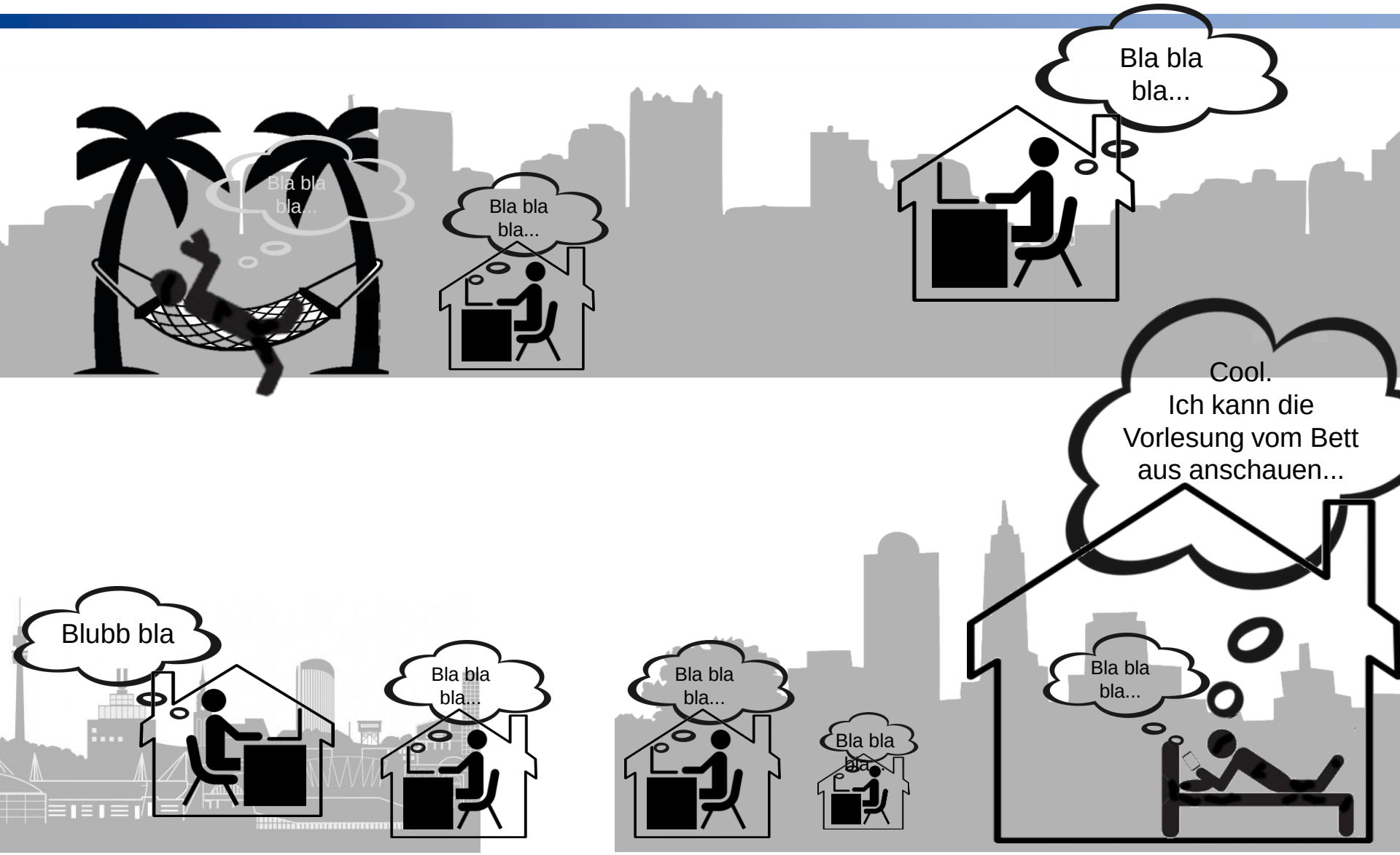
Vorlesung Datenkommunikation WS 2022/23

Klaus Wehrle

Communication and Distributed Systems
Chair of Computer Science 4
RWTH Aachen University

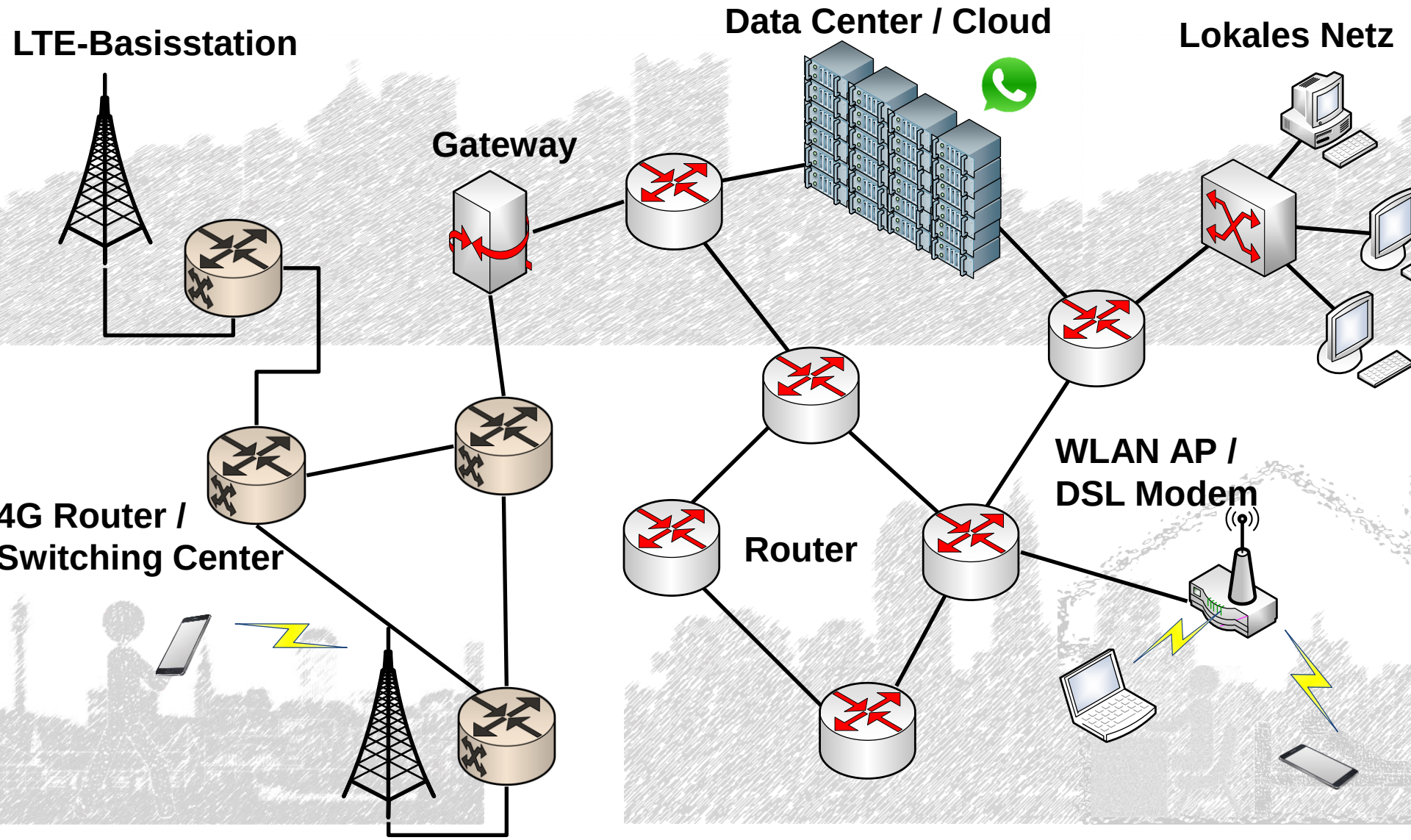
https://www.comsys.rwth-aachen.de

Grundlage: vernetztes System

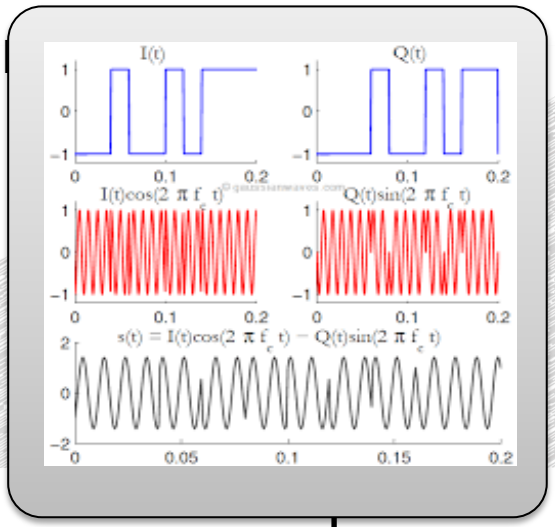


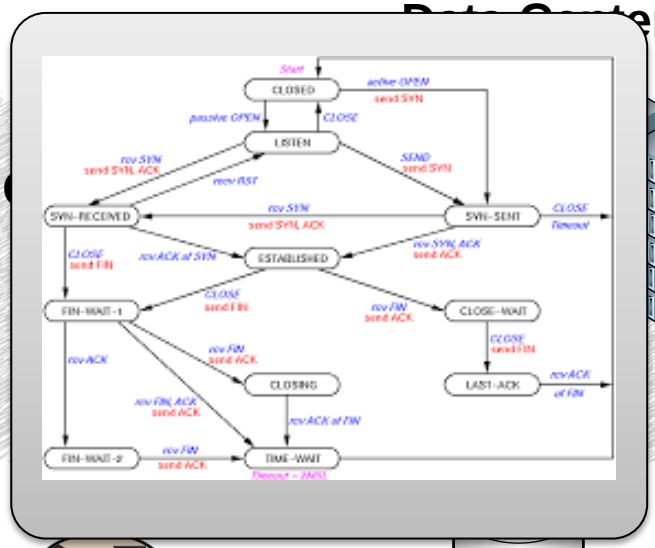
Grundlage: vernetztes System

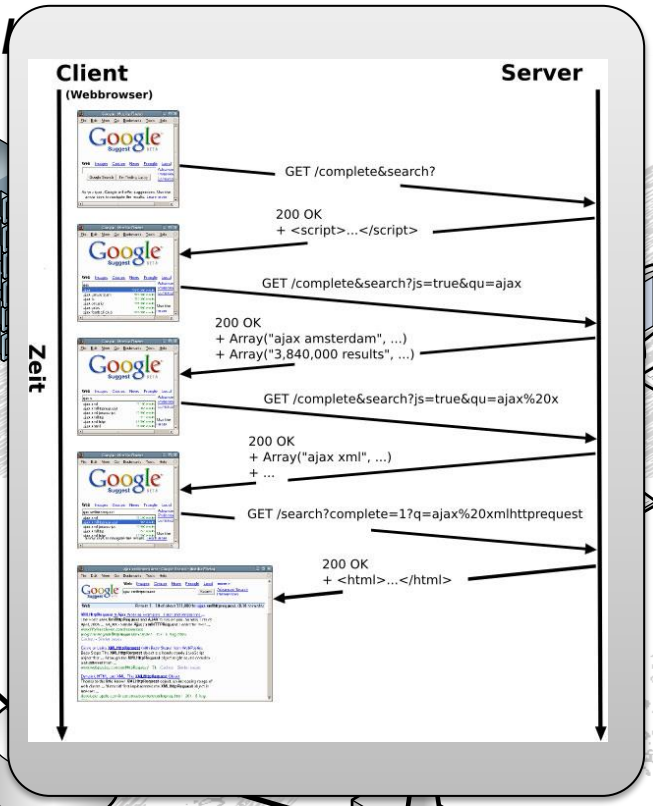
Grundlage: vernetztes System



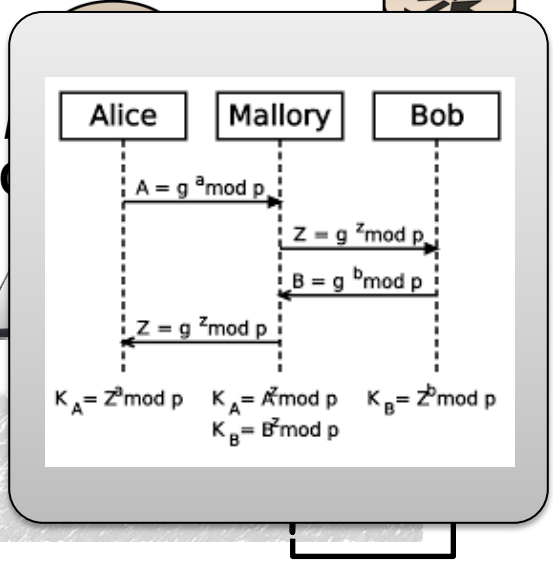
Grundlage: vernetztes System







4G Router /
Switching C



Router

Überblick

- **Vorstellung**
  - Research Vision
  - Teaching Agenda
  - COMSYS Team
- **Vorlesung „Datenkommunikation“**
  - Organisatorisches
  - Unterlagen
  - Übungen
  - Klausuren
- **Motivation & Themen der Vorlesung**

Wer sind wir?

- **Lehrstuhl für Kommunikation und verteilte Systeme (COMSYS, i4)**

Vorlesung:

Klaus Wehrle

Übungsbetrieb:

Mirko Stoffers

Justus Breyer

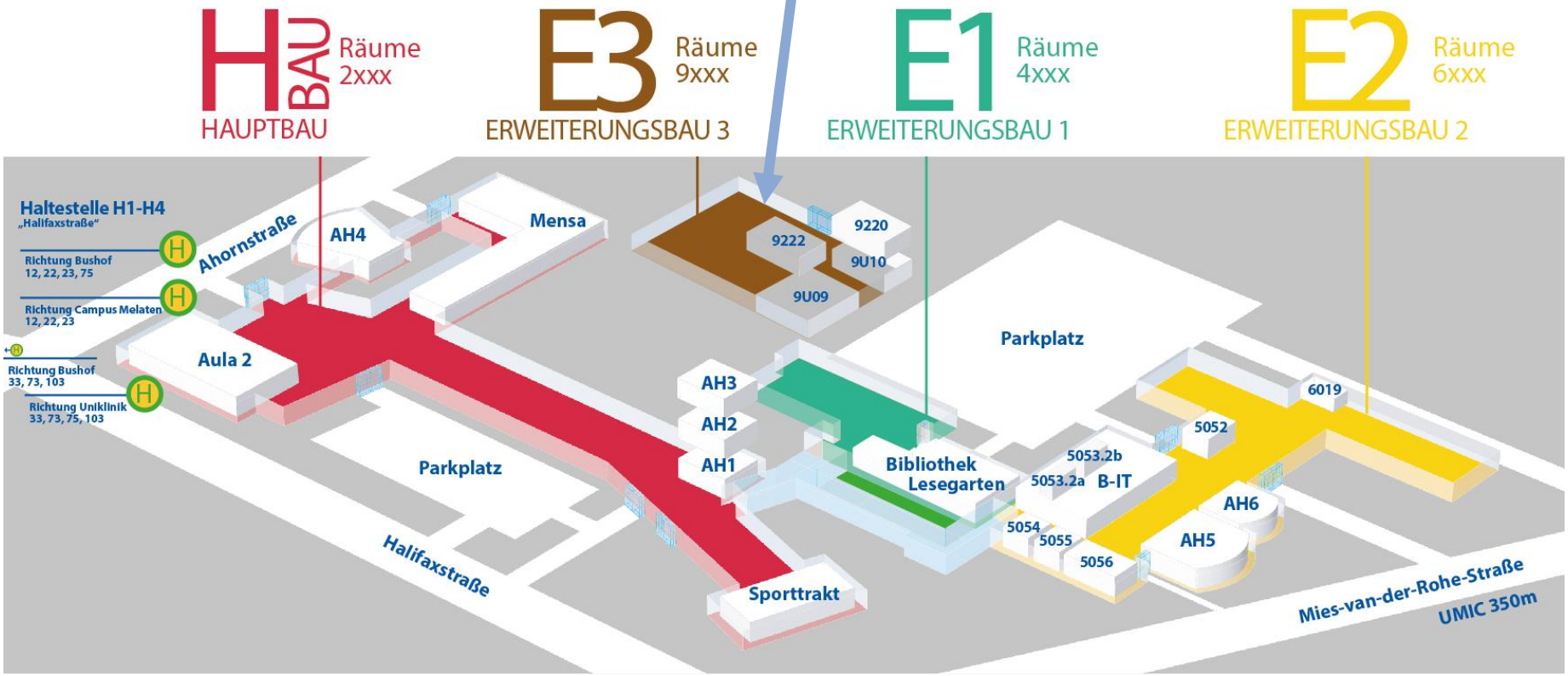
Team-Assistentinnen:

Claudia Förster

Joyce Sturm

Wo findet man uns?

- **Informatikzentrum – Gebäude E3**
  - Erdgeschoss
    - Eingang im Keller



COMSYS-Überblick

Research at COMSYS

- Research Vision

Teaching at COMSYS

- Teaching Agenda

COMSYS Team

- Your academic/innovational career with COMSYS

COMSYS
Research Vision

http://comsys.rwth-aachen.de

The Internet, even mobile, is working!

… why doing research in communication?

- **Challenges caused by…**
  - Large-scale distributed systems
  - Heterogeneous devices with diverse resources
  - Utilization of new/ubiquitous data sources
  - Spontaneous interaction and cooperation
  - Mobility of users, devices, and applications
  - Limited resources in wireless networks
  - Extreme performance requirements

Sensor networks **Mobile communication** **Large-scale distributed systems**

Scalability?

Reliability?

Adaptability?

Performance?

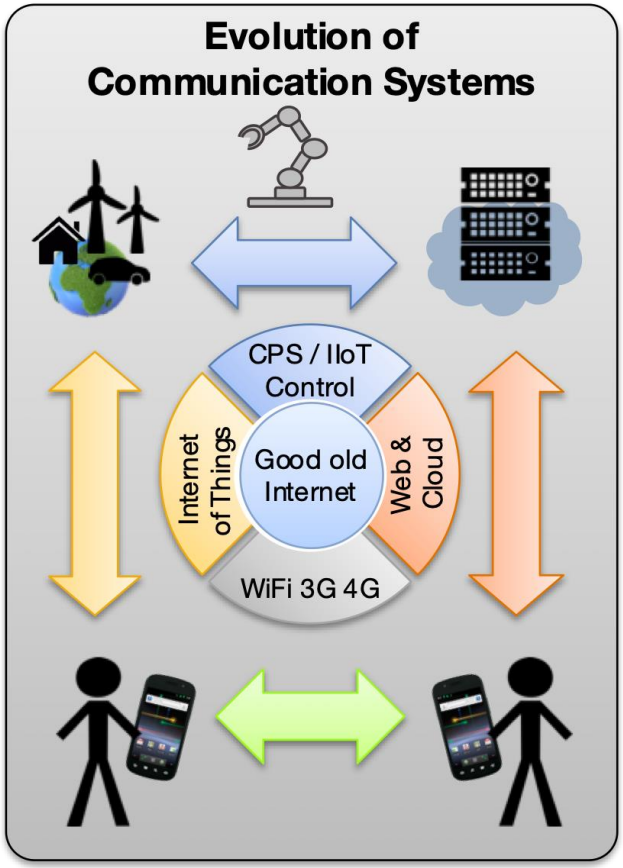
Mobility?

Privacy?

Security?

→ ***Internet of Things (IoT), Industrial IoT (IIoT), Cyber-Physical Systems***

Research Vision



Communication is an enabling technology for all aspects of our daily life and makes the world an amazing distributed system

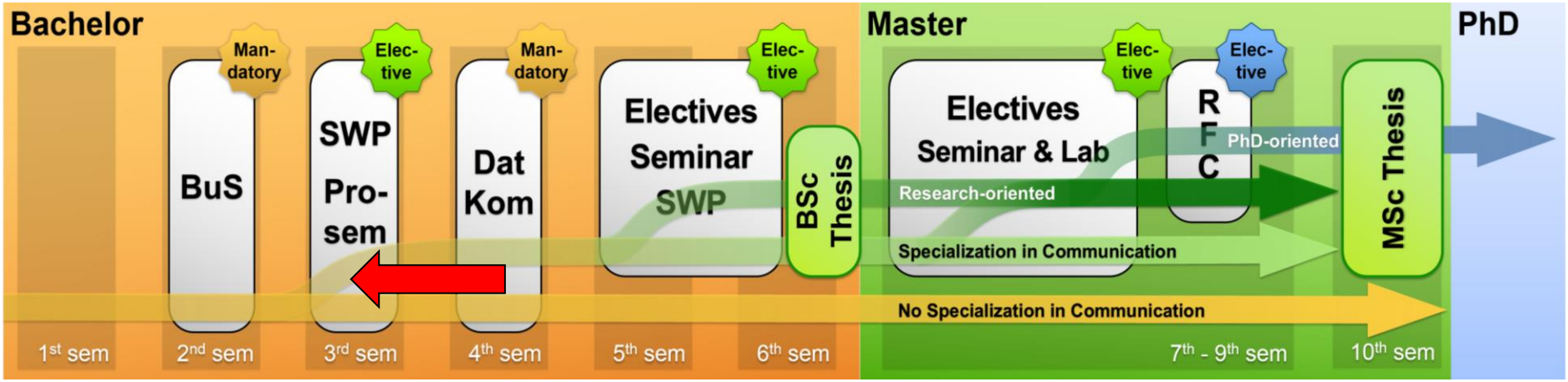
- COMSYS tackles all (interesting) research challenges of the modern connected world
- by **analyzing**, **understanding** and **engineering** the **fundamentals** of communication systems, distributed systems, and their usage in various areas
- Teaching and training of next generation's experts
- 36 years of experience in a great team

System Analysis Group	Security & Privacy Group	Cyber-Physical-Systems	Network Architectures

COMSYS
Teaching Agenda

http://comsys.rwth-aachen.de

Teaching Agenda – B.Sc. und M.Sc.



https://www.comsys.rwth-aachen.de/teaching/comsys-course-overview/

Man-datory Pflichtveranstaltungen

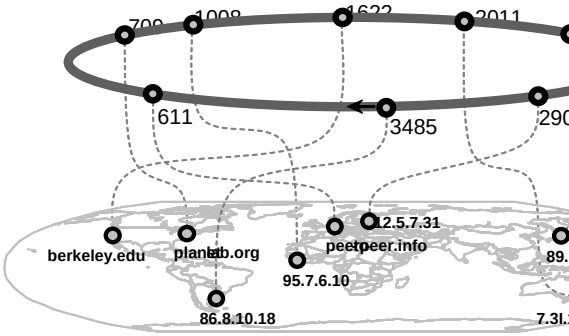
Elec-tive Bachelor & Master

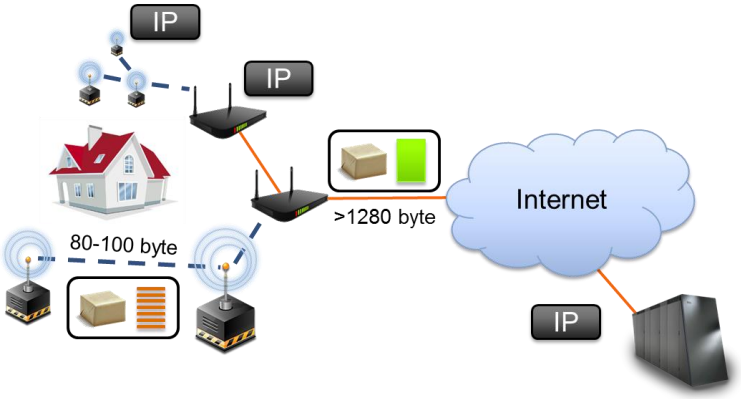
Elec-tive Nur Master

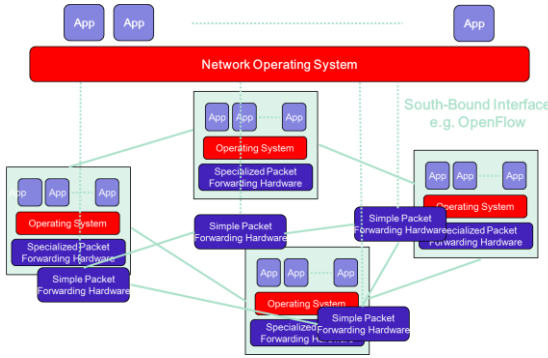
- **Electives (Wahlpflichtfächer)**
  - Advanced Internet Technology
  - Communication Systems Engineering
  - Mobile Internet Technology
  - Research Focus Class

Vorlesung: Advanced Internet Technology (AIT)

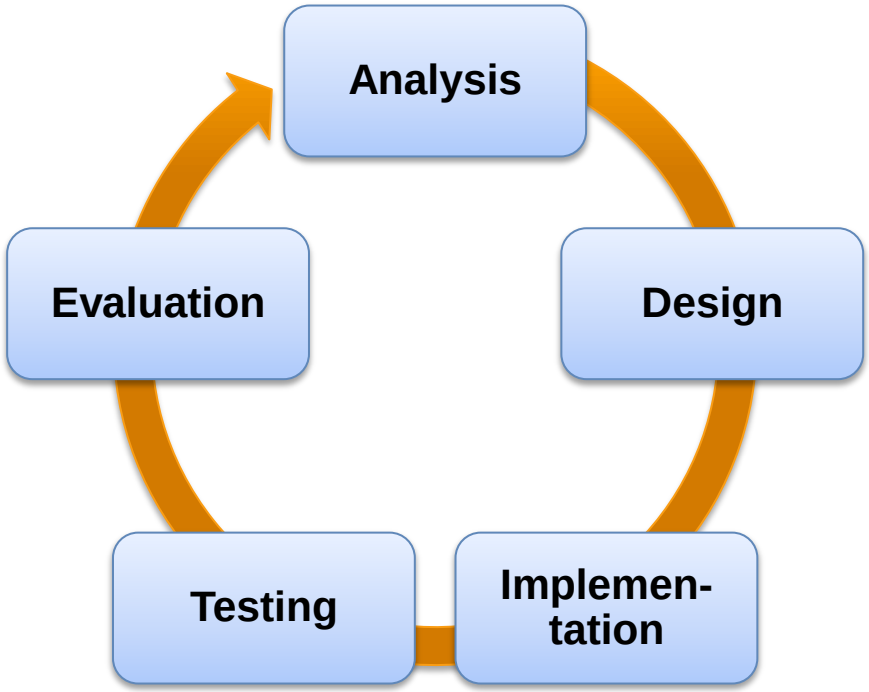
- **Modernere/aktuelle Entwicklungen im Bereich Kommunikationssysteme**
  - Peer-to-Peer, Blockchain, Cloud Computing
  - Software Defined Networking, Quality of Service
  - Cyber-physical Systems, Internet of Things
  - Security & Privacy Aspects



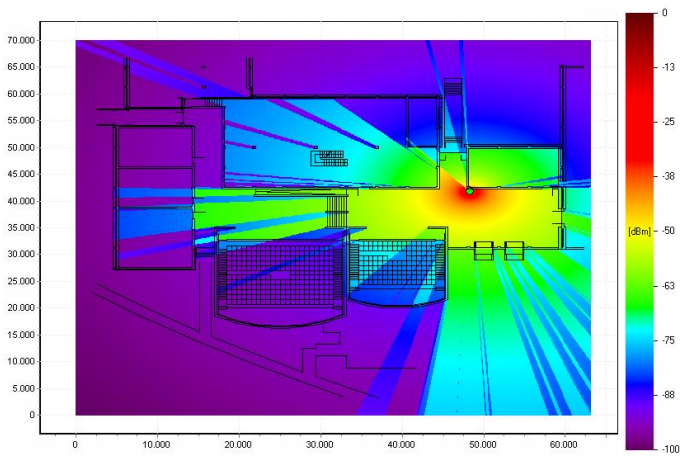


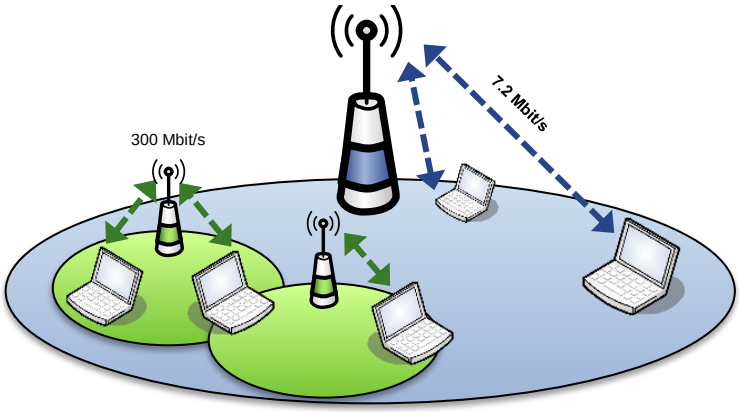


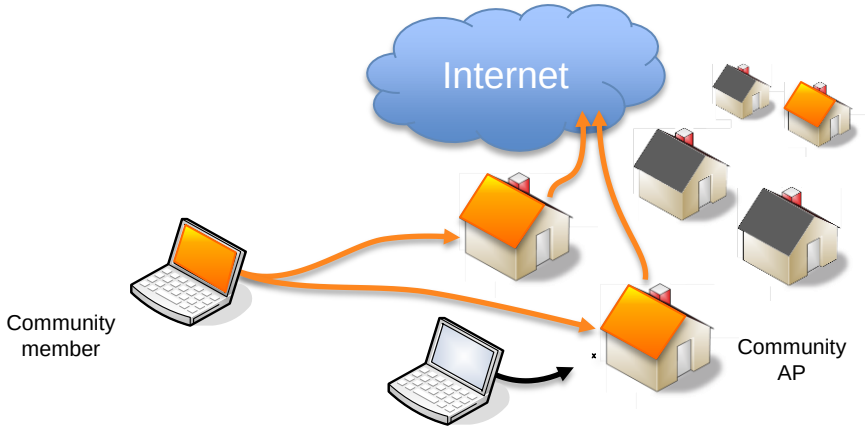
- **Entwicklung von Kommunikationssystemen**
  - … basierend auf den Grundlagen dieser Vorlesung
    - Netzwerkprogrammierung
    - Protokoll-Design
    - Networking & Kernel
    - Testing
    - Simulationen
    - Internet-Messungen



- **Datenkommunikation in drahtlosen Netzen**
  - Modulation, Kodierung, Signalausbreitung
  - Medienzugriff
  - WLAN und 2/3/4/5/6G-Netze als Beispiele
  - Internet-Protokolle und Mobilität



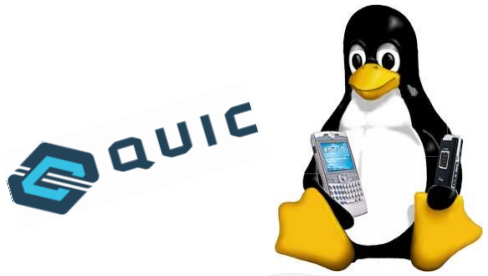


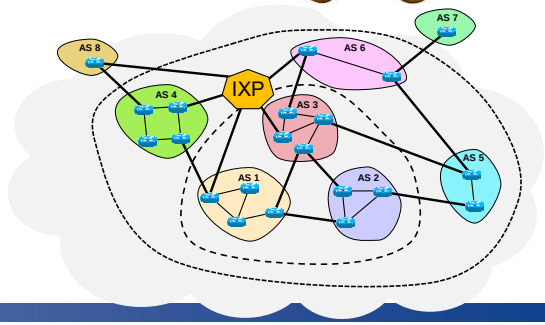


Seminars and Labs

- **Seminars @ COMSYS**
  - Everywhere: Write paper & give presentation
  - Special @COMSYS
    - Conference-style organization
    - Insight into daily business of researchers
    - Chance to work on your own publication

- **Labs @ COMSYS**
  - Transport Protocols (B.Sc.)
    - Develop and implement your own TCP or QUIC
  - Practical Internet eXperience (M.Sc.)
    - Operate your own network + interesting projects
    - Coming to B.Sc. soon





Spezialvorlesung: Research Focus Class

- **Research Focus Class (RFC)**
  - Credits wie für normale Vorlesungen
  - Sehr forschungsorientiert
  - Nur bei genügend (>5) interessierten Teilnehmern

- **Inhalte & Konzept**
  - Wechselnde Themen
    - Internet of Things, Privacy, Mobile Health, …
  - Einführung durch COMSYS-Mitarbeiter – Grundlagen
  - Auswahl von behandelten Themen im Team – hot topics
  - Vorträge und Diskussionen organisiert durch Studierende – interaktiv
  - Simulationen, Prototypen, Analysen, Experimente – echte Forschung!

Security & Privacy Lunch

- **Not an official course, but informal meetings**
  - Bring your lunch and watch current scientific talks with us!
    - Ok… currently not… but maybe the pandemic ends sometime…
  - Mailing list: https://lists.comsys.rwth-aachen.de/listinfo/sp-lunch
    - Subsribe to get information on dates and topics

- **Who can join?**
  - Bachelor & Master students
  - No credits, thus you cannot take it as kind of elective…
  - … but if you are interested in current Security & Privacy research, just step in!

Interesse an einem Hiwi-Job oder einer Bachelorarbeit?

- **Aktuelle Themenauswahl Bachelorarbeiten**
  - https://www.comsys.rwth-aachen.de/teaching/available-theses
  - Nur innerhalb des RWTH-Netzes verfügbar

- **Aktuelle Hiwi-Jobs**
  - https://www.comsys.rwth-aachen.de/research/hiwi-calls

Interesse an einer Bachelorarbeit?

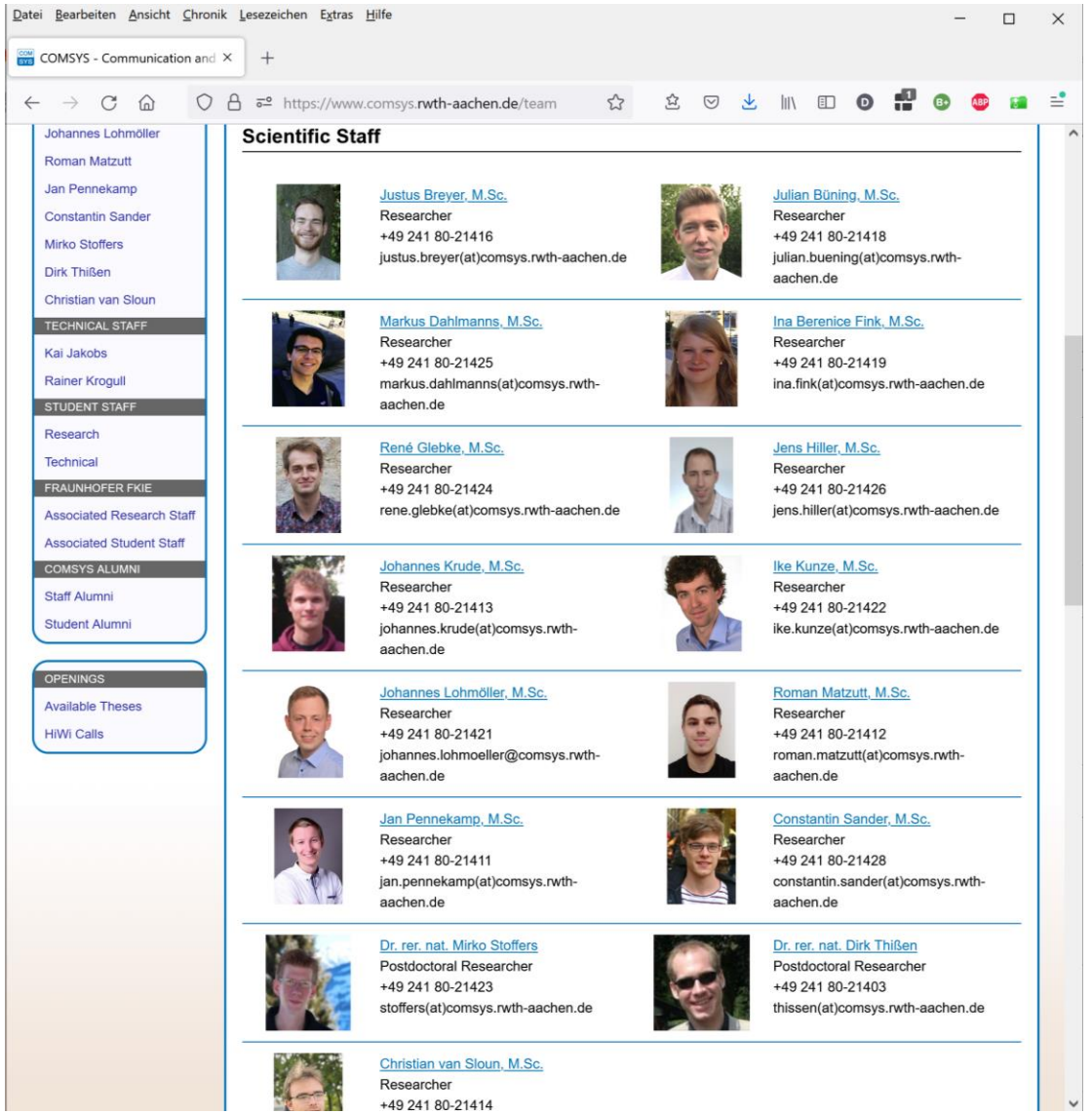
- **Werde Teil von COMSYS**
  - Zeige Interesse!
  - Wir arbeiten an vielen Themen
    - Siehe Forschungsbereiche auf den Webseiten
    - Interesse an einem Bereich? Komm vorbei!
      - *Viele Themen für Bachelorarbeiten werden „an der Tür“ vergeben*
      - Diese Themen kommen nie auf die Webseite
      - Eventuell kann man sein Thema mit beeinflussen
    - Bitte beachten: wir haben nur endliche Kapazitäten
      - *Überzeuge uns, warum Du ein Thema bekommen sollst ☺*

COMSYS Team and Career Opportunities

http://comsys.rwth-aachen.de

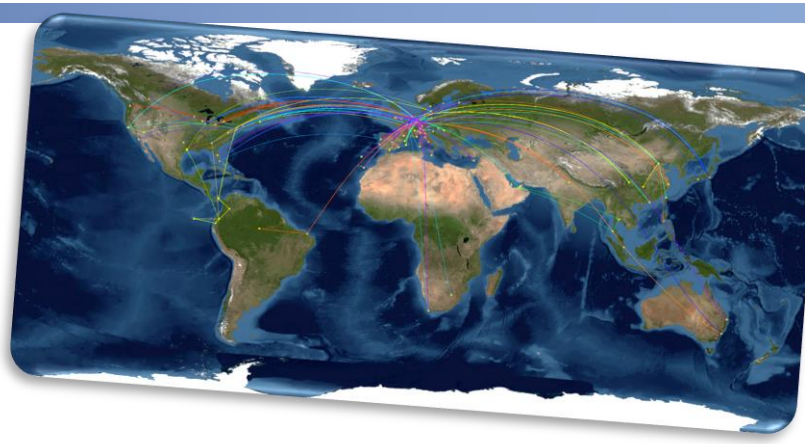
COMSYS Team

- **Currently…**
  - 1 professor
  - 2 junior profs*
  - 2 postdocs
  - 12 researchers
  - 15-20 student science assistants
  - 2 external researchers
  - 2 BBQs
  - 1 billard table
  - 1 foosball table
  - ...



* = training on the job ;-)

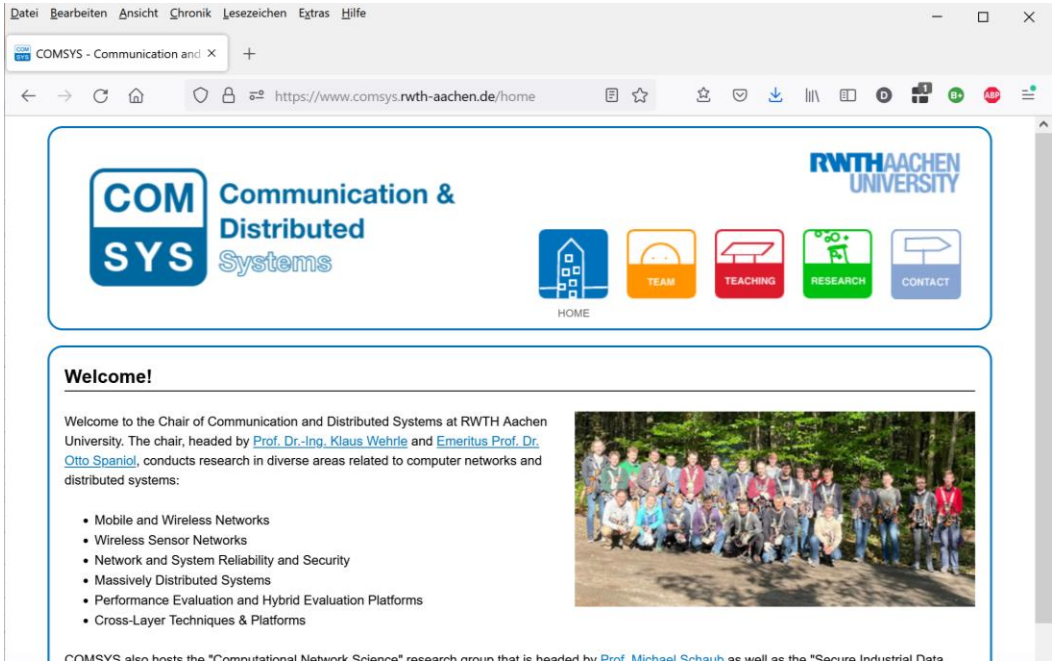
Opportunities for Students



- **Interested in research?**
  - Come along, talk to us
    - Maybe „just“ a thesis topic is found…
    - … but sometimes, even conference papers are outcome of a thesis!
      - You might be financed by us to attend the conference ☺
- **Interested in a student helper job?**
  - Again: come along, talk to us
    - Hiwi jobs are rare, they seldom come to public announcements
- **FOLKS@COMSYS**
  - The COMSYS club
  - E.g. scholarships / best thesis awards for students with outstanding achievements in communication systems

Homepage

- **All important information is available here**
  - https://www.comsys.rwth-aachen.de



  - COMSYS in social media

    /comsysrwth

    @COMSYS_rwth

Datenkommunikation

Organisatorische Informationen

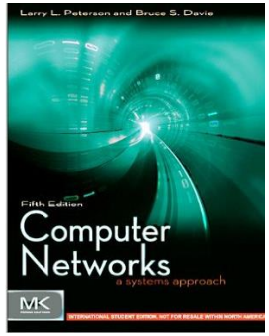
Organisatorisches

- **Termine der Vorlesungen:**
  - Donnerstag, 12:30 – 14:00
  - Freitag, 12:30 – 14:00
  - Vorlesung umfasst nur 3 SWS, daher fallen nach Ankündigung einzelne Termine aus
    - morgen (13.10.): fällt aus
    - nächsten Dienstag (18.10.): Vorlesung statt Übung

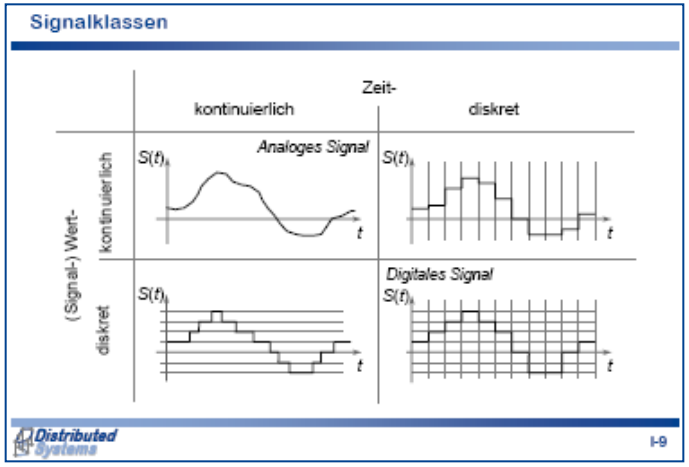
- **Übungen:**
  - Kleingruppenübungen
  - Großübung / Diskussionsstunde

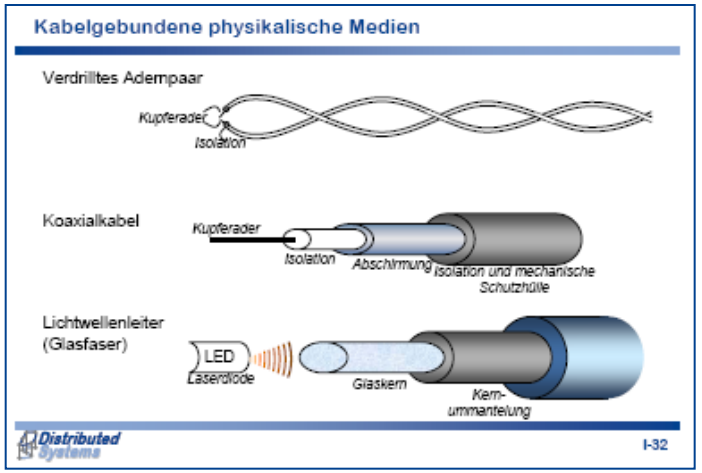
Materialien

- **Vorlesungsunterlagen und weitere Infos:**
  - Folien
    - Sind im Moodle-Lernraum verfügbar (nach der jeweiligen Vorlesung)
  - Bücher
    - A.S. Tanenbaum: „Computernetzwerke“, Pearson Studium, 5. Auflage, 2012
    - J.F. Kurose, K.W. Ross: „Computernetzwerke: der Top-Down-Ansatz”, Pearson Studium, 6. Auflage, 2014
    - L.L. Peterson, B.S. Davie: „Computer Networks – A Systems Approach“, Morgan Kaufmann, 2011
  - Sonstiges
    - In den Folien oder der Vorlesung wird eventuell auf weitere Literatur verwiesen



Folien + Erläuterungen = Skript





Nachrichtentechnische Kanäle lassen sich in die obigen Signalklassen einordnen. Die Signalklassen machen eine Aussage über den Signalverlauf, welcher durch seinen Signalwert-Verlauf (y-Achse *s*) über die Zeit (x-Achse *t*) bestimmt ist. Die unterschiedlichen Signalklassen ergeben sich aus der Kombination des Wert- und Zeitverlaufs:

Kontinuierlich: stetiger Verlauf (kein Abstand zwischen je zwei Punkten)

Diskret: sprunghafter Verlauf (Einschränkung auf bestimmte Werte)

Nun kann der kontinuierliche und der diskrete Fall auf den Signalwertverlauf und auf den Zeitverlauf sinnvoll angewendet werden und in beliebigen Kombinationen auftreten. Insgesamt können dabei 2·2 = 4 Signalklassen unterschieden werden:

Signal- und zeitkontinuierlich: z.B. analoges Telefon, Rundfunk: Weder der Signalwert, noch der Zeitverlauf wird zerhackt. Das entspricht dem klassischen analogen Signal

Signalkontinuierlich und zeitdiskret: z.B. periodisches Messen von analogen Werten eines technischen Prozesses: Der technische Prozess könnte z.B. ein Überwachungsprozess sein, in dem von einem Sensor alle 10 Sekunden oder auch auf Anfrage die bestehende Lichtintensität in [Lux] gemeldet wird.

Signaldiskret: digitale Übertragung mit beliebigen Signalwechseln (zeitkontinuierlich) oder festem (meist isochronem) Taktraster (zeitdiskret). In diesem Fall spricht man von einem digitalen Signal

Die zweiwertige digitale Übertragung (Binärübertragung) ist ein Spezialfall des signaldiskreten Verlaufs, da hier die zulässigen Signalwerte auf zwei begrenzt werden.

Es gibt mehrere Verfahren im Bereich der Telekommunikation, um ein Signal einer bestimmten Klasse in ein Signal einer anderen Klasse zu wandeln. Hierbei ist die Wandlung vom kontinuierlichen zum diskreten Fall von besonderer Relevanz: Die Digitalisierung in der Telekommunikation bedeutet, dass signal-/zeitkontinuierliche Signalklassen in entsprechende diskrete Signalklassen gewandelt werden. Das bedeutendste und am meisten verbreitete Telekommunikationssystem, welches eine solchen Wandlung benutzt, ist das Fernsprechen, also das Telefonnetz. Zur Analog-Digitalwandlung wird dort das Pulse Code Modulation (PCM) Verfahren eingesetzt, welches noch im weiteren Verlauf dieses Kapitels erklärt wird.

Verwendete physikalische Medien

Kupferdoppeladern (UTP - Unshielded Twisted Pair): Zwei Kupferdrähte, die verdrillt sind, um Störeinflüsse abzuschwächen. Diese Medienform ist sehr billig und meist schon (aufgrund von Telekommunikationsanlagen) verlegt. Neben UTP gibt es auch noch die teureren, abgeschirmten Kabel mit Kupferdoppeladern, die sinngemäß STP (Shielded Twisted Pair) genannt werden.

Derzeit sind UTP Kabel weit verbreitet, die in unterschiedliche Kategorien aufgeteilt werden: UTP 1 bis UTP 5 (häufig auch als CAT 1 bis CAT 5 bezeichnet), wobei UTP 3 (16 MHz), UTP 4 (20 MHz), UTP 5 (100 MHz) und UTP 6 (200 MHz) für aktuelle Netzwerke interessant sind. Derzeit wird UTP 7 standardisiert, wobei für die Spezifikation eine maximle Frequenz von 600 MHz vorgeschlagen wurde (Damit wäre es durchaus möglich, ATM mit 622 Mbit/s zu realisieren!). UTP 5 Kabel sind derzeit wohl am weitesten verbreitet. Damit ist es möglich, Übertragungsraten von bis zu 155 MBit/s zu erreichen.

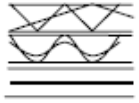
Koaxialkabel: Besser abgeschirmte Kupferkabel, die allerdings ziemlich unflexibel sind.

Lichtwellenleiter: Glaskern, in dem sich die Lichtwellen ausbreiten. Darum befindet sich ein Mantel mit einer größeren optischen Dichte als der Glaskern, so dass es zur Totalreflexion kommt und die Lichtwellen sich nur innerhalb des Glaskerns ausbreiten. Vorteilhaft ist die immense Bandbreite (bis TerraHz), gute Abhörsicherheit und die sehr niedrige Dämpfung. Man unterscheidet die folgenden Arten von Glasfasern:

Multimode mit Stufenindex

Multimode mit Gradientenindex

Monomode



Auch im Bereich der Glasfasern lassen sich unterschiedliche Reichweiten erzielen: Sie reichen von etwa 2 km bei Multimodefasern bis zu 40 km bei Monomode Glasfasern.

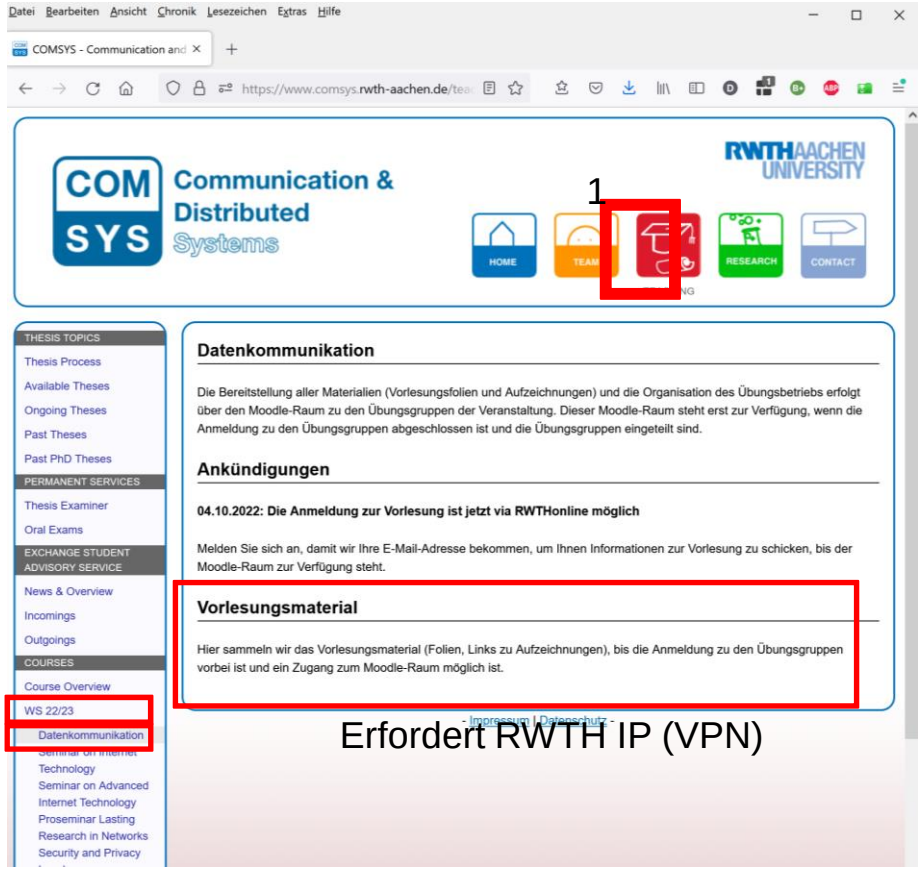
Organisatorisches

- **Moodle-Lernraum zur Vorlesung**
  - Verknüpft mit der Vorlesung in RWTHonline
  - Bereitstellung von Folienkopien, Übungen und sonstigem Material

- **Abwicklung des Übungsbetriebs über Moodle**
  - Bereitstellung von Übungsblättern
  - Abgabe von Lösungen
  - Punkteverwaltung
  - Diskussionsforum
  - ACHTUNG:
    Zugang zum Lernraum und Online-Teilnahme an den Übungsgruppen nur durch Anmeldung zu den Kleingruppenübungen in RWTHonline

Bis wir einen Lernraum haben…

- … Material über unsere Website abrufbar
  - https://www.comsys.rwth-aachen.de/teaching/ws-2223/datenkommunikation
  - Nur aus RWTH-IP-Adressbereich!
    - VPN verwenden!
    - Anleitung beim IT-Center verfügbar



Erfordert RWTH IP (VPN)

Übungsgruppen – Termine

- **Wöchentliche Übungen**
  - Wöchentliche Ausgabe eines Übungsblatts zur Bearbeitung (Moodle)
  - Abgabe der Übungen in Dreiergruppen (in Ausnahmefällen Zweiergruppen)
  - Korrektur der Übungen durch studentische Hilfskräfte
  - In den Gruppen: Fragemöglichkeit zu Korrekturen, freiwillige Präsenzübungen
  - Termine der Kleingruppenübungen:
    - Mittwoch 10:30 – 12:00 Uhr (2 Gruppen)
    - Mittwoch 12:30 – 14:00 Uhr (3 Gruppen)
    - Mittwoch 16:30 – 18:00 Uhr (3 Gruppen)
    - Donnerstag 08:30 – 10:00 Uhr (3 Gruppen)
    - Donnerstag 10:30 – 12:00 Uhr (3 Gruppen)
    - Donnerstag 14:30 – 16:00 Uhr (1 Gruppe)
    - Donnerstag 18:30 – 20:00 Uhr (2 Gruppen)

Übungsbetrieb

- **Abgabe der Übungen (online)**
  - In Dreiergruppen, in Ausnahmefällen in Zweiergruppen
    - Einzelabgaben werden nicht akzeptiert
    - *Möglichst mit Studierenden der gleichen PO-Version!*
      - *Letzte Übung hat vermutlich unterschiedliche Aufgaben für Bachelor Informatik PO 2018 und PO 2022!*
      - *Im Lernraum werden wir abfragen, wer die Vorlesung nach PO 2018 besucht*
      - *Bildung von Abgabeteams später im Lernraum*
  - Elektronisch über den Lernraum
    - Abgabe der Lösungen eines Übungsblatts als **ein pdf-Dokument**
    - Bitte KEINE Abgabe in .docx oder ähnlichem!
- **Rückgabe der Korrekturen**
  - Elektronisch über Moodle

PO-Version

- **Bisher:**
  - Vorlesung „Datenkommunikation und Sicherheit“
  - im 4. Semester (Sommer)
- **Ab diesem Semester:**
  - Vorlesung „Datenkommunikation“
  - im 3. Semester (Winter)
- **Unterschiede:**
  - kein Themenblock zu Sicherheit, dafür Anwendungsschicht
  - andere Übungen (am Ende des Semsters)
  - andere Klausur
  - andere Vorlesungsinhalte im letzten Block (Sicherheit nur als Videos)
- **Festlegung über Moodle nächste Woche!**

Übungsgruppen – Anmeldung

- **Anmeldung über RWTHonline**
  - 17 Übungsgruppen eingerichtet
  - Anmeldung als Team und mit Priorisierung bestimmter Übungsgruppen möglich
    - Gemeinsame Abgabe ist *nur innerhalb einer Übungsgruppe* möglich
  - Aber: endgültige Gruppenzuteilung durch uns!
- **Anmeldung bis Donnerstag, 20. Oktober 2022 !!**
  - Danach: Gruppenaufteilung wird in Moodle übernommen
  - Aus- und auch Abgabe der Übungen über Moodle
- **Zweites Anmeldeverfahren für Vorlesung/Globalübung**
  - *Hat nur den Zweck, Termine in den RWTH-Kalender zu übertragen und eine Mailingliste zu haben, bis Gruppenaufteilung in Moodle übernommen ist*

Übungsbetrieb

- **Diese Woche**
  - Kein Übungsbetrieb
  - Anmeldung zu den Übungsgruppen (bis 20.10.)
- **Nächste Woche**
  - Zuteilung zu den Übungsgruppen
  - Ausgabe des 1. Übungsblatts nach Übernahme der Gruppenaufteilung in Moodle
  - Beginn der Kleingruppenübungen: 26./27. Oktober
  - Festlegung der Veranstaltungsvariante (mit Sicherheit?)
- **Übungsblätter**
  - Dienstag um 12:00 Uhr Aus-/Abgabe

Übungsbetrieb

- **Zusätzlich:**
  - Diskussionsstunde – jeden Dienstag, 12:30 – 14:00
  - Diskussion von Lösungen zu den Übungsblättern
  - Fragestunde zum aktuellen Übungsblatt und zur Vorlesung
  - Erster Termin:
    - 18.10.: *Vorlesung statt Diskussionsstunde!*

Klausur

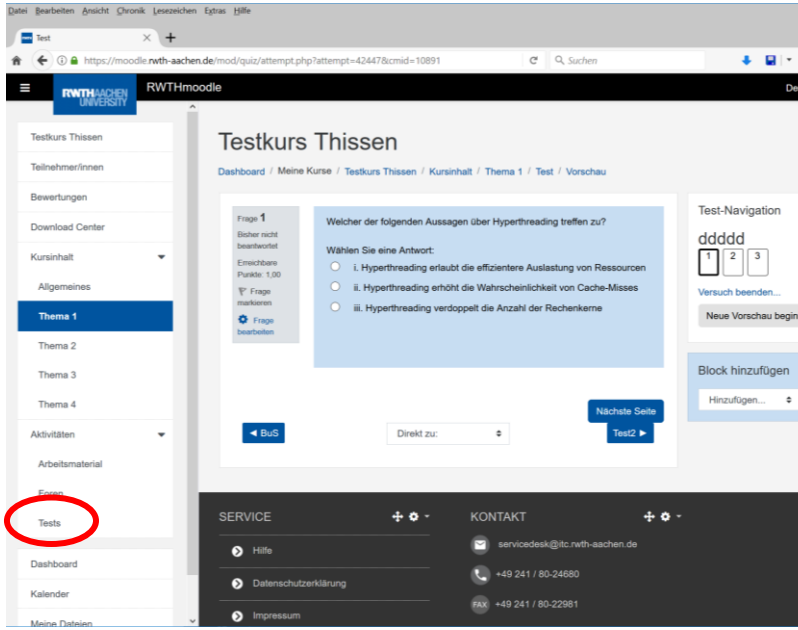
- **Zulassung zur Klausur**
  - Erlangung von mindestens 50% der erreichbaren Punkte aus den Übungen
- **Klausur**
  - Erste Klausur: Montag, 6. Februar 2023
  - Wiederholungsklausur: Montag, 13. März 2023
  - Dauer jeweils 90 Minuten
  - Umfasst den Stoff der Vorlesung und der Übung
  - Ende des Semesters auch Stunden zur „Klausurvorbereitung“
- **Ansprechpartner bei Fragen zum Übungsbetrieb**
  - Mirko Stoffers, Justus Breyer (**`datkom@comsys.rwth-aachen.de`**)

Klausur: Notenbonus

- **Notenbonus für die Klausur durch Übungsbearbeitung**
  - Bis zu zwei Notenstufen (0.3, 0.7) Verbesserung in der Klausur durch Bearbeitung der Übungsblätter
    - Klausur muss bestanden werden – erst dann wird Notenbonus berücksichtigt
    - Nur gültig für den ersten Klausurtermin in WS 2022/23 (6. Februar)
      - Mitnahme des Notenbonus in den zweiten Klausurtermin nur mit sehr guter Begründung möglich (Krankheit, zeitgleiche Klausur)
  - Bonussystem
    - Ab 50% der Punkte in den Übungsblättern: Klausurzulassung
    - Ab 70% der Punkte in den Übungsblättern: Verbesserung um eine Notenstufe (+0,3) in der Klausur
    - Ab 90% der Punkte in den Übungsblättern: Verbesserung um zwei Notenstufen (+0,7) in der Klausur

Quiz am Ende jeder Vorlesung

- **Quiz-System:**
  - Online-Test zur letzten Vorlesungsstunde
  - Feedback für Dozenten und Studierende
  - Ggfs. Wiederholung bestimmter Inhalte in folgender Vorlesungsstunde
- **Ablauf**
  - Test online verfügbar nach jeder Vorlesung, für 24h
  - Folien erst im Anschluss verfügbar
- **Anreiz**
  - Teilnahme an Online-Quiz kann bis zu 20% zusätzlicher Punkte bei den Übungspunkten einbringen, dh. ggf. eine Notenstufe (+0.3) in der Klausur
  - Allerdings: 50% der Übungspunkte müssen über die Übungsblätter erreicht werden, um die Klausurzulassung zu erhalten. Es ist lediglich ein Bonus.
  - Genauere Details folgen per E-Mail via RWTHmoodle



Anmeldung zur Klausur

- **Anmeldung via RWTHonline**
  - Separate Anmeldung zum 1. und 2. Termin
  - Wer noch nicht weiß, welchen Termin er wahrnimmt: bitte zum ersten Termin anmelden
    - Abmeldung bis drei Werktage vor dem Prüfungstermin möglich
  - Rechtzeitig anmelden! Nachmeldungen sind so gut wie unmöglich!

Datenkommunikation

Motivation & Themen

The Internet

- **The Internet …?!?!**
  - It works!
  - So, what's the big deal with it?

  - What is our job here?

A Look into History

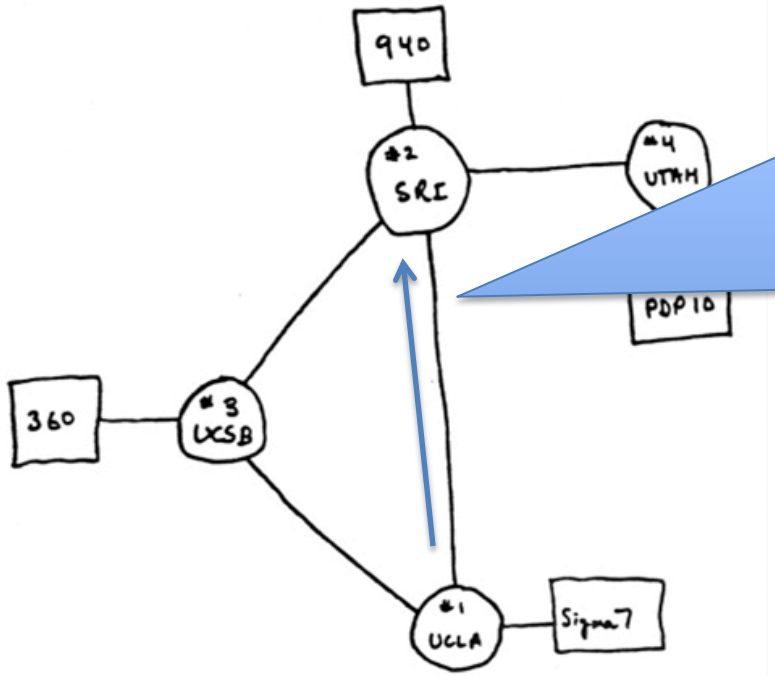
I think there is a world market for maybe five computers

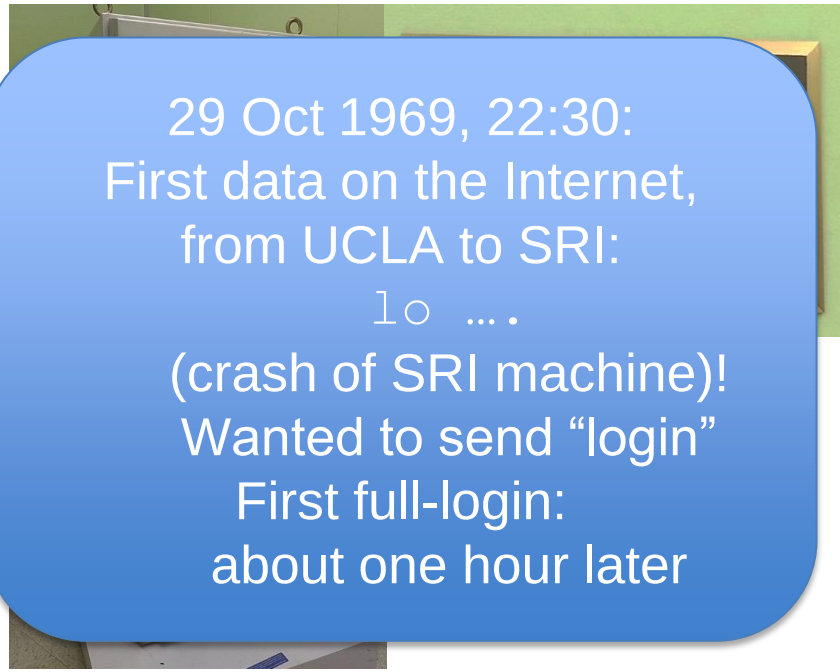
Thomas J. Watson, 1943; Chairman and CEO of International Business Machines (IBM)

1943

A Look into History: Research Network

- **"Initial Internet": Research Network**
  - ARPANET: UCLA, Stanford, UCSB, and Utah (one computer per site)
  - Decentralized connectivity of distributed, independent systems
    - Can operate and communicate in the presence of connection/node failures





29 Oct 1969, 22:30:
First data on the Internet,
from UCLA to SRI:
lo ….
(crash of SRI machine)!
Wanted to send "login"
First full-login:
about one hour later

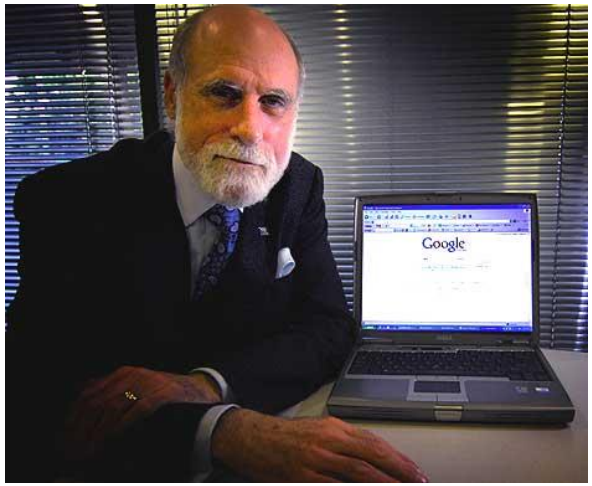
1969

A Look into History: E-Mail

Ray Tomlinson creates first email program

1971

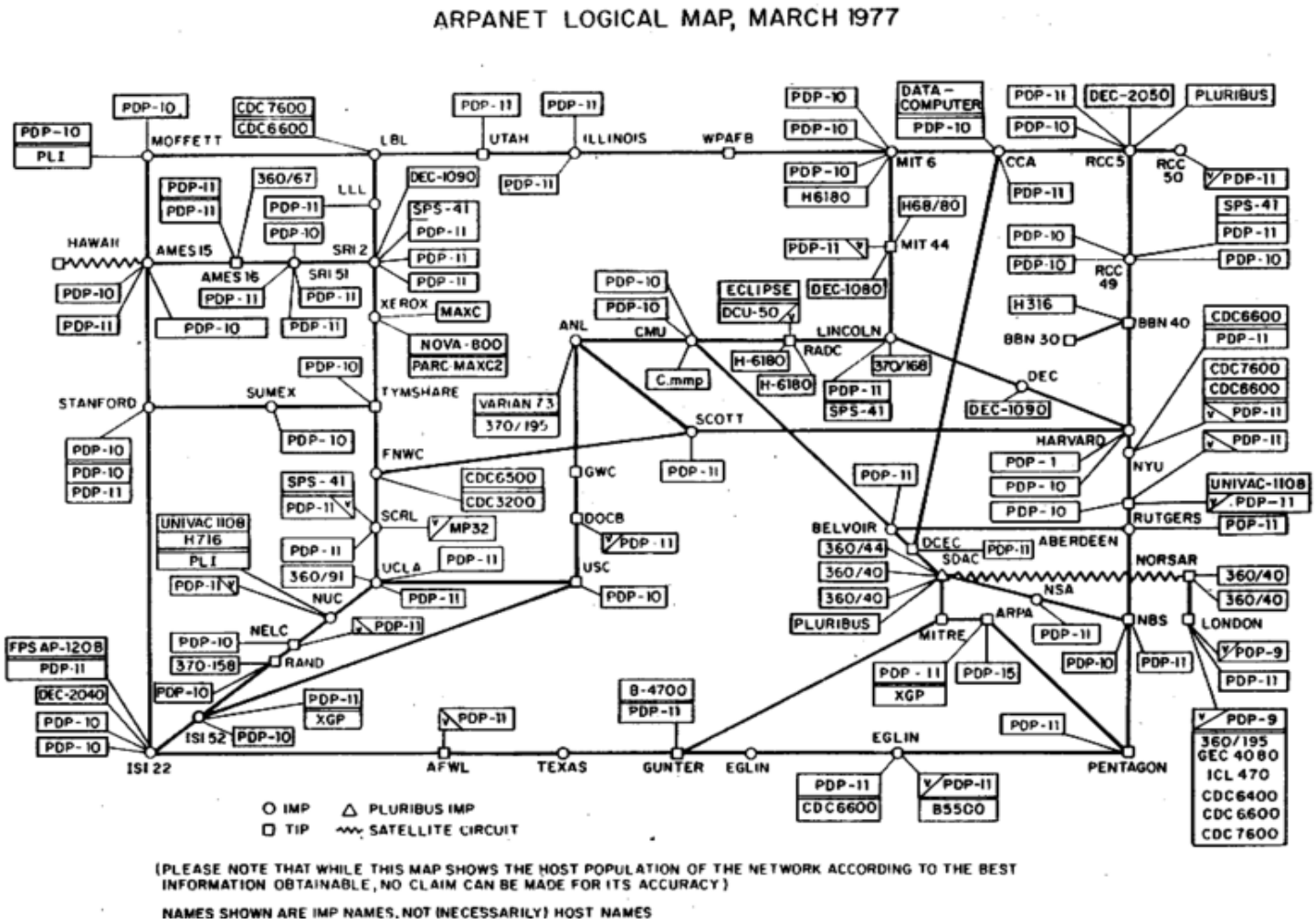
A Look into History: TCP/IP



TCP / IP defined by Vint Cerf & Bob Kahn

1974

A Look into History: Internet Growth

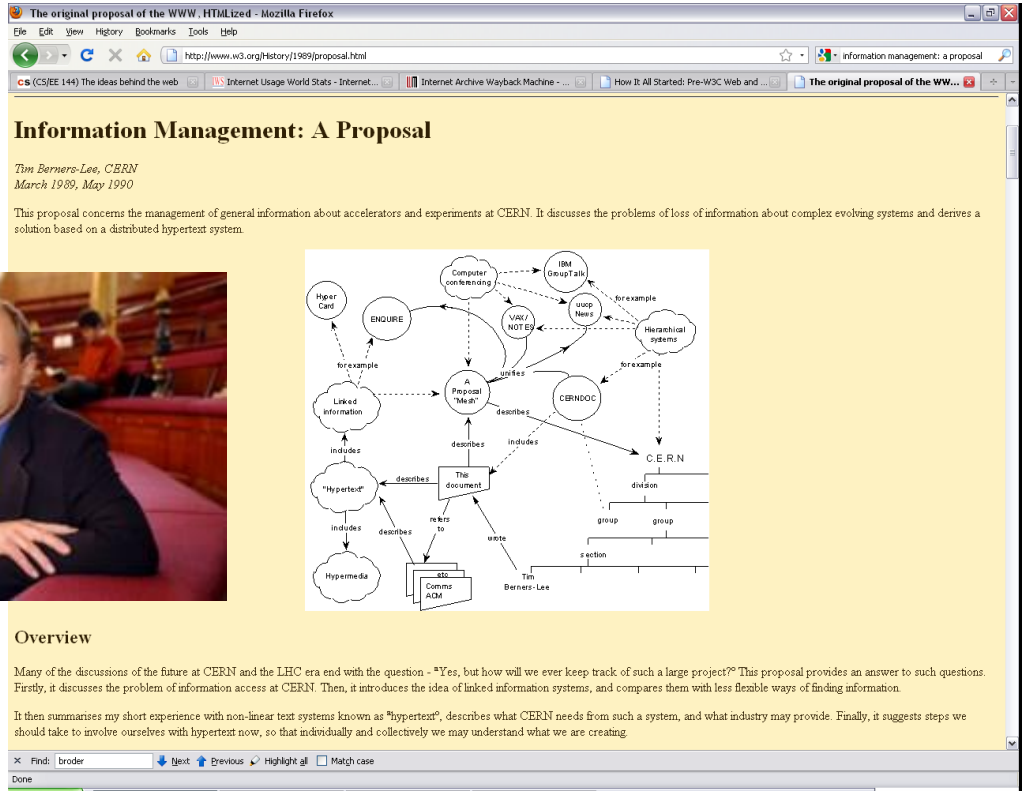


(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

1977

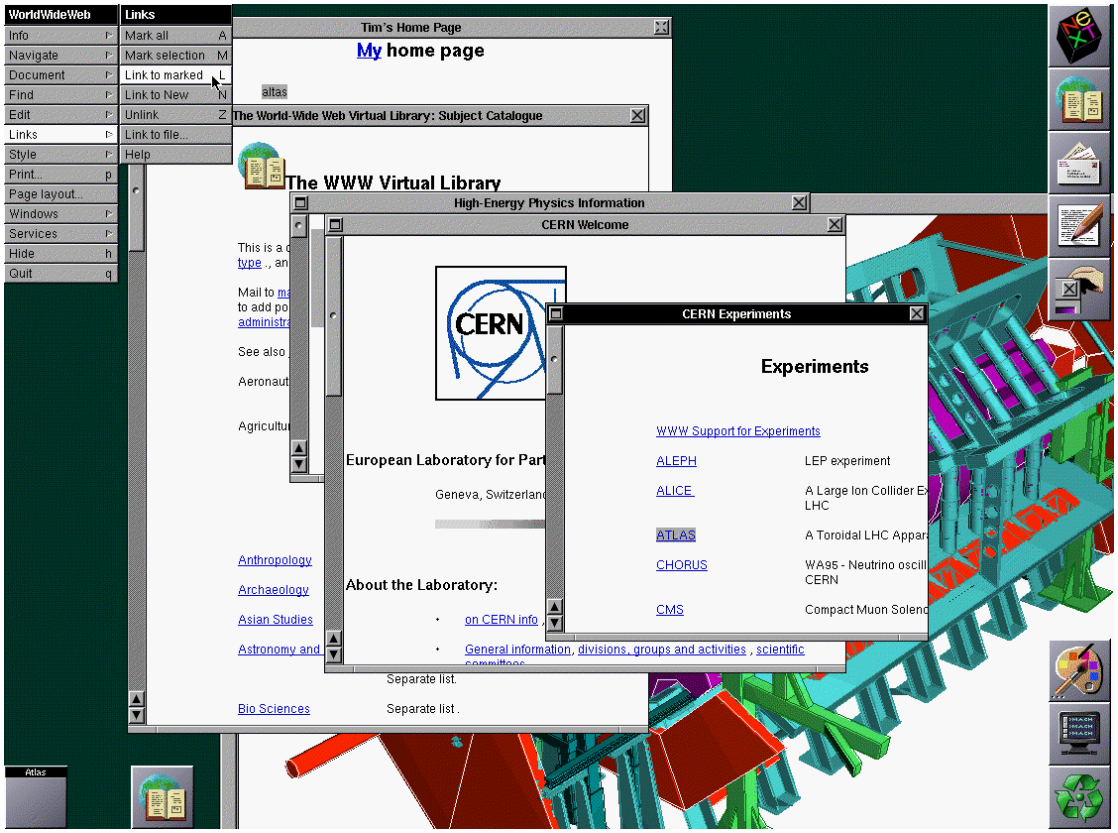
A Look into History: The Web



Tim Berners-Lee writes “Information Management: A proposal” at CERN

1989

A Look into History: The Web



First browser developed at CERN

1990

A Look into History: The Web

First paper appears on the project at Hypertext conference

→ Only accepted as a poster!

1971 1991

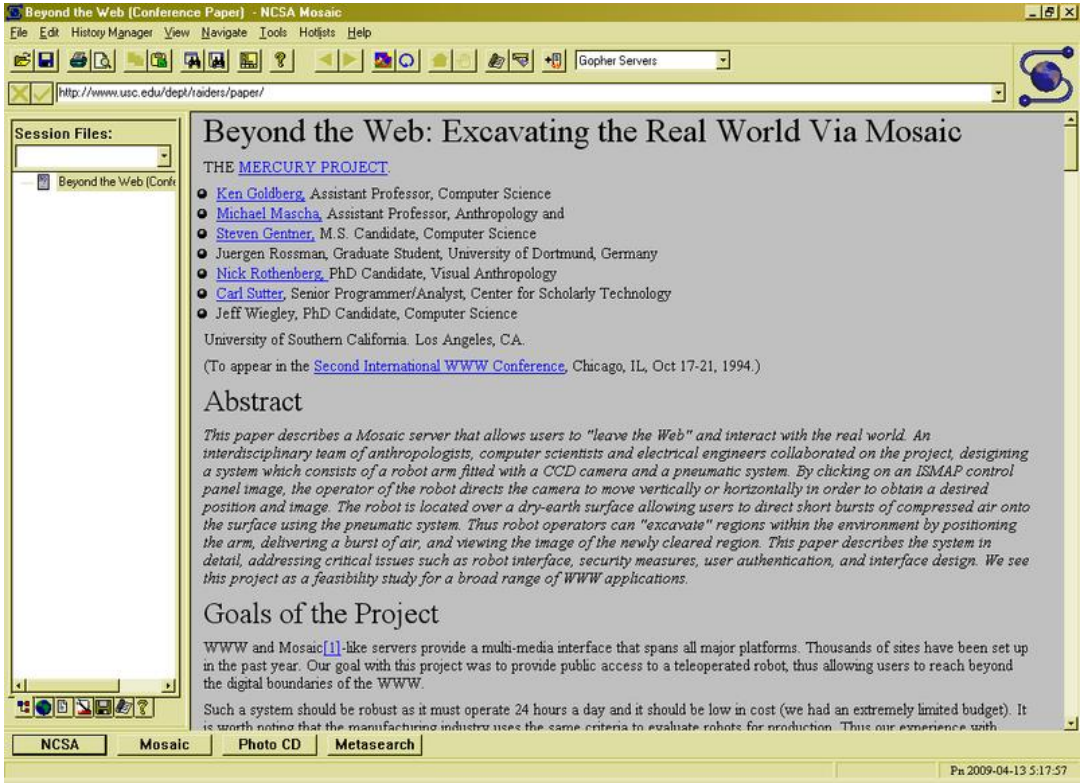
Humans interact with humans

E-Mail

WWW

Humans interact with machines

A Look into History: The Web



Mosaic became the first graphical browser

CERN agrees to allow public use of web protocol royalty-free!

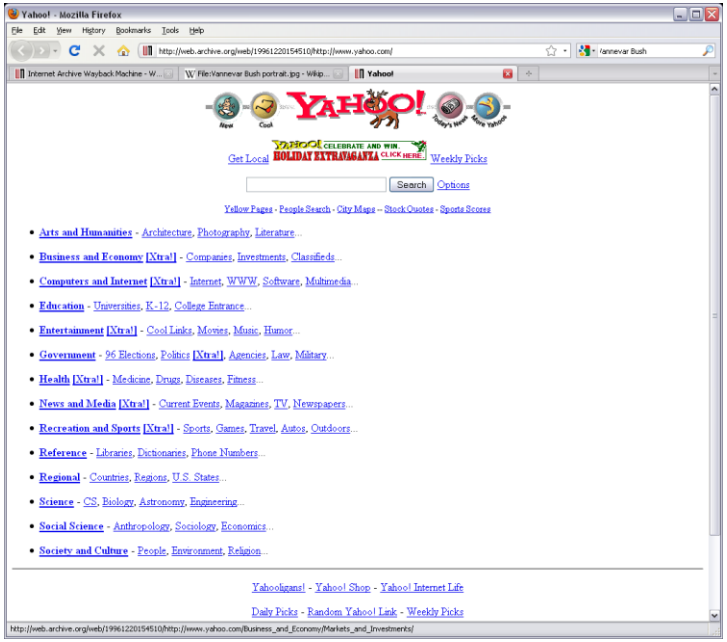
1993

A Look into History: The Web

→ Mosaic goes commercial

→ Traditional dialups (AOL, CompuServe, Prodigy) begin to sell Internet access

→ "Jerry's Guide to the world wide web" started …
it eventually became Yahoo

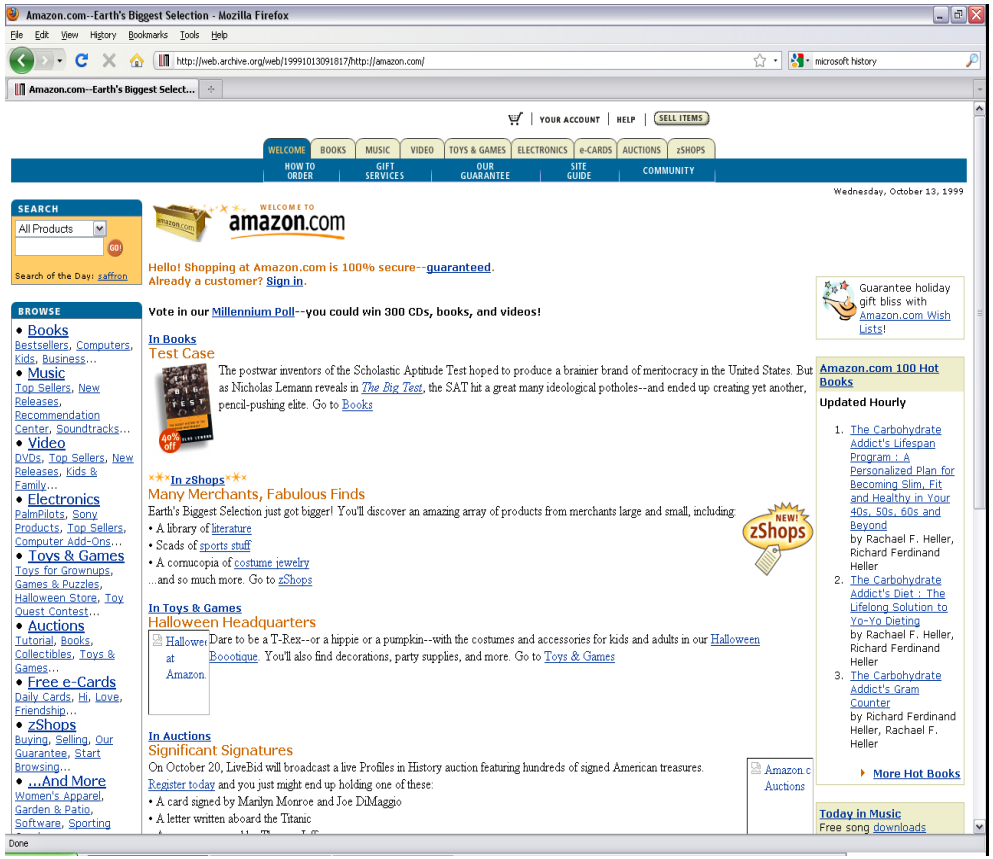


Yahoo
circa
1996

1994

A Look into History: The Web

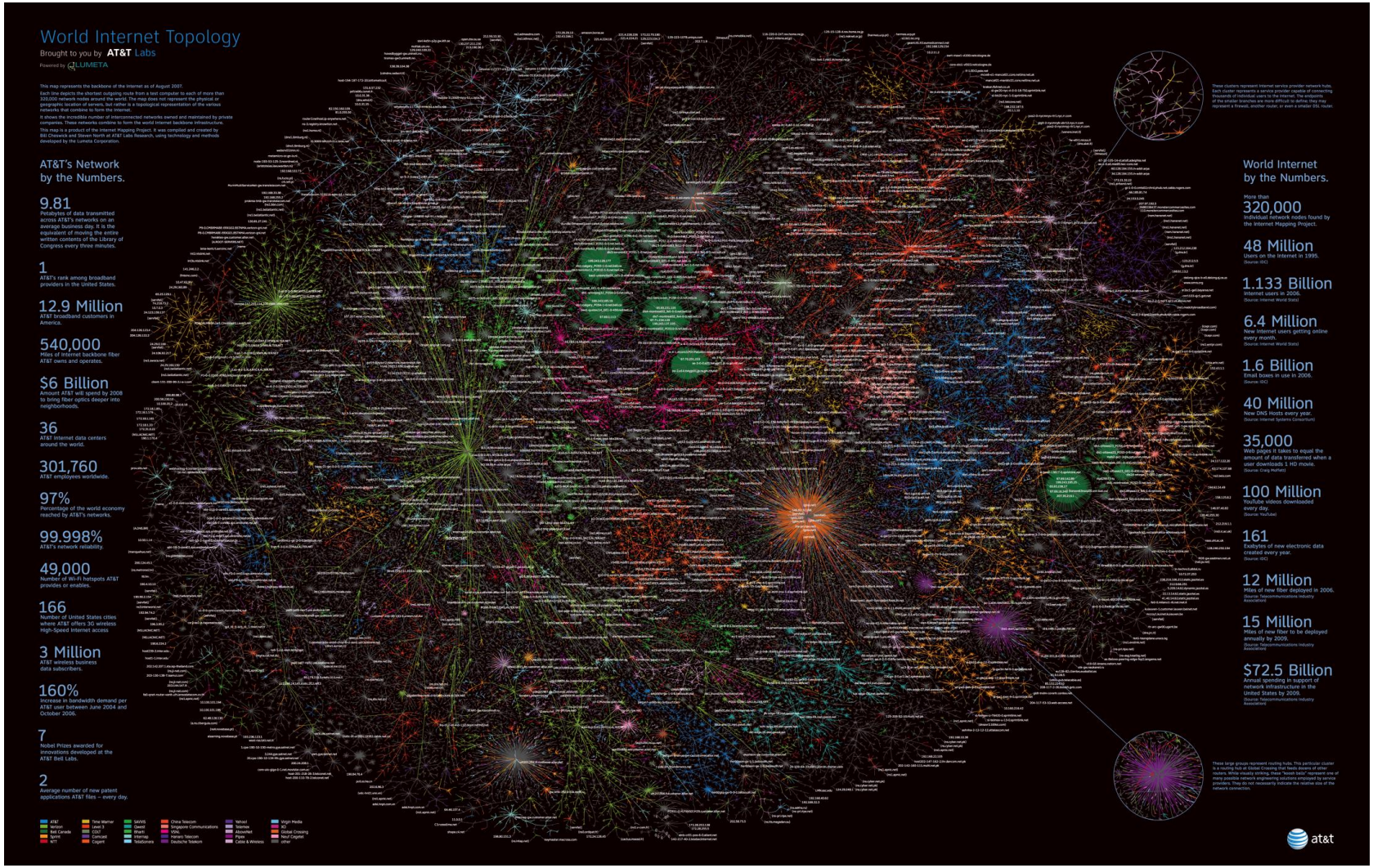
Amazon arrives and the commercialization of the web begins



Amazon circa 1999

1995+

A Look into History: Internet 2007 (only Backbone)



2007

A Look into History: Summary

Size per device

Number of Devices

Mainframe age (60's & 70's): One computer for many

PC age (80's & 90's): One computer for each, partially networked

Cloud computing
Mobile, ubiquitous computing (Today, > 2000): Many computers for each, networked

Communication Systems Today and in Future

Evolution of Communication Systems

Specialized Communication Systems

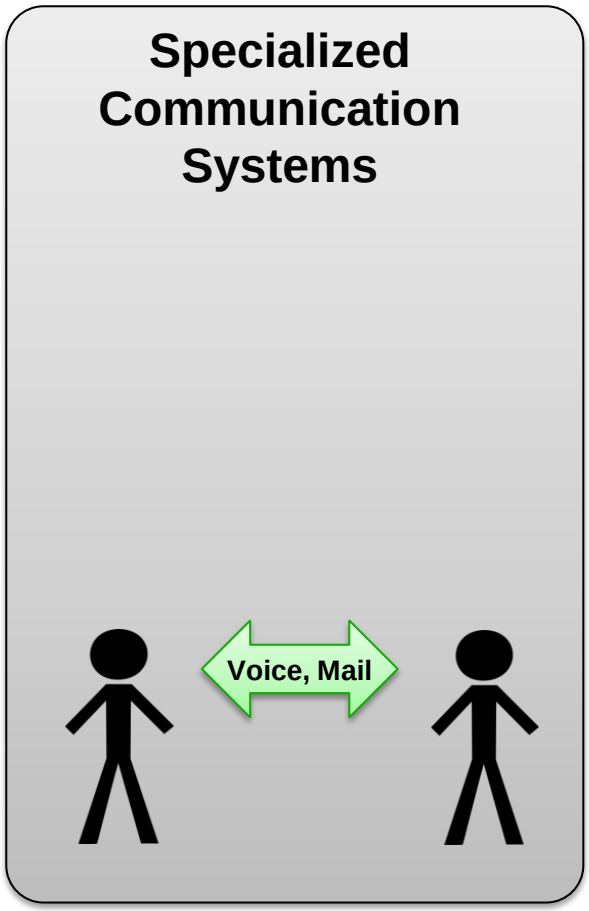
tomorrow

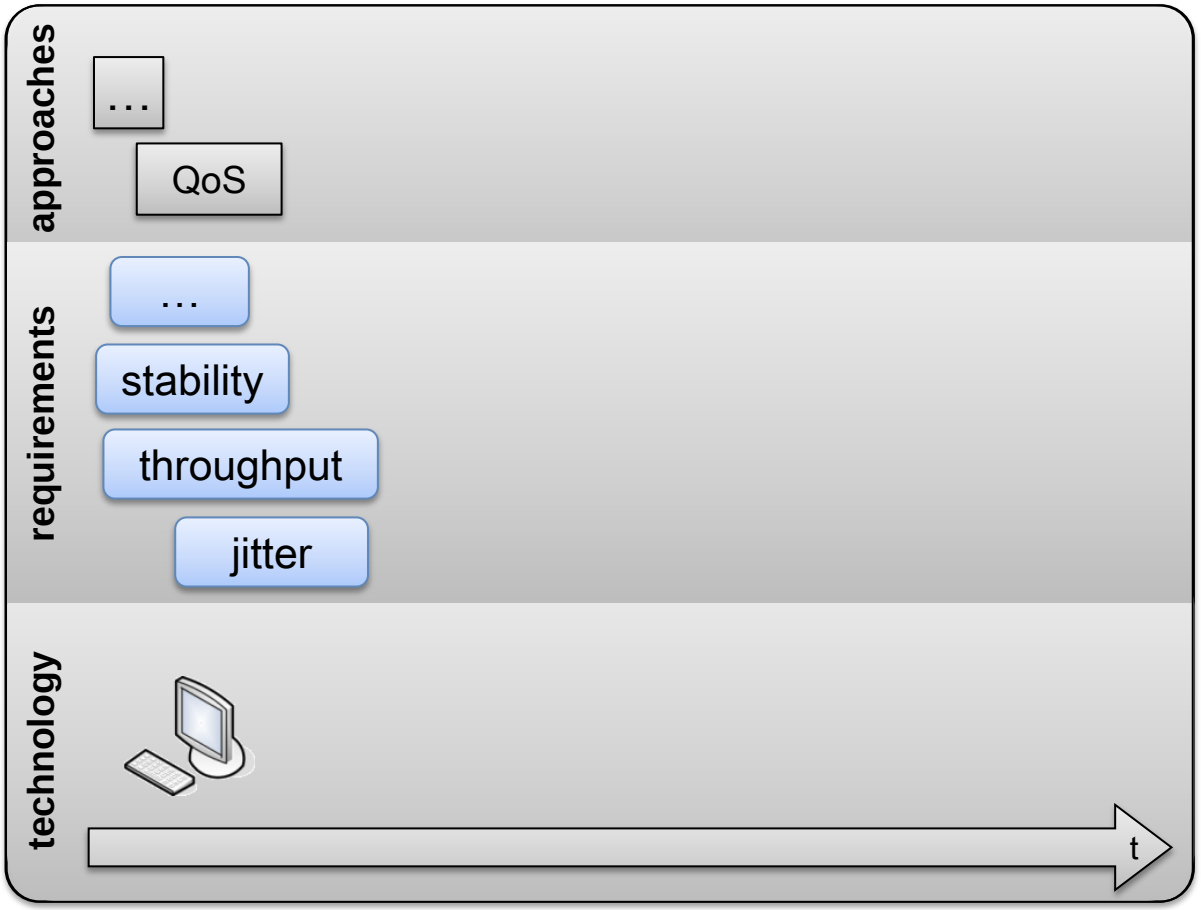
future

Communication Systems Today

Specialized Communication Systems

driven by technology and user demands

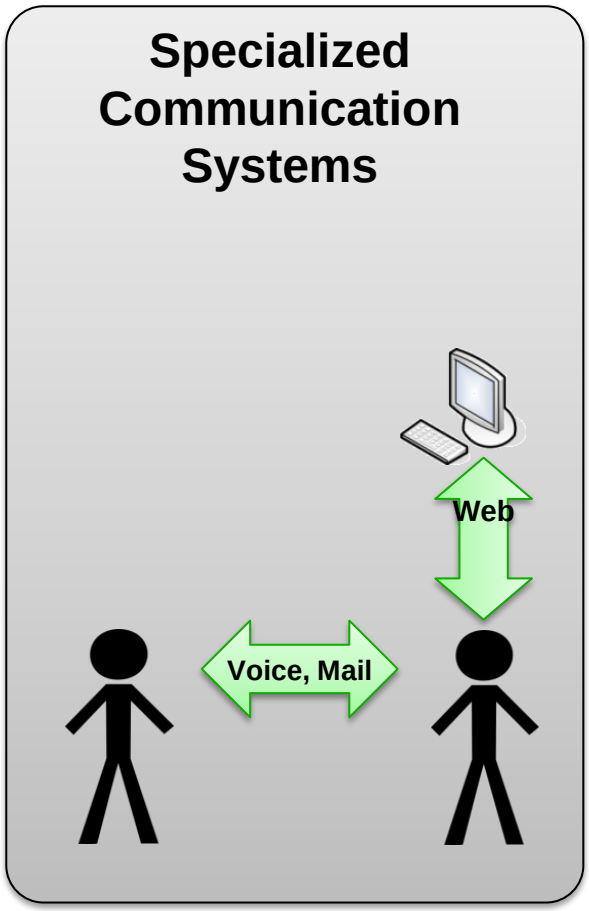


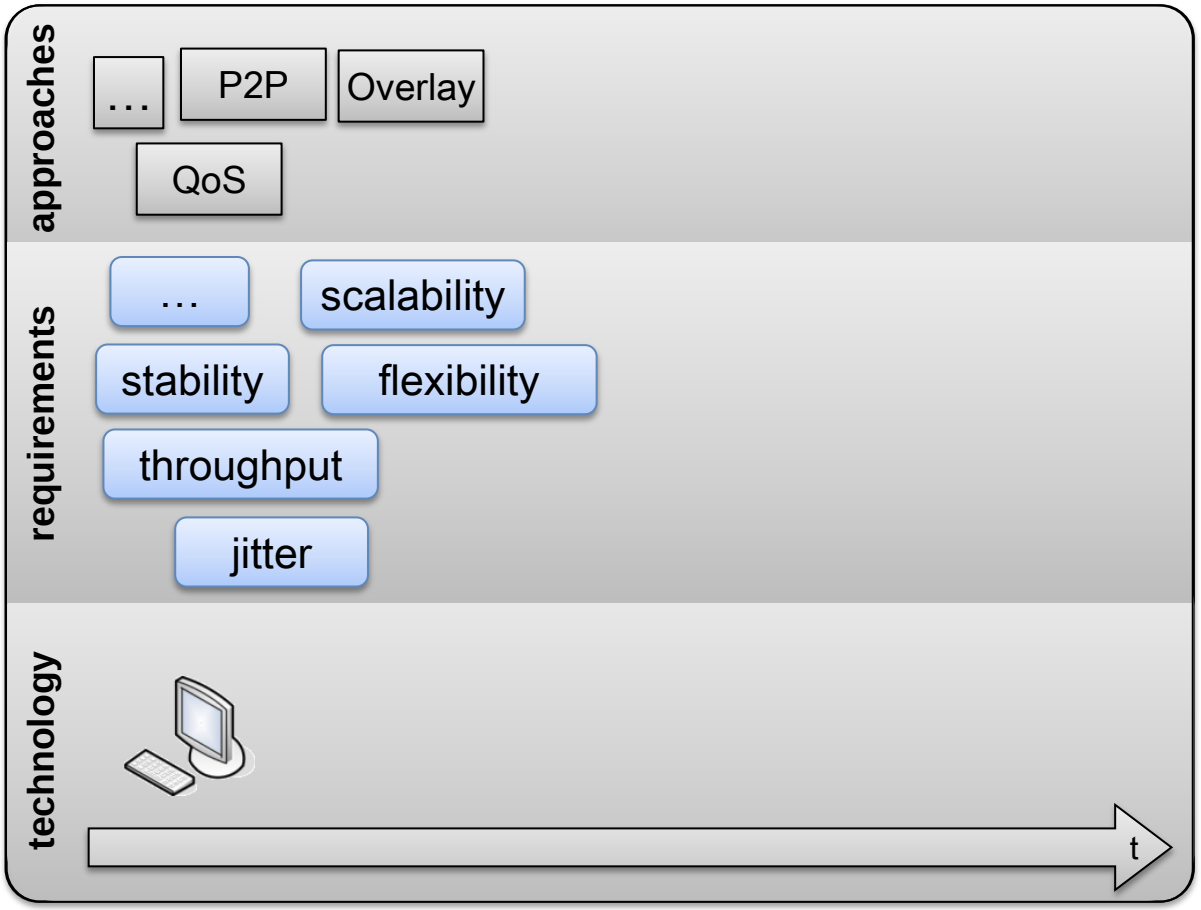


Communication Systems Today

Specialized Communication Systems

driven by technology and user demands

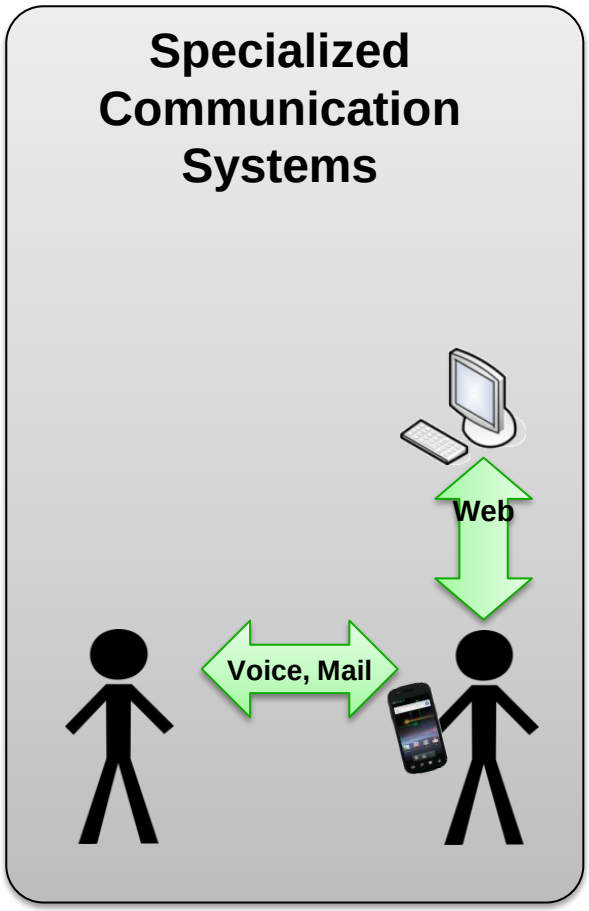


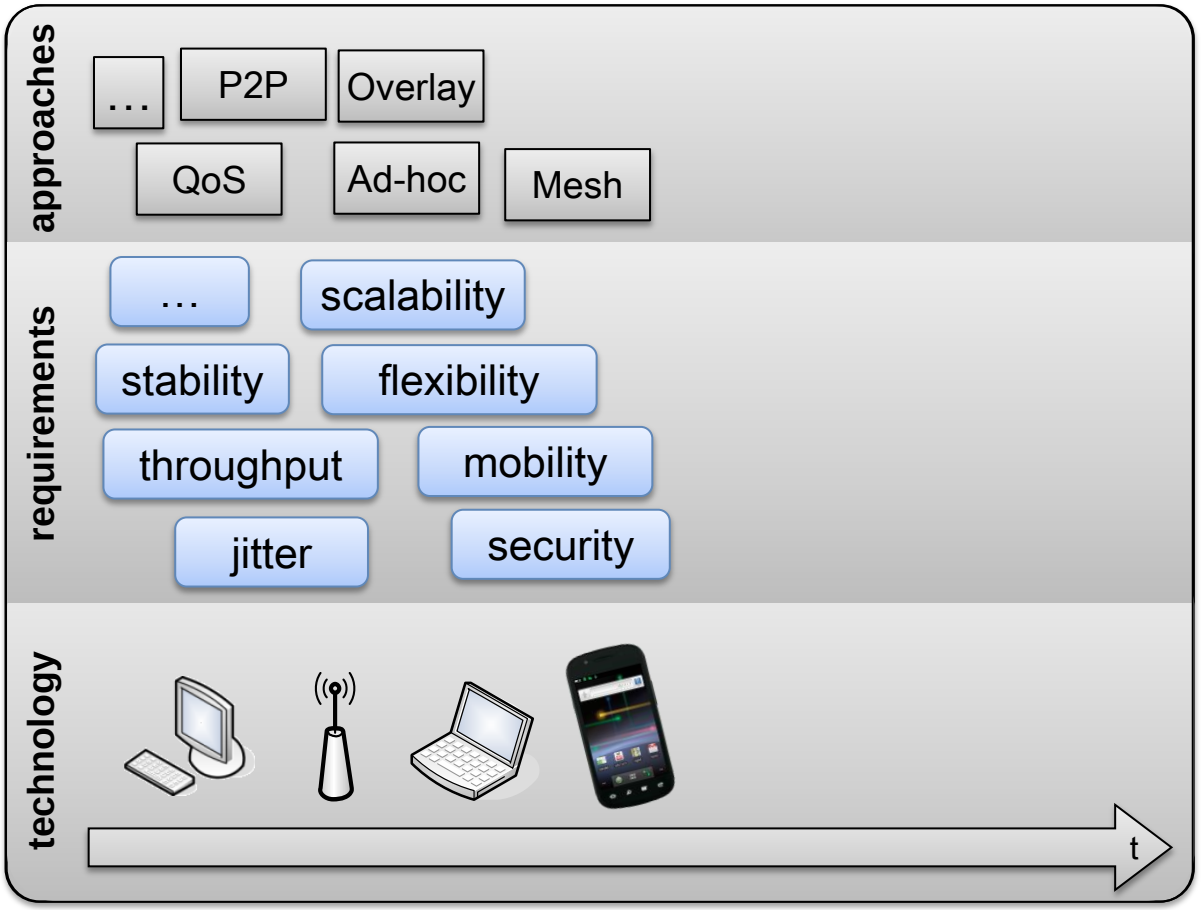


Communication Systems Today

Specialized Communication Systems

driven by technology and user demands

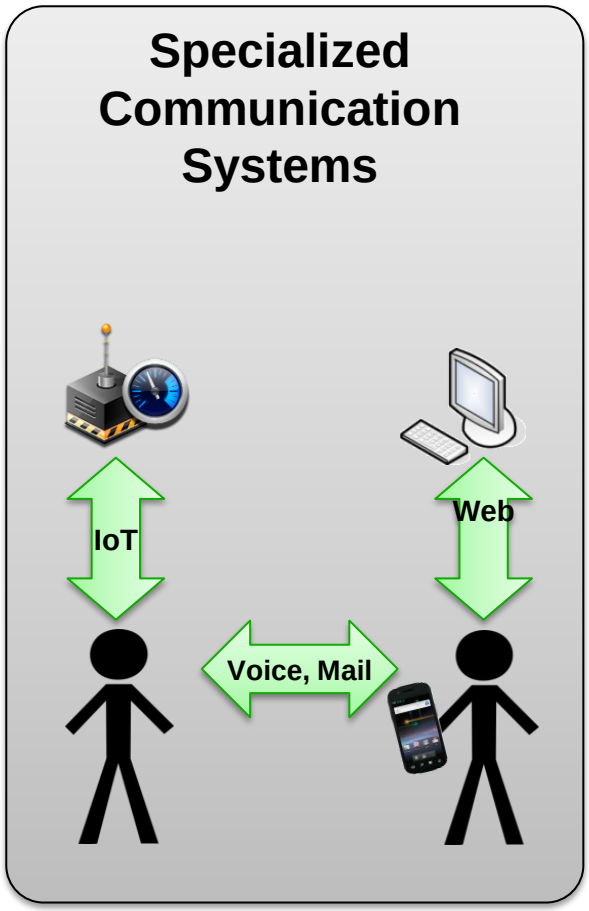


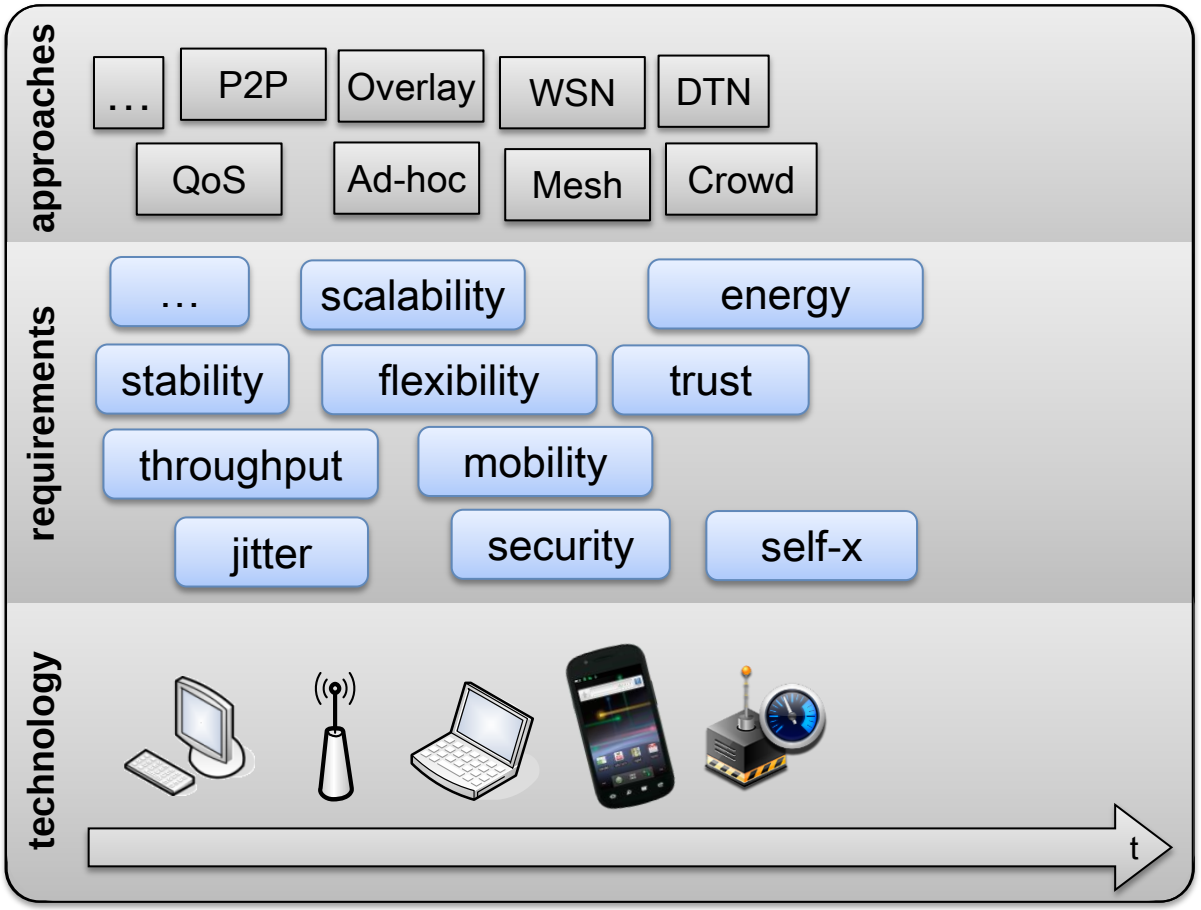


Communication Systems Today

Specialized Communication Systems

driven by technology and user demands

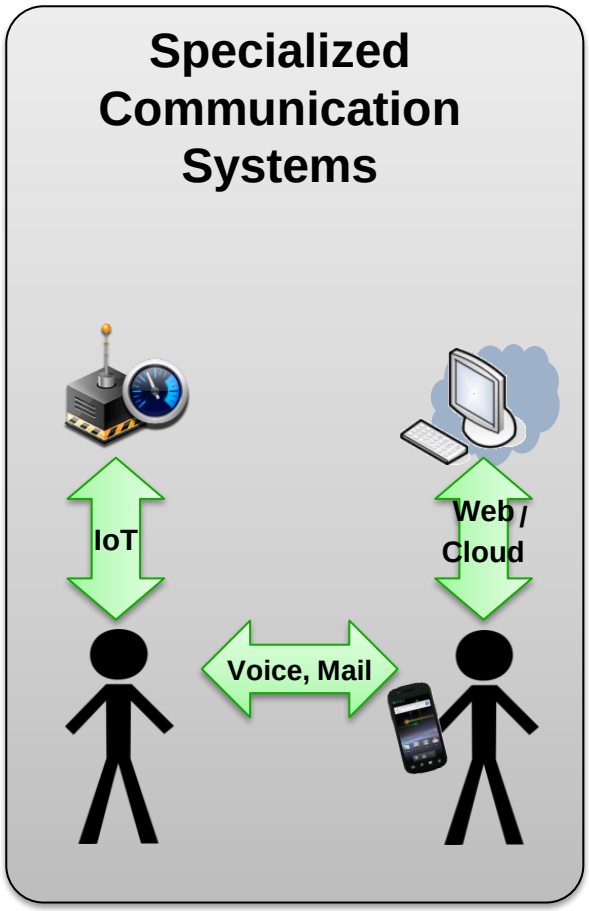


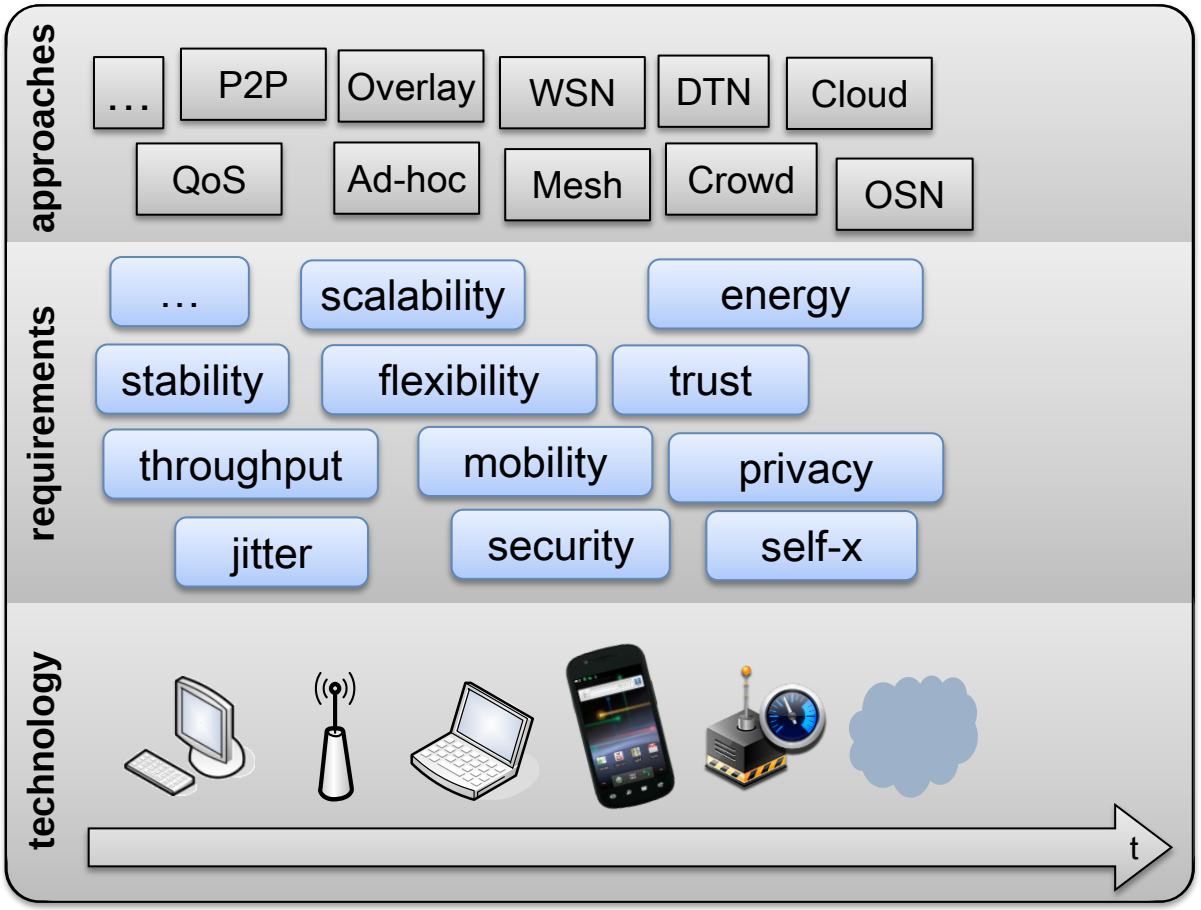


Communication Systems Today

Specialized Communication Systems

driven by technology and user demands

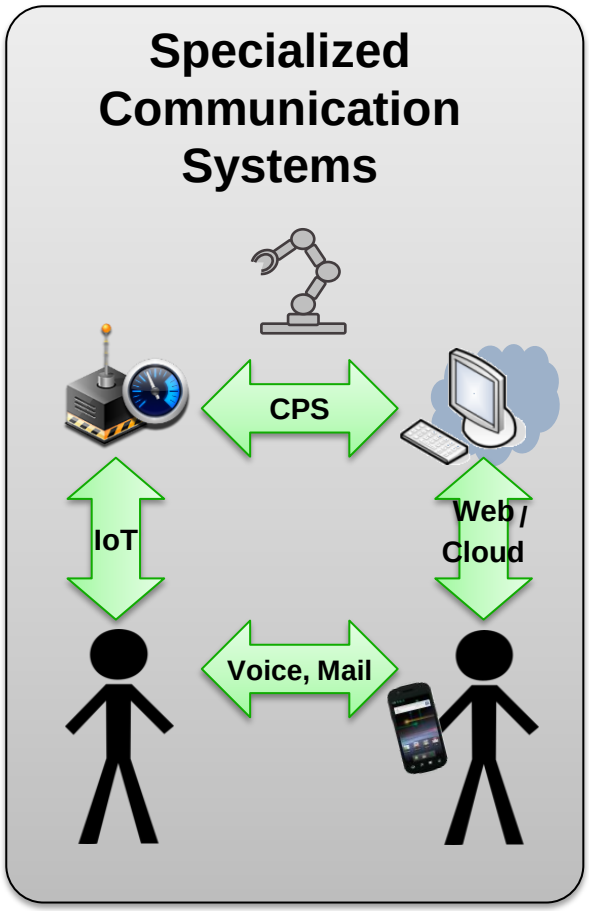


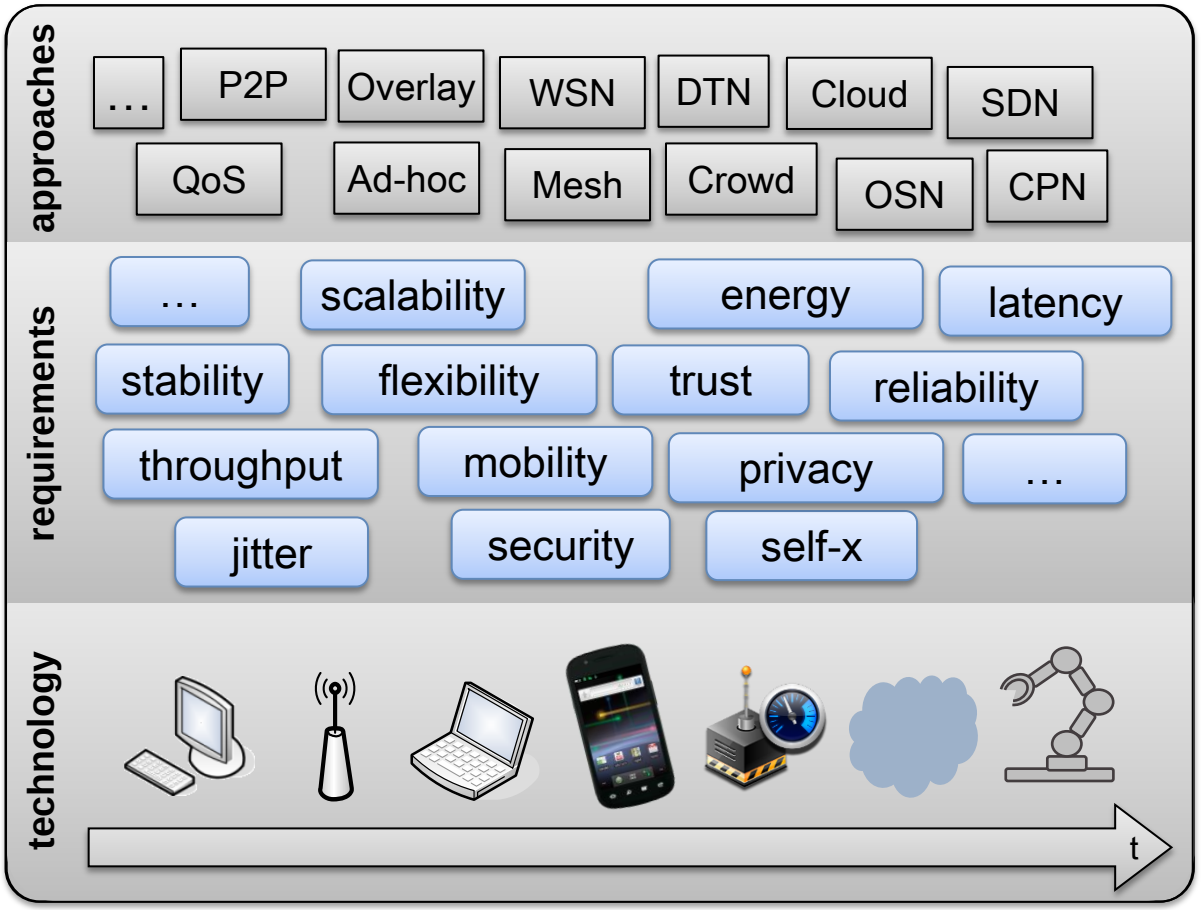


Communication Systems Today

Specialized Communication Systems

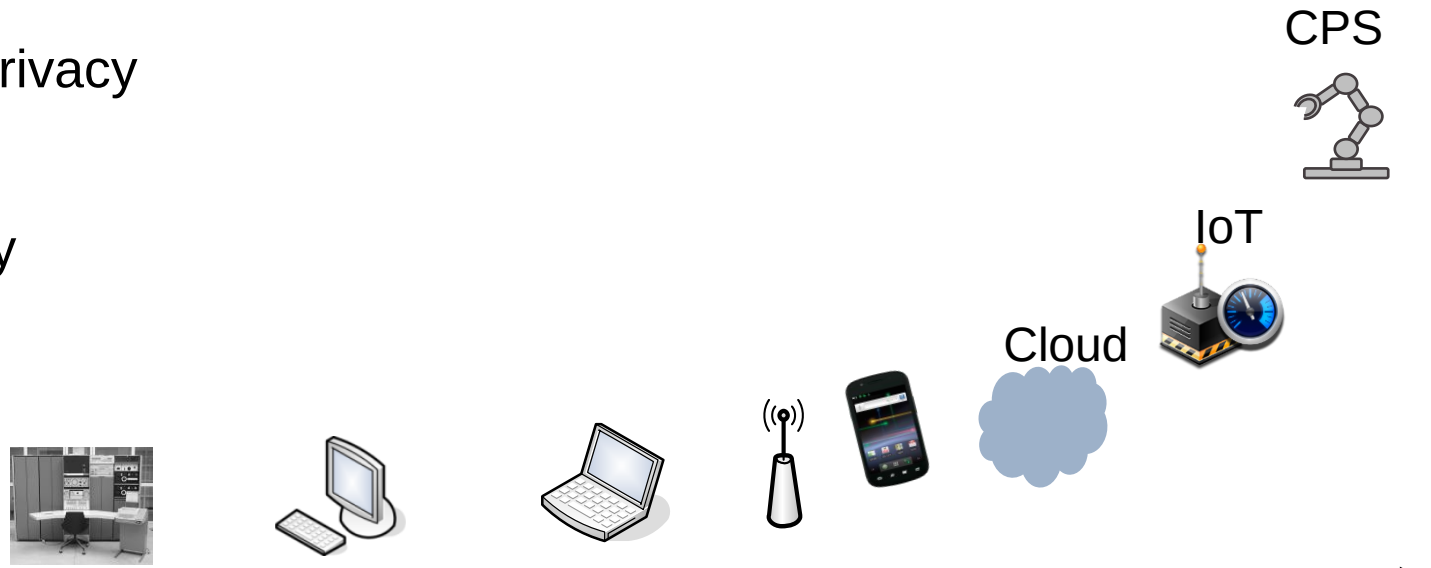
driven by technology and user demands





Specialized Communication Systems

- **Ongoing evolution:**
  - Continuous growth
  - Billions/Trillions of users and systems
- **Challenges**
  - Scalability
  - Security, Privacy
  - Energy
  - Adaptability
  - Reliability
  - …



- **Diese Vorlesung: *Grundlagen* der Datenkommunikation!**
  - Dienste, Protokolle und Referenzmodelle
  - Technische und nachrichtentechnische Grundlagen: Medien, Signale, Bandbreite, Leitungscodes und Modulation, Multiplexing
  - Lokale Netze: Strukturierung des Datenstroms, Fehlererkennung/-behebung, Flusssteuerung, Medienzugriff, Ethernet
  - Internet und Internet-Protokolle:
    - Vermittlungsschicht: IP, Routing
    - Transportschicht: TCP
  - Grundlagen der Sicherheit in/von Kommunikationsnetzen
    - Grundlagen: Verschlüsselung, Authentifizierung, Integrität
    - Sichere Internet-Protokolle
  - Anwendungsschicht

Viel Spaß in der Vorlesung! ☺

https://www.comsys.rwth-aachen.de

Datenkommunikation
Kapitel 1: Einführung

Klaus Wehrle

Communication and Distributed Systems
Chair of Computer Science 4
RWTH Aachen University

http://www.comsys.rwth-aachen.de

Kapitel 1: Einführung

- **Einführung und Begriffe**
  - Was ist Datenkommunikation?
  - Information, Daten, Signale
  - Netze
- **Allgemeine Grundlagen**
  - Dienste
  - Protokolle und Schichten
  - Kommunikationsarchitekturen

Was ist Datenkommunikation?

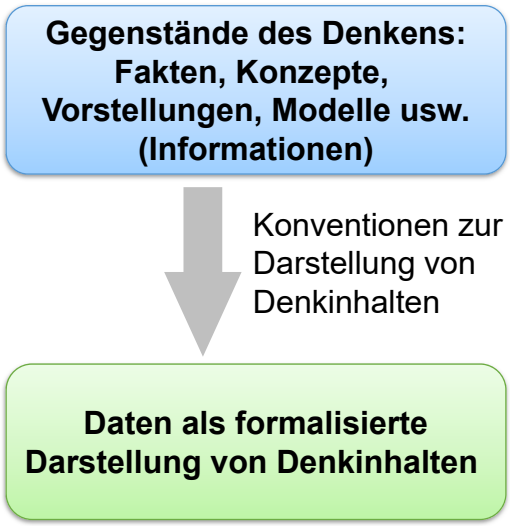
- **Datenkommunikation:**
  - Verarbeitung und Transport von digitalen Daten zwischen Computern und/oder anderen Geräten (i.A. über kleine oder größere Entfernungen)
  - Oder: Datenkommunikation ist der Oberbegriff für jeden Datenaustausch über *immaterielle Träger* und *Entfernungen* zwischen Menschen und/oder Maschinen
    - Immaterielle Träger:
      - Energieflüsse, meist elektrische Ströme, elektromagnetische Wellen
    - Gegensatz: materieller Datentransport (z.B. Brief, Versand von USB-Sticks)
    - Noch ein Gegensatz: Datenbanken, Dateien:
      Datentransport über die Zeit hinweg, der Ort bleibt gleich

Als Datenkommunikation bezeichnet man generell alle Methoden, die Nutzdaten von einer Quelle (Sender) hin zu einer Senke (Empfänger) übermitteln. Gemeinsam haben all diese Methoden, dass technisch gesehen der Sender zur Übermittlung der Daten eine physikalische Größe (z.B. elektrische Spannung) zeitlich variiert, und dass der Empfänger diese Variationen misst.

Datenaustausch zwischen Geräten kann auch z.B. über CDs oder USB-Stick erfolgen, aber dies ist keine Datenkommunikation mehr.

Der Begriff *Daten*

- **Daten (universell)**
  - *Darstellung von Sachverhalten (Fakten), Konzepten, Vorstellungen und Anweisungen in formalisierter Weise, die für die Kommunikation, Interpretation und die Verarbeitung durch Menschen und/oder technische Mittel geeignet ist*
  - Allgemeine Beispiele für Datendarstellungen:
    - Gesprochene Sprache
    - Zeichen-/Gebärden-Sprache
    - Geschriebene Sprache



Modell zur Erzeugung von Daten durch den Menschen

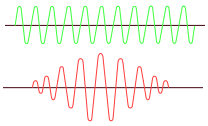
Was sind „Daten“ in der Datenkommunikation?

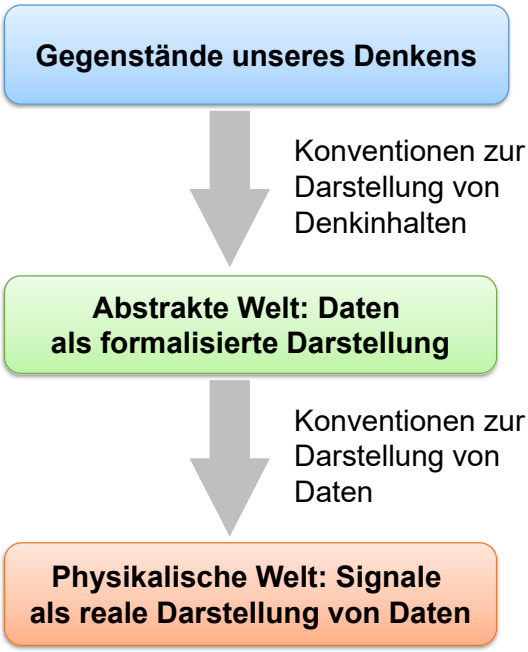
Ausgangspunkt bei der Datenkommunikation sind die unterschiedlichsten Gegenstände unseres Denkens. Dazu zählen Fakten, Konzepte, Vorstellungen und Modelle.

Bevor diese Gegenstände in irgendeiner Form übertragen werden können, müssen sie in einem ersten Schritt in eine formalisierte Darstellung transformiert werden, in die sogenannten *Daten*. Unter dem Begriff „Daten“ versteht man also die Darstellung von Sachverhalten (Fakten), Konzepten, Vorstellungen und Anweisungen in formalisierter Weise, die für die Kommunikation, Interpretation und die Verarbeitung durch Menschen und/oder technische Mittel geeignet ist. Beispiele für Daten sind z.B. Sätze in gesprochener Sprache, geschriebener Sprache, aber auch in Zeichen- und Gebärdensprache.

Der Begriff *Signal*

- **Signal**
  - *Physikalische Darstellung (Repräsentation) von Daten durch charakteristische räumliche und/oder zeitliche Veränderungen der Werte physikalischer Größen*
  - Reale physikalische Repräsentation abstrakter Darstellungen (der Daten)





COM SYS | RWTH AACHEN UNIVERSITY

I-5

Wichtig im Kontext der Vorlesung ist hingegen der Begriff „Signal“. Während Daten lediglich eine abstrakte Darstellung einer für den Menschen (meist) wesentlichen Information darstellen, ist es für die Datenkommunikation nötig, Daten über einen physikalischen Kanal zu transportieren (Kabel, Glasfaser, Funk, …). Die Daten müssen dazu in eine physikalische Form gebracht werden, die einen Transport über den zu verwendenden Kanal erlauben. Am Beispiel Kupferkabel: Daten können in Form von Strompulsen über das Kupferkabel transportiert werden. Durch Veränderung z.B. der Stärke des Stroms können unterschiedliche Daten (0 oder 1) repräsentiert werden.

Oft erfolgt eine Verwechslung (oder unsaubere Trennung) der Begriffe „Daten“ und „Signal“, da Daten nur in objektivierter, physikalisch dargestellter Form, d.h. als Signale, erfassbar, speicherfähig, übertragbar und verarbeitbar sind. Jede konkrete Datendarstellung ist somit mit einer spezifischen Signalrepräsentation verbunden, weswegen die konzeptionelle Unterscheidung oft nicht unmittelbar sichtbar ist. Beispiele wären:

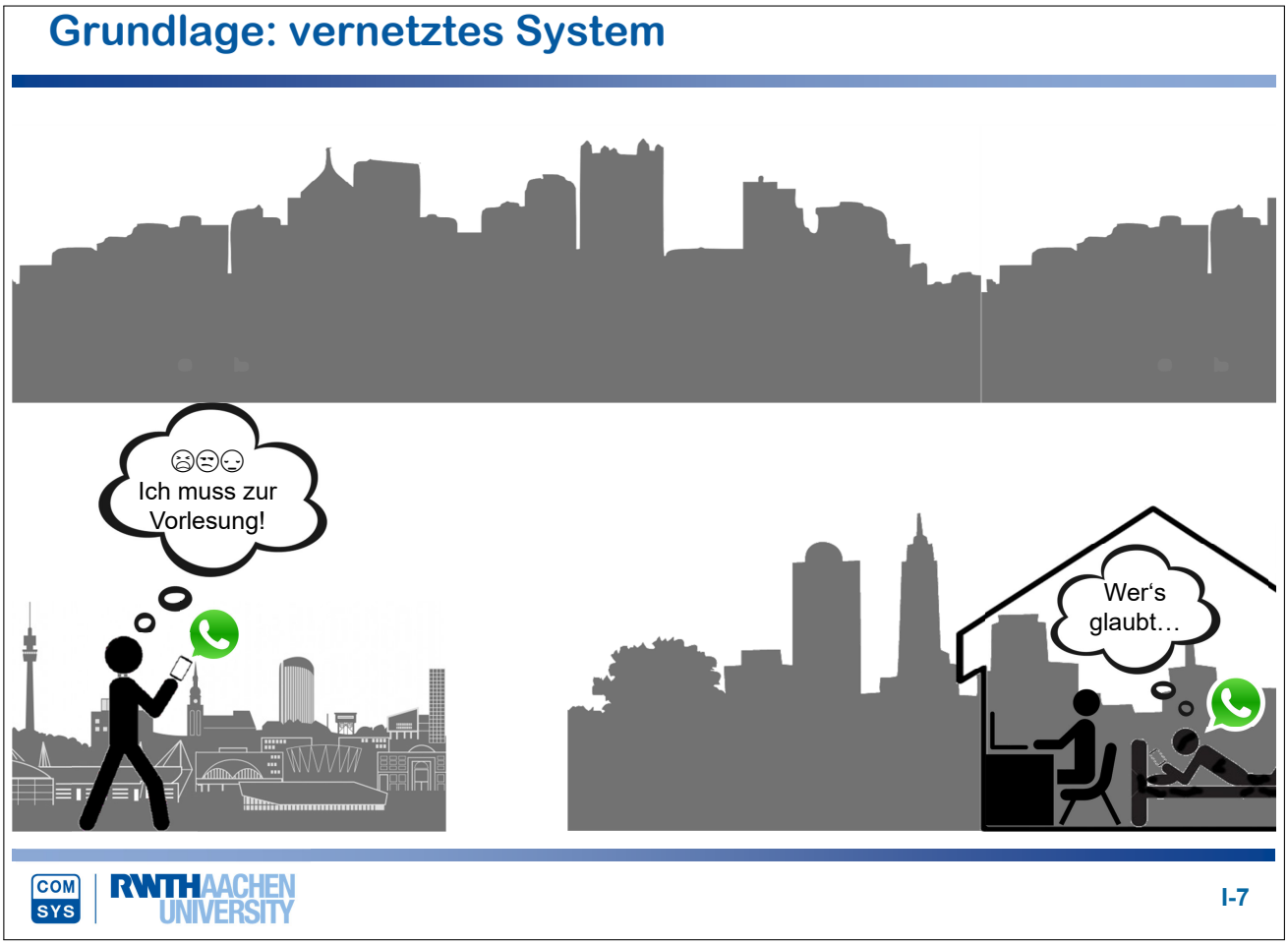
- Laute einer Sprache (Daten) beim Sprechen als akustische Schwingungen (Signale)
- Druckbuchstaben auf Papier als optische Signale abstrakter Schriftzeichen (Daten)
- Darstellung von Sprachlauten (Daten) durch elektrische Sprechströme (Signale)

Der Begriff *Information*

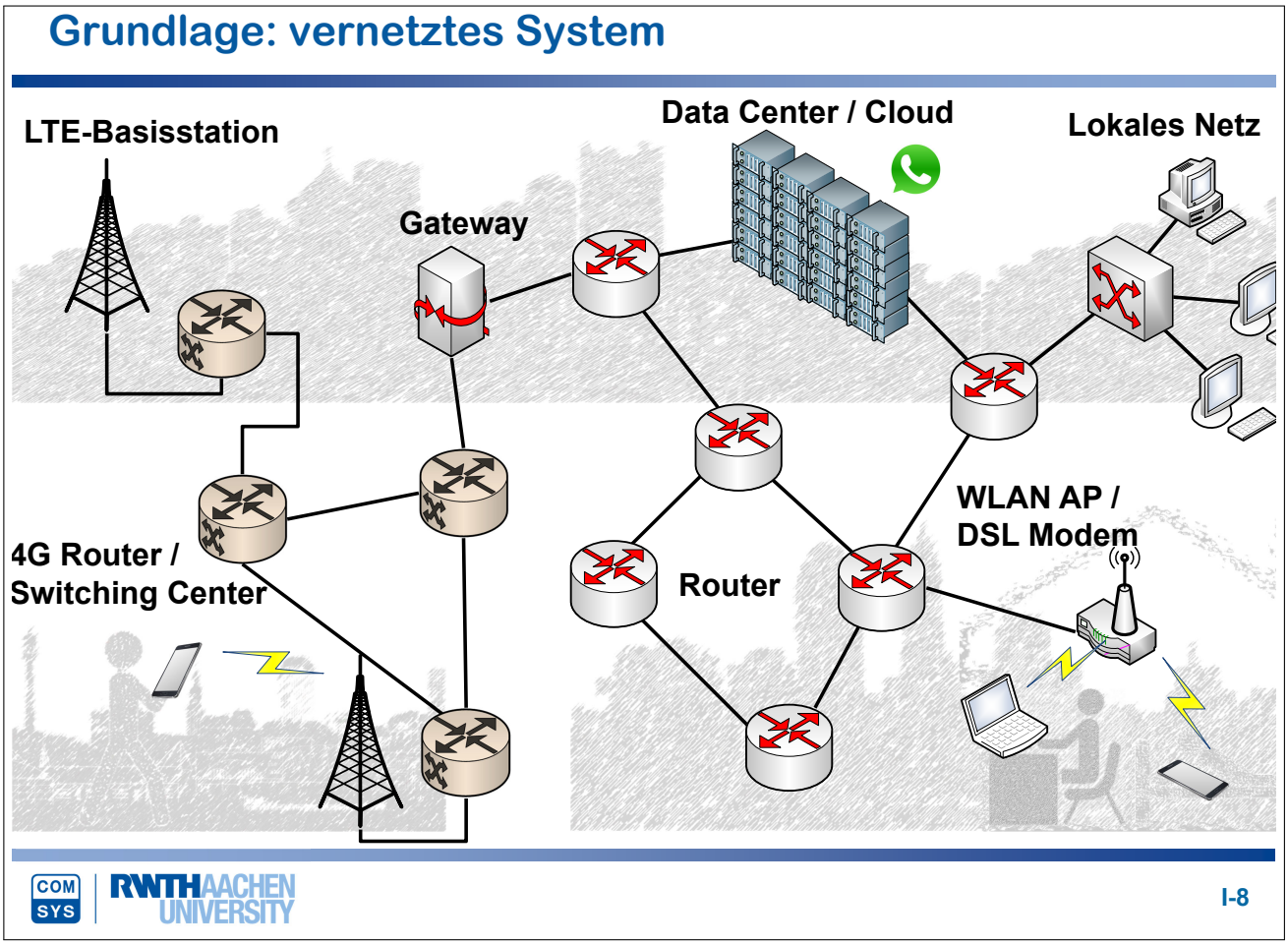
- **Information:**
  - *Bedeutung, die ein Mensch (bzw. bestimmte Anwendungen) den Daten aufgrund von Vereinbarungen (Konventionen), die ihnen zugrundeliegenden, zuordnen kann*
  - Informationsbegriff bezieht sich damit vorwiegend auf den Menschen
    - Enge Definition des Informationsbegriffs in der Vorlesung (verglichen mit der Alltagssprache)

  → Verwendung des Begriffs „Information“ bei präziser Ausdrucksweise in der Datenkommunikation möglichst vermeiden

Es ist Aufgabe des Menschen, Daten so zu interpretieren, dass daraus die übertragene *Information* gewonnen wird. An dieser Stelle wird somit die Unterscheidung zwischen Syntax und Semantik sichtbar: Zur Darstellung der Daten genügt eine Syntax, d.h. Regeln, wie gewisse Sachverhalte darzustellen sind, während die Semantik, also die Bedeutung dieser dargestellten Sachverhalte, erst durch die Interpretation durch den Menschen zustande kommt.



Ein alltägliches Szenario? Ermöglicht wird es durch die Datenkommunikation.



Grundlage für die Datenkommunikation ist ein vernetztes System, das sich im wesentlichen aus Endsystemen und Vermittlungseinrichtungen (Router, Switch) zusammensetzt. Endsysteme bieten dabei sowohl Personen den interaktiven Zugang zur Datenkommunikation als auch Softwaresystemen und Geräten wie beispielsweise Datenbanken oder Sensoren.

Bezüglich der Vernetzung von Endsystemen und Vermittlungseinrichtungen gibt es keine Einschränkungen – Verbindungen können über Kabel, Funk oder andere Techniken hergestellt werden.

Auf dieser Folie ist ein kleiner Ausschnitt der Kommunikationsmöglichkeiten gezeigt. Ein Smartphone is via LTE mit einem LTE-Provider verbunden, das andere in seinem lokalen Netz per Wi-Fi mit seinem DSL-Provider. Der LTE-Provider verfügt über sein eigenes Netz, einen sogenannten Backbone, der alle Funkzellen des Providernetzes miteinander verbindet und Daten bzw. Telefongespräche vermitteln kann. Der DSL-Provider ist über einen Backbone an andere Datennetze („das Internet“) angeschlossen. Zusätzlich zu sehen ist ein kabelgebundenes lokales Netz, das z.B. innerhalb einer Firma alle Rechner und Peripheriegeräte verbindet.

Bestimmte Exchange Points (hier Gateway genannt) verbinden LTE- und Datennetze miteinander. Angeschlossen ist hier auch ein Data Center, in dem z.B. die WhatsApp-Services laufen.

Kategorisierung von Netzen nach Ausdehnung

- **Personal Area Network (PAN)**
  - Vernetzung von persönlichen Geräten eines Benutzers
- **Local Area Network (LAN)**
  - Heimnetz, Firmennetz
- **Metropolitan Area Network (MAN)**
  - Campus, Stadtbereich
- **Wide Area Networks**
  - Länder, Kontinente
- **Internet**
  - Weltumfassender Zusammenschluss aller Netze

Das bekannteste Netz weltweit ist das Internet. Das Internet selbst ist allerdings kein eigenständiges Netz, sondern beschreibt die Kopplung der unterschiedlichsten Netztypen zu einem weltweiten Kommunikationsverbund. Eine erste Einteilung dieser verschiedenartigen Netze lässt sich dabei anhand ihrer Ausdehnung vornehmen:

• PAN *(Personal Area Network)*: Netz wenige Meter um den Benutzer herum. Beispiel Bluetooth – Kopplung eines drahtlosen Headsets mit dem Handy.

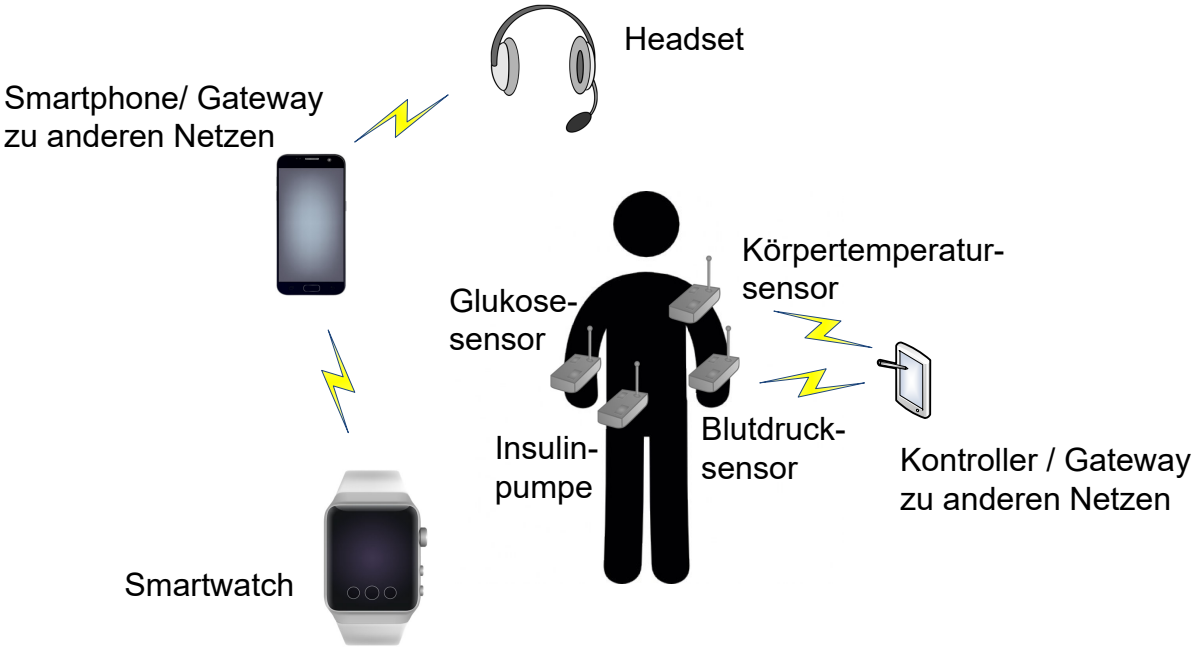
• LAN *(Local Area Network)* : lokales Netz einer Organisation, welches schnellen Datenaustausch innerhalb der Organisation und gemeinsame Nutzung von Ressourcen (Drucker, Dateiserver, …) erlaubt. Beispiel: WLAN in der eigenen Wohnung.

• MAN *(Metropolitean Area Network)*: Stadtnetz, welches die lokalen Netze innerhalb eines geographischen Bereichs koppeln soll. Beispiel: RWTH-Backbone zur Kopplung aller lokalen Netze der Institute/Lehrstühle der Uni.

• WAN *(Wide Area Network)*: Weitverkehrsnetz zur Kopplung lokaler und/oder Stadtnetze innerhalb eines ganzen Landes oder gar Kontinents. Beispiel: X-WIN des DFN (deutsches Forschungsnetz) zur Kopplung aller Universitäten und Forschungseinrichtungen.

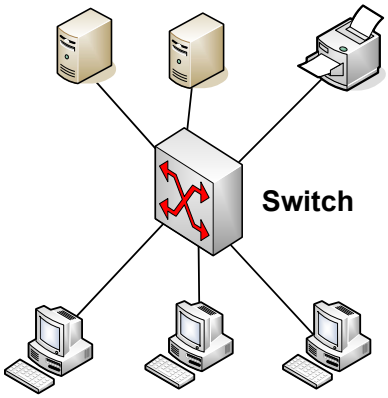
Wir werden später sehen, dass die Flexibilität, praktisch beliebige Netze und damit verknüpfte Übertragungstechniken in einem Netz, dem Internet, zusammenzufassen, dadurch erreicht wird, dass die Kopplung dieser Netze auf einer Protokollebene erfolgt, die oberhalb der Festlegungen von Übertragungsspezifika liegt.

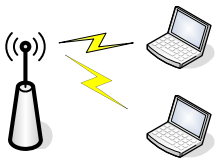
Personal Area Network (und Body Area Network)

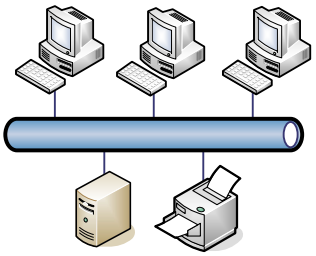


Spezialfall Body Area Network: Vernetzung verschiedener Geräte, die den menschlichen Körper überwachen/kontrollieren.

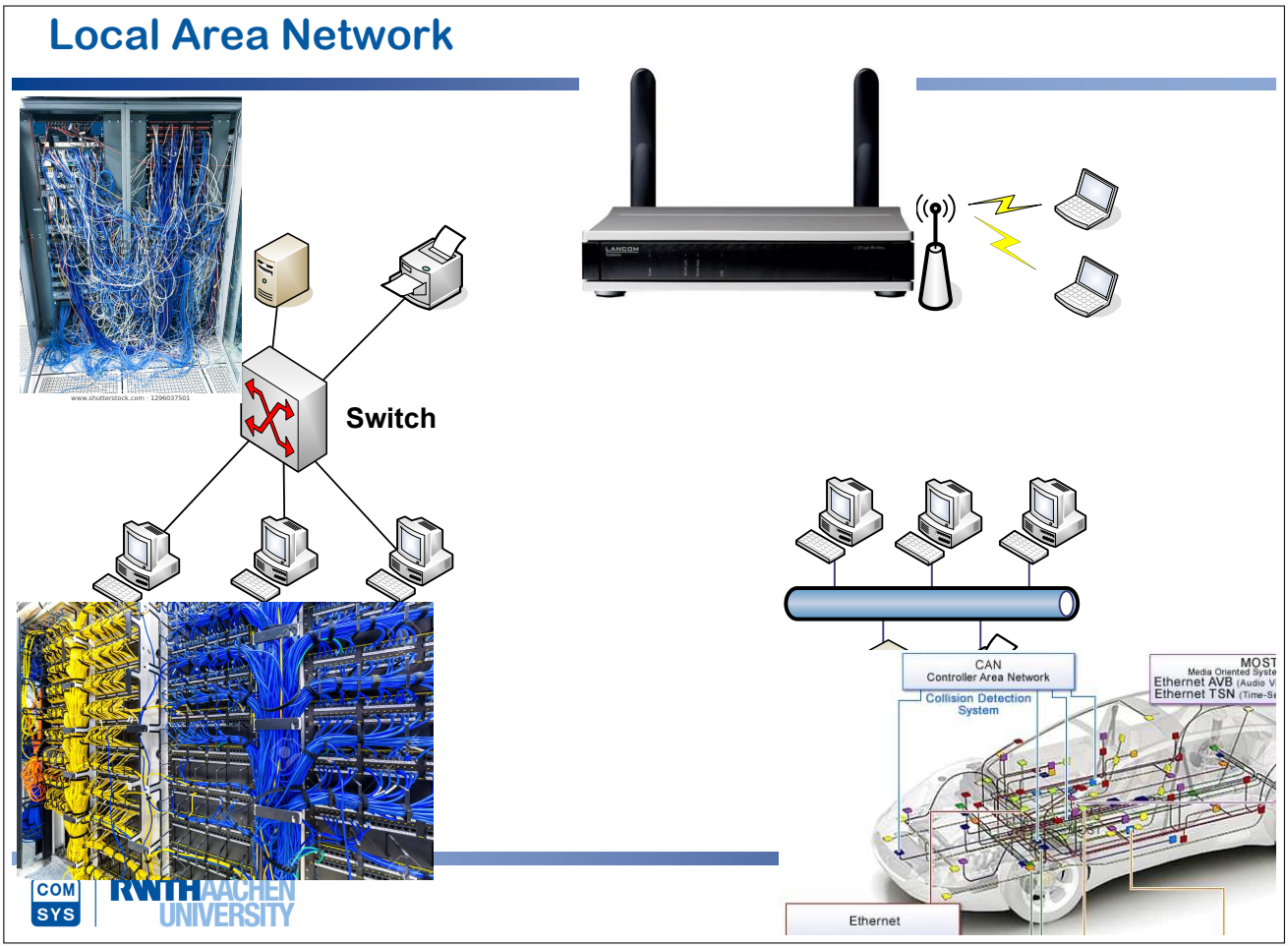
Local Area Network





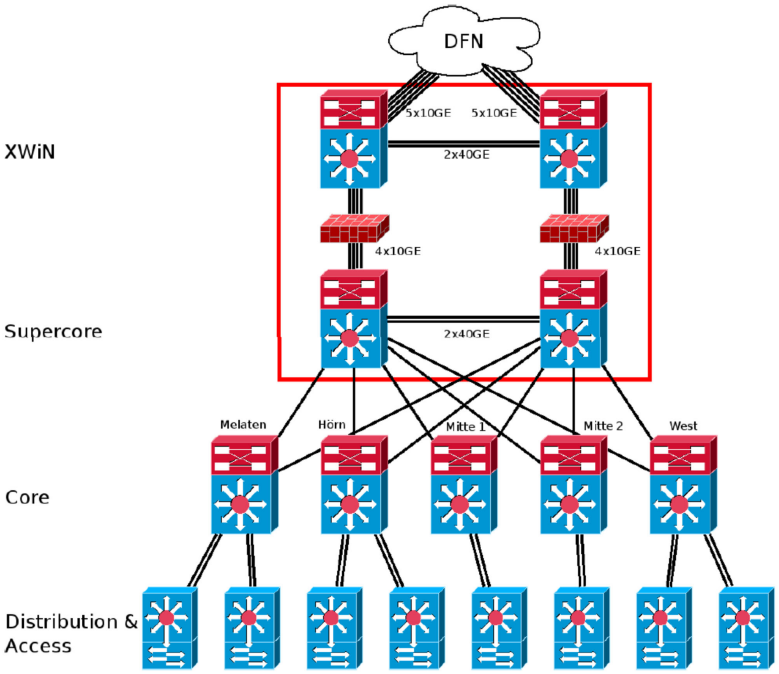


Es gibt unterschiedliche Möglichkeiten, Rechner per Kabel oder Funk zu einem lokalen Netz zu verbinden. Dies wird in Kapitel 3 näher betrachtet.

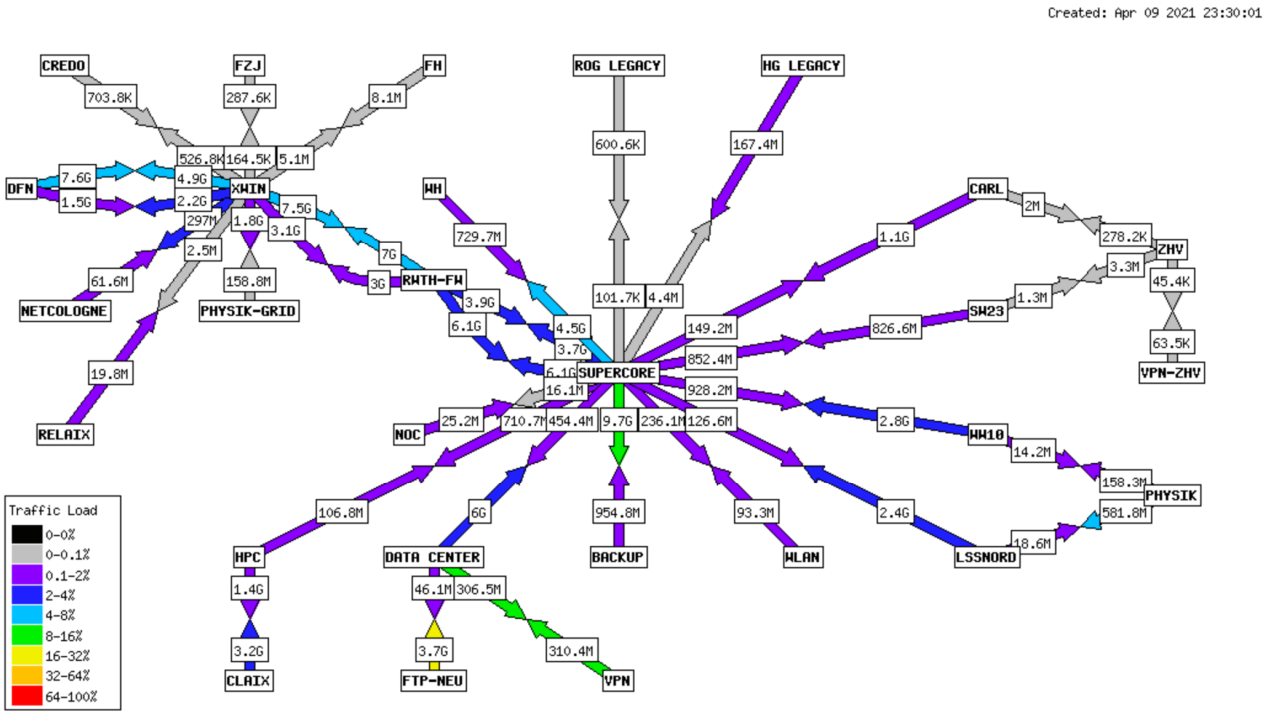


Es gibt unterschiedliche Möglichkeiten, Rechner per Kabel oder Funk zu einem lokalen Netz zu verbinden. Dies wird in Kapitel 3 näher betrachtet.

Metropolitan Area Network – RWTH-Netz



RWTH Weather Map

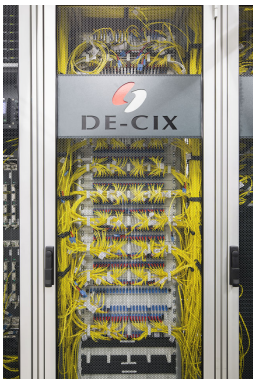


Deutsches Forschungsnetz: X-WIN

X-WiN-Topologie: Glasfasern

Glasfaser Bestand
Kernnetzknoten Bestand

Zentraler Knoten Frankfurt – Verbindung zum Europäischen Forschungsnetz

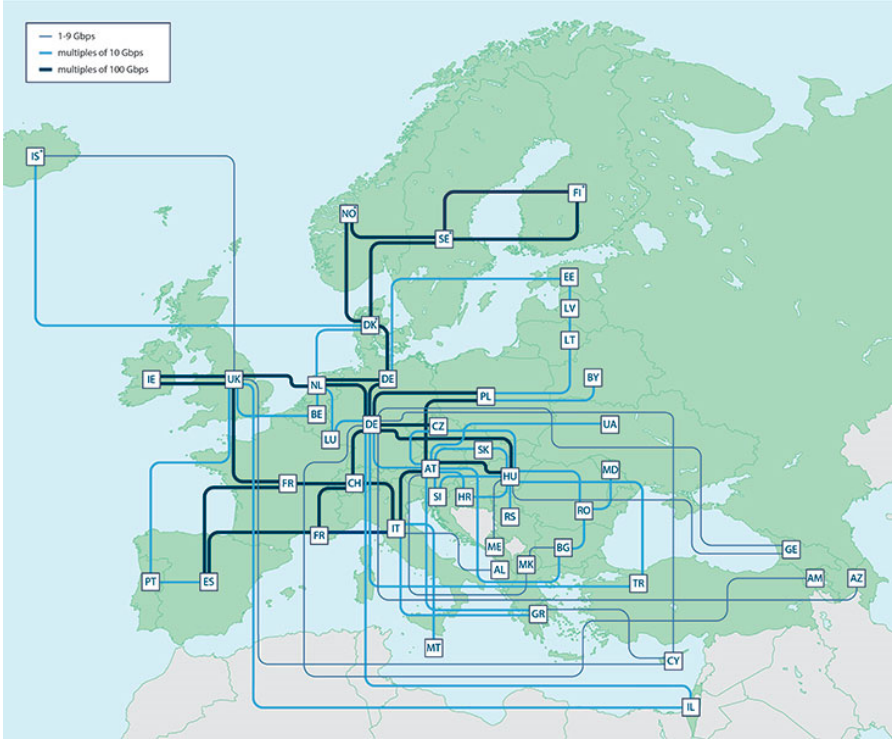


Stand: Oktober 2018

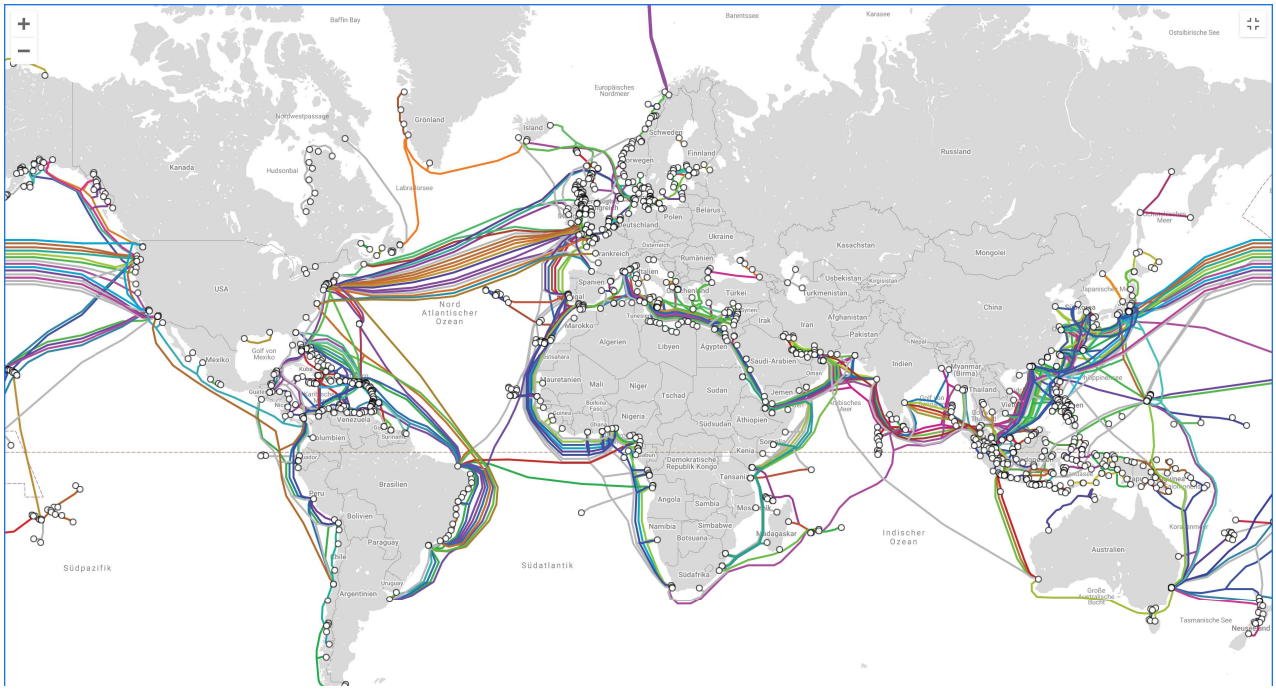
Netzwerke (Beispiel Géant)

Zentraler Knoten Frankfurt – Verbindung zum Europäischen Forschungsnetz.

In Frankfurt und Hamburg befinden sich die interkontinentalen Verbindungen.



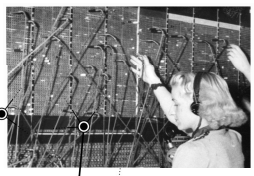
Verbindungen weltweit

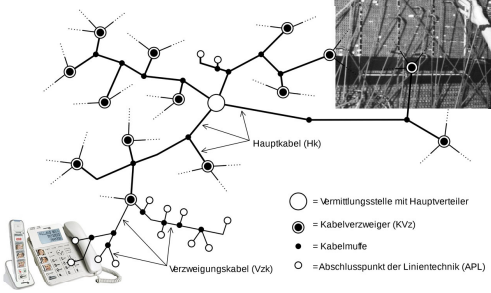


https://www.submarinecablemap.com/#/

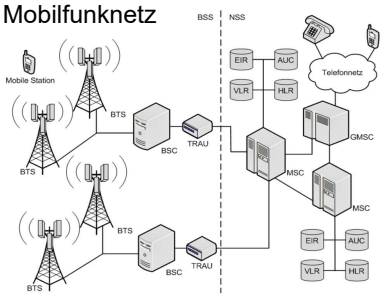
Weitere Netze...

Telefonnetz

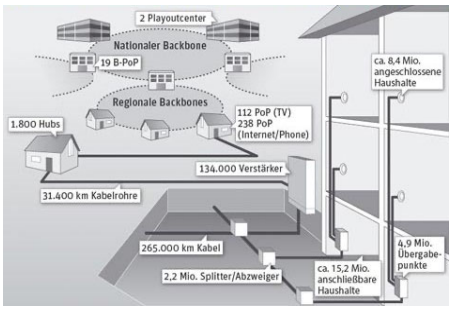




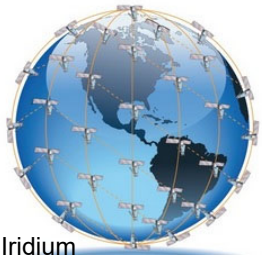
Mobilfunknetz



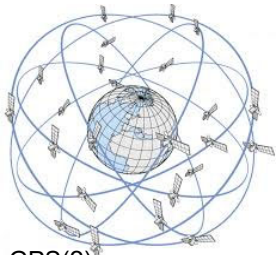
Kabel(TV)Netz



Satellitennetze



Iridium



GPS(?)

