Rechnernetze I

SoSe 2024

Roland Wismüller
Universität Siegen
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 18. März 2024

Inhaltsverzeichnis

Rechnernetze I

SoSe 2024

0 Organisation

Zu meiner Person

- Studium der Informatik an der Techn. Univ. München
  - dort 1994 promoviert, 2001 habilitiert
- Seit 2004 Prof. für Betriebssysteme und verteilte Systeme
- **Forschung:** Sichere komponentenbasierte Systeme; parallele und verteilte Systeme
- Vorsitzender des Prüfungsausschusses Informatik; Mentor für Bachelor Informatik 2012 mit Vertiefung Mathematik
- **e-mail:** roland.wismueller@uni-siegen.de
- **Tel.:** 0271/740-4050
- **Büro:** H-B 8404
- **Sprechstunde:** Mo., 14:15-15:15 Uhr

Zur Fachgruppe „Betriebssysteme / verteilte Systeme“

Andreas Hoffmann
andreas.hoffmann@uni-...
0271/740-4047
H-B 8405

- El. Prüfungs- und Übungssysteme
- IT-Sicherheit
- Web-Technologien
- Mobile Anwendungen

Felix Breitweiser
felix.breitweiser@uni-...
0271/740-4719
H-B 8406

- Betriebssysteme
- Programmiersprachen
- Virtuelle Maschinen

Sven Jacobs
sven.jacobs@uni-...
0271/740-2533
H-B 8407

- El. Prüfungs- und Übungssysteme
- Generative KI
- Web-Technologien

Lehrangebot

Vorlesungen/Praktika

- Rechnernetze I, 5/6 LP (Bachelor, SoSe)
- Rechnernetze Praktikum, 5/6 LP (Bachelor, WiSe)
- Rechnernetze II, 6 LP (Master, SoSe)

- Betriebssysteme I, 5/6 LP (Bachelor, SoSe)
- Parallelverarbeitung, 6 LP (Master, WiSe)
- Verteilte Systeme, 5/6 LP (Bachelor, WiSe)

Lehrangebot ...

Projektgruppen

- z.B. Sichere Kooperation von Software-Komponenten
- z.B. Konzepte zum sicheren Management von Linux-basierten Thin-Clients

Abschlussarbeiten (Bachelor, Master)

- Themengebiete: sichere virtuelle Maschine, Parallelverarbeitung, Mustererkennung in Sensordaten, eAssessment, ...

Seminare

- Themengebiete: IT-Sicherheit, Programmiersprachen, Mustererkennung in Sensordaten, ...
- Ablauf: Blockseminare (30 min. Vortrag, 5000 Worte Paper)
- Master: vorher Vorlesung „Wissenschaftliches Arbeiten“!

Zur Vorlesung

Vorlesung (3 SWS)

- Mo., 08:30 - 10:00 Uhr, H-C 3305, 14-tägig ab 15.04.
- Do., 10:15 - 11:45 Uhr, H-C 3305

Übung (2 SWS)

- Zwei alternative Übungsgruppen:
  - Mo., 10:15-11:45, H-C 6321 bzw. H-A 4111
  - Fr., 12:15-13:45, H-C 8326 bzw. H-A 4111
- Übungsbeginn: Mo. 22.04. (H-A 4111)

Zur Vorlesung ...

Übung (2 SWS) ...

- Ca. jede zweite Woche praktische Aufgaben im H-A 4111
  - bitte Information auf der Webseite beachten
  - mit der Nutzung der Rechner akzeptieren Sie die Benutzerordnung (siehe Webseite)!
  - wegen Kennungen bitte ggf. noch **unverzüglich** für Vorlesung und Übung in unisono anmelden!

Zur Vorlesung ...

Industriezertifikat CCNA

- CCNA: Cisco Certified Network Associate
  - Grundstufe der Cisco-Zertifikate
  - weltweit anerkannt, gehören zu den begehrtesten in der Netzwerkindustrie
- Vorlesungsbegleitend zu RN_I und RN-Praktikum möglich
  - RN_I (SoSe): CCNAv7 *Introduction to Networks*
  - RN-Praktikum (WiSe): CCNAv7 *Switching, Routing, and Wireless Essentials* & CCNAv7 *Enterprise Networking, Security, and Automation*
- Zusätzliches Selbststudium mit Online-Materialien
- Externe Zertifizierungprüfung, Kosten ca. 135,- Eur

Anmerkungen zu Folie 9:

Die Universität Siegen (Fachgruppe Betriebssysteme und verteilte Systeme) ist seit 2012 eine anerkannte lokale *Cisco Networking Academy* im Rahmen des *Cisco Networking Academy* Programms.

