull @ <crlf>.<crlf>	/* ab hier normales Nachrichtenformat */
S: 250 OK	
C: QUIT	/* Beenden der Verbindung */
S: 221 <beta.edu> Server Closing	

S = Server, empfangender MTA / C = Client, sendender MTA

POP3-Prozess

Abholen der eMails vom Server mittels POP3:



- Authorisierungsstatus:
USER name
PASS string
- Transaktionsstatus
STAT
LIST [msg]
RETR msg
DELE msg
NOOP
RSET
QUIT

- Damit werden die vollständigen eMails vom Server auf den Client transferiert
- Wahlweise können hierbei die Nachrichten auch gelöscht werden

Idee: Zugriff auf Server-Verzeichnisse wie auf lokale Verzeichnisse

- Daten verbleiben auf dem Server. Alle Aktionen führt der Client durch
- Das Protokoll ist komplexer als bei POP3 und erlaubt die Manipulation von Mailverzeichnissen auf dem Server (Erstellen, Umbenennen, Löschen; Setzen/Löschen von Flags; Suchen von Mails)
- Reduziert die zu übertragenen Daten, da zunächst nur die Nachrichten-Header übertragen werden
- Offline-Betrieb und Resynchronisation mit dem Server möglich
- Spezifiziert in **RFC 3501**

Mit IMAP verbleiben die eMail auf dem Server

Typische Zugangsdaten eines ISP

Wie kann ich meine E-Mails über eine gesicherte Verbindung (SSL oder TLS) versenden und empfangen?

[Druckansicht](#)
[FAQ #118](#)

Möchten Sie E-Mails mit Ihrem eigenen E-Mail Programm verschlüsselt versenden und empfangen, aktivieren Sie bitte die entsprechende Option in den Konto-Einstellungen. Diese Einstellungsmöglichkeit finden Sie in der Regel in Ihrem E-Mail Programm unter dem Menüpunkt **Extras > E-Mail-Einstellungen** bzw. **Konten** oder **Kontoeinstellungen**. Die Option selbst wird unterschiedlich benannt. Bei Outlook / Outlook Express z. B. heißt der Menüpunkt **Server erfordert eine verschlüsselte Verbindung (SSL)**, in Mozilla Thunderbird ist die **Verbindungssicherheit: SSL/TLS** einzustellen. Bitte lesen Sie dies, falls notwendig, in der Dokumentation zu Ihrer Software nach.

Die notwendigen E-Mail-Server lauten wie folgt:

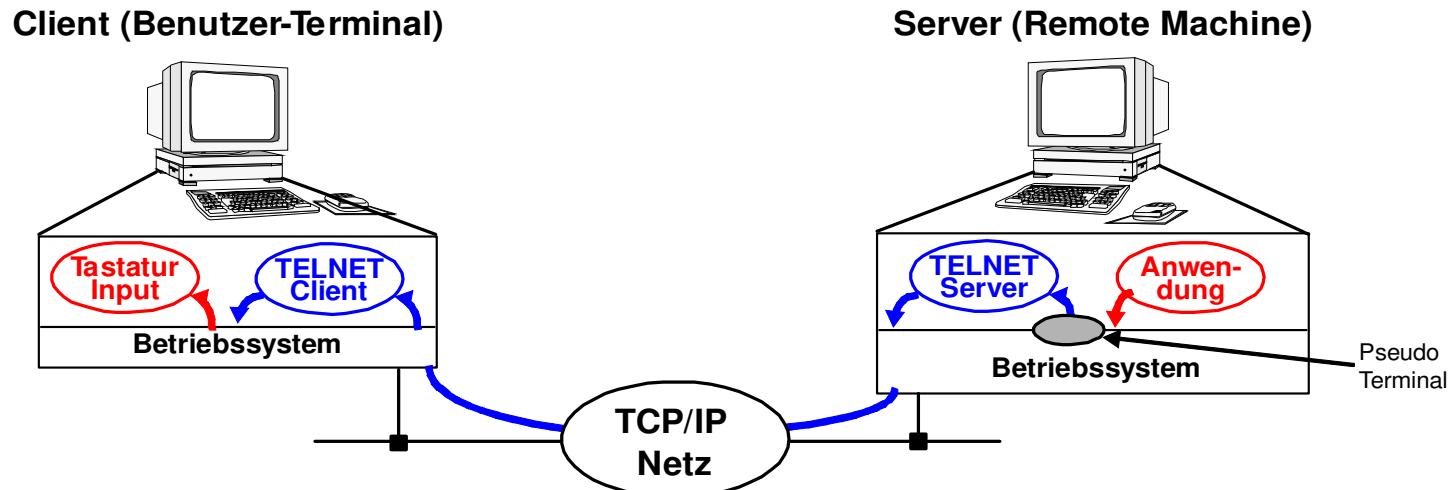
Posteingang (POP3):	pop3.strato.de
Posteingang (IMAP):	imap.strato.de
Postausgang (SMTP):	smtp.strato.de

Sofern Ihr E-Mail Programm Portangaben benötigt, lauten diese:

Protokoll	Port
POP3 (SSL/TLS)	995
IMAP (SSL/TLS)	993
SMTP (SSL/TLS)	465

Telnet - Entferntes Arbeiten

- TCP ermöglicht den transparenten, interaktiven Gebrauch von „entfernten“ Maschinen
- verbreitetes Protokoll: TELNET, welches auf einer Client/Server-Kommunikation basiert
- Ein „Pseudo-Terminal“ des Servers interpretiert Zeichen, als kämen sie von der eigenen Tastatur



- bei Antwort des Servers umgekehrter Weg (Pseudo-Teminal fängt Antwort ab, leitet sie über TCP an den Client weiter, der die Ausgabe am Bildschirm macht)
- **Benutzername und Passwort werden unverschlüsselt übertragen**

rlogin und rsh als Alternative zu Telnet (für Unix)

- *rlogin* ist eine sehr flexible Alternative zu Telnet. Sogenannte Trusted Hosts können sich login-Name und Zugriffsrechte auf Dateien teilen.
- Vorteile gegenüber TELNET:
 - Bei rlogin auf einem Trusted Host entfällt die Abfrage des Passworts.
 - Da ausschließlich unter Unix verwendet, vereinfacht sich die Kommunikation zwischen Client und Server: beide Seiten kennen so etwas wie Standard Input und Output, Standard Error ...
 - Umgebungsvariablen des Benutzers (z.B. Terminaltyp) werden automatisch übertragen, so dass entfernte Sitzungen große Ähnlichkeit mit lokalen Sitzungen haben.
- *rsh* ist eine Variante von rlogin:
 - Ziel: Auf einfache Art und Weise einzelne Kommandos auf der Remote Machine ausführen (rsh machine command).
 - Automatische Authentifizierung erlaubt die Benutzung nicht nur interaktiv, sondern auch aus Programmen heraus (ohne Passwortabfrage).

Das ssh-Protokoll

- **ssh** adressiert die Sicherheitsprobleme von telnet und rlogin. Es ist ein Protokoll zur Erstellung einer sicheren Verbindung zwischen zwei Systemen. Alle während der Verbindung gesendeten und empfangenen Daten werden mit einer 128 Bit-Verschlüsselung verschlüsselt.
- ssh unterstützt verschiedene Authentisierungsarten:
 - Bei der so genannten hostbased-Authentifizierung akzeptiert ein Rechner ohne eigene account-spezifische Tests die Vorgaben eines fremden Rechners. Es wird höchstens die Identität des fremden Rechners überprüft.
 - Die Authentifikation mit einem Passwort ist derzeit die "übliche" Methode, um sich an einem Rechner anzumelden. Die Sicherheit dieses Mechanismus beruht auf der Geheimhaltung des Passwortes, dessen Übertragung allerdings verschlüsselt wird
 - Um auch das Übertragen eines verschlüsselten Passwortes zu vermeiden, werden die so genannten public-key-Verfahren eingesetzt

SSH: Port-Forwarding

- Port-Forwarding:
 - verschlüsselte Verbindung zwischen zwei beliebigen Ports
 - kann auch ohne Shell genutzt werden
 - lokaler Port führt direkt auf den Zielport, als wäre dieser lokal
- universell einsetzbar, u. a.:
 - FTP sicherer machen (Tunneln des Kommando-Ports)
 - POP3/SMTP (Mailversand) sicherer machen
 - X Window Datenverkehr absichern
- Aufruf:
 - **ssh -L port:zielHost:zielPort <Rechner>** leitet *localhost:port* via *Rechner* zu *zielHost:zielPort* weiter
 - **ssh -R port:zielHost:zielPort <Rechner>** leitet *Rechner:port* via *localhost* zu *zielHost:zielPort* weiter

SNMP (Simple Network Management Protocol)

Die Objekte des Internet-Managements sind Rechner und vor allem Router
Ähnlich wie bei SMTP wird das Internet-Management durch zwei unabhängige standardisierte Teilbereiche beschrieben:

- Das Protokoll SNMP, das festlegt, wie Management-Information kommuniziert wird (Formate und Bedeutung von SNMP-Nachrichten) und
- die Spezifikation der Daten (MIB \circlearrowright *Management Information Base*).
Die MIB spezifiziert die Informationseinheiten (*items*), die vorgehalten werden müssen, und welche Operationen darauf erlaubt sind.

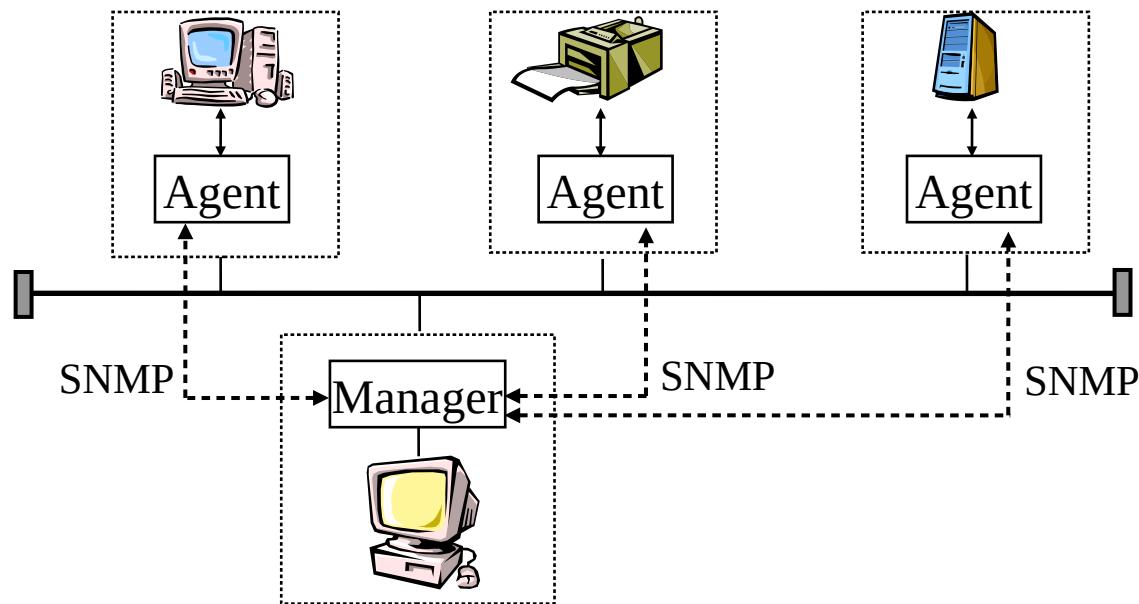
Wie bei den anderen Anwendungsdiensten, funktioniert auch das Management nach dem *Client/Server*-Prinzip.

In jedem Objekt (vor allem Router) muss ein *Server* installiert sein, der die in der MIB spezifizierten Informationen sammelt, diese gegebenenfalls einem *Client* zur Verfügung stellt (per SNMP) und von einem *Client* Kommandos entgegennimmt.

Für das Ausführen von Management-Funktionen ist eine Authentifizierung erforderlich ist.

SNMP (Simple Network Management Protocol)

Verwaltete Ressourcen integrieren **SNMP-Agenten** (Software-Prozess)
Die Agenten verwalten die Managementinformationen der Komponente
z.B. Anzahl eingegangener/verlorener Pakete
Der **Manager** (Software-Prozess) dient der Kommunikation mit den Agenten
Protokoll: SNMP (verwendet UDP)

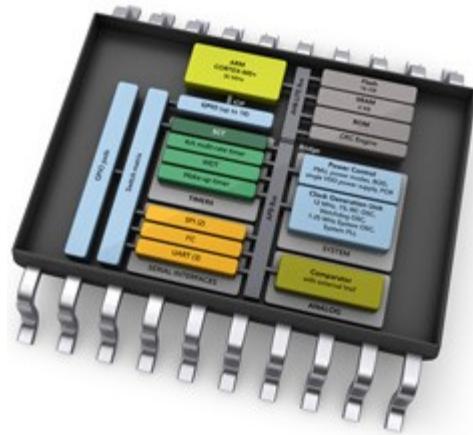


FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

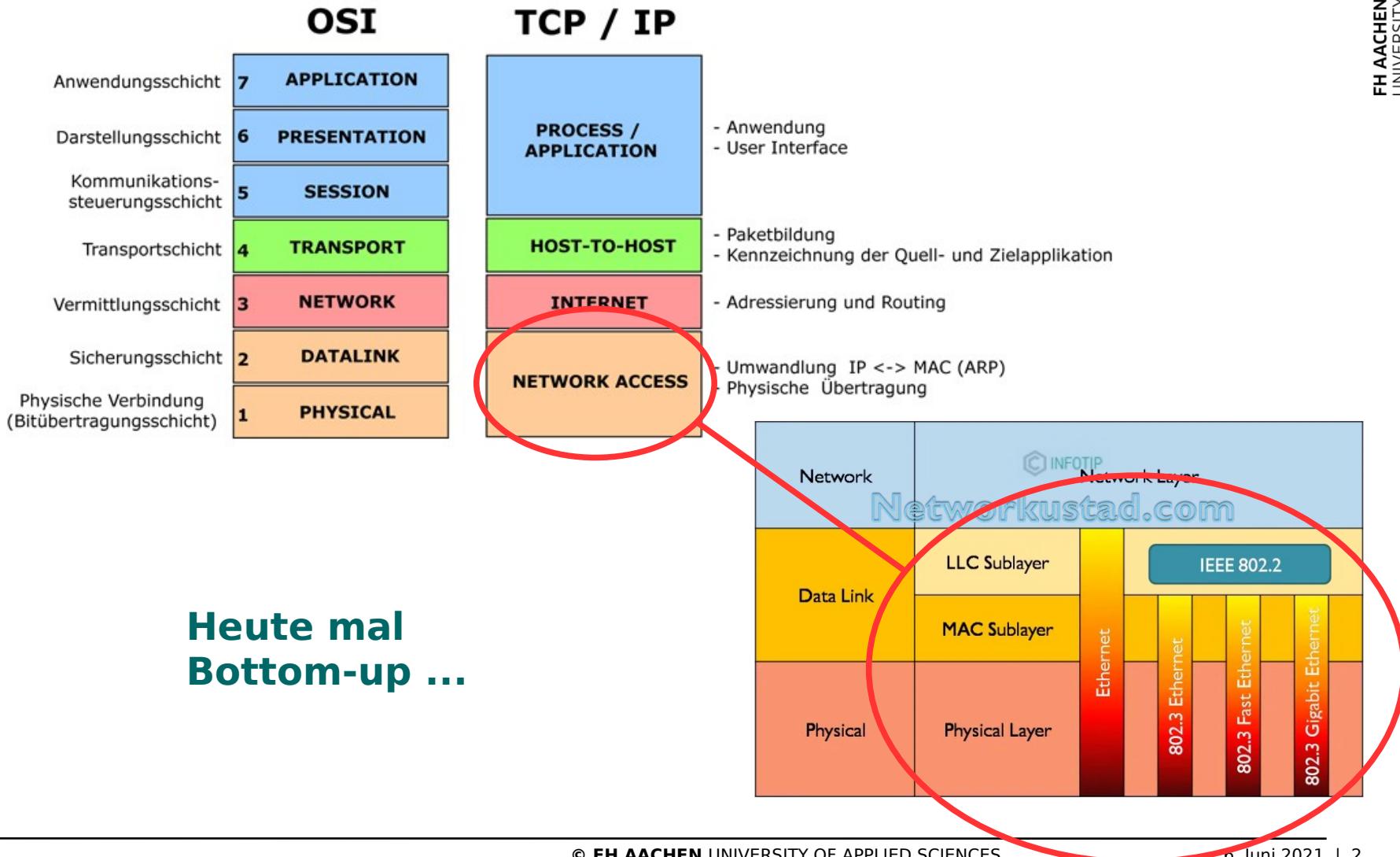
Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Schichten 1 und 2 im ISO/OSI Referenzmodell und im TCP/IP Referenzmodell



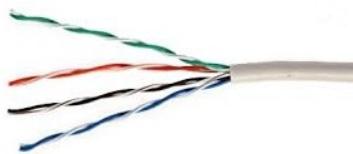
Beispiele für Layer 1/2 Standards: Ethernet Standards

Ethernet-Standard	Bezeichnung	Datenrate	Kabeltechnik	Erscheinungsjahr
802.3	10Base5	10 Mbit/s	Koaxialkabel	1983
802.3a	10Base2	10 Mbit/s	Koaxialkabel	1988
802.3i	10Base-T	10 Mbit/s	Twisted-Pair-Kabel	1990
802.3j	10Base-FL	10 Mbit/s	Glasfaserkabel	1992
802.3u	100Base-TX, 100Base-FX, 100Base-SX	100 Mbit/s	Twisted-Pair-Kabel, Glasfaserkabel	1995
802.3z	1000Base-SX, 1000Base-LX	1 Gbit/s	Glasfaserkabel	1998
802.3ab	1000Base-T	1 Gbit/s	Twisted-Pair-Kabel	1999
802.3ae	10GBase-SR, 10GBase-SW, 10GBase-LR, 10GBase-LW, 10GBase-ER, 10GBase-EW, 10GBase-LX4	10 Gbit/s	Glasfaserkabel	2002
802.an	10GBase-T	10 Gbit/s	Twisted-Pair-Kabel	2006

Übertragungsmedien: Schicht 1

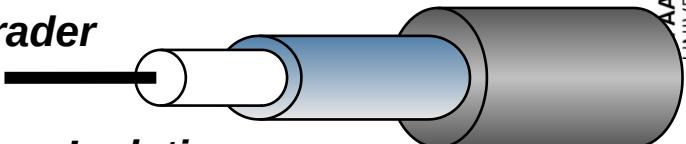
Kupferdoppelader (Twisted Pair)

Kupferader

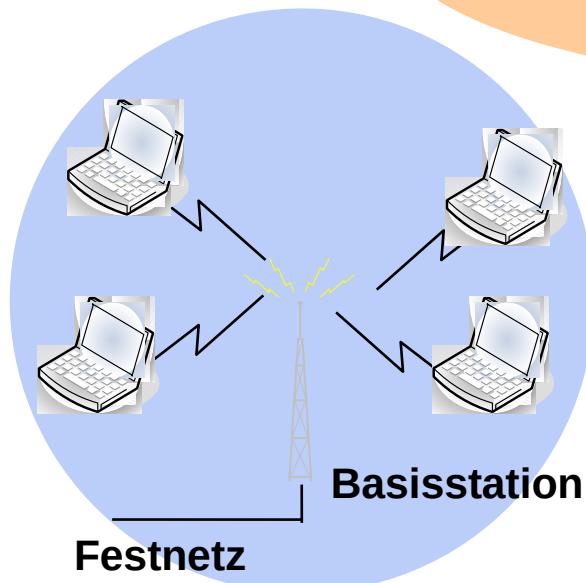


Koaxialkabel Abschirmung

Kupferader

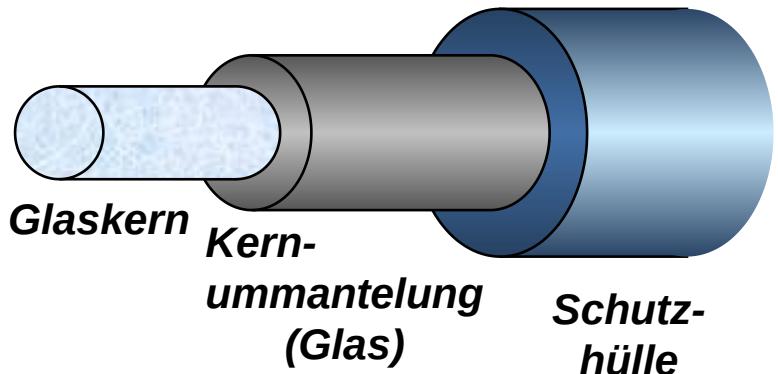


Funk



Unterschiedliche Medien
(verschieden in Übertragungs-
technik, Kapazität und
Bitfehlerrate)

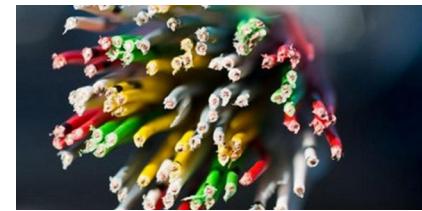
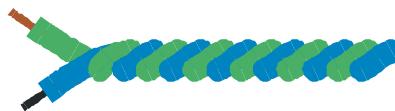
Glasfaser



Übertragung von Informationen

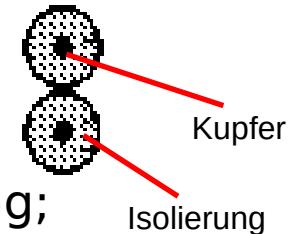
- Die Übertragung von (z.B. binären) Informationen geschieht über physikalische Größen wie z.B. Spannung oder elektromagnetische Wellen (Funk, Licht)
- Diese physikalischen Größen besitzen unterschiedliche Eigenschaften:
 - Amplitude (höhe der Spannung, Stärke des Lichtes..)
 - Frequenz
 - Phase
- Die Veränderung dieser Eigenschaften im Rahmen der Datenübertragung nennt man **Modulation**
 - Amplitudenmodulation (AM)
 - Frequenzmodulation (FM)
 - Phasenmodulation (PM)
- Verschiedene Modulationsarten können auch kombiniert werden!

Kupferdoppelader



Eigenschaften:

- Daten werden als **Spannungsniveaus** übertragen
 - > **Anfällig für Störungen**: Geräte oder andere Kabel in der Umgebung, die elektromagnetische Felder aussstrahlen, verfälschen die Darstellung der Bits auf dem Kupferkabel
- Besteht aus zwei gegeneinander isolierten, verdrillten Kabeln
 - > Verdrillen reduziert elektromagnetische Interferenzen
 - > Trotzdem: **Bit Error Rate (BER) $\leq 10^{-5}$**
 - > Einfach (bzgl. Kosten und Wartung)
 - > Oft existiert Twisted Pair bereits zur Telefon-Verkabelung;
 - > dies senkt die Vernetzungskosten
 - > Kabel dürfen nur eine Länge bis ca. hundert Meter haben (das Kupfer wirkt als Widerstand und schwächt das sich ausbreitende Signal ab!)
→ geringere Internet-Bandbreite!



Twisted Pair bei der Vernetzung

Unterscheidung nach Kategorie

Kategorie 3

Gemeinsame Umhüllung für vier Kupferdoppeladern

Kategorie 5

Wie Kategorie 3, aber mehr Windungen/cm (weitere Reduktion der elektromagnetischen Interferenzen)

Umhüllung besteht aus Teflon (bessere Isolierung, Qualität der Signale bleibt auf längere Strecken akzeptabel)

Kategorie 6,7

Die Paare sind zusätzlich einzeln mit Silberfolie umwickelt

Unterscheidung nach Abschirmung

UTP Kabel (Unshielded Twisted Pair)

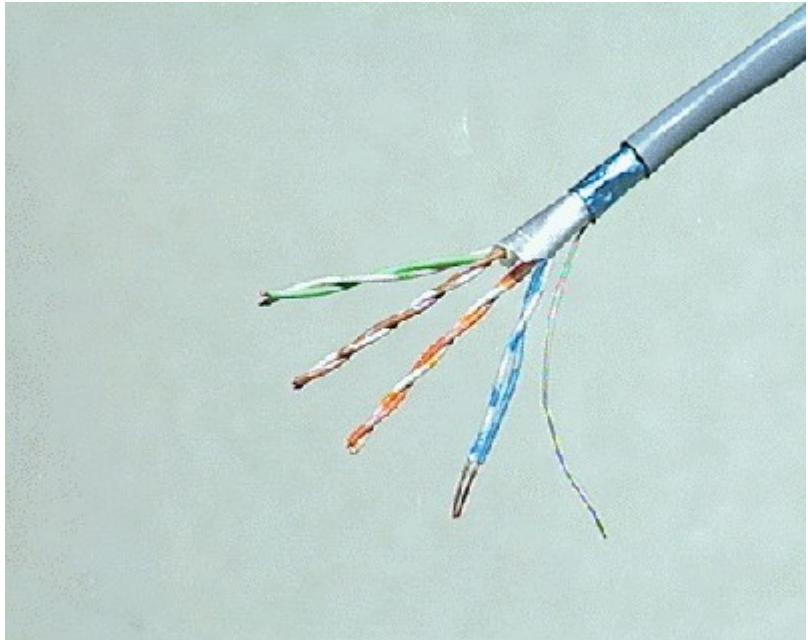
Keine Abschirmung des Kabels

STP Kabel (Shielded Twisted Pair)

Abschirmung des Kabels, dadurch günstigere Eigenschaften

Trotzdem in der Praxis oft UTP

Kupferdoppelader (Twisted Pair)

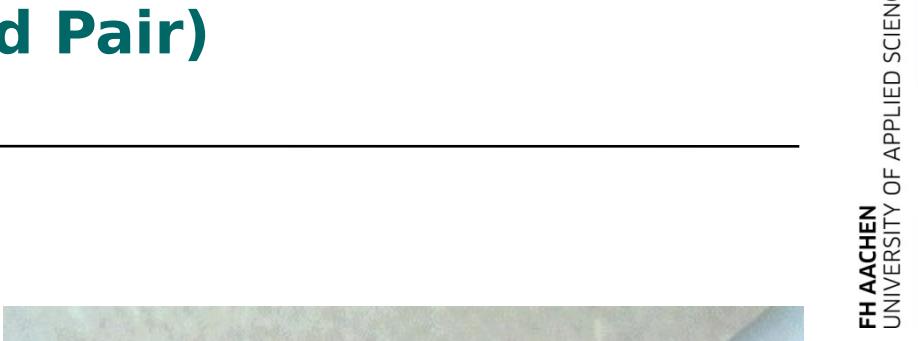


S/UTP-Kabel (cat 5)



S/STP-Kabel (cat 7)

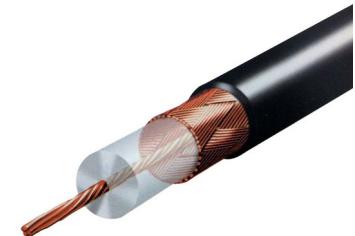
RJ45 Stecker



Koaxialkabel

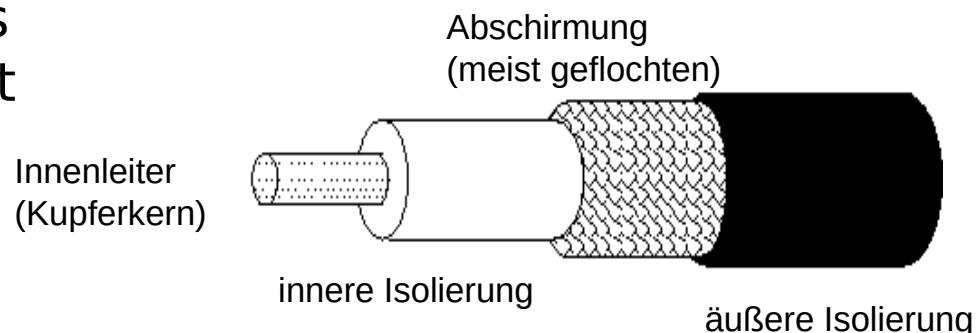
Charakteristika:

- Übertragung durch **Spannungsniveaus** wie bei Twisted Pair
- Besser abgeschirmt und damit weniger störanfällig als Twisted Pair:
- Bit Error Rate $\sim 10^{-9}$
 - > Größere Entferungen überbrückbar
 - > Höhere Datenraten möglich