+ Radio, TV, TMC, Alarm/Warn/Melde/Leitsysteme (Straßen/ Bahn/Flugverkehr, Industrie, Tsunamie, Pandemien (?!?), uvm)

Quellen: Wikipedia

Datenkommunikation = Rechnernetze?

- **Datenkommunikation**
  - … setzt die Vernetzung von Geräten voraus (Rechnernetze)
  - … behandelt die Übertragung von Daten als Signale
  - … ?

Zur Datenkommunikation ist es nötig, ein vernetztes System vorliegen zu haben. Abhängig von den verwendeten Kommunikationsmedien zur Verbindung der Geräte können die Daten zur Übertragung als konkrete physikalische Signale dargestellt werden.

Bei der Behandlung der Datenkommunikation müssen also Kommunikationsmedien und Signale behandelt werden. Aber: was umfasst die Datenkommunikation noch?

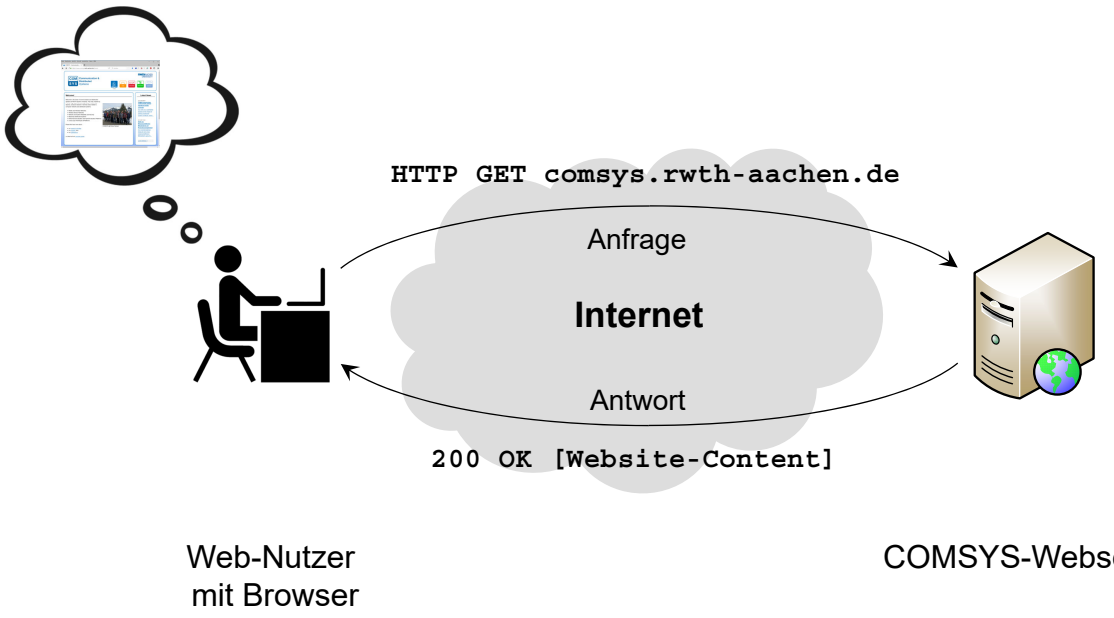
Aufgaben eines Kommunikationssystems

- **Ein Netzwerk ist mehr als nur das verbindende „Kabel“**
  - Wie werden die Daten als Signal dargestellt?
  - Welches angeschlossene System darf wann senden?
  - Wie können Übertragungsfehler erkannt werden?
  - Wie werden Endpunkte gefunden/adressiert?
  - Wie können entfernte Systeme in anderen Netzen erreicht werden?
  - Wie wird Datenstau in einem komplexen Netz vermieden?
  - Wie können Ressourcen ohne zentrale Kontrolle fair genutzt werden?
  - Wie werden die Daten bei den End-Systemen übermittelt?
  - Welche Kodierungsregeln und Semantiken gibt es hierbei?
  - Wie können Nachrichten sicher versendet werden?
  - Was bedeutet „sicher“?

Bei genauerer Betrachtung fällt eine Vielzahl von Aufgaben an, die ein Kommunikationssystem erbringen muss. Es ist nicht damit getan, Daten in Signale zu konvertieren, diese zu übertragen und auf Empfängerseite wieder in Daten zurückzuübersetzen. Es ist eine Vielzahl weiterer Aufgaben notwendig, z.B. die Erkennung von Übertragungsfehlern, die das Medium verursachen kann, eine Zugriffsregelung auf das Medium, Entscheidungen über die Wegwahl zur Weiterleitung von Daten über Zwischenknoten hinweg usw.

- **Einführung und Begriffe**
  - Was ist Datenkommunikation?
  - Information, Daten, Signale
  - Netze
- **Allgemeine Grundlagen**
  - Dienste
  - Protokolle und Schichten
  - Kommunikationsarchitekturen

Beispiel: Informationsabruf im WWW



COM SYS | RWTH AACHEN UNIVERSITY

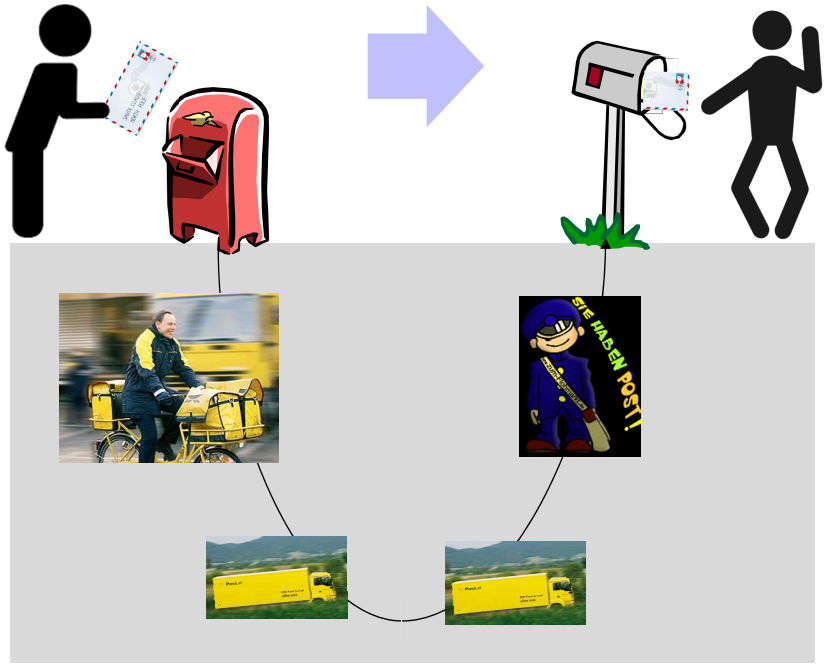
I-22

Generell stellen Kommunikationsnetze Dienste zur Verfügung – sie bieten den Dienst an, Daten zu übertragen. Alle, die die Datenübertragung in Anspruch nehmen, sind Dienstnutzer (auch: Dienstnehmer). Ein Beispiel ist der Zugriff auf die COMSYS-Webseite: der Benutzer (Browser) und der Webserver, der die Webseite hostet, sind Dienstnehmer.

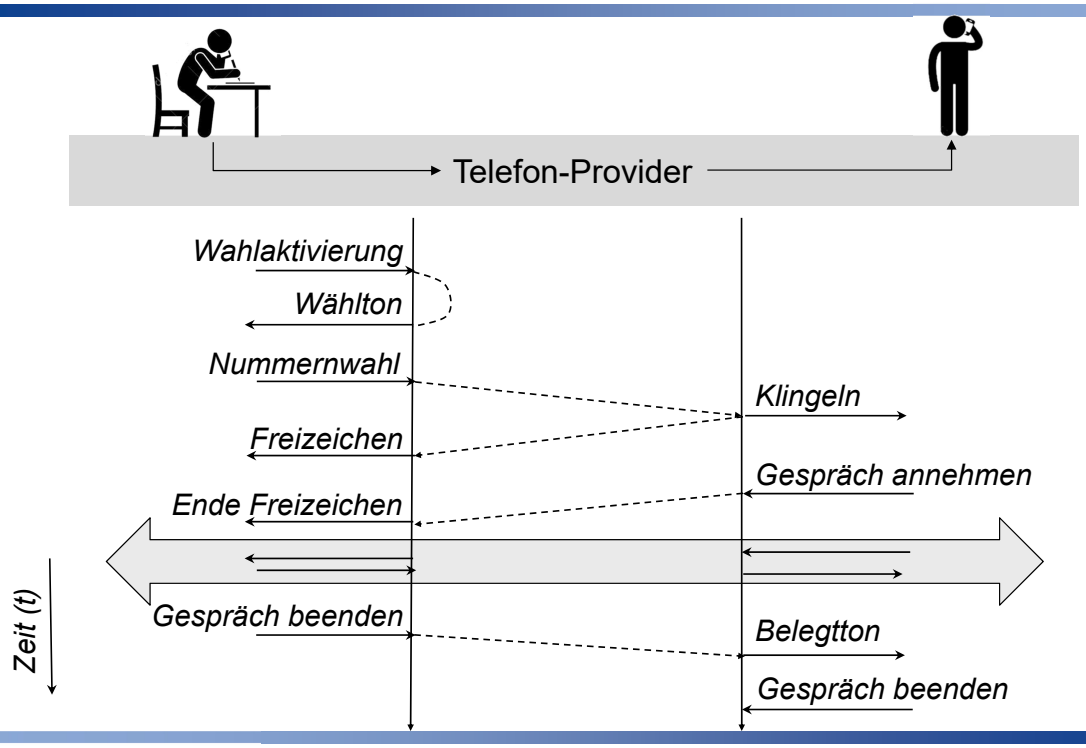
(Bitte nicht durcheinanderbringen: der Weberver selber bietet dem Web-Nutzer zwar auch einen Dienst an, aber dies ist in unserem Kontext nicht mit dem Begriff „Dienst“ gemeint. Wird im Bereich der Datenkommunikation von „Dienst“ gesprochen, ist stets ein Kommunikationsdienst gemeint, dem gegenüber auch der Webserver nur ein Dienstnehmer ist.)

Die Dienstnutzer „Web-Nutzer“ und „Webserver“ nehmen den Dienst eines Netzes (hier generell: Internet) in Anspruch, der die Anfrage vom Nutzer an den Server sowie die zugehörige Antwort zurück übermittelt. Der Dienst des „Internets“ sorgt dafür, dass Anfrage und Antwort jeweils am richtigen Ziel ankommen.

Beispiel: Post



Beispiel: Telefon



Gemeinsamkeiten der Beispiele

- **Welche Ähnlichkeiten weisen diese Kommunikationssysteme auf?**
  - *Kommunikationsdienste* (Dienste eines Kommunikationssystems) zur Übertragung von Daten in verteilten Umgebungen
  - Ablaufvorschriften / Mechanismen zur Koordination der Kommunikation *(Protokolle)*
  - *Nachrichten* werden ausgetauscht (zur Koordination, zum Austausch der Inhalte)
  - Kommunikation kann verschiedene Phasen aufweisen
  - Interaktion der Nutzer mit System erfolgt nur lokal
  - System kann hierarchisch strukturiert sein – mit verschiedenen Abstraktionsstufen – bleibt dem Nutzer in der Regel verborgen
  - ...

Gemeinsamkeiten sind:

• Es werden Nachrichten ausgetauscht

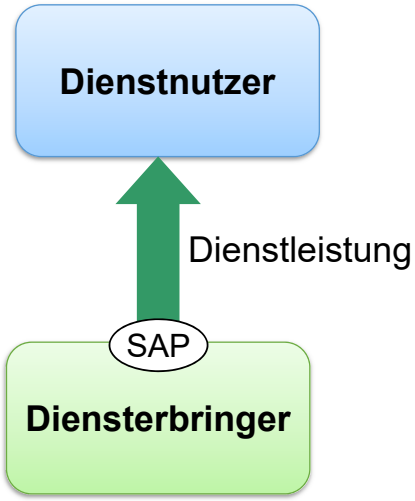
• Wir haben in jedem Fall ein System dazwischen (= Kommunikationssystem, bezogen auf die Datenkommunikation)

• Das Kommunikationssystem kann aus mehreren, irgendwie strukturierten Komponenten bestehen

Letztendlich kann man sagen: zwei Kommunikationspartner nutzen die *Dienste* eines Kommunikationssystems, um eine Nachricht zu übertragen, wobei das Kommunikationssystem selbst noch weiter strukturiert sein kann.

Charakterisierung von Diensten

- **Kommunikationsdienste**
  - *Dienstnutzer*: nimmt einen Dienst in Anspruch
  - *Diensterbringer*: bietet einen Dienst an
  - Ein Dienst wird im Rahmen einer *Dienstleistung* erbracht
    - Weist unterschiedliche Eigenschaften (z.B. bestätigt, unbestätigt) auf
    - Umfasst die Abwicklung eines Auftrags, welcher im Rahmen des Dienstes spezifiziert ist
  - *Dienstzugangspunkt* (Service Access Point, SAP): Schnittstelle zur Dienstnutzung



COM SYS | RWTH AACHEN UNIVERSITY

I-26

Ein Kommunikationssystem soll eine Reihe allgemein brauchbarer, wohldefinierter und geregelter Funktionen anbieten. Eine derartige Funktionalität fassen wir unter dem Begriff des *Dienstes* zusammen. Allgemein sind Dienste Zusammenstellungen zusammengehöriger Funktionen, die wiederholt in ähnlicher Form benötigt werden und daher durch Instanzen bereitgestellt werden können, die sich auf diese Funktionen spezialisieren. Die Instanzen werden *Diensterbringer* (oder *Dienstgeber*) genannt.

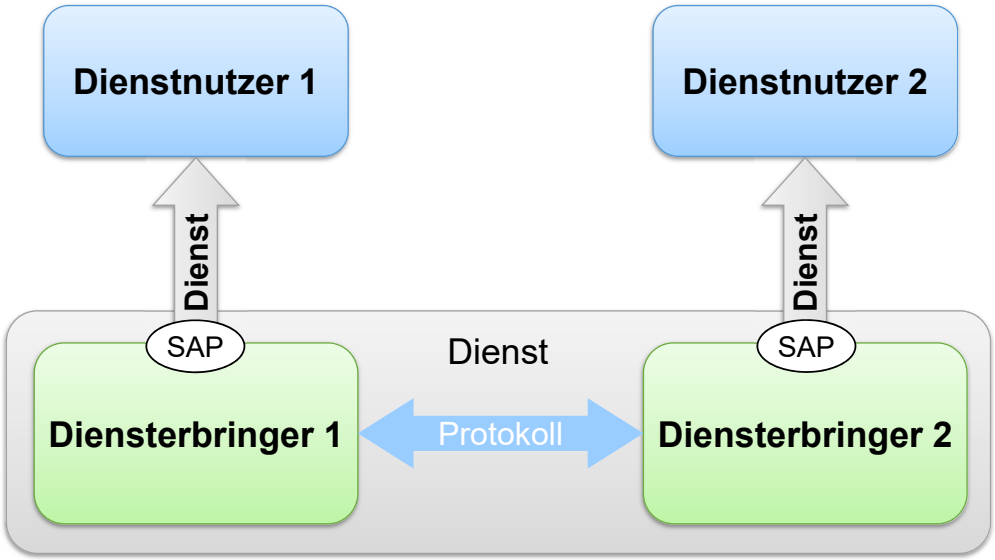
Die Instanzen, die die Funktion nutzen, sind die *Dienstnutzer* (oder *Dienstnehmer*). Sowohl Sender als auch Empfänger von Daten sind Dienstnutzer. Eine Änderung der Implementierung des Diensterbringers ist möglich, ohne dass der Dienstnutzer es bemerkt.

Beispiel: Kommunikationsdienst „Telefondienst“

- Kommunikationssystem: eine Sparte des Telefon-Providers
- Medium: verschiedene Kabel, Satellit, Funk
- Dienstnutzer: Kunde des Dienstes „Telefondienst“: Telefonierende Personen
- Dienstzugangspunkt: Telefon

Dienst und Protokoll – Übersicht

- **Grundlegendes Modell eines Kommunikationssystems**
  - Basierend auf der Unterscheidung zwischen *Dienst* und *Protokoll*



Ein Dienst regelt zwar, welche Funktionalität ein Kommunikationssystem anbietet, aber die Details der Diensterbringung bleiben dem Dienstnutzer verborgen – er sieht nur festgelegte Schnittstellen zur Dienstnutzung. Allerdings muss auch genau definiert sein, wie der Dienst erbracht wird – auf jedem der beteiligten Rechner läuft lokal eine Instanz (z.B. als Modul des Betriebssystem-Kernels), die mit den Instanzen auf anderen Rechnern nach fest definierten Regeln interagieren muss, um global einen einheitlichen Kommunikationsdienst anbieten zu können.

Protokolle sind Verhaltenskonventionen, die die zeitliche Folge der Kommunikation zwischen den diensterbringenden Instanzen festlegen und das Format (Syntax und Semantik) der auszutauschenden Dateneinheiten bestimmen. Die Dateneinheiten werden Protocol Data Units (PDUs) genannt.

Kommunikationssysteme basieren daher auf

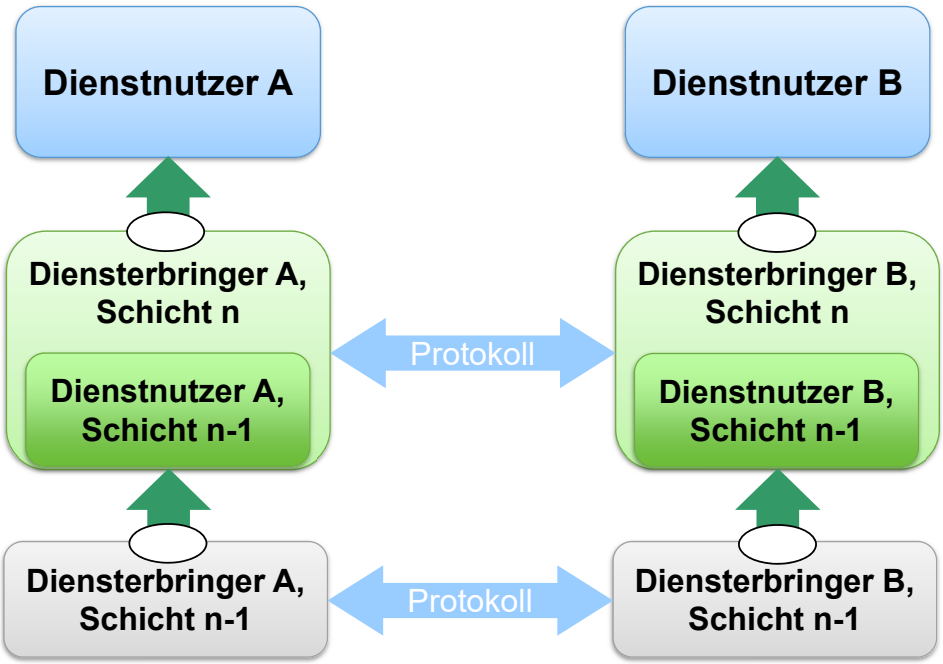
- Funktionalität = *Kommunikationsdienst* [engl. (Communication) Service]
- Realisierung der Funktionalität = Implementierung der Funktionalität als lokal ausführbare Instanzen der Diensterbringer. Auch „Protokollinstanz“ genannt.
- Regeln zur Erbringung der Funktionalität zwischen zwei räumlich verteilten Instanzen des Kommunikationssystems = *Kommunikationsprotokoll* [engl. (Communication) Protocol]

Dienstzugangspunkt: Eindeutig gekennzeichneter Zugangspunkt für die Inanspruchnahme eines Dienstes

und die Wechselwirkungen/Interaktionen zwischen den Dienstnutzern. Diese eindeutige Kennzeichnung entspricht einer Adresse, unter der die jeweiligen Dienstnutzer erreicht werden können. Im Englischen: Service Access Point, SAP

Dienst und Protokoll – Übersicht

- Grundlegendes *Modell eines Kommunikationssystems*



Generell gibt es zwei Möglichkeiten, einen Diensterbringer zu implementieren.

1. Der Diensterbringer kann als ein monolithisches Stück Software entwickelt werden, welches alle Aufgaben des Kommunikationsdienstes erbringt. Das ist zwar effizient umzusetzen, bringt aber den üblichen Nachteil einer monolithischen Software mit sich: mangelnde Flexibilität (da die Funktionalität an einer einzelnen Anwendung ausgerichtet ist) oder Überladung (da alle möglichen Funktionalitäten umgesetzt werden – Kommunikationssysteme sind sehr komplexe Systeme).
2. Die in der Praxis verwendete Methode ist die Modularisierung – teile die gesamte Funktionalität in logische Bereiche auf, die unabhängig voneinander umgesetzt und (bestimmten Regeln folgend) beliebig kombiniert werden können. In der Praxis kommen Schichtenmodelle zum Einsatz: von sehr anwendungsnahen bis hin zu sehr hardwarenahen Funktionalitäten werden die Funktionalitäten in einige logische Schichten gruppiert. Jede Schicht stellt eine Teilfunktionalität bereit und agiert als Diensterbringer für die nächsthöhere Schicht. Diese nächsthöhere Schicht kann die Funktionalität der tieferen Schicht nutzen und agiert selbst als Erbringer weiterer Funktionalitäten für die wiederum nächsthöhere Schicht. Auf jeder Schicht können unterschiedliche Varianten implementiert werden und je nach Bedarf einer Anwendung kann die entsprechende Variante verwendet werden. Dadurch erzielt man eine höhere Flexibilität.

Betrachten wir nun die Partner einer Schicht:

- Sie bieten einen Dienst für eine höhere Schicht an
- Sie kennen nur den direkt unterliegenden Dienst
- Sie benutzen diesen unterliegenden Dienst (außer der untersten Schicht); generell kann man auch sagen, dass ein Medium genutzt wird, da die Partner einer Schicht den Dienst als „Medium“ sehen, welches die Daten zustellt. Ein tatsächliches Medium nutzt dabei nur die unterste Schicht, ansonsten hat

man ein virtuelles Medium.

- Sie „unterhalten sich“ gemäß Regeln (= Protokollen) - z.B. „Telefon“-Schicht: wählen/klingeln/besetzt

Dienstprimitive

- **Grundlegende Dienstprimitive nach OSI-Modell**
  - *Dienstfunktionen*
    - `CONNECT`: Verbindungsaufbau
    - `DISCONNECT`: Verbindungsabbau
    - `DATA`: Datenaustausch
    - `ABORT`: Abbruch durch Dienstnutzer oder Diensterbringer
  - Typen bzw. Funktionen eines *Dienstprimitivs*:
    - `Request`: Anfordern eines Dienstes
    - `Indication`: Anzeige am Partner-Dienstzugangspunkt
    - `Response`: Antwort des Partner-Dienstzugangspunkts
    - `Confirmation`: Bestätigung der Diensterbringung am anfordernden Dienstzugangspunkt

Dienstprimitive stellen die – an einen zeitlichen Ablauf gebundene – abstrakte Beschreibung der Wechselwirkung an den Dienstzugangspunkten zur Nutzung eines Dienstes dar

Hier im Beispiel stelle ein Kommunikationsdienst vier Funktionen bereit:

- Connect: Herstellen einer Kommunikationsbeziehung zwischen den Kommunikationspartnern (z.B. Vereinbarung von Parametern)

- Disconnect: gegenseitige Einigung der Partner auf die Beendigung der Kommunikation

- Data: Austausch von Daten

- Abort: Beendigung der Kommunikationsbeziehung durch den Dienstanbieter oder einseitiger Abbruch durch einen der Dienstnutzer

Diese vier Dienstfunktionen werden durch die auf der Folie aufgeführten Primitive angeboten; im Kommunikationssystem erfolgt die Dienstnutzung durch Verwendung der Dienstprimitive.

Anmerkung: das sogenannte OSI-Modell wird später in diesem Kapitel noch behandelt. Hier soll dieses Modell (ohne genaueres Wissen, wie es aussieht) nur dazu dienen, eine konkrete Syntax für Dienstprimitive bereitzustellen.

Dienstprimitive – Beispiel

- **Die Benennung eines Dienstprimitivs besteht aus folgenden Komponenten:**

Name der Schicht/Anwendung	Dienstleistung	Dienstgrundtyp	Parameter
Physical (Ph) Data Link (DL) Network (N) Transport (T) HTTP FTP ...	Connect (Con) Data (Dat) Abort (Abo) Provider Abort (PAbo) Disconnect (Dis) ...	Request (Req) Indication (Ind) Response (Rsp) Confirmation (Cnf)	*(entsprechend)*

- **Beispiel:**
  - T-Con.Req(Adressen) = Verbindungsaufbauanforderung an der Schnittstelle zum Transportdienst
  - HTTP-Get[Dat.Req](URL) = Anforderung der Webseite, die durch URL identifiziert wird

Die Beschreibung der Dienstprimitive erfolgt hier gemäß dem Schema

<Schichtabkürzung> -<Dienstleistung>. <Diensttyp>

Das OSI-Modell definiert 7 Schichten. Hier kann zunächst verallgemeinernd davon ausgegangen werden, dass es sich dabei um Softwaremodule zur Datenübertragung handelt, die jeweils einen bestimmten Kommunikationsdienst erbringen. Der genauere Zweck der Schichten wird später noch erläutert. Jede der sieben OSI-Schichten hat eine Abkürzung, die sich an die englische Bezeichnung der Schicht anlehnt; so heißt z.B. die erste Schicht, die Bitübertragungsschicht, im Englischen Physical Layer mit der Abkürzung Ph.

Typische Dienstleistungen sind Aufbau, Datenübertragung, Rücksetzen, Abbruch. Diese fließen wie die Schichtbezeichnungen ebenfalls als Abkürzungen der entsprechenden englischen Begriffe in die Dienstprimitive ein, also z.B.

Verbindungsaufbau : Connect = Con

Übertragung : Data = Dat

Der Diensttyp umfasst die bereits erwähnten vier Grundtypen von Ereignissen, die aus der Abwicklung der Dienstleistung hervorgehen

Anfrage : Request = Req

Anzeige : Indication = Ind

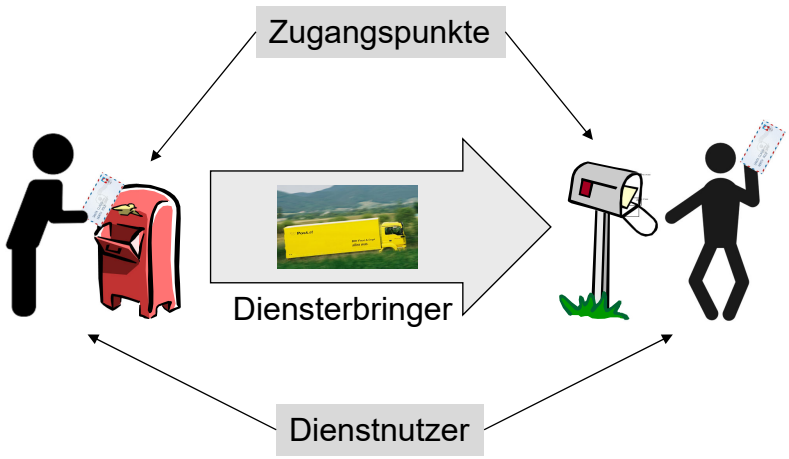
Antwort : Response = Rsp

Bestätigung : Confirmation = Cnf

In der Realität erfolgt die Umsetzung üblicherweise nicht sauber nach diesem Modell.

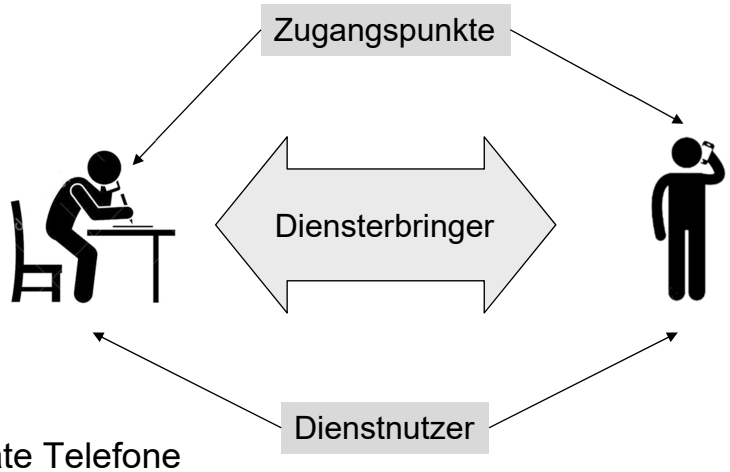
Dienste: Beispiele

- **Beispiel: Post**
  - *Dienstnutzer*: Postkunden
  - *Diensterbringer*: Post
  - *Zugangspunkte*: öffentliche und private Briefkästen
  - *Dienstprimitive*: Einwerfen eines adressierten und frankierten Briefes (`DATA.Req`), Postbote wirft Brief in Briefkasten des Empfängers (`DATA.Ind`) (in den USA: red flag als Indikator, dass Post ankam bzw. zum Abholen bereit ist)



Dienste: Beispiele

- **Beispiel: Telefon**
  - Dienstnutzer:
    Telefonkunden
  - Dienstanbieter:
    Telefon-Provider
  - Zugangspunkte:
    Öffentliche und private Telefone
  - Dienstprimitive:
    Nummer wählen (`Con.Request`), Klingeln (`Con.Indication`), ...



Dienstablauf

- **Abfolge von Dienstprimitiven ist nicht zufällig**
  - Abfolge muss spezifiziert werden können
- **Zeitablaufdiagramme (*Weg-Zeit-Diagramme*)**
  - Pro Dienstzugangspunkt eine Zeitachse (vertikal)
  - Dienstprimitive: Pfeile
  - Weitere Zusammenhänge (neben Abfolge) durch Kommentierungen
    - Kausalitäten
    - Exakte Zeitbedingungen
    - Kommunikationsaktivitäten, z.B. Senden oder Verlust von Paketen
  - Zeitablaufdiagramme können nicht alle Zusammenhänge spezifizieren
- **Weitergehende Spezifikationstechniken**
  - Zustandstabellen und Zustandsübergangsdiagramme (Automaten), Spezifikationssprachen

Bislang wurden lediglich die statischen Beschreibungsaspekte, konkret die Dienstzugangspunkte und die Dienstprimitive, betrachtet. Interessant und komplex wird die Dienstbeschreibung, wenn zusätzlich noch dynamische Abläufe, d.h. das zeitliche Auftreten von Dienstprimitiven an den beteiligten SAPs, betrachtet werden.

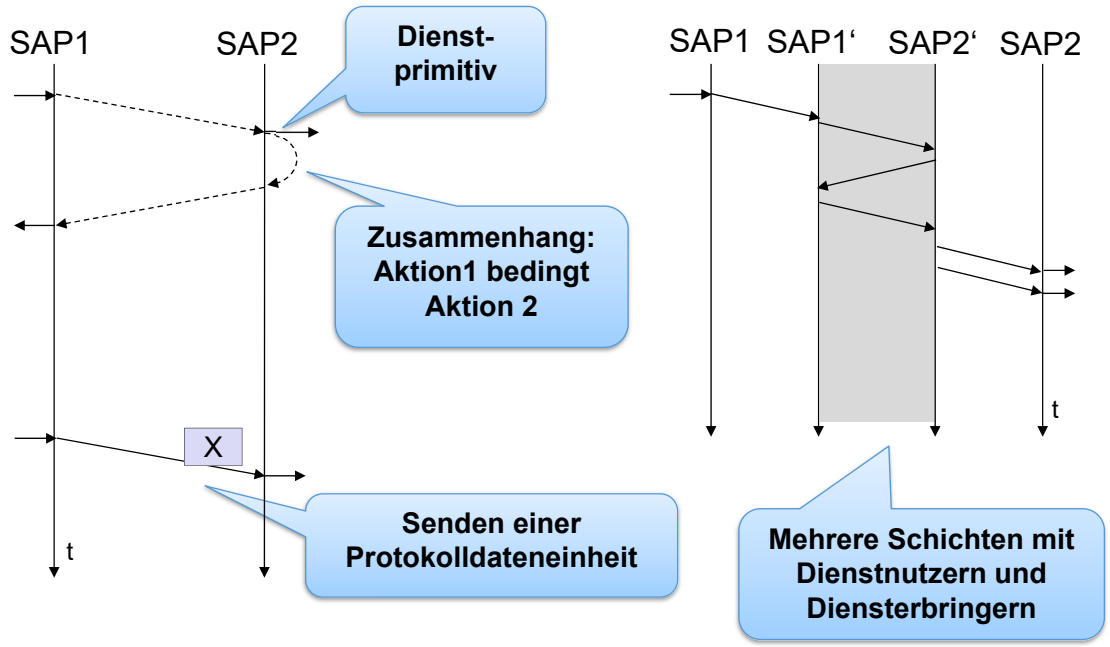
Eine einfache Möglichkeit zur Darstellung von Kommunikationsdiensten mit Betrachtung dynamischer Abläufe sind Weg-Zeit-Diagramme. Diese erlauben es, den Austausch von Dienstprimitiven zwischen zwei Dienstzugangspunkten einfach darzustellen. Solche Darstellungen können dann als Ausgangspunkt für die Implementierung der zugehörigen Protokolle verwendet werden.

Diese Diagramme werden hier knapp eingeführt und ab und zu zur Visualisierung verwendet; sie haben allerdings starke Nachteile, da die Visualisierung komplex wird, sobald ein dritter Dienstzugangspunkt verwendet wird; und auch einfache Verzweigungsmöglichkeiten im Dienstablauf (if-then-else) lassen sich nicht gut darstellen.

Zur tatsächlichen Protokollentwicklung bieten sich stattdessen mächtigere Spezifikationssprachen oder Automaten an – auch wenn die Protokollentwicklung oft auf eine vollständige formale Spezifikation verzichtet.

Dienstablauf

- **Beschreibung durch Weg-Zeit-Diagramme**

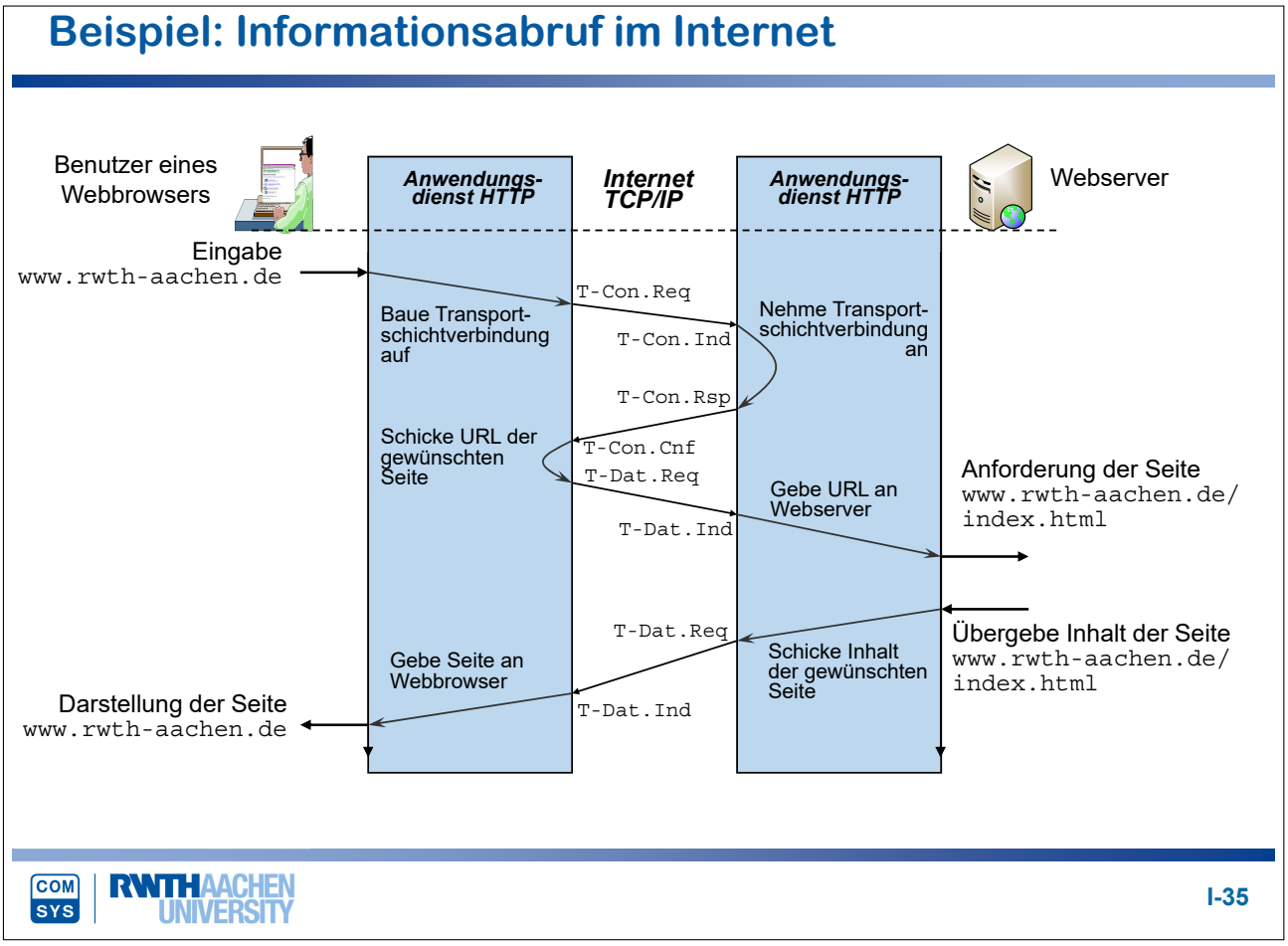


Horizontal werden die beteiligten SAPs angegeben, die vertikalen Achsen sind Zeitachsen, an denen die auftretenden Dienstprimitive aufgeführt sind. Die Ursache-Wirkung-Beziehungen zwischen den an getrennten SAPs auftretenden Dienstprimitiven erfolgt durch Pfeile von einer vertikalen Zeitachse zu einer anderen.

Unterschied der Notation bei den Pfeilen: die gestrichelten Pfeile stellen eine Interaktion der Kommunikationsteilnehmer (mittels Dienstprimitiven) dar, die nicht näher spezifiziert ist. Werden stattdessen (wie unten links dargestellt) durchgezogene Pfeile verwendet, ist uns die Implementierung bekannt (d.h.: wir wissen, welches Protokoll zur Interaktion eingesetzt wird und welche konkreten Nachrichten ausgetauscht werden).

Hinter den gestrichelten Pfeilen könnten sich auch noch komplexere Interaktionen verbergen. Z.B. könnte bei unserer Dienstimplementierung (vorborgen vor dem Dienstnutzer) noch die Nutzung eines weiteren Dienstes nötig werden (siehe rechts auf der Folie) – unser Diensterbringer wäre hier nicht monolithisch implementiert, sondern nutzt selbst einen weiteren Diensterbringer über SAP1‘ zur Erbringung seines Dienstes. Gestrichelte Pfeile sollen also andeuten, dass uns die real notwendigen Zwischeninteraktionen nicht bekannt sind.

Zu jedem Dienstprimitiv können außerdem Kontrollparameter angegeben werden. Dies sind z.B. Adressen von Sender und Empfänger oder Anforderungen an die Dienstqualität (beispielsweise: schnelle Übertragung notwendig, garantierte maximale Zeit bis zur Auslieferung der Daten).



Die betrachteten Partner-Protokollinstanzen sind in diesem Beispiel die HTTP-Instanzen im Webserver und im Webbrowser (Client). HTTP ist ein sogenanntes Anwendungsprotokoll (HyperText Transfer Protocol), das zum Austausch von Dokumenten im WWW entwickelt wurde. Die HTTP-Instanzen bieten kommunizierenden Anwendungen eine Dienstschnittstelle an, stellen also eine Implementierungsmöglichkeit der obersten Schicht eines Kommunikationssystems dar. Sie nutzen Dienste der sogenannten Transportschicht, um Anfrage und Antwort zuverlässig zwischen räumlich verteilten Rechnern zu übertragen.

Der Benutzer des Webbrowsers gibt eine URL ein. Für den Web-Browser nicht sichtbar wird eine Verbindung zum Webserver aufgebaut, über die die Daten zuverlässig übertragen werden können. Der Verbindungsaufbau ist bestätigt, da die anfragende Instanz wissen muss, ob die Gegenseite die Verbindungsanfrage annimmt und wann sie zum Empfang von Daten bereit ist.

Die Anforderung der gewünschten Webseite hingegen erfolgt unbestätigt. Für die anfragende Instanz ist es nicht wichtig zu wissen, ob und wann die Anfrage auf der Gegenseite angekommen ist, da die unterliegende Schicht die Zustellung garantiert. Ebenso erfolgt die Übertragung der Antwort unbestätigt.

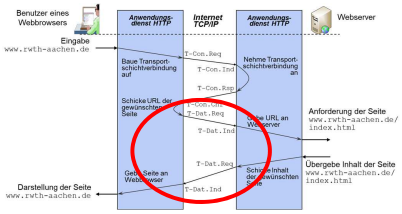
Wann bestätigte und wann unbestätigte Dienste eingesetzt werden, hängt also zum einen von den Anforderungen der Kommunikation selbst ab, zum anderen aber auch von den unterliegenden Diensten.

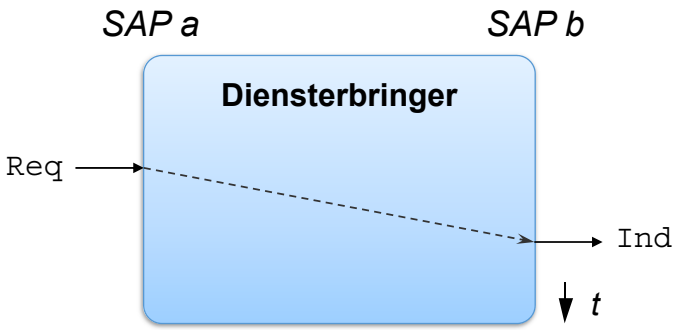
Dieses Beispiel beinhaltet mehrere Begriffe, deren Verständnis wichtig ist:

- Verbindungsorientierte Kommunikation (Verbindungsaufbau)
- Bestätigte/unbestätigte Kommunikation.

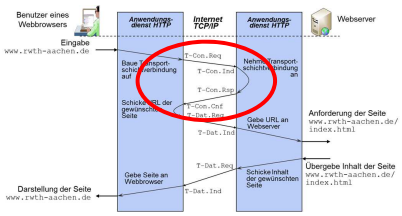
Bestätigter/unbestätigter Dienst

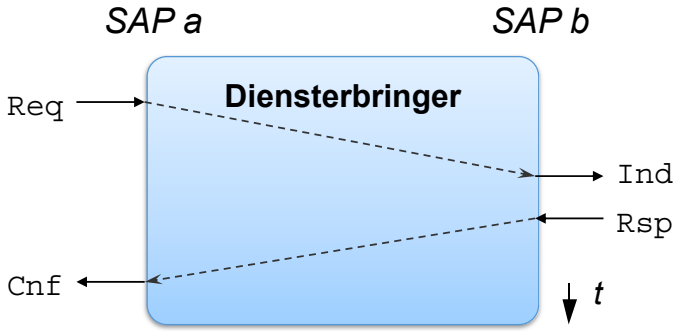
- **Unbestätigter Dienst**





- **Bestätigter Dienst**





COM SYS | RWTH AACHEN UNIVERSITY

I-36

Bei Nutzung eines unbestätigten Dienstes übergibt man dem Kommunikationsdienst an seinem SAP die Daten und diese werden – eventuell unter Zuhilfenahme weiterer Dienste – an den SAP des Empfängers ausgeliefert. Der Sender bekommt keine Rückmeldung, ob die Daten tatsächlich ausgeliefert bzw. angenommen wurden. Ob die Daten zuverlässig ausgeliefert werden oder nicht, hängt somit von der konkreten Dienstimplementierung ab (d.h. vom verwendeten Protokoll und den genutzten Diensten tieferer Schichten).

Beispiel: Versand einer E-Mail

Bei einem bestätigten Dienst hingegen bekommt der sendende Dienstnutzer eine Rückmeldung über die erfolgreiche Zustellung.

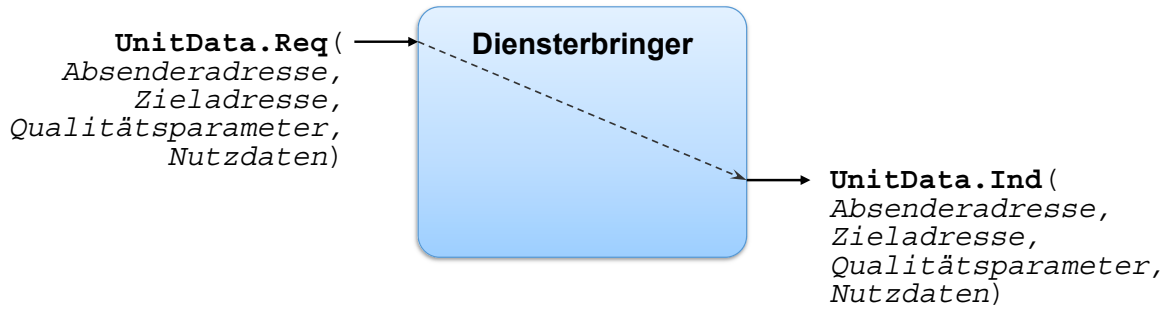
Beispiel: Anzeige, wann eine Nachricht über WhatsApp zugestellt wurde.

Bitte beachten: ob ein Dienst bestätigt oder unbestätigt ist, sagt noch nichts über die Zuverlässigkeit einer Datenübertragung aus – bei einer Übertragung im Internet kann es z.B. durch Netzüberlastung zu einem Verlust einzelner übertragener Dateneinheiten kommen, egal, ob der Dienst bestätigt ist.

Verwendet man einen bestätigten Dienst, erkennt man solche Verluste auf jeden Fall daran, dass keine Bestätigung erfolgt; der Dienstnutzer kann die Anfrage noch einmal stellen. Bei einem unbestätigten Dienst gibt es diese Möglichkeit nicht. Allerdings kann der Dienst intern (also im Protokoll) mit Bestätigungen arbeiten (die dem Dienstnutzer verborgen bleiben), so dass der Dienstnutzer auch bei Verzicht auf Bestätigungen davon ausgehen kann, dass die Daten zuverlässig übertragen werden.

Dienst und Verbindung

- **Verbindungsloser Dienst**
  - Dienstleistung ohne Kontext
    - Jeder Datenaustausch wird isoliert betrachtet, ohne jegliche Berücksichtigung vorhergegangener Kommunikationsvorgänge
  - Kontrollparameter als Teil der übergebenen Dateneinheit
  - Keine feste Verbindung zwischen den Kommunikationspartnern



Eine weitere Unterscheidung von Diensten kann anhand des Begriffs der „Verbindung“ erfolgen: Dienste können verbindungslos oder verbindungsorientiert sein.

Verbindungslose Dienste sind die einfachere Variante und werden daher zuerst betrachtet.

Das einzige Dienstprimitiv, das angeboten wird, ist UnitData, also ein „einzelnes Datum“. In diesem Dienstprimitiv müssen alle für die Übertragung relevanten Kontrollinformationen enthalten sein. Das sind zum Beispiel sämtliche Adressierungsinformation (Absender- und Zieladresse) und die Qualitätsparameter, in denen der Sender seine Wünsche bzgl. der Datenübertragung formuliert.

Verbindungslose Dienste sind normalerweise auch unbestätigt.

Dienst und Verbindung

- **Verbindungsorientierter Dienst**
  - Dienstleistung im Kontext, d.h. Diensterbringung abhängig von früher erbrachten Dienstleistungen
    - Vor dem Datenaustausch zwischen Dienstnutzern: *Verbindungsaufbau* durch die beteiligten Instanzen des Dienstanbieters
      - *Protokollabhängige Aushandlung von Übertragungsparametern*, z.B. Kommunikationspartner, Dienstqualität, Übertragungsweg
    - Datenaustausch innerhalb der Verbindung erfolgt unter Berücksichtigung des aktuellen Verbindungszustands
  - Weniger Kontrolldaten notwendig, aber
    - Zeitaufwand für Herstellung des Kontextes (Verbindungsaufbau)
    - Aufwand für die Verwaltung des Kontextes

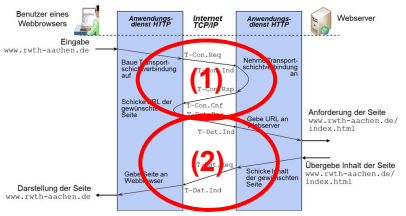
Der Kontext bei verbindungsorientierten Diensten entsteht durch früher erbrachte Dienstleistungen. Vor dem Datenaustausch muss bei einem verbindungsorientierten Dienst zuallererst eine Verbindung aufgebaut (d.h.: ein Kontext generiert) werden. Dies geschieht durch einen sogenannten Verbindungsaufbau, bei dem die Dienstnutzer einen Kontext aushandeln, d.h. Kontrollparameter definieren, die auf alle im Rahmen der Verbindung ausgetauschten Dienstprimitive angewendet werden.

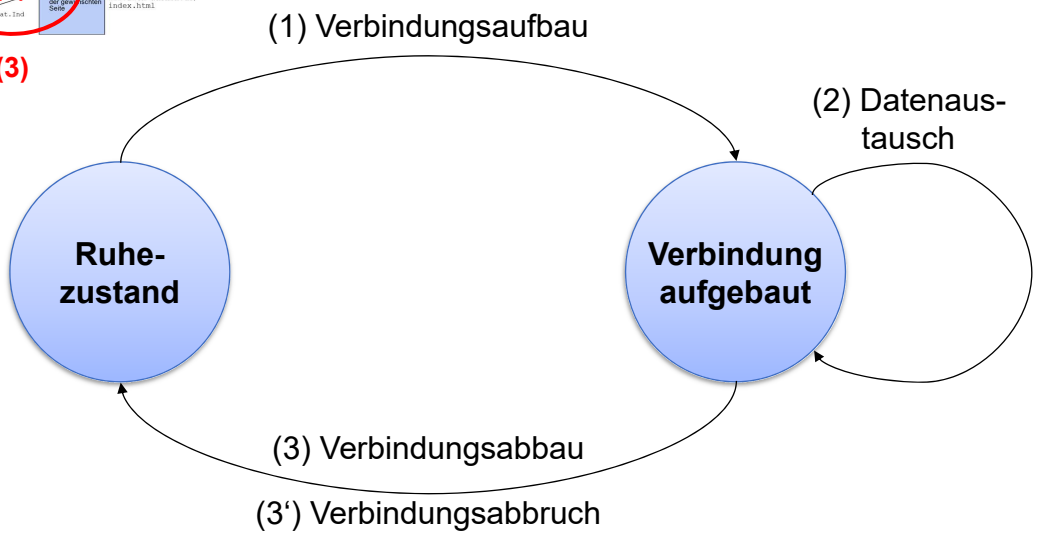
Der Zustand einer Verbindung zum Zeitpunkt *t* setzt sich aus den Parametern zusammen, zusätzlich jedoch auch aus (theoretisch) allen Kommunikationsvorgängen, die von Zeitpunkt 0 (Verbindungsaufbau) bis Zeitpunkt *t* innerhalb dieser Verbindung stattgefunden haben. Wie lange dieses „Gedächtnis“ zurückreicht und wie viele/welche Daten hierbei gespeichert werden (wenn überhaupt), ist jedoch protokollspezifisch. Als Beispiel sei der (später im Detail besprochene) Protokollmechanismus der „Sequenznummern“ erwähnt, der jedem Datenpaket eine spezifische, ganzzahlige Nummer zuordnet. Zusammen mit entsprechenden Bestätigungen von Empfängerseite ermöglicht dies eine Erkennung von Paketverlusten - hierbei speichert der Sender Datenpakete nur solange zwischen, bis eine ordnungsgemäße Bestätigung eintrifft.

Durch Aufbau und Verwaltung eines Kontextes scheint alles komplexer und aufwändiger zu werden. Man hat aber den großen Vorteil, dass man sich die explizite Übertragung einiger Kontrollparameter in jedem Dienstprimitiv sparen kann, weil diese bereits im Kontext beschrieben sind. Zum Beispiel: Beim verbindungsorientierten Dienst kann beim Verbindungsaufbau der Weg durch ein komplexes Netzwerk zwischen den beteiligten Kommunikationspartnern ermittelt und als Kontext aufbewahrt werden. Bei einem verbindungslosen Dienst muss die Wegwahl bei jedem einzelnen Dienstaufruf erfolgen. Ein wesentlicher Vorteil verbindungsorientierter Dienste ist somit, die Tatsache, dass nach einem Verbindungsaufbau nicht mehr jedes Datenpaket einer Verbindung sämtliche (bereits ausgehandelte) Parameter beinhalten muss (z.B. Adresse, Dienstqualität etc.), was eine Reduzierung des Anteils der übertragenen Kontrollinformationen und somit letztendlich auch der Netzlast selbst bedeutet.

Beispiele von vorher: die Versendung von Briefen ist verbindungslos – jeder Brief muss alle Kontrollinformationen enthalten. Ein Telefongespräch ist verbindungsorieniert – das Wählen einer Nummer startet einen Verbindungsaufbau, alle Sprachdaten folgen einem festgelegten Weg durch das Telefonnetz.

Verbindungsorientierte Dienste: Phasen



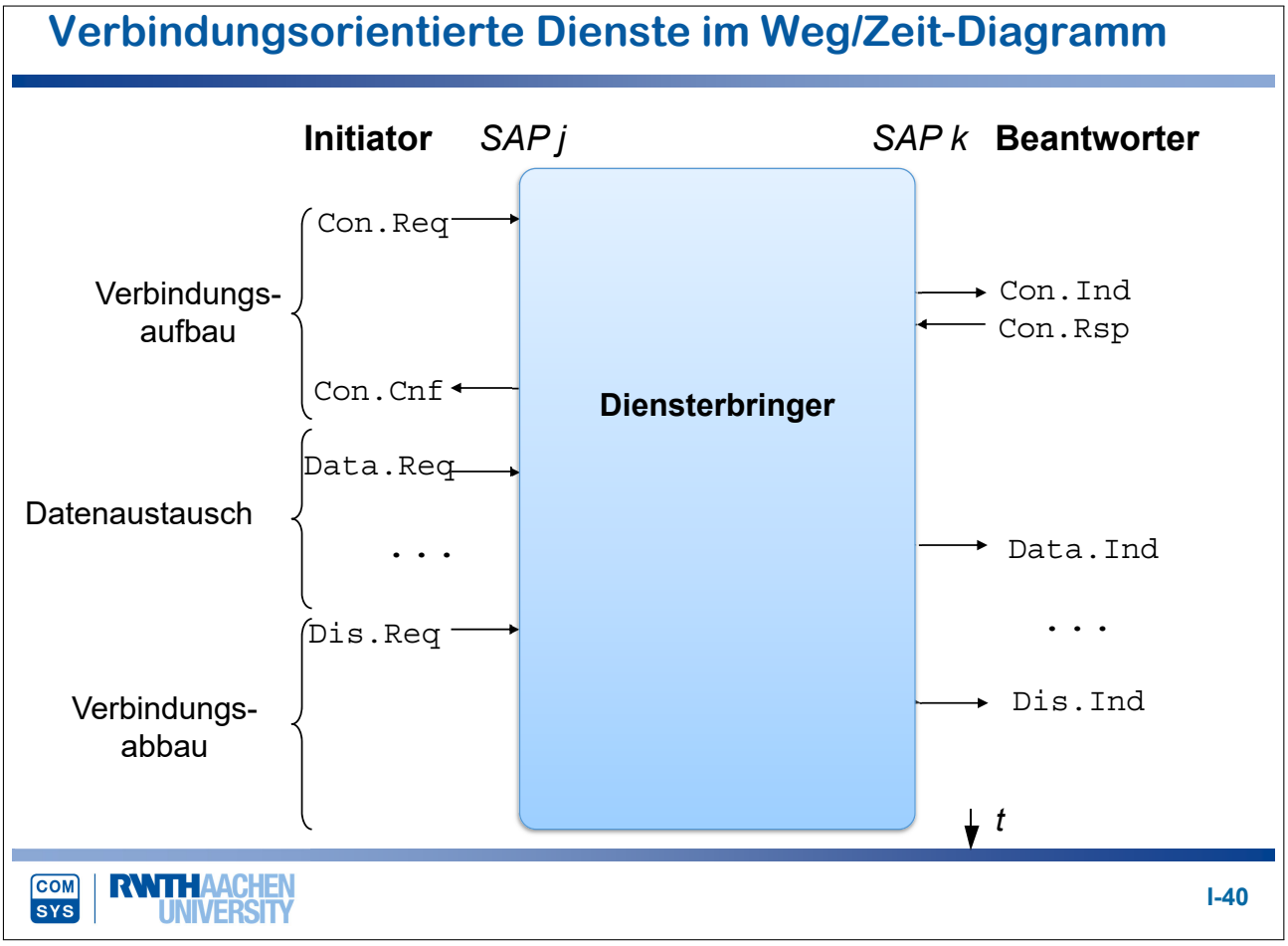


COM SYS | RWTH AACHEN UNIVERSITY

I-39

Die Klasse der verbindungsorientierten Dienste und das im Bereich der Datenkommunikation zentrale Konzept der Verbindung werden im Folgenden vertieft.

Ein verbindungsorientierter Dienst kann übersichtlich mithilfe eines endlichen Automaten in die vier Betriebsphasen „Verbindungsaufbau“, „Datenaustausch“, „Verbindungsabbau“ und „Verbindungsabbruch“ aufgeteilt werden.



(ausgeblendete Folie – wurde in der Vorlesung nicht behandelt)

Ein Verbindungsaufbau/-abbau hat im Weg/Zeit-Diagramm folgendes Aussehen:

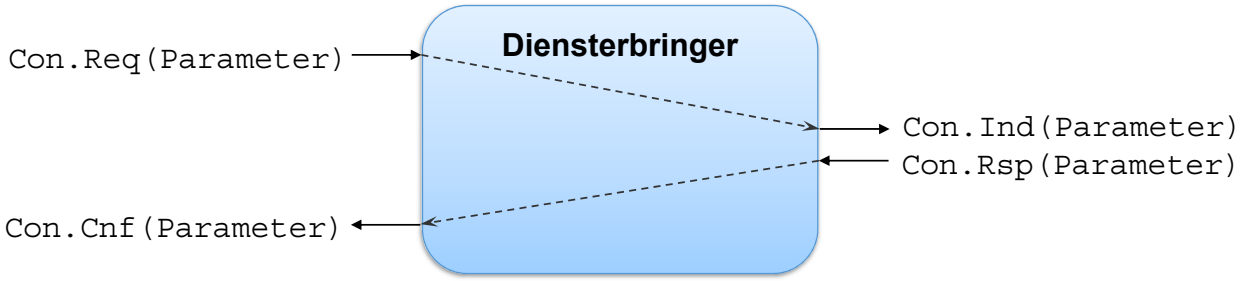
Phase 1: Verbindungsaufbau durch Con.Req/Ind/Rsp/Cnf

Phase 2: Datenaustausch durch Data.Req/Ind

Phase 3: Verbindungsabbau durch Dis.Req/Ind (unbestätigter Verbindungsabbau)

Üblicherweise ist der Verbindungsaufbau bestätigt, damit ein Dienstnutzer weiß, ab wann die Datenübertragung möglich ist. Datenaustausch und Verbindungsabbau können wie hier unbestätigt, oder auch bestätigt implementiert werden, abhängig von den unterliegenden Diensten.

Phase 1: Verbindungsaufbau



- **Parameter z.B.**
  - Initiatoradresse
  - Responderadresse
  - Qualitätsparameter
  - Wenige Nutzdaten

COM SYS | RWTH AACHEN UNIVERSITY I-41

Im Folgenden werden die drei Phasen eines verbindungsorientierten Dienstes genauer behandelt, beginnend mit der Verbindungsaufbauphase. Zunächst sollen die beim Verbindungsaufbau beteiligten Parameter näher betrachtet werden.

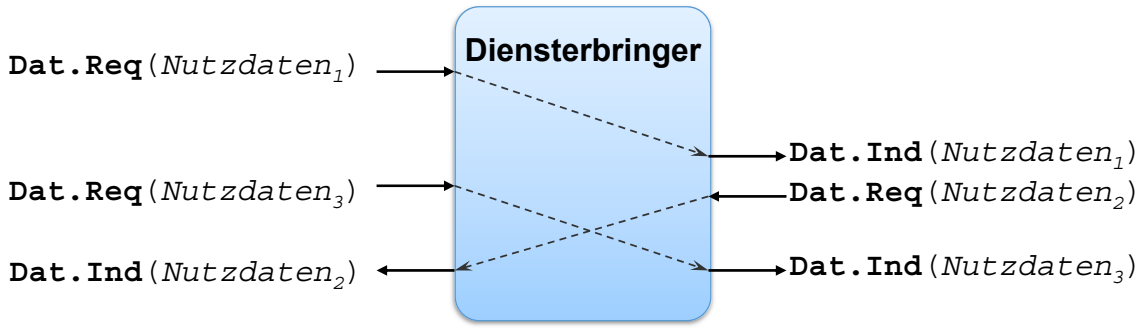
Folgende Parameter werden in den Dienstprimitiven zum Verbindungsaufbau genutzt:

- Initiator-/Responderadresse: SAP-Adressen
- Qualitätsparameter: weitere Kontrollparameter. Denkbare Qualitätszusicherungen eines Datenaustauschs sind verlustfreie Übertragung, reihenfolgetreue Auslieferung, hochpriorer Datenaustausch, …
- Kurze Nutzdaten: Dienstnutzer kann während des Verbindungsaufbaus zusätzliche Daten austauschen. Die zusätzlichen Daten stehen i.d.R. in direktem Zusammenhang mit der aufzubauenden Verbindung. Sie können z.B. für die Angabe genutzt werden, für welche überliegenden Protokolle die Verbindung dienen soll (z.B. Transportverbindung zum Zweck des Dateitransfers). Der Responder hingegen kann die kurzen Nutzdaten dazu verwenden, einen Statuscode z.B. für den Grund der Ablehnung einer Verbindung anzugeben.

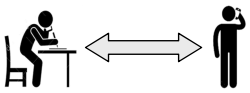
Ein interessanter Aspekt des Verbindungsaufbaus betrifft die Aushandlung der Qualitätsparameter. Der Initiator gibt gewisse Parameter, die er sich wünscht, vor. Diese werden vom Dienst bzw. vom Responder geprüft und können heruntergestuft werden.

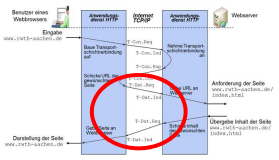
Konkretes Beispiel: Der Initiator wünscht sich einen Datendurchsatz von 200 MBit/s, der Dienst kann ihm aufgrund des zugrundeliegenden Mediums nur 100 MBit/s anbieten und der Responder reduziert letztendlich auf 50 MBit/s.

Phase 2: Unbestätigter Datenaustausch



- **„Ungeregelter“ Datenaustausch**
  - Beide Seiten gleichberechtigt, können beide jederzeit senden
  - Keine Rückmeldung an sendenden Dienstnutzer, ob die Nutzdaten zugestellt wurden

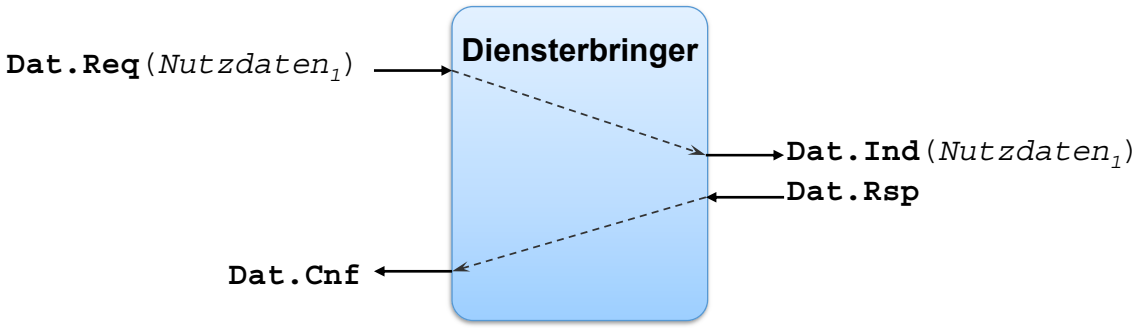




Die zweite Phase ist der Datenaustausch, also die eigentliche „Produktiv-Phase“, in der die Nutzdaten über die aufgebaute Verbindung ausgetauscht werden. Im Gegensatz zum bestätigten Datenaustausch besitzt der unbestätigte Datenaustausch dabei den Vorteil, dass Sender und Empfänger nur lose synchronisiert sind; das bedeutet, dass sich Sender und Empfänger nur wenig abstimmen müssen und sich entsprechend wenig im Fortschreiten ihrer Arbeit behindern.

Wie die Folie zeigt, beinhalten die beim unbestätigten Datentransfer verwendeten Dienstprimitive Dat.Req und Dat.Ind nur einen Parameter (die unbestätigt zu übertragenden Nutzdaten). Alle weiteren Parameter wurden bereits beim Verbindungsaufbau ausgetauscht/ausgehandelt. (Zumindest in diesem Beispiel – es können durchaus noch mehr Parameter vorhanden sein; der Phantasie der Protokollentwickler sind keine Grenzen gesetzt.)

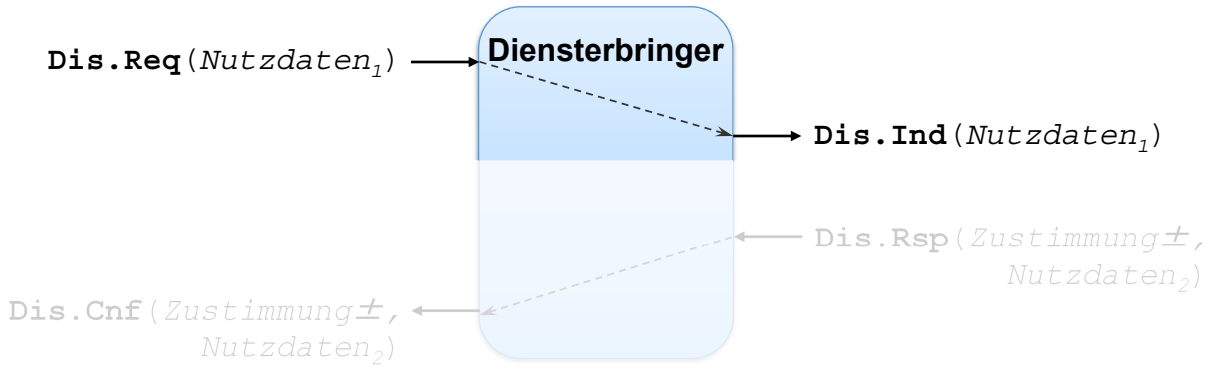
Phase 2: Bestätigter Datenaustausch



- **„Geregelter“ Datenaustausch**
  - Sendender Dienstnutzer erhält Rückmeldung z.B. über erfolgreiche Zustellung oder Ablehnung der Daten

Die Dienstprimitive Dat.Rsp und Dat.Cnf der bestätigten Variante beinhalten in diesem Beispiel gar keine Parameter mehr und sind z.B. nur eine Zustellungsbenachrichtigung

Phase 3: Verbindungsabbau



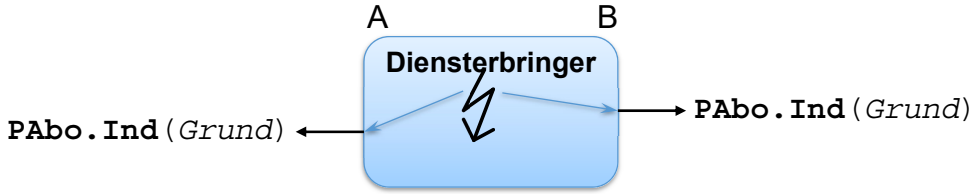
- **Geregelte Beendigung der Verbindung**
  - Bestätigt oder unbestätigt
  - Zur Freigabe von für die Kommunikation reservierten Ressourcen
  - z.B. bei der Transportschichtverbindung im HTTP-Beispiel

Der Verbindungsabbau wird durch einen der Dienstnehmer durch die Inanspruchnahme des entsprechenden Dienstprimitivs „Disconnect“ (oder „Release“) eingeleitet.

Der wesentliche Unterschied zwischen dem bestätigten und dem unbestätigten Verbindungsabbau besteht darin, dass beim bestätigten Verbindungsabbau der zweite Teilnehmer dem Verbindungsabbau widersprechen kann. Diese Möglichkeit ist durch den Parameter „Zustimmung+/-“ in Rel.Rsp/Cnf gegeben. In den Nutzdaten könnte der Dienstnutzer dann angeben, warum er es ablehnt, die Verbindung abzubauen.

Phase 3‘: Verbindungsabbruch

- **Abbruch durch Diensterbringer (Provider Abort, PAbo)**
  - z.B. Abbruch der LTE-Verbindung aufgrund überlasteter Zelle



- **Nutzerabbruch (User Abort, UAbo)**
  - z.B. abrupte Beendigung einer TLS-Verbindung, wenn eine korrupte Dateneinheit empfangen wird

COM SYS | RWTH AACHEN UNIVERSITY

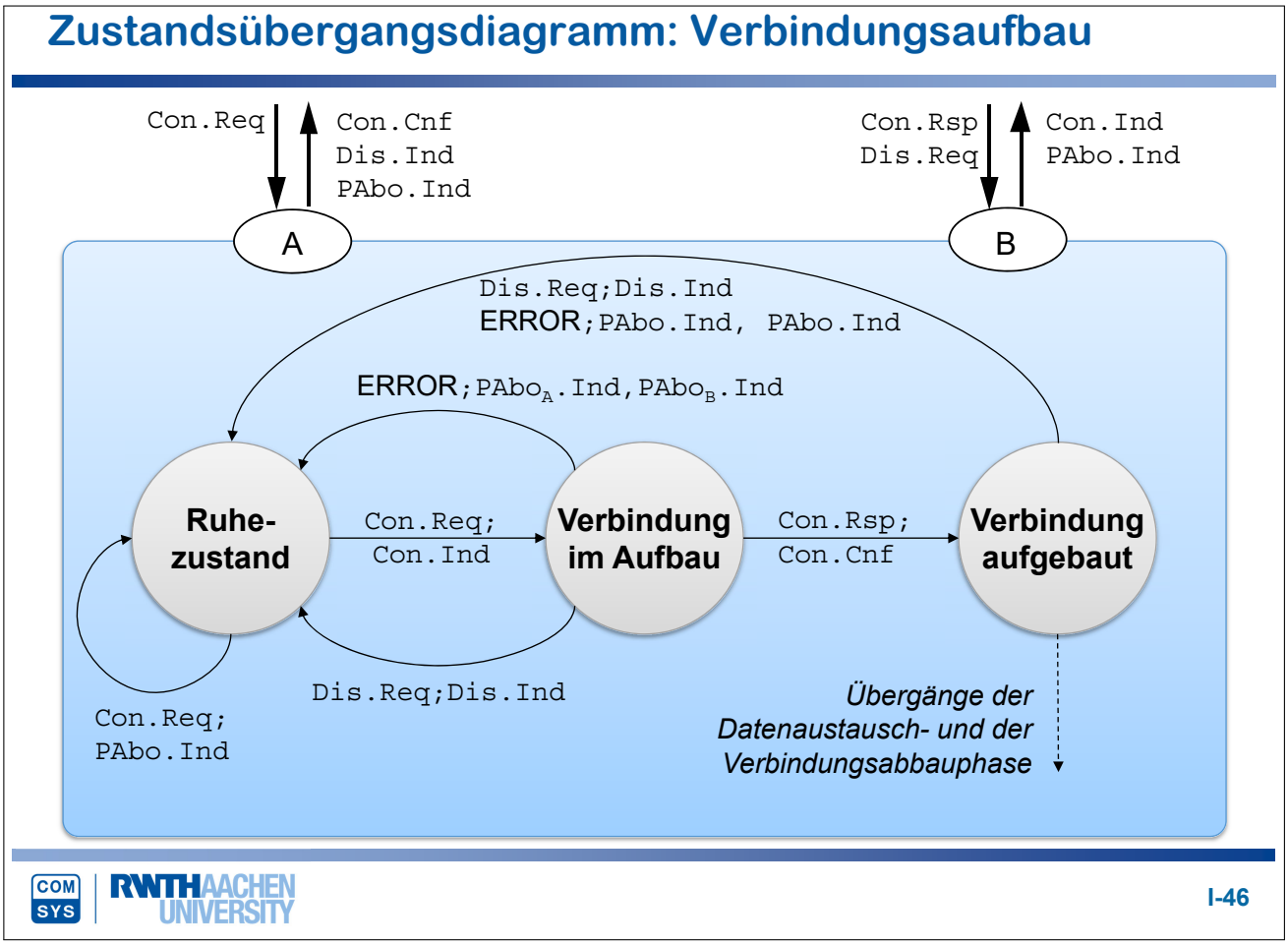
I-45

Abschließend soll auch die vierte und letzte Phase des verbindungsorientierten Dienstes behandelt werden, der Verbindungsabbruch. Wie der Begriff „Abbruch“ schon vermuten lässt, handelt es hier um einen ungewünschten Abbau der Verbindung aufgrund einer Ausnahmesituation. Diese Phase kommt also im regulären Betrieb nicht vor.

Die Ausnahmesituation kann in den unterliegenden Schichten aufgetreten sein, wofür das Dienstprimitiv PAbo zur Verfügung steht:

- Provider Abort (PAbo): werden durch den Dienst selbst oder die Diensterbringer tieferer Schichten verursacht
- User Abort (UAbo): Instanzen der über den Dienst kommunizierenden Dienstnutzer oder höhere Schichten

Der Dienst UAbo wird von der Instanz der betrachteten Schicht genutzt, wenn die Ausnahmesituation gerade in dieser Schicht auftritt und statt eines geordneten Verbindungsabbaus ein erzwungener Verbindungsabbruch notwendig ist.



Ein Weg-Zeit-Diagramm beschreibt das zeitliche Verhalten an den beteiligten Dienstzugangspunkten. Dieser von außen beobachtbare Ablauf ist aufgrund der Ursache-Wirkung-Beziehung von Dienstprimitiven gegeben, die sich geeignet durch sogenannte *Zustandsübergangsdiagramme* spezifizieren lässt (die Arbeitsweise eines solchen Diagramms entspricht im wesentlichen dem eines endlichen Automaten). Wie die Abbildung andeutet, wird damit das Verhalten des unterliegenden Mediums (Dienstes) beschrieben. Dieses befindet sich in einem gewissen Zustand – im Beispiel ist zwischen drei unterschiedlichen Zuständen zu unterscheiden. Gewisse eingehende Dienstprimitive werden in jedem Zustand akzeptiert und führen zu Zustandsübergängen: im Zustand „Verbindung in Aufbau“ existieren zu den Primitiven Con.Rsp und Dis.Req Übergänge; außerdem kann hier auch noch ein spontaner Übergang durch einen Abbruch stattfinden. Die Zustandsübergänge werden also durch das Dienstprimitiv ausgelöst, das vor dem „;“ angegeben ist. Nach dem Semikolon sind ein oder mehrere Dienstprimitive angegeben, die an den entsprechenden SAPs auszugeben sind. Die Beschriftung kann also als „Ereignis ; ausgelöste Aktion“ verstanden werden.

- Syntax: <auslösendes Dienstprimitiv ; resultierendes Dienstprimitiv>, z.B. Con.Req;Con.Ind
- Falls die erste Angabe fehlt, handelt es sich um einen spontanen Übergang.

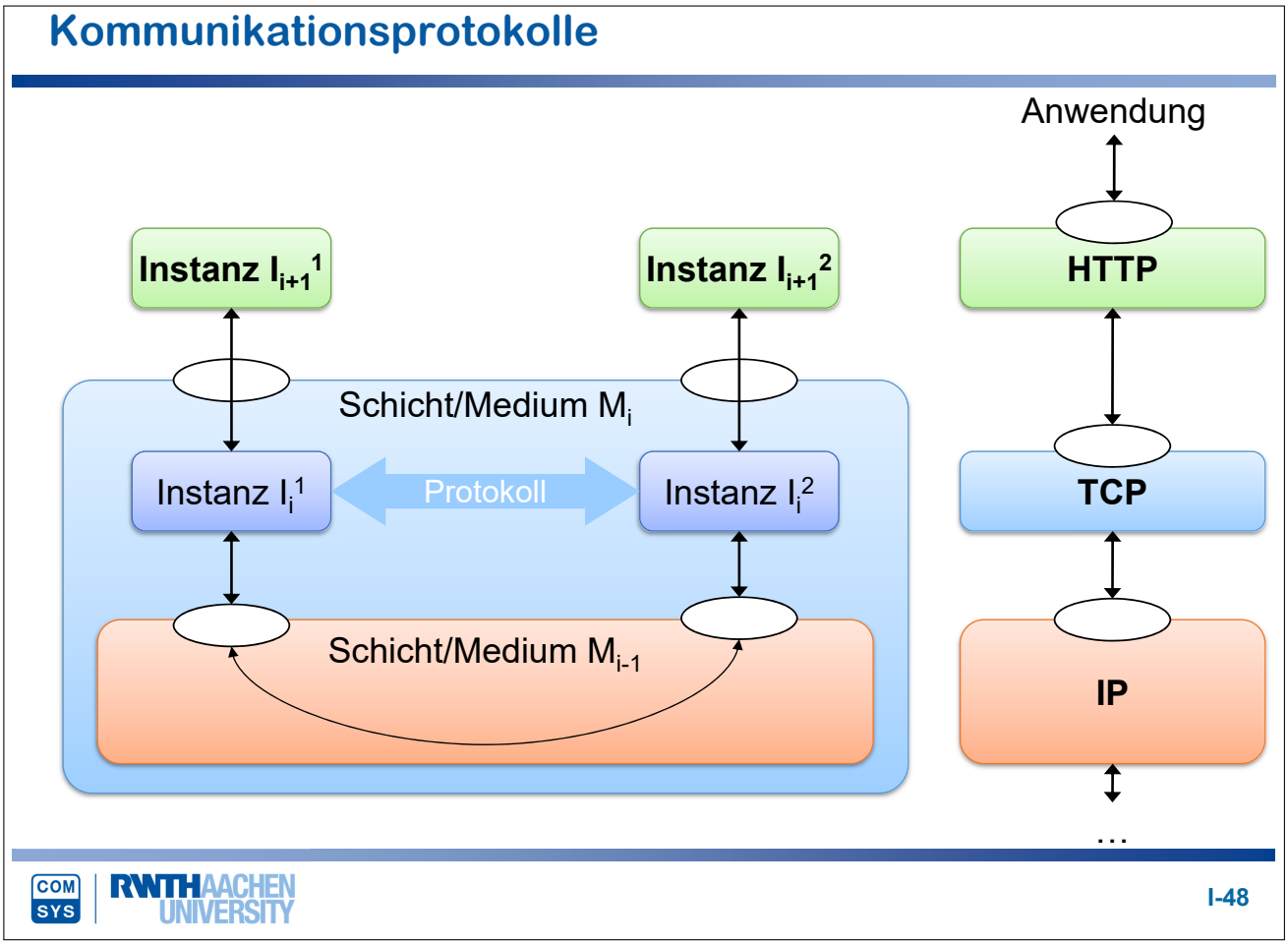
Ein Weg-Zeit-Diagramm beschreibt einen konkreten Weg durch das Zustandsübergangsdiagramm, durch welches das Verhalten des Mediums dargestellt ist. Das Zustandsübergangsdiagramm verfügt also über größere Ausdrucksstärke.

Der gewünschte Normalweg führt vom Ruhezustand durch Con.Req zum „Verbindung in Aufbau” und durch Con.Rsp zu „Verbindung aufgebaut”. Von hier geht es dann in die diversen anderen Dienstphasen des Datenaustauschs und des Verbindungsabbaus.

Zu bemerken ist noch, dass in obigem Beispiel die Initiative zum Verbindungsaufbau

immer vom Dienstnehmer am SAP A ausgehen muss, während der Dienstnehmer am SAP B eine Verbindung nur akzeptieren, ablehnen oder auch abbauen kann. Im Normalfall werden jedoch beiden Dienstnehmern sämtliche Dienstprimitive zur Verfügung stehen (d.h. alle Aktionen können sowohl von der einen wie auch von der anderen Seite ausgelöst werden). SAP A und B beschreiben also keine konkreten Geräte, sondern eher mögliche Rollen bei der Dienstnutzung.

- **Einführung und Begriffe**
  - Was ist Datenkommunikation?
  - Information, Daten, Signale
  - Netze
- **Allgemeine Grundlagen**
  - Dienste
  - Protokolle und Schichten
  - Kommunikationsarchitekturen



Bislang wurde behandelt, welche Kommunikationsfunktionalität in Form von Diensten an einer Schichtgrenze (der Dienstschnittstelle) angeboten wird. D.h.

- Der Kommunikationsdienst beschreibt, welche Funktionalität (was) angeboten wird.
- Das Kommunikationsprotokoll beschreibt, wie diese Funktionalität erbracht wird.

Zur Untersuchung der Protokolle ist es also notwendig, in die Schicht hineinzuschauen und die darin ‚arbeitenden' Komponenten zu beleuchten. Im Kommunikationsbereich hat sich für diese Komponenten der Begriff der „Instanzen" (engl. *entities*) durchgesetzt.

Ein Protokoll legt das Verhalten einer Instanz (*entity*) fest. Je zwei Partnerinstanzen (*peer entities*) realisieren das Protokoll und erbringen dadurch den entsprechenden Kommunikationsdienst.

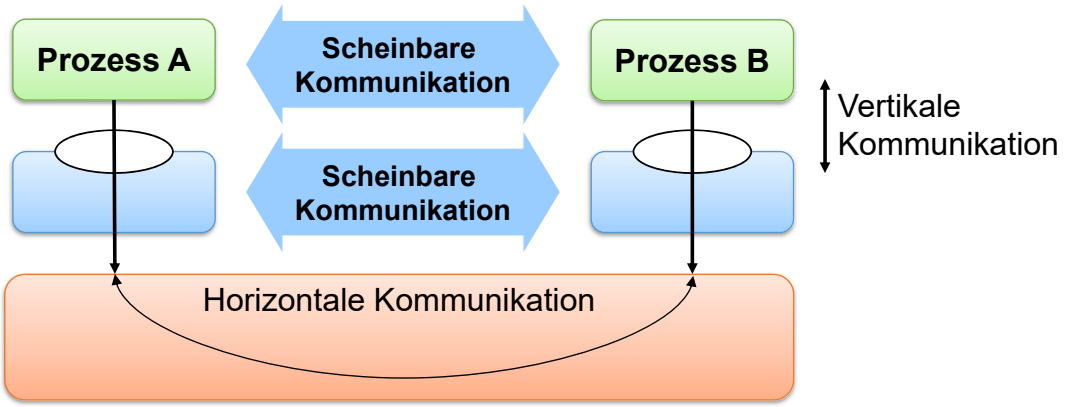
Charakteristisch für ein Kommunikationsprotokoll ist, dass es zwischen zwei Protokollinstanzen abläuft. Im Folgenden soll der oben angedeutete Zusammenhang zwischen Protokoll und Dienst verdeutlicht werden.

Die Partnerinstanzen, zwischen denen das Protokoll abläuft, erbringen einen Dienst des Mediums M_i und setzen dazu auf einem Dienst des Mediums M_{i-1} auf.

In der Abbildung auf der Folie wird gemäß dem zuvor eingeführten Dienstmodell von Medium gesprochen. Dieser Begriff ist gleichbedeutend mit dem Begriff der Schicht oder Ebene. In diesem Beispiel sind die Instanzen I_i^1 *und* I_i^2 die Partnerinstanzen. Sie erbringen für die darüberliegenden Instanzen Schicht-*i*-Dienste und nutzen die vom darunterliegenden Medium angebotenen $Medium_{i-1}$-Dienste.

Scheinbare & tatsächliche Kommunikation

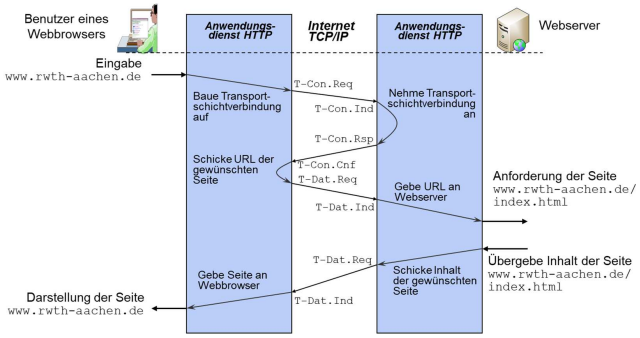
- **Unterschied: Horizontale und vertikale Kommunikation**
  - *Horizontal*: Kommunikation zwischen Instanzen der Protokolle
    - Kommunikation scheinbar direkt zwischen Kommunikationspartnern
  - *Vertikal*: tatsächliche Kommunikation innerhalb eines Protokollstapels
    - Tiefer liegenden Schichten erbringen den Kommunikationsdienst

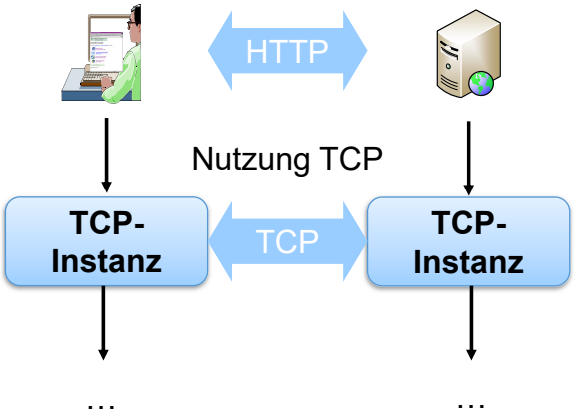


Kommunikationsprotokolle regeln intern innerhalb einer Schicht den Datenaustausch zwischen Instanzen der jeweiligen Schicht. Die Kommunikation der Instanzen über das Protokoll findet scheinbar direkt zwischen den Instanzen statt. Dies wird auch horizontale Kommunikation genannt. Nur auf der untersten Schicht findet horizontale Kommunikation über das physikalische Medium tatsächlich statt; auf allen höheren Ebenen ist die horizontale Kommunikation nur virtuell.

Tatsächlich erbringen aber die tiefer liegenden Schichten den Kommunikationsdienst. Die Protokolldateneinheiten werden an die nächsttiefer liegende Schicht weitergereicht und durch diese zur Protokollinstanz des Kommunikationspartners gebracht. Da die nächsttiefere Schicht dadurch eventuell wieder auf tiefere Schichten zurückgreift, durchlaufen die zu versendenden Daten einen Stapel an Diensten, die durch eigene Protokolle realisiert werden. Man spricht hierbei von einem Protokollstapel (Protokoll-Stack). Durch diesen Protokoll-Stack werden die Daten vertikal weitergereicht.

Scheinbare & tatsächliche Kommunikation

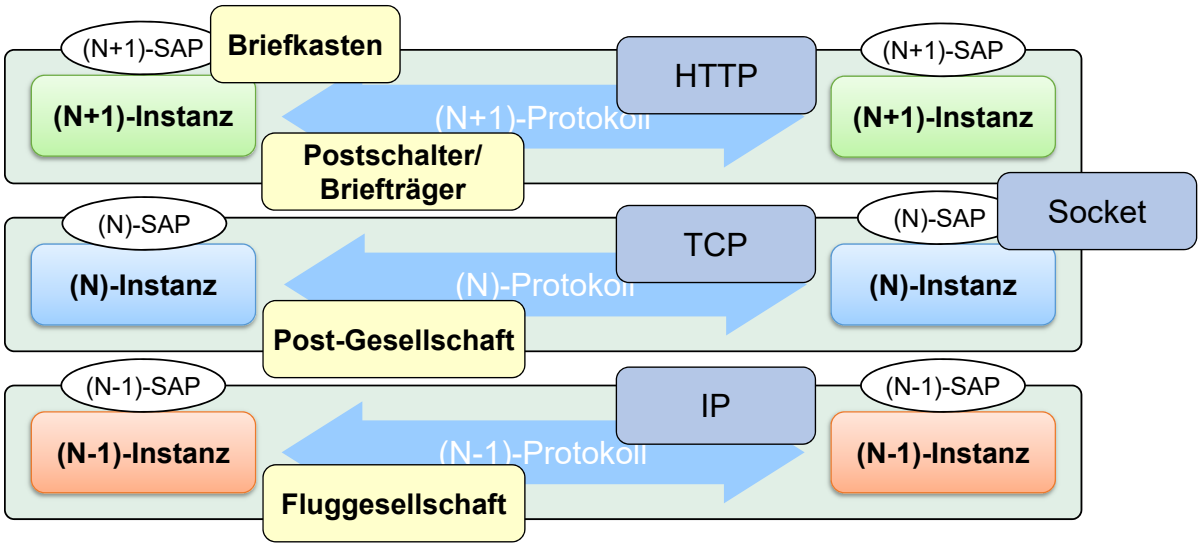




- **Beispiel Webseite**
  - Browser und Webserver kommunizieren (scheinbar) horizontal
  - Tatsächliche Kommunikation vertikal via TCP
  - TCP-Instanzen kommunizieren ebenso (scheinbar) horizontal
  - …

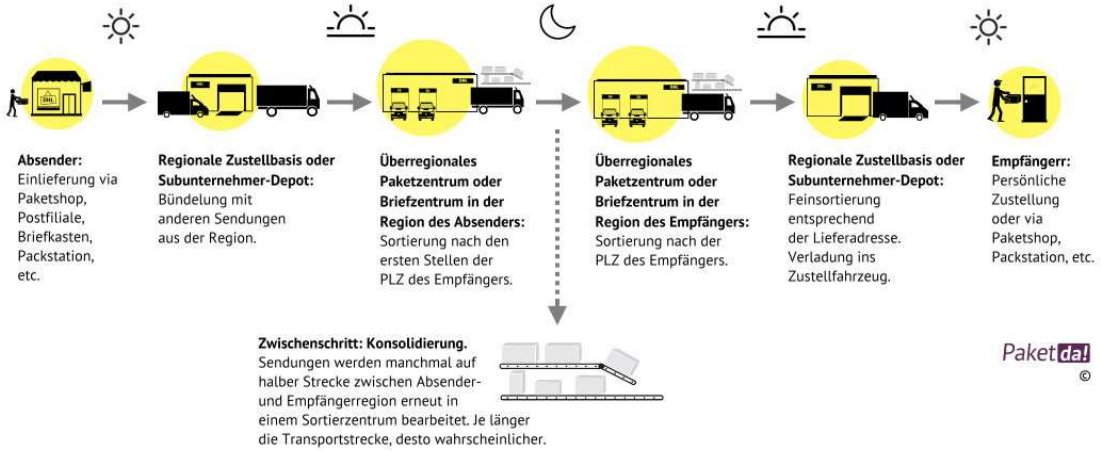
Protokolle und Protokollinstanzen

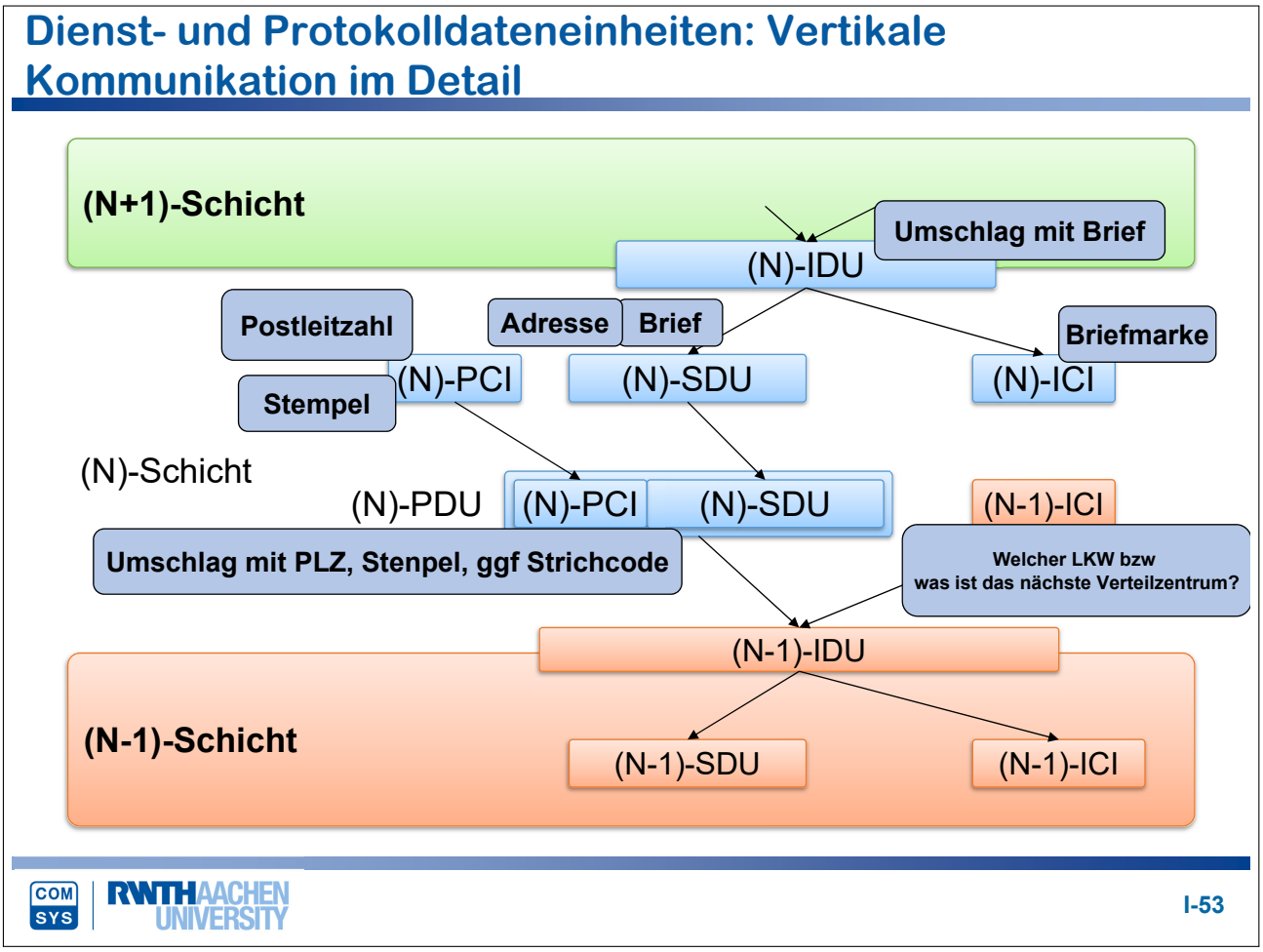
- *Protokolle*: Regeln/Formate für den Datenaustausch zwischen Rechnersystemen
- *Protokollinstanzen*: Protokollimplementierungen in den Rechnersystemen
  - *Schichtung* von Protokollinstanzen



Der Dienst, den eine Schicht erbringt, wird durch das Zusammenwirken von Protokollinstanzen dieser Schicht (knapp auch Schichteninstanzen) auf den beteiligten Rechnersystemen erbracht – eine Schicht zieht sich also über ein gesamtes Kommunikationssystem hinweg. Diese Protokollinstanzen sind konkrete Implementierungen eines Protokolls auf einem Rechnersystem. Das Protokoll enthält alle Regeln, die für die Erbringung des Dienstes auf dieser Schicht erforderlich sind, sowie Formate, in denen die Daten zwischen den Protokollinstanzen ausgetauscht werden. Das Protokoll wird stets nur zwischen den Instanzen auf dieser Schicht abgearbeitet.

Auf einer Schicht existieren meist unterschiedliche Protokolle, die verschiedene Dienste erbringen können – wie beispielsweise ein Protokoll für verbindungsorientierte Kommunikation und ein Protokoll für verbindungslose Kommunikation, welche die vorher bereits dargestellten Dienstprimitive und Regeln zu ihrem Austausch definieren.


Absender:
Einlieferung via Paketshop, Postfiliale, Briefkasten, Packstation, etc.
Regionale Zustellbasis oder Subunternehmer-Depot:
Bündelung mit anderen Sendungen aus der Region.
Überregionales Paketzentrum oder Briefzentrum in der Region des Absenders:
Sortierung nach den ersten Stellen der PLZ des Empfängers.
Überregionales Paketzentrum oder Briefzentrum in der Region des Empfängers:
Sortierung nach der PLZ des Empfängers.
Regionale Zustellbasis oder Subunternehmer-Depot:
Feinsortierung entsprechend der Lieferadresse. Verladung ins Zustellfahrzeug.
Empfängerr:
Persönliche Zustellung oder via Paketshop, Packstation, etc.
Zwischenschritt: Konsolidierung.
Sendungen werden manchmal auf halber Strecke zwischen Absender- und Empfängerregion erneut in einem Sortierzentrum bearbeitet. Je länger die Transportstrecke, desto wahrscheinlicher.
Paket da!
©



Im sogenannten ISO/OSI-Referenzmodell (wird später erläutert) wird das Zusammenspiel von zwei Instanzen an der Dienstschnittstelle in einem Dienstmodell genau festgelegt. Die Zusammenarbeit erfolgt in den folgenden Schritten:

(1) Die *(N+1)*-Instanz übergibt an der Dienstschnittstelle eine *(N)*-Interface Data Unit (IDU).

(2) Die *(N)*-Instanz teilt die *(N)*-IDU in zwei Teile auf:

a) transparent zu übertragende Nutzdaten: *(N)*-SDU (Service Data Unit)

b) Steuerinformationen für die Dienstschnittstelle: *(N)*-ICI (Interface Control Information)

(3) Zur Übertragung der *(N)*-SDU ist gemäß dem vereinbarten Kommunikationsprotokoll eine *(N)*-PCI (Protocol Control Information) zu erzeugen, die gemeinsam mit der *(N)*-SDU die *(N)*-PDU bildet. Im Verlauf der Vorlesung werden die PCI als Header bezeichnet, da sie meist der SDU vorangestellt werden.

Diese (N)-PDU wird transparent zwischen den Protokollinstanzen der Schicht N übertragen.

(4) Zur Übertragung der PDU durch die darunterliegende Schicht ist entsprechende Kontrollinformation für die untere Schnittstelle zu erzeugen, die *(N)*-PDU und diese *(N-1)*-ICI bilden die *(N-1)*-IDU.

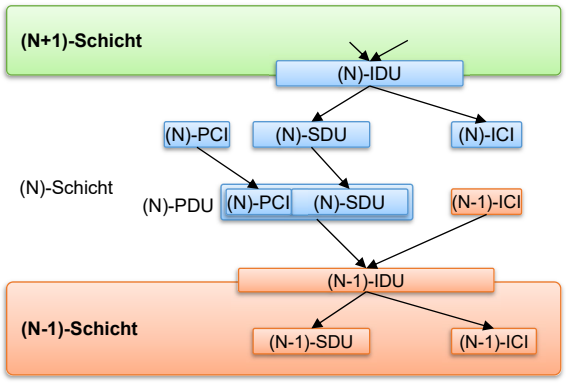
Zusammenfassung der Begriffe:

- *(N)*-Schnittstellendateneinheit (IDU): Dies ist die Dateneinheit, die die *(N+1)*-Schicht an die *(N)*-Schicht über den Dienstzugangspunkt (Service Access Point - SAP) übergibt. Sie enthält die Nutzdaten *(N)*-SDU und Schnittstellenkontrollinformationen *(N)*-ICI.
- *(N)*-Schnittstellenkontrollinformationen (ICI): Diese Dateneinheit enthält Informationen über den auszuführenden Dienst, z.B: Parameter des Dienstes, Länge der SDU, ...
- *(N)*-Dienstdateneinheit (SDU): Daten der *(N+1)*-Schicht (*(N+1)*-PDU), die transparent zwischen *(N)*-SAPs übertragen werden.
- *(N)*-Protokollkontrollinformationen (PCI): Informationen, die (über vertikale Kommunikation) zwischen den einzelnen *(N)*-Instanzen ausgetauscht werden, um den Protokollablauf zu steuern.
- *(N)*-Protokolldateneinheit (PDU): Daten, die zwischen *(N)*-Partnerinstanzen ausgetauscht werden. Eine *(N)*-PDU setzt sich aus *(N)*-PCI und *(N)*-SDU zusammen.

Packets inside the Linux Kernel

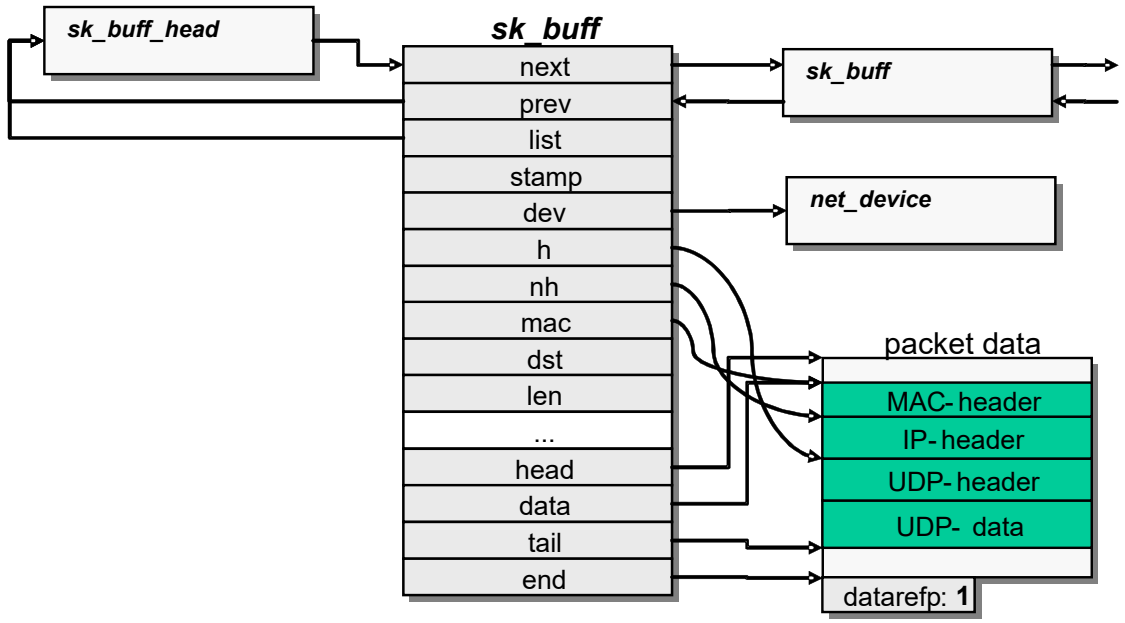
Auszug aus der Vorlesung
Communication Systems Engineering

- **Some questions:**
  - How are packets represented in the Kernel?
  - How to realize SDU, PDU, IDU, ICI, etc.?
  - Which variables do we need?
  - One structure for all protocols
    or individual structures for each protocol?
  - …

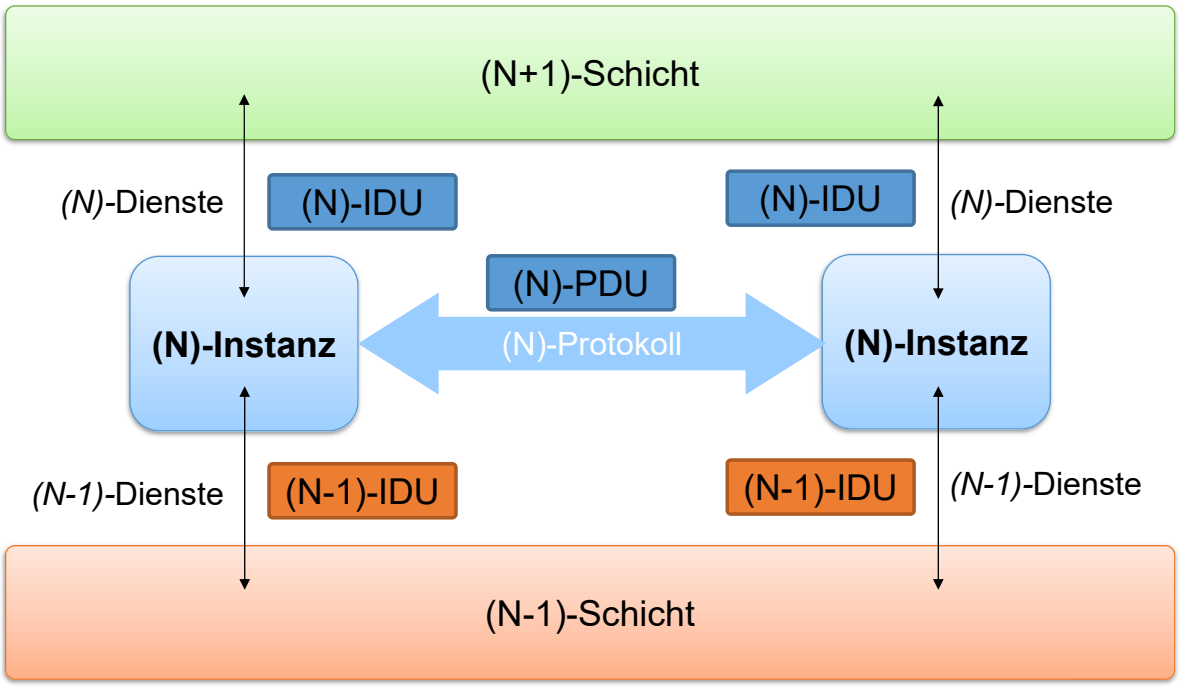


Socket Buffer: Represents Packets in the Linux Kernel

Auszug aus der Vorlesung
Communication Systems Engineering

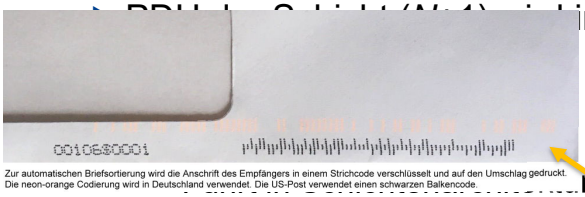


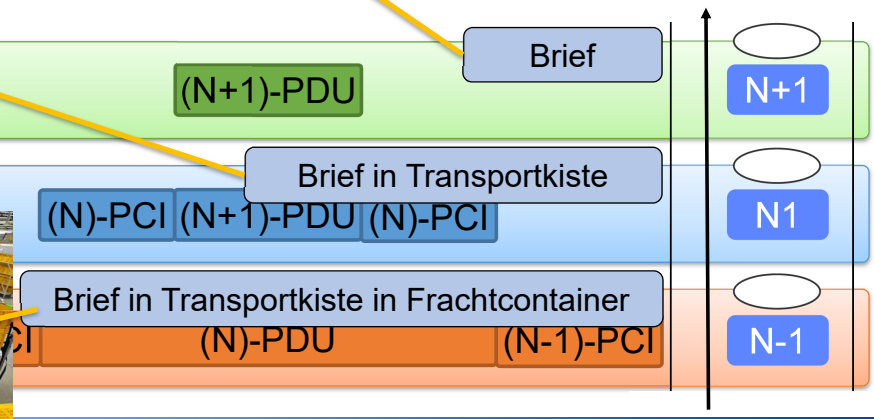
Kommunikation innerhalb/zwischen Systemen



Transparenz der Schichten

- **Transparenz**
  - in Schicht *N* als Nutzdaten betrachtet
  - Inhalts, *Transparenz* der Benutzerdaten (SDU)
  - in den unteren Schichten zu einer ständigen
  - ng des Anteils der PCIs am gesamten Datenumfang



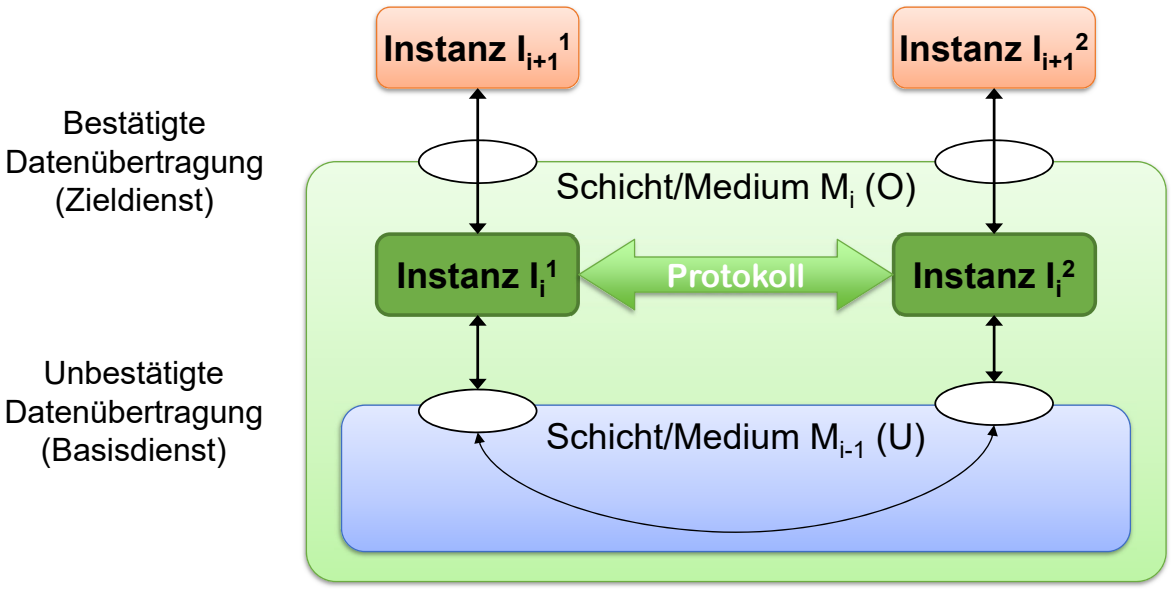


Jede Schicht soll unabhängig von allen anderen implementiert werden können, um Interoperabilität zwischen beliebigen Implementierungen auf allen Schichten zu gewährleisten. Dies hat aber auch einen Nachteil: die PDU einer Schicht, die an die nächsttiefere Schicht weitergereicht wird, wird dort als SDU aufgefasst – als Nutzdateneinheit, die unangetastet bleibt und mit neuen Kontrollinformationen versehen wird. Dadurch besteht eine PDU auf unterster Ebene aus den eigentlichen Nutzdaten selbst plus einer ganzen Kette an PCIs. Die zu übertragene Datenmenge wird umso mehr aufgebläht, je mehr Schichten verwendet werden. Dies kann einen enormen Overhead erzeugen – und das nicht nur bei der Menge der zu übertragenen Informationen, sondern auch bei der Verarbeitung der Daten durch den gesamten Protokoll-Stack.

- **Beispiel: Alternating Bit Protocol**
  - Dienst vs. Protokoll
  - Interaktion zwischen Schichten

Im Folgenden werden die eingeführten Konzepte von Dienst, Protokoll und Schichtung anhand eines einfachen Beispiels im Zusammenhang dargestellt.

Beispiel: „Alternating Bit Protocol“



Als Beispiel soll ein Dienst betrachtet werden, den ein Medium „O“ erbringt. Dieses Medium bietet die Funktionalität „bestätigte Datenübertragung“. Dazu ist die Definition eines Protokolls zwischen den Instanzen des Diensterbringers notwendig. Das Protokoll erbringt die virtuell horizontale Kommunikation zwischen den Instanzen, nutzt zur Übertragung aber einen Basisdienst, den das Medium „U“ anbietet – dieser Basisdienst ist unbestätigt und unzuverlässig.

Alternating Bit Protocol: Informelle Beschreibung

- **Dienst zur bestätigten Datenübertragung**
  - Auf Basis eines Dienstes zur unbestätigten Datenübertragung
  - Nötig: *Quittungen*
    - Mit alternierenden Quittungsnummern 0 und 1
    - Nachdem eine quittierte Datenübertragung vollständig beendet ist (Quittung erhalten), geht die Instanz in einen Ruhezustand über, bis die nächste Übertragungsanforderung eintrifft
  - Dienstprimitive
    - `DtReq`
    - `DtInd`
    - `DtRsp`
    - `DtCnf`

Als Zusammenfassung und Verdeutlichung der eingeführten Konzepte wird das Alternating Bit Protocol betrachtet. Es ist ein einfaches Protokoll, das eine elementare und wichtige Funktionalität innerhalb der Datenkommunikation erbringt, nämlich

- (i-1)-Dienst: unbestätigte und unzuverlässige Datenübertragung
- i-Dienst: bestätigte Datenübertragung

Um diese Dienstdifferenz von einem unbestätigten, unzuverlässigen zu einem bestätigten Dienst ausgleichen zu können, muss das Protokoll eine Quittierung einführen, durch die letztendlich sichergestellt wird, dass die übertragenen Daten auch beim Empfänger angekommen sind.

Protokollfunktionalität: Quittierung und Fehlerbehandlung

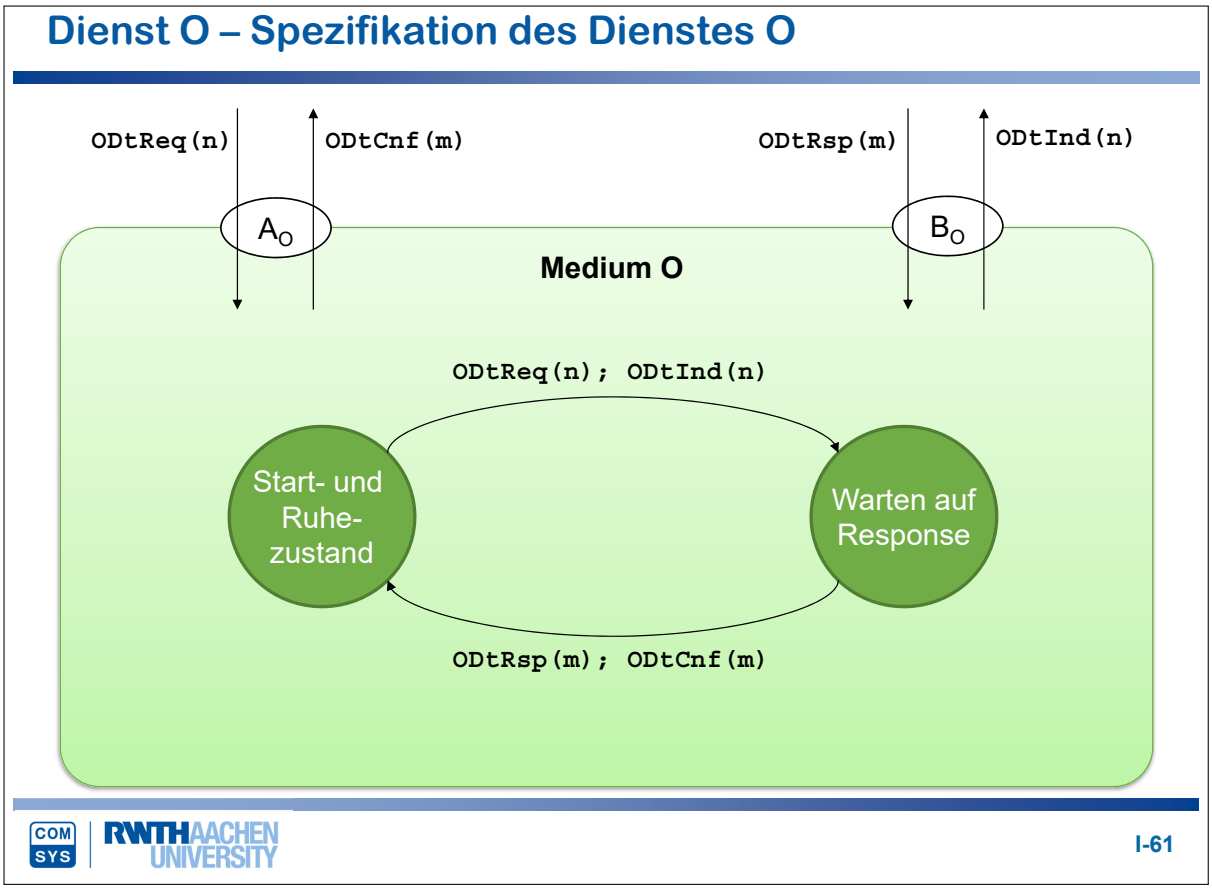
Die Beschreibung der Protokollfunktionalität umfasst drei Aspekte:

• Ablauffestlegungen innerhalb einer Instanz: Zustandsübergangsdiagramm (endlicher Automat - Protokollautomat)

• Bei den Übergängen zu erbringende Funktionalität: Erweiterung des endlichen Automaten um Aktionen auf interne Variablen

• Protokolldateneinheiten: Bitmuster oder Datenstruktur

Das Beispielprotokoll kommt aufgrund seiner Einfachheit noch ohne die Aktionen (Punkt 2) aus. Die beiden anderen Aspekte werden im folgenden anhand des Beispielprotokolls aufgezeigt.

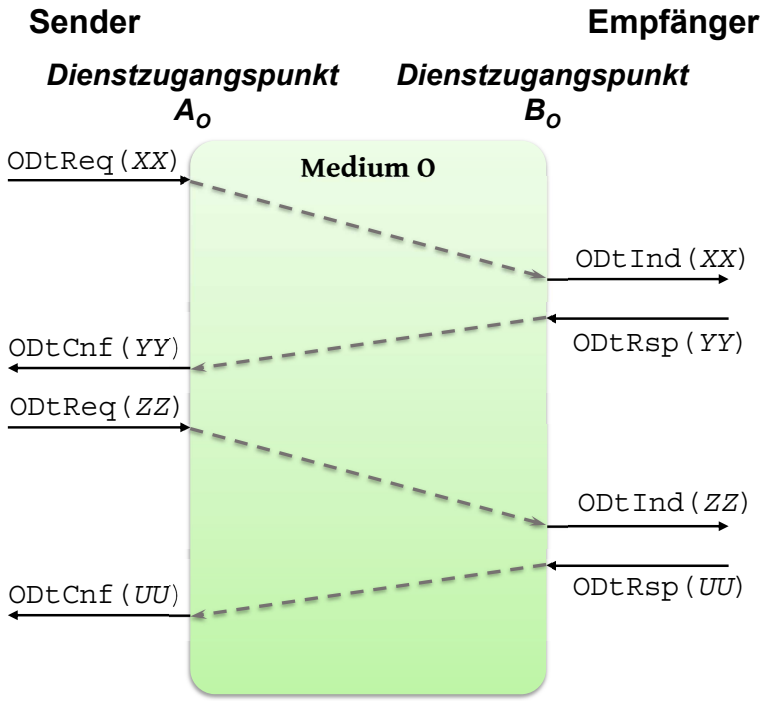


Wie das Diagramm (sehr grober Protokollautomat) für den zu realisierenden Dienst O aus dem Beispiel zeigt, handelt es sich hier um einen einfachen Dienst mit nur einer Dienstfunktion ODt, die einen bestätigten Datentransfer realisiert.

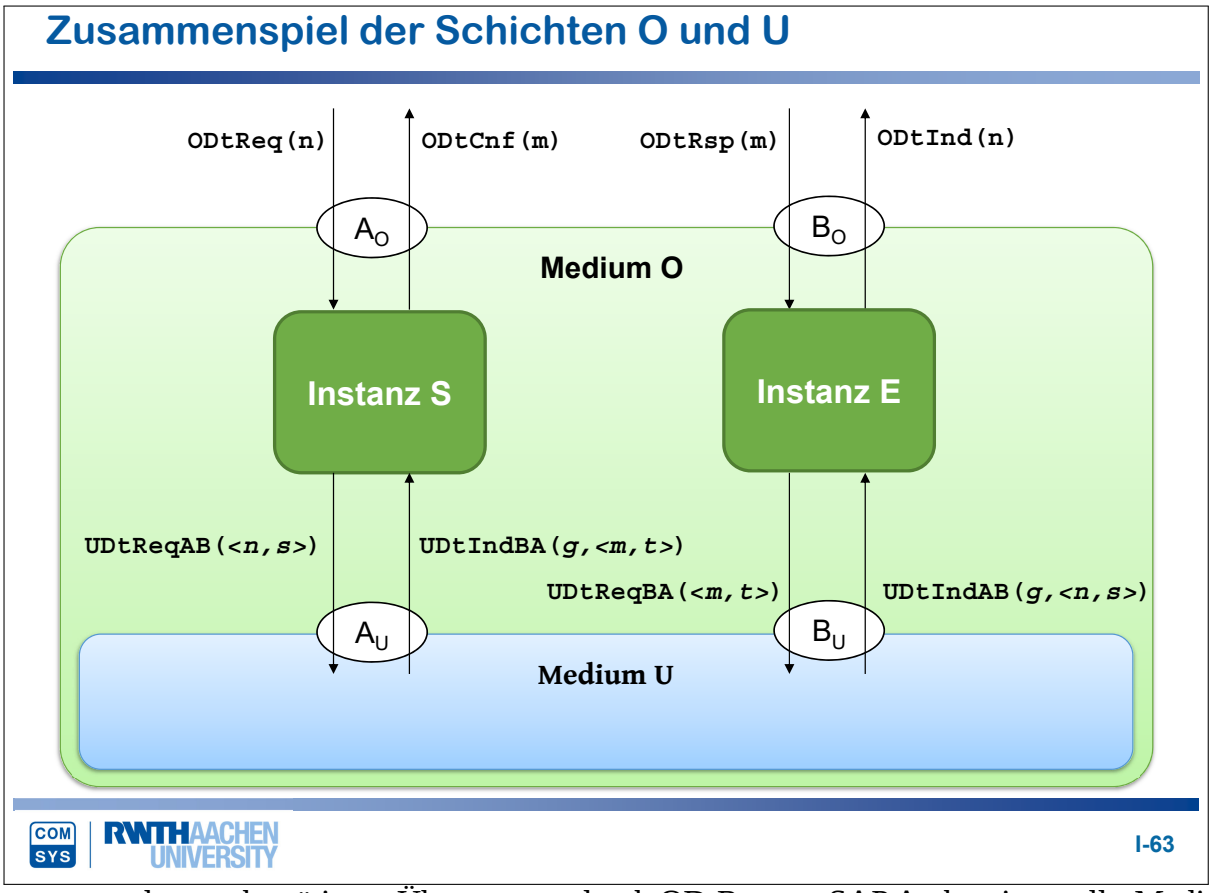
Aus der Sicht des Dienstnutzers von O werden sämtliche Störungen, wie Bitfehler oder Verlust ganzer Dateneinheiten, vollständig durch das Protokoll des Dienstes verdeckt. (In der Realität geht das nur bis zu einem gewissen Grad, weshalb man i.d.R. immer noch Ausnahmesituationen bei einer solchen Dienstschnittstelle berücksichtigen sollte.)

Die Initiierung der bestätigten Datenübertragung erfolgt über den Zugangspunkt A_O – dieser beschreibt die Rolle des sendenden Dienstnutzers. Zugangspunkt B_O beschreibt dementsprechend die Rolle des empfangenden Dienstnutzers. Über B_O werden zwar auch Dateneinheiten übertragen, die aber lediglich eine Bestätigung transportieren.

Dienstablauf in einem Weg-Zeit-Diagramm



Die Arbeitsweise eines Protokolls für O bzw. der Instanz, die dieses Protokoll ausführt, lässt sich am besten begreifen, wenn man typische Abläufe durch den Protokollautomaten nachvollzieht. Auf dieser Folie ist ein solcher Ablauf in einem Weg-Zeit-Diagramm dargestellt.



Die Nutzdaten *n* werden zur bestätigten Übertragung durch ODtReq am SAP A_O bereitgestellt. Medium O realisiert nun das Alternating Bit Protocol. Die Protokollinstanz *S*(ender) benutzt den unbestätigten Übertragungsdienst am unteren Zugangspunkt A_U und überträgt neben den Nutzdaten *n* außerdem eine Sequenznummer *s*. Diese wird bei der ersten Übertragung auf 0 gesetzt. Über das Medium wird also <*n*,0> übertragen.

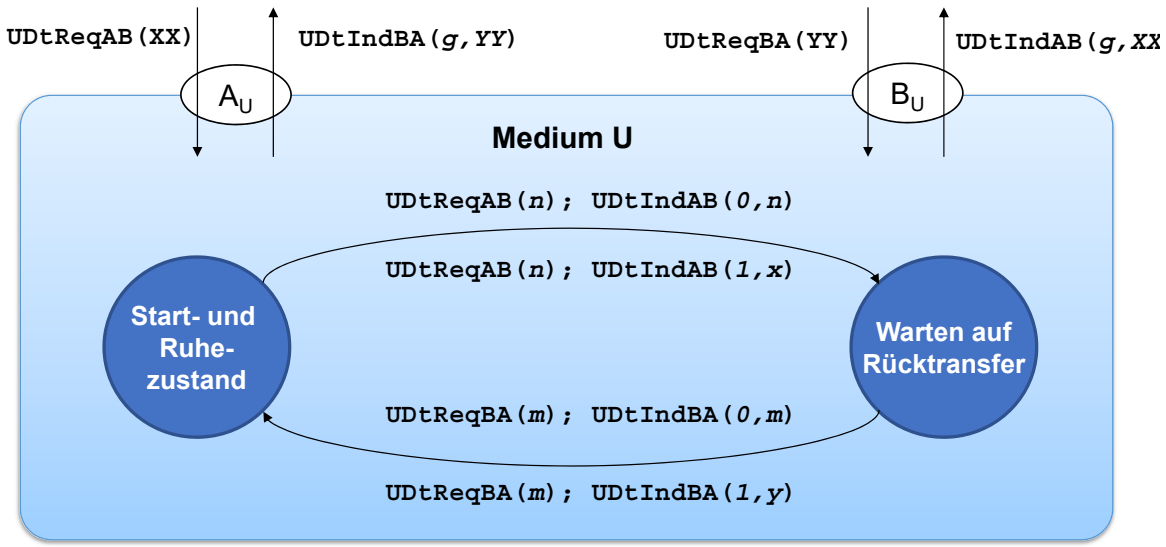
Bei einem störungsfreien Protokollablauf würden die Daten über B_U an die Protokollinstanz *E*(mpfänger) ausgeliefert (UDtIndAB), und über ODtInd an den Dienstnutzer des Mediums O weitergereicht. Der Dienstnutzer generiert eine Quittung *m*, welche durch E wiederum um einen Kontrollparameter *t* erweitert wird, bevor sie über UDtReqBA an S zurückgesendet wird. Der Parameter t dient zur Quittierung des vorherigen Requests und ist ebenfalls eine Sequenznummer. Ist der Wert gleich zu dem vorher verwendeten Wert für *s*, wird die Übertragung als erfolgreich angesehen und die enthaltene Bestätigung *m* an den Dienstnutzer des Mediums O weitergereicht.

Anhand des Beispielablaufs lässt sich ein wichtiger Aspekt, der bei einer Protokollbeschreibung auftritt, feststellen:

In einer Protokollbeschreibung ist neben dem Zusammenwirken der am oberen und unteren Zugangspunkt auftretenden Dienstprimitive auch genau zu definieren, welche Protokolldateneinheiten (Protocol Data Units, PDU) über das Medium auszutauschen sind. In diesem Fall ist es der Kontrollparameter „Sequenznummer“, der zusammen mit den zu übertragenden Nutzdaten die PDU bildet. Die Information, die in Form von solchen in ihrer Syntax genau festzulegenden PDUs beschrieben werden muss, wird im Beispiel immer in den spitzen Klammern <PDU-Inhalt> geschrieben.

Zusätzlich zur Sequenznummer wird ein Kontrollparameter *g* eingeführt, der allerdings kein Bestandteil der PDUs im Medium O mehr ist, sondern der Interaktion zwischen U und O innerhalb eines Protokoll-Stacks dient. Über diesen Parameter kann eine Instanz des Mediums U der Instanz des Mediums O im geleichen Protokoll-Stack signalisieren, ob Daten korrekt empfangen wurden oder ob sie verfälscht wurden. *g* ist also eine Interface Control Information, während die Sequenznummer eine Protocol Control Information ist.

Basisdienst U – Spezifikation des Dienstes U



- **Parameter *g*: Signalisierung von U an O, dass die Daten während der Übertragung verfälscht wurden**
  - *g*=0: korrekter Empfang; *g*=1: Daten verfälscht

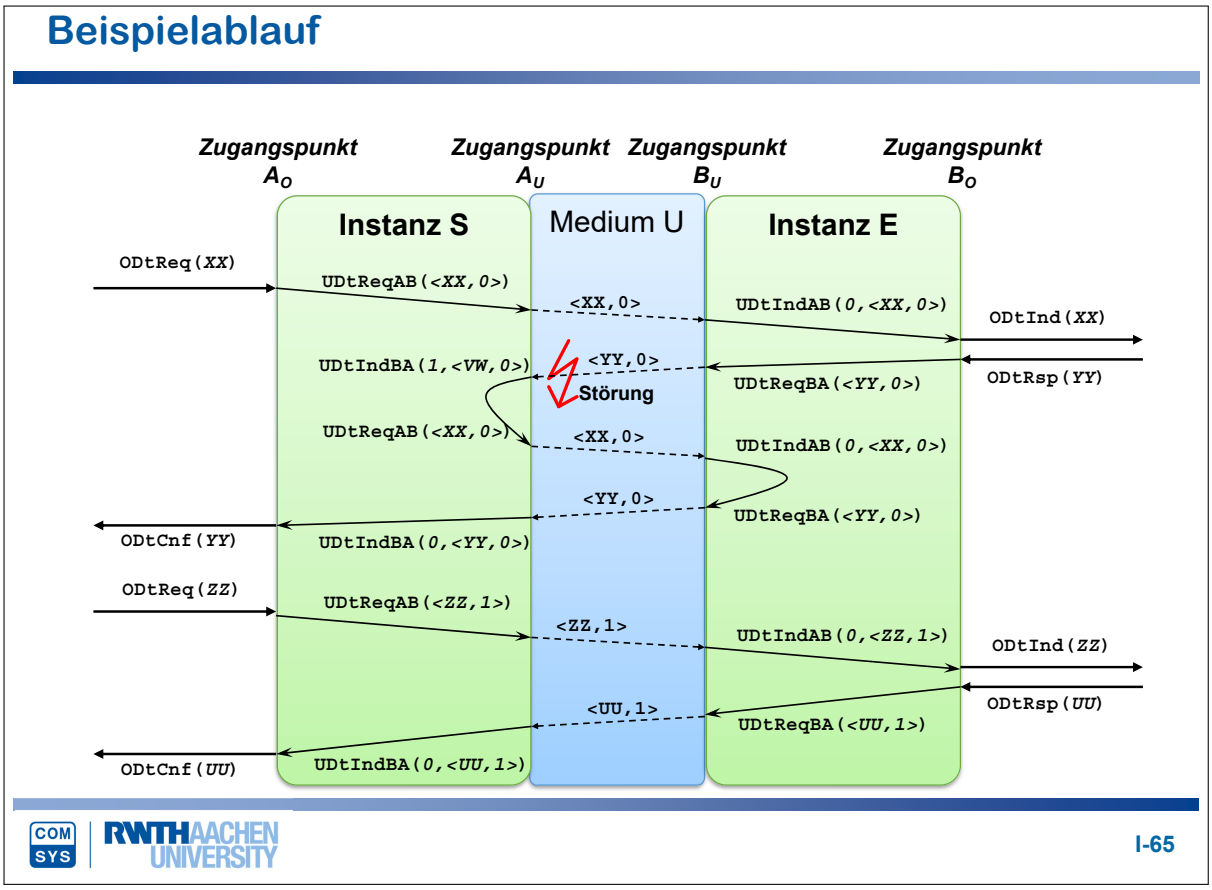
Der Basisdienst U unterstützt eine unbestätigte Datenübertragung in beide Richtungen. Aus der Sicht des Dienstnehmers (in diesem Fall die Instanzen S(ender) und E(mpfänger), die das Alternating Bit Protocol realisieren), ist diese Dienstschnittstelle also symmetrisch.

Betrachtet man jetzt aber das Zustandsübergangsdiagramm, erkennt man recht deutlich, dass zunächst eine Übertragung in der Richtung A→B erfolgen muss und daraufhin eine Übertragung in die entgegengesetzte Richtung erfolgt.

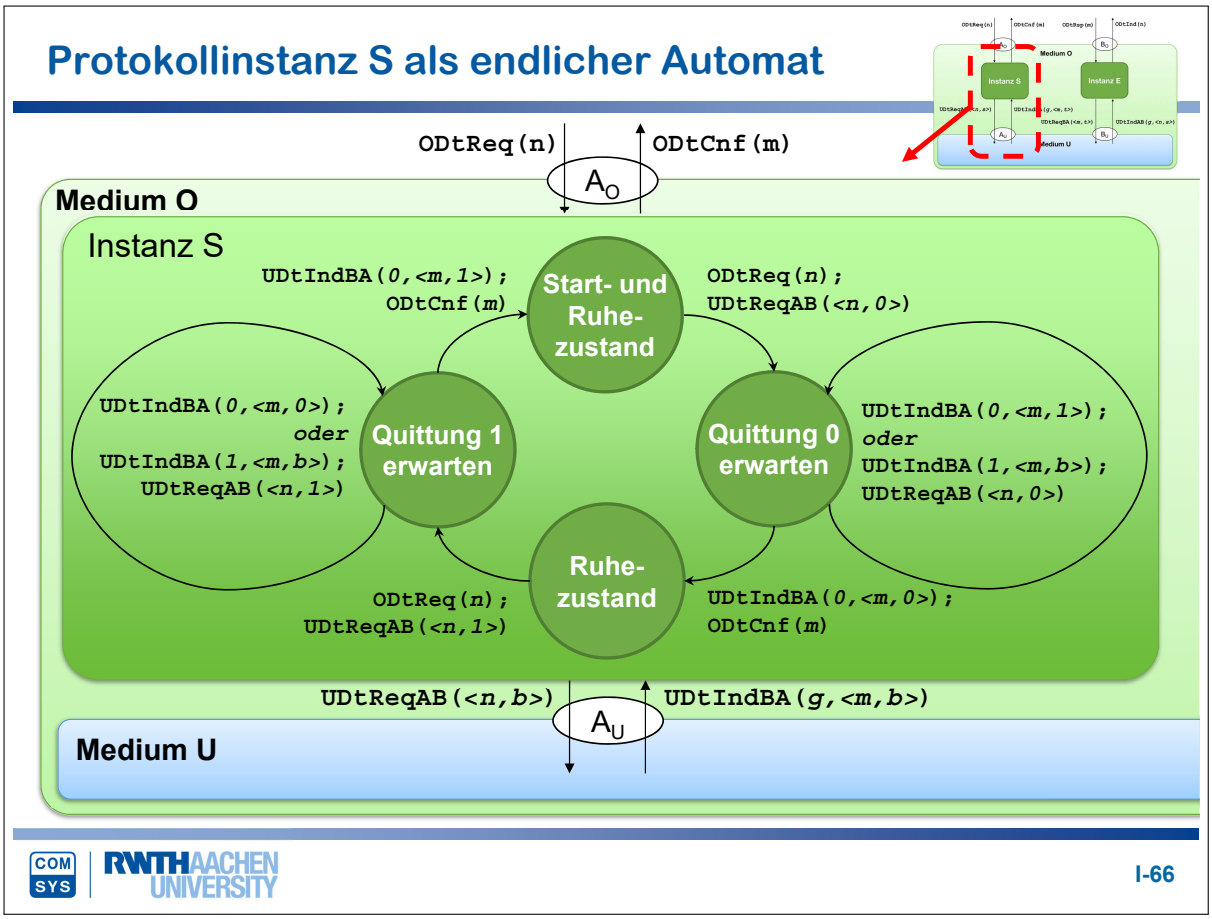
Der Basisdienst U berücksichtigt die Problematik, dass Störungen auftreten können, in folgender Weise:

Beim Basisdienst U werden dem Dienstnehmer medieninterne, nicht behebbare Fehler durch einen entsprechenden auf 1 gesetzten Kontrollparameter angezeigt.

In den endlichen Automaten, welche das Verhalten der Instanzen S und E beschreiben, wird dieser Parameter ausgewertet. Bei der Anzeige eines Übertragungsfehlers muss eine Übertragungswiederholung erfolgen, da der Dienstnutzer des Mediums O erwartet, dass die Daten übertragen werden. Zusammen mit der Beschreibung der Dienste U und O bilden die Instanzen S und E eine vollständige Beschreibung des vorliegenden Kommunikationsszenarios.



Hiermit sind alle vier Komponenten, also die zwei Instanzen S und E sowie der Ziel- und Basisdienst O und U, beschrieben.



Schnittstellen des Protokollautomaten für S:

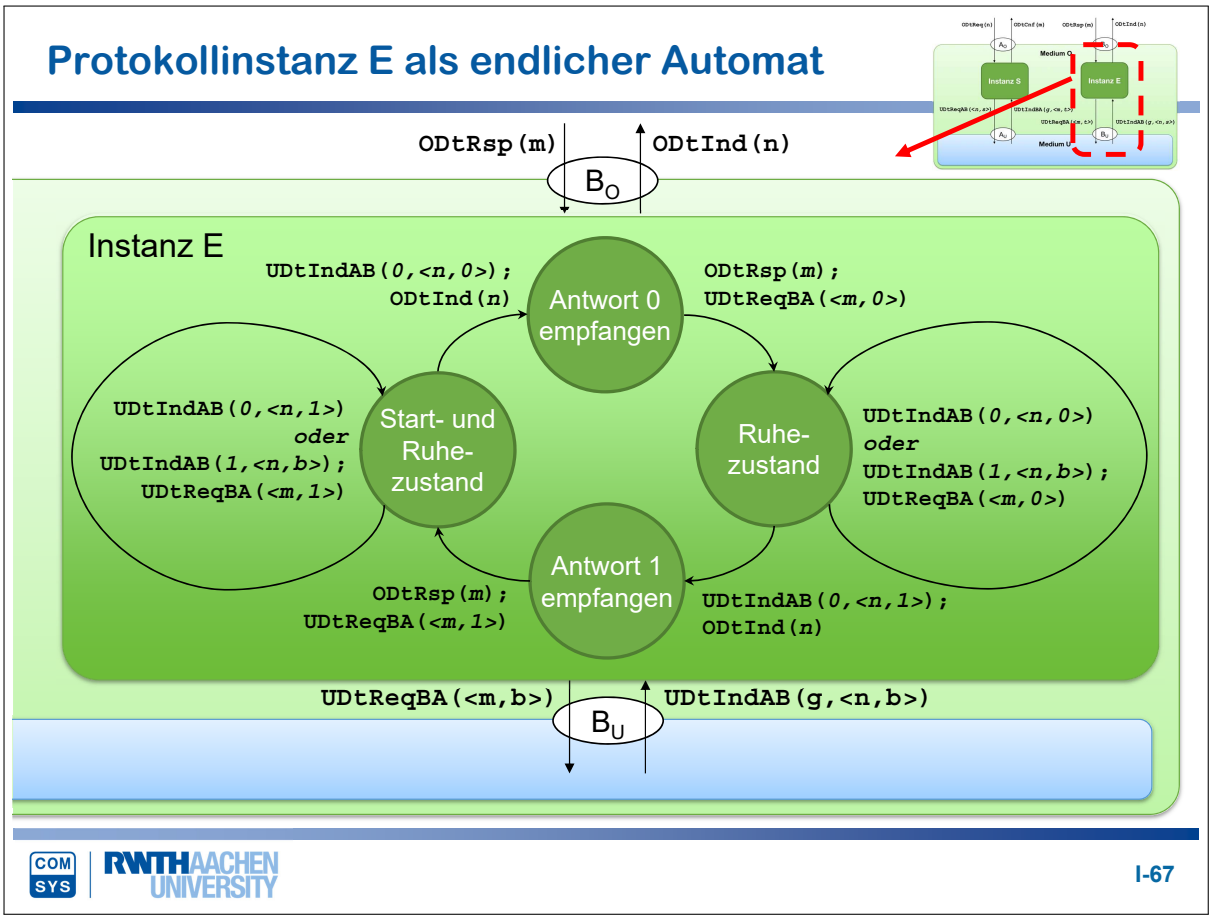
- ODtReq(n): Anforderung der Dienstleistung „bestätigte Datenübertragung der Nutzdaten n“
- UDtReqAB(<n,0>): Anforderung der Dienstleistung „unbestätigte Datenübertragung von einem Dienstzugangspunkt A_U nach B_U“ mit Nutzdaten n und Sequenznummer 0
- UDtIndBA(g,<m,b>): Anzeige der Dienstleistung „unbestätigte Datenübertragung von einem Dienstzugangspunkt B_U nach A_U“ mit den Parametern
  - g=Verfälschung ja/nein (1/0),
  - n, m=Nutzdaten,
  - b=Sequenznummer (0 oder 1)

Aus diesen Beschreibungen lässt sich die Bedeutung der Dienstprimitive ableiten, die die Ausgaben des Automaten bilden: ODtCnf(m) ist die Bestätigung der Übertragung von Nutzdaten m, UDtReqAB(<n,b>) ist die unbestätigte Übertragung der Nutzdaten n und der Sequenznummer b.

Der endliche Automat enthält insgesamt vier Zustände. Der Startzustand ist der oberste. Aus diesem kommt man in den Zustand “Quittung 0 erwarten” durch die Eingabe ODtReq(n) (eine Anfrage vom oberen Dienstzugangspunkt, Nutzdaten n bestätigt zu übertragen). Hierauf erfolgt die Ausgabe UDtReqAB(<n,0>), d.h. es werden die Nutzdaten n und die Sequenznummer b=0 an den unteren Dienstzugangspunkt A_U zur unbestätigten Übertragung weitergegeben.

In diesem Zustand “Quittung 0 erwarten” wartet die Instanz jetzt auf die Bestätigung, die durch eine entsprechende Anzeige UDtInd am unteren Dienstzugangspunkt mit der Sequenznummer 0 erfolgt.

Das Alternating Bit Protocol kommt deshalb mit einer alternierenden, d.h. nur die Werte 0 und 1 annehmenden Sequenznummer aus, weil eine Instanz immer erst eine weitere Nachricht senden darf, nachdem die zuvor gesendete Nachricht quittiert wurde.

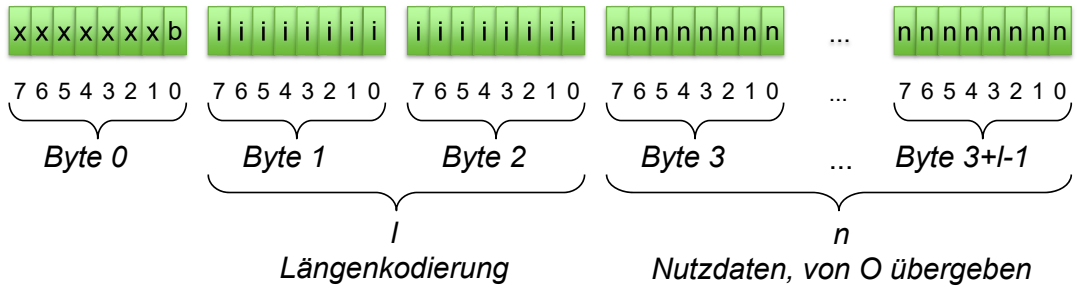


Die Auswertung des Verfälschungsparameters erfolgt je nachdem, wo man sich im alternierenden Ablauf gerade befindet, im “Start- und Ruhezustand” oder im “Ruhezustand” (selbiges für Instanz S in den anderen Zuständen). Die Reaktion der Instanz E auf eine Verfälschungsanzeige (also Wert = 1) besteht in der Übertragung von irgendwelchen Nutzdaten (hier: m) mit einem bewusst falsch gesetzten Sequenznummern-Wert. Im Beispiel: Instanz S erwartet anfangs die Sequenznummer 0, Instanz E antwortet bei einer Verfälschung aber mit der Sequenznummer 1.

Beim Alternating Bit Protocol wird eine negative Quittung und somit die Aufforderung zum nochmaligen Senden durch die Übertragung des gerade nicht-erwarteten Sequenznummern-Wertes erreicht.

Beschreibung von Protokolldateneinheiten

- **Format**



- Byte 0:
  - Kodierung der Sequenznummer *b* im niederwertigsten Bit
- Byte 1+2:
  - Kodierung der Länge *l* der Nutzdaten *n*
- Folgebytes:
  - Nutzdaten *n*

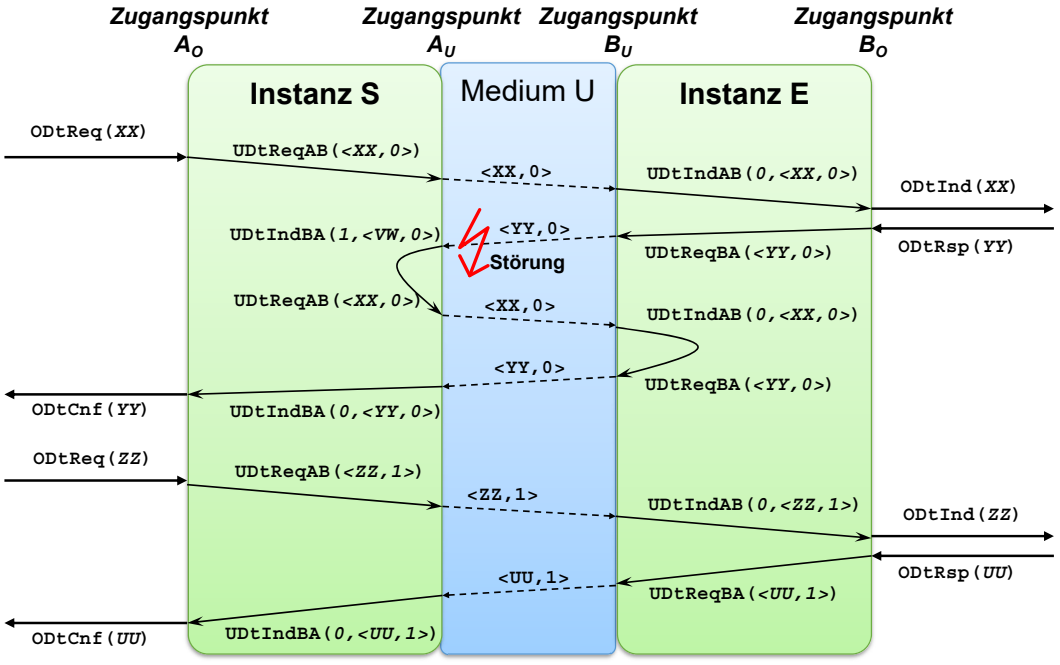
Die Protokolldateneinheiten unterscheiden sich von den Dienstprimitiven in einem ganz elementaren Punkt: während Dienstprimitive in einer Protokollnorm nur in ihrer Semantik beschrieben werden, muss zu jeder im Protokoll benutzten PDU auch der genaue syntaktische Aufbau festgeschrieben werden.

Der Grund hierfür liegt auf der Hand: während Dienstprimitive nur an internen Schnittstellen eines Kommunikationssystems auftreten, deren Realisierung implementierungsabhängig ist, werden PDUs tatsächlich zwischen den heterogenen Systemen ausgetauscht. Nur wenn einem System, welches eine PDU von einem entfernten System erhält, die genaue Struktur kennt, kann es diese auch verarbeiten.

Im Beispiel-Protokoll, dem Alternating Bit Protocol, kam u.a. eine PDU vor, in der die Nutzdaten n und die Sequenznummer b, notiert als <n,b>, zu beschreiben ist. Für diese PDU ist nun in einer Protokollbeschreibung das genaue Bitmuster festzuschreiben. Hier ist jetzt genau anzugeben,

- in welcher Reihenfolge die nieder- und höherwertigen Bits angegeben sind; Entscheidung hier: höherwertiges Bit vor niederwertigem
- in welchen Bytes die zu kodierenden Protokollelemente enthalten sind; Entscheidung hier: Sequenznummer b im ersten Byte; Länge der Nutzdaten in den zwei folgenden Bytes; Nutzdaten in den übrigen Bytes.

Vollständiger Beispielablauf



Hiermit sind alle vier Komponenten, also die zwei Instanzen S und E sowie der Ziel- und Basisdienst O und U, beschrieben.

- **Einführung und Begriffe**
  - Was ist Datenkommunikation?
  - Information, Daten, Signale
  - Netze
- **Allgemeine Grundlagen**
  - Dienste
  - Protokolle und Schichten
  - Kommunikationsarchitekturen

Kommunikationsarchitekturen / Referenzmodelle

- **Architektur von Kommunikationssystemen**
  - Geschichtete Systeme
  - Notwendig: Standardisierung
    - Genaue Abgrenzung der Funktionalität der einzelnen Schichten
    - Definierte Schnittstellen zwischen den Schichten
    - …
- **Relevante Architekturen**
  - ISO/OSI
  - TCP/IP (Internet)

Hier behandelt: ISO/OSI und TCP/IP – das erstere deshalb, da seine Terminologie sich bis heute gehalten hat und es eine gute Strukturierungshilfe bereitstellt; das zweite, da es das heute relevante Modell ist.

Andere Architekturen wie ISDN und ATM waren eine Zeit lang von Relevanz, wurden aber mittlerweile zum großen Teil von weiteren Entwicklungen abgelöst.

Standardisierungsgremien

- **Nötig für den weltweiten Einsatz: Standardisierung**
- **International Standards Organization - ISO**
  - Organisation, die auf freiwilliger Basis arbeitet (seit 1946)
  - Mitglieder: Standardisierungsorganisationen vieler Länder
  - Hat 200 Technical Comittees (TC) für sehr weites Spektrum von Standards
    - TCs haben spezifischen Aufgaben, z.B. TC97 für Computer und Informationsverarbeitung
    - TCs haben Subkomitees, die wiederum in Arbeitsgruppen unterteilt sind
  - Bahnbrechende Leistung von ISO in der Datenkommunikation: *ISO/OSI-Referenzmodell* als erstes Modell zur Strukturierung einer Kommunikation zwischen Rechnern (*OSI: Open Systems Interconnection*)
    - Bahnbrechendes Konzept, aber keine vernünftigen Produkte!

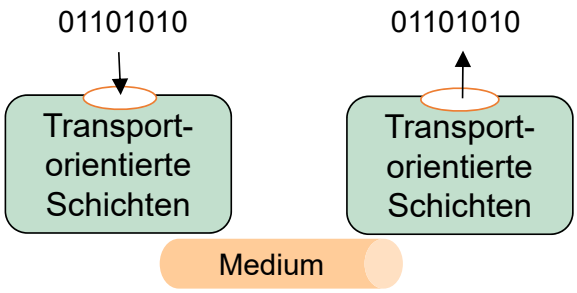
www.iso.ch

Ein sehr umfassendes Modell zur Umsetzung von Kommunikationssystemen ist das ISO/OSI-Referenzmodell (seit 1984 ein ISO-Standard). Es teilt den komplexen Kommunikationsvorgang in 7 Schichten auf, die jeweils logisch schlüssige und möglichst unabhängig behandelbare Teile eines Kommunikationsvorgangs umfassen und durch wohldefinierte Schnittstellen miteinander verbunden sind. Die hier gewählte Aufteilung hat sich als sinnvoll erwiesen; auch wenn das ISO/OSI-Modell in der Praxis nicht eingesetzt wird, haben sich andere Modelle im Wesentlichen an die hier definierte Aufteilung und Terminologie gehalten - zumindest ein Teil des OSI-RMs dient den standardisierten und auch herstellerspezifischen Protokollwelten aus dem Bereich der Rechnerkommunikation als konzeptionelle Basis.

Daher wird das ISO/OSI-Modell im weiteren Verlauf der Vorlesung zwar nur eine untergeordnete Rolle spielen, seine generelle Struktur bestimmt aber den Ablauf der Vorlesung.

Aufgabenbereiche im ISO/OSI-Referenzmodell

- **Transportorientierte Schichten**
  - Transparente Ende-zu-Ende-Übertragung von Daten
    - Inhalt (Semantik) der Daten transparent
    - Kein Bezug zu Kooperationsbeziehung der Dienstnutzer
    - Elementare Datenübertragung, Funktionsumfang abhängig vom bereitzustellenden Dienst



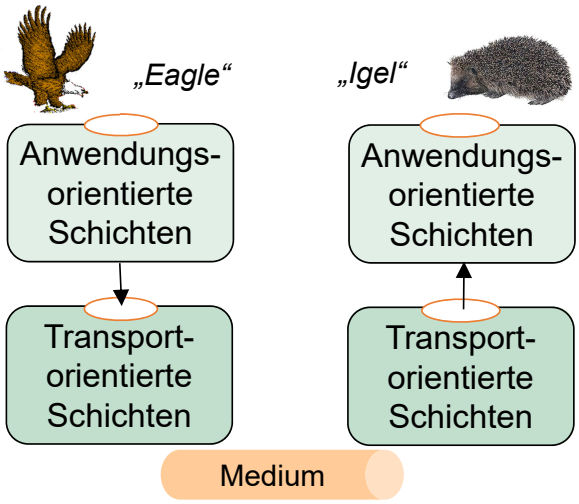
Generell lassen sich die im ISO/OSI-Modell definierte Schichten in zwei Gruppen aufteilen: transportorientierte und anwendungsorientierte Schichten.

Die transportorientierten Schichten dienen, wie der Name schon sagt, ausschließlich zum Transport der Daten. Weitere Aspekte, die z.B.

die Kooperation der Anwendungen oder die Inhalte der Daten selbst betreffen, werden nicht berücksichtigt.

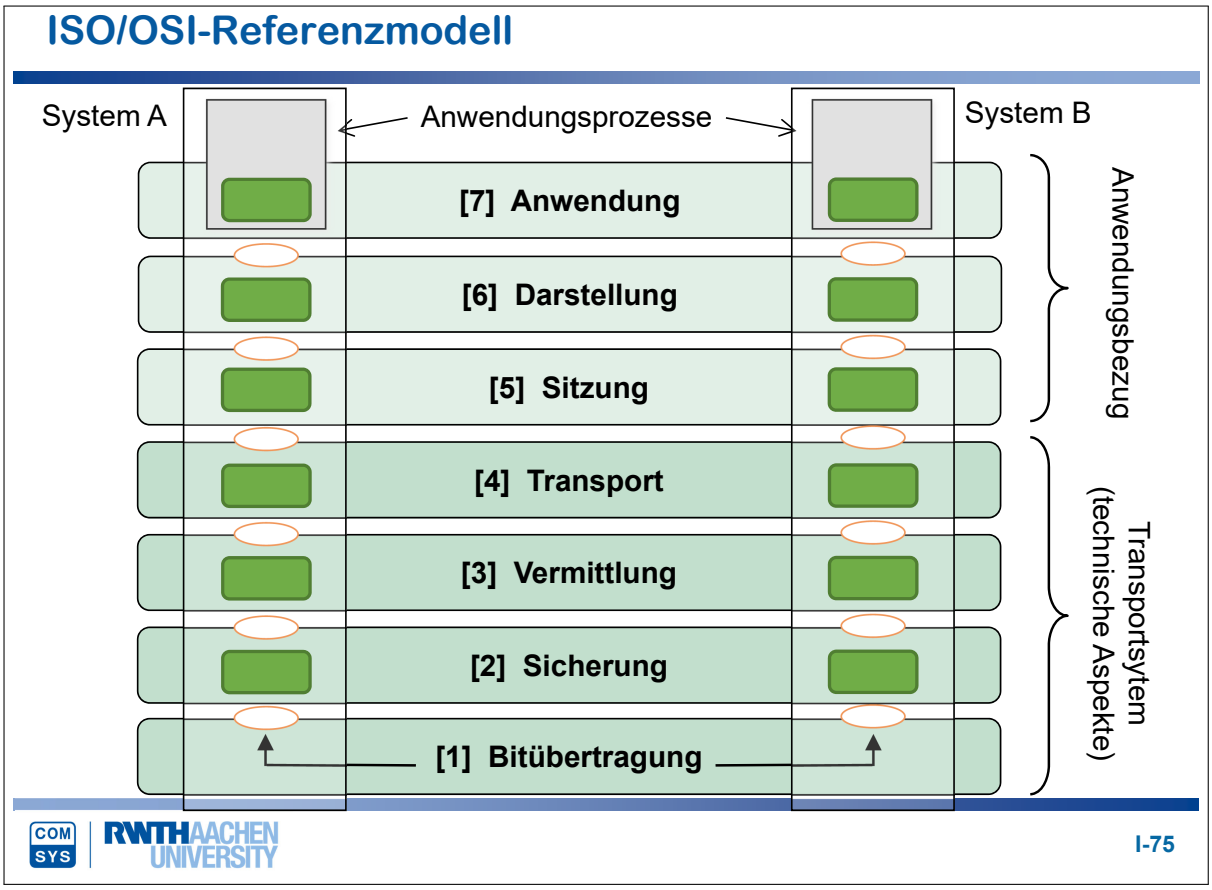
Aufgabenbereiche im ISO/OSI-Referenzmodell

- **Anwendungsorientierte Schichten**
  - Anwendungsbezogene Aspekte der Datenübertragung
    - Semantik der Daten ist wichtig
    - Kooperation der Teilnehmer unter formalen Gesichtspunkten (Strukturierung, Präsentation), z.B.
      - Steuerung des Kommunikationsablaufs
      - Kompensation von Fehlverhalten durch verteilte Transaktionen



Hierfür sind die anwendungsorientierten Schichten zuständig. Diese berücksichtigen Kooperationsaspekte, soweit sie in formaler Weise behandelbar sind. Teilschichten des Anwendungssystems kümmern sich außerdem um die Steuerung des Kommunikationsablaufs und um die Informationsdarstellung.

Der Zusammenhang mit dem Transportsystem findet dort im Anwendungssystem seinen Niederschlag, wo ein etwaiges Fehlverhalten, welches vom Transportsystem auf der reinen Übertragungsebene nicht erkannt werden kann, kompensiert werden muss.



Beim OSI-RM steht der *Modellierungsaspekt* im Mittelpunkt und weniger die durch OSI genormten Protokolle.

Diese Aussage gilt auch in Bezug auf die TCP/IP-Protokolle der Internet-Welt, wie auf einer der nachfolgenden Folien gezeigt wird.

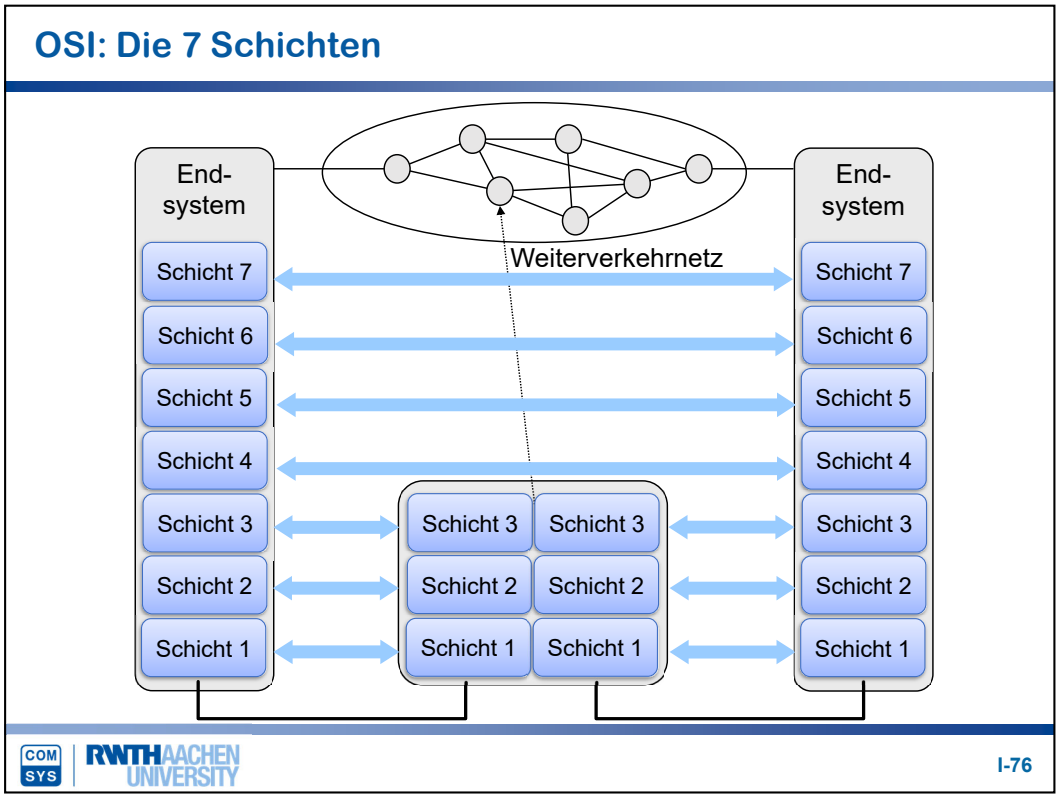
Im OSI-RM bilden die unteren vier Schichten das sogenannte Transportsystem (transportorientierte Schichten), die drei oberen Schichten realisieren das sogenannte Anwendungssystem (anwendungsorientierte Schichten).