Im Rahmen dieses Programms bieten wir die Kurse CCNAv7 *Introduction to Networks*, CCNAv7 *Switching, Routing, and Wireless Essentials* und CCNAv7 *Enterprise Networking, Security, and Automation* an, die auf das Industriezertifikat *CCNA* vorbereiten. Das weltweit anerkannte Industriezertifikat CCNA bescheinigt ein grundlegendes Wissen im Bereich Netzwerke, insbesondere die Fähigkeit, kleinere und mittlere Netzwerke installieren, konfigurieren und betreiben zu können. Dazu gehören u.a. Kenntnisse über Ethernet, WLAN, IPv4, IPv6 und WAN-Protokolle, VLANs, Routing-Protokolle wie RIP, OSPF und EIGRP sowie die Zugriffskontrolle über ACLs. Das Zertifikat kann beim Berufseintritt einen sichtbaren Bewerbervorteil bedeuten, wenn Ihr Beruf auch Tätigkeiten im Bereich Netzwerkadministration beinhaltet (beispielsweise auch bei Informatik-Lehrern!).

Für Studierende bieten wir die Vorbereitung auf die Zertifizierung kostenlos und vorlesungsbegleitend an. Die Voraussetzung dafür ist die Teilnahme an der Vorlesung und Übung Rechnernetze I und dem Rechnernetze-Praktikum (Vertiefungspraktikum Bachelor Informatik). Ein Einstieg in die CCNA-Kurse ist immer (und nur!) im Sommersemester möglich (mit Rechnernetze I).

Organisatorische Hinweise zum CCNA:

- Falls Sie an dem Programm teilnehmen wollen, melden Sie sich bitte in der ersten Übung an. Sie werden dann bei *Cisco Netspace* registriert und erhalten Zugang zu den Online-Materialien des ersten Kursteils.
- Die Online-Materialien müssen Sie zu Hause selbständig durcharbeiten. Jeder Kursteil besteht aus 14-17 Modulen, wobei nach ca. jedem dritten Modul ein kurzes ***Module Group Exam*** steht, eine eine automatisch ausgewertete elektronische Prüfung. Diese ist **bis zu einem festgelegten Stichtag** zu bearbeiten.
- Zu jedem Modul gibt es auch praktische *Labs*, die Sie in den Übungen bzw. im Rechnernetze-Praktikum bearbeiten.
- Am Ende jedes Kursteils gibt es nochmals ein umfangreicheres ***Final Exam***, das wie die *Module Group Exams* eine elektronische Prüfung ist. Zusätzlich gibt es ein ***Hands On Skills Exam***, bei dem Sie im Labor ein reales Netzwerk aufbauen und konfigurieren müssen. Für beide *Exams* werden wir **gegen Semesterende** jeweils einen Termin mit Ihnen vereinbaren.
- Wenn Sie im *Final Exam* des letzten Kursteils mindestens 75% der Punkte erreichen, erhalten Sie einen ***Voucher***, der Ihnen einen Rabatt von 58% für die eigentliche Zertifizierungsprüfung einräumt, so daß die Kosten von 270 Eur (+ MwSt) auf 135 Eur reduziert werden. Der *Voucher* gilt dabei ein halbes Jahr.

- Diese **Zertifizierungsprüfung** findet in einem *Pearson VUE Test Center* statt.
  - In Siegen gibt es leider kein Test Center mehr. Die nächstgelegenen Test Center sind in Marburg, Koblenz, Köln und Soest.
- Das Zertifikat gilt drei Jahre und muß dann ggf. (über eine erneute Prüfung) erneuert werden.
- Unabhängig von dem Zertifikat erhalten Sie für jeden erfolgreich abgeschlossenen Kursteil auch eine **Teilnahmebestätigung** von Cisco, die Sie für Ihre Bewerbungsunterlagen nutzen können.

CISCO

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706

Direct: 408 526 4000
FAX: 408 526 4100
www.cisco.com

16.03.2017

Dear

Congratulations on completing the Cisco® **CCNA Routing and Switching: Scaling Networks** course as part of the Cisco Networking Academy® program. This hands-on, lab-oriented course has prepared you for exciting career opportunities in the technology industry.