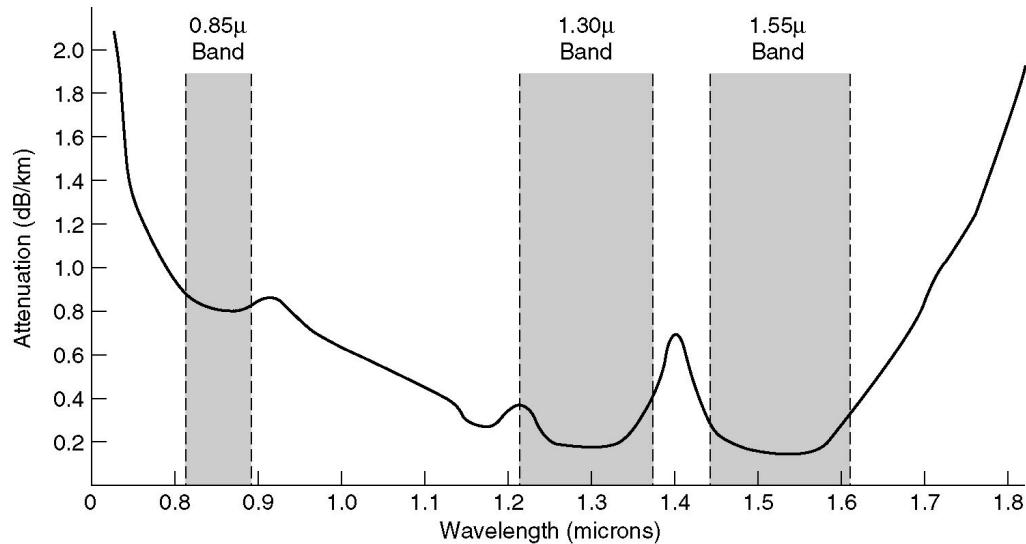
Aufbau

- Isolierter Kupferdraht im Zentrum (Innenleiter)
- Abschirmung besteht aus netzförmigen Kupferdraht
- Innere Isolierung trennt Innenleiter von der Abschirmung

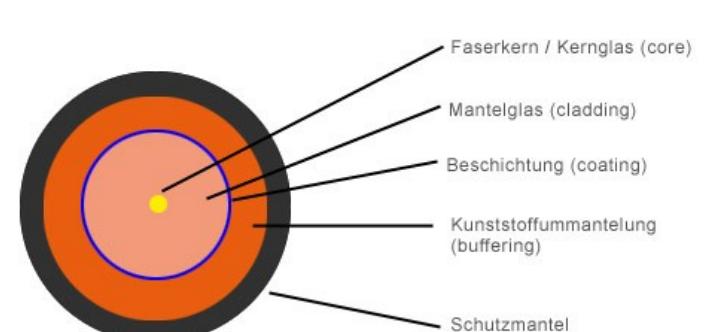


Charakteristika:

- Übertragung durch Lichtpulse im Bereich um 0.85, 1.3 und 1.55 μm
 - > Beschränkung auf diese Bänder aufgrund von Absorption des Lichts im Glaskern (ähnlich zu Dämpfung auf Kupferkabel)
- Unempfindlich gegenüber elektromagnetischen Störungen
 - > Bit Error Rate: $\sim 10^{-12}$
 - > Sehr große Entferungen mit sehr hohen Datenraten möglich



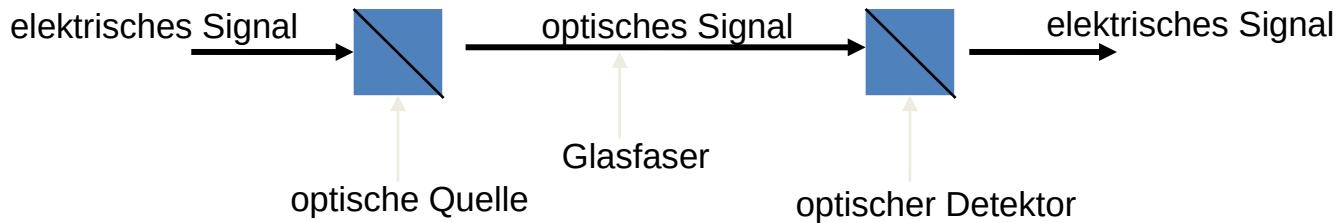
Aufbau eines Glasfaserkabels



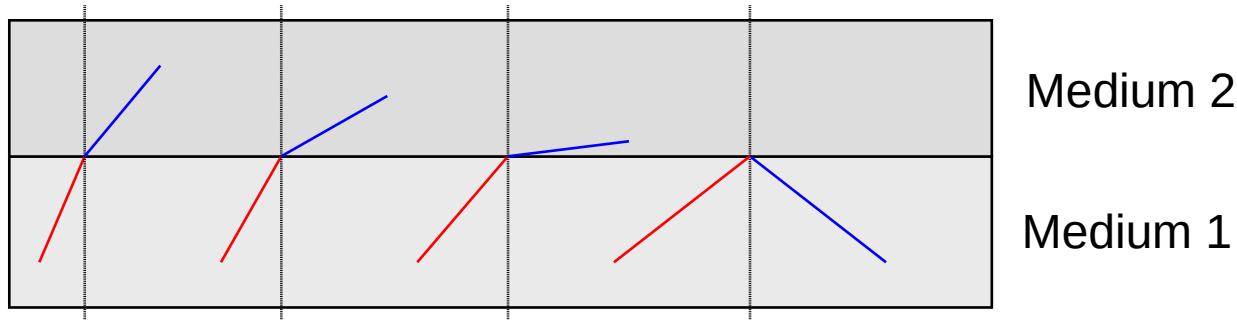
Optische Übertragung

Aufbau eines optischen Übertragungssystems

optische Quelle (konvertiert elektrische in optische Signale; normalerweise in der Form „1 – Lichtpuls“ / „0 – kein Lichtpuls“ – **Amplitudenmodulation**)
Übertragungsmedium
Detektor (konvertiert optische in elektrische Signale)



Physikalisches Grundprinzip: Totalreflexion des Lichts an einem anderen Medium



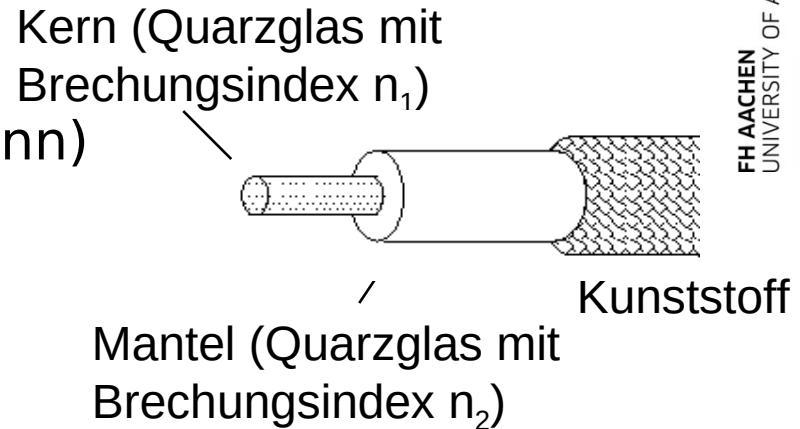
Brechungsindex:
Gibt Brechungswirkung relativ zur Luft an

Glasfaser: Aufbau

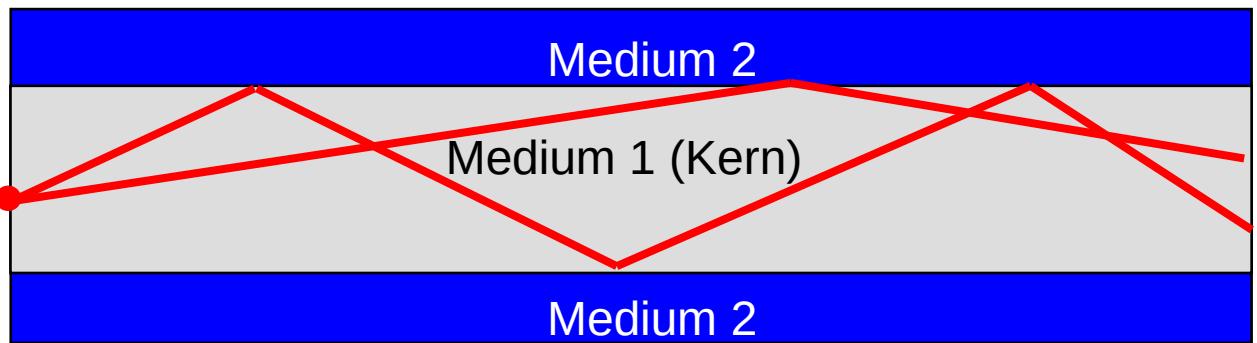
Aufbau:

- Kern: optisches Glas (extrem dünn)
- (innere) Glasummantelung
- (äußere) Kunststoffhülle

Die Übertragung findet im
Kern des Kabels statt!



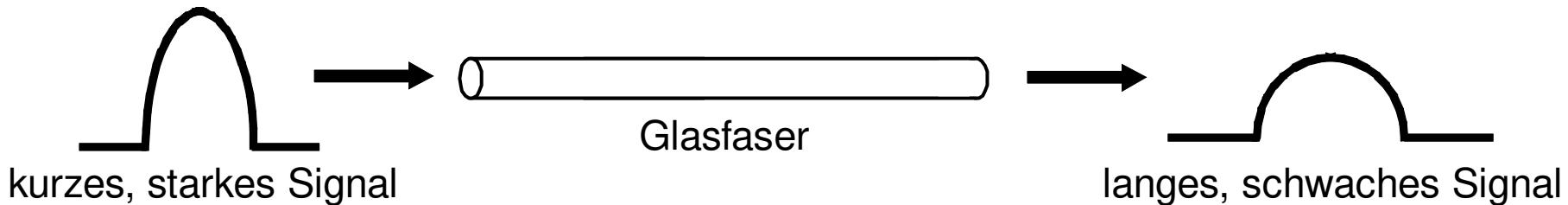
optische Quelle
(LEDs, Laser)



Glasfaser: Dispersion

Größtes Problem bei Glasfaser: **Dispersion**

- Begrenzt Übertragungsstrecke
- Lichtpuls besteht aus mehreren Wellen (Strahlen)
 - > Einfallswinkel dieser Strahlen unterschiedlich
- Lichtstrahlen kommen im Medium unterschiedlich schnell vorwärts:
 - > Wege (**Moden**) der Strahlen unterschiedlich lang (abhg. von Einfallswinkel)
 - > Strahlen eines Impulses kommen zeitversetzt am Ende des Kabels an
 - > Intensität der Impulse nimmt ab, benachbarte Impulse verschwimmen
- (Weitere Faktoren können ebenso Dispersion verursachen)



Glasfasertypen

Kennzeichnend bei der Unterscheidung ist das Profil:

X-Achse: Größe des Brechungsindex

Y-Achse: Dicke des Kerns und der Mantelschicht

Monomode-Faser

Kerndurchmesser: 8 -10 µm

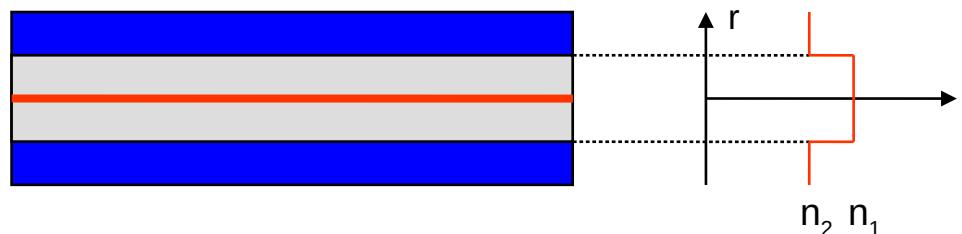
Alle Strahlen können nur noch einen Weg nehmen

Keine Dispersion (homogene Signalverzögerung)

50 GBit/s über 100 km

Teuer wegen geringem Kerndurchmesser

Achtung: Monomode bedeutet nicht, dass nur eine Welle gleichzeitig unterwegs ist. Es bedeutet, dass alle Wellen „den gleichen Weg“ nehmen. Damit wird Dispersion verhindert.



Glasfasertypen

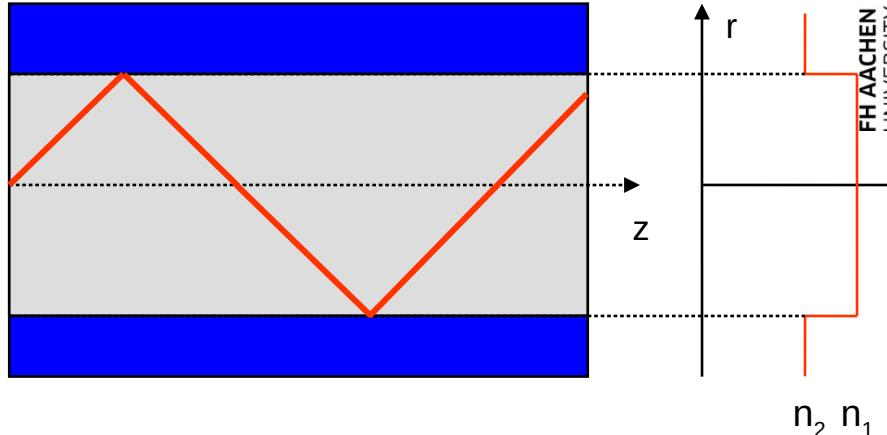
Multimode-Faser mit Stufenindex

Kerndurchmesser: 50 µm

Unterschiedliche Wege für Lichtwellen,
je nach Einfallswinkel

Starke Dispersion

Bis zu 1 km



Multimode-Faser mit Gradientenindex

Kerndurchmesser: 50 µm

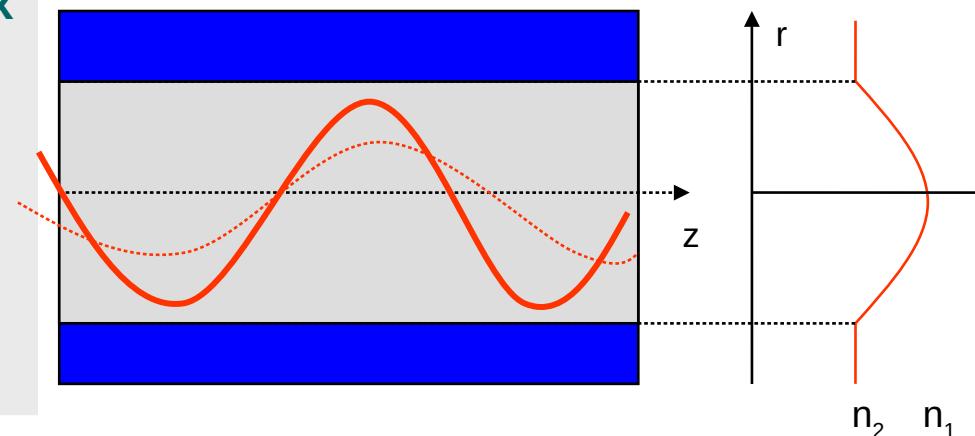
Brechungsindex ändert sich fließend

Leicht unterschiedliche Wege für

Lichtwellen

Geringere Dispersion

Bis zu 30 km



Glasfaser: Strahlungsquellen und -empfänger

Strahlungsquellen

Leuchtdioden (LED, Light Emitting Diode)

nahezu monochromatisch

billig und zuverlässig (z.B. gegenüber Temperaturschwankungen)

gewisses Wellenlängenspektrum, d.h. höhere Dispersion und somit geringe Reichweite

keine sehr hohe Kapazität

Laserdioden

teuer und empfindlich

hohe Kapazität

geringes Wellenlängenspektrum und damit hohe Reichweite



Strahlungsempfänger

Photodioden (*mit nachgeschaltetem Verstärker*)

unterscheiden sich insbesondere bei Signal-to-Noise Ratio

Durch verbesserte Materialeigenschaften der Fasern, präzisere Lichtquellen und damit Verkleinerung der Abstände zwischen den nutzbaren Wellenlängen wird die Anzahl der verfügbaren Kanäle laufen erhöht.

Weitere Eigenschaften einer Verbindungsstrecke

- Die **Datenübertragung** durch ein Kabel kann unterschiedliche Übertragungsrichtungen unterstützen:
- Simplex: Datenübertragung nur in eine Richtung
- Halb-Duplex: Datenübertragung in beide Richtungen, aber zu einem Zeitpunkt nur in eine Richtung
- Voll-Duplex: Datenübertragung zeitgleich in beide Richtungen

Verbindung von Rechnern

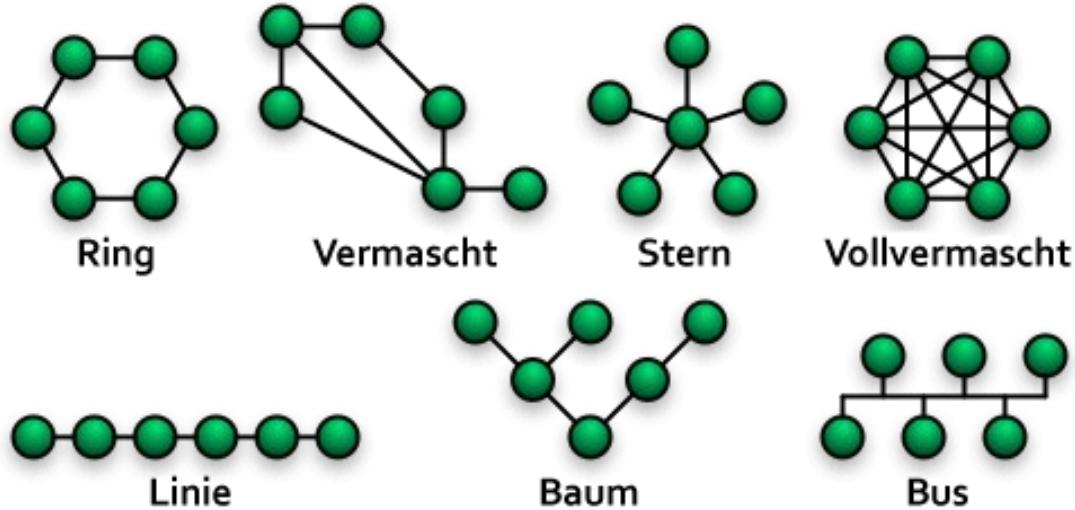
Point-to-Point (Punkt-zu-Punkt)

- Ein Paar von Rechnern ist durch eine direkte Leitung (Kupfer, Glasfaser oder Funk) verbunden
- Normales Vorgehen in Backbones – Verlegung von Verbindungen zwischen je zwei Stationen zur Datenweiterleitung (Router)
- Auch in lokalen Netzen verwendet

Multi-Access-Netz (gemeinsames Medium)

- Nur in lokalen Netzen verwendet
- Alle Stationen sind an ein einziges Medium angeschlossen
- Sendet eine Station Daten, werden sie an alle Stationen ausgeliefert
- Jeder Rechner kontrolliert jedes Paket, ob es für ihn bestimmt ist
- Wie können wir denn mehrere Rechner an eine Leitung anschließen?

Wiederholung: Statische Netztopologien



Topologie	Durchmesser	Bisektionsbr.	Knotengrad
Ring	$N/2$	2	2
Stern	2	1	1 bzw. N-1
Linie	$N-1$	1	1 bzw 2
Bus	1	1	1
Vollvermacht	1	$N/2 * N/2$	$N-1$
Baum (binär)	$2\log_2 N$	1	1, 2 oder 3

Netzinfrastruktur

Bei der Planung eines LANs möchte man die Komplexität des Netzes begrenzen (durch Segmentierung)

Administrative Gründe:

- Wartbarkeit
- Flexibilität der Infrastruktur
- Sicherheit

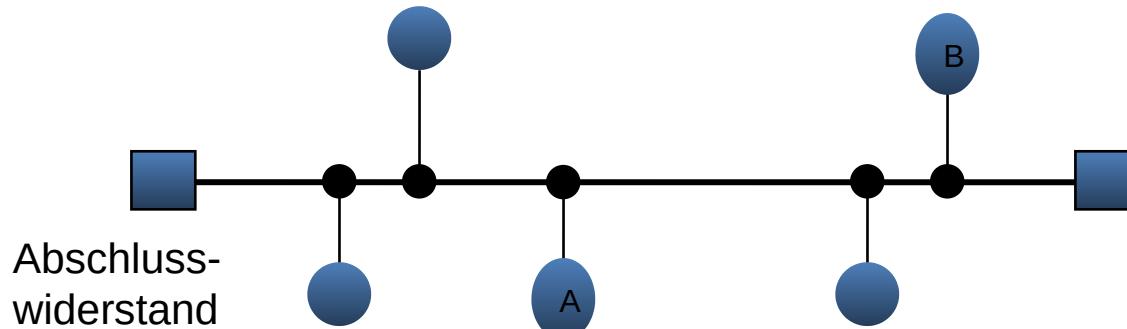
Technische Gründe:

- Maximale Segmentlänge
- Maximale Knotenanzahl im Netz
- Erhöhung der Verzögerung des Zugangs bei Erhöhung der Netzbenutzung
- Erhöhung der Kollisionsgefahr bei Erhöhung der Netzbenutzung

Netztopologien: Der Bus

Bus: Multi-Access-Netz!

- + Einfach, preiswert, einfacher Anschluss neuer Knoten
- + Passive Ankopplung der Stationen, der Ausfall eines Knotens ist kein Problem für die anderen Knoten
- Nur eine Station zu einem Zeitpunkt kann senden; alle anderen Stationen können nur empfangen
- Begrenzung der Zahl anschließbarer Stationen
- Passive Ankopplung der Stationen, daher begrenzte Ausdehnung des Busses (aber: Repeater zur Kopplung mehrerer Busse)



Beispiel:
**Ethernet über
Koaxkabel**

Netztopologien: Der Ring

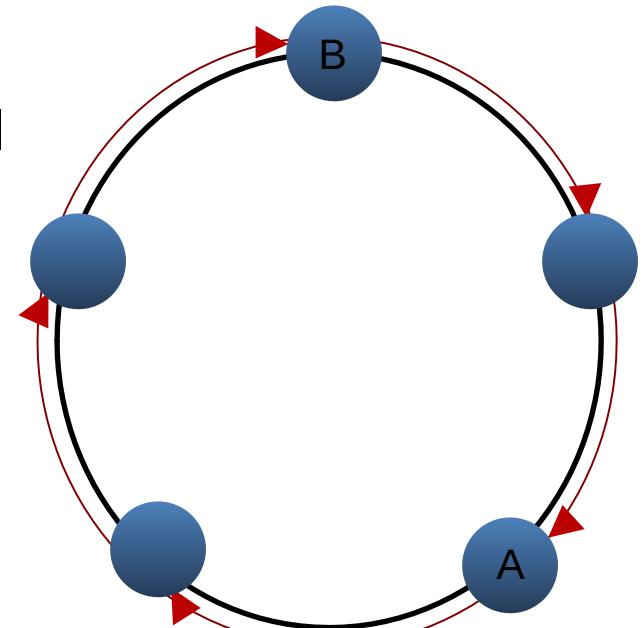
Ring: Point-to-Point

- Reihe von Punkt-zu-Punkt-Verbindungen
- **Aktive Knoten**: fungieren als Repeater
- Ausfall des gesamten Rings bei Unterbrechung einer Verbindung
- Ausfall des gesamten Rings bei Ausfall eines Knotens (Bypass als Abhilfe)
- Große Ausdehnung möglich
- (aufgrund der aktiven Knoten)
- Einfaches Einfügen neuer Knoten

Variante: bidirektonaler Ring

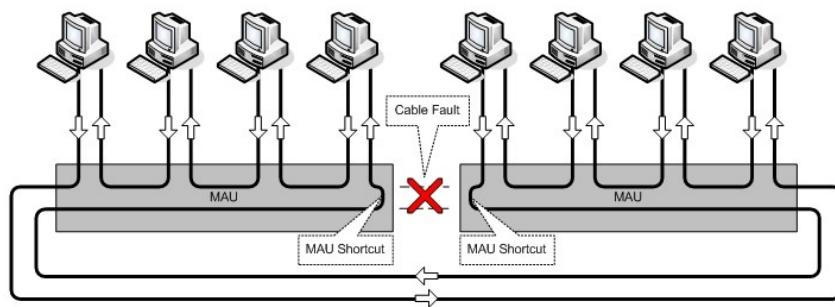
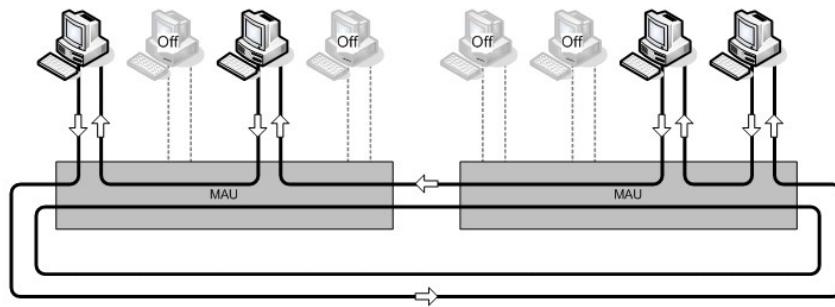
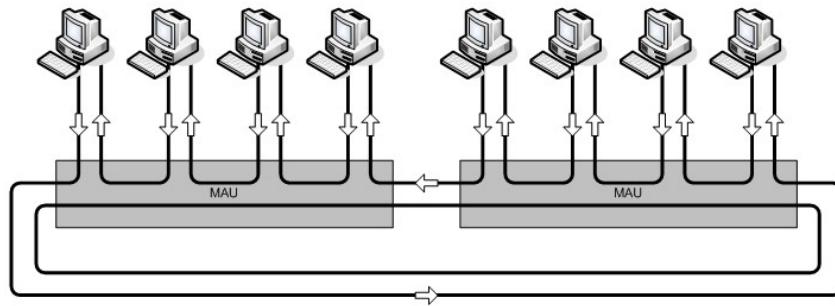
Knoten sind durch zwei gegenläufige

- Ringe miteinander verbunden



Beispiel: **Token Ring, FDDI**

Netztopologien: Fehlerverhalten bei Token Ring

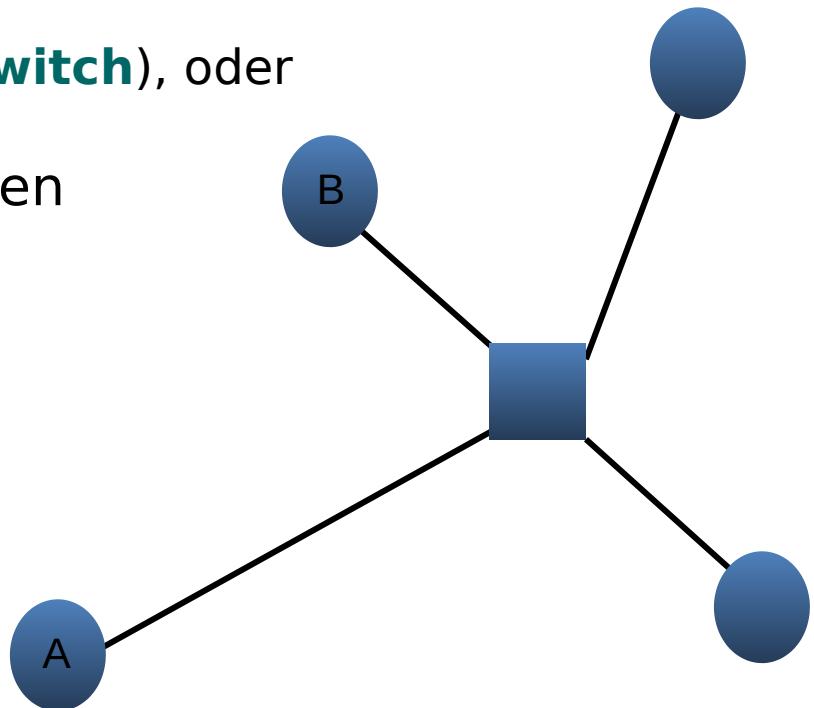


Media Access Unit (MAU)

Netztopologien: Der Stern

Stern

- Ausgezeichneter Knoten als zentrale Station
 - > Nachricht von Station A wird durch die zentrale Station an Station B weitergeleitet
 - > Punkt-zu-Punkt-Verbindungen (**Switch**), oder
 - > Broadcast (**Hub**)
 - > Verwundbarkeit durch zentralen Knoten (Redundanz möglich)

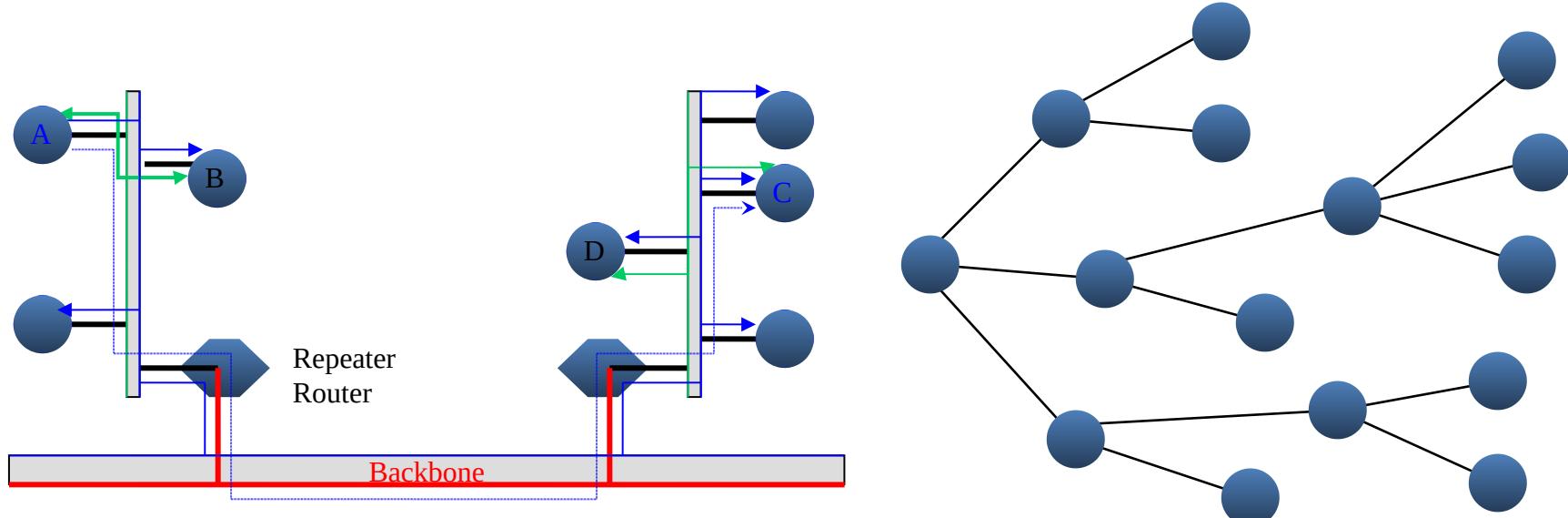


Beispiel: **Fast/Gigabit Ethernet**
über TP-Kabel

Netztopologien: Der Baum

Baum: Zusammenschluss mehrerer Busse oder Sterne

- Verzweigungselemente aktiv (Router) oder passiv (Repeater)
- Überbrückung größerer Strecken
- Gute Anpassung an vorgegebene geographische Gegebenheiten
- Minimierung der erforderlichen Kabellänge



- Neben den Computern bzw. Datensenken- und quellen gibt es **weitere Komponenten** in der Netzinfrastruktur!
- Typische Aufgaben:
 - Verstärken eines schwach gewordenen Signals
 - Bei Multi-Access Netzen: Begrenzung einer ‚Kollisionsdomäne‘
 - Kopplung unterschiedlicher physikalischer Netze
 - logische Verbindung mehrerer Netze zur Datenweiterleitung
 - ...