Die wesentlichen Aufgaben der Schichten sind:

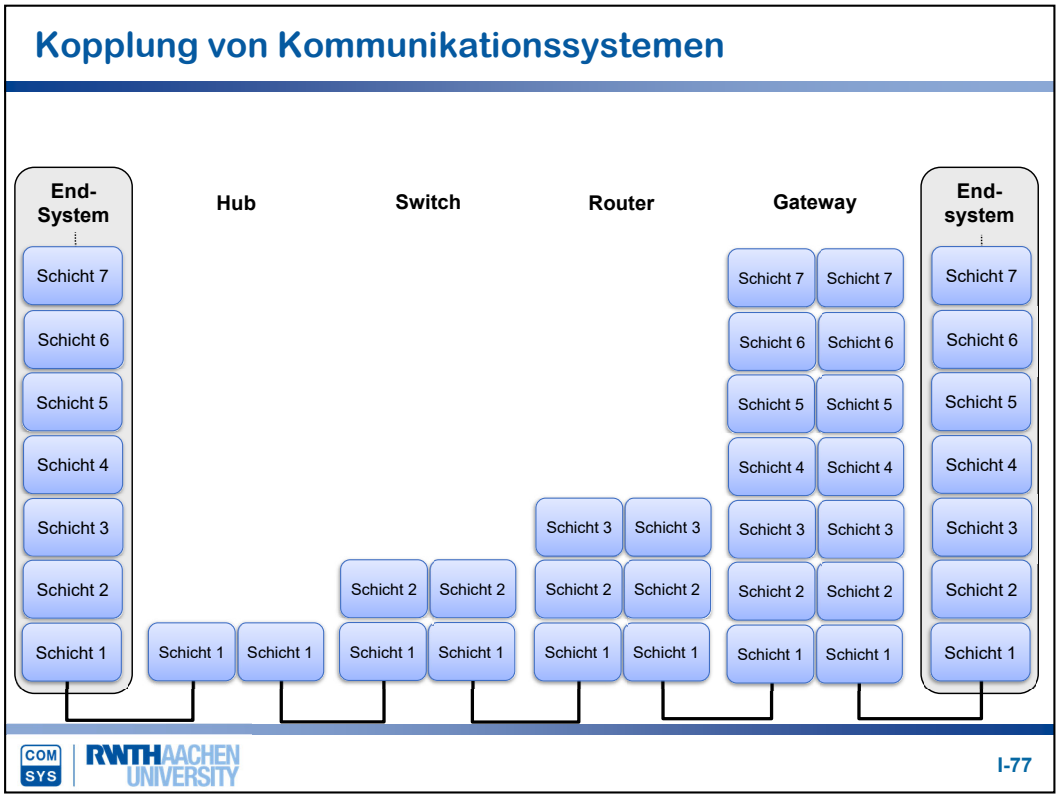
- Bitübertragungsschicht: Darstellung von Bits auf dem physikalischen Medium, Normierung von Kabeltypen und Steckern
- Sicherungsschicht: Erkennung und Behebung von Übertragungsfehlern zwischen zwei direkt miteinander verbundenen Geräten, Flusskontrolle zur Vermeidung der Überlastung des Empfängers, Regelung des Medienzugriffs
- Vermittlungsschicht: Verbindung von Teilstrecken/Netzen zu einem globalen Netzwerk, weltweit eindeutige Adressierung, Bestimmung von günstigen Pfaden durchs Netz, Weiterleitung der PDUs, evtl. Auch Staukontrolle zur Vermeidung der Netzüberlastung und Regelung der Übertragungsqualität.
- Transportschicht: Adressierung von Anwendungsprozessen, Ende-zu-Ende-Verbindung, Behebung von Fehlern auf Schicht 3 (Erkennung von Datenverlust, Staukontrolle, …)
- Sitzungsschicht: Strukturierung der Übertragung, Ablaufsteuerung/-koordinierung, Verwaltung von Dialogen

- Darstellungsschicht: Kommunikation wird trotz unterschiedlicher lokaler Datenformate der Teilnehmer ermöglicht, Konvertierung der Daten in Transportformat
- Anwendungsschicht: Stellt Dienste für bestimmte Anwendungsarten zur Verfügung, z.B. Dateitransfer, entfernter Prozeduraufruf, E-Mail. Z.B. Definition von Nachrichtentypen

Das OSI-RM definiert verschiedene Protokolle für jede Schicht, die unterschiedliche Teile des Funktionsumfangs einer Schicht implementieren. Es hängt von einer konkreten Anwendung ab, welche Implementierungen und damit welche Funktionen genutzt werden.



Der Normalfall ist, dass Zwischenknoten im Netz, die Daten nur weiterleiten sollen (Router), nur die untersten drei Schichten implementieren müssen. Eine Kopplung ist allerdings auch mit mehr oder weniger Schichten möglich.

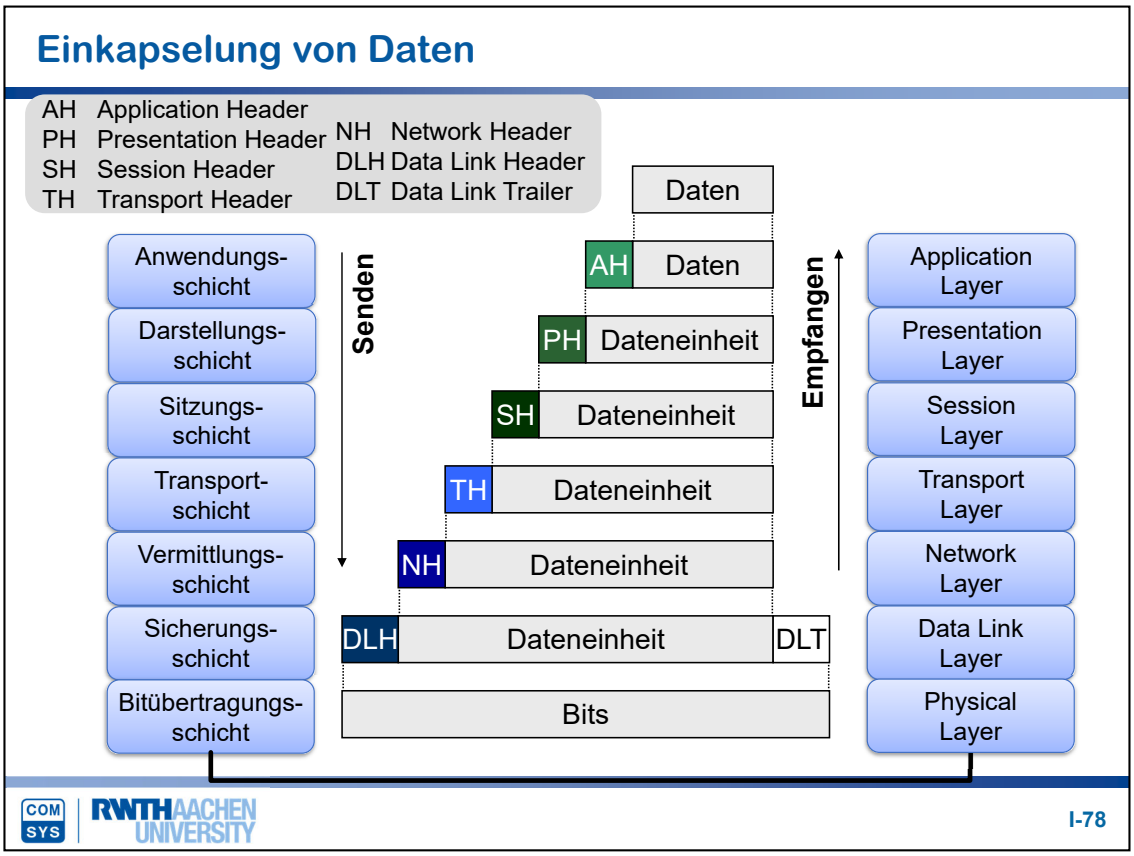


Kopplung auf Schicht 1: Repeater, nur Signalverstärkung; Verlängerung der Ausdehnung einer physikalischen Leistung

Kopplung auf Schicht 2: Switch, direkte Kopplung von Rechnern innerhalb eines lokalen Netzes; auch Bridge, Kopplung von lokalen Netzen, die unterschiedliche Schicht-2-Protokolle verwenden

Kopplung auf Schicht 3: Router, Kopplung unabhängiger Netze, auch über weite Strecken hinweg, Wegwahl innerhalb des Netzverbundes

Kopplung auf Schicht 7: Gateway, Kopplung von Systemen mit unterschiedlichen Anwendungsprotokollen. (Bitte Vorsicht: der Begriff „Gateway“ wird teilweise auch in anderer Bedeutung verwendet, z.B. in gewissem Kontext auch für Router, siehe Kapitel 4. Daher werden die hier dargestellten Gatewys auch schon mal „Application Gateways“ genannt.)



Wie wir gesehen haben, benutzen Instanzen die Dienste darunter liegender Instanzen. Entsprechend dem beschriebenen Zusammenspiel der Schichten werden die eigentlich zu übertragenden Daten in jeder Schicht um weitere Kontrolldaten (PCI) angereichert, die sich in einem Paketkopf (Header) oder Paketanhang (Trailer) befinden. Diese zusätzlichen Kontrolldaten müssen auf der empfangenden Seite in umgekehrter Reihenfolge wieder entfernt und ausgewertet werden, ehe die Daten an die nächsthöhere Schicht weitergereicht werden.

ISO/OSI-Modell und Realität

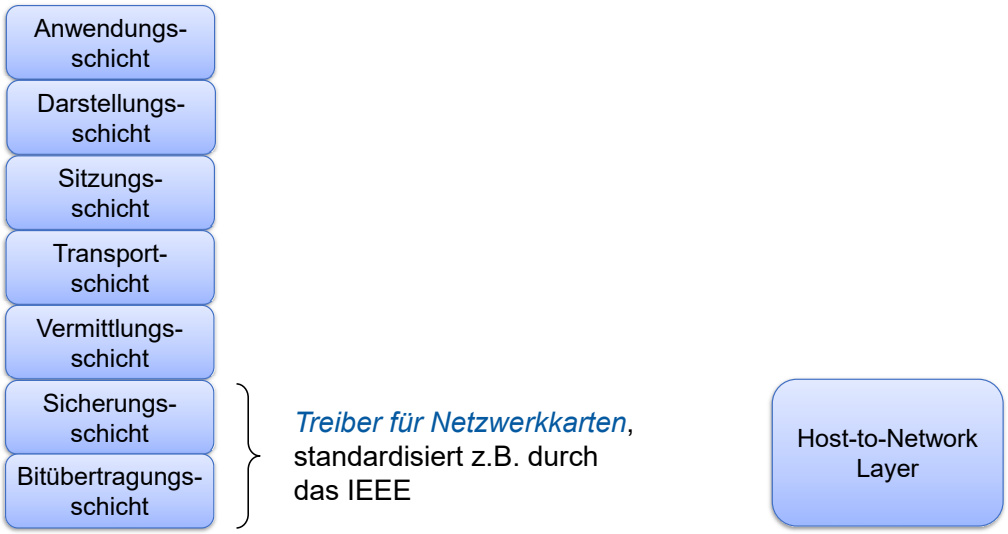
- **Situation**
  - ISO/OSI-Basisreferenzmodell ist der „Klassiker“
    - Gut geeignet zur logischen Strukturierung
    - Oft weit weg von der Strukturierung einer Anwendung
    - Manchmal zu detailliert und komplex … sagen zumindest Kritiker
  - Tipp
    - Versuchen Sie in diesem Modell zu „denken“ – ordnen Sie immer wieder die Problemstellungen in Kommunikationssystemen sorgfältig in dieses Modell ein – das erleichtert die Beherrschung des umfangreichen Stoffs ungemein ☺
- **Modellierung vs. Implementierung**
  - Schichten werden in der Realität oft nicht sauber getrennt implementiert
  - Daten werden zwischen Schichten möglichst nicht kopiert

Implementierungen des OSI-RM haben sich nicht durchgesetzt – aus verschiedensten Gründen: Komplexität, Implementierungsfehler, Politik, Dauer der Standardisierung. Die Terminologie der „7 Schichten“ hat sich allerdings bis heute gehalten, da die vorgenommene Strukturierung an sich durchaus sinnvoll ist. Auch bei heutigen Protokollimplementierungen hält man sich oft an diese Einteilungen.

In der Praxis findet dann aber nicht unbedingt eine saubere Implementierung statt; aus Effizienzgründen werden verschiedene Aspekte der Schichten gemischt oder aneinander angepasst.

Schichtenmodelle in der Praxis

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



Ein großer Nachteil des ISO/OSI-RM ist der Overhead, der durch die Vielzahl an Schichten entsteht – zum einen wird eine große Menge an Kontrollinformationen zu den eigentlichen Nutzdaten hinzugefügt, zum anderen muss auf jeder Schicht eine Protokollinstanz eine PDU erstellen bzw. diese bei Empfang kontrollieren.

In der Praxis werden daher üblicherweise Schichten zusammengefasst, um den Umfang an Kontrollinformationen zu reduzieren und die Funktionalitäten effizienter implementieren zu können.

Die Sicherungs- und die Bitübertragungsschicht werden gemeinsam für bestimmte Netzwerke implementiert, als Treiber für Netzwerkkarten. Standardisiert werden Protokolle dieser Schichten z.B. durch das IEEE (Institute of Electrical and Electronics Engineers)

Standardisierung von „lokalen“ Netzen

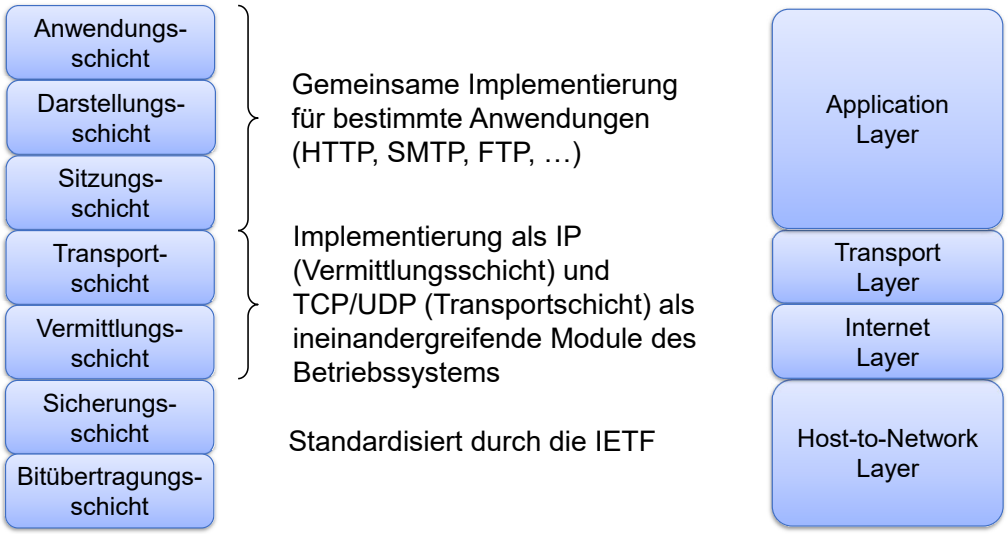
- **Institute of Electrical and Electronic Engineers (IEEE)**

IEEE Networking the World™
www.ieee.org

- 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 (TAG)
- 802.8 Fiber Optic TAG
- 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 TAG
- 802.19 Coexistence TAG
- 802.20 Mobile Broadband Wireless Access (MBWA)
- 802.21 Media Independent Handover
- 802.22 Wireless Regional Area Networks (WRAN)
- 802.23 Emergency Services
- 802.24 Vertical Applications TAG

Schichtenmodell (TCP-IP-Modell) des Internets

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

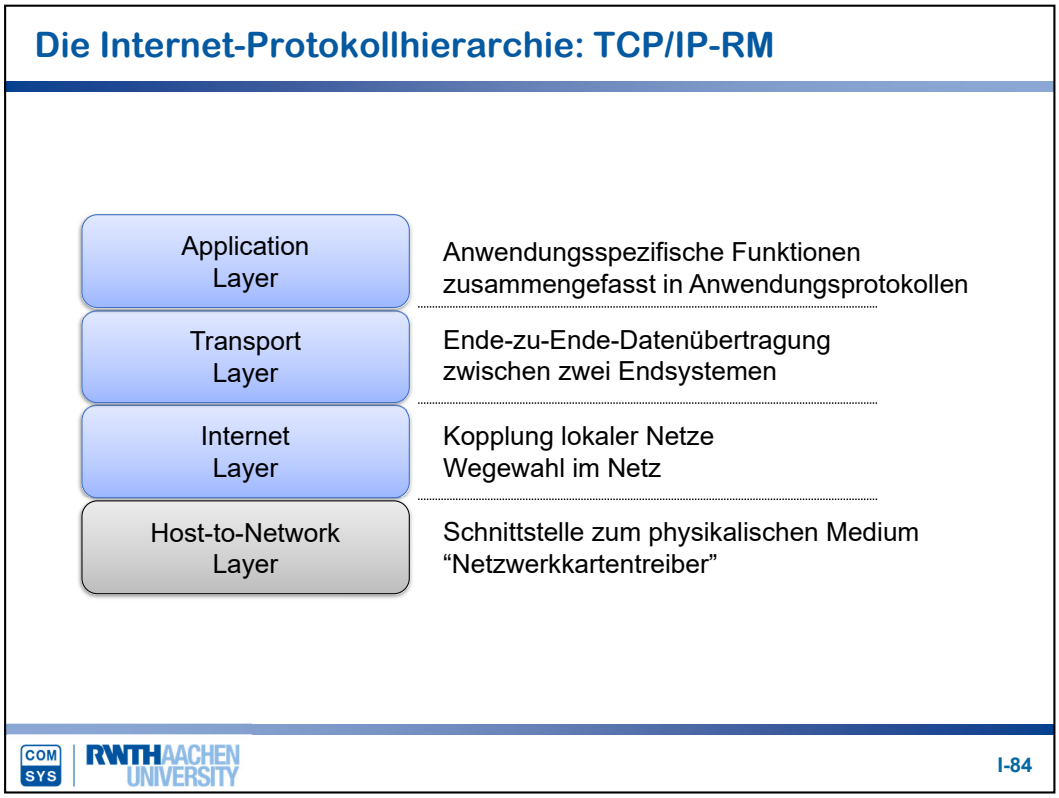


Während in der Praxis eine Vielzahl an Protokollen der Schicht 1/2 existiert, standardisiert die Internet Engineering Task Force die Protokolle ab Schicht 3, um eine Kopplung aller möglichen Netze mit beliebigen Schicht-1/2-Protokollen zu ermöglichen und um die Nutzung wichtiger Anwendungen netzweit einheitlich zu gestalten.

Standardisierungsgremien

- **Internet Engineering Task Force (IETF)**
  - Forum für die technische Koordination der Arbeiten zum ARPANET, dem Vorläufer des Internets (seit 1986)
  - Entwicklung zur großen, offenen und internationalen Gemeinschaft von Administratoren, Herstellern und Forschern
  - Beschäftigt sich mit der Evolution der Internet-Architektur und der reibungslosen Operation des Internets
  - Verschiedene Arbeitsgruppen zu einzelnen Aspekten des Internets
  - COMSYS ist auch aktiv in der IETF: RFCs 3662, 3754, 6069, 6253, ...

www.ietf.org



Durchgesetzt in der Praxis hat sich das TCP/IP-Referenzmodell (oder auch: Internet-Referenzmodell), das durch die IETF standardisiert wird. Es ist schlanker als das OSI-RM und damit effizienter einzusetzen… bietet allerdings auch weniger Wahlmöglichkeiten, welche Funktionen auf einer Schicht implementiert werden sollten.

Der Host-to-Network Layer wird von der IETF nicht betrachtet. Es wird lediglich verlangt, dass eine Schnittstelle zur Übertragung von IP-Paketen bereitgestellt wird. Die Spezifizierung der untersten Schicht wird anderen Gremien überlassen, z.B. der IEEE.

Die im Zusammenhang mit dem Internet entwickelten Protokolle werden als TCP/IP-Protokolle bezeichnet und umfassen neben dem Transportprotokoll TCP (Transmission Control Protocol) und dem Vermittlungsprotokoll IP (Internet Protocol) unter anderem auch Protokolle der Anwendungsschicht (z.B. File Transfer Protocol FTP oder HyperText Transfer Protocol HTTP).

Das TCP/IP-Modell ist als Modell weniger relevant. An dieser Stelle hat die ISO die wesentlichen Grundlagen erarbeitet, die z.T. auch Eingang in das TCP/IP-RM gefunden haben. Interessant im Zusammenhang mit TCP/IP sind vielmehr die konkreten Protokolle, die das Modell auffüllen. Hier besteht ein genau umgekehrtes Verhältnis zu der ISO, deren Protokolldefinitionen zu keinem Zeitpunkt die hohe Akzeptanz der TCP/IP-Protokolle erreicht hat. Ein kurzes Fazit lautet also:

Während die ISO die modelltechnischen Grundlagen für Kommunikationssysteme lieferte, resultierten aus den Internet-Aktivitäten allgemein akzeptierte Protokollstandards in Form der TCP/IP-Protokollfamilie.

Fazit

- **Datenkommunikation:**
  - Sammlung aufeinander aufbauender Dienstinstanzen und Protokolle, die gemeinsam die gesamte Funktionalität eines 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 als *Netzwerkkarten (und deren Treiber)* implementiert. Innerhalb eines lokalen Netzes können Geräte mittels dieser Funktionalität kommunizieren.
    - TCP/IP-Referenzmodell baut auf den Netzwerkkarten(treibern) auf
      - IP und TCP/UDP sind als *Teil des Betriebssystems* implementiert
      - Über Anwendungen werden *Dienste* auf dem System realisiert (Web, Email, Zeitsynchronisation). Die Anwendungen koordinieren sich untereinander über Anwendungsprotokolle (HTTP, SMTP, NTP, ...), die wiederum die Dienste von TCP, UDP nutzen. Diese nutzen den Dienst von IP.

Lessons learned

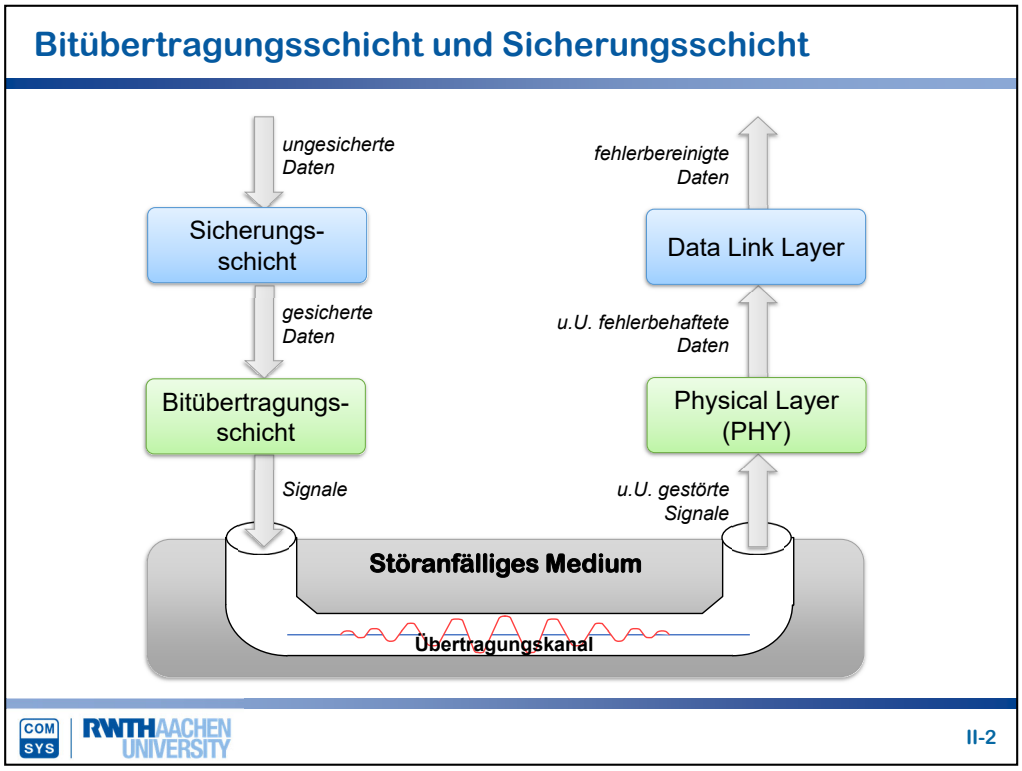
- **Wichtige Begriffe/Konzepte des Kapitels**
  - Kommunikationsdienst vs. –protokoll
    - Dienst beschreibt das „was“, Protokoll das „wie“
  - Bestätigte und unbestätigte Dienste
    - Bitte deutlich unterscheiden von Bestätigungen im Rahmen von Protokollen!
  - Verbindungslose und –orientierte Dienste
    - Verbindungsorientierte Dienste bauen einen Kontext auf, verbindungslose Dienste versenden alle PDUs unabhängig voneinander
  - Schichten
    - Implementierung von Kommunikationssystemen als geschichtete Sammlung von Protokollen, die sich auf bestimmte Aufgaben spezialisieren
  - Referenzmodelle
    - ISO/OSI-RM als theoretisches Modell, TCP/IP-RM in der Praxis

Datenkommunikation
Kapitel 2: Bitübertragungsschicht

Klaus Wehrle

Communication and Distributed Systems
Chair of Computer Science 4
RWTH Aachen University

http://www.comsys.rwth-aachen.de



Die beiden untersten Schichten sorgen gemeinsam für eine fehlerfreie Datenübertragung zwischen benachbarten Rechnern. „Benachbart“ heißt, dass die kommunizierenden Rechner durch ein physikalisches Medium direkt miteinander verbunden sind.

Die Bitübertragungsschicht bietet als Medium nachrichtentechnische Kanäle, die einen ungesicherten, längenbeschränkten, ungepufferten Nachrichtenaustausch zwischen einer beschränkten Anzahl von angeschlossenen Einrichtungen unterstützen:

- ungesicherte Verbindung zwischen Systemen (d.h. es können Fehler bei der Übertragung auftreten)
- Übertragung unstrukturierter Bitfolgen über das physikalische Medium
- Normung der physikalischen Schnittstellen zwischen Rechner und Medium
- Umsetzung von Daten in Signale

Die Sicherungsschicht erweitert einen nachrichtentechnischen Kanal zum eigenständigen, abstrakten Medium „gesicherter Kanal“. „Gesichert“ heißt hierbei: fehlerfrei:

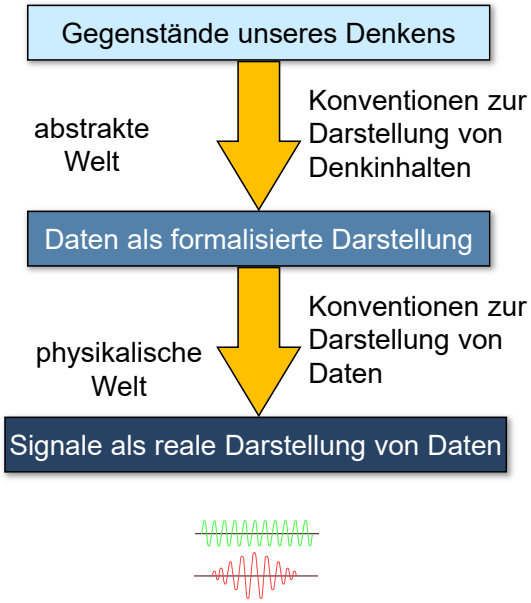
- Zerlegung des Bitstromes in Rahmen (Frames) als Basiseinheit der Übertragung
- Fehlererkennung und –behandlung (Quittungen und Übertragungswiederholungen, siehe Alternating Bit Protocol)
- Flusskontrolle (Vermeidung der Überlastung/Pufferüberlauf des Empfängers)
- Regelung des Zugriffs auf ein Medium, welches gemeinsam von mehreren Stationen verwendet wird.

Themenübersicht

- **Datenkommunikation**
  - Dienste, Protokolle und Referenzmodelle
  - Technische und nachrichtentechnische Grundlagen: Medien, Signale, Bandbreite, Leitungscodes und Modulation, Multiplexing
  - Lokale Netze: Strukturierung des Datenstroms, Fehlererkennung/-behebung, Flusssteuerung, Medienzugriff, Ethernet
  - Internet und Internet-Protokolle:
    - Vermittlungsschicht: IP, Routing
    - Transportschicht: TCP
  - Grundlagen der Sicherheit in/von Kommunikationsnetzen
    - Grundlagen: Verschlüsselung, Authentifizierung, Integrität
    - Sichere Internet-Protokolle
  - Anwendungsschicht

Wiederholung: Daten und Signale

- **Daten**
  - Darstellung von Sachverhalten, Konzepten, Vorstellungen und Anweisungen in formalisierter Weise
- **Signal**
  - Physikalische Darstellung von Daten durch charakteristische räumliche und/oder zeitliche Veränderungen der Werte physikalischer Größen
  - Reale physikalische Repräsentation der Daten

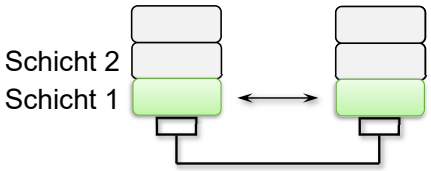


Kurze Wiederholung: Daten und Signale.

Gegenstände unseres Denkens (beispielsweise Fakten, Konzepte, Vorstellungen und Modelle) werden in formalisierter Weise dargestellt: als *Daten*. Unter dem Begriff „Daten“ versteht man also die Darstellung von Sachverhalten (Fakten), Konzepten, Vorstellungen und Anweisungen in formalisierter Weise, die für die Kommunikation, Interpretation und die Verarbeitung durch Menschen und/oder technische Mittel geeignet ist. Beispiele für Daten sind die gesprochene Sprache, die geschriebene Sprache oder Zeichen- und Gebärdensprache.

Durch entsprechende Konventionen können Daten in physikalische *Signale* gewandelt werden, beispielsweise in Form von sinusförmigen Schwingungen. Dies ist eine grundlegende Notwendigkeit – Signale sind eine Repräsentation der Daten, die über ein physikalisches Medium übertragen werden können. Die Art der verwendbaren Signale hängt direkt vom Medium selbst ab (Strom, Licht, …).

- **Aufgaben der Bitübertragungsschicht (Schicht 1)**
  - Englisch: Physical Layer, PHY
  - Unterste Schicht, sitzt direkt auf dem physikalischen Medium
    - Übertragung unstrukturierter Bitfolgen über physikalisches Medium
    - Umfasst u.a. physikalischen Anschluss, Umsetzung Daten ⇔ Signale
    - Ungesicherte Verbindung zwischen Systemen
  - Normung vor allem der physikalischen Schnittstelle Rechner/Medien
    - Beispiele (nur Schnittstelle, nicht die entsprechenden Protokolle):
      - Ethernet (Yellow Cable, BNC, RJ-45 etc.)
      - USB
      - Wi-Fi
      - LTE/5G



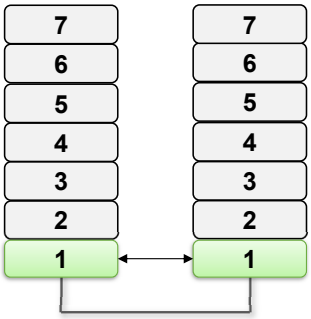
Die unterste Schicht im ISO/OSI-Modell ist die Bitübertragungsschicht (Physical Layer). Da die Bitübertragungsschicht direkt auf dem physikalischen Medium sitzt (z.B. ein Kabel, aber auch die Luftschnittstelle beim Mobilfunk), wird das physikalische Medium auch als „Schicht 0“ bezeichnet.

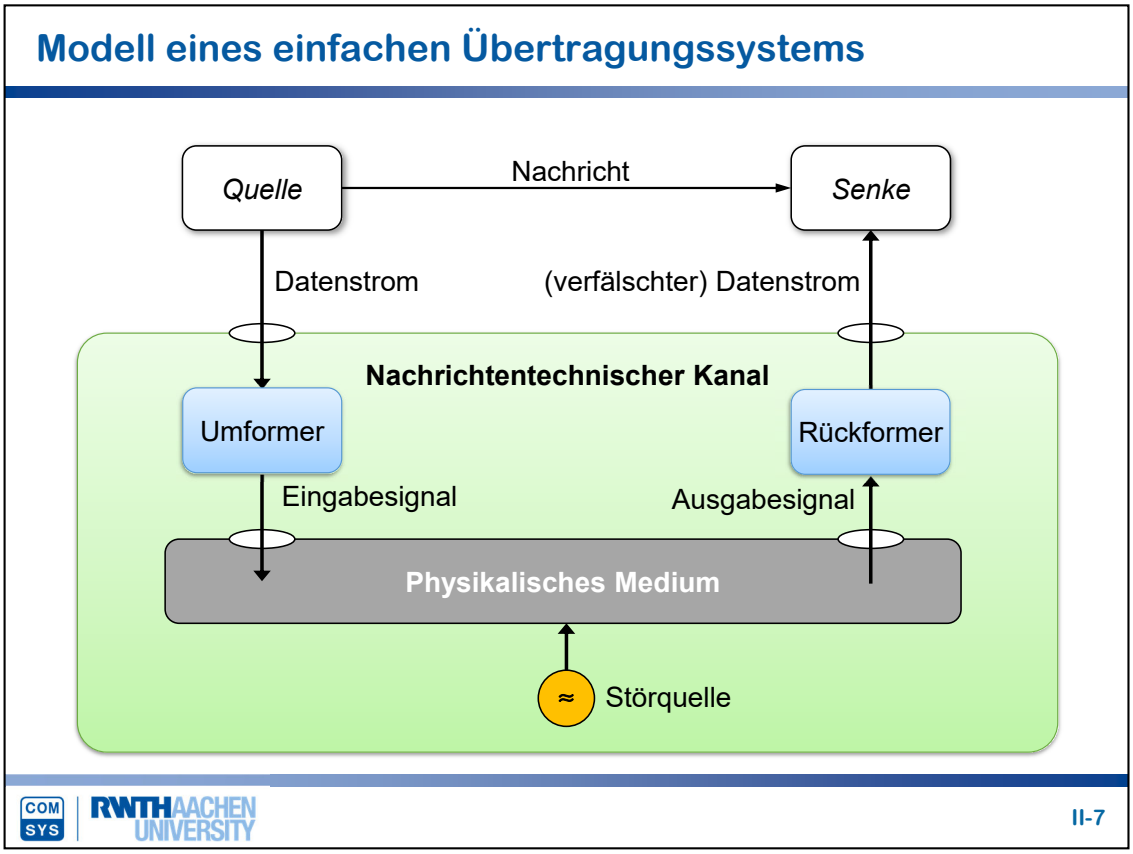
Die Aufgaben der Bitübertragungsschicht sind demzufolge hardwarenah: Festlegung des physikalischen Mediums und der Repräsentation digitaler Informationen auf dem Medium durch Signale, der Pinbelegung in Steckern, …

Zu berücksichtigen ist, dass die Übertragung von Daten ungesichert ist: durch Störeinflüsse können Daten während der Übertragung verfälscht werden, aber es wird nicht versucht, solche Situationen zu erkennen oder mit Fehlern umzugehen – dies bleibt der nächsthöheren Schicht überlassen.

Kapitel 2: Bitübertragungsschicht

- **Grundlagen**
  - Übertragungsmedien
  - Signale und Bandbreite
- **Übertragung von Signalen**
  - Umformung, Basisband, Modulation
  - Übertragungsparameter, Störeinflüsse
  - Leitungscodes und Modulationsverfahren
  - PCM
- **Kanalnutzung**
  - Multiplexing

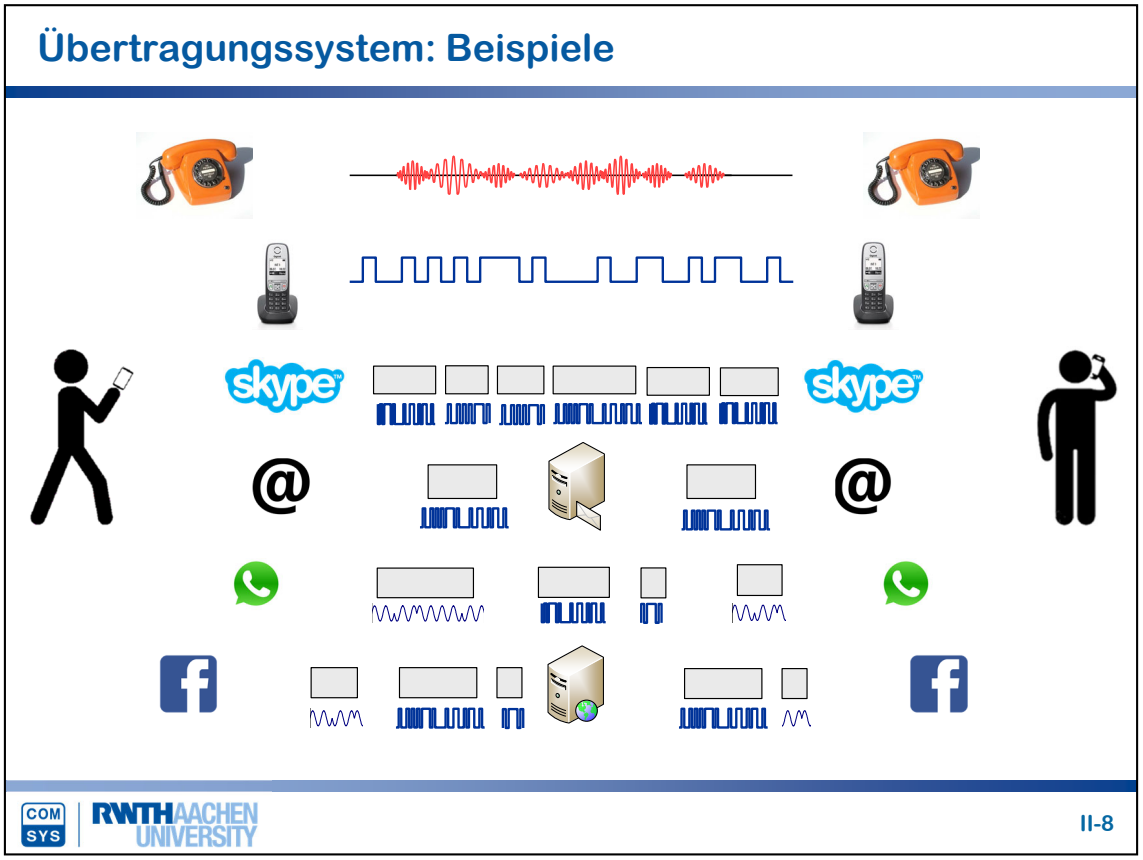




Im Folgenden werden Aspekte der Übertragungsmedien und der Schicht 1 betrachtet, d.h. nachrichtentechnische Grundlagen der Datenkommunikation. Die Behandlung dieser Problematik geht aber nur soweit, wie es für einen Informatiker relevant ist.

Das obige Modell eines einfachen Übertragungssystems unterscheidet folgende drei Ebenen:

- *Quelle*, die eine Nachricht (ein Folge von Eingabesignalen – meist digital vorliegende Daten, letztlich eine Folge von Bits) an die *Senke* überträgt
- *Nachrichtentechnischer Kanal*: zur Übertragung des Eingabesignals (der Daten / des Bitstrom)s über ein konkretes physikalisches Medium müssen die Daten in andere Signale transformiert werden, die das Medium transportieren kann. Die Transformation muss so erfolgen, dass die entstehenden Signale gut übertragen und auf der Gegenseite eindeutig rücktransformiert werden können. Dafür sind die beiden Bestandteile *Umformer* und *Rückformer* im nachrichtentechnischen Kanal zuständig. Durch den Umformer entsteht ein Eingabesignal: eine physikalische Größe, wie etwa der Amplitudenwert einer elektrischen Spannung, dessen Wert sich im Verlauf der Zeit ändert.
- Das *Medium*, durch welches die räumliche Distanz zwischen Quelle und Senke letztendlich überbrückt wird. Das physikalische Medium gehört mit zum nachrichtentechnischen Kanal. Die Übertragung des Eingabesignals über das Medium ergibt auf der Seite der Senke ein Ausgabesignal. Während der Übertragung können Störungen auftreten, die zu einer Verfälschung des Signalverlaufs führen. Im Modell des nachrichtentechnischen Kanals werden alle möglichen auftretenden Störungen in einer Störquelle zusammengefasst. Das Ausgabesignal ist dann eine Funktion des Eingabesignals und des Signals der Störquelle. Die Senke erhält nach Rücktransformation des Ausgabesignals einen eventuell verfälschten Bitstrom.



Es gibt eine Vielzahl von möglichen Umsetzungen eines Übertragungssystems. Das älteste System ist das analoge Telefon. Hier wird das Eingabesignal (Sprache, ein analoges Signal) durch ein Mikrofon erfasst und so umgeformt (moduliert), dass sie auf dem Telefonkanal als elektromagnetische Welle (analog) übertragen werden können. Das Medium ist ein Kupferkabel.

Beim digitalen Telefon wird das Eingabesignal (Sprache) als digitales Signal auf das Medium (Kupferkabel) gesetzt. Das Medium ist also das gleiche wie beim analogen Telefon, nur die Umformung ist unterschiedlich. Auch das digitale Signal ist ein elektromagnetisches Signal.

Bei Voice-over-IP oder Videotelefonie wie mit Skype wird das Eingabesignal auch digital übertragen, allerdings nicht als Strom von Audiodaten (Bitfolge), sondern als Strom von Paketen, die jeweils einen Teil der Daten enthalten.

Neben diesen synchronen Formen der Kommunikation gibt es auch asynchrone Systeme, bei denen Quelle und Senke nicht gleichzeitig aktiv sein müssen. Dazu zählt E-Mail – die Quelle überträgt die Nachricht als Paket hin zu einem Mailserver, von dem die Senke die Nachricht jederzeit abrufen kann.

Bei WhatsApp wird eine Chatnachricht ebenfalls digital als Paket übertragen, wird der Senke allerdings direkt zugestellt. Hier erfolgt die Übertragung über ein

analoges Medium – einen Funkkanal.

Eine ganz andere Form der Übertragung sind Systeme wie Facebook – die Daten werden digital in Form von Paketen auf die Facebook-Server gepostet, wo andere Personen sie jederzeit abrufen können.

Während es diese Vielzahl von Kommunikationsformen gibt (analog/digital, synchron/asynchron, Bitstrom/Paketstrom, …), interessiert uns auf der technischen Ebene nur die physikalische Übertragung der Daten –Pakete existieren auf dieser Ebene noch nicht, es werden nur unstrukturierte Bitfolgen betrachtet, die übertragen werden müssen. Dies kann digital oder analog erfolgen. Ebenso kann die Umformung von/zu digital/analog erfolgen.

Ganz generell haben analoge und digitale Signale allerdings eine Gemeinsamkeit: beide sind elektromagnetische Signale.

Umformung digital/analog

- **Mögliche Umformungen:**
  - *Analog → Analog:*
    - Ursprüngliches Telefon (engl.: POTS = Plain Old Telephone System), (analoger) Rundfunk

  - *Analog → Digital:*
    - Digitale Telefonie, Voice-over-IP (VoIP)

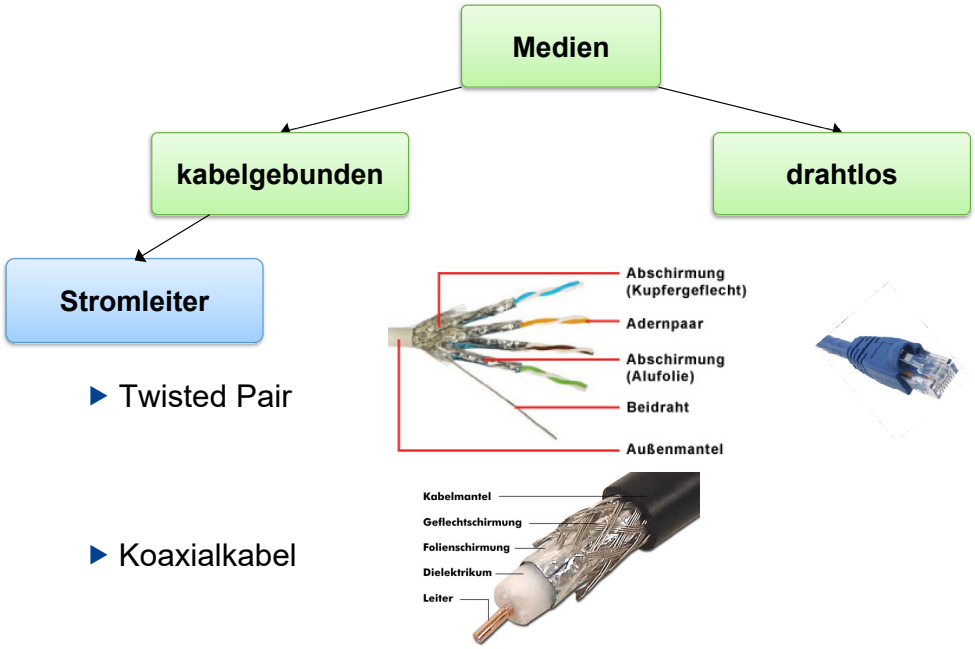

  - *Digital → Analog:*
    - Digitaldatenübertragung über analoges Netz (MODEM-Technik, DSL) oder mittels Funk-/Satellitentechnik

  - *Digital → Digital:*
    - Leitungscodierung im Basisbandverfahren


Ein wesentlicher Aspekt des Übertragungssystems ist, welche Art von Signalen verwendet werden, um Daten über ein bestimmten Medium zu transportieren, und wie die Umformung der zu übertragenen Daten auf das medienbezogene Signal stattfindet. Im Modell des Übertragungssystems ist eine Umformer- und eine Rückformer-Einheit vorgesehen, welche das Übertragungsverfahren durchführt.

Je nach Art der zu übertragenen Daten und des medienbezogenen Signals können vier Arten von Umformung gebildet werden.

Physikalische Medien



COM SYS | RWTH AACHEN UNIVERSITY

https://www.elektronik-kompendium.de/sites/net/0603191.htm
https://www.itwissen.info/RG-6-Kabel-RG-6-cable.html

II-10

Generell kann man Übertragungsmedien in kabelgebunden und drahtlos unterteilen. Betrachten wir zunächst die kabelgebundenen Medien – diese kann man noch weiter unterteilen. Eine Unterkategorie ist die der Stromleiter – auf diesen Medien können Daten analog als elektromagnetische Welle oder digital als Folge von Strompulsen übertragen werden.

Wichtige Stromleiter sind Twisted Pair (verdrillte Kupferdoppelader) und Koaxialkabel.

Twisted Pair (TP): Zwei Kupferdrähte werden verdrillt, um Störeinflüsse abzuschwächen. Solch ein Kabelpaar wurde bereits für das analoge Telefonnetz verwendet: die Sprache wurde aufmoduliert und als elektromagetisches Signal über solche ein Adernpaar übertragen. Als TP-Kabel bezeichnet man heute vier solche Adernpaare die in einer gemeinsamen Ummantelung enthalten sind. Mittels sogenannter RJ45-Stecker wird diese Kabelform heutzutage z.B. beim Ethernet eingesetzt.

Die Basisvaiante (vier Pare im Außenmantel) wird als „Unshielded Twisted Pair“ (UTP) bezeichnet. Neben UTP gibt es auch noch teurere, abgeschirmte Kabel (STP, Shielded Twisted Pair), bei denen die einzelnen Adernpaare durch Alufolie gegeneinander abgeschirmt werden und ein Kupfergeflecht die vier Adernpaare noch einmal nach außen hin abschirmt. Je mehr Abschirmung, desto weniger Störfelder erzeugen die Kabel in der Umgebung und umso weniger Störungen auf dem Kabel rufen elektromagnetische Störsignale in der Umgebung hervor.

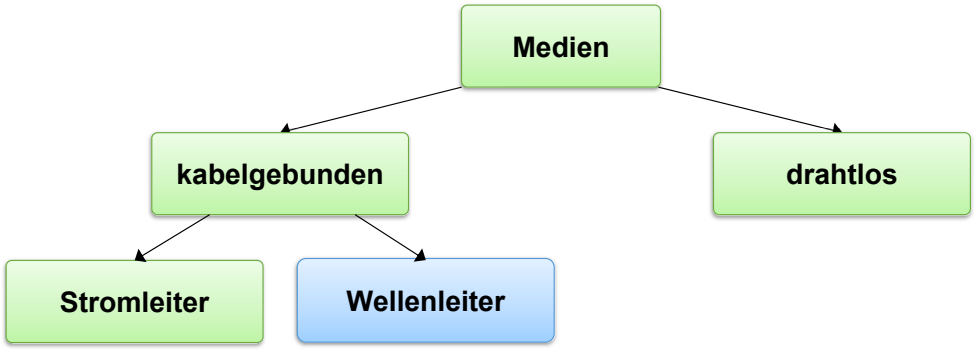
.

TP-Kabel werden in unterschiedliche Kategorien aufgeteilt, UTP/STP 1 bis 7 (häufig auch als CAT 1 bis CAT 7 bezeichnet), die mit zunehmender Nummer eine größere Bandbreite bieten, die sich zur Übertragung nutzen lässt; eine größere Bandbreite bedeutet auch eine höhere erzielbare Datenrate. Zudem lassen sich bei höheren Kategorien größere Entfernungen überbrücken, ohne dass die Signalqualität stark vermindert wird. Bandbreite und Signalqualität werden im weiteren Verlauf des Kapitels noch genauer behandelt.

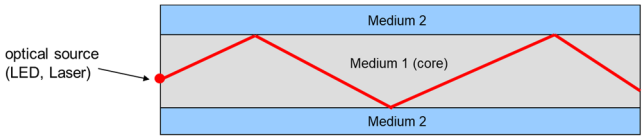
CAT 3 bis 7 werden für Netzwerke verwendet – CAT 3 dürfte noch in älteren Netzen zu finden sein, CAT-5-Kabel sind derzeit möglicherweise (noch) am weitesten verbreitet, CAT 6 oder 7 werden mittlerweile bevorzugt verlegt, da sie die größte Bandbreite bieten und damit die heute interessanten Datenraten erreichen können. Eine Kategorie 8 ist bereits auch geplant. Weitere Details zu den Kategorien sind hier uninteressant; bei Interesse sei z.B. auf Wikipedia verwiesen (http://de.wikipedia.org/wiki/Twisted-Pair-Kabel).

Koaxialkabel: Besser abgeschirmte Kupferkabel, die allerdings ziemlich unflexibel sind. Dadurch können eine höhere Bandbreite und eine bessere Signalqualität erzielt werden. In den Anfangszeiten in Ethernet benutzt, aufgrund der einfacheren Handhabung mittlerweile durch TP-Kabel abgelöst.

Physikalische Medien



- Glasfaser



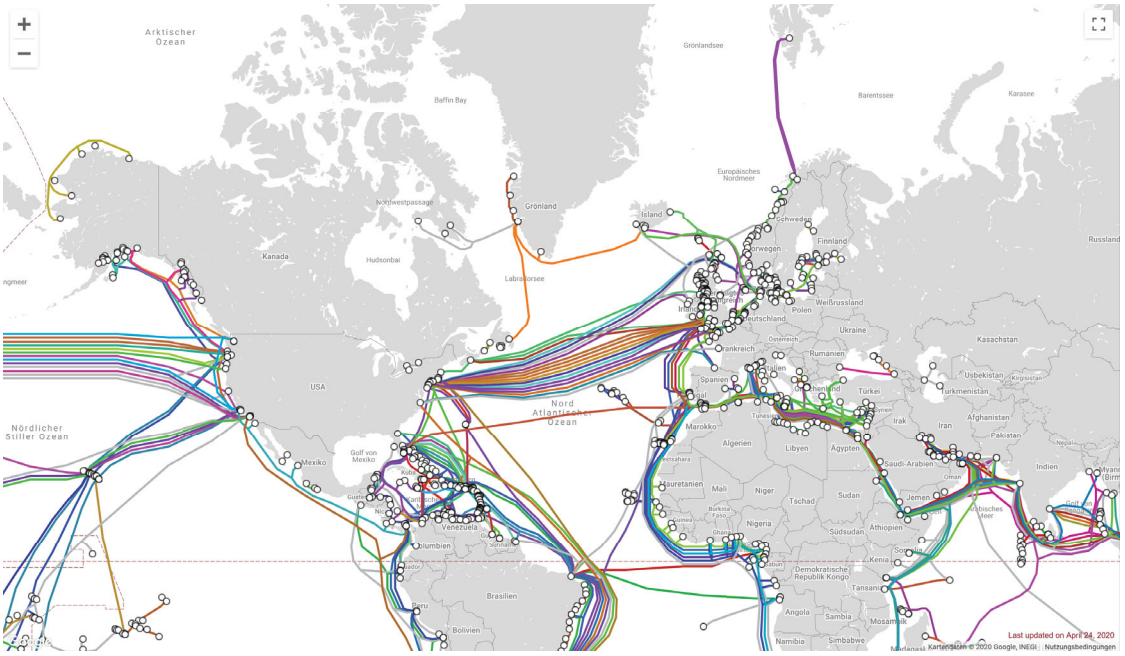
Bei Glasfaser werden Daten im Form von Licht übertragen – einer elektromagnetischen Welle.

Das Licht wird in den Glaskern des Kabels geleitet. Um den Kern herum befindet sich ein Mantel mit einer größeren optischen Dichte als der Glaskern, so dass es zur Totalreflexion kommt und die Lichtwellen sich nur innerhalb des Glaskerns ausbreiten. Durch eine schützende Ummantelung wird das Glas vor Beschädigungen geschützt, aber auch vor Störungen: es können keine Lichtsignale von außen eindringen und die Übertragung stören.

Glasfaserkabel haben typischerweise eine höhere Bandbreite als Kupferkabel, und die Signalqualität bleibt über längere Strecken sehr gut, so dass sehr hohe Datenraten über weitere Strecken erzielt werden können. Während Kupferkabel bevorzugt in lokalen Netzen eingesetzt werden, werden Glasfaserkabel in Backbones verwendet.

Wie bei den Kategorien von TP-Kabeln gibt es auch unterschiedliche Arten von Glasfaser, die sich in Bandbreite und Signalqualität unterscheiden (http://de.wikipedia.org/wiki/Lichtwellenleiter).

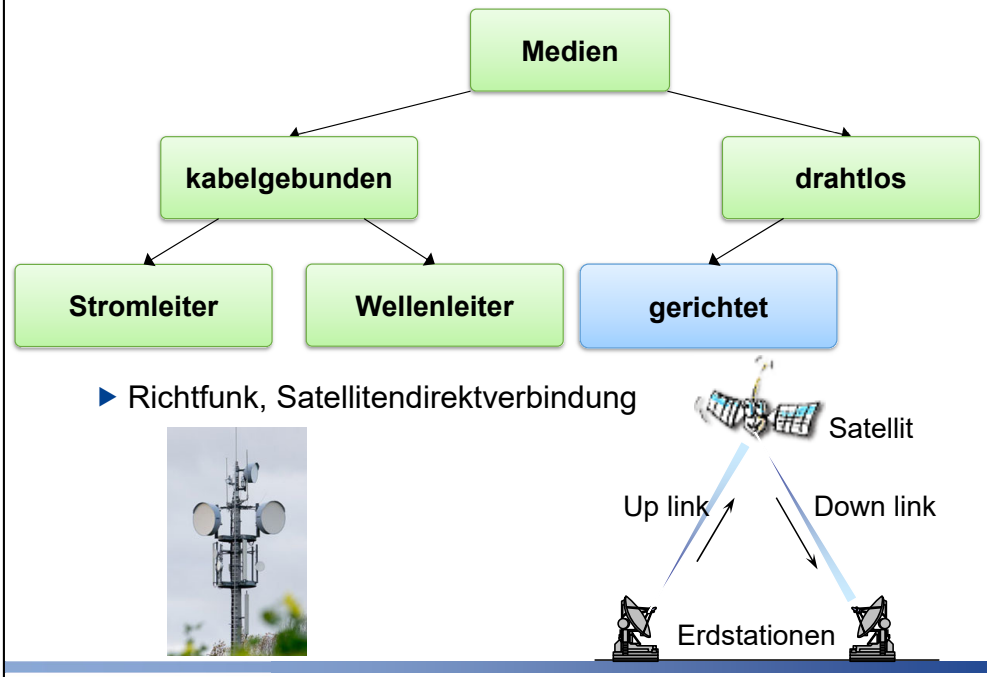
Physikalische Medien - Glasfaser



Video zur Unterwasserübertragung: https://youtu.be/4xGxotBk8AM?t=23412

(Das Video beinhaltet noch deutlich mehr drum herum, nicht von der Länge abschrecken lassen. Der Unterwasserteil beginnt bei ca. 6:30)

Physikalische Medien

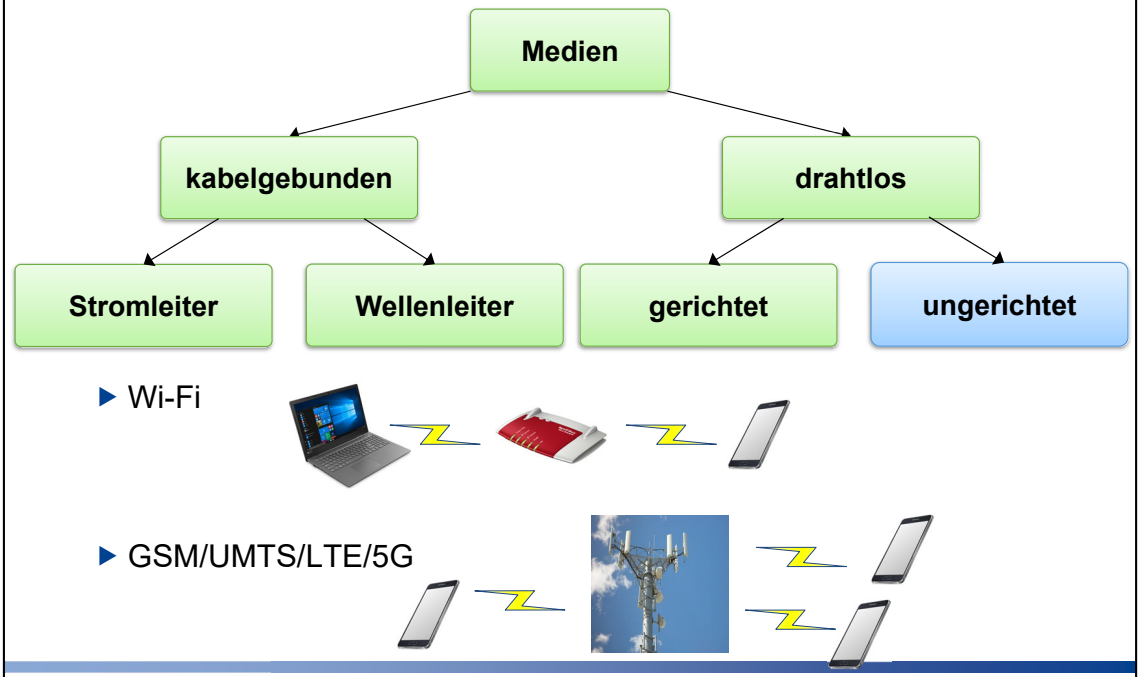


- Richtfunk, Satellitendirektverbindung

Auf drahtlosen Kanälen werden Daten analog als elektromagnetische Welle übertragen. Digitale Daten werden auf den Funkkanal aufmoduliert.

In gerichteten System verwendet man Frequenzen von 10^9 - 10^{11} Hz. Beispiel Satellitendirektverbindung: ein Transponder im Satellit empfängt Daten auf einem Kanal und sendet auf einem anderen Kanal. Pro Kanal steht eine hohe Bandbreite zur Verfügung.

Physikalische Medien



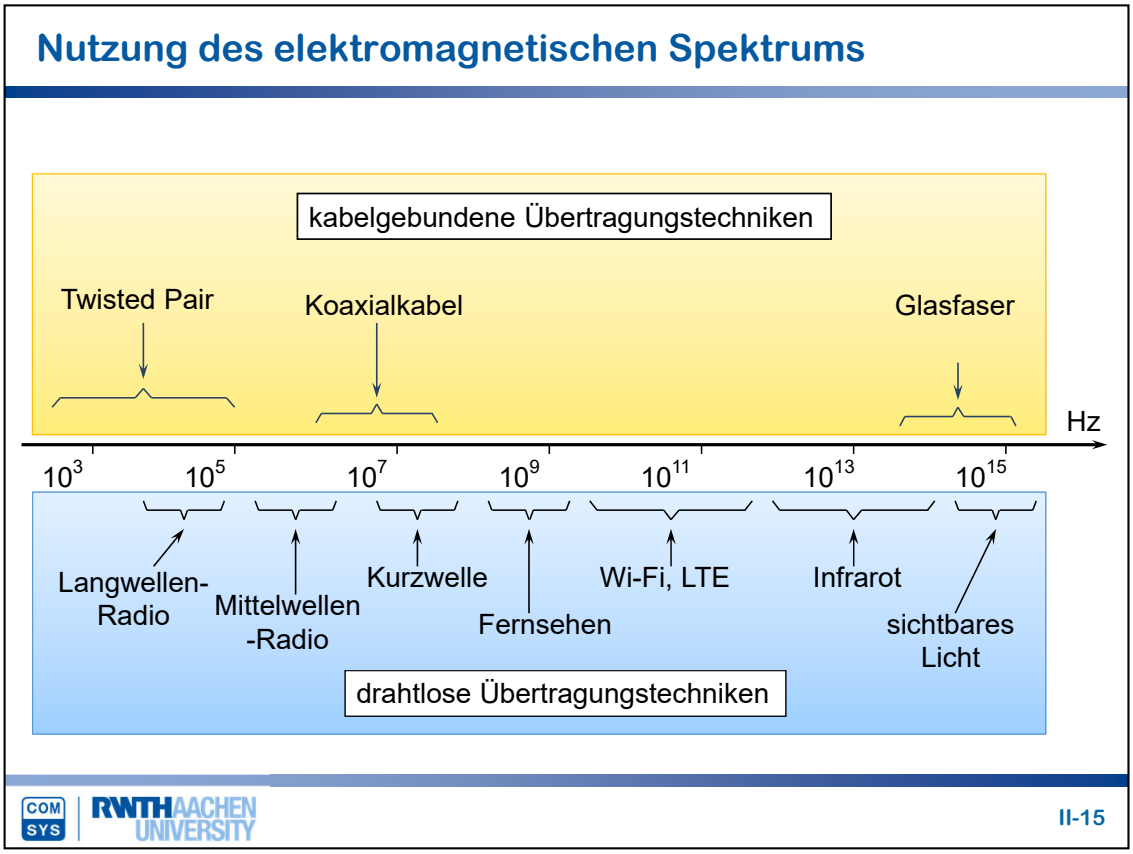
http://legmannews.com/global-lte-base-station-enodeb-market-2017-2022-by-players-regions-product-types-applications/

II-14

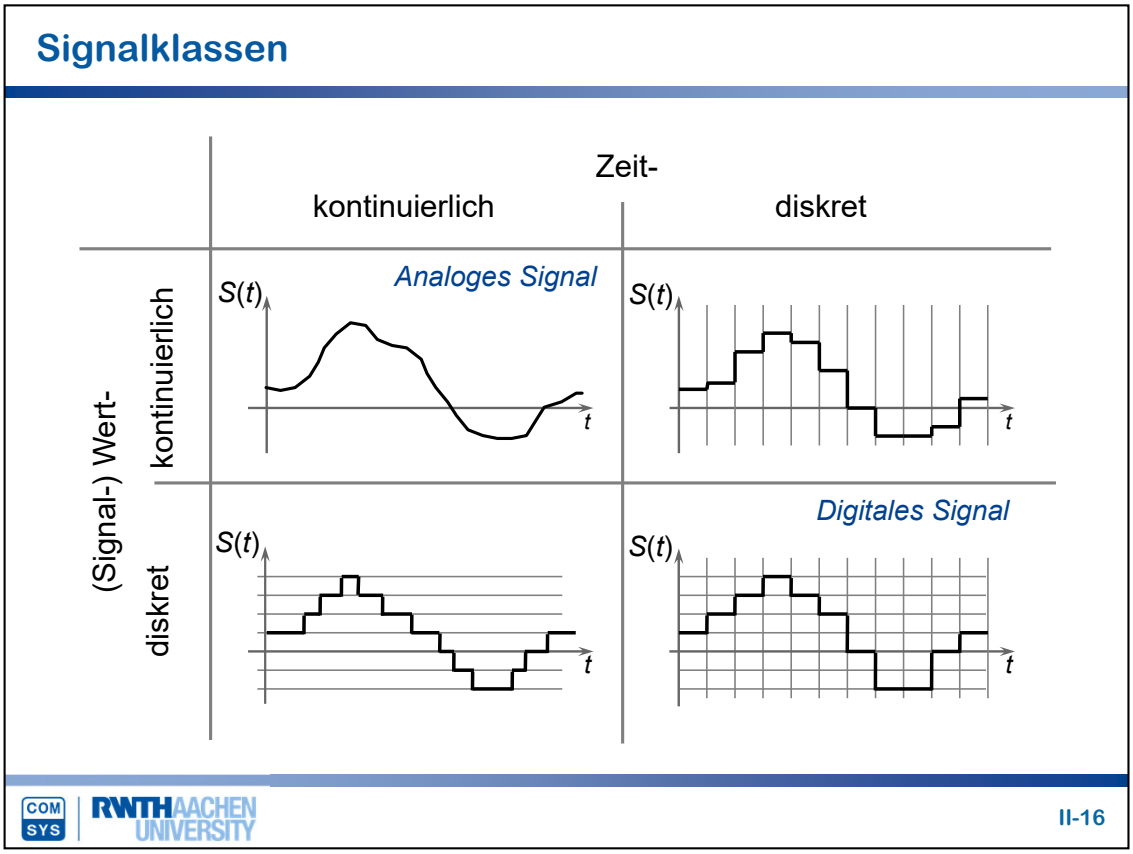
Auf drahtlosen Kanälen werden Daten analog als elektromagnetische Welle übertragen. Digitale Daten werden auf den Funkkanal aufmoduliert. Es werden Frequenzen im Bereich 10^4 - 10^9 Hz verwendet.

Bei ungerichteten System gibt es meist eine Basisstation (oder Access Point), zu der sich alle Geräte in Empfangsreichweite verbinden. Die Geräte (Laptops, Smartphones, …) kommunizieren über diese zentrale Komponente mit anderen drahtlosen Geräten oder in kabelgebundene Netze.

Das generelle Prinzip von Wi-Fi und Mobilfunknetzen wie LTE ist hierbei das gleiche. Nur die Umsetzung ist aufgrund des Einsatzzwecks unterschiedlich: Wi-Fi ermöglicht Kommunikation im lokalen Umfeld mit hohen Datenraten, Mobilfunknetze erzielen geringere Datenraten, bieten dafür aber eine sehr gute Mobilitätsunterstützung.



Gemeinsam haben letztlich alle relevanten Medien, dass elektromagnetische Signale zur Datenübertragung genutzt werden. Gezeigt werden hier die Frequenzbänder verschiedener Übertragungsmedien.



Ganz allgemein kann der Begriff des Signals noch weiter gefasst und auf andere Bereiche angewendet werden. Ein Signal kann Informationen transportieren, indem die Signalparameter modifiziert werden (z.B. +1 Volt und -1 Volt Spannung auf einem Kupferkabel).

Neben den hier interessanten Signalen gibt es auch ortsabhängige und somit räumliche Signale, die für das Speichern von Daten verwendet werden. Dazu zählen beispielsweise optische Speicher (wie beschriebenes oder bedrucktes Papier), optische Platten (wie CD-R oder DVD) oder magnetische Speicher wie z.B. die Festplatte.

Interessant für uns sind zeitabhängige Signale, deren Parameter im laufe der Zeit verändert werden, um z.B. eine Bitfolge übertragen zu können. Als Grundsatz gilt dabei: Jedes ortsabhängige Signal ist in ein zeitabhängiges Signal überführbar („Lesen“, Abtasten) und umgekehrt („Schreiben“, Aufzeichnen). Diese Vorlesung behandelt ausschließlich zeitabhängige Signale. Für die Klassifikation solcher Signale sind die Parameter „Zeit“ und „Signalwert“ von Interesse. Signale werden graphisch häufig durch Ihren von der Zeit abhängigen Signalwert dargestellt.

Die auf nachrichtentechnischen Kanälen eingesetzten Signale lassen sich in vier Klassen einordnen. Die Signalklassen machen eine Aussage über den Signalverlauf, welcher durch seinen Signalwert-Verlauf (y-Achse *s*) über die Zeit (x-Achse *t*) bestimmt ist. Die unterschiedlichen Signalklassen ergeben sich aus der Kombination des Wert- und Zeitverlaufs:

- Kontinuierlich: stetiger Verlauf (kein Abstand zwischen je zwei Punkten)
- Diskret: sprunghafter Verlauf (Einschränkung auf bestimmte Werte)

Der kontinuierliche und der diskrete Fall können in beliebigen Kombinationen auf Wert- und Zeitverlauf angewendet werden. Insgesamt können somit 4 Signalklassen unterschieden werden:

- Signal- und zeitkontinuierlich: z.B. analoges Telefon, Rundfunk: Weder der Signalwert, noch der Zeitverlauf wird zerhackt. Das entspricht dem klassischen analogen Signal.

- Signalkontinuierlich und zeitdiskret: z.B. periodisches Messen von analogen Werten eines technischen Prozesses: Der technische Prozess könnte z.B. ein Überwachungsprozess sein, in dem von einem Sensor alle 10 Sekunden oder auch auf Anfrage die bestehende Lichtintensität in [Lux] gemeldet wird.
- Signaldiskret: digitale Übertragung mit beliebigen Signalwechseln (zeitkontinuierlich) oder festem (meist isochronem) Taktraster (zeitdiskret). In diesem Fall spricht man von einem digitalen Signal.

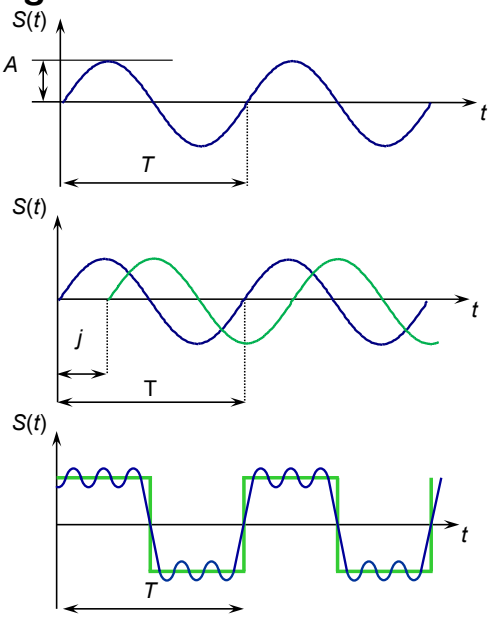
Wenn möglich, wird heutzutage die physikalische Übertragung von Daten digital vorgenommen, da hierdurch eine Übertragung der Daten über weitere Strecken mit besserer Qualität möglich ist. Dies kann man sich einfach anhand des Beispiels „Sprache“ klar machen: überträgt man Sprachdaten analog, werden die Wellenformen im laufe der Übertragung durch Störeinflüsse verzerrt, d.h. die Amplitudenwerte werden modifiziert. Da jeder Amplitudenwert gültig ist, kann der Empfänger nicht erkennen, wo eine Modifikation stattgefunden hat. Gibt man die Sprachinformation wieder, hat man im Hintergrund ein Rauschen. Müssen die Sprachinformationen weitergeleitet werden, werden sie bereits verzerrt weitergeleitet und auf der nächsten Teilstrecke weiter verfälscht. Digitale Signale sind nicht so anfällig – da nur bestimmte Signalwerte erlaubt sind, kann der Empfänger die erhaltene Signalfolge abtasten und die verzerrten Signale auf die nächsten erlaubten Werte korrigieren. So werden die meisten Störungen rausgefiltert und auch bei einer Weiterleitung über mehrere Strecken keine Fehler propagiert.

Die zweiwertige digitale Übertragung (Binärübertragung) ist ein Spezialfall des signaldiskreten Verlaufs, da hier die zulässigen Signalwerte auf zwei begrenzt werden. Dieser Spezialfall wird in der Praxis oft verwendet (siehe im weiteren Verlauf des Kapitels: Leitungscodes).

Es gibt mehrere Verfahren im Bereich der Datenkommunikation, um ein Signal einer bestimmten Klasse in ein Signal einer anderen Klasse zu wandeln.

Periodische und digitale Signale

- **Kenngrößen periodischer Signale:**
  - Periode T, Frequenz $f = 1/T$, Amplitude A, Phase j
- **Beispiele:**
  - Sinus-Schwingung
  - Phasendifferenz j
  - Rechteckschwingung

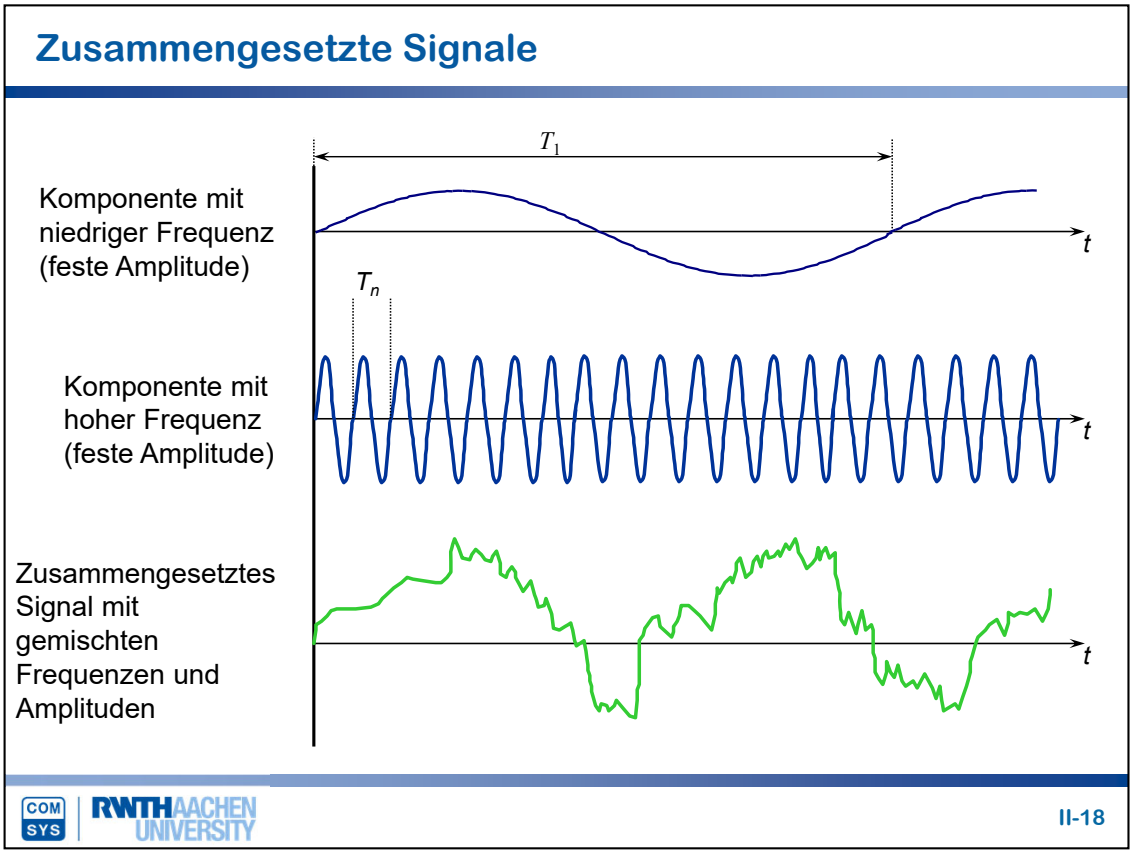


Zur Übertragung werden häufig periodische Sinus-Schwingungen verwendet, wie sie im ersten Beispiel dargestellt sind. Auf dieser Folie wird für die Darstellung der periodischen Signale eine Zeitfunktion gewählt, welche eine Zuordnung von Signalwert und Zeit vornimmt.

Sinus-Schwingungen sind durch die folgenden Parameter gekennzeichnet:

- Periode T: Gibt die zeitliche Dauer einer Schwingung an. Der Kehrwert der Periode ist die Frequenz f, die in Hertz [Hz] angegeben wird.
- Amplitude A: Die Amplitude beschreibt die maximale Auslenkung der Schwingung vom Mittelwert.
- Phase φ: Beschreibt die aktuelle Position im Ablauf einer Schwingung. Wird verwendet, um die Verschiebung des Beginns einer Sinus-Schwingung gegenüber einem idealisierten Nullpunkt anzugeben, wie im zweiten Beispiel gezeigt.

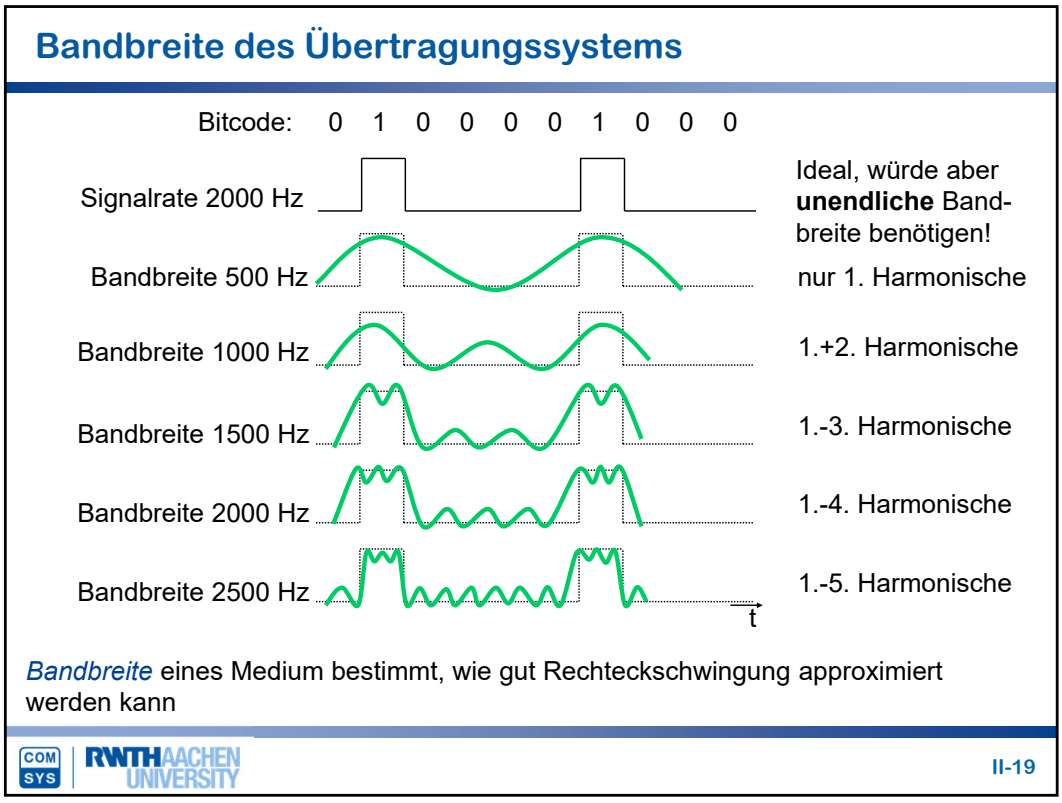
Zur Übertragung digitaler Signale kann z.B. eine Folge positiver und negativer Spannungen auf ein Kupferkabel gegeben werden: +1 Volt für eine 1, -1 Volt für eine 0. Auch dies kann als Schwingung angesehen werden. Mit Sinus-Schwingungen lässt sich, wie im dritten Beispiel angedeutet, solch eine (digitale) Signalfolge darstellen. Häufig wird dazu eine idealisierte Rechteckschwingung mit senkrechten Flanken angenommen, welche allerdings in der Realität nicht übertragen werden kann. Das liegt darin begründet, dass dazu theoretisch eine Sinus-Schwingung mit unendlich hoher Frequenz benötigt wird, was wiederum eine unendliche Bandbreite des Mediums voraussetzt (Erläuterung auf den nächsten Folien).



Eine Eigenschaft elektromagnetischer Schwingungen ist, dass sich verschiedene Schwingungen bei gleichzeitiger Übertragung über ein Medium vermischen: es entsteht eine Schwingung, deren Verlauf sich aus einer Addition der jeweils zu einem Zeitpunkt vorliegenden Amplituden aller Schwingungen ergibt.

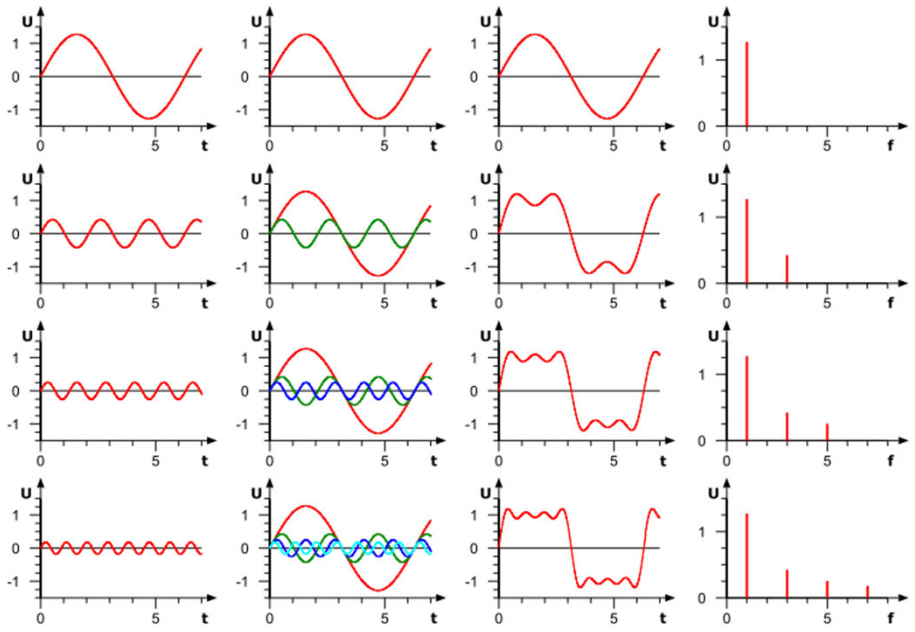
Je nach Mischung verschiedener Sinus-Schwingungen entsteht eine mehr oder weniger irreguläre Schwingung, wie unten auf der Folie dargestellt.

Ideale Sinus- und Cosinus-Schwingungen werden auch harmonische Schwingungen genannt.



Betrachten wir dies umgekehrt: egal, auf welchem Medium man ein Signal überträgt, es setzt sich aus harmonischen Schwingungen zusammen.
Versucht man, ein digitales Signal zu übertragen (Rechteckschwingung, ganz oben auf der Folie), klingt dies erst mal simpel: warum nicht einfach für eine bestimmte Zeitdauer z.B. auf einem Kuperkabel für eine 1 einen Spannungspuls von +1 Volt erzeugen, für eine 0 einen Spannungspuls von -1 Volt. Auf einem Kupferkabel wird solch ein Signal dann allerdings als “Gemisch” von harmonischen Schwingungen übertragen, die überlagert die idealisierte Rechteckschwingung ergeben. Die sogenannte Fourier-Transformation erlaubt es, ein Eingangssignal dahingehend zu analysieren, welche harmonischen Schwingungen (Frequenz und Amplitude) überlagert werden müssen, um das Eingangssignal zu erzeugen. Die Fourier-Analyse soll hier nicht weiter betrachtet werden; wichtig ist nur das Ergebnis: zur Darstellung einer Rechteckschwingung werden unendliche viele harmonische Schwingungen benötigt, d.h. auch Schwingungen mit unendlich hoher Frequenz.
Dies ist aber nur bei unendlicher Bandbreite möglich. Mit Bandbreite bezeichnet man die Breite des Frequenzspektrums, welches über ein Medium übertragen werden kann. Um die harten Pegelwechsel an den Flanken der Rechteckschwingung darzustellen, braucht man unendlich hohe Frequenzen, damit auch eine unendliche Bandbeite. Je nachdem, welches Frequenzspektrum ein Medium übertragen kann, spricht man davon, welche Bandbreite es hat. Der Einfluss der Bandbreite auf die Darstellung der idealisierten Rechteckschwingung ist auf der Folie dargestellt.
Nebenbemerkung: die Grundschwingung des Signals wird als 1. Harmonische bezeichnet. Die 2. Harmonische hat die doppelte Frequenz der Grundschwingung, die 3. Harmonische die dreifache Frequenz usw. Die Periode der Grundschwingung entspricht bei periodischen Signalen der Periode der zu übertragenden Signals.
Von der nutzbaren Bandbreite hängt auch die erreichbare Signalrate ab (d.h. letztendlich auch die Datenrate, da mit jedem Signal eine bestimmte Anzahl von Bits übertragen wird) – je mehr Wechsel eines Parameterwerts man pro Sekunde vornimmt, desto höhere Frequenzen werden zur Darstellung benötigt, um das Signal auf Empfängerseite wieder korrekt interpretieren zu können; je geringer die Bandbreite ist, desto weniger Signale (und damit auch Bit) können pro Sekunde auf das Medium gegeben werden.
Die Abhängigkeiten zwischen Signalrate und Bandbreite werden durch die Formeln von Nyquist und Shannon beschrieben, die im Folgenden noch behandelt werden.

Beispiel: Erzeugung einer Rechteckschwingung

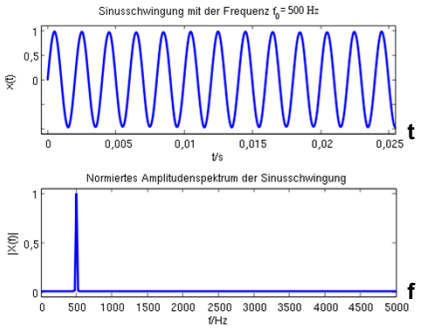


Auf der Folie sieht man dies noch einmal anders dargestellt: die erste Spalte zeigt verschiedene harmonische Schwingungen mit unterschiedlichen Amplituden. In der zweiten Spalte werden die zeilenweise hinzugefügten harmonischen Schwingungen gemeinsam gezeigt, die dritte Spalte zeigt jeweils das Signal, welches aus Überlagerung der betrachteten harmonischen Schwingungen resultiert. Man kann sehen, dass sich das Signal immer weiter einer Rechteckschwingung annähert.

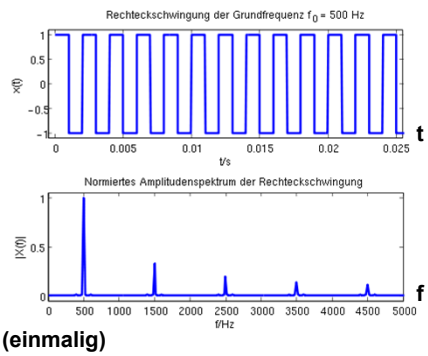
Die vierte Spalte zeigt eine übliche Darstellung, wie sie die Analyse eines überlagerten Signals liefern würde – es ist dargestellt, welche harmonischen Schwingungen mit welcher Amplitude im überlagerten Signal enthalten sind. Diese Darstellung ist äquivalent zu der in der zweiten Spalte, allerdings übersichtlicher.

Approximation eines Rechtecksignals

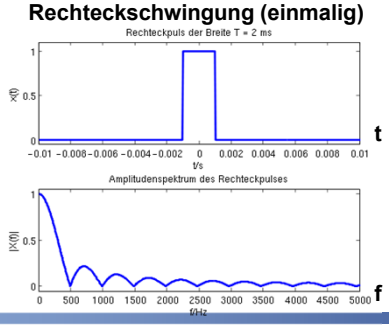
Sinusschwingung (periodisch, endlos)



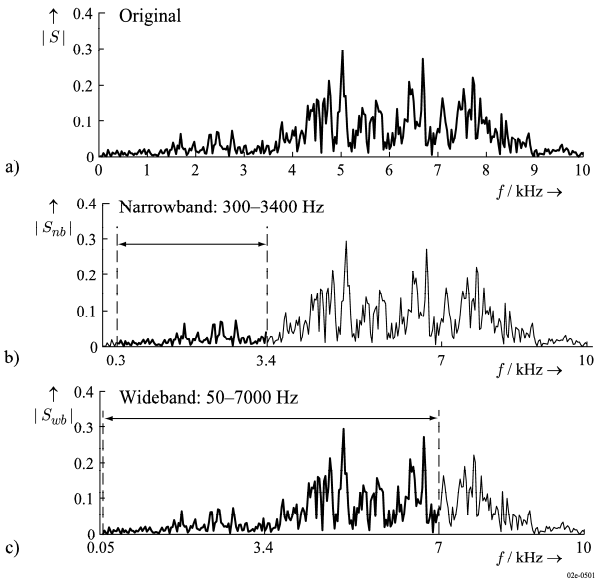
Rechteckschwingung (periodisch, endlos)



Rechteckschwingung (einmalig)



Comparison: Telephone Speech and Wideband Speech



Original

Today's telephone speech

Future wideband telephone speech ("HD-Voice")

Quelle: Vorlesung Digital Speech Transmission | Prof. Dr.-Ing. Peter Jax

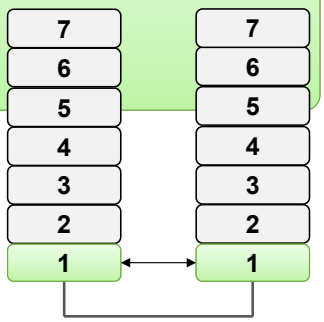
Beispiel Bandbreitenbegrenzung bei Sprache.

b) ist der Normale“ Telefonkanal 300-3400Hz (z.B. Codec G.711)

c) ist „High Definition Sound Performance (HDSP)“ bzw. HD-Voice (Codec G.722)

Kapitel 2: Bitübertragungsschicht

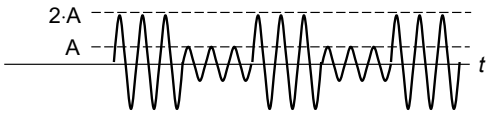
- **Grundlagen**
  - Übertragungsmedien
  - Signale und Bandbreite
- **Übertragung von Signalen**
  - Umformung, Basisband, Modulation
  - Übertragungsparameter, Störeinflüsse
  - Leitungscodes und Modulationsverfahren
  - PCM
- **Kanalnutzung**
  - Multiplexing



Übertragungsverfahren und Umformung

- **Bei digitalen Kanälen wird im Wesentlichen unterschieden zwischen:**
  - „Direkte“ Weitergabe des Eingabesignals
    - Umformung digital → digital
    - *Basisbandübertragung*

  - Aufprägung des Eingabesignals auf harmonische Trägerschwingung, d.h. Eingabesignale werden einer Trägerfrequenz aufmoduliert
    - Umformung digital → analog



- **Zudem: Umwandlung analog → digital**
  - Z.B. Digitalisierung von Sprache zur Übertragung über digitale Kanäle

COM SYS | RWTH AACHEN UNIVERSITY II-24

Die Umformung analog → analog spielt in der Datenkommunikation keine große Rolle. Die anderen Konvertierungen sind allerdings (je nach verwendetem Medium) nötig.

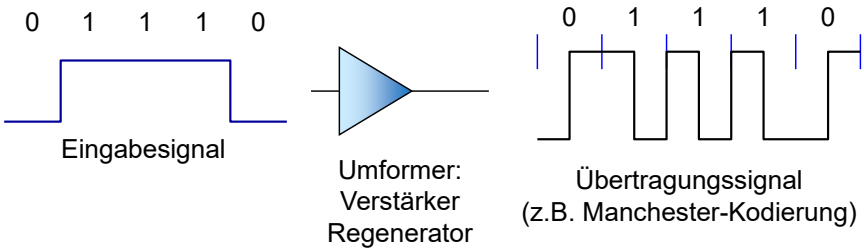
Die einfachste Form eines Übertragungsverfahrens besteht darin, das rechteckförmige digitale Signal direkt mit einer eventuellen Pegelanpassung (also Verstärkung) in das Medium einzuspeisen – z.B. indem zur Übertragung einer „1“ für eine definierte Dauer eine Spannung von +1 Volt an das Kabel angelegt wird, und zur Übertragung einer „0“ eine Spannung von -1 Volt. Dieses Übertragungsverfahren nennt man *Basisbandübertragung*. Es kommt beispielsweise auf Kupferkabeln in Ethernet zum Einsatz.

Durch die Art des Verfahrens erzeugt man eine Rechteckschwingung auf dem Medium. Wie auf Folie 18 dargestellt, verwendet man dadurch die gesamte Bandbreite des Mediums zur Übertragung. Während der Übertragung wird die Rechteckschwingung verzerrt, da das medium nur eine begrenzte Bandbreite hat.

Bei der Übertragung über ein analoges Medium nutzt man eine geeignete Trägerschwingung dazu, das eigentliche (digitale) Nutzsignal zu transportieren. Die digital vorliegenden Daten werden der Trägerschwingung aufgeprägt, d.h. modifizieren ihre Parameter. Z.B. könnte man zur Übertragung einer „1“ die Trägerschwingung mit einer Amplitude von a aussenden, für eine „0“ mit einer Amplitude von 2·a. Zur Übertragung des entstehenden Signals wird eine bestimmte Bandbreite benötigt, aber keine unendliche Bandbreite wie bei der Basisbandübertragung. Dieses Verfahren eignet sich also, wenn man nur einen Kanal mit einer begrenzten Bandbreite zur Verfügung hat und das Eingabesignal an diese Bandbreite anpassen muss. Der Umformer wird als Modulator, der Rückformer als Demodulator bezeichnet. Unterstützt das Übertragungssystem beide Übertragungsrichtungen, so werden die beiden Komponenten zusammengefasst als *Modem* bezeichnet.

Auch die Konvertierung analoger Quellsignale in digitale Signale ist möglich – dies findet z.B. bei der Telefonie statt, wo analoge Sprachsignale in digitale Signale überführt werden, die dann übertragen werden. Diese digitalen Sprachdaten werden dann allerdings wieder umgeformt, um auf einem Medium (Telefonkabel oder Funk) übertragen werden zu können. Mehrere Signalkonversionen können also hintereinander ausgeführt werden, wobei Analog → Analog und Analog → Digital nie verlustfrei ist!

Basisbandübertragung



- **Einfachste Form der Signalwandlung:**
  - Umformung digital → digital
  - Eingabesignal muss an Mediencharakteristik angepasst werden
  - Abbildungsvorschrift auf das Übertragungssignal wird als *Leitungscodierung* bzw. Übertragungscodierung bezeichnet

Basisbandübertragungsverfahren

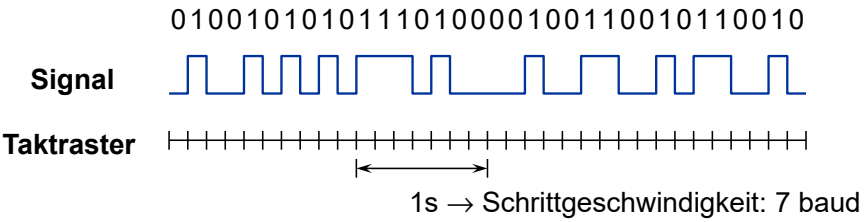
Hier werden die digitalen Daten einfach als digitale Signale, d.h. im Beispiel von Kupferkabel als Folge von Rechteckspannungen übertragen.

Die moderne digitale Übertragungstechnik verwendet Basisbandverfahren bis in den Gigabit-Bereich (Bekanntestes Beispiel: Ethernet). Dabei sind folgende Eigenschaften angestrebt:

- Da unter Umständen eine große Anzahl gleicher Werte auftreten kann (z.B. eine längere Folge von 111...11), kann eine Gleichstromkomponente auftreten. Gewünscht ist eine Eliminierung des Gleichstromanteils.
- Wiedergewinnung des Takts aus dem Signal (selbsttaktender Code)
- Erkennung von Signalfehlern auf Signalebene
- Hohe Effizienz, d.h. Übertragung möglichst vieler Bits pro Sekunde

Zum Erreichen dieser Eigenschaften benutzt man meist bestimmte Leitungscodes. Die Zuordnungsvorschrift, welche das digitale Eingabesignal auf das (ebenfalls digitale) Übertragungssignal abbildet, wird Leitungscodierung bzw. Übertragungscodierung genannt.

Digitale Signalübertragung



- ***Schritt***
  - Minimales Zeitintervall T_{min} zwischen aufeinanderfolgenden Änderungen der Signalkoordinate
    - Wird auch als *Schrittdauer* bezeichnet
  - Wichtig: Digitales Signal mit fester Schrittdauer T („Schritt-Takt“)
- ***Schrittgeschwindigkeit***
  - Bei Digitalsignalen mit festem Zeitraster: Kehrwert der Schrittdauer
  - Einheit: *baud* = 1/s

Im Zusammenhang mit dem digitalen Signal existieren zahlreiche Begriffe, die fest im Sprachgebrauch innerhalb des Datenkommunikationsbereichs verankert sind.

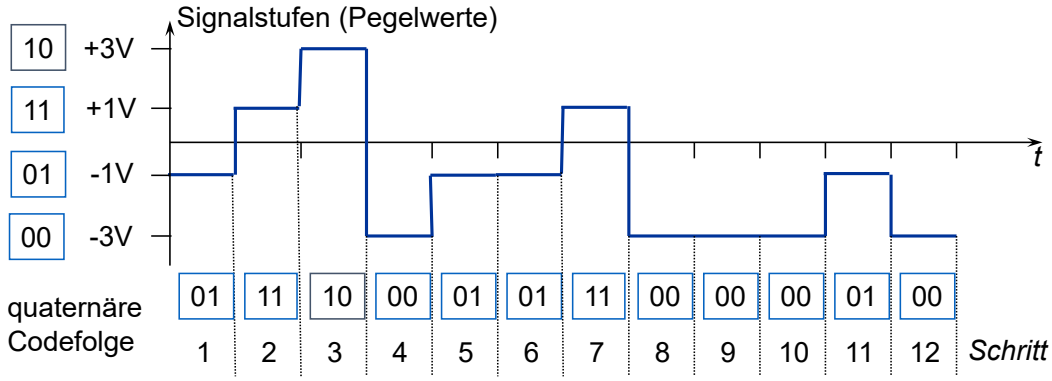
Den Ausgangspunkt bildet das oben angegebene digitale Signal, welches in diesem Fall zwei diskrete Signalwerte annehmen kann. Ein Begriff, welcher im Zusammenhang mit der Diskretisierung des Zeitverlaufs relevant ist, ist das Taktraster. Beim Empfänger definiert das Taktraster den Zeitpunkt, zu welchem ein Signal detektiert werden kann. Darauf bezogen ist ein Signalschritt die minimale Zeiteinheit, in der ein Sender ein Signal auf das Medium geben kann.

Die Schrittgeschwindigkeit ist die Anzahl der Signalschritte pro Zeiteinheit, welche in [baud] gemessen wird. Sie gibt an, wie viele Signale (und damit in Folge auch: wie viele Bit) ein Sender pro Sekunde auf das Medium setzen kann.

Bitte beachten: ein Signal entspricht nicht notwendigerweise genau einem Bit, so dass die Datenrate (Übertragungsrate) auf einem Medium, die in Bit/s gemessen wird, zwar von der Schrittgeschwindigkeit abhängt, aber nicht den gleichen Wert haben muss.

Mehrwertiges Digitalsignal

- **Zweiwertiges Signal**
  - Digitales Signal mit nur zwei Werten des Signalparameters
- **Mehrwertiges Signal**
  - Die (diskrete) Signalkoordinate kann mehr als zwei Werte annehmen
  - Z.B. 2B1Q (ISDN): zwei Bit pro Koordinatenwert (quaternäres Signal)



Bei dem vorherigen Beispiel sind wir davon ausgegangen, dass ein Signal immer genau ein Bit codiert. In diesem Fall spricht man von einem zweiwertigen Signal: der Signalparameter kann nur zwei Zustände annehmen. Jedoch besteht die Möglichkeit, mit einem Signalschritt mehrere Datenwerte (eine sog. Codefolge) zu codieren, wie das obige Beispiel zeigt. In diesem Fall spricht man von *mehrwertigen* oder *mehrstufigen* Signalen, d.h. die (diskrete) Signalkoordinate kann mehr als zwei Werte annehmen. Die Anzahl *n* der diskreten Werte (Kennwerte, Stufen), die ein Signalelement annehmen kann, wird wie folgt gekennzeichnet:

n = 2: binär (= zweiwertiges Signal)

n = 3: ternär

n = 4: quarternär (= DIBIT)

...

n = 8: oktonär

Gehen wir von einem gleichen Taktraster aus, so können also bei mehrwertigen Signalen mehr Daten übertragen werden als bei zweiwertigen Signalen. Hier wird jetzt der Unterschied zwischen der *Schrittgeschwindigkeit* und der *Übertragungsgeschwindigkeit (Datenrate)* klar. Die Schrittgeschwindigkeit (in baud) bezieht sich ausschließlich auf das Taktraster, also die Anzahl der potentiellen Signalparameterwechsel. Im Gegensatz dazu bezieht sich die Übertragungsgeschwindigkeit auf die Menge der tatsächlich übertragenen Daten. Für zweiwertige Signale wird in jedem Schritt ein Bit codiert. Deshalb gilt in diesem Fall:

- *Schrittgeschwindigkeit in baud = Übertragungsgeschwindigkeit in bit/s*

Die Übertragungsgeschwindigkeit wird in diesem Fall als *Bitrate* (oder auch *Datenrate*) bezeichnet.

Für mehrstufige Signale (mit *n* möglichen Wertestufen) gilt:

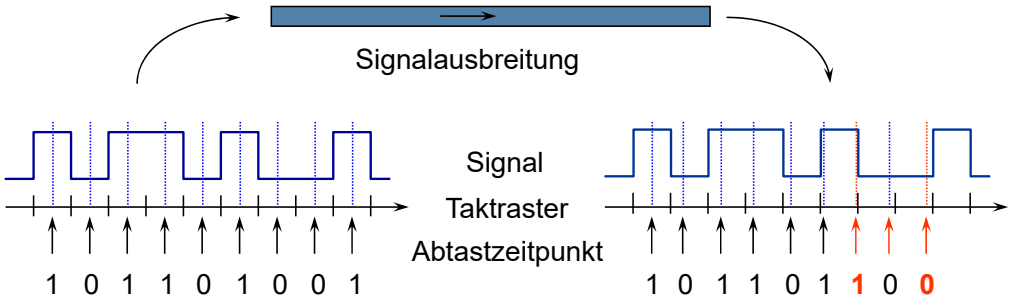
- *Übertragungsgeschwindigkeit in Bit/s = Schrittgeschwindigkeit* * ld*(n)*

wobei ld für den Logarithmus zur Basis 2 steht. Bei DIBIT wäre demnach 1 baud = 2 bit/s.

(Bitte beachten: dies ist eine Brutto-Übertragungsgeschwindigkeit, also das erreichbare Maximum. In der Praxis kann die erzielbare Datenrate niedriger liegen, in Abhängigkeit von der gewählten Codierung.)

Fehlerquelle: Bitfehler durch fehlerhafte Synchronisation

- **Wesentlich für eine fehlerfreie Übertragung sind korrekt *synchronisierte* Sender und Empfänger**
  - Ungenaue Abtastzeitpunkte können u.U. zu sporadischen Bitfehlern führen, obwohl die Signalfolge korrekt übertragen wurde
  - Falsch synchronisierte Sender und Empfänger führen häufig zu einer vollständig falsch interpretierten Bitfolge



Es gibt eine Vielzahl an Leitungscodes, die im Laufe der Zeit entwickelt wurden, um im Basisband zu übertragen. Generell versucht man eine Robustheit gegenüber Störungen zu schaffen (üblicherweise nur Verwendung binärer Codes, damit die Signalwerte nicht zu ähnlich sind) und Probleme durch Taktverlust oder Gleichstromübertragung zu vermeiden (häufiger Wechsel des Übertragungspegels).

Das Beispiel auf dieser Folie zeigt die Auswirkung von Taktverlust - ungenaue Abtastzeitpunkte, die zu sporadischen Bitfehlern führen. Das Problem resultiert schlicht daraus, dass die Uhren zweier Rechner nicht exakt gleich schnell laufen und daher die Dauer eines Schritts auf beiden Rechnern unterschiedlich ist.

Der weitaus schlimmere Fall tritt ein, falls Sender und Empfänger gänzlich falsch synchronisiert sind und den Beginn einer Bitfolge unterschiedlich interpretieren. Dies führt in der Regel zu einem vollständigen Ausfall der Kommunikation, da Sender und Empfänger sich nicht mehr verstehen können.

Ein unvermeidbares Problem hierbei ist, dass eine exakte Synchronisation zweier beliebiger Rechner nicht möglich ist; die internen Rechneruhren, die den Takt vorgeben, laufen mit leicht unterschiedlichen Geschwindigkeiten, so dass das auf der Folie gezeigte Problem über einen längeren Zeitraum hinweg definitiv auftreten wird. Vermieden werden kann es entweder durch einen externen Taktgeber, durch eine künstliche Beschränkung der Länge von Bitfolgen, die übertragen werden können, oder durch eine Selbsttaktung innerhalb des verwendeten Codes.

Die völlig falsche Synchronisation kann vermieden werden, indem man der Übertragung einen „Startcode“ vorstellt; dieser wird als Präambel bezeichnet, siehe nächstes Kapitel.

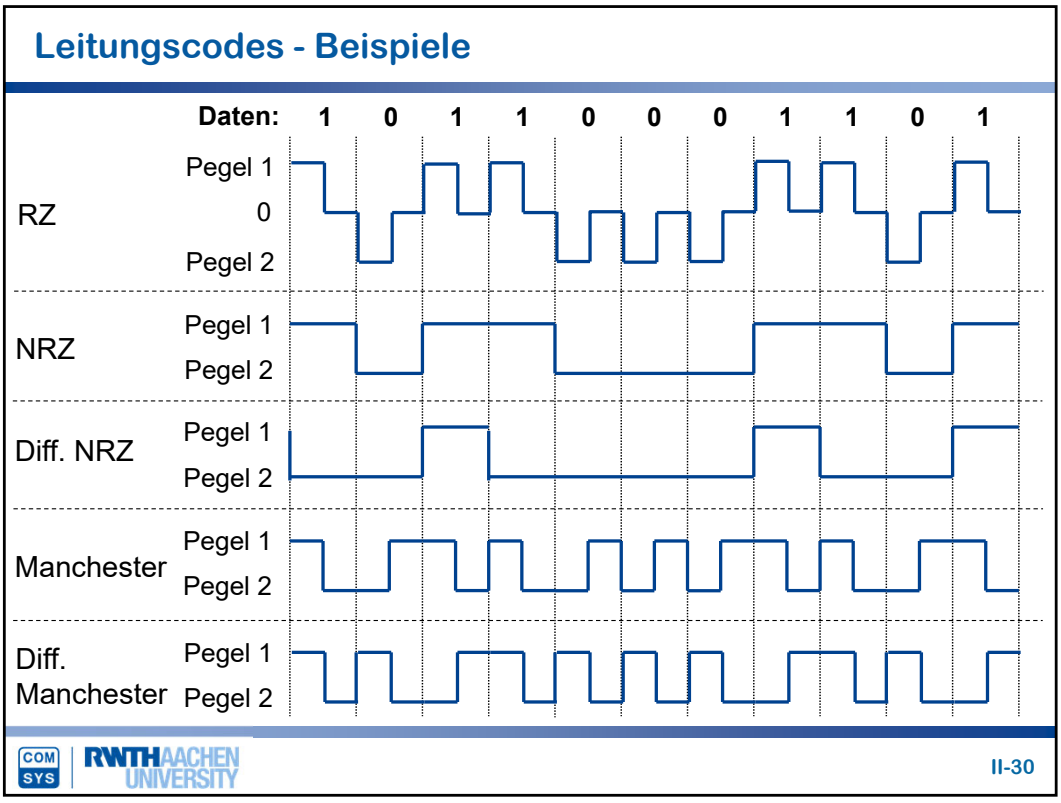
Aspekte beim Design von Leitungscodes

- *Effizienz*
  - Wie viele Bits können pro Schritt codiert werden?
- *Robustheit*
  - Wie kann man eine Fehlinterpretation eines empfangenen Signals aufgrund von Verzerrungen vermeiden?
- *Synchronisation/Taktrückgewinnung*
  - Ist eine Synchronisation zwischen Sender und Empfänger nötig oder kann zusammen mit den Daten ein Takt codiert werden?
- *Gleichstromfreiheit*
  - Eine Gleichspannung kann nicht über lange Strecken übertragen werden, daher müssen Pegelwechsel erfolgen

Probleme mit Gleichstrom treten dadurch auf wenn längere Zeit auf einem Pegel übertragen wird. Beim einfachen Code, der bisher als Beispiel verwendet wurde, hätte man z.B. einen Gleichstrom, der über die Leitung fließt, wenn 1000 1en am Stück übertragen würden (1000 Schittdauern +1 Volt).

Gleichstrom möchte man aus mehreren Gründen vermeiden. Zum einen ist die Übertragung eines Gleichstroms über lange Leitungen nicht möglich, da jedes Kabel als Widerstand auf den Gleichstrom wirkt und ihn abschwächt. Zum anderen kann man bei Verwendung gleichstromfreier Codes über das gleiche Kabel Daten übertragen und gleichzeitig sogenannte Repeater (siehe nächstes Kapitel) mittels Gleichstrom fernspeisen. Zudem würden bei Verwendung von Gleichstrom zur Datenübertragung z.B. Kondensatoren in Repeatern oder anderen Netzkomponenten immer weiter aufgeladen, bis sie voll wären und gar kein Strom mehr fließen würde.

Im Folgenden werden mehrere mögliche Leitungscodes betrachtet, die im Laufe der Zeit entwickelt wurden, um Robustheit, Taktrückgewinnung und Gleichstromfreiheit zu erzielen und trotzdem effizient Daten übertragen zu können.



Der RZ-Code codiert eine die Bitwerte 1 und 0 durch zwei unterschiedliche Pegel. Dies können z.B. +1 und -1 Volt sein: für eine "1" würde für einen Schritt ein Puls von +1 Volt auf die Leitung gesetzt, für eine "0" ein Puls von -1 Volt. Nach der Übertragung eines Bits geht der RZ-Code auf einen dritten Pegel über – für einen Schritt wird ein Pegel von 0 gehalten. Daher resultiert der Name dieses Codes: nach der Übertragung jedes Bits kehrt man zum Nullwert zurück. Durch diese konstante Rückkehr zum Nullwert bettet man ein Taktsignal in die Übertragung ein: pro Bit benötigt man nun zwei Schritte und wechselt bei jedem Schritt garantiert den Pegel. Dadurch ist Taktrückgewinnung garantiert. Allerdings auf Kosten der Effizienz: der RZ-Code hat eine Effizienz von 50%, da er pro Schritt sozusagen nur ein halbes Bit darstellt. Da die Schrittgeschwindigkeit nicht beliebig erhöht werden kann (Bandbreite, siehe folgende Folien), kann mit diesen Codes nur die Hälfte der Datenrate erreicht werden wie bei Verwendung von NRZ-Codes. Auch Gleichstromfreiheit kann der RZ-Code nicht garantieren – werden lange 1- oder 0-Folgen übertragen, sendet man nur auf einem Pegel (plus 0-Pegel).

NRZ-Codes (Non-Return-to-Zero) nutzen nur zwei unterschiedliche Pegel, um Signale darzustellen. Dies können z.B. wie oben +1 und -1 Volt sein. Pro Schritt wird ein Bit übertragen, indem für die Schrittdauer ein konstantes Signal auf einem der Pegel auf die Leitung gesetzt wird. Beim einfachen NRZ (in der Literatur auch NRZ-L genannt) wird für eine "1" ein Puls von z.B. +1 Volt ausgesendet, für eine "0" ein Puls von -1 Volt. Dies wurde bisher bereits als Beispiel verwendet. Einen 0-Pegel gibt es hier nicht – daher in Anlehnung an den RZ-Code der Name NRZ.
Eine Variante ist das differentielle NRZ (auch NRZI-M oder NRZ-M), bei dem eine "1" durch einen Wechsel des Pegels dargestellt wird und eine "0" durch Beibehaltung des Pegels. Die Bezeichnung "differentiell" bezieht sich also darauf, dass jeder Bitwert nun basierend auf dem vorher anliegenden Pegel codiert wird.
Die Codes dieser Kategorie ereichen keine Selbsttaktung und Gleichstromfreiheit, falls lange Folgen des gleichen Eingabesignals übertragen werden müssen. Bei differentiellem NRZ sorgen lange 0-

Folgen für Probleme, bei einfachem NRZ zusätlich auch lange 1-Folgen.
Dafür sind diese Codes allerdings effizient, da pro Schritt ein Bit übertragen werden kann, und robust, da die Pegelwerte gut voneinander unterschieden werden können.
(Mehrwertige Codes wäre effizienter, da sie pro Schritt mehrere Bits übertragen können, aber im Normalfall auch weniger robust, da der Empfänger aufgrund von Verzerrungen mehr Probleme haben kann, den korrekten Signalwert zu identifizieren.)

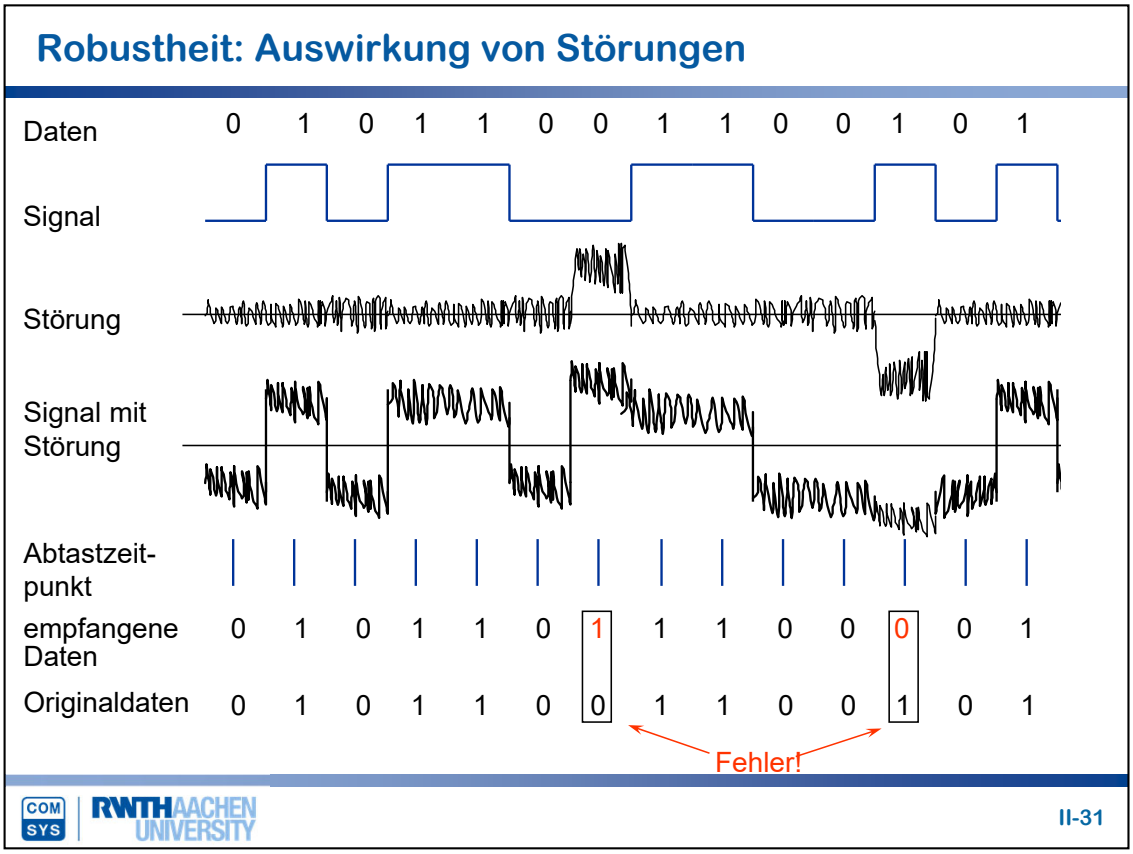
Ene weitere wichtige Klasse sind sogenannte Biphase-Codes. Diese sind selbsttaktend und vermeiden Gleichstrom, da sie regelmäßige Pegelwechsel wie RZ, aber ohne 0-Pegel garantieren.
Beim Manchester-Code (auch Biphase-L) wird die 1 durch einen Wechsel vom hohen zum niedrigen Pegel dargestellt, die 0 durch einen Wechsel vom niedrigen zum hohen Pegel.
Beim differentiellen Manchester-Code wird wieder differentiell codiert: zur Darstellung der 0 wird am Anfang des Bitwerts der Pegel gewechselt, zur Darstellung der 1 wird am Anfang des Bitwerts weiterhin der Pegel verwendet, auf dem zuvor übertragen wurde.
Hier hat man wie beim RZ-Code wieder ein Effizienzproblem: Biphase-Codes benötigen zwei Schritte, um ein einzelnes Bit zu übertragen

Es hängt also von der Codierung ab, ob tatsächlich die komplette mögliche Datenrate ausgenutzt werden kann oder nicht.

Bei dem Manchester- und dem differentiellen Manchester-Code gibt es leider je zwei Definitionen, bei denen die Darstellung der 1 und der 0 vertauscht sind. Daher ist es in der Praxis wichtig, darauf zu achten, welche Variante eingesetzt wird, bevor man einen Netzwerkkartentreiber schreibt. Daher soll es z.B. für die Klausur auch ausreichen, wenn man das Prinzip kennt, aber 1- und 0-Repräsentation verwechselt.

(Sollten entsprechende Übungsaufgaben kommen, wird allerdings genau die hier abgebildete Variante erwartet!)

Des weiteren gibt es noch viele Varianten von RZ-, NRZ- und Biphase-Codes, die hier allerdings nicht weiter betrachtet werden.



Störeinflüsse können sich derart auswirken, dass das empfangene Signal nicht mehr korrekt interpretiert werden kann. Im Beispiel wird eine NRZ-codierte Signalfolge angenommen. Dieser wird ein Störsignal überlagert, welches zwei Spitzen aufweist. Diese Spitzen bewirken, dass der Empfänger anstelle eines niedrigen Pegels einen hohen detektiert (wie im ersten Fehler) beziehungsweise umgekehrt (im zweiten Fehler). Durch diese falsche Interpretation wird die ursprünglich gesendete Bitfolge fehlerhaft reproduziert und weitergegeben.

4B/5B – Kodierung

Symb.	Code Gruppe	Binärdarstellung
0	11110	0000
1	01001	0001
2	10100	0010
3	10101	0011
4	01010	0100
5	01011	0101
6	01110	0110
7	01111	0111
8	10010	1000
9	10011	1001
A	10110	1010
B	10111	1011
C	11010	1100
D	11011	1101
E	11100	1110
F	11101	1111

Symb.	Code Gruppe	Bedeutung
Q	00000	Quiet
I	11111	Idle
H	00100	Halt
J	11000	1st of sequential SD-Pair
K	10001	2nd of sequential SD-Pair
T	01101	End
R	00111	Reset
S	11001	Set

- Verwende differentielles NRZ mit vorheriger Codierung der Daten
- Übertragung von 4 Bit in 5 Takten
- Code-Effizienz: 80%

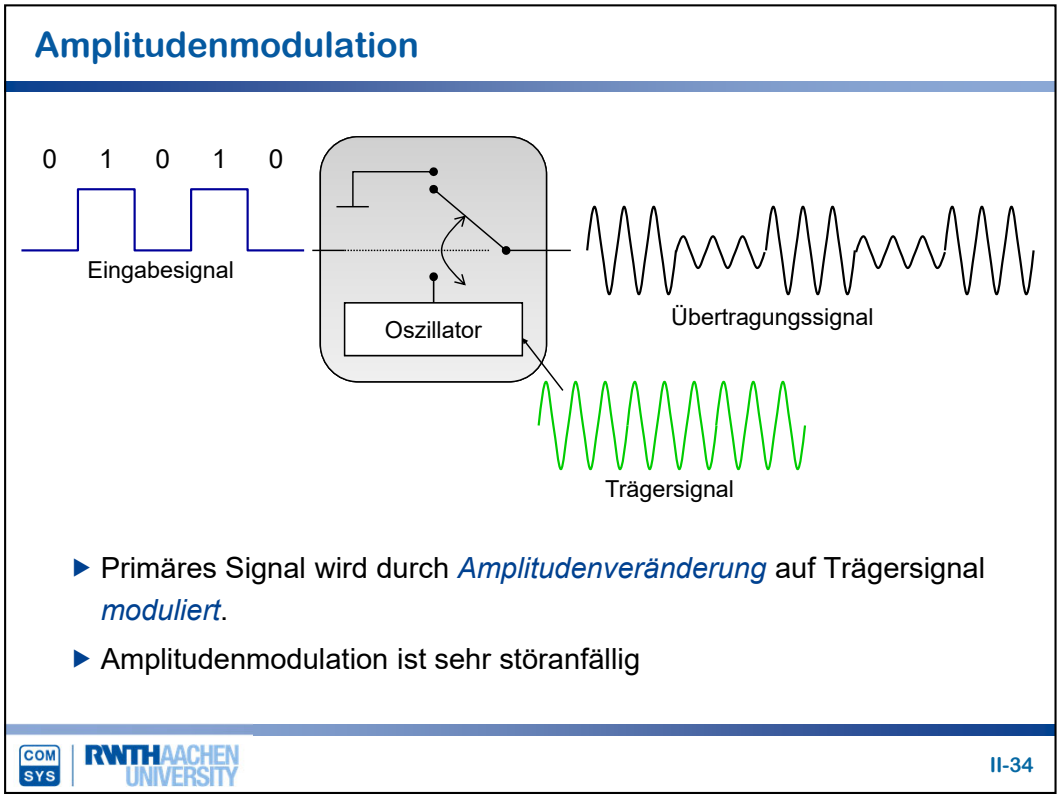
Heutige Codierungsverfahren kombinieren Taktung mit höherer Effizienz. Der einfachste Code dieser Art ist der 4B/5B-Code. Dieser verwendet das differentielle NRZ-Verfahren, welches bei Übertragung einer 1 einen Pegelwechsel vornimmt, bei der Übertragung einer 0 den vorherigen Pegelwert im nächsten Schritt beibehält. Damit sind lange 0er-Folgen bezüglich Taktung und Gleichstrom problematisch – aber diese werden durch eine Codierung der Daten vor der Umformung beseitigt. Man bildet je 4 Bit der zu übertragenden Daten auf 5 Bit Eingabesignal ab, die auf dem Medium codiert werden. Da man dazu nur 16 der 32 möglichen 5-Bit-Codewörter benötigt, kann man sich diejenigen mit möglichst wenigen Nullen aussuchen, so dass bei der Übertragung maximal drei Nullen in Folge auftreten können.

Die nicht benötigten 5-Bit-Codewörter kann man zur schnellen Übertragung von Steuersignalen nutzen.

4B3T-Kodierung im ISDN

Binärwert	S_1	S_n	S_2	S_n	S_3	S_n	S_4	S_n
0001	0-+	1	0-+	2	0-+	3	0-+	4
0111	-0+	1	-0+	2	-0+	3	-0+	4
0100	-+0	1	-+0	2	-+0	3	-+0	4
0010	+-0	1	+-0	2	+-0	3	+-0	4
1001	+0-	1	+0-	2	+0-	3	+0-	4
1110	0+-	1	0+-	2	0+-	3	0+-	4
1011	+-+	2	+-+	3	+-+	4	---	1
0011	00+	2	00+	3	00+	4	--0	2
1101	0+0	2	0+0	3	0+0	4	-0-	2
1000	+00	2	+00	3	+00	4	0--	2
0110	-++	2	-++	3	--+	2	--+	3
1010	++-	2	++-	3	+--	2	+--	3
1111	++0	3	00-	1	00-	2	00-	3
0000	+0+	3	0-0	1	0-0	2	0-0	3
0101	0++	3	-00	1	-00	2	-00	3
1100	+++	4	-+-	1	-+-	2	-+-	3

- In drei Takten werden vier Bit übertragen.
- Nach jeder Übertragung wird ein neuer Zustand angenommen, der die Kodierung der nächsten vier Bit bestimmt.
- Die Taktrate wird reduziert, wobei gleichzeitig zu große unterschiedliche Strompotentiale auf Sender- und Empfängerseite vermieden werden.
- Einsatzbereich: ISDN auf der U_{K0}-Schnittstelle.



Modulierte Übertragung

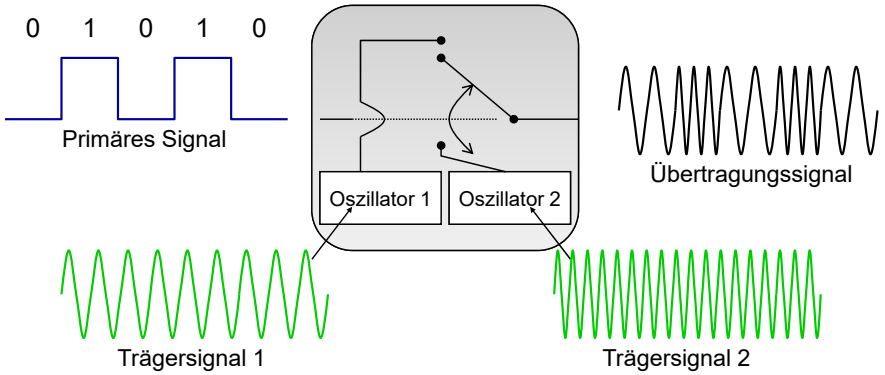
Eine Basisbandübertragung ist nur möglich, wenn man das gesamte Frequenzspektrum des Mediums nutzen kann. Bei Funkübertragung beispielsweise ist dies nicht möglich – das gesamte Frequenzspektrum bei Funkübertragung ist aufgeteilt in einzelne Kanäle, die für bestimmte Zwecke genutzt werden dürfen (Rundfunk, TV, WLAN, GSM, UMTS, LTE, …). Im Gegensatz zu Basisband wird dieses Vorgehen Breitband genannt.

Ist die Übertragung digitaler Daten mittels der Basisbandübertragung nicht möglich, kann man die digitalen Daten an ein analoges Trägersignal (harmonische Schwingung) binden und durch Veränderung seiner Parameter (Amplitude, Frequenz, Phase) die digitalen Daten übertragen. Man nennt den Vorgang der Beeinflussung der Parameter des Trägersignals durch das zu sendende Signal *Modulation*. Es gibt drei Arten von Modulation: Amplitudenmodulation, Frequenzmodulation und Phasenmodulation.

Amplitudenmodulation: Hier wird, abhängig vom Zustand des Modulationssignals die Amplitude des Trägers eingestellt. Im Beispiel auf der Folie entspricht eine „1“ im Übertragungssignal einer Trägerschwingung mit hoher Amplitude, eine „0“ wird durch eine Trägerschwingung mit niedriger Amplitude angegeben. Im Extremfall könnte man auch eine Null-Amplitude verwenden – was einfach „kein Signal“ ist, wir senden also für eine „0“ nichts aus. Amplitudenmodulation ist jedoch sehr störanfällig, da die meisten Störeinflüsse die Amplitude eines Signals verfälschen.

Der Begriff des Schritts findet auch bei den Modulationsverfahren Anwendung. Auch hier ist eine Änderung der Signalkoordinate (in dem Fall der Amplitude) nötig, so dass auch hier Schrittgeschwindigkeiten festgelegt werden müssen. Diese hängen wieder von der Bandbreite des verwendeten Kanals ab (siehe folgende Folien).

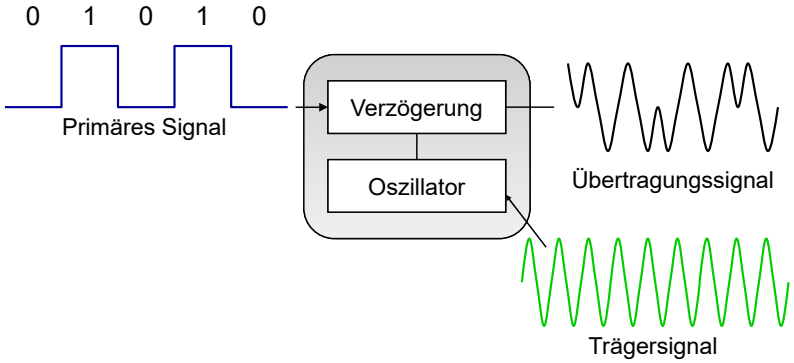
Frequenzmodulation



- Primäres Signal wird durch gezielte *Änderung der Frequenz des Trägersignals* moduliert

Frequenzmodulation: Hier verwendet man mehrere Frequenzen, die nicht sehr weit auseinander liegen, aber trotzdem noch eindeutig erkennbar sind. Der Sender sendet nun für eine „0“ oder „1“ für eine Schrittdauer ein Signal auf der dem Bitwert zugeordneten Frequenz aus. Dieses Verfahren wird beispielsweise beim Rundfunk auf der Ultrakurzwelle verwendet.

Phasenmodulation



- Primäres Signal wird mittels gezielter *Phasensprünge* des Trägersignals moduliert.
- Für Datenkommunikation bestes Verfahren

Phasenmodulation: Hier werden die digitalen Signale durch Verschieben der Phase des Signals codiert. Dies ist das beste, aber auch das aufwendigste Verfahren der analogen Übertragung. Das rührt daher, dass Phasensprünge besser zu detektieren sind als Frequenz- oder Amplitudenänderungen. Zusätzlich ist die Wahrscheinlichkeit der Beeinflussung von Phasensprüngen durch externe Ereignisse wesentlich geringer als etwa bei der Amplitudenmodulation.

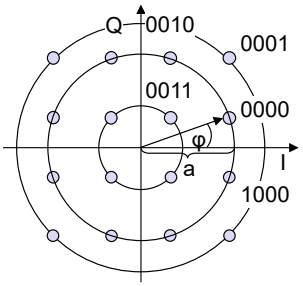
Hier gezeigt ist eine differentielle Phasenmodulation: soll eine 0 codiert werden, wird ein Phasenwechsel um 180° vorgenommen, zur Codierung einer 1 erfolgt kein Phasenwechsel.

Bitte beachten: man kann sowohl analoge als auch digitale Primärsignale auf ein Trägersignal aufmodulieren. Zur besseren Unterscheidung wird in der Praxis für die Modulation digitaler Primärsignale oft der Begriff des „Keyings“ verwendet: Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), Phase Shift Kying (PSK).

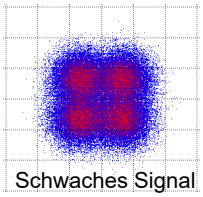
Die vorgestellten Modulationsarten (Amplituden-, Frequenz- und Phasenmodulation) können auch beliebig miteinander kombiniert werden. Dies geschieht beispielsweise bei dem QAM-Verfahren (Quadratur-Amplituden-Modulation).

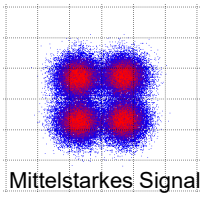
Kombination der Modulationsverfahren möglich

- **Beispiel Quadraturamplitudenmodulation (QAM):**
  - Kombiniert Amplituden- und Phasenmodulation
  - Möglichkeit, *n* Symbole in einen Taktschritt zu kodieren
  - 2^n diskrete Signalwerte
  - Bitfehlerrate wächst mit *n*, aber geringer als vergleichbare (reine) Phasenmodulationsverfahren
  - Beispiel: 16-QAM (Signalwerte repräsentieren 4 Bit)
    - Bis hin zu 32768-QAM



Beispiel: DVB-S 4-QAM-Signal


Schwaches Signal


Mittelstarkes Signal

  - Verwendet z.B. in DSL, WLAN, LTE, DVB-S

COM SYS | RWTH AACHEN UNIVERSITY

II-37

Wie bei mehrwertigen digitalen Signalen können auch auf analogen Medien mehrere Bit pro Signal kodiert werden. Dies kann durch Verwendung mehrerer Amplitudenlevel, Trägerfrequenzen oder Phasenlagen vorgenommen werden (was z.B. in den ersten Modems gemacht wurde, um die Bitrate zu erhöhen).

Durchgesetzt hat sich allerdings QAM, da es sich als robuster erwiesen hat, mit einer gleichzeitigen Modifikation zweier Signalparameter zu arbeiten. In heutigen Kommunikationsnetzen wird dieses Verfahren sehr oft verwendet, z.B. bei DSL, WLAN, LTE, …

Auf der Folie dargestellt ist zur Veranschaulichung 16-QAM: 16 Zustände erlauben die gleichzeitige Übertragung von 4 Bit in einem Signal, z.B. (vereinfacht dargestellt, die tatsächliche physikalische Realisierung ist komplexer):

Symbole 0011 und 0001 haben gleiche Phase φ, aber unterschiedliche Amplituden a.

0000 und 1000 haben unterschiedliche Phasen, aber gleiche Amplitude.

Üblich sind heute aber höhere Schemata, z.B. 256-QAM.

Zu beachten ist allerdings: je mehr Signalwerte erlaubt werden, desto ähnlicher werden benachbarte Werte einander. Wenn Störeinflüsse im Medium auftreten, welche die Signalwerte verfälschen, dann sind beim Empfänger die einzelnen Signalwerte umso schwerer voneinander zu unterscheiden, je mehr Signalwerte verwendet wurden. Deshalb implementieren unsere Kommunikationssysteme üblicherweise adaptive Verfahren, die die Bitrate reduzieren, wenn zu starke Störungen vorliegen, indem eine QAM-Variante mit weniger Zuständen verwendet wird.

Bei der Variante 4-QAM (4 Zustände, Codierung von 2 Bit pro Signal) liegen alle Zustände auf einem Kreis gleicher Amplitude. Diese Variante hat den speziellen Namen QPSK (Quaternary Phase Shift Keying), da die Unterscheidung der Signalformen nur über die Phase erfolgt.

Ausbreitungsgeschwindigkeit von Daten

- **Max. Lichtgeschwindigkeit ($c = 3 \cdot 10^8$ m/s) im Vakuum**
  - *Ausbreitungsgeschwindigkeit* auf Leitungen < Lichtgeschwindigkeit
    - z.B. näherungsweise $0{,}6 \cdot c$ bei Kupferkabel
    - Führt zu *Latenz* (Ausbreitungsverzögerung, Signallaufzeit)
  - *Bandbreiten-Verzögerungs-Produkt* (auch: Speicher-/Pfadkapazität) des Mediums durch begrenzte Ausbreitungsgeschwindigkeit
  - Beispiel: Übertragung RWTH → UPM:
    - Strecke: 1.400 km;
      Latenz: 1.400 km / $2 \cdot 10^8$m/s → 7 ms
    - Bei einer Übertragungsrate von 64 kBit/s: 448 Bit Speicherkapazität
    - Bei einer Übertragungsrate von 2 MBit/s: 14.000 Bit Speicherkapazität

Ein wichtiger, nicht zu unterschätzender Faktor, der sich aus der Schrittgeschwindigkeit ergibt, ist die Tatsache, dass jedes Medium eine gewisse Speicherkapazität besitzt, die beispielsweise bei Zugriffsprotokollen berücksichtigt werden muss (im nächsten Kapitel behandelt). In der Literatur findet sich dafür häufig der Begriff des „Bandbreiten-Verzögerungsprodukts“. (ACHTUNG: mit Bandbreite ist hier die Datenrate gemeint!)

Diese Speicherkapazität, welche durch die begrenzte Ausbreitungsgeschwindigkeit entsteht, kann durchaus gravierende Dimensionen annehmen. Im gezeigten Beispiel stehen über größere Distanzen meist Leitungen mit einer hohen Übertragungsrate zur Verfügung, über welche die Daten aus mehreren lokalen Netzen geleitet werden. Als Beispiel: Ein einziger Kanal mit 622 Mbit/s erreicht bei einer Länge von 5.000 km dadurch eine Speicherkapazität von 25ms·622Mbit/s = 15,55 MBit! Das bedeutet, dass der Sender bereits 15,55 Mbit gesendet hat, bevor das erste Bit beim Empfänger ankommt.

Die Verzögerung bei der Zustellung von Daten über eine Leitung wird auch *Latenz* genannt. Eine anschauliche Darstellung des Zusammenhangs zwischen der Latenz und der Datenrate in Abhängigkeit von der Paketgröße der übertragenen Daten findet sich hier:

https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/transmission-vs-propogation-delay/transmission-propagation-delay-ch1/index.html

Bitte beachten: in der Rechnung auf der Folie wurde angenommen, dass eine direkte Verbindung zwischen Aachen und Madrid existiert. In der Realität hat man eine Vielzahl von Teilstrecken, über die die Daten übertragen werden, die durch Router oder Switches gekoppelt sind. Innerhalb dieser Komponenten tritt noch weitere Latenz auf: eingehende Datenpakete werden zunächst gepuffert, bis sie verarbeitet und weitergeleitet werden können. Wie lange sie gepuffert warden, hängt davon ab, wie viele weitere Pakete der Router/Switch bereits in seiner Warteschlange hat, die vorher weitergeleitet werden müssen. Wir haben also eine variable Pufferungslatenz (Queuing Latency). Ist ein Paket an der Reihe,

wird es verarbeitet – auch dies erzeugt eine (konstante) Latenz. In der Realität wäre daher das Bandbreiten-Verzögerungs-Produkt noch wesentlich höher.

Zusammenfassung: wichtige Begriffe

- **Wesentliche Parameter der Kommunikation**
  - *Schritt*
    - Zeitintervall, in dem ein Signal auf das Medium gegeben wird
  - *Schrittgeschwindigkeit*
    - Anzahl der Signale pro Sekunde, die auf das Medium gegeben werden können
  - *Datenrate* (Übertragungsrate, Übertragungsgeschwindigkeit, Kapazität)
    - Anzahl der Bit pro Sekunde, die auf das Medium gegeben werden können
    - *Sendedauer einer Bitfolge*: Dauer, alle Bits auf das Medium zu geben
  - *Ausbreitungsgeschwindigkeit* (Signalgeschwindigkeit)
    - Geschwindigkeit, mit der ein Signal sich auf dem Medium ausbreitet
  - *Latenz* (Ausbreitungsverzögerung, Signallaufzeit)
    - Zeitverzögerung zwischen Aufsetzen eines Signals auf das Medium durch den Sender und Empfang des Signals durch den Empfänger

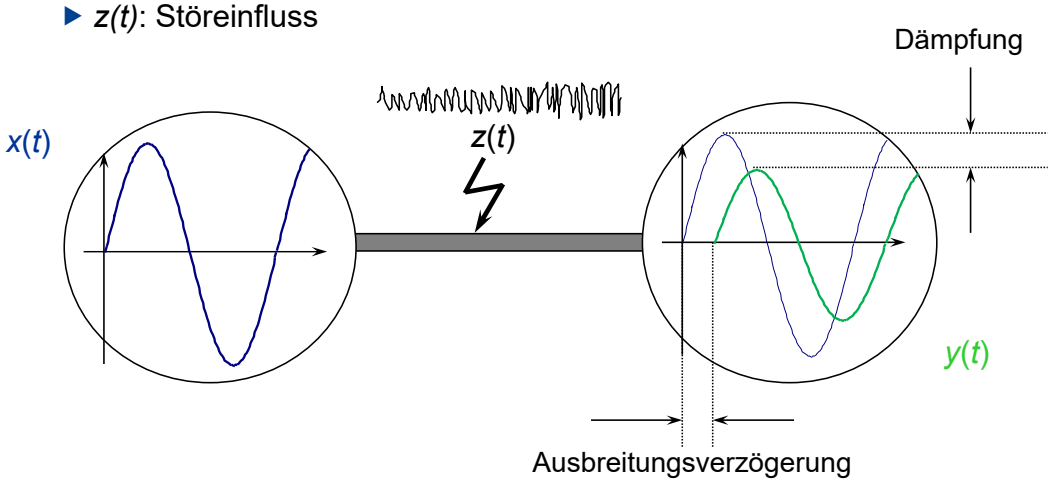
Zusammenfassend: aus der Schrittgeschwindigkeit und der gewählten Codierung ergibt sich die Datenrate. Aus der Ausbreitungsgeschwindigkeit und der Länge des Mediums (Entfernung zwischen Sender und Empfänger) ergibt sich die Latenz.

Datenrate und Latenz zusammen bestimmen zwei wesentliche Faktoren:

- Wird nicht ein einzelnes Bit betrachtet, sondern eine Bitfolge, die übertragen werden soll, setzt sich die gesamte Zeit, bis der Empfänger die Datenfolge erhalten hat, zusammen aus der gesamten Sendedauer für diese Bitfolge (wie lange dauert es, bei gegebener Datenrate alle Bits aufs Medium zu geben) und der Latenz (wie lange dauert es, bis das letzte Bit nach Aufsetzen aufs Medium beim Empfänger ankommt).
- Bandbreiten-Verzögerungs-Produkt: eine gewisse Menge an Daten befindet sich zu jeder Zeit im Transfer (auf dem Medium); wie groß diese Datenmenge ist, hängt vom Verhältnis der Datenrate und der Latenz ab (wie viele Bits können auf das Medium gegeben werden, bevor das erste Bit den Empfänger erreicht).

Signalübertragung über ein Medium

- ***y(t) = F(x(t), z(t))***, mit …
  - *x(t)*: Eingabesignal
  - *y(t)*: Ausgabesignal
  - *z(t)*: Störeinfluss



Nun werden Qualitätsaspekte von Medien betrachtet, welche interessanterweise unabhängig von den diversen Medientypen behandelt werden können, da bei allen Medien die gleichen Einflüsse die Qualität eines Signals beeinflussen können.

Auf Seiten des Empfängers wird ein Ausgabesignal $y(t)$ abgetastet und in das Primärsignal rücktransformiert. Dieses Ausgabesignal hängt vom Eingabesignal $x(t)$ sowie von Störeinflüssen $z(t)$ während der Übertragung auf dem Medium ab: $y(t) = F(x(t), z(t))$.

Zwei Einflüsse, die das Eingabesignal $x(t)$ während der Übertragung modifizieren, sind die Ausbreitungsverzögerung und die Dämpfung. Die Ausbreitungsverzögerung ist unkritisch, wenn sie konstant bleibt. Bei Verwendung von drahtloser Kommunikation z.B. können allerdings sowohl Sender als auch Empfänger mobil sein, so dass sich die Entfernung zwischen ihnen und somit auch die Signalverzögerung kontinuierlich ändert. Dies führt zu Verzerrungen im Signal, die der Empfänger erkennen und herausfiltern muss.

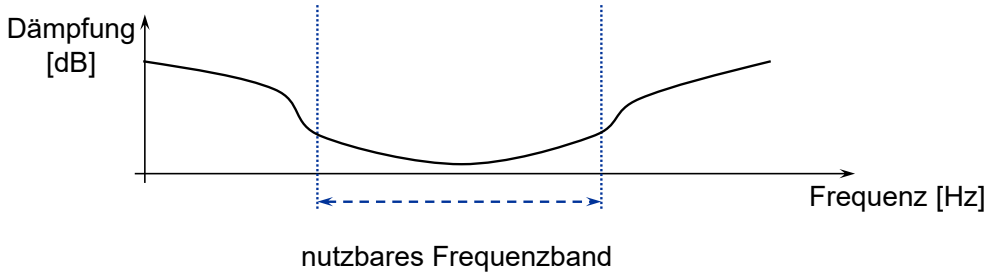
Die Dämpfung ist eine Eigenschaft jedes Mediums. Ein elektromagnetisches Signal wird während der Ausbreitung auf dem Medium abgeschwächt (die Amplitude verringert sich). Je größer die zurückgelegte Strecke eines Signals über das Medium ist, desto stärker wird ein Signal gedämpft. Wird die Amplitude zu gering, kann der Empfänger die Signale nicht mehr eindeutig erkennen. Durch die Dämpfung wird daher die maximal mögliche Distanz bestimmt, über die kommuniziert werden kann. So ist z.B. ein wesentlicher Vorteil eines optischen Mediums (Glasfaserkabel), dass dieses eine um den Faktor 10^2 bis 10^3 geringere Dämpfung aufweist als ein Kupferkabel.

Ausbreitungsverzögerung und Dämpfung führen dazu, dass der Empfänger ein abgeschwächtes und verzerrtes Signal erhält.

$z(t)$ ist eine abstrakte Größe, die sich aus mehreren Störeinflüssen zusammensetzen kann. Generell beschreibt sie, dass fremdinduzierte Signale (Rauschen) sich mit dem eigentlichen Eingabesignal überlagern und es dadurch weiter verzerren. Störeinflüsse sind medienabhängig.

Bandbreite und Dämpfung

- *Bandbreite*
  - Für Übertragung aufgrund der Dämpfung nutzbares Frequenzband



  - [dB] = 10 · $\log_{10}(S_1/S_2)$
  - 10 dB = 10, 20dB = 100, 30 dB = 1000, -20 dB = 0,01, 0dB = 1

Die Dämpfung, die das Signal während des Transports auf dem jeweiligen Medium erfährt, bestimmt letztendlich auch die Bandbreite dieses Mediums. Der Zusammenhang zwischen Dämpfung und Bandbreite ergibt sich aufgrund folgender Eigenschaft der Dämpfung:

- Die Dämpfung ist frequenzabhängig. Es lässt sich ein Frequenzspektrum angeben, außerhalb dessen die Dämpfung keine Auswertung des Ausgabesignals mehr zulässt. Dies ist abhängig von der Länge des verwendeten Mediums – je größer die zu überbrückende Distanz wird, desto größere Teile des Frequenzspektrums werden zu stark gedämpft, so dass die nutzbare Bandbreite abnimmt.

Die Dämpfung ist eine zentrale Kenngröße eines Mediums. Aus der Bandbreite lässt sich auf die maximale Schrittgeschwindigkeit (d.h. Anzahl der Pegelwechsel bei Leitungscodes) schließen, mit der ein Kanal betrieben werden kann.

Daher gibt es auch Längenbeschränkungen für alle Medien, um das für eine Zieldatenrate notwendige Frequenzspektrum trotz Dämpfung noch verwenden zu können.

Ist es nötig, größere Entfernungen zu überbrücken, können sogenannte Repeater eingesetzt werden, die in regelmäßigen Abständen (bevor das Signal zu stark gedämpft wurde) im Kabel (bzw. der Funkstrecke) zwischengeschaltet werden. Ein Repeater tastet die empfangenen Signale ab und rekonstruiert den Bitstrom. Dieser Bitstrom wird auf dem anderen Ausgang nach dem verwendeten Leitungscode/Modulationsverfahren neu in Signale mit voller Stärke gewandelt. Damit kann die Reichweite eines Netzes trotz Dämpfung prinzipiell ins Unendliche gesteigert werden.

Nötig ist dies z.B. auch bei den Unterseekabeln – alle 50 km wird ein Repeater benötigt.

Hinweis: Bei der physikalischen Einheit Dezibel (dB) handelt es sich um eine logarithmische Größe, die das Verhältnis zwischen zwei Werten ausdrückt. Hier ist S_1 das Eingabesignal, S_2 das Ausgabesignal. Bitte beachten: es gibt auch negative Dezibelwerte; in diesem Fall wäre der Wert negativ, wenn $S_2 > S_1$ ist, also eine Verstärkung stattgefunden hätte.

Störeinflüsse

- **Die bisher behandelten Kenngrößen stellen medienbedingte Abweichungen dar**
  - Zusätzlich können *Störungen* auftreten, die das Signal beeinflussen
- **Beispiel für Störeinflüsse:**
  - Weißes Rauschen (Grundstöranteil)
  - Echobildung (durch zeitverschobenes Eingabesignal)
  - Nebensprechen (gegenseitige Medienbeeinflussung)
  - Brummsignale (niederfrequente Störsignale)
  - Störimpulse (kurzzeitig mit hoher Amplitude)
  - Weitere/ähnliche Störfaktoren bei Funkübertragung:
    - Beugung, Brechung, Reflektion (Mehrwegeausbreitung), etc.
    - Siehe Vorlesung „Mobile Internet Technology“

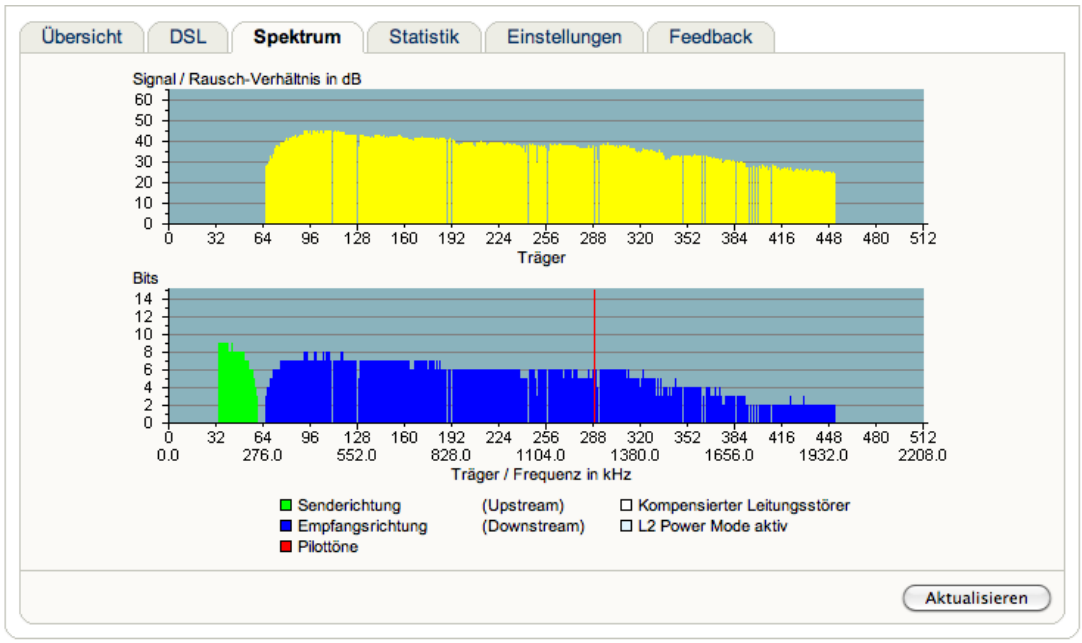
Störeinflüsse sind folgendermaßen charakterisiert:

Die Störeinflüsse (im Modell unter $z(t)$ zusammengefasst) sind additive Komponenten zu den medienbedingten Abweichungen, die in ihrem genauen Verlauf nicht vorab bestimmbar sind.

Bei den oben angegebenen Arten von Störeinflüssen lassen sich solche feststellen, die im Grundsatz ständig vorhanden sind, wie das weiße Rauschen (Grundstöranteil) oder auch die Brummsignale (niederfrequente Störungen - Beispiel: 50 Hz, die von Stromversorgungsleitungen stammen). Andere Störeinflüsse können von anderen benachbart verlaufenden Medien erzeugt werden, wie etwa das Nebensprechen (Störungen von nebenliegenden Leitungen). Man kann aber auch quasi sich selbst stören, nämlich dann, wenn Echo-Bildungen von den eigenen Signalen entstehen. Die letzte Art, die Störimpulse, fasst alle weiteren Störungen zusammen, deren Ursache häufig gar nicht genau festgestellt werden kann (z.B. ein Umschaltvorgang im Medium, der kurzfristig zu Störimpulsen führen kann).

Störeinflüsse können sowohl kurzfristig als auch langfristig auftreten. Kurzfristige Störungen können beispielsweise durch transiente, stochastische Prozesse oder Impulsstörungen auftreten. Lange anhaltende Störungen werden beispielsweise durch Bündelfehler (Echobildung, Nebensprechen, (thermisches) Rauschen, Anschalten von induktiven Lasten (z.B. Motor)), aber auch durch Synchronisationsfehler hervorgerufen.

Signal/Rausch-Verhältnis bei DSL

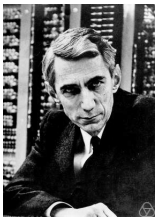


Beispiel DSL: dargestellt ist hier ein Beispiel der Signalqualität in Abhängigkeit der Frequenz bei einer DSL-Leitung. In Relation dazu sieht man im unteren Teil der Abbildung die erzielte Datenrate auf der Leitung – bei höherem Signal/Rausch-Abstand hat man die Möglichkeit, mehr Signalstufen zu verwenden und erzielt damit höhere Datenraten.

Nyquist- und Shannon-Theorem

- **H. Nyquist (1924):**
  - *Störungsfreier Kanal* mit eingeschränkter Bandbreite:
    - Maximale Datenrate [Bit/s] = **2 · B · ld(n)**
      - *B*: Bandbreite des Kanals
      - *n*: Anzahl diskreter Signalstufen

- **C. Shannon (1948)**
  - Bandbreitenbeschränkter Kanal mit *zufälligem Rauschen*:
    - Maximale Datenrate [Bit/s] = **B · ld(1 + S/N)**
      - *S*: Stärke des empfangenen Signals
      - *N*: Stärke des Störsignals
      - S/N: Signal-Rausch-Abstand (*Signal to Noise Ratio, SNR*)



COM SYS | RWTH AACHEN UNIVERSITY

Bildquelle: Wikipedia

II-44

Zwei wichtige Theoreme für die Datenrate, die theoretisch auf einem gegebenen Kanal erreicht werden kann, wurden von Nyquist und Shannon aufgestellt.

Nyquist geht von einem Kanal ohne Störungen aus. Dieser Kanal erlaubt aufgrund von Dämpfung nur die Verwendung einer bestimmten Bandbreite B. Das Nyquist-Theorem erlaubt die Berechnung der maximalen Datenrate, die auf diesem Kanal unter Verwendung einer Codierung mit n Signalstufen erreicht werden kann.

Shannon vernachlässigt die verwendete Codierung, berücksichtigt dafür aber Störsignale, die durch externe Einflüsse auf den Kanal einwirken. In Abhängigkeit von der Stärke der Störsignale kann nur eine bestimmte Datenrate erzielt werden. Der Faktor S/N ist hierbei der Signal-Rausch-Abstand – das Verhältnis von Signalstärke zu Stärke des Rauschens. (Dieser wird in der Praxis auch oft in dezibel angegeben.)

Für die Berechnung der in der Praxis erzielbaren Datenrate sind beide Faktoren von Bedeutung – bei vorgegebener Bandbreite ist stets das *Minimum* der Ergebnisse der beiden Theoreme maßgeblich, um die maximal erzielbare Bandbreite abzuschätzen. Bitte beachten Sie, dass es sich tatsächlich nur um eine Abschätzung handelt, da die beiden Theoreme theoretisch erzielbare Datenraten ermitteln. In der Praxis können weitere Effekte dazu führen, dass noch geringere Datenraten erreicht werden. Diese ignorieren wir hier.

Beispiel: Nyquist- und Shannon-Theorem

- **Gegebener Kanal:**
  - Bandbreite von 3.000 Hz
  - Zweiwertiges Signal (Z.B. NRZ-Codierung)
  - Signal-Rauschabstand von 30 dB

- **Maximale Datenrate R_{max}:**
  - *Nyquist:*
    - $R_{ny} = 2 \cdot 3000$ [Hz] $\cdot$ ld 2 [Bit] = 6.000 Bit/s
  - *Shannon:*
    - 30 dB = $10 \cdot \log_{10}(S/N) \rightarrow S/N = 10^3 = 1.000$
    - $R_{Sh} = 3000 \cdot \text{ld}(1 + 1000) \approx 30.000$ Bit/s
  - $R_{max} = \min\{R_{ny}, R_{Sh}\} = 6.000$ Bit/s

Umrechnung von Dezibel in absolute Werte für S/N:

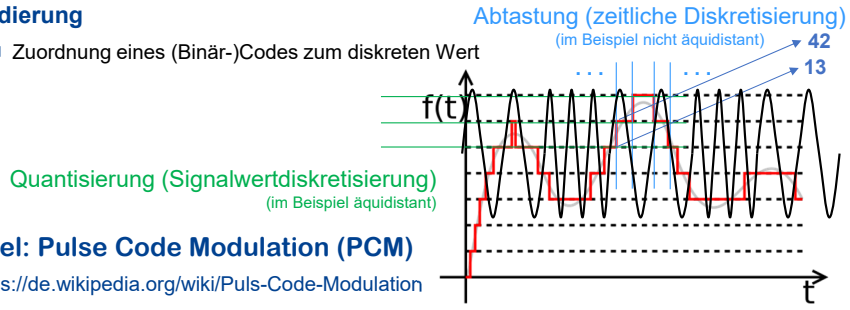
S/N = 1000 -> 30dB

= 100 -> 20dB

= 10 -> 10dB

Wandlung kontinuierlicher in diskrete Werte

- **Konvertierung analoger in digitale Werte und zurück**
  - Übertragung analoger Signale in digitaler Darstellung
    - z.B. bei digitalen Telefonkanälen (ISDN, GSM, …)
    - Verwendung von (Binär-)Codes zur Codierung der digitalen Werte
  - **Abtastung** des analogen Signals in regelmäßigen Intervallen (Zeitdiskretisierung)
  - **Quantisierung** der abgetasteten Werte (Signalwertdiskretisierung)
    - Definition von Quantisierungsintervallen
  - **Codierung**
    - Zuordnung eines (Binär-)Codes zum diskreten Wert



- **Beispiel: Pulse Code Modulation (PCM)**
  - https://de.wikipedia.org/wiki/Puls-Code-Modulation

COM SYS | RWTH AACHEN UNIVERSITY

Interessantes Video zum Thema PCM und Quantisierung:
https://xiph.org/video/vid2.shtml

II-46

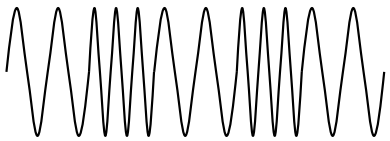
Bisher ausgespart wurde die Übertragung analoger Signale in digitaler Darstellung, da der Fokus der Vorlesung auf der Übertragung digitaler Signale liegt. Zum Abschluss soll aber auch dieser Fall kurz betrachtet werden.

Zur Übertragung von analogen Signalen über digitale Übertragungssysteme muss man die Signale vorher *digitalisieren*, d.h. in digitale Daten umsetzen. Dies erreicht man durch eine zeitdiskrete *Abtastung* des analogen Signals und eine anschließende Umsetzung des abgetasteten, kontinuierlichen Wertes in einen diskreten Wert. Bei dieser Umsetzung teilt man die analoge Werteskala in endlich viele Bereiche ein. Jedem der Bereiche (*Quantisierungsintervalle*) ordnet man einen digitalen Wert zu (*Codierung*). Da den analogen Werten eines Quantisierungsintervalls mit der Breite a nur ein digitaler Wert zugeordnet wird, erhält man einen Quantisierungsfehler von max. $\pm a/2$. Diesen Wert kann man bei genügend Intervallen in Kauf nehmen, da dieser Fehler der maximale Fehler ist, den man auch nach der Übertragung hat. Bei einer analogen Übertragung würden noch Störsignale hinzukommen, die meist noch durch analoge Komponenten verstärkt werden, und einen größeren Fehler als $a/2$ verursachen.

Ein Standardverfahren ist die sogenannte Pulse Code Modulation.

Abtasttheorem

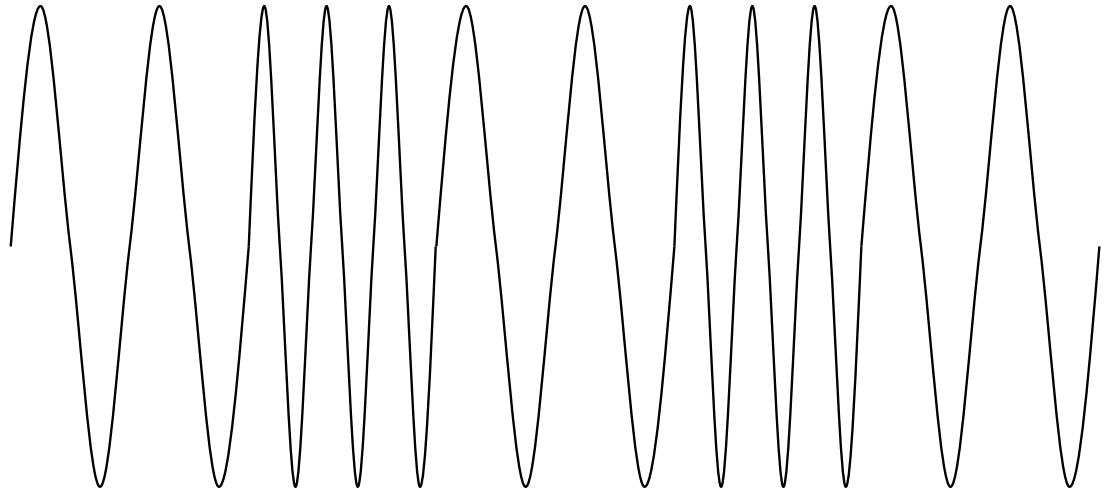
- **Abtasttheorem von Nyquist/Shannon u.A.**
  - Zur fehlerfreien Rekonstruktion des Signalverlaufs eines abgetasteten Analogsignals ist eine *Mindestabtasthäufigkeit* (Abtastfrequenz f_A) erforderlich
  - Eine Signalfunktion, die nur Frequenzen im Frequenzband *B* (bandbegrenztes Signal) enthält, wobei *B* gleichzeitig die höchste Signalfrequenz ist, wird durch ihre diskreten Amplitudenwerte im Zeitabstand $\mathbf{t_0 < 1/(2B)}$ vollständig bestimmt
  - Andere Formulierung:
    - Die Abtastfrequenz f_A muss mehr als doppelt so hoch sein wie die höchste im abzutastenden Signal vorkommende Frequenz f_{Grenz}:
      - $\mathbf{f_A > 2 * f_{Grenz}}$



Für die Abtastung eines Signals ist es erforderlich, dass die Abtastrate mehr als doppelt so hoch ist wie die maximale relevante Frequenz, die im Signal vorkommen kann. Dies wird durch das Abtasttheorem festgelegt. (Dieses Ergebnis wurde von mehreren Forschergruppen unabhängig voneinander erzielt, so dass das Theorem auch unter verschiedensten Namen zu finden ist. Generell ist es am einfachsten, nur schlicht „Abtasttheorem“ zu sagen.)

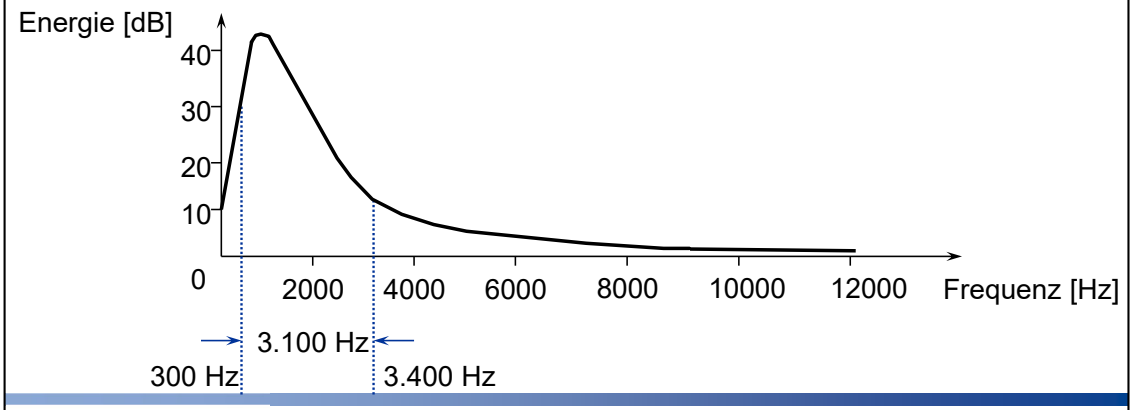
Abtasttheorem

- Die Abtastfrequenz f_A muss mehr als doppelt so hoch sein wie die höchste im abzutastenden Signal vorkommende Frequenz f_{Grenz}:
  - $\mathbf{f_A > 2 * f_{Grenz}}$



Frequenzspektrum eines Signals

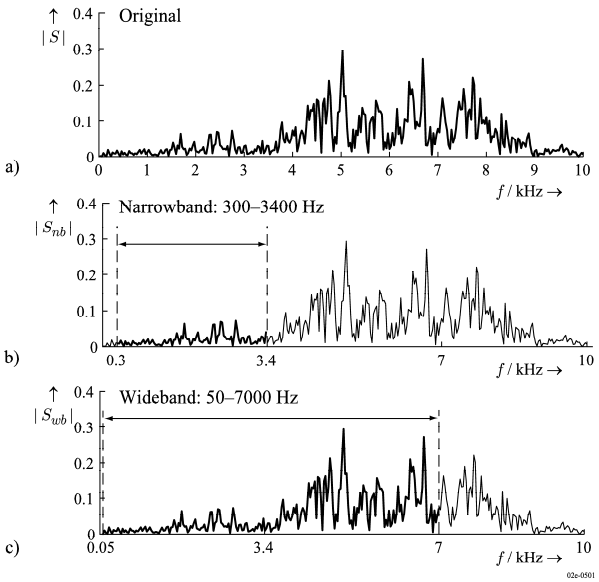
- **Bandbegrenztes Signal:**
  - Signale können ein natürlich begrenztes Frequenzspektrum umfassen oder durch technische Mittel auf einen Ausschnitt ihres Spektrums (Bandbreite) begrenzt werden
    - Beispiel: ITU-Standardtelefonkanal – Kontinuierliches Frequenzspektrum der menschlichen Stimme begrenzt auf 300 – 3.400 Hz



Wenden wir das Abtasttheorem oder generell PCM nun auf ein Beispiel an: die Übertragung von Sprache über ein Telefonnetz. Das ursprüngliche Telefonnetz war analog, und hier wurde die Designentscheidung getroffen, die Bandbreite der menschlichen Sprache künstlich zu beschränken, so dass nur die wichtigsten Signalkomponenten übertragen werden. Diese Begrenzung wurde genormt auf den Bereich von 300 – 3.400 Hz.