By completing this course you have earned a Certificate of Completion for **CCNA Routing and Switching: Scaling Networks**, and acquired competencies that include the following:

- Understand, configure, and troubleshoot enhanced switching technologies such as VLANs, Rapid Spanning Tree Protocol (RSTP), Per VLAN Spanning Tree Plus Protocol (PVST+), and EtherChannel
- Understand, configure, and troubleshoot first hop redundancy protocols (HSRP) in a switched network
- Understand, configure, and troubleshoot wireless routers and wireless clients
- Configure and troubleshoot routers in a complex routed IPv4 and IPv6 network using single-area OSPF, multiarea OSPF, and Enhanced Interior Gateway Routing Protocols (EIGRP)
- Manage Cisco IOS® Software licensing and configuration files

Technological literacy is more important today than ever before, and Cisco is proud to provide you with the knowledge and skills necessary to build and maintain computer networks.

Please accept my best wishes for your continued success.

Sincerely,

Chuck Robbins
Chief Executive Officer
Cisco Systems, Inc.

CISCO Cisco Networking Academy® Mind Wide Open™

Certificate of Course Completion

CCNA Routing and Switching: Scaling Networks

During the Cisco Networking Academy® course, administered by the undersigned instructor, the student was able to proficiently:

- Understand, configure, and troubleshoot enhanced switching technologies such as VLANs, Rapid Spanning Tree Protocol (RSTP), Per VLAN Spanning Tree Plus Protocol (PVST+), and EtherChannel
- Understand, configure, and troubleshoot first hop redundancy protocols (HSRP) in a switched network
- Understand, configure, and troubleshoot wireless routers and wireless clients
- Configure and troubleshoot routers in a complex routed IPv4 or IPv6 network using single-area OSPF, multiarea OSPF, and Enhanced Interior Gateway Routing Protocol (EIGRP)
- Manage Cisco IOS® Software licensing and configuration files

Student

Universitaet Siegen
Academy name

Germany
Location

Manfred Stettner
Instructor

16.03.2017
Date

Instructor Signature

Zur Vorlesung ...

Industriezertifikat CCNA ...

- Mehr Infos: `http://www.bs.informatik.uni-siegen.de/cisco`
- Anmeldung in den ersten Übungsstunden


CISCO

Cisco Certifications

Roland Wismüller

HAS SUCCESSFULLY COMPLETED THE CISCO CERTIFICATION REQUIREMENTS AND IS RECOGNIZED AS A

Cisco Certified Network Associate Routing and Switching

CISCO CERTIFIED CCNA ROUTING & SWITCHING

CERTIFICATION DATE September 16, 2013
VALID THROUGH September 16, 2016
CISCO ID NO. CSCO12447888

Validate this certificate's authenticity at
www.cisco.com/go/verifycertificate
Certificate Verification No. 415343161150ASYM

John Chambers
Chairman and CEO
Cisco Systems, Inc.

9533973
0924

Zur Vorlesung ...

Information, Folien und Ankündigungen

- **Auf der Vorlesungs-Webseite:**

  `http://www.bs.informatik.uni-siegen.de/lehre/rn1`

- Ggf. Aktualisierungen/Ergänzungen kurz vor der Vorlesung
  - auf das Datum achten!
- Dort auch Links zu den CCNA-Materialien
- Zugangsdaten für geschützte Bereiche:
  - werden in der Vorlesung bekanntgegeben!
- Es gibt auch einen Moodle Kurs mit Aufzeichnungen aus dem SoSe 2020 (mit Ergänzungen), Selbsttests, etc.

Zur Vorlesung ...

Prüfung

- **Alle Studiengänge:**
  - 1-stündige **elektronische** Klausur, ohne Hilfsmittel
  - voraussichtliche(!) Termine:
    - SoSe: Mo., 05.08.2024 (ohne Gewähr!)
    - WiSe: Mo., 03.02.2025 (ohne Gewähr!)
  - Zeit / Ort wird noch verbindlich bekanntgegeben (unisono!)