Netzinfrastruktur

Zur Kopplung von Netzen werden spezielle Netzwerkknoten benötigt. Diese lassen sich hierarchisch bzgl. ihrer Funktionalität anordnen:

Repeater

- vergrößert einzelne lokale Netze gleichen Typs (physikalisch)

Hub

- Koppelt mehrere gleichen lokale Netze

Brücke

- Koppelt mehrere eventuell unterschiedliche lokale Netze

Switch

- Wie Hub, aber 'intelligenter' bzgl. Datenweiterleitung

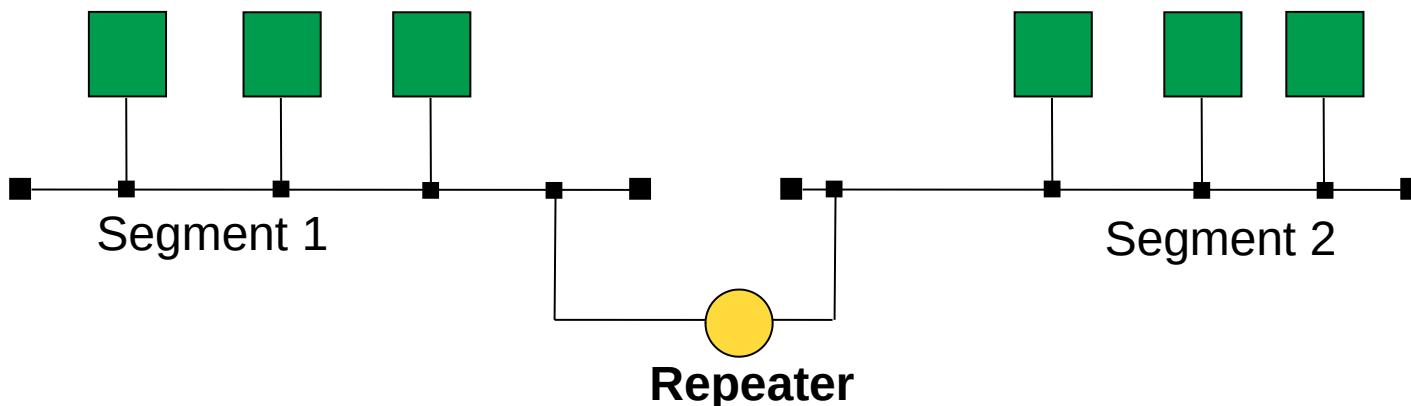
Router

- Verbindet mehrere LANs mit gleichem Netzwerkprotokoll über weite Strecken

Infrastrukturkomponenten: Repeater

Repeater:

- Verknüpfung von zwei Netzen zur Vergrößerung der Ausdehnung
- Arbeitet auf der Bitebene
 - > Kann einkommende Signale als „0“ oder „1“ interpretieren
 - > Empfang und **Auffrischung** des Signals - ein empfangenes Bit wird auf der anderen Seite neu als Stromimpuls codiert
- Kein Verstehen von Adressen höherer Schichten, alle Daten werden weitergeleitet (das Netz bleibt z.B. ein Multi-Access-Netz)

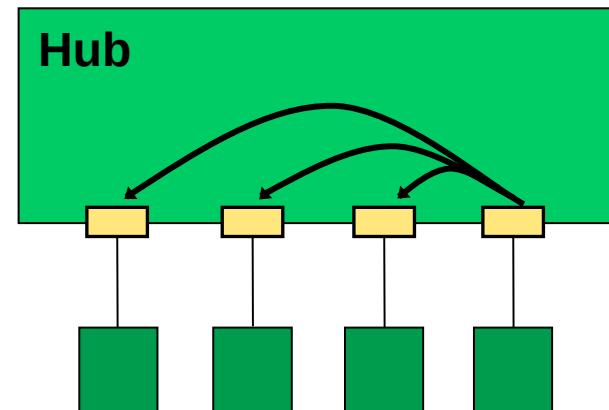


Infrastrukturkomponenten: Hub



Hub = „Repeater mit mehr als zwei Anschlüssen“

- Signalauffrischung wie beim Repeater
- An einen Anschluss kann ein einzelner Rechner oder ein ganzer Bus angeschlossen werden
- Multi-Access-Netz: der Hub gibt ein empfangenes Signal auf allen Anschlüssen wieder aus
- Praktisch wie ein ‚Bus‘ mit mehreren Anschlüssen:
- **Gemeinsamer Übertragungskanal**, d.h.
- Stationen können nicht gleichzeitig senden und empfangen, nur eine Station auf einmal kann senden
- Geringe Sicherheit, da alle Stationen mithören können

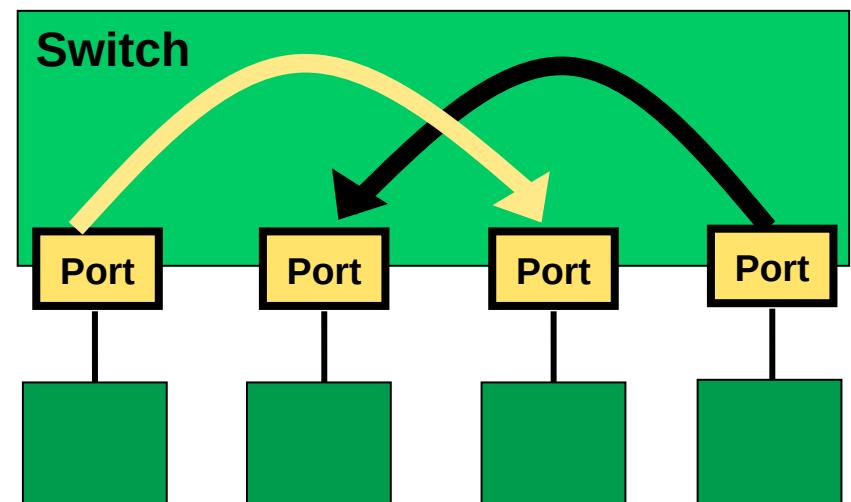
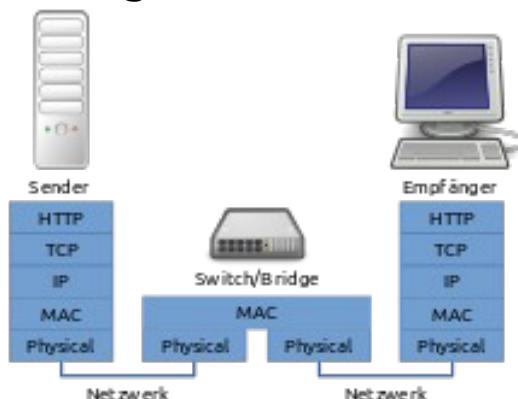


Infrastrukturkomponenten: Switch



Wie Hub, aber:

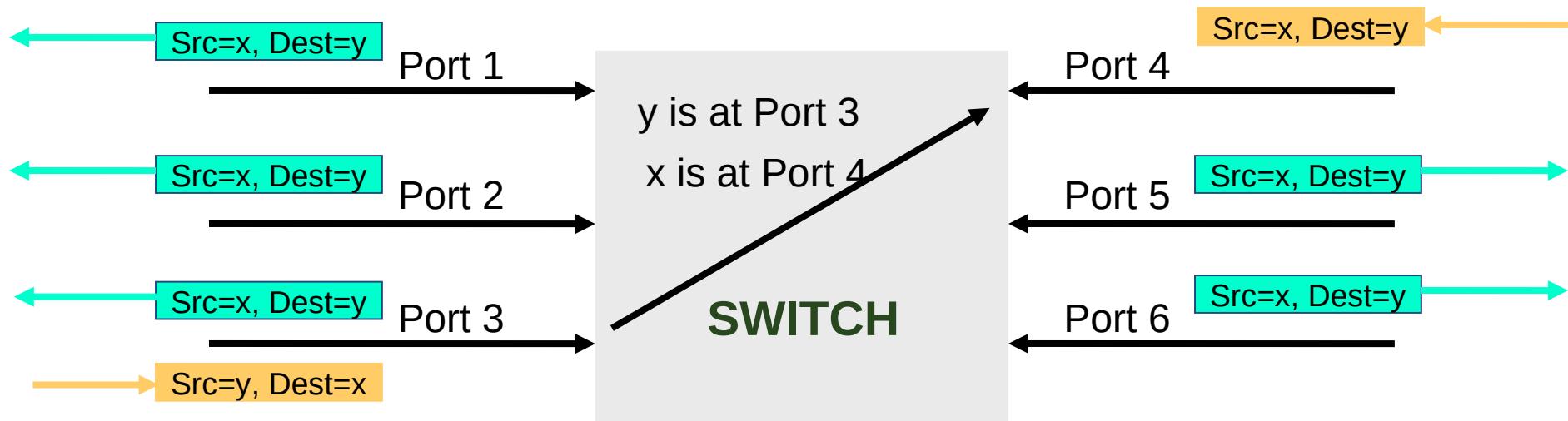
- *Punkt-zu-Punkt-Kommunikation* zwischen zwei Stationen
 - > Switch kann Layer-2-Adressen (MAC-Adressen) der angeschlossenen Stationen verstehen, lernt sie und kann Daten gezielt weiterleiten
 - > Stationen können gleichzeitig senden und empfangen
 - > Nur der adressierte Empfänger erhält die Daten, andere Stationen können nicht mithören
- Vermeidung von Kollisionen („Mikrosegmentierung“)
- Puffer für jeden Port
- „Layer-3-Switch“
- Integriert mit Routing



Switches - Lernen von Adressen

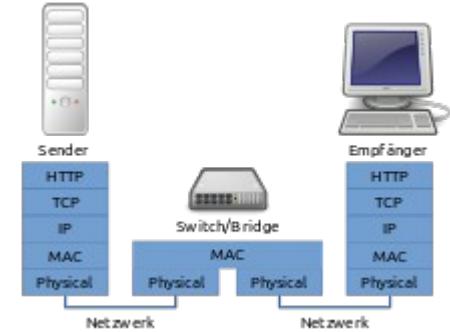
Autokonfiguration von Switches

- Zu Beginn unbekannt, welcher Rechner an welchem Port hängt
- Werden Daten für eine unbekannte Adresse auf einem Port empfangen: Weiterleitung an alle anderen Ports (Broadcast)
- Die Absenderadresse der Daten kann für den Port, über den sie empfangen wurden, gespeichert werden



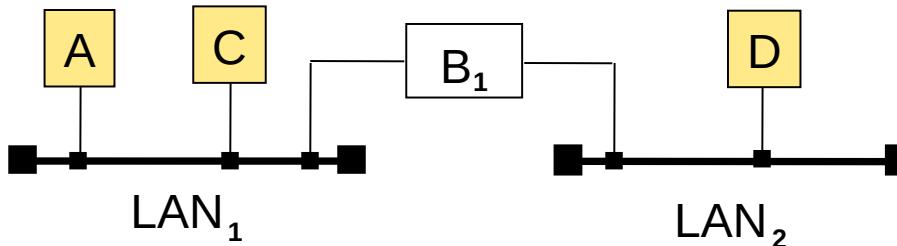
Infrastrukturkomponenten: Brücken

- Brücken dienen der Kopplung von zwei oder mehreren LANs
- Können auch auf LLC-Ebene Arbeiten (Switches in der Regel nur auf MAC Layer), und dadurch unterschiedliche Netztechnologien verbinden
- Dienen der Trennung von Kollisionsdomänen



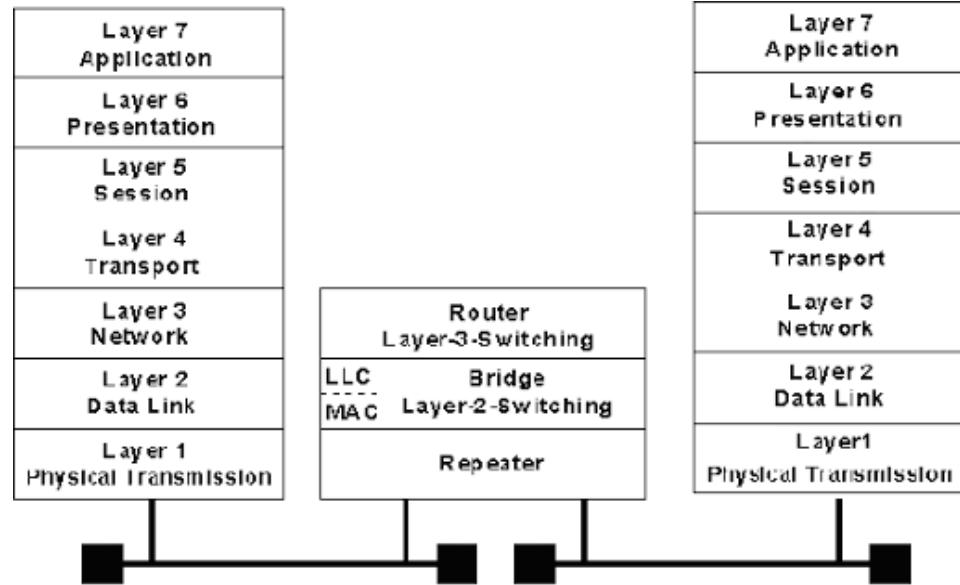
Hauptaufgaben:

- geeignete Weiterleitung der Daten
- Anpassung an unterschiedliche LAN-Typen
- Reduzierung des Verkehrs in einem LAN-Segment, d.h. Pakete, die von A an C gesendet werden, werden von der Brücke nicht in LAN₂ durchgelassen. Dadurch kann Station D parallel senden.
- Aufhebung physikalischer Längenbegrenzungen
- Erhöhte Zuverlässigkeit durch Abgrenzung der LAN-Segmente

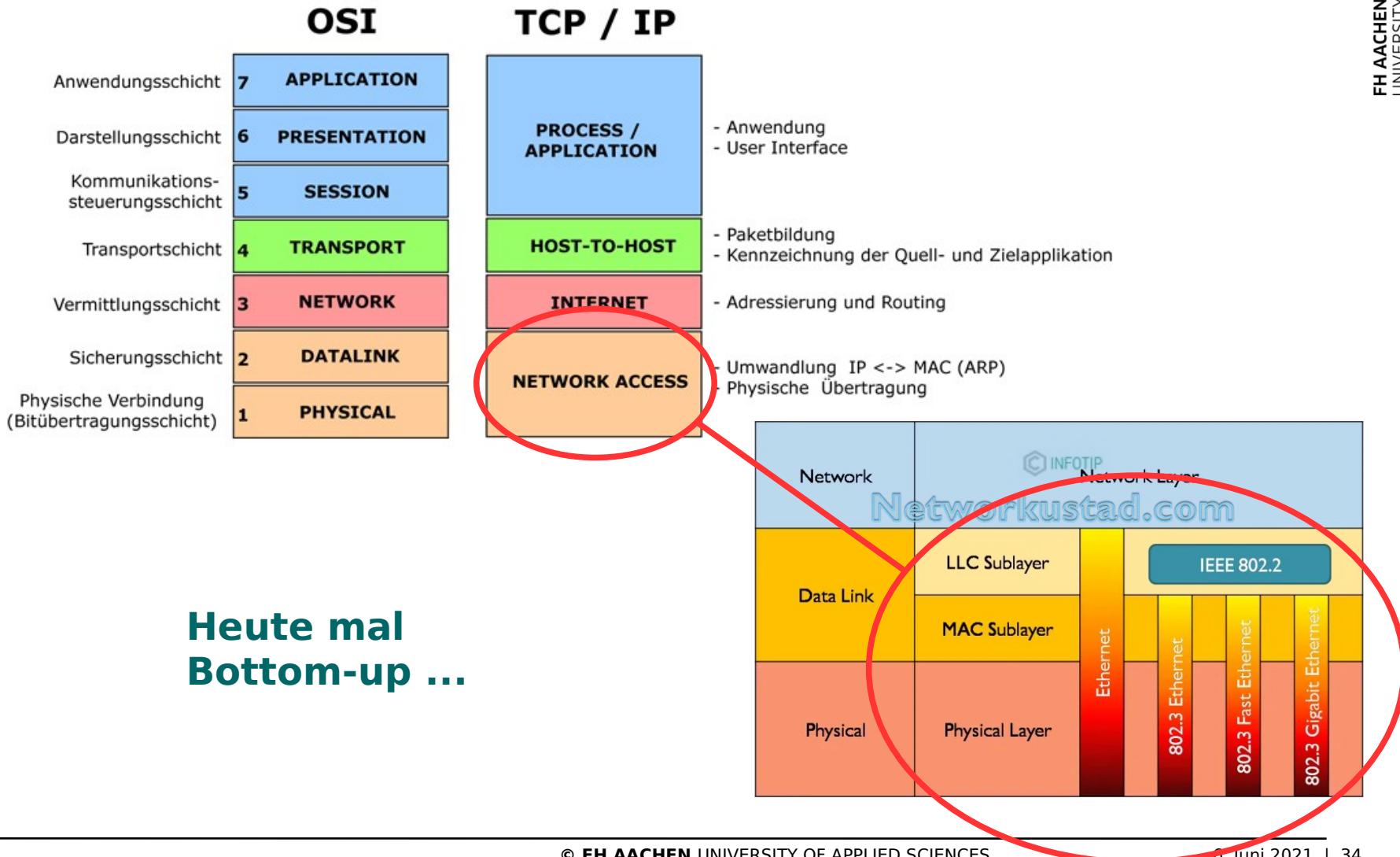


Infrastrukturkomponenten: Router

- Weiterleitung von Datenpaketen auf Layer 3 (Netzwerk-Schicht).
- Logische Verbindung unterschiedlicher Netze zum Routing der Daten
- Switches und Bridges arbeiten nur auf MAC-Ebene, verarbeiten also keine Adressen mit einer hierarchischen Struktur und keinem geographischen Bezug (ARP erforderlich!)
- Mit Brücken und Switches gekoppelte LANs bilden ein “großes LAN”, obwohl eine Trennung oft wünschenswert wäre (z.B. in Bezug auf Verwaltung oder Fehler)



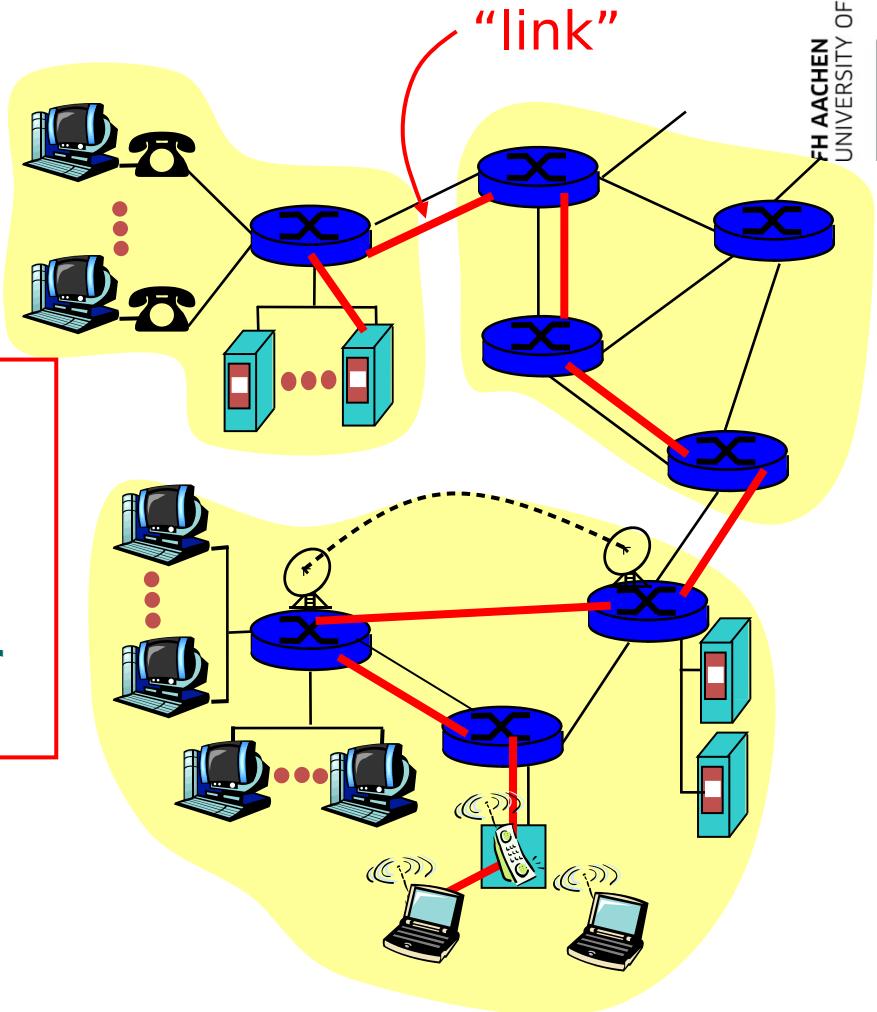
Schichten 1 und 2 im ISO/OSI Referenzmodell und im TCP/IP Referenzmodell



Sicherungsschicht: Die Link-Layer

Die Sicherungsschicht hat die Aufgabe, die Datagramme über das gemeinsame Kommunikationsmedium zu einem anderen Knoten zu transportieren

Direkte Kommunikation innerhalb einer Netzwerktechnologie



Dienste der Sicherungsschicht

Medium ACcess (MAC) Layer

- Das **Kanalzugriffsprotokoll** beschreibt, nach welchen Regeln auf ein Übertragungsmedium zugegriffen werden darf, d.h. ein Rahmen auf der Verbindungsleitung übertragen werden darf
- Bei Punkt-zu-Punkt-Verbindungen ist das Leitungszugriffsprotokoll einfach. Der Sender kann einen Rahmen senden, wann immer die Verbindungsleitung frei ist (bei vollduplex immer)
- Bei Multi-Access-Netzen teilen sich mehrere Teilnehmer eine Verbindungsleitung, z.B. nach dem Bus-Prinzip. Hier übernimmt der MAC Layer die Koordination der Leitungsnutzung
- Der MAC-Layer liefert eine eindeutige Kennung für jedes Netzwerkgeräte bzw. jede Netzwerk-Karte → **MAC** Adresse

Dienste der Sicherungsschicht

MAC und LLC Layer = Sicherungsschicht

- Die Sicherungsschicht verkapselt die Daten in sog. Rahmen (Frames)
- Der Rahmen hat in der Regel einen **Header und Trailer** (ggf. mehrere Felder).
- Aufgabe der Schicht ist es, das Format zu beschreiben, und einen einfachen, **sicheren** Datenpaket-orientierten Übertragungsdienst zur Verfügung zu stellen
- Diese Pakete müssen auch Adressen enthalten → MAC Adressen
- Datenkommunikation nur innerhalb des selben LANs
- Zuverlässige Datenübertragung: Zusicherung einer fehlerfreien Übertragung eines Datagrams. Einsatz ähnlicher Verfahren wie bei den Transportprotokollen.
- Leitungen mit niedriger Fehlerrate verzichten ggf. auf diesen Dienst (Glasfaser), Funknetze typischerweise nicht!

Weitere Dienste der Sicherungsschicht

Flusskontrolle:

- Flusskontrolle kennen wird schon von den Sliding-Window-Protokollen. Auch die Sicherungsschicht passt die Transmissionsrate den Fähigkeiten des Empfängers an und verhindert so etwaige Pufferüberläufe

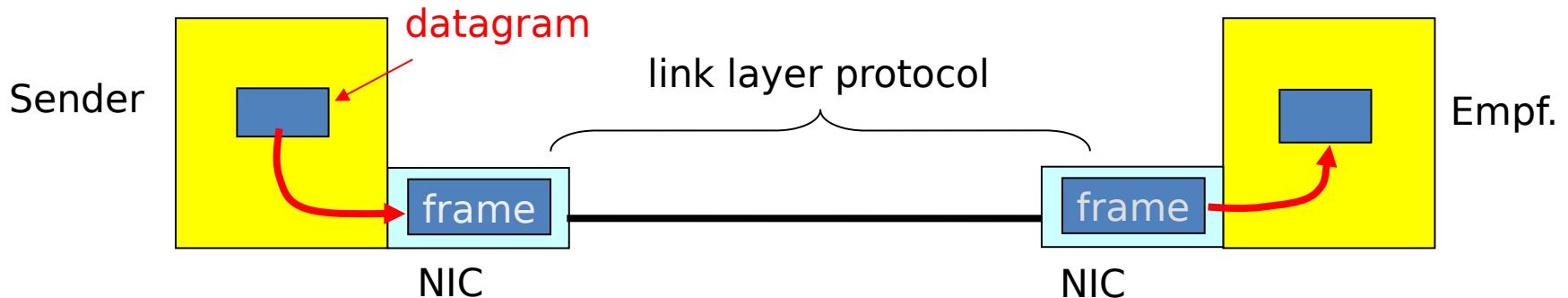
Fehlererkennung:

- Fehler werden durch elektromagnetisches Rauschen und Signaldämpfung verursacht
- Durch den Einsatz redundanter Fehlererkennungsbits kann der Empfänger Fehler erkennen (NAK oder Eliminieren des Rahmens)

Fehlerbehebung:

- Je nach Redundanz der Fehlererkennungsbits können einige Fehler sogar ohne erneute Übertragung korrigiert werden
- Forward Error Correction (FEC)

Kommunizierende Netzwerkartenkarten



Die Sicherungsschicht wird in einer Netzwerkartenkarte (aka NIC) implementiert

- Ethernet card, PCMCI card, 802.11 card

NIC-Adapter des Senders:

- Verkapselt das Datagramm in einen Rahmen
- Setzt ggf. Fehlerprüfbits, implementiert etwaige Mechanismen für die zuverlässige Übertragung und Flusskontrolle, etc.