Bei der Einführung digitaler Telefonie war die Bedingung, dass die Sprachqualität des analogen Netzes beibehalten werden sollte. Daher wurde 3.400 Hz als die obere zu berücksichtigende Grenzfrequenz festgelegt.

Comparison: Telephone Speech and Wideband Speech



Original

Today's telephone speech

Future wideband telephone speech ("HD-Voice")

Quelle: Vorlesung Digital Speech Transmission | Prof. Dr.-Ing. Peter Jax

Beispiel Bandbreitenbegrenzung bei Sprache.

b) ist der Normale“ Telefonkanal 300-3400Hz (z.B. Codec G.711)

c) ist „High Definition Sound Performance (HDSP)“ bzw. HD-Voice (Codec G.722)

Pulse Code Modulation (PCM) – Beispiel

- **Beispiel digitale Sprachübertragung:**
  - Grenzfrequenz: 3.400 Hz
  - Abtastfrequenz: 8.000 Hz (> 6.800 Hz)
    d.h. alle 125 µs wird abgetastet
  - Kodierung der Signalwerte: 8 Bit (entspricht 2^8 = 256 Intervallen)
  - Datenrate: 8.000 Hz · 8 Bit = 64.000 Bit/s = 64 kBit/s

Die Digitalisierung der Sprache erfolgt nun mittels PCM. Die obere Grenzfrequenz wurde auf 3.400 Hz festgelegt. Nach Abtasttheorem ergibt sich daraus eine (theoretische) Abtastrate von mindestens 6.800 Hz. In der Praxis wird ein Faktor von mindestens 2,2 verwendet, da unsere Hardware nicht perfekt ist und die Verwendung von Faktor 2 Störsignale erzeugen würde. (Die Details sind hier nicht relevant.)

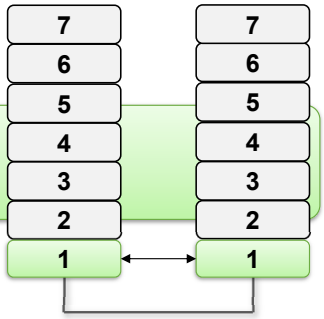
Daher sollte man eine Abtastrate von mindestens 7.480 Hz verwenden. Da Informatiker gerne in Bytes denken und 8 somit eine Schöne Zahl ist, wurde auf 8.000 Hz aufgerundet.

8.000 Mal die Sekunde wird somit die Amplitude des analogen Eingabesignals gemessen und in einen digitalen Wert umgewandelt. Hier hat man für die Telefonie festgelegt, dass die Approximation des Originalsignals gut genug ist, wenn 256 verschiedene Amplitudenwerte möglich sind. Jeder abgetastete Wert wird daher mit 8 Bit codiert.

Dadurch erzeugt man einen Datenstrom mit 64.000 Bit/s. Dieser Wert hat sich bis heute als nötige Datenrate für die Übertragung von Sprache gehalten.

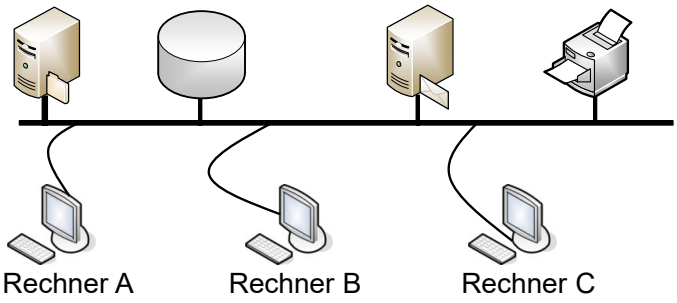
(Schaut man genauer in Standards wie GSM – LTE hinein, stellt man fest, dass deutlich geringere Datenraten verwendet werden. Dies liegt daran, dass eine Komprimierung der Sprachdaten erfolgt.

- **Grundlagen**
  - Übertragungsmedien
  - Signale und Bandbreite
- **Übertragung von Signalen**
  - Umformung, Basisband, Modulation
  - Übertragungsparameter, Störeinflüsse
  - Leitungscodes und Modulationsverfahren
  - PCM
- **Kanalnutzung**
  - Multiplexing



Dedizierter und mehrfach genutzter Kanal

- **Dedizierter Kanal**
  - Nachrichtentechnischer Kanal verbindet genau eine Quelle mit genau einer Senke
  - Betriebsarten: simplex, halbduplex oder duplex
- **Mehrfach genutzter Kanal (*Shared Medium*)**
  - Mehr als zwei Dienstnutzer greifen auf dasselbe Medium zu
  - Beispiel: Lokales Netz (Wifi) im Büro oder zu Hause



Der letzte Aspekt, welcher im Zusammenhang mit dem nachrichtentechnischen Kanal behandelt wird, betrifft die Frage, wie dieser eigentlich genutzt wird. Bisher wurde i.d.R. vom einfachsten und bequemsten Fall ausgegangen, dass der Kanal von genau zwei Dienstnutzern, der Quelle und der Senke, genutzt wird. In diesem Fall sagt man, dass der Kanal diesen beiden Dienstnutzern gewidmet ist; man spricht von einem dedizierten Kanal. Hierbei lässt sich weiter zwischen verschiedenen Betriebsarten unterscheiden:

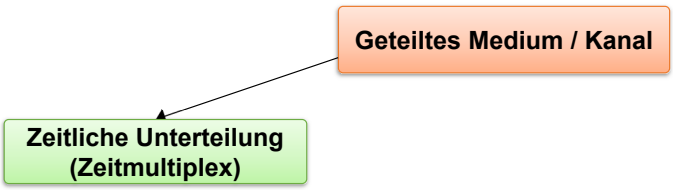
• *simplex*: Senden nur in eine Richtung möglich

• *halbduplex*: Senden nur in eine Richtung zur Zeit erlaubt, aber Wechsel der Senderichtung möglich

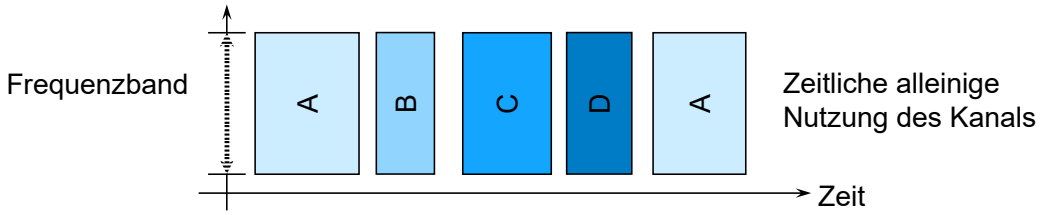
• *duplex*: Senden in beiden Richtungen gleichzeitig möglich

Sehr viel mehr protokolltechnische Überlegungen muss man sich machen, wenn man es mit einem Halbduplex-Kanal zu tun hat, der von mehreren Dienstnutzern gleichzeitig genutzt werden möchte und daher vor der sinnvollen Nutzung als erstes zugeteilt bzw. reserviert werden muss. Bei einem von mehreren Dienstnutzern genutzten Kanal sind aufgrund konkurrierender Sendeversuche zusätzliche Maßnahmen zur Zugriffskontrolle nötig. Andererseits bietet ein solcher Kanal aber auch den Vorteil von Broadcasts, d.h. die Daten werden nur einmal auf das Medium gegeben und gleichzeitig von allen Teilnehmern empfangen, die an dieses Medium angeschlossen sind.

Mehrfachnutzung von Medien: Zeitmultiplex



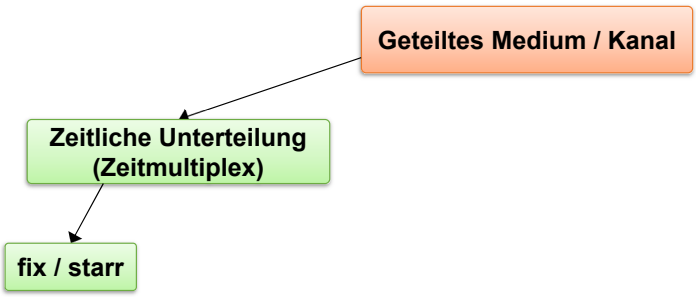
- **Zeitmultiplex (Time Division Multiplexing, TDM)**



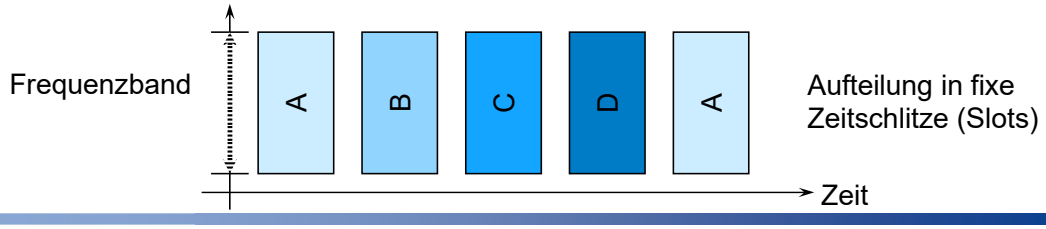
Oft hat man ein Medium mit einer hohen Bandbreite, die von einem Sender alleine nicht ausgenutzt wird. Zusätzlich hat man mehrere Sender, die das Medium benutzen wollen. Also müssen sich diese Sender das Medium teilen. Dieses Teilen nennt man auch *Multiplexen* (Bündelung) des physikalischen Mediums. Man bezeichnet das ganze Medium, welches von allen benutzt wird als Übertragungsweg. Die einzelnen virtuellen Kanäle, die durch Multiplexen entstehen, werden als *Übertragungskanal* bezeichnet. Somit besteht ein Übertragungsweg aus mehreren Übertragungskanälen, die man auf unterschiedliche Art erzeugen kann.

Zeitmultiplex: Bei diesem Multiplexverfahren bekommt ein Sender das Medium nur für eine gewisse Zeitperiode zugeteilt, besitzt aber in dieser Zeitspanne die volle Bandbreite des Mediums. Zwischen den einzelnen Zuteilungen des Mediums liegt meist noch eine kleine Schutzzeit, um die einzelnen Zeitschlitze vor Störungen benachbarter Übertragungen zu schützen.

Mehrfachnutzung von Medien: Zeitmultiplex

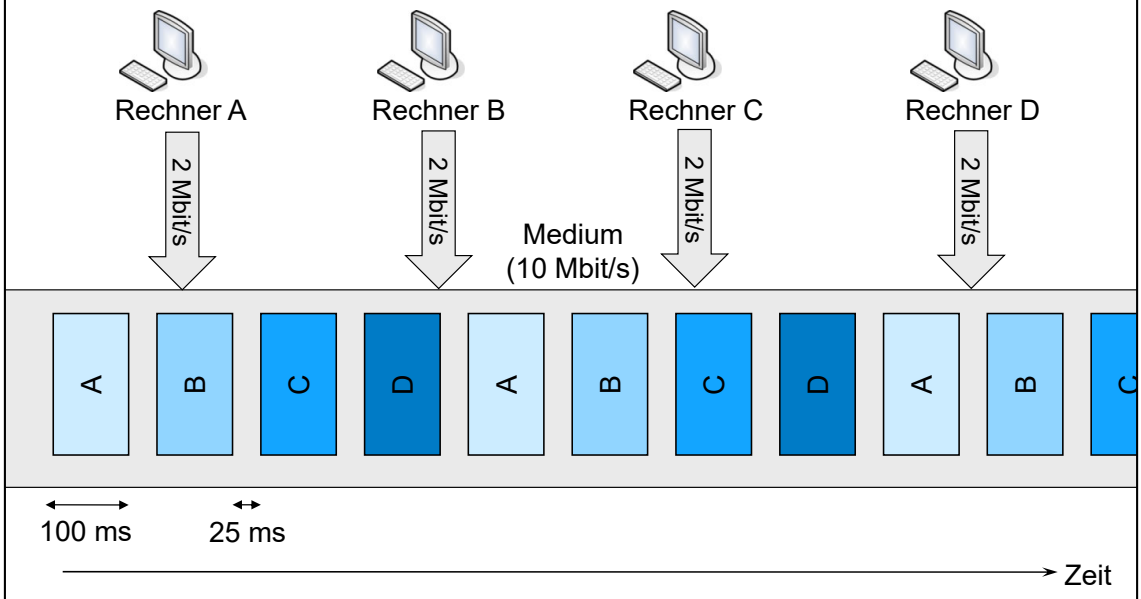


- **Starrer Zeitmultiplex (Time Division Multiplexing, TDM)**

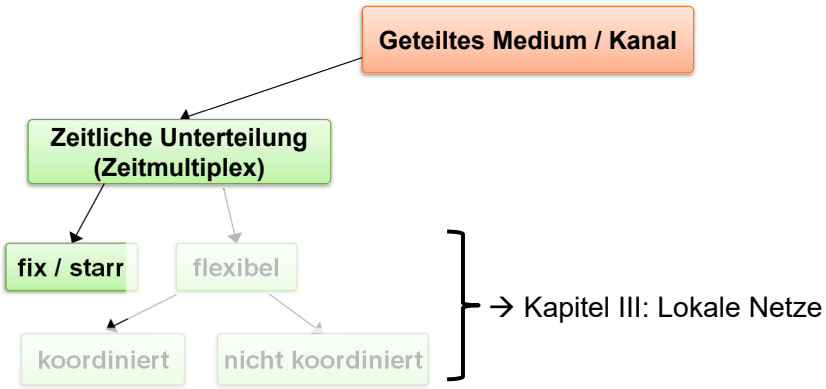


Eine einfache Implementierung ist die Verwendung eines starren Schedules – es gibt einen Scheduler, der entscheidet, welche Slots welcher Station zugeteilt werden – hier bekommt jede Station jeden vierten Slot.

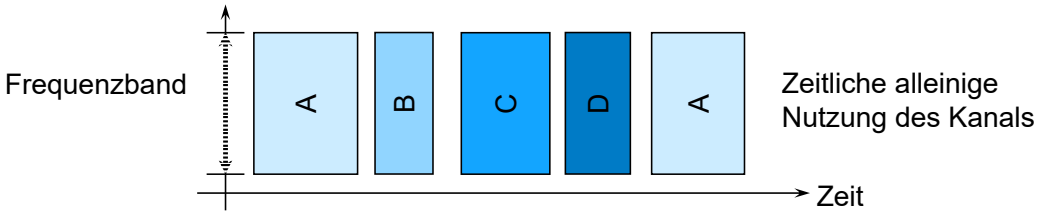
Zeitmultiplex



Mehrfachnutzung von Medien: Zeitmultiplex

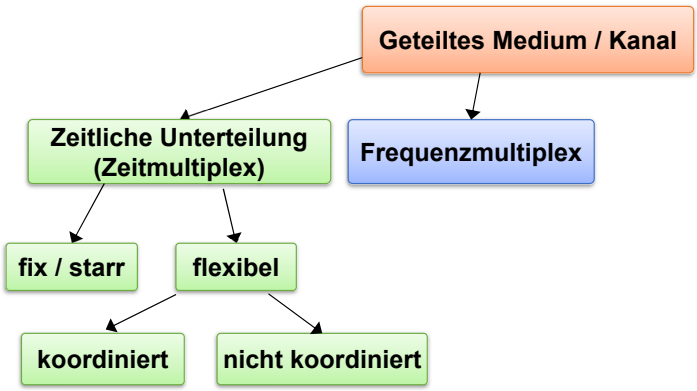


- **Flexibler Zeitmultiplex (Time Division Multiplexing, TDM)**

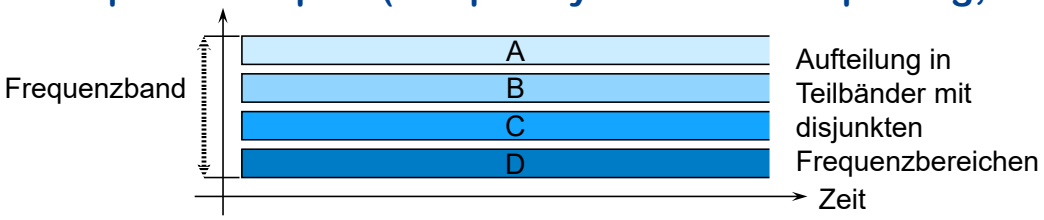


TDMA kann aber auch flexibel implementiert werden. Dies kann koordiniert geschehen (ein Scheduler kann dynamisch Slots zuweisen bzw. alle Stationen einigen sich untereinander auf einen Schedule) oder nicht koordiniert (Stationen treten in Wettbewerb). Diese Varianten werden im nächsten Kapitel anhand der Beispiele Token Ring und CSMA/CD detaillierter betrachtet.

Mehrfachnutzung von Medien durch Multiplexen



- **Frequenzmultiplex (Frequency Division Multiplexing, FDM)**

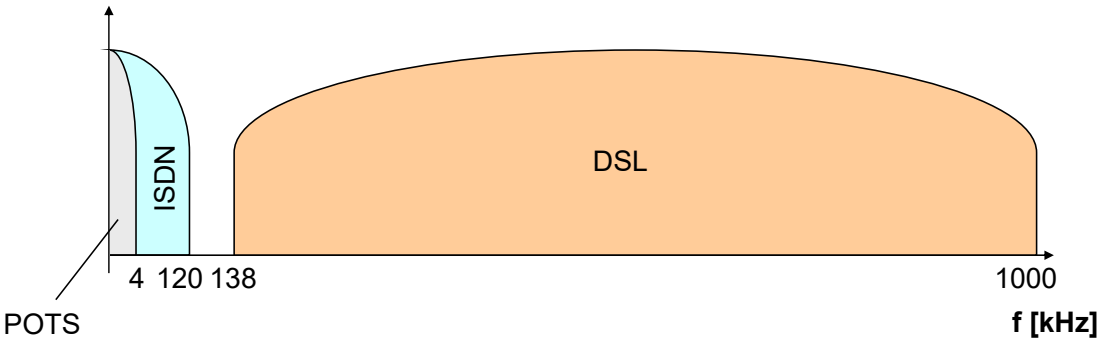


Frequenzmultiplex: Bei einer breitbandigen Übertragung kann man das Frequenzspektrum in mehrere Frequenzbereiche aufspalten und somit Unterkanäle erzeugen. Diese Kanäle können dann unabhängig von den anderen benutzt werden. Sie besitzen natürlich nur einen Teil der ursprünglichen Bandbreite. Meist ist die Summe der Bandbreiten der Unterkanäle sogar kleiner als die gesamte Bandbreite, da man zwischen den Kanälen kleine Bereiche frei lässt, um gegenseitige Störungen der einzelnen Bänder zu verhindern. Dieses Verfahren eignet sich nicht für die Übertragung im Basisband, es ist eine Modulation der Daten auf einen Kanal notwendig.

Wellenlängenmultiplex: Beim Wellenlängenmultiplex handelt es sich physikalisch betrachtet um eine spezielle Variante des Frequenzmultiplex (Frequenz f und Wellenlänge λ besitzen eine feste Beziehung: Lichtgeschwindigkeit ist im Vakuum $c = \lambda * f$, daher sind Frequenz und Wellenlänge begrifflich austauschbar), bei der verschiedene Wellenlängen über eine einzelne Glasfaser übertragen werden – dies ist möglich, da sich die einzelnen Wellenlängen kaum oder nicht beeinflussen. Da die Frequenzen im optischen Bereich liegen, hat sich der Begriff Wellenlängenmultiplex (WDM, Wavelength Division Multiplex) durchgesetzt.

Multiplexing bei DSL

- **Klassische Telefonkanalstruktur**
  - Analoges Telefon: 300 – 3400 Hz
  - ISDN: bis 120 kHz
- **DSL**
  - Nutze Frequenzbereich von 138 kHz bis in den MHz-Bereich



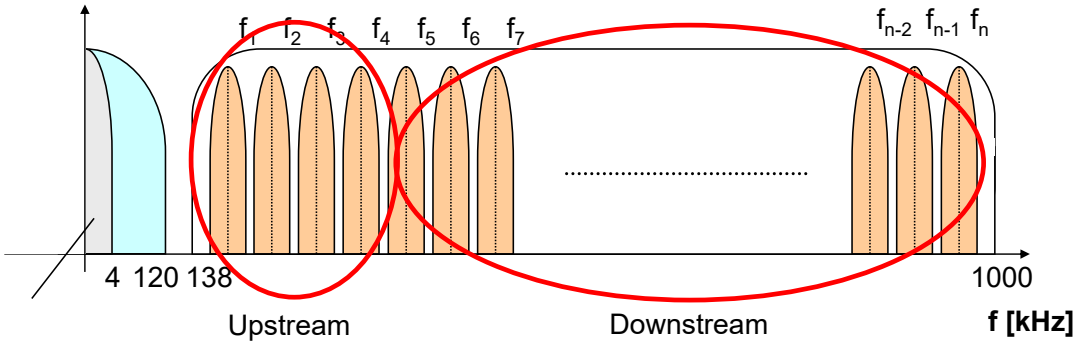
Bereits zu Zeiten des analogen Telefonnetzes konnten mithilfe eines Modems digitale Daten über den analogen Kanal verschickt werden: durch Modulation auf die Bandbreite bis 3.400 Hz wurden Datenraten bis 56 kbit/s möglich.

Ein enormer Fortschritt war ISDN (Integrated Services Digital Network) – es ermöglichte Datenraten bis 144 kbit/s, indem einfach eine größere Bandbreite zugelassen wurde. Dies erforderte den Austausch der Hardware auf beiden Seiten des Telefonkabels, so dass nicht nur Signale bis 3.400 Hz, sondern bis 120 kHz berücksichtigt wurden.

DSL (Digital Subscriber Line) führt das Grundkonzept von ISDN konsequent fort: nutze ein größeres Frequenzband (eine größere Bandbreite) auf der existierenden Telefonverkabelung, um höhere Datenraten erzielen zu können.

DSL: Kanalstruktur

- **DSL**
  - FDM: *erstelle Unterkanäle mit je 4 kHz Bandbreite*
    - Dämpfungs-/Störungsabhängige Verwendung der Kanäle
    - Modulation pro Kanal anpassbar



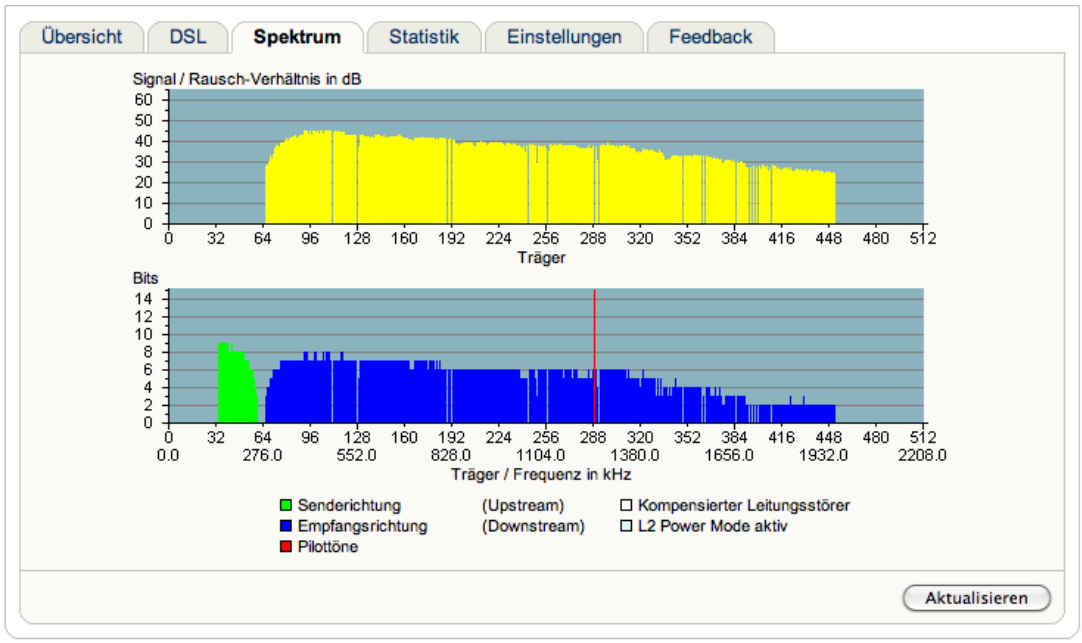
COM SYS | RWTH AACHEN UNIVERSITY

II-60

DSL unterteilt den Frequenzbereich ab 138 kHz in Kanäle zu je 4 kHz Bandbreite. Dadurch wird es möglich, unabhängig voneinander mehrere Unterdatenströme parallel über das Kabel zu übertragen. Dies hat mehrere Vorteile:

- Es wird auf einem einzigen Kabel eine Duplexkommunikation ermöglicht. Noch dazu erlaubt das Prinzip, je nach Bedarf die Anzahl der Unterkanäle auf Up- und Downstream zu verteilen, um unterschiedliche Datenraten in beide Richtungen erzielen zu können.
- Auf jedem Kanal kann QPSK oder QAM mit unterschiedlicher Zahl an Zuständen verwendet werden. Je nach SNR auf einem bestimmten Kanal kann die Modulation angepasst werden, um die maximal mögliche Datenrate ohne zu hohe Bitfehlerrate zu erreichen. Pro Kanal wird bei Systemstart eine Messung durchgeführt, um die erzielbare Modulation ermitteln zu können.
- Durch Dämpfung auf dem Kabel sind Kanäle in Abhängigkeit ihrer Positionierung im Frequenzbereich nur über eine bestimmte Strecke hin nutzbar. Je nach Abstand eines DSL-Anschlusses von der nächsten Vermittlungsstelle kann entschieden werden, wie viele Kanäle verwendet werden können (da hohe Frequenzen stärker gedämpft werden als niedrige). Aus diesem Grund können in Abhängigkeit des Wohnortes (d.h. der Distanz zur nächsten Vermittlungsstelle) auch unterschiedliche Datenraten erreicht werden.

Signal/Rausch-Verhältnis bei DSL

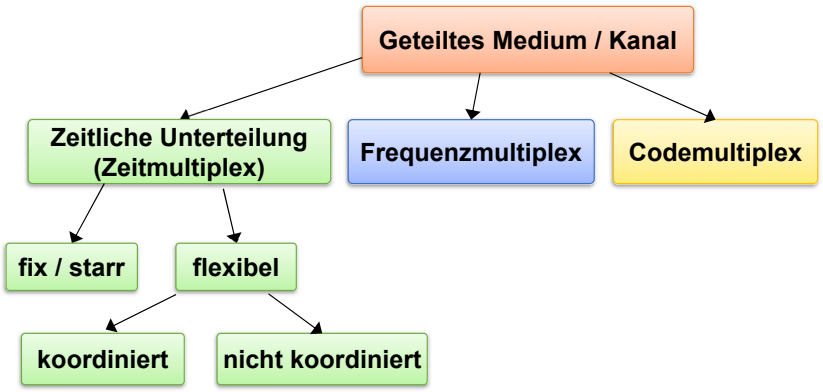


Beispiel DSL: dargestellt ist hier ein Beispiel der Signalqualität in Abhängigkeit der Frequenz bei einer DSL-Leitung. In Relation dazu sieht man im unteren Teil der Abbildung die erzielte Datenrate auf der Leitung – bei höherem Signal/Rausch-Abstand hat man die Möglichkeit, mehr Signalstufen zu verwenden und erzielt damit höhere Datenraten.

Signal/Rausch-Verhältnis bei VDSL

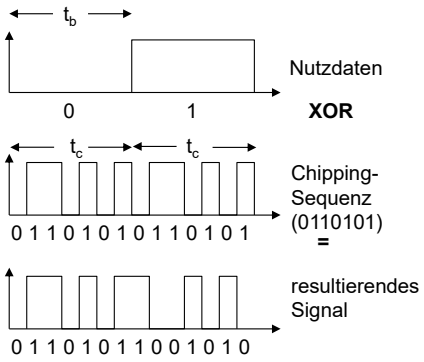
Beispiel DSL: dargestellt ist hier ein Beispiel der Signalqualität in Abhängigkeit der Frequenz bei einer DSL-Leitung. In Relation dazu sieht man im unteren Teil der Abbildung die erzielte Datenrate auf der Leitung – bei höherem Signal/Rausch-Abstand hat man die Möglichkeit, mehr Signalstufen zu verwenden und erzielt damit höhere Datenraten.

Mehrfachnutzung von Medien: Codemultiplex



Mehrfachnutzung von Medien: Codemultiplex

- **Codemultiplex**:
  - Alle Sender nutzen das gleiche Frequenzband und senden gleichzeitig
  - Signal (Bit) wird auf der Senderseite mit einer für den Sender eindeutigen Pseudozufallszahl (Chipping-Sequenz) XOR-verknüpft
  - Empfänger kann mittels bekannter Chipping-Sequenz und einer Korrelationsfunktion das Originalsignal restaurieren
  - Einsatz bspw bei UMTS
    - Variable Codes/Codelängen für mobile Geräte
      - 1,92 Mbps = 4 Bit Code
      - 30 kbps = 256 Bit Code

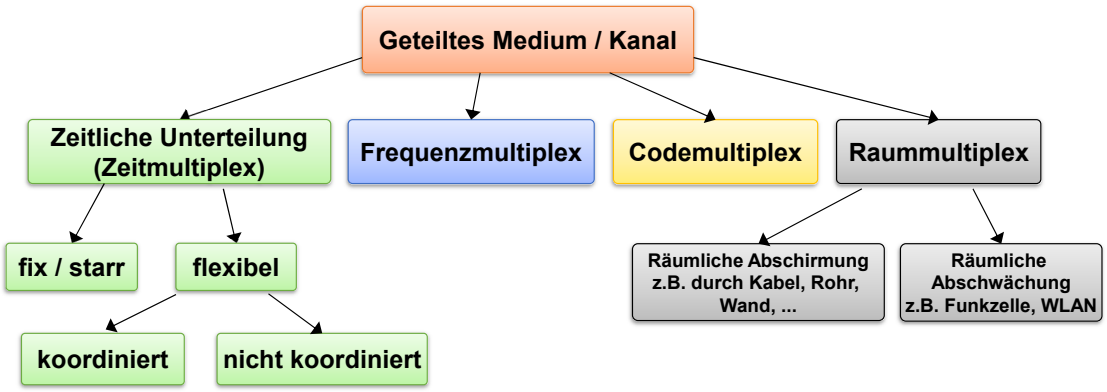


t_b: bit period
t_c: chip period

Codemultiplex: eine weitere Möglichkeit besteht darin, alle Sender gleichzeitig die komplette Bandbreite nutzen zu lassen, indem statt eines einzelnen Bits jeweils eine bestimmte Bitfolge (Chipping-Sequenz, Codefolge) übertragen wird. Die Übertragungen der unterschiedlichen Sender überlagern sich auf dem Medium, aber bei geeigneter Wahl der Chipping-Sequenzen kann ein Empfänger eine einzelne Übertragung herausfiltern.
Dieses Multiplexprinzip sollte eine Alternative zur gemeinsamen Verwendung von FDM und TDM darstellen und wird z.B. bei UMTS genutzt.

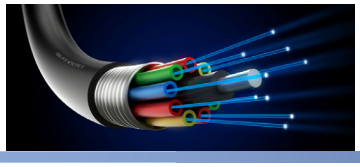
Generell gilt der Zusammenhang, dass je länger die Codefolge ist, umso geringer ist die Nutzdatenrate. Dafür erhöht sich die Anzahl möglicher Nutzer, bzw. verringert sich die Sendeleistung. Je nach Anwendung können den Sendern unterschiedlich lange Codemuster zugewiesen werden, wie dies bei UMTS erfolgt. Für eine Bruttodatenrate von 1,92 Mbps wird bei diesem Verfahren eine Codefolge mit der Länge 4 Bit verwendet. Bei 30 kbps ist die Codefolge 256 Bit lang. Dabei wird eine konstante Chip-Rate von 3,84 Mcps eingesetzt.

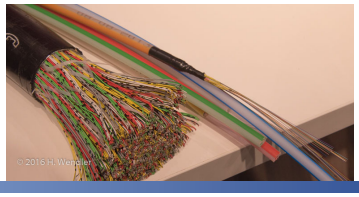
Mehrfachnutzung von Medien: Raummultiplex



- **Raummultiplex („Kupfermultiplex“)**

Bündelung von Adern(paaren)

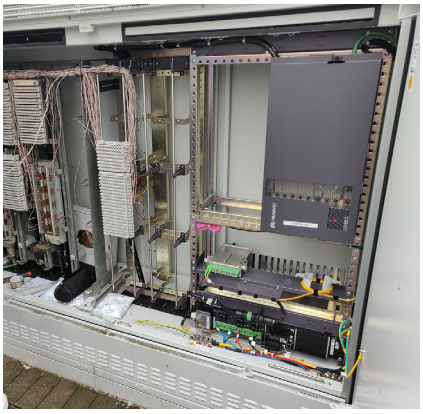




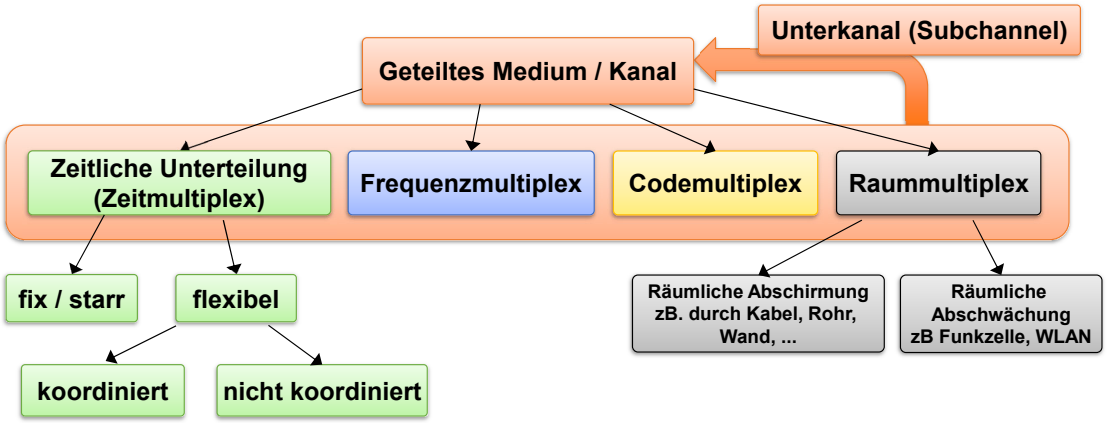
https://mg-heute.de/karnevalsscherz-deutsche-glasfaser-macht-moenchengladbach-fit-fuer-die-zukunft/
https://www.fiercetelecom.com/telecom/at-t-level-3-centurylink-slower-spending-result-lower-north-american-optical-capex-says

Raummultiplex: Die Mehrfachnutzung des Mediums wird beim Raummultiplex dadurch realisiert, dass mehrere physikalische Leitungen (Adern(paare) bei Kupfer- oder Glasfaserkabel, Zellen bei drahtlosen Netzen, …) zusammengefasst werden.

Beispiele Raummultiplex ;-)



Weitere Unterteilung von Kanälen in Unterkanäle



- **Submedium = Shared Medium?**
  - Verwendung von FDM zur Einteilung des Mediums in Frequenz(unter)kanäle
  - Verwendung von TDM pro Frequenzkanal
  - Z.B. GSM, UMTS, LTE, (Wi-Fi)

Multiplexverfahren können auch in Kombination eingesetzt werden – oft wird ein gegebenes Frequenzband mittels FDM in schmalere Kanäle unterteilt und die einzelnen Frequenzkanäle danach mittels TDM weiter unterteilt, um Slots koordiniert oder nicht koordiniert Stationen zuzuteilen.

Diese Kombination findet man z.B. bei GSM: das gesamte GSM-Frequenzspektrum wird in Frequenzkanäle unterteilt und pro Frequenzkanal weist ein Scheduler den Stationen nach einem festen Schedule Slots zu (immer jeder achte Slot).

Bei UMTS und LTE findet dieses Prinzip auch Verwendung, allerdings mit bedarfsgesteuerten Schedules.

Bei Wi-Fi wird das Frequenzspektrum auch in Frequenzkanäle eingeteilt, allerdings überlappen diese Frequenzkanäle. Daher kann man eigentlich nicht von FDM sprechen sondern von irgendwas-wie-beinahe-FDM, aber der generelle Ansatz ist gleich. Jeder Frequenzkanal wird mittels nicht-koordiniertem TDM weiter unterteilt.

Denkbar ist auch das umgekehrte Vorgehen: das Medium wird mit TDM unterteilt und jeder Slot noch einmal weiter mittels FDM. Dies erfordert allerdings eine aufwendigere Koordination.

Zusammenfassung

- **Umformung von Quellsignalen in Signale im Medium**
  - Anpassung digitaler Daten an das Medium, z.B. Strom auf Kupferkabel
- **Mediencharakteristiken und Übertragungsparameter**
  - Bandbreite eines Mediums bestimmt zusammen mit der gewählten Codierung die maximal erzielbare Datenrate
  - Datenrate und Latenz als Charakteristiken einer konkreten Übertragungsstrecke
- **Codierung und Multiplexing**
  - Leitungscodes im Basisband, Modulationsverfahren im Breitband
  - Multiplexverfahren zur Aufteilung einer Übertragungsstrecke in Kanäle

Lessons learned

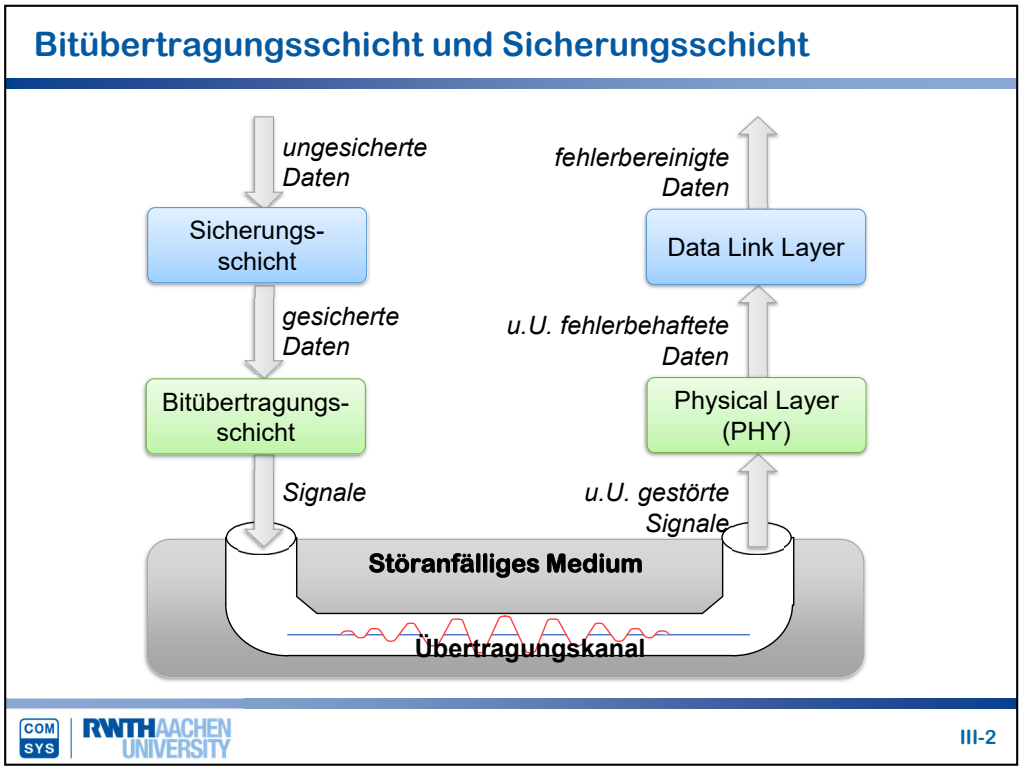
- **Wichtige Begriffe/Konzepte des Kapitels**
  - Digitale vs. analoge Signale
    - Beide sind elektromagnetische Signale
  - Bandbreite
    - Beschränkt die minimale Dauer eines Schritts und damit die Datenrate
  - Leitungscodierung/Modulation
    - Anpassung der Daten an die Mediencharakteristiken: Basis-/Breitband
  - Latenz
    - Verzögerung eines Signals bei der Übertragung über ein Medium
  - Nyqusit- und Shannon-Theorem
    - Bestimmung der möglichen Datenrate
  - Multiplexing
    - Gemeinsame Nutzung eines Mediums (meist TDM und/oder FDM)

Datenkommunikation
Kapitel 3: Sicherungsschicht

Klaus Wehrle

Communication and Distributed Systems
Chair of Computer Science 4
RWTH Aachen University

http://www.comsys.rwth-aachen.de



Zur Erinnerung: die beiden untersten Schichten sorgen gemeinsam für eine fehlerfreie Datenübertragung zwischen benachbarten Rechnern. „Benachbart“ heißt, dass die kommunizierenden Rechner durch ein physikalisches Medium direkt miteinander verbunden sind.

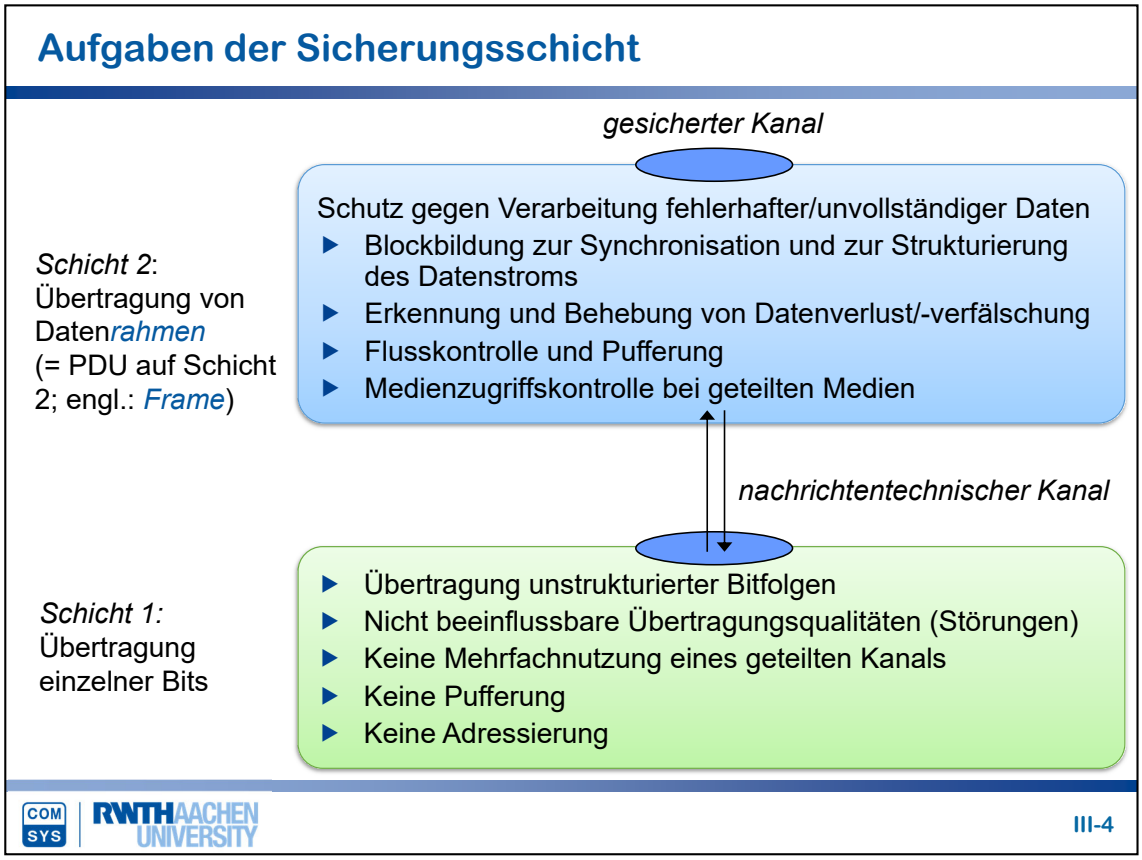
Die Bitübertragungsschicht realisiert einen nachrichtentechnischen Kanal zur Übertragung von Bitfolgen, allerdings können bei der Übertragung Fehler auftreten.

Die Sicherungsschicht erweitert einen nachrichtentechnischen Kanal zum eigenständigen, abstrakten Medium „gesicherter Kanal“. „Gesichert“ heißt hierbei: fehlerfrei. Notwendig dazu sind:

- Strukturierung des Bitstromes in Rahmen (Frames) als Basiseinheit der Übertragung
- Fehlererkennung und –behandlung (Quittungen und Übertragungswiederholungen, siehe Alternating Bit Protocol)
- Flusskontrolle (Vermeidung der Überlastung/Pufferüberlauf des Empfängers)
- Regelung des Zugriffs auf ein Medium, welches gemeinsam von mehreren Stationen verwendet wird.

Themenübersicht

- **Datenkommunikation**
  - Einführung, Begriffe und allgemeine Grundlagen
  - Technische und nachrichtentechnische Grundlagen: Medien, Signale, Bandbreite, Leitungscodes und Modulation, Multiplexing
  - Lokale Netze: Strukturierung des Datenstroms, Fehlererkennung/ -behebung, Flusssteuerung, Medienzugriff, Ethernet
  - Internet und Internet-Protokolle:
    - Vermittlungsschicht: IP, Routing
    - Transportschicht: TCP
  - Anwendungsorientierte Schichten



Die Sicherungsschicht hat die Aufgabe, höheren Schichten einen Übertragungskanal zur Verfügung zu stellen, über den Daten vollständig und fehlerfrei (= *gesichert*) übertragen werden. Der Sicherungsschicht steht aber nur ein ungesicherter Schicht-1-Dienst zur Verfügung, der unstrukturierte Bitfolgen ohne Kontrollmechanismen überträgt *(nachrichtentechnischer Kanal).*

Die Sicherungsschicht muss daher einen Schutz gegen Missinterpretation empfangener Bitfolgen bieten. Dazu können entweder einzelne Bits oder Blöcke von Bits gesichert werden. Im Allgemeinen ist die Sicherung von Blöcken effizienter und daher wird nur dieses Vorgehen behandelt. Der Empfänger kann durch Blockbildung innerhalb des empfangenen Datenstroms Muster identifizieren und Datenblöcke daraufhin als korrekt oder falsch klassifizieren. Die effiziente Erkennung und Behandlung von Fehlern setzt daher zwingend voraus, dass die Daten vor der Übertragung in Blöcke strukturiert werden.

Um verschiedene Arten von Fehlern oder Problemen zu erkennen und zu behandeln, hat die Sicherungsschicht die folgenden Aufgaben (die je nach Protokoll nicht unbedingt alle implementiert werden (müssen)):

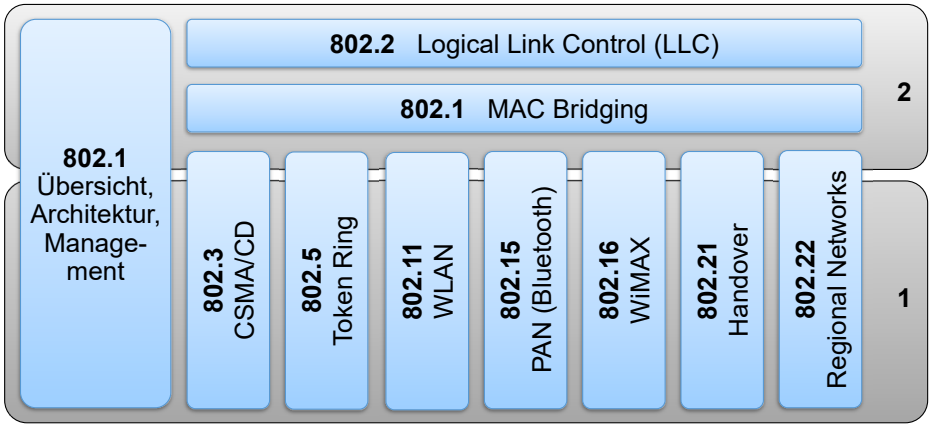
- Synchronisation, Strukturierung des Datenstroms/Erkennen von Blockgrenzen: Ganz allgemein müssen Zeichen bzw. Bitfolgen innerhalb eines unstrukturierten Bitstroms identifiziert werden können. Meist wird der Bitstrom dazu in sogenannte *Rahmen* unterteilt, um zur Bewältigung der restlichen Aufgaben eine fest definierte Struktur verwenden zu können. Die Anfänge eines Rahmens müssen erkannt werden, vor allem auch nach Übertragung fehlerhafter (bspw. abgeschnittener) Rahmen.
- Codetransparenz: Es muss sichergestellt werden, dass beliebige Zeichen- bzw. Bitfolgen übertragen werden können. Ob hierbei Probleme auftreten können, hängt von der Art der gewählten Strukturierung/Rahmenbildung ab und kann daher auch als Unterpunkt des vorherigen Punktes angesehen werden.
- Fehlererkennung und -behandlung: Es sollen sowohl Fehler bei der Datenübertragung als auch Fehler im Protokollablauf erkannt und behoben werden. Fehler bei der Datenübertragung können durch Suche nach Mustern in den empfangenen Blöcken erfolgen – die Behandlung eines Fehlern erfolgt z.B. einfach durch eine erneute Übertragung wie beim Alternating-Bit-Protokoll. Um Fehler im Protokollablauf erkennen zu können, müssen Protokolle vollständig spezifiziert werden und es muss auch definiert werden, wie in welcher Fehlersituation zu verfahren ist.
- Flusskontrolle und Pufferung: Überlastsituationen müssen verhindert werden – es soll nicht vorkommen, dass ein Empfänger überlastet wird, da er die empfangenen Daten nicht schnell genug verarbeiten kann und sie verwerfen

muss. Dazu müssen Daten gepuffert werden, aber es muss auch möglich sein, der Gegenstelle zu signalisieren, dass keine weiteren Daten mehr gepuffert werden können, wenn der Puffer voll ist.

- Koordinierter Medienzugriff: Vergabe von Senderechten/Medienzuteilung bei geteiltem Übertragungsmedium. Dies kann durch die bereits im letzten Kapitel vorgestellten Multiplexverfahren erreicht werden, doch sind diese i.A. zu unflexibel.
- Verfahren zur Steuerung der Übermittlung: z.B. Initialisierung, Terminierung, Identifikation, Halbduplex-/Vollduplexbetrieb.

LAN/MAN: Standardfamilie IEEE 802

- **IEEE: Amerikanischer Verband der Elektroingenieure**
  - Im Bereich von LAN/MAN Netzen sehr verbreitete Standards
  - Standardfamilie IEEE 802 unterteilt Schicht 2 in *Medium Access Control* (MAC) und Logical Link Control (LLC)



Die IEEE wurde bereits als die zentrale Standardisierungsorganisation im Bereich der lokalen Netze erwähnt.

Laut IEEE wird Schicht 2 in zwei Teilfunktionalitäten gegliedert. Der obere Teil ist unabhängig von dem konkreten Medium und stellt der nächsthöheren Schicht eine einheitliche Schnittstelle zum Zugriff auf das Netz zur Verfügung. Hier sind all die Funktionen definiert, die unabhängig vom konkreten Netzaufbau sind: Flusssteuerung, Quittierungsmechanismus, ... – zusammengefasst als 802.2, Logical Link Control.

Der untere Teil umfasst hardwarenähere Aspekte. Dies umfasst die Festlegung von Übertragungsmedien und Netztopologien – und damit zusammenhängenden Aufgaben. vom Medium hängt es ab, welches Fehlererkennungs- oder -korrekturverfahren eingesetzt wird. (Bei Glasfaser treten kaum Bitfehler auf, weshalb eine einfache CRC oder gar ein Parity-Bit ausreicht, bei Funkübertragung sind Bitfehler die Regel, weshalb meistens CRC zusammen mit Fehlerkorrekturverfahren eingesetzt wird.) Vom Medium hängt es ab, welche Arten des Medienzugriffs möglich sind. (CSMA/CD beispielsweise, welches bei Ethernet über Kupfer- oder Glasfaserkabel eingesetzt wird, funktioniert nicht bei drahtloser Kommunikation – WLAN definiert deshalb eine Variante namens CSMA/CA.)

Sicherungsschicht – Überblick

Layer 2b: Logical Link Control

1. Protocol Data Units
- Rahmenbildung
- Blockerkennung
  - Längenfeld
  - Layer-1-Codierung
  - Code-Violation
  - Character-Stuffing
  - Bit-Stuffing

2. Fehlererkennung und -behandlung
- Prüfsummen
  - Paritätsüberprüfung, CRC
- Proaktive Fehlerbehandlung
  - Hinzufügen von Redundanz
  - Hamming-Code
- Reaktive Fehlerbehandlung
  - Automatic Repeat Request (ARQ)

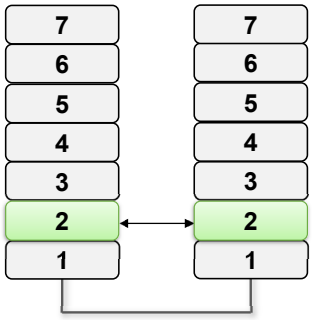
3. Flusskontrolle
- Stop and Wait
- Sliding Window

Layer 2a: Medium Access Control

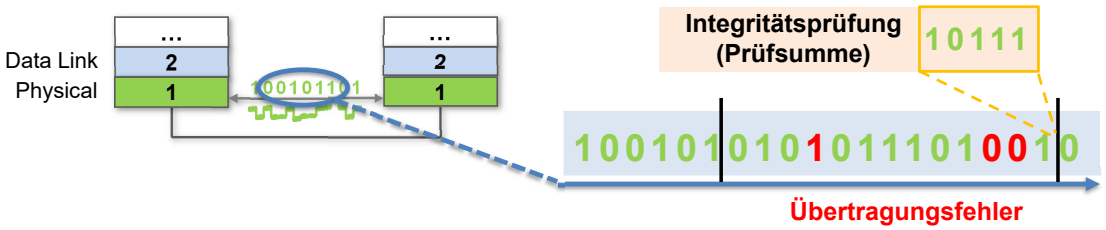
4. Medium Access Control
- LAN-Topologien
- Medium access control
  - Geregelter Zugriff (Polling, Token Ring)
  - Konkurrierender Zugriff (Aloha, CSMA/CD)
- Ethernet

Kapitel 3: Sicherungsschicht

- **Rahmenbildung**
  - Blockbildung, Codetransparenz
- **Fehlererkennung/-behandlung und Flusskontrolle**
  - Fehlererkennende Codes (CRC)
  - Proaktive und reaktive Fehlerbehebung (FEC und ARQ)
  - Flusskontrolle durch Sliding Window
  - Beispielprotokoll: HDLC
- **Medienzugriff**
  - Topologien
  - Token Passing, CSMA/CD, CSMA/CA
- **Standards für lokale Netze**
  - Ethernet



Rahmenbildung



- **Schicht 1: Unstrukturierter Bitstrom**
  - Übertragungsfehler (gekippte Bits) können nicht erkannt werden
- **Benötigt:**
  - *Integritätsprüfung* zur Erkennung von Fehlern → zusätzliche, redundante Daten (= *Prüfsumme*)
  - *Struktur im Bitstrom (= Rahmen)*
    - um festzulegen, welche Bits von der Prüfsumme abgedeckt werden
    - um die Prüfsumme von den Datenbits abgrenzen zu können

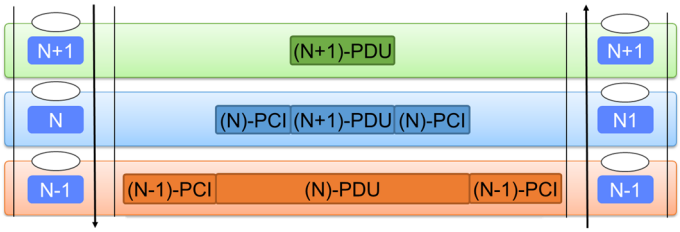
Auf Schicht 2 wird nicht mehr die Übertragung einzelner Signale (einzelne Bits bzw. Bitkombinationen bei Verwendung mehrwertiger Signale) betrachtet, sondern die Übertragung von Bitfolgen fester oder variabler Länge. Dies ist z.B. notwendig, um Übertragungsfehler zu erkennen, die durch Störeinflüsse auf Schicht 1 zustande gekommen sind. Anhand eines einzelnen empfangenen Bits wird man nicht entscheiden können, ob es korrekt ist – dies wird erst möglich, wenn man Folgen von Bits betrachtet.

Aber diese Fehlererkennung ist nicht der einzige Sinn, den die Rahmenbildung hat. Während auf Schicht 1 zwar bestimmte Aufgaben umgesetzt werden müssen, wurden im ISO/OSI-Referenzmodell keine Protokolle für diese Schicht definiert; erst auf Schicht 2 werden Protokolle eingesetzt: bei der Betrachtung einzelner Bits können keine Protokollkontrollinformationen übertragen werden, da der Empfänger aufgrund eines einzelnen Bits nicht entscheiden kann, ob Nutzdaten oder Kontrollinformationen vorliegen. Bei Codes wie 4B/5B liegen einzelne Bitkombinationen für Kontrollinformationen vor, aber ihr Umfang ist relativ eingeschränkt. Erst wenn man die übertragenen Daten mit Struktur versieht, kann man einfach erkennen, was Kontroll- und was Nutzdaten sind.

Rahmenbildung bezeichnet also den Vorgang, einen zu übertragenden Bitstrom in Blöcke aufzuteilen, die einer festen Struktur folgen, um die für Protokolle nötigen Kontrollinformationen hinzufügen und die zur gesicherten Übertragung notwendigen Mechanismen (Prüfsumme) implementieren zu können.

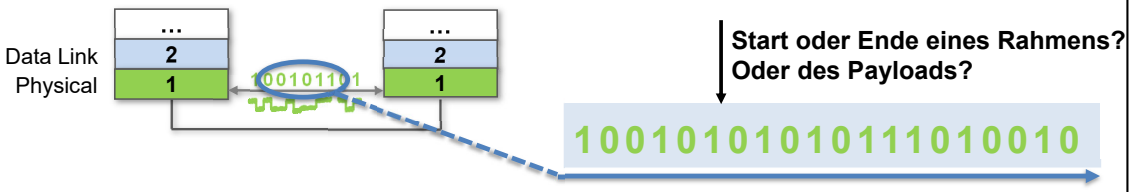
Zur Erinnerung: PDUs

- **Schicht 2 führt PDUs ein**
  - Nutzdaten (*Payload*) + Kontrollinformationen (zur Koordination der Protokolloperationen)
    - Hier: Hinzufügen einer Prüfsumme zur Fehlererkennung
    - Andere Kontrollinformationen:
      - Adressen, Sequenznummern, …
      - Üblicherweise als *Header* den Daten vorangestellt
      - Auf Schicht 2 auch als *Trailer* an die Daten angehangen



  - Herausforderung: Erkennung von *Beginn und Ende eines Rahmens*

Erkennung von Rahmenbeginn/-ende



- **Wie können Start/Ende eines Rahmens erkannt werden?**
  - Payload darf nicht mit Begrenzungen durcheinandergebracht werden (Codetransparenz)
    - *Implizite Codetransparenz*: Payload wird nicht modifiziert
      - (1) Verwendung einer Längenangabe
      - (2) Spezielle Kontrollzeichen auf Schicht 1 oder 2
      - (3) Codeverletzung auf Schicht 1
    - *Explizite Codetransparenz*: Payload muss modifiziert werden
      - (4) Character-Stuffing
      - (5) Bit-Stuffing

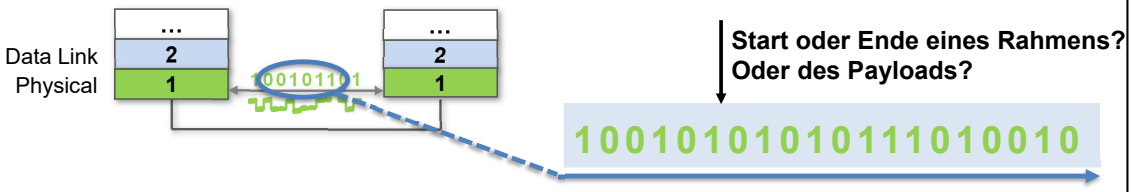
Statt den Begriff des Rahmens zu nutzen, verwenden wir im Folgenden oft den Begriff des Blocks – um darzustellen, dass uns eine weitere Rahmenstruktur noch nicht interessiert, sondern es zunächst nur um Kennzeichnung von Anfang und Ende einer zusammengehörigen Bitfolge geht. Ob diese nur aus Daten (Payload) oder auch aus weiteren Kontrollinformationen (Header) besteht, ist zunächst uninteressant.

Wir betrachten in dieser Vorlesung nur die synchrone Übertragung. Bei der synchronen Übertragung wird eine Folge von Zeichen/Bits innerhalb eines Blocks (bzw. Rahmens) übertragen. Anfang und Ende des Blocks sind durch bestimmte Zeichen-/Bit-Muster eindeutig festgelegt und können somit vom Empfänger erkannt werden. Eine Synchronisation ist somit nur am Anfang eines Blocks notwendig (dies wird dadurch erreicht, dass dem Blockstartmuster einige Zeichen/Bits (Präambel) vorangestellt werden, die nur der Synchronisierung dienen).

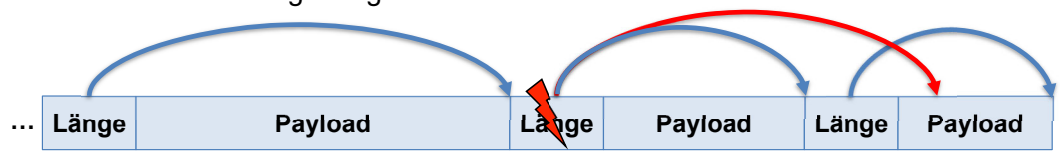
Als Sonderfall kann hierbei die Vorgehensweise betrachtet werden, nur den Anfang eines Blockes eindeutig zu kennzeichnen und die Länge des Nutzdatenfeldes durch ein zusätzliches, sich an das Blockstartmuster anschließendes Längenfeld zu spezifizieren.

Alternativ kann man auch den Leitungscode von Schicht 1 zur Blockbildung „missbrauchen“.

Codetransparenz: Längenangabe



- **Wie können Start/Ende eines Rahmens erkannt werden?**
  - Payload darf nicht mit Begrenzungen durcheinandergebracht werden (Codetransparenz)
    - *(1) Verwendung einer Längenangabe*
      - Einfach…
      - … aber Probleme bei Verlust der Synchronisation, z.B. durch Bitfehler in Längenangabe



Bei dieser einfachen Methode wird eine weitere Kontrollinformation vor den Daten eingefügt, welche die Anzahl der Bits oder Bytes im Block angibt. Sobald die Implementierung der Sicherungsschicht im Empfänger dieses Feld liest, weiß sie, wie lang der Rahmen ist. Mehr Kontrollinformationen sind nicht nötig - was ein klarer Vorteil zu anderen Vorgehensweise ist.

Bei dieser Methode kann es jedoch zu Problemen kommen, falls das Längenfeld während der Übertragung verfälscht wird. Der Empfänger kann zwar typischerweise erkennen, dass er einen verfälschten Block erhalten hat, kann aber den Anfang des nächsten Blocks nicht mehr korrekt erkennen.

Verwendung spezieller Kontrollzeichen

(2) Spezielle Kontrollzeichen auf Schicht 2

- Definition spezieller Zeichen im verwendeten Alphabet
- Beispiel: ASCII für text-basierte Codierung

← `02 72 69 76 76 79 32 87 79 82 76 68 03` `02 … 03` `00 00` `02 … 03`

STX (=02): **Start of text**
ETX (=03): **End of text**

- Markieren Blockbegrenzung
- Können nicht im Payload vorkommen

		000	001	010	011	100	101	110	111
0000	0	NUL	(DLE)	SP	0	@	P	`	p
0001	1	(SOH)	DC1	!	1	A	Q	a	q
0010	2	(STX)	DC2	"	2	B	R	b	r
0011	3	(ETX)	DC3	#	3	C	S	c	s
0100	4	(EOT)	DC4	$	4	D	T	d	t
0101	5	(ENQ)	(NAK)	%	5	E	U	e	u
0110	6	(ACK)	(SYN)	&	6	F	V	f	v
0111	7	BEL	(ETB)	´	7	G	W	g	w
1000	8	(BS)	CAN	(	8	H	X	h	x
1001	9	(HT)	EM	)	9	I	Y	i	y
1010	10	(LF)	SUB	*	:	J	Z	j	z
	⋮								

Bei zeichenorientierter Übertragung wird ein Datenstrom als Abfolge von Zeichen angesehen. Einige der Zeichen (hier grau/gelb hervorgehoben) werden als Steuerzeichen zur Regelung des Kommunikationsablaufs verwendet, z.B. STX (START OF TEXT) als Startmuster einer Übertragung, ETX (END OF TEXT) als Endmuster der Übertragung.

Verwendung spezieller Kontrollzeichen

(2) Spezielle Kontrollzeichen auf Schicht 1

- Definition spezieller Zeichen im verwendeten Code
- Beispiel: 4B/5B-Code

Layer 2: `---` `(Start Delimiter)` `0001` `0010` `0011` `(End)` `(Start Delimiter)` …

Layer 1: `00000 11000 10001 01001 10100 10100 01101 11000 10001` …

Symbol	Channel code	Payload value
0	11110	0000
1	01001	0001
2	10100	0010
3	10101	0011
4	01010	0100
5	01011	0101
6	01110	0110
7	01111	0111
8	10010	1000
9	10011	1001
A	10110	1010
B	10111	1011
C	11010	1100
D	11011	1101
E	11100	1110
F	11101	1111

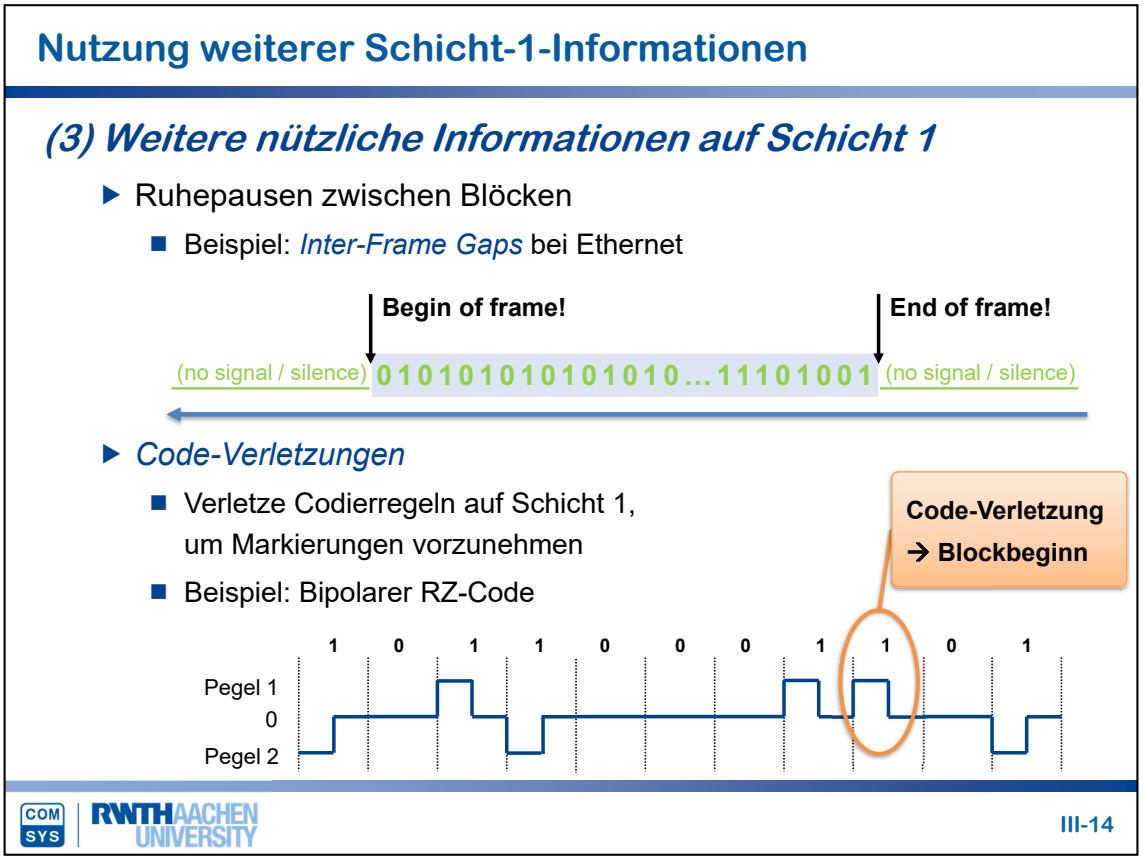
Symbol	Channel code	Protocol command
Q	00000	Quiet
I	11111	Idle
H	00100	Halt
J	11000	1st of sequential SD-Pair
K	10001	2nd of sequential SD-Pair
T	01101	End
R	00111	
S	11001	

`JK`: Start of Frame
`TT`: End of Frame

- Markieren Blockbegrenzung
- Können nicht im Payload vorkommen

Alternative: je nach Leitungscode existieren solche speziellen Zeichen bereits auf Schicht 1.

Im vorherigen Kapitel wurde bereits der 4B/5B-Code vorgestellt, der entsprechende Kontrollzeichen mit sich bringt.



Bietet der Leitungscode auf Schicht 1 keine Steuerzeichen, kann man eventuell andere Informationen nutzen.

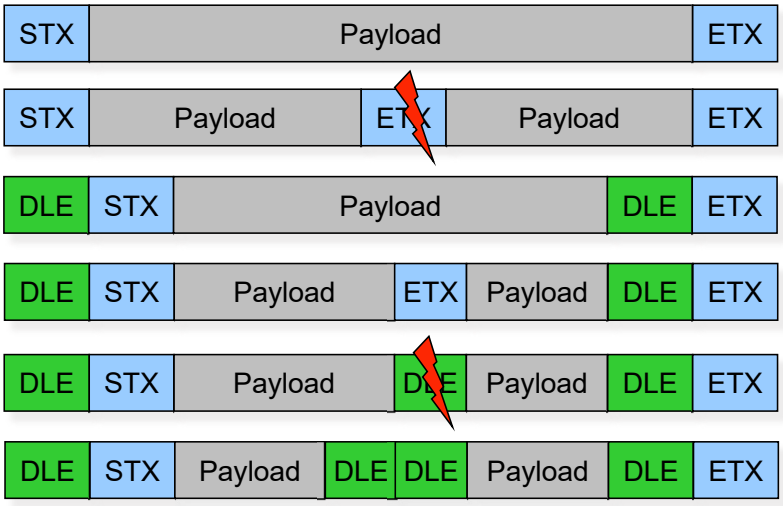
Z.B. kann man Ruhepausen zwischen je zwei übertragenen Blöcken lassen. Das Ende eines Blocks erkennt man dann dadurch, dass eine Ruhepause folgt; ebenso lässt sich der Beginn des nächsten Blocks einfach nach Ende der Ruhepause erkennen. Diese Technik wurde bereits oben zusammen mit der Längenangabe genannt. Diese einfache Technik der Blockbegrenzung wird z.B. bei Ethernet eingesetzt.

Eine andere Möglichkeit sind explizite Code-Verletzungen (Code Violation). Hier im Beispiel dargestellt ist ein vorher nicht behandelter Leitungscode, eine Variante des RZ-Codes. Bei dieser wird eine logischen Null durch eine Ruhepause dargestellt (Halten des Null-Pegels für eine definierte Zeit), die logische 1 abwechselnd durch einen Wechsel von hohem und niedrigem Pegel zum Null-Pegel. Verwendet man nun z.B. zwei Mal in Folge einen Wechsel vom hohen Pegel zum Null-Pegel, ist dies nicht im Code definiert – der Code wird verletzt. Der Sender kann diese Verletzung absichtlich hervorrufen, um einen Blockbeginn zu kennzeichnen.

Character-Stuffing

(4) Markierung von Kontrollsequenzen

- DLE (= Data Link Escape) markiert das folgende Zeichen als Kontrollinformation



Blöcke werden in Variante (2) durch die Zeichen STX und ETX begrenzt, welche dem Empfänger mitteilen, welche empfangenen Bits die Dateneinheit darstellen. Oben wurde gesagt, dass diese Zeichen nicht im Block vorkommen können.

Im Payload müssen aber beliebige Bit-/Bytefolgen übertragbar sein – und die Bitfolgen, die STX/ETX darstellen, können durchaus auch Teil der zu übertragenen Daten sein. In der zweiten Zeile dargestellt ist so ein Problemfall: das ETX-Zeichen ist zufällig im Nutzdatenteil enthalten. In diesem Fall wird der Empfänger es als die hintere Rahmenbegrenzung betrachten und die Daten für abgeschlossen halten. Wir benötigen hier explizite Codetransparenz – es muss sichergestellt sein, dass die übertragenen Daten nie mit Kontrollinformationen verwechselt werden. Dazu müssen im Datenfeld zufällig auftauchende Kontrollsequenzen maskiert werden.

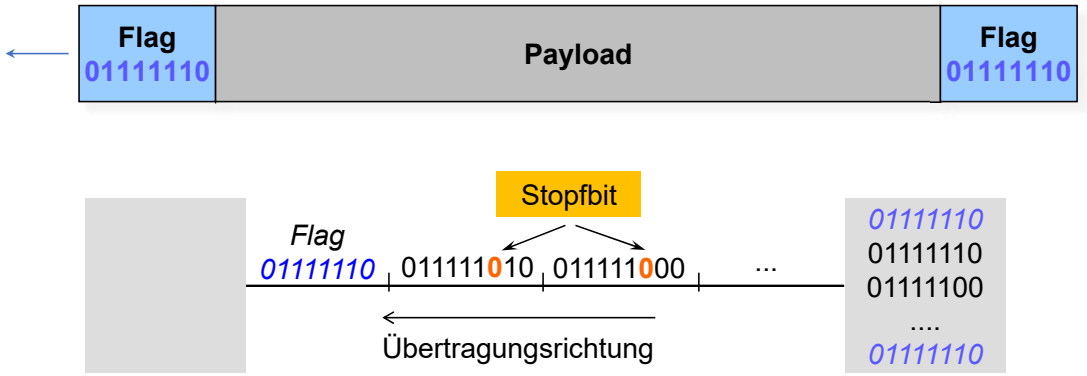
Ein Beispiel, wie explizite Codetransparenz erreicht werden kann, zeigt Zeile 3 der Abbildung auf der Folie: es wird festgelegt, dass einem Steuerzeichen immer ein DLE-Zeichen (Data Link Escape) vorangestellt sein muss. Nun ist das Auftreten eines ETX im Datenteil kein Problem mehr (Zeile 4). Das DLE sorgt hier also für die Codetransparenz – Steuerzeichen werden nur als solche erkannt, wenn ein DLE davor steht.

Dafür tritt ein neues Problem auf (Zeile 5): tritt die DLE-Bitfolge im Datenteil auf, wird sie als Beginn eines Steuerzeichens interpretiert, der Datenteil wird wieder nicht korrekt verarbeitet. Dies wird vermieden, indem der Sender ein weiteres DLE vor dieser Bitfolge im Datenteil einfügt. Ein doppeltes DLE (Zeile 6) bedeutet, dass das zweite DLE als Datenzeichen aufzufassen ist. Auf diese Art und Weise kann auch die DLE-Bitfolge als Teil der Daten übertragen werden.

Bit-Stuffing

(5) Maskierung von angeblichen Kontrollsequenzen

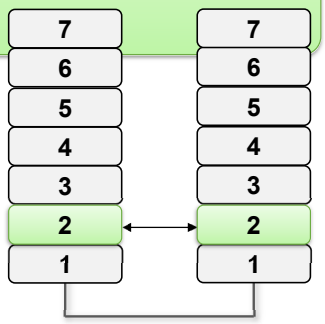
- Blockbegrenzung (Flag) ist eine ausgezeichnete Bitfolge (hier: 01111110)
  - *Bit-Stuffing* zur Vermeidung des Auftretens von 01111110 im Payload
    - Füge nach 5 aufeinanderfolgenden „1“-en eine „0“ ein, die der Empfänger wieder entfernt



Bit-Stuffing verfolgt das gleiche Prinzip wie bei der zeichenorientierten Übertragung: Anfang und Ende eines Rahmens werden durch eine spezielle Bitfolge (hier beispielhaft: *01111110*) gekennzeichnet, die nicht innerhalb des Blocks auftreten darf. Um dies zu vermeiden, fügt man nun im Datenteil nach fünf aufeinanderfolgenden Einsen eine Null ein (auch dann, wenn gerade eine Null folgt). Somit erkennt der Empfänger deutlich an sechs aufeinanderfolgenden Einsen das Blockende und weiß, dass er bei einer Folge von fünf Einsen und einer Null die Null entfernen muss.

Kapitel 3: Sicherungsschicht

- **Rahmenbildung**
  - Blockbildung, Codetransparenz durch Character/Bit Stuffing
- **Fehlererkennung/-behandlung und Flusskontrolle**
  - Fehlererkennende Codes (CRC)
  - Proaktive und reaktive Fehlerbehebung (FEC und ARQ)
  - Flusskontrolle durch Sliding Window
  - Beispielprotokoll: HDLC
- **Medienzugriff**
  - Topologien
  - Token Passing, CSMA/CD, CSMA/CA
- **Standards für lokale Netze**
  - Ethernet



Fehlerursachen, Fehlertypen

- **Übertragungsfehler**
  - Störeinflüsse führen zu *falsch detektierten Bits*
  - *Hardwareinduzierte Fehler*, die auf dem Übertragungsmedium entstehen (seltener auch in der Anschlusselektronik der kommunizierenden Stationen)
    - Art und Häufigkeit von Übertragungsmedium und Datenrate abhängig
      - *Einzelbitfehler*
      - Bündelfehler (*Fehlerburst*)
  - Des weiteren: Synchronisierungsfehler
    - Alle Bits werden falsch abgetastet

Störeinflüsse bei der Übertragung digitaler Daten führen normalerweise zur fehlerhaften Erkennung der Bits. Sowohl die Art der Störeinflüsse als auch die Häufigkeit, mit der Signale verfälscht werden, hängen von der Art des Mediums ab. Drahtlose Medien weisen im Allgemeinen deutlich höhere Fehlerraten auf als kabelgebundene Medien, Kupfermedien immer noch höhere Fehlerraten als Glasfaser.

Man unterscheidet generell zwei Fehlertypen:

- *Einzelbitfehler*: Fehler, bei denen nur ein Bit gekippt (= verfälscht) wurde, verursacht z.B. durch Spitzen im Rauschen.
- *Bündelfehler (Fehlerbursts)*: länger anhaltende Fehler, die über viele Bits in Folge gehen (z.B. durch längere Überspannungen oder Interferenz mit benachbarten Funkzellen). Achtung: ein Fehlerburst zeichnet sich dadurch aus, dass über eine längere Folge von Bits Verfälschungen aufgetreten sind; es müssen aber nicht notwendigerweise alle Bits der Folge verfälscht sein.

Ob ein Störeinfluss sich als Einzelbitfehler oder als Fehlerburst auswirkt, hängt von der Datenrate ab, mit der gesendet wird – je höher die Datenrate ist (je kürzer die Schrittdauer ist), desto mehr Bits können durch einen kurzen Störeinfluss verfälscht werden.

Eine weitere Fehlerklasse sind die *Synchronisierungsfehler*: der Empfänger ist falsch synchronisiert und tastet alle Bits falsch ab. Dieses Problem lässt sich lösen, indem einem Rahmen eine Präambel mit fest definierten Pegelwechseln vorangestellt wird und/oder selbsttaktende Leitungscodes eingesetzt werden. Diese Fehlerklasse soll hier nicht genauer betrachtet werden.

Fehlerwirkungen: Rechenbeispiel

- **Eine Störung von 20 ms führt …**
  - bei Telex (50 Bit/s, Signaldauer: 20 ms) zu einem Fehler von 1 Bit Länge → Einzelfehler
  - bei ISDN (64 kBit/s, Signaldauer: 15,625 µs) zu einem Fehler von 1280 Bit Länge → Fehlerburst
  - bei SDH (Weitverkehrsnetz, unterschiedliche Datenraten möglich)
    - 155 MBit/s, Signaldauer: 6,45 ns: zu einem Fehler von ca. 3,1 Mbit Länge
    - 622 MBit/s, Signaldauer: 1,61 ns: zu einem Fehler von ca. 12,4 Mbit Länge
    - 2,4 GBit/s, Signaldauer: 0,4 ns: zu einem Fehler von ca. 48 Mbit Länge

    → extreme Fehlerbursts!

(SDH: Synchronous Digital Hierarchy; analog dazu in den USA: Sonet. Weitverkehrstechnologie, nicht Fokus der Vorlesung.)

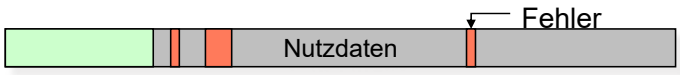
Fehlerhäufigkeiten

- **Maß für die Fehlerhäufigkeit:**
  - *Bitfehlerrate* (bit error rate, BER) = $\frac{\text{Summe gestörte Bits}}{\text{Summe übertragene Bits}}$
  - Abhängig von
    - Übertragungsmedium
    - Gesamtlänge des Übertragungswegs
- **Typische Wahrscheinlichkeiten für Bitfehler:**
  - Analoges Fernsprechnetz: $2 \cdot 10^{-4}$
  - Funkstrecke: 10^{-3} – 10^{-4}
  - Ethernet (10Base2): 10^{-9} – 10^{-10}
  - Glasfaser: 10^{-10} – 10^{-12}

Die Bitfehlerrate ist ein typisches Maß für die Wahrscheinlichkeit des Auftretens eines Übertragungsfehlers. Zu jedem Übertragungsmedium wird man eine Angabe zur BER finden.

Fehlerwirkungen

- **Die Auswirkung eines Fehlers ist abhängig davon, welche Bits des Rahmens betroffen sind:**
  - (Nutz-)Datenfehler: Bits innerhalb der Nutzdaten



  - Protokollfehler: Bits in Protokollkontrolldaten, Steuerzeichen, Adressen oder sonstigen protokollrelevanten Daten

Treten innerhalb der Nutzdaten Fehler auf, werden die Daten zwar korrekt zugestellt, aber die nächsthöhere Schicht des Empfängers verarbeitet fehlerhafte Daten, was letztlich zu Problemen auf den höheren Schichten oder in der empfangenden Anwendung führt.

Treten Fehler in den Kontrollinformationen auf, wird im Allgemeinen der Protokollablauf gestört; z.B. könnten die Daten dem falschen Rechner zugestellt oder auf dem empfangenden Rechner dem falschen Prozess zugestellt werden.

Beide Arten von Fehlern müssen gleichermaßen erkannt und behandelt werden.

Fehlererkennung und -behandlung

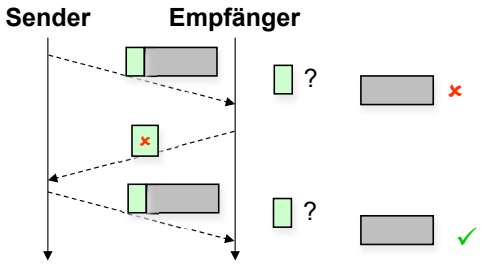
- **Erste Notwendigkeit: Fehlererkennung**
  - Erzeuge „Muster“ im Block, anhand derer der Empfänger die Korrektheit / das Auftreten von Fehlern erkennen kann
    - Durch künstliches Hinzufügen von Redundanz zum Block beim Sender
    - *Error detecting codes*

Teilweise kann man Fehler einfach ignorieren. Beispielsweise filtert das menschliche Hirn in gewissem Umfang Störungen heraus, so dass bei Übertragung von Audio und Video einzelne gestörte Bildteile oder Audiosequenzen keine Probleme verursachen.

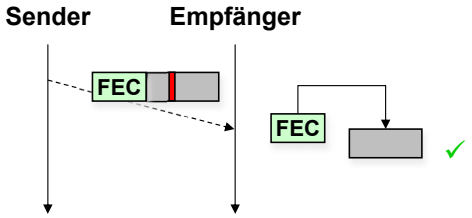
Bei den meisten Daten allerdings ist es notwendig, sie fehlerfrei zu erhalten (Webseite, E-Mail, Datei, …). Daher werden Techniken zur Fehlererkennung benötigt. Ganz generell gesprochen fügt der Sender Redundanzen zu einem zu übertragenden Block hinzu, anhand derer der Empfänger Übertragungsfehler erkennen lassen. Die Daten werden so codiert, dass eine Fehlererkennung möglich wird: man spricht von „Error Detecting Codes“.

Fehlererkennung und -behandlung

- **Nach Erkennung: Fehlerbehandlung**
  - Reaktiv: reagiere auf erkannten Fehler durch Neuübertragung
    - *Automatic Repeat Request (ARQ)*



  - Proaktiv: Blöcke enthalten ausreichend Redundanzen zur Korrektur
    - *Error correcting codes (Forward Error Correction, FEC)*



Um einen erkannten Fehler zu beheben, gibt es zwei Möglichkeiten: proaktives oder reaktives Vorgehen.

Bei proaktivem Vorgehen geht man davon aus, dass Fehler auftreten werden und codiert die Daten proaktiv so, dass Fehler nicht nur erkannt, sondern sogar korrigiert werden können. Dazu werden deutlich größere Mengen an Redundanzen benötigt als zur reinen Fehlererkennung. Diese Art von Codes heißt „Error Correcting Codes“.

Treten Fehler relativ selten auf, bietet sich eher ein reaktives Vorgehen an: wird ein Fehler erkannt, wird der zugehörige Block verworfen und noch einmal übertragen. Die Klasse dieser Verfahren wird „Automatic Repeat Request“ genannt und das simpelste Verfahren ist das Alternating-Bit-Protokoll.

Fehlererkennung und -korrektur: Redundanzen

- **Hinzufügen von Redundanzen:**
  - Länge des gesamten Blocks: n Bit (2^n mögliche „Code“wörter)
  - Länge der Nachricht (Header und Payload):
    m ($<n$) Bit (2^m mögliche Nachrichten ohne Redundanz)
    - Nachricht = Rahmen ohne Prüfbits
  - k Prüfbits ($k = n - m$)
- **Forme eine Sphäre von Codewörtern um jede erlaubte Nachricht**
  - *Hamming-Abstand* D:
    - Hamming-Abstand zweier Codewörter:
      - Anzahl der Positionen, in denen sich zwei Codewörter unterscheiden
    - Hamming-Abstand eines Codes:
      - Minimum der Hamming-Abstände aller möglichen Codewort-Paare
  - Dann:
    - $D \geq 2t + 1$ → Code kann t Fehler korrigieren
    - $D \geq t + 1$ → Code kann t Fehler erkennen

Die generellen Möglichkeiten, die man mit einem fehlererkennenden oder –korrigierenden Code erreichen kann, lassen sich mittels des Hamming-Abstands abschätzen. Als einfachstes Beispiel kann man sich die Betrachtung einzelner Bits hernehmen: gültige Nachrichten sind „0“ und „1“. Um einen Fehler erkennen zu können, benötigt man einen Hamming-Abstand von 2, daher codiert man die „0“ als „00“ und die „1“ als „11“. Unter der Annahme, dass lediglich ein einzelner Fehler auftreten kann, kann der Empfänger bei Erhalt einer „00“ eine „0“ decodieren, bei Erhalt einer „11“ eine „1“. Werden eine „01“ oder eine „10“ empfangen, kann der Empfänger von einem Bitfehler ausgehen, da es keine gültigen Codeworte sind. (Allerdings funktioniert dies immer nur unter der Annahme, dass nur ein einzelner Fehler aufgetreten ist!)

Zur Korrektur von Fehlern benötigt man mehr Redundanz: um einen Fehler zu korrigieren, benötigt man einen Hamming-Abstand von 3. Betrachtet man wieder einzelne Bits, kann man eine „0“ als „000“ und eine „1“ als „111“ codieren. Wird nur z.B. eine „110“ empfangen, geht man davon aus, dass nur eine Stelle verfälscht worden sein kann und korrigiert deswegen zu „111“. Man legt sozusagen eine Sphäre von Codeworten um jede erlaubte Folge und interpretiert „000“, „100“, „010“ und „001“ als „0“, „111“, „110“, „101“, „011“ als „1“.

Fehlererkennung: Paritätsüberprüfung

- **Paritätsüberprüfung**
  - Nur ein Bit Redundanz
  - Gerade oder ungerade Parität
  - Beispiel:

1101010	0110001	0001000	1000100	1101001	1110010	1101011

  - Parity-Bit bei gerader Parität: 1
  - Block inklusive Redundanz:

1101010	0110001	0001000	1000100	1101001	1110010	1101011	1

Die simpelste Form des Hinzufügens von Redundanz zur Fehlererkennung ist die Ergänzung der Daten um ein Paritätsbit (engl.: Parity Bit), welches mit zum Empfänger übertragen wird. Die Bildung des Parity-Bits lässt sich auf zweierlei Art und Weise vornehmen:

- *Gerade Parität*: Gesamtzahl der 1en einschließlich des Parity-Bits ist gerade.
- *Ungerade Parität*: Gesamtzahl der 1en einschließlich des Parity-Bits ist ungerade.

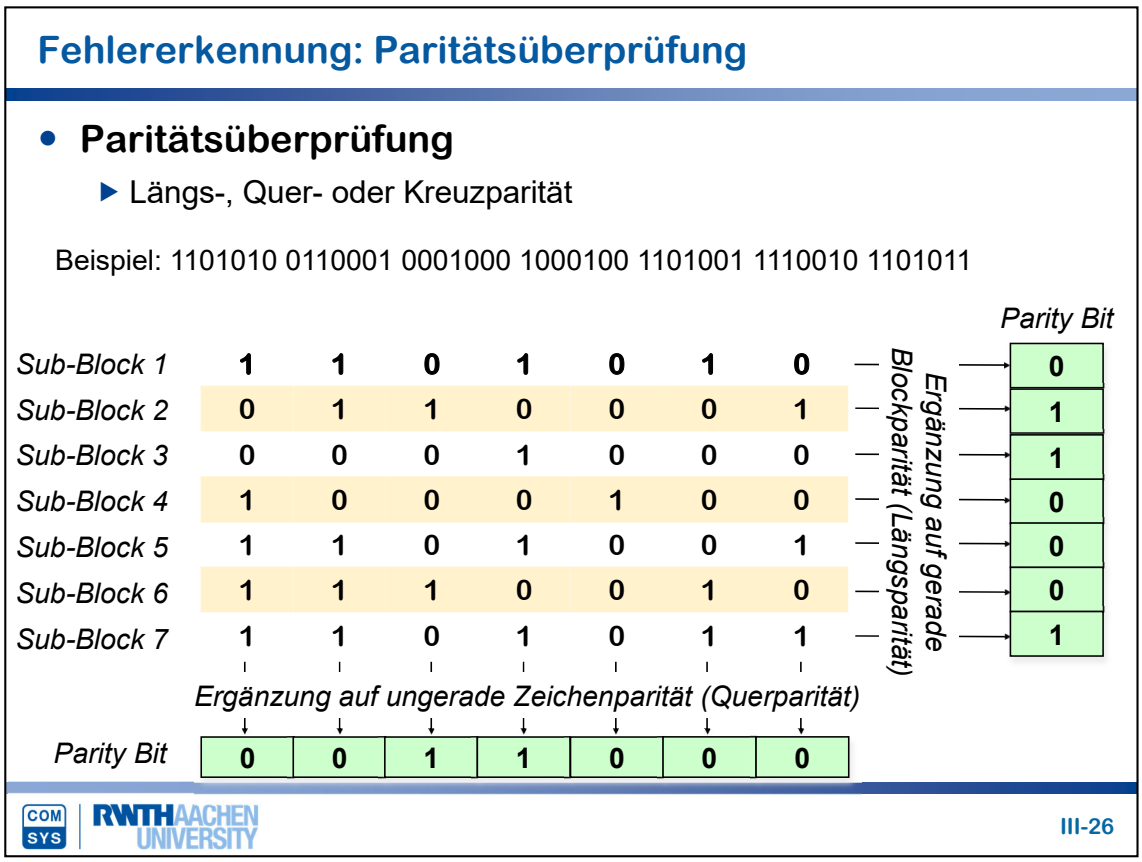
Im Allgemeinen findet gerade Parität verbreiteter Einsatz.

Durch dieses Verfahren kann die Verfälschung eines einzelnen Bits erkannt werden: unterscheiden sich zwei Nachrichten in nur einem Bit, haben sie einen Hamming-Abstand von 1. Durch das Hinzufügen des Parity-Bits wächst der Hamming-Abstand auf 2, so dass man einen einzelnen Fehler erkennen, aber keinen korrigieren kann.

Der Empfänger empfängt einen gesamten Rahmen (inklusive Parity-Bit) und berechnet selbst, welchen Wert das Parity-Bit aufgrund der empfangenen Bitwerte haben müsste. Diesen Wert vergleicht er mit dem empfangenen Parity-Bit-Wert. Sind die Werte gleich, geht der Empfänger von einem korrekten Empfang aus; sind die Werte unterschiedlich, hat der Empfänger einen Übertragungsfehler erkannt.

Allerdings kann der Fehler nicht korrigiert werden – dies kann man schon anhand des Hamming-Abstands sehen, sich aber auch intuitiv herleiten: egal, welches Bit verfälscht ist, das Parity-Bit hat den gleichen Wert. Es könnte sogar sein, dass alle „normalen“ Bits des Rahmens korrekt sind und das Parity-Bit verfälscht wurde. Zudem könnte es auch sein, dass 3, 5, 7, … Bits verfälscht sind – dies hätte die gleiche Auswirkung auf das Parity Bit.

Schlimmer ist allerdings, dass sich zwei Fehler gegenseitig aufheben und Fehler in einer gerade Anzahl von Bits nicht erkannt werden können.



Man kann zudem zwischen zwei Modi unterscheiden, die bei der Verwendung des Parity Bit zur Fehlererkennung eingesetzt werden können:

• *Block- oder Längsparität*: Berechnung der Parität über Blöcke wie auf der vorherigen Folie.

• *Zeichen- oder Querparität*: Sicherung von Einzelbits in aufeinanderfolgenden Blöcken. Bei Einsatz dieses Verfahrens muss man sich nur Gedanken machen, wie die Parity-Bits übertragen werden – eventuell als eigener Block.

Teilt man einen Block in Sub-Blöcke auf, ist auch mehr möglich – eine *Kreuzsicherung*: gleichzeitige Anwendung von Längs- und Querparität wie auf dieser Folie gezeigt. Über jeden Sub-Block wird die Längsparität berechnet (hier beispielhaft gerade Parität) und für jede Bitposition de Querparität über alle Sub-Blöcke (hier beispielhaft ungerade Parität).

Dies hat den Effekt, dass der Hamming-Abstand zwischen zwei Blöcken, die sich in nur einem Bit unterscheiden, auf drei anwächst - durch einen einzigen Bitfehler werden direkt zwei Paritätsbits verfälscht, die zusammen die Bestimmung der Position des Fehlers und somit seine Korrektur erlauben.

Allerdings führt auch hier das Auftreten von zwei oder mehr Fehlern zu Fehlinterpretationen.

Eine Paritätsüberprüfung ist also nur dann sinnvoll, wenn man davon ausgehen kann, dass nur Einzelbitfehler auftreten können. In der Datenkommunikation ist dies selten der Fall – verwendet werden Parity-Bits aber zum Beispiel bei der Speicherverwaltung (RAM).

Fehlererkennung: Cyclic Redundancy Check (CRC)

- **Verallgemeinerung der Paritätsüberprüfung**
  - Größere Zahl an Prüfbits, um mehr Fehler erkennen zu können
  - Einsatz von CRC zur Berechnung einer Prüfsumme über den Block (*Frame Check Sequence, FCS*)

- **Bildung der CRC-Prüfsumme:**
  1. Zu prüfende Bitfolge wird als Polynom aufgefasst
  2. Erweiterung um 0-Folge (Anzahl 0en = Grad des Prüfpolynoms), Teilung durch vereinbartes Prüfpolynom (Generatorpolynom)
  3. FCS ist Rest der Division, wird an die Bitfolge angehängt
  4. Beim Empfänger wird neu dividiert (einschließlich Rest), bei fehlerfreier Übertragung muss das Ergebnis 0 sein

Das gängige Verfahren in der Datenkommunikation ist der Cyclic Redundancy Check. Dieses Verfahren ist in der Lage, auch Fehler-Bursts sehr zuverlässig zu erkennen. Die zu übertragenden Daten (Protokoll- und Nutzdaten) können beliebige Länge haben; mittels CRC wird eine Frame Check Sequence (FCS) berechnet, an letzter Stelle in den Rahmen eingefügt und mit übertragen. Anhand der FCS kann der Empfänger ermitteln, ob während der Übertragung Fehler in den vorhergehenden Daten aufgetreten sind.

Für die Berechnung der FCS wird ein sogenanntes Generatorpolynom festgelegt. Der Sender fasst die Bitfolge der Daten als Polynom auf und teilt sie durch das Generatorpolynom (Modulo-2-Arithmetik). Der Rest, der bei der Division übrigbleibt, wird an die Bitfolge angehängt. Durch Anhängen des Rests an die Daten entsteht eine erweiterte Bitfolge (bzw. ein Polynom), welche Vielfaches des Generatorpolynoms ist.

Dadurch entsteht eine Art Codiervorschrift: eine übertragene Bitfolge ist nur dann gültig, wenn sie (als Polynom interpretiert) ein Vielfaches des Generatorpolynoms ist. Der Empfänger teilt die gesamte empfangene Bitfolge durch das gleiche Generatorpolynom. Bei korrektem Empfang darf bei der Division kein Rest bleiben.

Dieses Verfahren erkennt nicht nur mit sehr hoher Wahrscheinlichkeit fehlerhafte Datenblöcke, es ist auch einfach in Hardware zu implementieren, so dass die Berechnung der FCS effizient erfolgen kann und zu keiner spürbaren Verzögerung im Prozess des Versendes der Daten führt.

Fehlererkennung: CRC-Beispiel

- **Gegeben:**
  - Zu sendender Block: 110011
  - Generatorpolynom: $x^4 + x^3 + 1$
    - Polynom in Modulo-2-Binärarithmetik: 11001
- **Verfahren:**
  - Addition/Subtraktion Modulo-2 entspricht bitweiser XOR-Verknüpfung
  - Dividend ist teilbar durch Generatorpolynom, falls der Dividend mindestens so viele Stellen besitzt wie das Generatorpolynom (führendes Bit muss 1 sein)
  - Länge der FCS = Grad des Generatorpolynoms = 4

Fehlererkennung: CRC-Beispiel – Senden

- **Gegeben:**
  - Zu sendender Block: 110011
  - Generatorpolynom: 11001
- **Berechnung der FCS:**

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

- **Zu übertragender, erweiterter Block: 11 0011 *1001***

Fehlererkennung: CRC-Beispiel – Empfangen

- **Gegeben:**
  - Zu sendender Block: 110011
  - Generatorpolynom: 11001
- **Empfangen eines korrekten Blocks:**

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

- **Kein Rest**
  - Somit (mit sehr hoher Wahrscheinlichkeit) Daten fehlerfrei

Erhält der Empfänger bei Division durch das Generatorpolynom 0, geht er von einer fehlerfreien Übertragung aus.

Dies ist nicht notwendigerweise korrekt; wie bei der Paritätsüberprüfung gibt es auch hier Kombinationen von Fehlern, die nicht erkannt werden können.

Fehlererkennung: CRC-Beispiel – Empfangen

- **Gegeben:**
  - Zu sendender Block: 110011
  - Generatorpolynom: 11001
- **Empfangen eines verfälschten Blocks:**

```
111111 1000 ÷ 11001 = 101001
11001
001101 1
  1100 1
  0001 0000
     1 1001
     0 1001  = Rest ≠ 0
```

- **Rest ungleich 0, somit Fehler in Übertragung**

Fehlererkennung: CRC-Leistungsfähigkeit

- **International genormte Prüfpolynome:**
  - CRC-12 $= x^{12} + x^{11} + x^3 + x^2 + x + 1$
  - CRC-16 $= x^{16} + x^{15} + x^2 + 1$
  - CRC-CCITT $= x^{16} + x^{12} + x^5 + 1$
  - Ethernet: $= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

- **CRC-16 und CRC-CCITT entdecken**
  - alle Einzel- und Doppelfehler + alle Fehler ungerader Bitanzahl
  - alle Fehlerbursts der Länge ≤ 16
  - 99,997 % aller Fehlerbursts der Länge 17
  - 99,998 % aller Fehlerbursts der Länge ≥ 18

  → **Verbleibende Fehlerrate < 0.5 * 10^{-5} der BER**

Das CRC-Verfahren reduziert das Risiko, verfälschte Daten zu verarbeiten, kann aber nicht ganz ausschließen, dass verfälschte Daten als korrekt interpretiert werden. Die Wahrscheinlichkeit für so einen Fall ist allerdings extrem gering.

CRC wird nicht nur in der Datenkommunikation eingesetzt, sondern z.B. auch bei komprimierten Archiven.

Allerdings ist nur eine Erkennung von Fehlern möglich, keine Korrektur. Verfälschte Rahmen müssen verworfen werden, da man nicht feststellen kann, an welcher Stelle Fehler aufgetreten sind.

Vorwärtsfehlerkorrektur (Forward Error Correction)

- **Redundanz (durch Prüfsumme)**
  - Bislang nur zur Überprüfung der Daten
  - Jetzt: proaktive Fehlerbehandlung
    - Redundanz zur *Rekonstruktion verfälschter Datenblöcke*
- **Beispiel:**
  - Zu senden sind die Blöcke
    - 0101 – B1
    - 1111 – B2
    - 0000 – B3
  - Berechne vierten Block (XOR): 1010 – B4
  - Sende alle vier Blöcke an den Empfänger gesendet
    - Bei korrektem Erhalt dreier Blöcke kann der vierte berechnet werden
    - Schutz gegen Verlust eines ganzen Blockes

Für die Behebung von Übertragungsfehlern wird mehr Redundanz benötigt. Beispielsweise könnte eine zu übertragende Dateneinheit in Blöcke aufgeteilt werden, die mittels XOR verknüpft werden, um einen Prüfblock zu berechnen. Wird jeder dieser Blöcke zusätzlich mit einer CRC versehen, kann der Empfänger feststellen, welche der Blöcke korrekt übertragen wurden; wird einer der Blöcke als fehlerhaft identifiziert, kann er aus den anderen Blöcken rekonstruiert werden.

Forward Error Correction: Redundante Pakete

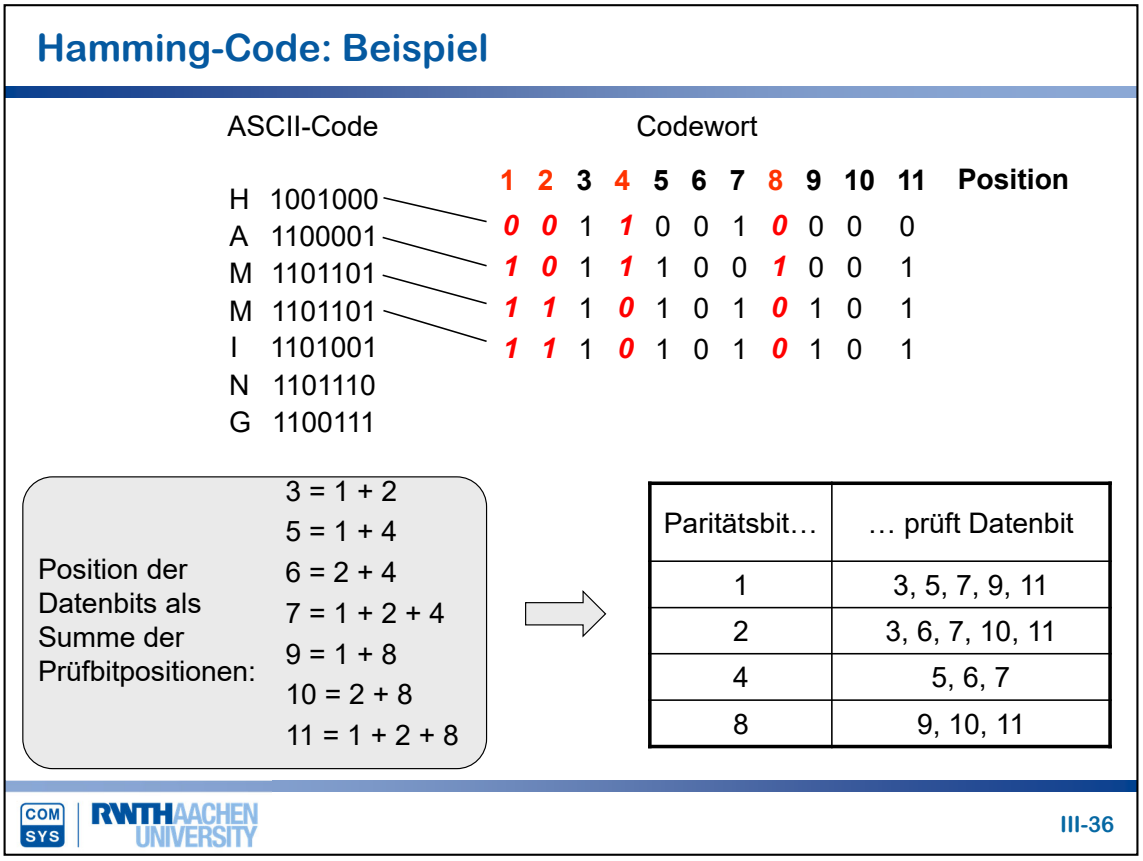
- **Über wie viele Blöcke *n* soll ein XOR-Block berechnet werden?**
  - Tradeoff: Erhöhe *n*…
    - … weniger Overhead durch seltenere XOR-Blöcke
    - … größere Verzögerung bis zur Korrektur, falls ein Fehler auftritt
    - … mehr Speicher zum Zwischenspeichern von Blöcken zur Korrektur
    - … größeres Risiko, dass zwei oder mehr Blöcke verfälscht sind

Hamming-Code

- **Idee:**
  - Erlaube die Korrektur von Einzelbitfehlern
    - Nötig: Hamming-Abstand von 3
  - Nutze dazu mehrere Paritätsbits, die jeweils (überlappend) mehrere Bits der Nachricht prüfen
    - Fehler können durch Kombination der Paritätsbits korrigiert werden
- **Realisierung eines „minimalen“ Codes**
  - Jede natürliche Zahl kann durch eine Summe von Zweierpotenzen ausgedrückt werden
  - Forme Codewörter $w = z_1, \ldots, z_n$ durch Einschieben der Paritätsbits an den Positionen, die eine Zweierpotenz sind
  - Jedes der Paritätsbits prüft alle anderen Positionen, zu deren Darstellung in Zweierpotenzen die Position des Paritätsbits nötig ist

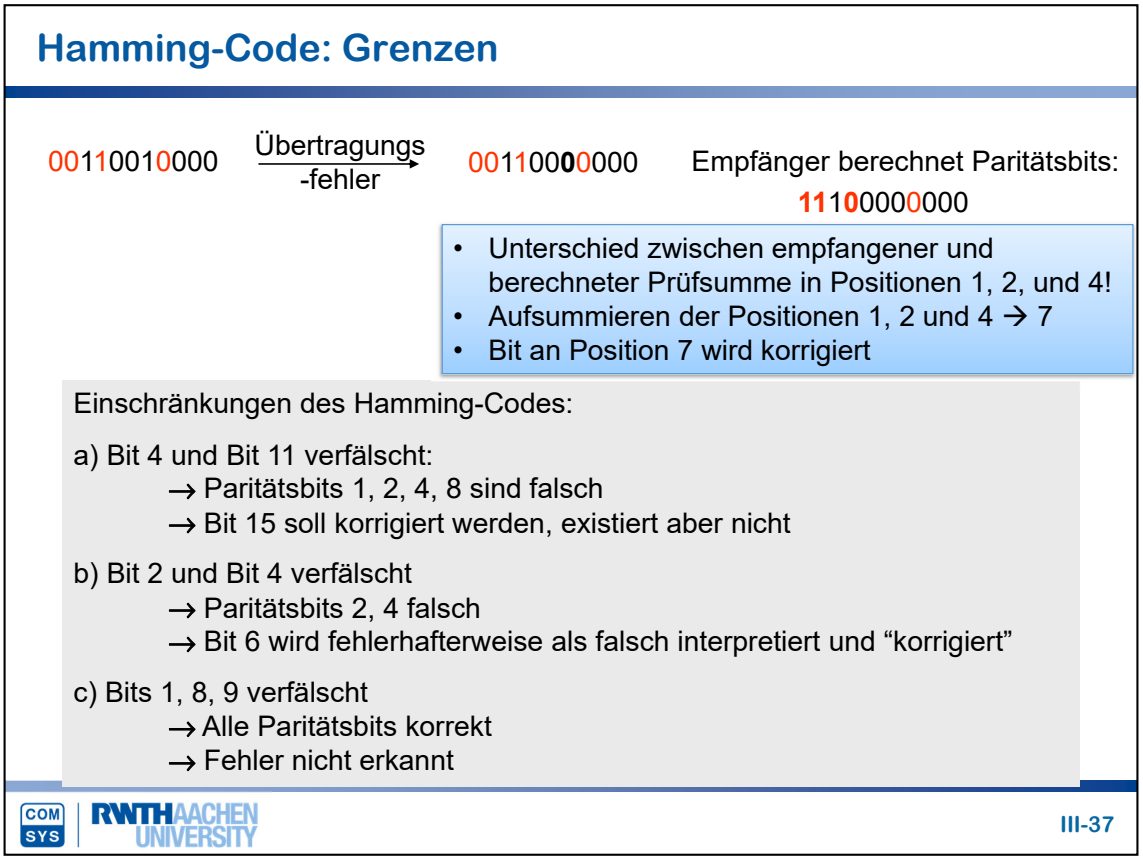
Anderer Ansatz: erzeuge keine redundanten Blöcke, sondern – wie bei CRC – pro Block eine Redundanz, die die Korrektur von Fehlern in diesem Block ermöglicht. In dieser Vorlesung wird nur die einfachste Variante betrachtet: der Hamming-Code, der einen Bitfehler innerhalb eines Blockes korrigieren kann. Dazu ist es notwendig, zwischen je zwei beliebigen, möglichen Blöcken einen Hamming-Abstand von mindestens drei zu erreichen.

Hierzu muss man Regeln festlegen, um zusätzliche Bits berechnen zu können, damit man einen minimalen Overhead erzielt.. Beim Hamming-Code wird hierzu eine Menge an Parity-Bits verwendet, die jeweils einen Teil der Nachricht prüfen, so dass jedes Datenbit von mindestens 2 Parity-Bits geprüft wird und keine zwei Datenbits von der gleichen Kombination an Parity-Bits geprüft werden.



Der Empfänger der Daten berechnet alle Paritätsbits und prüft, welche davon falsch sind. Die Positionen der falschen Paritätsbits werden aufsummiert und ergeben dadurch die Position des Bits, welches falsch ist und korrigiert werden muss.

Hamming hat gezeigt, dass diese Form der Berechnung von Prüfbits einen Hamming-Abstand von 3 erzeugt. Problem daher: dieser Code eignet sich aufgrund des Hamming-Abstands nur zur Korrektur von Einzelbitfehlern. Die Korrektur oder eventuell sogar die Erkennung mehrerer Fehler ist nicht möglich.



Der Hamming-Code eignet sich nur zur Korrektur von 1-Bit-Fehlern.

In der Realität (speziell in drahtlosen Netzen mit deutlich hoher BER) treten aber meist Fehler-Bursts auf. In der Praxis eingesetzte Verfahren sind daher deutlich komplexer und verlangen mehr Redundanz. Daher werden solche Verfahren auch nur dann eingesetzt, wenn die BER so hoch ist, dass ohne Fehlerkorrektur keine Kommunikation mehr möglich wäre (drahtlose Netze) oder wenn eine Fehlerkorrektur die einzige Möglichkeit ist, Fehler zu beheben (Broadcast-Netze wie DVB-T, Speichermedien wir DVD).

In kabelgebundenen Netzen wird meist nur auf die deutlich einfachere Fehlererkennung zurückgegriffen (zumal heutige Netze eine relativ geringe BER aufweisen). Im Fall eines erkannten Fehlers kann eine Behebung z.B. durch erneute Übertragung der verfälschten Daten erfolgen.

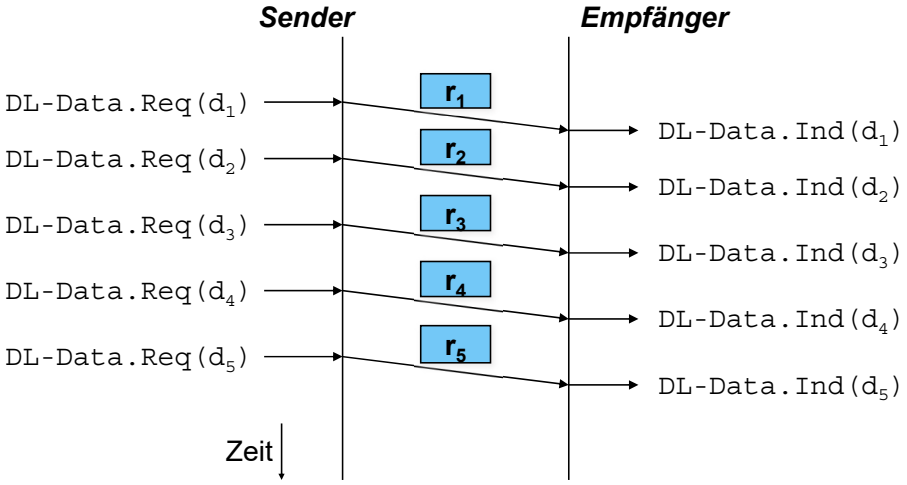
Fehlerbehandlung: Sendewiederholungsverfahren

- **Sendewiederholungsverfahren (*Automatic Repeat Request, ARQ*)**
  - Reaktive Fehlerbehandlung
  - Strategien:
    - Stop-and-Wait
    - Go-Back-N
    - Selective Repeat/Reject

Wird eine Fehlererkennung verwendet, müssen als verfälscht erkannte Rahmen verworfen werden. Die einzige Möglichkeit zur Fehlerbehebung ist eine Neuübertragung der Rahmen. Während Rahmen auf Empfängerseite verworfen werden, muss eine Sendewiederholung durch den Sender erfolgen. Daher sind Protokollmechanismen notwendig, durch die der Empfänger dem Sender signalisieren kann, welche Rahmen korrekt angekommen sind und welche neu übertragen werden müssen. Solche Mechanismen werden als „Automatic Repeat Request“ (ARQ) bezeichnet.

Übertragung ohne Fehlerbehandlung

- **Sender und Empfänger sind immer bereit**
  - Ständiger Datenfluss vom Sender zum Empfänger
  - Übertragungsfehler werden nicht berücksichtigt

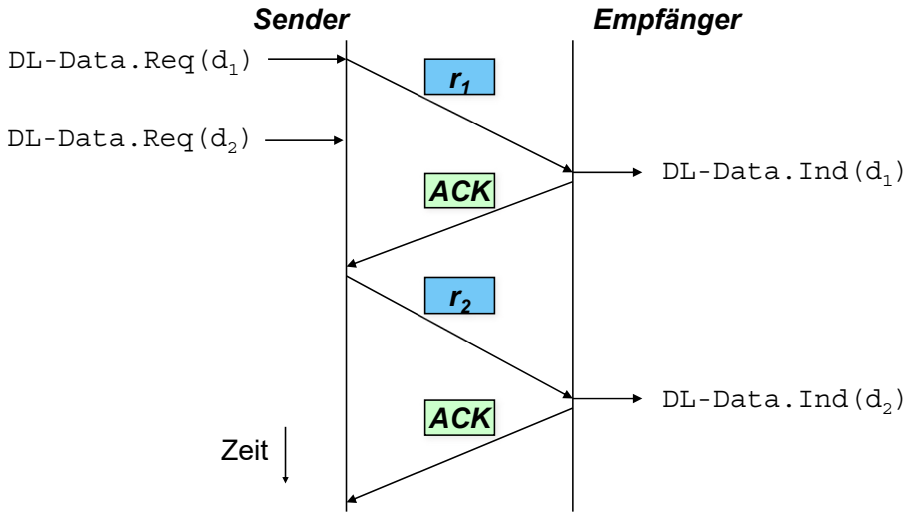


Protokolle auf Schicht 2 wären sehr einfach, wenn es keine Fehlersituationen gäbe. Können Fehler ignoriert werden (z.B. Audioinformationen) oder werden fehlerkorrigierende Codes eingesetzt, kann die Übertragung wie hier dargestellt erfolgen.

Notation: Auf Senderseite sollen Daten d_i übertragen werden; dazu werden sie an die Sicherungsschicht übergeben. Diese überträgt sie als Rahmen r_i – die Daten d_i zusammen mit z.B. einer CRC-Prüfsumme zur Fehlererkennung und eventuell weiteren Kontrollinformationen, die hier aber nicht von Bedeutung sind.

Fehlerbehandlung: Stop-and-Wait

- **Empfang eines Rahmens muss *bestätigt* werden**
  - Sender muss auf Bestätigung (*ACK*) warten, bevor er den nächsten Rahmen sendet



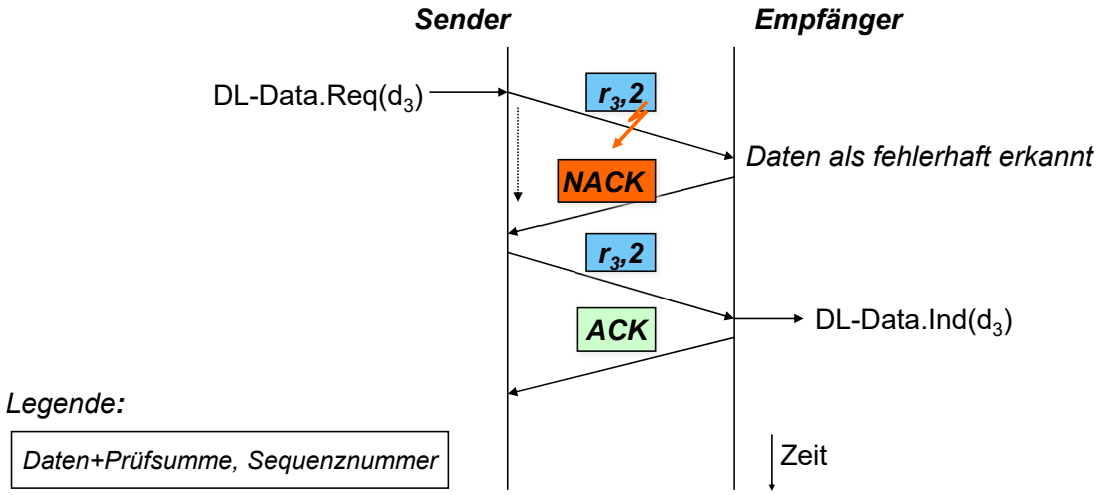
Die einfachste Möglichkeit der Fehlerbehandlung wird durch die Bestätigung des Empfangs realisiert. Beim *Stop-and-Wait-Verfahren* hat der Empfänger jede Dateneinheit explizit durch eine Kontrollnachricht ACK positiv (oder durch NACK negativ) zu bestätigen. (Dies ist im Wesentlichen das Alternating Bit Protocol aus Kapitel 1.) Die Bestätigung erfolgt nur schichtenintern, sie wird nicht an den Dienstnutzer weitergeleitet. Es handelt sich also um einen unbestätigten, aber zuverlässigen Dienst.

Als positiven Nebeneffekt der Quittierung kann man ansehen, dass der Empfänger nicht überlastet werden kann – bei unbestätigter Datenübertragung wie auf der vorherigen Folie könnte es vorkommen, dass der Empfänger die Rahmen nicht in der Geschwindigkeit verarbeiten kann, wie sie bei ihm ankommen und aufgrund dessen Daten verwerfen muss. Dies ist hier nicht mehr möglich.

Allerdings ist das Verfahren sehr ineffizient: einen großen Teil der Zeit verbringt der Sender mit dem Warten auf Bestätigungen, so dass die Kapazität der Leitung nicht ausgenutzt werden kann – insbesondere dann, wenn das Bandbreiten-Verzögerungs-Produkt groß ist, denn Bestätigungen erhält der Sender erst frühestens nach dem Doppelten der Latenz (plus Verarbeitungszeit des Rahmens durch den Empfänger und Sendezeit der Bestätigung, aber diese werden oft als verhältnismäßig klein angesehen und vernachlässigt).

Fehlerbehandlung: explizite Übertragungswiederholung

- **Explizite Übertragungswiederholung**
  - Fehlerhafte Rahmen explizit durch *NACK* (Negative Acknowledgement) anfordern



Anmerkung zur Legende: r_i bezeichnet den Rahmen, also Daten + Kontrollinformationen der Schicht. Trotzdem wird die Sequenznummer (die auch eine Kontrollinformation ist) hier explizit zusätzlich angegeben, um die Abläufe besser darstellen zu können.

Wie auf der vorherigen Folie erwähnt, steht dem Empfänger je nach Implementierung auf eine Kontrollnachricht „NACK“ (negative acknowledgement, in Protokollen auch oft kurz als „NAK“ bezeichnet) zur Verfügung, über die er den Empfang eines fehlerhaften Rahmens signalisieren kann. Der Sender reagiert auf den Erhalt dieser Kontrollnachricht durch eine Wiederholung des vorherigen Rahmens.

Sollte der Sender weder eine Bestätigung durch eine ACK-PDU erhalten, noch eine Aufforderung zur Übertragungswiederholung durch eine NACK-PDU, so sendet er nach einer bestimmten Zeitspanne auch in diesem Fall den Rahmen erneut – denn auch die ACK/NACK-PDU kann durch eine Störung verfälscht werden, so dass der Sender nicht weiß, ob sein Rahmen korrekt angekommen ist.

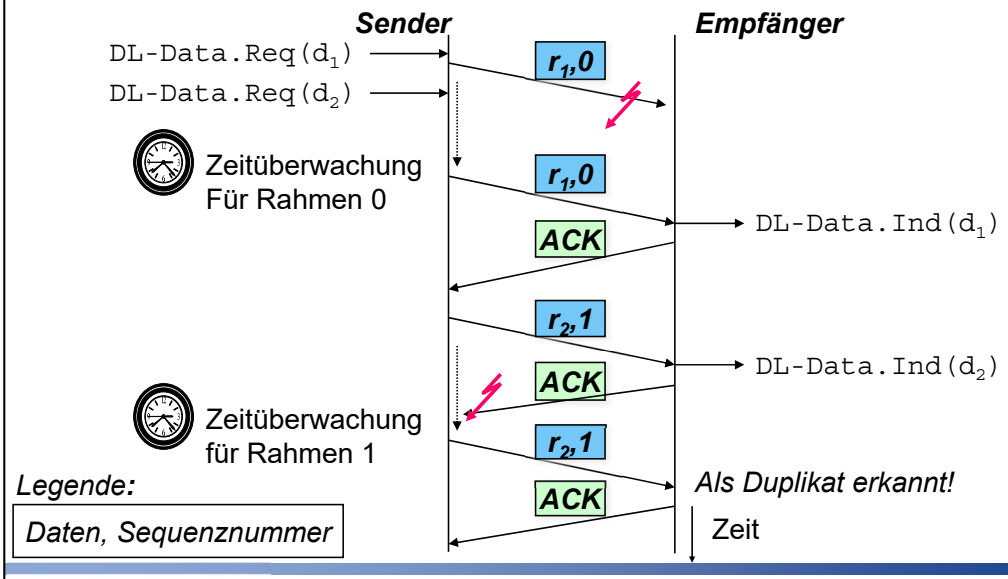
Wichtig ist hier zudem, dass der Sender bei der Übertragung eines Rahmens als weitere Kontrollinformation im Header eine Sequenznummer angibt, damit der Empfänger eindeutig entscheiden kann, ob der empfangene Rahmen auch der erwartete ist: angenommen, der Empfänger empfängt Rahmen 3 (hier mit der Sequenznummer 2) korrekt und sendet ein ACK zurück. Durch eine Störung wird das ACK verfälscht und der Sender kann sich nicht sicher sein, ob Rahmen 3 angekommen ist und überträgt ihn erneut. Ohne Sequenznummer würde der Empfänger denken, er würde nun Rahmen 4 empfangen, da er Rahmen 3 korrekt bestätigt hat. Er würde die Nutzdaten fälschlicherweise noch einmal verarbeiten. Anhand der Sequenznummer kann der Empfänger diese Situation erkennen.

Die Sequenznummern werden oft als N(S) bezeichnet; analog dazu können auch die Quittungen eine Bestätigungsnummer N(R) enthalten, um dem Sender mitzuteilen, auf welche Sequenznummer sich eine Quittung bezieht.

Bei Vollduplexbetrieb können sowohl N(S) als N(R) im Header eines Rahmens angegeben werden, um zugleich in eine Richtung Daten zu übertragen und aus der anderen Richtung erhaltene Daten zu quittieren (Piggybacking).

Fehlerbehandlung: implizite Übertragungswiederholung

- **Paketverlust**
  - Zeitüberwachung (*Timeout*) auf Senderseite & erneute Übertragung



Erhält der Sender nach einer bestimmten Zeitspanne (*Timeout*) keine Bestätigung für einen bestimmten Rahmen, so sendet er ihn erneut. Dieser Mechanismus ist (wie auf der letzten Folie erwähnt) notwendig, wenn negative Quittungen verwendet werden, kann aber darüber hinaus negative Quittungen auch ganz ersetzen: im Falle des korrekten Empfangs eines Rahmens sendet der Empfänger ein ACK, im Falle eines fehlerhaften Empfangs gibt er schlicht gar keine Rückmeldung. Die Erkennung eines Fehlers durch den Sender erfolgt somit nur dadurch, dass nach einer bestimmten Zeitspanne keine Antwort empfangen wurde – der Sender hat einen Timeout und überträgt den Rahmen neu.

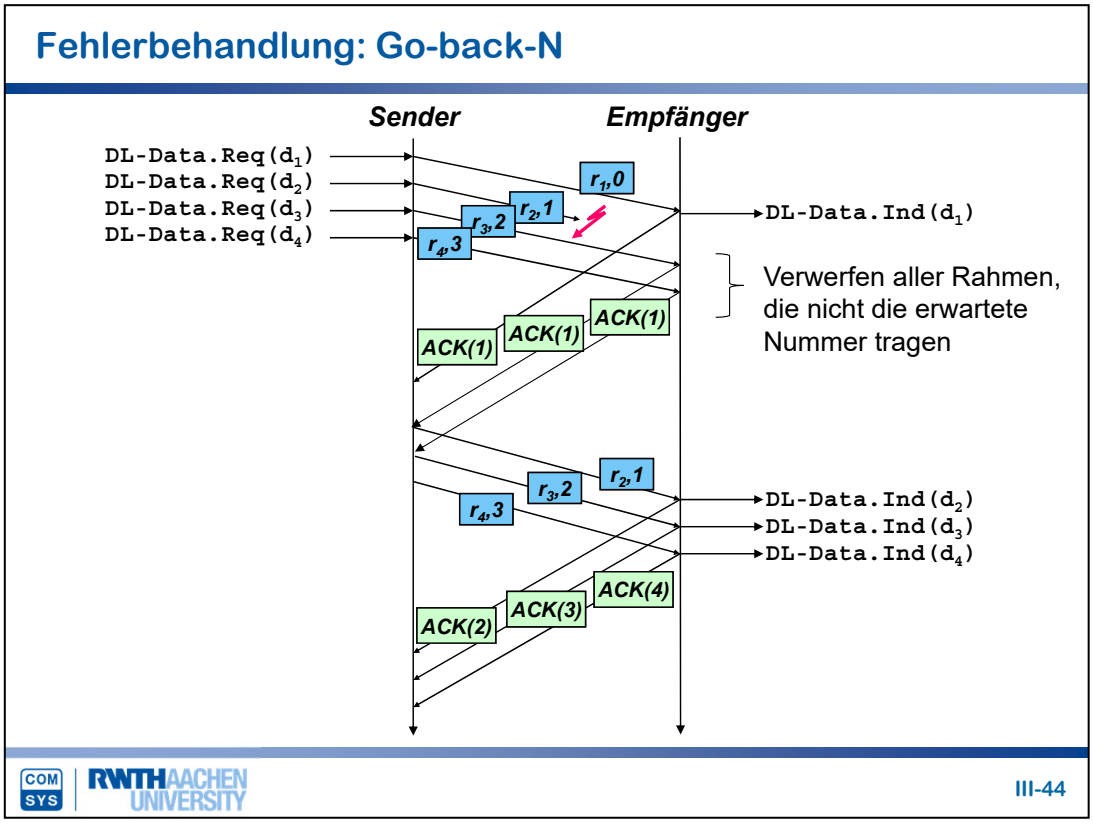
Die Größe des Timeouts ist sehr wichtig:

- Der Timeout-Wert muss natürlich zumindest zwei mal so groß sein wie die Latenz auf der Leitung (plus Verarbeitungs- und Sendezeit des Empfängers) – vorher kann keine Quittung eintreffen.
- Ein zu niedriger Timeout (nur knapp über der zweifachen Latenz) führt dazu, dass der Sender sehr schnell beginnt, Rahmen zu wiederholen. Es kann durch schwankende Latenzen oder Verarbeitungszeiten auf Empfängerseite allerdings durchaus vorkommen, dass kein Rahmenverlust auftrat, sondern ein Rahmen oder dessen Bestätigung schlicht etwas später ankommt. Somit wird ein Rahmen (sinnlos) mehrfach übertragen, was zur Verschwendung der Leitungskapazität führt.
- Ein zu hoher Timeout sorgt dafür, dass Verluste erst spät erkannt werden. Dies behindert die Kommunikation und führt zu einer Leistungssenkung.

Fehlerbehandlung: erweiterte ARQ-Verfahren

- **Stop-and-Wait**
  - Ineffizient: nach jedem Rahmen auf Quittung warten
    - Latenz zwischen Sender und Empfänger erzeugt lange Wartezeiten
    - Schlechte Ausnutzung der Leitung
  - Sinnvoller: erlaube Übertragung mehrerer Rahmen ohne auf sofortige Quittierung zu warten
- **Regelungen zur Quittierung bei mehreren Rahmen**
  - *Go-Back-N (Reject)-Verfahren*
    - Sämtliche Rahmen ab dem fehlerhaften müssen erneut übertragen werden
  - *Selective Repeat*
    - Nur der als fehlerhaft angegebene Rahmen muss wiederholt werden

Da Stop-and-Wait sehr ineffizient ist, erlaubt man dem Sender, mehrere mit Sequenznummern versehene Rahmen zu versenden, ohne auf eine Quittung warten zu müssen. Falls nun allerdings ein Rahmen verfälscht ankommt, muss geregelt werden, wie der Empfänger mit diesem und allen folgenden noch eintreffenden Rahmen verfährt. Die gängigsten Mechanismen sind Go-Back-N und Selective Repeat.



Go-back-N

Der Empfänger bestätigt dem Sender jeden korrekt erhaltenen Rahmen. Bitte beachten: üblicherweise wird in der Praxis der Erhalt eines ACK(i) verstanden als „Bis i-1 ist alles korrekt angekommen; der nächste Rahmen, den ich erwarte, ist derjenige mit Nummer i“.

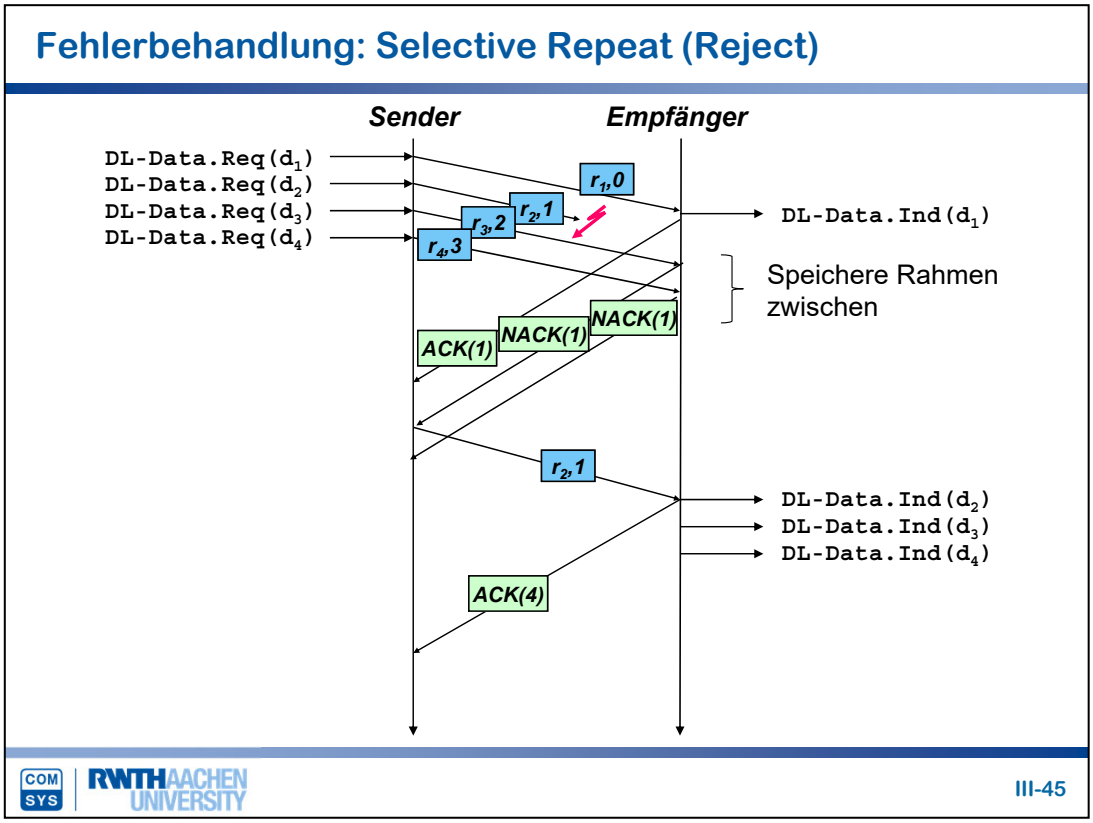
Wird ein Rahmen bei der Übertragung beschädigt, kann der Empfänger den Rahmen zwar (meist) als fehlerhaft erkennen, verwirft ihn aber. Es gibt verschiedene Varianten von Go-Back-N, je nach Quittungsverhalten des Empfängers in so einem Fall. Gemeinsam habe aber alle Varianten, dass der Sender alles ab dem fehlerhaften Rahmen neu übertragen muss.

Varianten:

1. Auf der Folie dargestellt: bei Eintreffen eines Rahmens mit nicht erwarteter Nummer wiederholt der Empfänger die vorherige Bestätigung. Der Sender erkennt einen Rahmenverlust anhand einer doppelten ACK-PDU und wiederholt alles ab dem im ACK genannten Rahmen.
2. Durch eine NACK-PDU wird der Sender angewiesen, die Übertragung ab einer gewissen Rahmennummer erneut zu starten. Dieses Verfahren ist bis auf das unterschiedliche PDU-Format (NACK statt ACK) identisch mit Verfahren 1 (NACK(1) statt ACK(1) bei Erhalt von r_3,2 im Beispiel auf der Folie).
3. Der Empfänger bestätigt ausschließlich korrekt empfangene Rahmen (in obiger Abbildung würde nur das erste ACK(1) gesendet, auf die nachfolgend eintreffenden Rahmen würde der Empfänger nicht reagieren). Der Sender erkennt Rahmenverluste nur durch Ablaufen eines Timeouts (fehlendes ACK) und muss nach dem zuletzt bestätigten Rahmen wieder aufsetzen.

Man sollte Go-back-N nur bei Leitungen mit einer geringen Latenz und/oder niedriger Datenrate einsetzen, damit nicht viele schon korrekt ausgelieferte Rahmen erneut übertragen werden müssen.

Des Weiteren bietet sich Go-Back-N an, wenn der Empfänger nicht viele Ressourcen zur Verfügung hat und daher nicht in der Lage ist, Rahmen zwischenzuspeichern, die nicht die erwartete Nummer tragen.



Selective Repeat

Bei diesem Verfahren wiederholt der Sender nur einzelne Rahmen. Allerdings muss der Empfänger die anderen Rahmen zwischenspeichern, um die Reihenfolgetreue zu erhalten (Empfängerbuffer wird benötigt).

Auch bei Selective Repeat gibt es zwei (bzw. drei) Ansätze:

1. *Explizite Übertragungswiederholung:* alle in korrekter Reihenfolge empfangenen Rahmen werden normal durch eine ACK-PDU bestätigt. Der Empfänger fordert fehlende/fehlerhafte Rahmen durch eine NACK-PDU erneut an. Diese Situation ist oben dargestellt. Wie bei Go-Back-N reagieren wir auf die negative Quittung mit einer Wiederholung, wiederholen allerdings nur den explizit als nicht korrekt empfangen gemeldeten Rahmen. Sobald dieser angekommen ist, schickt der Empfänger eine kumulative Quittung für alle bis zu diesem Zeitpunkt korrekt in Folge empfangenen Rahmen. (Wie auch bei Go-Back-N können auch positive Quittungen wiederholt werden, wenn keine NACK-PDU zur Verfügung steht. Es gibt also letztlich die gleichen drei Varianten wie bei Go-Back-N.)
2. *Implizite Übertragungswiederholung*: alle in korrekter Reihenfolge empfangenen Rahmen werden normal durch eine ACK-PDU bestätigt. Der Sender wiederholt nach einem Timeout für einen Rahmen nur diesen, der Sender schickt eine Quittung für diesen und alle folgenden bereits korrekt angekommenen Rahmen (kumulative Quittung). Bitte beachten: in obigem Beispiel würden unnötigerweise auch die Rahmen ab r_3 wiederholt werden, da es für diese kurz nach der Wiederholung von Rahmen r_2 auch Timeouts gibt. Wie viele Rahmen wir unnötigerweise wiederholen, hängt hier von der Latenz zwischen Sender und Empfänger ab, da die Wiederholungen abgebrochen werden können, sobald die kumulative Quittung eingeht. Daher ist die Verwendung von NACK-PDUs im Allgemeinen sinnvoller.

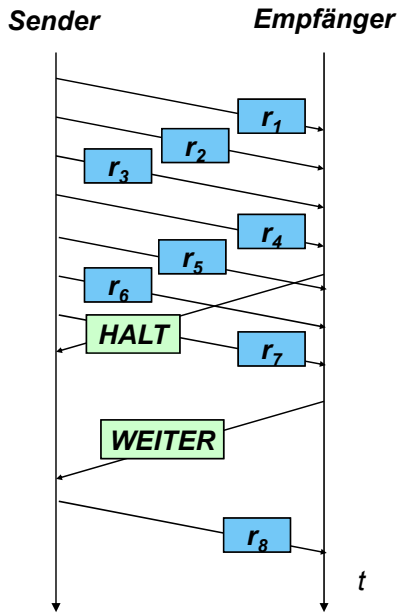
Flusskontrolle (Flow Control)

- **Synonyme Begriffe**
  - Flusssteuerung
  - Flussregulierung
- **Aufgabe**
  - Der Empfänger wird vor einem zu großen Zufluss von Rahmen eines Senders geschützt

Flusskontrolle (Vermeidung des Überlaufens des Buffers des Empfängers)
Schränkt man bei den vorherigen Verfahren nicht ein, wie viele unbestätigte Rahmen gleichzeitig unterwegs sein dürfen ohne quittiert worden zu sein, kann der Empfänger überlastet werden; auf Empfängerseite wird ein Buffer bereitgestellt, in dem empfangene Dateneinheiten zwischengespeichert werden, bis sie lokal verarbeitet werden können. Ist dieser Buffer voll, muss der Sender gezwungen werden, keine weiteren Rahmen mehr zu versenden, da diese beim Empfänger nicht zwischengespeichert werden können und man wiederum Rahmenverluste hat. Daher fügt man eine Beschränkung ein, um den Sender stoppen zu können. Dies nennt man Flusskontrolle oder auch Flusssteuerung: der Empfänger steuert, wie viel der Sender verschicken darf.

Flusskontrolle mit Halt-/Weiter-Meldungen

- **Einfachste Methode**
  - Sender-Empfänger-Flusssteuerung
    - Meldungen
      - Halt
      - Weiter



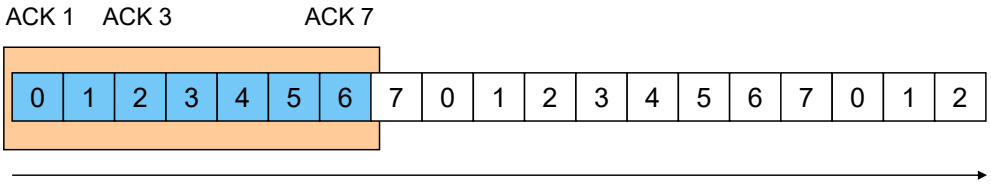
Die einfachste Methode der Flusskontrolle basiert auf zwei Meldungen:

• Mit der HALT-Meldung zeigt der Empfänger an, dass er nicht mit dem Sender Schritt halten kann und der Buffer volläuft. Diese Meldung sollte nicht zu spät gesendet werden, da der Sender bis zur Ankunft der HALT-Meldung noch weiter Rahmen senden kann.

• Mit der WEITER-Meldung wird später signalisiert, dass der Empfänger genügend Rahmen verarbeitet hat und so neue Rahmen annehmen kann.

Flusskontrolle: Sliding Window

- **Verwendung eines „Sendefensters“**
  - Sender und Empfänger vereinbaren ein *Übertragungsfenster* (entspricht max. Größe des Pufferspeichers des Empfängers)
    - Sender nummeriert die zu versendenden Rahmen des Datenstroms im Bereich 0, 1, 2, ..., MODULUS-1, 0, … mit W < MODULUS durch
    - Fenstergröße W bedeutet: der Sender darf maximal W fortlaufend nummerierte Rahmen verschicken, ohne eine Bestätigung zu bekommen
    - Empfänger bestätigt empfangene Rahmen durch Quittungen (ACK)
    - Sender rückt Fenster vor, sobald ein ACK eintrifft, und darf neue Rahmen verschicken



COM SYS | RWTH AACHEN UNIVERSITY http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/ III-48

Das gängige Verfahren zur Flusskontrolle ist Sliding Window: lege ein „Fenster“ über den zu versendenden Datenstrom, durch das der Sender nur einen Ausschnitt der Daten sehen kann. Nur diese Daten dürfen versendet werden, bevor der Sender auf eine Bestätigung warten muss. Die Menge der Rahmen, die ein Sender senden darf, ohne auf eine Bestätigung warten zu müssen, nennt man Fenstergröße. Die Fenstergröße sollte vom Empfänger eingestellt werden können, um dessen Buffergröße zu repräsentieren. Dadurch kann der Sender Netzkapazität wann immer möglich ausnutzen, während der Empfängerbuffer nicht überlastet wird.

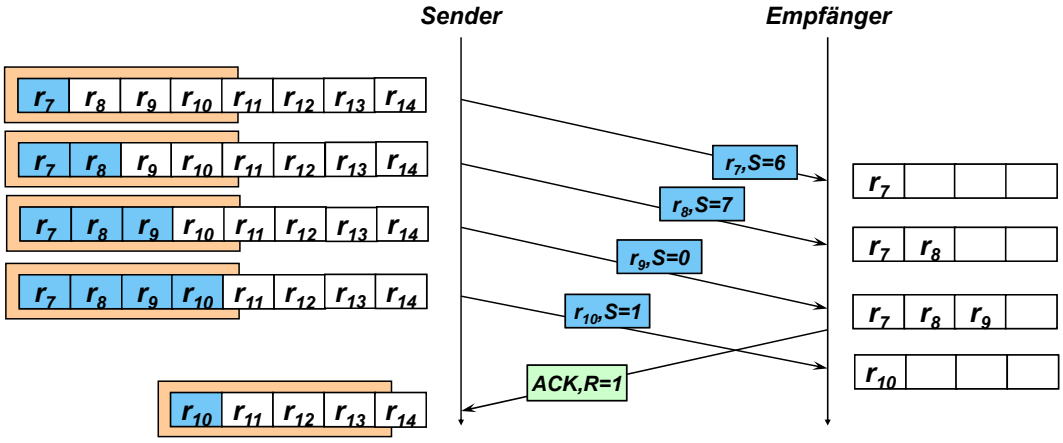
Ist die Fenstergröße statisch vorgegeben, kann der Empfänger lediglich Bestätigung für Rahmen verzögern, um nicht überlastet zu werden. Es ist jedoch nicht vorteilhaft, wenn während dieser Verzögerung der Timer beim Sender abläuft und dieser seine Rahmen wiederholt, obwohl sie beim Empfänger schon korrekt ankamen. Für solche Fälle gibt es bei manchen Protokollen eine Meldung wie RECEIVE NOT READY, welche die letzten Rahmen bestätigt, aber mitteilt, dass der Empfänger noch eine Weile braucht, um seine Buffer zu leeren.

Die Wahl der Fenstergröße wird dadurch beschränkt, dass im Header des Rahmens nur eine bestimmte Zahl n an Bit für die Sendefolgennummern spendiert werden und es daher nur 2^n unterschiedliche Sendefolgenummern gibt. Wird das Fenster zu groß gewählt, können zwei unbestätigte Rahmen mit gleicher Nummer unterwegs sein, so dass Quittungen nicht mehr eindeutig wären.

Sliding Window wird auch kreditbasierte Flusskontrolle genannt: der Sender hat Kredit für die Versendung einer bestimmten Zahl von Rahmen, bevor er auf eine Quittung warten muss.

Sliding Window

- **Sliding Window mit Fenstergröße 4 für eine Senderichtung**



S: Sequenznummer (des zuletzt gesendeten Rahmens)
R: Nächste erwartete Sequenznummer = Quittierung bis Folgenummer R-1

Beispiel: HDLC-Protokoll

- **High-Level Data Link Control (HDLC)**
  - Bit-orientiertes Sicherungsschichtprotokoll
    - Anwendungszenario: Anschluss an Datennetze über Telefonnetze (X.25)
  - Funktionen:
    - Framing durch spezielle Bitkombination, Codetransparenz durch Bitstuffing in Payload und CRC-Prüfsumme
    - Fehlerkorrektur durch CRC + ARQ, Piggybacking von Quittungen
    - Flusskontrolle (Sliding Window)
  - Varianten zum HDLC-Protokoll:
    - LAPD (ISDN, Link Access Procedure, D-Kanal)
    - *LLC (IEEE 802.2, Logical Link Control)*
    - *PPP (Point-to-Point Protocol)*

Auf den folgenden Folien wird das HDLC-Protokoll vorgestellt, welches sehr simpel ist, aber alle bisher vorgestellten Aspekte der Sicherungsschicht implementiert und sich daher gut eignet, die Umsetzung zu erklären.

HDLC kam als ein Standardprotokoll in X.25-Netzen zum Einsatz, die ursprünglich von Telefonnetzbetreibern verwendet wurden, um Datenaustausch über Telefonnetze zu ermöglichen. X.25 wurde nach und nach von Frame Relay abgelöst, dieses dann von ATM und dieses wiederum von SDH – so dass HDLC heute vermutlich kaum noch im Einsatz ist. Varianten dieses Protokolls haben sich allerdings gehalten – beispielsweise als PPP, welches Internetprovider für die Einwahl der Kunden verwenden. Mit Hilfe von PPP teilt der Provider dem Rechner/Router des Kunden wichtige Daten mit, z. B. dessen IP-Adresse oder den zu verwendenden DNS-Server. Zunächst wurde PPP über Modem- oder ISDN-Verbindungen verwendet, heutzutage auch zum Datenaustausch über GPRS-/UMTS-Mobilfunkdatenverbindungen oder DSL-Verbindungen.

HDLC: Medienzugriff

- **HDLC baut auf einem Befehl/Antwort-Schema auf:**
  - *Leitsteuerung*: Aussenden eines Befehls
  - *Folgesteuerung*: Reaktion (Meldung) auf eine Leitsteuerung
- **Wer kann Leitsteuerung aussenden?**
  - HDLC definiert Leitstationen und Folgestationen
  - Asymmetrischer Fall mit zentraler Steuerung
    - Eine feste Leitstation, ein oder mehrere Folgestationen
    - Leitstation versendet Empfangsaufruf und Sendeaufrufe, kein spontaner Zugriff einer Folgestation auf das Medium
  - Symmetrischer Fall mit gleichberechtigten Stationen
    - Hybridstationen alternieren als Leit- und Folgestationen

(Ausgeblendete Folie – nur informativ, nicht wichtig für Vorlesung)

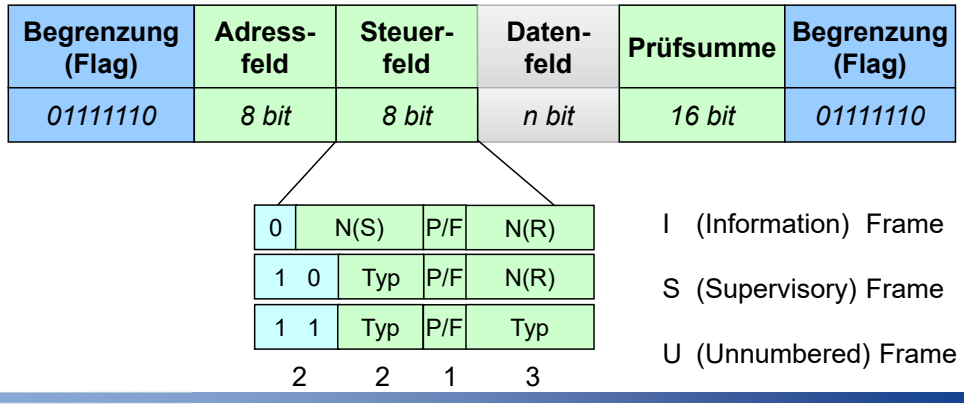
Der einzige bei HDLC implementierte Mechanismus, der noch nicht zuvor erläutert wurde, ist der Medienzugriff. Auch dieser ist bei HDLC einfach gehalten, bietet aber unterschiedliche Varianten. Generell basiert der Medienzugriff allerdings auf einer Koordination durch eine zentrale Station, die Leitsteuerung genannt wird. Alle anderen Stationen sind Folgestationen, die lediglich auf Sendeaufforderungen der Leitsteuerung reagieren (Polling).

HDLC erlaubt mehrere Arbeitsbetriebsarten:

- *Aufforderungsbetrieb* (NRM — Normal Response Mode): entspricht Polling: eine Folgestation darf nur nach ausdrücklicher Erlaubnis durch die Leitstation Meldungen senden.
- *Spontanbetrieb* (ARM — Asynchronous Response Mode): eine Folgestation kann jederzeit Meldungen an Leitstation senden.
- *Gleichberechtigter Spontanbetrieb* (ABM — Asynchronous Balanced Mode): Beide Hybridstationen dürfen jederzeit Meldungen und Befehle übermitteln.

HDLC: Rahmenaufbau

- **Eigenschaften:**
  - Synchronisation durch Rahmenbegrenzer *01111110*
    - Sequenz darf nicht im Rahmen vorkommen → Bitstuffing
  - Quittierung, Flusskontrolle → Sequenz-/Bestätigungsnummern
  - Fehlererkennung durch CRC → Prüfsumme



Zum Austausch von Daten definiert HDLC ein einheitliches Rahmenformat sowohl für den Datenaustausch als auch für Kontrollkommandos. HDLC unterscheidet dazu drei unterschiedliche Rahmentypen, die sich anhand der ersten Bits des Steuerfeldes unterscheiden lassen:

- Information (I-Frames): Datenübertragung
- Supervisory (S-Frames): Fehler-/Flusskontrolle
- Unnumbered (U-Frames): Verbindungsauf-/-abbau

Neben den Sequenz-/Quittungsnummern N(S) und N(R) gibt es ein Typ-Feld, welches verschiedene Steuerbefehle unterscheidet (z.B. RR - Receive Ready) sowie das Poll/Final-Bit (P/F).

Da Rahmenbegrenzungen verwendet werden, muss auch Bitstuffig eingesetzt werden: dazu wird nach einer Folge von fünf Einsen immer eine Null eingefügt.

HDLC: Aufbau des Steuerfeldes

Steuerfeldformat für	Bit-Nummer 1	2	3	4	5	6	7	8
I-Rahmen (Nutzdatenrahmen) [I=Information]	0	N(S)			P/F	N(R)		
S-Rahmen (Steuerrahmen) [S=Supervisory]	1	0	S	S	P/F	N(R)		
U-Rahmen (Steuerrahmen) [U=Unnumbered]	1	1	M	M	P/F	M	M	M

- **I-Block:** Übertragung von Nutzdaten, *N(S)* ist aktuelle Rahmennummer
  Piggybacking von Quittungen über N(R) möglich
- **S-Block:** Steuerrahmen mit Quittungsnummer *N(R)*
  S-Bits des Typ-Felds spezifizieren den Rahmen, verschiedene Bestätigungen sowie Flusskontrolle implementiert
- **U-Block:** Steuerrahmen ohne Nummer
- **P/F Bit:** Übergibt Sendeberechtigung zwischen Leit- und Folgestationen

Das P/F-Bit dient dazu, einer anderen Station anzuzeigen, dass die Datenübertragung in ihre Richtung beendet ist. Grund hierfür ist, dass Rahmen nicht beliebig lang sein sollten – ein Bitfehler innerhalb eines Rahmens führt dazu, dass der Rahmen vom Empfänger verworfen wird, weshalb es in Abhängigkeit von der BER eines Mediums sinnvoll ist, die Nutzdaten auf mehrere Rahmen aufzuteilen. Sendet die Leitsteuerung nun eine Folgen von Rahmen an eine Folgestation, setzt sie im letzten Rahmen das P/F-Bit auf 1. Dies hat für die Folgestation die Bedeutung „Poll“, d.h. sie darf nun senden. Auch sie kann nun mehrere Rahmen an die Leitsteuerung senden und setzt im letzten Rahmen das P/F-Bit auf 1, was für die Leitsteuerung die Bedeutung „Final“ hat, d.h. nun ist sie wieder an der Reihe.

HDLC: Rahmentypen

Type	Name	Fields 1	2	3	4	5	6	7	8	
I	**I** (Data Frame)	0	N(S)			P	N(R)			Sende Rahmen *N(S)*, bestätige *N(R)* -1 in die andere Richtung (Piggybacking)
S	**RR** (Receive Ready)	1	0	0	0	P/F	N(R)			Signalisierung "empfangsbereit", bestätige *N(R)* -1 in die andere Richtung (ACK, WEITER)
	RNR (Receive not Ready)	1	0	1	0	P/F	N(R)			Signalisiere "temporär nicht empfangsbereit", bestätige *N(R)* -1 in die andere Richtung (HALT)
	REJ (Reject)	1	0	0	1	P/F	N(R)			NACK für *N(R)*, ACK bis *N(R)*-1. Anforderung einer Übertragungswiederholung ab *N(R)* (Go-back-N)
	SREJ (Selective Reject)	1	0	1	1	P/F	N(R)			ACK bis *N(R)*-1, NACK nur für *N(R)*
U	**SABM** (Set Asynchronous Balanced Mode)	1	1	1	1	P	1	0	0	Verbindungsaufbau
	DISC (Disconnect)	1	1	0	0	P	0	1	0	Ankündigung eines Verbindungsabbaus
	UA (Unnumbered ACK)	1	1	0	0	F	1	1	0	Generelles ACK (z.B. im Verbindungsaufbau)
	CMDR (Command Reject)	1	1	1	0	F	0	0	1	Rahmen/Kommado ungültig (falsches Rahmenformat, ungültige Sequenznummer, …)
	FRMR (Frame Reject)	1	1	1	0	F	0	0	1	
	DM (Disconnect Mode)	1	1	1	1	F	0	0	0	Verbindungsabbau

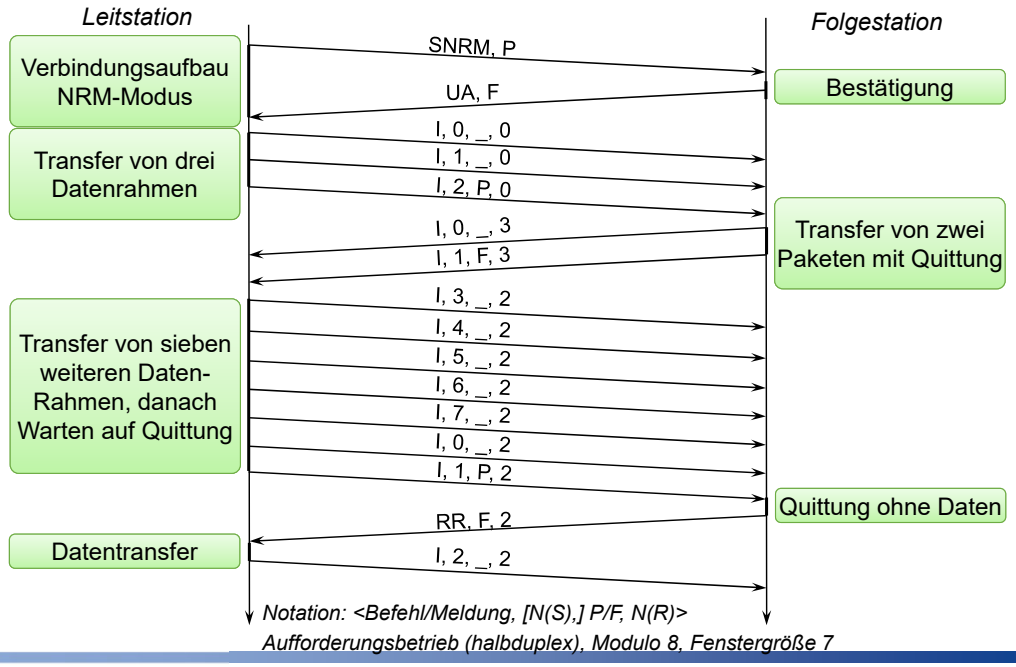
Bitte beachten: die Liste der U-Frames ist nicht vollständig, sondern soll einfach ein paar Beispiele zeigen, was für Kommandos gegeben werden können.

Siehe auch http://de.wikipedia.org/wiki/High-Level_Data_Link_Control.

Wir sehen hier ein paar Mechanismen, die zuvor erläutert wurden:

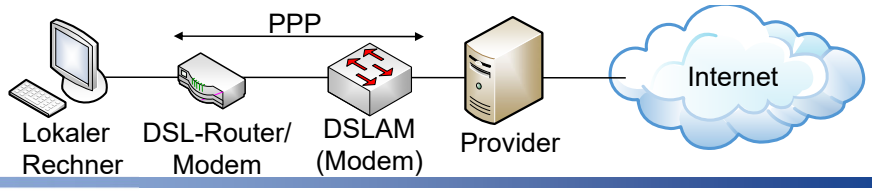
- Quittungsmechanismus mit ACK und NACK
- RNR entspricht einer „HALT-Meldung“

HDLC: Beispielablauf



PPP (Point-to-Point Protocol)

- **Großteil des Internets: Punkt-zu-Punkt Verbindungen**
  - Verbindungen im WAN zwischen Routern
  - Heimanbindung über Modem und Telefonleitung (DSL)
- ***PPP* (Vereinfachte Version von HDLC)**
  - Effiziente Anpassung an verschiedene PHYs + unterstützt diverse Schicht-3-Protokolle und deren Optionsverhandlung
    - Steuerprotokoll (LCP, Link Control Protocol) zum Verbindungsaufbau, Verbindungstest, Verbindungsverhandlung, Verbindungsabbau
    - Verhandlung von Schicht-3-Optionen unabhängig vom Schicht-3-Protokoll (separates Network Control Protocol, NCP)



COM SYS | RWTH AACHEN UNIVERSITY

III-56

„Einloggen“ über Modem bzw. Terminal-Emulation auf einem Server ergibt lediglich (textorientierte) Terminal-Funktionalität und eröffnet nicht die gesamte Anwendungsbreite des Internets (WWW, FTP etc.), da der eigene Rechner nicht als vollwertiger Internet-Host auftritt. Dies wurde erst durch die Protokolle SLIP (Serial Line IP, standardisiert als RFC 1055) und PPP (Point-to-Point Protocol, standardisiert als RFC 1661/1662) ermöglicht, die die transparente Übertragung von IP-Paketen erlauben.

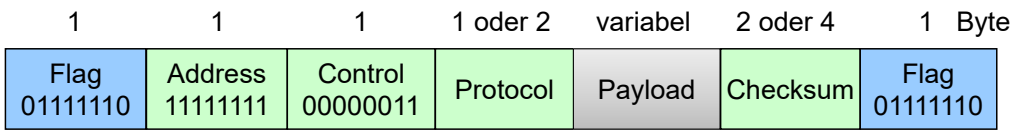
Das aus den 80er Jahren stammende SLIP hatte einige Nachteile (keine Fehlererkennung, nur IP, keine dynamische Adresszuweisung, keine Authentifikation), so dass es 1994 durch das leistungsfähigere PPP ersetzt wurde.