Literatur

- Larry L. Peterson, Bruce S. Davie: Computernetze: *Ein modernes Lehrbuch*. 3. Auflage, dpunkt.verlag Heidelberg, 2004.
- Skript: kein ausformuliertes Skript, aber Anmerkungen zu etlichen Folien in der 2-auf-1 Version
  - gedruckte Version: bitte beim Fachschaftsrat fragen!
- Zur Vertiefung:
  - Online-Materialien Cisco CCNA „Introdution to Networks“ (siehe Webseite)
  - A. Tanenbaum. Computernetzwerke, 4. Auflage, Pearson Studium, 2003

Gliederung der Vorlesung

- Einführung
- Protokolle, Protokollhierarchie
- Direktverbindungsnetze, Ethernet
- LAN Switching
- Internetworking, IP
- Routing
- UDP, TCP, Sicherung der Übertragung
- Datendarstellung
- Anwendungsprotokolle
- Netzwerksicherheit

Ziel der Vorlesung

- Grundwissen jedes Informatikers im Bereich Netzwerke
- Verständnis der Probleme und ihrer Lösungen
- Grundverständnis gängiger Netzwerkprotokolle
- Grundlage für weiterführende Lehrveranstaltungen
  - Rechnernetze-Praktikum (WiSe)
  - Rechnernetze II (Ergänzung/Vertiefung der Themen, SoSe)
  - Parallelverarbeitung (WiSe)
  - ...

Rechnernetze I

SoSe 2024

1 Einführung

1 Einführung ...

Inhalt

- Motivation
- Verbindungsstrukturen
- Anforderungen an Netze
- Leistungsparameter

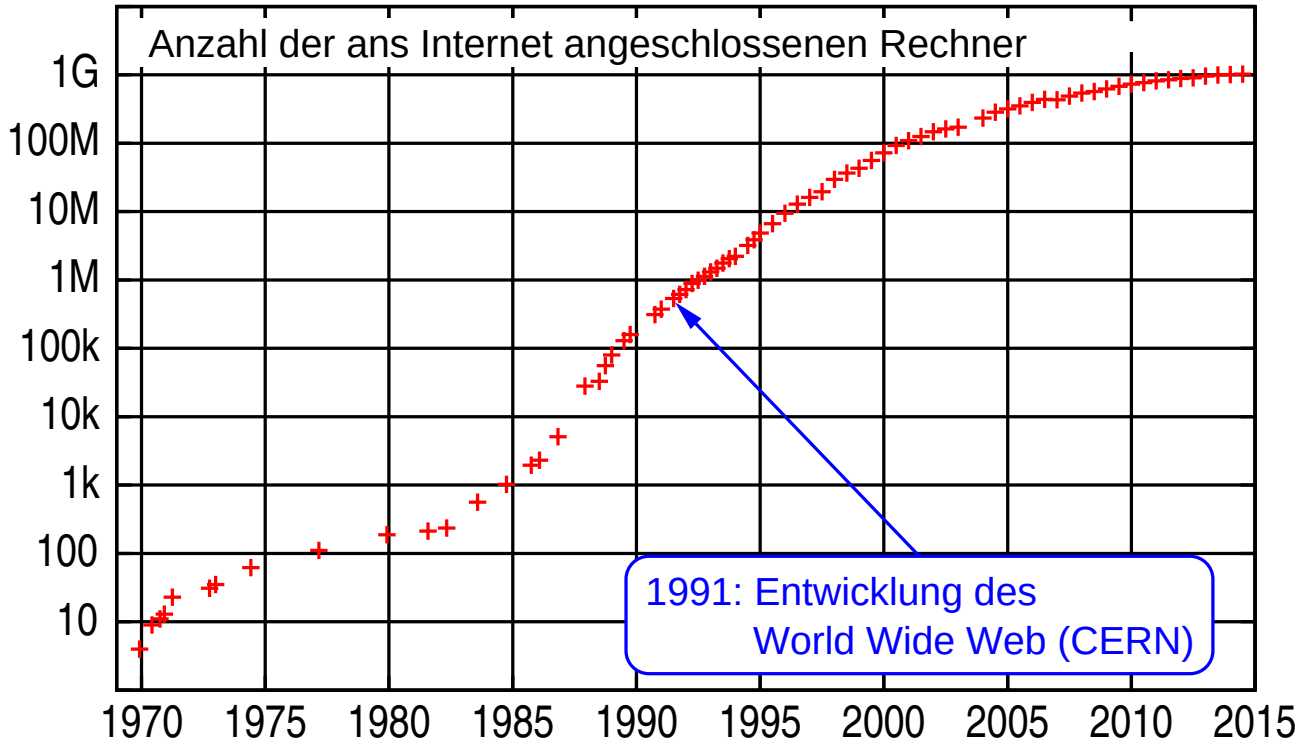
- Peterson, Kap. 1.2
- CCNA, Kap. 1

The Network is the Computer

- Vernetzungsaspekt wird zunehmend wichtiger als lokale Datenverarbeitung
- Boom im Bereich der Vernetzung / Netzwerktechnik
  - ausgelöst durch WWW / Internet