NIC-Adapter des Empfängers

- Prüft auf Fehler, implementiert etwaige Methoden für die zuverlässige Übertragung und Flusskontrolle, etc.
- Extrahiert das Datagramm und leitet es an Vermittlungsschicht des Empfängers weiter

NIC-Adapter ist halbautonom
Implementiert Bit- und Sicherungsschicht

Datenrahmen in der Sicherungsschicht bei Ethernet

Ein Ethernet-Frame im Kontext mit maximalen IPv4- / TCP-Daten

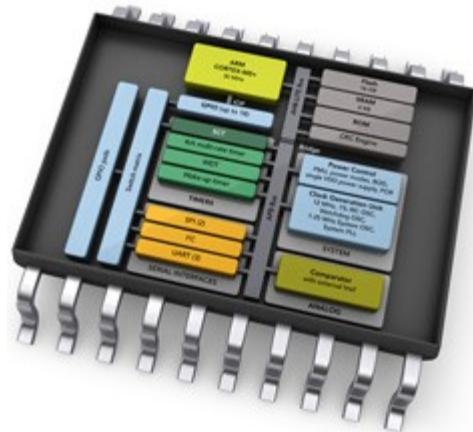
Schicht 4: TCP-Segment							TCP-Header	Nutzlast (1460 bytes)			
Schicht 3: IP-Paket							IP-Header	Nutzlast (1480 bytes)			
Schicht 2: Ethernet-Frame			MAC-Empfänger	MAC-Absender	802.1Q-Tag (opt.)	EtherType	Nutzlast (1500 bytes)		Frame Check Sequence		
Schicht 1: Ethernet-Paket+IPG	Präambel	Start of Frame	Nutzlast (1518/1522 bytes)							Interpacket Gap	
Oktette	7	1	6	6	(4)	2	20	20	(6-)1460	4	12

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

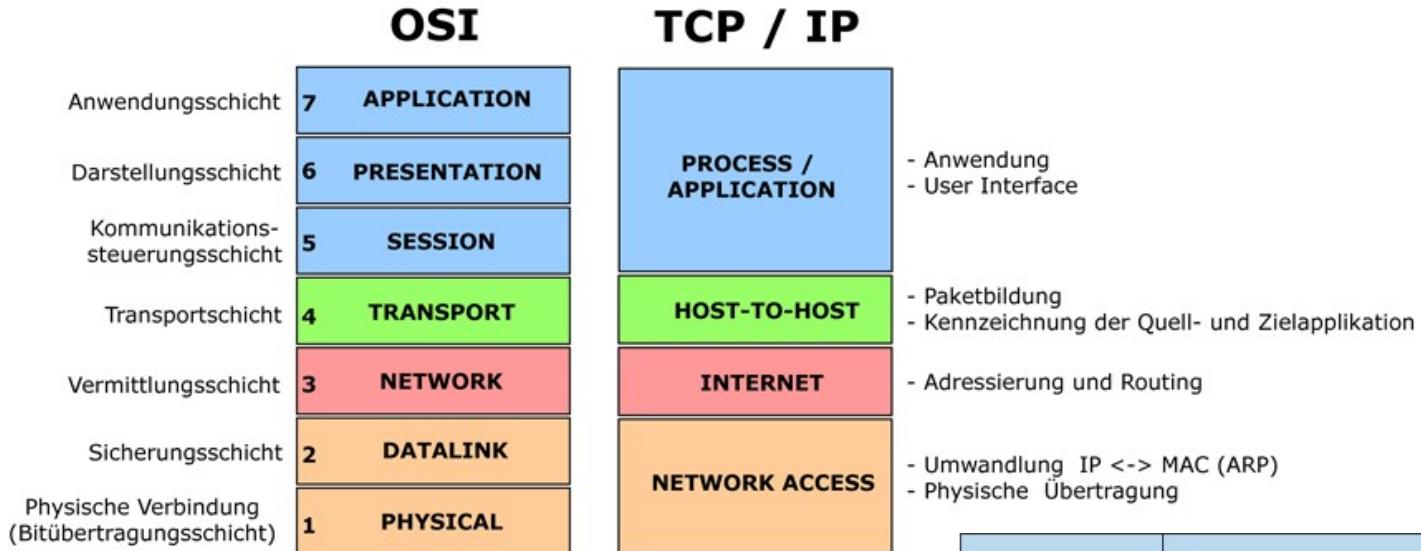
Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge

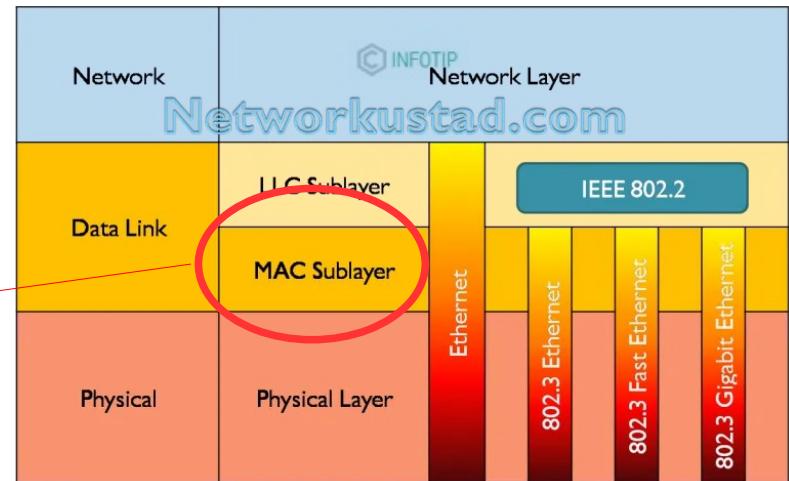


Schichten 1 und 2 im ISO/OSI Referenzmodell und im TCP/IP Referenzmodell



Medium Access Control:

Welcher Kommunikationsteilnehmer kann wann und wie lange das Medium nutzen?



Mehrfachzugriffsprotokolle

Zwei Grundtypen von Verbindungsleitungen:

- Punkt-zu-Punkt
 - PPP für Einwahlverfahren mit Modem/ISDN
 - Verbindungsleitung zwischen einem Ethernet-Switch und Rechner
- **Broadcast (geteiltes Medium)**
 - Traditionelle Ethernet
 - 802.11 Wireless LAN

Mehrfachzugriffsprotokolle

Verteilter Algorithmus zur Regelung der gemeinsamen Nutzung eines geteilten Übertragungsmediums

- **Wann** kann **wer** etwas senden?
- Störungsfreier Kanalzugriff

Hierbei ist zu berücksichtigen, dass die zur Koordination notwendige Kommunikation auch über das geteilte Medium abgewickelt werden muss!

- **Kontrollnachrichten**
- Keine out-of-band-Signalisierung

Eine Ideales Mehrfachzugriffsprotokoll

Broadcast-Medium mit Maximalrate R bps

1. Ein einzelner Knoten kann mit der Rate R übertragen
2. M Knoten können mit einer mittleren Rate von R/M übertragen
3. Das Protokoll ist dezentral, d.h. es gibt keine Master-Knoten, die ausfallen und das ganze System zum Absturz bringen können
4. Das Protokoll ist einfach und kann somit kostengünstig implementiert werden

MAC-Protokolle: Eine Taxonomie

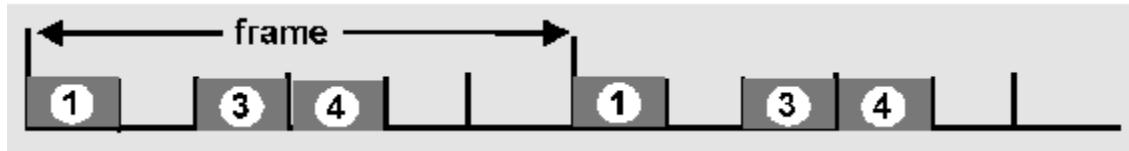
Drei grobe Kategorien:

- **Kanalaufteilungsprotokolle (Multiplexing)**
 - Aufteilung des Übertragungskanals in kleine Übertragungseinheiten (Zeitfenster, Frequenzen, Kodierung)
 - Exklusive Nutzung der Übertragungseinheiten für einzelne Knoten
- **Zufallszugriffsprotokolle**
 - Keine Unterteilung des Kanals, jeder überträgt mit der gesamten Leitungskapazität. Allerdings sind Kollisionen von Nachrichten möglich und müssen korrekt behandelt werden
- **Rotationsprotokolle**
 - Der Zeitpunkt, wann ein Knoten senden kann wird durch eine spezielle Koordinierung nach einem flexiblen Rotationsprinzip im Übertragungsmedium ausgetauscht.

Kanalaufteilungsprotokolle: Zeitmultiplexing

TDMA: Time Division Multiple Access

- Aufteilung des Kanals in kleine Zeiteinheiten, die sich in Runden wiederholen
- Jeder Knoten kann eine bestimmte Anzahl an Zeiteinheiten in jeder Runde Nutzen
- Ungenutzte Einheiten (slots) bleiben ungenutzt
- Beispiel: 6-Knoten LAN, 1,3,4 haben Daten, Slots 2,5,6 idle

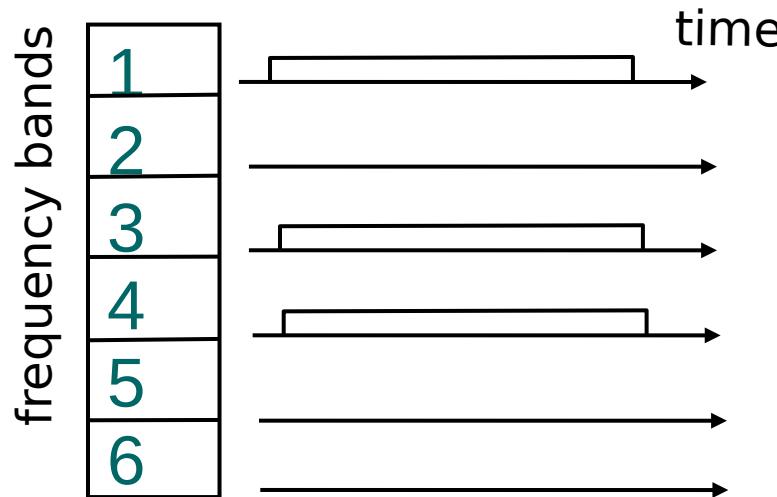


- Die Zeiteinheiten können meist auch gruppiert angefordert werden
- Anforderung entspricht Konfiguration der NIC-Karte

Kanalaufteilungsprotokolle: Frequenzmultiplexing

FDMA: Frequency Division Multiple Access

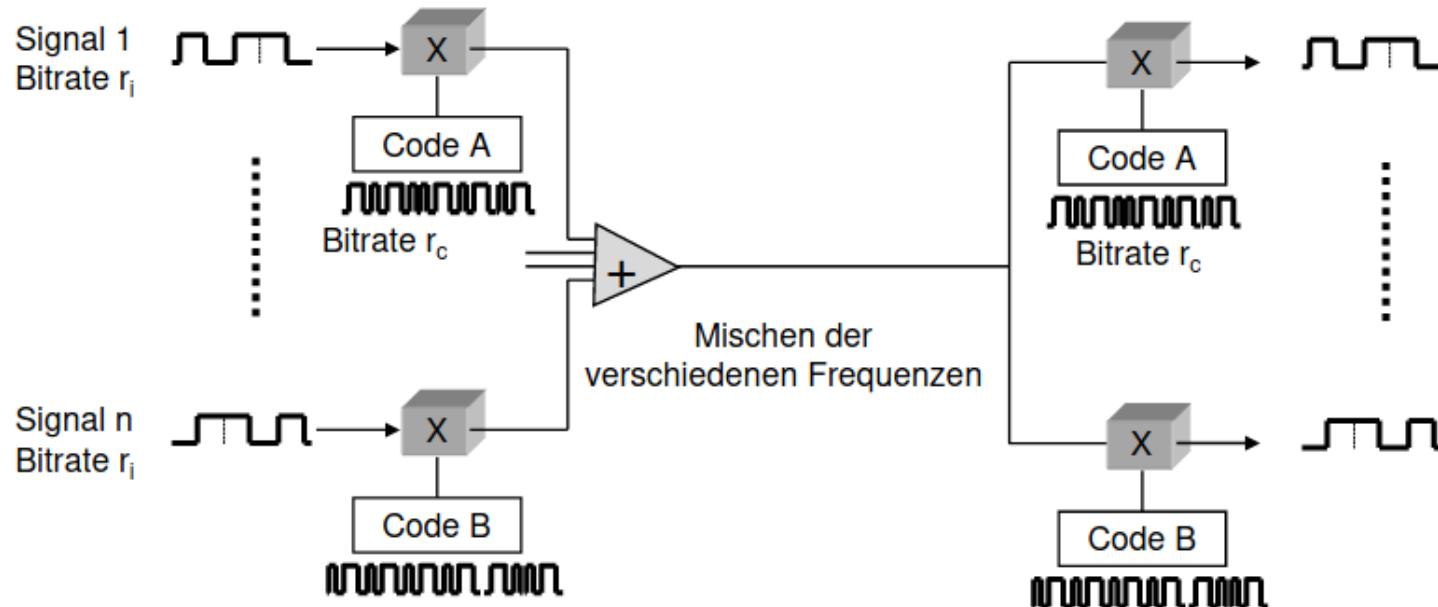
- Aufteilung des Kanals in Frequenzbänder
- Nach Nyquist kann ein rauschfreier Kanal mit einem Frequenzband der Breite x Hz binäre Signale nicht mit mehr als $2x$ Bps übertragen (Nyquist-Bandbreite)
- Jede Station nutzt ein Frequenzband
- Die Kapazität nicht genutzter Frequenzbänder geht verloren
- Beispiel: 6-Knoten LAN, 1,3,4 haben Daten, 2,5,6 idle



Kanalaufteilungsprotokolle: Frequenzmultiplexing

CDMA: Code Division Multiple Access

- Keine Unterteilung in Zeitslots oder Frequenzen
- Nutzung von ‚orthogonalen Codes‘ bei der (De-) Modulation, die sich im Medium überlagern/mischen, aber trotzdem auf Empfängerseite eindeutig wiederhergestellt werden können
- Vorteil: Es gibt keine ungenutzten Ressourcen wie bei TDMA/FDMA



Zufallszugriffsprotokolle

Sendet ein Knoten, so

- nutzt er die vollständige Kanalrate R .
- Keine a priori Koordination

Kollisionen treten auf, wenn mehrere Knoten gleichzeitig senden:

- Wie werden diese erkannt?
- Berücksichtigung der Signal-Ausbreitungsgeschwindigkeit!
- Wie werden diese behandelt (z.B. erneutes Verschicken nach einem zufälligen Zeitintervall) via ‚delayed retransmissions‘

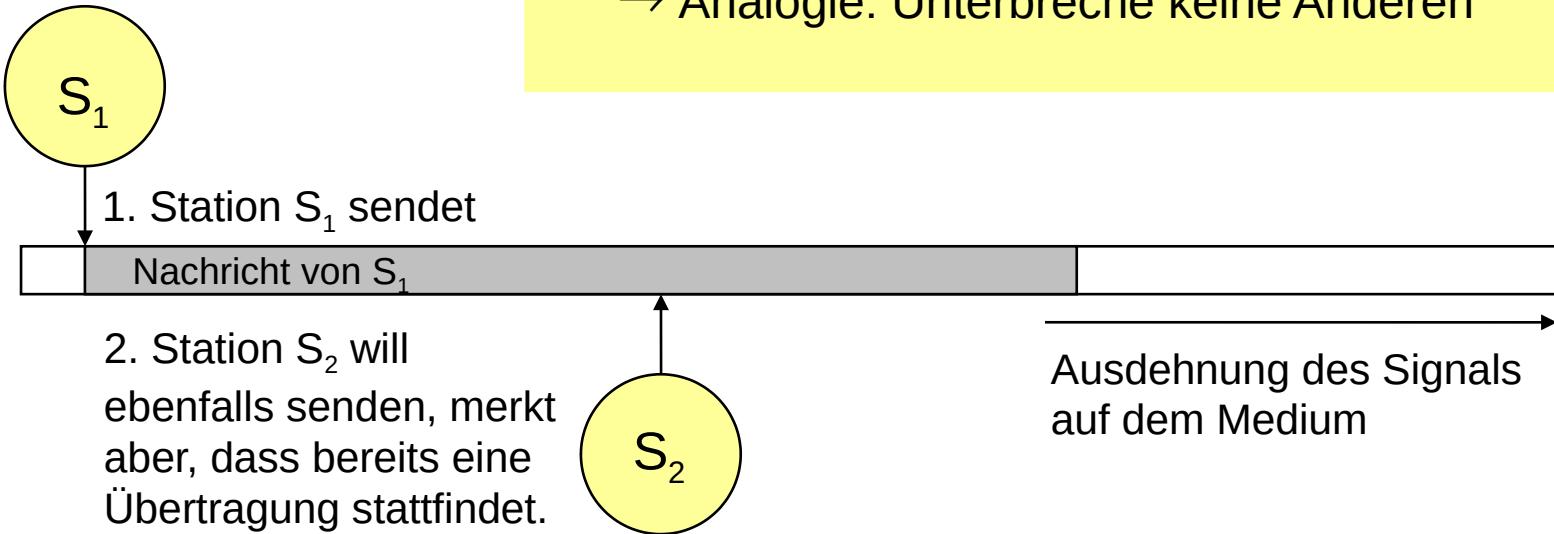
Beispielprotokolle:

- slotted ALOHA
- ALOHA
- **CSMA**, CSMA/CD, CSMA/CA

Carrier Sense Multiple Access (CSMA)

Prinzip:

- höre vor der Übertragung das Medium ab
- sende nur, falls das Medium frei ist
- Analogie: Unterbreche keine Anderen

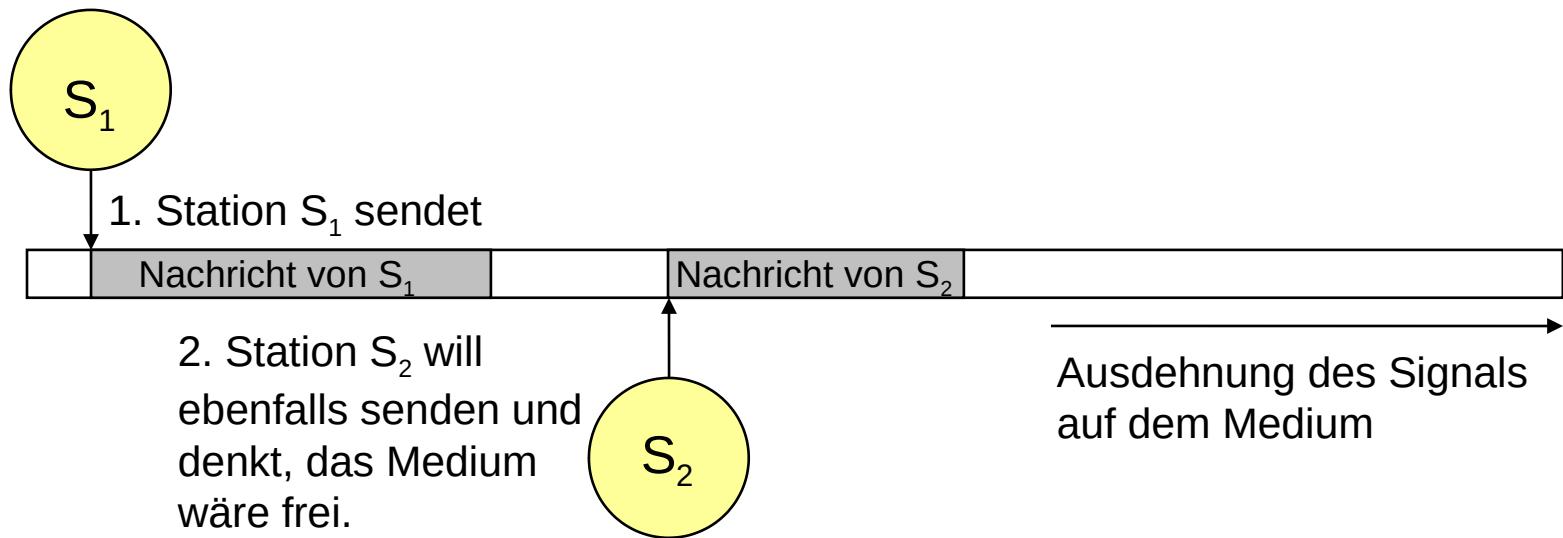


Vorteile: einfach, da die Stationen nicht koordiniert werden müssen; mit einigen Erweiterungen trotzdem gute Ausnutzung der Netzkapazität

Nachteil: kein garantierter Zugriff, es ist eine große Verzögerung möglich
→ keine Echtzeitfähigkeit!

Problem bei CSMA

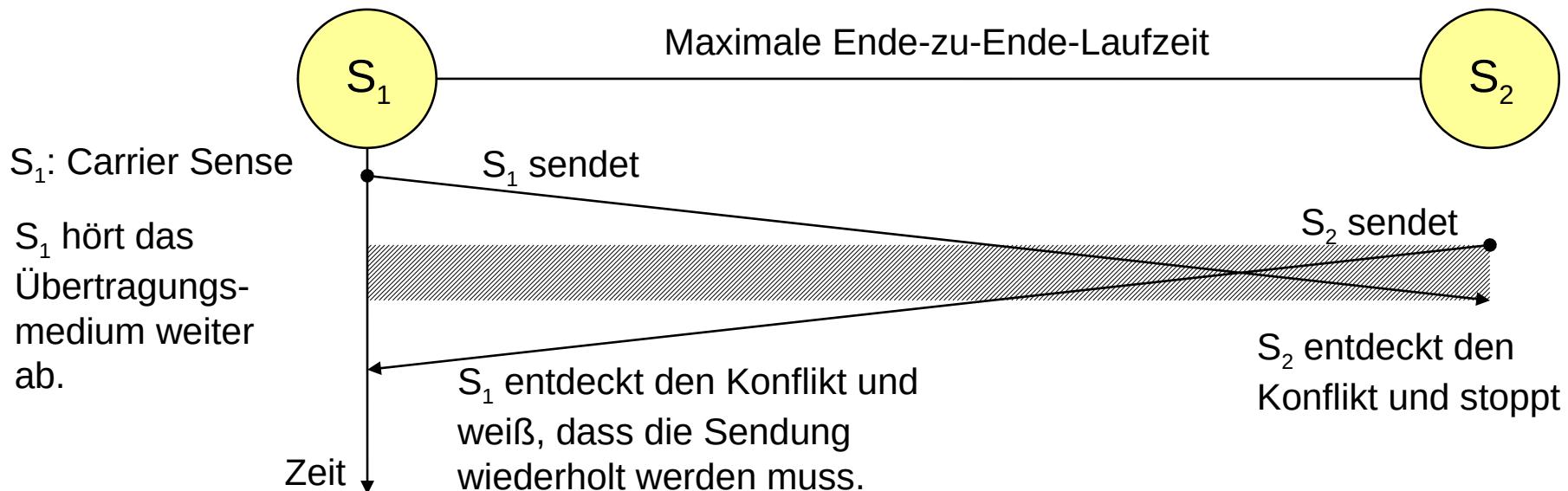
Problem: die Nachricht, die S_1 sendet, breitet sich mit endlicher Geschwindigkeit auf dem Medium aus. Daher kann es sein, dass S_2 denkt, das Medium wäre frei, obwohl S_1 schon mit der Sendung begonnen hat. Beide Nachrichten überlagern sich auf dem Medium und werden unbrauchbar.



Prüfe auf Kollision (CD)

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

- wie CSMA, zusätzlich: höre während der Sendung das Medium weiter ab und breche die Übertragung ab, wenn eine Kollision auftritt
- sende ein Jamming-Signal, damit jede Station weiß, dass eine Kollision aufgetreten ist und die Nachricht nutzlos geworden ist.
- Wichtig: Man muss so lange senden, dass man ein Jamming -Signal während dessen auch mitbekommt



Anmerkung: mit zunehmender Ausdehnung des Netzes steigt die Gefahr eines Konflikts.
Daher ist diese Technik nur für "kleine" Netze geeignet.

Ethernet

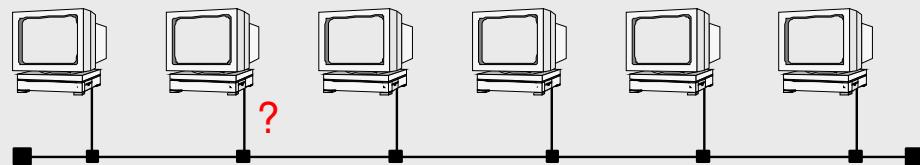
basiert auf dem Standard IEEE 802.3 “**CSMA/CD**”

(Carrier Sense Multiple Access/Collision Detection)

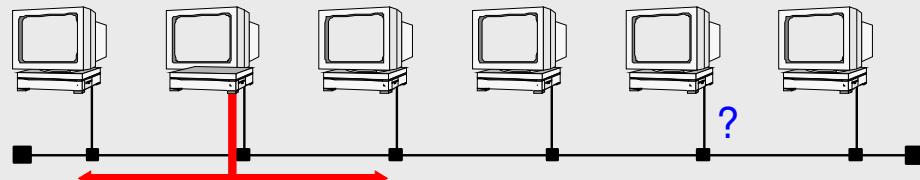
mehrere (passive) Rechner - ein gemeinsames Medium (Random Access)

ursprünglich Bustopologie:

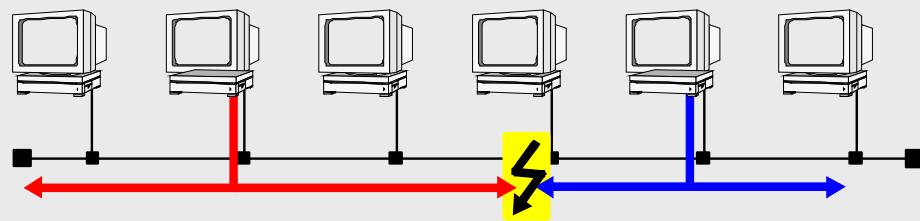
1. Ist das Medium frei?
(Carrier Sense)



2. Übertragen



3. Prüfe auf Kollision (Collision Detection)
Falls ja: sende Jamming und breche ab.
Danach weiter mit Binary Exponential
Backoff Algorithmus



Vorgehen bei Kollisionen

Problem:

Wenn zwei Nachrichten eine Kollision haben, versuchen die Stationen eine erneute Übertragung. Wann sollen die Stationen die Wiederholung starten?

Binary Exponential Backoff

Um nach einer Kollision die gleichzeitige Wiederholung der kollidierten Sendungen zu vermeiden (Folgekollision), wird eine zufällige Wartezeit aus einem vorgegebenen Intervall gezogen. Das Intervall wird *klein* gehalten, um große Wartezeiten bis zur Wiederholung zu vermeiden. Dadurch ist allerdings das Risiko eines Folgekonflikts groß. Kommt es so zu einer weiteren Kollision, wird das Intervall vor dem nächsten Versuch vergrößert, um mehr Spielraum für alle sendenden Parteien zu schaffen.

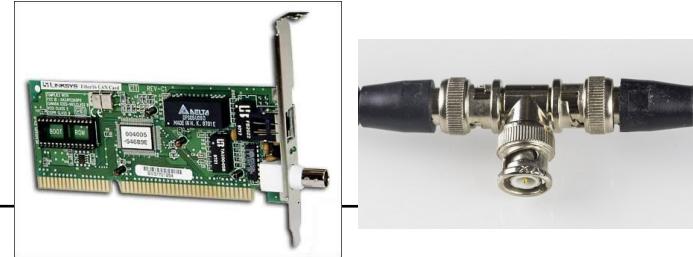
Die Wartezeit wird dabei folgendermaßen ermittelt:

- Hatte eine Station bereits i Kollisionen, würfelt sie eine Zahl x aus dem Intervall $[0, 2^i-1]$ (bei Kollision 10 - 15 bleibt das Intervall fix bei $[0, 2^{10}-1]$, nach der 16. Kollision erfolgt ein Abbruch)
- Sobald das Medium frei ist, wartet der Sender x Zeitslots, wobei ein Zeitslot der minimalen Ethernet-Rahmenlänge von 512 Bit, also bei 10 Mbit/s 51.2 µs, entspricht.
- Nach dem x -ten Zeitslot beginnt die Station erneut mit Carrier Sense.

Datenübertragung bei Ethernet

- Verwendet CSMA/CD, bei Kollisionen wird das Binary Exponential Backoff-Verfahren angewendet
- Ziel: Realisieren großer Netzwerke, die aber trotzdem eine geringe Kollisionswahrscheinlichkeit haben...
... Resultat: die maximale Ausdehnung wird auf **2500 Meter** festgelegt
Die maximale Zeit zur Entdeckung einer Kollision ist knapp zweimal so lang wie die Signallaufzeit auf dem Medium.
Bei einer Signalgeschwindigkeit von ungefähr 100000 km/s (10 µs/km) erhält man (unter Berücksichtigung der Zeit in 4 Repeatern) eine maximale Ende-zu-Ende-Signallaufzeit von 25 µs. Die maximale Konfliktdauer ist damit ca. **50 µs**. Um einen Konflikt mit Sicherheit zu erkennen, muss die Station mindestens 50 µs auf dem Medium horchen und senden → bei 10MBit/s mindestens 500Bit
Darauf basierend wurde für eine Senderate von 10 MBit/s eine *minimale Rahmenlänge* (aufgerundet **64 Byte**) definiert, um eine Kollisionserkennung auch im Worst-Case zu ermöglichen!