Typisches Szenario beim Zugriff eines PCs auf das Internet via Modem

- Anruf beim Internetprovider via Modem und Aufbau einer physikalischen Verbindung
- Anrufer sendet mehrere LCP-Pakete im PPP-Rahmen zur Auswahl der gewünschten PPP-Parameter, Authentifizierung
- Austausch von NCP-Paketen, um Vermittlungsschicht zu konfigurieren; z.B. kann hier dynamisch eine IP-Adresse zugewiesen werden
- Der Anrufer kann nun genauso wie ein festverbundener Rechner Internet-Dienste nutzen
- Zur Beendigung der Verbindung wird via NCP die IP-Adresse wieder freigegeben und die Vermittlungsschichtverbindung abgebaut
- Über LCP wird die Schicht 2-Verbindung beendet, schließlich trennt das Modem die physikalische Verbindung

PPP – Rahmenformat

- **Rahmenformat an HDLC angelehnt**



- Zeichenorientiert (Länge des Payload endet immer an Byte-Grenze)
- Codetransparenz durch Character Stuffing
- Typischerweise nur *Unnumbered*-Frames, bei hohen Fehlerraten (Mobilkommunikation) kann jedoch auch der zuverlässigere Modus mit Sequenznummern und Bestätigungen gewählt werden
- „Protocol“ definiert Art des Payloads, z.B. IPv4, IPv6
- Falls nicht anderweitig verhandelt, ist die maximale Länge der Nutzlast auf 1500 Byte begrenzt

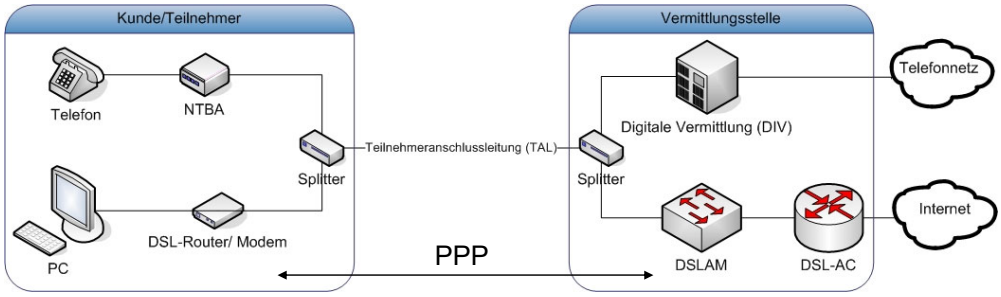
PPP hat im wesentlichen das Rahmenformat von HDLC übernommen (Kompatibilität). Es werden die gleichen Flags zur Kennzeichnung des Rahmenbeginns und -endes verwendet. Das Adressfeld ist immer auf 255 gesetzt, das Steuerfeld (Control) hat üblicherweise den Wert 3 (unnummerierte Datenübertragung). Die Checksumme kann wie HDLC 2 Byte lang sein, ist heutzutage aber 4 Byte lang (wie bei allen modernen Protokollen).

Zudem wird hinter dem Steuerfeld ein Feld eingefügt, welches dem Empfänger anzeigt, welches höhere Protokoll im Datenteil verwendet wird, damit der Empfänger weiß, mit welcher Protokollinstanz er die Daten weiterverarbeiten muss.

Durch zusätzliche Verhandlung kann der Header verkleinert werden.

PPP und DSL: Systemaufbau

- **Komponenten des DSL-Systems**
  - *DSL-Modem* für alle Unterkanäle
  - *Splitter* zur Trennung des Telefonkanals (bis 120 kHz) vom Datenkanalbereich (ab 138 kHz)
  - *DSLAM* (Digital Subscriber Line Access Multiplexer) als Modembank zum Anschluss mehrerer Haushalte beim DSL-Anbieter, misst Kanalqualitäten



http://upload.wikimedia.org/wikipedia/commons/9/92/DSL.jpg

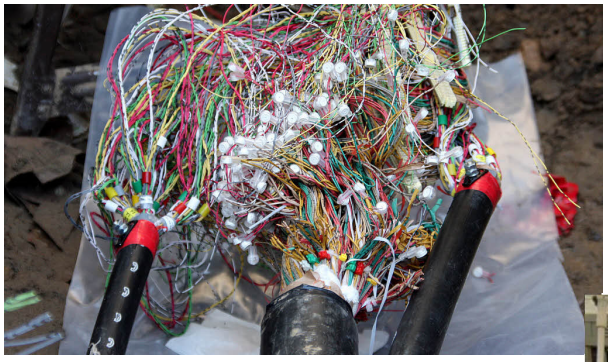
Das DSL-Grundprinzip ist das gleiche wie beim ursprünglichen Modem. Auf Seiten des Heimnetzes wird das DSL-Modem die zu übertragenen Daten auf die verwendbaren Unterkanäle aufteilen und modulieren. Auf der Seite des Providers existiert für jeden Haushalt eine Gegenstelle, d.h. auch ein Modem. Die Gesamtheit der Modems formt den DSLAM, der die Daten aller Haushalte demoduliert und auf eine gemeinsame Leitung zur Weiterleitung ins Internet multiplext.

Um parallel Telefonie über die Anschlussleitung zu ermöglichen, wurde der untere Bereich des Frequenzbands freigelassen (POTS/ISDN). Ein sogenannter Splitter trennt bzw. vereint die nieder- und die hochfrequenten Anteile der übertragenen Signale.

Dieser Splitter ist heutzutage meist nicht mehr nötig, da die Sprachübertragung für das Telefon bei den Providern auch über IP läuft und die Sprach-Pakete wie alle anderen Datenpakete an das Modem übertragen und erst dort getrennt werden.

Um einen Kommunikationskanal zwischen den Modems auf beiden Seiten einzurichten, wird PPP eingesetzt.

Bilder Telefon”verkabelung” und DSLAM



Bilder Telefon”verkabelung” in New Delhi

DSL-Varianten

- **Symmetrische Varianten**
  - High Data Rate Digital Subscriber Line (HDSL)
  - Symmetric Digital Subscriber Line (SDSL)
- ***Asymmetrische Varianten***
  - Asymmetric Digital Subscriber Line (ADSL)
    - Frequenzband bis 1,1 MHz (Up: 1 Mbit/s, Down: 8 Mbit/s)
  - ADSL2/ADSL2+
    - Frequenzband bis 2,2 MHz (Up: 1 Mbit/s, Down: 24 Mbit/s)
  - Very High Data Rate Digital Subscriber Line (VDSL)
    - Frequenzband bis 12 MHz (Up: 3,5 Mbit/s, Down: 25 Mbit/s)
  - VDSL2
    - Frequenzband bis 30 MHz (Up/Down: bis zu 200 Mbit/s)

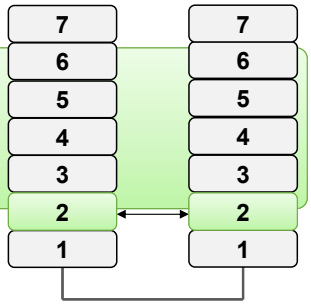
(Ausgeblendete Folie)

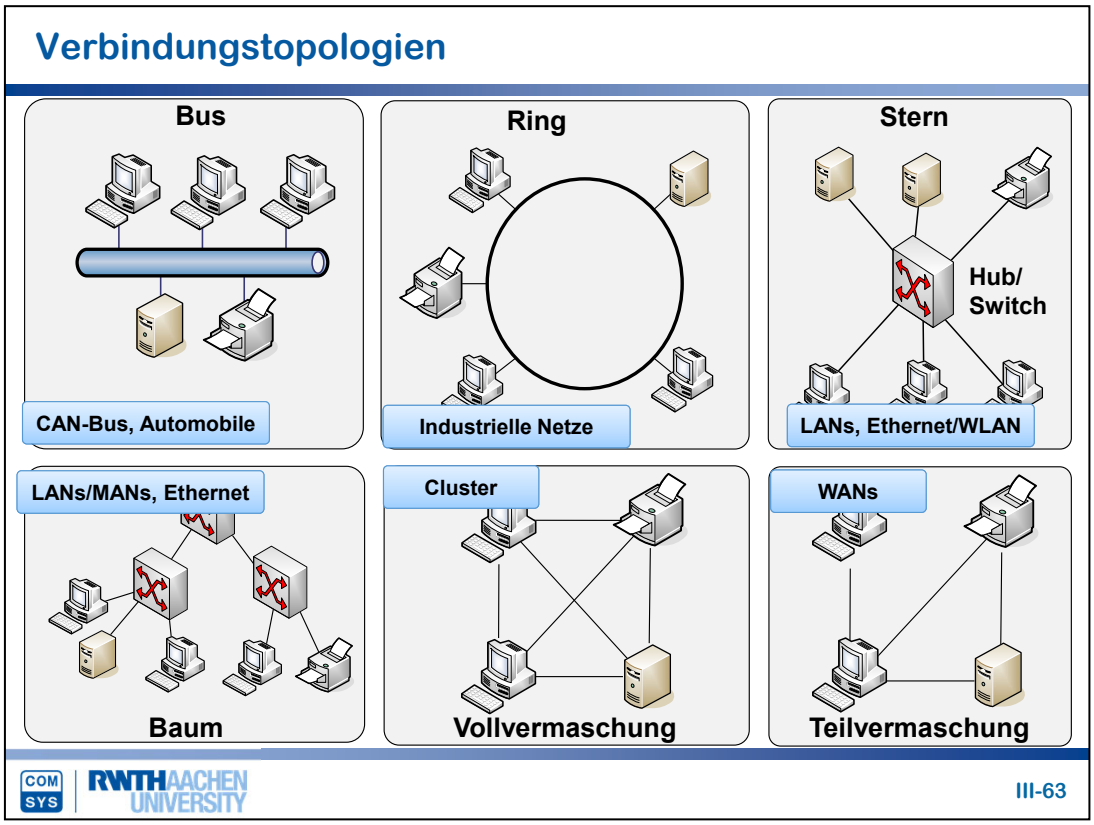
Je nach Aufteilung der Unterkanäle kann man symmetrische Varianten umsetzen (gleiche Datenrate in beiden Richtungen) oder asymmetrische Varianten (höhere Datenrate im Downstream). Asymmetrische Varianten werden typischerweise von den DSL-Anbietern angeboten, da Privatnutzer typischerweise mehr Daten herunter- als hochladen.

Die Verbesserungen der neuen Varianten werden durch Verwendung einer größeren Bandbreite erreicht – VDSL ist über alte Telefonverkabelung dabei aber nicht mehr zu realisieren. Die Distanz zur Vermittlungsstelle muss drastisch verkürzt werden, wozu die DSL-Anbieter Outdoor-DSLAMs installieren, die die Entfernung zu den Privathalshalten auf wenige hundert Meter reduzieren. Mittlerweile werden die DSLAMs sogar innerhalb von Privathäusern selbst installiert und von dort mit Glasfaserkabel mit den Standorten der Provider verbunden, um die Übertragungsstrecke über Telefonkabel noch weiter zu reduzieren und somit die nutzbare Bandbreite noch einmal drastsich zu erhöhen.

Kapitel 3: Sicherungsschicht

- **Rahmenbildung**
  - Blockbildung, Codetransparenz durch Character/Bit Stuffing
- **Fehlererkennung/-behandlung und Flusskontrolle**
  - Fehlererkennende Codes (CRC)
  - Proaktive und reaktive Fehlerbehebung (FEC und ARQ)
  - Flusskontrolle durch Sliding Window
  - Beispielprotokoll: HDLC
- **Medienzugriff**
  - Topologien, Shared Medium
  - Token Passing, CSMA/CD, CSMA/CA
- **Standards für lokale Netze**
  - Ethernet





LANs (lokale Netze) haben nur eine geringe Ausdehnung von wenigen zehn Metern bis zu wenigen Kilometern. Dadurch sind aber sehr große Übertragungsraten im Bereich von 100 Mbit/s bis 10 Gbit/s möglich. LANs werden meist von privaten Firmen und Privatleuten betrieben, da sie nicht unter die Fernmeldehoheit fallen. Ihre Ausdehnung ist deshalb meist auf ein (Privat-) Grundstück beschränkt. Man verwendet klassischerweise meist Busse, Sterne oder Ringe, um schnelle Erreichbarkeit zwischen jedem Paar von Rechnern zu ermöglichen. Heutzutage sind vorwiegend Sterne oder Bäume – welche kaskadierende Sterne sind – im Einsatz.

Ein MAN (Nahverkehrsnetz) erstreckt sich meist über das Gebiet einer Stadt (oder eines Campus‘). Ein MAN soll bei sehr vielen angeschlossenen Knoten (bzw. LANs) auch eine sehr hohe Übertragungsrate und sehr geringe Ausfallzeiten garantieren. Eingesetzt werden meist Ringe und Bäume.

WANs (Weitverkehrsnetze) sind in ihrer Ausdehnung nicht begrenzt. Sie werden meist von staatlichen Einrichtungen oder TK-Gesellschaften betrieben. In vielen Fällen sind die einzelnen WANs auf Staatsgrenzen beschränkt, da jede Gesellschaft den allgemeinen Standard etwas anders benutzt. Trotzdem ist es möglich, länderübergreifend zu kommunizieren (über bestimmte Übergangsknoten). Ein WAN besteht meist aus einem teilvermaschten Netz, da Leitungen bedarfsgetrieben verlegt werden; oft kann man aber auch (hierarchische) Ringstrukturen erkennen.

Vollvermaschung ist in Rechnernetzen nicht üblich – dieses Konzept bietet sich bei Rechenclustern an, um direkten schnellstmöglichen Datenaustausch ohne

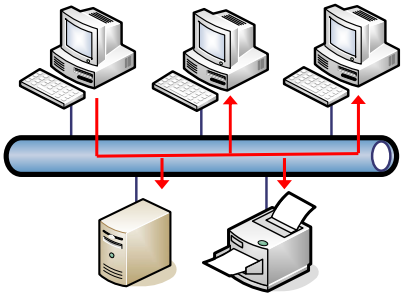
Kollisionsrisiko zwischen jedem einzelnen Rechnerpaar zu ermöglichen.

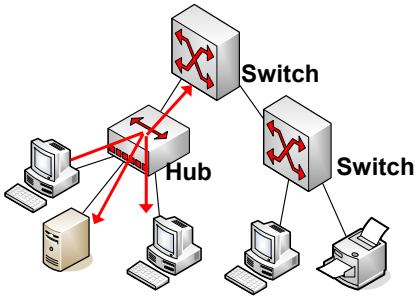
Einen guten Überblick über die Topologien samt vor-/Nachteilen bietet https://de.wikipedia.org/wiki/Topologie_(Rechnernetz).

Broadcast-Netze

- **Bus, Ring und eventuell Stern und Baum sind *Broadcast-Netze***
  - Geteiltes Medium (Shared Medium)
  - Sendet eine Station, erhalten alle anderen Stationen die Signale
    - Ein Netzsegment ist eine *Kollisionsdomäne*
  - Gleichzeitiges Senden zweier Stationen führt zu einer *Kollision*

→ Regelung des Medienzugriffs nötig





Bus und Ring sind sogenannte Broadcast-Netze – sendet eine Station auf dem Medium, erhalten alle angeschlossenen Stationen die Signale. Diese Zustellung der Daten an alle nennt man „Broadcast“. Der Stern (und auch der Baum) sind Broadcast-Netze, wenn als zentrale Komponente ein Hub eingesetzt wird.

Voll- und Teilvermaschte Netze beinhalten nur direkte Verbindungen zwischen genau zwei Rechnern/Netzkomponenten – daher sind dies keine Broadcast-Netze, wenn das Medium vollduplex ist. Ist es nur halbduplex, hat man eine Richtungskonkurrenz. (Ein Beispiel hierfür ist WLAN: Laptop und Access Point kommunizieren auf einem Funkkanal miteinander.)

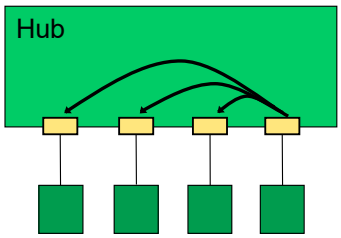
Ein Broadcast-Netz (und auch ein Halubduplex-Kanal) bildet eine Kollisionsdomäne: es wird ein geteiltes Medium verwendet und wenn zwei oder mehr Stationen gleichzeitig senden, tritt eine Kollision auf. Die Signale überlagern sich auf dem Medium, es kann keine Übertragung mehr identifiziert bzw. korrekt dekodiert werden.

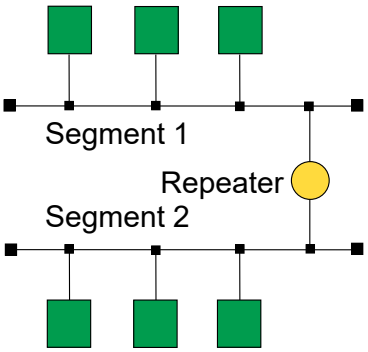
Durch Komponenten wie Switches werden Kollisionsdomänen aufgetrennt: ein Switch kennt die Adressen der angeschlossenen Stationen und kann Rahmen gezielt weiterleiten.

Daher ist in solchen Netzen eine Regelung des Medienzugriffs nötig. Welche Zugriffsregelungen Sinn machen, hängt stark davon ab, wie die Rechner untereinander verbunden sind (Topologie).

Zentrale Komponente beim Stern: Hub

- **Nur Bitübertragungsschicht**
  - *Empfang und Auffrischung von Signalen* (wie Repeater)
    - Kann auch zur Vergrößerung des Netzes verwendet werden
  - Keine Unterbrechung der Kollisionsdomäne
    - Hub broadcastet empfangenes Signal auf allen anderen Ports
    - Nur eine Station zur Zeit kann senden



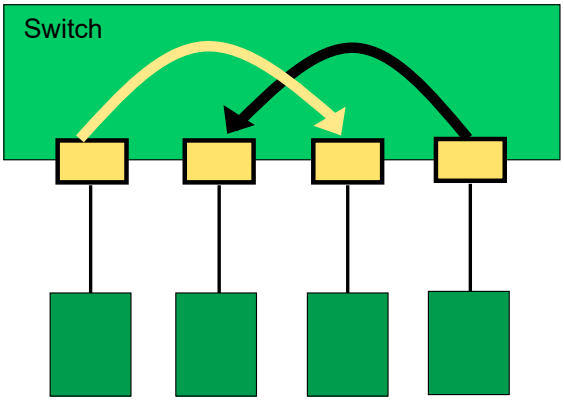


Die erste Komponente, die als zentrale Einheit im Stern eingesetzt wurde, war der Hub. Wie auch die Repeater hat er eine einfache Funktionalität: er empfängt auf einem Port (= Anschluss) Daten und leitet sie auf allen anderen weiter. Da die Signale auf dem empfangenen Port abgetastet und auf den anderen Ports neu erzeugt werden, nimmt er eine Signalauffrischung vor. Damit arbeitet der Hub auf der Bitübertragungsschicht.

Wird ein Hub als zentrale Komponente einer Sterntopologie (oder in einem Baum) eingesetzt, hat man ein Broadcast-Netz und damit die gleichen Probleme wie bei Bus und Ring.

Zentrale Komponente: Switch

- **Bitübertragungs- und Sicherungsschicht**
  - *Punkt-zu-Punkt-Verbindungen* zwischen je zwei Stationen
  - Lernt die Adressen angeschlossener Stationen
  - Puffer für jeden Port
    - Trennt Kollisionsdomänen
    - Keine Kollisionen mehr!
  - Höherer Durchsatz als Hub
  - Realisiert durch hochratigen internen Bus

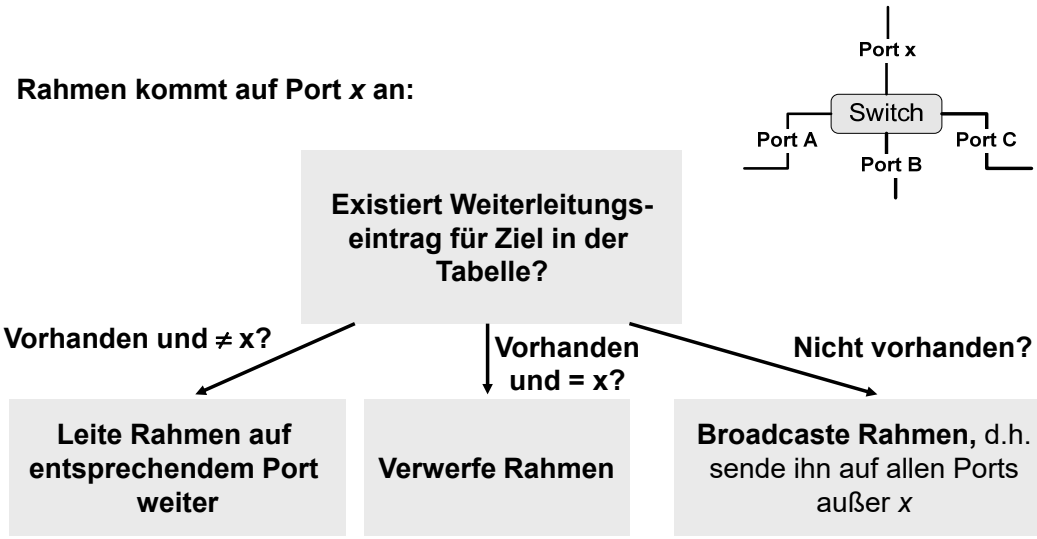


Der Switch wird mittlerweile (hoffentlich) überall als zentrale Komponente eingesetzt. Er verwaltet eine Weiterleitungstabelle und lernt die Adressen der angeschlossenen Stationen. Er kann auch zur Kopplung ganzer Netzsegmente eingesetzt werden (allerdings ohne Protokollwandlung, alle Segmente müssen das gleiche Protokoll einsetzen! Für eine zusätzliche Protokollwandlung würde man eine sogenannte Brücke benötigen), wird aber üblicherweise zur Kopplung einzelner Stationen eingesetzt – als Resultat können keine Kollisionen mehr auftreten und verschiedene Paare von Stationen können simultan Daten austauschen. Der Gesamtdurchsatz des Netzes wird gegenüber dem Hub deutlich erhöht.

Achtung, Falle: im heutigen Sprachgebrauch wird sehr viel als Switch bezeichnet, was eigentlich gar kein Switch mehr ist. Ein sogenannter „Layer-3-Switch“ ist ein Switch, der zusätzlich zu den hier dargestellten Funktionen auch über eine integrierte Routing-Funktionalität verfügt (siehe nächstes Kapitel). Wenn im weiteren Verlauf der Vorlesung/Übungen von „Switch“ die Rede ist, ist stets die eigentliche, hier dargestellte Ursprungsvariante gemeint.

Aufbau von Weiterleitungstabellen

- **Automatisches Lernen von Adressen angeschlossener Stationen**



Ein Switch verwaltet eine Weiterleitungstabelle, in der festgehalten wird, welche Rechner (bzw. Netzwerkkarten mit bestimmten MAC-Adressen) sich hinter welchem Anschluss (Port) befinden. In der Tabelle wird bei jedem Eintrag neben der MAC-Adresse und dem zugehörigen Port auch das Alter des Eintrags festgehalten.

Die Weiterleitungstabelle des Switches muss nicht manuell konfiguriert werden. Statt dessen lernt der Switch, welche Stationen angeschlossen sind, im laufenden Betrieb. Empfängt der Switch einen Rahmen, schaut er sich zunächst die Zieladresse des Rahmens an und durchsucht seine bisherige Weiterleitungstabelle, ob er einen Eintrag für die entsprechende Adresse hat. Ist dies der Fall, wird der Rahmen über den im Eintrag angegebenen Port weitergeleitet. Eine Ausnahme ist, dass der Rahmen über den Port weitergeleitet werden müsste, über den der Switch ihn empfangen hat – in diesem Fall hat sich der Rahmen bereits in dem Segment, in dem das Ziel zu finden ist, ausgebreitet und braucht nicht noch einmal ausgesendet zu werden.

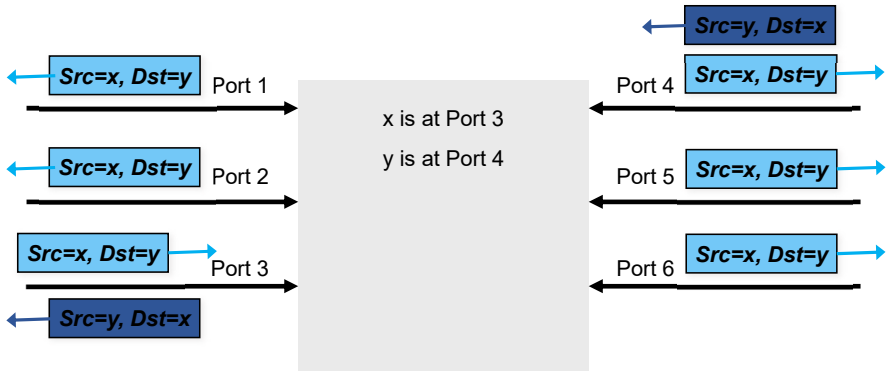
Ist kein Eintrag vorhanden, wird der Rahmen auf allen Ports (außer dem, über den er empfangen wurde) weitergeleitet, damit er auf jeden Fall beim Ziel ankommt.

Gleichzeitig untersucht der Switch auch bei jedem Rahmen, ob er einen neuen Weiterleitungseintrag anlegen muss, indem er sich die Absenderadresse des Rahmens anschaut. Existiert zu dieser Adresse bereits ein Eintrag, wird das Alter des Eintrags zurückgesetzt. Ist im Eintrag ein anderer Port abgespeichert als der, über den der aktuelle Rahmen empfangen wird, wird auch diese Angabe aktualisiert. Existiert noch kein Eintrag, wird ein neuer Eintrag mit der Absenderadresse und dem Port, auf dem der Rahmen empfangen wurde, angelegt.

Das Alter eines Eintrags wird mitgeführt, damit veraltete Einträge nach bestimmter Zeit automatisch gelöscht werden.

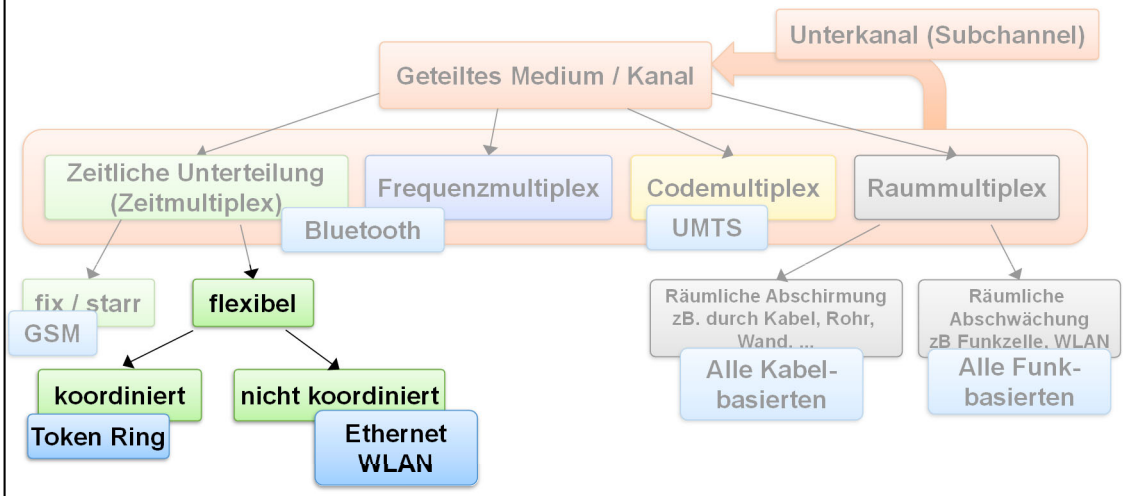
Lernen von Adressen

- **Daten für eine unbekannte Adresse auf einem Port empfangen**
  - Broadcast des Rahmens auf alle anderen Ports
  - Absenderadresse des Rahmens kann für den Port, über den er empfangen wurde, gespeichert werden



Medienzugriff

- **Problem:**
  - Mehrere Stationen als Dienstnutzer eines einzigen physikalischen Mediums (*Shared Medium*) können *Kollisionen* verursachen



Sind mehrere Stationen an ein physikalisches Medium angeschlossen (z.B. Twisted Pair, Koaxialkabel, aber auch ein gemeinsames Frequenzband bei drahtlosen Netzen), muss geregelt werden, wer das Medium zu einem bestimmten Zeitpunkt belegen darf. Die Grundlagen der Verfahren *Frequenzmultiplex (FDM), Zeitmultiplex (TDM)* und *Codemultiplex (CDM)* wurden schon in Kapitel 2 (Bitübertragungsschicht) vorgestellt. Diese Verfahren kann man nutzen, um einen physikalischen Kanal statisch auf mehrere Stationen aufzuteilen. Eine Kombination aus FDM und synchronem TDM wird beispielsweise bei GSM eingesetzt, CDM bei UMTS – Netze, bei denen eine garantierte Datenrate (zur Telefonie) bereitgestellt werden soll.

Der Nachteil eines statischen Multiplexings ist allerdings, dass nicht auf Änderungen in der benötigten Datenrate einer Station eingegangen werden kann. Internet-Verkehr ist oft *burstartig*, d.h. die Übertragung von Daten erfolgt in unregelmäßigen Zeitintervallen, aber dann wird direkt eine große Menge an Daten auf einmal übertragen. Wir benötigen also die Möglichkeit, spontan bei Bedarf quasi beliebige Mengen an Daten übertragen zu können, während ein Multiplexing uns dauerhaft eine konstante Datenrate zur Verfügung stellt.

Die Lösung ist ein dynamisches Multiplexing. Vor allem für LANs (aber auch für MANs) existiert eine Reihe von Varianten des *asynchronen* Zeitmultiplex (asynchron=keine feste Einteilung in Rahmenstruktur bzw. Zeitschlitze), die im Folgenden beschrieben werden. (Die Multiplexingverfahren können auch kombiniert werden – so wird z.B. bei WLAN die gesamte Bandbreite zunächst

statisch in verschiedene Frequenzbereiche unterteilt, dann aber auf jedem dadurch entstehenden Frequenzkanal wieder dynamisches Zeitmultiplexing eingesetzt.)

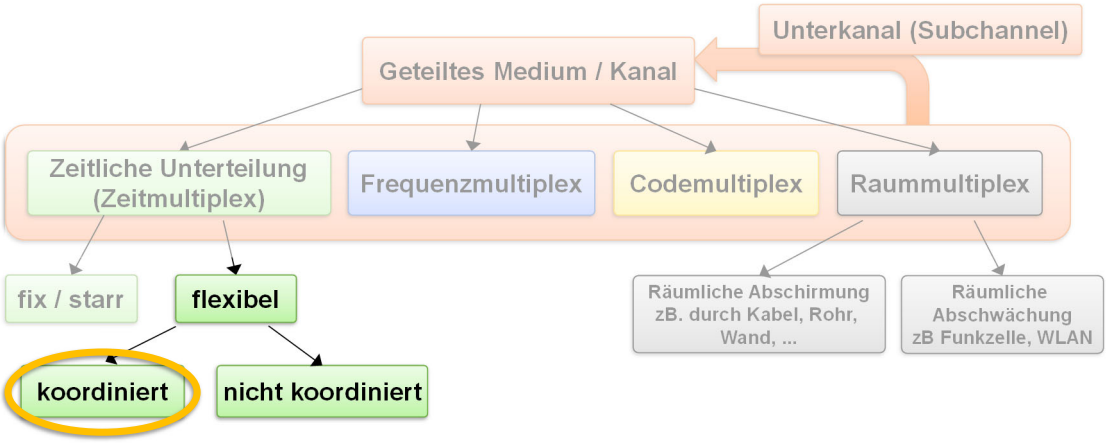
Der größere Teil der existierenden Zugriffsprotokolle ist dezentral organisiert. Grundsätzlich lassen sich die Verfahren in zwei Gruppen einteilen:

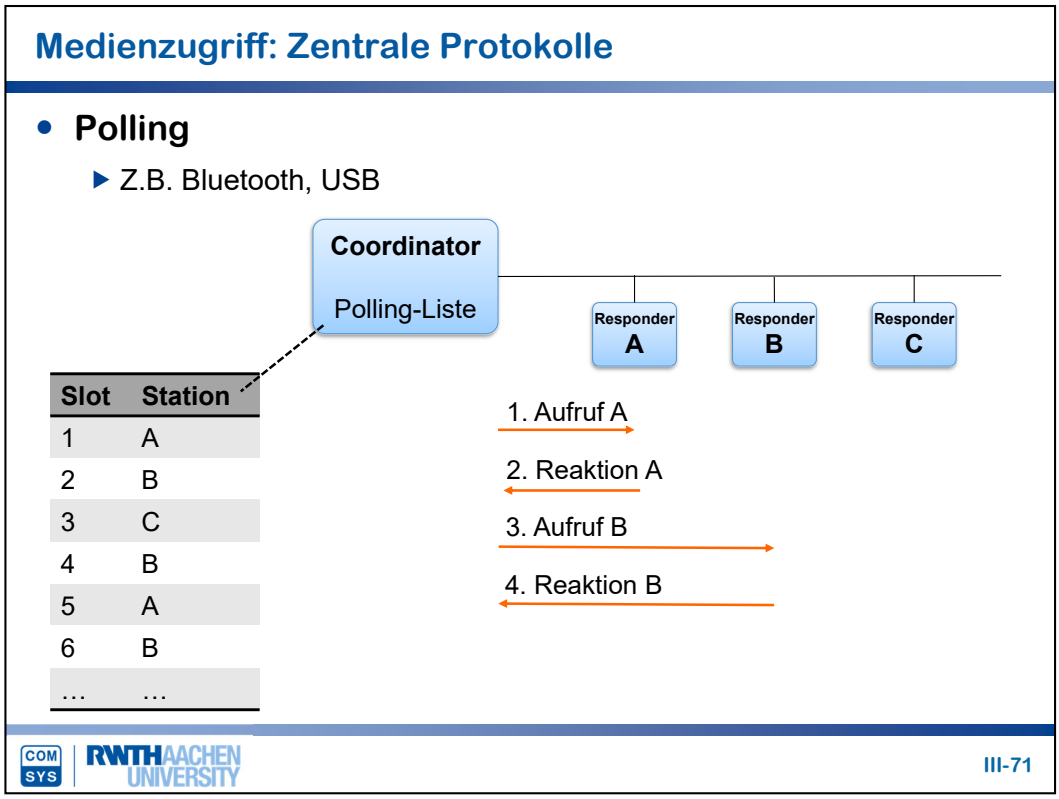
- *Konkurrierender Zugriff (Contention)*: Hier gibt es keine spezielle Zuteilung, d.h. wenn eine Station senden will, so prüft sie bei Bedarf gegebenenfalls, ob das Medium frei ist, und sendet dann. Es kann natürlich zu Kollisionen kommen, falls mehrere Stationen gleichzeitig anfangen zu senden – die Signale aller Stationen überlagern sich auf dem Medium und werden dadurch unbrauchbar. Kollisionen werden aufgelöst, indem zu einem späteren Zeitpunkt eine erneute Übertragung versucht wird. In diese Kategorie fallen z.B. CSMA/CD (Ethernet) und CSMA/CA (WLAN).
- *Geregelter Zugriff* (= kontrollierte Zuteilung): es gibt keine Kollisionen und keine dadurch bedingte Sendewiederholungen. Die Stationen teilen unter sich die Sendeberechtigung auf. Es existieren wiederum mehrere Möglichkeiten:
    - Bei der zyklischen Zuteilung wandert ein Bitmuster, welches das Senderecht repräsentiert (Token), von einer Station zur nächsten. Möchte eine Station senden, so muss sie warten, bis sie das Token erhält, dann kann sie eine gewisse Datenmenge senden (z.B. bei Token Ring).
    - Die Zuteilung kann auch *zentral* durch Aufforderung erfolgen
    - Andere Konzepte kombinieren Verfahren, z.B. das Token-Ring-Prinzip mit konkurrierendem Zugriff, indem zusätzliche Buffer verwendet werden.

Während die bisher in diesem Kapitel behandelten Verfahren bei jedem Netz benötigt werden, ist der Medienzugriff nur dann notwendig, wenn man Netze mit geteiltem Medium hat – und welches Verfahren sinnvoll ist, hängt sowohl vom Medium und der Topologie des Netzes ab, als auch von den Anforderungen der sendenden Stationen. Daher gibt es in diesem Bereich eine Vielzahl an Verfahren und Standards für lokale Netze unterscheiden sich vorwiegend in diesem Punkt.

Medienzugriff

- **Problem:**
  - Mehrere Stationen als Dienstnutzer eines einzigen physikalischen Mediums (*Shared Medium*) können *Kollisionen* verursachen





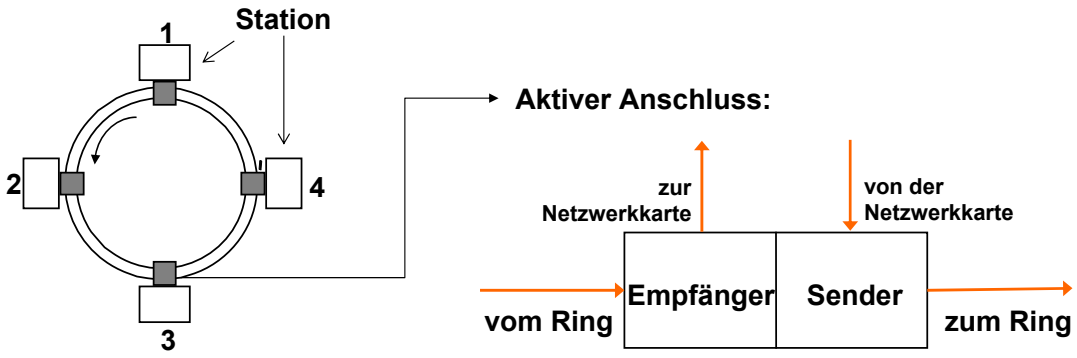
Polling (Poll/Select – Roll call polling)

Hier unterscheidet man einen zentralen Rechner (Leitstation, Coordinator) und viele angeschlossene Stationen (Folgestationen, Responder). Der zentrale Rechner entscheidet, welche Station wann das Senderecht bekommt. Dieses Prinzip wird z.B. bei Bluetooth eingesetzt: ein Coordinator fragt zyklisch alle anderen Geräte ab, jedes Gerät hat danach seinen reservierten Zeitslot, um Daten zu senden.

Abfragen mit gemeinsamer Busleitung: als Variante zum obigen Mechanismus kann eine gemeinsame Leitung (gesonderter Kanal) verwendet werden, auf der Stationen dem Coordinator einen Sendewunsch mitteilen können, damit dieser seine Aufrufliste anpassen kann. Eine Variante, hierbei sogar ohne Master auszukommen, ist das Daisy-Chaining bei kabelgebundenen Netzen; mehr Varianten existieren allerdings für drahtlose Netze.

Medienzugriff: Token Passing (Token Ring, IEEE 802.5)

- **Token Ring: Garantierter Medienzugriff innerhalb einer bestimmten Zeitperiode**
  - Stationen sind Punkt-zu-Punkt zu Ring verbunden
  - Stationen sind aktiv an das Medium gekoppelt (*Repeater*)



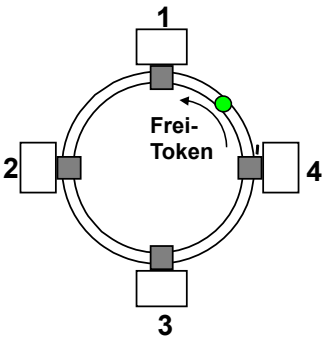
Bei Token Ring sind alle teilnehmenden Stationen physikalisch in einer Ringstruktur verbunden. Man kann auch alle Stationen auf andere Art verschalten und lediglich einen logischen Ring durch alle Stationen ziehen – dies wurde beispielsweise basierend auf einer Bus-Topologie bei 802.4 (Token Bus) vorgenommen.

Auf dem Ring wird eine erlaubte Senderichtung festgelegt; damit hat jede Station eindeutig einen Vorgänger und einen Nachfolger. Die Stationen sind durch sogenannte Repeater aktiv an das Netz gekoppelt: die einkommende Signale werden rücktransformiert in ihre digitale Darstellung; falls sie nicht für die empfangene Station selber sind, sondern über die andere Leitung weitergeleitet werden müssen, werden sie neu in die Übertragungssignale transformiert. Damit nimmt jede Station eine Signalauffrischung vor (sie arbeitet als Repeater), so dass ausgedehnte Netze entstehen können.

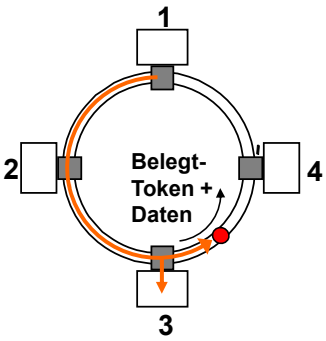
Token Ring basiert zwar auf Koaxialkabel (mit Twisted Pair in 1/4/16 Mbit/s) dennoch ist es für verschiedene Physical Layer implementiert. Zur Datencodierung wird der Manchester-Code verwendet. Zudem wurde eine Variante standardisiert, die Glasfaser benutzt (FDDI) und damit noch größere Abstände zwischen den einzelnen Stationen erlaubt. Dadurch wurde FDDI eine Zeitlang zum beliebten Netztyp für MANs (auch an der RWTH).

Ablaufbeispiel Token-Ring

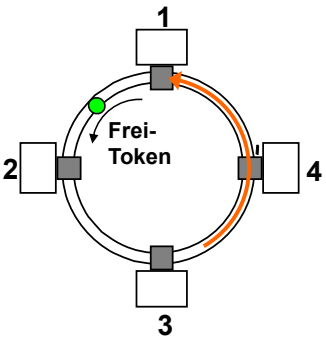
- **Regelung des Zugriffs durch Token Passing**
  - Zeitliche Obergrenze für Medienzugriff durch Token Holding Time



- „Frei“-Token kreist
- 1 hat Sendewunsch



- 1 hat Token belegt
- 1 sendet an 3
- 3 kopiert



- 1 vernichtet Daten
- 3 setzt Quittungsbits (Piggybacking)
- 1 setzt Token auf „frei“

Der Medienzugriff bei Token Ring ist koordiniert und kontrolliert: nur wer eine bestimmte Bitfolge, das Token empfängt, darf senden.

Durch zyklische Weitergabe des Senderechts nach der eigenen Übertragung und die Festlegung einer maximalen Sendedauer pro Station (Token Holding Time, per Default 10ms) erlangt jede Station nach einer berechenbaren Zeit spätestens wieder das Senderecht – einer Station wird dadurch eine maximale Wartezeit bis zum sendebeginn sowie eine feste Datenrate zugesichert.

Erhält eine Station das Senderecht, sendet sie nicht das Token weiter über den Ring, sondern ihre eigenen Rahmen: durch Modifikation eines bestimmten Bits im Token wird aus dem Token der Anfang eines Datenrahmens. Es können beliebig viele Rahmen versendet werden, aber maximal für die Dauer der THT.

Die Rahmen umrunden den gesamten Ring – der Empfänger der Daten wird sie lediglich kopieren, aber trotzdem über den Ring weiterleiten. Dadurch kommen die Daten irgendwann wieder beim Sender an, der sie vom Ring nimmt und stattdessen wieder ein Token erzeugt und auf den Ring setzt.

Token Ring – Mechanismen

- **Zugriffskontrolle durch Token Passing**
  - Modifikation eines Bits im Token → Rahmenbeginn
  - Empfänger kann Quittungsbits an Rahmen anhängen
  - Priorisierung einzelner Stationen möglich (Reservierung)
- **Token-Management nötig**
  - Zentralisierte Station zur Überwachung (*Monitor*)
    - Erzeugung eines neuen Tokens nach Tokenverlust
    - Erkennung endlos kreisender Rahmen
  - Reaktion auf verdoppelte Token
  - Reaktion auf Ausfall des Monitors
  - Überbrückung bei Ausfall einer Netzschnittstelle

Das Token ist ein 3 Byte langes Bitmuster, das zwischen den Stationen weitergeleitet wird. Will eine Station senden, kippt sie einfach ein bestimmtes Bit innerhalb des Tokens und macht das Token damit zum Beginn eines Rahmens. Ein „Belegt“-Token gibt es also eigentlich gar nicht, das „Frei“-Token wird einfach durch einen korrekten Rahmen ersetzt.

Die Quittierung des korrekten Empfangs eines Rahmens ist bei Token Ring direkt in den Medienzugriff integriert: da ein Rahmen den Ring vollständig umrundet und erst durch den Empfänger wieder vom Netz genommen wird, kann der Empfänger eine Quittung an den Rahmen anhängen. Der Sender erhält auch diese Quittung und weiß, ob die Daten korrekt empfangen wurden.

Durchläuft ein Rahmen eine Station, kann diese im Header bestimmte Reservierungsbits setzen, um sich selbst eine höhere Priorität zuzuweisen. Als Effekt wird bei der nächsten Erzeugung eines „Frei“-Tokens ein Token höherer Priorität erzeugt, welches andere Stationen weiterleiten müssen, ohne selbst Daten zu senden; erst eine Station höherer Priorität (im Normalfall die, die die Priorität gesetzt hat) darf das Token nehmen.

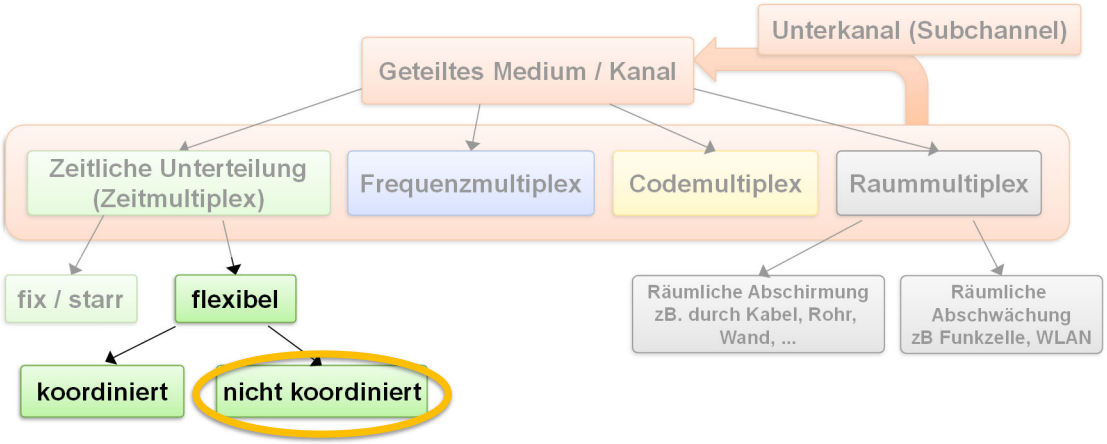
Beim Token Ring können mehrere Fehlersituationen auftreten, die speziell durch die Verwendung eines Tokens zustande kommen:

- Verlust des Tokens: Eine Monitorstation überwacht immer, ob bei ihr ein Token vorbeikommt. Wenn nach einer bestimmten Zeit kein Token mehr vorbeikommt, erzeugt sie ein neues Token.

- Endlos kreisender Rahmen: Fällt ein Sender aus, bevor er seinen Rahmen wieder vom Ring nehmen kann, wird dieser endlos auf dem Ring weiter übertragen und keine andere Station kann mehr senden. Um dies zu verhindern, setzt der Monitor bei jedem Rahmen, welcher vorbeikommt, das sogenannte M-Bit. Wenn nun ein Rahmen mit gesetztem M-Bit ankommt, macht dieser Rahmen bereits eine zweite Umrundung, was nur bei einer Fehlfunktion des Senders der Fall sein kann. Die Monitorstation leert den Ring und erzeugt ein neues Token.
- Doppelte Token: Wenn eine Station, die gerade sendet, einen Datenrahmen bekommt, in welchem nicht ihre Adresse als Sendeadresse steht, bricht sie die Sendung ab. Man hat jetzt kein Token mehr und verfährt nach Fall 1.
- Ausfall des Monitors: Wenn der Monitor ausfällt, so kann jede Station zum neuen Monitor werden. Den Ausfall des Monitors erkennt man daran, dass die obigen Fehler nicht behandelt werden. Wenn dies geschieht, so senden alle Stationen bestimmte Steuerrahmen. Jede Station lässt nur solche Steuerrahmen durch, deren Quelladresse kleiner als die eigene ist. So erhält nur eine Station ihren Rahmen wieder. Diese ist der neue Monitor.
- Ausfall einer Netzschnittstelle: Wenn eine Netzschnittstelle ausfällt, so schließt ein Relais. Somit ist die Leitung überbrückt und zwar eine Station nicht mehr erreichbar, alle anderen können aber noch kommunizieren.

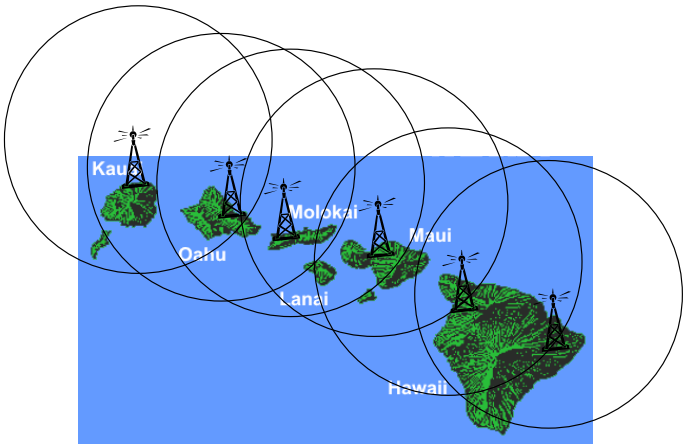
Medienzugriff

- **Problem:**
  - Mehrere Stationen als Dienstnutzer eines einzigen physikalischen Mediums (*Shared Medium*) können *Kollisionen* verursachen



MAC-Protokolle mit konkurrierendem Zugriff: ALOHA

- **Stationen übertragen Daten jederzeit nach Bedarf**
  - Nicht alle Stationen in Reichweite
  - Koordination zwischen Stationen zu aufwändig

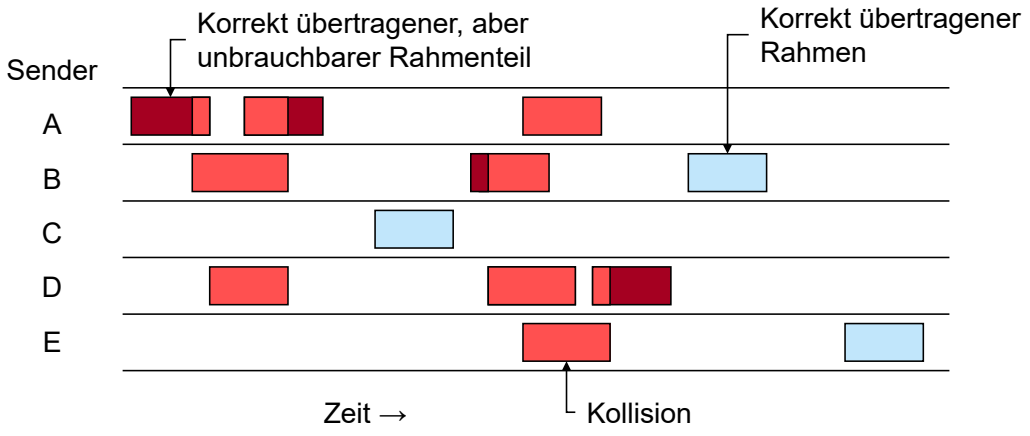


ALOHA

Dieses Protokoll wurde für die Kommunikation zwischen Stationen auf verschiedenen Inseln (Hawaii) mittels Funk entwickelt. Die Stationen können die anderen Stationen bis zu einer bestimmten Entfernung empfangen (z.B. nur die direkten Nachbarn), aber nicht weiter entfernte Stationen. Da eine Koordination zwischen den Stationen sehr aufwändig gewesen wäre, hat man komplett drauf verzichtet: wenn ein Sender etwas zu senden hat, so sendet er.

MAC-Protokolle mit konkurrierendem Zugriff: ALOHA

- **Stationen übertragen Daten jederzeit nach Bedarf**
  - *Kollisionen* führen zu gestörten Rahmen
  - Empfänger schickt Bestätigung, wenn Rahmen korrekt empfangen
  - Maximale Kanalauslastung 18% (analytischer Wert)

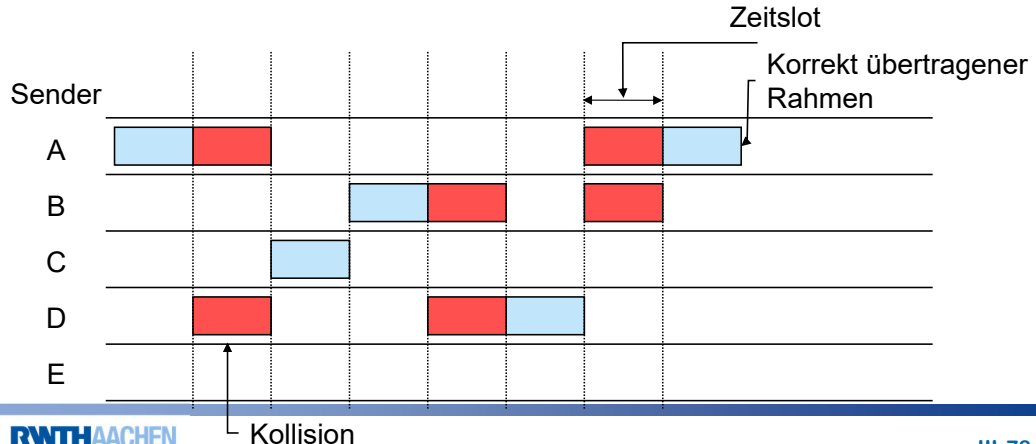


Das spontane Senden kann natürlich zu Kollisionen führen, welche der Sender nicht sofort bemerkt, da sich die verschiedenen Sender gegenseitig zum Teil nicht hören können.

Im schlimmsten Fall überlappen sich die gesendeten Rahmen in nur einem Bit und beide sind trotzdem unbrauchbar – damit ist die maximale Kollisionszeit also zweimal so lang wie die Dauer der Übertragung eines Rahmens maximaler Länge. Der (analytisch ermittelte) maximale Durchsatz liegt hier etwa bei 18% (nicht sehr effektiv). Damit man bei Kollisionen wenigstens nur die Übertragungszeit eines Rahmens verliert, hat man Slotted Aloha konzipiert.

Konkurrierender Zugriff: Slotted ALOHA

- **Übertragung der Rahmen in festen *Zeitslots***
  - Rahmenübertragung nur zu Beginn eines Zeitslots
  - Nur total überlappende Kollisionen treten auf
  - Kollisionszeit: Übertragungszeit von ein statt zwei Rahmen
  - Maximale Kanalauslastung auf 36% verbessert



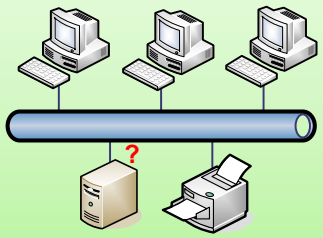
Slotted ALOHA

Durch die Vorgehensweise, Zeit in *Slots* fester Länge einzuteilen und das Senden eines Rahmen nur zu Beginn eines Slots zu gestatten, wird die maximale Kanalauslastung auf 36% gesteigert. Dies ist immer noch sehr gering, erforderte aber zusätzlich noch eine Synchronisationen zwischen allen Stationen, was die Komplexität des Netzes erhöhte.

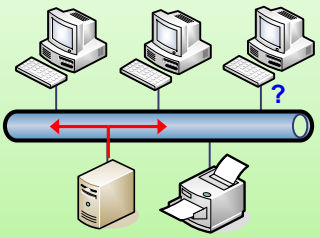
IEEE 802.3: CSMA/CD

- **Verbessertes ALOHA-Prinzip für LANs**
  - Ursprünglich entwickelt für Bustopologie (Broadcast-Netz)
  - Prinzip: Listen before Talk (*Carrier Sense Multiple Access, CSMA*)
    - Standardisiert in IEEE 802.3, eingesetzt in Ethernet
    - Konkurrierendes Zugriffsverfahren
    - Sende nur, wenn Medium aktuell frei ist (Kollisionsvermeidung)

1. Ist das Medium frei? (Carrier Sense)



2. Übertragen



COM SYS | RWTH AACHEN UNIVERSITY

III-79

Aus ALOHA ging CSMA/CD hervor. Während ALOHA für Funknetze entwickelt wurde, bei denen nicht alle Stationen sich gegenseitig hören können, ist CSMA/CD eine Modifikation für kleine Netze (mit geringen Latenzen), in denen sich alle angeschlossenen Stationen gegenseitig hören können.

Beim CS (Carrier Sense)-Verfahren hören die Stationen im Gegensatz zu ALOHA das Medium ab, bevor sie zu senden beginnen. Dadurch lässt sich vermeiden, dass eine sendende Station gestört wird.

Ist das Medium frei, wartet die Station noch 9,6 µs (Interframe Spacing) bevor sie zu senden beginnt. Dieses Spacing soll vermeiden, dass eine Station, die erfolgreich Medienzugriff erlangt hat, beliebig viele Rahmen in Folge senden kann. Durch die erzwungene Wartezeit zwischen zwei Rahmen können andere Stationen mit Sendewunsch das Medium als frei erkennen und auch zum Senden kommen.

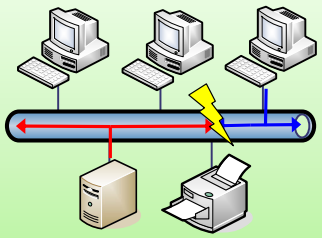
Es kann passieren, dass eine Station bereits sendet während eine zweite Station aufs Medium lauscht um festzustellen, ob sie senden kann. Durch die endliche Ausbreitungsgeschwindigkeit auf dem Medium kann es vorkommen, dass die zweite Station das Medium als frei erkennt, obwohl die erste Station bereits sendet (Abbildung rechts auf der Folie). In diesem Fall fängt auch die zweite Station guten Gewissens an zu senden.

IEEE 802.3: CSMA/CD

- **Verbessertes ALOHA-Prinzip für LANs**
  - Listen while Talk (with *Collision Detection, CD*)
    - Empfang von Signalen während der eigenen Aussendung von Daten bedeutet, dass eine Kollision aufgetreten ist:
      - Erkennbar an Spannungsspitzen
      - Jamming-Signal; Abbruch der Sendung
      - Erneuter Versuch nach statistisch verteilter Verzögerungszeit

3. Prüfe auf Kollision (Collision Detection)

Falls ja: sende Jamming-Signal und
breche ab. Danach weiter mit z.B.
“Binary Exponential Backoff “-Algorithmus



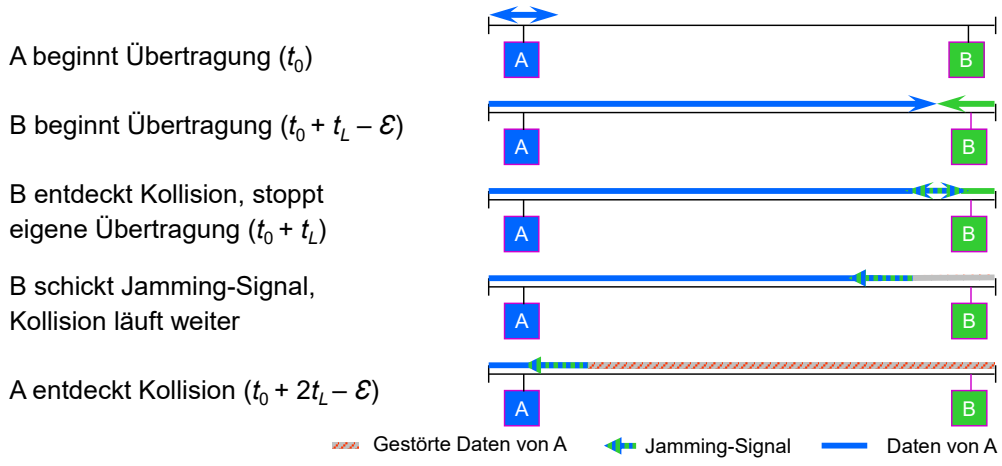
In diesem Fall treffen sich die Signale irgendwo auf dem Medium und überlagern sich (= Kollision). Die beiden Übertragungen löschen sich dabei nicht gegenseitig aus, wie es diese Folie scheinbar zeigt. Die Signale beider Stationen breiten sich über das gesamte Medium aus und jede angeschlossene Station empfängt die Überlagerung. Die beiden sendenden Stationen oder auch beliebige andere an das Medium angeschlossene Stationen bemerken die Kollision an Spannungsspitzen (die durch die Überlagerung entstehen).

So eine Kollision sollte schnellstmöglich erkannt werden, damit das Medium nicht für längere Zeit unnütz belegt wird. Dies wird durch das CD ermöglicht: sendende Stationen hören während des gesamten Sendevorgangs mit, ob sie andere Signale außer ihren eigenen hören.

Wenn eine (beliebige) Station bemerkt (an Spannungsspitzen und Codeverletzungen), dass eine Kollision auftritt, so sendet sie ein sogenanntes *Jamming-Signal*. Dieses Signal ist eine definierte Bitfolge, welche mit höherer Spannung ausgesendet wird, um alle anderen Signale auf dem Medium zu übertönen und auf jeden Fall erkannt zu werden. Dieses soll allen Stationen ermöglichen, die Kollision zu erkennen. Nach Erhalt des Jamming-Signals stoppen alle Sender ihre Sendung.

CSMA/CD – Beispielablauf

- **Zu berücksichtigen: Laufzeiten**
  - t_L : Signallaufzeit von A nach B (Propagation Delay)
  - $2\ t_L$: Signallaufzeit von A nach B und zurück (Round Trip Delay)



Wichtig bei CSMA/CD ist das Propagation Delay: die Laufzeit eines Signals von einem bis zum anderen Ende eines Übertragungsmediums. Diese beträgt in obigem Beispiel t_L. Sie entspricht der Latenz, wobei die Latenz üblicherweise als Ausbreitungsverzögerung zwischen zwei Stationen angesehen wird, das Propagation Delay hier die Verzögerung von einem äußersten Ende des Netzes zum anderen bezeichnet. Die Latenz zwischen je zwei Stationen im Netz ist abhängig von der Position der Stationen, das Propagation Delay ist das Maximum der Latenzen im Netz.

Im Beispiel beginnt B zu senden (zum Zeitpunkt ($t_0 + t_L - \mathcal{E}$), kurz bevor zum Zeitpunkt ($t_0 + t_L$) das Signal von A die Station B erreicht. B erkennt die Kollision praktisch unverzögert nach deren Auftreten. B stoppt die begonnene Sendung und bringt ein starkes Störsignal auf den Bus (Jamming-Signal). Die von der Kollision und dem Jamming-Signal herrührende Signalstörung läuft nach A zurück und wird zum Zeitpunkt ($t_0 + 2t_L - \mathcal{E}$) von A entdeckt. A stoppt seine Sendung ebenfalls; der bisher gesendete MAC-Teilrahmen ist zerstört.

Damit A erkennt, dass das detektierte Kollisionssignal den von ihm abgesandten MAC-Rahmen betrifft, darf die Sendung nach $2t_L$ (Round Trip Delay) noch nicht beendet sein. 802.3-Rahmen müssen also eine Mindestlänge haben, damit eine Kollision zuverlässig erkannt wird!

Nachdem beide kollidierenden Stationen die Übertragung gestoppt haben, erfolgt ein erneuter Anlauf nach einer gewissen Verzögerungszeit.

Kollisionserkennung bei Ethernet (10 Base2)

- ***Slot Time*** **(= Round Trip Delay)**
  - Minimale Dauer für das Senden eines Rahmens zur zuverlässigen Kollisionserkennung
- **Festlegung der maximalen Netzausdehnung**
  - Maximales Round Trip Delay berechenbar
    - 2800 Meter beim klassischen Ethernet
    - 51,2 µs Slot Time
  - Bei Datenrate von 10 MBit/s: in 51,2 µs werden 64 Byte versendet
  - Es müssen immer mindestens 64 Byte versendet werden, um Kollisionen zuverlässig erkennen zu können
    - Sonst: Padding auf Mindestlänge

Um die Dauer berechenbar zu machen, die ein Sender aktiv bleiben muss, bis er sicher sein kann, dass keine Kollision mehr auftreten wird, wurde die maximale Ausdehnung beim anfänglichen Ethernet auf 2800 Meter festgelegt. Diese Entfernung war mit dem verwendeten Koaxialkabel aufgrund der Dämpfung nicht zu überbrücken, daher musste alle 500 Meter ein Repeater eingefügt werden, der das Signal wieder auffrischte. Die Ausbreitungszeit eines Signals von einem Ende des größtmöglichen Netzes bis zum anderen inklusive der Verzögerungen durch Verarbeitung in den Repeatern betrug 51,2µs (bzw. etwas weniger – man hat so aufgerundet, dass man innerhalb dieser Zeit eine Zweierpotenz an Bits versenden kann). Bei der Datenrate von 10 MBit/s, die erreicht werden sollte, können in dieser Zeit 64 Byte versendet werden. Daher muss eine Station bei Ethernet immer Rahmen mit mindestens 64 Byte aussenden, auch wenn sie eigentlich gar nicht so viel zu senden hat. Muss eigentlich nur weniger gesendet werden, wird Padding vorgenommen; der Payload des Rahmens wird mit Füllbits auf die nötige Anzahl Bytes gebracht.

Binary Exponential Backoff

- **Vorgehen nach Kollision**
  - Beide (bzw. alle) Stationen brechen Übertragung ab
  - Vermeidung von Folgekollisionen?
    - Stationen ziehen zufällige Wartezeit bis Übertragungswiederholung
- ***BEB-Algorithmus*:**
  - Anpassung der Wartezeit an die Zahl wartender Stationen
    - Nach der *i*-ten Kollision: ziehe Wartezeit *x* aus $[0, 2^i-1]$
    - Wartezeit bis zum nächsten Carrier Sense: $x \cdot 51{,}2\ \mu s$
    - Nach 10 – 15 Kollisionen konstantes Intervall $[0, 2^{10} - 1]$
    - Nach 16 Kollisionen gebe auf
  - Jede Station verwaltet ihr *i* selbst
    - Einzelne Stationen können benachteiligt werden!

Wenn eine sendende Station eine Kollision erkannt hat und aufhört zu senden, wartet sie eine gewisse Zeit lang und hört dann das Medium wieder ab, ob sie ihre Übertragung wiederholen kann. Es kann natürlich weiterhin zu Kollisionen kommen, vor allem weil während des Wartens noch einige sendewillige Stationen dazukommen können. Die Wartezeit einer Station nach einem erfolglosen Versuch errechnet sich nach dem exponentiellen Backoff-Algorithmus. Dieser berechnet die Anzahl der Slot-Times (die Slot-Time von 51,2 µs entspricht 64 Byte), die gewartet werden muss. Dies garantiert, dass, falls eine Station anfängt, das Medium zu nutzen, nach Ablauf einer Slot-Time auf jeden Fall alle anderen Stationen wissen, dass das Medium in Benutzung ist und weiterhin warten.

Man wartet somit immer ein Vielfaches der Slot-Time. Wenn zwei Stationen die gleiche Zufallszahl ziehen, dann gibt es wieder eine Kollision. Der exponentielle Anstieg des Bereichs, aus dem Zufallszahlen gezogen werden, garantiert, dass bei wenigen Stationen diese nicht allzu lange warten müssen (wir ziehen nur kurze Wartezeiten, die sich hoffentlich trotzdem unterscheiden). Warten viele Stationen, so wird das Intervall der Wartezeiten nach jeder Kollision vergrößert, so dass nach einigen Schritten jede voraussichtlich eine andere Zufallszahl ziehen: nach der ersten Kollision zieht die Station eine Zufallszahl aus dem Bereich [0, 1] und wartet diese Anzahl von Slots. Nach der i-ten Kollision zieht sie eine Zufallszahl aus dem Intervall $[0, 2^i-1]$ und wartet diese Anzahl von Slots. Ab dem 10ten Wiederholungsversuch bleibt das Intervall allerdings immer bei $[0, 2^{10}-1]$. Nach 16 erfolglosen Versuchen bricht die Station ab und verwirft das Paket.

Vergleich von CSMA/CD und Token Ring

CSMA/CD

- **Vorteile**
  - Einfaches Protokoll
  - Installation im laufenden Betrieb einfach möglich
  - Passive Kabel
  - Nur geringe Verzögerung bei niedriger Last

Token Ring

- **Vorteile**
  - Sehr guter Durchsatz und hohe Effizienz unter hoher Last
  - Automatische Erkennung und Elimination von Kabelbruch
  - Prioritäten möglich
  - Kurze Rahmen möglich, Rahmenlänge nur durch THT begrenzt
  - Echtzeitbetrieb möglich

Vergleich von CSMA/CD und Token Ring

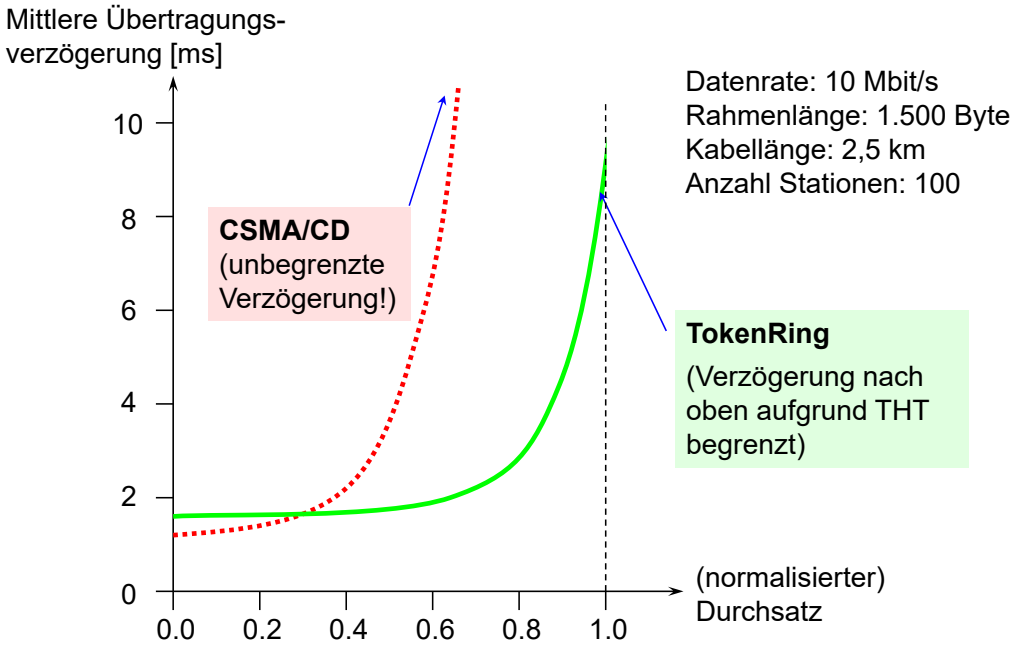
CSMA/CD

- **Nachteile**
  - Kollisionserkennung benötigt minimale Rahmenlänge
  - Keine Prioritäten
  - Nicht deterministisch, deshalb kein Echtzeitbetrieb möglich
  - Begrenzte Netzausdehnung
  - Geringe Effizienz durch viele Kollisionen, problematisch bei höherer Last

Token Ring

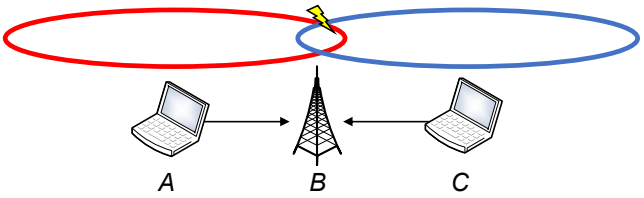
- **Nachteile**
  - Komplexes Fehlermanagement (Token-Verlust, Monitorausfall)
  - Unnötige Verzögerung unter niedriger Last

Verzögerungen von CSMA/CD vs. Token Ring



Drahtlose Netze und CSMA/CD

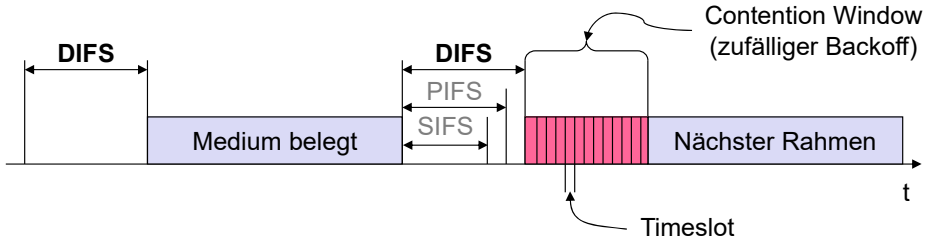
- **Kann CSMA/CD auch in drahtlosen Netzen eingesetzt werden?**
  - *Hidden-Station-Problem*
    - *A* sendet an *B*, *C* ist nicht in Reichweite von *A*
    - C möchte an *B* senden und führt CS aus – Medium frei
      - Carrier Sense schlägt fehl
    - Kollision bei *B*, *A* stellt Kollision nicht fest
      - Collision Detection schlägt fehl
    - *A* ist gegenüber C versteckt (*hidden* für *C*), ebenso *C* gegenüber *A*



In drahtlosen Netzen tritt das ALOHA-Problem auf: es sind nicht notwendigerweise alle Stationen in Reichweite aller anderen Stationen. Ein typisches Beispiel ist Wi-Fi: mobile Geräte können überall rund um den Access Point verteilt sein und sich gegenseitig nicht sehen.

CSMA/CA

- **Modifikation: *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA)**
  - Verzichte auf Kollisionserkennung
  - Kollisionsvermeidung durch zufällige Wartezeit vor Sendungsbeginn
    - Ähnlich BEB



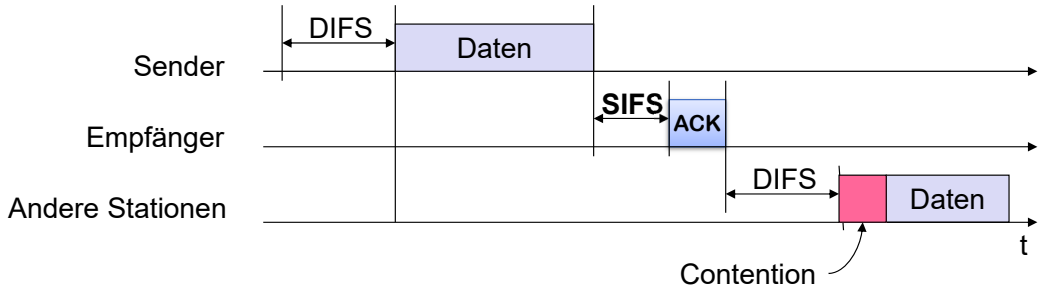
CSMA/CA versucht Kollisionen zu vermeiden, indem Sendewünsche zufällig verteilt werden. Eine sendewillige Station führt nach wie vor CSMA durch und wartet danach einen Inter-Frame Space wie bei Ethernet. Dieser Space ist hier mit DIFS bezeichnet: Distributed Coordination Function Inter Frame Space. Ist das Medium immer noch frei, würde eine Station bei Ethernet nun anfangen zu senden. Bei CSMA/CA hingegen hängt es davon ab, ob das Medium zuvor belegt war oder nicht. War es vorher belegt, gibt es ein nicht zu vernachlässigendes Risiko, dass auch andere Stationen warten und gleichzeitig anfangen würden zu senden. Daher wird nun zufällig eine zusätzliche Wartezeit aus einem vorgegebenen Intervall gezogen (dynamisches Intervall, ähnlich zu BEB). Ist das Medium nach Ablauf der Zufallszeit immer noch frei, beginnt eine Station zu senden.

Garantierte Kollisionsvermeidung erreicht man dadurch nicht.

Die anderen IFSs „Point Coordination Function IFS“ und „Short IFS“ dienen dazu, bestimmte Arten von Übertragungen priorisieren zu können, indem sie eher und ohne Zufallswartezeit Medienzugriff bekommen.

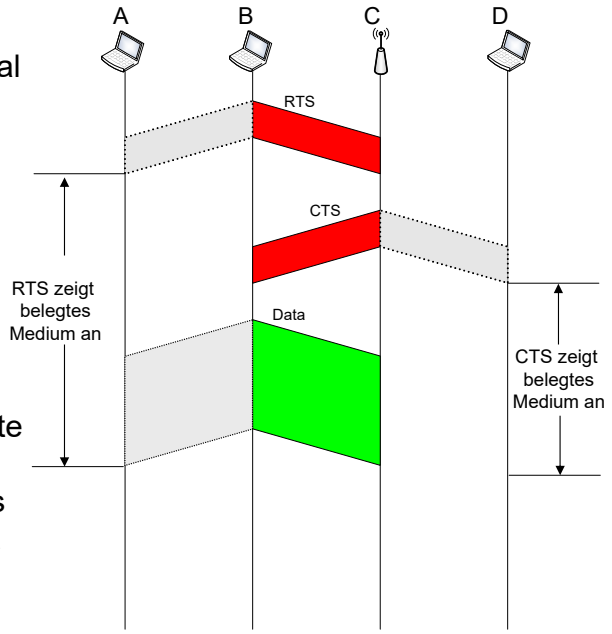
CSMA/CA

- **Hohe Bitfehlerrate – hohes Risiko Rahmenverlust**
  - Quittungsmechanismus integriert
  - Quittung wird priorisiert (SIFS)



Alternative MACA – Multiple Access with Collision Avoidance

- **MACA:**
  - Vor Übertragung wird Kanal reserviert
  - Request to send (RTS) wird verwendet, um Empfänger nach Empfangsbereitschaft zu fragen
  - Clear to send (CTS) teilt Senderecht zu
  - Alle Stationen in Reichweite des Empfängers erhalten das CTS und wissen, dass sie “hidden” sind und nicht senden sollten



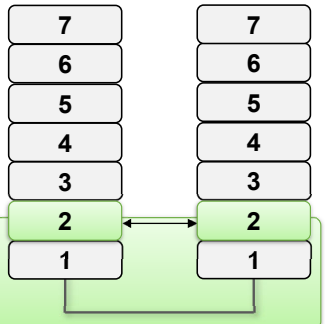
Sollen Kollisionen vermieden werden, kann man MACA verwenden. Eine Station führt immer noch CSMA durch, aber falls der Kanal nicht belegt ist, beginnt die Station nicht mit Versendung des Rahmens, sondern sendet zunächst einen Kontrollrahmen: ein RTS, mit dem sie den Empfänger nach Empfangsbereitschaft fragt.

Der Empfänger antwortet mit einer anderen Kontrollnachricht (CTS), falls er nicht bereits andere anstehende Anfragen hat. Der Sender darf nach Erhalt eines CTS anfangen zu senden. Erhält er kein CTS, muss er seine Übertragung zurückstellen.

Der Sinn des CTS ist auch, dass alle Stationen um den Empfänger – potentielle Hidden Stations – von der anstehenden Übertragung erfahren und selbst keinen Übertragungsversuch starten. Gleiches gilt für alle Stationen rund um den Sender (die das RTS erhalten) – diese sollten auch nicht senden, da der Sender typischerweise auch eine Quittung erhält, die nicht gestört werden soll.

Kapitel 3: Sicherungsschicht

- **Rahmenbildung**
  - Blockbildung, Codetransparenz durch Character/Bit Stuffing
- **Fehlererkennung/-behandlung und Flusskontrolle**
  - Fehlererkennende Codes (CRC)
  - Proaktive und reaktive Fehlerbehebung (FEC und ARQ)
  - Flusskontrolle durch Sliding Window
  - Beispielprotokoll: HDLC
- **Medienzugriff**
  - Topologien
  - Token Passing, CSMA/CD, CSMA/CA
- **Standards für lokale Netze**
  - Ethernet

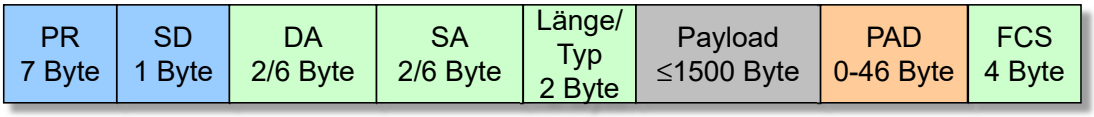


Ethernet

- **Implementierung von CSMA/CD in der Praxis**
  - 70er Jahre: experimentelles Netzwerk, Koaxialkabeln, Datenrate von 3 MBit/s; entwickelt von der Xerox Corporation als ein Protokoll für LANs mit sporadischem aber burst-artigem Verkehrsverhalten
  - 1980: gemeinsame Weiterentwicklung durch DEC, Intel und Xerox zu einer 10 MBit/s-Variante („klassisches“ Ethernet)
  - Bus-Topologie mit einer maximalen Segmentlänge von 500 Metern, Anschlussmöglichkeit für maximal 100 passive Stationen; Repeater zum Zusammenschluss mehrerer Segmente
  - Gängiges Medium: Kupferkabel; aber auch Glasfaser-Kabel kommen zum Einsatz (erhöht Segmentlänge)
  - Frühe 90er Jahre: Bus-Topologie wird mehr und mehr von Stern-Topologie mit Hub oder Switch verdrängt (auf Twisted-Pair oder Glasfaserkabel basierend)

In den Anfangszeiten der lokalen Netze haben sich zwei Konzepte prominent entwickelt: 802.5 wurde unter dem Namen „Token Ring“ vertrieben, 802.3 mit leichten Variationen unter dem Namen „Ethernet“. Bitte beachten: Die 802.3/5 definieren den MAC-Layer – für konkrete Produkte werden auch Festlegungen für den Physical-Layer benötigt!

Rahmenformat nach IEEE 802.3 / Ethernet



PR : Präambel zur Synchronisation (7 x 10101010)
SD : *Start-of-frame Delimiter* zeigt Blockbeginn an (10101011)
DA : *Destination Address*
SA : *Source Address*
Länge : Anzahl der Bytes im Datenfeld (IEEE 802.3), Typ des Payloads im Datenfeld (Ethernet)
Payload: Datenfeld, das maximal 1500 Byte umfassen darf
PAD : *Padding*, um zu kurze Datenfelder auf die nötige Länge zu ergänzen
FCS : *Frame Check Sequence*, CRC

802.3 – Rahmenformat

- Präambel: Dient zur anfänglichen Synchronisation der Stationen durch konstant auftretenden Pegelwechsel.
- Start Delimiter: zeigt den Beginn eines Rahmens an: 10101011.
- Zieladresse, Absenderadresse: laut ursprünglichem Standard: 2 oder 6 Byte; heute: 6 Byte MAC-Adresse.
- Längenfeld: gibt die Länge des Datenfelds (in Byte) an. Bei der Implementierung von 802.3 in Ethernet wird allerdings dieses Feld für einen anderen Zweck missbraucht: der Empfänger muss wissen, welcher Typ von Daten im Datenteil enthalten ist. Laut IEEE steht diese Angabe im LLC-Header, mit dem der Datenteil beginnt. In der Praxis verzichtet man aber auf den Einsatz von LLC, daher muss die Typangabe im 802.3-Rahmen selbst untergebracht werden. Bei Ethernet wird daher die Typangabe in das Längenfeld eingetragen; es werden nur Werte größer 1500 erlaubt, um eine Verwechslung mit einer Längenangabe zu vermeiden.
- Payload: Die Daten werden ohne Bitstuffing o.ä. angefügt, da kein End Delimiter verwendet werden muss (Längenangabe). Bei Verwendung des Längenfelds für die Angabe des Typs von Daten erhält man die Datenfeldlänge implizit dadurch, dass es ein „Interframe Spacing“ gibt; eine Station, die feststellt, dass das Medium frei ist, muss noch für 9,6µs warten, bevor sie zu Senden anfangen darf. Zwei Rahmen können also nicht direkt aufeinander folgen. Die letzten 4 empfangenen Byte müssen also die FCS gewesen sein.
- PAD: Dieses Feld enthält eventuell Padding, d.h. wahllos eingefügt Bits. Zu kurze Datenfelder werden dadurch aufgestockt, damit die Rahmenlänge (ohne Präambel) mindestens 64 Byte beträgt, die vorgeschriebene Mindestlänge zur Kollisionserkennung.
- FCS: Dieses Feld dient zur Fehlererkennung: CRC-Verfahren mit 32 Bit-Polynom

Als Leitungscodierung verwendet Ethernet den Manchester-Code. Bei Überlagerung von Signalen kommt es zu Codeverletzungen, so dass alle Stationen Kollisionen erkennen können.

Ethernet

- **Ethernet: der heutige LAN-Standard**
  - Einfach zu verstehen, umzusetzen und zu überwachen
  - In der Anschaffung billig
  - Bringt Flexibilität bzgl. der Topologie mit sich
  - Über die Zeit Entwicklung von 4 Klassen von Ethernet-Varianten:

• (Klassisches) Ethernet	→	10 Mb/s
• Fast Ethernet	→	100 Mb/s
• Gigabit Ethernet	→	1000 Mb/s
• 10Gigabit-Ethernet	→	10000 Mb/s

  - Des Weiteren:
    - 40G-Ethernet, 100G-Ethernet

„Ethernet“ bezeichnet nicht einen einzelnen Standard, sondern wird im Sprachgebrauch für eine ganze Standardfamilie verwendet. Die Grundprinzipien sind bei allen Unterstandards gleich, sind aber auf unterschiedliche Medien angepasst und können durch weitere Anpassungen unterschiedliche Datenraten erzielen. Diese Unterschiede betreffen also im Wesentlichen die Umsetzung des Physical-Layers, während der MAC-Layer größtenteils gleich umgesetzt wird (~802.3).

Datenrate und Kollisionserkennung

- **Festlegung der minimalen Rahmenlänge im klassischen Ethernet auf 64 Byte**
  - Kompatibilität: behalte minimale Rahmenlänge bei Fast Ethernet bei
- **Fast Ethernet (IEEE 802.3u)**
  - Datenrate von 100 MBit/s: 64 Byte werden in 5,12 µs versendet!
  - Reduktion der Ausdehnung auf gut 200 Meter notwendig, um Kollisionen zuverlässig erkennen zu können
- **Gigabit Ethernet (IEEE 802.3z)**
  - Datenrate von 1 GBit/s: 64 Byte werden in 0,512 µs versendet!
  - Reduktion der Ausdehnung auf 20 Meter?

Umsetzung bei Ethernet

	Ethernet	Fast Ethernet	Gigabit Ethernet
Maximale Ausdehnung	bis zu 2800 Meter	205 Meter	200 Meter
Datenrate	10 MBit/s	100 MBit/s	1000 MBit/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-Code, ...
Topologie	Bus oder Stern	Stern	Stern
Maximale #Repeater	5	2	1

Für einen konkreten Netzstandard werden die unteren beiden Schichten also gemeinsam standardisiert. Dies betrifft auf dem Physical-Layer die Art des physikalischen Mediums, seine Länge, Steckertypen zum Anschluss von Stationen, Topologie des Netzes, Codierung, Schrittgeschwindigkeit, … - hier dargestellt ist nur eine Teilmenge der Möglichkeiten. Z.B. gibt es beim Classical Ethernet nicht nur die Möglichkeit, den Manchester-Code zu verwenden, sondern auch Modulationsverfahren. Hier dargestellt sind die gängigsten Varianten.

Dargestellt ist hier auch ein Aspekt des MAC-Layers: die erlaubte Rahmenlänge. Durch die Wahl von CSMA/CD als MAC-Verfahren ergibt sich die Notwendigkeit einer minimalen Rahmenlänge.

Die maximale Ausdehnung ist auch abhängig von CSMA/CD. Eine maximale Ausdehnung von 2800 Metern bei Ethernet auf Basis von Kupferkabel allerdings wäre zunächst nicht möglich, wenn auch durch CSMA/CD erlaubt: durch Signalabschwächung auf dem Medium können Kupferkabel nur über wenige 100 Meter die notwendige Bandbreite zur Verfügung stellen. Sollen größere Ausdehnungen erzielt werden, ist der Einsatz von Repeatern zur Signalauffrischung notwendig. Prinzipiell kann durch Repeater auch eine beliebig große Ausdehnung erzielt werden – aber durch CSMA/CD sind keine größeren Ausdehnungen erlaubt.

Die einzige „Optimierung“ beim klassischen Ethernet waren sogenannte *Brücken (Bridges),* die Kollisionsdomänen auf unterschiedlichen Segmenten trennen sollten.

Man setzt zur Verbindung zweier Segmente nicht mehr einen Repeater zur puren Signalauffrischung ein, sondern implementiert in den Brücken auch Schicht-2-Protokolle. Hierdurch ist eine Brücke in der Lage, MAC-Adressen zu verstehen. Die Brücke kann lernen, welche Stationen mit welchen Adressen über welches der Segmente, die sie koppelt, erreichbar sind. Damit kann sie auch entscheiden, ob ein auf einem Segment empfangener Rahmen auf das andere Segment weitergeleitet werden muss oder nicht. Die Brücke führt also eine Weiterleitungstabelle zu allen ihr bekannten MAC-Adressen. Damit wird die Zahl von Kollisionen auch in ausgedehnten Netzen stark reduziert. Trotzdem darf die maximal erlaubte Ausdehnung nicht überschritten werden, da miteinander kommunizierende Stationen natürlich auch in den am weitesten auseinanderliegenden Segmenten platziert sein können und die Rahmen auch durch die Brücken über alle Segmente hinweg weitergeleitet werden.

Gigabit-Ethernet: Rahmenformat

- **Beibehaltung der Segmentlänge**
  - Minimale Rahmenlänge von 64 Byte beibehalten → Reduktion der Netzausdehnung auf 20 Meter wäre erforderlich!
  - Daher: größere minimale Rahmenlänge aber Beibehaltung des alten Rahmenformats durch eine *Carrier Extension*

- Padding auf 64 Byte innerhalb des Rahmens
- Padding auf 520 Byte außerhalb des Rahmens
- Entfernung des „nodata“-Anteils bei Übergang von Gigabit- in Fast-Ethernet

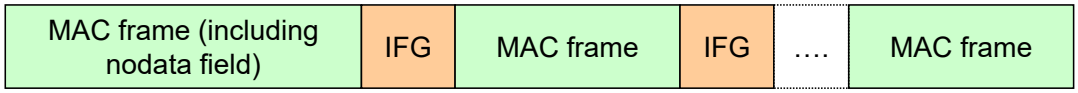
Um zu vermeiden, dass bei erneuter Steigerung der Datenrate um den Faktor 10 die Ausdehnung des Netzes für CSMA/CD um den Faktor 10 reduziert werden muss, hat man zu einem schmutzigen Trick gegriffen: führe ein zweites Padding-Feld außerhalb des Rahmens ein. Ist ein Rahmen kleiner als 64 Byte, wird das innere Padding-Feld genutzt, um den Rahmen auf 64 Byte aufzufüllen. Danach (oder falls ein Rahmen zwischen 64 und 520 Byte Größe hat) wird Padding hinter dem Rahmen vorgenommen (Nodata-Feld). Dieses Vorgehen wird Carrier Extension genannt.

Hierdurch ermöglicht man gemischten Betrieb von Fast- und Gigabit-Ethernet. Wird ein Rahmen < 520 Byte von einer Fast-Ethernet-Station über einen Switch (der beide Varianten spricht) an eine Gigabit-Station weitergeleitet, fügt diese ein Nodata-Feld an. Umgekehrt wird solch ein Feld beim Übergang von einem Gigabit- in ein Fast-Ethernet entfernt.

Da mittlerweile wohl nur noch Switches eingesetzt werden, ist eine Kollision mehr möglich, aber der Standard führt trotzdem noch diesen Trick für die Kollisionserkennung mit sich…

Gigabit-Ethernet: Rahmenformat

- **1500 Byte maximale Rahmenlänge sehr beschränkt**
  - *Frame Bursting*: sende mehrere Rahmen ohne erneutes CSMA
  - Senden von bis zu 5 Rahmen in Folge möglich, verbunden durch *Inter-Frame Gaps* (IFGs)
    - Medium scheint für andere Stationen belegt
    - Beibehaltung des vorherigen Rahmenformats



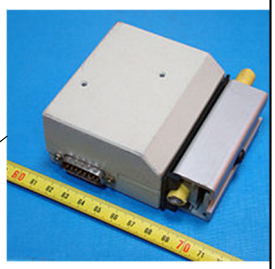
Um den Effekt von Carrier Extension bei Versendung vieler kleiner Frames zu reduzieren, hat man Frame Bursting eingeführt: sende mehrere Rahmen, ohne zwischendurch erneut CSMA durchführen zu müssen (was im Normalfall sowieso überflüssig ist, da Switches eingesetzt werden). Statt dessen sendet man mehrere Rahmen in Folge und um diese voneinander zu trennen, werden bestimmte Bitfolgen als spezieller Interframe-Gap eingefügt. Damit scheint das Medium für konkurrierende Stationen durchgehend belegt. Ab dem zweiten Rahmen ist nun kein Carrier Extension mehr notwendig.

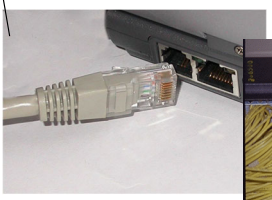
Benennung von Ethernet-Varianten

- **Benennung von Ethernet-Varianten:**
  - Datenrate in MBit/s (10, 100, *1000* oder 10G)
  - Übertragungstechnik (z.B. *Base* für Basisband, Broad für Broadband)
  - Maximale Segmentlänge á 100 Metern oder *Medientyp*
- **Beispiele:**
  - 10Base-T: 10 MBit/s, Basisband, Twisted-Pair-Kabel
  - 100Base-T2: 100 MBit/s, Basisband, 2 Adern des Twisted-Pair-Kabels
  - 1000Base-X: 1000 MBit/s, Basisband, Glasfaser-Kabel
- **Von der Variante hängen noch Parameter wie z.B. min. Rahmenlänge ab (Ausbreitungsgeschwindigkeiten):**
  - 1000Base-X: minimale Rahmenlänge 416 Bytes
  - 1000Base-T: minimale Rahmenlänge 520 Bytes

Ethernet-Varianten: klassisches Ethernet

Bezeichnung	Kabel/ Topologie	Segment-länge	Knoten pro Segment
10Base5 (Thick Ethernet)	Dickes Koax (Bus)	Max. 500m	Max. 100
10Base2 (Thin Ethernet)	Dünnes Koax (Bus)	Max. 185m	Max. 30
10BaseT	Twisted Pair (Stern)	Max. 100m	Max. 1.024
10BaseF	Glasfaser (Stern)	Max. 2.000m	Max 1.024





COM SYS | RWTH AACHEN UNIVERSITY

III-100

Technische Realisierungen von CSMA/CD

- *10Base5* Thick Ethernet: Bus, dickes gelbes Koax-Kabel, 10 MBit/s Übertragungsrate, Segmentlänge = 500m, 100 Rechner pro Segment, max. 5 Segmente =2500m Länge (wegen Signallaufzeit bzw. Slot-Time), über Repeater verbunden. Medienzugriff über Transceiver, den man ins Kabel einsticht. Extra Leitung (Attachment Unit Interface, AUI) zum Rechner (Transceiverkabel, max. 50m).
- *10Base2* Thin Ethernet (Cheapernet): Bus, dünnes Koaxkabel, 10 Mbit/s Übertragungsrate, 185m pro Segment (höhere Dämpfung), max. 5 Segmente, 30 Rechner pro Segment, Anschluss direkt an Karte über T-BNC-Stecker (Transceiver in Adapterkarte integriert).
- *10BaseT* Twisted Pair: Sternförmige Kopplung über zentrale Komponente (z.B. Hub, Switch). 10 MBit/s Übertragungsrate, UTP-Kabel mit max. 100m von Station zur zentralen Komponente, theoretisch max. 1024 Stationen pro Segment, maximal 5 Segmente.

Ethernet-Varianten: Fast Ethernet (IEEE 802.3u)

- **Einschränkung der Topologie**
  - Sterntopologie
  - Datenrate100 MBit/s → Reduktion der Ausdehnung

Bezeichnung	Kabel	Segmentlänge	Vorteile
100Base-T4	Vier verdrillte Adernpaare Cat3	Max. 100m	Ältere Kabel wiederverwendbar
100Base-TX	Zwei verdrillte Adernpaare Cat5	Max. 100m	Vollduplex bei 100 Mbit/s
100Base-F	Glasfaser	Max. 800m	Vollduplex, längere Strecken

Bei Fast Ethernet wurde das Konzept von IEEE 802.3 (insbesondere das Zugriffsverfahren) übernommen und die Datenrate auf 100 MBit/s erhöht. Bei einer Übertragungsrate von 100 MBit/s müsste man in einer Slot-Time von 51,2 µs nun 640 Byte senden, um das ganze Kabel zu belegen. Diese minimale Rahmengröße ist aber zu groß. Deshalb wurde die maximale Ausdehnung auf 100m beschränkt und nur noch eine sternförmige Netztopologie (Anschluss erfolgt an einen Hub oder Switch - 100m Länge zum Hub entsprechen dann 200m Netzausdehnung) zugelassen. Folgende Kabelarten werden unterstützt:

- 100 Base-T4: 4 Adernpaare (UTP-3/4/5-Kabel). Drei Adernpaare werden zur Datenübertragung genutzt, eines zur Übertragung von Kontrollinformationen. Auf jedem Adernpaar erfolgt eine Übertragung mit 33.33 Mbit/s (8B6T Code). Die Übertragung erfolgt im Halbduplex-Modus.
- 100 Base-TX: 2 Adernpaare (UTP-5-Kabel, STP-1/2-Kabel), eins pro Richtung. Man benutzt die 4B/5B-Codierung zur Übertragung (vollduplex).
- 100 Base FX: Hier verwendet man Multimode-Glasfaserkabel (jeweils eines pro Richtung) zur Vollduplexübertragung. Die maximale Länge der Übertragungsstrecke zum Hub beträgt 400m – was eigentlich länger ist als durch CSMA/CD erlaubt! Allerdings werden bei Verwendung von Glasfaser nur noch Switches eingesetzt, so dass keine Kollisionen mehr auftreten können und man auf Kollisionserkennung keine Rücksicht mehr nehmen muss.

Ethernet-Varianten: Gigabit Ethernet (IEEE 802.3z)

- **Weitere Steigerung der Datenrate**
  - Sterntopologie
  - Datenrate 1.000 MBit/s
  - Auch UTP möglich (802.3ab)

Bezeichnung	Kabel	Segmentlänge
1000Base-T	4 ungeschirmte Adernpaare (UTP-5)	100m
1000Base-CX	2 geschirmte Adernpaare	25m
1000Base-SX	Multimode-Glasfaser	2 - 550 m
1000Base-LX	Multi-/Monomode-Glasfaser	2 - 5.500 m

Auch hier zu sehen: bei Glasfaserverkabelung wird vorausgesetzt, dass keine Kollisionen mehr verursacht werden können und die Kollisionserkennung wird ignoriert.

Gigabit Ethernet: 1000Base-T

- **Wegen Verbreitung von UTP-5-Kabeln: Gigabit-Ethernet soll auch für diesen Kabeltyp möglich sein**
  - Kompatibilität: Länge von bis zu 100 m soll beibehalten werden
  - 8B/10B-Codierung: 1.250 MBaud zur Übertragung notwendig!
  - Probleme mit Übersprechen, Dämpfung, ...
- **Lösung:** ***parallele Nutzung aller vier Doppeladern***
  - Steuerung des Senders durch Switch
  - Modulationsverfahren: PAM5 (Pulsamplitudenmodulation mit fünf Zuständen)
    - Über die 4 Adernpaare kann pro Signalschritt 1 Byte übertragen werden
  - Duplexbetrieb durch Echokompensation
  - → Reduktion auf 125 MBaud pro Adernpaar

Probleme beim Übergang zu Gigabit-Ethernet:

- Kompatibilität zu älteren Ethernet-Varianten sollte beibehalten werden, um ein gemischtes Zusammenschalten von Stationen zu ermöglichen. Damit darf sich auch das Rahmenformat nicht ändern, wodurch wieder die Netzausdehnung reduziert werden müsste (auf ca. 20 Meter… „Very local area network“???).
- Erreichen der nötigen Datenrate über relativ schlechte Kabel – was ermöglicht wurde, indem auf allen vier Adernpaaren parallel ein ganzes Byte codiert wird. Um Duplexbetrieb zu erreichen, werden alle vier Adernpaare simultan in beide Richtungen genutzt, was spezielle Filter nötig macht, damit die eigene Sendung nicht Störeinflüsse auf die empfangenen Signale ausübt.

Aktuelle Ethernet-Entwicklungen

- *10-Gigabit-Ethernet* (802.3ae)
  - Nur noch Switch als zentrale Komponente erlaubt
    - Keine Notwendigkeit mehr für CSMA/CD
    - Aus Kompatibilitätsgründen mit implementiert (Rahmenformat, Größen)
  - Zwei Varianten der Bitübertragungsschicht
    - 10 Gbit/s für LAN
    - 9,615 Gbit/s zum Übergang zu SDH (WAN)
  - Anfangs nur Glasfaser, mittlerweile auch Twisted Pair
- **Mittlerweile**
  - 40G-Ethernet, 100G-Ethernet
    - Beibehaltung des Rahmenformats
    - 7 Meter über Twisted Pair, bis zu 40 km über Glasfaser
  - EtherCAT, Ethernet Powerlink, …

Bei 10G-Ethernet müsste man erneut die Ausdehnung verringern, da für die höheren Datenraten eine höhere Bandbreite benötigt wird – und höhere Frequenzen werden stärker gedämpft. Ziel war: 50m bei CAT6, 100m bei CAT7. Um dies mit 10 Gigabit zu erreichen, hat man einige Änderungen auf der Bitübertragungsschicht eingeführt: zum einen wird nicht mehr ein Signal über ein Adernpaar geschickt, sondern nur noch über eine Ader. Damit kann man die Daten auf 8 Adern verteilen. Als Resultat wird aber der Einfluss von Störungen durch das Verdrillen nicht mehr reduziert – daher fügt man dem Signal vor der Übertragung ein Rauschen hinzu: die Signale werden auf Senderseite so verzerrt, dass sie nach Dämpfung und Störeinflüssen auf Empfängerseite geschätzt so ankommen, wie es gewünscht ist. Zusätzlich wird PAM16 verwenden, also ein Code mit 16 Zuständen, so dass 4 Bit pro Signal pro Ader übertragen werden können.

Des weiteren hat Ethernet sich mittlerweile auch in vielen anderen Bereichen durchgesetzt – beispielweise auch in Busnetzen, bei denen die Zustellung von Daten zeitkritisch ist oder priorisiert werden können muss (wozu früher Token Bus konzipiert wurde). Hierzu gibt es eigene Mechanismen (beispielsweise EtherCAT, Ethernet Powerlink), die zumindest das Rahmenformat von Ethernet übernehmen.

Zusammenfassung

- **Aufgabe der Sicherungsschicht**
  - Gesicherte Übertragung von Daten zwischen benachbarten Stationen
    - Umfasst Rahmenbildung, Fehlererkennung/-behebung, Flusskontrolle, Medienzugriff
  - Standardverfahren wie CRC (Fehlererkennung), Go-Back-N/Selective Repeat (Fehlerbehebung), Sliding Window (Flusskontrolle)
  - Vielfältige Mechanismen zum Medienzugriff
    - Dezentral, geregelt/konkurrierend
    - Nur notwendig in Netzen mit geteiltem Medium
    - Sinnvoller Mechanismus hängt von Medium und Topologie ab
  - Standard für lokale kabelgebundene Netze: Ethernet

Lessons Learned

- **Wichtige Begriffe/Konzepte des Kapitels**
  - Rahmenbildung notwendig für Fehlererkennung
  - Codetransparenz durch Character-/Bitstuffing
  - Fehlererkennung durch Redundanzen; bevorzugtes Verfahren: CRC
  - Fehlerkorrektur durch FEC oder ARQ
  - Go-Back-N vs. Selective Repeat/Reject als prominente ARQ-Verfahren
  - Flusskontrolle durch Sliding Window
  - Medienzugriff angepasst an Medium und Topologie
    - Bevorzugt: asynchrones Zeitmultiplex, dezentral koordiniert
    - Ethernet / CSMA/CD erfolgreichstes Verfahren
    - In drahtlosen Netzen angepasste Mechanismen (CSMA/CA) notwendig

Datenkommunikation
Kapitel 4: Vermittlungsschicht

Klaus Wehrle

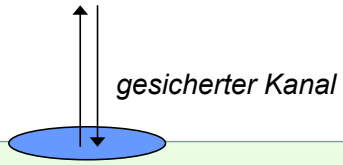
Communication and Distributed Systems
Chair of Computer Science 4
RWTH Aachen University

http://www.comsys.rwth-aachen.de

Themenübersicht

- **Datenkommunikation**
  - Einführung, Begriffe und allgemeine Grundlagen
  - Technische und nachrichtentechnische Grundlagen: Medien, Signale, Bandbreite, Leitungscodes und Modulation, Multiplexing
  - Lokale Netze: Strukturierung des Datenstroms, Fehlererkennung/ -behebung, Flusssteuerung, Medienzugriff, Ethernet
  - Vermittlungsschicht
    - Leitungsvermittlung
    - Internet und Internet-Protokolle:
    - Routing
  - Transportschicht
    - TCP und Co.
  - Anwendungsschicht

Aufgaben der Vermittlungsschicht



Schicht 2:
Gesicherter Übertragung von Rahmen

- Blockbildung zur Synchronisation und zur Strukturierung des Datenstroms
- Erkennung und Behebung von Datenverlust/-verfälschung
- Flusskontrolle und Pufferung
- Medienzugriffskontrolle bei geteilten Medien

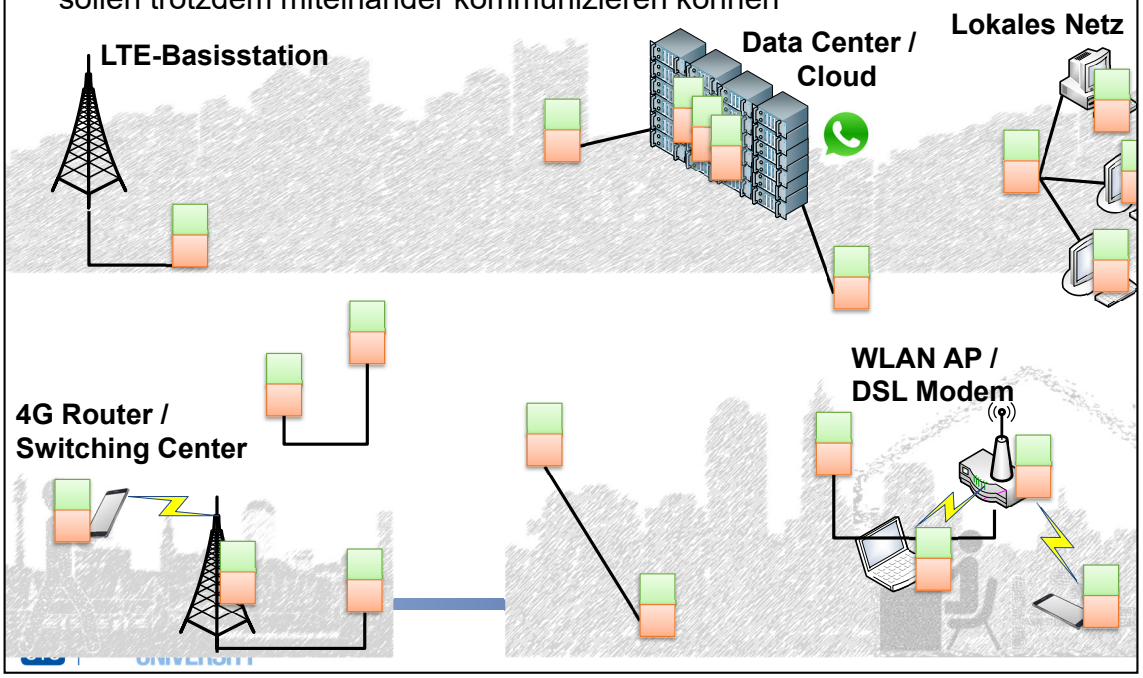
Die Vermittlungsschicht ist am engsten mit dem Begriff eines Netzwerkes verbunden und wird deshalb auch oft als Netzwerkschicht bezeichnet. Im TCP/IP-Modell heißt sie „Internet-Schicht“ (= die Schicht, die Kommunikation über Netzgrenzen hinweg ermöglicht). Dies liegt darin begründet, dass erst diese Schicht die lokalen Punkt-zu-Punkt-Verbindungen, d.h. die einzelnen physikalischen Links zwischen benachbarten Netzwerkknoten, verknüpft und Verbindungen zwischen entfernten Systemen über eine Vielzahl von Zwischensystemen (Vermittlungsknoten, typischerweise Router) hinweg erlaubt *(Endsystemverbindung).* Hierzu stellt die Vermittlungsschicht eine Adressierung zur Verfügung, die die weltweit eindeutige Identifikation einzelner Systeme ermöglicht (in der Praxis IP-Adressen) und anhand derer eine geeignete Wegewahl durchgeführt werden kann. Im lokalen Netz (bzw. auf jeder Teilstrecke) muss diese Adressierung dann von der Vermittlungsschicht wieder auf die MAC-Adressen der Schicht 2 abgebildet werden.

Die Datenübertragung selbst findet weiterhin über die unteren Schichten 2 und 1 statt.

Viele der Protokollmechanismen der Schicht 2 können sich auch in der Vermittlungsschicht wiederfinden, so z.B. auch Verfahren zur Flusskontrolle und Datensicherung. Hierbei muss man sich den veränderten Bezug dieser Mechanismen vor Augen halten: im Gegensatz zur Schicht 2 sind die hierbei beteiligten Systeme nicht benachbart, sondern möglicherweise Tausende von Kilometern voneinander entfernt und es müssen neue Fehlerquellen berücksichtigt werden, die durch das Zusammenwirken unterschiedlicher Netze sowie durch die Vermittlungsknoten entstehen. Achtung: diese Mechanismen werden hier kaum behandelt und sind auf obiger Folie mit „Eventuell: …“ bezeichnet, da diese Funktionen in der Praxis (IP) keine Verwendung finden.

Vermittlungssysteme

- Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können

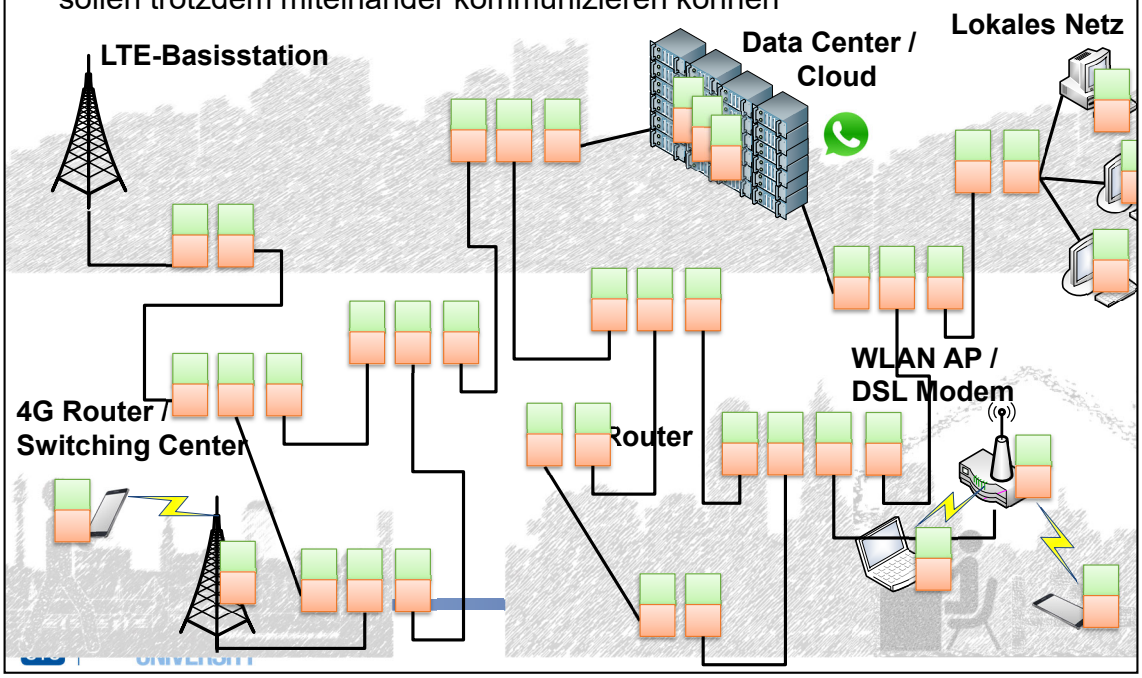


Die Hauptaufgabe der Vermittlungsschicht ist die Schaffung von Endsystemverbindungen zwischen beliebigen Endsystemen im weltweiten Netzwerk. Dazu ist die Kopplung aller lokalen Netze notwendig. Dies ließe sich zwar auf Schicht 2 durch Brücken erreichen, die auch eine Protokollwandlung vornehmen können, aber aus verschiedenen Gründen wäre dies problematisch.

Daher findet eine tatsächliche Kopplung von Netzen erst auf Schicht 3 statt, auf der „Vermittlungsknoten“ – im Internet: Router – zur Verbindung von LANs und zur Weiterleitung von Daten zwischen beliebigen LANs eingesetzt werden.

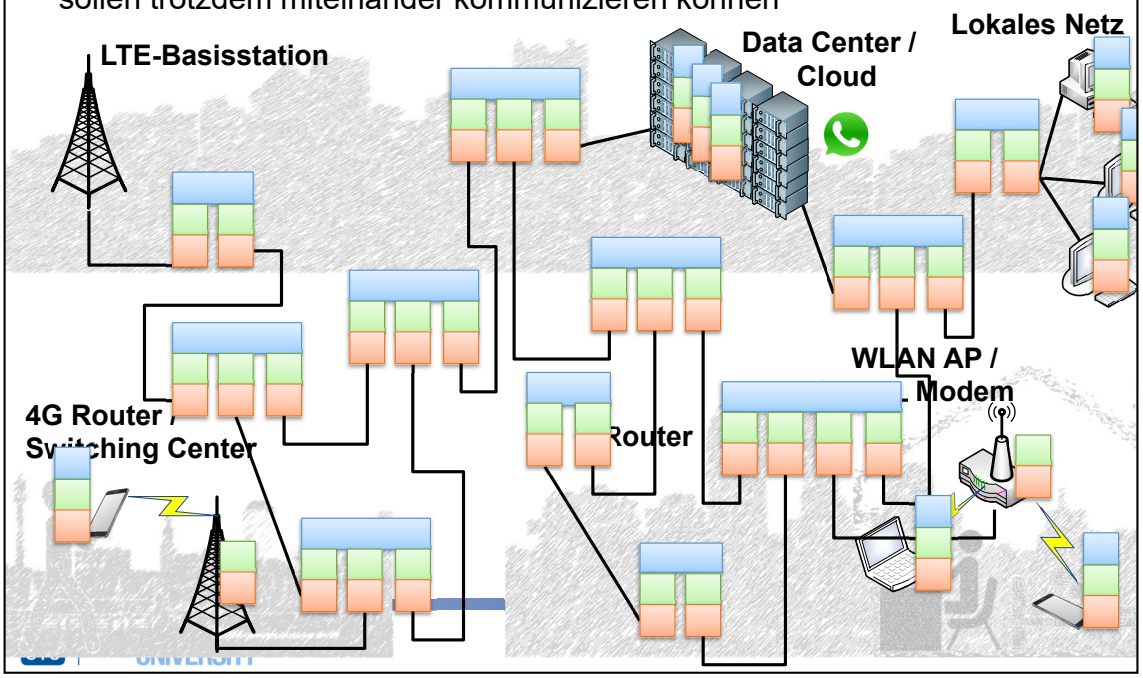
Vermittlungssysteme

- Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können



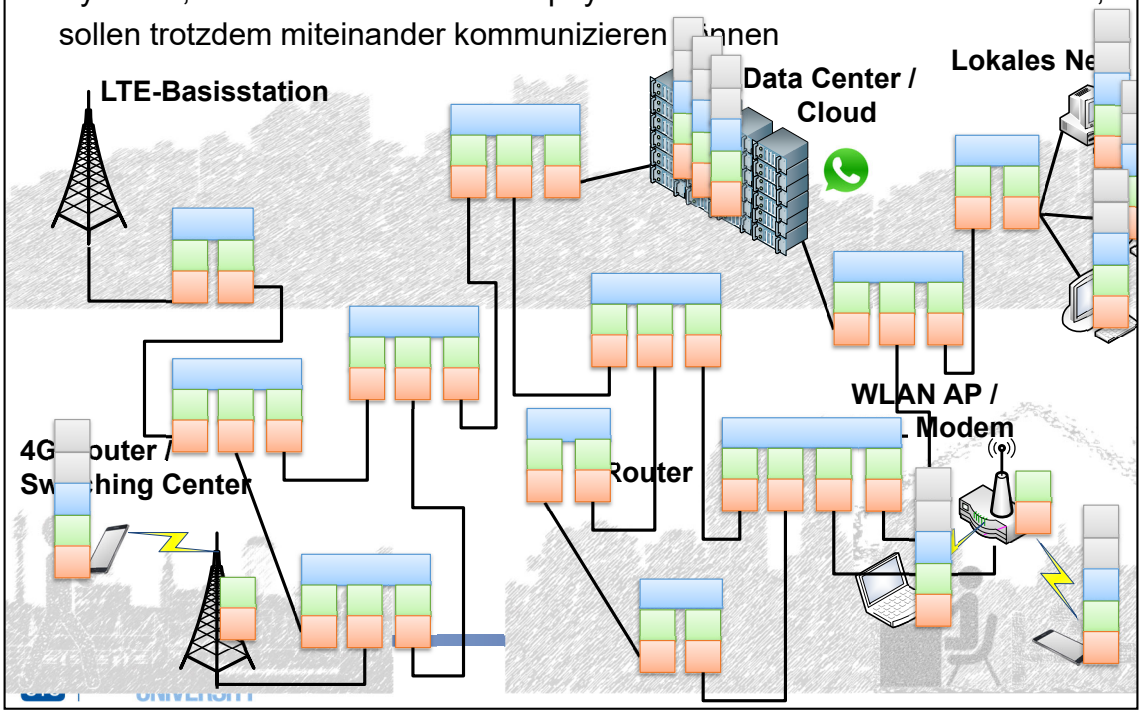
Vermittlungssysteme

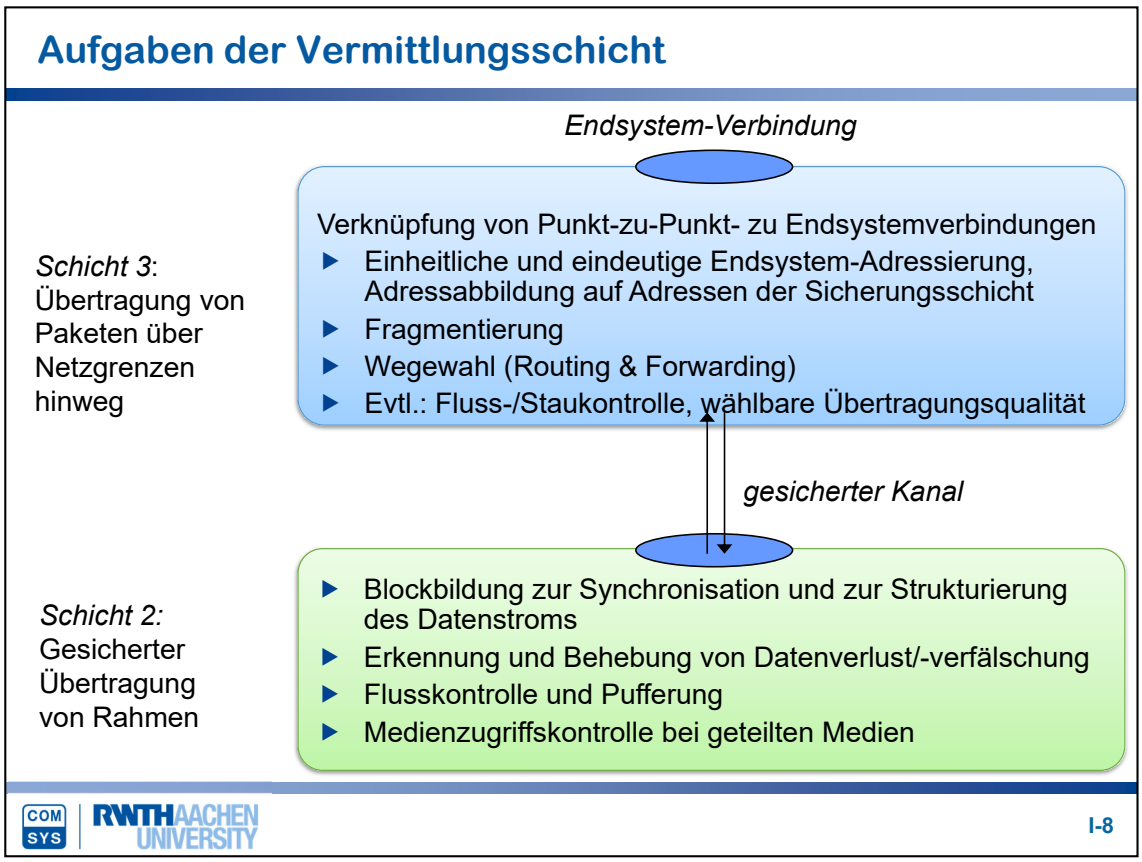
- Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können



Vermittlungssysteme

- Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können





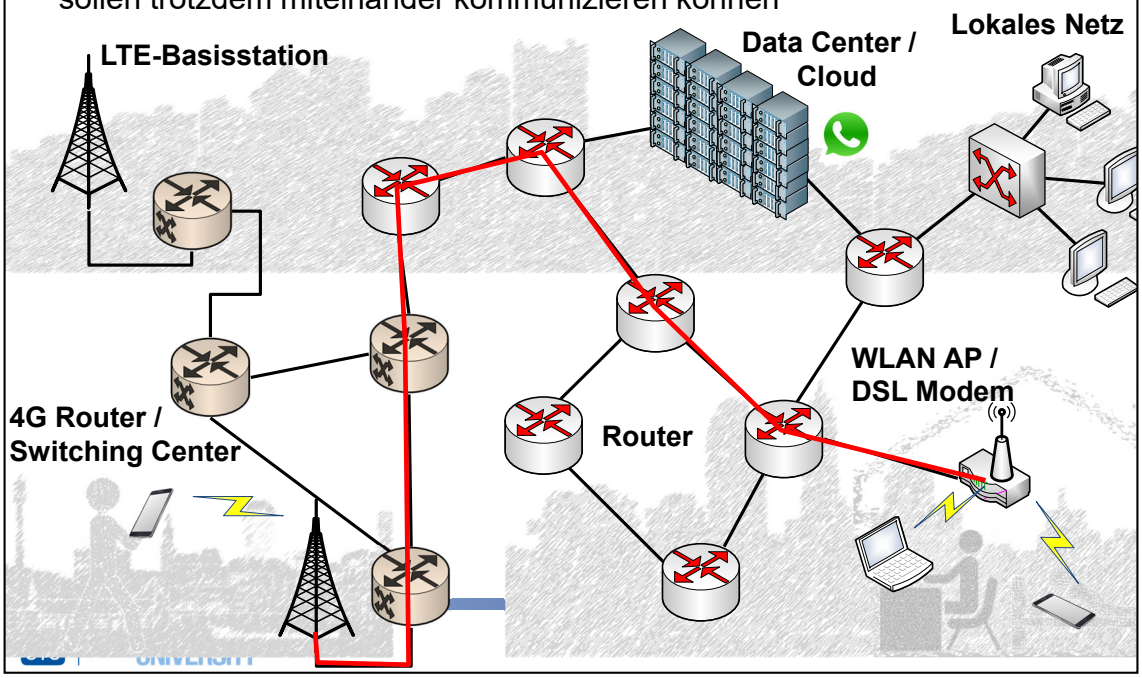
Die Vermittlungsschicht ist am engsten mit dem Begriff eines Netzwerkes verbunden und wird deshalb auch oft als Netzwerkschicht bezeichnet. Im TCP/IP-Modell heißt sie „Internet-Schicht“ (= die Schicht, die Kommunikation über Netzgrenzen hinweg ermöglicht). Dies liegt darin begründet, dass erst diese Schicht die lokalen Punkt-zu-Punkt-Verbindungen, d.h. die einzelnen physikalischen Links zwischen benachbarten Netzwerkknoten, verknüpft und Verbindungen zwischen entfernten Systemen über eine Vielzahl von Zwischensystemen (Vermittlungsknoten, typischerweise Router) hinweg erlaubt *(Endsystemverbindung).* Hierzu stellt die Vermittlungsschicht eine Adressierung zur Verfügung, die die weltweit eindeutige Identifikation einzelner Systeme ermöglicht (in der Praxis IP-Adressen) und anhand derer eine geeignete Wegewahl durchgeführt werden kann. Im lokalen Netz (bzw. auf jeder Teilstrecke) muss diese Adressierung dann von der Vermittlungsschicht wieder auf die MAC-Adressen der Schicht 2 abgebildet werden.

Die Datenübertragung selbst findet weiterhin über die unteren Schichten 2 und 1 statt.

Viele der Protokollmechanismen der Schicht 2 können sich auch in der Vermittlungsschicht wiederfinden, so z.B. auch Verfahren zur Flusskontrolle und Datensicherung. Hierbei muss man sich den veränderten Bezug dieser Mechanismen vor Augen halten: im Gegensatz zur Schicht 2 sind die hierbei beteiligten Systeme nicht benachbart, sondern möglicherweise Tausende von Kilometern voneinander entfernt und es müssen neue Fehlerquellen berücksichtigt werden, die durch das Zusammenwirken unterschiedlicher Netze sowie durch die Vermittlungsknoten entstehen. Achtung: diese Mechanismen werden hier kaum behandelt und sind auf obiger Folie mit „Eventuell: …“ bezeichnet, da diese Funktionen in der Praxis (IP) keine Verwendung finden.

Vermittlungssysteme

- Systeme, die nicht direkt durch ein physikalisches Medium verbunden sind, sollen trotzdem miteinander kommunizieren können



Vermittlung

- **Aufgabe: Bereitstellung eines Kommunikationswegs durch das Netz auf Anforderung**
  - Schicht-3-Verbindung (*Endsystemverbindung*)
    - Temporäre oder permanente Bereitstellung möglich
    - Mit oder ohne Zustand
  - Über eine Kette von *Vermittlungsstellen (Routern)*
    - Zwischen Vermittlungsstellen Austausch von Kontrollnachrichten (*Signalisierung*)
  - Arten der Vermittlung:
    - Leitungsvermittlung oder *Speicher-/Paketvermittlung*
    - *Verbindungsorientierte/verbindungslose* Vermittlungstechniken

Aufgabe der Vermittlung ist es also, zum Zwecke einer Datenübertragung zwischen zwei Teilnehmern einen geeigneten Kommunikationsweg über eine Kette von Vermittlungsstellen bereitzustellen. Dies kann entweder verbindungsorientiert erfolgen (Verbindungsaufbauphase vor dem Datenaustausch, in der der Übertragungsweg festgelegt wird) oder verbindungslos (jedes Datenpaket wird isoliert betrachtet und zum Empfänger weitergeleitet).

Verbindungen auf Schicht 3 werden auch Endsystemverbindungen genannt, da sie zwei entfernte Endsysteme (Teilnehmer) koppeln. Wie wir in Kapitel 5 sehen werden, dient erst die Transportschicht (Schicht 4) dazu, innerhalb eines Endsystems nochmals zwischen einzelnen Anwendungen zu differenzieren (Ende-zu-Ende-Verbindungen).

Als Signalisierung wird die vermittlungstechnische Kommunikation zwischen einem Endgerät (Rechner) und dem Netz sowie die Kommunikation im Netzinneren bezeichnet.

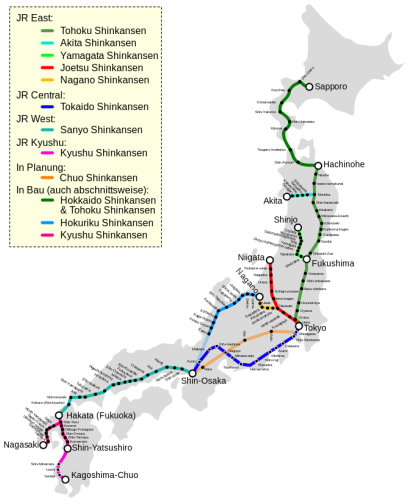
In der Telekommunikation unterscheidet man zwei grundsätzlich verschiedene Formen der Vermittlung:

- Durchschaltevermittlung (Leitungsvermittlung)
- Speichervermittlung bzw. Paketvermittlung

Diese beiden Vermittlungsarten werden auf den nächsten Folien erläutert.

Leitungs- und Paketvermittlung

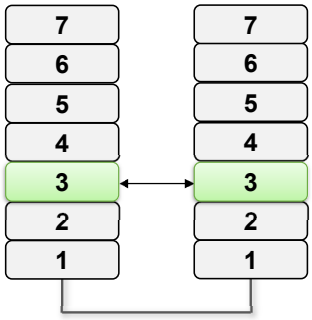
- **Leitungsvermittlung**
  - Geplantes Schienennetz
  - Pro Strecke eine Linie



- **Paketvermittlung**
  - Bekannte Streckenführungen
  - Keine globale Planung
    - Z.B. Verwendung eines Navis, um dem für ein Fahrzeug aktuell besten Weg zu folgen

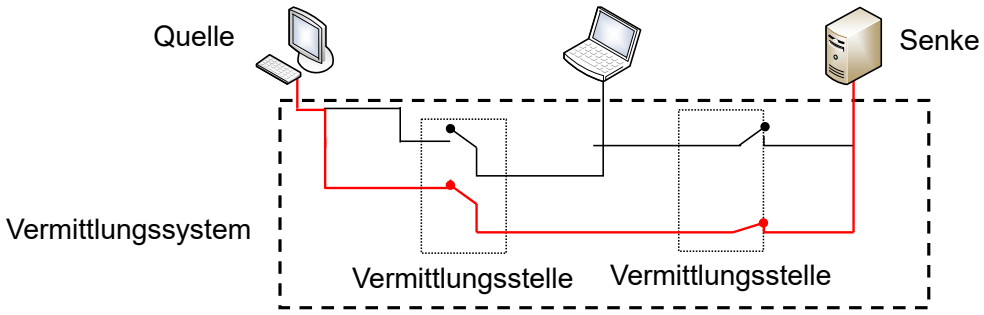
https://de.wikipedia.org/wiki/Datei:Shinkansen_Zukunft.svg#/media/File:Shinkansen_Zukunft.svg
https://www.verkehr.nrw/

- **Vermittlungsverfahren**
  - Leitungsvermittlung, Speicher-/Paketvermittlung
  - Beispiel ATM
- **Die Vermittlungsschicht im Internet – IP**
  - IPv4: Adressen, Subnetze, CIDR, NAT
  - IPv4-Header
  - Hilfsprotokolle: ARP, DHCP, ICMP
  - IPv4 vs. IPv6
- **Wegewahlverfahren (Routing)**
  - Hierarchien, einfache Verfahren
  - Distance Vector
  - Link State



Leitungsvermittlung (Circuit Switching)

- **Schalten einer *durchgängigen, dedizierten Verbindung***
  - Von der Quelle zur Senke über das Vermittlungssystem
  - Verbindung besteht aus nachrichtentechnischen Kanälen
    - Geschaltet für Dauer der Verbindung
    - Reihenfolgetreue Übertragung garantiert
  - Z.B. klassisches Telefonnetz



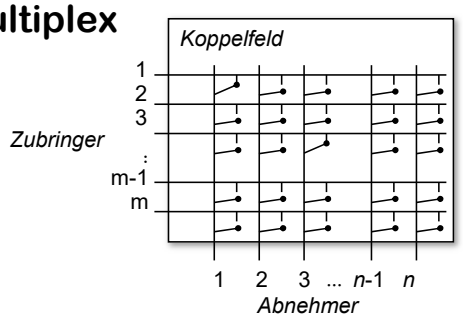
Bei der Leitungsvermittlung (engl. Circuit Switching, anderer Begriff: Durchschaltevermittlung) wird durch das Vermittlungssystem (d.h. die beteiligten Vermittlungsstellen) eine durchgängige dedizierte Verbindung von der Quelle zur Senke geschaltet. Dazu existiert eine Verbindungsaufbauphase vor dem Datenaustausch, in der die Verbindung fest geschaltet wird (in dieser Phase werden auch die wichtigsten Steuerinformationen ausgetauscht, wie z.B. Adressierungs- und Abrechnungsinformation).

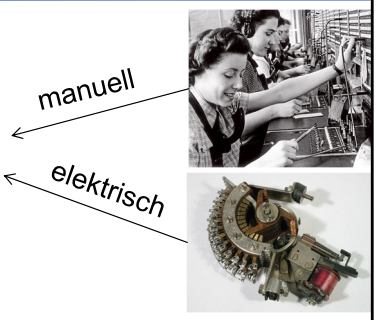
Das bekannteste Beispiel eines Kommunikationsnetzes, welches auf der Basis der Leitungsvermittlung arbeitet, ist das Telefonnetz. Wichtig ist, dass die nachrichtentechnischen Kanäle, die Bestandteil der geschalteten Verbindung sind, während der gesamten Dauer der Verbindung ausschließlich von den beteiligten Teilnehmern genutzt werden.

Die Kernfunktionalität der Leitungsvermittlung besteht in der Kopplung von Medien, und hier lassen sich zwei grundlegende Verfahren unterscheiden, wie auf der nächsten Folie gezeigt wird.

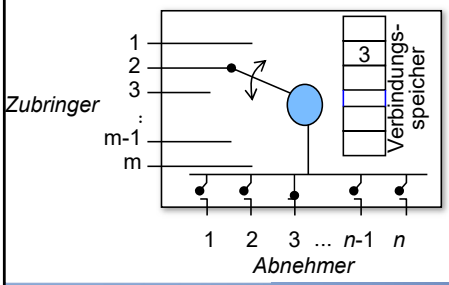
Leitungsvermittlung: Medienkopplungsverfahren

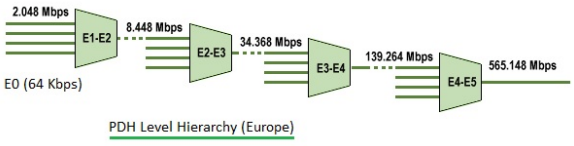
- **Raummultiplex**





- **Zeitmultiplex**





COMSYS | RWTH AACHEN UNIVERSITY IV-14

Die verallgemeinerte Aufgabenstellung der Medienzuordnung besteht in der Kopplung von *m* Zubringerleitungen an *n* Abnehmerleitungen.

Durch die Verwendung eines sogenannten *Raummultiplex*-Koppelfeldes kann jede Zubringerleitung auf beliebige Abnehmerleitungen geschaltet werden, indem die entsprechenden Schaltelemente der Matrix geschlossen werden. Diese beliebige Medienzuordnung ist möglich, weil jedes Schaltelement einzeln und unabhängig voneinander angesteuert werden kann.

Sehr viel eingeschränktere Möglichkeiten bietet hingegen das *Zeitmultiplex*-Verfahren, bei welchem zu jedem Zeitpunkt immer nur eine Zubringerleitung mit einer Abnehmerleitung nach Wahl verbunden werden kann. Jeder Zubringerleitung wird ein Zeitschlitz zugeordnet.

Im Verbindungsspeicher ist zu jeder Zubringerleitung vermerkt, auf welche Abnehmerleitung geschaltet werden soll.

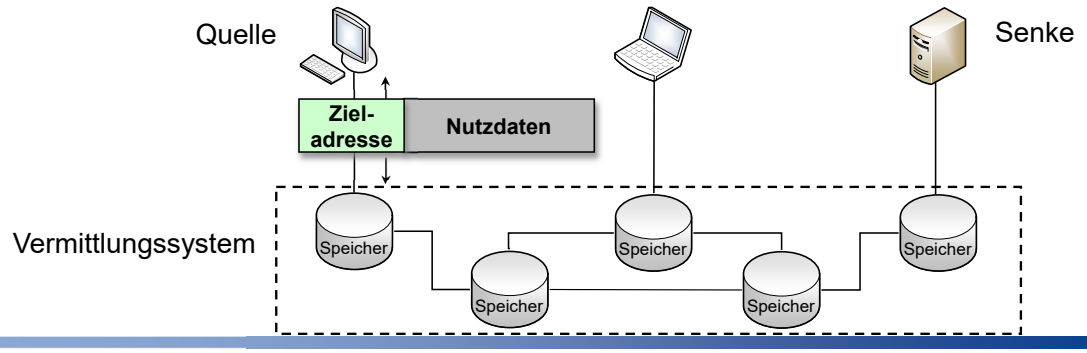
Für beide Verfahren gilt: Ist ein durchgehender Übertragungskanal (eine Leitung) zwischen den Teilnehmern aufgebaut, so sind Übertragungsverzögerungen auf physikalisch bedingte signaltechnische Laufzeiten beschränkt. Bitfolgen werden reihenfolgetreu übertragen, damit wird die Reihenfolge von PDUs des Absenders beim Empfänger beibehalten. Dies mag logisch klingen, doch bei anderen Verfahren, die im Folgenden noch erläutert werden, kann es durchaus vorkommen, dass PDUs beim Empfänger nicht in der korrekten Reihenfolge ankommen – diese Eigenschaft ist also ein klarer Vorteil von Leitungsvermittlung.

Streng genommen realisiert die Zeitmultiplex-Variante keine echte Leitungsvermittlung mehr, da Daten nicht direkt weitergeleitet werden können, sondern solange zwischengespeichert werden müssen, bis der passende Zeitslot zur Weiterleitung beginnt. Dies ist bereits ein Element der Speichervermittlung. Wird allerdings eine statische Zuordnung von Zeitslots zu Zubringerleitungen vorgenommen, behält die Weiterleitung trotz der Zwischenspeicherung alle Eigenschaften der Leitungsvermittlung – man kann dies als Vermittlung auf einer *virtuellen Leitung* bezeichnen.

Bei Verwendung des Zeitmultiplex gibt es auch die Möglichkeit, Zeitlots anforderungsgesteuert zuzuweisen (statistisches Zeitmultiplex). Dies führt uns zur Speichervermittlung.

Speicher-/Paketvermittlung (Packet Switching)

- „*Store & Forward*“-Prinzip
  - Vermittlungsstellen puffern Daten (Pakete) und leiten sie später weiter
    - Zeitlich unabhängige Weiterleitung je Paket
  - Keine „Besetzt“-Fälle durch fest geschaltete Leitungen
  - Bessere Bandbreitenausnutzung
  - Verlust von Daten, Zustellung in falscher Reihenfolge möglich



Bei der Speichervermittlung werden die zu vermittelnden Daten in einem Netzknoten zwischengespeichert und bei freier Teilstrecke zu einem geeigneten benachbarten Netzknoten weitergeschickt, bis sie bei der Senke angekommen sind.

Im Englischen heißt diese Form der Vermittlung treffend *Store-and-forward Switching*, was genau die beiden Aktionen umfasst (speichern und weiterschicken), die ein Netzknoten auszuführen hat. Ein Beispiel für ein nicht rechnergestütztes Transportsystem, das gemäß dem Store-and-Forward-Prinzip arbeitet, ist die Briefpost, wobei die Knoten die Postämter (bzw. Brief- und Paketzentren) sind.

- Bei der Speichervermittlung treten *keine Besetztfälle* durch für andere geschaltete Leitungen auf, sondern nur eventuell Wartezeiten: ist die ausgehende Leitung besetzt oder stark belastet, so werden die Daten zwischengespeichert und erst verzögert weitergegeben.
- Bei der Speichervermittlung kann es durchaus zu *Reihefolgevertauschungen* von Daten kommen, z.B. durch Verwendung unterschiedlicher Routen für verschiedene Dateneinheiten. Es besteht keine Zeitbeziehung zwischen den Dateneinheiten!
- Man kann *variable Datenraten* benutzen, d.h. hat man z.B. burstartigen Verkehr, so kann bei einem Burst – vorausgesetzt, das Netz ist nicht stark belastet – die gesamte Kapazität ausgenutzt werden, bei Sendepausen steht das Netz anderen zur Verfügung. Im Gegensatz dazu hätte man bei der Leitungsvermittlung dauerhaft eine feste Datenrate zur Verfügung.
- Während bei der Leitungsvermittlung eine Leitung durchgeschaltet ist, der alle Daten folgen, muss bei der Speichervermittlung in jeder Dateneinheit der Empfänger *adressiert* werden, damit die Vermittlungsstelle die einzelnen Pakete zustellen kann – es besteht kein Bezug zwischen den verschiedenen Dateneinheiten eines Senders.
- Es kann auch zu *Verlusten von Daten* kommen, wenn ein Zwischenknoten mehr Dateneinheiten zugestellt bekommt, als er weiterleiten kann und irgendwann sein Speicher überläuft. Daher müssen bei den Endsystemen mehr Protokollmechanismen vorhanden sein als bei Leitungsvermittlung (Fehlererkennung und -behandlung, Reihenfolgeerhaltung, Paketverluste, ...).

Eine Ausprägung der Speichervermittlung stellt die sogenannte Paketvermittlung (engl. Packet Switching) dar. Bei diesem Verfahren werden Daten in Form einzelner Pakete (= PDU auf Schicht 3) übertragen (dabei ist entweder eine feste Länge oder eine maximale Länge vorgegeben) und in den Vermittlungsstellen entlang des Übertragungsweges zwischengespeichert. Um eine Weiterleitung zu ermöglichen, müssen die Pakete als PCI zumindest die Adresse des Zielrechners enthalten. Die weitergehenden Möglichkeiten, Paketvermittlung zu betreiben (verbindungsorientiert oder verbindungslos) werden im Folgenden erläutert.

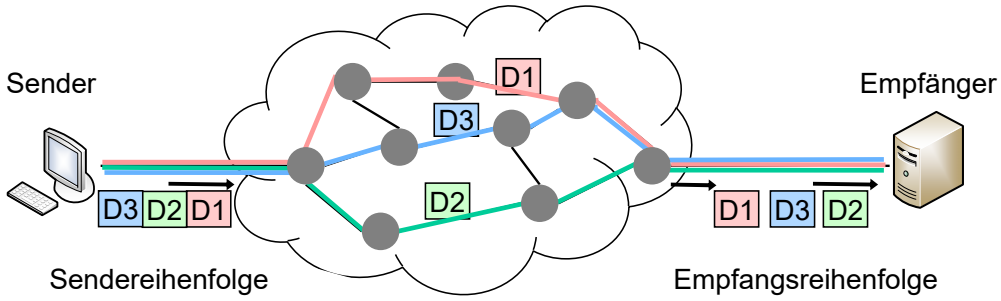
Verbindungslose Paketvermittlung

- ***Pakete*** **bilden geschlossene Vermittlungseinheiten**
  - U.A. mit Quell-/Zieladresse
  - Pakete mit gleicher Quell- und Zieladresse werden unabhängig voneinander weitergeleitet
- **Vermittlungsstellen besitzen keine Informationen über Verbindungskontext!**
  - \+ Verbindungsaufbau, -überwachung und -abbau entfallen
  - \+ Bessere Nutzung der Netzkapazität möglich
  - \- Einzelne Datagramme können unterschiedliche Wege nehmen
  - \- Die reihenfolgerichtige Auslieferung beim Empfänger ist nicht gesichert

Die verbindungslose Paketvermittlung basiert auf der isolierten Betrachtung jedes einzelnen Datenpakets. Jedes Paket enthält in den entsprechenden Kontrollfeldern im Paketheader ausreichende Adressierungsinformationen, die es den Vermittlungsstellen ermöglichen, Pakete dem gewünschten Empfänger zuzustellen. Verbindungsauf- und -abbau können somit komplett entfallen. Dies bringt aber auch Schwächen des Mechanismus‘ mit sich: da zwei Pakete des selben Paketstroms auf unterschiedlichen Pfaden zum Ziel geleitet werden können, kommen sie unter Umständen in falscher Reihenfolge beim Empfänger an. Zudem können keine Garantien wie bei der verbindungsorientierten Vermittlung gegeben werden, wenn nicht jedes Paket im Header alle notwendigen Parameter enthält – und selbst dann ist nicht gewährleitet, dass Garantien erfüllt werden können, da jeder Vermittlungsknoten für sich selbst entscheidet, wie ein Paket weiterzuleiten ist und nicht weiß, ob der Vermittlungsknoten, der das Paket erhält, auch in der Lage sein wird, die Garantien zu erfüllen.

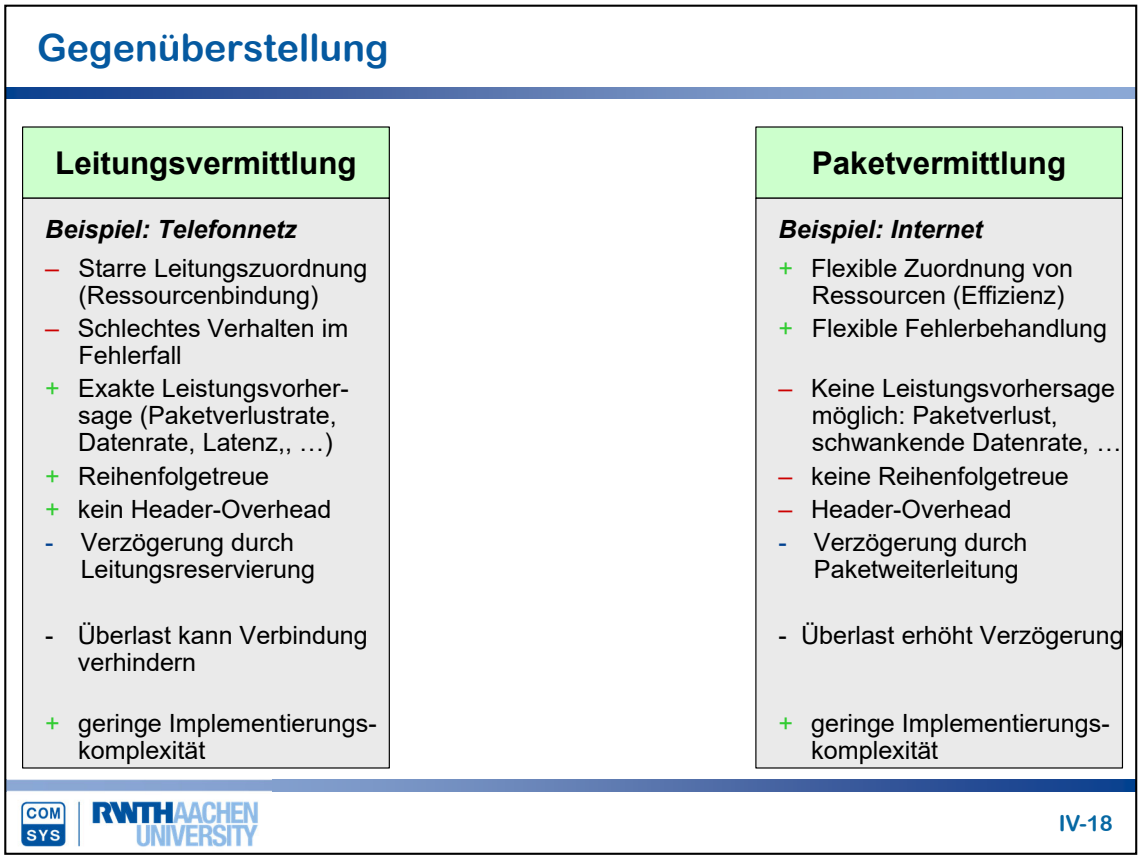
Verbindungslose Paketvermittlung – Ablauf

- **Verkehr in einem paketvermittelten Netz**
  - Jedes Paket mit eigenem Weg
  - Eine Reservierung von Verarbeitungskapazität ist nicht möglich



Dx Paket *x*

Paketvermittlungsstelle



Grundsätzlich gilt, dass der Vorteil des einen Verfahrens (Leitungs-/Paketvermittlung) der Nachteil des anderen ist und umgekehrt.

Die Leitungsvermittlung kann grob als starr und einfach, die Speichervermittlung als flexibel und komplex zusammengefasst werden. Der wesentliche Unterschied, welcher alle weiteren positiven und negativen Eigenschaften nach sich zieht, liegt in der Art, wie die Vermittlungsverfahren die Kommunikationsressourcen, sprich den nachrichtentechnischen Kanal, belegen:

Die Leitungsvermittlung nimmt eine feste Reservierung der Kommunikationsressourcen während der gesamten Verbindungsdauer vor, während die Speichervermittlung die Ressourcen immer nur für ein Paket bzw. eine Nachricht beansprucht.

Der Einsatz virtueller Leitungen kombiniert zwar die Vorteile beider Verfahren, aber auch ihre Nachteile. Zudem bringt dieses Verfahren eine höhere Komplexität mit sich und kann schnell ineffizient werden – siehe dazu wieder ATM: die geteilte Nutzung der Leitungen wird durch Zeitmultiplex ermöglicht, welches eine feste Zellgröße erfordert. Wählt man die Zellen zu klein, wird der Header-Overhead enorm groß und die Nutzdatenrate für Anwendungen geht in den Keller. Wählt man die Zellen zu groß, dauert die Weiterleitung einer einzelnen Zelle so lange, dass alle Zellen in den Switches stark verzögert werden können; zudem können die Ressourcen nicht mehr effizient genutzt werden, wenn Anwendungen die Zellen nicht komplett füllen können: niemand sonst kann die dadurch frei bleibende Netzkapazität nutzen. Dies war gerade das Problem mit ATM: die Zellgröße war ein Kompromiss wischen der datenorientierten Internetfraktion, die große Zellen für Datenpakete haben wollte, und der Sprachfraktion, die kleine Zellen für die Echtzeitübertragung von Sprachinformationen forderten. Die gewählte Zellgröße stellte beide Fraktionen nicht zufrieden, und speziell für die datenorientierte Übertragung war ATM höchst ineffizient, da ein großer Teil der Netzkapazität für Headerinformationen verloren ging.

Verbindungsorientierte Paketvermittlung

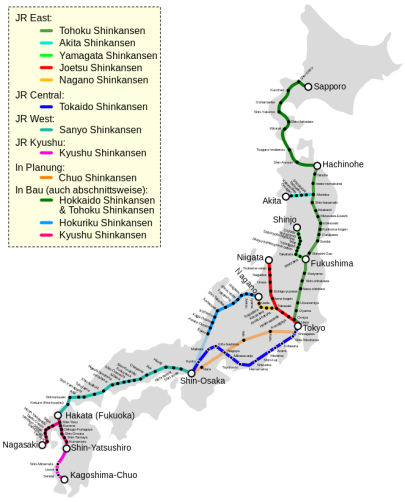
- ***Virtuelle Leitung*** **(Virtual Circuit, VC)**
  - Bidirektionaler fester Übertragungsweg (vollduplex)
    - Wird bei Verbindungsaufbau festgelegt
  - Kann mehrere (Paket-)Vermittlungsstellen umfassen
- **Kombiniere Vorteile von Circuit- und Packet Switching**
  - Aufbau einer Verbindung, ähnlich Leitungsvermittlung
    - Möglichkeit von Garantien, z.B. reihenfolgetreue Übertragung
  - Teilung von Leitungen wie bei Paketvermittlung
    - Ressourceneffizienz, Flexibilität

Bei der *verbindungsorientierten Paketvermittlung* findet vor dem eigentlichen Datenaustausch ein Verbindungsaufbau statt, der neben verschiedenen Verbindungsparametern (z.B. für die Dienstqualität) den Weg für alle zu dieser Verbindung gehörenden Pakete festgelegt. Für den Benutzer erscheint die Verbindung wie eine echte, durchgehende Leitung, obwohl es eine solche in der Realität nicht gibt - deshalb auch die Bezeichnung *virtuelle Verbindung*.

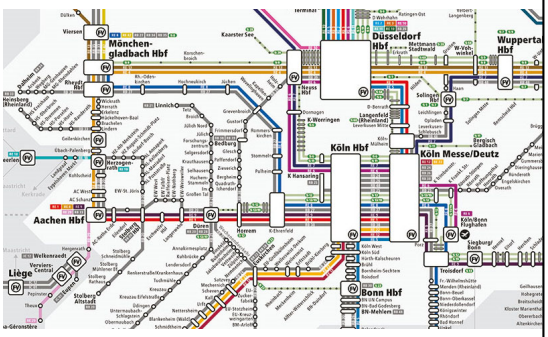
Wird ein Übertragungsweg vom Netzbetreiber längerfristig eingerichtet („Standleitung“), so spricht man auch von *fester virtueller Verbindung* (Permanent Virtual Circuit, PVC). Ist eine gesonderte Verbindungsaufbauprozedur erforderlich (Signalisierung), wird die Verbindung als *gewählte virtuelle Verbindung* bezeichnet (Switched Virtual Circuit, SVC).

Circuit Switching und Virtual Circuits

- **Leitungsvermittlung**
  - Geplantes Schienennetz
  - Pro Strecke eine Linie



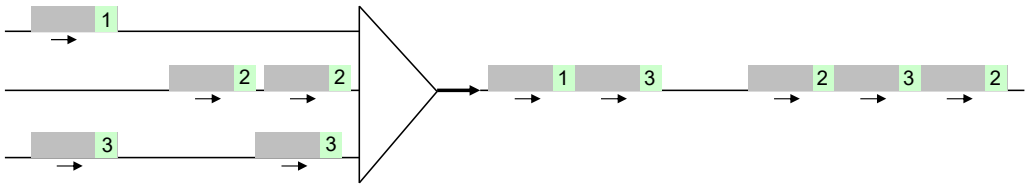
- **Virtuelle Leitung**
  - Geplantes Schienennetz
  - Eine Linie fährt immer die gleiche Strecke
    - Aber Linien können Teil-strecken gemeinsam nutzen



https://de.wikipedia.org/wiki/Datei:Shinkansen_Zukunft.svg#/media/File:Shinkansen_Zukunft.svg
http://www.kcitf-nrw.de/regionalverkehrsplan/

Virtual Circuit: Asynchronous Transfer Mode (ATM)

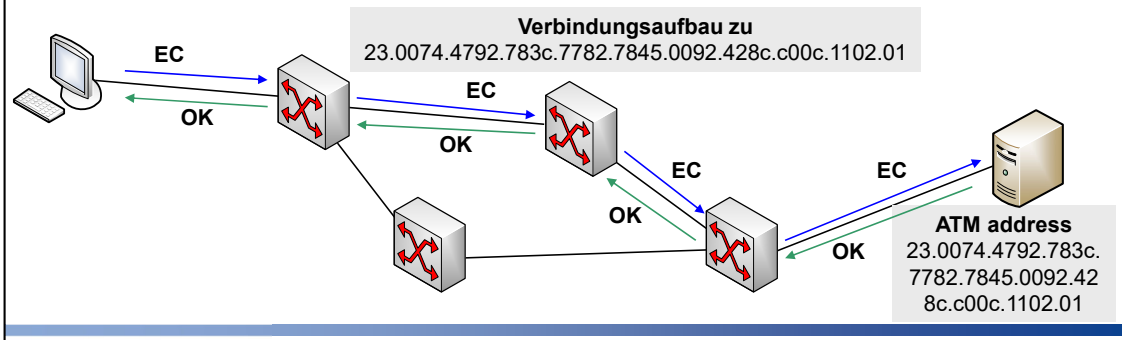
- **Zellenbasiertes Multiplexing und Switching**
  - *ATM-Zelle*: Paket fester Länge mit 5 Byte Header + 48 Byte Payload
  - Schaltet *virtuelle Verbindungen*
  - *Multiplexing* verschiedener Verbindungen auf eine Leitung
    - Konstante Zellgröße ermöglicht Weiterleitung in Zeitmultiplex-Raster
    - Reservierung von Kapazität möglich
    - *Admission Control* vor Annahme einer Verbindung, um Überlastung des Netzes zu verhindern
    - Bei Überlast werden Zellen verworfen



Ein Versuch, Leitungs- und Paketvermittlung zu kombinieren, wurde mit ATM vorgenommen. Typischerweise werden bei der Paketvermittlung die Ressourcen des Netzes geteilt zwischen mehreren Stationen genutzt und es ist schwer, konkrete Garantien zu geben, da Pakete unterschiedliche Längen haben können, wodurch indeterministische Verzögerungen bei der Weiterleitung von Paketen auftreten können. ATM verwendet Pakete fester Länge (Zellen), um ein Zeitmultiplexing mit fester Slotlänge einsetzen zu können. Dadurch ist bei Bedarf die Reservierung einer garantierten Bandbreite möglich. Mechanismen wie Admission Control (Zugangskontrolle) sorgen dafür, dass nicht zu viele Teilnehmer gleichzeitig Garantien erfragen und das Netz überlasten können.

Verbindungsaufbau bei ATM

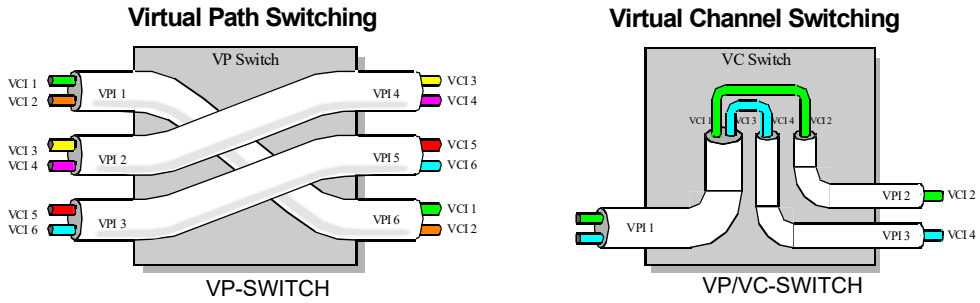
- **Zellweiterleitung durch Switches**
  - Anhand von *Verbindungs-IDs*, nicht Rechneradressen
  - Vor Beginn der Kommunikation: Verbindungsaufbau
    - Kontrollpaket ermittelt Pfad durchs Netz, Empfänger antwortet mit Quittung
    - Passierte Switches legen eine Verbindungs-ID an
      - Ermöglicht weniger (kleinere) Einträge in den Weiterleitungstabellen



Vor Beginn einer Kommunikation erfolgt ein Verbindungsaufbau, bei dem auch Reservierungen von Bandbreite in den einzelnen Switches vorgenommen werden können. Bei Verbindungsaufbau bekommt eine Kommunikationssitzung eine ID zugewiesen; eine Weiterleitung der Zellen erfolgt dann nicht mehr aufgrund einer Wegewahl anhand der Zieladresse: die Vermittlungsknoten agieren als Switches, die in einem festen Eintrag die ID einer physikalischen Leitung zuordnen und die folgenden Zellen der Verbindung immer über diese Leitung weiterleiten.

Verbindungs-IDs: Pfad und Kanal

- **Hierarchischer Aufbau der Verbindungs-IDs**
  - Identifier für virtuellen Pfad (*Virtual Path Identifier*, VPI)
  - Identifier für virtuellen Kanal (*Virtual Channel Identifier*, VCI)
  - Hierarchische Struktur ermöglicht Kombination mehrerer Routen-Informationen in der Weiterleitungstabelle durch Vergabe des gleichen VPI

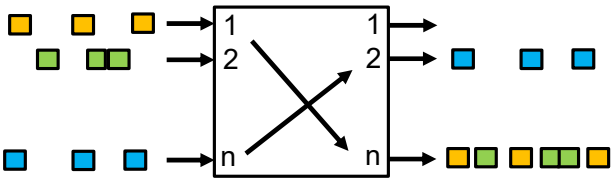


Ein Problem bei der Verwendung des Switchings ist die Explosion der Weiterleitungstabellen; für jede aktuell existierende Verbindung muss ein Switch einen eigenen Eintrag verwalten, was zu extrem großen Weiterleitungstabellen führt, die unperformant werden. Daher wird eine hierarchische ID-Struktur erstellt – soweit möglich, wird ein neuer Eintrag mit bereits existierenden Einträgen aggregiert, um die Anzahl der Weiterleitungseinträge zu minimieren.

Switching

- **Weiterleitungstabellen in den Switches**
  - `Old Header` enthält VPI/VCI auf dem vorherigen Teilstück
  - `New Header` enthält VPI/VCI zur Verwendung auf dem nächsten Teilstück
  - `In` und `out` bezeichnen die Netzwerkkarten, über die ein Paket eingeht / ausgesendet werden muss

Eingangsleitungen Switch Ausgangsleitungen



Switching-Tabelle

In	Old Header	Out	New Header	
1	a	n	a	
2	c	n	d	
…	…	…	…	
n	b	2	e	

Da die ID durch jeden Switch geändert wird, ist eine Modifikation des Headers notwendig. Zudem enthält der Header noch eine CRC-Checksumme, die der Korrektur von 1-Bit-Fehlern im Header dient. Diese CRC muss also vor der Weiterleitung auch neu berechnet werden. Dies führt zu einer Verzögerung der Zellen vor der Weiterleitung. Um den Prozess zu beschleunigen, werden die CRC-Prüfsummen bereits bei Verbindungsaufbau vorberechnet und mit in der Weiterleitungstabelle abgespeichert. Bei Empfang einer Zelle wird nun lediglich geprüft, auf welcher Leitung sie weitergesendet werden muss, der alte Zellheader verworfen und der vorberechnete Header aus der Weiterleitungstabelle eingefügt.

Gegenüberstellung

Leitungsvermittlung	Virtuelle Leitung	Paketvermittlung
Beispiel: Telefonnetz	***Beispiel: ATM***	***Beispiel: Internet***
– Starre Leitungszuordnung (Ressourcenbindung)	+ Flexible Zuordnung von Ressourcen (Effizienz)	+ Flexible Zuordnung von Ressourcen (Effizienz)
– Schlechtes Verhalten im Fehlerfall	~ Beschränkte Fehlerbehandlung	+ Flexible Fehlerbehandlung
+ Exakte Leistungsvorher-sage (Paketverlustrate, Datenrate, Latenz,, …)	+ Exakte Leistungsvorher-sage (Paketverlustrate, Datenrate, Latenz,, …)	– Keine Leistungsvorhersage möglich: Paketverlust, schwankende Datenrate, …
+ Reihenfolgetreue	+ Reihenfolgetreue	– keine Reihenfolgetreue
+ kein Header-Overhead	– Header-Overhead	– Header-Overhead
- Verzögerung durch Leitungsreservierung	– Verzögerung durch Verbindungsaufbau und Paketweiterleitung	- Verzögerung durch Paketweiterleitung
- Überlast kann Verbindung verhindern	– Überlast kann Verbindung verhindern und erhöht Verzögerung	- Überlast erhöht Verzögerung
+ geringe Implementierungs-komplexität	– hohe Implementierungs-komplexität	+ geringe Implementierungs-komplexität

Grundsätzlich gilt, dass der Vorteil des einen Verfahrens (Leitungs-/Paketvermittlung) der Nachteil des anderen ist und umgekehrt.

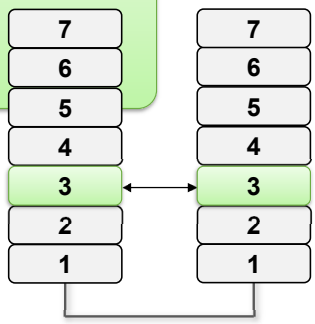
Die Leitungsvermittlung kann grob als starr und einfach, die Speichervermittlung als flexibel und komplex zusammengefasst werden. Der wesentliche Unterschied, welcher alle weiteren positiven und negativen Eigenschaften nach sich zieht, liegt in der Art, wie die Vermittlungsverfahren die Kommunikationsressourcen, sprich den nachrichtentechnischen Kanal, belegen:

Die Leitungsvermittlung nimmt eine feste Reservierung der Kommunikationsressourcen während der gesamten Verbindungsdauer vor, während die Speichervermittlung die Ressourcen immer nur für ein Paket bzw. eine Nachricht beansprucht.