(Animierte Folie)

Entwicklung des Internet



1.2 Strukturen von Rechnernetzen

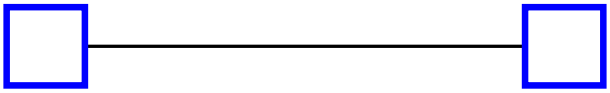
★

Grundelemente eines Rechnernetzes

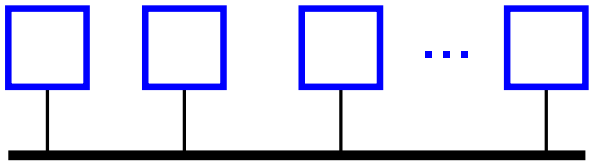
- **Knoten**: Endgeräte (Rechner, Host), Vermittlungsknoten (Switch, Router, ...)
- **Verbindungen** („Leitung“): Kabel, Glasfaser, Funk, ...

Verbindungsstrukturen

- **Punkt-zu-Punkt Verbindung**:



- **Mehrfachzugriffsverbindung (Bus)**:



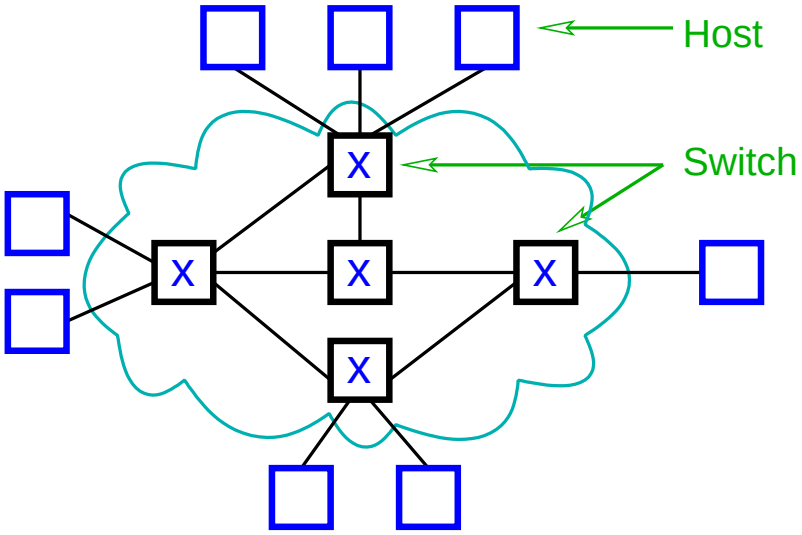
Anmerkungen zu Folie 20:

Die Sterne-Markierungen rechts oben auf den Folien geben eine unverbindliche**(!)** Einschätzung für die Prüfungsrelevanz der jeweiligen Folie an:

- ★★★: Die Inhalte sind sehr wichtig; sie sollten auf jeden Fall bis ins Detail verstanden (und ggf. auch auswendig gelernt) werden; sehr hohe Wahrscheinlichkeit einer entsprechenden Prüfungsfrage.
- ★★: Die Inhalte sind wichtig; sie sollten möglichst im Detail verstanden (und ggf. auch grob auswendig gelernt) werden; hohe Wahrscheinlichkeit einer entsprechenden Prüfungsfrage.
- ★: Die Inhalte sind als Hintergrundwissen wichtig; sie sollten auch verstanden werden; eine direkte(!) Prüfungsfrage dazu ist aber eher unwahrscheinlich.
- (ohne): Hintergund- oder Ergänzungswissen bzw. erläuternde Beispiele.