Entwicklung des Ethernet



- 70er Jahre: experimentelles Netzwerk auf Basis von Koaxialkabeln, Datenrate von 3 Mb/s. Entwickelt von der Xerox Corporation als ein Protokoll für LANs mit sporadischem aber burst-artigem Verkehrsverhalten.
- 1980: gemeinsame Weiterentwicklung durch die Digital Equipment Corporation, die Intel Corporation und Xerox zu einer 10 Mb/s-Variante.
- Original Ethernet-Struktur: Bus-Topologie mit einer maximalen Segmentlänge von 500 Metern, Anschlussmöglichkeit für maximal 100 passive Stationen. Repeater werden verwendet, um mehrere Segmente zusammenzuschließen.
- Gängigstes Medium: Kupferkabel. Aber auch Glasfaser-Kabel kommen zum Einsatz (erhöht die Segmentlänge).
- Frühe 90er Jahre: die Bus-Topologie wird mehr und mehr von einer Stern-Topologie verdrängt, bei der ein zentraler Switch Punkt-zu-Punkt-Verbindungen (auf Twisted-Pair oder Glasfaserkabel basierend) zu allen Stationen realisiert. Der Switch bietet hierbei den Vorteil, dass mehrere Verbindungen parallel ablaufen können.

Ethernet

Basiert auf IEEE 802.3 „CSMA/CD“

4 Klassen von Ethernet-Varianten:

- Standard Ethernet → 10 Mb/s Kaum noch im Einsatz
- Fast Ethernet → 100 Mb/s Ehemals meistverbreitete Variante
- **Gigabit Ethernet** → 1000 Mb/s Heute meist verbreitet
- 10Gigabit-Ethernet → 10000 Mb/s Standardisiert

Ethernet hat sich im LAN-Bereich durchgesetzt. Es wird in der überwiegenden Anzahl der LANs als Infrastruktur eingesetzt:

- Es ist einfach zu verstehen, umzusetzen und zu überwachen
- Das Netzwerk ist in der Anschaffung **billig**
- Bringt **Flexibilität** bzgl. der Topologie mit sich
- Es gibt **keine Kompatibilitäts-Probleme**, jeder Hersteller kennt und befolgt den Standard
- Das Netz ist prinzipiell nicht echtzeitfähig...

Daten zu Ethernet

Parameter	Ethernet	Fast Ethernet	Gigabit Ethernet
Maximale Ausdehnung	bis zu 2800 Meter	205 Meter	200 Meter
Kapazität	10 Mb/s	100 Mb/s	1000 Mb/s
Minimale Rahmenlänge	64 Byte	64 Byte	520 Byte
Maximale Rahmenlänge	1526 Byte	1526 Byte	1526 Byte
Signal-darstellung	Manchester-Code	4B/5B-Code, 8B/6T-Code, ...	8B/10B, ...
Maximale Anzahl Repeater	5	2	1

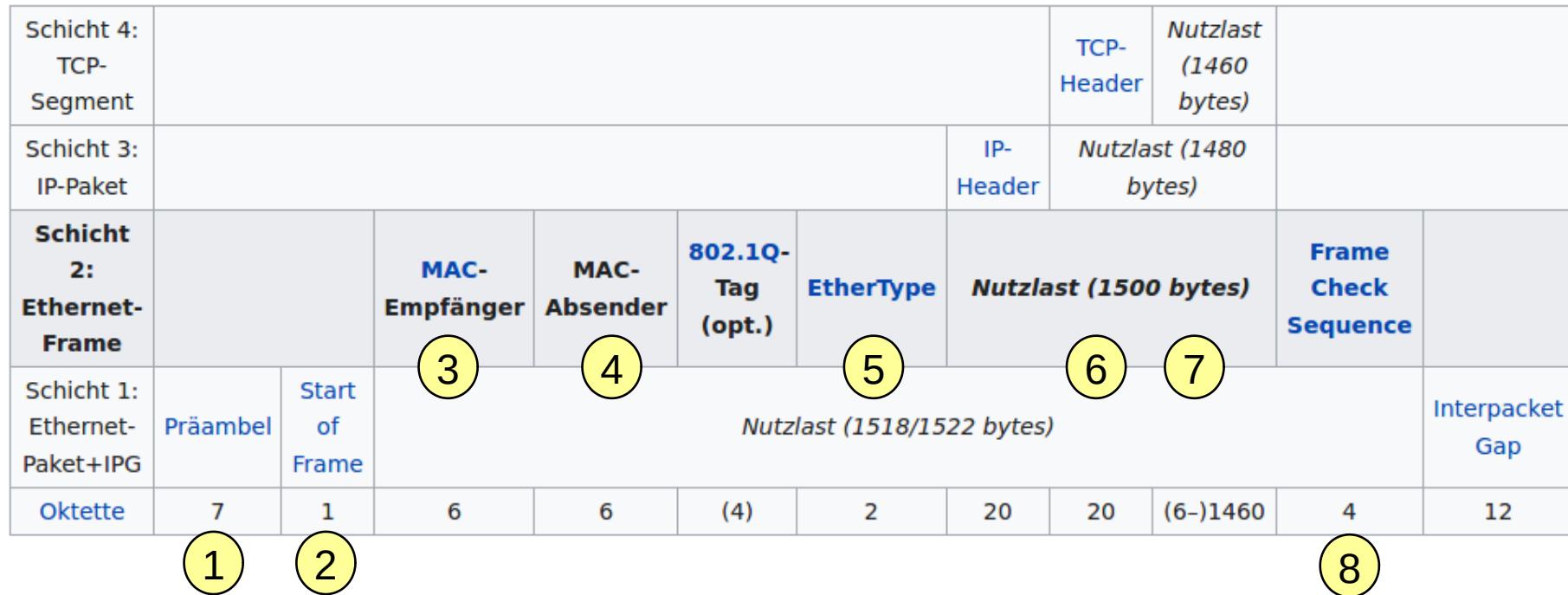
Zusätzlich:

Ein weiterer „Rahmen“ wird zum „Jamming“ benutzt:

Dies ist ein 4-Byte-Störsignal zur Kollisionserkennung

Nachrichtenformat bei Ethernet

Ein Ethernet-Frame im Kontext mit maximalen IPv4- / TCP-Daten



- 1: 7 Byte Synchronisation
Jedes Byte beinhaltet 10101010
- 2: 1 Byte Start Frame Delimiter
Markierung des Rahmenbeginns durch das Byte 10101011
- 3: 6 Byte Destination Address
- 4: 6 Byte Source Address
- 5: 2 Byte Length/Type
Angabe der Länge des Datenfelds bzw. Typ des Schicht-3-Protokolls

- 6: (0-1500) Byte Daten
- 7: n Byte Padding
Auffüllen des Rahmens auf mindestens 64 Byte (Kleinere Fragmente werden im Netz verworfen, abgesehen von dem Jamming-Signal)
- 8: 4 Byte Frame Check Sequence
Verwendung eines zyklischen Codes

Der Ethernet-Rahmen

- *Präambel*: kennzeichnet eine folgende Übertragung und synchronisiert den Empfänger mit dem Sender.
- Der *Start-of-Frame-Delimiter*: (bzw. die beiden aufeinanderfolgenden Einsen) zeigen an, dass endlich Daten folgen.
- *Destination Address*: das erste Bit kennzeichnet den Empfänger: entweder eine einzelne Station (1. Bit = 0) oder eine Gruppenadresse (1. Bit = 1; Broadcast ist auch hier durch 11...1 gegeben).
- *Length/Type*: Bei einem Wert bis 1500 wird die Angabe als Länge des Datenteils aufgefasst (dies ist der Fall beim so genannten CSMA/CD), bei einem Wert ab 1536 wird hier angegeben, an welches Schicht-3-Protokoll die Daten weitergegeben werden sollen (verwendet bei Ethernet).
- *FCS*: Checksumme, 32-Bit CRC. Diese erstreckt sich über die Felder DA, SA, Length/Type, Data/Padding.

Ethernet-Varianten

Angabe der verwendeten Ethernet-Variante durch 3 Namenskomponenten:

- 1 - Kapazität in Mb/s (also 10, 100, 1000 oder 10G)
- 2 - Übertragungstechnik (z.B. **Base** für Basisband, Broad für Broadband)
- 3 - maximale Segmentlänge in Einheiten von 100 Metern, bzw. Art des verwendeten Mediums

Beispiele:

- 10Base-T: 10 Mb/s, Basisband, Twisted-Pair-Kabel
- 100Base-T2: 100 Mb/s, Basisband, 2 Twisted-Pair-Kabel
- 1000Base-X: 1000 Mb/s, Basisband, Glasfaser-Kabel

Von der Variante hängen noch Parameter wie z.B. die minimale Rahmenlänge ab (wegen unterschiedlicher Signallaufzeiten):
1000Base-X: minimale Rahmenlänge 416 Bytes
1000Base-T: minimale Rahmenlänge 520 Bytes

Fast Ethernet

Prinzip: behalte Ethernet bei, aber mache es schneller:

- Kompatibilität mit existierenden Ethernet-Netzen
- 100 MBit/s an Übertragungsrate, erreicht durch bessere Technik, effizientere Codes, Nutzung mehrerer Leitungspaare, Switches, ...
- Resultat: IEEE 802.3u, 1995

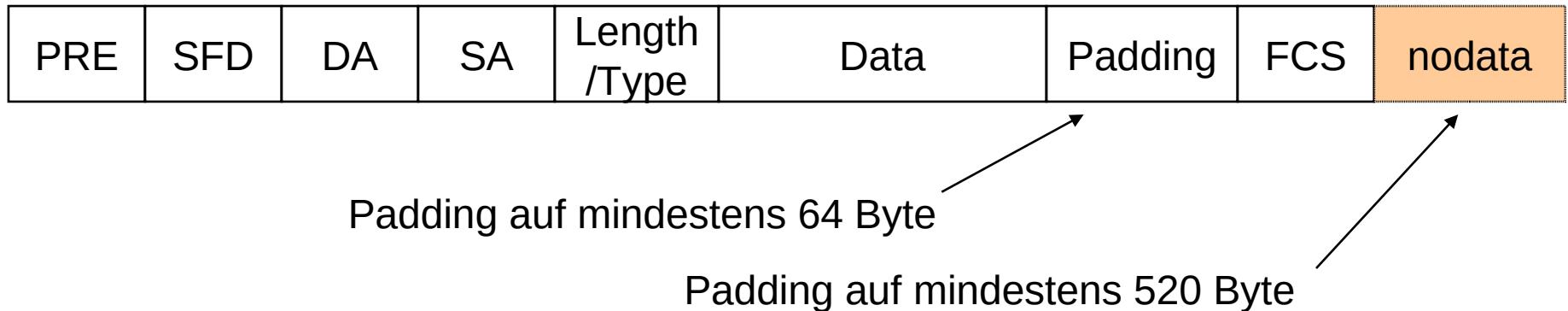
Problem:

- Die minimale Rahmenlänge zur Kollisionserkennung bei Ethernet beträgt 64 Byte. Bei 100 Mb/s wird der Rahmen aber ca. 10 Mal so schnell abgesendet, so dass eine Kollisionserkennung nicht mehr gewährleistet ist.
- ***Resultat: für Fast Ethernet musste die Ausdehnung ca. um den Faktor 10 auf etwas mehr als 200 Meter reduziert werden...***

Wechsel zu Gigabit-Ethernet

Gigabit-Ethernet (IEEE 802.3z, 1998)

- Kompatibilität zu Fast Ethernet beibehalten!
- Problem: zur Kollisionserkennung wäre eine **Reduktion** der Ausdehnung **auf 25m notwendig...**
 - > Daher: **Ausdehnung von Fast Ethernet beibehalten (200m)**
 - > Neue minimale Rahmenlänge von 520 Byte
 - > Trotzdem Beibehaltung des alten Rahmenformats: füge zweites Padding-Feld hinter dem Rahmen an (**Carrier Extension**)



Wechsel zu Gigabit-Ethernet

Auch maximale Rahmenlänge unzureichend

- Ermögliche Versenden mehrerer aufeinanderfolgender Rahmen (**Frame Bursting**) ohne wiederholt CSMA/CD anzuwenden
- Versendung von bis zu 5 Rahmen in Folge, getrennt durch „Inter-Frame Gaps“ (IFGs)
 - > Spezielles Bitmuster, Medium bleibt für andere Stationen belegt



- Im Normalfall werden nur noch Switches eingesetzt
 - > Es treten keine Kollisionen mehr auf
 - > Maximale Kabellänge nur noch durch die Signalabschwächung bestimmt
 - > Trotzdem ist CSMA/CD noch implementiert (half duplex erlaubt!)

1000Base-T/X (Gigabit Ethernet)

1000Base-T

- Basiert auf Fast Ethernet (mindestens UTP Kat. 5)
- Nutzung aller 4 Kabelpaare mit quaternärem Code, Kabellänge: 100 m

1000Base-SX

- Multimode-Glasfaser mit 550 m Segmentlänge
- Übertragung auf dem 850nm-Band

1000Base-LX

- Übertragung auf 1300nm
- Mono- oder Multimode mit Reichweite bis 5000 m

1000Base-LH

- Monomode Übertragung auf 1550nm
- Reichweite bis zu 70 km

10-Gigabit Ethernet (IEEE 802.3ae)

- Erst nur für Glasfaser spezifiziert (LX oder SX), bis 40km
- Mittlerweile auch für Twisted Pair (Kat. 6 oder 7)
 - > Wüste Trickserei auf Schicht 1, um trotz Dämpfung und Verzerrung 50 bzw. 100m Reichweite zu erreichen
- Verzicht auf CSMA/CD, da sowieso keine Kollisionen mehr auftreten können (Sterntopologie mit Switch)

Mittlerweile

- 40G-Ethernet, 100G-Ethernet (802.3ba)
 - > Beibehaltung der Rahmenformate...
 - > 7 Meter über Twisted Pair, bis zu 40 km über Glasfaser

Gibt es auch Alternativen zu Ethernet?

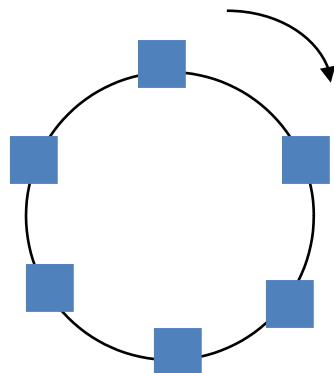
Token Ring (IEEE 802.5)

Ursprüngliche Konkurrenz zu Ethernet

- “Token”-Verfahren, nur wer ein bestimmtes **Token** (=Bitfolge) besitzt, darf senden
- Die Rechner teilen sich einen Ring aus Punkt-zu-Punkt-Verbindungen
- Das Token wird zyklisch weitergegeben

Eigenschaften:

- Garantierter Zugriff, keine Kollisionen
- Sehr gute Ausnutzung der Netzkapazität,
- hohe Effizienz
- Fair, garantierte Antwortzeiten
- Aber: aufwändig und teuer



Weitergabe des Tokens

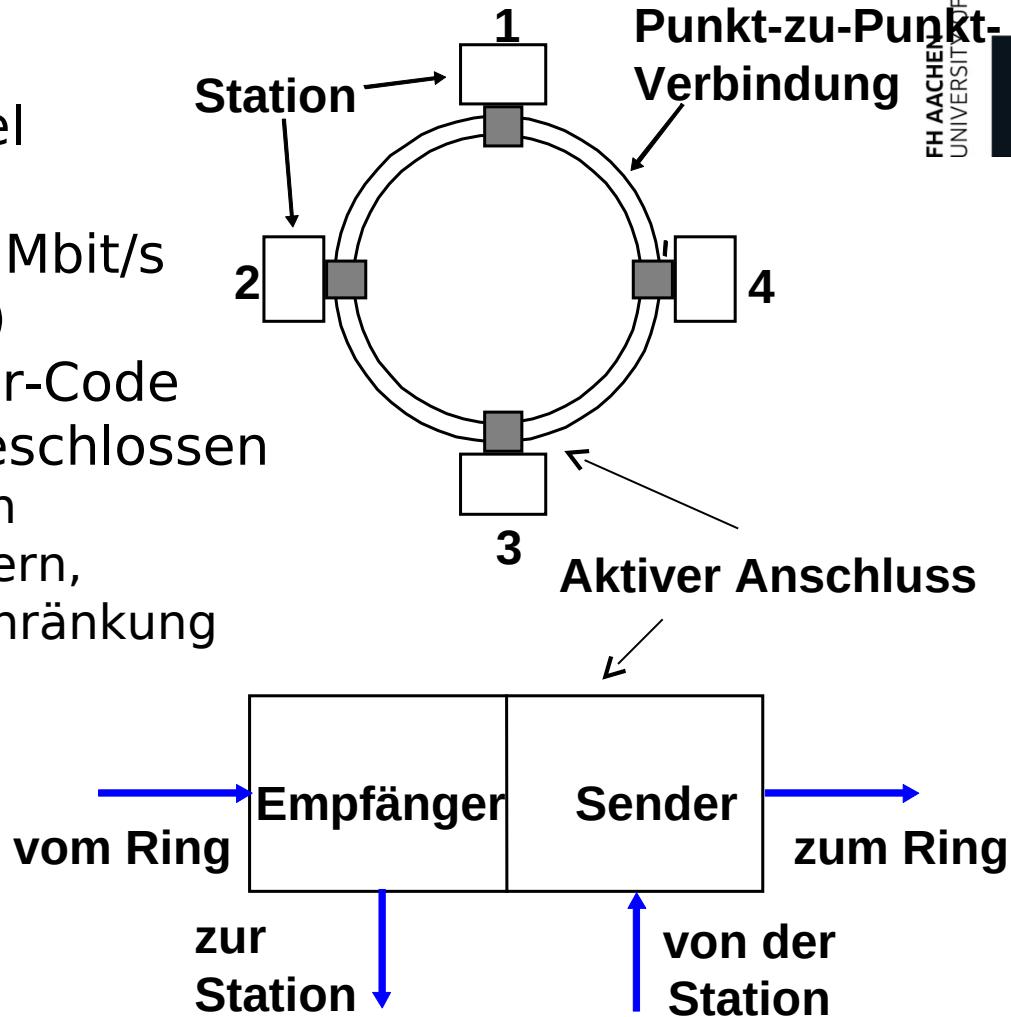
Token Ring

Parameter

- Twisted Pair, Koaxialkabel oder Glasfaser
- Datenrate von 4 bzw. 16 Mbit/s (100MBit/s mit Glasfaser)
- Differentieller Manchester-Code
- Stationen sind aktiv angeschlossen
- Empfangene Signale werden regeneriert (wie bei Repeatern, daher prinzipiell keine Beschränkung der Ausdehnung)



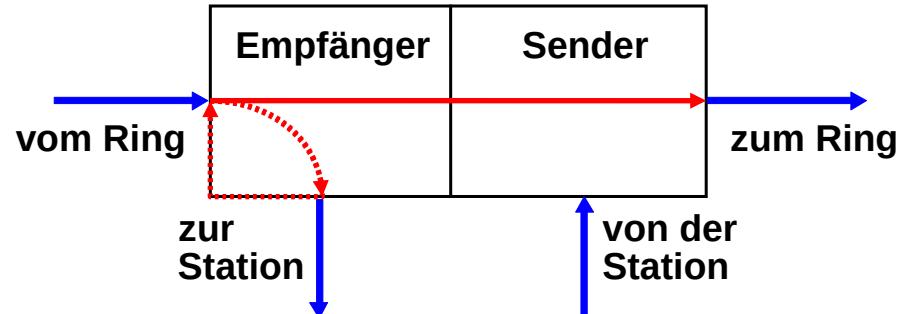
Media Access Unit (MAU)



Senden und Empfangen

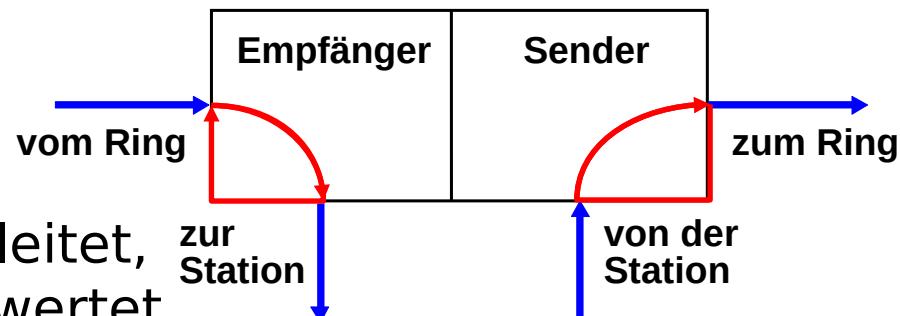
Grundzustand

- Daten werden vom Ring seriell empfangen
- An Station gerichtete Daten
- werden kopiert
- Daten werden seriell
- weitergeleitet



Sendezustand

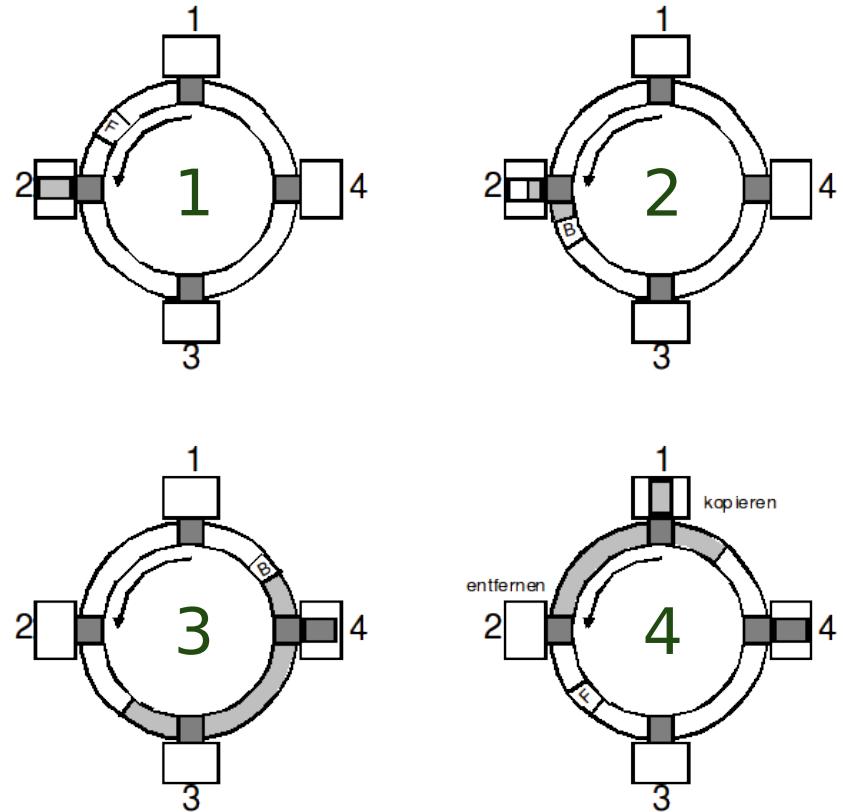
- Der Ring wird aufgetrennt
- Eigene Daten werden
- seriell gesendet
- Vom Ring kommende
- Daten werden nicht weitergeleitet,
- sondern in der Station ausgewertet



Zugriff beim Token Ring

Beispiel: Station 2 sendet an Station 1

1. Station 2 wartet auf freies Token (Sendeberechtigung, 3-Byte-Token)
2. Station 2 wandelt freies Token in belegtes um (= Rahmen-Header); danach sendet 2 den Rahmen (und ggfs. weitere Rahmen, für maximal 10 ms)
3. Station 2 beendet den Rahmen und wartet, bis dieser wieder bei ihr ankommt
4. Station 1 kopiert den Rahmen; Station 2 entfernt ihn vom Ring und erzeugt ein neues, freies Token



Token Ring als Alternative?

Token Ring WAR eine Konkurrenz zu Ethernet:

- Fair, garantierte Antwortzeiten, keine Kollisionen
- Aber: Verwaltungsaufwand! Was passiert, wenn durch einen Übertragungsfehler das Token verfälscht wird?

→ Ethernet ist **DER** Standard für lokale Netze

Fazit: Ethernet

Standardnetz für LANs

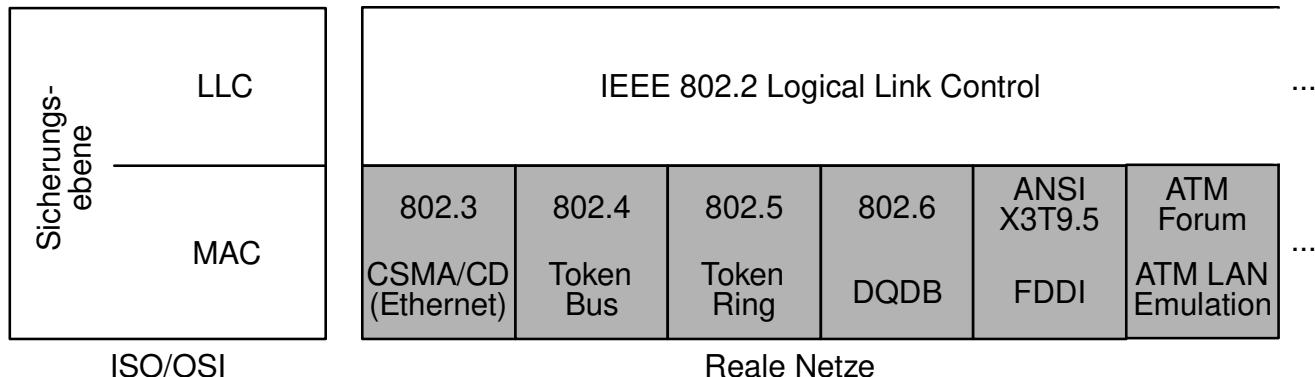
- Vielzahl von Varianten, flexible Topologie
 - > Heutzutage vorwiegend Stern, teilweise erweitert zu Bäumen
- Einschränkungen durch Beibehaltung der Kompatibilität zu älteren Varianten
 - > Rahmenformat
 - > Kollisionserkennung heute kaum noch nötig
- Oft auch im MAN verwendet

Wird durch weitere Technologien z.B. im Bereich Funkübertragung und Weitverkehrsnetze ergänzt:

- WLAN im drahtlosen Bereich
- DSL zum Anschluss von Heimnetzen an das Internet
- SDH im Weitverkehrsbereich

Schicht 2: Aufteilung in zwei Aufgabenbereiche

- **Logical Link Control (LLC)** (Schicht 2b)
 - Einteilung der zu sendenden Daten in *Rahmen (Frames)*
 - Multiplexing mehrerer Layer-3 Protokolle über das gleiche Medium
 - Bereitstellung einer (möglichst) fehlerfreien Übertragung zwischen benachbarten Knoten durch:
 - *Erkennung (und Behebung) von Übertragungsfehlern*
 - *Flusskontrolle* (Vermeidung der Überlastung des Empfängers)
 - *Pufferspeicher*
- **Medium Access Control (MAC)** (Schicht 2a)
 - Regelung des Zugriffs auf den Kommunikationskanal bei Broadcast-Netzen



Sicherungsschicht

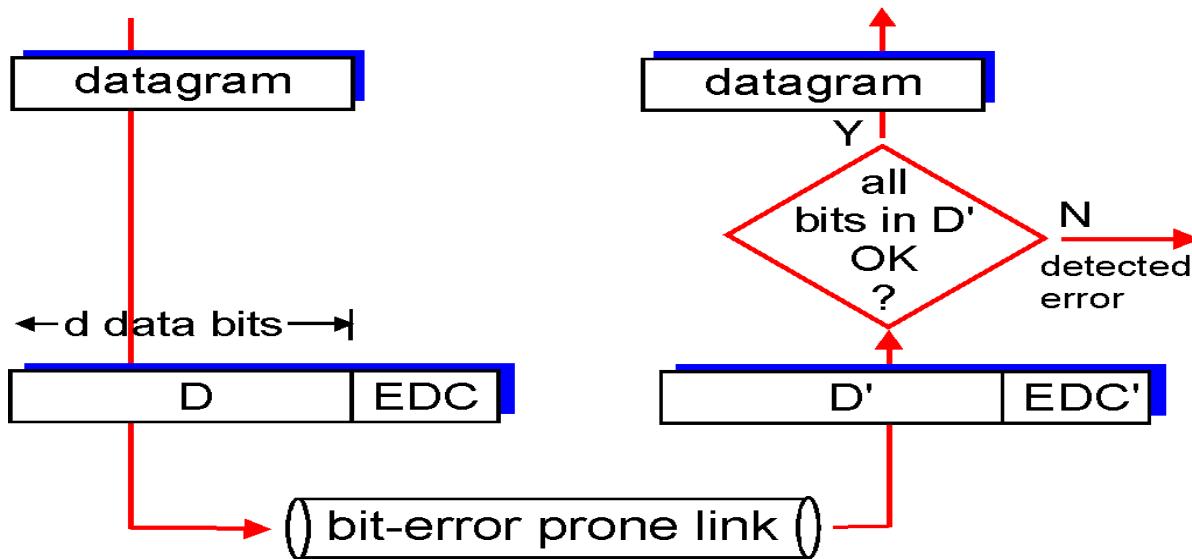
**Wie können eigentlich Fehler erkannt und
ggf. sogar behoben werden?**

Fehlererkennung

EDC = Error Detection and Correction bits (Redundante Information)

D = Daten, die durch EDC geschützt werden

- Durch Fehlererkennungs- und Korrektur-Mechanismen kann der Empfänger manchmal – aber nicht immer – erkennen, dass Bitfehler aufgetreten sind
- Höhere Redundanz hilft