Der Einsatz virtueller Leitungen kombiniert zwar die Vorteile beider Verfahren, aber auch ihre Nachteile. Zudem bringt dieses Verfahren eine höhere Komplexität mit sich und kann schnell ineffizient werden – siehe dazu wieder ATM: die geteilte Nutzung der Leitungen wird durch Zeitmultiplex ermöglicht, welches eine feste Zellgröße erfordert. Wählt man die Zellen zu klein, wird der Header-Overhead enorm groß und die Nutzdatenrate für Anwendungen geht in den Keller. Wählt man die Zellen zu groß, dauert die Weiterleitung einer einzelnen Zelle so lange, dass alle Zellen in den Switches stark verzögert werden können; zudem können die Ressourcen nicht mehr effizient genutzt werden, wenn Anwendungen die Zellen nicht komplett füllen können: niemand sonst kann die dadurch frei bleibende Netzkapazität nutzen. Dies war gerade das Problem mit ATM: die Zellgröße war ein Kompromiss wischen der datenorientierten Internetfraktion, die große Zellen für Datenpakete haben wollte, und der Sprachfraktion, die kleine Zellen für die Echtzeitübertragung von Sprachinformationen forderten. Die gewählte Zellgröße stellte beide Fraktionen nicht zufrieden, und speziell für die datenorientierte Übertragung war ATM höchst ineffizient, da ein großer Teil der Netzkapazität für Headerinformationen verloren ging.

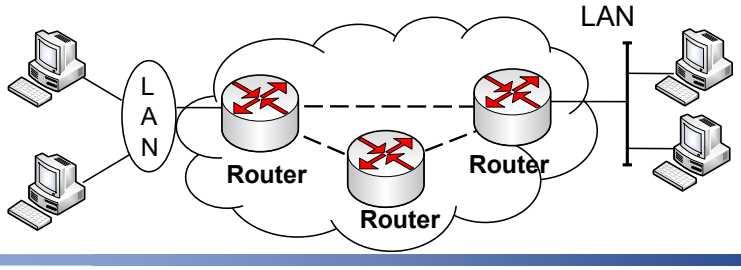
Kapitel 4: Vermittlungsschicht

- **Vermittlungsverfahren**
  - Leitungsvermittlung, Speicher-/Paketvermittlung
  - Beispiel ATM
- **Die Vermittlungsschicht im Internet – IP**
  - IPv4: Adressen, Subnetze, CIDR, NAT
  - IPv4-Header
  - Hilfsprotokolle: ARP, DHCP, ICMP
  - IPv4 vs. IPv6
- **Wegewahlverfahren (Routing)**
  - Hierarchien, einfache Verfahren
  - Distance Vector
  - Link State



Vermittlungsschicht im Internet: IP (Internet Protocol)

- **Historie:**
  - Entwickelt vom amerikanischen Verteidigungsministerium (Departement of Defense, DoD)
  - Bereits 1983 im damaligen ARPANET eingesetzt (anfangs vier Hosts)
- **Realisierung und Entwicklung:**
  - Verbreitet ist vorwiegend Version 4 (*IPv4*)
  - Weiterentwicklung im Projekt IPng (IP next generation) der IETF (Internet Engineering Task Force) zu *IPv6*



Die Entwicklung der Protokolle der TCP/IP-Familie geht auf eine Initiative des amerikanischen Verteidigungsministeriums (Department of Defense, DoD) in den 60er Jahren zurück, welche das primäre Ziel besaß, ein möglichst ausfallsicheres Netzwerk zu schaffen. Hierbei sollte die Kommunikation auch über große Entfernungen hinweg in Krisenzeiten (z.B. Atomkrieg) sichergestellt werden. Dahingehende Untersuchungen wurden von der ARPA (Advanced Research Project Agency) durchgeführt, die sich schließlich für ein paketvermitteltes Netzwerk entschied und das hierzu entwickelte Protokoll IP erstmals 1983 einsetzte (die beteiligten Rechner bildeten das sogenannte ARPANET). Nach der Abspaltung des militärischen Teils in das sogenannte MILNET entstand aus dem ARPANET schließlich das heute als Internet bekannte Netzwerk.

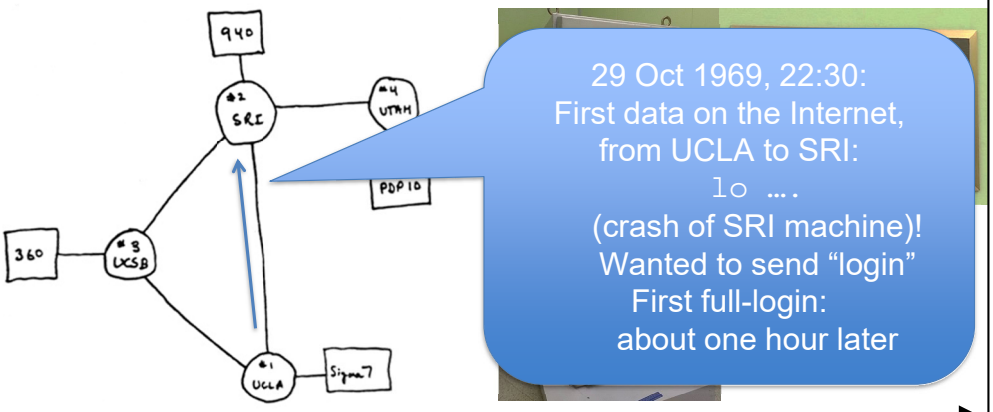
Von da an entwickelte sich das Internet weiter wie kein anderes Netzkonzept mit dem Ergebnis, dass es heute ein weltweites (Daten-)Kommunikationsnetz ist, mit über einer Milliarde angeschlossener Rechner.

Die ursprüngliche entwickelte Version von IP war IPv4 (Versionen 1 – 3 gab es nie). Diese Version ist jetzt bereits seit ungefähr 35 Jahren im Einsatz. Die Entwicklung von IP ging allerdings weiter, die letzten größeren Arbeiten an IP erfolgten im Rahmen der Arbeitsgruppe „IP Next Generation“, aus der IPv6 hervorging (eine Version 5 gibt es ebenfalls nicht).

Die zentrale Aufgabe des IP-Protokolls bzw. der dieses Protokoll realisierenden Protokollinstanzen ist die Zustellung von Daten an einen entfernten Rechner – dazu definiert es ein Adressschema, Regeln zur verbindungslosen Paketverarbeitung und -weiterleitung sowie entsprechende Headerinformationen, die eine vernünftige Weiterleitung ermöglichen.

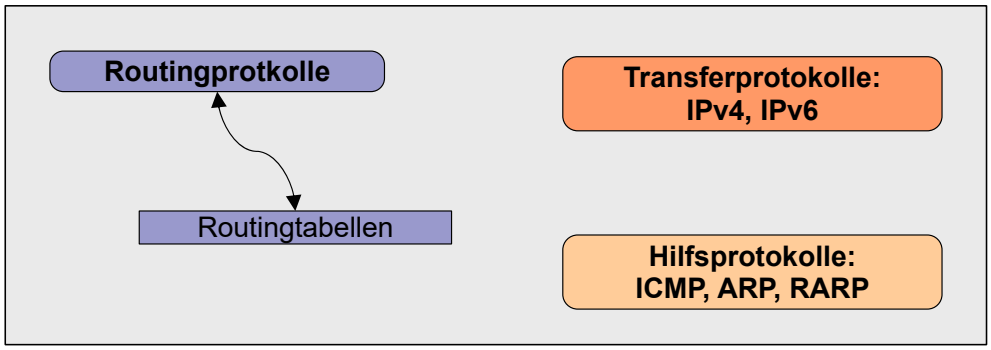
A Look into History: Research Network

- **"Initial Internet": Research Network**
  - ARPANET: UCLA, Stanford, UCSB, and Utah (one computer per site)
  - Decentralized connectivity of distributed, independent systems
    - Can operate and communicate in the presence of connection/node failures



Vermittlungsschicht im Internet

- **Grobe Aufteilung in drei Aufgabenbereiche:**
  - *Datentransfer* über ein globales Netz: Adressierung, Paketformat, Regeln zur Paketverarbeitung (Forwarding)
  - *Wegewahl* durch die Zwischenknoten (Routing)
  - *Hilfsprotokolle* z.B. zum Austausch von Informationen über den Netzstatus (Fehlermeldungen, Signalisierung)



IP wirkt zusammen mit einigen Hilfs- und Kontrollprotokollen, die Teilaufgaben wahrnehmen, sowie mit Routing-Protokollen, die festlegen, wie die Router (= Vermittlungsstellen im Internet) untereinander Informationen austauschen können, um bei der Weiterleitung eines Pakets eine Wegewahl zu treffen. Die Weiterleitung der Pakete selbst wird als Forwarding bezeichnet.

Eigenschaften des Internet Protocol (IPv4)

- **Paketvermittelt, *verbindungslos***
  - Ungesicherte Übertragung:
    - Ein Paket kann verloren gehen
    - Ein Paket kann dupliziert werden
    - Pakete können in falscher Reihenfolge ankommen
    - Pakete können endlos im Netz zirkulieren
  - Nicht behebbare Fehler der zugrundeliegenden Schicht 2 können von IP im Allgemeinen ebenfalls nicht behandelt werden
    - Mit dem Protokoll *ICMP (Internet Control Message Protocol)* existiert jedoch eine Möglichkeit zur Fehleranzeige
  - Keine Flusskontrolle, keine Staukontrolle

Die zentrale Eigenschaft von IP besteht darin, dass es sich um ein unzuverlässiges, verbindungsloses Protokoll handelt. Dementsprechend werden also Probleme wie Paketverlust, Duplizierung von Paketen oder Reihenfolgeumstellung im Paketstrom von IP nicht behoben. Diese Aufgabe wird den höheren Schichten überlassen.

Selbst Fehler, die in der Schicht 2 nicht behoben werden konnten und deshalb nach oben in die IP-Schicht durchschlagen, werden meist unbehandelt nach oben zur Transportschicht weitergegeben.

Aufgrund der Verbindungslosigkeit bietet IP auch keine Flusskontrolle.

Die einzige Form der Reaktion auf erkannte Fehlersituationen kann darin bestehen, dass eine IP-Instanz dem Sender des IP-Pakets über das Protokoll ICMP einen aufgetretenen Fehler mitteilt. Diese Information kann für den Sender wichtig sein, damit er weiß, warum seine Nachricht nicht beim Empfänger angekommen ist.

IP kann sowohl zur Kopplung von Teilnetzen z.B. in einem lokalen unternehmensweiten Netz genutzt werden, als auch zum Aufbau eines öffentlichen Netzes, wie es das Beispiel des Internet deutlich macht – der Trend hat sich sogar dahin entwickelt, alles über IP laufen zu lassen, also z.B. auch Telefonnetze auf IP umzustellen.

IP: Aufgaben

- **Transparente Ende-zu-Ende-Kommunikation zwischen Rechnern auch über unterschiedliche Netztypen hinweg**
  - Bereitstellung *weltweit eindeutiger Adressen*
    - IPv4: 32 Bit
    - Hierarchische Struktur
  - Definition von *Verarbeitungs-/Weiterleitungsregeln* samt eines zugehörigen *Paketformats*
    - Header mit Kontrollinformationen
    - Maximale Paketgröße: 64 kByte (in der Praxis: 1500 Byte)
  - Zusammenspiel mit *Routing* (Wegermittlung)
  - Zusammenspiel mit den tieferen Schichten (*ARP*)

IP-Adressen

- **Eindeutige IP-Adressen für jeden Host und jeden Router**
  - IP-Adressen (IPv4) sind 32 Bit lang
    - *Dotted Decimal Notation*: Darstellung einer Adresse als 4 Dezimalwerte, die jeweils 8 Bit der Adresse entsprechen
  - *Hierarchisch strukturiert und netzbezogen*, d.h. Maschinen mit Anschluss an mehrere Netze haben mehrere IP-Adressen
  - Struktur der Adresse: *Netzwerk*-Anteil für physikalisches Netz (z.B. **137.226**.0.0) und *Host*-Anteil für einen Rechner (z.B. 137.226.**12.221**) in diesem Netz
  - Um unterschiedlich große Netze installieren zu können, wurden ursprünglich mehrere Adressklassen definiert
    - Heute wird zwar noch die Terminologie verwendet, aber nicht mehr die Klassenaufteilung!

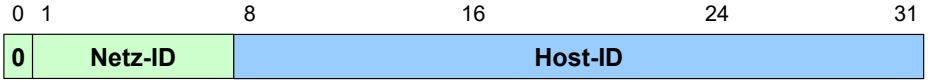
Um IP-Pakete weiterleiten zu können, muss ein Router alle möglichen Endsysteme kennen, und wissen, wie er diese erreichen kann. Jedes Netzwerkinterface ist eindeutig über eine MAC-Adresse adressierbar – aber da MAC-Adressen über keine Struktur verfügen, müsste jeder Router einen Eintrag pro Netzwerkkarte weltweit haben, wodurch Weiterleitungstabellen unhandhabbar groß werden. Daher definiert IP eigene Adressen. Gewählt wurde ein 32 Bit großer Adressraum, um ausreichend viele Adressen für das weltweite Netz zur Verfügung zu haben. Da es somit aber immer noch 2^{32} mögliche Adressen gibt, wären die Weiterleitungstabellen immer noch zu groß, um sie verwalten zu können.

Deshalb benutzt man eine Adressstruktur, die es nicht erforderlich macht, dass jeder Router jeden Host kennt. Man unterteilt die Adresse in einen Netzteil und einen Hostteil. Somit muss ein Router nur noch wissen, wie er ein bestimmtes Netz erreichen kann, es ist also nur noch ein Eintrag pro Netz nötig. Für sein eigenes Netz, das über den Router erreicht werden kann (LAN, MAN oder WAN), muss er zusätzlich wissen, wie er die einzelnen Hosts erreichen kann.

Allerdings wird keine starre Aufteilung der IP-Adresse in Netz- und Hostteil vorgenommen. Da Router nur noch einen einzigen Eintrag für ein ganzes Netz abspeichern, können Adressen mit gleichem Netzwerk-Anteil nur an einem einzigen Standort verwendet werden. Ungenutzte Adressen mit diesem Netzwerk-Anteil können nirgendwo sonst verwendet werden. Da reale Netze sehr unterschiedliche Größe haben können, würde eine einheitliche Festlegung der Netzgröße dazu führen, dass in einigen Netzen viele Adressen brachliegen und dass andererseits sehr große Netze eventuell nicht mit genügend Adressen versorgt werden können. Daher entschloss man sich dazu, unterschiedliche Netzgrößen, d.h. unterschiedlich lange Netzwerk-Anteile zuzulassen. Entsprechend der Netzgröße wurden mehrere Adressklassen definiert. (Bitte beachten: der klassenbasierte Adressierungsansatz hat sich als nicht praktikabel erwiesen und wird nicht mehr verwendet – die Klassenterminologie wird allerdings immer noch im Sprachgebrauch benutzt.)

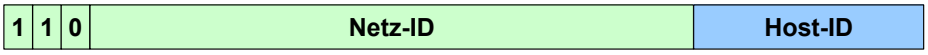
IP-Adressen / Adressklassen

- **Ursprüngliches Adressschema: Adressklassen**
  - Class A für Netze mit bis ca. 16 Mio. Hosts



  - Class B für Netze mit bis zu 65.534 Hosts

  - Class C für Netze mit bis zu 254 Host



  - Class D für Gruppenkommunikation (Multicast)

  - Class E, reserviert für zukünftige Anwendungen

Da nicht alle Netze gleich groß sind, d.h. die gleiche Anzahl Rechner umfassen, hat man zu Beginn ein Klassenschema eingeführt: die ersten Bits der Adresse definieren, welcher Klasse ein Netz angehört. Es wurde definiert, dass es drei unterschiedliche Netzgrößen geben kann (Klasse A – C):

- Klasse A: Die ersten 8 Bit der IP-Adresse identifizieren ein Netz; ein Netz dieser Klasse kann bis zu 2^{24} Hosts besitzen. Adressen dieser Klasse beginnen immer mit einer 0, daher gibt es 2^{7} solche Netze.
- Klasse B: Die ersten 16 Bit der IP-Adresse identifizieren ein Netz; ein Netz dieser Klasse kann bis zu 2^{16} Hosts besitzen. Adressen dieser Klasse beginnen immer mit 10, daher gibt es 2^{14} solche Netze.
- Klasse C: Die ersten 24 Bit der IP-Adresse identifizieren ein Netz; ein Netz dieser Klasse kann bis zu 2^{8} Hosts besitzen. Adressen dieser Klasse beginnen immer mit 110, daher gibt es 2^{21} solche Netze.

Soll ein IP-Paket weitergeleitet werden, liest ein Router zunächst nur die ersten Bits der Ziel-IP-Adresse aus und weiß daraufhin, zu welcher Klasse das Netz gehört, in dem der Zielrechner liegt und für welches Präfix der Adresse (8, 16 oder 24 Bit) er einen passenden Eintrag in seiner Weiterleitungstabelle suchen muss.

Des weiteren gibt es eine Klasse für Multicast-Adressen, d.h. Adressen, über die man Daten mit einem Paket an mehrere Empfänger zustellen kann. Und wie es bei Protokollentwicklern üblich ist, wurde ein Teil des Adressraums freigehalten (Klasse E), um eventuell später für neue Arten von Anwendungen weitere Adresstypen definieren zu können.

- Klasse D: Diese Klasse umfasst alle Adressen, die mit 1110 beginnen und ist für Multicast-Gruppen reserviert. Es sind 2^{28} Gruppen möglich.
- Klasse E: Alle Adressen, die mit 1111 beginnen; reserviert für zukünftige Verwendung.

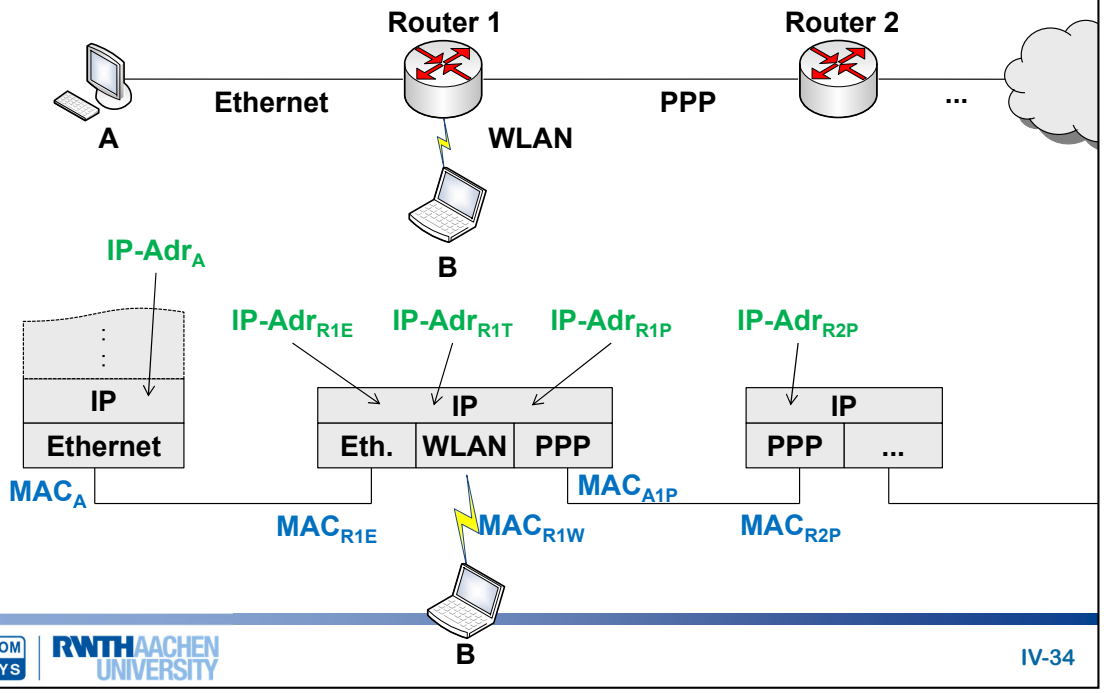
In der Praxis sind meist noch einige Adressen pro Klasse für spezielle Zwecke reserviert, wie z.B. Host-ID = 1...1 für Broadcast-Pakete (werden an alle Rechner innerhalb eines Netzes zugestellt) oder 127.0.0.1 als Loopback-Adresse, bei der keine Ausgabe der Daten aufs Netz erfolgt, sondern die Daten auf dem eigenen Rechner wieder durch den Protokollstack nach oben durchgereicht werden. Damit hat man eine Standardadresse zur Verfügung, über die eine Anwendung auf einem Rechner ohne Kenntnis der eigenen IP-Adresse über IP mit einer anderen lokalen Anwendung kommunizieren kann. Die Adresse wird daher auch als „localhost“ bezeichnet.

Außerdem gibt es einige Konventionen – so ist z.B. die höchste Adresse in einem Netz für Broadcast in diesem Netz reserviert; an diese Adresse versendete Pakete werden von jedem Rechner des lokalen Netzes entgegengenommen und verarbeitet. Router leiten Broadcast-Pakete nicht weiter, so dass ein Broadcast auf das eigene lokale Netz beschränkt ist. Die niedrigste Adresse ist für das Netz reserviert und wird üblicherweise an keinen Rechner vergeben.

Eine 4-Byte-IP-Adresse kennzeichnet einen Rechner weltweit eindeutig. Damit es keine Konflikte gibt, werden die Netz-IDs von NICs (Network Information Centers) vergeben.

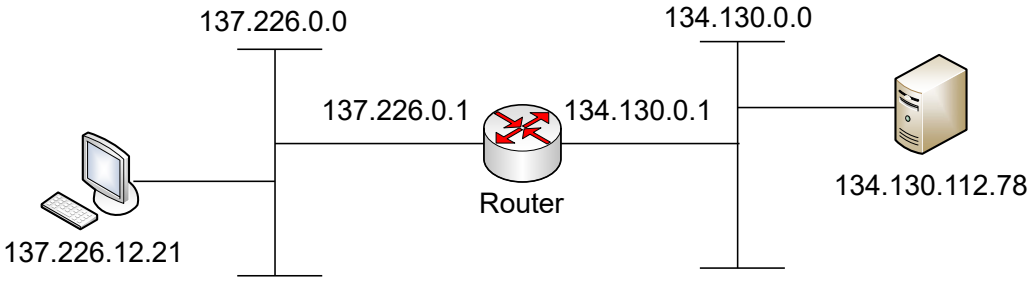
IP-Adressen

- Jedes Interface benötigt eine eigene IP-Adresse:



IP-Adressen

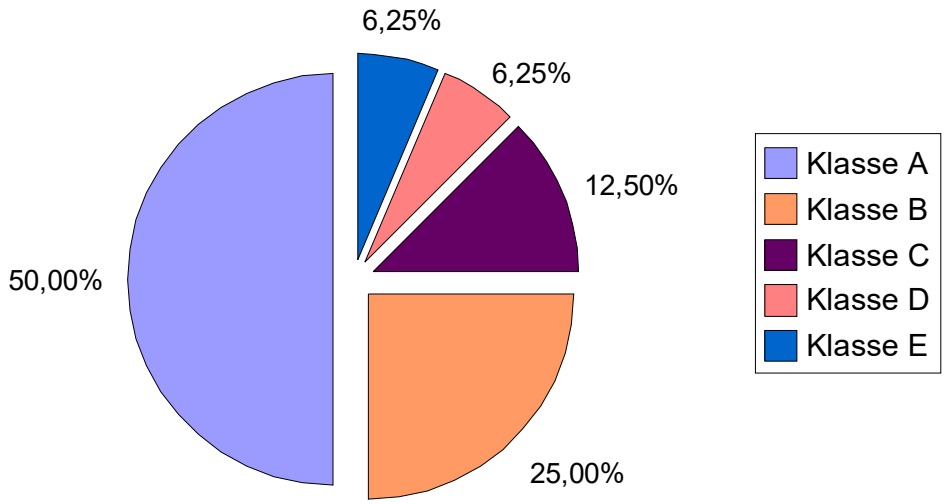
- **Einrichten eines IP-Netzes**
  - Kaufen/mieten eines Adressblocks (z.B. Klasse B: 137.226.x.x)
  - Jeder Host hat (wenigstens) eine weltweit eindeutige IP-Adresse
  - Router oder Gateways, die mehrere Netze miteinander verknüpfen, haben für jedes angeschlossene Netz eine IP-Adresse



Binärformat	10001001 11100010 00001100 00010101
Dotted Decimal Notation	137.226.12.21

Aufteilung des Adressraums:

- **Aufteilung der IP-Adressen auf die Klassen**



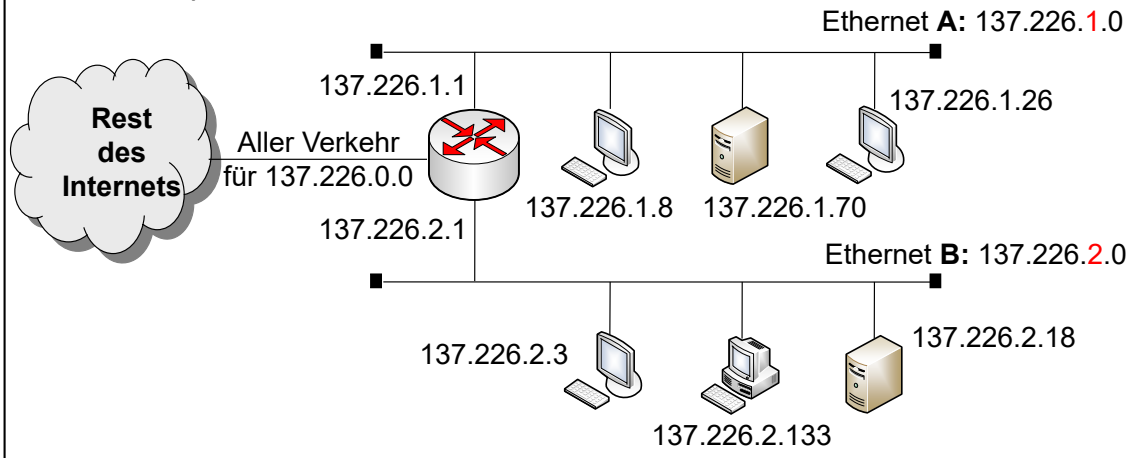
COM SYS | RWTH AACHEN UNIVERSITY IV-36

In den ersten Jahren des Internets wurden recht großzügig Klasse-A-Netze vergeben, da niemand mit einem solchen Anwachsen des Internets gerechnet hatte. Auch Klasse-B-Netze wurden bereits freizügig verteilt – was macht man beispielsweise, wenn man ein Netz mit 500 Rechnern aufbauen will? Ein Klasse-C-Netz wäre zu klein, also besorgt man sich einen Klasse-B-Adressblock. Daher war recht bald abzusehen, dass man in eine Adressknappheit hineinlief.

Als Lösung, den Adressraum effizienter aufteilen zu können, wurde daher das Konzept der *Subnetze* eingeführt.

IP-Subnetze

- **Erweiterung der Adresshierarchie: *Subnetze***
  - Zerlege ein durch ein Präfix identifiziertes Netz in kleinere Netze
  - Zerlegung identifiziert durch *Subnetz-Maske*
  - Beispiel: Subnetz-Maske 255.255.255.0



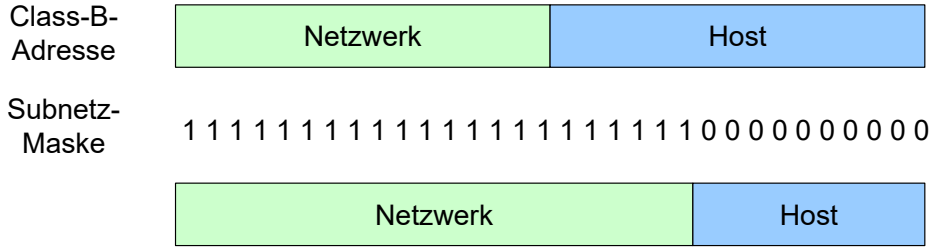
Durch eine Subnetzmaske lässt sich ein großes Netz in mehrere kleinere Netze zerlegen. Dazu wurde das Konzept der Subnetzmasken eingeführt, über die zusammengehörige Adressen innerhalb eines größeren Adressraums identifiziert werden können.

Bitte beachten: die 137.226.1.0 und 137.226.2.0 werden „Netzadressen“ genannt – auch wenn das Netz selbst gar keine Adresse hat. (Nur Hosts/Router haben Adressen.)

Das, was man als „Netzadresse“ bezeichnet, ist die Basisadresse des gesamten Netzes, die in Routingtabellen verwendet wird – siehe dazu Folie 33 – 35.

IP-Subnetze

- **„Zerlegung“ eines Netzes in Subnetze**
  - Einige Bits der Host-Adresse werden als Netzwerk-ID genutzt
  - Eine Subnetz-Maske identifiziert die Bits, die zur Identifizierung des Netzes genutzt werden
  - Router ermitteln durch Kombination einer IP-Adresse und einer Subnetz-Maske, in welches Teilnetz ein Paket geschickt werden muss



  - *Schreibweise: Netzwerk-ID/22* (= 22 Einsen in der Subnetz-Maske)

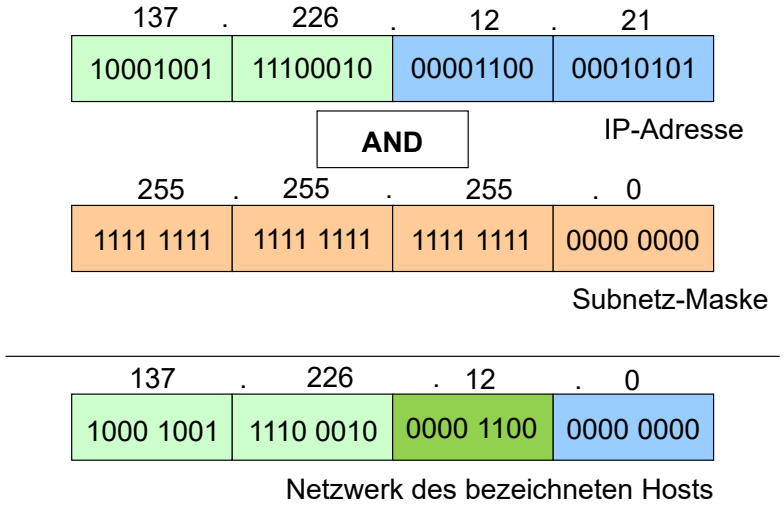
Subnetze führen eine dritte Hierarchiestufe ein. Ein außenstehender Router weiß nichts von der Einteilung in Subnetze und muss daher nicht mehr Informationen speichern. Ein netzinterner Router muss jedoch sein Netz in Subnetze unterteilen. Dabei hängt jedes Subnetz an einem physikalischen Ausgangsport des Routers.
Subnetze identifiziert der Router anhand der Subnetzmaske. Diese muss bei allen Hosts eines Netzes gleich sein. Die Subnetzmaske besteht aus einer Folge von x Einsen, die angeben, welcher Teil der IP-Adresse als Netzidentifikator verwendet wird. Dieser 1er-Folge folgen (32-x) 0en.

Beispiel:
- Klasse-B-Netz: Basisadresse 137.226.0.0
- Subnetzmaske: 255.255.240.0 (255.255.1111 0000.0000 0000)
- (137.226.aaaa bbbb.bbbb bbbb) a = Subnetz-Adresse b = Hostadresse innerhalb des Subnetzes

Damit kann man 16 ($=2^4$) Subnetze bilden, die jeweils aus 4094 ($=2^{12}-2$) Hosts bestehen können. Die niedrigste und höchste Adresse eines Subnetzes sind für Netz-ID und für Broadcast im Subnetz reserviert. Durch die Größe der Subnetzmaske kann man sich also Varianten von vielen kleinen Subnetzen bis hin zu wenigen großen Subnetzen aussuchen. Nicht alle Subnetze müssen gezwungenermaßen die gleiche Größe haben.

IP-Subnetze - Berechnung des Zielhosts

Der Eingangs-Router der RWTH, der das IP-Paket empfängt, berechnet, wo sich Host ‘137.226.12.21‘ befindet



Der Router berechnet das Subnetz ‘137.226.12.0‘ und sucht den entsprechenden Eintrag in seiner Weiterleitungstabelle

Unterschiedliche Router können auch unterschiedliches Wissen über die Zerteilung eines Adressbereichs haben. Router weltweit brauchen die interne Subnetz-Struktur der RWTH nicht zu kennen – sie benötigen nur eine Weginformation hin zu 137.226.0.0/16, welches das Netz der RWTH identifiziert. Innerhalb der RWTH ist allerdings dieser Adressbereich in Subnetze mit der Maske /24 unterteilt; daher wird der Eingangsrouter der RWTH zur Ermittlung des Zielnetzes bei Ankunft eines Pakets für z.B. den Rechner 137.226.12.21 eine Verknüpfung dieser Zieladresse mit /24 vornehmen und sucht in seiner Weiterleitungstabelle nach dem Eintrag 137.226.12.0/24. Die Verknüpfung ist eine einfache AND-Operation.

Wegewahl bei IP

- **Jedes System besitzt eine Routingtabelle**
- **Anhand der Zieladresse wird ein Eintrag bestimmt, der die Weiterleitung festlegt:**
  1. Durchsuche Host-Adressen
  2. Durchsuche Netzwerkadressen
  3. Suche nach Default-Eintrag
- **Zwei Möglichkeiten:**
  1. Rechner direkt erreichbar (direct route)
  2. Rechner indirekt erreichbar (indirect route)

Die Wegwahl, die von IP durchgeführt wird, basiert auf dem Vorhandensein einer Weiterleitungstabelle (auch: Routingtabelle) in jedem Netzknoten (Router und Hosts), der IP implementiert. Ausgehende IP-Pakete indizieren dabei anhand der Zieladresse *einen* Eintrag dieser Tabelle und können so einem Netzadapter (z.B. Ethernet oder SDH) zugeordnet werden, auf dem der gewünschte Zielrechner entweder direkt *(direct route)* oder über einen zwischengeschalteten Router *(indirect route)* erreichbar ist (siehe Beschreibung des Flags G auf der nächsten Folie). In beiden Fällen wird das IP-Paket in einen entsprechenden MAC-Rahmen verkapselt - bei der direct route wird die MAC-Adresse des gewünschten Zielrechners eingesetzt, bei der indirect route die MAC-Adresse des entsprechenden Routers - die IP-Adresse wird natürlich nicht verändert.

Um die Weiterleitungsinformationen nicht für jeden Rechner einzeln in der Tabelle eintragen zu müssen, kann als „Destination“ auch ein ganzes Subnetz eingetragen sein (erkennbar am nichtvorhandenen H-Flag, siehe nächste Folie). Jedoch werden volle Rechneradressen stets vor Subnetz-Adressen betrachtet. Danach erst kommt der Default-Eintrag *(default-Router)* zum Einsatz.

Wegwahl bei IP

Beispiel: manuell konfigurierter Rechner 137.226.12.184:

Destination	Gateway (Next Hop)	Netmask	Flags	Ref	Use	Inter-face
137.226.12.0	*	255.255.255.0	U	0	0	eth0
default	137.226.12.1	0.0.0.0	UG	0	0	eth0

Beispiel: automatisch konfigurierter Rechner 137.226.12.98:

Destination	Gateway (Next Hop)	Netmask	Flags	Ref	Use	Inter-face
137.226.12.0	*	255.255.255.0	US	0	0	eth0
default	137.226.12.1	0.0.0.0	UGS	0	0	eth0
127.0.0.0	127.0.0.1	255.0.0.0	UHS	0	0	lo0
localhost	127.0.0.1	255.255.255.255	UHS	0	0	lo0
137.226.12.98	127.0.0.1	255.255.255.255	UHS	0	0	lo0

Die Weiterleitungstabelle eines Rechners kann mithilfe des UNIX-Befehls netstat -r angezeigt werden.

Destination beinhaltet den DNS-Namen/die IP-Adresse des Zielrechners, **Gateway** die Adresse des nächsten Routers/des lokalen Netzwerkadapters auf dem Weg zum Zielrechner (abhängig von den Flags) und **Interface** den Namen des Netzadapters, über welchen der Zielrechner bzw. nächste Router erreicht werden kann. Ein „*“ bei **Gateway** soll anzeigen, dass sich der Zielrechner im gleichen Netz befindet und keine Weiterleitung über Router hinweg mehr erfolgen muss. (In diesem Fall werden die Daten direkt an den Zielrechner zugestellt, wofür die MAC-Adresse des Zielrechners benötigt wird; diese wird über ARP ermittelt, siehe später im Kapitel.)

Flags:

- **U**: Up, d.h. verfügbar.
- **G**: Gateway, d.h. der in der Spalte **Gateway** angegebene Knoten ist ein IP-Router, der das Paket weitervermitteln soll (indirect route). Bei Einträgen ohne G bezeichnet diese Spalte die Adresse des lokalen Netzadapters. (Oft herrscht das Missverständnis, dass jeder Rechner im Internet eine IP-Adresse benötigt. Tatsächlich muss sogar *jeder Netzwerkadapter* eine IP-Adresse besitzen. Router, d.h. Rechner mit mehr als einem Netzadapter, besitzen somit mehrere IP-Adressen.)
- **S**: statische Route (keine Veränderung durch Routingprotokolle)
- **H**: Host, d.h. angegebene Adresse ist Endsystemadresse (ansonsten wird ein Subnetz beschrieben)

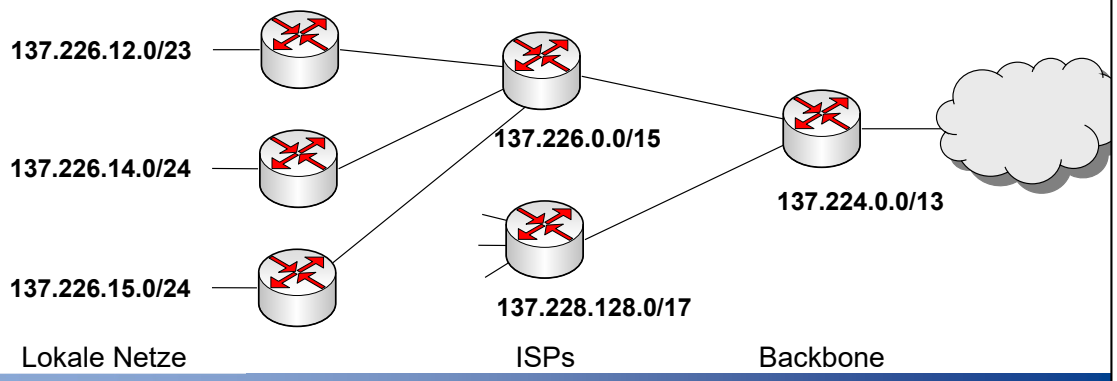
Ref gibt an, welche Routen aktuell von Kommunikationsvorgängen genutzt werden, **Use** zählt die Anzahl der gesandten Pakete auf der jeweiligen Route.

Wird kein passender Eintrag gefunden, greift auf jeden Fall der Eintrag 0.0.0.0/0 – dies ist die *Default-Route*, über die alle Pakete versendet werden, für die keine näheren Weginformationen vorliegen. Der Sinn ist, nicht für jedes Netz weltweit einen Eintrag haben zu müssen, sondern eine Vielzahl von Paketen in eine Richtung weiterleiten zu können, von der man weiß, dass dort zentrale Knotenpunkte stehen, die genauere Informationen besitzen. (Z.B. muss ein Router, der nur ein lokales Netz eines Lehrstuhls der RWTH anschließt, alles, was nicht für das eigene LAN gedacht ist, an den Gebäuderouter weiterleiten – unabhängig vom Zielnetz.) Wie die Routing-Tabellen berechnet werden, wenn Einträge sich dynamisch an die Netzsituation anpassen sollen, regeln Routing-Protokolle – die später im Kapitel behandelt werden.

„Gateway“ wird in der Routing-Terminologie auch als „Next Hop“ bezeichnet – der nächste Hop auf dem Weg zum Ziel.

Classless Inter-Domain Routing (CIDR)

- **CIDR:**
  - Schlechte Ausnutzung des Adressraums selbst mit Subnetzeinteilung
  - Ersetzen der festen Klassen durch *Netzwerk-Präfixe variabler Länge*
    - Es können auch kleine Klasse-C-Netze zu größeren Subnetzen vereint werden
    - *Route Aggregation* – Router kombinieren Einträge weitest möglich



Die Einführung von Subnetzen brachte allerdings auch einen Nachteil mit sich. Router benötigen für jedes Zielnetz einen eigenen Weiterleitungseintrag. Durch die Aufsplittung großer Netze in kleinere wuchsen Routingtabellen stark an.

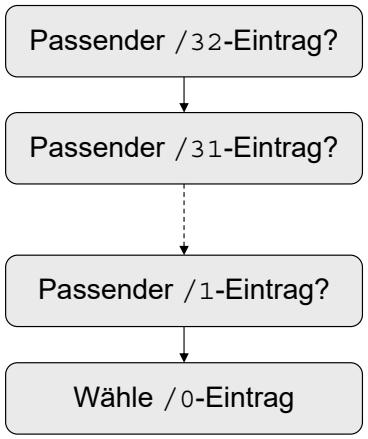
In den 90er Jahren wurde das bisherige “classful routing” zugunsten von “classless routing” aufgegeben. Dies geschah vor allem, um die Routingtabellen handhabbarer zu machen. Es wurde festgelegt, dass Subnetzmasken quasi beliebige Längen haben können und dadurch auch mehrere kleinere Netze durch einen Weiterleitungseintrag erschlagen werden können, wenn ihr Adress-Präfix gleich ist (Route Aggregation).

Einige Beispiele wären:

- Backbone-Router (z.B. an Transatlantik-Link) kann beispielsweise so konfiguriert werden, dass er nur die ersten 13 Bit der IP-Adressen betrachtet. Dadurch werden die Routing-Tabellen relativ klein gehalten und es entsteht wenig Rechenaufwand zur Bestimmung der Ausgangsleitung, über die ein Paket weitergeleitet wird.
- Router eines Providers betrachtet z.B. nur die ersten 15 Bit.
- Router in Firmennetz betrachtet z.B. die ersten 25 Bit.

Longest Prefix Match

- **Suche nach dem Routing-Eintrag mit der *größten Netzwerkpräfix-Überdeckung* der Zieladresse**
  - Ermittle den am genauesten spezifizierten Weg
    - Teilbereiche aus Adressräumen können herausgeschnitten werden



- Suche nach Übereinstimmung auf 32 Bits
  - Z.B. *Host-Route, Loopback-Adresse* `127.0.0.1`
- Suche nach Übereinstimmung auf 31 Bits
- Suche nach Übereinstimmung auf 1 Bits
- Kein Match gefunden
  - Verwende *Default-Route* `0.0.0.0/0`

Es kann durch die flexible Aufteilung des Adressraums vorkommen, dass mehrere Einträge der Routingtabelle auf die Zieladresse eines weiterzuleitenden Pakets passen – würde der Lehrstuhl für Informatik 4 beschließen, die RWTH zu verlassen und nach Köln zu ziehen aber sein Subnetz des RWTH-Netzes mitnehmen, wäre dies möglich, allerdings hätten Router außerhalb der RWTH das Problem, dass Daten an 137.226.0.0/16 nach Aachen, Daten an 137.226.12.0/23 nach Köln weitergeleitet werden müssten. Dazu wurde das Longest-Match-Verfahren eingeführt: enthält eine Routingtabelle mehrere passende Einträge, wird derjenige mit dem längsten übereinstimmenden Präfix (d.h. der längsten passenden Subnetzadresse) verwendet. Wird keiner gefunden, passt wieder auf jeden Fall 0.0.0.0/0, also die kürzest mögliche Subnetzmaske – die Default-Route.

Longest Prefix Match

- **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

- **Flexiblere Aufteilung und Anordnung von Netzbereichen**
  - z.B. Mitnahme des Adressblocks der Firma bei Umzug an einen neuen Standort
  - Aber: Vergrößerung von Routingtabellen
  - Und trotzdem: freie Adressbereiche sind ausgegangen…

NAT – Network Address Translation

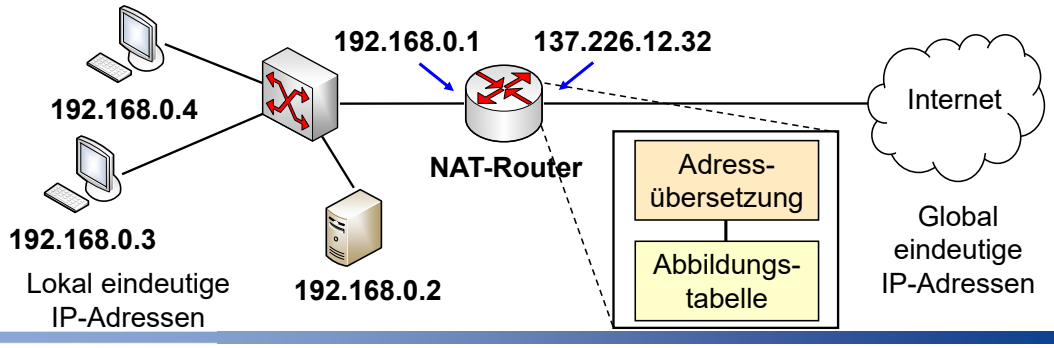
- **Notwendigkeit der Adressierung**
  - Internet: jeder Rechner benötigt eine *eindeutige IP-Adresse*, um global mit anderen Rechnern kommunizieren zu können
  - Lokales Netz: findet Kommunikation nur innerhalb des LAN statt, kann man sich einen *beliebigen Adressbereich* wählen
    - Pakete dürfen das LAN nicht verlassen!
- **Private Adressen**
  - Standardisiert: *private Adressblöcke*, die jeder intern frei nutzen kann:
    - 10.0.0.0 - 10.255.255.255
    - 172.16.0.0 - 172.31.255.255
    - 192.168.0.0 – 192.168.255.255
  - Private Adressen werden im Internet nicht geroutet

Drei Adressblöcke bei IP sind als private Blöcke freigehalten – jeder kann sich mit diesen Blöcken ein eigenes lokales Netz aufbauen, ohne dafür offiziell Adressen erwerben zu müssen. Allerdings kommt es dadurch dazu, dass mehrere unabhängige LANs die gleichen Adressen nutzen. Damit ist keine eindeutige Adressierung über das lokale Netz hinaus mehr gegeben, daher verwerfen Router Pakete mit Zieladressen aus diesen Bereichen einfach. Man kann sich also ein eigenes IP-Netz aufbauen, allerdings nur mit Rechnern aus dem eigenen Netz kommunizieren.

Diese Adressblöcke hat man sich zu Nutze gemacht, um mit der Adressknappheit bei IP umzugehen.

Network Address Translation (NAT)

- **Prinzip von NAT:**
  - Rechner im lokalen Netz verwenden private Adressen
  - Das Netz bekommt eine global eindeutige Adresse zugewiesen
  - Zugangsrouter nimmt *transparente Umsetzung* zwischen Adressen vor
    - Speicherung in Abbildungstabelle
    - Keine Änderungen an Endgeräten erforderlich

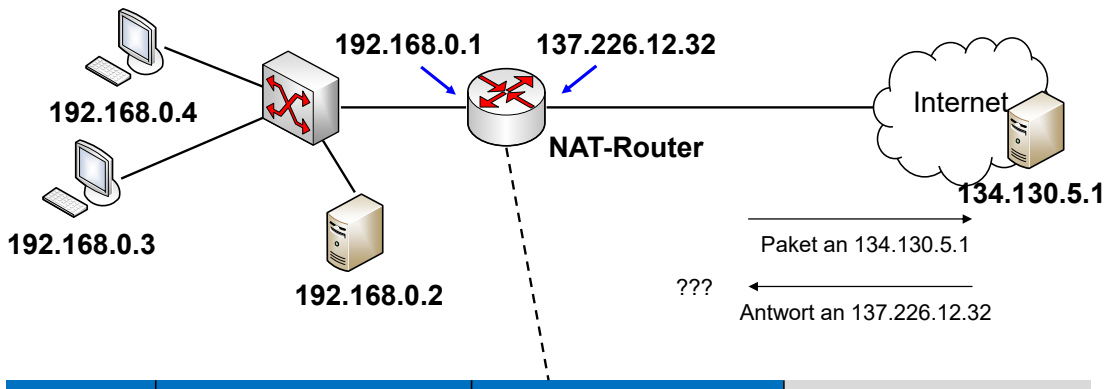


Ziel von NAT ist es, Netze mit privaten (und somit nicht routbaren) IP-Adressen an das Internet anzubinden (die Verwendung routbarer Adressen ist zwar ebenfalls möglich, es werden im Normalfall aber nur private Adressen verwendet, um Fehler durch falsch konfigurierte Zugangsrouter zu vermeiden). Hierzu nimmt der Router des lokalen Netzes eine Adressumsetzung vor, falls an der internen Netzwerkschnittstelle Pakete mit einer privaten Absenderadresse empfangen werden, die für ein externes Netz bestimmt sind. Diese Adressumsetzung kann dabei statisch (1:1) oder dynamisch (n private Adressen auf m öffentliche Adressen) erfolgen. Im Normalfall sind damit Rechner innerhalb des Netzes von außen nicht adressierbar. (Aber eine entsprechende statische Abbildung in umgekehrter Richtung kann angelegt werden, falls es notwendig ist.)

Somit ergeben sich zwei wesentliche Vorteile dieser Technologie: zum einen sind die Rechner im privaten Adressbereich geschützt, da sie von außen nicht zugreifbar sind, zum anderen wird nur eine geringe Anzahl an öffentlichen IP-Adressen benötigt.

Dabei kann ein ganzer privater Adressbereich auf nur eine einzige öffentliche IP-Adresse abgebildet werden.

NAT – Beispiel



Protokoll	lokal		global		Ziel	
	IP-Adresse	Port	IP-Adresse	Port	IP-Adresse	Port
TCP	192.168.0.4	53211	137.226.12.32	53211	134.130.5.1	80
TCP	192.168.0.3	48331	137.226.12.32	48331	134.130.5.1	80
TCP	192.168.0.4	48331	137.226.12.32	48332	141.18.62.55	25
TCP	192.168.0.2	49999	137.226.12.32	48333	123.12.123.12	12345

Problem bei NAT: was passiert, wenn mehrere Rechner des lokalen Netzes gleichzeitig nach Außen kommunizieren wollen? Alle verwenden die gleiche globale Adresse, so dass keine eindeutige Rückübersetzung möglich ist, wenn ein Antwortpaket von außen kommt. Hier könnte man natürlich einen Broadcast der Antwortpakete im lokalen Netz vornehmen (was in manchen Fällen auch gemacht wird), was aber im allgemeinen nicht effizient und meist auch nicht gewünscht ist.

Üblich ist daher, in der Abbildungstabelle noch Zusatzinformationen zu speichern, die eine eindeutige Rückübersetzung erlauben. Hierzu nimmt man im lokalen Netz eine Adresserweiterung vor, die dem eigentlichen Schichtenkonzept wiederspricht: man verwendet die *Schicht-4-Adresse* eines ausgehenden Pakets zur Identifizierung des Rechners. Dies ist der sogenannte *Port* (TCP/UDP – Kapitel 5), der die sendende *Anwendung* auf dem sendenden Rechner identifiziert. Dadurch schafft man einen neuen Adressraum von 16 Bit, anhand dessen die lokalen Rechner unterschieden werden können. Dieses Prinzip wird auch NAT-Overloading genannt.

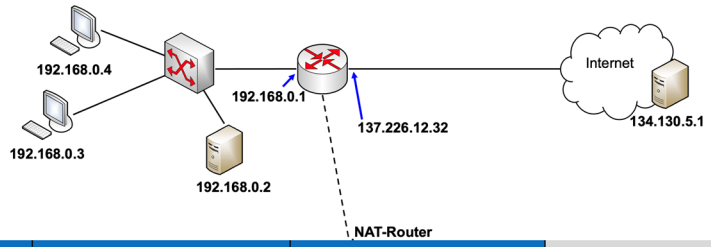
Sollte es vorkommen, dass zwei Rechner den gleichen Port nutzen, um Daten zu versenden, kann der NAT-Router nicht nur die IP-Adresse, sondern auch den Port vor der Weiterleitung des Pakets modifizieren (und muss natürlich die entsprechenden Informationen mit in seiner Abbildungstabelle speichern). Dieses Verfahen wird auch bei der Anbindung von Haushalten durch ISPs vorgenommen – man bekommt eine einzige IP-Adresse zugewiesen, der heimatliche NAT-Router schließt alle lokalen Rechner mit privaten Adressen an.

Im Beispiel auf der Folie sieht man dieses Prinzip. Ohne die Hinzunahme des Ports als Adressinformation wüsste der NAT-Router bei der Antwort von 134.130.5.1 nicht, an welchen lokalen Rechner – 192.168.0.4 und 192.168.03 – das Paket weitergeleitet werden müsste.

(Achtung: die Speicherung der Zieladressen ist nicht unbedingt notwendig; die in der Tabelle blau hinterlegten Einträge alleine sind bereits ausreichend, eine eindeutige Rückübersetzung vornehmen zu können. Ob die Zieladressen mit abgespeichert werden oder nicht, hängt von der Implementierung ab.)

An der Tabelle zeigen sich die Möglichkeiten von NAT: in Zeile 1 und 2 wird jeweils beim ausgehenden IP-Paket nur die Absender-IP-Adresse ersetzt. In Zeile 3 verwendet Rechner 192.168.0.4 den gleichen Port wie 192.168.0.3 zuvor bereits; hier wird auch die Portnummer ersetzt, um Einträge eindeutig zu halten. Eine weitere Möglichkeit zeigt Zeile 4: es gibt auch NAT-Implementierungen, die den Port immer ersetzen, egal, ob sie bereits einen identischen Eintrag haben oder nicht, und beginnend mit einer bestimmten Nummer einfach alle Ports aufsteigend verwenden.

Der Eintrag „Protokoll“ ist zwar nicht für das Mapping wesentlich, wird allerdings z.B. mit eingetragen, damit nur Pakete ins lokale Netz gelassen werden, die das korrekte Transportprotokoll verwenden (Sicherheitsaspekte) oder eine einfache Konfiguration des NAT-Routers ermöglichen (Definition von Regeln, die auf alle TCP-Verbindungen angewendet werden).



Protokoll	lokal		global		Ziel	
	IP-Adresse	Port	IP-Adresse	Port	IP-Adresse	Port
TCP	192.168.0.4	53211	137.226.12.32	53211	134.130.5.1	80
TCP	192.168.0.3	48331	137.226.12.32	48331	134.130.5.1	80
TCP	192.168.0.4	48331	137.226.12.32	48332	141.18.62.55	25
TCP	192.168.0.2	49999	137.226.12.32	48333	123.12.123.12	12345

Problem bei NAT: was passiert, wenn mehrere Rechner des lokalen Netzes gleichzeitig nach Außen kommunizieren wollen? Alle verwenden die gleiche globale Adresse, so dass keine eindeutige Rückübersetzung möglich ist, wenn ein Antwortpaket von außen kommt. Hier könnte man natürlich einen Broadcast der Antwortpakete im lokalen Netz vornehmen (was in manchen Fällen auch gemacht wird), was aber im allgemeinen nicht effizient und meist auch nicht gewünscht ist.

Üblich ist daher, in der Abbildungstabelle noch Zusatzinformationen zu speichern, die eine eindeutige Rückübersetzung erlauben. Hierzu nimmt man im lokalen Netz eine Adresserweiterung vor, die dem eigentlichen Schichtenkonzept wiederspricht: man verwendet die *Schicht-4-Adresse* eines ausgehenden Pakets zur Identifizierung des Rechners. Dies ist der sogenannte *Port* (TCP/UDP – Kapitel 5), der die sendende *Anwendung* auf dem sendenden Rechner identifiziert. Dadurch schafft man einen neuen Adressraum von 16 Bit, anhand dessen die lokalen Rechner unterschieden werden können. Dieses Prinzip wird auch NAT-Overloading genannt.

Sollte es vorkommen, dass zwei Rechner den gleichen Port nutzen, um Daten zu versenden, kann der NAT-Router nicht nur die IP-Adresse, sondern auch den Port vor der Weiterleitung des Pakets modifizieren (und muss natürlich die entsprechenden Informationen mit in seiner Abbildungstabelle speichern). Dieses Verfahen wird auch bei der Anbindung von Haushalten durch ISPs vorgenommen – man bekommt eine einzige IP-Adresse zugewiesen, der heimatliche NAT-Router schließt alle lokalen Rechner mit privaten Adressen an.

Im Beispiel auf der Folie sieht man dieses Prinzip. Ohne die Hinzunahme des Ports als Adressinformation wüsste der NAT-Router bei der Antwort von 134.130.5.1 nicht, an welchen lokalen Rechner – 192.168.0.4 und 192.168.03 – das Paket weitergeleitet werden müsste.

(Achtung: die Speicherung der Zieladressen ist nicht unbedingt notwendig; die in der Tabelle blau hinterlegten Einträge alleine sind bereits ausreichend, eine eindeutige Rückübersetzung vornehmen zu können. Ob die Zieladressen mit abgespeichert werden oder nicht, hängt von der Implementierung ab.)

An der Tabelle zeigen sich die Möglichkeiten von NAT: in Zeile 1 und 2 wird jeweils beim ausgehenden IP-Paket nur die Absender-IP-Adresse ersetzt. In Zeile 3 verwendet Rechner 192.168.0.4 den gleichen Port wie 192.168.0.3 zuvor bereits; hier wird auch die Portnummer ersetzt, um Einträge eindeutig zu halten. Eine weitere Möglichkeit zeigt Zeile 4: es gibt auch NAT-Implementierungen, die den Port immer ersetzen, egal, ob sie bereits einen identischen Eintrag haben oder nicht, und beginnend mit einer bestimmten Nummer einfach alle Ports aufsteigend verwenden.

Der Eintrag „Protokoll“ ist zwar nicht für das Mapping wesentlich, wird allerdings z.B. mit eingetragen, damit nur Pakete ins lokale Netz gelassen werden, die das korrekte Transportprotokoll verwenden (Sicherheitsaspekte) oder eine einfache Konfiguration des NAT-Routers ermöglichen (Definition von Regeln, die auf alle TCP-Verbindungen angewendet werden).

Der Aufbau eines IP-Pakets

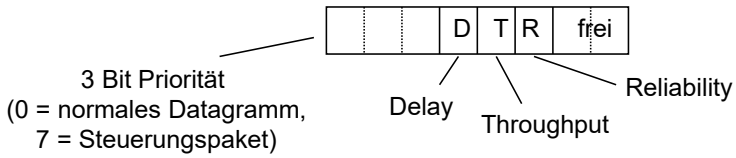
Version	IHL	Type of Service	Total Length
Identification		DF MF	Fragment-Offset
Time-to-Live		Protocol	Header Checksum
Source Address			
Destination Address			
Options (0 oder mehr Wörter)			
Data			

Der IP-Header

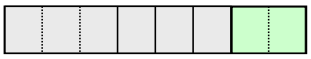
- **Version**
  - IP-Versionsnummer (hier immer 4)
  - Erlaubt den gleichzeitigen Einsatz mehrerer Versionen
    - Anhand des Werts ermittelt ein Router den Aufbau des folgenden Headers
- **IHL**
  - IP-Header-Length (in Worten à 4 Byte; zwischen 5 und 15, je nach Optionen)
- **Total Length**
  - Länge des gesamten Datagramms (in Byte, $\leq 2^{16}-1 = 65535$ Bytes)
  - In der Praxis üblicherweise maximal 1500 Byte
- **Source Address / Destination Address**
  - IP-Adressen von sendendem und empfangendem Rechner

Der IP-Header

- **Type of Service**
  - Angabe der gewünschten Eigenschaften bei der Übertragung: ist Geschwindigkeit, Durchsatz oder Zuverlässigkeit wichtig? Welche Priorität hat das Paket?



  - Nicht eingesetzt, darum heute: Neudefinition der Bedeutung zur Unterstützung von Quality of Service und Stauerkennung



Das Type-of-Service-Feld sollte dazu dienen, Pakete nach Typ des Payloads klassifizieren zu können: z.B. Echtzeitdaten, Dateiübertragung, …

Routerhersteller haben es allerdings stets ignoriert, da jede weitere Überprüfung eines Headerfeldes und entsprechende Aktionen zur unterschiedlichen Behandlung von Paketen zu einer Erhöhung der Bearbeitungszeit pro Paket führen. Stattdessen werden alle Pakete gleich (und einfach) behandelt, um die Anzahl weitergeleiteter Pakete pro Sekunde maximieren zu können.

Im Laufe der Jahre wurde die Bedeutung des Feldes daher umdefiniert. Die vorderen 6 Bits dienen der QoS-Unterstützung, behalten also im Wesentlichen ihren ursprünglichen Zweck bei, allerdings mit anderer, effizienterer Implementierung (hier nicht behandelt). Die letzten beiden Bits wurden der Stauerkennung zugeordnet, um TCP eine effizientere Arbeit zu ermöglichen (siehe Kapitel 5).

Der IP-Header

- **Time-to-Live (TTL)**
  - Lebenszeit von Datagrammen begrenzen auf maximal 255 Hops (verhindert endloses Kreisen von Paketen im Netz)
  - Der Zähler wird von jedem Router um 1 verringert
  - Bei 0 wird das Datagramm verworfen
- **Protocol**
  - Welches Transportprotokoll wird im Datenteil verwendet?
  - An welchen Transportprozess ist das Paket daher beim Empfänger weiterzugeben?
- **Header Checksum**
  - Prüfsumme über den Header
  - Muss bei jedem Hop neu berechnet werden (da sich TTL ändert)

TTL: war ursprünglich als tatsächliche Zeitangabe gedacht, wurde aber zur einfacheren Implementierung so modifiziert, dass jeder empfangende Router den Zähler um 1 verringert, unabhängig von der tatsächlich verbrachten Zeit im Router bzw. auf einem Link. Der Router, der den Wert auf 0 runterzählt, verwirft das Paket, d.h. der initiale Wert gibt an, wie viele Teilstrecken ein Paket maximal zurücklegen darf.

Header Checksum: ist keine CRC, sondern einfacher: der Header wird in 16-Bit-Blöcke zerlegt und diese (Einerkomplement) aufsummiert. Vom Ergebnis wird wieder das Einerkomplement gebildet.

Der IP-Header

- **Identification**
  - Eindeutige Kennzeichnung eines Pakets
- **DF: Don't Fragment**
  - Das Paket darf nicht fragmentiert werden
  - Zu große Pakete mit gesetztem DF-Flag werden verworfen
- **MF: More Fragments**
  - "1" - es folgen weitere Fragmente
  - "0" - letztes Fragment eines Pakets
- **Fragment Offset**
  - Folgenummern der Fragmente eines Pakets
  - Sagt aus, an welche Stelle des Datenteils eines Pakets (gerechnet in 8-Byte-Stücken) ein Fragment gehört

Anmerkung: es gibt drei Flag-Bits: MF, DF und eins, welches für zukünftige Zwecke reserviert wurde.

Fragmentierung

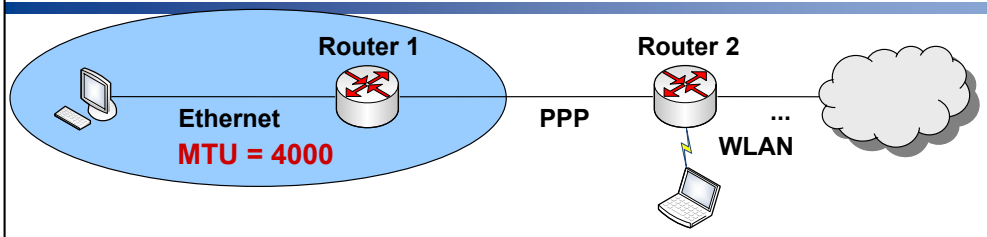
- **Verwendung von Fragmentierung**
  - In jedem Netz gibt es eine Maximallänge für Dateneinheiten
    - *MTU – Maximum Transfer Unit*
    - Z.B. Ethernet: Datenteil darf maximal 1500 Byte groß sein
  - Jeder Router prüft, ob er ein IP-Paket über die ermittelte Leitung senden kann oder ob es dort die Maximallänge überschreitet
    - Zu große IP-Pakete werden fragmentiert, damit sie weitergeleitet werden können
  - Aber: der Datenteil des Pakets formt eine logische Einheit
    - Wie kann diese wieder zusammengesetzt werden, bevor der Zielrechner die Daten verarbeitet?

Eine Aufgabe von IP ist die Fragmentierung. Typischerweise wird der Sender Pakete maximaler Größe erstellen und versenden (1500 Byte). Allerdings hat jedes Schicht-2-Protokoll eine MTU (= maximale Größe des Payloads). Ist ein IP-Paket größer als die MTU, muss IP das Paket fragmentieren (= in Teilpakete zerlegen), damit es über mehrere Pakete verteilt weitergeleitet werden kann. Bei Ethernet ist die MTU 1500 Byte – was auch der Grund dafür ist, dass in der Praxis keine größeren IP-Pakete erstellt werden.

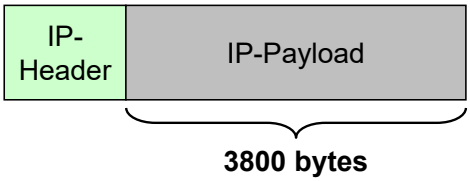
Trotzdem kann es sein, dass bereits der Sender oder ein Router auf dem Weg zum Ziel ein IP-Paket zerlegen muss, da es ansonsten nicht über das nächste Schicht-2-Protokoll weitergeleitet werden kann.

Damit der Empfänger Pakete wieder korrekt zusammensetzen kann, müssen während der Fragmentierung ein paar Kontrollinformationen hinzugefügt werden, die dem Empfänger mitteilen, an welcher Stelle des Originalpakets ein Teilpaket anzuordnen ist.

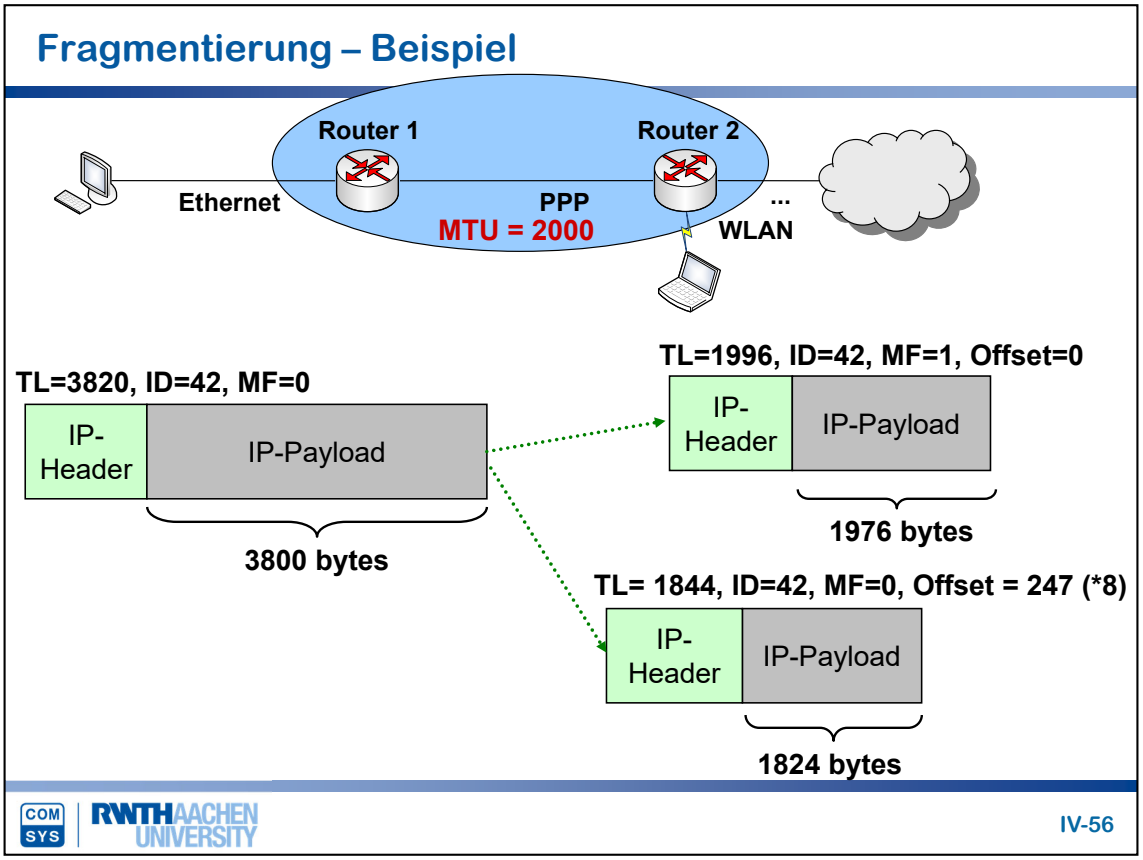
Fragmentierung – Beispiel



TL=3820, ID=42, MF=0



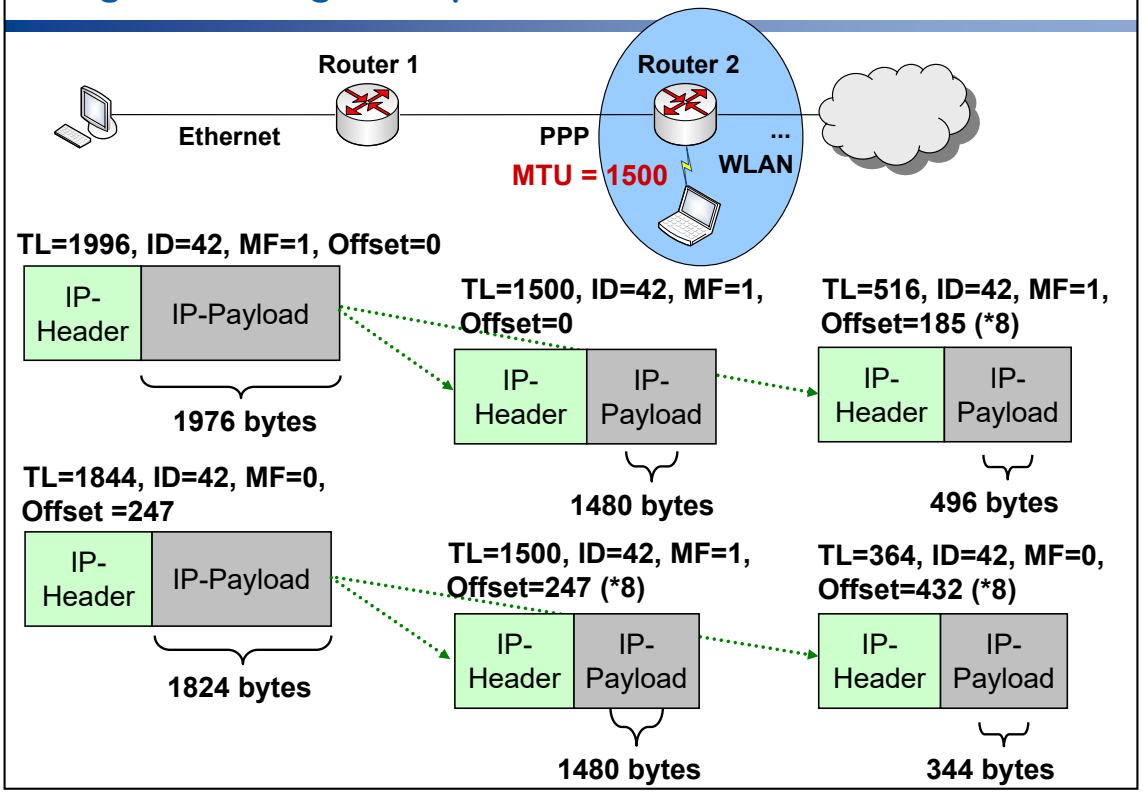
TL: Total Length
ID: Identifiction
MF: More Fragments



Die Kontrollinformationen zum Erkennen der Fragmentierung beim Empfänger sind

- ID: alle Fragmente eines Pakets tragen die ID des Ursprungspakets, damit der Empfänger identifizieren kann, welche Fragmente zusammengehören.
- MF: dieses Flag wird bei allen außer dem letzten gesetzt. Der Empfänger weiß bei Erhalt eines IP-Pakets mit gesetztem Flag, dass dies kein komplettes Paket ist, sondern er mit der Verarbeitung warten muss, bis alle Fragmente empfangen wurden. Im letzten Fragment ist das Bit auf 0 gesetzt, um anzuzeigen, dass keine weiteren Fragmente folgen.
- Offset: der Offset gibt (in Vielfachen von 8 Byte) an, an welcher Stelle im Originalpaket das Fragment anzuordnen ist. Hierdurch kann der Empfänger Fragmente in die richtige Reihenfolge sortieren und auch feststellen, ob einzelne Fragmente verlorengegangen sind.

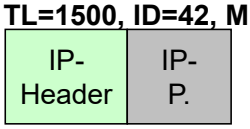
Fragmentierung – Beispiel



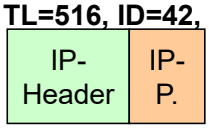
Zusammensetzung der Fragmente

- **Empfänger nutzt MF und Offset, um Fragmente zu erkennen und zusammenzusetzen**

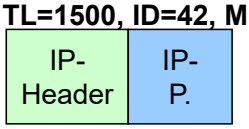
TL=1500, ID=42, MF=1, Offset=0



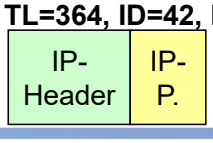
TL=516, ID=42, MF=1, Offset=185 (*8)



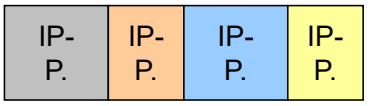
TL=1500, ID=42, MF=1, Offset=247 (*8)



TL=364, ID=42, MF=0, Offset=432 (*8)



- IP-Pakete mit MF=1 oder mit MF=0 aber Offset>0 sind Fragmente
- Sortierung der Fragmente anhand des Offsets
- Unterscheidung der Fragmente unterschiedlicher Pakete durch ID



Der IP-Header

- **Options: Spielraum für zukünftige Erweiterungen**
  - Länge: Vielfaches von 4 Byte, daher ist möglicherweise Padding notwendig
  - Fünf Optionen definiert, wenn auch nicht genutzt
    - *Security*: wie geheim sind die transportierten Informationen? (z.B. zur Umgehung bestimmter Router)
    - *Strict Source Routing*: Vollständiger Pfad vom Quell- zum Zielhost, definiert durch die IP-Adressen der zu passierenden Router
    - *Loose Source Routing*: die aufgelisteten Router müssen in angegebener Reihenfolge durchlaufen werden, zusätzliche Router sind erlaubt
    - *Record Route*: Aufzeichnung der IP-Adressen der durchlaufenen Router (maximal 9 IP-Adressen möglich!)
    - *Time Stamp*: Aufzeichnung von IP-Adressen mit Zeitstempel für jeden Router (je 32 Bit)

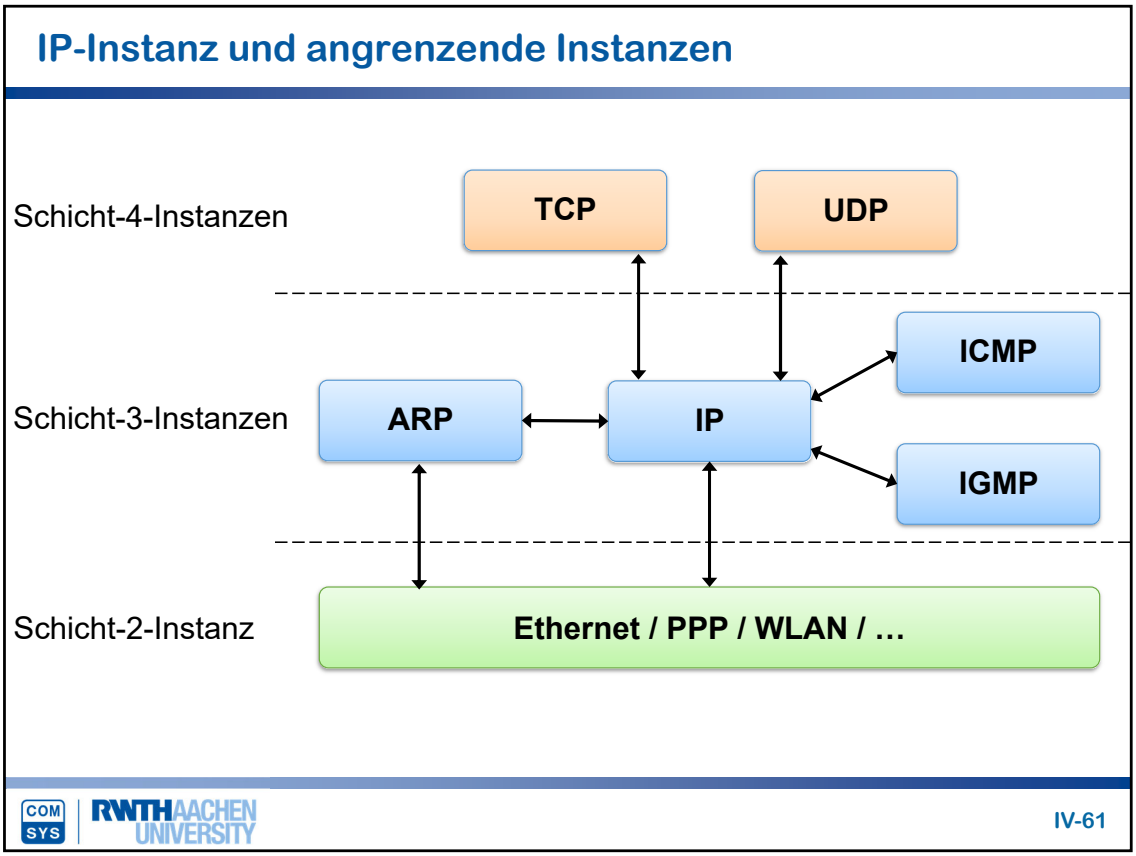
Mittels Source Routing kann der Sender einer Nachricht festlegen, welchen Weg die Nachricht durch das Netz nehmen soll, d.h. nicht die Router treffen anhand ihrer Tabellen die Entscheidungen, sondern der Sender des Pakets. Beim Strict Source Routing gibt der Sender die exakte Route vor. Alle angegebenen Hosts/Router müssen durchlaufen werden, in der angegebenenen Reihenfolge, und ohne dabei andere Hosts/Router zu durchlaufen. Beim Loose Source Routing wird zwar eine Route angegeben, und die in dieser Route aufgeführten Hosts müssen in der angegebenen Reihenfolge durchlaufen werden, dazwischen dürfen aber auch andere Hosts und/oder Router durchlaufen werden.

Source Routing wird kaum verwendet, und Pakete mit Source-Routing-Informationen werden häufig verworfen. Dies hat seinen Grund in Sicherheitsproblemen: so öffnet Source Routing z.B. die Möglichkeit, Router gezielt zu überlasten.

Zudem ist der Optionsteil auf maximal 40 Byte beschränkt – wovon ein bisschen bereits benötigt wird, um die verwendete Option anzuzeigen. Damit können maximal 9 Adressen eingetragen werden, was bei der heutigen Netzstruktur zu wenig ist. Gleiches gilt für die Record-Route-Option und erst recht für die Timestamp-Option, bei der zu jedem Router insgesamt 8 Byte an Informationen abgespeichert werden müssen.

Überprüfung des IP-Headers

- **Überprüfungen, die nach dem Empfang eines IP-Datagramms am Header durchgeführt werden:**
  - Test der IP-Versionsnummer
  - Überprüfung der korrekten Länge des Headers und des Pakets
  - Prüfsummenbildung über den IP-Header
  - Überprüfung der Paketlebenszeit
  - Überprüfung der Protokoll-ID
  - Überprüfung der Gültigkeit von Quell- und Zieladresse
    - Private Adressen sind nicht erlaubt
- **Negatives Ergebnis bei einer Überprüfung**
  - Paket verwerfen, Fehlermeldung per ICMP an Sender schicken



Wir wollen die Ausführungen zur IP-Instanz mit der Beschreibung des Zusammenspiels mit den oben, unten und insbesondere seitlich angrenzenden Protokollinstanzen abschließen.

Nachdem im Protokollstack von oben, also üblicherweise von einer TCP- oder UDP-Instanz, die zu übertragenden Daten übergeben wurden, hat die IP-Instanz die Aufgabe, die Daten anhand einer IP-Adresse weiterzuvermitteln. IP-Adressen spannen allerdings nur einen logischen Adressraum auf – physikalisch kommunizieren alle Geräte im Netz immer noch über eine Netzwerkkarte, die anhand einer MAC-Adresse (Schicht-2-Adresse) adressiert werden muss. Zur Ermittlung dieser Adresse dient ARP. Die IP-Instanz beauftragt die lokale ARP-Instanz damit, zu der Ziel-IP-Adresse die zugehörige MAC-Adresse zu ermitteln.

Ist die gesuchte MAC-Adresse bestimmt, werden die Daten der lokalen Schicht-2-Instanz übergeben (also z.B. Ethernet) und von dieser an die entfernte Schicht-2-Instanz übertragen. Dort werden dann die Daten an die überliegende IP-Instanz (und im folgenden die TCP-/UDP-Instanz) weitergegeben.

Die beiden Hilfsprotokolle ICMP und IGMP werden von der IP-Instanz für besondere Zwecke genutzt.

ICMP dient zur Übermittlung von Kontrollnachrichten auf IP-Ebene und wird üblicherweise verwendet, um bei Problemen während der Übermittlung eine entsprechende Benachrichtigung an den Absender zu übermitteln.

IGMP (Internet Group Management Protocol, hier nicht weiter behandelt) dient zur Übermittlung und Verwaltung von Informationen zu Gruppenzugehörigkeiten und ermöglicht damit Multicast.

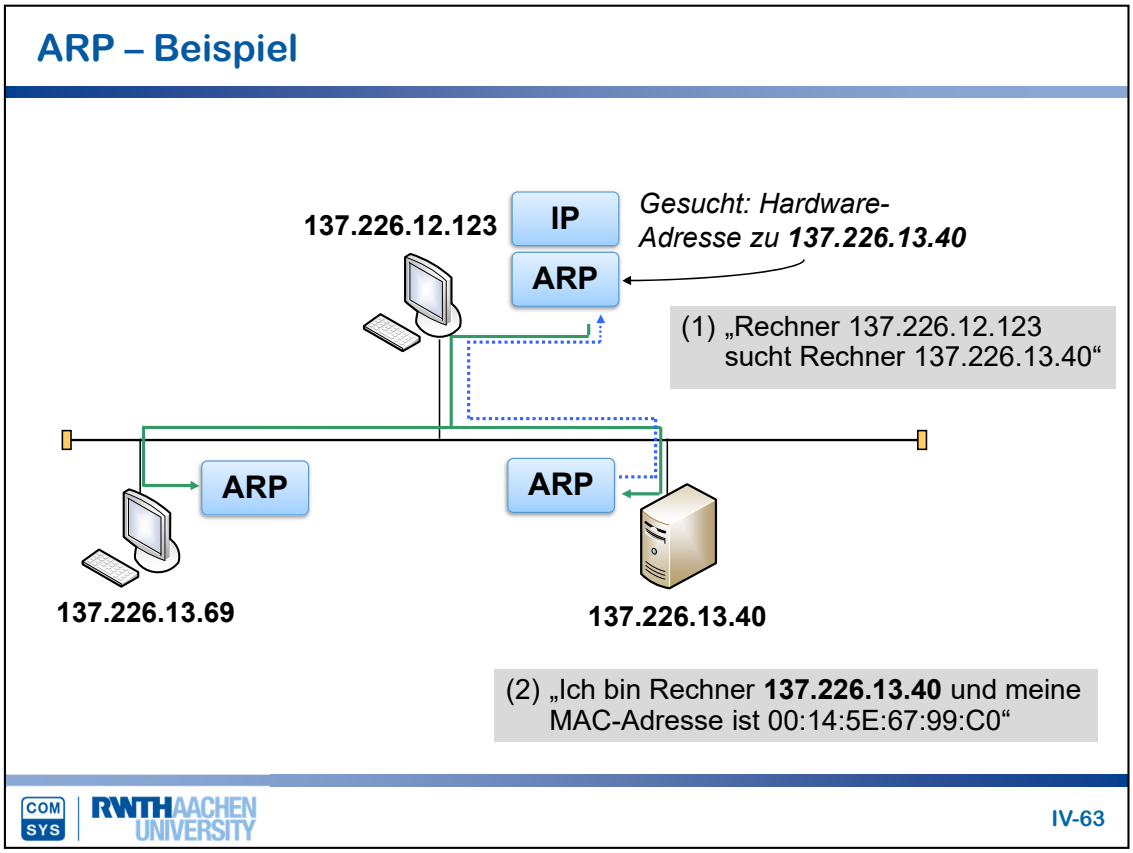
Address Resolution Protocol (ARP)

- **Aufgabe:**
  - Umsetzen IP-Adresse → Schicht-2-Adresse (MAC-Adresse)
  - Beispiel Rechner `messenger`:
    - IP-Adresse: `137.226.13.40` → Ethernet-Adresse: `00:14:5E:67:99:C0`
- **Vorgehensweise:**
  - ARP erhält eine IP-Adresse zur Adressauflösung
  - ARP sendet einen Broadcast im LAN unter Angabe der IP-Adresse
  - Alle Stationen im Netz empfangen die Anfrage, doch nur diejenige, die ihre eigene IP-Adresse erkennt, antwortet
  - Die Antwort wird bei der anfragenden Station gespeichert (ARP-Cache), um ein erneutes Anfragen zu vermeiden
  - Dieser Eintrag muss nach einem festgelegten Zeitintervall wieder gelöscht werden

ARP (Address Resolution Protocol)
Beim Versenden eines IP-Pakets gibt die IP-Instanz das Ziel in Form einer IP-Adresse an. Um jedoch das Paket zum nächsten Router oder Host zu senden, muss man die physikalische Adresse dieser Station kennen. Das Problem besteht nun darin, die Zuordnung von MAC-Adresse zu IP-Adresse eindeutig zu lösen. Dieses Problem tritt nicht bei der Weiterleitung von Paketen zwischen Routern auf, da es hier eine dedizierte Leitung gibt, über die genau ein anderes Gerät (ein Router) angeschlossen ist. Sobald allerdings über eine Leitung mehrere Geräte angeschlossen sind, d.h. in einem LAN, muss vor der Weiterleitung des Pakets die Ziel-MAC-Adresse ermittelt werden.

Dabei gibt es drei Möglichkeiten:
- Man speichert in jedem Host die Zuordnung IP→MAC in einer Datei. Dies klappt nur für kleine Netze, da der Verwaltungsaufwand zu hoch ist, jede Änderung auf jedem Rechner zu korrigieren.
- Man unterhält einen speziellen Server in seinem Netz, der die Zuordnung vornimmt. Die MAC-Adresse dieses Servers wird in allen anderen Geräten abgespeichert. Dies bringt einen Verwaltungsaufwand mit sich, und zudem liegt das Netz lahm, wenn dieser Server ausfällt.
- Man löst das Problem dezentral, indem man den gewünschten Rechner durch Broadcast ermittelt.

Die ersten beiden Verfahren werden kaum benutzt. Die dezentrale Variante hat sich durchgesetzt und wird durch ARP implementiert.



Der Ablauf des ARP-Protokolls sieht folgendermaßen aus:

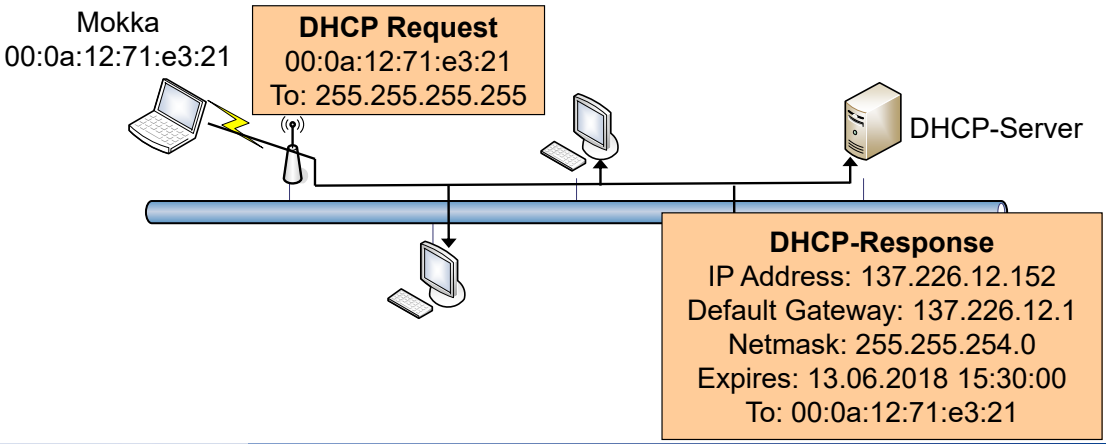
- Der Sender eines Pakets sieht in seinem ARP-Cache nach, ob er die MAC-Adresse zu der Ziel-IP-Adresse hat. Wenn ja, benutzt er diese. Die Werte im Cache werden nach einer gewissen Zeit (ohne Nachfrage) gelöscht, um zu vermeiden, dass veraltete Informationen weitergenutzt werden (falls sich z.B. durch einen Wechsel der Netzwerkkarte die MAC-Adresse eines Rechners ändert oder falls Zuordnungen durch Neukonfiguration der IP-Adressen ungültig werden).
- Wenn die MAC-Adresse nicht vorhanden ist, sendet er durch Broadcast (auf Schicht 2) einen ARP-Request, bei dem er seine eigene IP- und MAC-Adresse und die IP-Adresse des Ziel-Hosts versendet.
- Alle Rechner im LAN erhalten diese Nachricht und vergleichen die IP-Adresse mit ihrer eigenen. Zusätzlich vermerken sie die Zuordnung von IP-/MAC-Adresse des Senders in ihrem Cache; diese ist nun in allen ARP-Caches des LANs gespeichert. (In der Praxis ist es jedoch oft so, dass nur der betroffene Rechner diese Zuordnung speichert, alle anderen ignorieren die ARP-Nachricht.)
- Der Rechner, der die gesuchte IP-Adresse besitzt, sendet nun einen ARP-Reply an den Sender zurück (Unicast, da die MAC-Adresse des Senders bekannt ist), in dem seine MAC-Adresse steht. Der Sender speichert die Zuordnung in seinem Cache und sendet das eigentliche Paket ab.

Es gibt eine Sicherheitslücke bei diesem Verfahren: der ARP-Reply könnte von einem anderen Rechner kommen, der vorgibt, die gesuchte Adresse zu haben und somit alle Pakete erhält, die für den echten Rechner gedacht waren. Falls mehrere ARP-Replies kommen, verwirft der Initiator die zusätzlichen, so dass der erste Antwortende gewinnt.

DHCP: Dynamic Host Configuration Protocol

- **Einfache Konfiguration von vernetzten Rechnern**
  - Weist IP-Adresse zu und liefert Informationen zur Netzkonfiguration: DNS-Server-Adresse, Domain-Namen, Subnetz-Masken, Router etc.

  → *automatische Integration* eines Rechners in das Internet bzw. Intranet

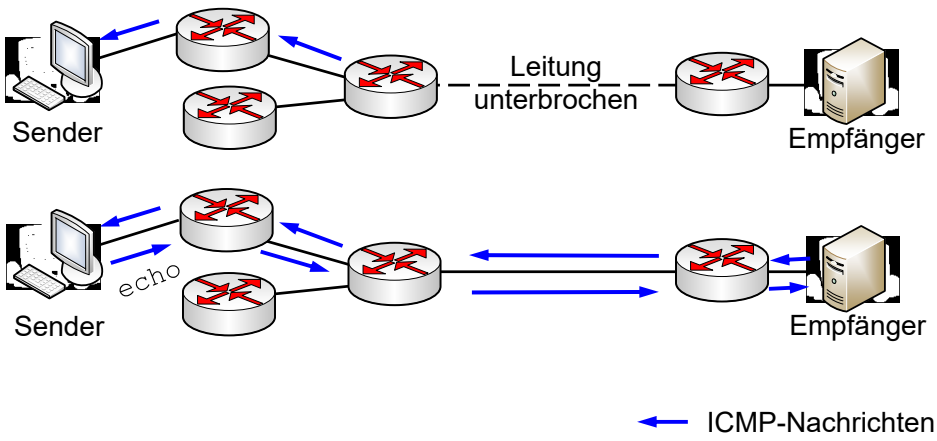


Ein Anwendungsprotokoll, das in diesem Kontext wichtig ist, ist das Dynamic Host Configuration Protocol (DHCP).

DHCP erlaubt das einfache Aufbauen eines Netzwerks mit Hosts, die eventuell nur zeitweise Teil dieses Netzes sein sollen und bei Bedarf IP-Adressen zugewiesen bekommen sollen. Dafür stellt der Netzwerkadministrator einen Bereich seiner verfügbaren Adressen einem speziellen DHCP-Server zur Verfügung. Wenn ein Client dem Netz hinzugefügt wird und eine IP-Adresse für die Kommunikation benötigt, sendet er eine DHCP-Anfrage als Broadcast auf Schicht 2. Der im Netz befindliche DHCP-Server wiest dem Client eine IP-Adresse zu und sendet ihm due gesammte Netzkonfiguration mit: die verwendete Subnetzmaske, die Adresse eines Standardgateways (Zugangsrouter des LANs, benötigt, um Daten an Hosts außerhalb des eigenen LANs verse mnden zu können), Adressen von DNS-Servern usw. Der Client verfügt nun über alle Informationen, umsowohl im Netz als auch aus dem Netz heraus zu kommunizieren.

Steuerung von IP: ICMP

- **IP: unzuverlässiger Datenaustausch**
  - Für Fehlerfälle oder Testzwecke wird das *ICMP (Internet Control Message Protocol)* verwendet



IP ist ein unzuverlässiges Protokoll. Man bekommt keine Bestätigung über die Zustellung eines Pakets oder eine Meldung über verlorengegangene Pakete. Außerdem besitzt es keinen Mechanismus, der auf Überlast reagiert.

Mit ICMP können einige dieser Mängel behoben werden. Zwar werden keine Bestätigungen für die Zustellung von Paketen versendet, aber über ICMP kann eine Meldung versendet werden, falls z.B. ein Paket in einem Router verworfen wurde oder wenn ein Router stark belastet wird. Dies dient dazu, den Absender davon abzuhalten, denselben Fehler oft zu wiederholen. ICMP-Meldungen können sowohl von Routern als auch von Endsystemen verschickt werden.

Hosts können mittels ICMP auch ohne Vorliegen einer Fehlersituation eine ICMP-Nachricht an einen anderen Host versenden, beispielsweise, um zu prüfen, ob ein Host erreichbar ist (ping).

Man kann u.A. folgende Meldungen mit ICMP versenden:

- Zieladresse nicht erreichbar (destination unreachable): Ein Paket konnte wegen eines Ausfalls (Leitung, Router, oder Endsystem) nicht zugestellt werden.
- Zeit abgelaufen (time exceeded): Ein Paket wurde wegen Ablauf des TTL-Wertes vernichtet. Hinweis auf Kreisen des Pakets.
- Falscher Parameter (parameter problem): Ein Paket wurde wegen eines unzulässigen Wertes im IP-Header verworfen.
- Reduktion der Datenrate eines Senders (source quench): Ein überlastetes Kommunikationssystem fordert den Sender auf, die Übertragungsrate zu senken.
- Umleiten (redirect): Ein Paket sollte besser an einen anderen Router gesendet werden (z.B. da keine

geeigneten Routinginformationen bekannt sind).

Die einzelnen Meldungen können in der ICMP-Nachricht noch genauer erläutert, bspw. wird angezeigt, welches Gerät ausgefallen ist (Router oder Host), zudem kann der Header des IP-Pakets, welches die Meldung ausgelöst hat, mitgesendet werden, um dem Absender eine Analyse des Fehlers zu ermöglichen.

Wichtig zu merken ist, dass die sendende IP-Instanz zwar einen Bericht zur Analyse der Fehlerursache bekommt, ihn aber nicht dazu verwendet, die Zuverlässigkeit für die Paketübertragung zu erhöhen; beispielsweise wird keine Neuübertragung verworfener Pakete vorgenommen, sondern lediglich die Fehlerursache notiert.

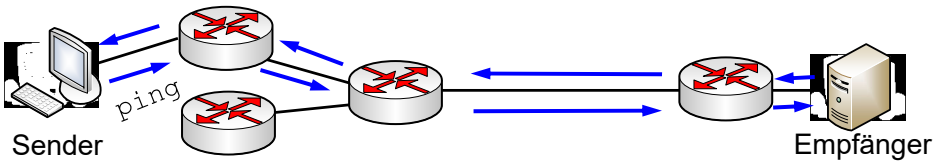
Außer den obigen Meldungen in einem Fehlerfall kann man mit ICMP auch Funktionen zur Diagnose einzelner IP-Rechner ausführen:

- Echo und Echoantwort (echo and echo reply): Der Sender sendet eine Echoanfrage an den Partner (ping) und kann dabei beliebige Daten mitsenden. Der Empfänger muss nun eine Antwort auf diese Echoanfrage zurücksenden. Dabei sendet er die Nutzdaten einfach wieder zurück. Die Nutzdaten dienen zur Analyse des Transports bei unterschiedlicher Nutzlast.
- Zeitstempel und Zeitstempelantwort (timestamp and timestamp reply): Bei dieser Anfrage sendet der Absender seine Systemzeit t_0 zum Zielhost. Dieser sendet wiederum diese Zeit, die Empfangszeit der ICMP-Anfrage t_1 und die Absendezeit der Antwort t_2 zurück. Somit kann der Initiator Rückschlüsse über die Übertragungsverzögerung auf der Verbindung und die Belastung des Zielhosts ziehen. (Achtung: da zwei beliebige Rechner nicht unbedingt synchronisierte Uhren haben, sollte das Ergebnis mit Vorsicht betrachtet werden!)

ICMP benutzt IP zum Transport seiner Meldungen, man rechnet es jedoch trotzdem zur IP-Schicht. Enthält ein IP-Paket eine ICMP-Meldung, wird das Protocol-Feld im IP-Header entsprechend gesetzt (Wert = 1). Die ICMP-Meldung (bzw. ICMP-PDU) steht im Datenteil des IP-Pakets.

ICMP: ping

- **ping: Testen der Erreichbarkeit eines Rechners und Messung der RTT zu diesem Rechner**
  - Einfache ICMP-Anfrage/-Antwort: echo
    - Ein Empfänger antwortet auf einen echo-Request so schnell wie möglich mit einem echo-Reply
  - Sender misst Zeit zwischen Aussenden der Anfrage und Erhalt der Antwort
  - Mehrmalige Wiederholung, Bildung von Mittelwerten



ping ist eine einfache Anwendung, mit der man schnell feststellen kann, ob ein bestimmter Rechner erreichbar ist. Zudem misst ping die aktuelle Round-Trip-Time zum Zielrechner.

Das Prinzip ist simpel: der anfragende Rechner sendet einen echo-Request aus und startet einen Timer. Der Empfänger eines echo-Requests beantwortet diesen schnellstmöglich mit einem echo-Response. Bei Erhalt der Antwort stoppt der anfragende Rechner seinen Timer.

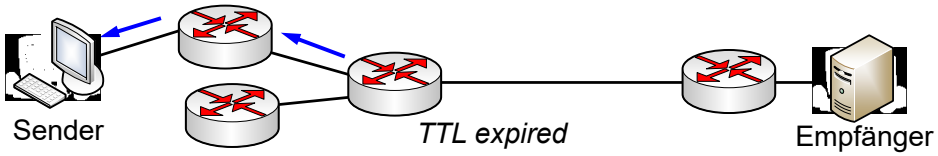
Diese Anwendung dient der Netzadministration: man kann schnell ermitteln, ob Rechner erreichbar sind und welche Verzögerung die Netze aktuell erzeugen.

Beispiel: `ping` von Berlin nach Aachen

```
$ ping -c 10 www.rwth-aachen.de
PING www.rwth-aachen.de (137.226.107.63) 56(84) bytes of data.
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=1 ttl=117 time=12.1 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=2 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=3 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=4 ttl=117 time=12.1 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=5 ttl=117 time=12.2 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=6 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=7 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=8 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=9 ttl=117 time=12.0 ms
64 bytes from www.vr.cms.rwth-aachen.de (137.226.107.63): icmp_seq=10 ttl=117 time=12.0 ms

--- www.rwth-aachen.de ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9010ms
rtt min/avg/max/mdev = 12.011/12.087/12.277/0.102 ms
```

ICMP: `traceroute`

- **`traceroute`: Ermittlung des Pfads zu einem Zielrechner**
  - Versendung eines IP-Pakets an den Empfänger mit TTL=1
    - Erster Router auf dem Weg verwirft das Paket und sendet ICMP-Nachricht `TTL expired` zurück an den Sender
  - Versendung des Pakets mit TTL=2
    - Zweiter Router auf dem Weg verwirft das Paket und sendet ICMP-Nachricht `TTL expired` zurück an den Sender
  - …



  - Sender ermittelt alle Router auf dem Weg zum Empfänger

traceroute ist eine einfache Anwendung zur Ermittlung von Pfaden im Internet. An sich sollte diese Möglichkeit über die IP Option „Record Route“ bestehen – doch diese Option ist nicht verwendbar, da das Optionsfeld nur 40 Byte umfasst und maximal neun Routeradressen aufnehmen könnte.

Statt dessen macht sich traceroute ICMP zunutze. Der Sender versendet ein IP-Paket an den Empfänger, setzt aber absichtlich die TTL auf 1. Der erste Router auf dem Weg hin zum Empfänger zählt vor der Weiterleitung des Paktes die TTL runter – und verwirft das Paket, da die TTL nun 0 ist. Der Router sendet nun mittels ICMP eine Meldung an den Sender zurück, dass das Paket verworfen wurde. Der Sender gelangt auf diesem Weg an die IP-Adresse des ersten Routers.

Nun wird das Spiel mit TTL=2 wiederholt. Der erste Router leitet das Paket nun weiter, aber der zweite zählt die TTL auf 0 runter und verwirft es. Auch der zweite Router sendet eine entsprechende ICMP-Meldung an den Sender zurück.

Der Sender inkrementiert die TTL so lange und wiederholt den Prozess, bis der Empfänger erreicht wird. Der Sender kennt nun den Pfad, auf dem das Paket hin zum Empfänger geleitet wurde.

Das ist zumindest die Theorie – schaut man genauer in traceroute hinein, stellt man fest, dass man äußerst vorsichtig beurteilen sollte, was für einen Pfad man ermittelt hat.

Traceroute: Sample Output

```
$ traceroute www.rwth-aachen.de
traceroute to www.rwth-aachen.de (137.226.107.63), 30 hops max, 60 byte packets
 1  130.149.221.190 (130.149.221.190)  0.127 ms  0.145 ms  0.172 ms
 2  ta-inet.gate.tu-berlin.de (130.149.235.193)  0.898 ms  0.984 ms  1.075 ms
 3  e-n-hft.gate.tu-berlin.de (130.149.126.57)  0.659 ms  0.791 ms  0.997 ms
 4  cr-tub1-te0-0-0-15.x-win.dfn.de (188.1.235.117)  0.929 ms  0.925 ms  0.999 ms
 5  cr-han1-hundredgige0-6-0-0-7.x-win.dfn.de (188.1.144.189)  5.641 ms  5.926 ms  5.923 ms
 6  kr-aac18-4.x-win.dfn.de (188.1.242.98)  12.222 ms  12.354 ms  12.247 ms
 7  fw-xwin-1-vl106.noc.rwth-aachen.de (134.130.3.229)  11.945 ms  11.815 ms  11.878 ms
 8  n7k-ww10-1-vl158.noc.rwth-aachen.de (134.130.3.243)  12.660 ms  12.670 ms  12.092 ms
 9  n7k-ww10-2-et1-1.noc.rwth-aachen.de (137.226.139.42)  12.913 ms n7k-sw23-2-et1-
1.noc.rwth-aachen.de (137.226.38.58)  12.604 ms n7k-ww10-2-et1-1.noc.rwth-aachen.de
(137.226.139.42)  12.860 ms
10  stone-rz-ids-pfo-1.noc.rwth-aachen.de (134.130.9.60)  12.023 ms  12.003 ms  11.963 ms
```


Im Trace sieht man in Zeile 9, dass die drei Antworten von zwei unterschiedlichen Rechnern kommen – zum Load Balancing werden hier einkommende Pakete auf mehrere Rechner verteilt (z.B. zur Prüfung, Firewalls), so dass nicht alle Pakete exakt den gleichen Weg nehmen. Genauso kann es sein, dass aufgrund es Routings schon im Kernnetz Pakete auf unterschiedlichen Wegen weitergeleitet werden, so dass man einen Pfad ermittelt, den es gar nicht gibt.

Auch sind Router nicht verpflichtet, ICMP-Nachrichten zu verarbeiten und zu verschicken. Da die Bearbeitung/Erstellung von ICMP-Nachrichten Rechenzeit braucht, die Router sinnvoller einsetzen könnten, kann es sein, dass sie bei Verwerfen eines Pakets keine ICMP-Nachricht zurücksenden. In dem Fall würde man keine Informationen über den entsprechenden Hop bekommen.

Das „neue“ IP – IPv6

IPv4 (September 1981) → IPv6 (Dezember 1995)

- **Warum ein neues Protokoll, wenn IPv4 gut funktioniert?**
  - IP-Adressen gehen aus (NAT als Hilfslösung)
  - Aufwand für Konfiguration von Rechnern (DHCP als Hilfslösung)
  - Keine Unterstützung verschiedener Dienste (DiffServ als Hilfslösung)
  - Keine Sicherheitsmechanismen wie Verschlüsselung und Authentifizierung (IPSec als Hilfslösung)
  - Keine Unterstützung von Mobilität aufgrund ortsgebundener Adressen (Mobile IP als Hilfslösung)
  - Relativ aufwändige Paketverarbeitung in Routern (Zeitaufwand)
  - Großer Umfang der Routingtabellen (Verwaltungsaufwand)
  - Kein vernünftiger Optionsmechanismus für Weiterentwicklungen

COM SYS | RWTH AACHEN UNIVERSITY IV-70

IPv4 hat sich als ein gut funktionierendes Protokoll über Jahre bewährt. Trotzdem hat sich die IETF bereits vor Jahrzehnten dazu entschlossen, IP zu überholen und eine neue Version zu etablieren. Folgende Gründe waren die entscheidenden für das Entwickeln eines neuen Protokolls:

- Anwachsen des Internets: Der große Erfolg des Internets führte zu einer stark wachsenden Benutzerzahl und vor allem auch zu einem starken Anstieg der Netzbelastung. Die steigenden Benutzerzahlen führen dazu, dass der IP-Adressraum ausgeschöpft ist und die Einrichtung neuer Netze immer aufwändiger wird (NAT), und dass als Folge der nachträglichen Einführung von Subnetzen die Routingtabellen in Kernroutern sehr umfangreich sind, was den Weiterleitungsprozess von Paketen verlangsamt.
- Geänderte Anforderungen an die Datenübertragung: erfordern Vertraulichkeit und Authentifizierung, Mobilitätsunterstützung, usw. IP bietet solche Funktionalitäten nicht, daher wurden im Laufe der Zeit viele Erweiterungen entwickelt, die IPv4 um fehlende Funktionalitäten ergänzen sollen. Aufgrund des beschränkten Optionsmechanismus bietet IPv4 nicht viel Spielraum, solche Erweiterungen effizient zu integrieren.
- Hohe Datenraten: einige Protokollfunktionalitäten (bspw. Fragmentieren in einem Router) reduzieren die Effizienz der Weiterleitung von Paketen. Eine effizientere Paketverarbeitung in Routern ist unabdingbar, um die Kapazität der Netze zu erhöhen.

Dennoch verlief die Einführung von IPv6 nur schleppend. Grund: Zusätze zum alten IP (IPSec, Mobile IP, NAT, …) erfüllen bereits die mit IPv6 gesetzten Erwartungen, so dass lange Zeit eigentlich kein direkter Handlungsbedarf bestand. Erst langsam verbreitet sich auch IPv6.

IPv6 - Eigenschaften

- **Änderungen bei IPv6**
  - *Adressgröße 128 Bit* (8 Gruppen zu je 4 Hexadezimal-Zahlen)
    - Reduktion des Umfangs von Routing-Tabellen durch mehr Struktur in den Adressen
    - Multi-Homing: transparente Verwendung mehrerer Adressen gleichzeitig
    - *Autokonfiguration von Adressen* durch einen Rechner selbst (aber auch noch durch DHCP)
  - *Vereinfachung des Protokolls (und in Folge des Headers)* zur beschleunigten Verarbeitung von IPv6-Paketen im Router
    - *Verbesserter Optionsmechanismus*, feste Headerlänge
    - Keine Checksum
    - *Keine Fragmentierung* im Router mehr, dadurch Entlastung der Router
  - *Markieren von Paketen* für speziellen Verkehr
  - *Zusatzdienste als Optionen*: z.B. Authentifizierung und Vertraulichkeit

IPv6 – Adressen

- **Beispiele für IPv6-Adressen**
  - 8 Hexadezimalwerte
    - Beispiel: `2a00:8a60:1014:0000:0002:0000:0345:5f23`
  - Führende Nullen können weggelassen werden:
    - `2a00:8a60:1014:0:2:0:345:5f23`
  - An einer Stelle können Nuller-Blöcke weggelassen werden
    - `2a00:8a60:1014::2:0:345:5f23`
  - Integration des IPv4-Adresraums als `::ffff:ip4-address`
    - `::ffff:137.226.12.21` oder `::ffff:89e2:c15`
  - Hierarchische Struktur wie in IPv4, mit Präfixschreibweise
    - `2a00:8a60:1014::/64`

IPv6 – Adressen

- **IPv6-Adressen können sein…**
  - *Unicast-Adressen* einzelner Hosts
  - *Multicast-Adressen*: Adressieren alle Mitglieder einer Gruppe von Hosts
  - *Anycast-Adressen*: Adressieren ein Mitglied einer Gruppe von Hosts
  - Typ der Adresse wird durch das Präfix bestimmt:

Reserviert:	00000000 = 0000::/8 (= ::/8)	
Unspezifizierte Adresse	00…00 (128 Bit)	= ::/128
Loopback-Adresse:	00…01 (128 Bit)	= ::1/128
IPv4-Adresse:	0….01…1	= ::ffff:0:0/96
Multicast-Adresse:	11111111	= ff00::/8
Broadcast	ff01::1, ff02::1	
Link-local Unicast-Adresse	1111111010	= fe80::/10
Unique local Unicast-Adresse	1111110	= fc00::/7
Global Unicast-Adresse	alles andere	

Wie auch bei IPv4 gibt es Unicast-Adressen, die einzelne Hosts eindeutig identifizieren. Broadcast-Adressen zur Versendung von Daten an alle Hosts in einem Netz gibt es der Terminologie nach nicht mehr – bei IPv6 wird dies auch als Multicasting bezeichnet, die Broadcast-Adresse eines Netzes ist nun eine spezielle Multicast-Adresse.

Anycast-Adressen sind ein neues Konzept, welches es bei IPv4 nicht gab: wie eine Multicast-Adresse adressiert man über eine Anycast-Adresse eine Gruppe von Hosts. Während allerdings Daten bei Multicasting an alle Gruppenmitglieder zugestellt werden, werden sie beim Anycasting nur an genau ein Gruppenmitglied versendet. Dies soll es ermöglichen, einen Anwendungsdienst (z.B. Suchmaschine, Streaming-Server) zu replizieren und mehrmals anzubieten (Beispiel Mirror-Server). Ruft ein Client den Anwendungsdienst auf, wird er nach einem bestimmten Prinzip (welches im Netz implementiert sein muss, z.B. als Teil des Routing-Algorithmus) mit einem Server verbunden. Dies wäre z.B. bei Video-on-Demand sinnvoll, um jeden Client mit dem geographisch nächsten Host zu verbinden, auf dem der Anwendungsdienst läuft (Reduktion der Latenz). Für Anycasting wird kein eigener Adressbereich reserviert, man verwendet Unicast-Adressen. Dadurch bleibt für einen Dienstnutzer verborgen, ob hinter einer IP-Adresse ein Empfänger oder eine Empfängergruppe steckt.

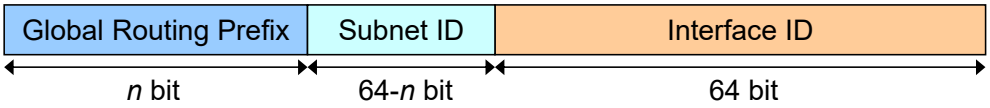
Dieses Prinzip findet auch bereits in IPv4-Netzen Einsatz, meist aber auf höheren Schichten implementiert.

Eine spezielle Adresse ist die Loopback-Adresse. Während in IPv4 ein ganzes Klasse-A-Netz dafür reserviert wurde, ohne Wissen der eigenen IP-Adresse Anwendungen auf dem gleichen Host ansprechen zu können, ist dies bei IPv6 nur noch eine einzige Adresse. (Da die Praxis gezeigt hat, dass auch in IPv4 nur eine einzige Adresse verwendet wurde.)

Um während einer Übergangszeit IPv4 und IPv6 parallel betreiben zu können, wurde auch vorgesehen, ein Subnetz in IPv6 vorzuhalten, welches den gesamten IPv4-Adressbereich integriert, so dass auch nur-IPv4-fähige Hosts von einem IPv6-fähigen Host aus angesprochen werden können.

Adressbeispiel

- **Global Unicast: dreigeteilte Hierarchie**



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

  → Interface ID: `00:1D:72:FF:FF:8E:74:6C` (64 bit)

  (bzw. `001D:72FF:FF8E:746C` zur Darstellung in der IP-Adresse)
- COMSYS-Webserver: `2a00:8a60:1014::142`

Die Entwicklung der Global-Unicast-Adressen wurde direkt mit mehr Hierarchiestufen vorgenommen als bei IPv4. Jedes Netz bekommt nach wie vor eine Netz-ID sowie eine Host-ID (Interface-ID). Beide Teile sind fix 64 Bit lang.

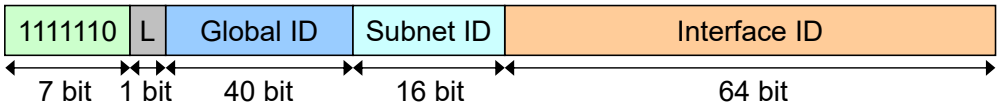
Die Netz-ID ist weiter strukturiert, um den Umfang von Routingtabellen zu reduzieren. Die ersten *n* (bis zu 32) Bit identifizieren eine Organisation (z.B. die RWTH Aachen) und werden von regionalen Vergabestellen (Regional Internet Registry) koordiniert vergeben. Geht die Vergabestelle geschickt vor, lassen sich hinterher eventuell sogar weniger als *n* Bit nutzen, um geographische Bereiche zu identifizieren.

Die Subnet-ID ist nur innerhalb der Organisation von Interesse und dient dazu, den zugewiesenen Adressraum auf mehrere unabhängige Einheiten zu verteilen (z.B. einzelne Lehrstühle an der RWTH). Die Aufteilung ist aber ganz dem Netzbereiber überlassen.

Die Interface-ID identifiziert eindeutig einen Rechner im Netz. Adressen können auch bei IPv6 manuell konfiguriert oder über DHCP bezogen werden, es gibt aber durch die sogenannte “Autokonfiguration”. um eine einfachere Zuweisung von Adressen zu ermöglichen: ein Host generiert z.B. basierend auf seiner MAC-Adresse einen 64-Bit-Identifier. Auf der Folie dargestellt ist ein sehr einfaches Beispiel. In der Praxis sollten kryptographische Funktionen zur Generierung verwendet werden, damit niemand aus der IP-Adresse auf die physikalische Adresse eines Hosts zurückschließen kann.

Adressbeispiel

- **Link-local Unicast:**
  - Jeder Host hat zumindest eine Link-local-Adresse
  - Schema: `fe80::/64 + interface ID`
  - Z.B. für Austausch von Kontrollnachrichten mit dem lokalen Router
- **Unique local Unicast**



  - Kommunikation zwischen Subnetzen
    - Global eindeutig, aber typischerweise nicht geroutet
    - L = 1: lokal zugewiesene, zufällige Global ID /
    - L = 0: global zugewiesene, eindeutige Global ID

Link-local-Unicast-Adressen sind nur innerhalb eines Netzes gültig. Sie werden IPv6-intern verwendet, z.B. zur Autokonfiguration.

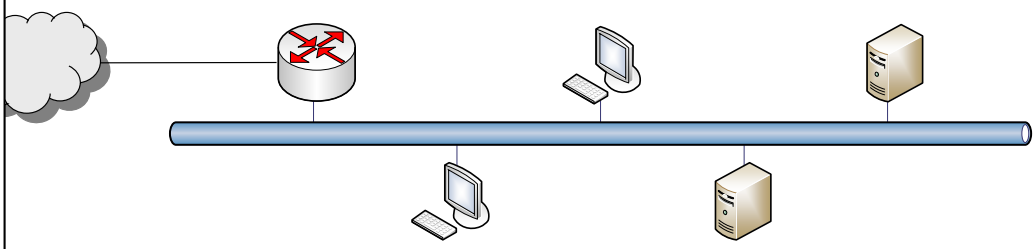
Als Ersatz privater Adressen gibt es zudem noch die Unique-local-Unicast-Adressen. Hiermit kann man IP-basierte Netze erstellen, die keine Verbindung zum Intenet benötigen. In diesen Adressen kann man durch die Subnet-ID weitere Struktur schaffen, also auch mehrere Subnetze schaffen, die miteinander verbunden sind und private Adressen verwenden. Im Internet sollen diese Adressen alleridngs nicht geroutet werden.

Die vorgeschaltete Global ID soll erreichen, dass auch all diese Netze unterschiedliche Adressen verwenden. Momentan ist aber immer L=1 gesetzt.

Es ist vorgesehen, dass ein Host gleichzeitig mindestens eine Link-local- und eine Global-Unicast-Adresse besitzt. Zudem kann ein Host noch auf mehrere Multicast-Adressen hören.

Autokonfiguration

- **Ablauf der Autokonfiguration**
  - *1. Generiere Link-Local-Adresse*



MAC-Adresse `00:1D:72:8E:74:6C`
→ Link-Local-Adresse `fe80::001D:72FF:FF8E:746C`

In IPv6 gibt es drei Möglichkeiten, einem Host eine IP-Adresse zuzuweisen: manuelle Konfiguration, Einsatz eines DHCPv6-Servers zur Adressvergabe, Autokonfiguration.

Die Autokonfiguration eines Hosts umfasst drei Schritte

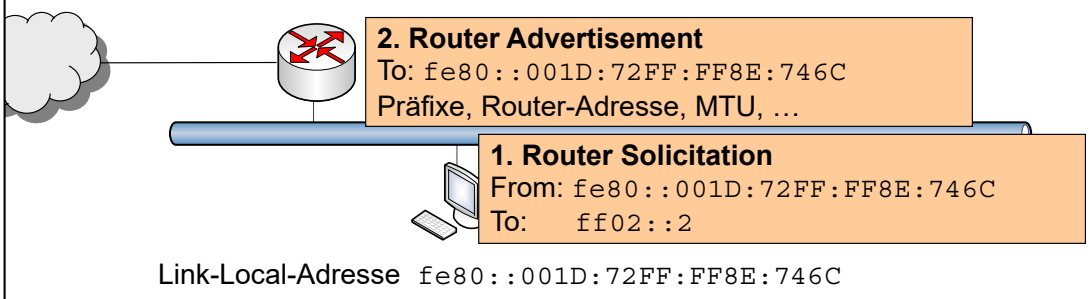
1. Zuweisung einer eigenen Link-local-Unicast-Adresse durch Berechnung der Interface-ID z.B. aus der eigenen MAC-Adresse
2. Ermittlung der Netzkonfiguration: Routeradressen, Netzwerkpräfixe, …
3. Duplicate Address Detection zur Sicherstellung, dass die gewählte Adresse im Netz noch nicht verwendet wird (MAC-Adressen müssen nicht notwendigerweise eindeutig sein)

Autokonfiguration

- **Ablauf der Autokonfiguration**

2. Router Discovery

- 1. „Router Solicitation“-Nachricht per Multicast an `ff02::2` versenden
- 2. Router antworten mit „Router Advertisement“
  - Lokale Netzwerkpräfixe, Router-Adressen, MTU



Schritte 2 und 3 werden mittels ICMPv6 ausgeführt; der speziell für diese Zwecke ausgelegte Teil von ICMPv6 wird als Neighbor Discovery Protocol bezeichnet und dient zum Auffinden anderer Rechner im eigenen Netz – sowohl andere Hosts als auch Router. Die wesentlichen Nachrichtentypen sind Router Solicitation, Router Advertisements, Neighbor Solicitation und Neighbor Advertisements.

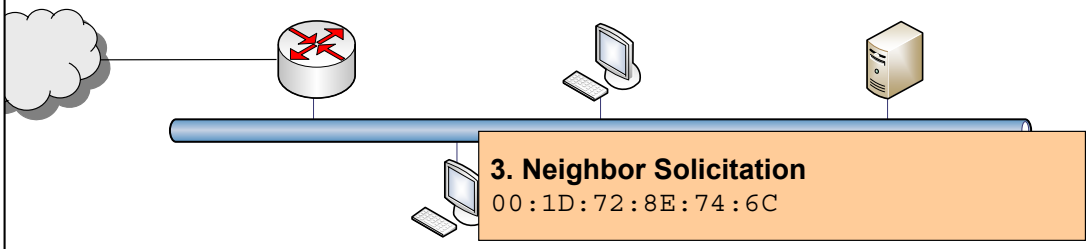
Router Discovery ist notwendig, da eine Link-local-Unicast-Adresse keine Kommunikation mit Hosts außerhalb des eigenen Netzes erlaubt. Hierzu benötigt man das eigene Netzpräfix sowie Routeradressen (die man bei IPv4 zusammen mit der IP-Adresse über DHCP bezieht). Dazu wird eine Nachricht vom Typ „Router Solicitation“ an die Muticast-Adresse gesendet, auf der alle Router im Netz lauschen. Der Router wird eine Antwort schicken, in der er alle konfigurierten Netzwerkpräfixe, seine eigene Adresse, die lokale MTU und eine Gültigkeitsdauer der Angaben zurücksendet. Der Host kann sich nun eine Global-Unicast-Adresse generieren und weltweit kommunizieren.

Autokonfiguration

- **Ablauf der Autokonfiguration**

3. Duplicate Address Detection

- „Neighbor Solicitation“ zur Abfrage der MAC-Adresse von Nachbarn
- Sende eigene MAC-Adresse als Payload mit
  - Hosts mit der MAC-Adresse antworten mit „Neighbor Advertisement“
  - Keine Antwort = eindeutige Adresse gefunden



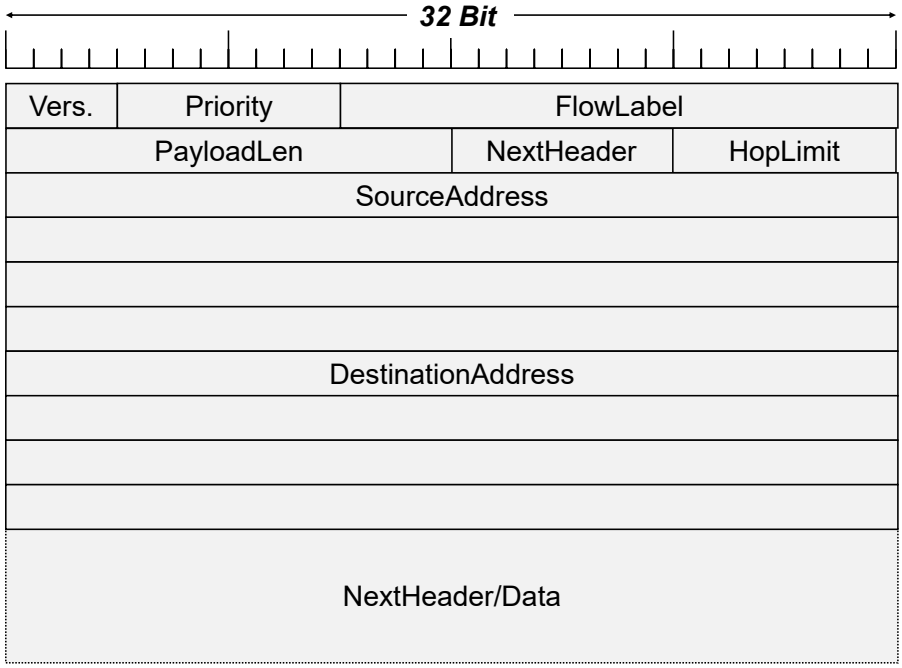
Link-Local-Adresse `fe80::001D:72FF:FF8E:746C`

Dabei bleibt allerdings ein Problem: es ist nicht garantiert, dass die selbst generierte Interface-ID nicht bereits in Benutzung ist. Daher ist in einem letzten Schritt eine Prüfung notwendig, ob man eine bereits in Verwendung befindliche Adresse gewählt hat. Die Duplicate Address Detection ist sehr simpel: führe einen Multicast an alle bereits konfigurierten Hosts im eigenen Netz durch. Dabei wird die eigene MAC-Adresse mitgesendet. Hosts, die die gleiche MAC-Adresse verwenden, werden dazu aufgefordert, zu antworten. Kommt keine Antwort, kann man davon ausgehen, dass die autogenerierte Interface-ID eindeutig sein wird. Der zur Neighbor-Discovery eingesetzte Nachrichtentyp „Neighbor Solicitation“ ist aber auch für andere Zwecke einsetzbar; er ist generell dazu gedacht, die MAC-Adresse der Nachbarn abzufragen, um ein Bild des eigenen Netzes zu bekommen. Dadurch wird auch ARP ersetzt.

Path MTU Discovery

- **Vermeidung von Fragmentierung**
  - *Erkunde minimale MTU* auf einem Pfad (PMTU)
    - Sende Pakete orientiert an lokaler MTU
    - Erster Router, der geringere MTU auf nächstem Hop hat, verwirft das Paket und sendet ICMP-Meldung „Packet too big“ zurück
    - Sender passt PMTU an und cached Wert für spätere Verwendung
  - Zweck: passe Größe von Paketen an minimale MTU an, um Fragmentierung zu vermeiden
  - Kleinste unterstützte MTU: 1280 Byte
    - Keine weitere Path MTU Discovery falls Wert geringer ist
    - IPv6 geht immer von mindestens 1280 Byte MTU aus

IPv6 Haupt-Header



- **Version:** IP-Versionsnummer, Wert: 6
- **Priority / Traffic Class:** ursprünglich 4 Bit für Priorität (zur Definition unterschiedlicher Klassen von Übertragungsarten, die im Router unterschiedlich behandelt werden): z.B.: 1 – News, 4 – FTP, 6 – Telnet, 8 bis 15 – Echtzeitverkehr.
  Mittlerweile auf 8 Bit erweitert und als „Traffic Class“ bezeichnet.
- **FlowLabel:** alle Pakete eines Datenstroms können das gleiche Label bekommen und von Routern anhand des Labels weitergeleitet werden (differenzierte Behandlung von Datenströmen)
- **PayloadLen:** Paketlänge nach dem 40-Byte-Header
- **NextHeader:** 8-Bit-Selektor. Gibt den Typ des folgenden Erweiterungs-Headers an (oder das verwendete Transportprotokoll, falls keine Erweiterungsheader folgen)
- **HopLimit:** Wird bei jedem Knoten um 1 dekrementiert. Bei Null wird das Paket verworfen
- **SourceAddress:** Die IPv6-Adresse des Senders des Pakets
- **DestinationAddress:** Die IPv6-Adresse des Empfängers (nicht unbedingt das endgültige Ziel, wenn es einen Routing-Erweiterungsheader gibt, z.B. zur Unterstützung von Mobilität).
- **NextHeader/Data**: Optionsmechanismus – hänge Optionen in Form beliebig vieler weiterer Header zwischen den IPv6-Header und den eigentlichen Payload (der auch mit einem Header, meist eines Transportschichtprotokolls beginnen wird).

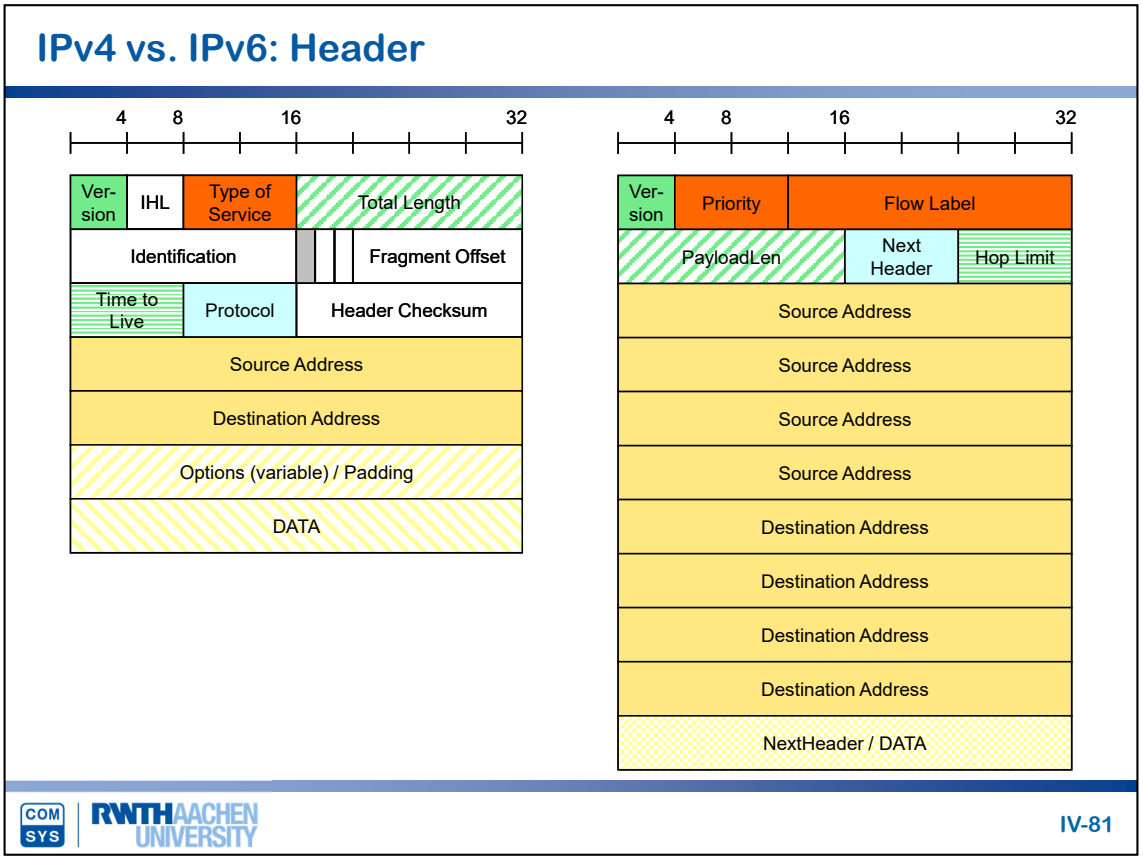
Erweiterungs-Header

- **6 Klassen von Erweiterungs-Headern (Optionen):**
  - Hop-by-Hop (Informationen für Teilstrecken)
    - Alle Router müssen dieses Feld prüfen
    - Z.B. Unterstützung von Jumbogrammen, d.h. Paketen mit Überlänge (Hierbei wird eine Längenangabe eingetragen)
  - Zieloptionen (Zusatzinformationen für das Ziel)
  - Routing (Definition einer vollen oder teilweise festgelegten Route)
  - Fragmentierung (Verwaltung von Fragmenten)
    - Verwendung, wenn Sender Fragmentierung vornehmen muss
  - Authentifikation (des Senders)
  - Verschlüsselung (Informationen zur Verschlüsselung der Daten)
  - Zieloptionen (Zusatzinformationen für das Ziel)

Die Auslagerung von Optionen in Erweiterungsheader ermöglicht es, beliebig viele Optionen einfach zu einem IP-Paket hinzuzufügen. Generell lasen sich alle Optionen in eine von sechs Klassen einordnen.

- Hop-by-Hop: jeder Router, der das Paket weiterleitet, muss die Inhalte dieser Erweiterungsheader prüfen. Eine Möglichkeit zum Einsatz wäre die effizientere Kommunikation zwischen Rechnern in kontrollierbarem Umfeld – so könnte man in einem Rechencluster zur Verringerung des Overheads durch Header sogenannte Jumbogramme versenden. Durch solch einen Erweiterungsheader werden alle Router angewiesen, die Längenangabe im Header zu ignorieren und stattdessen den folgenden Wert zu verwenden
- Fragmentierung: Auslagerung der Fragmentierungsinformationen in einen Erweiterungsheader, da Fragmentierung bei modernen Netzen als relativ selten notwendig eingeschätzt wurde. Durch Path MTU Discovery kann der Sender schon eine passende Fragmentierung vornehmen.
- Authentifizierung und Verschlüsselung dienen dazu, Informationen zur Authentizität des Absenders und zur Verschlüsselung des Paketinhaltes zu übermitteln. Beide Erweiterungsheader wurden als IPSec auch als Aufsätze zu IPv4 implementiert und werden in Kapitel 6 behandelt.
- Routing-Header können z.B. ein Source-Routing ermöglichen. Zusammen mit den Zieloptionen realisiert Mobile IPv6 eine Mobilitätsunterstützung, bei der man seine IP-Adresse behalten kann und unter dieser Adresse auch erreichbar ist, wenn man das Netz gewechselt hat (siehe Vorlesung “Mobile Internet Technology”)

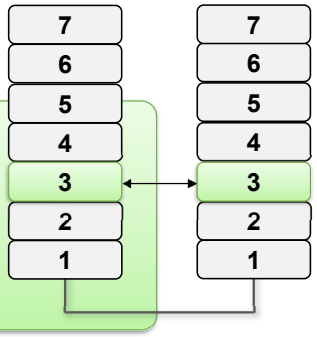
Die Optionen für das Ziel sind zwei Mal aufgelistet, da die Reihenfolge auf der Folie auch diejenige ist, in der die Erweiterungsheader aneinandergehängt werden sollten; je nach Zielinformation bieten sich zwei Stellen an, an denen sie eingebettet werden.



Der IPv6-Header ist zwar länger, doch dies liegt nur an den längeren Adressen. Ansonsten ist er „aufgeräumt worden“, was die Paketverarbeitung in den Routern beschleunigt.

Kapitel 4: Vermittlungsschicht

- **Vermittlungsverfahren**
  - Leitungsvermittlung, Speicher-/Paketvermittlung
  - Beispiel ATM
- **Die Vermittlungsschicht im Internet – IP**
  - IPv4: Adressen, Subnetze, CIDR, NAT
  - IPv4-Header
  - Hilfsprotokolle: ARP, DHCP, ICMP
  - IPv4 vs. IPv6
- **Wegewahlverfahren (Routing)**
  - Hierarchien, einfache Verfahren
  - Distance Vector
  - Link State



Routing

- **Paketweiterleitung**
  - Verbindungslos: unabhängige Wegewahlentscheidung für jedes Paket (z.B. IP)
  - Verbindungsorientiert: Wegewahlentscheidung nur bei Verbindungs-herstellung (z.B. ATM, Telefonnetz)
    - Alle innerhalb einer Verbindung ausgetauschten Pakete folgen dem bei Verbindungsherstellung festgelegten Weg
- **Routing**
  - Ermittlung von Wegen für das Forwarding
  - Beteiligte Komponenten beim Routing
    - *Routing-Protokoll*: Protokoll zum Austausch von Routing-Informationen
    - *Routing-Algorithmen*: Ermittlung von günstigsten Wegen im Netz, Erzeugung von Einträgen für Routing-Tabelle
    - *Routing-Tabellen*: Halten der Wegdaten

Die Paketweiterleitung selbst wird als “Forwarding” bezeichnet. “Routing” ist der Prozess der Wegeermittlung, so dass es Regeln für die Paketweiterleitung gibt.

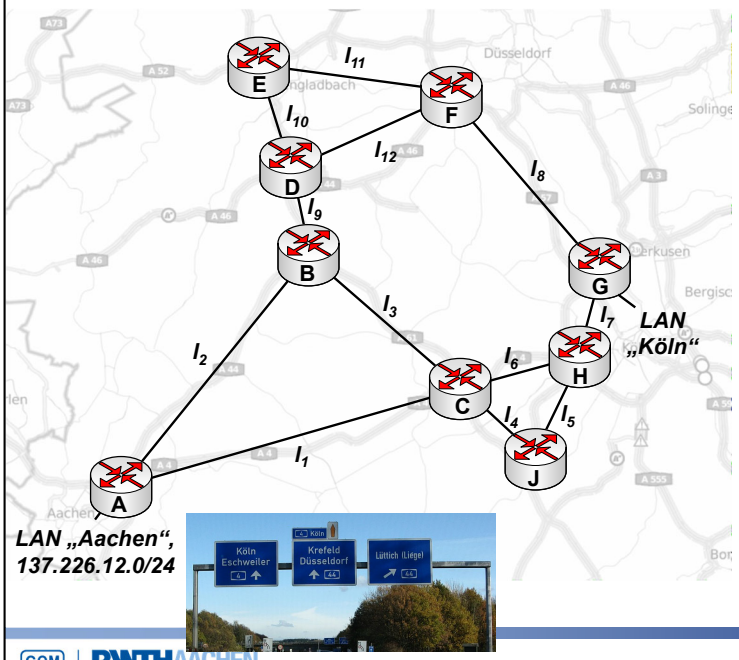
Umgangssprachlich wird auch schon mal der Weiterleitungsprozess als Routing bezeichnet, aber konzeptionell sollte man zwischen den Begriffen Routing und Weiterleitung sauber trennen.

Um Daten weiterleiten zu können, muss ein Rechner oder ein Router über eine Routing-Tabelle verfügen, die die notwendigen Weiterleitungsinformationen enthält. Diese kann statisch konfiguriert werden, was z.B. auf den Endrechnern und auf vielen Routern, die LANs anschließen, passiert. Endrechner sind üblicherweise nur durch einen einzigen Router mit dem Rest des Internets gekoppelt, daher gibt es nur einen einzigen Weg, den Daten an entfernte Rechner nehmen können. Dieser ändert sich nie, so dass die entsprechenden Informationen statisch eingepflegt werden können. Gleiches gilt für viele Router, die lokale Netze anschließen – auch hier gibt es oft nur zwei Anschlüsse: den ins lokalen Netz und einen hin zu einem Router auf höherer Ebene.

Sobald man den lokalen Bereich allerdings verlässt, existieren meist mehrere mögliche Pfade hin zum Ziel – hier benötigt man nun Routing-Protokolle, über die sich Router über den aktuellen Netzzustand austauschen können, um auf Basis der so ermittelten Informationen durch einen Routing-Algorithmus kürzeste Wege durchs Netz zu berechnen und daraus Einträge für die Routing-Tabelle zu berechnen. Durch regelmäßigen Informationsaustausch mit anderen Routern kann so auch dynamisch auf Änderungen im Netz reagiert werden.

Routing-Protokolle und -Algorithmen werden im Folgenden zusammen auch als Routing-Verfahren bezeichnet.

Prinzip einer Routing-Tabelle



Routing-Tabelle für C:

nach	Next Hop	Link
A	A	l_1
B	B	l_3
D	B	l_3
E	B	l_3
F	B	l_3
G	H	l_6
H	H	l_6
J	J	l_4

„nach A“ in tatsächlicher Routing-Tabelle z.B. „137.226.12.0/24“

COMSYS | RWTH AACHEN UNIVERSITY

IV-84

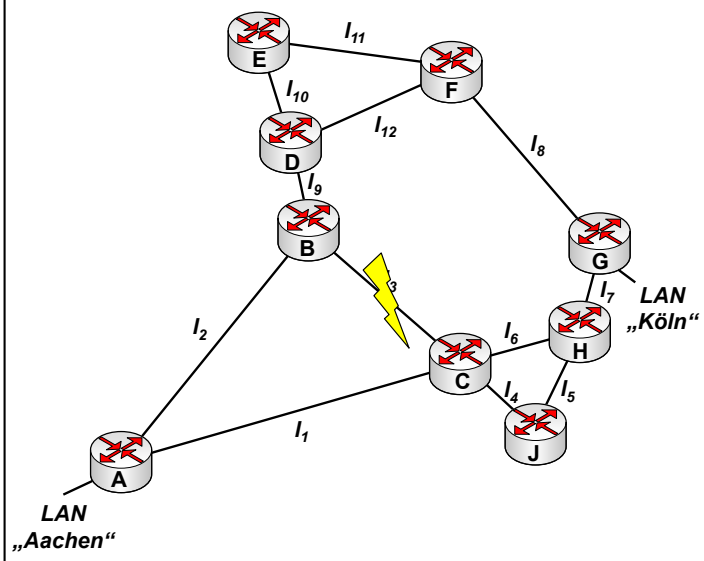
Eine Routing-Tabelle haben wir bereits oben gesehen – sie enthält eine Menge von Weiterleitungsregeln der Form (X,Y): „Pakete an einen Rechner in Zielnetz X müssen an Router Y weitergeleitet werden“. Ist der Zielrechner im gleichen Netz wie der sendende Knoten, hat man einen Eintrag der Form (Z,*): „Der Sender sitzt selbst in Zielnetz Z. Mittels z.B. ARP muss die MAC-Adresse bestimmt werden, damit die Daten über Schicht 2 zugestellt werden können.“

Im lokalen Bereich kann man diese Regeln manuell konfigurieren. In einem einfachen lokalen Netz braucht man nur zwei Regeln – eine der Form (Z,*), damit alle Rechner im lokalen Bereich zugestellt werden können und eine der Form (X, Z_Router), so dass alle Pakete für Rechner außerhalb des eigenen Netzes an den lokalen Router weitergeleitet werden.

Ein Kernrouter in einem Backbone benötigt eine umfangreichere Tabelle – er muss im Extremfall einen Eintrag für jedes Subnetz im Internet besitzen. Auch wenn viele Einträge mittels CIDR zusammengefasst werden können, hat man immer noch sehr umfangreiche Routingtabellen.

Als Analogie kann man ein Autobahnkreuz als Router betrachten und die Beschilderung als Routing-Tabelle, die den Paketen (Autos) den Weg weist.

Dynamische Anpassung der Tabelle: Ausfall eines Links



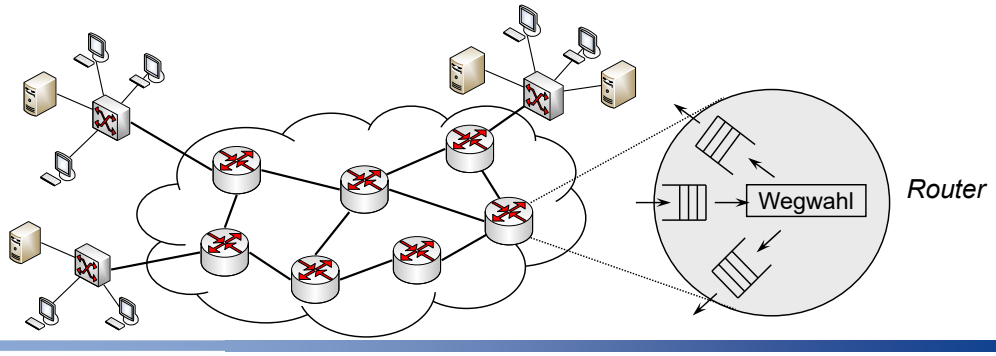
Routing-Tabelle für C:

nach	Next Hop	Link
A	A	l_1
B	**A**	$\mathbf{l_1}$
D	**A**	$\mathbf{l_1}$
E	**A**	$\mathbf{l_1}$
F	**H**	$\mathbf{l_6}$
G	H	l_6
H	H	l_6
J	J	l_4

In lokalen Netzen sind Routen normalerweise statisch konfiguriert, da es kaum alternative Pfade gibt. In Backbones hingegen müssen Einträge in Routing-Tabellen dynamisch und automatisiert konfigurierbar sein, um schnell auf Änderungen im Netzzustand reagieren zu können. Dazu werden Routing-Verfahren benötigt.

Routing: Kosten

- ***Kosten* beim Routing:**
  - Wegwahlentscheidung für ein eingehendes Paket, orientiert an:
    - Niedriger mittlerer Paketverzögerung
    - Hohem Netzdurchsatz
    - …
  - Generell: ermittle Weg mit geringsten "Kosten"

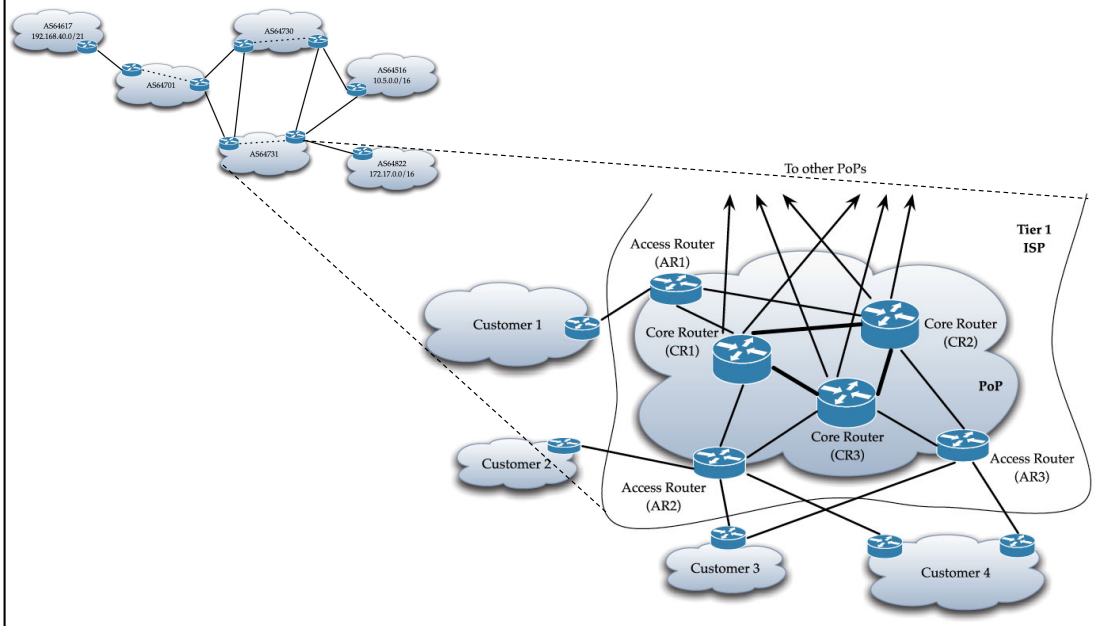


https://www.autoextrem.de/artikel-ratgeber/2375-stau-vorschau-2017-jahr-autobahn-voll.html

Generell wird beim Erstellen der Einträge der Routing-Tabelle eine Kostenminimierung angestrebt: gibt es mehrere Wege hin zum Ziel, wird der günstigste genommen.

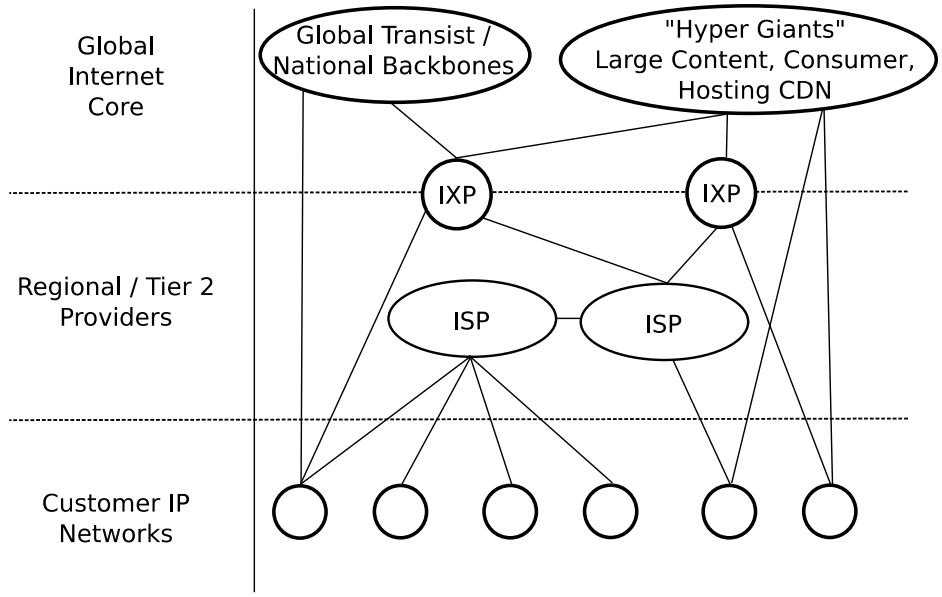
Was genau “Kosten” sind, ist nicht festgelegt. Es können tatsächliche monetäre Kosten für die Verwendung einer Leitung sein, aber auch die Anzahl der zu passierenden Router bis zum Ziel (Verarbeitungskosten), die erwartete Übertragungsverzögerung über einen Weg, der erzielbare Durchsatz, die Fehlerrate, die Anzahl der aktuell auf Übertragung wartenden Pakete … - hier bleibt Freiraum, eine passende Metrik für die Kosten zu wählen.

Autonome Systeme (Außen- und Innensicht)



Aufbau des Internet

- *Hierarchische Struktur*



COM SYS | RWTH AACHEN UNIVERSITY

IV-88

Das Internet hat eine hierarchische Struktur mit traditionell drei Ebenenen: Tier-1 bis Tier-3.

Tier-1 sind nationale Backbones (z.B. deutschlandweiter Telekom-Backbone), globale Transit-Netze (die die Backbones verbinden) und sogenannte “Hyper Giants”, z.B. die Netze großer Content Distribution Networks (CDNs). Diese können sowohl untereinander verbunden sein, als auch mit Tier-2-Netzen. Die Verbindungsstellen werden Internet Exchange Point (IXP) genannt.

Tier-2 sind Netze regionaler Anbieter. (Internet Service Provider, ISP)

Ter-3 sind Netze lokaler Anbieter, z.B. NetAachen, Campus, Firma.

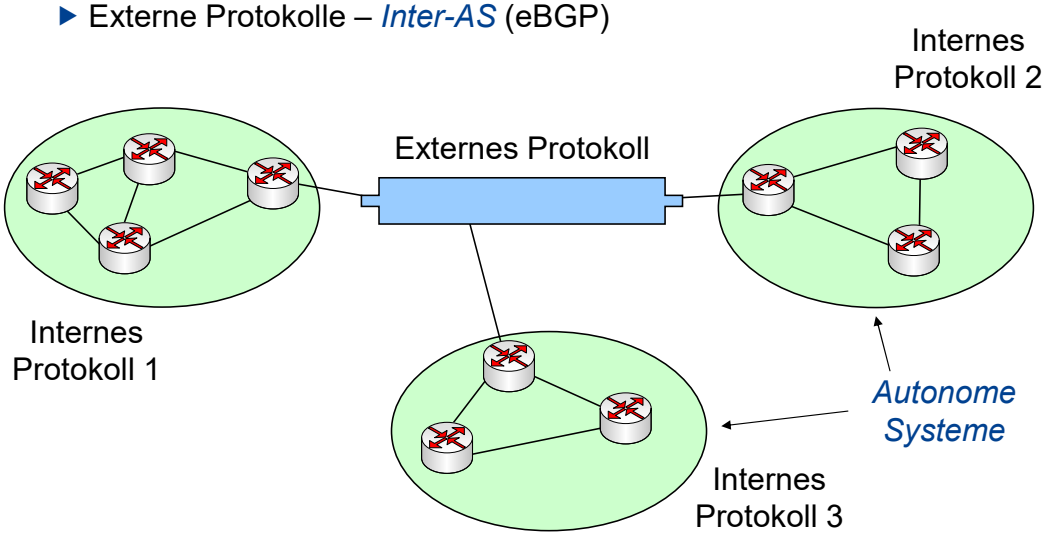
Während lange Zeit eine klare Trennung der Ebenen und festgelegte Verbindungsstrukturen bestanden, hat in den letzten (ca. 10) Jahren ein Wandel stattgefunden. Die klare Hierarchie wurde etwas aufgeweicht, um den neuen Strukturen des Internets gerecht werden zu können.

Das Internet besteht somit aus unterschiedlichen Bereichen, die unter lokaler Kontrolle einer Organisation stehen. Um zu vermeiden, dass alle Router in all diesen Bereichen das gleiche Routing-Protokoll verwenden und mit allen

anderen Routern weltweit Informationen austauschen müssen (Skalierbarkeit!), wurde das Prinzip der *autonomen Systeme* eingeführt – ein AS ist ein administrativer Bereich des Internets, der unter der Kontrolle einer Organisation steht und innerhalb dessen alle Router das gleiche Routing-Protokoll (und den gleichen Routing-Algorithmus) verwenden. Diese Aufteilung reduziert auch den Kommunikationsaufwand der Router untereinander: Router innerhalb eines AS müssen nur mit anderen Routern des gleichen AS Informationen austauschen; nur die Router, die das eigene AS mit anderen ASs verbinden, müssen auch mit diesen Informationen austauschen.

Routing im Internet

- **Zwei Klassen von Routing-Protokollen**
  - Interne Protokolle – *Intra-AS* (OSPF, RIP, iBGP)
  - Externe Protokolle – *Inter-AS* (eBGP)



Man unterscheidet generell zwischen internen Protokollen (die innerhalb autonomer Systeme eingesetzt werden) und externen Protokollen (die autonome Systeme koppeln).

An Intra-AS- und Inter-AS-Protokolle und -Algorithmen werden unterschiedliche Anforderungen gestellt. Innerhalb eines AS können die genauen Anforderungen dabei noch von der Größe und dem Aufbau des AS abhängen. Daher gibt es eine Vielzahl unterschiedlicher interner Protokolle, während es nur ein externes Protokoll gibt (da alle ASs auf einheitliche Art gekoppelt werden müssen).

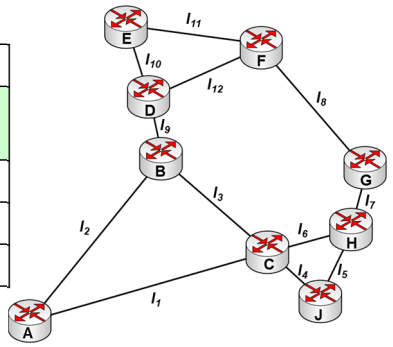
Bei der Entwicklung eines Routing-Verfahrens (sowohl intern als auch extern) müssen unterschiedliche Aspekte berücksichtigt werden:

- Dynamik: soll sich die Wegwahl adaptive an den Netzzustand anpassen? Wenn ja, wie schnell kann/muss dies geschehen?
- Zentralisation: wird die Wegwahl in einem dynamischen Autonomen System zentral getroffen oder erfolgt die Ermittlung günstiger Wege verteilt durch die Router selbst?
- Interaktion: wird die Wegwahl in einem dynamischen, dezentral organisierten Autonomen System isoliert durch jeden Router getroffen oder interagieren die Router zur Ermittlung von Wegen?

Statisches Routing (Static Routing, Directory Routing)

- **Nicht adaptiv, einfach, viel benutzt**
  - Routing-Tabelle enthält eine Zeile für jeden Zielknoten
  - Jede Zeile: *n* Einträge (beste, zweitbeste, etc. Übertragungsleitung zum Ziel) zusammen mit einer relativen Gewichtung
  - Vor Weiterleitung: ziehe Zufallszahl und wähle entsprechenden Eintrag
- **Tabelle in Knoten C:**

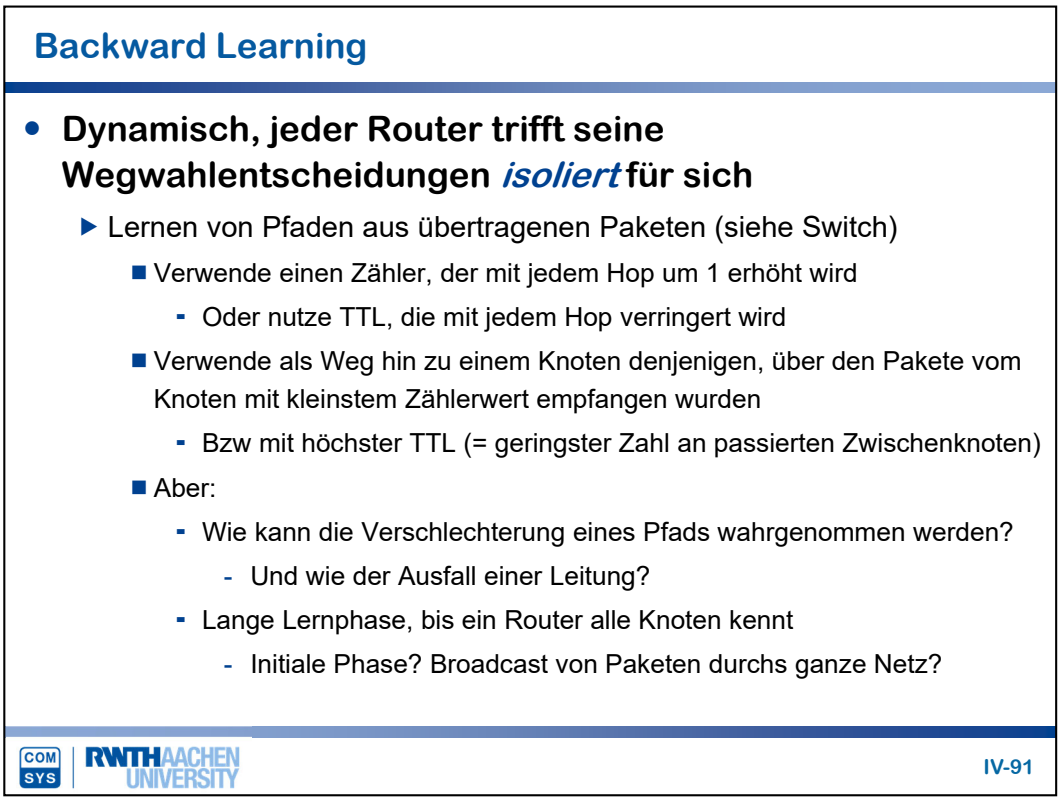
Ziel	1. Wahl		2. Wahl		3. Wahl	
	Next Hop	Gew.	Next Hop	Gew.	Next Hop	Gew.
D	B	0,6	A	0,3	H	0,1
G	H	0,7	J	0,25	B	0,05
…	…	…	…	…	…	…



Ablauf des statischen Routings: jeder Knoten unterhält eine Tabelle mit einer Zeile für jeden möglichen Zielknoten. Der komplette Inhalt dieser Tabellen wird statisch in jedem Rechner konfiguriert. Eine Zeile enthält *n* Einträge, welche die beste, zweitbeste, etc. Übertragungsleitung für dieses Ziel angeben (zusammen mit einer relativen Gewichtung, welche die Eignung der Leitung zur Übertragung angibt). Vor der Weiterleitung eines Pakets wird eine Zufallszahl gezogen und eine der Alternativen anhand der Gewichtung ausgewählt.

Statisches Routing in seiner einfachsten Form wird, wie bereits erwähnt, auch in IP-Netzen verwendet: Hierbei wird in jedem Rechner statisch eine einzelne Route konfiguriert, über die ein Ziel (Endsystem oder Netzwerk) erreicht werden kann. Existiert nur eine Route, benötigt man natürlich keine Gewichtung – dieses Vorgehen kann dann verwendet werden, wenn es mehrere mögliche Wege gibt und man eine Lastverteilung vornehmen will.

Bitte beachten: ist hier und im Folgenden die Rede davon, dass „ein Routing-Eintrag hin zu einem Knoten“ existiert, ist dies eine Abstraktion – in der Realität ist solch ein „Knoten“ der Zugangsrouter zu einem LAN, also einem ganzen Netz von Knoten. Mit einem bestimmten IP-Adressbereich.



Backward-Learning realisiert eine sehr einfache und intuitive Idee – im Prinzip erfolgt das Lernen auf die gleiche Art wie bei Switches. Aufgrund der komplexeren Topologie ergeben sich allerdings im Vergleich zu Switches in LANs einige Nachteile und Probleme.

Nachteile

- Nur Änderungen zum Besseren werden zur Kenntnis genommen. Eine Erhöhung der Kosten ist nicht vorgesehen.
- Ausfälle oder Überlastung von Übertragungsleitungen werden nicht weitergemeldet. Ein Switch wird daher nach einer gewissen Zeit einen Eintrag löschen, falls einige Zeit keine Rahmen mehr von einer Adresse empfangen wurden. In ähnlicher Weise müsste ein Router periodisch sein Wissen verwerfen.

Probleme

- Während der Lernperiode ist das Routing nicht optimal.
- Bei häufigem Verwerfen des gesammelten Wissens nehmen viele Pakete Wege unbekannter Qualität. Bei seltenem Verwerfen ergibt sich ein schlechtes Reaktionsverhalten in Fehlerfällen.

Dynamische Routing-Protokolle

- **Verbesserte Wegewahl**
  - Interaktion zwischen Routern
    - Austausch von Wissen über den Netzzustand
  - Erweiterte Kostenwerte
    - Kosten orientiert an Entfernung, Latenz, Datenrate, …
    - … oder sind einfach immer 1 (wie bei Backward Learning)
- **Zwei Klassen von Protokollen zum Austausch von Routing-Informationen**
  - *Distance Vector*
  - *Link State*

Heutige Routing-Verfahren basieren meist auf dem Informationsaustausch zwischen Routern, damit Routing-Entscheidungen anhand des globalen Zustands eines Netzes getroffen werden können. Die beiden Klassen von Verfahren, die eingesetzt werden, sind Link State Routing und Distance Vector Routing. Beide betrachten ein Netzwerk als einen Graphen, in dem es kürzeste Wege zu finden gilt.

Die Kanten sind dabei mit Kosten für ihre Nutzung beschriftet. Hier können verschiedene Kostenmaße eingesetzt werden: aktuelle Auslastung der Ausgangsleitung, Latenz auf der Übertragungsstrecke, Datenrate auf der Übertragungsstrecke, Fehlerrate auf der Übertragungsstrecke…

Oft wird statt „Kosten“ auch der Begriff „Metrik“ verwendet – denn bei der Berechnung günstiger Pfade interessieren uns die Gesamtkosten für die Nutzung eines bestimmten Pfades; die Einzelkosten der Teilstrecken müssen also zu einem Gesamtwert für die Güte eines Pfades aggregiert werden. Dazu verwendet man eine Metrik – ein Abstandsmaß, welches die Einzelwerte der Teilstrecken zu einem Gesamtwert aggregiert, der die Güte eines Pfades angibt.