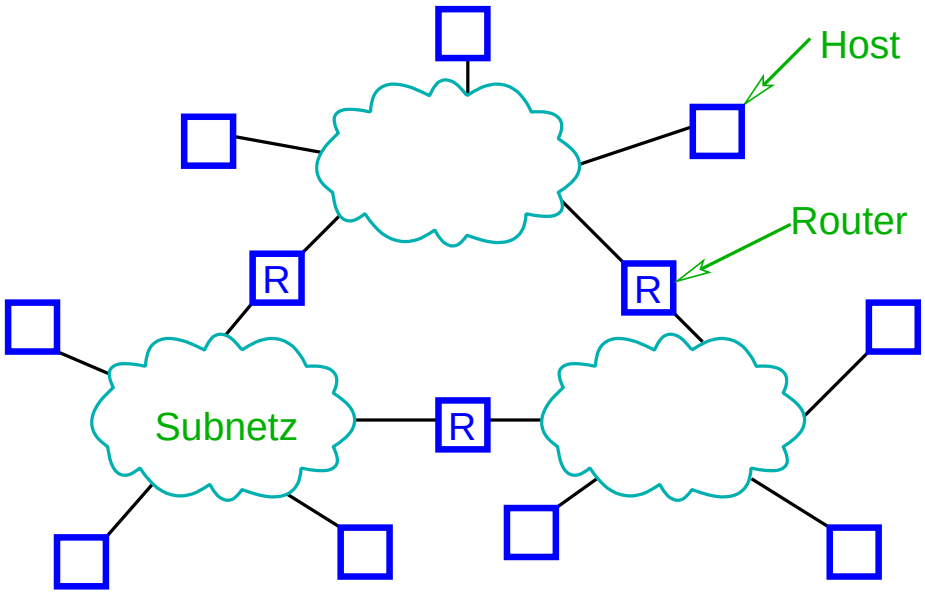
Verbindungsstrukturen ...

- **Vermitteltes Netzwerk**
  - Punkt-zu-Punkt Verbindungen mit Vermittlungsknoten (Switch)



Verbindungsstrukturen ...

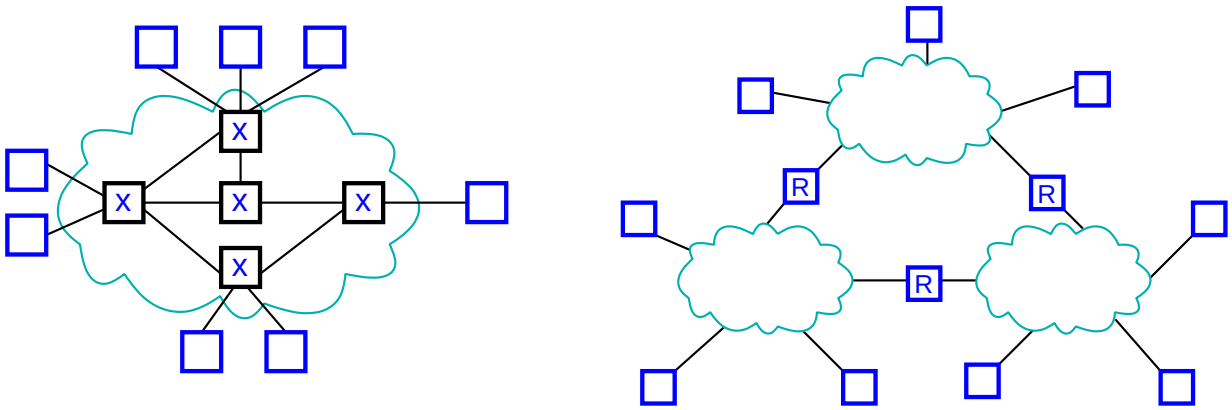
- **Zusammenschluß mehrere Netze (Internetwork)**
  - Kopplung mehrerer Subnetze durch Knoten (Router)