Fehlererkennung auf Bit-Ebene

- Idee für die Verwendung von Redundanz
 - Daten werden in Form eines Codes erwartet
 - Ausnutzung der „**Distanz**“ zwischen gültigen Codewörtern, d.h. **nicht alle möglichen Bitkombinationen sind gültige Codewörter**
- **Hamming-Abstand**
 - Anzahl unterschiedlicher Bits zweier Codewörter c_1 und c_2 , d.h. Anzahl der 1-Bits von $c_1 \text{ XOR } c_2$.
 - Beispiel: $d(10001001, 10110001) = 3$
 - Hamming-Abstand D von vollständigem Code C :

$$D(C) := \min\{ d(c_1, c_2) \mid c_1, c_2 \in C; c_1 \neq c_2 \}$$

Fehlererkennung- und Korrektur

- Hamming-Abstand eines Code bestimmt die Fähigkeit des Codes, Fehler zu erkennen und zu beheben
 - **erkennen** von e -Bit-Fehlern: Hamming-Abstand $e+1$ notwendig
 - **beheben** von e -Bit-Fehlern: Hamming-Abstand $2e+1$ notwendig

Beispiele

- Fehlererkennender Code:
 - Code mit einem einzigen Paritätsbit
 - Hamming-Abstand des Codes = 2
(Änderung eines einzelnen Bits erfordert Änderung des Paritätsbits)
 - Erkennung eines 1-Bit-Fehler möglich
(allg.: Fehler mit ungerader Anzahl von Bits)
- Fehlerbehebender Code (vereinfacht): 00000 00000, 00000 11111,
11111 00000, 11111 11111
Hamming-Abstand → des Code = 5
 - Korrektur von 2-Bit-Fehlern möglich
Beispiel: 00000 00111 → 00000 1111
 - **FEC (Forward Error Correction)**

7 bits of data	(count of 1-bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

Hamming-Code

- Wie kann z.B. ein Code entwickelt werden, der automatisch alle 1-Bit Fehler korrigieren kann?
- Bisherige Erkenntnis: Hamming-Distanz muss mindestens $2e+1 = 3$ sein (mit $e=1$)
- Übertragung von m Datenbits
 r Prüfbits
 $n = m + r$ Gesamtbits
- Wenn n Bits übertragen werden, gibt es auch n Fehlermöglichkeiten mit einem Einzelbitfehler. D.h. jede der 2^m verschiedenen Datenwörter braucht jeweils $(n+1)$ Repräsentationen (n Fehler + eine fehlerfreie Version)
- Die Summe aller Repräsentationen muss in 2^n passen (Gesamtzahl aller darstellbarer Kombinationen).
- Abschätzung für r :
- Beispiel: $m=7 \rightarrow r=4$

$$(n+1) \cdot 2^m \leq 2^n$$

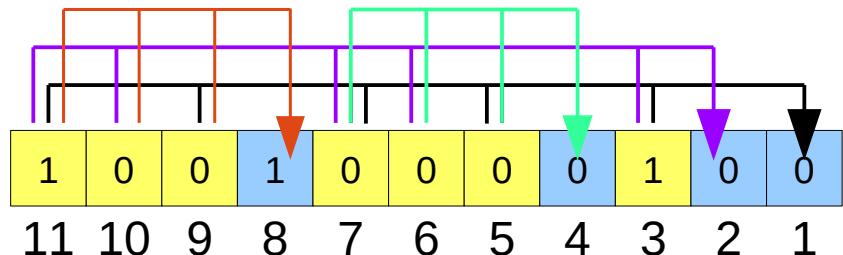
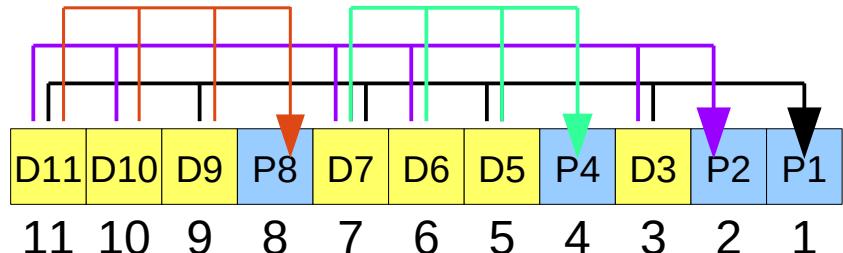
$$\text{mit } n=m+r$$

$$(m+r+1) \cdot 2^m \leq 2^m \cdot 2^r$$

$$(m+r+1) \leq 2^r$$

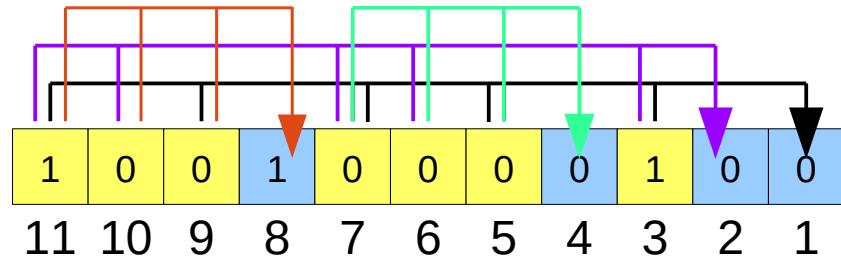
Hamming-Code (11/7)

- Mit dem Hamming Code kann das theoretische Minimum von r realisiert werden. (11/7) bedeutet: 11 Bit übertragen, 7 Datenbits
- Die Prüfbits befinden sich (ab 1 gezählt) an allen Positionen von 2-er Potenzen:
- Die Prüfbits geben das Parity-Bit (even) aller DatenBits an, die das entsprechende Prüfbit (als Dualzahl) gesetzt haben.
- Beispiel:
Übertragung von ASCII ,A': 0x41 = 1000001

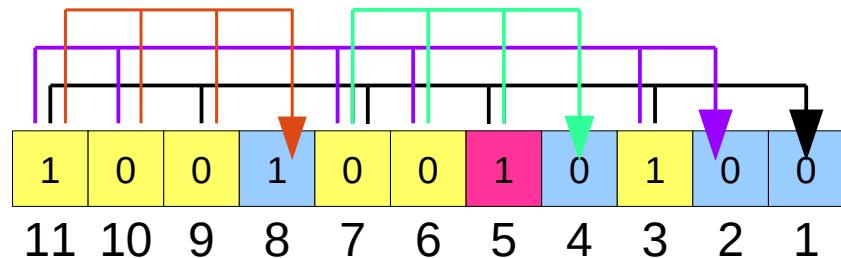


Hamming-Code (11/7)

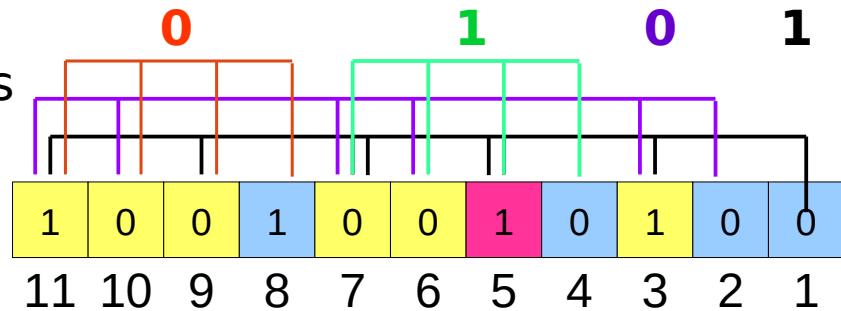
- Beispiel:
Übertragung von
ASCII ',A': $0x41 = 1000001$



- Fehler bei der Übertragung.



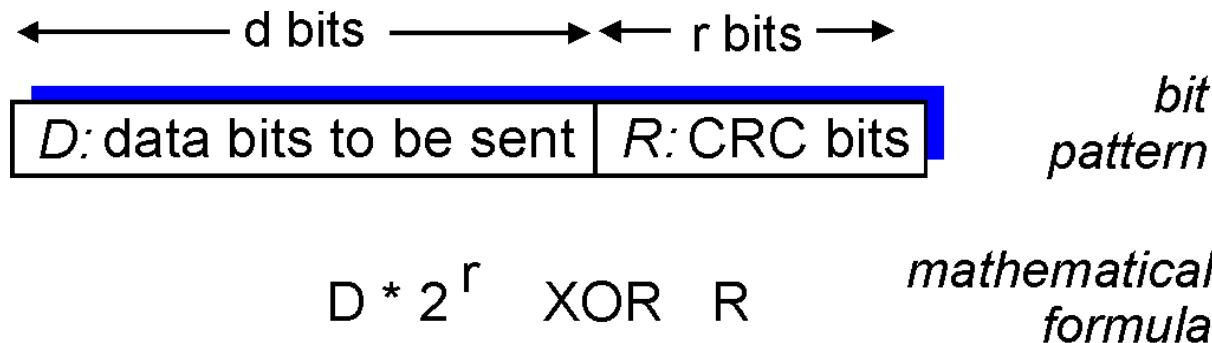
- Empfänger berechnet
Paritätsbits nochmal incl.
der übertragenen Paritätsbits
→ müsste immer 0 sein ...
- Hier $101 = 5 \rightarrow$
Fehler ist an Position 5



Eine Hardware-Lösung

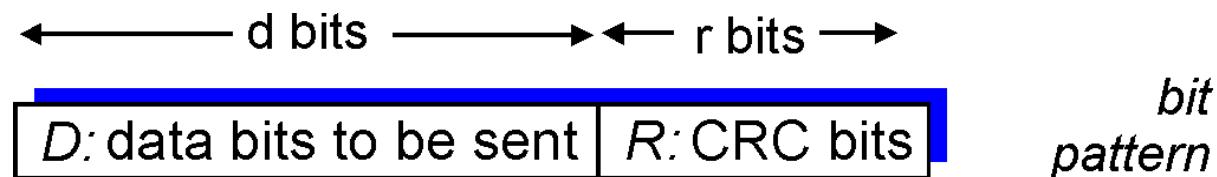
Cyclic Redundancy Check (CRC)

- Polynom-Codes: Interpretiere Datenbits D als Bitkette eines Polynoms, dessen Koeffizienten die 0-1-Werte der Bitkette sind
$$u^3 + u + 1 \rightarrow \begin{matrix} 1 & 0 & 1 & 1 \end{matrix}$$
- Prüfung basiert auf Polynom-Arithmetik



Prüfsummen: Cyclic Redundancy Check

- Sender und Empfänger einigen sich auf ein gemeinsam verwendetes Bitmuster der Länge $r+1$ Bit, das als **Generator G** bezeichnet wird. Das höchstwertige Bit hiervon ist 1
- Konzept: Berechne die r CRC-bits **R** so, dass die $d + r$ Bits (als Binärzahl interpretiert) mit der **Modulo-2-Arithmetik** genau durch G teilbar sind
 - Empfänger kennt G, teilt $\langle D, R \rangle$ durch G. Falls der Rest ungleich 0, so liegt ein Fehler vor!
 - Kann Burst-Fehler von weniger als $r+1$ Bits und jede ungerade Fehlerzahl erkennen

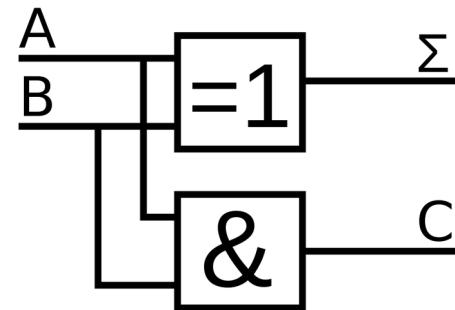


$$D * 2^r \text{ XOR } R$$

mathematical formula

Modulo-2 Arithmetik

- Die Rechenoperationen werden in der Modulo-2-Arithmetik einfacher, da hierbei keine Überträge zu berücksichtigen sind!
- Addition und Subtraktion führen so zu dem gleichen Ergebnis.
- Wir können einfach mit XOR arbeiten!
- Digitaltechnik, hier **Halbaddierer**: Eine Addition ist logisch ein XOR, das Carry ein UND



Wie kann R berechnet werden?

Es gibt eine Bitkombination n, so dass gilt:

$$(D \cdot 2^r) \text{ XOR } R = n G$$

D.h., R soll so gewählt werden, dass G in $D \cdot 2^r$ ohne Rest teilbar ist

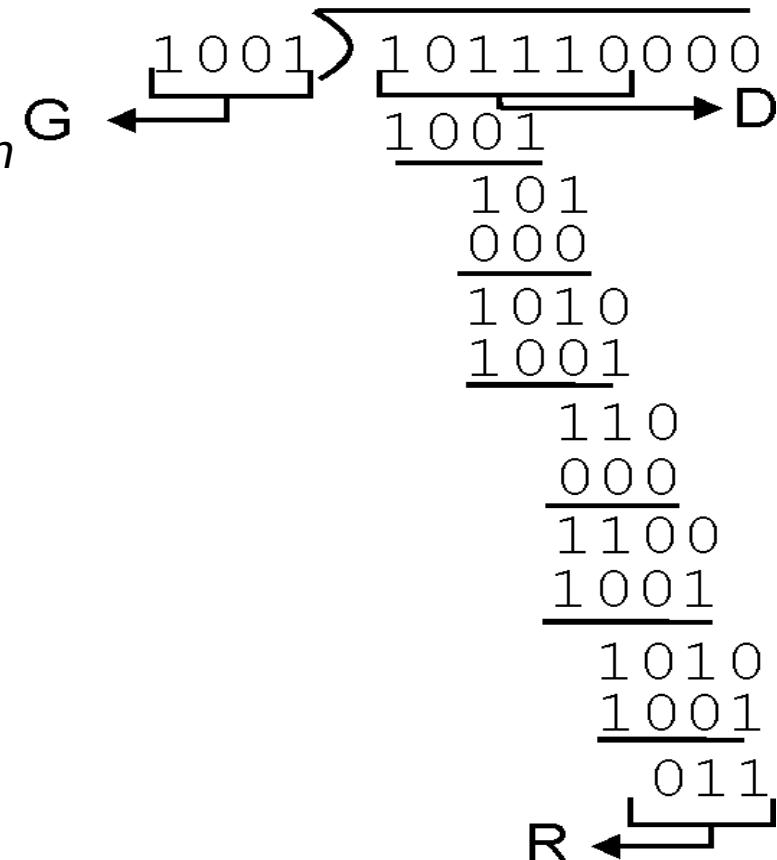
\leftrightarrow

$$D \cdot 2^r = nG \text{ XOR } R$$

\leftrightarrow

Damit kann man R berechnen, denn
wenn man **D · 2^r durch G teilt, ist der
Rest des Wertes genau R**

$$R = \text{Rest} \left[\frac{D \cdot 2^r}{G} \right]$$



Aufgabe

- Wir betrachten das CRC-Verfahren am Beispiel des Generatorpolynoms $x^4 + x^3 + 1$. Berechnen Sie die CRC-Prüfsumme zur Bitfolge 10110101110

Wir berechnen 101101011100000:11001

101101011100000

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

11001

011111011100000

11001

001100111000000

11001

0000011000

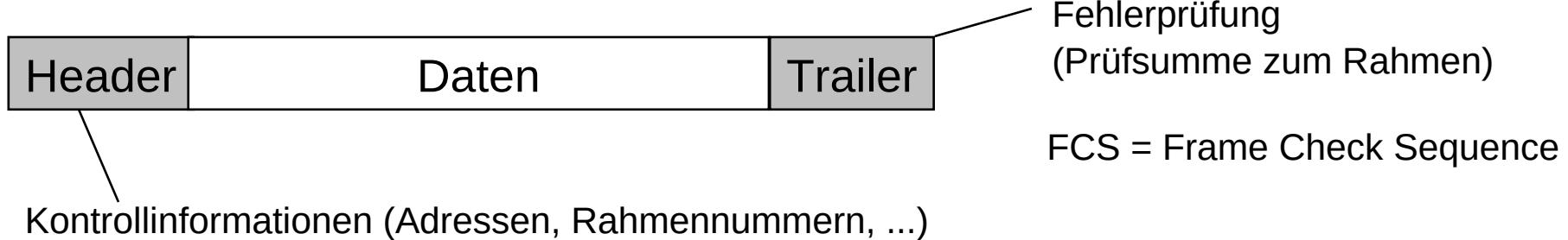
11001

0000100

R ist somit 0100, gesendet wird 101101011100100

Rahmenerstellung in der Sicherungsschicht

- Einteilung einer Nachricht in einheitliche Einheiten (einfachere Übertragung)
- wohldefinierte Schnittstelle nach oben (Schicht 3)
- Kennzeichnung der Einheiten:



Fehlererkennung: **ARQ** (Automatic Repeat Request)

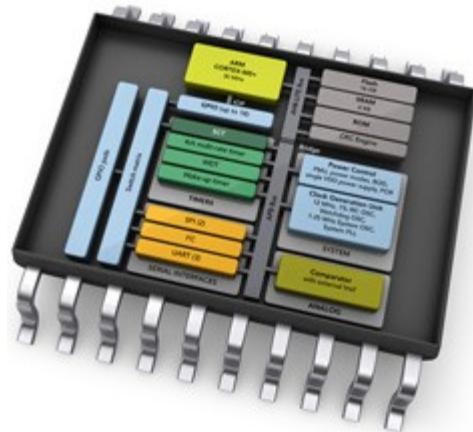
- Verwendung eines fehlererkennenden Codes (CRC)
- Fehler werden erkannt, können aber nicht korrigiert werden! Daher müssen verfälschte Daten neu angefordert werden.
- Einführung einer **Flusskontrolle** (wie bei TCP: Sliding Window):
 - Nummerierung der zu sendenden Datenblöcke
 - Quittierung von Blöcken durch den Empfänger
 - Wiederholung fehlerhaft übertragener Blöcke

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



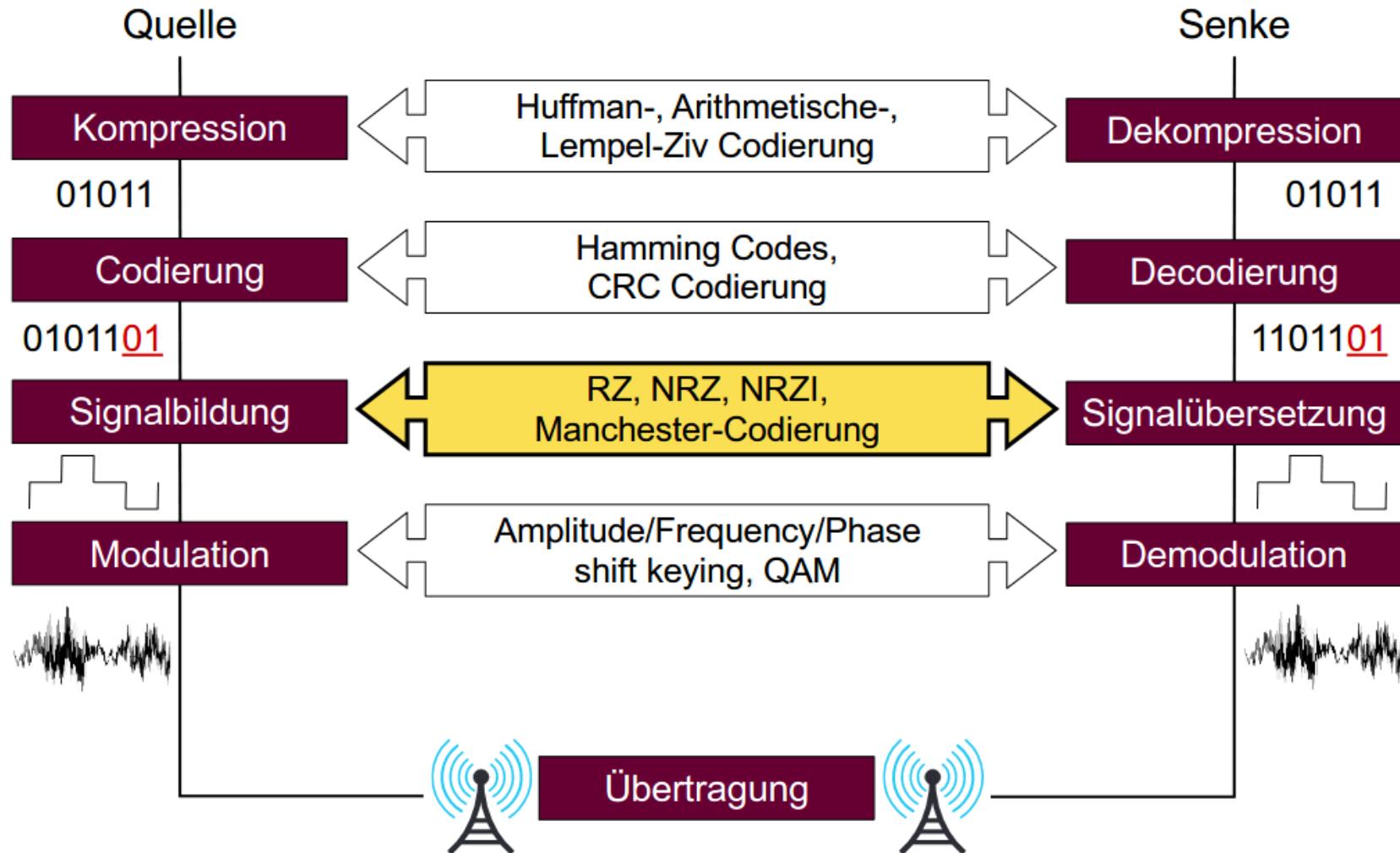
Bisher:

- Bitübertragungsschicht:
 - Physikalisches Medium
- Sicherungsschicht:
 - Zugriff aufs Medium bei Mehrfachzugriff
 - Sicherungsverfahren

Heute:

- Bitübertragungsschicht:
 - **Leitungskodierung**

Übersicht über die Kodierungsverfahren (in unterschiedlichen Schichten)



Kodierung von Informationen

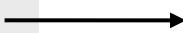
Shannon:

„Das grundlegende Problem der Kommunikation besteht darin, an einer Stelle genau oder angenähert eine Nachricht wiederzugeben, die an einer anderen Stelle ausgewählt worden ist.“

Ziel: sinnvolle Darstellung der zu übertragenden Information

Kodierungsverfahren:

- **(Kanal)-Codierung**
(Schicht 2 und 4)
- **Signalbildung**
(Leitungskodierung,
Schicht 1)



Darstellung der zu übertragenden Daten in Codewörtern, die den Eigenschaften des Übertragungskanals angepasst sind (Redundanz).
→ **Sicherung gegen Übertragungsfehler** durch fehlererkennende bzw. -korrigierende Codes

→ **Physikalische Darstellung von Digitalsignalen**

Basis- und Breitband

Die Übertragung von Informationen kann entweder auf dem **Basisband** oder auf **Breitband** erfolgen. Dies bedeutet:

Basisband

Das Basisband ist der natürliche Frequenzbereich des Nutzsignals (untere Grenzfrequenz f_{\min} gleich oder nahe bei 0 Hz). Die digitalen Informationen werden ‚direkt‘ in physikalische Größen übersetzt und so über die Leitung übertragen. Hierzu sind Kodierungsverfahren notwendig, die festlegen, wie bei der Übertragung eine 0 bzw. eine 1 repräsentiert werden. Es kann nur je ein Signal übertragen werden

Breitband

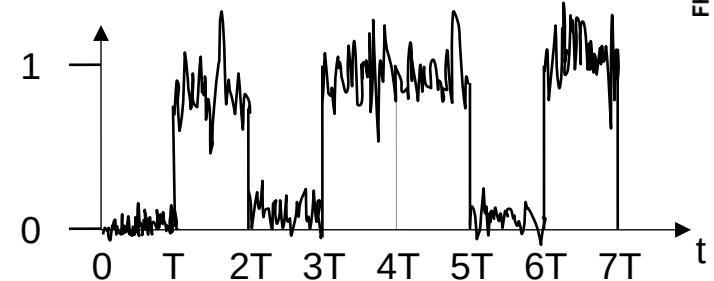
Die digitalen Nutzdaten werden nicht direkt übertragen, sondern einem oder mehreren hochfrequenten Trägern aufmoduliert. Durch die Verwendung verschiedener Trägerwellen (Frequenzen) können dann mehrere Informationen gleichzeitig übermittelt werden

Erst durch optische Übertragungstechnik und Funknetze wurde die Breitbandtechnik benötigt und verbreitet

Leitungscodes im Basisband: Anforderungen

Wie sollen digitale Signale z.B. auf Kupferkabel elektrisch repräsentiert werden?

- Möglichst hohe **Widerstandsfähigkeit gegen Dämpfung**



- Effizienz:** möglichst hohe Übertragungsraten durch Codewörter

binärer Code: +5V / -5V ?

ternärer Code: +5 V / 0V / -5V ?

quaternärer Code: 4 Zustände (Codierung von 2 Bit gleichzeitig)

- Taktrückgewinnung** beim Empfänger (**Synchronisation**), dazu möglichst häufige/regelmäßige Pegelwechsel
- Gleichstromfreiheit:** positive und negative Signale treten ungefähr gleich oft auf → kein nennenswerter elektrischer Gleichstrom-Fluss
- Robustheit:** Können längere Sequenzen von 0 und 1 noch als solche noch erkannt werden? Können fehlerhafte Bits erkannt werden?

Baud-Rate vs. Bit-Rate

- Wenn die Zeitdauer (Schrittdauer) eines **Symbols** bzw. **Codeelements** T ist, ist die Schrittgeschwindigkeit

$$v_s = \frac{1}{T} \quad (\text{Einheit: } \mathbf{Baud}), \text{ Symbolrate}$$

- Die Übertragungsgeschwindigkeit (äquivalente Bitrate) ist dann

$$v_u = v_s \cdot \log n \quad (n = \text{Anzahl diskreter Zustände des Codeelements})$$

Bei binären Codeelementen stimmen somit **Bitrate** und **Baudrate** (Schrittgeschwindigkeit) überein, falls nur Codeelemente für Daten übermittelt werden (es gibt auch Codeelemente für z.B. die Rahmenstruktur)

Leitungscodes

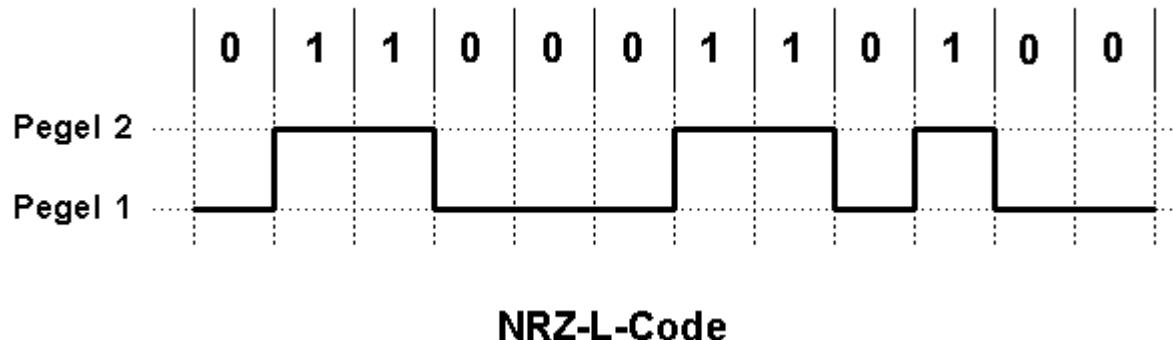
■ NRZ / NRZ-L: Non-Return-to-Zero

Kein automatisches Zurückfallen auf einen Grundpegel.