In den Folien sowie auch bei Übungsaufgaben (sofern nicht explizit angegeben) werden additive Metriken verwendet, d.h. die Kosten eines Pfades ergeben sich aus der Summe der Kosten der Teilstrecken. Ein kleinerer Wert ist dabei besser.

Werden andere Kostenmaße verwendet, kann es allerdings auch andere Berechnungsvorschriften geben und es könnte auch ein größerer Wert besser sein.

Dynamische Routing-Protokolle

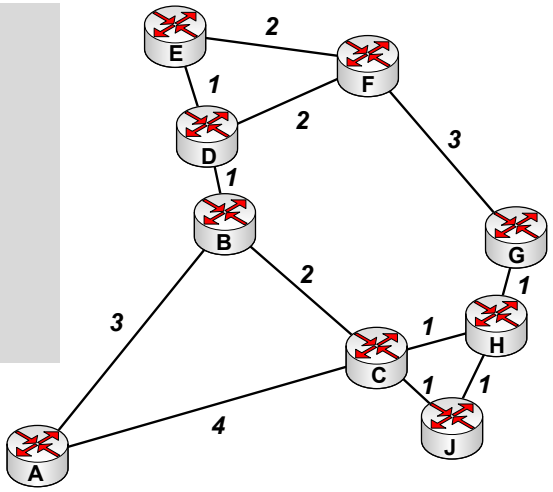
- **Basis: Graphentheorie**

Graph: $G = (N,E)$

N = Menge von Routern
= { *A, B, C, D, E, F, G, H, J*}

E = Menge von Verbindungen
= {(*A,B*), (*A,C*), (*B,C*), (*B,D*), (*D,E*), (*E,F*), (*D,F*), (*F,G*), (*G,H*), (C,H), (C,J), (H,J)}

Kostenfunktion c, z.B. $c(A,C) = 4$



Distance Vector Routing

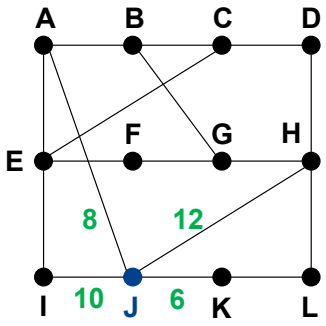
- **Erstes im Internet verwendete Verfahren**
  - Implementiert als *Routing Information Protocol, RIP*
  - Jeder Router hat Routing-Tabelle mit Einträgen
    `(Ziel, Next Hop, Kosten)`
  - Lokaler Austausch globalen Wissens
    - Periodische Generierung von *Abstandsvektoren* (Ziel, Kosten) aus allen Einträgen und Versendung an benachbarte Router
      - Vektor von C: `((A,4), (B,2), (C, 0), (D,3), (E,4), …)`
      - Notation mit $D_x(y)$ = Kosten von x zu y:
        $[D_x(y): y \in N]$
    - Bei Empfang eines Abstandsvektors eines Nachbarn v aktualisiere eigenen Abstandsvektor und Routing-Tabelle:
      - $D_x(y) = \min \{c(x,v) + D_v(y)\}$ for each node $y \in N$

Einfaches Prinzip: tausche mit den Nachbarn Informationen über die eigene Routing-Tabelle aus, indem periodisch Abstandsvektoren aus der Routing-Tabelle generiert werden. Die nötigen Informationen, die anderen Routern zugestellt werden sollten, sind lediglich die Informationen, welche Ziele erreicht werden können und mit welchen Kosten. Auf diese Art wird aus der Routing-Tabelle eine Menge an Tupeln (bekanntes Ziel, Kosten zum Ziel) errechnet – ein Abstandsvektor.

Ein Router, der solch einen Vektor erhält, muss nur wissen, welche Kosten durch die Übertragungsleitung zwischen dem sendenden Router *A* und ihm selbst entstehen, um zu wissen, welche Kosten entstehen würden, um andere Ziele über *A* zu erreichen – addiere auf die im Abstandsvektor enthaltenen Kosten jeweils die Übertragungskosten hin zu *A* auf. Entsteht hierbei ein Wert, der kleiner ist als der, der aktuell in der Routing-Tabelle zu einem Ziel eingetragen ist (oder existiert zu einem Ziel noch gar keine Information), dann speichere den neuen Routingeintrag (Ziel, Next Hop, Kosten) als: (Ziel, A, von A empfangene Kosten + Übertragungskosten hin zu A).

Bei der Implementierung RIP sind die Kosten sehr einfach gehalten: sie betragen für jede Leitung 1, man versucht also nur die Anzahl zu passierender Router zu minimieren.

Distance Vector Routing – Beispiel



Grün: Kosten der Leitungen von J zu anderen Knoten

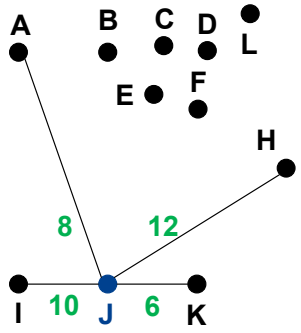
J erhält folgende Abstandsvektoren:

von A nach…		von I nach…		von H nach…		von K nach…	
A		A		A		A	
B		B		B		B	
C		C		C		C	
D		D		D		D	
E		E		E		E	
F		F		F		F	
G		G		G		G	
H		H		H		H	
I		I		I		I	
J		J		J		J	
K		K		K		K	
L		L		L		L	

Neue Routing-Tabelle von *J*:

Ziel	Next Hop	Kosten
A		
B		
C		
D		
E		
F		
G		
H		
I		
J		
K		
L		

Distance Vector Routing – Beispiel



Grün: Kosten der Leitungen von J zu anderen Knoten

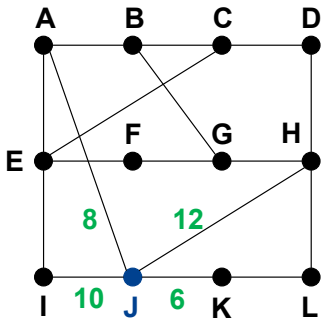
J erhält folgende Abstandsvektoren:

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

Neue Routing-Tabelle von *J*:

Ziel	Next Hop	Kosten
A		
B		
C		
D		
E		
F		
G		
H		
I		
J		
K		
L		

Distance Vector Routing – Beispiel



Grün: Kosten der Leitungen von J zu anderen Knoten

J erhält folgende Abstandsvektoren:

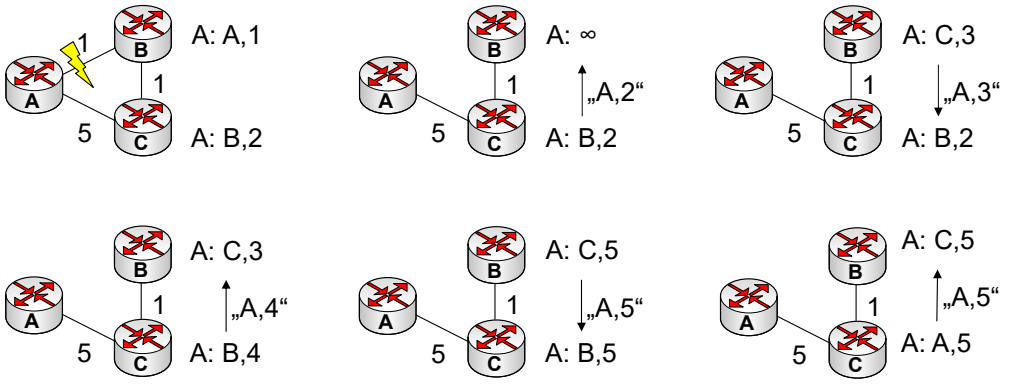
von A nach…		von I nach…		von H nach…		von K nach…	
A	0	A	24	A	20	A	21
B	12	B	36	B	31	B	28
C	25	C	18	C	19	C	36
D	40	D	27	D	8	D	24
E	14	E	7	E	30	E	22
F	23	F	20	F	19	F	40
G	18	G	31	G	6	G	31
H	17	H	20	H	0	H	19
I	21	I	0	I	14	I	22
J	9	J	11	J	7	J	10
K	24	K	22	K	22	K	0
L	29	L	33	L	9	L	9

Neue Routing-Tabelle von *J*:

Ziel	Next Hop	Kos-ten
A	A	8
B	A	20
C	I	28
D	H	20
E	I	17
F	I	30
G	H	18
H	H	12
I	I	10
J	-	0
K	K	6
L	K	15

Distance Vector Routing – Probleme

- **Probleme bei Distance Vector Routing**
  - Bei größeren Netzen *langsame Konvergenz* zu einem konsistenten Zustand (langsame Informationsausbreitung)
    - *Bouncing Effect* – langsames Hochschaukeln von Einträgen benachbarter Router



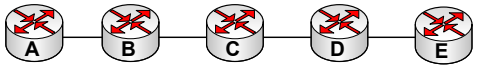
Das Distance Vector Routing war das ursprüngliche Routing-Verfahren, das im Arpanet unter dem Namen RIP implementiert wurde. Ein Problem ist die langsame Konvergenz im Falle eines Systemausfalls oder einer Änderung der Topologie: Änderungen der Informationen eines Routers werden nur an benachbarte Router weitergereicht. Diese verarbeiten die empfangenen Informationen und reichen entsprechende Änderungen wieder an ihre Nachbarn weiter. Durch diese lokale Weitergabe der Änderungen verbreitet sich eine Topologie-Änderung nur langsam – besonders bei Implementierungen wie RIP, bei denen nur alle 30 Sekunden eine Weitergabe von Änderungen erfolgt.

Darüber hinaus treten durch die lokale Weitergabe aber noch weitere Probleme auf: der sogenannte Bouncing-Effect, bei dem die Kosten in Routing-Einträgen sich aufgrund der Weitergabe veralteten Wissens zwischen zwei Routern langsam hochschaukeln, und Count-to-Infinity als Extremfall, wenn keine alternative Route gefunden werden kann.

Die Problematiken wurden zwar in einer Reihe von Erweiterungen des Algorithmus‘ zu beheben versucht, jedoch konnte keine entscheidende Verbesserung erzielt werden. Deshalb wurde das Verfahren schließlich breitflächig vom sogenannten Link State Routing abgelöst.

Distance Vector Routing – Probleme

- **Probleme bei Distance Vector Routing**
  - Bei größeren Netzen *langsame Konvergenz* zu einem konsistenten Zustand (langsame Informationsausbreitung)
    - *Count-to-Infinity* – Extremfall des Bouncing Effect
      - Beispielszenario: Router *A* fällt aus:



					Abstand in Hops zu *A*
	1	2	3	4	1. Austausch
∞ →	3	← 2	3	4	2. Austausch
	3	4	3	4	3. Austausch
	5	4	5	4	4. Austausch
	5	6	5	6	5. Austausch
	7	6	7	6	6. Austausch
	7	8	7	8	...
	...	...	...	...	je nach Festlegung der Obergrenze
	∞	∞	∞	∞	

Implementierung: RIP

- **Erstes Intra-AS-Protokoll im Internet**
  - Abstandsvektoren werden alle 30 Sekunden ausgetauscht
    - Konvergenz im Minutenbereich!
  - 4 Bit für Kosten – maximaler Abstand zu Zielen ist 14 Hops
    - 15 wird als „unendlich“ = unerreichbar angesehen
  - Maximal 25 Abstandswerte können in einer Nachricht versendet werden
- **Erweiterungen: RIPv2, RIPv3**
  - Authentifizierung bei Informationsaustausch
  - Multicasting von Abstandsvektoren für schnellere Konvergenz
  - IPv6-Unterstützung

Trotz der Probleme wurde RIP weiterentwickelt und kann auch heute noch innerhalb von Autonomen Systemen eingesetzt werden – es bleibt jedem AS-Betreiber überlassen, ein eigenes Protokoll auszuwählen.

Implementierung: BGP

- *Border Gateway Protocol*
  - Aktuell: BGPv4
  - *Einziges Inter-AS-Protokoll*
  - Distance Vector mit kleinen Modifikationen
    - Tausche keine Abstandsvektoren aus, sondern komplette Pfade hin zu den Zielen
    - Damit möglich:
      - Berücksichtigung von Policies (z.B. Vermeidung der Paketweiterleitung durch bestimmte ASs)
      - Kein Bouncing-Effekt, kein Count-to-Infinity
  - Verfügbar als eBGP (exterior BGP) und iBGP (interior BGP)
  - Auch Unterstützung von IPv6

Bei BGP werden keine Abstandsvektoren ausgetauscht, sondern komplette Pfade. Dadurch können der Bouncing-Effect und Count-to-Infinity vermieden werden; zudem ist es für Routerbetreiber möglich, eigene Policies durchzusetzen, z.B. die Vermeidung bestimmter AS bei der Weiterleitung von Paketen.

BGP ist das Protokoll, das zum Routing zwischen autonomen Systemen eingesetzt wird (eBGP). Zusätzlich gibt es auch eine Variante zum Einsatz innerhalb autonomer Systeme (iBGP).

Link State Routing

- **Algorithmus:**
  1. Erkennung der Nachbarn (`Hello`-Pakete)
  2. Kostenschätzung/-messung (`Echo`-Pakete)
  3. Erzeuge Link-State-Paket (*Link State Advertisement, LSA*)
     - Inhalt: Absender, Sequenznummer, Alter, Liste mit Nachbarn + Kosten
  4. Verteile Link-State-Paket auf angeschlossenen Netzen
     - Periodisches oder ereignisgesteuertes *Flooding* mit Sequenznummern
     - Vernichten von Duplikaten und veralteten Link-State-Informationen.
     - Bestätigung von erhaltenen Link-State-Paketen
  5. Berechnung von Routing-Einträgen aus empfangenen Informationen
     - Z.B. Dijkstra-Algorithmus
- **Implementierung als *OSPF (Open Shortest Path First)***

Anfang der 80er Jahre wurde das Distance Vector Routing von einem effizienteren Verfahren mit Link State Routing abgelöst (die derzeit im Internet eingesetzten Routing-Protokolle OSPF und IS-IS basieren auf diesem Verfahren). Die wichtigsten Vorteile stellen dabei die schnellere Reaktionsfähigkeit (Konvergenz) beim Ausfall einzelner Knoten/Verbindungen sowie die Unterstützung mehrerer Wege zum Zielsystem dar.

Das Prinzip von Link-State-Algorithmen ist die globale Verteilung von lokalem Wissen – jeder Router ermittelt für sich seine Nachbarn sowie die Kosten der Leitungen hin zu den Nachbarn und sendet diese Information per Flooding ins Netz. Um das Flooding effizient zu gestalten, werden zwei Einträge in den Routing-Paketen verwendet:

- Sequenznummer zur Elimintion von Duplikaten
- Altersangabe – Zeitangabe seit Generierung des LSAs. Nach Ablauf, d.h. bei Überalterung des Pakets werden die enthaltenen Angaben nicht mehr zur Erstellung von Routing-Tabellen berücksichtigt, damit man Netzinkonsistenzen aufgrund veralterter Link-Informationen vermeidet.

Da sichergestellt werden muss, dass alle Router alle Nachrichten erhalten, damit jeder seine Routing-Tabelle auf Basis der gleichen Informationen berechnet, wird der Erhalt von Routing-Paketen zwischen benachbarten Routern bestätigt.

Hat ein Router die Informationen aller anderen Router erhalten, kennt er die genaue Topologie des Netzes und kann damit beginnen, kürzeste Wege hin zu allen Routern (und damit zu den an sie angeschlossenen Teilnetzen) zu berechnen. Dies kann z.B. durch Anwendung des Dijkstra-Algorithmus‘ erfolgen.

Flooding

- **Ähnlich wie Broadcasting:**
  - Jedes eingehende Paket wird auf *jeder* Übertragungsleitung weitergeleitet, außer auf derjenigen, auf der es eintraf
    - Aber: Paketflut, Zirkulieren von Paketen
- **Maßnahmen zur Eindämmung der Flut**
  - Erkennung von Duplikaten durch Nummerierung der Pakete
  - Kontrolle der Lebensdauer eines Pakets durch Zählen der zurückgelegten Teilstrecken (wie TTL)
  - Optimierung: *selektives Flooding* – Weiterleitung nicht auf allen, sondern nur auf ausgewählten Leitungen
    - Nicht in OSPF, verwendet z.B. in drahtlosen Mesh-Netzen

Flooding ist zwar das einfachst denkbare Verfahren und kommt ohne Algorithmen und Tabellen aus, führt aber zu enormen Problemen: wenn jeder Router jede Nachricht an jeden anderen weiterleitet, werden massig Duplikate erzeugt, die auch weitergeleitet werden, was im schlimmsten Fall dazu führt, dass Pakete auf ewig im Netz zirkulieren und die Netzlast immer weiter steigt.

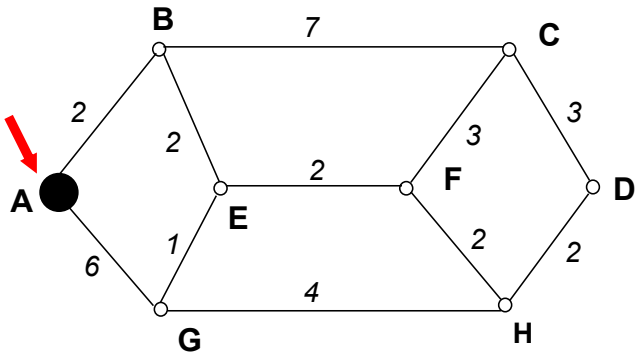
Daher muss man noch Zusatzinformationen in den Paketen mitschicken: zum einen kann durch die TTL die Anzahl der Weiterleitungsschritte begrenzt werden, zum anderen sollte ein Router jedes Paket nur einmal weiterleiten und nicht später bei ihm über andere Wege eintreffende Duplikate noch einmal. Daher werden Sequenznummern in den Paketen verwendet, und ein Router merkt sich, welche Pakete er bereits empfangen hat und wird später eintreffende Duplikate verwerfen.

Dijkstra-Algorithmus

- **Algorithmus von Dijkstra (1959)**
  1. Mache den Ausgangsknoten zum “Arbeitsknoten” und markiere ihn als “permanent” (d.h. die Entfernung zu ihm wird sich nicht mehr ändern)
  2. Betrachte alle Nachbarknoten des Arbeitsknotens und berechne aufgrund der Kosteninformationen an den Kanten ihre Distanz zum Ausgangsknoten.
     Packe alle Nachbarn mit ihren Distanzen in eine “Arbeitsmenge”, falls sie sich nicht schon mit geringeren Kosten darin befinden oder bereits als permanent markiert sind
  3. Wähle aus der Arbeitsmenge den Knoten mit geringster Distanz zur Quelle.
     Mache ihn zum Arbeitsknoten und markiere ihn als permanent.
     Gehe zurück zu 2.
  4. Ein Abbruch erfolgt, sobald die Arbeitsmenge leer ist.

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**
  1. Markiere A als permanent
  2. Betrachte die Nachbarn von A (B, G)



Permanent: {A}

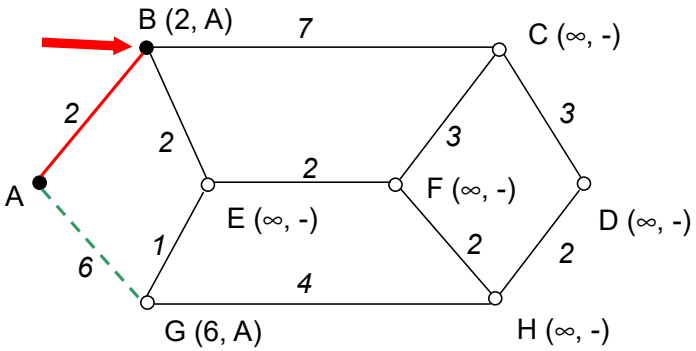
Arbeitsmenge:
{B, G}

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

  Um später den Pfad auslesen zu können, speichern wir jeweils die Information, wie wir einen Knoten erreicht haben

  3. B wird Arbeitsknoten (permanent), da er die kürzere Distanz zu A hat
  4. Betrachte die Nachbarn von B

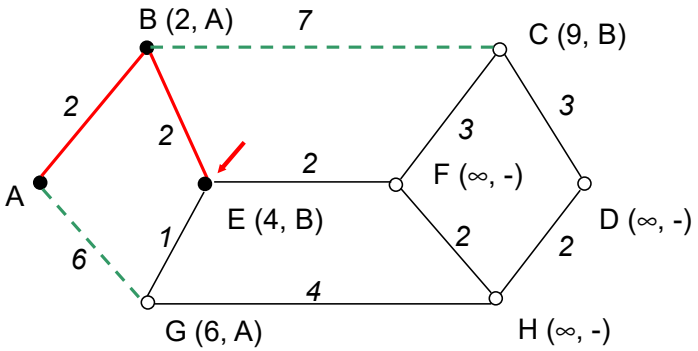


Permanent: {A, B}

Arbeitsmenge:
{C, E, G}

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**
  5. E wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat
  6. Betrachte die Nachbarn von E



Permanent: {A, B, E}

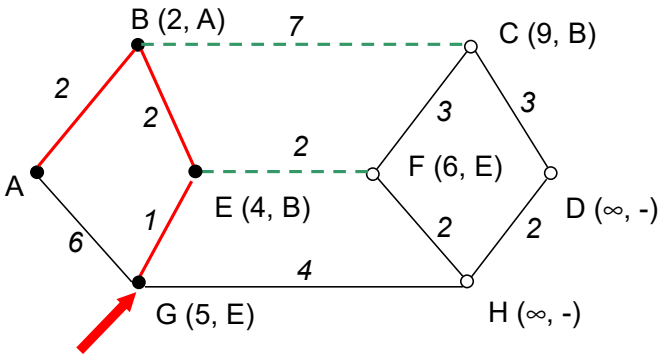
Arbeitsmenge:
{C, F, G}

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

  Vorheriges Label von G wird überschrieben, da ein kürzerer Weg hin zu ihm gefunden wurde

  7. G wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat
  8. Betrachte die Nachbarn von G

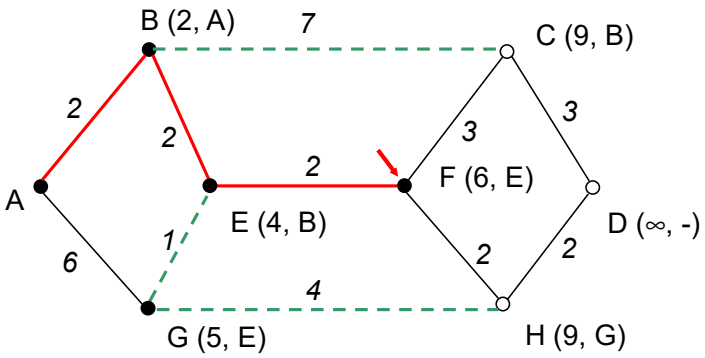


Permanent: {A, B, E, G}

Arbeitsmenge:
{C, F, H}

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**
  9. F wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat
  10. Betrachte die Nachbarn von F

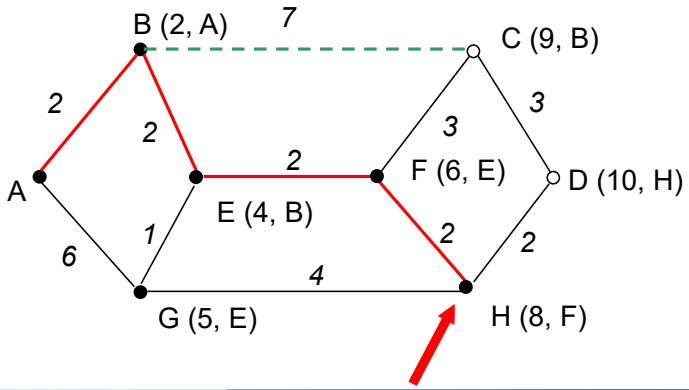


Permanent:
{A, B, E, F, G}

Arbeitsmenge:
{C, H}

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**
  11. H wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat
  12. Betrachte die Nachbarn von H

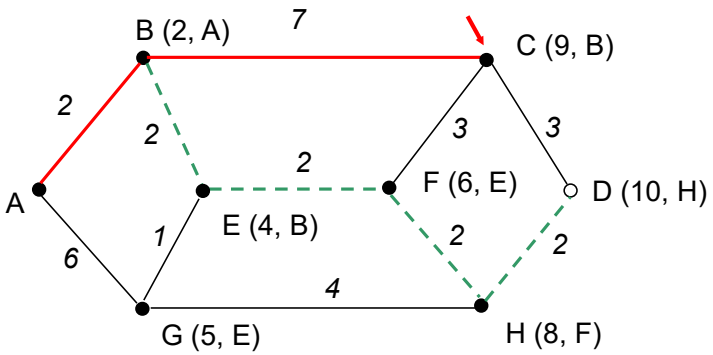


Permanent:
{A, B, E, F, G, H}

Arbeitsmenge:
{C, D}

Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**
  13. C wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat
  14. Betrachte die Nachbarn von C



Permanent:
{A, B, C, E, F, G, H}

Arbeitsmenge:
{D}

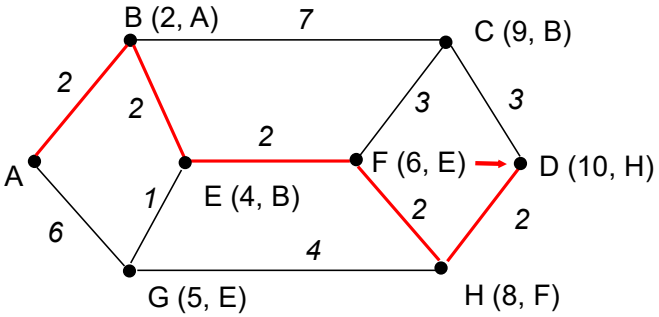
Dijkstra-Algorithmus – Beispiel

- **Beispiel: Berechnung des kürzesten Pfades von A nach D**

  15. D wird Arbeitsknoten (permanent), da er die kürzeste Distanz zu A hat

      Da D permanent ist, kann kein kürzerer Pfad hin zum Ziel mehr existieren; der kürzeste Weg ist gefunden und kann nun aus den Labeln vom Ziel zur Quelle ausgelesen werden

  In der Routing-Tabelle von A wird gespeichert
  „Ziel D, Next Hop B, Kosten: 10“



Permanent:
{A, B, C, D, E, F, G, H}

Arbeitsmenge:
{}

Errechnete Routing-Tabelle

- **Dijkstra-Algorithmus aus vorherigem Beispiel**
  - Ermittlung des kürzesten Weges zu D
  - Aber: kürzeste Wege zu allen Knoten wurden gleichzeitig ermittelt
  - Daher vollständige Generierung der Routing-Tabelle durch einen Lauf:

nach	Next Hop	Kosten
B	B	2
C	B	9
D	B	10
E	B	4
F	B	6
G	B	5
H	B	8

Open Shortest Path First (OSPF)

- **Weit verbreitete Implementierung: OSPF**
  - 1990 standardisiert durch IETF (RFC 1247)
  - OSPFv3: Unterstützung von IPv6
  - Unterstützt viele verschiedene Metriken als Kosten
  - Erzeuge LSAs je nach Konfiguration alle 10 bis 30 Sekunden
  - Flooding von LSAs mittels Multicast an `224.0.0.5`
  - Lastverteilung möglich – Speicherung alternativer Wege, probabilistische Verteilung von Daten über alternative Wege
  - Authentifizierung beim Austausch von LSAs

Zusammenfassung

- **IP als zentrales Protokoll zur Kopplung von Netzen**
  - Verbindungslos, unzuverlässig
  - Definiert Weiterleitungsregeln, Adressschema
  - Network Address Translation zum Umgang mit Adressknappheit
  - IPv4 vs. IPv6
- **Hilfsprotokolle**
  - ARP zur Ermittlung des Zielknotens auf dem letzten Hop
  - DHCP (Schicht-7-Protokoll) zur automatischen Konfiguration
  - ICMP zum Austausch von Kontrollnachrichten
- **Routingprotokolle**
  - Distance Vector vs. Link State
  - In autonomen Systemen können Netzprovider bevorzugte Protokolle wählen

Datenkommunikation
Kapitel 5: Transportschicht

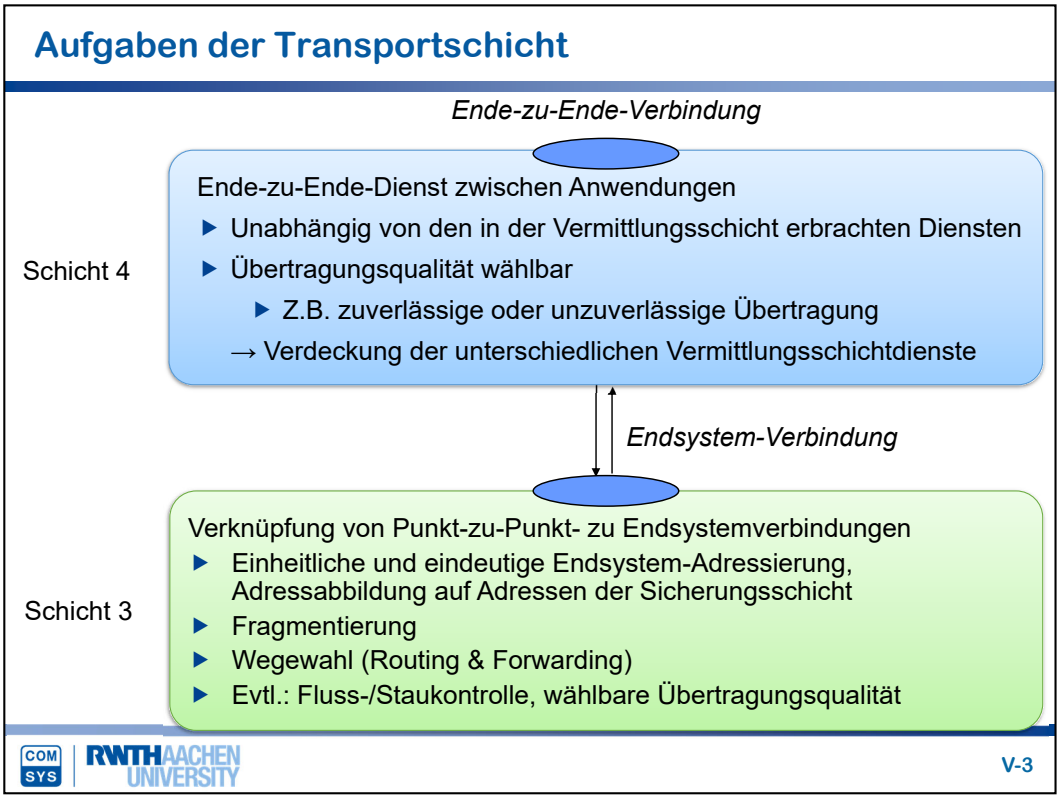
Klaus Wehrle

Communication and Distributed Systems
Chair of Computer Science 4
RWTH Aachen University

http://www.comsys.rwth-aachen.de

Themenübersicht

- **Datenkommunikation**
  - Einführung, Begriffe und allgemeine Grundlagen
  - Technische und nachrichtentechnische Grundlagen: Medien, Signale, Bandbreite, Leitungscodes und Modulation, Multiplexing
  - Lokale Netze: Strukturierung des Datenstroms, Fehlererkennung/ -behebung, Flusssteuerung, Medienzugriff, Ethernet
  - Vermittlungsschicht
    - Leitungsvermittlung
    - Internet und Internet-Protokolle:
    - Routing
  - Transportschicht
    - TCP und Co.
  - Anwendungsschicht



Anwendungen benötigen für die Kommunikation untereinander einen meist zuverlässigen und einheitlichen Dienst, welcher die unterschiedlichen Leistungen der Vermittlungsschicht verdeckt. Dieser Dienst wird von der vierten Schicht im ISO/OSI-Basisreferenzmodell, der *Transportschicht*, erbracht, welche in diesem Kapitel behandelt wird.

Die Transportschicht ist nicht einfach „nur eine weitere Schicht“ im OSI-Modell. Sie repräsentiert das *Kernstück* der gesamten Protokollhierarchie, welche die netzwerkbezogenen, technischen Eigenschaften des Kommunikationssystems von den anwendungsorientierten Aspekten trennt. Die Anwendungen können somit auf einen einheitlichen (und, je nach gewählter Protokollinstanz, auch zuverlässigen) Dienst zugreifen, unabhängig von den verwendeten Technologien und der Topologie des darunterliegenden Netzwerkes. Im Gegensatz zur Vermittlungsebene, bei der sich die Kommunikation auf Rechnerknoten bezieht, werden in der Transportschicht *Anwendungen* adressiert, die miteinander kommunizieren.

Instanzen der Transportschicht finden sich nur in den Endsystemen, d.h. die Instanzen sind völlig unabhängig von den unterliegenden Netzen. Außerdem werden auf der Schicht 4 nicht Rechnersysteme adressiert sondern Anwendungen, d.h. es werden evtl. mehrere Transportschicht-Verbindungen zwischen zwei Anwendungen über eine Schicht-3-Verbindung geführt (Multiplexing). Außerdem ist es möglich, Quality of Service (Dienstgüte, Dienstqualität) für eine Transportverbindung festzulegen.

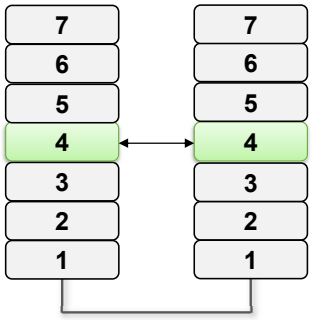
Funktionalitäten zur Erbringung solcher Dienste sind die Flusssteuerung des Datenstromes zwischen zwei Teilnehmern, das Multiplexen mehrerer Verbindungen auf eine Endsystemverbindung (zur Optimierung der Nutzung einer zur Verfügung stehenden Endsystemverbindung) und das Splitten einer Schicht-4-Verbindung auf mehrere Schicht-3-Verbindungen zur Erhöhung von Qualität und Sicherheit.

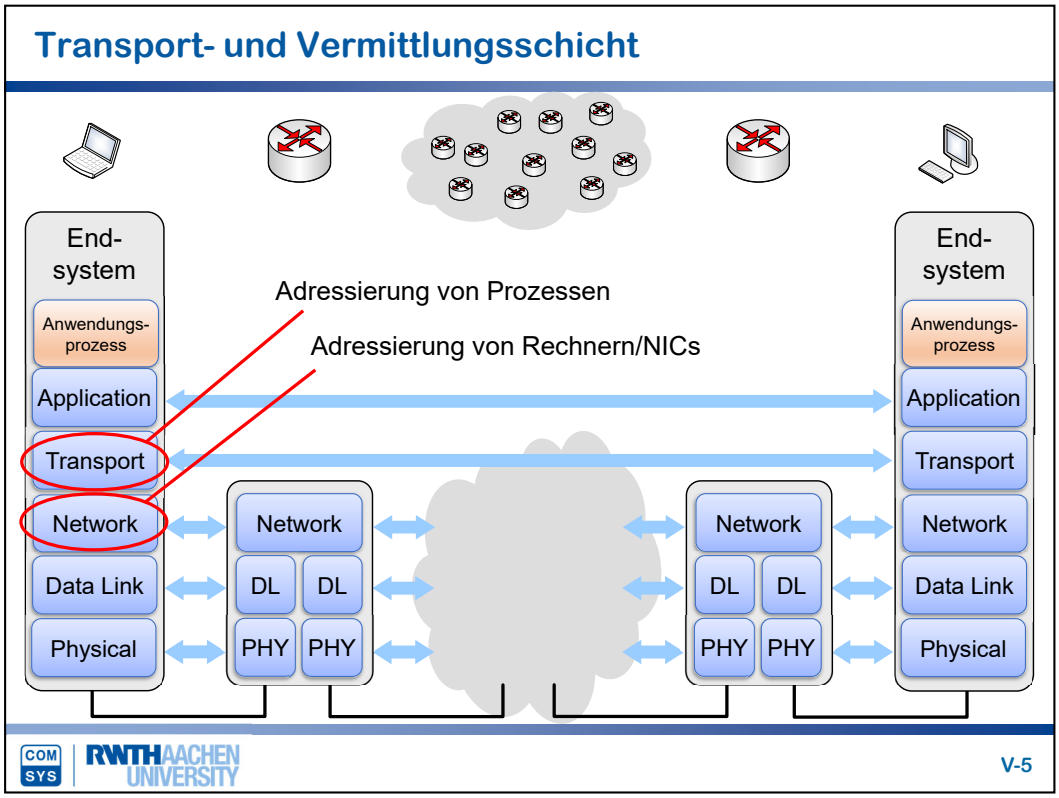
Der Transportschicht kommt – besonders im Internet – eine zentrale Bedeutung zu, da das dort verwendete IP zur Kommunikation zwischen Rechnern verbindungslos und unzuverlässig arbeitet:

- Eine Reihenfolgetreue der Pakete ist nicht gewährleistet
- Pakete können während der Übertragung verloren gehen oder verfälscht werden

Die Behandlung dieser Probleme wird – wenn benötigt – von der Transportschicht übernommen.

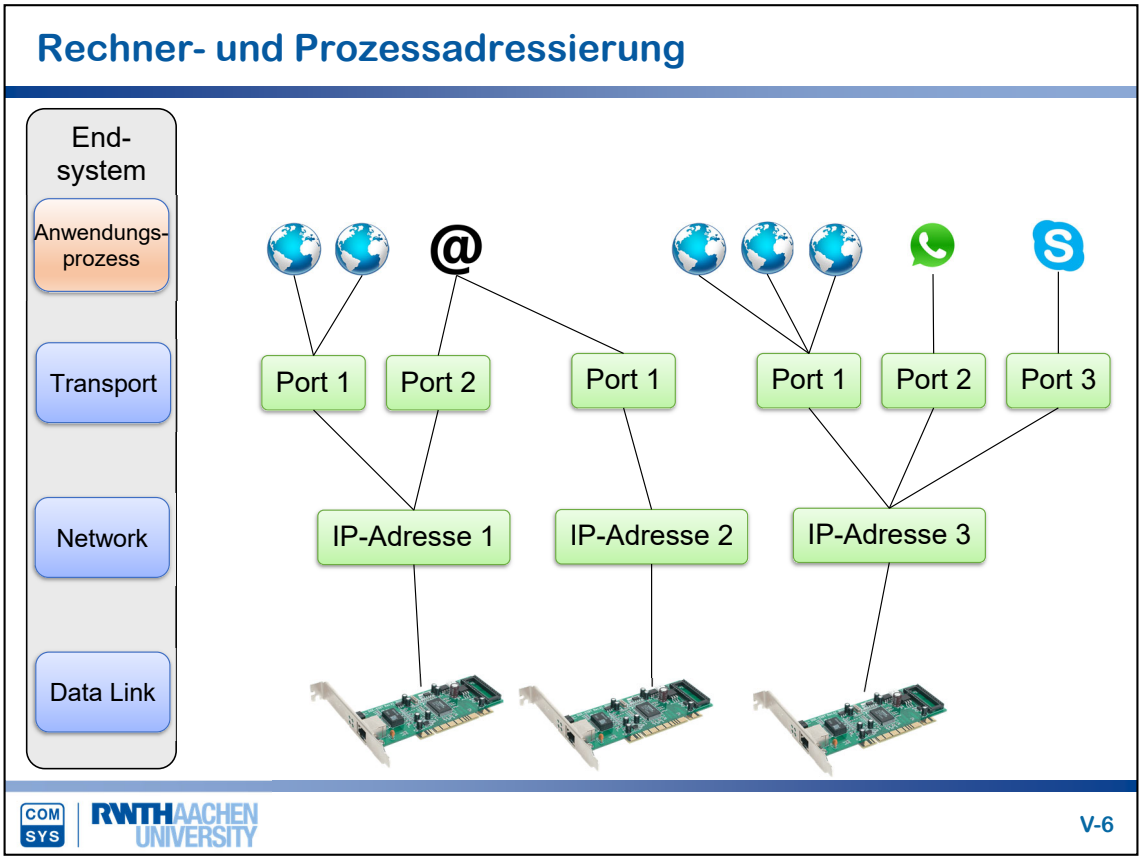
- **Protokollmechanismen der Transportschicht**
  - Prozessadressierung, strombasierte vs. paketbasierte Kommunikation, verbindungsorientiert/verbindungslos
- **Die Transportschicht im Internet**
  - TCP (Transmission Control Protocol)
    - Adressierung, Sockets
    - TCP-Verbindung
    - Flusskontrolle
    - TCP-Header
    - Staukontrolle (Congestion Control)
  - UDP (User Datagram Protocol)





Da die Transportschicht die kommunizierenden Anwendungen auf den Endrechnern adressiert und vom Netzwerk trennt, ist ihre Implementierung auch nur auf den Endsystemen notwendig. Router (oder generell: Vermittlungsknoten) im Netz brauchen sie nicht zu implementieren.

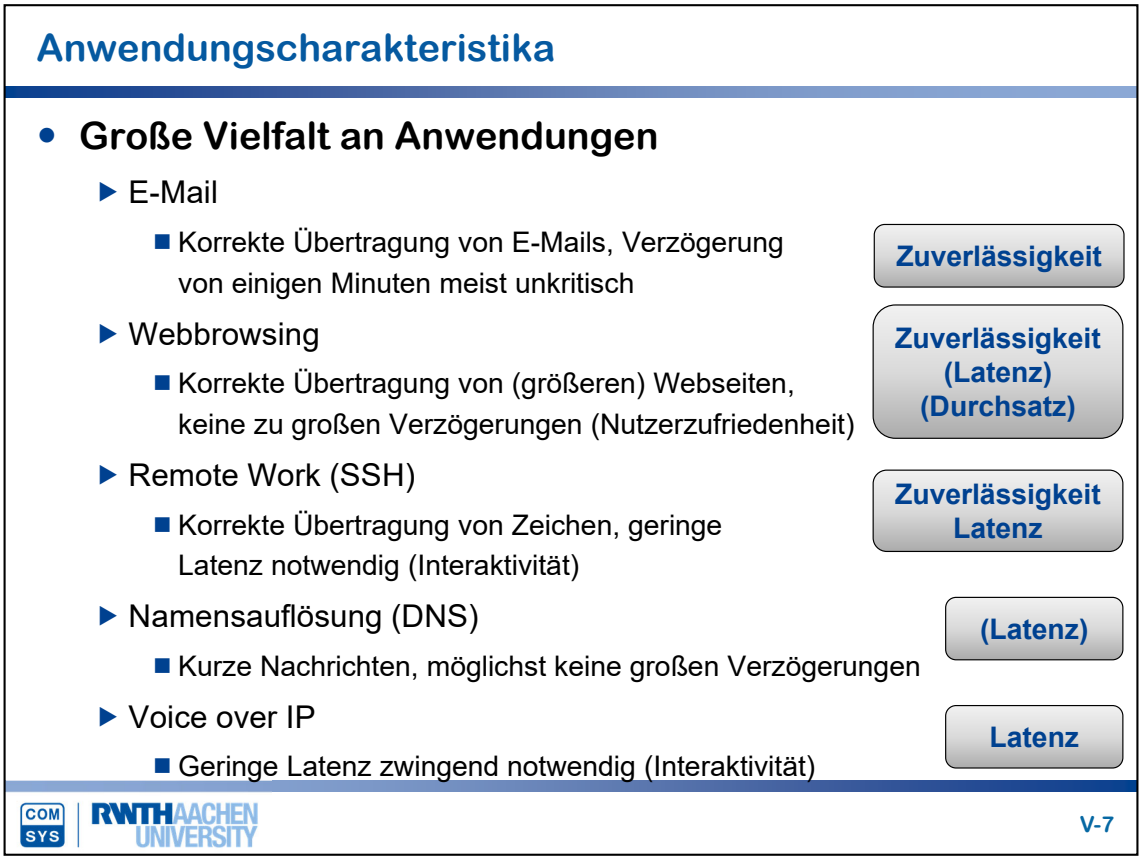
Während auf der Vermittlungsschicht (Schicht 3) eine Endsystemverbindung besteht, wird auf der Transportschicht eine Ende-zu-Ende-Verbindung zwischen den Teilnehmern (Anwendungen) aufgebaut. Die Transportinstanzen verständigen sich untereinander über das *Transportprotokoll*, also durch den Austausch von Transport-Protokolldateneinheiten (TPDUs: Transport Protocol Data Units). Diese TPDUs werden über die Endsystemverbindung übertragen.



Jeder Rechner muss anhand einer IP-Adresse adressierbar sein – aber man kann durchaus einem Rechner bzw. genauer gesagt einer Netzwerkkarte auch mehrere IP-Adressen zuweisen.

Die Transportschicht dient dazu, mehrere Anwendungsprozesse auf eine IP-Adresse zu multiplexen. Dazu werden sogenannte Port-Nummern verwendet: verschiedene gleichzeitig auf einem Rechner laufende Anwendungen können anhand eines lokalen Index voneinander unterschieden werden.

Unter Umständen ist es sogar möglich, mehrere Anwendungsprozesse gemeinsam über einen Port zu adressieren.



Die Transportschicht hat aber nicht nur die Aufgabe, die Kommunikationsverbindungen verschiedener Anwendungsprozesse auf eine Netzwerkkarte zu multiplexen. Die zweite wesentliche Aufgabe ist, den Anwendungsprozessen einen Kommunikationsdienst bereitzustellen, der ihre Anforderungen erfüllt – dazu muss die Transportschicht Probleme, die auf der Netzwerkschicht auftreten können, vor den Anwendungen verbergen.

Das große Problem für die Transportschicht ist, dass sie mit dem Best-Effort-Dienst von IP leben und versuchen muss, die Anforderungen der verschiedenen Anwendungen zu erfüllen. Anwendungen wie E-Mail oder Webanwendungen benötigen eine zuverlässige Übertragung (vollständig, reihenfolgetreu) und darüber hinaus eventuell noch mehr – Webseiten sind heute oft sehr umfangreich, aber die Nutzer erwarten eine schnelle Anzeige der Inhalte, so dass auch eine relativ geringe Latenz und ein hoher Durchsatz benötigt werden. Noch kritischer sind Remote-Login-Anwendungen wie SSH: eine zuverlässige Übertragung der Kommandos ist notwendig, aber um dem Anwender ein Gefühl der Interaktivität zu vermitteln, muss auch eine geringer Latenz vorliegen.

DNS wird verwendet, um logische Rechnernamen in IP-Adressen aufzulösen, so dass wir Rechner im Web mit Namen adressieren können. Die hier ausgetauschten Nachrichten sind relativ klein und die Namensauflösung sollte schnell sein, um schnell die eigentliche Datenübertragung hin zur aufgelösten IP-Adresse beginnen zu können.

Interaktive Anwendungen, bei denen Sprache und/oder Bild übertragen werden,

benötigen zwingend eine geringe Latenz. Wird Video übertragen, ist noch dazu ein relativ großer Durchsatz wichtig. Zuverlässigkeit spielt hingegen keine allzu große Rolle, da einzelne fehlende Teile eines Videos oder einer Sprachsequenz dem Benutzer nicht auffallen.

Ganz generell lassen sich diese Anforderungen in Zuverlässigkeit einerseits und Latenz andererseits aufteilen. Als Lösung hat man auf der Transportschicht mehrere Protokolle definiert, die unterschiedliche Anforderungen erfüllen: TCP für Zuverlässigkeit, UDP für Einfachheit und damit geringe Latenz. Im Laufe der Zeit wurden weitere Transportprotokolle entwickelt, die sich alle nicht durchgesetzt haben; erst aktuell kommt mit QUIC ein neues Transportprotokoll, dass gleichzeitig Zuverlässigkeit und geringe Latenz erreichen soll.

Verbindungslos oder verbindungsorientiert?

- **Anforderung Zuverlässigkeit**
  - Oft größere Menge an Daten (z.B. komplexe Webseiten)
    - *Strombasiert* – gesamter Datenstrom muss in Segmente unterteilt und beim Empfänger reassembliert werden
      - Übertragung der Segmente in unabhängigen IP-Paketen
    - Kontext zwischen den Segmenten notwendig

    → *Verbindungsorientierte Kommunikation*

- **Anforderung Latenz**
  - Oft geringe Mengen an Daten (ein Paket) oder zeitkritische Daten ohne hohe Zuverlässigkeitsanforderungen
    - *Paketbasiert* – Versendung einzelner Pakete ohne Kontext oder Kontextverwaltung in der Anwendung
    - Kein Kontext auf Transportschicht notwendig

    → *Verbindungslose Kommunikation*

Die große Vielfalt an Anwendungen sorgt für unterschiedliche Anforderungen – die nicht alle in einem einzigen Transportprotokoll erfüllt werden können. Denn zur Erfüllung der Anforderungen sind bereits unterschiedliche Grundprinzipien nötig.

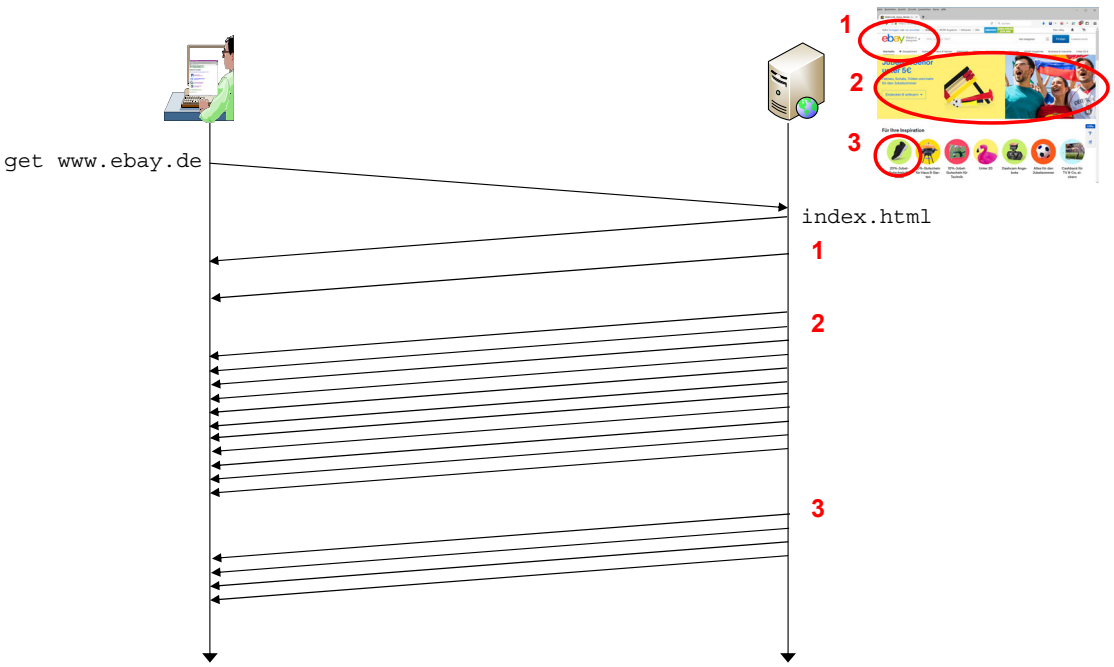
Um Zuverlässigkeit erreichen zu können, muss man mit Quittungsmechanismen arbeiten – und oft sind die Datenmengen so umfangreich, dass die zu übertragenden Daten als Datenstrom aufgefasst werden können, der in mehrere Pakete aufgeteilt werden muss. Daher benötigt man einen Kontext bei der Übertragung, muss also ein verbindungsorientiertes Protokoll definieren.

Um geringe Latenz zu erzielen, stört ein Verbindungsaufbau – er fügt eine Latenz hinzu. Gerade wenn nur kurze Nachrichten übertragen werden, die in einzelne Pakete passen (wie z.B. bei DNS), wäre eine verbindungslose Übertragung sinnvoller. Paketverlust ist in den meisten Fällen zwar vielleicht ärgerlich, aber nicht kritisch, so dass eine Anfrage einfach noch einmal wiederholt werden kann, wenn keine Antwort kommt.

Bei Anwendungen wie Telefonie ist eine verbindungsorientierte Übertragung auch hinderlich, da ihre Kontrollmechanismen einen Overhead erzeugen können, der die Übertragung verzögert. Hier bevorzugt man verbindungslose Kommunikation und übernimmt eine Kontextverwaltung auf Anwendungsebene. Zuverlässigkeit ist nicht kritisch, da das menschliche Hirn kleinere Mengen an fehlenden Sprachinformationen ausfiltert.

Neben diesen beiden Anforderungen ist auch Durchsatz wichtig – für die Übertragung moderner Webseiten, die dem Benutzer ein Gefühl von Interaktivität geben sollen, oder auch für z.B. Videostreaming. Einen hohen Durchsatz könnte man allerdings mit beiden Techniken erreichen. (Allerdings muss man vorsichtig vorgehen, um das Netz nicht zu überlasten, wie im Folgenden dargestellt.)

Strombasierte Übertragung



Ein gutes Beispiel für strombasierte Übertragung ist die Übertragung von Webseiten. Moderne Webseiten bestehen aus vielen unterschiedlichen Elementen, die oft sehr groß sind und über mehrere Pakete verteilt werden müssen. Hier im Beispiel wird zuerst das Gerüst der Webseite übertragen, danach in einem Paket das Logo, dann eine sehr große Abbildung, zum Schluss eine kleinere Abbildung.

Man muss eine variable Menge an Daten in Pakete zerteilen – dazu werden die gesamten zu übertragenden Daten als Bytestrom gesehen, der in Pakete segmentiert werden muss.

Welche Probleme kann IP verursachen?

- Paketverlust
  - Oft keine Fehlerbehebung auf Schicht 2 implementiert
  - Router können Pakete verwerfen, z.B. bei Überlast

  → Fehlerbehebung (ARQ) Stauvermeidung
- Pakete können in falscher Reihenfolge ankommen
  - Durch Übertragung auf unterschiedlichen Pfaden
- Pakete können dupliziert werden
  - Durch Neuübertragung bei verlorengegangener Quittung

  → Reihenfolge- und Duplikaterkennung
- Verzögerung von Paketen
  - Durch hohe Routerlast

  → Stauvermeidung
- Keine Flusskontrolle vorhanden
  - Flusskontrolle auf Schicht 2 nur für einzelne Hops
    - Keine Ende-zu-Ende-Vermeidung von Überlastung eines Empfängers
  - Keine Vermeidung von Router-Überlastung

  → Flusskontrolle Stauvermeidung

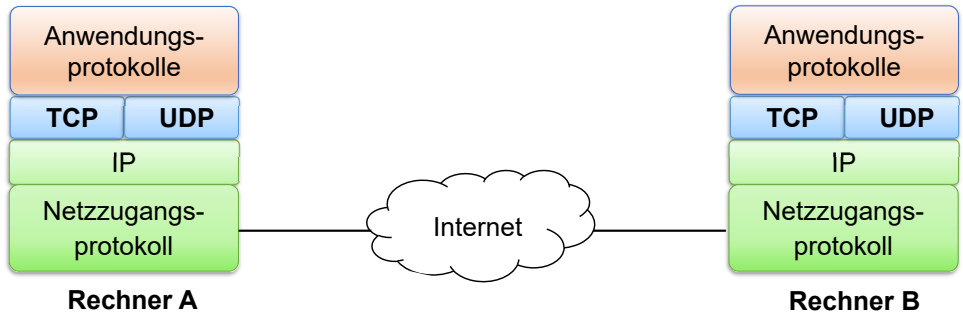
IP bietet weder Zuverlässigkeit noch gibt es die Möglichkeit, eine bestimmte Latenz zu garantieren. Selbst wenn auf Schicht 2 entsprechende Mechanismen z.B. zur Sicherstellung der Zuverlässigkeit realisiert sind, treten auf Schicht 3 wieder neue Probleme auf, so dass die Transportschicht wieder mit vielen Problemen konfrontiert ist, die bereits auf Schicht 2 existierten – nur jetzt auf globaler Ebene.

Umsetzung der Transportschicht

- ***Verbindungsorientiert*, Hauptaspekt Zuverlässigkeit**
  - Sequenznummern und ARQ zur Behebung von Paketverlust, Reihenfolgeumstellung, Paketduplizierung
  - Ende-zu-Ende-Flusskontrolle pro Anwendung
  - Staukontrolle zur Vermeidung von Routerüberlast
  - Möglichkeit eines hohen Durchsatzes trotz Staukontrolle
- ***Verbindungslos*, Hauptaspekt Latenz**
  - Keine Kontrollmechanismen (Vermeidung von Overhead)
  - Stauvermeidung dennoch sinnvoll

Die Transportschicht im Internet

- **Hauptsächlich zwei Protokolle:**
  - ***TCP** (Transmission Control Protocol)*: Zuverlässiges, verbindungsorientiertes Transportprotokoll über unzuverlässigem IP
  - ***UDP** (User Datagram Protocol)*: Verbindungsloses Transportprotokoll, fügt lediglich eine Anwendungsschnittstelle zu IP hinzu



Im Internet werden auf Transportebene vorwiegend TCP und UDP genutzt. Der wesentliche Unterschied zwischen den beiden Transportprotokollen ist, dass TCP verbindungsorientiert arbeitet, während UDP eine Paket-orientierte Übertragung anbietet. Häufig wird TCP in einem Atemzug mit IP genannt: man spricht von TCP/IP. Das deutet bereits darauf hin, dass TCP im Vergleich zu UDP die bessere Ergänzung zum Paket-orientierten IP darstellt. Die Unzuverlässigkeit des auf der Vermittlungsschicht eingesetzten IP wird durch die Verwendung von TCP vollständig verdeckt. Dadurch können die Anwendungen auf einen gesicherten Dienst zugreifen.

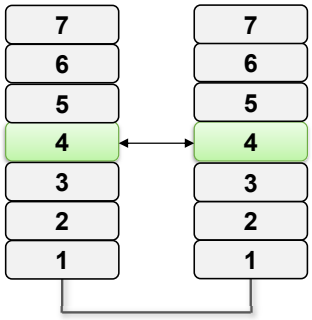
Die Dienste der Transportschicht können von den Anwendungsprotokollen genutzt werden, wobei diese in den meisten Fällen auf TCP aufsetzen. Beispiele für solche Anwendungsprotokolle sind FTP (File Transfer Protocol, Dateiübertragung), SSH (Secure Shell, Remote Terminal), SMTP (Simple Mail Transfer Protocol, E-Mail) oder HTTP (HyperText Transfer Protocol, Webseiten).

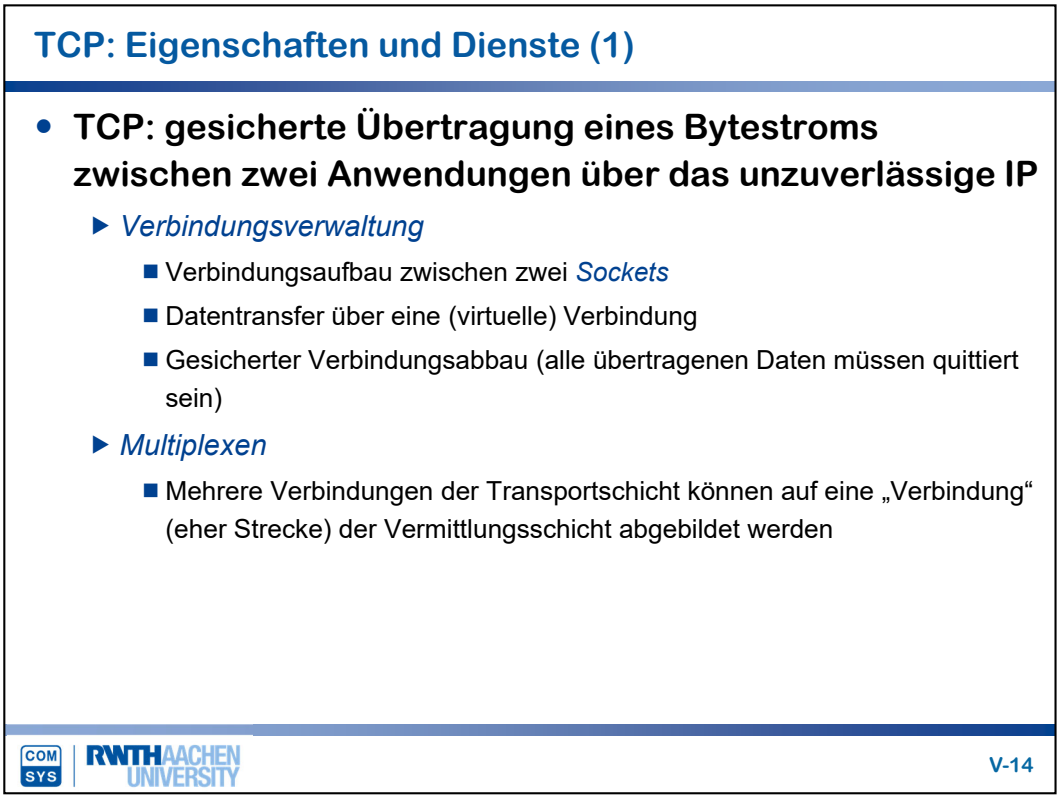
Es gibt aber auch Anwendungen, die UDP verwenden - vor allem, wenn der Zusatzaufwand der gesicherten Übertragung nicht benötigt wird, da nur kurze, nicht-zeitkritische Nachrichten versendet werden. Dazu zählen beispielsweise SNMP (Simple Network Management Protocol, Netzwerkmanagement) oder DNS (Domain Name System, Abbildung logischer Namen auf IP-Adressen). Es gibt sogar zeitkritische Anwendungen, die auf keinen Fall erneute Paketübertragungen wollen, weil die Neuübertragung den ganzen Datenstrom ausbremst und die neuübertragenen Daten bei Ankunft eventuell schon veraltet sind. Beispiele hierfür sind Audio- und Videoübertragungssysteme, die ihre Daten in Echtzeit übertragen (Live-Streaming, Videokonferenzen). Diese Anwendungen nehmen eher einen Paketverlust in Kauf (was kurzfristig zu einer schlechteren Ton- oder Bildqualität führt) als eine Paketwiederholung (d.h. Anhalten des Audio-/Videostroms und warten auf das fehlende Paket).

Neben TCP und UDP sind mehrere weitere Transportprotokolle definiert worden, von denen viele allerdings nie oder nur sehr rudimentär eingesetzt wurden. Neuere Entwicklungen, die noch weiter Verbreitung finden könnten, sind SCTP (Stream Control Transmission Protocol) und QUIC (Quick UDP Internet Connections); diese werden in dieser Vorlesung allerdings nicht weiter betrachtet. Gerade QUIC

stellt eine wichtige Entwicklung dar, da es verbindungsorientierte Kommunikation mit reduzierten Latenzen ermöglichen soll.

- **Protokollmechanismen der Transportschicht**
  - Prozessadressierung, strombasierte vs. paketbasierte Kommunikation, verbindungsorientiert/verbindungslos
- **Die Transportschicht im Internet**
  - TCP (Transmission Control Protocol)
    - Adressierung, Sockets
    - TCP-Verbindung
    - Flusskontrolle
    - TCP-Header
    - Staukontrolle (Congestion Control)
  - UDP (User Datagram Protocol)





Eine TCP-Verbindung dient zur Übertragung eines Bytestromes von Anwendung A zu Anwendung B. Es werden also einzelne Bytes ausgetauscht, keine Nachrichten.

TCP ist ein robustes Protokoll, welches sich dynamisch an die Eigenschaften und die aktuelle Verfügbarkeit des darunter liegenden Netzes anpassen kann. Das ist auch notwendig, weil es sich beim Internet um eine äußerst heterogene Topologie von unzähligen Netzwerken und verschiedenen Komponenten (Hosts, Router, Switches) handelt, die unterschiedlich leistungsfähig sind und unterschiedliche Eigenschaften besitzen.

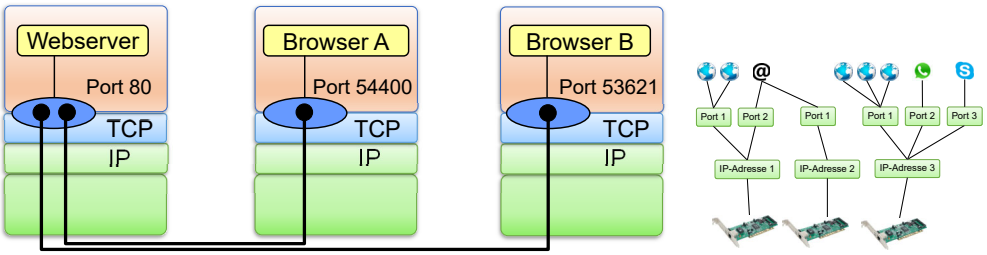
Auch bei TCP gibt es die klassischen Phasen bei der Verbindungsverwaltung:

Verbindungsaufbau, *Datentransfer* und *Verbindungsabbau*. Um der Zuverlässigkeit der Datenübertragung zu genügen, ist der Verbindungsabbau zusätzlich gesichert, d.h. es müssen sämtliche übertragenen Daten vom Empfänger quittiert sein, bevor ein Verbindungsabbau eingeleitet werden kann.

TCP setzt das Konzept des Multiplexens um, indem es Ports einführt. Ein Rechner kann über die (verbindungslose) IP-Schicht mehrere Ende-zu-Ende-Verbindungen zu beliebigen anderen Rechnern aufbauen. Diese werden anhand der Portnummern identifiziert.

TCP: Adressierung

- **Identifikation von Kommunikationsendpunkten geschieht über *Ports***
  - 16-Bit-Nummer
  - Portnummern bis 1023 für Standarddienste reserviert (z.B. 80 für HTTP, 25 für SMTP, 22 für SSH)
  - *Socket*: umfasst IP-Adresse eines Rechners und einen Port
    - Notation: (IP-Adresse:Portnummer) → internetweit eindeutig



Protokolle der Anwendungsschicht können die von TCP bzw. UDP bereitgestellten Dienste an *Ports* in Anspruch nehmen. Ports erlauben die Adressierung mehrerer Kommunikationsendpunkte im selben System. Falls zwischen zwei Rechnern gleichzeitig mehrere Verbindungen bestehen, können die Pakete anhand der Portnummern dem richtigen Kommunikationsfluss zugeordnet werden. Protokolle der Anwendungsschicht, über die weitverbreitete Dienste realisiert werden (wie beispielsweise FTP als Protokoll für die Übertragung von Dateien), haben eigene Ports reserviert. Dadurch ist es jedem möglich, einen Webserver (auf Port 80) anzubieten, ohne dafür sorgen zu müssen, dass alle Clients die verwendete Portnummer erfahren. Möchte man nicht, dass jeder den Dienst nutzen kann, kann man aber auch einen anderen Port wählen. Die Ports bis 1023 sind standardisiert und sind für bestimmte Dienste vorgesehen. Oft können Anwendungen diese Ports nur mit Root-Rechten verwenden. Die Ports von 1024 – 49151 sind registrierte Ports – sie werden auch üblicherweise für bestimmte Anwendungen verwendet, aber auch von Standardbenutzern verwendbar. Die höheren Porst können dynamisch verwendet werden.

Bitte beachten: Standardports werden nur auf der Serverseite verwendet, siehe Abbildung auf der Folie. Auf Clientseite wird kein Standardport ausgewählt – die TCP-Implementierung im Betriebssystem wird bei Initiierung eines Verbindungsaufbaus einfach einen freien Port außerhalb der reservierten Nummern wählen und von dieser Adresse aus einen Verbindungsaufbau vornehmen. Der standardisierte Port ist nur auf Seiten des Servers notwendig, der kontaktiert werden muss.

Es ist auch möglich, über TCP direkt zu kommunizieren, ohne ein Anwendungsprotokoll zu verwenden. Instanzen der Anwendungsschichtprotokolle werden einen sogenannten Socket erzeugen, um darüber zu kommunizieren. Diese Sockets sind eine Datenstruktur mit Buffer, die an eine IP-Adresse und einen Port gebunden werden und damit einen Kommunikationsendpunkt darstellen. Ein Socket kann eine TCP-Verbindung zu einem anderen Socket herstellen. Die Protokolle der Anwendungsschicht verwenden dann eine Variable des Datentyps, um Daten über den Socket zu versenden bzw. zu empfangen.

Implementiert man seine eigene Anwendung, benötigt man nicht notwendigerweise ein Anwendungsschichtprotokoll, man kann auch direkt auf Sockets zugreifen und über diese kommunizieren

Standardisierte Port-Nummern (well-known ports)

- **Viele Anwendungen nutzen TCP als Protokoll**
  - Allerdings muss der richtige Port gewählt werden, um auf der Gegenseite mit der richtigen Anwendung zu kommunizieren
    - 13: daytime
    - 20/21: FTP (File Transfer Protocol)
    - 25: SMTP (Simple Mail Transfer Protocol)
    - 53: DNS (Domain Name System)
    - 80: HTTP (HyperText Transfer Protocol)
    - …

```
> telnet relay.rwth-aachen.de 25
Trying 134.130.3.1...
Connected to sokrates .
Escape character is '^]'.
220 sokrates ESMTP Sendmail 8.8.5/8.8.5;
Mon, 4 Aug 2007 17:02:51 +0200
HELP
214-This is Sendmail version 8.8.5
214-Topics:
214-    HELO    EHLO    MAIL    RCPT    DATA
214-    RSET    NOOP    QUIT    HELP    VRFY
214-    EXPN    VERB    ETRN    DSN
214-For more info use "HELP <topic>".
...
214 End of HELP info
```

COM SYS | RWTH AACHEN UNIVERSITY V-16

HELO datkom.example.org

MAIL FROM:<klaus@comsys.rwth-aachen.de>

RCPT TO:<thissen@comsys.rwth-aachen.de>

DATA

Dies ist eine Testmail.

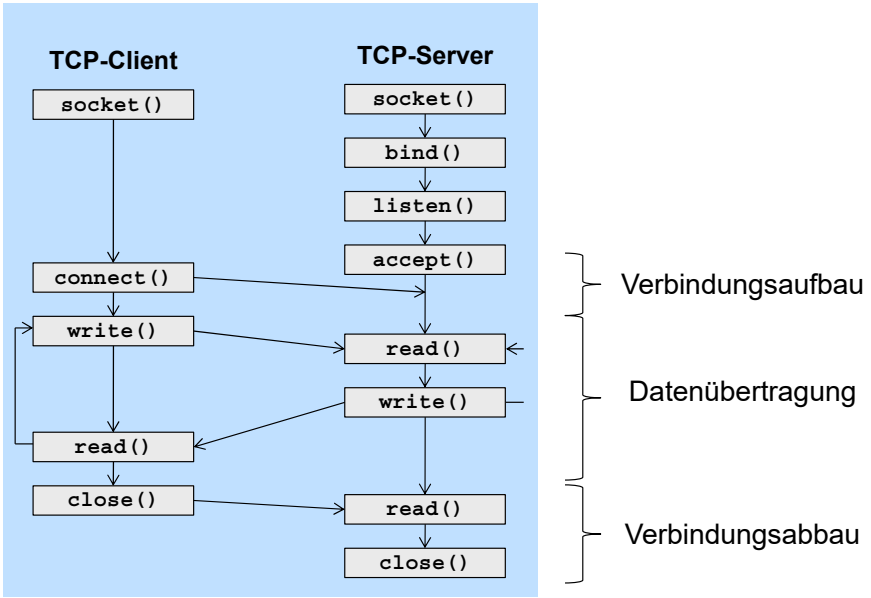
.

TCP: Verbindungsorientierung

- **Sockets können *aktiv* oder *passiv* erstellt werden**
  - Aktiver Modus: *Initiator* fordert auf einem Socket eine TCP-Verbindung an
    - Meist verwendet durch einen *Client*
  - Passiver Modus: *Responder* informiert TCP, dass er auf eine eingehende Verbindung wartet
    - Typischerweise verwendet durch einen *Server*
    - Spezifikation eines speziellen Sockets (Clients), von dem eine eingehende Verbindung erwartet wird (*fully specified passive open*)
    - Alle Verbindungen annehmen (*unspecified passive open*)
    - Geht ein Verbindungsaufbauwunsch ein, wird ein neuer Socket erzeugt, der als Verbindungsendpunkt dient

Wird ein Socket erstellt, kann er in den aktiven oder passiven Modus versetzt werden. Aktiv bedeutet, dass über den Socket eine Verbindung zu einem spezifizierten Kommunikationspartner (IP-Adresse und Port, also ein konkreter Socket) aufgebaut wird. Passiv heißt, dass die TCP-Instanz auf eingehende Verbindungsaufbauwünsche wartet. Bei üblichen Client/Server-Anwendungen verwendet der Client einen aktiven, der Server einen passiven Socket.

Ablauf bei der Kommunikation über TCP-Sockets



Bei der Programmierung von netzwerkfähigen Anwendungen mittels Sockets werden einem Programmierer Primitive bereitgestellt, die für Auf- und Abbau der Verbindung sowie für die Datenübertragung verwendet werden können und somit die Schnittstelle einer Anwendung zu TCP darstellen.

Socket-Primitive in TCP

- **Kommunikation via TCP durch einen Satz von Primitiven, die ein Anwendungsprogrammierer verwenden kann:**

Primitiv	Bedeutung
SOCKET	Erstellung eines neuen Netzzugangspunkts
BIND	Verknüpfe eine lokale Adresse mit dem Socket
LISTEN	Warte auf ankommende Verbindungswünsche
ACCEPT	Nimm einen Verbindungswunsch an
CONNECT	Initiierung eines Verbindungsaufbaus
WRITE (SEND)	Sende Daten über die Verbindung
READ (RECEIVE)	Empfange Daten auf der Verbindung
CLOSE	Freigabe der Verbindung

Je nach Programmiersprache sind die Primitive unterschiedlich benannt; eventuell werden sogar mehrere Primitive in einer einzigen Funktion zusammengefast (z.B. LISTEN und ACCEPT in Java).

Server

```
   int main()
   {
10    int listenfd, connfd;
      socklen_t len;
      struct sockaddr_in servaddr, cliaddr;
      char buffer[1024];
      time_t ticks;

15    listenfd = socket(AF_INET, SOCK_STREAM, 0);
      memset(&servaddr,0,sizeof(servaddr));
      servaddr.sin_family = AF_INET;
      servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
      servaddr.sin_port = htons(8013);
20    bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
      listen(listenfd,5);
      while(1){
         len = sizeof(cliaddr);
         connfd=accept(listenfd,(struct sockaddr*)&cliaddr, &len);
25       printf("connection from %s, port %d\n",
         inet_ntop(AF_INET,&cliaddr.sin_addr,buffer,sizeof(buffer)),
         ntohs(cliaddr.sin_port));
         ticks=time(NULL);
         snprintf(buffer,sizeof(buffer),"%.24s\r\n",ctime(&ticks));
30       write(connfd,buffer,strlen(buffer));
         close(connfd);
      }
      return 0;
   }
```


1-8: declarations of the used library functions

10-14: declaration of the used variables: File descriptors, `sock_addr` structures...

15: instantiation of a IPv4 (`AF_INET`) / TCP (`SOCK_STREAM`) socket

16-19: fill in the `sockaddr_in` structure: IP-address (`INADDR_ANY`) and port (`8013`), converting into network byte order

20: configuration of the sockets with `bind`(...). It will listen to port `8013` and accepts connections on all IP addresses (in accordance with `sockadd_in`)

21: switch socket to passive mode (`listen`)

23: tell `accept`, how large our `sockadd_in` structure is

24: `accept` waits for incoming connections and returns a file descriptor *`connfd`* of the connection

26-28: prints the connecting parameters

29-30: generate a string with data in `buffer` and sends the answer (`write`)

31: close the active connection and wait for the next one

Client

```
   int main(int argc, char **argv)
   {
10    int sockfd;
      struct sockaddr_in destaddr;
      char buffer[1024];

      if(argc!=3){
         printf("Syntax: simplex-client address port\n");
15       return 0;
      }
      sockfd = socket(AF_INET, SOCK_STREAM, 0);
      memset(&destaddr,0,sizeof(destaddr));
      destaddr.sin_family = AF_INET;
20    destaddr.sin_port = htons(atoi(argv[2]));
      inet_pton(AF_INET,argv[1],&destaddr.sin_addr);
      connect(sockfd,(struct sockaddr*)&destaddr,sizeof(destaddr));
      while(1){ /* endless loop */
         int r;
25       r = read(sockfd, buffer, 1024);
         if(r<=0)break; /* no data to read or connection closed */
            printf("%s",buffer);
         }
         close(sockfd);
30       return 0; /* never reached */
   }
```


1-7: declarations of the used library functions

10-12: declaration of the used variables:
File descriptors, `sock_addr` structures...

13-16: test command line parameters

17: initializing the socket

18-21: fill in the `sockaddr_in` structure: destination address and port with command line parameters, converting the port number into network byte order

22: initiate connection. Waits until connection works

23-31: infinite loop to read data, abort with `break`

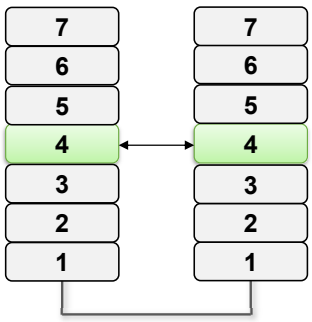
25: read data (maximum 1024 byte) into `buffer`

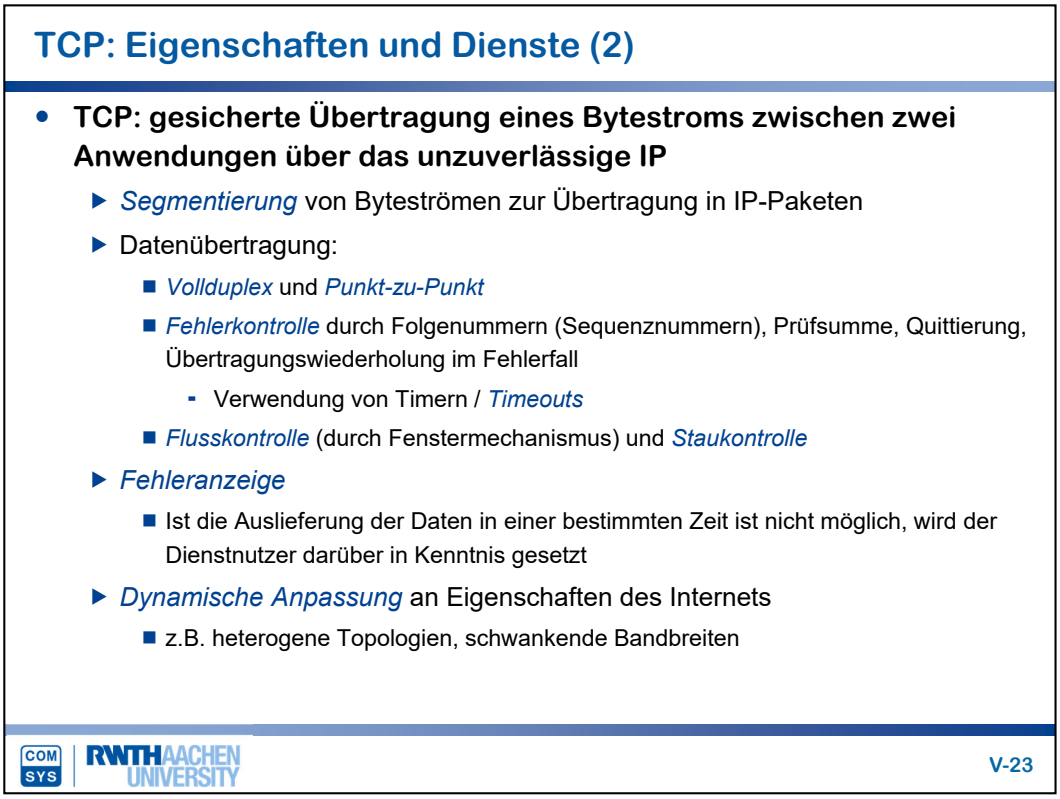
26: abort, if connection failed (`read`(...) returns `-1`)

27: print the received data

29: close socket

- **Protokollmechanismen der Transportschicht**
  - Prozessadressierung, strombasierte vs. paketbasierte Kommunikation, verbindungsorientiert/verbindungslos
- **Die Transportschicht im Internet**
  - TCP (Transmission Control Protocol)
    - Adressierung, Sockets
    - TCP-Verbindung
    - Flusskontrolle
    - TCP-Header
    - Staukontrolle (Congestion Control)
  - UDP (User Datagram Protocol)





Die Gewährung der Zuverlässigkeit erfordert u.a. eine reihenfolgetreue Auslieferung der einzelnen Pakete, welche durch IP auf Vermittlungsebene nicht gewährleistet ist. Dies wird bei TCP durch die Verwendung von Folgenummern erreicht. Jedes Paket wird mit einer Sequenznummer versehen, die im TCP-Header vermerkt ist. Der Empfänger hat somit die Möglichkeit, die ankommenden Pakete wieder in die richtige Reihenfolge zu bringen und kann gegebenenfalls Pakete erneut anfordern, falls diese unterwegs verloren gingen oder beschädigt wurden. Zur Erkennung einer Fehlersituation ist es oft notwendig, das Ausbleiben einer Quittung zu erkennen, wozu Timeouts eingesetzt werden.

Die Flusskontrolle basiert bei TCP auf einem Fenstermechanismus, wie er bereits in Kapitel 3 (Sicherungsschicht) ausführlich beschrieben wurde. Die Staukontrolle hat eine ähnliche Aufgabe wie die Flusskontrolle – die Flusskontrolle soll eine Überlastung des Empfängers vermeiden, die Staukontrolle eine Überlastung des Netzes. Dies ist keine einfache Aufgabe, da TCP nach Design unabhängig vom Netz ist. Auf die Fluss- und vor allem die Staukontrolle bei TCP wird im weiteren Verlauf dieses Kapitels noch detaillierter eingegangen, da sie zentrale Mechanismen des Transportprotokolls darstellen.

TCP: Verbindungsorientierung zur Zuverlässigkeit

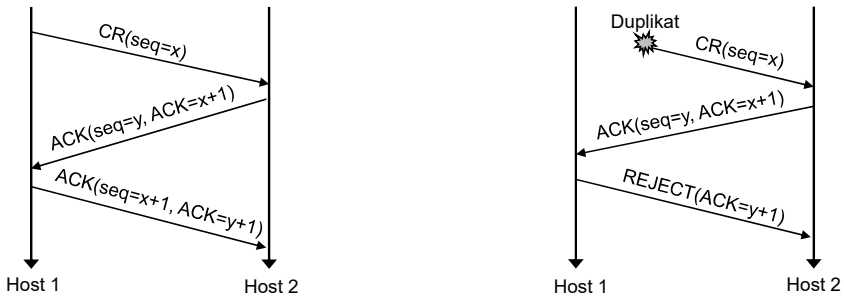
- **TCP zerlegt einen Bytestrom in *Segmente***
  - Datensegmente
  - Kontrollsegmente für Verbindungsaufbau, -verwaltung, -abbau
- **Verbindungsorientierung und Zuverlässigkeit bei TCP**
  - Verbindungsaufbau durch Kontrollsegmente, z.B. Reservierung von Speicher (Buffer) auf beiden Seiten
  - Zerteilung der Daten in Segmente, die jeweils mit einer Sequenznummern versehen werden (im TCP-Header)
  - Empfänger schickt Quittungen für korrekt empfangene Segmente (Wiederholung unquittierter Segmente, Reihenfolgewiederherstellung)
  - Verbindungsabbau durch Kontrollsegmente, z.B. Freigabe von Buffer

→ ***virtuelle Verbindung über verbindungsloses IP***

TCP nennt sich zwar ein verbindungsorientiertes Protokoll, was aber im Gegensatz zu den tieferen Schichten nicht bedeutet, dass ein fester Pfad durch das Netz aufgebaut wird. Bei einer Verbindungsorientierung auf Schicht 3 würde genau dies geschehen – der beste Weg für Dateneinheiten über ein komplexes Netz hinweg würde ermittelt und alle Dateneinheiten entlang der gleichen Route geschickt. Im Internet ist diese Schicht allerdings verbindungslos (IP). TCP baut auf IP auf und muss dessen Dienste nutzen, so dass keine feste Verbindung erzwungen werden kann. Stattdessen wird der nächsthöheren Schicht vorgegaukelt, dass TCP verbindungsorientiert ist, indem es die Probleme von IP verbirgt (Paketverlust, Reihenfolgeumstellung) und den Anschein erweckt, die Daten wären verbindungsorientiert und zuverlässig übertragen worden.

Verbindungsaufbau

- *Three-Way-Handshake*
  - Netz kann Pakete verlieren, verspätet ausliefern, duplizieren
  - Abhilfe: Sequenznummern und dreiteiliger Handshake



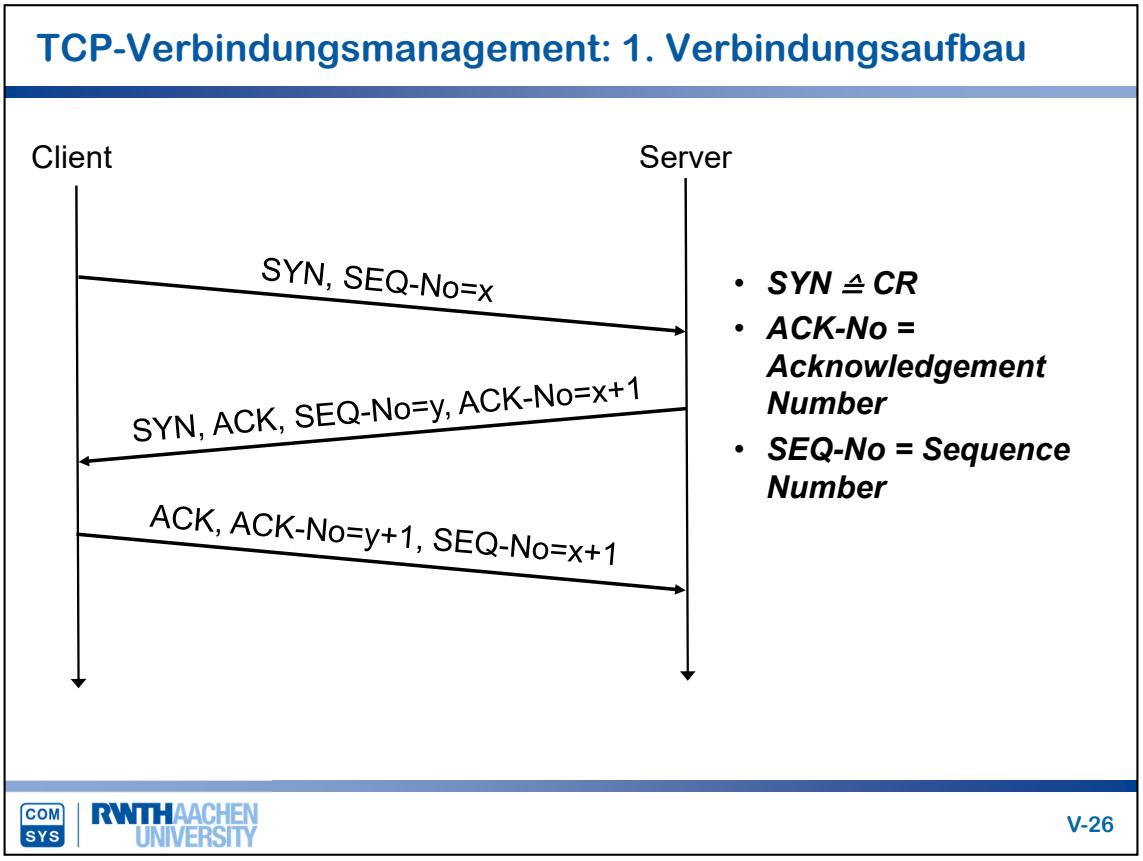
- **Kompromiss zwischen Komplexität und garantierter Einigung auf eine Verbindung**

Zweck des Verbindungsaufbaus ist es, die Kommunikationsbereitschaft der Gegenstelle zu testen und initiale Parameter auszuhandeln (z.B. Buffergrößen, Sequenznummern).

Alle Pakete können bei Verwendung von IP verloren gehen, daher ist nicht gewährleistet, dass ein Verbindungsaufbauwunsch (Connection Request, CR) beim Kommunikationspartner ankommt. Eine Quittierung der Verbindungsaufbaunachricht (Acknowledgement, ACK) ist also nicht nur notwendig, um dem Initiator der Verbindung die Kommunikationsbereitschaft zu signalisieren und Parameter mitzuteilen, sondern auch, damit der Initiator weiß, dass sein CR nicht verloren gegangen ist. Aber auch diese Quittung könnte verloren gehen, so dass der Empfänger in Bereitschaft ist, aber nie Daten folgen. Daher quittiert auch der Initiator den Erhalt der Quittung noch einmal. Zwar kann auch diese Quittung verloren gehen, so dass ihr Erhalt auch wieder bestätigt werden müsste, aber dieses Spielchen ließe sich endlos fortsetzen. Bei Verlust der letzten Quittung wird dies als nicht tragisch angesehen, da der Initiator im Normalfall auch direkt mit der Datenübertragungsphase beginnen wird, durch die implizit die verlorene Quittung ersetzt wird.

TCP arbeitet hier durchgängig mit Timern – trifft eine Quittung nicht innerhalb einer bestimmten Zeit ein, geht TCP von einem Datenverlust aus und wiederholt das Segment.

Während dieses Handshakes werden bereits Folgenummern verwendet, um gegen Fehler bei der Übertragung zu schützen. Es könnte z.B. sein, dass ein Duplikat der CR-Nachricht beim Empfänger eintrifft, wodurch dieser annehmen würde, dass ein neuer Verbindungsaufbauwunsch vorliegt. Trifft die zugehörige Quittung beim Initiator ein, kann er identifizieren, dass sich diese auf einen nicht mehr aktuellen Verbindungsaufbauwunsch bezieht, und wird eine negative Quittung zurückschicken, damit der Empfänger nicht unnötig Buffer für diese Verbindung reserviert.

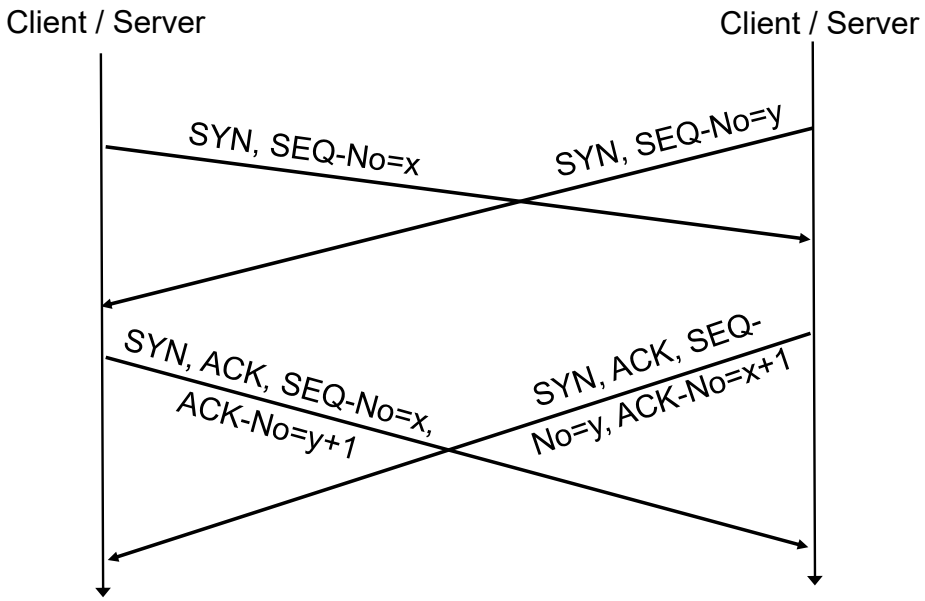


Wird ein Socket im aktiven Modus erstellt, wird ein Connection Request an den adressierten passiven Socket gesendet und nach dem Three-Way-Handshake quittiert. Dass es sich bei den ausgetauschten Segmenten um Nachrichten eines Verbindungsaufbaus handelt, wird durch Setzen eines bestimmten Flags im TCP-Header kenntlich gemacht. TCP definiert nur eine PDU-Struktur und erlaubt durch eine Menge von Flags, kenntlich zu machen, welchem Zweck das aktuelle Segment dient (Verbindungsaufbau (SYN), Daten (kein gesetztes Flag), Quittung (ACK), Verbindungsabbau (FIN), Zurücksetzen der Verbindung (RST), …). Im Connection-Request-Segment sendet der Client eine initiale Sequenznummer (x). Ebenso sendet der Server in seiner Quittung eine Sequenznummer (y) mit zurück. Hierbei werden das ACK-Flag und das SYN-Flag gesetzt: die Nachricht ist gleichzeitig eine Quittung über den Erhalt der CR-Nachricht (ACK-Flag – hierdurch wird signalisiert, dass die Nachricht eine Quittungsnummer enthält) als auch eine Annahme des Verbindungsaufbauwunsches (SYN-Flag). Die letzte Nachricht des Clients schließt den Three-Way-Handshake ab, die Verbindung ist aufgebaut.

Die Sequenznummern x und y werden auf beiden Seiten zufällig ausgewählt (sie müssen nicht bei 0 beginnen). Der Hauptzweck der Sequenznummern ist – wie auf Schicht 2 – die Einführung eines Quittungsmechanismus und einer Flusskontrolle. Während des Verbindungsaufbaus teilen sich die beiden Kommunikationspartner dadurch mit, mit welchen Sequenznummern sie im Folgenden beginnen werden, die auszutauschenden Daten durchzunummerieren.

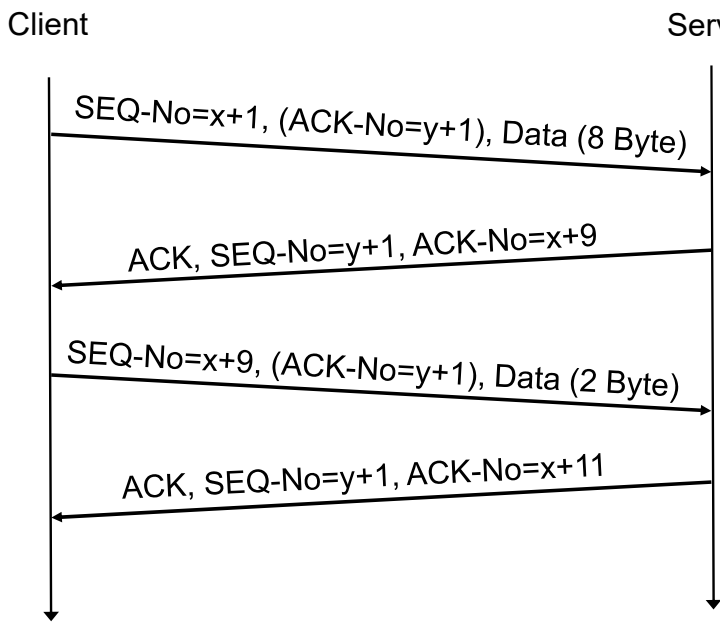
Gleichzeitig wird beim Verbindungsaufbau auf beiden Seiten Bufferspeicher reserviert, in dem empfangene Daten zwischengespeichert werden können, bis sie in richtiger Reihenfolge an die Anwendung weitergereicht werden können.

Alternativ: ungeregelter Verbindungsaufbau



Gibt es keine festgelegten Client- und Serverrollen, kann es eventuell passieren, dass beide Seiten simultan versuchen, eine Verbindung mit dem jeweils Anderen aufzubauen. Dies ist überflüssig, da TCP-Verbindungen vollduplex sind – und darüber hinaus kann zwischen zwei Kommunikationsendpunkten sowieso nur genau eine Verbindung existieren, da jede Verbindung eindeutig durch IP-Adressen und Ports der beiden Sockets identifiziert wird. Daher wird in diesem Fall die hier dargestellte Variante des Verbindungsaufbaus verwendet: jede Seite hat bereits eine SYN-Nachricht des Kommunikationspartners erhalten und sendet auf diese eine Quittung zurück. Danach ist die Verbindung aufgebaut.

TCP-Verbindungsmanagement: 2. Datenübertragung



- **Übliche Segmentgröße: 1460 Byte – vermeiden von IP-Fragmentierung**
- **Verwendung von Timern (Timeouts) zum Schutz gegen Paketverlust**

Nach dem Verbindungsaufbau besteht kein Unterschied mehr zwischen den Kommunikationspartnern. Die Verbindung ist bidirektional, Daten und Quittungen können in beide Richtungen übertragen werden. Im Beispiel auf der Folie sendet nur der Client Daten, der Server sendet Quittungen zurück. Der Client sendet einen Bytestrom, den er segmentiert – eine übliche Größe für den Payload, den ein Segment enthält, ist 1460 Byte – zuzüglich der Standard-Header von TCP und IP (je 20 Byte) entsteht so eine SDU von 1500 Byte Größe auf der Sicherungsschicht, was genau die Größe ist, die maximal in einem Ethernet-Rahmen übertragen werden kann, so dass IP-Fragmentierung in üblichen LANs vermieden wird. In der Praxis sind also die Schichten doch nicht völlig voneinander getrennt sondern so konfiguriert, dass sie optimal zusammenspielen.

Jedes Datensegment wird mit einer Sequenznummer versehen. Diese bezeichnet – basierend auf der beim Verbindungsaufbau ausgehandelten Sequenznummer (x+1) – die Position des ersten im Segment enthaltenen Bytes innerhalb des gesamten Bytestroms.

Die Quittungsnummern des Empfängers haben die gleiche Bedeutung wie bei der Flusskontrolle, die auf Schicht 2 behandelt wurde: die Quittungsnummer entspricht demjenigen Byte, welches als nächstes erwartet wird, alle vorherigen Bytes wurden korrekt empfangen. Der Unterschied zur Implementierung auf Schicht 2 ist also die Einheit der Nummerierung – Rahmen vs. Bytes, so dass bei TCP die Sequenznummern zweier aufeinanderfolgender Segmente im Normalfall keine aufeinanderfolgenden Nummern sein werden.

Da durch IP Pakete verloren gehen oder in falscher Reihenfolge ausgeliefert werden können, wird nur eine positive Quittung zurückgeschickt, falls das nächste empfangene Segment auch die Sequenznummer des nächsten erwarteten Bytes enthält. Der Sender startet beim Senden jedes Segments einen Timer; bei einem Timeout wird von einem Paketverlust ausgegangen und das Segment erneut übertragen.

Die in Klammern gesetzten Quittungsnummern bei den Segmenten vom Client an den Server sollen andeuten, dass es sich um keine notwendige Angabe handelt. Da keine Daten in Serverrichtung quittiert werden müssen, braucht der Client das ACK-Flag nicht zu setzen und daher auch keinen gültigen Wert als ACK-No einzutragen. In der Praxis werden diese Angaben aber trotzdem gemacht – dadurch werden schlicht und einfach vorherige Quittungen wiederholt, was dem Ablauf von TCP nichts schadet, aber gegen negative Auswirkungen des Verlusts vorheriger Quittungen schützt. Beispielweise könnte die dritte Nachricht des Three-Way-Handshakes von Folie 27 verlorengehen, so dass der Server irgendwann einen Timeout hat und seine Quittung wiederholt. Setzt der Client hier auf dieser Folie bei jedem seiner Segmente die Quittungsnummer (und das zugehörige Flag), wiederholt er die verlorene Quittung mit jedem Datensegment, so dass der Timer des Server stoppen würde und die Neuübertragung seiner vorherigen Quittung nicht vorgenommen wird.

Bitte nicht täuschen lassen: diese Folie scheint den Eindruck zu erwecken, dass die Fehlerbehebungsstrategie von TCP Stop-and-Wait ist – allerdings ist hier nur zur Vereinfachung diese Darstellung gewählt worden. Auch bei TCP können Strategien wie Go-Back-N und Selective Repeat eingesetzt werden.

TCP: Neuübertragung von Segmenten

- **Mehrere Verfahren zur Reaktion auf Fehler (fehlende Segmente, Segmente in falscher Reihenfolge)**
  - Ursprüngliches TCP: *Go-Back-N*
    - Quittungen werden nur für Segmente mit erwarteter Nummer gesendet
    - Quittungsnummer *n* bestätigt Empfang aller Bytes bis *n*-1
    - Nicht erwartete Segmente werden verworfen
    - Übertragungswiederholung aufgrund Ablauf von Retransmission Timer des Senders
  - Oder Aushandlung bei Verbindungsaufbau:
    - *Selective-Repeat* (RFC 1106)
      - Durch NAK kann fehlerhaftes Segment explizit angefordert werden
    - *Selective Acknowledge (SACK)* (RFC 2018)
      - Erlaubt Listen von positiven und negativen Quittungen für mehrere Segmente

Im Gegensatz zu Selective Repeat, das nie zum vollwertigen Internet-Standard wurde, gilt Selective Acknowledge heutzutage als Standard.

Reaktion des Empfängers

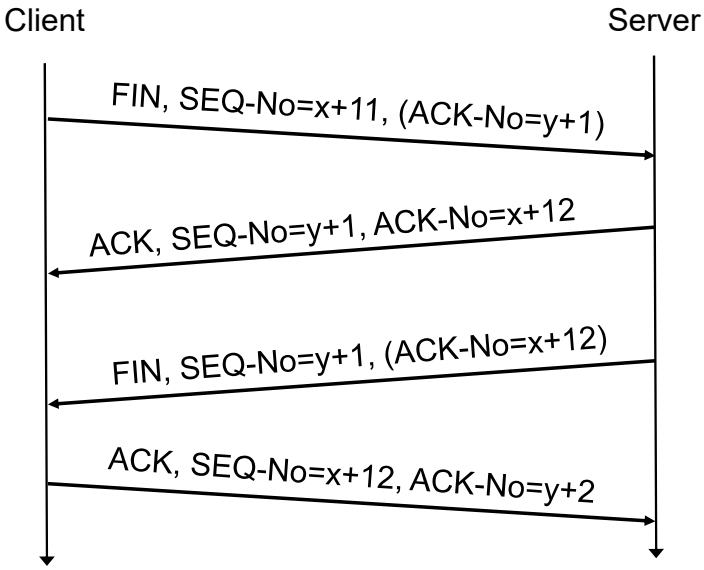
Ereignis beim Empfänger	Aktion beim Empfänger
Ankunft eines Segments mit der erwarteten Sequenznummer. Alle Daten bis dahin sind schon bestätigt worden.	Delayed ACK. Warte bis zu 500ms auf das nächste Segment. Wenn dieses nicht empfangen wird, verschicke die Bestätigung.
Ankunft eines Segments mit der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

TCP kann Quittungen auch absichtlich verzögern, um nicht zu viel Overhead durch Quittungsnachrichten zu erzeugen. Es wird nicht auf jedes korrekte Segment mit einem ACK reagiert, sondern nur auf jedes zweite (oder noch seltener, je nach Konfiguration). Hierbei muss man allerdings drauf achten, keinen versehentlichen Timeout zu erzeugen.

Nur in Fehlersituationen oder bei Behebung einer Fehlersituation erfolgt eine sofortige Reaktion.

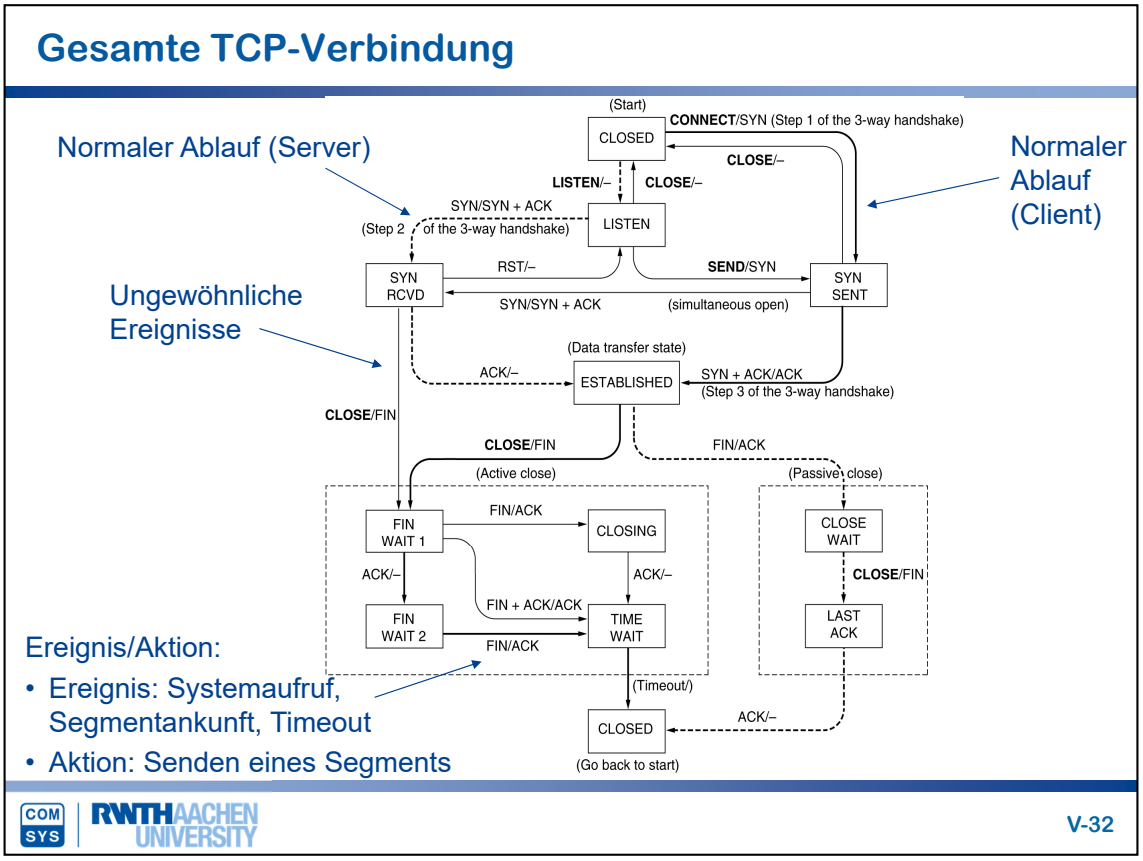
DUP-ACKs werden im Zusammenhang mit der Staukontrolle benötigt, siehe weiter unten.

TCP-Verbindungsmanagement: 3. Verbindungsende



- **Abbau von zwei Simplex-Verbindungen**

Wie der Verbindungsaufbau, wird auch der Verbindungsabbau durch das Setzen eines Flags (FIN) markiert. Der Abbau der beiden Übertragungsrichtungen kann hierbei unabhängig voneinander erfolgen – im obigen Beispiel kann der Client nach Austausch der ersten beiden Nachrichten keine Daten mehr an den Server versenden, der Server könnte aber noch – falls nötig – Daten an den Client senden. Wiederum werden Timer und Neuübertragung zum Schutz gegen Paketverlust verwendet.



Anhand von TCP kann man gut einen Bogen zurück zum ersten Kapitel schlagen – die Instanzen, die den Dienst der zuverlässigen, verbindungsorientierten Datenübertragung von TCP erbringen, implementieren einen Automaten, anhand dessen der Protokollablauf modelliert ist. Hierbei gibt es einen einzigen Automaten, der das Verhalten einer TCP-Instanz beschreibt; je nachdem, ob eine Instanz sich aktiv oder passiv verhält, werden unterschiedliche Übergänge des Automaten verwendet. Während wir in Kapitel 1 Dienstprimitive verwendet haben, um die Übergänge zu beschriften, wird hier nur dargestellt, welche Ereignisse eintreten, nicht welche Primitive dazu genau verwendet werden.

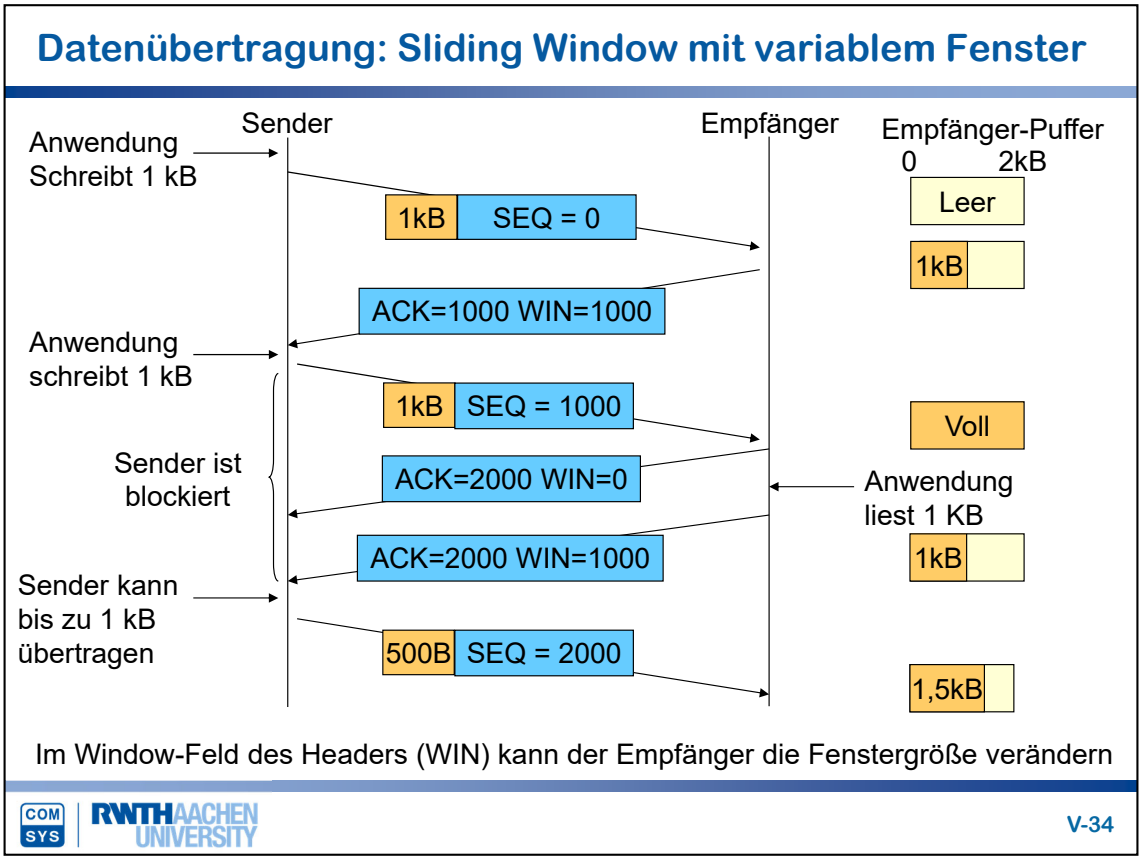
(Automaten werden auch zur Modellierung der anderen behandelten Protokolle verwendet – allerdings wurde bis hier gewartet, um noch einmal Bezug darauf zu nehmen, da TCP über die größte Funktionalität verfügt und daher den interessantesten Automaten hat. Bitte beachten: hinter dem Zustand „Established“ verbirgt sich ein komplexer Teilautomat, der die im Folgenden noch dargestellten Funktionalitäten der Datenübertragungsphase spezifiziert.)

Etwas seltsam erscheinen mag der TIME WAIT-Zustand – die Verbindung ist abgebaut und damit sind auch alle Daten ausgetauscht. Hier findet allerdings gewollt eine verzögerte Freigabe des Sockets statt um zu vermeiden, dass die gleiche IP-Adressen/Port-Kombination direkt von einer neuen Anwendung verwendet werden kann. Die Idee dahinter ist zu vermeiden, dass eventuell verspätet noch Paket-Duplikate der alten Verbindung eintreffen, die als Elemente der neuen Verbindung betrachtet werden und im schlimmsten Fall zu einem Rücksetzen der neuen Verbindung führen.

(Für die Praxis heißt dies: nicht wundern, wenn beim Testen von TCP-basierten Anwendungen nach Beendigung einer Verbindung nicht direkt eine neue Verbindung aufgebaut werden kann. Einfach etwas warten…)

Zustände der TCP-Verbindung

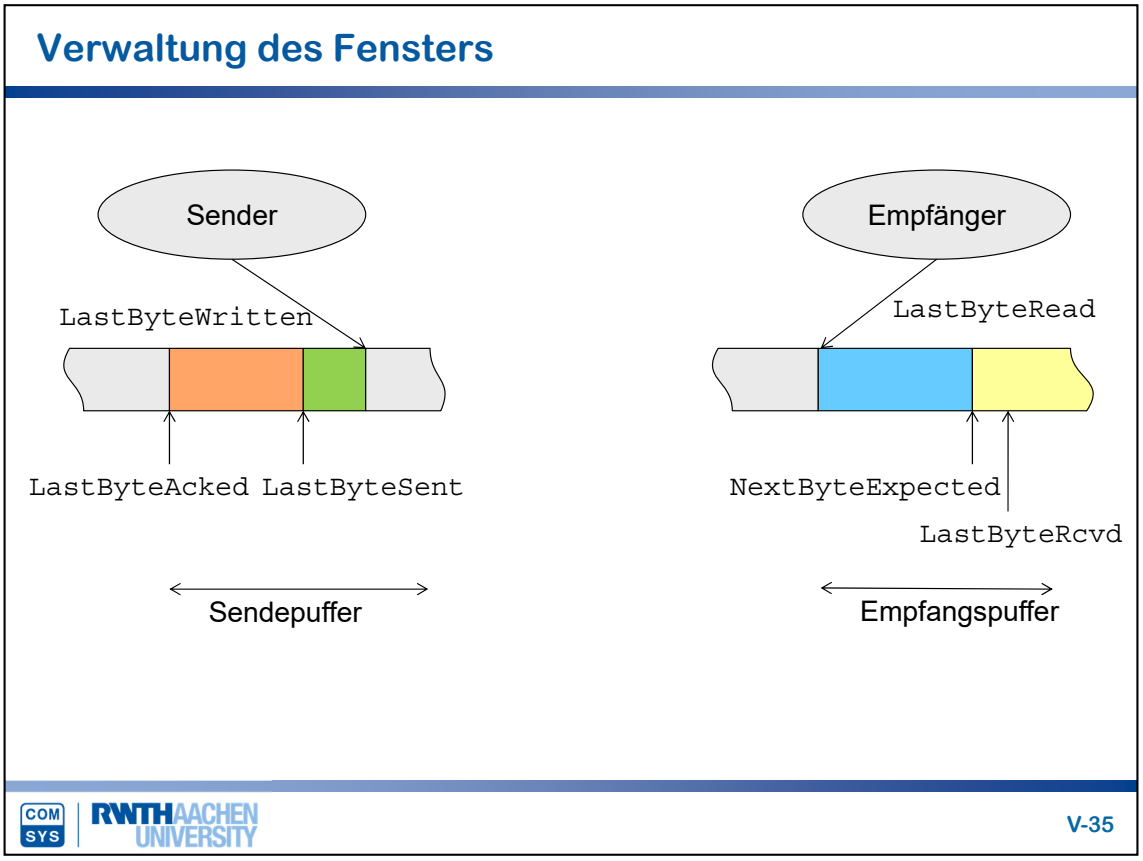
State	Description
CLOSED	Keine aktive Kommunikation
LISTEN	Passiver Modus – warten auf Connection Requests
SYN RCVD	Warten auf die dritte Nachricht des Verbindungsaufbaus
SYN SENT	Warten auf die zweite Nachricht des Verbindungsaufbaus
ESTABLISHED	Verbindung aufgebaut, Datenaustausch
FIN WAIT 1	Anwendung hat Verbindungsabbau gestartet
FIN WAIT 2	Einseitiger Verbindungsabbau quittiert
TIME WAIT	Warten auf Duplikate
CLOSING	Simultaner Verbindungsabbau; warte auf Quittung der anderen Seite
CLOSE WAIT	Verbindung einseitig abgebaut
LAST ACK	Warten auf Quittung für den Abbau der zweiten Richtung



Während der Phase der Datenübertragung wird wie auf Schicht 2 das Sliding-Window-Verfahren zur Flusskontrolle verwendet. Es gibt allerdings einen wesentlichen Unterschied zu dem Verfahren auf Schicht 2: TCP muss mehrere Verbindungen simultan verwalten können. Der insgesamt zur Verfügung stehende Bufferspeicher muss zwischen allen Verbindungen aufgeteilt werden. Darüber hinaus werden Daten auf Schicht 2 schnellstmöglich an Schicht 3 (IP) weitergegeben, um dort verarbeitet zu werden. Bei TCP verbleiben die Daten im Buffer, bis die Anwendung (bzw. das Protokoll der Anwendungsschicht) die Daten aus dem Socket ausliest. Dies heißt insgesamt: eventuell muss der maximal zur Verfügung stehende Bufferspeicher einer TCP-Verbindung im Laufe der Zeit an die Zahl der Verbindungen angepasst werden, und es hängt von der Lesegeschwindigkeit und –frequenz der Anwendung ab, wie stark sich die Größe des freien Bufferspeichers im Laufe der Zeit verändert. Der Sender muss also permanent über die Größe des aktuell zur Verfügung stehenden freien Bufferspeichers informiert werden.

Während des Verbindungsaufbaus teilt der Empfänger dem Sender die Größe des zur Verfügung stehenden Bufferspeichers mit, der exklusiv für diese Verbindung reserviert wurde. Den entsprechenden Wert nimmt der Sender als Fenstergröße des Sliding-Window-Verfahrens. In jeder Quittung teilt der Empfänger dem Sender mit, wieviel Platz aktuell noch vorhanden ist, um weitere Daten zu empfangen. Im obigen Beispiel dargestellt ist die Situation, dass der Buffer auf Empfängerseite voll läuft; in dem Fall kann sogar eine neue Fenstergröße von Null vorgegeben werden, der Sender wird gebremst. (Analog zum Beispiel der HALT-/WEITER-Kommandos im Foliensatz zu Schicht 2.)

Es besteht allerdings die Gefahr, dass die spätere Meldung des Empfängers, dass wieder Speicher vorhanden ist, verloren geht – um zu vermeiden, dass der Empfänger auf neue Daten wartet, während der Sender auf eine (verloren gegangene) Sendefreigabe wartet, wird wieder ein Timer eingesetzt (Persistence Timer). Dieser wird beim Sender gestartet, sobald ein Segment eingeht, das eine Fenstergröße von Null vorgibt. Nach Ablauf des Timers wird der Sender das nächste Segment abschicken und auf die Antwort des Empfängers warten, in der wieder die aktuelle Fenstergröße enthalten ist.



Sockets stellen eine Datenstruktur dar, die über einen Bufferspeicher verfügt. Dieser wird in Sende- und Empfangsbuffer unterteilt. Eine Anwendung schreibt Daten in den Sendebuffer und liest Daten aus dem Empfangsbuffer. Daten verbleiben im Sendebuffer, bis sie erfolgreich übertragen wurden, d.h. bis sie von der empfangenden TCP-Instanz quittiert und im dortigen Empfangsbuffer abgespeichdert wurde (bis die empfangende Anwendung sie entnimmt).

Zur Verwaltung des Bufferspeichers werden Zeiger eingesetzt, die auf die zur Verwaltung wichtigen Bufferpositionen zeigen und nach Ankunft eines Datensegments bzw. einer Quittung sowie bei einer Schreib-/Leseoperation einer Anwendung verrückt werden:

Senderseite

- LastByteWritten zeigt auf Position des Sendebuffers, bis zu der die Anwendung Daten geschrieben hat.
- LastByteSent hält mit, bis zu welcher Position im Bytestrom bereits Daten versendet wurden, LastByteAcked hält mit, bis zu welcher Position der Bytestrom bereits quittiert wurde. Alle vorherigen Bytes wurden erfolgreich zugestellt und aus dem Sendebuffer gelöscht. Dieser Bufferspeicher kann also von der Anwendung neu beschrieben werden (Ringbuffer).
- Die Daten zwischen LastByteAcked und LastByteWritten sind bereits gesendet, aber noch nicht quittiert. Die Menge dieser unquittierten Daten darf die Buffergröße des Empfängers nicht überschreiten.

Empfängerseite

- NextByteExpected gibt an, bis zu welcher Position der Bytestrom bereits komplett empfangen wurde.

- LastByteRead gibt an, bis zu welcher Position die empfangende Anwendung die Daten bereits ausgelesen hat.
- Die Menge der Daten zwischen LastByteRead und NextByteExpected belegt noch den Empfangsbuffer, die Differenz zur Gesamtgröße des Buffers entspricht dem noch freien Fenster.
- Falls Go-Back-N zur Fehlerbehebung verwendet wird, werden Segemente, deren Sequenznummer nicht “NextByteExpected” entspricht, verworfen. Es ist keine weitere Verwaltung notwendig. Wird stattdessen Selective Repeat eingesetzt, werden auch Segmente außerhalb der erwarteten Reihenfolge gespeichert und es sind zwei weitere Zeiger notwendig (hier nicht dargestellt), um Anfang und Ende solch eines Teilbytestroms festzuhalten. Eventuell sind sogar mehrere solche Zeigerpaare notwendig, falls vereinzelte Segmente empfangen werden.
- Mittels LastByteRcvd kann festgehalten werden, bis zu welcher Position des Bytestroms bereits Daten empfangen wurden (Ende des letzten Segments, welches außerhalb der erwarteten Reihenfolge empfangen wurde). Dies ist nur dann notwendig, wenn das “Selective ACK (SACK)”-Verfahren benutzt wird.

Verwaltung des Fensters

- **Verwendung der Zeiger bei der Flusskontrolle:**
  - Senderseite:
    - Größe des Senderbuffers: `MaxSendBuffer`
    - `LastByteSent - LastByteAcked ≤ AdvertisedWindow`
    - `EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)`
    - `LastByteWritten - LastByteAcked ≤ MaxSendBuffer`
    - Blockiere den Versuch der Anwendung, `y` Byte zu schreiben, falls `(LastByteWritten - LastByteAcked) + y > MaxSenderBuffer`
  - Empfängerseite:
    - Größe des Empfängerbuffers: `MaxRcvBuffer`
    - `LastByteRcvd - LastByteRead ≤ MaxRcvBuffer`
    - `AdvertisedWindow` =
      - `MaxRcvBuffer - ((NextByteExpected - 1) - LastByteRead)` [bei Go-Back-N, Selective Repeat]
      - `MaxRcvBuffer - (LastByteRcvd - LastByteRead)` [bei SACK]

AdvertisedWindow ist die Angabe, die im WIN-Feld jeder Quittung an den Sender übertragen wird.

EffectiveWindow ist der dem Sender noch zur Verfügung stehende freie Teil des Fensters; maximal so viele Byte dürfen noch übertragen werden, ohne dass eine weitere Quittung eingeht.

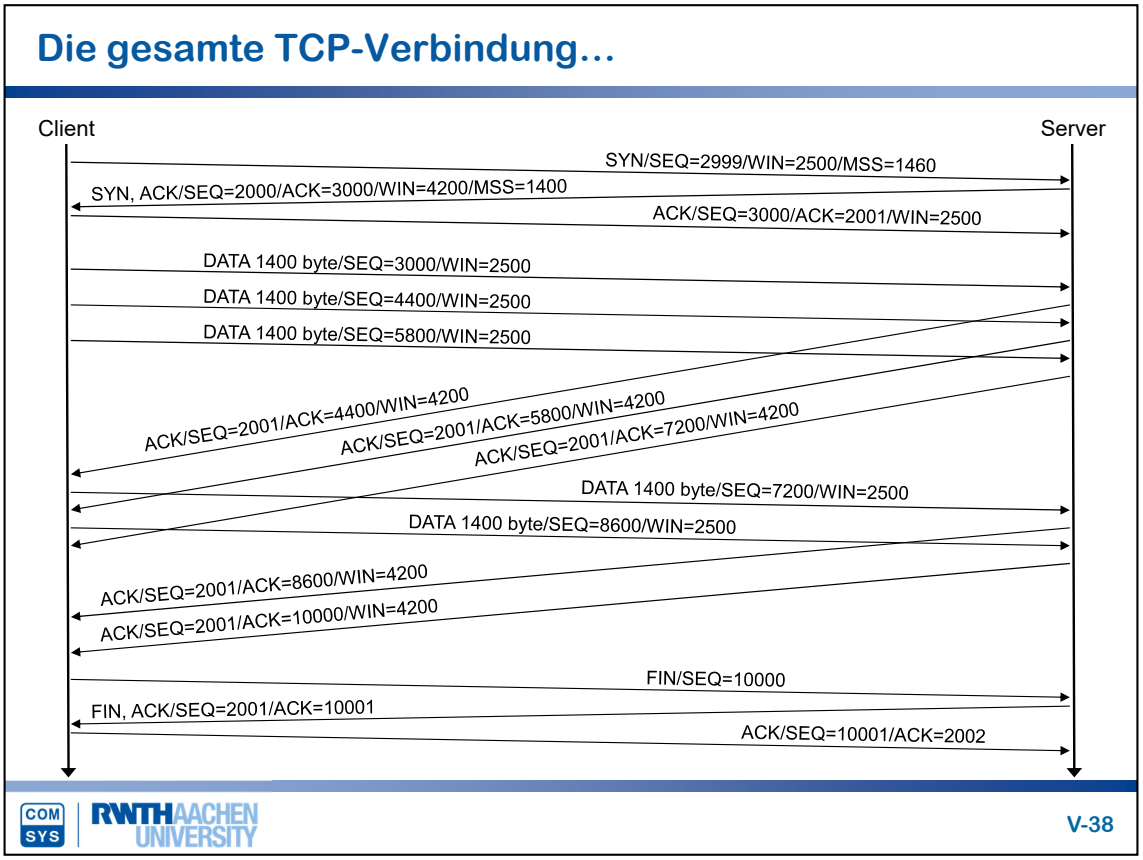
Silly-Window-Syndrom

- *Silly Window*
  - Sender wird durch ACK mit WIN=0 gestoppt
  - Empfänger liest 1 Byte aus Buffer
    - Wiederholung des vorherigen ACKs mit WIN=1
  - Sender schickt 1 Byte (plus 20 Byte TCP-Header + 20 Byte IP-Header)
    - ACK mit WIN=0
  - …
  - Unnötige Netzbelastung!
- **Lösungen**
  - *Nagle's Algorithmus*: Sender wartet, bis Segment maximaler Größe gesendet werden kann
  - *Clark's Solution*: Empfänger verzögert ACK, bis Buffer halb leer ist oder Platz für ein maximales Segment ist

Bei stupider Anwendung des Fenstermechanismus' kann es zum Silly-Window-Syndrom kommen: der Empfängerbuffer wird komplett gefüllt, da die empfangende Anwendung keine Daten entnimmt. Laut Protokoll wird der Sender durch ein ACK mit WIN=0 gestoppt. Entnimmt die Anwendung nur langsam Daten, kann es nun sein, dass nur wenige Byte aus dem Buffer gelesen werden. Der Sender wird entsprechend entsperrt und kann ein sehr kleines Segment schicken, so dass durch die TCP- und IP-Header sehr viel Overhead erzeugt wird. Danach ist der Empfängerbuffer wieder voll. Wiederholt sich dieses Spiel, da die Anwendung kurz hintereinander immer wieder kleine Mengen an Daten entnimmt, wird die Übertragung ineffizient.

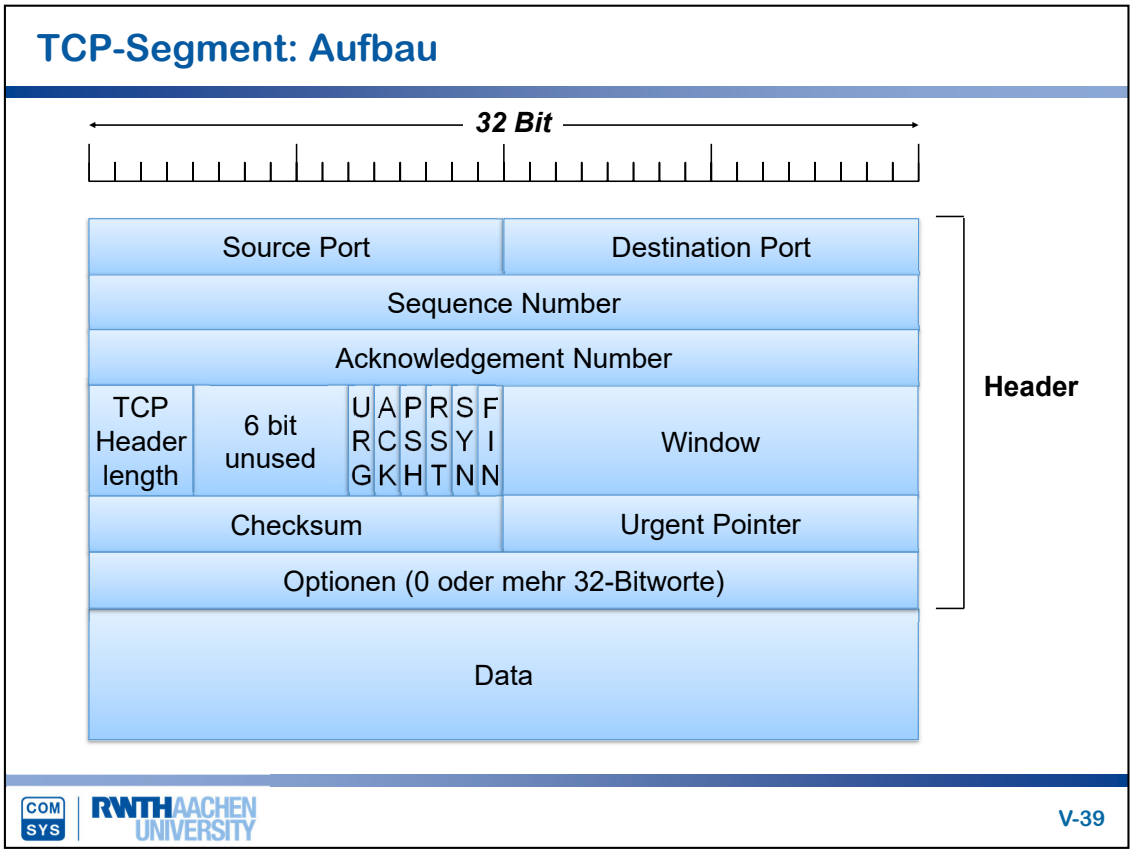
Für Abhilfe sorgen z.B. die Lösungen von Nagle (Senderseite) und Clarke (Empfängerseite). Nagle's Algorithmus hat sich durchgesetzt und soll laut Standard von jeder TCP-Implementierung per default aktiviert sein.

Bei Nagle's Algorithmus versendet TCP die von einer Anwendung übergebenen Daten nur dann sofort, wenn momentan keine Daten unbestätigt unterwegs sind oder wenn ein Segment maximaler Größe (oft 1460 Byte) versendet werden kann. Liegen weniger Daten vor und es stehen noch Bestätigungen des Empfängers aus, verzögert TCP die Versendung. Dadurch können Daten eventuell sehr stark verzögert werden. Implementiert man eine Anwendung, die geringe Mengen an Daten sofort versenden soll, muss man darauf achten, den Buffer zu flushen, also TCP mitzuteilen, Nagle's Algorithmus zu ignorieren und trotzdem ein kleines Segment zu senden.



Beispiel für eine fehlerfreie TCP-Übertragung, bei der die Daten auf Empfängerseite immer direkt verarbeitet werden und das Fenster damit immer die maximale Größe hat.

Mit enthalten in den ersten beiden Nachrichten des Verbindungsaufbaus ist hier auch der Parameter MSS (Maximum Segment Size) – die beiden Kommunikationspartner einigen sich hier auf die maximale Größe des Payloads eines Segments während der Datenaustauschphase. Die MSS wird auf jeder Seite so gewählt, dass die resultierenden IP-Pakete nie die Größe der lokal verwendeten MTU überschreiten, um IP-Fragmentierung zu vermeiden. Das Minimum der beiden vorgeschlagenen Werte wird verwendet.

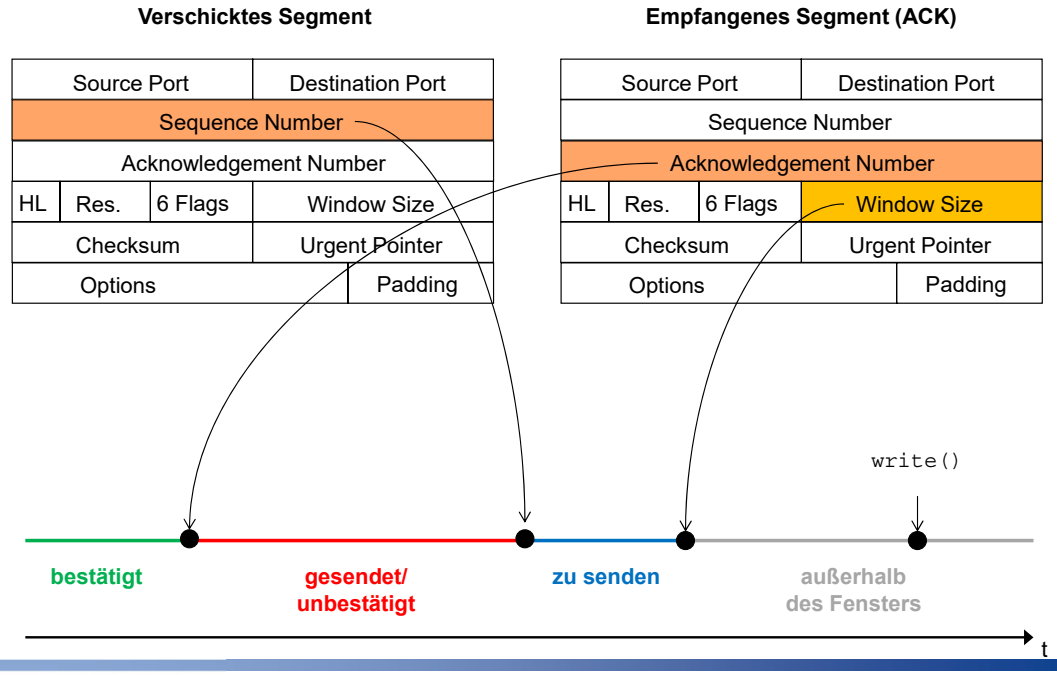


Bedeutung der Felder des TCP-Headers:

- *Source-* und *Destination-Port*: Portnummer von Sender bzw. Empfänger
- *Sequence Number/Acknowledgement Number*: 32-Bit-Sequenz- und Quittungsnummer für die Flusskontrolle
  - Sequenz- und Quittungsnummer zählen einzelne Bytes
  - Die Quittungsnummer gibt das nächste erwartete Byte an
  - Sequenznummern fangen nicht notwendigerweise bei 0 an, der Anfangswert wird beim Verbindungsaufbau festgelegt
  - Piggybacking: eine Quittung kann in einem Datensegment mitgeschickt werden
- *HL*: Header Length: der TCP-Header hat variable Länge: es können Optionen angegeben werden. Eine Angabe der Länge ist nötig (in 32-Bit-Worten), damit der Empfänger weiß, was noch Optionen sind und wo im Segment der Datenteil beginnt.
- *Res* = Reserved. Für spätere Verwendung freigehaltene Bits. Mittlerweile sind zwei dieser Bits als weitere Flags standardisiert; dies wird später noch erläutert.
- Sechs Flags:
  - *URG*: URGENT – gesetzt, wenn das Feld Urgent Pointer beachtet werden soll, um „wichtige“ Daten außerhalb der Flusskontrolle zu senden
  - *ACK*: gesetzt, wenn eine Quittung mitgesendet wird. Nur bei gesetztem ACK-Flag wird der Empfänger des Segments den in Acknowledgement Number befindlichen Wert beachten.
  - *PSH*: PUSH – direkte Weiterleitung der Daten, kein Warten auf weitere Daten (d.h. Versenden auf Senderseite, Verarbeitung auf Empfängerseite)
  - *RST*: RESET – Rücksetzen einer Verbindung, z.B. falls eine Verbindungsabfrage nicht akzeptiert werden kann
  - *SYN*: gesetzt beim Aufbau einer Verbindung

- *Window Size* (WIN): Größe des Bufferspeichers für die Verbindung - gibt an, wie viele Bytes der Empfänger aktuell noch puffern kann (AdvertisedWindow)
- *Checksum*: Prüfsumme über Header und Daten. Dient u.a. zur Verifikation, dass das Paket an das richtige Gerät ausgeliefert worden ist.
- *Urgent Pointer*: in dringenden Fällen können Daten ohne Beachtung der Flusskontrolle gesendet werden. Dieses Feld gibt an, wo die „dringenden Daten“ enden (Byteversatz von der aktuellen Folgenummer)

Flusskontrolle: Senderseite



In Ergänzung zu der vorherigen Folie zum Sende- und Empfangsbuffer sind hier der Bytestrom als Ganzes und der Bezug der Header-Angaben zu diesem Bytestrom dargestellt.

TCP-Header: Protokollevolution

- **TCP-Optionen**
  - Angaben zur Kontrolle/Verwaltung einer Verbindung
    - Erlaubt Weiterentwicklung des „alten“ TCP
      - Stark genutzt, da nur Endsysteme geändert werden müssen
  - Wichtige Optionen:
    - *Maximum Segment Size* (*MSS*): Vereinbarung einer maximalen Segmentgröße beim Verbindungsaufbau
    - *Window Scale*: Skalierung der Fenstergröße durch Angabe eines Faktors beim Verbindungsaufbau, mit dem der Wert in `WIN` multipliziert wird
      - Logarithmische Angabe, maximaler Wert:14
        - maximaler `WIN`-Wert: 65535, maximaler Skalierungsfaktor: 14
        - Damit: $65535 \cdot 2^{14} = 1.073.725.440$ Byte max. Fenstergröße
    - Vereinbarung des *Fehlerbehebungsmechanismus‘* für eine Verbindung
      - Go-Back-N, Selective Repeat, SACK, …

TCP verfügt über kein Versionsfeld zur Weiterentwicklung des Protokolls; stattdessen werden Änderungen durch Optionen umgesetzt. Dies ist einfacher als eine Änderung auf IP-Ebene, da zur Änderungen von TCP nur die Betriebssysteme von Endgeräten aktualisiert werden müssen, nicht auch alle Router. Die Optionen bei TCP werden heute stark genutzt, um veraltete Einstellungen und Mechanismen des ursprünglichen TCP abzuändern.

Über Optionen können beim Verbindungsaufbau Parameter der nachfolgenden Verbindung ausgehandelt werden, z.B.:

- Welcher Mechanismus soll zur Übertragungswiederholung verwendet werden? Go-Back-N, SACK, …
- Welche Maximum Segment Size soll verwendet werden?
- Window Scaling Factor – leider ist die Angabe einer Fenstergröße bei TCP durch 16 Bit beschränkt – ein größeres Fenster als 64 kByte kann nicht eingetragen werden, was bei den Datenraten heutiger Netze unbefriedigend ist. Daher können die Kommunikationspartner einen Skalierungsfaktor vereinbaren, mittels dessen der Wert des WIN-Feldes vervielfacht werden kann.

Aber Optionen können auch während der Datenaustauschphase verwendet werden. Zu beachten ist jedoch, dass ein IP-Paket maximal 1500 Byte umfassen soll, so dass mit steigender Zahl an Optionen der TCP-Payload geringer wird.

Berechnung der Prüfsumme

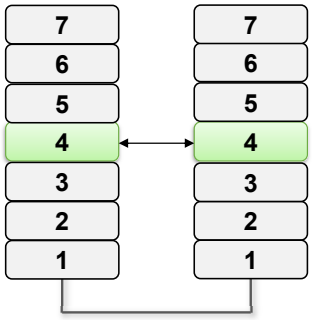
- **Prüfsumme: Validierung der Korrektheit der empfangenen Daten**
  - Berechnung über Pseudo-Header + TCP-Header + Daten
  - Einerkomplement der Summe aller 16-Bit-Worte
- ***Pseudo-Header*:**

Source address (IP)		
Destination address (IP)		
00000000	Protocol = 6	Länge des TCP-Segments

  - Prüfsumme berücksichtigt auch Korrektheit der IP-Adressen

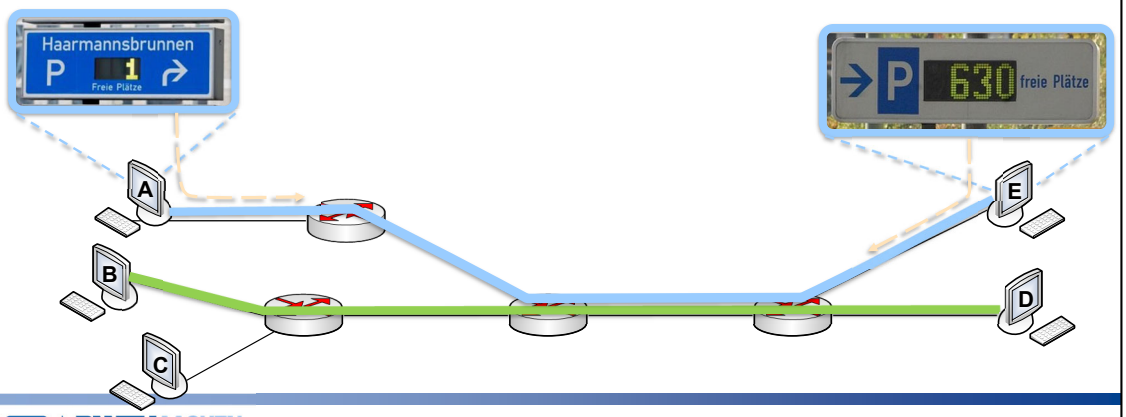
Während IP nur den Header auf Korrektheit prüft, führt TCP auch eine (einfache) Sicherung des Payloads durch. Zudem geht ein sogenannter Pseudo-Header mit in die Berechnung der Prüfsumme ein, durch den auch die IP-Adressen noch einmal mit eingeschlossen werden. Hierdurch soll vermieden werden, dass durch unentdeckte Fehler auf tieferer Ebene (Verfälschung der IP-Adressen) ein Paket an den falschen Host ausgeliefert wird.

- **Protokollmechanismen der Transportschicht**
  - Prozessadressierung, strombasierte vs. paketbasierte Kommunikation, verbindungsorientiert/verbindungslos
- **Die Transportschicht im Internet**
  - TCP (Transmission Control Protocol)
    - Adressierung, Sockets
    - TCP-Verbindung
    - Flusskontrolle
    - TCP-Header
    - Staukontrolle (Congestion Control)
  - UDP (User Datagram Protocol)



Flusskontrolle

- **Wiederholung:**
  - Flusskontrolle regelt die Datenmenge zwischen den Partnern einer TCP-Verbindung
  - Ziel: *Keine Überlastung des TCP-Partners* innerhalb einer Verbindung
  - Aber: Das verhindert keine *Überlastung des Netzes*!



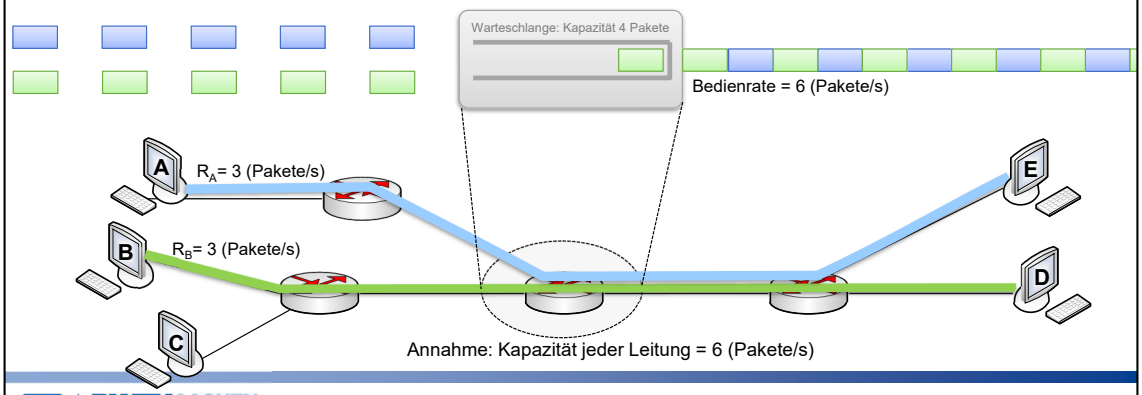
Die bislang vorgestellte Funktionalität von TCP entspricht im Wesentlichen der Funktionalität von Schicht 2. Allerdings gibt es einen bedeutenden Unterschied zwischen TCP und der Schicht 2: auf Schicht 2 wird nur die Übertragung über eine Leitung betrachtet. TCP setzt auf IP auf, durch welches all diese Teilstrecken zu einem komplexen Netz verbunden werden. TCP hat zwar logisch gesehen auch seinen Kanal (die Verbindung), kennt allerdings nicht die zugrundeliegende Netzstruktur. Auf IP-Ebene kann es zum Datenstau kommen (engl. Congestion), falls ein Router über all seine Leitungen in Summe mehr Daten empfängt, als er verarbeiten (d.h. weiterleiten) kann. Als Resultat wird der entsprechende Router eingehende IP-Pakete schlicht verwerfen.

Auf TCP hat dies die Auswirkung, dass keine Quittungen mehr eingehen und nach einem Timeout eine Überragungswiederholung initiiert wird. Dadurch wird die Überlastsituation im Netz noch verschlimmert.

Als (etwas hinkende) Analogie kann man hier die Medienzugriffskontrolle auf Schicht 2 herziehen: ist das Netz stark belastet, so dass es zu vielen Kollisionen kommt, werden die Wartezeiten der einzelnen Stationen so erhöht (randomisiert), dass das Kollisionsrisiko verringert wird; aber auch die erzielbare Datenrate der einzelnen sendenden Stationen wird dadurch reduziert.

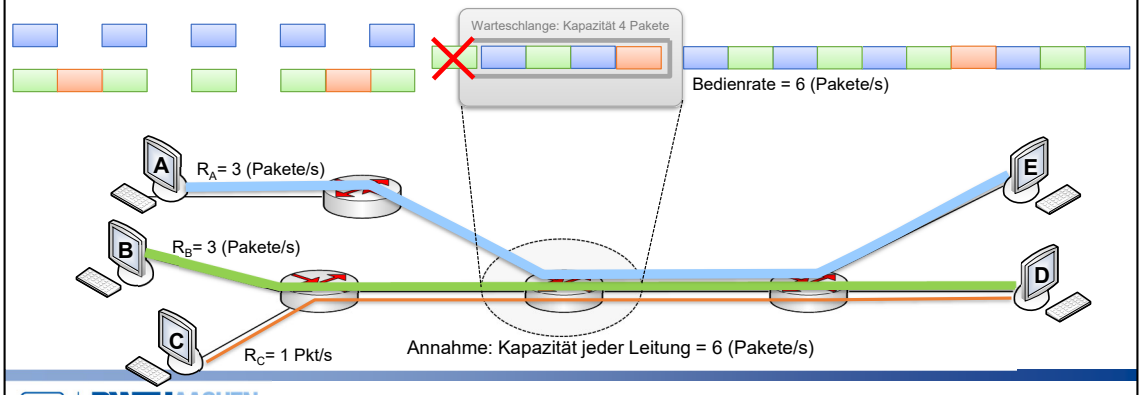
Ein ähnlicher Mechanismus ist auch bei TCP notwendig, nur dass hier Paketverlust der Indikator eines Problems ist, keine Kollisionen.

Stausituationen im Internet



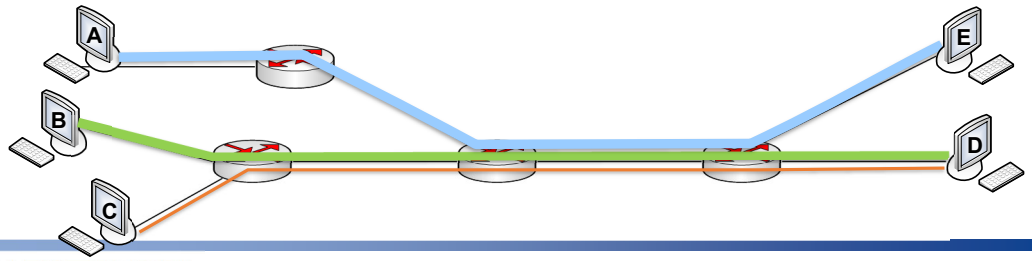
Stausituationen im Internet – Ursache und Definition

- **Überlastung des Kernnetzes: Stau im Internet**
  - Pakete können aufgrund von Überlastsituationen nicht mehr gepuffert werden → Paketverlust
- **Wann treten Stausituationen auf?**
  - Wenn Summe der Datenraten (Ankunftsraten R_j^{in}) auf ausgehendem Port *i* größer als die Kapazität (Bedienrate R_i^{out}) des Link *i* ($\sum_j R_j^{in} > R_i^{out}$) ist und die Wartschlange voll ist



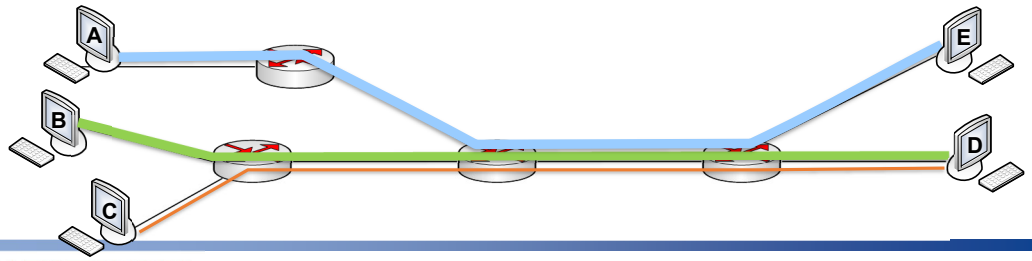
Stausituationen im Internet – Ursache und Definition

- **Überlastung des Kernnetzes: Stau im Internet**
  - Pakete können aufgrund von Überlastsituationen nicht mehr gepuffert werden → Paketverlust
- **Wann treten Stausituationen auf?**
  - Wenn Summe der Datenraten (Ankunftsraten R_j^{in}) auf ausgehendem Port *i* größer als die Kapazität (Bedienrate R_i^{out}) des Link *i* ($\sum_j R_j^{in} > R_i^{out}$) ist und die Wartschlange voll ist
- **Wie erkennt eine TCP-Instanz eine Stausituation?**
  - Sender bemerkt, dass Pakete nicht mehr bestätigt werden (vgl. Kapitel 3, Link Layer)
    - Ausbleiben einer Bestätigung → *Time-Out*
    - Erneute (kumulative) Bestätigung eines Pakets → *Duplicate ACK (DupACK)*



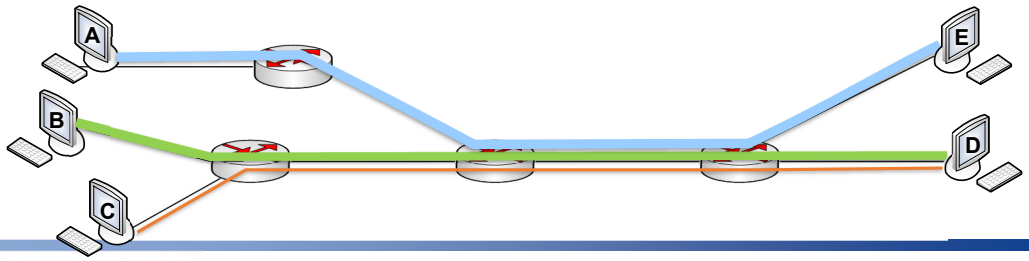
Reaktion auf Stausituation (Staubehandlung)

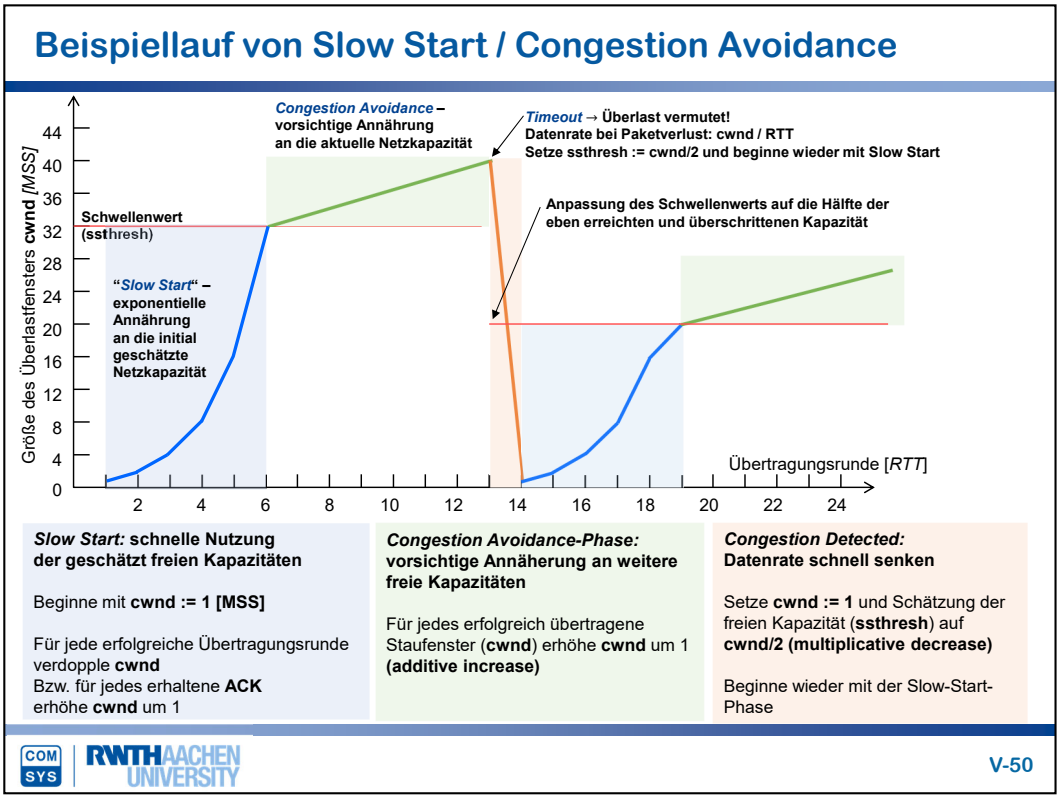
- **Wie soll ein TCP-Sender auf einen Stau reagieren?**
  - Überlast ist Ursache → *Reduktion der Last*
  - Welcher Sender reduziert? → *Die Sender, deren Pakete verloren gehen*
  - Kein Wissen über Netz-Kapazitäten, über Bottleneck-Links, über aktuelle Belastung, über andere Ströme



Reaktion auf Stausituation (Staubehandlung)

- **TCP-Staukontrolle (Congestion Control):**
  - Basiert auf grober Schätzung der freien Netzkapazität
  - Überlastfenster (Congestion Window, `cwnd`) limitiert (neben Sendefenster) die Anzahl der Segmente, die pro Runde (`RTT`) gesendet werden dürfen
  - Angenommene „sichere“ Kapazität (= `ssthresh / RTT`)
    - Initial sehr hoch, bzw. auf maximal Größe des Initial Window (Flusskontrolle, angegeben im TCP-Header, auch als `rwnd` bezeichnet)





Bemerkungen:

- Die initiale Größe des Congestion-Windows wurde im Laufe der Zeit angepasst, es sind mittlerweile auch größere Werte zulässig. Zur Vereinfachung werden wir in Vorlesung und Übung allerdings immer von einem Startwert von 1 ausgehen.
- Die Benennung „Slow Start“ ist irreführend – man fängt zwar langsam mit nur einem Segment an zu senden, um das Netz nicht direkt zu überlasten, steigert dann aber die Senderate exponentiell, um die freie Netzkapazität schnell ausnutzen zu können.
- Die Bestimmung des Congestion Window erfolgt allein durch den Sender.
- Bei Erreichen des Thresholds wird die Senderate nur noch linear gesteigert, um zu verhindern, dass man durch zu schnelle Steigerung plötzlich das Netz überlastet (Stauvermeidung, Congestion Avoidance).
- Die Verkleinerung des Thresholds auf die Hälfte rührt daher, dass angenommen wird, dass unter der vorherigen Konfiguration durch exponentielles Wachstum die Kapazität des Netzes stark überschritten wurde. Daher soll beim weiteren Senden ein vorsichtigeres Herantasten bis an die freie Kapazität erfolgen.

Der Überlastkontrollmechanismus wird auch A*dditive Increase/Multiplicative Decrease (AIMD)* genannt.

Das Überlastfenster cwnd wird am Anfang auf ein Segment gesetzt. Nach der erfolgreichen Übertragung eines Segments dürfen zwei weitere Segmente übertragen werden – bei erfolgreicher Übertragung eines vollen Fensters hat sich das Überlastfenster somit verdoppelt. Eine solche Übertragung eines vollen Fensters ist auf dieser Folie als „Übertragungsrunde“ bezeichnet. Die „Übertragungsrunde“ ist allerdings nur eine vereinfachte Darstellung zur Erläuterung des Algorithmus‘ – in der Realität wird der

Wert von cwnd mit jeder eintreffenden Quittung um die Größe der MSS erhöht. Dies entspricht insgesamt in einer gedachten Übertragungsrunde einer Verdopplung der Fenstergröße. Bei fehlerfreier Übertragung steigt die Größe des Überlastfensters also exponentiell an.

Hat das Überlastfenster einen Schwellenwert erreicht, so wird nur noch bei Übertragung eines vollen Sendefensters die Größe des Überlastfensters um eine MSS erhöht (linearer Anstieg). Man tastet sich sozusagen vorsichtig an den maximalen Wert der Netzkapazität heran. Dies ist der „Additive Increase“. Die vorherige Slow-Start-Phase soll nur verhindern, dass der „Additive Increase“ bei 1 beginnen muss – der Start soll beschleunigt werden. Im Prinzip verhindert der Slow-Start-Algorithmus also einen Slow-Start.

Tritt ein Segmentverlust auf, so wird das Überlastfenster wieder auf den Initialwert zurückgesetzt und der Schwellenwert auf die Hälfte des vorher erreichten Überlastungsfensters gesetzt. Diese Halbierung ist der „Multiplicative Decrease“.

Der Mechanismus produziert also immer einen sägezahnartigen Verlauf. Für längere Verbindungen ist er geeignet, um Überlastsituationen zu vermeiden, aber bei kleinen Datenmengen verschenkt man anfangs sehr viel Netzkapazität. Darüber hinaus führt ein Paketverlust zu einem deutlichen Einbruch in der Übertragungsrate und die Verbindung erholt sich nur langsam davon. Heutzutage kommen Varianten des Slow-Start-Mechanismus‘ zum Einsatz, die ebenfalls effektiv bei der Verhinderung von Netzüberlastungen sind, sich aber schneller von Paketverlusten erholen – Ziel ist es, das „Multiplicative Decrease“ nicht nur auf den Schwellwert, sondern auch auf das Übertragungsfenster anzuwenden, also zu verhindern, dass das Fenster auf 1 zurückfällt.

Stau oder kein Stau?

- **Paketverlust = Stau?**
  - *Massive Überlast*
    - Viele Pakete werden verworfen
    - Große Reduktion der Last erforderlich
    - → Indiz: Der Sender erhält keine ACKs, aber es erfolgt Timeout, da keine Datenpakete ankommen
  - *Leichte Überlast*
    - Einzelne Pakete werden verworfen
    - Große Reduktion der Rate wäre übertrieben
    - → Indiz: Der Sender erhält Duplicated-ACKs, da weitere Pakete durchkommen und bestätigt werden
  - *Übertragungsfehler (z.B. in drahtlosen Netzen)*
    - Einzelne Pakete sind defekt
    - Reduktion der Rate nicht erforderlich
    - → Indiz: Der Sender erhält Duplicated-ACKs

Timeout → Slow Start

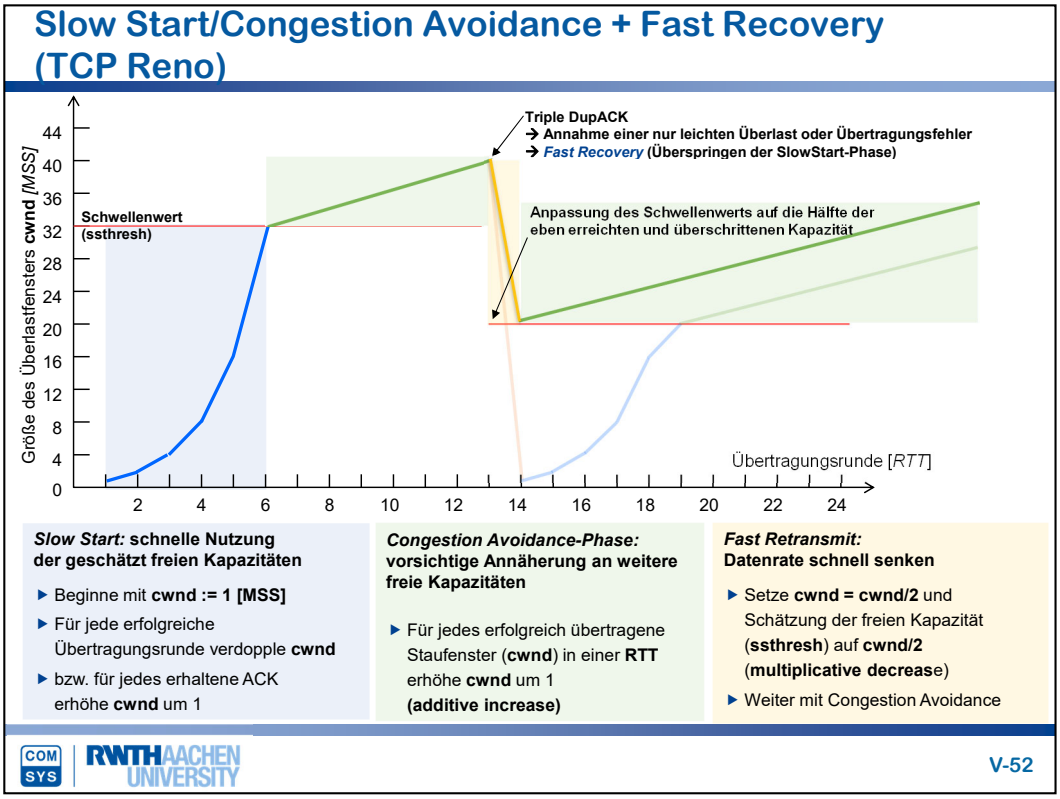
TCP Tahoe (erste Version TCP): Triple DupACK → Slow Start

TCP Reno (zweite TCP Version) Triple DupACK → Fast Recovery

Gehen nur einzelne Segmente verloren, ohne dass eine wirkliche Überlastsituation vorliegt, reagiert der Slow-Start-Algorithmus falsch – eine drastische Reduktion der Datenrate ist nicht nötig. Um zu vermeiden, dass der Slow-Start ausgelöst wird, schickt der Empfänger bei jedem erhaltenen Segment eine Quittung zurück; kann er keine positive Quittung schicken, da ein Segment fehlt, sendet er die zuletzt gesendete Quittung noch einmal. Der Sender erhält dadurch Feedback: es liegt keine vollständige Überlast vor, da immer noch einige seiner Segmente beim Empfänger ankommen. Daher kann er eine direkte Wiederholung des fehlenden Segments initiieren.