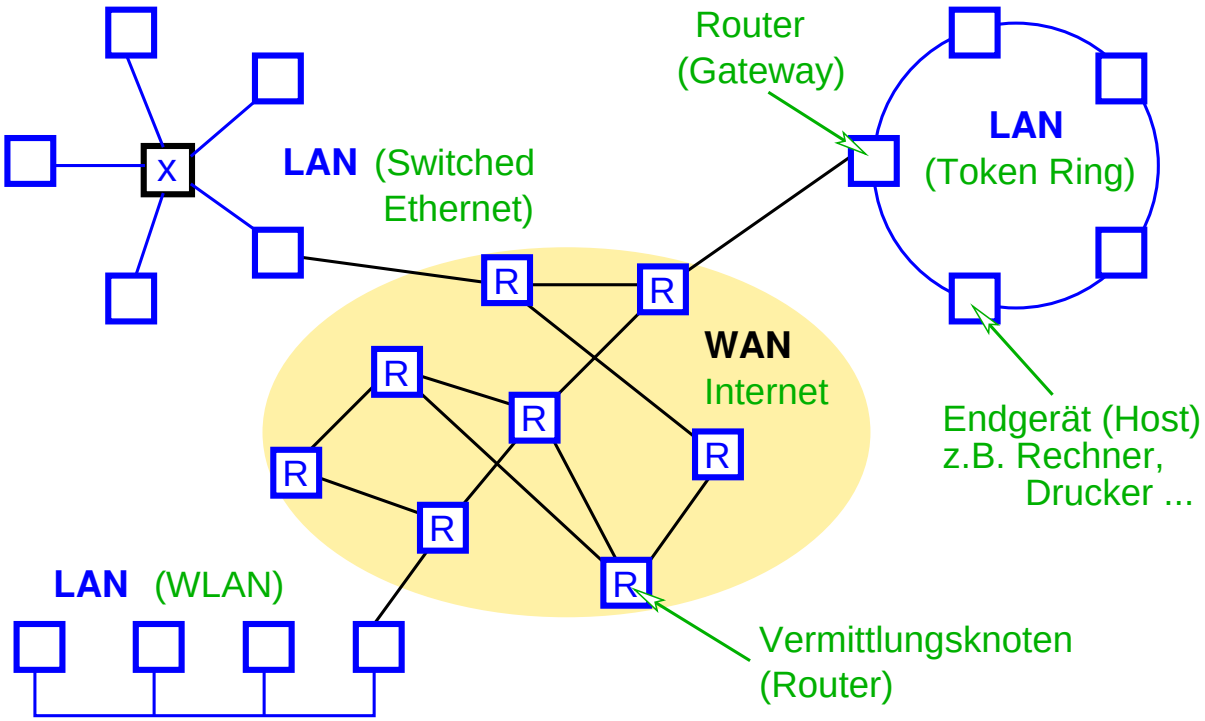
Allgemeine Struktur eines Netzwerks

- Ein Netzwerk besteht aus
  - mehreren Knoten, verbunden durch eine Leitung

  oder
  - mehreren Netzwerken, verbunden durch ein oder mehrere Knoten



Beispiel für ein Netzwerk



Klassifikation nach geographischer Ausdehnung

- **SAN**: *System Area Network*
  - Hochgeschwindigkeitsnetz, innerhalb eines Raums
- **LAN**: *Local Area Network*
  - $\leq$ 1 km, innerhalb eines Gebäudekomplexes, z.B. Ethernet
- **MAN**: *Metropolitan Area Network*
  - $\leq$ 10 km, innerhalb einer Stadt
- **WAN**: *Wide Area Network*
  - länder-bzw. weltumspannend, z.B. Internet
- Einsatz jeweils unterschiedlicher Technologien