Hier z.B.:

0 = negative Spannung (konstant 0V), Pegel 1

1 = positive Spannung (konstant +5V), Pegel 2



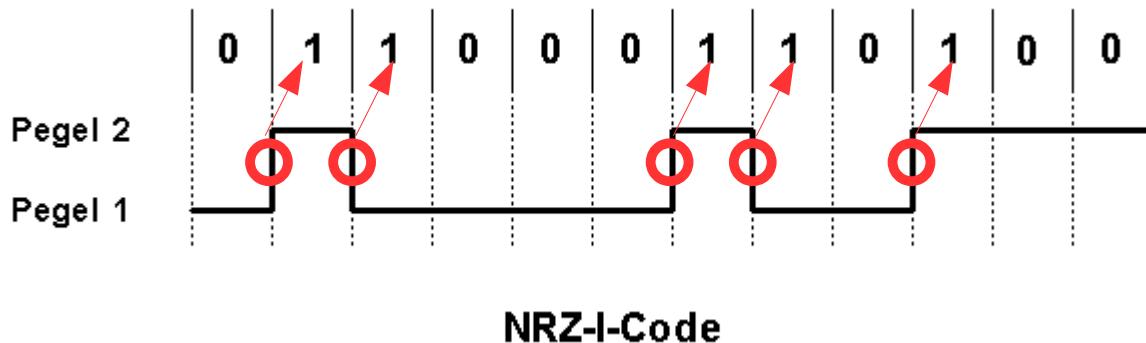
Nachteil: bei langen 0- oder 1-Folgen Taktverlust und keine Gleichstromfreiheit

Beispiel: UART, RS232 (serielle Schnittstellen)

Leitungscodes

■ NRZ-I: NRZ-Inverted-Code

- 0 = kein Polaritätswechsel
1 = Polaritätswechsel



Vorteil: Leitungen können vertauscht werden, Polarität egal

Nachteil: Taktverlust und keine Gleichstromfreiheit bei langen 0-Folgen

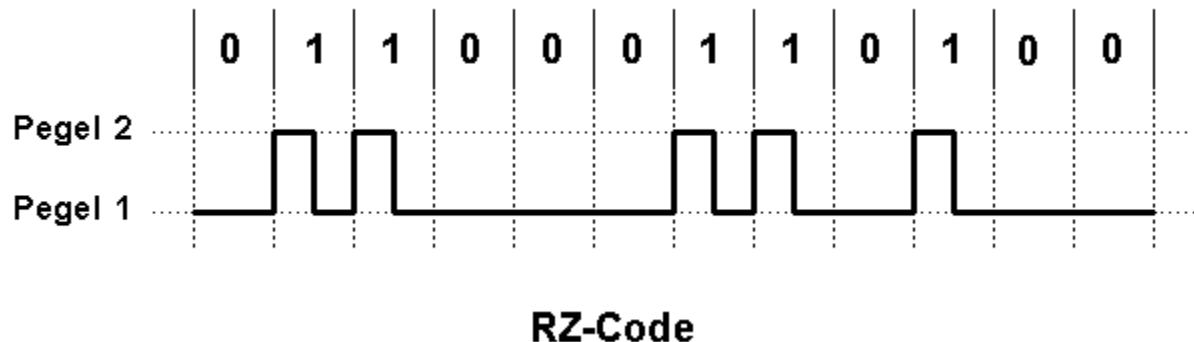
Beispiel: 100Base-FX, USB, CD-ROM, Festplatten

Leitungscodes

■ RZ: Return to Zero (hier unipolar)

0 = 0V

1 = T/2 lang 1, T/2 lang 0



Vorteil: Taktrückgewinnung bei 1-Folgen

Nachteil: Keine Gleichstromfreiheit, kein Takt bei langen 0-Folgen

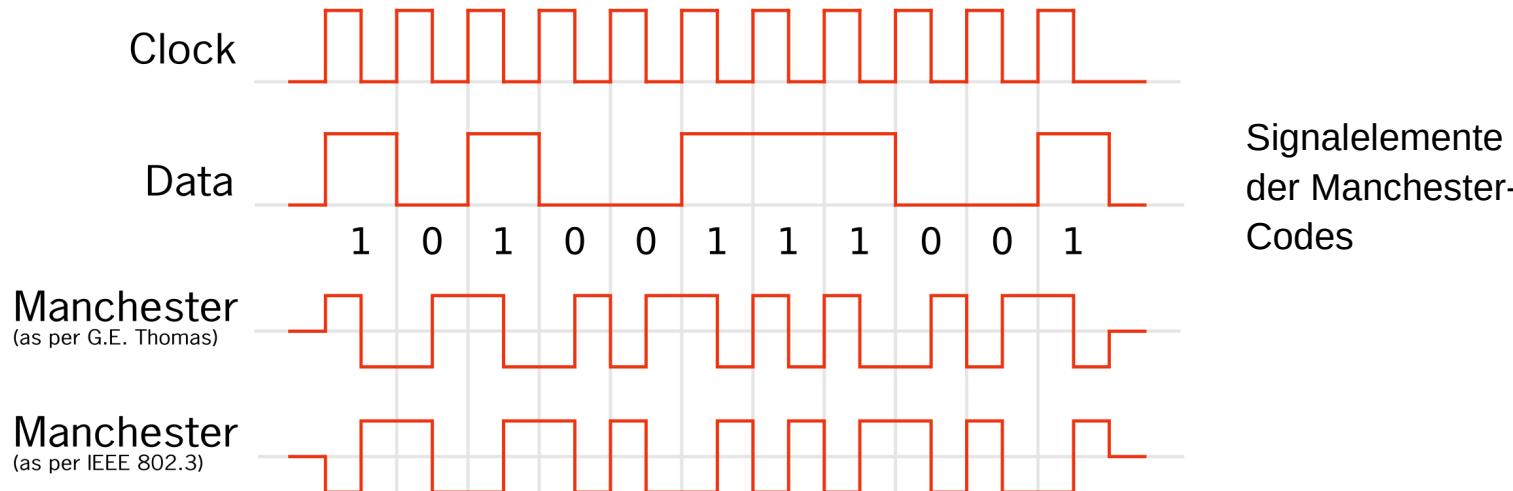
Beispiel: IrDA - Fernbedienung

Leitungscodes

Manchester-Code (1B/2B-Code)

Lange Folgen gleicher Signale werden durch einen Pegelwechsel in der Mitte jedes Bits verhindert. Nach G.E. Thomas:

- 0 = Polaritätswechsel von negativ (-5V) nach positiv (+5V)
- 1 = Polaritätswechsel von positiv (+5V) nach negativ (-5V)



Vorteile: Gleichstromfrei, Taktrückgewinnung möglich

Nachteil: Doppelte Bandbreite im Vergleich zu NRZ, **Bitrate = Baudrate/2**

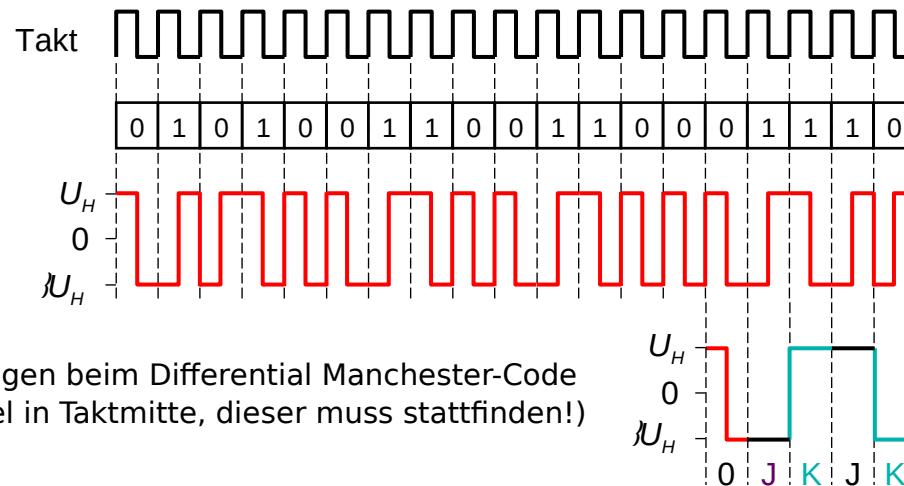
Beispiel: 10Base2

Leitungscodes

Differentieller Manchester-Code

Wie beim Manchester-Code erfolgt ein Pegelwechsel in der Bitmitte, zusätzlich allerdings:

- 0 = Pegelwechsel am Anfang des Symbols
- 1 = kein Pegelwechsel am Anfang des Symbols



Vorteile: Gleichstromfrei, Taktrückgewinnung möglich

Nachteil: Doppelte Bandbreite im Vergleich zu NRZ

Beispiel: Token Ring, Magnetstreifen

4B/5B-Code

Nachteil des Manchester-Codes:

- 50% Effizienz, d.h. **1B/2B-Code** (ein Bit wird auf zwei *Symbole* kodiert)

Eine Verbesserung stellt der **4B/5B-Code** dar:

- vier Bit werden in fünf *Symbole* kodiert: 80% Effizienz

Arbeitsweise:

- Pegelwechsel bei 1, kein Pegelwechsel bei 0 (Differentieller NRZ-Code)
- Kodierung von hexadezimalen Zeichen: 0, 1, ..., 9, A, B, ..., F (4 Bit) in 5 Bit, so dass lange Nullenblöcke vermieden werden.
- Auswahl der günstigsten 16 der möglichen 32 Codewörter (maximal 3 Nullen in Folge)
- Weitere 5 Bit-Kombinationen für Steuerinformationen
- Erweiterbar auf 1000B/1001B-Codes?

Beispiel für eine 4B/5B-Codetabelle

Decimal Value	Data (4Bits)	Transmitted Symbols	Symbol Assignment	
0	0000	00000	Quiet -line state	(status)
1	0001	00001	Invalid	
2	0010	00010	Invalid	
3	0011	00011	Invalid	
4	0100	00100	Halt -line state	(status)
5	0101	00101	Invalid	
6	0110	00110	Invalid	
7	0111	00111	R-Reset (logical 0)-control	(control)
8	1000	01000	Invalid	
9	1001	01001	Data	
10	1010	01010	Data	
11	1011	01011	Data	
12	1100	01100	Invalid	
13	1101	01101	T-Ending delimiter	(control)
14	1110	01110	Data	
15	1111	01111	Data	
16		10000	Invalid	
17		10001	K-starting delimiter	(control)
18		10010	Data	
19		10011	Data	
20		10100	Data	
21		10101	Data	
22		10110	Data	
23		10111	Data	
24		11000	J-starting delimiter	(control)
25		11001	S - set (logical 1) - control	(control)
26		11010	Data	
27		11011	Data	
28		11100	Data	
29		11101	Data	
30		11110	Data	
31		11111	Idle-line state	(status)

Worst case:
11100|01110
<→
3 Nullen

Breitbandkommunikation

Bei der Breitbandkommunikation wird das **digitale** Signal auf ein **analoges** Trägersignal **aufmoduliert**

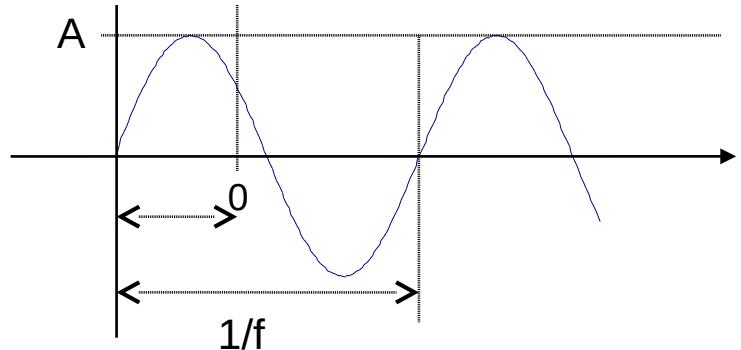
Daten werden physikalisch als elektromagnetische Wellen dargestellt:

$$s(t) = A \cdot \sin(2 \cdot \pi f t + \varphi)$$

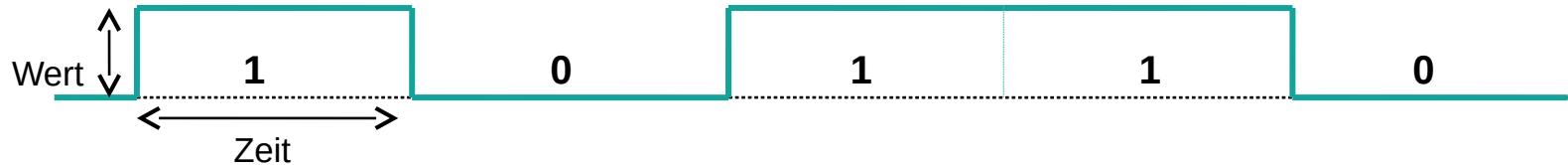
A: Amplitude

f: Frequenz

φ : Phase



Modulation digitaler Signale

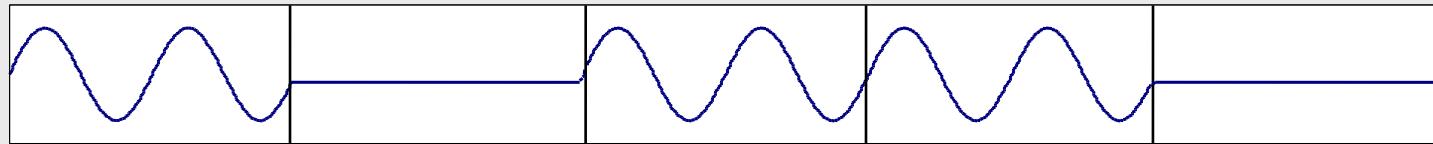


Die Umwandlung der digitalen Signale kann auf verschiedene Arten erfolgen, basierend auf den Parametern einer analogen Welle:

$$s(t) = A \cdot \sin(2 \cdot \pi f t + \phi)$$

Amplitude Frequenz Phase

Amplitudenmodulation (Amplitude Shift Keying, ASK)



- Bei einer 1 wird eine große Amplitude ausgestrahlt, bei einer 0 eine kleine
- Technisch einfach zu realisieren, wird üblicherweise bei optischer Übertragung verwendet
- Störanfällig wegen Abschwächung des Signals

Modulation digitaler Signale

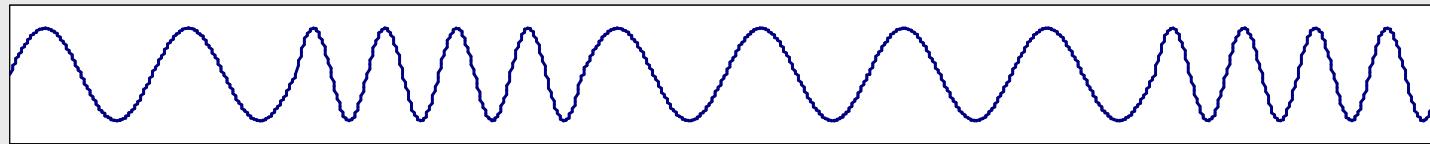


Die Umwandlung der digitalen Signale kann auf verschiedene Arten erfolgen, basierend auf den Parametern einer analogen Welle:

$$s(t) = A \cdot \sin(2 \cdot \pi f t + \phi)$$

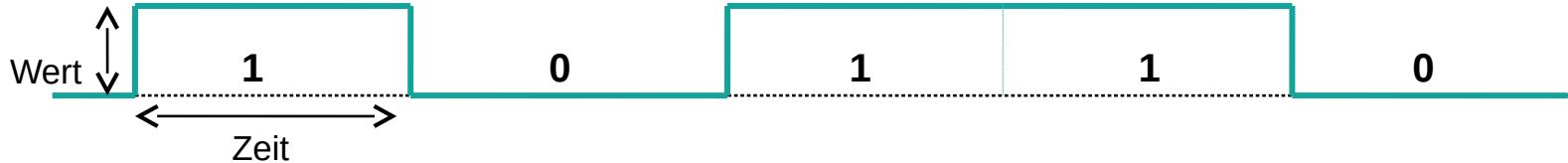
Amplitude Frequenz Phase

Frequenzmodulation (Frequency Shift Keying, FSK)



- Für 0 und 1 werden verschiedene Frequenzen ausgestrahlt
- Verschwenderischer Umgang mit Frequenzen
- Für Telefonübertragung

Modulation digitaler Signale

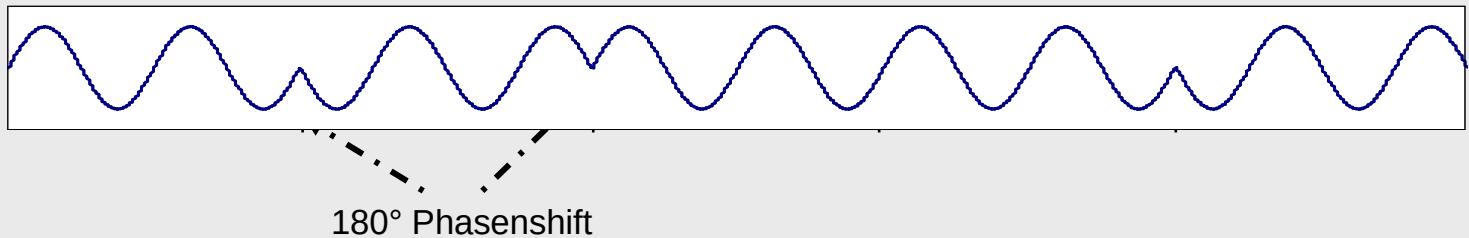


Die Umwandlung der digitalen Signale kann auf verschiedene Arten erfolgen, basierend auf den Parametern einer analogen Welle:

$$s(t) = A \cdot \sin(2 \cdot \pi f t + \phi)$$

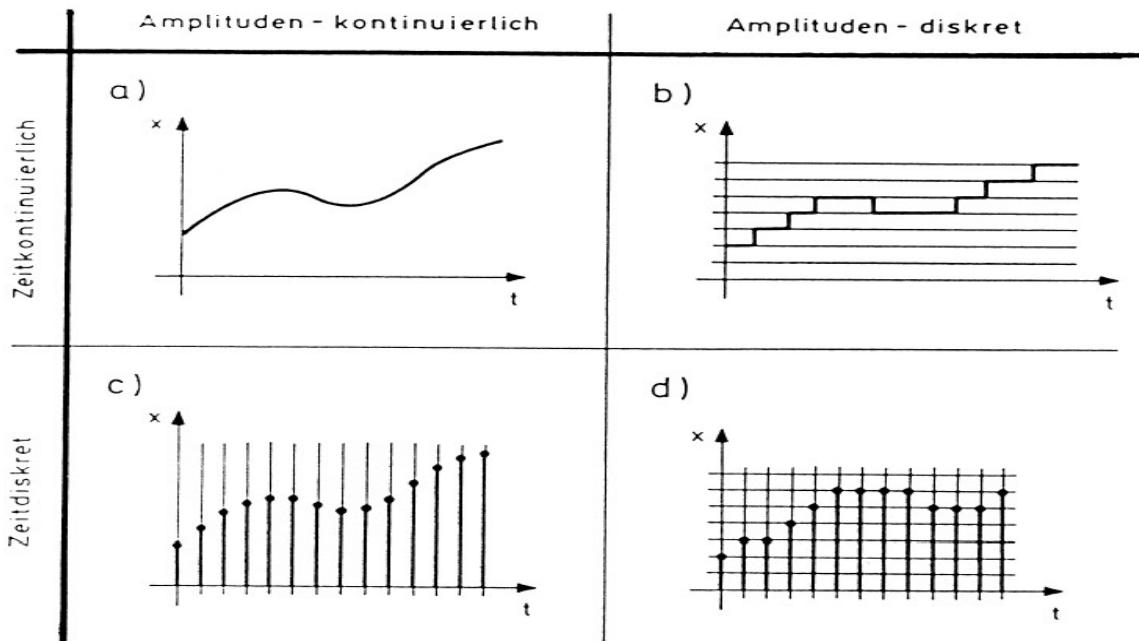
Amplitude Frequenz Phase

Phasenmodulation (Phase Shift Keying, PSK)



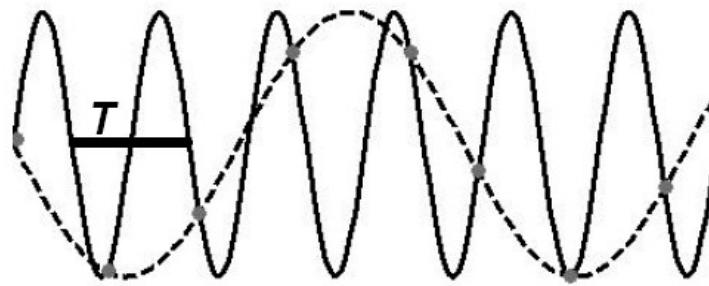
- Verwendung unterschiedlicher Phasenlagen für 0 und 1
- Komplexe Demodulation
- Störsicher
- In der drahtlosen Kommunikation oft bevorzugt

Abtastvorgang / Digitalisierung



- Quantisierung in der Zeit
 - Signal wird regelmäßig abgetastet
- Quantisierung der Amplitude
 - Signal wird mit diskreten Amplitudenwerten abgetastet

Abtastvorgang / Digitalisierung

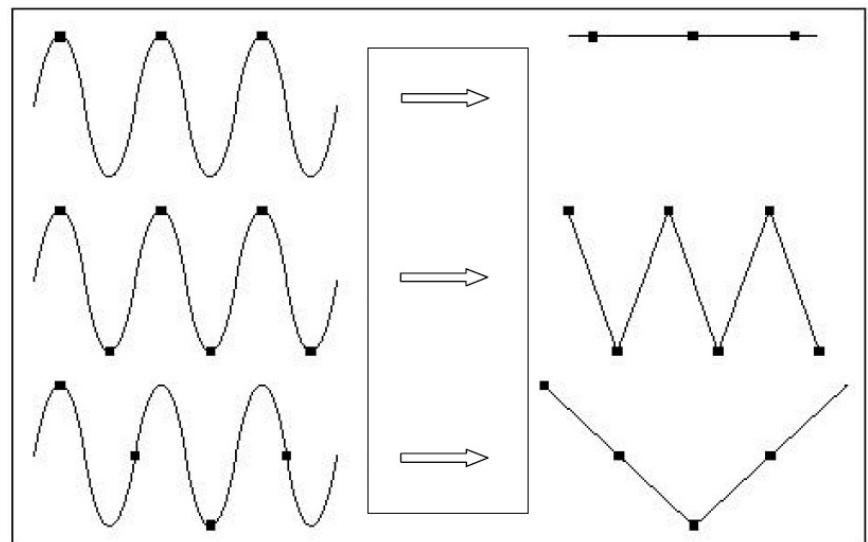


Abtastung: falsch



Abtastung: richtig

- Abtastintervall kürzer als halbe Periodendauer
- sonst: Signal nicht eindeutig charakterisiert



„Fehlerdeutungen“

Welche Übertragungsraten sind erreichbar?

Das Abtasttheorem von Nyquist

Aus einer Folge diskreter Werte kann das analoge Ausgangssignal rekonstruiert werden, wenn die Abtastrate mindestens das **Doppelte der oberen Grenzfrequenz** des ursprünglichen Analogsignals beträgt.

- Damit bestimmt die **Differenz** zwischen der höchsten und niedrigsten **Frequenz** (die *Bandbreite*) die **Abtastrate**
- **Pro Abtastvorgang** kann nur eine bestimmte Informationsmenge ***I*** (in **Bits!**) gewonnen werden
- Damit ergibt sich als Übertragungsrate in einem **beliebigen Breitbandkanal** der Bandbreite f eine maximale Übertragungskapazität von ***2 f I***

Schlussfolgerungen

- Die Bandbreite ist proportional zur erreichbaren Datenrate
 - möglichst breite Frequenzbänder nutzen! Je höher die Frequenz, desto einfacher ist dies
- Die in einem Abtastvorgang modulierten binäre Informationen sind proportional zur erzielbaren Bandbreite
 - Hat ein Signal V diskrete Niveaus, so ist $\log_2(V)$ die modulierte binäre Information
 - möglichst viel Information pro Abtastvorgang zu erhalten (also V zu erhöhen!)
- Höhere Frequenzen können eher breite Frequenzbänder aufnehmen

Begrenzung der Übertragung durch Rauschen

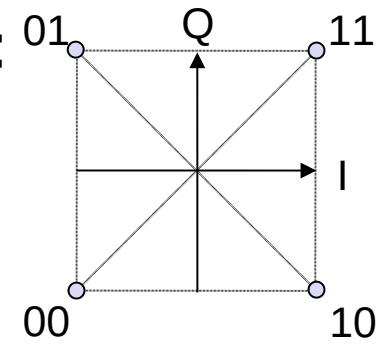
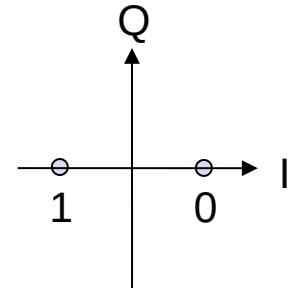
- Die in einem Abtastvorgang modulierbare Information hängt von einem **ausreichenden Signal-Rauschabstand** ab. Dieser kann zwar durch eine erhöhte Sendeleistung verstärkt werden, jedoch ist die **Leistungsflussdichte** selbst im freien Raum umgekehrt proportional zum Quadrat der Entfernung

Dies bedeutet, dass sie bei Verdoppelung der Distanz bereits auf ein Viertel, bei **Verzehnfachung des Abstandes sogar auf ein Hundertstel des Ausgangswertes** abfällt.
- Nach Shannon ist die Bandbreite begrenzt durch den Faktor $\log_2(1+\text{Signalrauschabstand})$

Beispiel: Eine Telefonleitung mit 30 db Rauschabstand und einer Bandbreite von 3 KHz kann maximal 30000 Bit/s übertragen
- Höhere Frequenzen verhalten sich ähnlich wie Licht. Damit wird dann eine Sichtverbindung benötigt

Fortgeschrittene Phasenmodulation

- **BPSK** (Binary Phase Shift Keying):
 - = einfaches PSK
 - > Bitwert 0: Sinuswelle
 - > Bitwert 1: invertierte Sinuswelle
 - Niedrige Datenraten
 - Robuste Übertragung
 - Auch oft als differentieller Code (DBPSK)
- **QPSK** (Quaternary Phase Shift Keying):
 - Zwei Bit werden gemeinsam codiert
 - Vier unterschiedliche Phasenlagen
 - Doppelte Datenraten verglichen mit BPSK

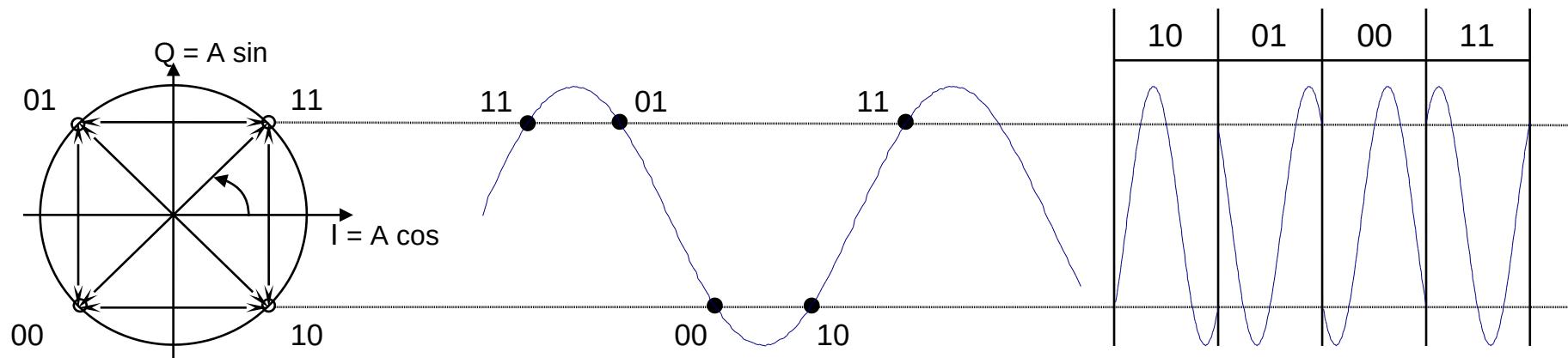


Fortgeschrittene PSK-Verfahren

Die Umtastung kann auch mehr als 2 Phasen umfassen: es kann zwischen M verschiedenen Phasen umgetastet werden, wobei M eine Zweierpotenz sein muss. Dadurch können mehr Informationen gleichzeitig übermittelt werden.

Beispiel: **QPSK** (Quaternary Phase Shift Keying)

- Umtastung zwischen 4 Phasen
- 4 Phasen erlauben 4 Zustände: **kodiere 2 Bit auf einmal**
- Damit doppelte Übertragungsrate



A = Amplitude des Signals

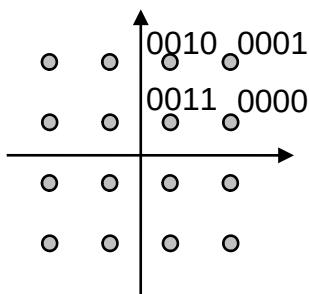
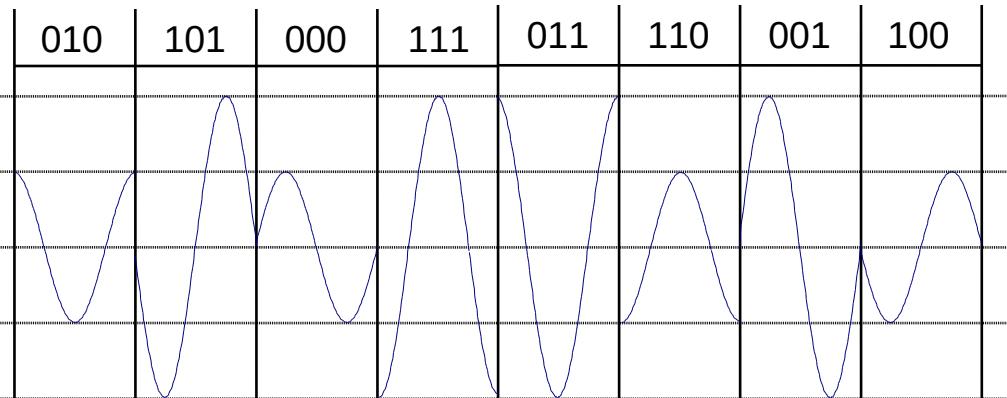
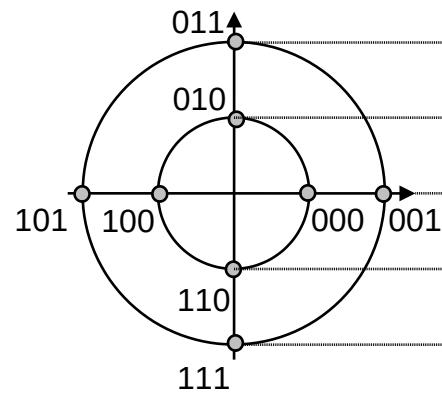
I = In-Phase, Signalkomponente (in Phase mit Trägersignal)

Q = Quadratur-Phase, Quadraturkomponente (senkrecht zur Trägerphase)

Fortgeschrittene PSK-Verfahren

Quadraturamplitudenmodulation (Quadrature Amplitude Modulation, **QAM**)

- Kombination aus ASK und QPSK
- n Bit können gleichzeitig übertragen werden ($n=2$ ist QPSK)
- Bitfehlerrate steigt mit zunehmendem n , aber weniger als bei vergleichbaren PSK-Verfahren



16-QAM: 4 Bit pro Signal:

- 0011 und 0001 haben gleiche Phase, aber unterschiedliche Amplitude
- 0000 und 0010 haben gleiche Amplitude, aber unterschiedliche Phase

WLAN: Signalausbreitung bei Funkwellen

- In freiem Raum geradlinige Ausbreitung
- die **Empfangsleistung ist proportional zu $1/d$** (d = Abstand zwischen Sender und Empfänger, $\gamma = 2$ in freiem Raum, 2.7-5 in der Stadt, 4-6 im Gebäude)
- Werte für γ größer als 2 ergeben sich durch Dämpfung, Reflektion, Streuung, Beugung, ...