Der Sender reagiert aber erst beim dritten DUP-ACK, welches er empfängt. Da IP keine reihenfolgetreue Auslieferung der Daten garantiert, kann es sein, dass ein DUP-ACK bedeutungslos ist, da kurz nach seiner Versendung die angemahnten Daten doch eintreffen. Dies würde zu einer unnötigen Wiederholung und damit Belastung des Netzes führen. Daher reagiert der Sender erst nach dem Eintreffen mehrerer Duplikate einer Quittung. Er wiederholt das anscheinend fehlende Segment in der Hoffnung, dass dadurch noch vor dem Timeout für das verlorengegangene Segment ein ACK zurückkommt und Slow Start vermieden wird. Dieser Mechanismus wird „Fast Retransmit“ genannt.

Unterschiedliche TCP-Varianten unterscheiden sich vorranging im verwendeten Congestion-Control-Algorithmus. Basierend auf dem implementierten Algorithmus haben die TCP-Varianten Namen, traditionell nach Städten oder Regionen benannt.



Bei Eintreffen des dritten DUP-ACK wird nicht nur eine sofortige Neuübertragung vorgenommen, die Verbindung wird auch eingefroren. Der Schwellwert wird um die Hälfte reduziert und das Überlastfenster auf die Größe des Schwellwertes gesetzt, um die Datenrate zu reduzieren. Mit jedem weiteren einkommenden DUP-ACK wird eine Überlastsituation unwahrscheinlicher, da immer mehr Segmente korrekt ankommen. Daher wird das Überlastfenster an die Zahl ankommender Pakete angepasst und eventuell noch weiteres Senden erlaubt. Trifft eine Quittung für das fehlende Segment ein (und vermutlich direkt auch für viele folgende Segmente), setze das Überlastfenster zurück auf den Threshold, um die Datenrate wieder vorsichtig weiter steigern zu können.

Durch dieses Vorgehen wird ein drastischer Abfall der Datenrate wie durch Slow Start vermieden; man setzt nach einer potentiellen Überlastsituation wieder mit einer höheren Datenrate (durch den Threshold vorgegeben) auf und kann auch während der Fehlerbehandlungsperiode weitersenden.

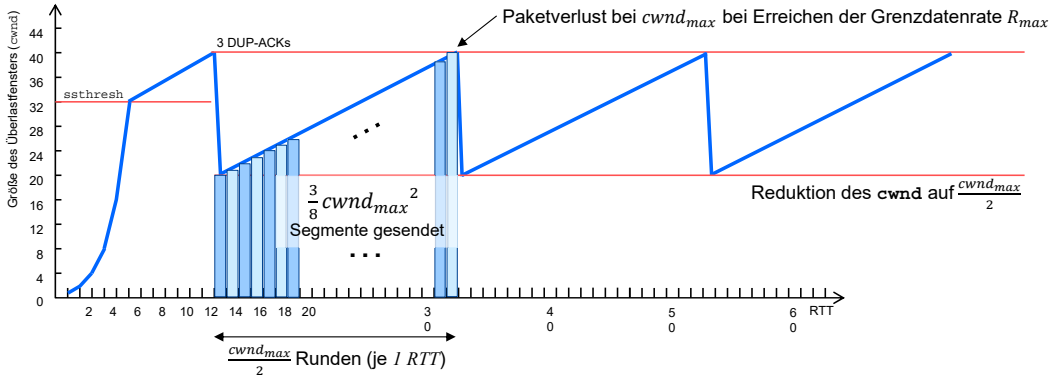
Insgesamt ist nun das AIMD-Prinzip besser umgesetzt: die Steigerung der Datenrate erfolgt linear (nach anfänglicher Slow-Start-Phase), die Verringerung multiplikativ (Halbierung der Datenrate bei Segmentverlust).

(Bitte beachten: das Vorgehen bei Fast Retransmit / Fast Recovery lässt sich schlechter in das Prinzip der Übertragungsrunden einpassen als Slowstart. Auf Folie 55 ist eine detailliertere Darstellung dessen, was passiert.)

Eine Frage ist noch, wie der initiale Threshold gewählt werden sollte. Die RFCs sagen dazu, dass er möglichst groß angesetzt werden solle. Z.B. könne er am Fenster des Empfängers orientiert werden (Window Size im TCP-Header, auch rwnd genannt), um wenn möglich schnell die durch den Empfänger erlaubte Kapazität erreichen zu können (Initial sshthresh = advertised window).

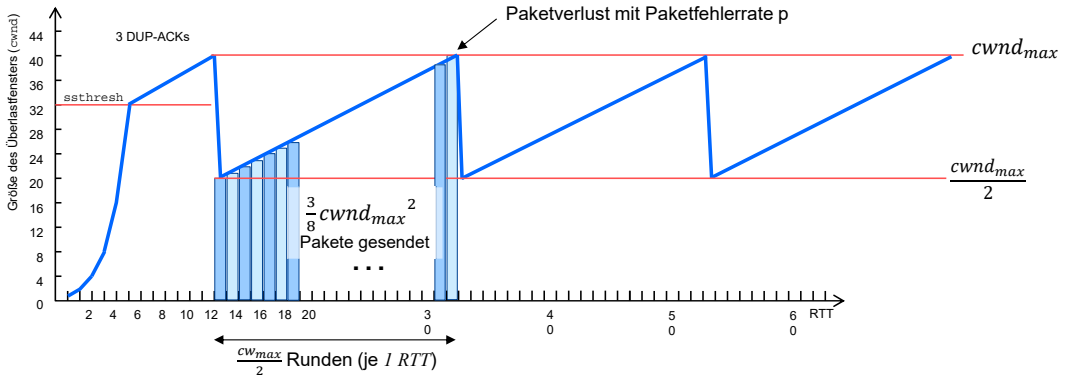
Bestimmte Betriebssysteme verwenden eventuell unterschiedliche Werte. Laut Wikipedia war in Microsoft Windows 9x, Me und NT das Empfangsfenster standardmäßig 8 Kilobyte groß. Windows 2000 und XP reservierten 16 Kilobyte. Laut der englischen Version des Artikels hatten Windows Vista und Windows 7 ein Empfangsfenster von 64 kB, hochskalierbar bis 16 MB (Autotuning).

TCP Goodput (Throughput made good) bei garantierter Datenrate



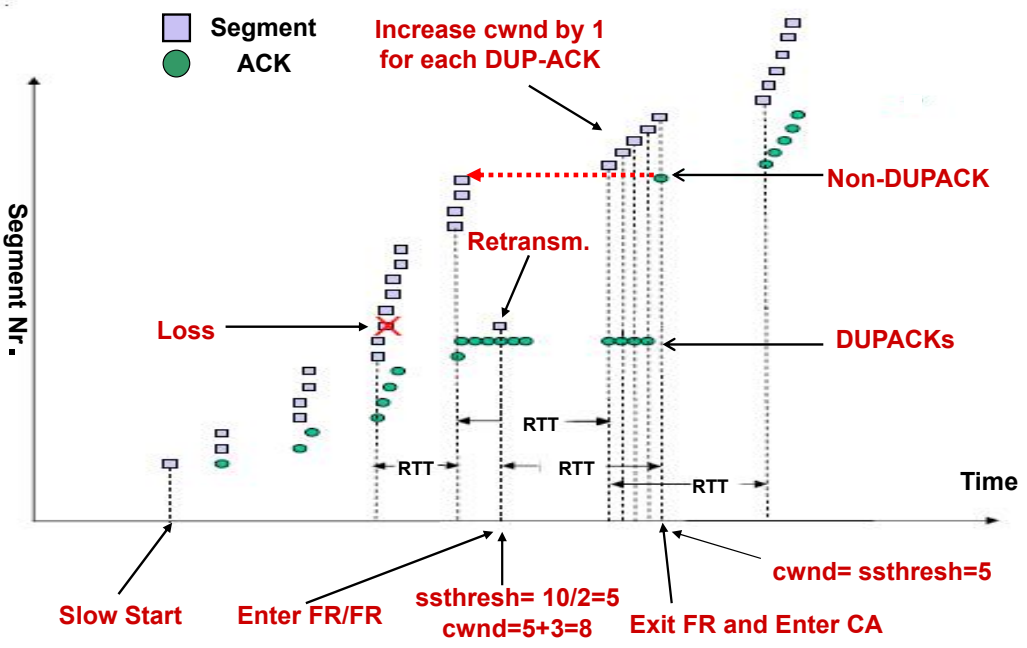
- **Annahme: Konstanter Durchsatz R_{max} ist für die TCP Verbindung garantiert**
  - Welche Datenrate erreicht die TCP Verbindung im Durchschnitt?
    - Datenrate entspricht der durchschnittlichen Rate pro Fast-Recovery-Abschnitt (1 Sägezahn)
    - Gesendete Pakete pro „Sägezahn“: $\#\boldsymbol{Pakete} = \frac{cwnd_{max}}{2} \cdot \left(\frac{cwnd_{max}}{2} + cwnd_{max}\right) / 2 = \frac{3}{8} cwnd_{max}^2$
    - Durchsatz pro „Sägezahn“: $R_{avg} = \frac{\#\boldsymbol{Pakete} \cdot \boldsymbol{MSS}}{\#\boldsymbol{Runden} \cdot \boldsymbol{RTT}} = \frac{\frac{3}{8} cwnd_{max}^2 \cdot \boldsymbol{MSS}}{\frac{cwnd_{max}}{2} \cdot \boldsymbol{RTT}} = \frac{3 \cdot cwnd_{max} \cdot \boldsymbol{MSS}}{4 \cdot \boldsymbol{RTT}} = \frac{3}{4} R_{max}$

TCP Goodput bei Paketverlustrate *p*



- **Neue Annahme: unbeschränkte Kapazität, aber Paketverlustrate *p***
  - Gesendete Pakete bis Verlust: $\#Pakete = \frac{cwnd_{max}}{2} \cdot \left(\frac{cwnd_{max}}{2} + cwnd_{max}\right) / 2 = \mathbf{\frac{3}{8} cwnd_{max}^2}$
  - $p = \frac{1}{\#\text{Pakete}} = \frac{1}{\frac{3}{8}cwnd_{max}^2} \Rightarrow cwnd_{max} = \sqrt{\frac{8}{3p}}$
- **Durchsatz:** $R_{\text{avg}} = \frac{\#packets \cdot MSS}{\#rounds \cdot RTT} = \frac{\frac{3}{8}cwnd_{max}^2 \cdot MSS}{\frac{cwnd_{max}}{2} \cdot RTT} = \frac{3}{4} cw_{max} \frac{MSS}{RTT} = \frac{3}{4}\sqrt{\frac{8}{3p}} \frac{MSS}{RTT} = \sqrt{\frac{3}{2}} \frac{\mathbf{MSS}}{\mathbf{RTT} \cdot \sqrt{p}}$

Beispiel FR/FR: Paketverlust



Die hier gezeigte Art der Darstellung wird in der TCP-Welt meist gewählt. Da sie allerdings deutlich komplexer ist als die vereinfachte Darstellung unter Zuhilfenahme von Übertragungsrunden, werden wir meist bei der einfacher verständlichen Übertragungsrundendarstellung bleiben.

Weitere Verbesserungen der Staukontrolle

- **Heute: Vielzahl von Erweiterung:**
  - *TCP SACK*: Empfänger verschickt Listen von positiven und negativen Quittungen (im Optionsfeld des Headers)
  - Binary Increase Congestion Control (*BIC*) für Hochgeschwindigkeitsnetze mit großer Latenz
  - *CUBIC* als Variante von BIC in Linux (2.6.19 – 3.1): Überlastfenster ist kubische Funktion der Zeit seit der letzten Überlastsituation
    - Fenster wächst zunächst konkav (schnelle Annäherung an Fenstergröße vor der letzten Überlastsituation), danach konvex (vorsichtiges Testen auf höhere Datenraten, schnelle Steigerung)
    - Entkopplung vom Empfang von Quittungen
    - Abgelöst durch TCP *Proportional Rate Reduction* in Linux 3.2
  - *Compound TCP* als TCP Reno mit Delay-Window
  - …

Die ursprünglichen Algorithmen sind bei vielen Betriebssystemen weiter verbessert worden, beispielsweise durch Selective Acknowledgements. Linux und Windows als prominente Kernel implementieren ihre eigenen Varianten.

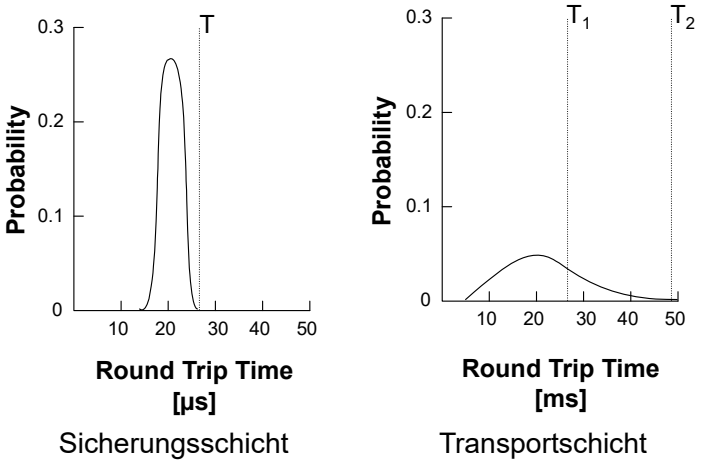
CUBIC: die Fenstergröße ist eine kubische Funktion der Zeit seitdem das letzte Mal eine Stausituation aufgetreten war; der Empfang von ACKs ist nicht nötig, um das Fenster zu vergrößern. Dadurch wird zum einen eine zu schnelle Reaktion auf fehlende ACKs vermieden, zum anderen ist eine bessere Fairness zwischen unterschiedlichen TCP-Strömen gegeben, da bei traditionellen TCP-Algorithmen das Fenster schneller wächst, wenn die RTT zum Empfänger sehr gering ist. Dadurch kann es passieren, dass TCP-Ströme zwischen relativ dicht beieinander befindlichen Stationen gegenüber denen, die weite Strecken zurücklegen, bevorzugt werden und eine höhere mittlere Datenrate bekommen. Mittlerweile abgelöst durch TCP Proportional Rate Reduction, welches die Fenstergröße nach einem Paketverlust nahe am Threshold hält.

Compound TCP: optimiert für Verbindungen mit hohem Bandbreiten-Verzögerungs-Produkt. Es basiert auf TCP Reno, ergänzt das Fenster allerdings um eine Delay-basierte Komponente. Dieser Teil des Fenster wächst bei kleiner Verzögerung zwischen den Kommunikationspartner sehr schnell an, um die zur Verfügung stehende Kapazität des Netzes schnell auszunutzen; später wird das Fenster verkleinert, um die Datenrate insgesamt relativ konstant und in der Nähe des Bandbreiten-Verzögerungs-Produktes zu halten.

Einen (unvollständigen) Überblick über TCP-Varianten bietet http://en.wikipedia.org/wiki/TCP_congestion-avoidance_algorithm.

Retransmission Timer

- **Wie sollte der Retransmission Timer gewählt werden?**
  - T_1: zu klein, zu viele Neuübertragungen
  - T_2: zu groß, zu lange Wartezeiten bei tatsächlichem Verlust



Sicherungsschicht | Transportschicht

Ein Problem bei der praktischen Umsetzung der Algorithmen ist: wann sollte ein Timeout ausgelöst werden, wie groß sollte der Retransmission Timer gewählt werden?

Während man auf der Sicherungsschicht immer nur einen einzelnen Link betrachtet und daher gute Schätzungen abgeben kann, wann eine Quittung eintreffen sollte, betrachtet man auf der Transportschicht ein Netzwerk aus Routern, die indeterministische Verzögerungen erzeugen können. Die Laufzeit eines Segments durchs Netz kann nur geschätzt werden und sich auch von Segment zu Segment ändern. Der Retransmission Timer kann daher nicht statisch festgelegt werden.

Berechnung des Retransmission Timers

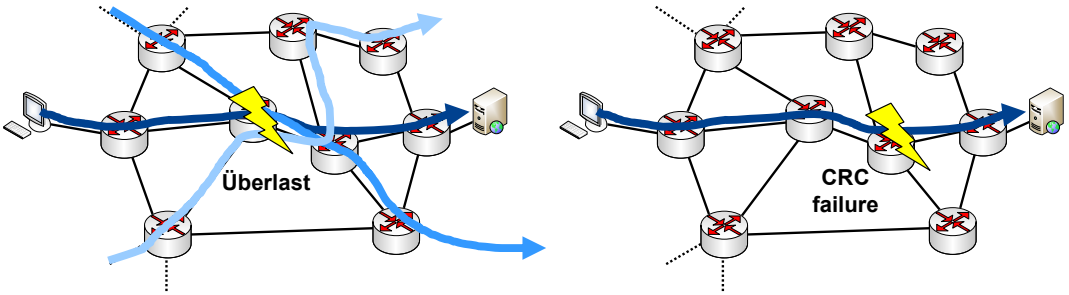
- **Algorithmus zur Berechnung (Jacobson, 1988)**
  - Stoppe Round-Trip-Time *t* für jedes Segment
  - Führe Variable *RTT* als Schätzung der aktuellen Round-Trip-Time
  - Initialisiere *RTT* mit einem Startwert
    - Z.B. 2 Sekunden in Linux, 3 Sekunden in Windows
  - Falls eine Quittung vor Ablauf des Retransmisson Timers eintrifft, aktualisiere *RTT*:
    - $RTT = \alpha \bullet RTT + (1 - \alpha) \bullet t_{letzte\ RTT}$
    - α ist Glättungsfaktor, oft 0.875
  - Wahl der Retransmission Timers *RTO*
    - Erster Ansatz: wähle Retransmission Timer als $\beta \bullet RTT$
    - Wie sollte β gewählt werden?

Berechnung des Retransmission Timers

- **Beispielalgorithmus (Jacobson, 1988)**
  - Wahl von β: orientiert an Standardabweichung der Round-Trip-Zeiten
    - $\beta = \alpha' \bullet \beta + (1 - \alpha') \bullet |RTT - t_{letzte\ RTT}|$
  - Dann: $RTO = RTT + 4 \bullet \beta$
    - „4“ wurde mehr oder weniger willkürlich gewählt (effiziente Implementierung der Multiplikation)
    - Ist *RTO* kleiner 1 Sekunde, wird auf 1 Sekunde aufgerundet
- **Zusätzlich:**
  - Mit jedem Timeout nach einer Neuübertragung verdopple den Wert des Retransmission Timers
    - Wiederholter Verlust scheint auf dauerhafte Überlastsituation hinzudeuten
  - Kann bis auf mehrere Minuten steigen!

Neuere Entwicklung: Explicit Congestion Notification

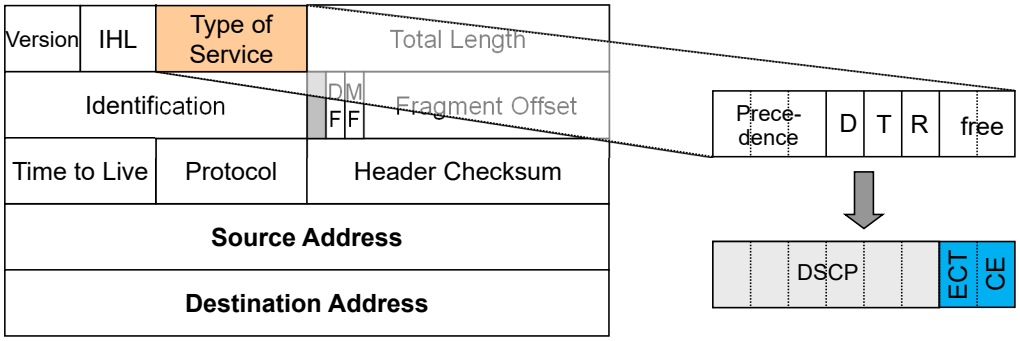
- **Router verwirft bei Überlast Pakete**
  - Sender erhält DUP-ACKs oder hat Timeout
- **Sender kann nicht entscheiden:**
  - Liegt eine Überlastung eines Routers vor?
  - Ist ein Paket aufgrund eines Bitfehlers verworfen worden?



  - Wissen nur auf Schicht 3 vorhanden!

Neuere Entwicklung: Explicit Congestion Notification

- **Überarbeitung des IP-Headers:**
  - Neudefinition des „Type-of-Service“-Felds
    - DSCP – Differentiated Service Code Point (QoS-Unterstützung)
    - *ECT – Explicit Congestion Notification (ECN) Capable Transport*
    - *CE – Congestion Experienced*

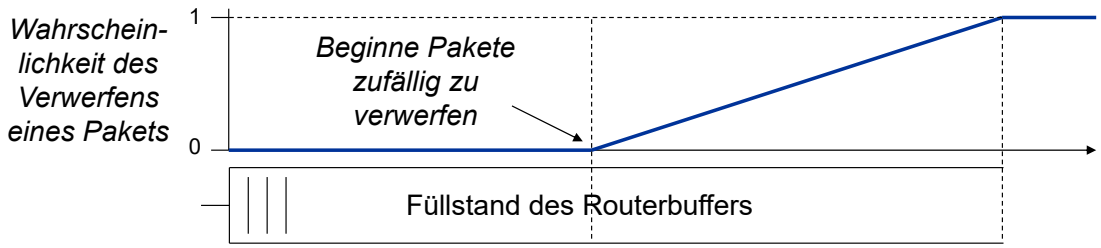


Bereits vor mehreren Jahren wurde das ToS-Feld des IP-Headers neu definiert. Da Routerhersteller die Informationen sowieso nicht berücksichtigten, wurden die sechs standardisierten Bits umdefiniert zu einem „Differentiated Service Code Point“. Dieser wird im Rahmen einer QoS-Erweiterung von IP benötigt, die hier nicht behandelt werden soll.

Die beiden letzten Bits des Feldes wurden zunächst unberührt gelassen: sie waren für spätere Verwendung freigehalten worden und im Laufe der Zeit von manchen Anwendungen für proprietäre Signalisierungszwecke missbraucht worden. Mittlerweile sind aber auch diese Bits neu definiert, um mit der Staukontrolle von TCP zusammenzuarbeiten.

Interaktion von TCP und IP

- **IP kann Überlastsituationen erkennen:**
  - ECT-Flag wird vom Sender gesetzt, wenn er die Stauerkennungs-erweiterung implementiert
  - Router können das CE-Flag setzen, wenn eine Überlastsituation vorliegt
    - *Random Early Detection* (RED)



  - Der Empfänger erkennt eine Überlastsituation anhand des CE-Flags

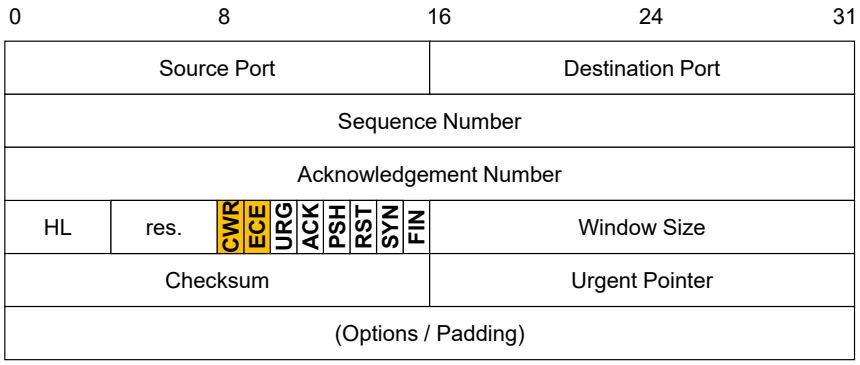
In einer einfachen Router-Implementierung werden empfangene IP-Pakete in einen Buffer eingereiht und der Reihe nach entnommen, auf ihre Zieladresse hin untersucht und weitergeleitet. Ist der Buffer voll, werden weitere Pakete verworfen.

Heutige Router implementieren allerdings „Random Early Detection“. Um Überlast zu vermeiden, beginnt ein Router bereits ab einem definierten Füllstand seines Buffers, zufällig Pakete zu verwerfen. Die Verwurfswahrscheinlichkeit steigt dabei mit zunehmendem Bufferfüllstand. Die Idee dahinter ist, mit einem Verwerfen von Paketen nicht zu warten, bis Überlast eingetreten ist, sondern Pakete schon vorher zu verwerfen, damit der Sender die Datenrate reduziert. Dadurch pendeln sich TCP-Verbindungen bei einer Datenrate ein, die das Netz weitgehend auslastet, aber nicht überlastet.

Kritisieren kann man nun allerdings, dass selbst ohne akute Überlast Pakete verworfen werden. Dies soll die ECN-Erweiterung vermeiden: bei Verwendung von ECN kann ein Router das CE-Flag im IP-Header setzen, statt Pakete zu verwerfen. Dadurch erfährt die empfangende IP-Instanz, dass im Netz eine Überlast droht. Diese Information muss nun an die sendende TCP-Instanz weitergegeben werden.

Modifikation von TCP

- **TCP und ECN: Verwendung zweier neuer Flags**
  - *ECN Echo (ECE)*: Empfänger setzt dieses Bit in den Quittungen für Pakete mit gesetztem CE-Flag
  - Sender passt *cwnd* an und signalisiert dies dem Empfänger über das *CWR-Flag (Congestion Window Reduced)*

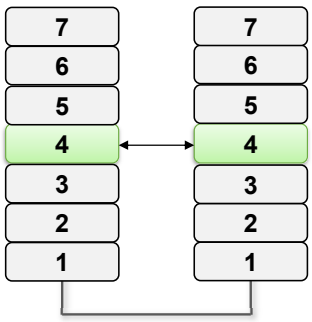


Dies erfolgt über den TCP-Header, durch Standardisierung zweier weiterer Flags. In der Quittung auf das Segment, welches ohne ECN aufgrund von RED verworfen worden wäre, setzt der Empfänger das ECE-Flag. Der Sender verhält sich daraufhin so, als wäre ein Paketverlust aufgetreten (Verringerung von cwnd nach FR/FR), ohne dass ein Timeout droht.

Steigerung der TCP-Performance

- **Auch außerhalb der Staukontrolle Verbesserungen**
  - *Fast Open*
    - Beschleunigung des Datenaustauschs
    - Daten bereits in Handshake-Nachrichten
    - Im Linux-Kernel ab Version 3.13 Standard
  - *Multipath-TCP*
    - Gleichzeitige Nutzung mehrerer Pfade
      - Erhöht Redundanz und Durchsatz
      - Z.B. Server mit Multihoming mehr Stabilität bei Ausfall eines Pfades
    - Mobilitätsunterstützung
      - Nutzung von mehreren Netzwerktechnologien gleichzeitig
      - Wechsel z.B. zwischen WiFi und Mobilfunk wird vereinfacht
  - … mehr dazu z.B. in “Communication Systems Engineering”

- **Protokollmechanismen der Transportschicht**
  - Prozessadressierung, strombasierte vs. paketbasierte Kommunikation, verbindungsorientiert/verbindungslos
- **Die Transportschicht im Internet**
  - TCP (Transmission Control Protocol)
    - Adressierung, Sockets
    - TCP-Verbindung
    - Flusskontrolle
    - TCP-Header
    - Staukontrolle (Congestion Control)
  - UDP (User Datagram Protocol)



UDP: „Keep it simple“

- **UDP: verbindungslose Kommunikation**
  - *Unzuverlässige* Datenübertragung
    - Paketverlust, Reihenfolgevertauschung und Duplikate werden nicht erkannt und nicht behandelt (keine Quittungen)
    - Fehlerhaft empfangene Pakete können zwar durch eine (optionale) Checksumme erkannt werden, werden aber einfach verworfen
  - Geringe Zuverlässigkeit, aber *kein Overhead durch Verbindungsverwaltung!*
    - Dadurch schnellere Übertragung kleiner Datenmengen
    - Möglich: Quittungen/Neuübertragungen in Anwendungen implementiert
  - Hauptzweck von UDP: Angabe von Sender- und Empfänger-Port auf dieser Ebene notwendig
    - Und: Nutzung für Multicast (nicht möglich bei TCP)
  - Geringer Funktionsumfang: nur 8 Byte Header

UDP - User Datagramm Protocol

UDP stellt einen einfachen, unbestätigten Datagrammdienst auf der Transportschicht zur Verfügung. UDP versendet Datagramme bis zu einer Größe von 64 kByte. Der Empfang wird nicht bestätigt, und es muss beachtet werden, dass Datagramme verloren gehen können.

Die einzigen Protokollfunktionen, die UDP ausführt, sind lokale Fehlerbehandlung (fehlerhafte PDUs werden verworfen) und Multiplexen (wie bei TCP mit Ports).

UDP wird häufig benutzt, um kleine Informationsmengen, die in keinem unmittelbaren Zusammenhang stehen, auszutauschen. Man möchte dabei auf den Verbindungsoverhead von TCP verzichten. UDP wird oft als „schnell“ bezeichnet, da es weder durch den Verbindungsoverhead noch durch Staukontrolle ausgebremst wird und man theoretisch mit höchsten Datenraten übertragen kann, ohne ausgebremst zu werden.

Außerdem kann es von Nutzen sein, dass man keine Fehlerbehebungsmechanismen ausführt, beispielsweise bei einer Videoübertragung. Wenn man 0 als Prüfsumme einträgt, so wird die Fehlerkontrolle ganz ausgeschaltet und es können auch fehlerhafte PDUs übergeben werden.

Verglichen mit dem TCP-Segment ist das UDP-Datagramm daher sehr einfach aufgebaut.

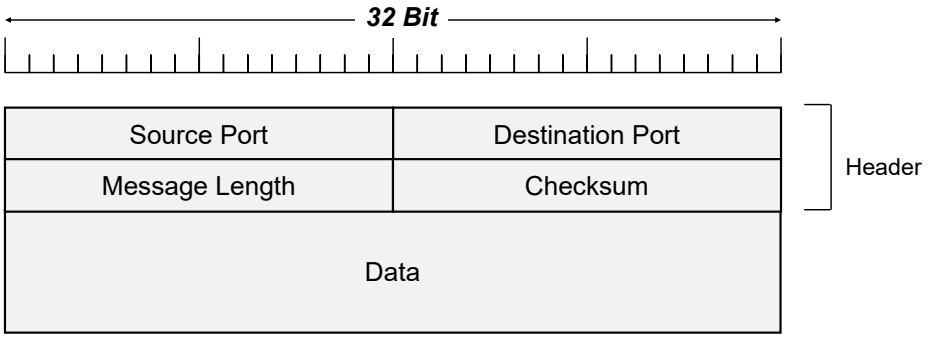
Die Adressierung der Verbindungsendpunkte verläuft analog zu TCP, wobei auch für UDP sog. Well-known ports vereinbart wurden, z.B.:

- 13: Daytime
- 53: Domain Name System
- 123: Network Time Protocol

Der Header von UDP umfasst nur zwei 32-Bit-Worte (Sender- und Empfängerport, ein Prüfsummenfeld (Mechanismus wie bei TCP) und ein 16-Bit-Längenfeld.

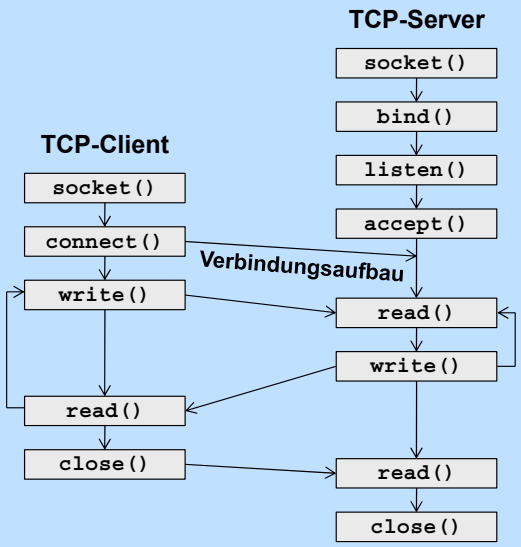
UDP: User Datagram Protocol

- **Kompaktes Header-Format**
  - Im Wesentlichen Portnummern
  - Checksumme ist optional

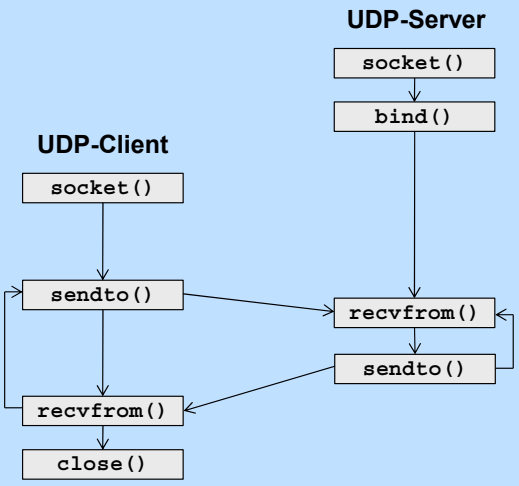


Ablauf bei der Kommunikation über Sockets

- TCP-basierte Kommunikation



- UDP-basierte Kommunikation



UDP-Sockets bieten einen geringeren Umfang an Primitiven als TCP-Sockets, da sie keinen Verbindungsauf- und –abbau verwenden.

Beispiel: UDP Echo Server

```
   int main()
   {
10    int sockfd;
      struct sockaddr_in servaddr, cliaddr;

      sockfd=socket(AF_INET,SOCK_DGRAM,0);
      bzero(&servaddr, sizeof(servaddr));
      servaddr.sin_family=AF_INET;
15    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
      servaddr.sin_port=htons(8012);
      bind(sockfd, (struct sockaddr*) &servaddr,  sizeof(servaddr));
      for (;;) { /* infinite loop */
          int n, socklen_t len;
20        char buffer[256];
          n=recvfrom(sockfd, buffer, 256, 0,(struct
                     sockaddr*)&cliaddr,&len);
          sendto(sockfd, buffer, n, 0, (struct sockaddr*)
                 &cliaddr, len);
      }
      return 0;
25 }
```

COM SYS | RWTH AACHEN UNIVERSITY

V-69

Description:

0-7: the usual `#include` instructions

10,11: required variables

12: `socket()` call IPv4 (`AF_INET`) and UDP (`SOCK_DGRAM`)

13-16: initialization of the `sockaddr` structure with the server address and port

17: `bind()` only required if fixed port

18-23: infinite loop

21: receive a packet (max. 256 Byte).
The received address information is stored in `cliaddr` (max. length `len`).

22: send back the received data exactly to the address and the port from which the data was sent

Client

```
   int main(int argc, char **argv)
   {
10    int sockfd; struct sockaddr_in servaddr;
      if(argc!=3){
         printf("usage: %s <addresse> <port>\n", argv[0]);
         exit(0);}
      memset(&servaddr, 0, sizeof(servaddr));
15    servaddr.sin_family=AF_INET;
      servaddr.sin_port=htons(atoi(argv[2]));
      inet_pton(AF_INET, argv[1], &servaddr.sin_addr);
      sockfd=socket(AF_INET,SOCK_DGRAM,0); /* Ipv4 & UDP */
      for (;;) {
20        /* an int and some buffer space */
         int n; char send_buffer[1024], recv_buffer[1024];
          /* read a line from stdin */
         if( fgets(send_buffer, 1024, stdin) == NULL ) break;
         sendto(sockfd, send_buffer, strlen(send_buffer), 0,
           (struct sockaddr*)&servaddr, sizeof(servaddr));
25       n=recvfrom(sockfd, recv_buffer, 1024, 0, NULL, NULL);
         recv_buffer[n]=0; /* make sure the string ends !! */
          /* write the received string to stdout */
         fputs(recv_buffer,stdout);
      };
30    close(sockfd);
      return 0;
   }
```

COM SYS | RWTH AACHEN UNIVERSITY V-70

Description:

0-7: the usual `#include` instructions

10: required variables

13: examining the command line parameters

14-17: initialization of the `sockaddr` structure with the destination address and port

18: Initialize `socket()`: IPv4 (`AF_INET`) and UDP (`SOCK_DGRAM`)

19-29: infinite loop

21: some variables for the loop block

23: read from `stdin`, with abort condition

24-25: send input and receive the result

27,28: add the string end and write the received string to `stdout`

Zusammenfassung

- **Zwei prominente Protokolle zur Kommunikation zwischen Prozessen:**
  - TCP: zuverlässige Datenübertragung durch Quittungsmechanismus und Flusskontrolle (virtuelle Verbindungsorientierung)
  - UDP: schnelle Übertragung, da es keine Kontrollmechanismen hat; füge einem IP-Paket lediglich Ports hinzu
- **Im Laufe der Jahre: Entwicklung weiterer Transportprotokolle**
  - SCTP (Stream Control Transmission Protocol)
  - DCCP (Datagram Congestion Control Protocol)
  - QUIC (Quick UDP Internet Connections)
  - …

TCP wird laufend weiterentwickelt – nicht nur die Verfahren zur Congestion Control werden ständig erweitert, auch in anderer Hinsicht erfolgen Erweiterungen. So gibt es mittlerweile z.B. eine Spezifikation zu „TCP Fast Open“, bei dem bereits im ersten SYN-Segment Daten mitgeschickt werden können, um den Start der Datenübertragungsphase zu beschleunigen. Eine andere wesentliche Neuerung ist Multipath-TCP, bei dem simultan Daten über mehrere IP-Adressen/Interfaces versendet werden können, die alle hinter einer TCP-Verbindung verborgen sind.

Zusatz-Vorlesung zu Transport-Evolution (Video, nicht klausurrelevant)

- **Ursachen und Probleme von TCP**
  - Evolution des Internet
    - Alter von TCP, Wandlung der Nutzung des Internet, Technologie Entwicklung
  - Beispiel: Parallelismus und TCP (Connection Racing)
- **Evolution von TCP**
  - Erweiterbarkeit von TCP und Limitierungen
    - WS, SACK, TS, ECN, IW10 und Pacing, TFO, MPTCP, ...
  - Middleboxes
- **QUIC**
  - Sinn und Ziele
  - Aufbau
  - Fundamentale Unterschiede zu TCP
- **Congestion Control**

https://www.ccs-labs.org/teaching/rn/animations/

Lessons Learned

- **Transportschicht dient zur Adressierung von Prozessen**
  - Verwendung von Ports
- **Unterschiedliche Anwendungscharakteristika**
  - Strombasierte Kommunikation (TCP)
    - Zuverlässigkeit durch ARQ
    - Flusskontrolle durch Sliding Window mit dynamischer Fenstergröße
    - Staukontrolle zur Vermeidung der Netzüberlastung
      - Wird immer noch weiterentwickelt
  - Paketbasierte Kommunikation (UDP)
    - Keine Kontrollmechanismen, daher gleiche Probleme wie IP
    - Aber kein Overhead durch Verbindungsaufbau und –verwaltung
      - Geringere Verzögerung (Latenz) beim Senden kleinerer Datenmengen
  - Aktuellere Entwicklung: QUIC – adressiert Zuverlässigkeit und Latenz

Frohe Feiertage und einen erfolgreichen Start in 2023

Vorlesung Datenkommunikation WS 2022/23

Klaus Wehrle

Communication and Distributed Systems
Chair of Computer Science 4
RWTH Aachen University

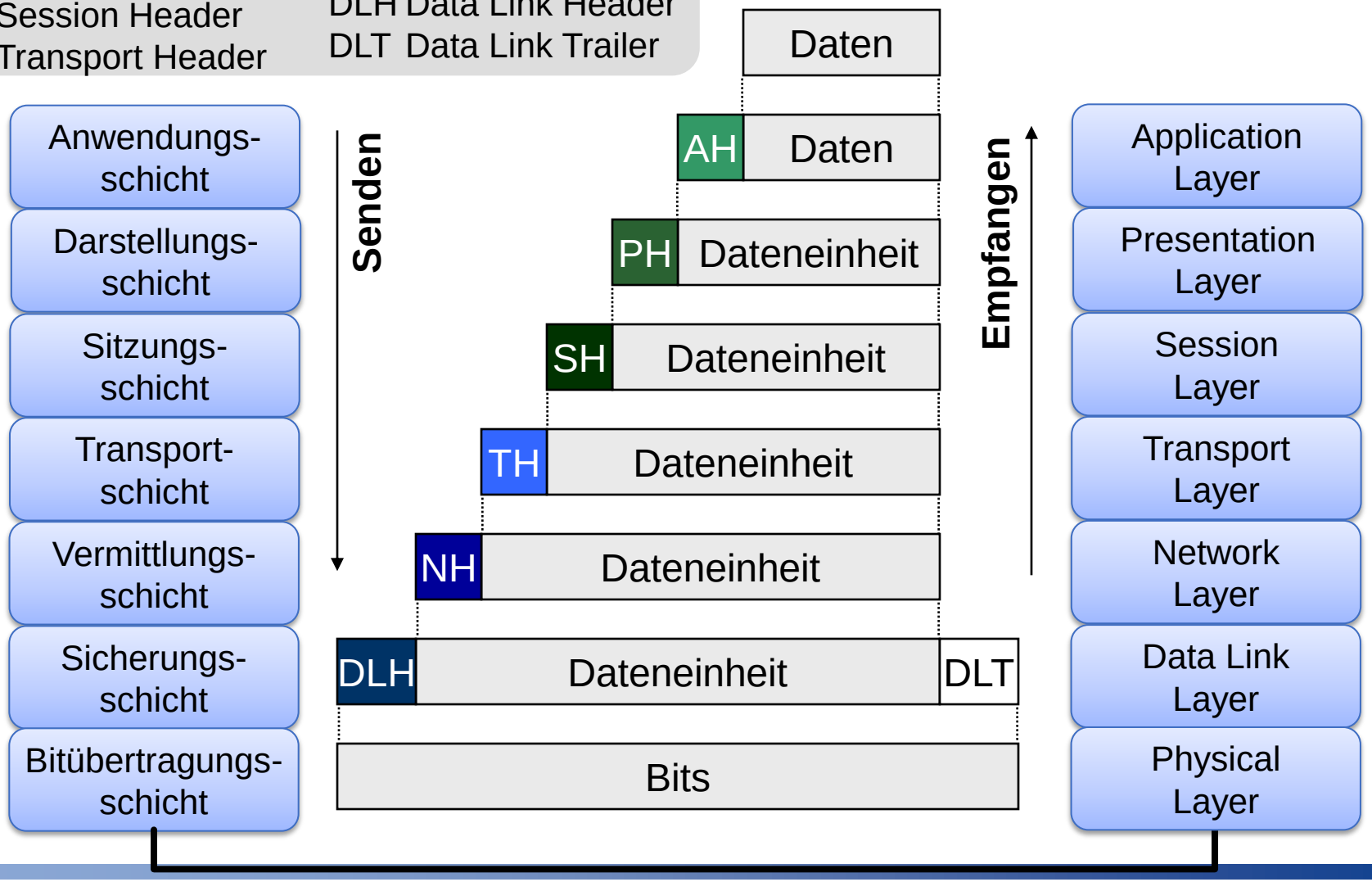
https://www.comsys.rwth-aachen.de

Themenübersicht

- **Datenkommunikation**
  - Einführung, Begriffe und allgemeine Grundlagen
  - Technische und nachrichtentechnische Grundlagen: Medien, Signale, Bandbreite, Leitungscodes und Modulation, Multiplexing
  - Lokale Netze: Strukturierung des Datenstroms, Fehlererkennung/ -behebung, Flusssteuerung, Medienzugriff, Ethernet
  - Vermittlungsschicht
    - Leitungsvermittlung
    - Internet und Internet-Protokolle:
    - Routing
  - Transportschicht
    - TCP und Co.
  - Anwendungsschicht

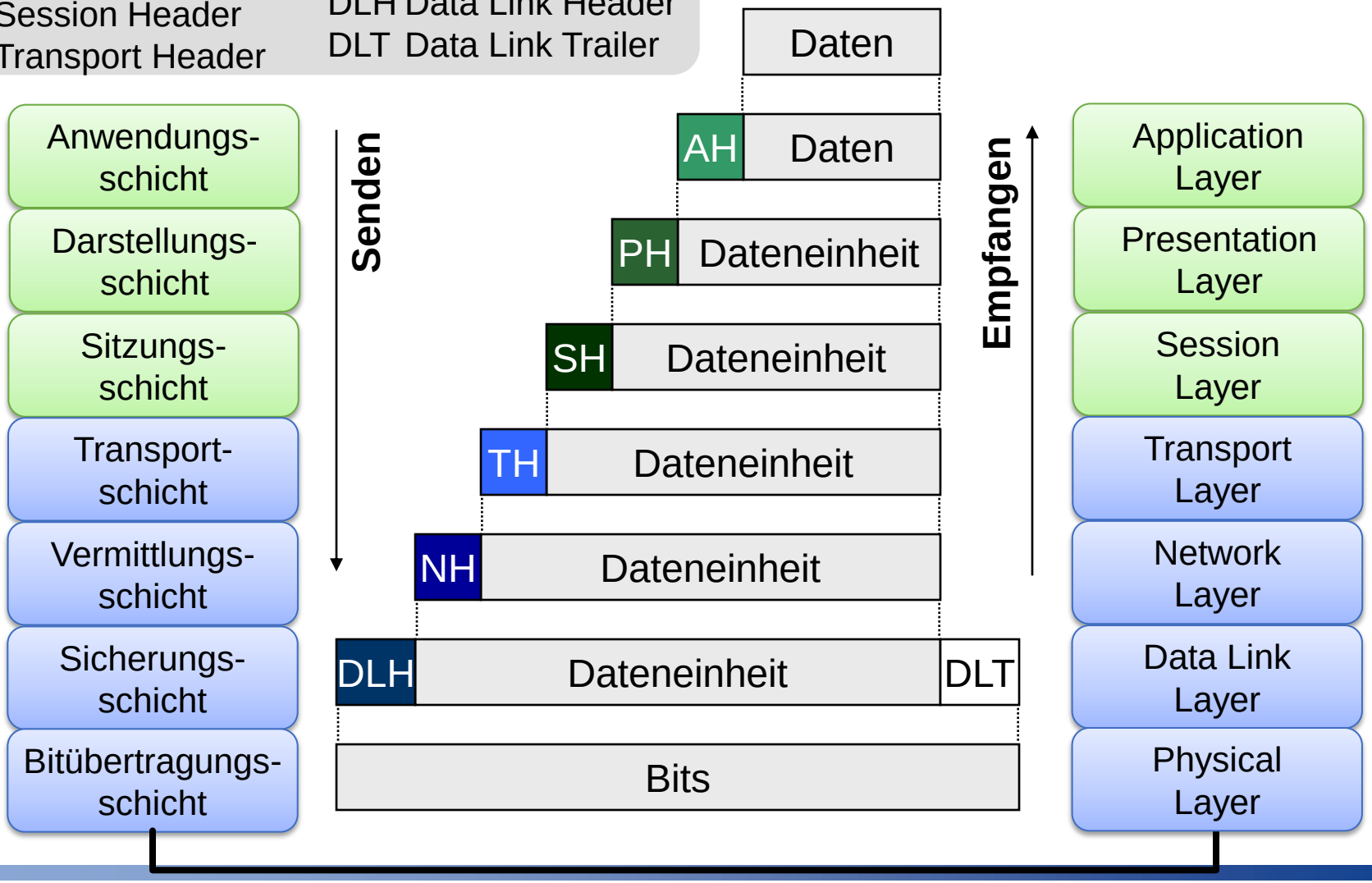
Einkapselung von Daten

AH Application Header
PH Presentation Header
SH Session Header
TH Transport Header
NH Network Header
DLH Data Link Header
DLT Data Link Trailer

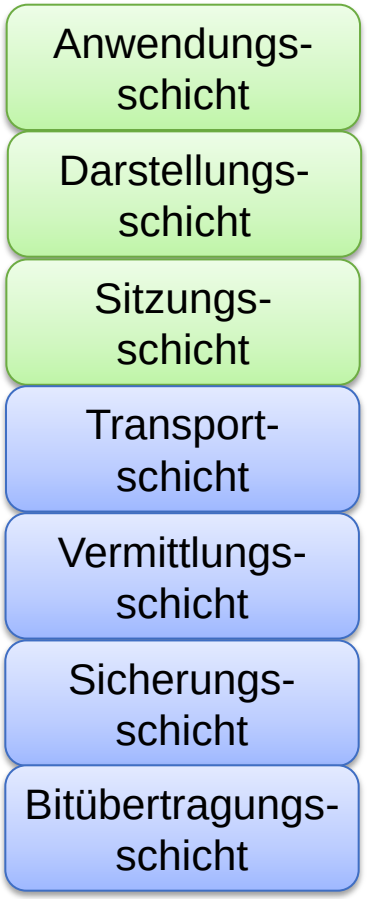


Einkapselung von Daten

AH Application Header
PH Presentation Header
SH Session Header
TH Transport Header
NH Network Header
DLH Data Link Header
DLT Data Link Trailer



Schicht 5 und 6

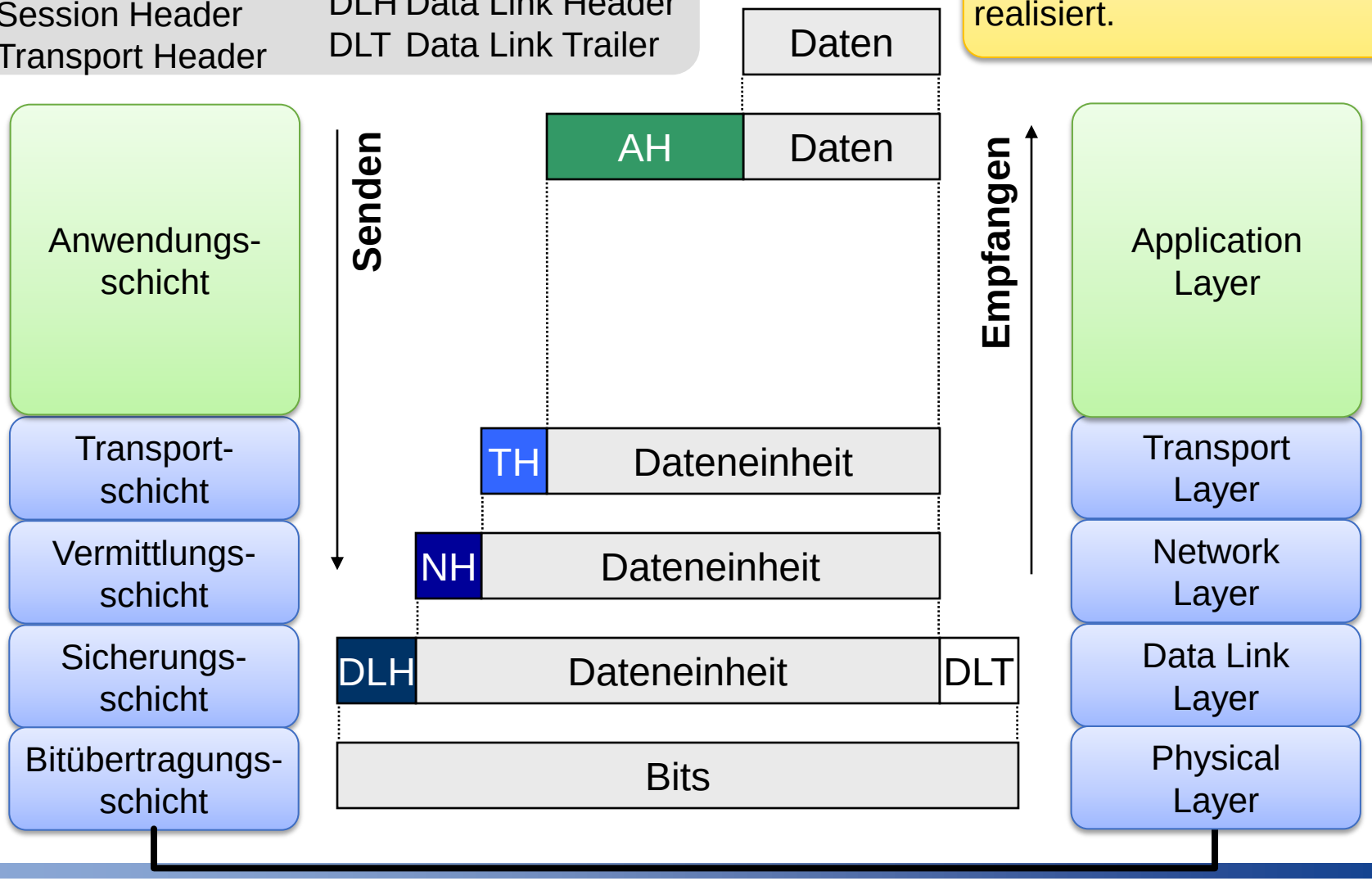


- **Sitzungsschicht:**
  - Endsysteme haben keinen gemeinsamen Zustand ihrer Verbindung
    - TCP hat einen Zustand (z.B. Receiver Window, Optionen)
    - Anwendung jedoch nicht
  - Beispiele für Zustandsinformationen
    - Position eines Videostreams, Codec, etc.
    - Konto, in das der Bankkunde eingeloggt ist
- **Darstellungsschicht:**
  - Endsysteme können gleiche Daten unterschiedlich darstellen
    - z.B. Character Encoding, Dateiformat
  - Darstellungsschicht stellt zusammenpassende Darstellung sicher

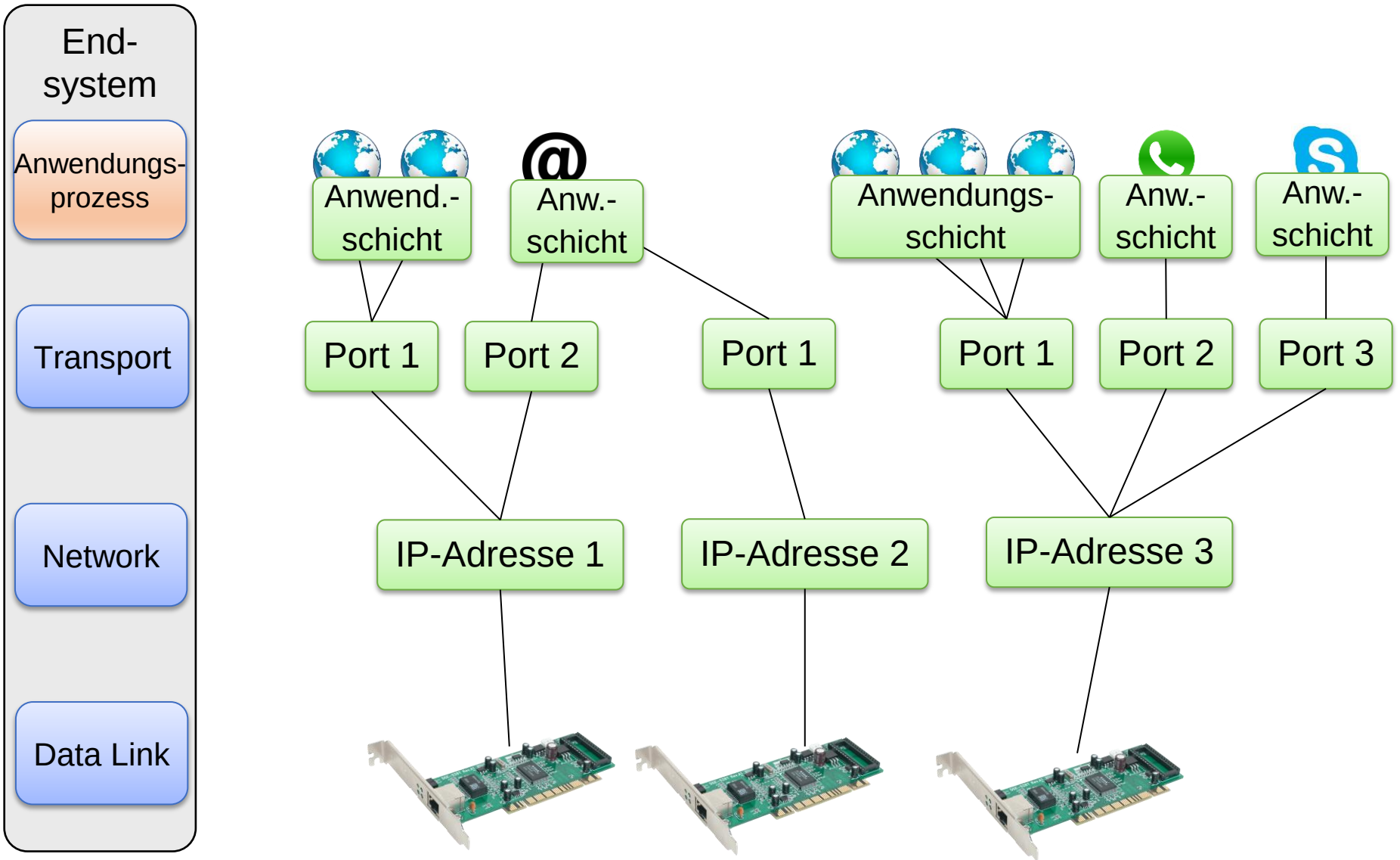
Einkapselung von Daten

AH Application Header
PH Presentation Header
SH Session Header
TH Transport Header
NH Network Header
DLH Data Link Header
DLT Data Link Trailer

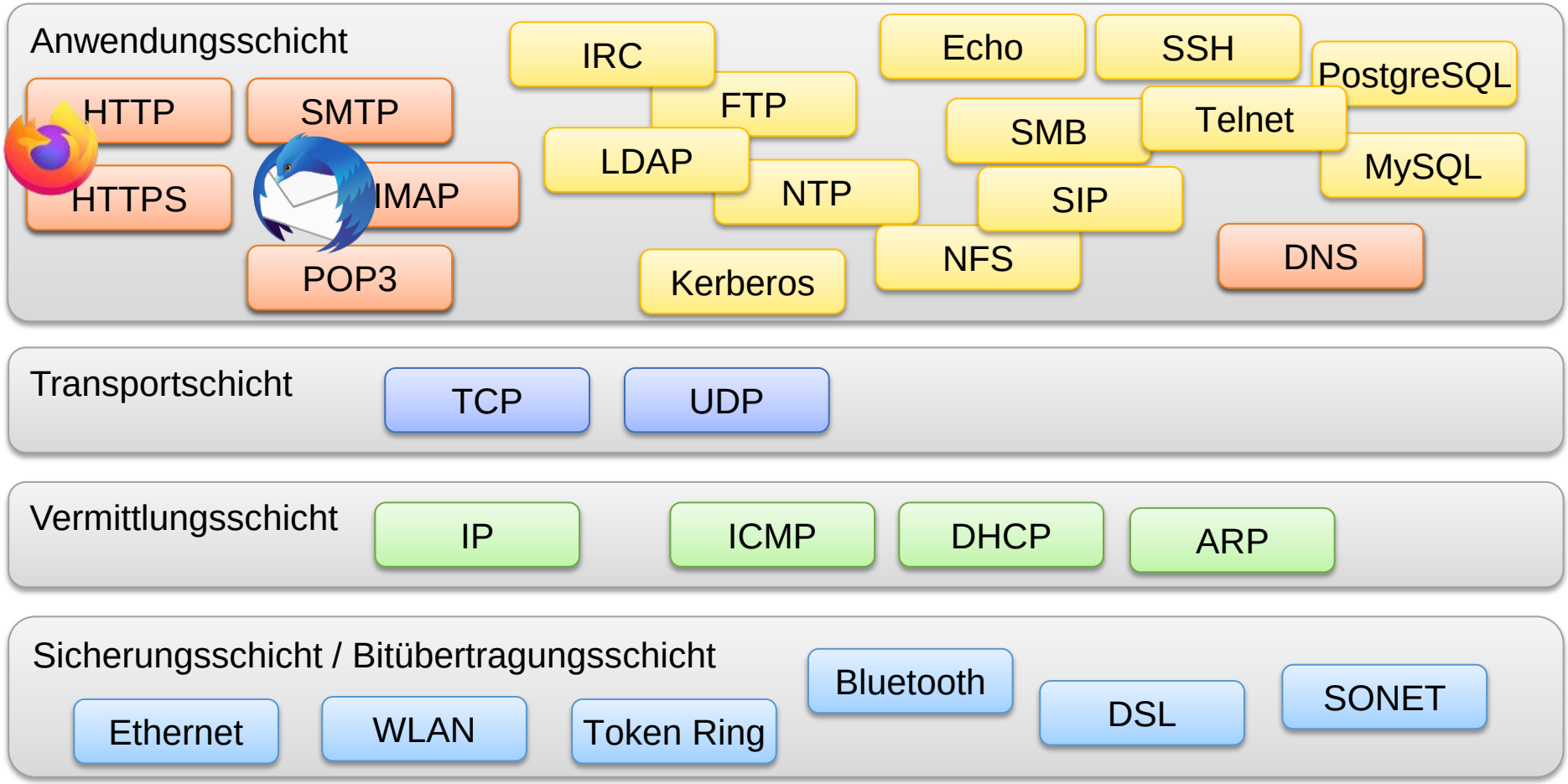
in der Praxis wird die Funktionalität von Schicht 5 und 6 gemeinsam mit dem Anwendungsprotokoll realisiert.



Rechner- und Prozessadressierung

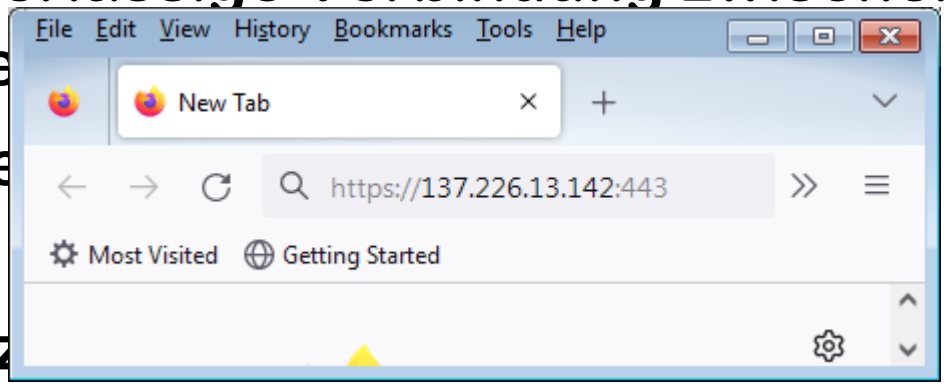


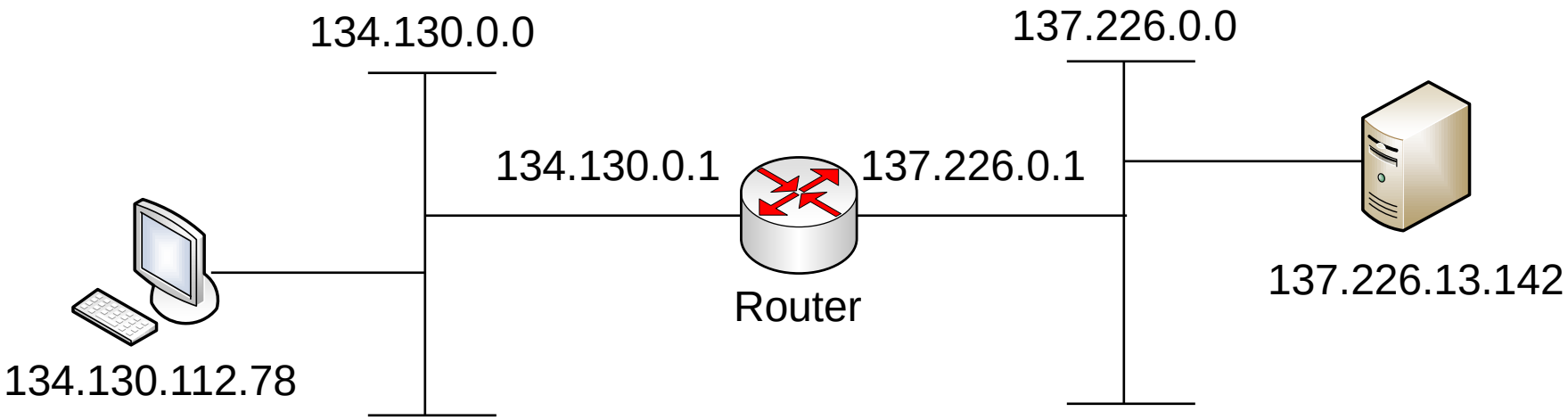
Protokolle der Anwendungsschicht



Wozu DNS?

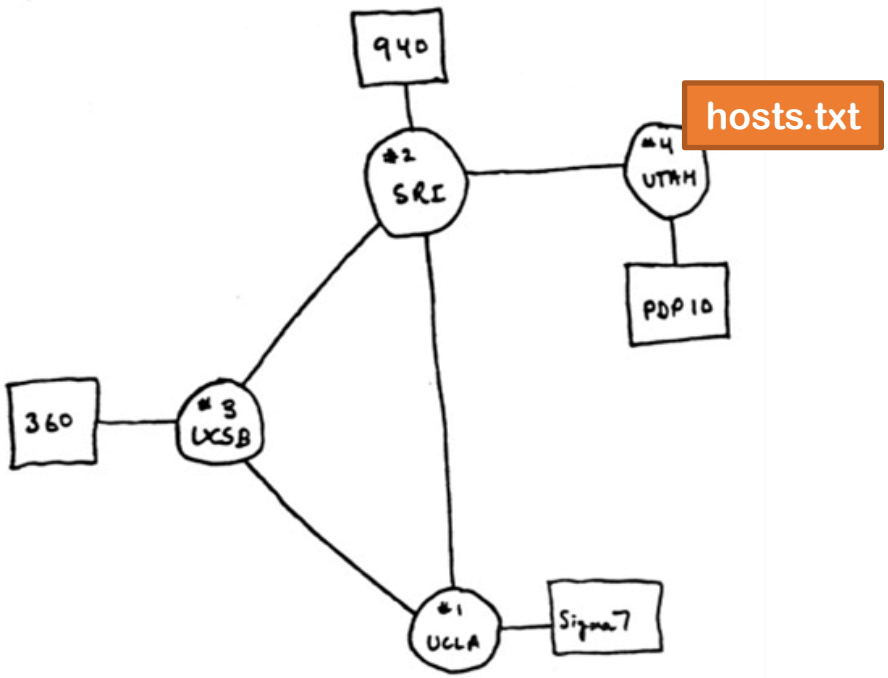
- TCP ermöglicht eine zuverlässige Verbindung zwischen zwei Instanzen der Anwe
- HTTP ermöglicht die Übe eine TCP-Verbindung
- Wozu benötigen wird jetz





www.comsys.rwth-aachen.de → 137.226.13.142

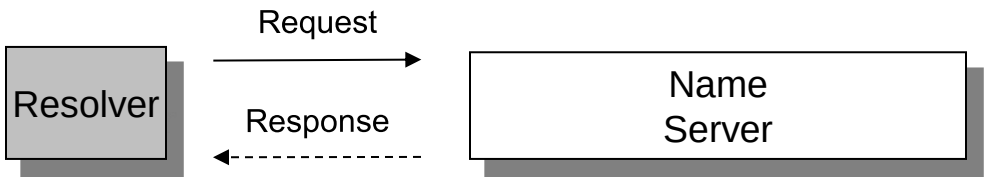
Namensauflösung im ARPANET



- Ursprünglich wurde eine Liste der Namen und IP-Adressen in einer Datei „hosts.txt“ gepflegt
- Diese Datei musste dann regelmäßig ausgetauscht werden
- nicht skalierbar!
- Datei existiert heute noch und kann für statische Festlegungen verwendet werden

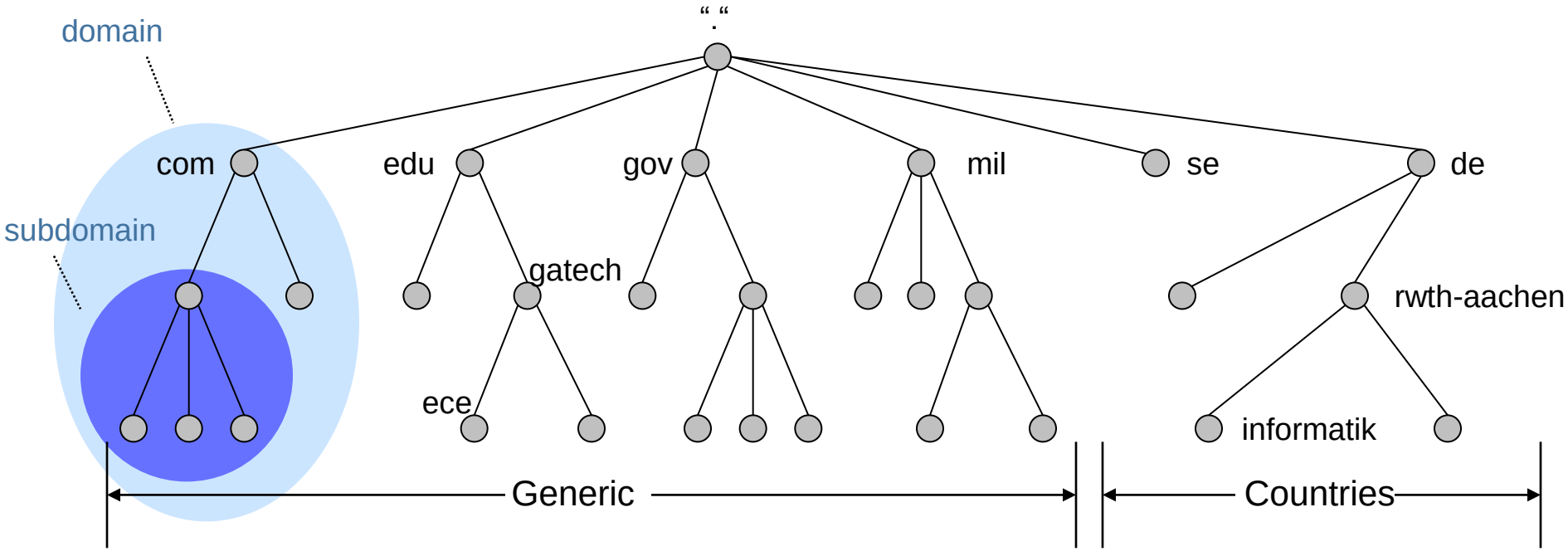
DNS als verteilte Datenbank

- **Verteilte Datenbank**
  - Einträge werden lokal verwaltet und können abgerufen werden
  - gute Skalierbarkeit!
- **Hierarchischer Namensraum**
  - ermöglicht Verteilung der Zuständigkeiten
- **Client-Server-Architektur ermöglicht weltweiten Zugang zur verteilten Datenbank**
  - *Name Server*: verwaltet einen Teil der Datenbank
  - *Resolver*: Client, der auf jedem Host läuft und Datenbankabfragen an die Name Server stellen kann



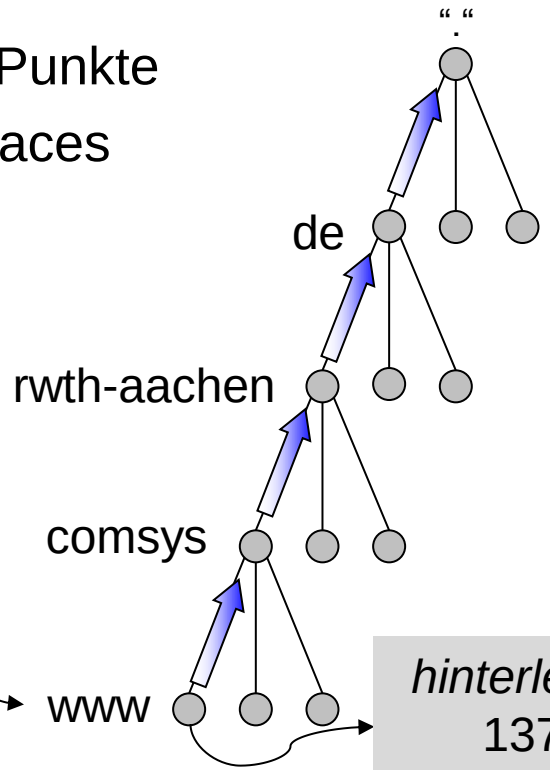
DNS Namespace

- **Alle Namen im Internet folgen einer Struktur**
  - Der Namespace ist ein *Baum*
  - Jeder Knoten hat ein *Label*
  - *Wurzel*: „.“
  - erste Ebene: *Top Level Domain* (TLD)



Domain-Namen

- **Domain-Name einer Organisation:**
  - Jeder kann einen Knoten im DNS-Namespace reservieren
    - und kontrolliert damit den gesamten Subtree darunter
- **Name eines Knotens:**
  - *Folge von Labels* getrennt durch Punkte bis zur Wurzel des DNS-Namespaces
- **Zuordnung zu IP-Adressen:**
  - Eintrag in der verteilten Datenbank, der dem Namen eine IP-Adresse zuteilt

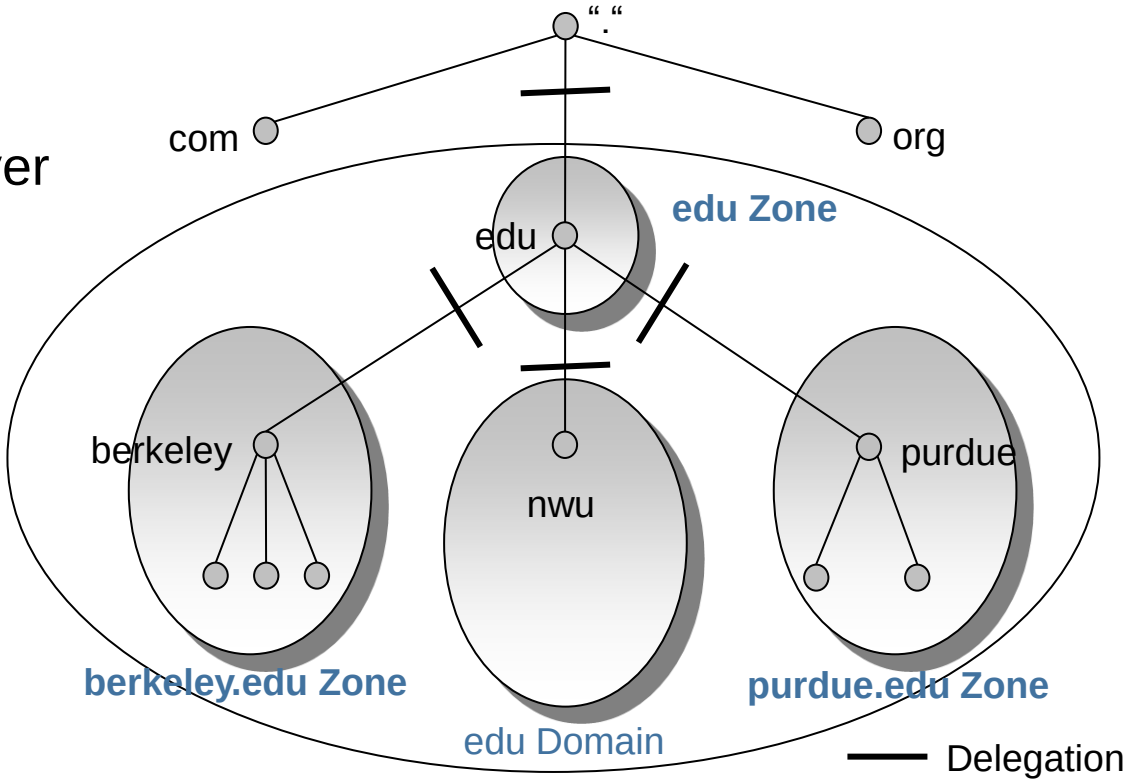


Domain-Name eines Rechners:
www.comsys.rwth-aachen.de.

hinterlegte IP-Adresse
137.226.13.142

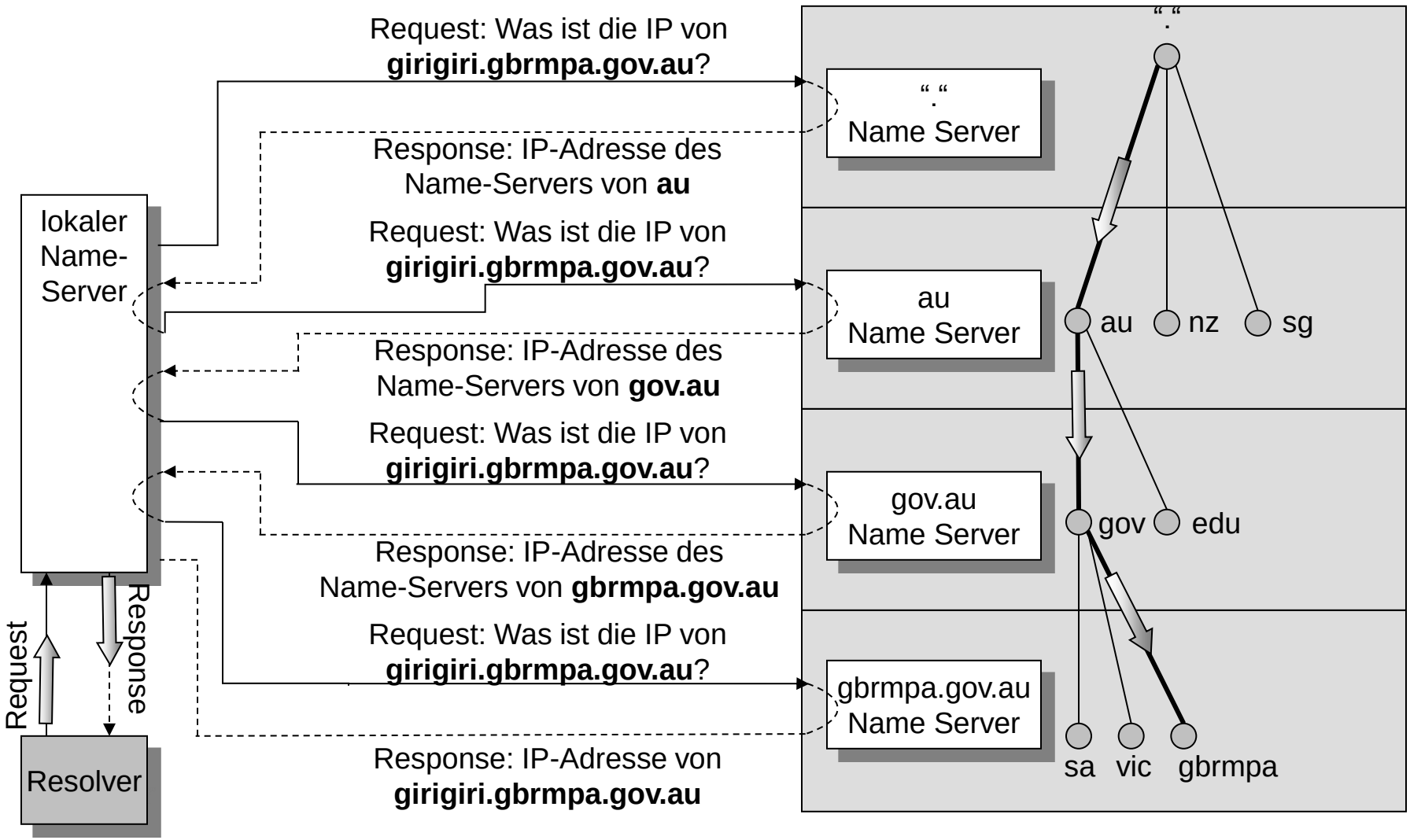
Domains und Zonen

- **Jeder *autoritative Name-Server* verwaltet nur einen (kleinen) Teil des DNS-Namespaces: eine *Zone***
  - Zone: Teil des Namespaces, der von einer Organisation verwaltet wird
  - Jede Zone muss über autoritative Name-Server verfügen
  - Diese Name-Server verwalten die Einträge dieser Zone und verweisen ggf. auf die zuständigen Zonen darunter



  - Frage: Wen kontaktieren wir, um eine IP-Adresse zu erfragen?

Namensauflösung



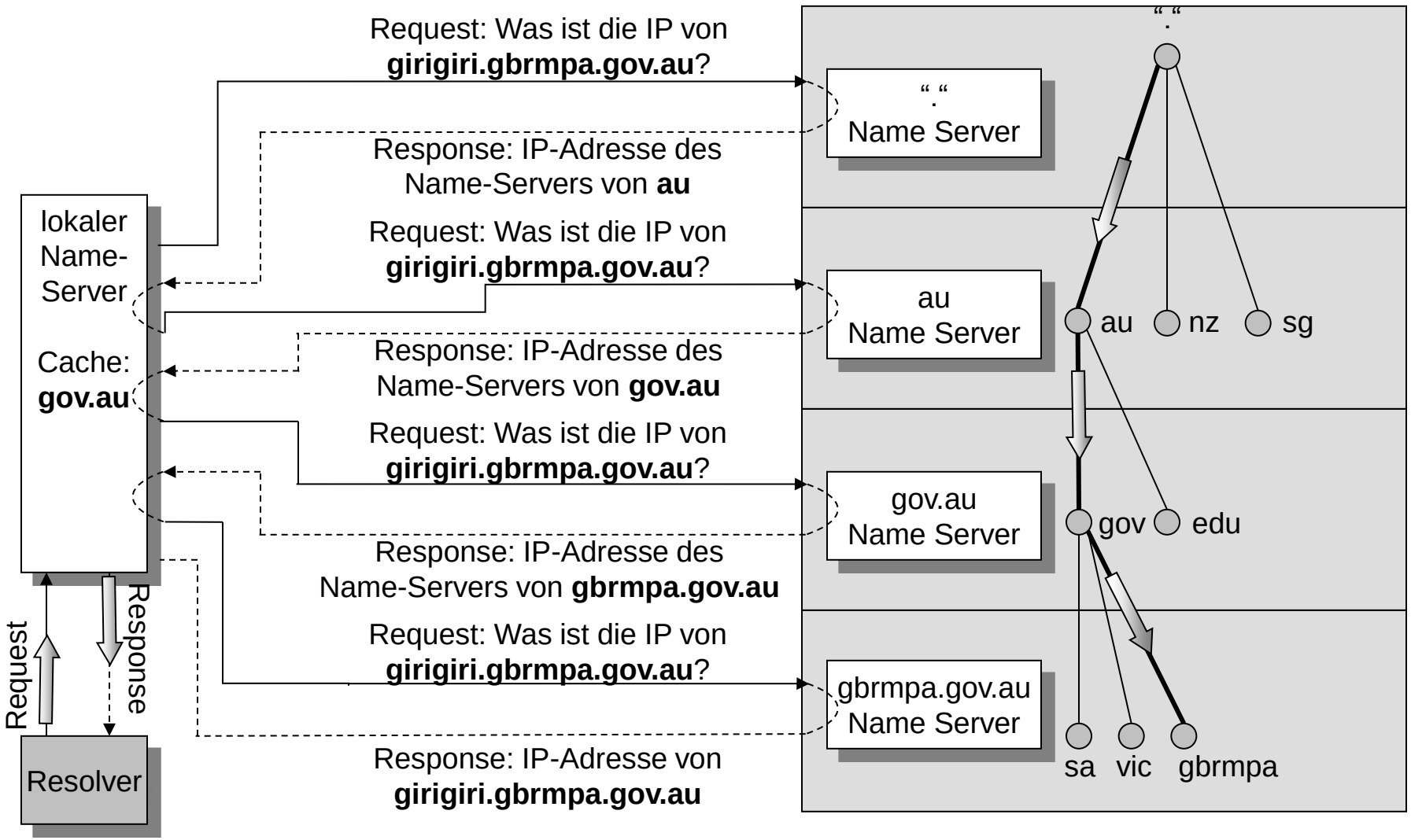
Namensauflösung

- **Zuordnung eines Namens zu einer IP-Adresse via DNS**
  - Resolver ist auf jedem Rechner installiert
    - Kennt die Adresse eines Name-Servers (typischerweise durch DHCP)
  - Resolver stellt Anfrage an lokalen Name-Server
  - Lokaler Name-Server ermittelt die Antwort und sendet sie zurück
    - aus der eigenen Datenbank
    - aus einem lokalen Cache
    - durch Anfrage anderer Server
  - Letzteres erfolgt durch die hierarchische Struktur des DNS von oben nach unten
- **Varianten der Auflösung:**
  - iterativ: Frage nacheinander die Layer des Baumes ab
  - rekursiv: Der befragte Server fragt die nächste Ebene an

Autoritativer Name-Server

- **Für jede Zone muss es (mindestens) einen autoritativen Name-Server geben**
- **Dieser Name-Server verwaltet die Einträge dieser Zone**
  - Er ist berechtigt, Änderungen vorzunehmen
  - Er ist die Instanz, die letztlich kontaktiert wird, wenn ein Eintrag aus dieser Zone benötigt wird
- **Eine Zone kann über mehrere autoritative Name-Server verfügen**
  - erhöht Ausfallsicherheit, ermöglicht Lastverteilung
  - Administrator muss sicherstellen, dass alle Name-Server dieselben Einträge ausliefern

Caching

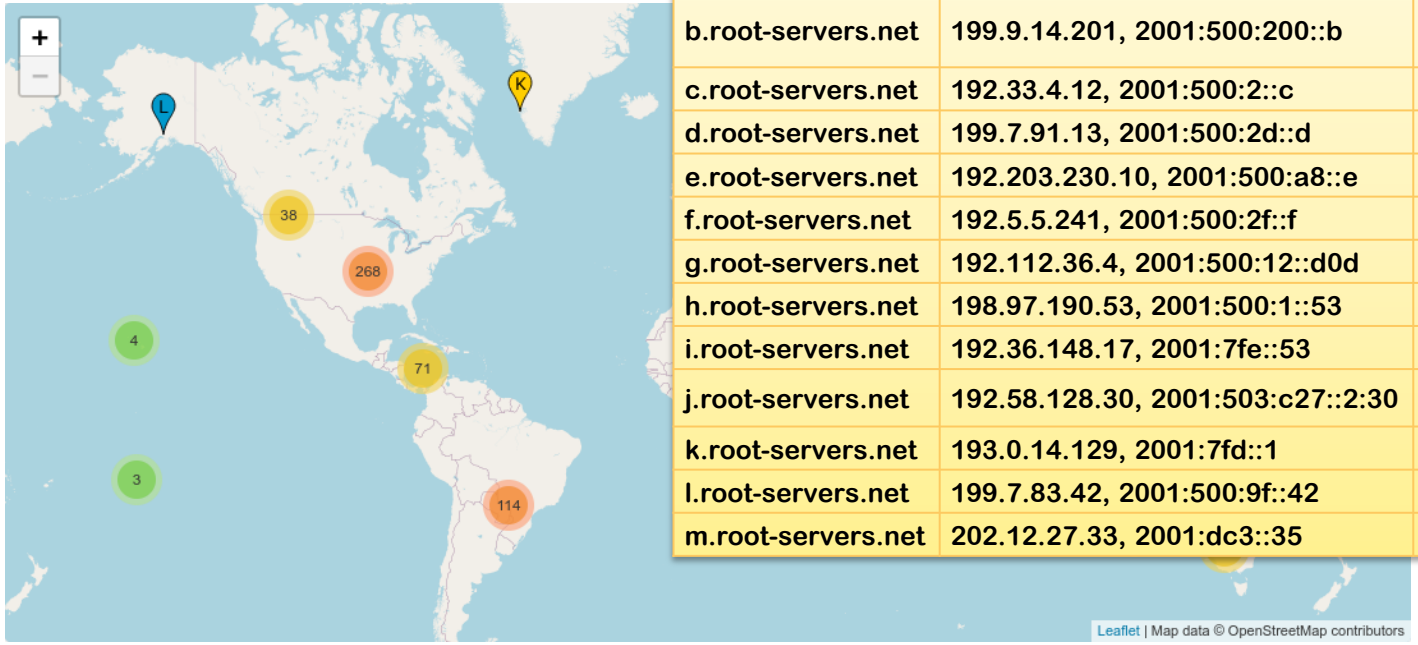

Request: Was ist die IP von **girigiri.gbrmpa.gov.au**?
Response: IP-Adresse des Name-Servers von **au**
Request: Was ist die IP von **girigiri.gbrmpa.gov.au**?
Response: IP-Adresse des Name-Servers von **gov.au**
Request: Was ist die IP von **girigiri.gbrmpa.gov.au**?
Response: IP-Adresse des Name-Servers von **gbrmpa.gov.au**
Request: Was ist die IP von **girigiri.gbrmpa.gov.au**?
Response: IP-Adresse von **girigiri.gbrmpa.gov.au**
lokaler Name-Server
Cache: **gov.au**
Request
Response
Resolver
" "
Name Server
au
Name Server
gov.au
Name Server
gbrmpa.gov.au
Name Server
au
nz
sg
gov
edu
sa
vic
gbrmpa

Caching

- **DNS-Antworten werden gecached**
  - schnellere Antwortzeiten, weniger Overhead
  - Aber: Was passiert, wenn der gespeicherte Eintrag nicht mehr aktuell ist?
- **Time-to-live für DNS-Einträge**
  - Jeder Eintrag bekommt eine Gültigkeitsdauer (in Sekunden)
    - Festlegung des Verwalters des Eintrags, wie lange Kommunikationspartner möglicherweise mit alten Werten arbeiten werden
  - Große TTL (Stunden bis Tage) zur Lastreduzierung
    - z.B., `` `dig NS de @a.root-servers.net` `` => 2 Tage
  - Kleine TTL (Sekunden bis Minuten) ermöglicht schnelle Änderungen
    - z.B., `` `dig A google.de @ns1.google.com` `` => 5 Minuten

Root Server

- **Zuständig für die Root-Zone**
- **13 logische Root-Server**
  - Schema: X.root-servers.net – X = a-m
  - derzeit (10.01.2023): 1607 physikalische Server, weltweit verteilt
  - Zugriff per Anycast



Hostname	IP Addresses	Operator
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	Verisign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California, Information Sciences Institute
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	Verisign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

Resource Records

- **Resource Record:**
  - Eintrag in der verteilten DNS-Datenbank
- **Struktur eines Resource Records:**

  `(label, TTL, class, type, value)`
  - `Label`: Name des Eintrags (z.B. Host-Name)
  - `TTL`: Zeit, wie lange der Eintrag gecached werden darf
  - `Class`: `IN` = Internet. DNS könnte auch für andere Zwecke eingesetzt werden.
  - `Type`: Typ des Eintrags (siehe nächste Folie)
  - `Value`: Wert des Eintrags (typabhängig)

Resource Record Types

Type	Beschreibung
SOA	start of authority (Informationen über den Verwalter des Teilbaums)
NS	Welcher Name-Server ist für diesen Teilbaum verantwortlich
A	IPv4-Adresse
AAAA	IPv6-Adresse
CNAME	Canonical name (effektiv ein Verweis auf einen anderen Namen)
MX	Mail-Server einer Domain
TXT	Beliebiger Text (wird auch für andere Zwecke verwendet, z.B. SPF)
SRV	Refers to a server which offers a certain service in the domain
RRSIG	Signature on a set of resource records
DNSKEY	Public key for zone
DS	Hash value of public key
TLSA	Crypthographic certificate for TLS
...	...

A/AAAA Record

- **A: name → IPv4 address**
  - **dig -t A www.google.de**

;; ANSWER SECTION:

www.google.de.	100	IN	A	172.217.22.227

  - **dig -t A www.windows.com**

windows.com.	3600	IN	A	23.100.122.175
windows.com.	3600	IN	A	191.239.213.197

…

- **AAAA: name → IPv6 address**
  - **dig -t AAAA www.google.de**

;; ANSWER SECTION:

www.google.de.	128	IN	AAAA

2a00:1450:4016:807::2003

CNAME: DNS Verweise

- **Name → Name**

- **Beispiel: tagesschau.de**
  - **dig www.tagesschau.de**

;; ANSWER SECTION:

www.tagesschau.de.	283	IN	CNAME	san.tagesschau.de.edgekey.net.
san.tagesschau.de.edgekey.net.	14963	IN	CNAME	e8178.g.akamaiedge.net.
e8178.g.akamaiedge.net.	20	IN	A	23.45.108.110

MX: E-Mail

- **E-Mail an foo@google.de**
- **Wer ist der Mail-Server für die Domain google.de?**
  - **dig -t MX google.de**

;; ANSWER SECTION:

google.de.	600	IN	MX	20 alt1.aspmx.l.google.com.
google.de.	600	IN	MX	30 alt2.aspmx.l.google.com.
google.de.	600	IN	MX	40 alt3.aspmx.l.google.com.
google.de.	600	IN	MX	10 aspmx.l.google.com.
google.de.	600	IN	MX	50 alt4.aspmx.l.google.com.

- **Danach A oder AAAA Lookup:**

aspmx.l.google.com.	293	IN	A	108.177.15.26
aspmx.l.google.com.	293	IN	AAAA	2a00:1450:400c:c07::1a

NS: Den Name-Server einer Domain finden

- **Beispiel google.de**
  - **dig -t NS google.de**

;; ANSWER SECTION:

google.de.	14806	IN	NS	ns3.google.com.
google.de.	14806	IN	NS	ns1.google.com.
google.de.	14806	IN	NS	ns4.google.com.
google.de.	14806	IN	NS	ns2.google.com.

Reverse DNS

- **Manchmal benötigen wir einen Namen zu einer IP-Adresse**
  - Beispiel traceroute: ICMP-Nachrichten enthalten nur IP-Adressen der Router
- **IP-Adressen sind ebenfalls hierarchisch aufgebaut**
  - Aber andersum als Domain-Namen!
- **Für Reverse-Lookups wird die IP-Adresse umgedreht und der PTR-Record in einer eigens dafür vorgesehenen Domain abgefragt**
  - .in-addr.arpa. für IPv4
  - .ip6.arpa. für IPv6

Reverse DNS

- **Beispiel google.de**
  - **www.google.de. 118 IN A 172.217.16.131**
  - **dig -t PTR 131.16.217.172.in-addr.arpa**

    ;; ANSWER SECTION:

    131.16.217.172.in-addr.arpa. 20875 IN PTR zrh04s06-in-f131.1e100.net.

    131.16.217.172.in-addr.arpa. 20875 IN PTR fra15s46-in-f3.1e100.net.

  - **www.google.de. 84 IN AAAA 2a00:1450:4001:808::2003**
  - **dig -t PTR 3.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.0.8.0.1.0.0.4.0.5.4.1.0.0.a.2.ip6.arpa**

    ;; ANSWER SECTION:

    3.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.0.8.0.1.0.0.4.0.5.4.1.0.0.a.2.ip6.arpa. 21265 IN PTR fra15s46-in-x03.1e100.net.

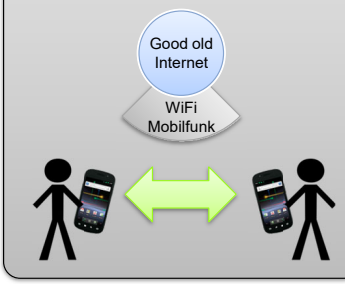
    3.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.0.8.0.1.0.0.4.0.5.4.1.0.0.a.2.ip6.arpa. 21265 IN PTR fra02s19-in-x03.1e100.net.

Zusammenfassung DNS

- **DNS ermöglicht Namensauflösung im Internet, d.h. insbesondere Zuordnung von Namen zu IP-Adressen**
- **DNS ist als hierarchische, verteilte Datenbank organsiert**
  - aufgeteilt in Zonen; autoritative Name-Server verwalten Zonen
- **Anfragen an die verteilte Datenbank können rekursiv oder iterativ erfolgen**
- **Caching ist wichtig für die Skalierbarkeit des Systems**
- **Verschiedene Resource Records existieren**
- **Weitere Funktionen von DNS:**
  - Mail-Server finden, Reverse-DNS
  - TLS-Zertifikate hinterlegen, SPF via TXT-Records, ...

Anwendungsschicht im Internet

Interaktion auf der Anwendungsebene



- Kommunikation zwischen Menschen
  - E-Mail
    - Als moderne Form des asynchronen Nachrichtenaustauschs
    - Oft auch zum Austausch von Dateien
  - Heute
    - Oft Foren, Chat, Diskussions-, Kollaborationsforen
      - Webbasiert
    - Messenger-Dienste
      - WhatsApp, Telegram, …
      - App-basiert auf Smartphone

Elektronische Post (E-Mail)

- **Eine der ersten Anwendungen im Internet:**
  - Kombination aus `SNDMSG` und `CPYNET`: Nachricht an `user@137.226.12.24`
    - `SNDMSG`: Nachricht (Text) in die private Nachrichtendatei des Ziel-Benutzers auf dem gleichen Rechner eintragen
    - `CPYNET`: Kopiere eine Datei von einem Rechner zu einem anderen (Internet-) Rechner
  - E-Mail nach RFC 822:
    - Füge eine (ASCII-)Nachricht in die Mail-Datei des adressierten Benutzers ein
    - Internationaler Austausch elektronischer Mitteilungen zwischen Personen

- **Wesentliche Charakteristik:**
  - Unterstützung des asynchronen Verhaltens von Sender und Empfänger(n)

  → Client-Server-Anwendung: Einführen von Mail-Server

Elektronische Post (E-Mail) war in der Vergangenheit diejenige Nutzungsform von Kommunikationsnetzen, die die Abläufe in der heutigen Arbeitswelt am stärksten beeinflusst hat.

Asynchron heißt hierbei, dass der sendende Teilnehmer unabhängig vom Verhalten des Empfängers senden kann und nicht vor, während oder nach dem Senden warten muss, bis der Empfänger bereit ist, die Nachricht zu empfangen. Entsprechend entsteht durch den Nachrichtenaustausch keine Synchronisierung von Sender und Empfänger.

Prinzipiell existieren zwei unterschiedliche Realisierungen der E-Mail-Technologie: SMTP (Simple Mail Transfer Protocol) im Internet und X.400 der OSI. Wie so oft konnte sich jedoch trotz der (theoretischen) Überlegenheit der OSI-Lösung der einfachere Internet-Standard auf breiter Front durchsetzen und wird heutzutage fast ausschließlich verwendet.

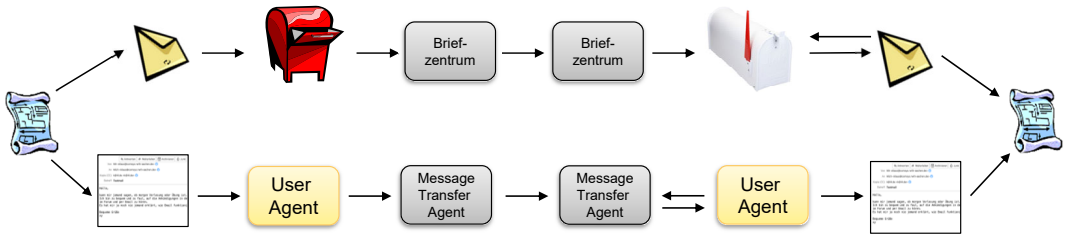
Elektronische Post (E-Mail)

- **Allgemeine Basisfunktionen:**
  - Erstellen von E-Mails
  - Übertragung zum Ziel
  - Benachrichtigung im Erfolgs-/Fehlerfall
  - Anzeige erhaltener Nachrichten
  - Speicherung von Nachrichten

- **Realisierung:**
  - Simple Mail Transfer Protocol (SMTP) im Internet
    - Transport von Emails zwischen Mailservern (vergleichbar mit Postdienst)
    - Ehemals X.400 bei OSI
  - IMAP, POP3, Webmail-Portale
    - Zugriff auf Mailserver (von Sender/Empfänger aus)

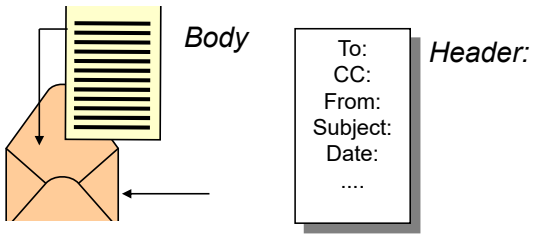
E-Mail: Allgemeines Modell

- **User Agent (UA)**
  - Lokales, grafik-/textorientiertes Programm
  - Ermöglicht Lesen und Versenden von E-Mail vom lokalen Rechner
  - z.B. Thunderbird, Mail, Outlook, Messenger
- **Message Transfer Agent (MTA)**
  - Hintergrundprozess
  - Zuständig für das Weiterleiten von E-Mails zum Zielrechner



SMTP: Format einer E-Mail

- **Umschlag (Envelope)**
  - Enthält alle Daten für den Transport der Mitteilung zu Empfänger/n („To:“)
  - Adressierung erfolgt mithilfe von DNS, z.B. klaus@comsys.rwth-aachen.de
  - Wird interpretiert von den MTAs
- **Kopfteil (Header)**
  - Enthält zusätzliche Felder wie z.B. Betreff, Kopie an („Subject:“, „CC:“)
  - Interpretiert von den User Agents
- **Hauptteil (Body)**
  - Enthält den eigentlichen Inhalt der Mitteilung (ursprünglich nur ASCII)



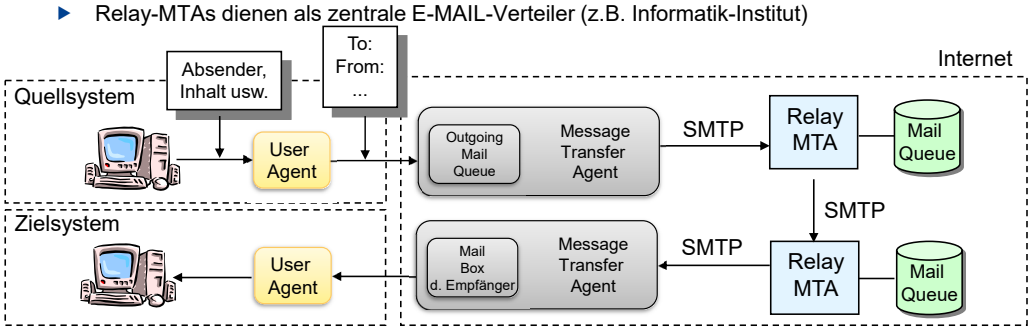
Mögliche Felder im Header einer SMTP-Nachricht:

To:	Empfänger-Adresse
CC:	Sekundäre Empfänger (CC=Carbon Copy, Wortabstammung vom Kohlepapier, das durch festes Aufdrücken des Stiftes eine (Kohle)kopie erstellt)
Bcc:	Weitere sekundäre Empfänger (werden beim Empfänger nicht angezeigt und daher auch aus dem SMTP-Header gelöscht)
From:	Adresse des Autors der Mitteilung
Sender:	Absenderadresse
Received:	Zeile, in die sich jeder MTA entlang es Weges zum Ziel einträgt
Return-Path:	Pfad zum Sender
Date:	Absendedatum und -zeit
Reply-to:	Antwortadresse
Message-ID:	Nachrichtennummer
In-Reply-To:	ggf. Nachrichtennummer der Nachricht, auf die geantwortet wurde
References:	Andere Nachrichtennummern auf die Bezug genommen wird

Keywords:	Vom Benutzer vergebene Schlüsselworte
Subject:	Titel („Betreff“) der Mitteilung

Internet Mail: Das SMTP-Modell

- **SMTP dient der E-Mail-Übermittlung**
  - zeichenorientiertes Protokoll, basierend auf 7-Bit-ASCII
  - nur wenige Kommandos, z.B. HELO, MAIL, RCPT, DATA, QUIT
- **UA erhält alle notwendigen Angaben vom Benutzer**
  - Mitteilung wird über Mail-Queue zum lokalen MTA übertragen
- **MTAs übertragen die Mitteilung zum Zielrechner**
  - Auslieferung einer E-Mail erfolgt über eine TCP-Verbindung (Port 25) zum Ziel-MTA (MTA unter UNIX: sendmail)
  - Relay-MTAs dienen als zentrale E-MAIL-Verteiler (z.B. Informatik-Institut)



SMTP: Beispielablauf

- Client öffnet TCP-Verbindung zum Mail Transfer Agent:
  - `telnet smtp-name.rwth-aachen.de 25`

```
220 mail-in-1a.itc.rwth-aachen.de ESMTP
                                    HELO myhostname.de
250 mail-in-1a.itc.rwth-aachen.de
                                    MAIL FROM dirk@i4.de
501 #5.5.2 syntax error 'MAIL FROM dirk@i4.de'
                                    MAIL FROM: dirk@i4.de
250 sender <dirk@i4.de> ok
                                    RCPT TO: klaus@comsys.rwth-aachen.de
250 recipient <klaus@comsys.rwth-aachen.de> ok
                                    DATA
354 go ahead
                                    Hallo zusammen
                                    (mehr Email-Body-Text)
                                    .
250 ok:  Message 33885312 accepted
                                    QUIT
221 mail-in-1a.itc.rwth-aachen.de
```


Der oben abgebildete Beispieldialog zeigt den Rechner myhostname.de in der Domäne i4.de in der Rolle des Senders S und den Mailserver smtp-name.rwth-aachen.de in der Rolle des Empfängers E.

Der Empfänger antwortet bei SMTP auf jedes Kommando des Senders mit einem dreistelligen Code, in dem das Ergebnis des Kommandos kodiert ist. Der Text hinter den Zahlen wird bei den meisten Antworten nicht vom Sender ausgewertet, sondern dient lediglich dazu, bei Fehlerfällen einem menschlichen Operator das Aufspüren von Fehlern zu erleichtern.

Mit dem Kommando HELO <sendername> identifiziert sich der Sender gegenüber dem Empfänger. Dieser antwortet darauf mit 250 <receivername>. Diese Begrüßung wird immer als erstes nach Aufbau der Transportverbindung ausgetauscht.

Das Kommando MAIL FROM: <senderuser> signalisiert dem Empfänger, dass eine Nachricht von dem Benutzer „senderuser“ übermittelt werden soll.

Mit dem Kommando RCPT TO: <recipientuser> wird der Empfänger der Nachricht spezifiziert. Es ist möglich, mit mehreren RCPT-Kommandos mehrere Empfänger zu spezifizieren.

Mit dem Kommando DATA wird der Beginn der Nachricht bekannt gegeben. Danach folgt die Nachricht Zeile für Zeile. Eine Zeile, die nur einen „.“ enthält, markiert das Ende der Nachricht.

Der Empfänger besteht den Empfang der Nachricht, legt sie in die lokale Mail-Queue und leitet sie später weiter oder legt sie in einem lokalen Postfach ab.

Mit dem Kommando QUIT wird der Verbindungsabbau eingeleitet.

Status- / Fehlermeldungen

Code	*Description*
Positive Completion Reply	
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
Positive Intermediate Reply	
354	Start mail input
Transient Negative Completion Reply	
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Format einer E-Mail


```
Received: from HERMES-MBX.comsys.rwth-aachen.de (2a00:8a60:1014::101c:a3f0:7884)
by HERMES-MBX.comsys.rwth-aachen.de (2a00:8a60:1014:0:dce8:101c:a3f0:7884)
with Microsoft SMTP Server (version=TLS1_2, cipher=TLS_ECDHE_RSA_WITH_AES_...)
id 15.1.2308.15
via Mailbox Transport; Sat, 14 Jan 2023 19:53:18 +0100
Received: from [IPV6:2a0a:a547:7a18:0:4ce:8d4a:d22:4c83]
by HERMES-MBX.comsys.rwth-aachen.de (2a00:8a60:1014:0:dce8:101c:a3f0:7884)
with Microsoft SMTP Server (version=TLS1_2, cipher=TLS_ECDHE_RSA_WITH_AES_256...)
id 15.1.2308.15; Sat, 14 Jan 2023 19:53:17 +0100
Message-ID: <19203bc6-105f-2880-88de-19d2f77f359c@comsys.rwth-aachen.de>
Date: Sat, 14 Jan 2023 19:53:17 +0100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0) Gecko/20100101
Thunderbird/102.5.1
From: Studi XYZ <xyz@rwth-aachen.de>
Subject: Testmail
To: Datkom <datkom@comsys.rwth-aachen.de>
CC: "test@test.de" <test@test.de>
Return-Path: Klaus@comsys.rwth-aachen.de

Hallo,

kann mir jemand sagen, ob morgen Vorlesung oder Uebung ist?

Ich bin zu bequem/zu faul, auf die Ankuendigungen in der Vorlesung,
im Forum und per Email zu verfolgen.
Es hat mir ja noch nie jemand erklaert, wie Email funktioniert.
Bequeme Gruesse
xyz
```

E-Mail Header

Header	Meaning
`To:`	Address of the main receiver (possibly several receivers or also a mailing list)
`Cc:`	Carbon copy, e-mail addresses of less important receivers
`Bcc:`	Blind carbon copy, a receiver which is *not* indicated to the other receivers
`From:`	Person who wrote the message
`Sender:`	Address of the actual sender of the message (possibly different to “From” person)
`Received:`	One entry per Message Transfer Agent on the path to the receiver
`Return Path:`	Path back to the sender (usually only e-mail address of the sender)
`Date:`	Transmission date and time
`Reply to:`	E-Mail address to which answers are to be addressed
`Message-Id:`	Clear identification number of the e-mail (for later references)
`In-Reply-to:`	Message-Id of the message to which the answer is directed
`References:`	Other relevant Message-Ids
`Subject:`	One line to indicate the contents of the message (is presented the receiver)

MIME (Multipurpose Internet Mail Extensions)

- **SMTP sieht nur einfache ASCII-Texte als Nachrichten vor (im Hauptteil)**
- **MIME erweitert den Hauptteil einer Nachricht um Formatinformationen.**
  - Hierzu werden neue Datenfelder für den Kopfteil einer Nachricht definiert:
  - `Content-Type:` definiert den Typ des Hauptteils.
    - Text, Multipart, Message, Application (Binary), Image, Audio, Video, …
  - `Content-Transfer-Encoding:` definiert die Transfer-Syntax, in der die Daten des Hauptteils übertragen werden.
    - `Base64`, `Quoted Printable`, `7 Bit`, `8 Bit` und `Binary`
- **Weitgehende Kompatibilität zur herkömmlichen Internet-Mail:**
  - Mit der Transfersyntax `Base64` ist es möglich, Binärdaten über Netze zu leiten, die nur 7-Bit-ASCII-Texte übertragen können
  - Die Transfersyntax `Quoted Printable` erlaubt nationale Sonderzeichen. Wird eine solche Mail von einem „alten“ Mail User Agent angezeigt, so werden nur diese Erweiterungen nicht korrekt angezeigt.

Während in der ursprünglichen Definition von SMTP (RFCs 821, 822) aus dem Jahre 1982 sowohl für den Kopf- wie auch für den Hauptteil einer Mitteilung nur der amerikanische 7-Bit-ASCII-Code verwendet werden konnte, wurde mit RFC 1522 die Multipurpose Internet Mail Extension (MIME) definiert, mit der beliebige Inhalte transportiert werden können. Hierzu werden einige neue Felder für den Kopfteil definiert, die den Aufbau und die Kodierung der Nachricht beschreiben:

Content-Type: Typ des Inhalts der Mitteilung: text, multipart, message, application, image, audio, video. Diese Klassen sind nochmals in mehrere Unterklassen aufgeteilt.

Content-Transfer Encoding: Übertragungssyntax (quoted-printable, Base64, 7 Bit, 8 Bit, Binary). Die ersten drei dieser Kodierungen werden auch von MTAs korrekt empfangen und weitergeleitet, die nur die ursprüngliche SMTP-Version implementieren.

Content-ID: Identifikator für den Inhalt

Content-Description: Textuelle Beschreibung des Inhalts

MIME-Version: Versionsnummer

Literatur:

[BorFre 92] N. Borenstein, N. Freed: RFC 1341 – MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies

[Postel 82] J. Postel: RFC 821: Simple Mail Transfer Protocol

[Crocker 82]: D.H. Crocker: RFC 822: Standard for the Format of ARPA Internet Text Mesages

E-Mail Header

- **RFC 2822: only suitably for messages of pure ASCII text without special characters.**
- **Nowadays demanded additionally:**
  - E-Mail in languages with special characters (e.g. French or German)
  - E-Mail in languages not at all using an alphabet (e.g. Japanese)
  - E-Mail not completely consisting of pure text (e.g. audio or video)
- **MIME keeps the RFC-2822 format, but additionally defines**
  - a structure in the Message Body (by using additional headers), and
  - coding rules for non-ASCII characters.

Header	Meaning
`MIME-Version:`	Used version of MIME is marked
`Content-Description:`	String which describes the contents of the message
`Content-Id:`	Clear identifier for the contents
`Content-Transfer-Encoding:`	Coding which was selected for the contents of the email (some networks understand e.g. only ASCII characters). Examples: base64, quoted-printable
`Content-Type:`	Type/Subtype regarding RFC 1521, e.g. text/plain, image/jpeg, multi-part/mixed

MIME - Standard Email-Kopf: Adressen & MIME-Ankündigung

Header

```
From: "Student XYZ" <x.y.z@rwth-aachen.de>
To: <datkom-solutions@discord.org>
Subject: Lösungen der Quiz-Fragen von heute
Date: Tue, 17 Jan 2023 16:11:41 +0100
Message-ID: <001701be075d$ca257d00$732a0d81@rwth-aachen.de>
MIME-Version: 1.0
Content-Type: MULTIPART/MIXED;
BOUNDARY= "8323328-2120168431-824156555=:325"
```

Data

```
--8323328-2120168431-824156555=:325
```

MIME-Inhalt

```
Content-Type: TEXT/PLAIN; charset=US-ASCII
Hi guys, here are the solution for today's quizzes.
```

```
--8323328-2120168431-824156555=:325
```

MIME-Inhalt

```
Content-Type: IMAGE/JPEG; name="picture.jpg"
Content-Transfer-Encoding: BASE64
Content-ID: <PINE.LNX.3.91.960212212235.325B@localhost>
Content-Description:
/9j/4AAQSkZJRgABAQEAlgCWAAD/2wBDAAEBAQEBAQEBAQEBAQIBAQEBAQIBAQECAgICAgICAgIDAwQDAwMDAwICAwQD
BAQEAgMFBQQEBQQEBAT/2wBDAQEBAQEBAQIBAQIEAwIDBAQEBA
[…]
KKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKK
KKACiiigAooooAKKKKACiiigAooooAD//Z
---8323328-2120168431-824156555=:325 –
```

MIME - Standard Email-Kopf: Adressen & 1. MIME Content

```
From: "Student XYZ" <x.y.z@rwth-aachen.de>
To: <datkom-solutions@discord.org>
Subject: Lösungen der Quiz-Fragen von heute
Date: Tue, 17 Jan 2023 16:11:41 +0100
Message-ID: <001701be075d$ca257d00$732a0d81@rwth-aachen.de>
MIME-Version: 1.0
Content-Type: MULTIPART/MIXED;
BOUNDARY= "8323328-2120168431-824156555=:325"

--8323328-2120168431-824156555=:325

Content-Type: TEXT/PLAIN; charset=US-ASCII
Hi guys, here are the solution for today's quizzes. <3
```

MIME Version

Struktur

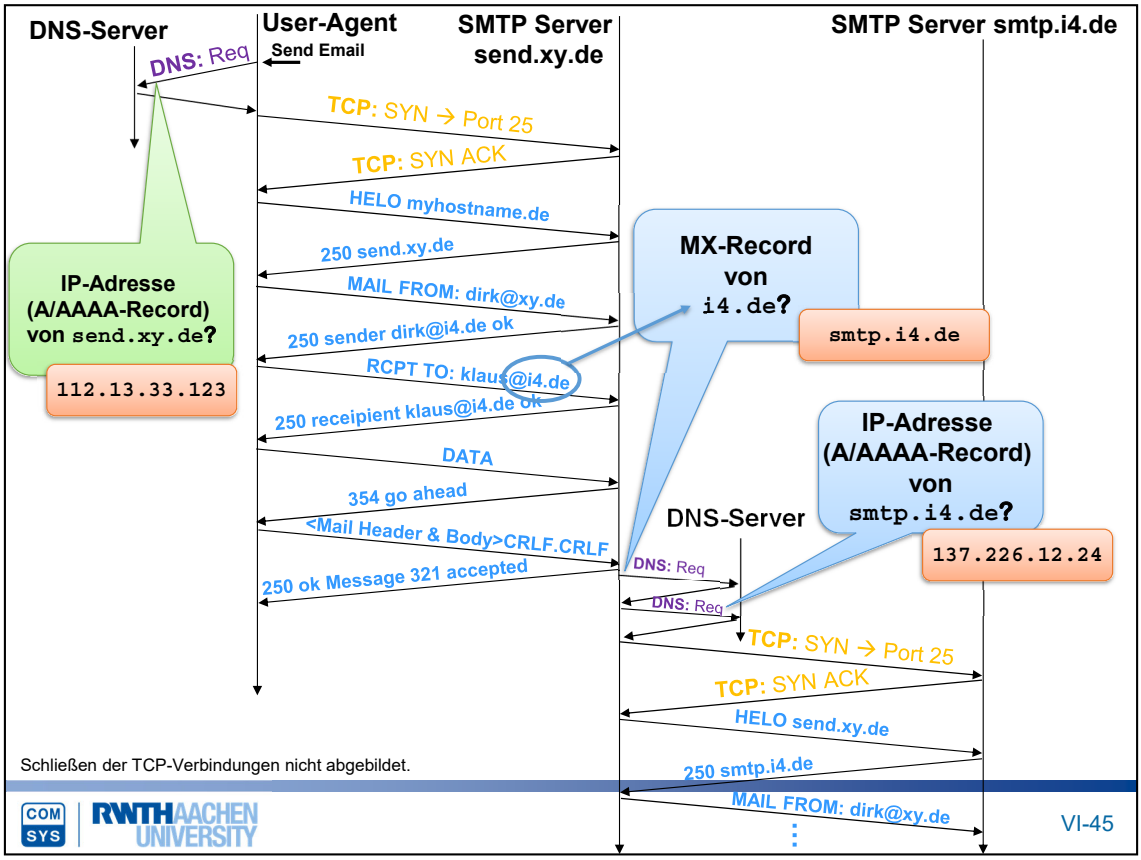
Trenner zwischen MIME Inhalten

Leerzeile: Beginn des Inhalts

MIME Inhalt 1: Text (in ASCII)

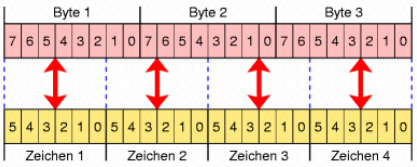
MIME – Parts

```
Hi guys, here are the solution for today's quizzes. <3

--8323328-2120168431-824156555=:325

Content-Type: IMAGE/JPEG; name="solutions.jpg"
Content-Transfer-Encoding: BASE64
Content-ID: <PINE.LNX.3.91.960212212235.325B@localhost>
Content-Description:
/9j/4AAQSkZJRgABAQEAlgCWAAD/2wBDAAEBAQEBAQEBAQEBAQIBAQEBAQIBAQECA
gICAgICAgIDAwQDAwMDAwICAwQDAwQEBAQEAgMFBQQEBQQEBAT/2wBDAQEBAQEBAQIBAQ
[…]
KKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACi
iigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAD//Z

---8323328-2120168431-824156555=:325 —
```


DNS-Server
User-Agent
Send Email
SMTP Server
send.xy.de
SMTP Server smtp.i4.de
DNS: Req
TCP: SYN → Port 25
TCP: SYN ACK
HELO myhostname.de
250 send.xy.de
MAIL FROM: dirk@xy.de
250 sender dirk@i4.de ok
RCPT TO: klaus@i4.de
250 receipient klaus@i4.de ok
DATA
354 go ahead
<Mail Header & Body>CRLF.CRLF
250 ok Message 321 accepted
IP-Adresse (A/AAAA-Record) von send.xy.de?
112.13.33.123
MX-Record von i4.de?
smtp.i4.de
DNS-Server
IP-Adresse (A/AAAA-Record) von smtp.i4.de?
137.226.12.24
DNS: Req
DNS: Req
TCP: SYN → Port 25
TCP: SYN ACK
HELO send.xy.de
250 smtp.i4.de
MAIL FROM: dirk@xy.de
Schließen der TCP-Verbindungen nicht abgebildet.
COM SYS
RWTH AACHEN UNIVERSITY
VI-45

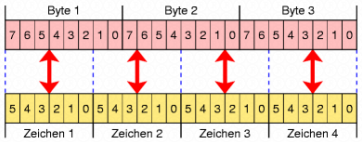
Base64 Encoding

- **Abbildung von beliebigen Byte-Folgen auf 7-Bit ASCII**
  - Gruppiere je drei Bytes in vier Gruppen à sechs Bit
    - 3x 8 Bit → 4x 6 Bit
    - Fülle Bytes mit 0,1 oder 2 Null-Bytes auf, so dass Byteanzahl durch drei teilbar ist
    - Kodiere die 3x6-Bit-Blöcke gemäß Tabelle
    - Kennzeichne Zahl der Padding-Blöcke mit einem „=„ pro Padding-Block



Wert			Zeichen	Wert			Zeichen	Wert			Zeichen	Wert			Zeichen
dez.	binär	hex.		dez.	binär	hex.		dez.	binär	hex.		dez.	binär	hex.	
0	000000	00	A	16	010000	10	Q	32	100000	20	g	48	110000	30	w
1	000001	01	B	17	010001	11	R	33	100001	21	h	49	110001	31	x
2	000010	02	C	18	010010	12	S	34	100010	22	i	50	110010	32	y
3	000011	03	D	19	010011	13	T	35	100011	23	j	51	110011	33	z
4	000100	04	E	20	010100	14	U	36	100100	24	k	52	110100	34	0
5	000101	05	F	21	010101	15	V	37	100101	25	l	53	110101	35	1
6	000110	06	G	22	010110	16	W	38	100110	26	m	54	110110	36	2
7	000111	07	H	23	010111	17	X	39	100111	27	n	55	110111	37	3
8	001000	08	I	24	011000	18	Y	40	101000	28	o	56	111000	38	4
9	001001	09	J	25	011001	19	Z	41	101001	29	p	57	111001	39	5
10	001010	0A	K	26	011010	1A	a	42	101010	2A	q	58	111010	3A	6
11	001011	0B	L	27	011011	1B	b	43	101011	2B	r	59	111011	3B	7
12	001100	0C	M	28	011100	1C	c	44	101100	2C	s	60	111100	3C	8
13	001101	0D	N	29	011101	1D	d	45	101101	2D	t	61	111101	3D	9
14	001110	0E	O	30	011110	1E	e	46	101110	2E	u	62	111110	3E	+
15	001111	0F	P	31	011111	1F	f	47	101111	2F	v	63	111111	3F	/

Base64 Encoding

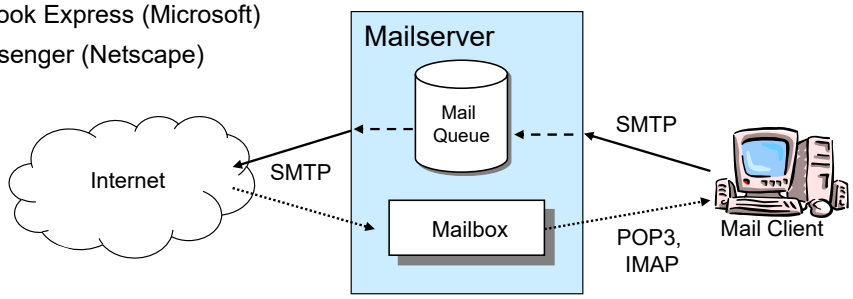


- **Beispiel:**
  - Quelltext (hex): 0C 04 CA 38 CA E8 72 4B
  - Binär (à 8 Bit): 0000 1100 0000 0100 1100 1010
    0011 1000 1100 1010 1110 1000
    0111 0010 0100 1011 0000 0000
  - Binär (à 6 Bit): 000000 010010 001101 000101
    011001 111000 100110 101011
    110011 011110 111100 000000
  - Base64 encoded:
    DATKOMrocks=

Wert			Zeichen	Wert			Zeichen	Wert			Zeichen	Wert			Zeichen
dez.	binär	hex.		dez.	binär	hex.		dez.	binär	hex.		dez.	binär	hex.	
0	000000	00	A	16	010000	10	Q	32	100000	20	g	48	110000	30	w
1	000001	01	B	17	010001	11	R	33	100001	21	h	49	110001	31	x
2	000010	02	C	18	010010	12	S	34	100010	22	i	50	110010	32	y
3	000011	03	D	19	010011	13	T	35	100011	23	j	51	110011	33	z
4	000100	04	E	20	010100	14	U	36	100100	24	k	52	110100	34	0
5	000101	05	F	21	010101	15	V	37	100101	25	l	53	110101	35	1
6	000110	06	G	22	010110	16	W	38	100110	26	m	54	110110	36	2
7	000111	07	H	23	010111	17	X	39	100111	27	n	55	110111	37	3
8	001000	08	I	24	011000	18	Y	40	101000	28	o	56	111000	38	4
9	001001	09	J	25	011001	19	Z	41	101001	29	p	57	111001	39	5
10	001010	0A	K	26	011010	1A	a	42	101010	2A	q	58	111010	3A	6
11	001011	0B	L	27	011011	1B	b	43	101011	2B	r	59	111011	3B	7
12	001100	0C	M	28	011100	1C	c	44	101100	2C	s	60	111100	3C	8
13	001101	0D	N	29	011101	1D	d	45	101101	2D	t	61	111101	3D	9
14	001110	0E	O	30	011110	1E	e	46	101110	2E	u	62	111110	3E	+
15	001111	0F	P	31	011111	1F	f	47	101111	2F	v	63	111111	3F	/

Internet-Mail: Verwaltung durch Mailserver

- E-Mail wird meist zentral über einen Mailserver abgewickelt (Relay-MTA)
- Mittels **POP3** (Post Office Protocol 3) holt der Client die vom Mailserver empfangenen und in der Mailbox gespeicherten Meldungen ab
  - einfache Funktionalität
- **IMAP** (Internet Message Access Protocol) dient dazu, die Nachrichten zentral auf einem Mailserver zu verwalten
  - IMAP erlaubt erweiterte Kommandos (z.B. Filterung)
- Beispiele für Mail-Client-Programme
  - Outlook Express (Microsoft)
  - Messenger (Netscape)



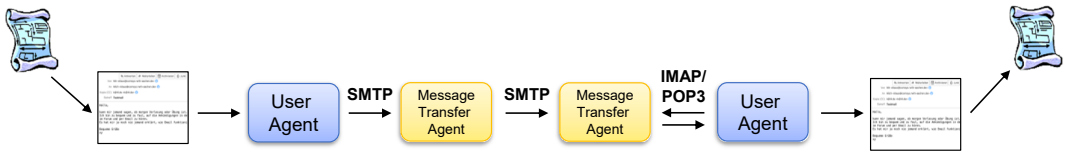
Werden Nachrichten zentral von einem Mailserver verschickt/empfangen (was meistens der Fall ist), so erfolgt der E-Mail-Empfang über die Protokolle POP3 (Post Office Protocol Version 3) oder IMAP (Internet Message Access Protocol).

Der wesentliche Vorteil von IMAP gegenüber POP3 besteht darin, dass es IMAP ermöglicht, Nachrichten von unterschiedlichen Rechnern aus zu lesen, da alle Nachrichten zentral verwaltet werden. Bei POP3 entsteht dagegen der Nachteil, dass man die Nachrichten nur auf dem Rechner, der zum Abrufen der E-Mails verwendet wurde, lokal vorliegen hat. Ein zweiter Vorteil von IMAP ist, dass man nur Teile der Nachricht (z.B. nur den Kopf) lesen kann, wodurch man sich das Herunterladen eines eventuell sehr großen Hauptteils sparen kann.

Die verschiedenen, heute auf dem Markt erhältlichen, Mail-Client-Programme bieten i.W. die gleiche Funktionalität an. In der Regel unterstützen diese sowohl POP3 als auch IMAP. Als Beispiele seien hier Mozilla Thunderbird, Apple Mail oder Microsoft Outlook genannt. Viele Nutzer rufen ihre E-Mails auch über eine Web-Applikation (z.B. Outlook Web Access) ab, wobei dann effektiv das HTTP-Protokoll zum Einsatz kommt.

EMail-Zugriffsprotokolle

- **SMTP: Zustellung zu und Speicherung in Empfänger-Server**
- **Email-Zugriffsprotokoll: Empfang vom eigenen Mail-Server**
  - POP: Post Office Protocol [RFC 1939]
    - Client verbindet zu POP3-Server über TCP-Port 110
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Client verbindet zu TCP-Port 143
  - HTTP: Gmail, OutlookWebAccess (OWA), etc.



POP vs. IMAP

- **POP3**
  - Zustandsloser Server
  - User Agent ruft E-Mails vom Server ab
  - Mails werden normalerweise vom Server gelöscht
  - Die letzten Änderungen befinden sich im User Agent
  - Einfaches Protokoll (`list`, `retr`, `del` innerhalb einer POP-Sitzung)
- **IMAP4**
  - Zustandsabhängiger Server
  - UA- und Server-Verarbeitung
  - Server sieht Ordner usw., die für User Agents sichtbar sind
  - Änderungen auf dem Server sichtbar
  - Komplexes Protokoll
  - Geeignet für gleichzeitige Mail-Verwaltung auf mehreren UAs

E-Mail-Zugriffsmodelle

- **Zugriffsmodelle:**
- **(siehe RFC1733)**
  - Offline
    (Mailbox-Verarbeitung beim Client)
  - Online
    (Mailbox-Zugriff z.B. über NFS (Network File Service))
  - Disconnected
    (Client übergibt Änderungen an Server)

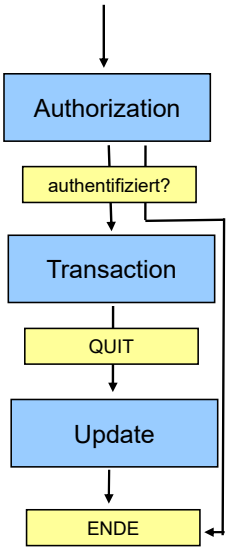
Eigenschaft	Offline	Online	Disc.
Unterstützung mehrerer MTAs	Nein	Ja	Ja
Kurze Verbindungszeit zum Server	Ja	Nein	Ja
Geringe Anforderung an Server-Ressourcen	Ja	Nein	Nein
Geringe Anforderung an Client-Ressourcen	Nein	Ja	Nein
Mehrere entfernte Mailboxen	Nein	Ja	Ja
Geringe Zugriffslatenzzeit	Nein	Ja	Nein
Offline-Mailbox-Verarbeitung	Ja	Nein	Ja

⇨ **POP3: Offline-Modell**

⇨ **IMAP4: kann alle drei Zugriffsmodelle unterstützen**

POP3: Übersicht

- **POP3 bietet keine zentrale Mailverwaltung**
  - z.B. keine Auswahl einzelner Nachrichten
- **Drei Zustände:**
  - Authorization
    - Benutzer muss sich mit Name & Passwort ausweisen
    - einfache Kommandos: `USER, PASS`
    - Passwortübertragung normalerweise unverschlüsselt!
    - Die verschlüsselte Übertragung des Passwortes mit Kommando **APOP** wird nur selten unterstützt.
  - Transaction
    - Eigentliche „Arbeitsphase“, d.h. einfache „Verwaltung“ eingegangener E-Mails
    - Kommandos: z.B. `STAT, LIST, RETR, DELE`
    - Beenden dieser Phase durch `QUIT`
  - Update
    - Fortschreiben des Zustands (z.B. Löschen von E-Mails)
    - Danach Beendigung der Sitzung



Kommandos (Auszug):

User - Benutzername

Pass - Passwort

Stat(us) - Anzahl und Größe der gesamten eingegangenen E-Mail

List - Größe jeder einzelnen E-Mail (Ohne „Betreff“-Zeile!)

Retr(ieve) - Abholen der Mail vom Server

Dele(te) - Markierung zum Löschen einer Mail

POP3 Protokoll

Authorizing phase

- `user` Benutzer
- `pass` Passwort
- `+OK` oder `-ERR` sind mögliche Server-Antworten

Transaktionsphase

- `list` für die Auflistung der Nachrichtennummern und der Nachrichtengrößen
- `retr` zum Anfordern einer Nachricht anhand ihrer Nummer
- `dele` löscht die entsprechende Nachricht

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged in
```

```
C: list
S: 1    498
S: 2    912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 2 contents>
S: .
C: dele 2
C: quit
S: +OK
```

IMAPv4: Übersicht

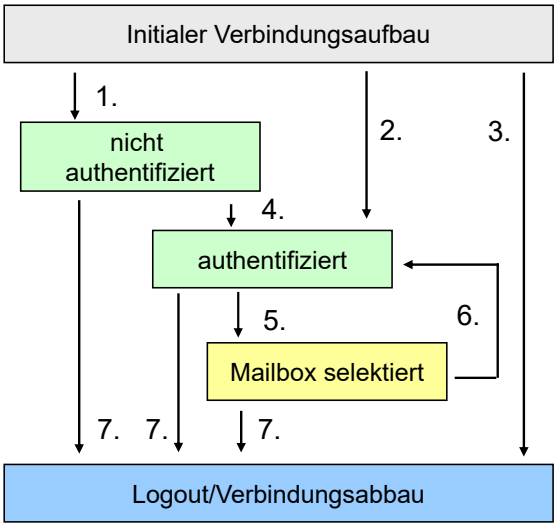
- **Vielzahl an Operationen zur zentralen E-Mail-Verwaltung**
  - Speicherung zentral auf Server oder lokal auf Seite des Clients (*online/offline operation*) mit Synchronisation
  - speziell geeignet für Nutzer mehrerer Rechner
- **Kommandos (Auszug):**
  - Authentifizierung (un- oder verschlüsselt) durch `LOGIN/ AUTHENTICATE`
  - Auswahl, Erzeugen und Verwalten von E-Mail-Verzeichnissen („Mailbox“) durch `SELECT, CREATE, RENAME, ...`
  - Nachrichtensuche durch `SEARCH`
  - Auswahl/Abholen von einzelnen Nachrichten(-teilen) durch `FETCH`
- **Weitere Funktionen:**
  - Setzen von Flags (z.B. Nachricht gelesen/beantwortet, ...)
  - Filterung (Automatisches Einsortieren, Weiterleiten, Löschen, Beantworten, ...) auf Seite des Clients realisiert

Akutelle Spezifikation: RFC 2060: Internet Message Access Protocol - Version 4rev1

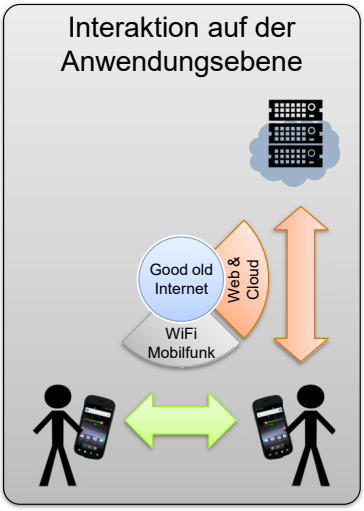
IMAPv4: Zustandsdiagramm des Servers

Ein IMAP-Server durchläuft die Zustände nicht authentifiziert, authentifiziert, Mailbox selektiert und Verbindungsabbau:

1. Herkömmlicher Verbindungsaufbau („OK-Greeting“)
2. Verbindungsaufbau „ohne“ Authentifizierung („PREAUTH-Greeting“)
   - Authentifizierung extern erfolgt
3. Verbindungszurückweisung
4. Erfolgreiche Authentifizierung
5. Erfolgreiche Auswahl einer Mailbox
6. Schließen einer Mailbox / Fehler bei Auswahl einer Mailbox
7. Verbindungsabbau/Logout



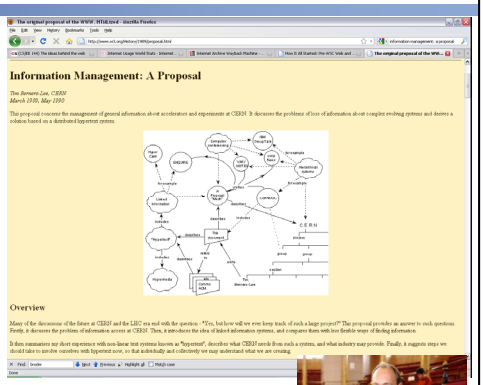
Evolution von Kommunikationssystemen

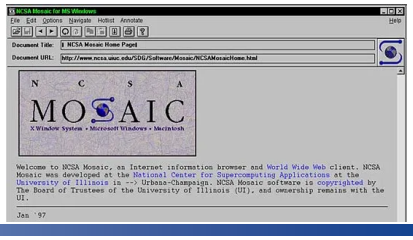


- Informationen auf Servern speichern, abrufen und tauschen
  - World Wide Web
    - Moderne Form einer Online-Bibliothek, -Mediathek, -Dienstleistung
  - Heute
    - Wesentlicher Pfeiler der Informationsgesellschaft
    - Ersetzt viele anderen Medien, wie Radio, TV, ...
- Vom synchronen Austausch Mensch <-> Mensch zu Mensch → Maschine, Mensch ← Maschine
  - Zunächst gebunden an logischen „Ort“ (Webserver)
  - Später „finden" nach Informationen über Suchmaschinen

Zur Entwicklung des World Wide Web (WWW)

- **Erfunden von Tim Berners-Lee am CERN**
  - Ziel: Einfacher weltweiter Austausch von Dokumenten zwischen Wissenschaftlern
- **Erster Prototyp Ende 1990**
  - grafisch (auf NEXTStep) und zeilenorientiert
- **Durchbruch des WWW durch WWW-Client Mosaic**
  - entwickelt von Marc Andreesen und Eric Bina (NCSA at UIUC)
  - ursprünglich für X-Windows-Systeme
  - als Quellcode per FTP kostenlos verfügbar → schnelle Verbreitung
  - Marc Andreesen gründete 1995 die Firma Netscape (→ Mozilla → Firefox/Thunderbird)
- **Gründung des W3-Konsortiums im Juli 1994**
  - vorrangiges Ziel: Weiterentwicklung des WWW, z.B. durch Standardisierung von HTML
  - Infos unter http://www.w3.org





Client/Server-Architektur des WWW

First paper appears on the project at Hypertext conference
→ Only accepted as a poster!

- **Idee:**
  - Vernetzen von Informationen, die auf Rechnern verfügbar sind bzw. dafür auf Rechnern bereit gestellt werden
  - Text, Bilder, (Videos), …
    eigentlich alle möglichen Inhalte, Objekte, Medien
    - Inhalte, die über Text hinaus gehen (*Hyper*-text)
      → Hypertext-Ressourcen
  - Grundlage: Webseite mit Links als Ausgangspunkt
  - Verbindungen ((Hyper-)Links) zwischen diesen Ressourcen erlaubt einfache Vernetzung dieser Ressourcen (Browsen)

- **Client-Server-Architektur**
  - Web-Browser zur Anzeige von Hypertext-Dokumenten/Hypermedia-Objekten
  - Hyperlinks erlauben Navigation

- **Lösungen zu folgenden Fragestellungen erforderlich:**
  - Eindeutige Adressierung einer Web-Seite
  - Transport einer Web-Seite
  - Beschreibung des Inhalts (insbes. der Hyperlinks) einer Web-Seite

Adressierung eines Web-Dokuments

- **Uniform Ressource Locator (URL)**
  - weist der Client-Software den Weg zu einer bestimmten Ressource
  - auch für Inhalte anderer Server (USENET, FTP, Dateimanager) verwendbar
  - z.B. `http://www.s-inf.de/Skripte/DatKom-2011-SS-Klausur1MitLoesung.pdf`

    Zugriffs-protokoll | Web-Server (DNS-Name oder IP-Adresse) | Verzeichnisangabe auf Server | Ressourcen-Objekt (i.d.F. PDF-Datei)
  - oder `http://user:password@servername:port/subdirectories/resource#abschnitt`
  - …mit Zugangskontrolle und abweichender Portangabe (http = 80, https=443)
  - … kann auch nur Ausschnitte von Ressourcen anfragen (Chunked, Range)
- **Ressourcenbezeichnung identifiziert Objekt auf Server eindeutig**
  - bei WWW: abgerufene Web-Seite
  - bei FTP: zu übertragene Datei
  - bei Mail: Empfänger der Mail
- **Web-Browser unterstützen eine Vielzahl von Protokollen**
  - z.B. http://, https://, ftp://, mailto://, telnet://, soap://, coap://, file://, nntp://, news://, rtsp://

Mit dem URL-Konzept wird ein Adressierungsschema für die Ressourcen sämtlicher Internet-Anwendungen geschaffen, wobei sich hinter einer Ressource sehr unterschiedliche Objekte verbergen können (eine adressierbare Ressource kann z.B. sein: eine Datei, eine Datenbank, ein News-Artikel oder eine Telnet-Sitzung).

Die sich über die verschiedenen Internet-Anwendungen erstreckende Ressourcen-Adressierung legt die Basis dafür, dass mithilfe des Web-Clients, also des Browsers, auf unterschiedlichste Internet-Server zugegriffen werden kann. Allerdings müssen selbstverständlich hierzu die verschiedenen Protokollmaschinen (FTP, NTTP, SMTP, ...) in der Browser-Software enthalten sein.

Falls der Port weggelassen wird, so gilt der für den jeweiligen Dienst vereinbarte Default-Port, also z.B. 21=FTP, 23=TELNET, 25=SMTP, 80=HTTP.

Das WWW-Anwendungsprotokoll: HTTP

- **HTTP (HyperText Transport Protocol)**
  - Version 0.9 (1991) und 1.0 (1996): RFC1945
  - Version 1.1 (1999): RFC2068
  - Ursprünglich eingeführt zur Übertragung von Web-Seiten und Web-Ressourcen
- Wesentliche Eigenschaften
  - Zustandsloses Protokoll
  - ASCII-Anwendungsprotokoll
  - setzt auf eine (sichere) TCP-Verbindung auf
  - Default-Port: 80 (http), 443 (https)
  - bis HTTP 1.0: Separate TCP-Verbindung pro HTTP-Abfrage
  - Ab HTTP1.1: Wiederverwender der TCP-Verbindung (Pipelining)
- Beispiele von Befehlen:
  - `GET`: Anfordern eines bestimmten Dokuments
  - `HEAD`: Anfordern von Informationen im Kopfteil eines Dokuments
  - `POST`: Anhängen von Daten an ein existierendes Dokument
  - `PUT`: Anlegen eines Dokuments

Die Bezeichnung von HTTP als ASCII-Protokoll besagt konkret, dass die Anfragen durch den Client und die Antworten durch den Server als ASCII-Strings über das Netz übertragen werden, d.h. eine Beispiel-Anfrage GET /anschrift/inf.htm wird als ASCII-Zeichenfolge übergeben, woraufhin der Server den Inhalt dieser Seite ebenfalls als ASCII-Zeichenfolge überträgt.

Die obige Anfrage ist ein Beispiel für eine einfache Anfrage. Würden wir an die Anfrage noch die Angabe HTTP/1.0 anfügen, so würde aus der einfachen eine volle Anfrage. Bei einer vollen Anfrage wird vom Server neben dem Seiteninhalt zusätzliche Steuerinformation angeben, die Auskunft über den Inhaltstyp und die gewählte Kodierung (oft MIME) gibt.

Das Verhalten des HTTP-Servers, die Verbindung nach einer Anfrage sofort zu schließen, resultiert aus einer zentralen Eigenschaft von HTTP: HTTP ist zustandslos, was die Entwicklung von HTTP-Clients und HTTP-Servern vereinfacht, aber ein sehr ineffizientes Protokollverhalten bewirkt.

Die von HTTP unterstützten Befehle können als Methoden auf dem Objekt „Web-Seite“ angesehen werden. In den Versionen vor 1.0 gab es ausschließlich den Befehl GET, der daher auch gar nicht angegeben werden musste. Weitere Befehle neben GET sind:

> Statt der ganzen Seite kann durch den Befehl HEAD auch nur die Steuerinformation einer Web-Seite abgefragt werden.
>
> Ein Beispiel, wann der POST-Befehl vom Client benötigt wird, ist das

Senden von der von einem Benutzer über ein Formular eingegebenen Information an den Server. Daneben kann durch den Befehl PUT eine Datei auf den Server hochgeladen werden, sofern der Client die Rechte dazu besitzt. Der zu PUT entgegengesetzte HTTP-Befehl ist DELETE, durch den die angegebene Datei gelöscht werden soll.

Beispiel einer HTTP-Anfrage und HTTP-Antwort

HTTP-Client

```
GET /index.html HTTP/1.1
Host:www.comsys.rwth-aachen.de
Pragma: no-cache
....
```

- Anfrage von Client an Server
- Befehlszeile: <Befehl> <URL> <Version>
- Client wünscht nicht zwischengespeicherte, d.h. aktuelle Version des Dokuments

Hinweis: Verbindung zwischen Client und Server wurde bereits zuvor aufgebaut

HTTP-Server

```
HTTP/1.1 200 OK
Date: Fri, 24 Sep 2022 09:45:51 GMT
Server: Apache/2.4.54 (Linux)
Transfer-Encoding: chunked
Content-Type: text/html

<HTML>
Gemäß HTML-Konventionen
strukturiertes Dokument
</HTML>
```

- Antwort-Zeile
- Datum
- Server
- Angaben zur Kodierung
- Art des Inhalt

- Hauptteil

Der Aufbau der HTTP-Nachrichten, die zwischen Client und Server ausgetauscht werden, orientiert sich an den bekannten Konzepten des E-Mail-Transportes. Das wird insbesondere an der Trennung einer HTTP-Nachricht in Kopf (Header) und Hauptteil (Entity) deutlich.

Wir gehen im Folgenden von einer vollen Anfrage aus, was an der angegebenen HTTP-Version ersichtlich ist. In HTTP lassen sich durch eine Reihe von Parametern die Anfragen bzw. Antworten präzisieren. So können in der Anfrage die vom Client akzeptierten Dokumententypen bzw. Sprachen eingeschränkt werden oder die genutzte Client-Software dem Server mitgeteilt werden.

Der Server antwortet bei voller Anfrage (die heute fast ausschließlich genutzt wird) immer mit der verwendeten HTTP-Version und einem Antwort-Code. So bedeutet der Code „200“ beispielsweise, dass die Anfrage erfolgreich bearbeit werden konnte. Zu jedem Antwort-Code wird auch ein erläuternder Text mit angegeben; bei dem Code “200“ ist das „OK“.

Auch der Server kann noch Parameter nutzen, um dem Client weitere Angaben zu machen. So kann analog zum Parameter „User-Agent“ die verwendete Server-Software angegeben werden, oder es kann im Parameter „Location“ ein Verweis auf eine weitere URL erfolgen.

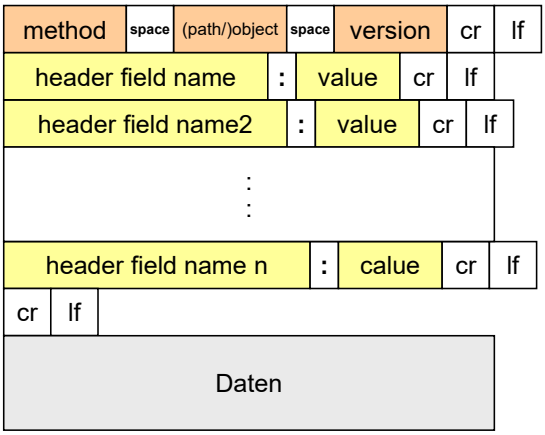
Die gerade in allgemeiner Form angegebenen Formate werden auf dieser Folie durch eine konkrete Anfrage bzw. Antwort verdeutlicht. Wir gehen bei diesem Beispiel implizit davon aus, dass die Verbindung zwischen Client und Server bereits besteht (also eine TCP-Verbindung auf Port 80).

Im Zusammenhang mit dem Header „Pragma“ ist folgender Sachverhalt wichtig: Der HTTP-Client kann abgefragte Web-Seiten im Cache halten und dadurch eine wiederholte Übertragung vermeiden.

Durch die Angabe „no-cache“ wird der Client aufgefordert, die Seite vom Server in jedem Fall zu fordern, selbst wenn sie sich im Cache befindet und seit diesem Zeitpunkt auch nicht verändert wurde. Der Header „Pragma“ (pragmatic) dient allgemein dazu, nicht standardisierte und möglicherweise Software-spezifische Aspekte zwischen Client und Server auszutauschen.

Die Antwort des Servers beginnt mit der obligatorischen Antwortzeile. Danach folgen verschiedene Parameter, die Informationen zu dem Server selbst (z.B. verwendete Software) und insbesondere das in der Antwort mitgelieferte Dokument geben. Nach den Parametern folgt der eigentliche Inhalt der HTML-Seite, der durch zwei sog. HTML-Tags, <HTML> und </HTML> begrenzt ist.

HTTP Request Header



Request Zeile: verpflichtend,
z.B. `GET /path/file.type`

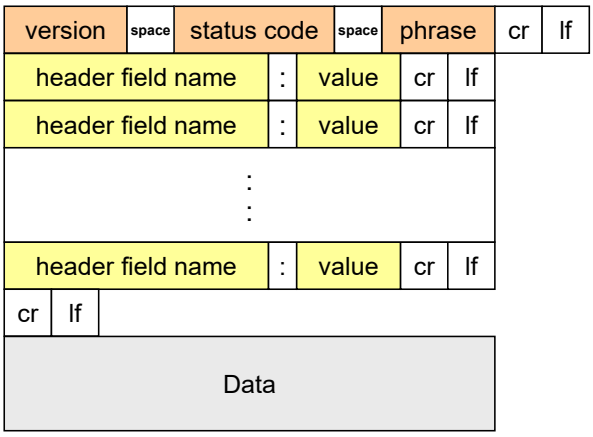
Header Zeilen: optional, weitere Informationen, Optionen, etc. zur Anfrage bzw. dem Objekt, z.B.:

```
Host: www.rwth-aachen.de
Accept-language: fr
Connection: close
User-agent: Firefox /5.0
```

Entity Body: optional.
Weitere Daten, die der Client dem Server übermittelt,
z.B: bei POST Methode

Space: Leerzeichen
cr/lf: carriage return/line feed (Leerzeile)

HTTP Response Header



Status Zeile: *Status Code* und Beschreibung geben Erfolg oder Erläuterung zur Anfrage an (menschenlesbar)

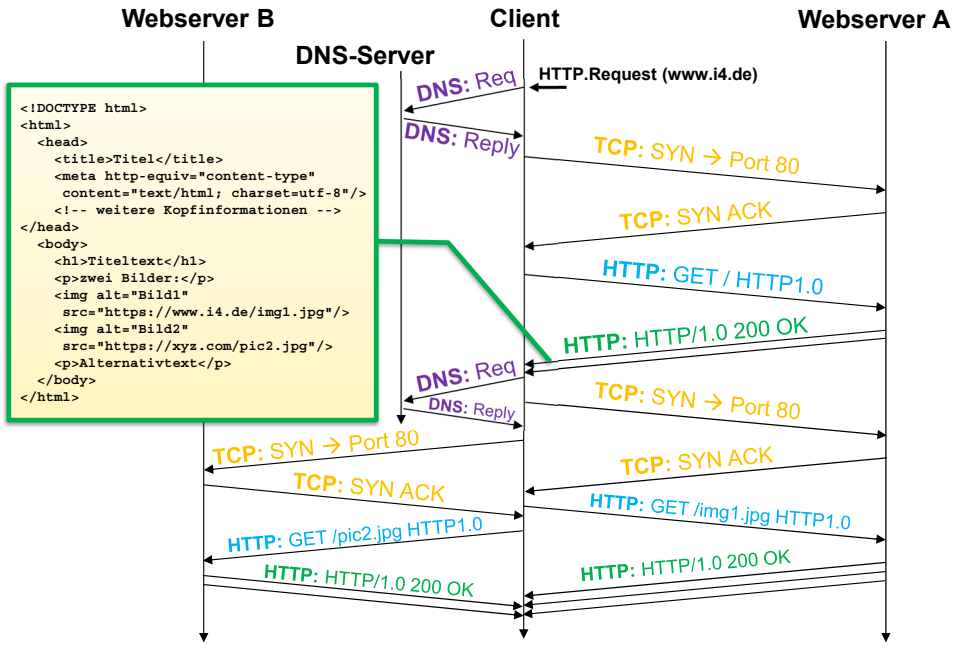
```
200 OK
400 Bad Request
404 Not Found
```

Kategorien der Statusmeldungen:

- `1xx`: Only for information
- `2xx`: Successful inquiry
- `3xx`: Further activities are necessary
- `4xx`: Client error (syntax)
- `5xx`: Server error

`HEAD` *Methode*: Der Server antwortet, aber überträgt nur Statuszeile und Header. Keine Nutzdaren (wird für Debbugging und Testen verwendet)

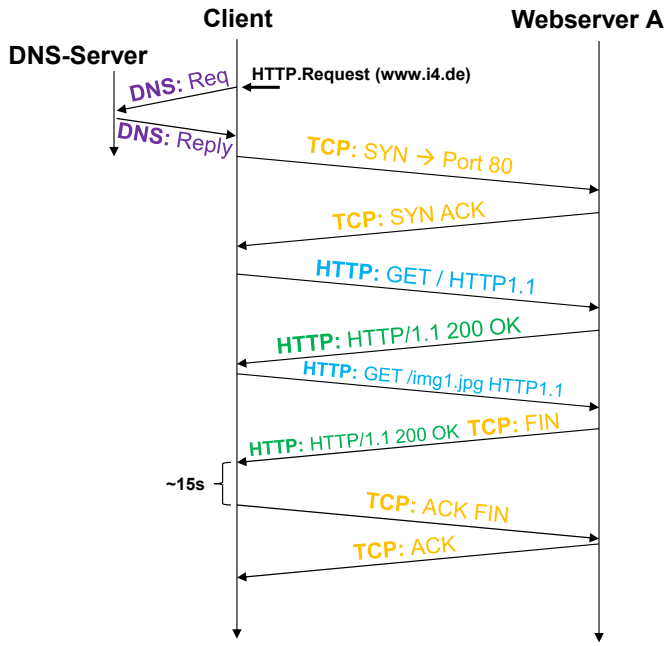
HTTP Request 1.0




```
<!DOCTYPE html>
<html>
  <head>
    <title>Titel</title>
    <meta http-equiv="content-type"
     content="text/html; charset=utf-8"/>
    <!-- weitere Kopfinformationen -->
</head>
  <body>
    <h1>Titeltext</h1>
    <p>zwei Bilder:</p>
    <img alt="Bild1"
     src="https://www.i4.de/img1.jpg"/>
    <img alt="Bild2"
     src="https://xyz.com/pic2.jpg"/>
    <p>Alternativtext</p>
  </body>
</html>
```

Schließen der TCP-Verbindungen nicht abgebildet.
Beim Übertragen des HTTP Requests schließt der Client die Verbindung einseitig.
Nach Aussenden der Antwort auch der Server.

HTTP Request 1.1 (Persistent Connection)



Ein Verbindungsaufbau zu einem HTTP-Server lässt sich mit Hilfe des Tools telnet realisieren:

```
telnet www.comsys.rwth-aachen.de
```

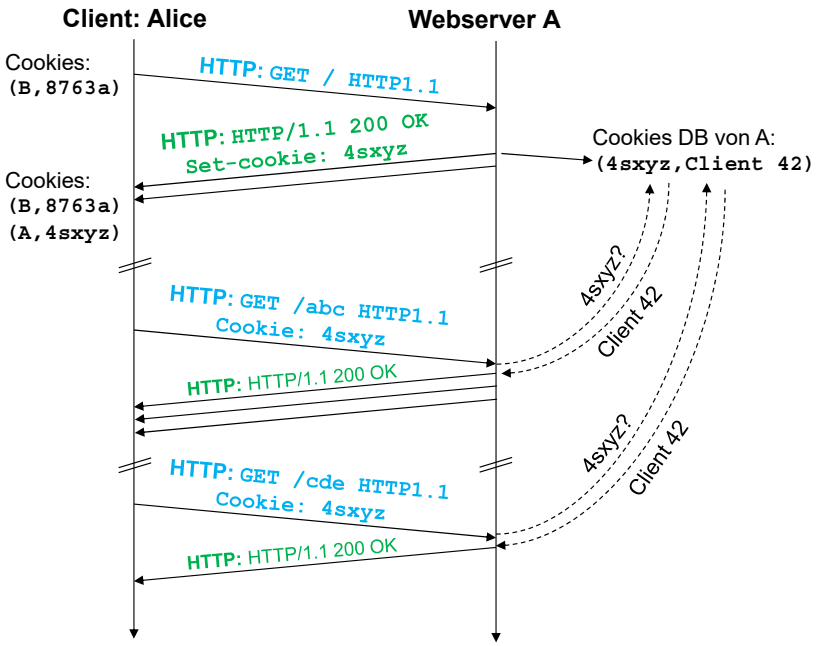
Da die meisten Server ihre Webseiten jedoch nicht mehr per HTTP ausliefern, muss eine verschlüsselte Verbindung mittels openssl aufgebaut werden. Dieser Aufruf benötigt einige Informationen mehr. Insbesondere muss sowohl die IP-Adresse oder der Name des Servers, zu dem die Verbindung aufgebaut werden soll (im folgenden Beispiel www.comsys.rwth-aachen.de) als auch der im daraufhin genutzten SSL-Zertifikat konfigurierte Servername (hier folks.i4.de) angegeben werden:

```
openssl s_client -crlf -servername folks.i4.de -
connect www.comsys.rwth-aachen.de:443
```

User Identification: Cookies

- **Eigenschaft von HTTP: zustandsloses Protokoll**
  - Eine HTTP-Sitzung entspricht einer TCP-Verbindung
  - Nach Beendigung der Verbindung "vergisst" der Webserver alles über die Anfrage
  - Einfaches Prinzip, ausreichend zum Surfen
  - Reduziert komplexe “Sitzungsverwaltung“ auf Webserver
  - Aber: moderne Prinzipien wie Online-Shops?
    - Speicherung von Informationen über verbundene Sitzungen?

Identifikation/Herstellung eines Kontexts zwischen Aufrufen: Cookies

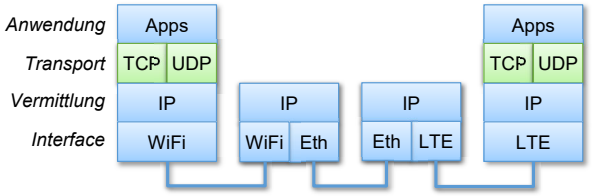


User Identification: Cookies

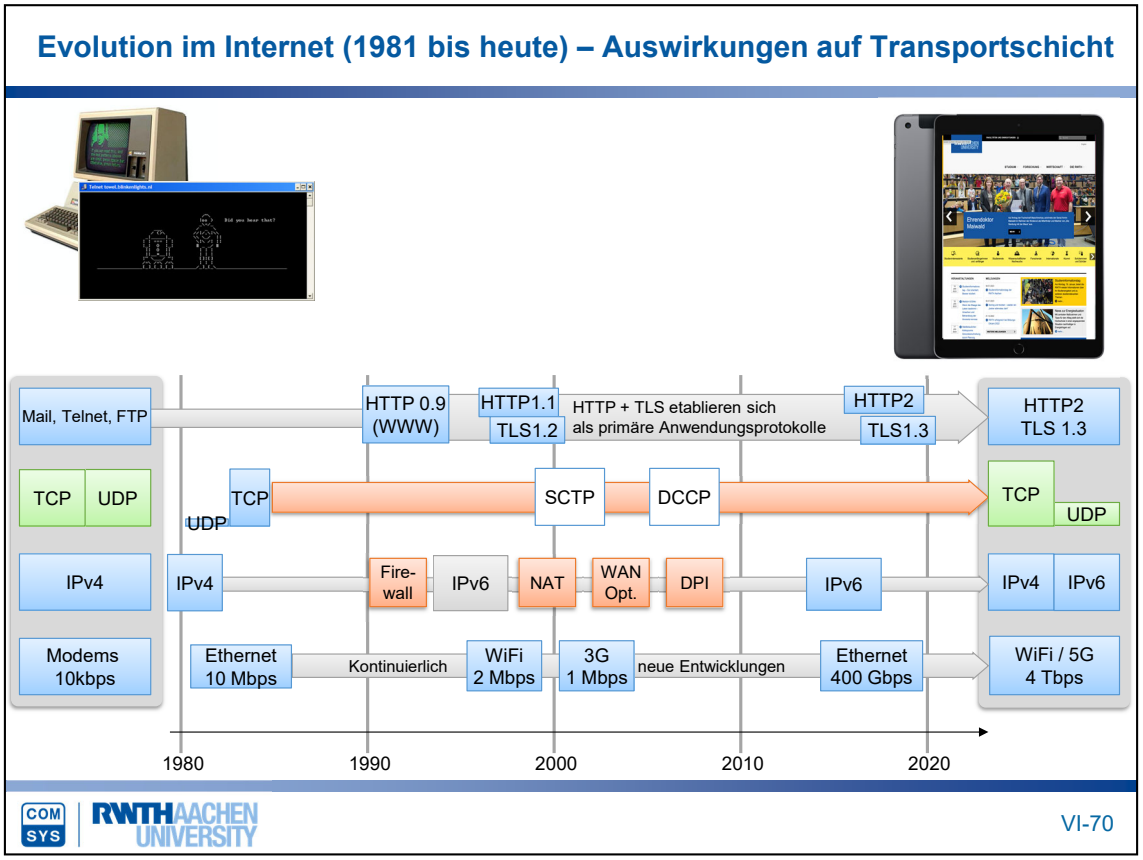
- **Eigenschaft von HTTP: zustandsloses Protokoll**
  - Eine HTTP-Sitzung entspricht einer TCP-Verbindung
  - Nach Beendigung der Verbindung "vergisst" der Webserver alles über die Anfrage.
  - Einfaches Prinzip, ausreichend zum Surfen
  - Aber: moderne Prinzipien wie Online-Shops?
    - Speicherung von Informationen über verbundene Sitzungen?
- **Lösung: neues Header-Feld `Set-cookie: ...`**
  - Server übergibt Client ein Cookie
  - Client speichert Cookie zusammen mit Servernamen
  - Bei nachfolgenden Anfragen trägt Client das passende Cookie in Kopfzeile
  - Server kann zusammenhängende Anfragen identifizieren
    - Heute oft Inhalt abhängig von der Identifikation der Sitzung
    - Sitzungs-abhängige Generierung von HTML-Inhalten (auf Server-Seite, oder beim Client)
- **Inhalt des Cookies:**
  - vom Server definiertes Name-Wert-Paar zur Identifizierung
  - optionale Name-Wert-Paare für z. B. Kommentare, Datum, TTL

Rückblick über 40 Jahre Transportschicht

- **Protokollmechanismen der Transportschicht**
  - Prozessadressierung, strombasierte vs. paketbasierte Kommunikation, verbindungsorientiert/verbindungslos
- **Die Transportschicht im Internet**
  - TCP (Transmission Control Protocol)
    - Adressierung, Sockets
    - TCP-Verbindung
    - Flusskontrolle
    - TCP-Header
    - Staukontrolle



  - UDP (User Datagram Protocol)
  - Entwicklungen auf der Transportschicht
    - Anforderungen an die Transportschicht im heutigen Internet
    - Möglichkeiten zur Anpassung der Transportschicht
      - Neue Transportprotokolle oder TCP-Optionen?
      - QUIC



Viel Spaß beim ersten Terminal-Blockbuster: telnet towel.blinkenlights.nl