Basierend auf der Empfangsleistung ergeben sich verschiedene Bereiche:

Übertragungsbereich

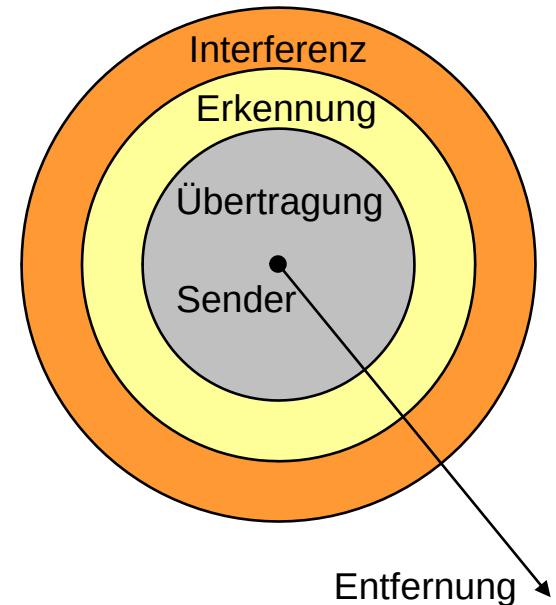
- Kommunikation möglich
- Akzeptable Fehlerrate

Erkennungsbereich

- Signalerkennung ist noch möglich
- Zu hohe Fehlerrate für Identifikation des Signals

Interferenzbereich

- Keine Signalerkennung
- Störung anderer Übertragungen



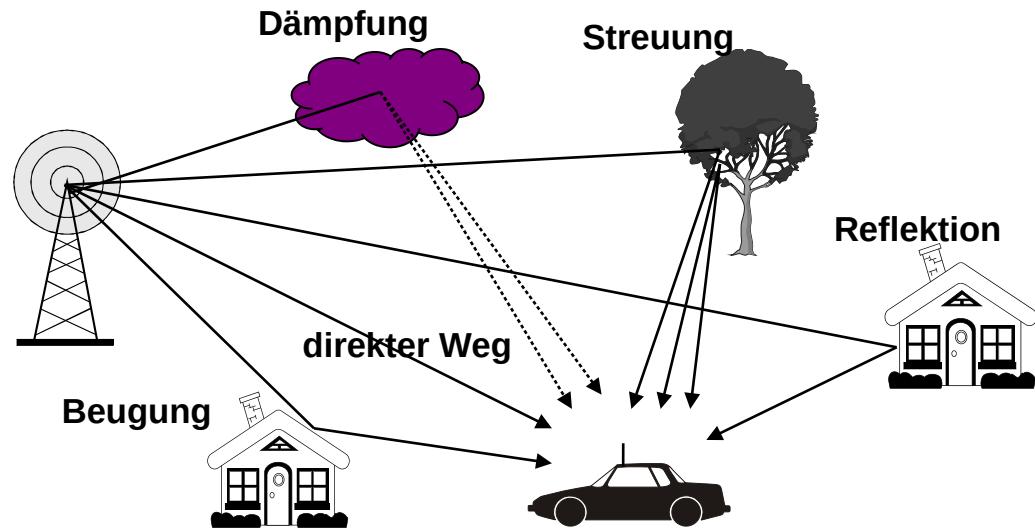
Mehrwegausbreitung

Faktoren, die die Ausbreitung beeinflussen:

- natürliche Umgebung: Gebirge, Wasser, Vegetation, Regen, Schnee
- künstliche Umgebung: Gebäude etc.

Ausbreitungsmechanismus:

- Dämpfung, Abschattung (Regen, Vegetation)
- Beugung/Ablenkung: scharfe und runde Kanten
- Reflektion an großen Flächen
- Streuung an kleinen Hindernissen
- Brechung: Brechungsindex wird mit zunehmender Höhe kleiner



- Wirkung:**
- der Empfänger erhält ein Signal auf mehreren Wegen zugleich
 - Abschwächung der Sendeleistung bei Reflektion, Beugung, ...
 - dasselbe Signal wird mit verschiedenen Phasenlagen empfangen, Interferenz findet statt
 - Das Signal wird **zeitlich gestreut**, Interferenz mit Nachbarsymbolen

Konsequenz

Bedingt durch diese Effekte benötigen drahtlose Kommunikationsnetze ausgeklügelte Mechanismen zur Fehlerkorrektur. Dies bedeutet jedoch, dass hierfür Bandbreite zur Verfügung gestellt werden muss. Der **Kommunikations-Overhead liegt damit erheblich über dem kabelgebundener Übertragung!**

Daumenregel:

Protokolle zur zuverlässigen Datenübertragung können weniger als 2/3 der verfügbaren drahtlosen Übertragungsrate tatsächlich nutzen

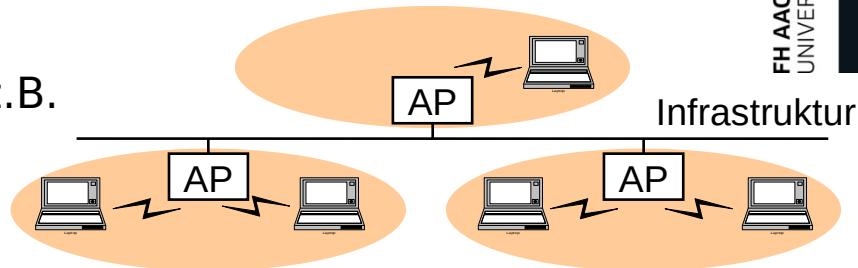
WLAN: Drahtloses Ethernet

- Drahtloses Äquivalent zu Ethernet: "Wireless LAN" (WLAN)
- Ausschließlich datenorientierte, breitbandige Internetzugangslösung
- Standardisiert von der IEEE als IEEE 802.11
 - 1997: IEEE 802.11 (Bandbreiten von maximal 2 Mb/s)
 - IEEE 802.11a mit 54 Mb/s durch Verwendung eines (störanfälligeren) Frequenzbandes (5 GHz) mit größerer Kapazität
 - 1999: IEEE 802.11b (Brutto-Datenrate von 11 MB/s bei einem Nutzdatenanteil von bis zu 6 Mb/s)
 - IEEE 802.11g: höhere Datenraten von bis zu 54 Mb/s im 2.4 GHz-Band
 - 802.11.n: MIMO-Technik erlaubt mehrere Kanäle bis zu 600 Mb/s. Optionale Nutzung des 5 GHz-Bandes

Aufbau eines WLAN

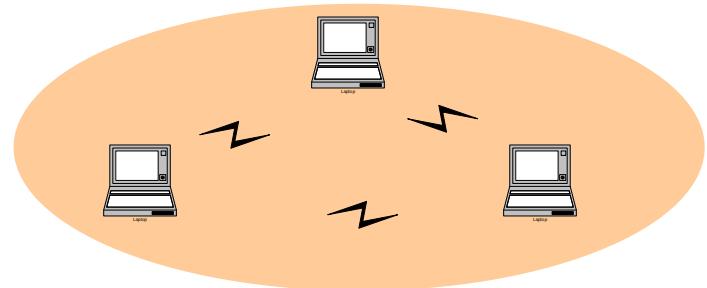
1. Netze mit fester Infrastruktur

- Infrastruktur bedeutet: feststehendes Netz, z.B. Ethernet oder Satellitenstrecken
- Zentraler Access Point (AP), drahtlose Geräte kommunizieren nur mit dem AP
- Kontrollfunktionalitäten (Medienzugriff, Mobilitätsmanagement, Authentisierung, ...) sind in der Infrastruktur realisiert
- Komplexität liegt in den Infrastrukturokomponenten, drahtlose Geräte brauchen nur ein Minimum an Funktionalität zu realisieren



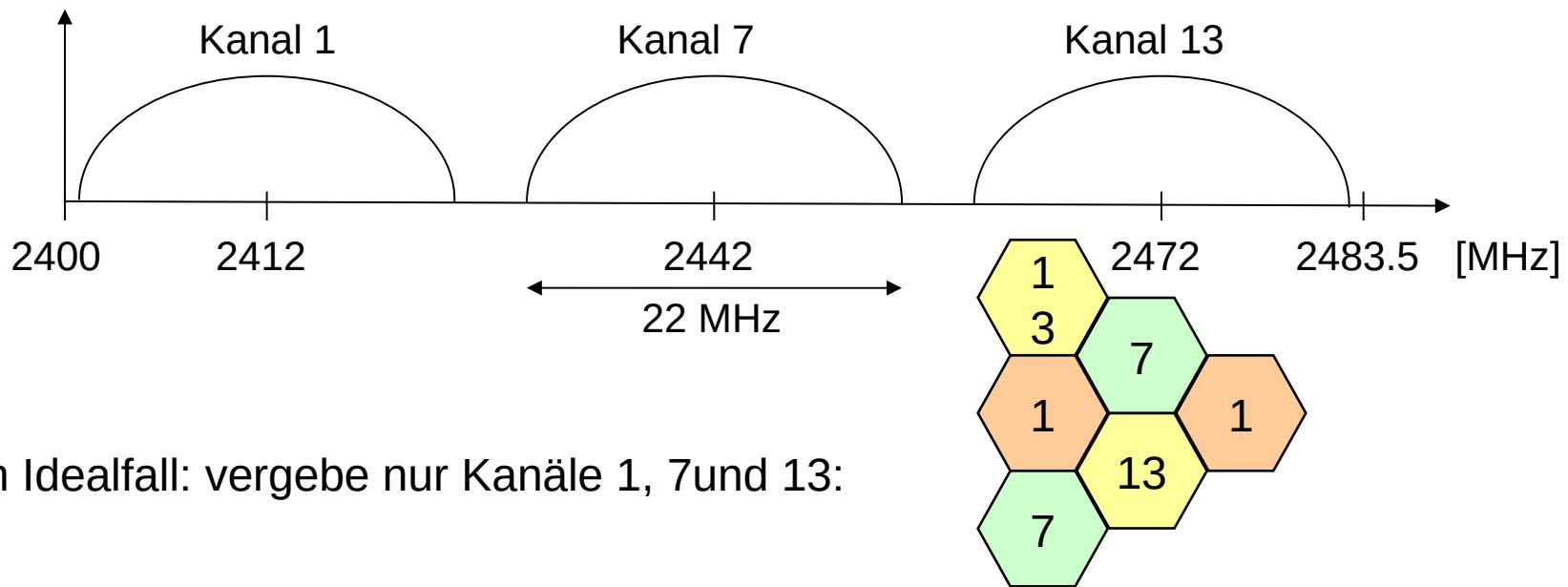
2. Ad-hoc-Netze

- Keine Infrastruktur – die drahtlosen Geräte kommunizieren direkt miteinander
- Höhere Komplexität der Geräte, da jedes Gerät alle Zugriffs- und Kontrollmechanismen implementieren muss



Kanäle bei IEEE 802.11b (g und n haben andere Breiten)

- Würden alle APs auf der gleichen Frequenz senden, würden in den Überlappungsbereichen Störungen auftreten.
- Daher: Aufteilung des gesamten Frequenzbereichs in Kanäle
- Kanäle belegen nie genau eine Frequenz, sondern „streuen“ auf die benachbarten Frequenzen. In IEEE 802.11b sind die Kanäle je 22 MHz breit
- 13 Kanäle in Deutschland (2412, 2417, 2422, ..., 2472 MHz), 11 in USA/Kanada
- Kanäle überlappen! Nicht-überlappende Kanalwahl:



- Im Idealfall: vergabe nur Kanäle 1, 7 und 13:

WLAN: IEEE 802.11

- Datenraten
 - 1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54, ... MBit/s Bruttodatenrate
 - Abhängig von Signalqualität wird die bestmögliche Datenrate gewählt
 - Nutzdatenrate wenig mehr als die Hälfte der jeweiligen Bruttodatenrate
- Kommunikationsbereich
 - 100m Außen-, 10m Innenbereich (Vergrößerung der Reichweite durch externe Antennen; auch möglich: Richtfunk über mehrere km)
 - Max. Datenrate bis ~5m
- Frequenzbereich
 - Freies 2.4 GHz-Band (2.4 - 2.4835 GHz) ISM = Industrial – Scientific – Medical
 - Optional 5 GHz-Band
- Sendeleistung
 - maximal 100 mWatt
- Lizenzfrei
- Starke Störungen auf dem Frequenzband (Mikrowellenherde, analoge TV-Übertragung, Überwachung, lizenfreie Stadtnetze), keine Qualitätsgarantien

Durchsatz eines WLAN

Mit wachsender Entfernung nimmt die Durchsatzrate in einem WLAN ab

- Faktisch wird die Modulationsart angepasst
- Hohe Übertragungsraten nur in Radien von 10m
- Bereits bei etwa 30m liegt die 802.11g-Rate bei 11 MBit/s
- Bautechnische Hindernisse reduzieren dies zusätzlich
- **Keine Mobilität** (relative Geschwindigkeit etwa 10km/h – **802.11p**)

Erneut muss erwähnt werden, dass diese Datenraten niemals der Anwendung zukommen. Bei WLANs findet man häufig nur TCP-Nutzraten von 50% und weniger!

Mit der Multiple-Inputs-Multiple-Outputs-Technik (**MIMO**) kann die **Datenrate** in Zukunft **deutlich gesteigert** werden **802.11n**

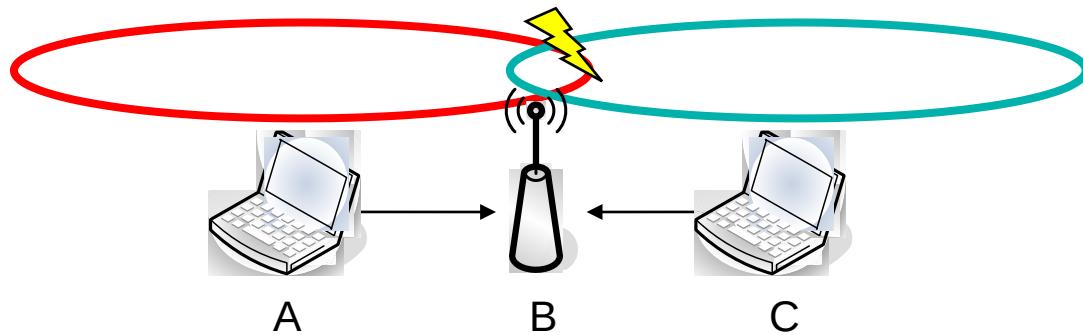
- Durch den Einsatz mehrere Antennen als Gruppe kann ein stärkeres Empfangssignal empfangen werden
 - Größere Distanzen bei gleichem Durchsatz
 - Höhere Datenraten bei gleicher Entfernung
- Zusätzlich wird versucht eine Richtwirkung zu nutzen

Medienzugriff in drahtlosen Netzen

- Verwendung von CSMA/CD?
 - Horche vor der Übertragung, ob bereits jemand sendet
 - Horche während der Übertragung, ob jemand anderes parallel sendet
- Probleme bei drahtloser Kommunikation
 - Sehr schwierig (teuer), gleichzeitig zu senden und zu empfangen
 - > Kollisionserkennung kann nicht durchgeführt werden
 - Kollision tritt beim Empfänger auf, aber nicht notwendigerweise beim Sender
 - > “Broadcastnetz” ohne garantierten Empfang aller Übertragungen: **Hidden Station**

Hidden-Station-Problem

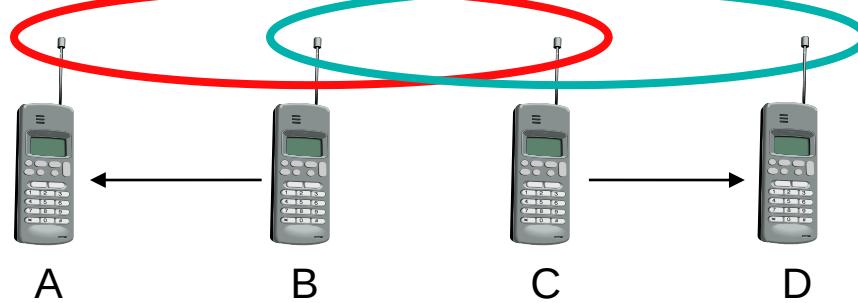
- **Hidden Station**
 - A sendet an B, C empfängt A nicht
 - C will an B senden, stellt freies Medium fest (CS schlägt fehl)
 - Kollision bei B, A bemerkt sie nicht (CD schlägt fehl)
 - A ist *hidden* (versteckt) für C



Exposed-Station-Problem

Exposed Station

- B sendet zu A, C will zu D senden
- C muss warten, da CS ein „besetztes“ Medium signalisiert
- da A aber außerhalb der Reichweite von C ist, ist dies unnötig

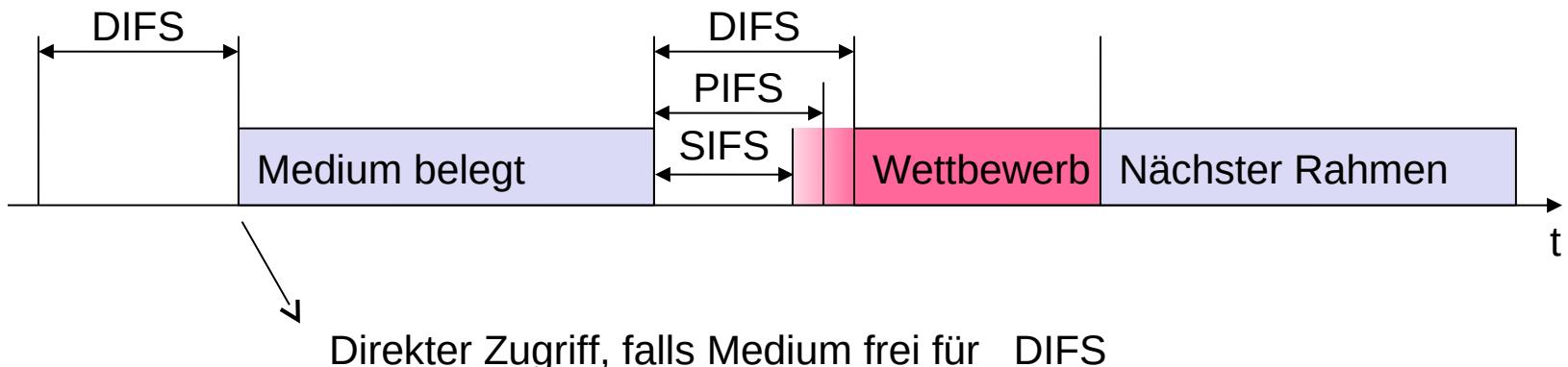


Medienzugriff bei WLAN

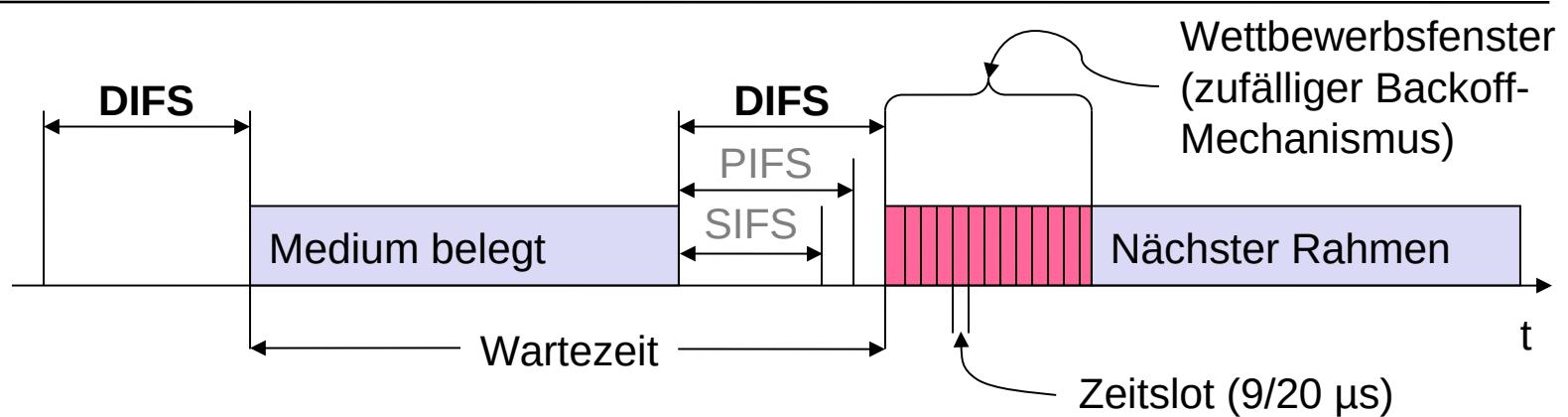
- CSMA/CA
 - **Carrier Sense Multiple Access with Collision Avoidance**
 - Kollisionen können nicht erkannt werden, darum wird versucht, sie zu vermeiden
 - Carrier Sense mit zufällsgetriebenen **Backoff-Mechanismus**
- Hier nicht relevant
 - **RTS/CTS-Handshake (Request to Send / Clear to Send)**
 - **Polling** durch den AP

Medienzugriff durch gestaffelte Zugriffskonzepte

- **Prioritäten** durch **Staffelung von Zugriffszeitpunkten**
 - Keine garantierten Prioritäten, aber wichtige Nachrichten bekommen eher Zugriff
 - Einteilung in Zyklen: Drei Intervalle (Inter Frame Space, IFS)
 - > Short IFS (**SIFS**) für **Bestätigungsnotizen**, 10 – 16 μ s
 - > Point Controlled IFS (**PIFS**) für Aktionen des AP beim **Polling**, 19 – 30 μ s
 - > Distributed Control IFS (**DIFS**) für **Datenübertragung**, 28-50 μ s

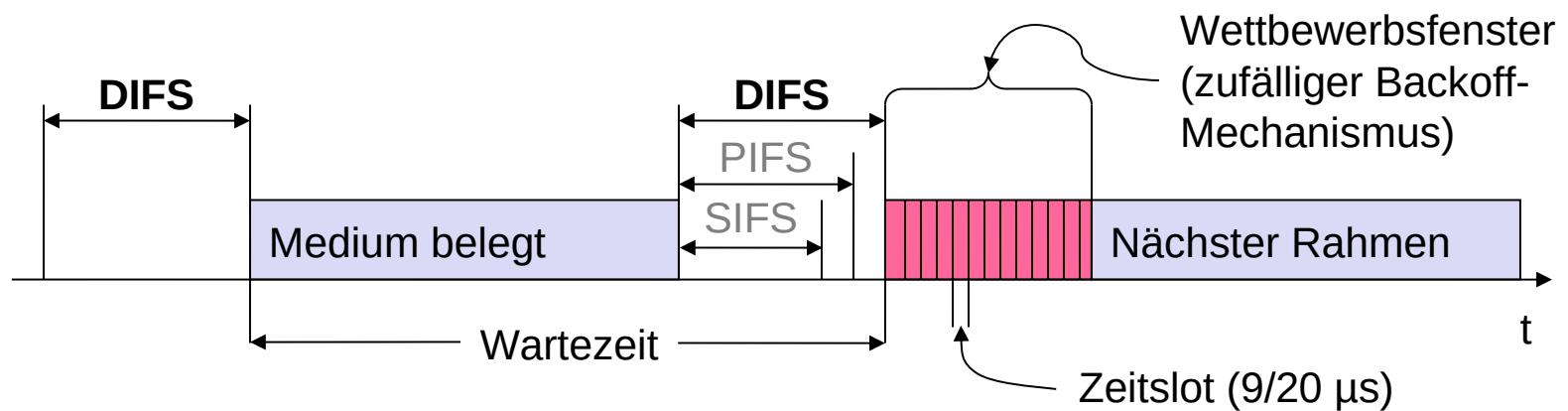


Kollisionsvermeidung durch Wartezeitverteilung



- Idee
 - Vor Beginn des Sendens: **Carrier Sense**
 - Falls Medium frei für mindestens eine Zeit von DIFS, starte direkt mit Übertragung
 - Falls Medium belegt: warte bei Freiwerden erneut für DIFS, wähle dann eine Backoff-Zeit vor nächsten Zugriffsversuch (**Kollisionsvermeidung**)
 - > Backoff-Zeit ist Vielfaches eines Zeitslots

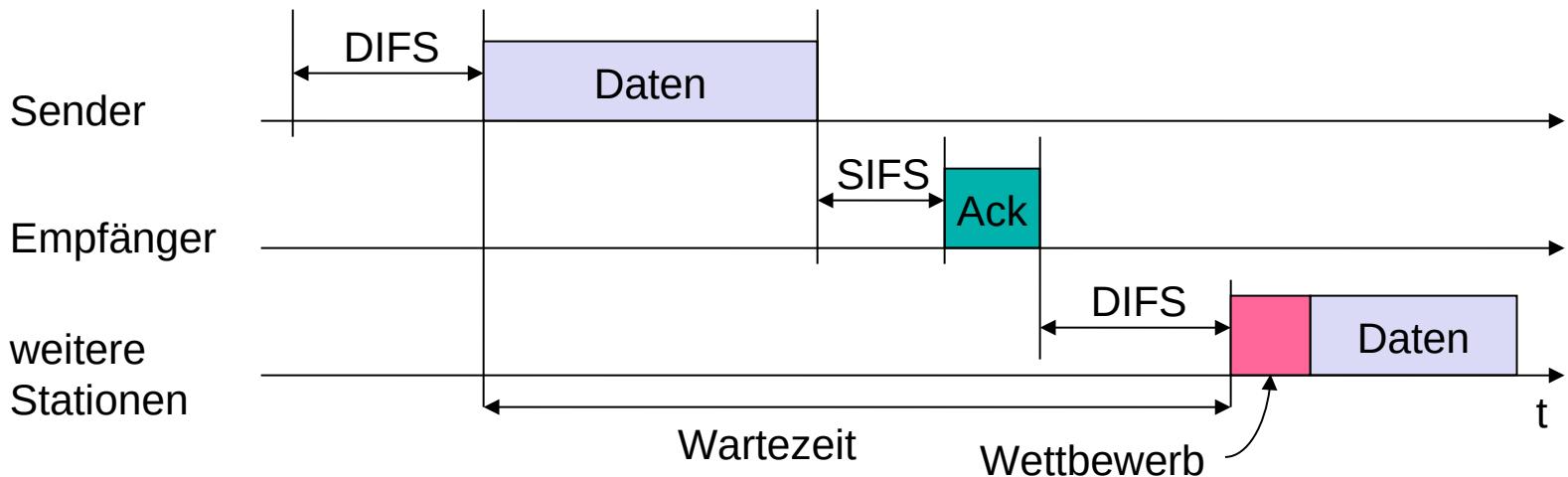
Kollisionsvermeidung durch Wartezeitverteilung



- Vorgehen
 - Falls Medium nach Ablauf der Backoff-Zeit noch immer frei, starte mit Übertragung
 - Falls Medium eher belegt wird:
 - > Stoppe Backoff-Zähler
 - > Verwende aktuellen Wert beim nächsten Versuch weiter

CSMA/CA - Quittungen

- Quittierung jeder Übertragung, da Kollisionen nicht erkannt werden können
 - **Direkte Bestätigung** jedes korrekten Datenrahmens
 - > Wichtige Kontrollinformation, daher werden diese bereits nach SIFS ohne jegliches Backoff versendet



FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de