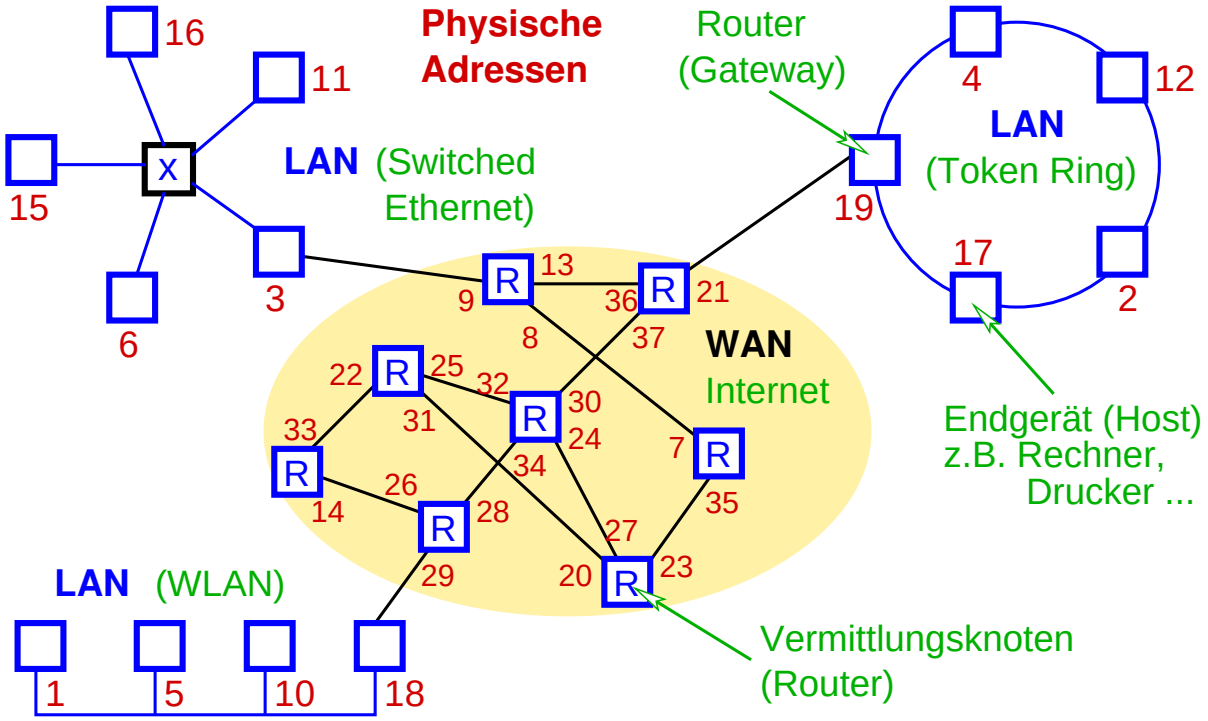
Wichtige Begriffe / Aufgaben

- **Adressierung**
  - physische Adresse: identifiziert Host* weltweit eindeutig, keine Information über das Netz des Hosts*
  - logische Adresse: identifiziert Netz und Host* in diesem Netz
  - Verwendung numerischer Adressen

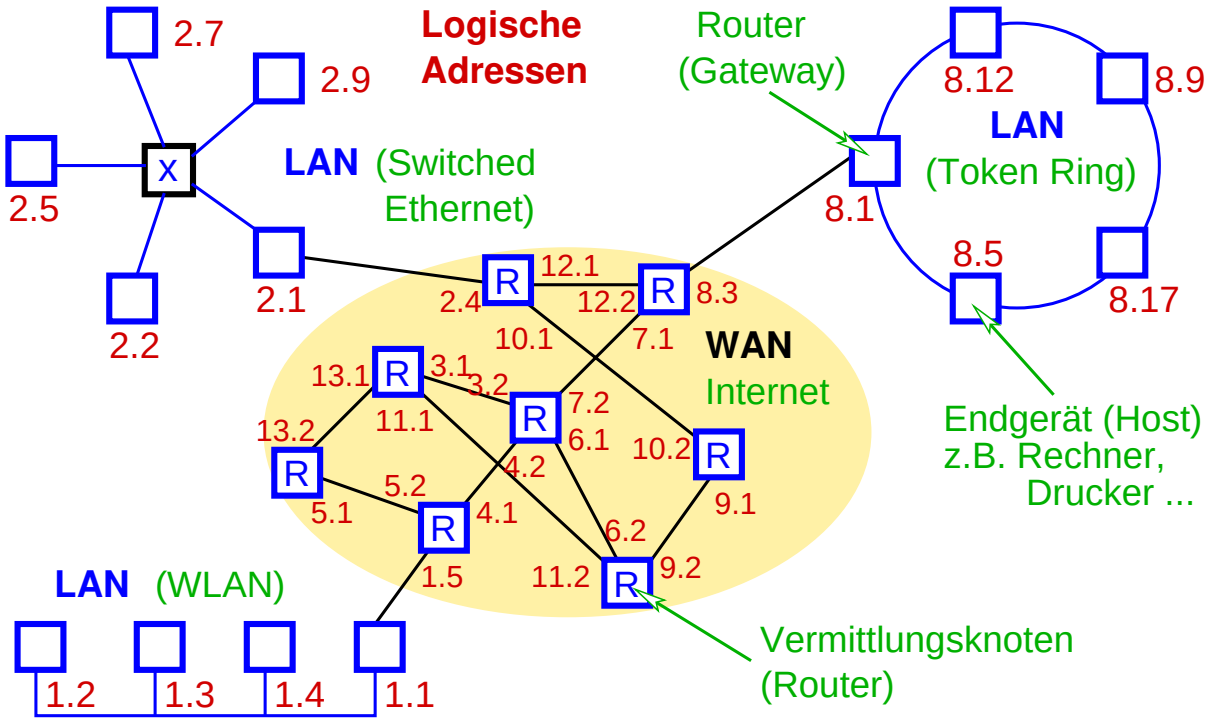
* genauer: Netzwerkkarte

- Anzahl der Empfänger
  - **Unicast**: genau einer
  - **Broadcast**: alle
  - **Multicast**: mehrere bestimmte
- **Routing / Forwarding (Vermittlung / Weiterleitung)**
  - Weiterleitung der Daten zum Empfänger durch Zwischenknoten

Beispiel für ein Netzwerk



Beispiel für ein Netzwerk



1.3 Multiplexing

- Ziel: Nutzung einer Verbindungsleitung für mehrere (unabhängige) Kommunikationsbeziehungen
- **Frequenzmultiplex**
  - Nutzung verschiedener Frequenzkanäle (z.B. Kabelfernsehen)
  - Variante: Wellenlängenmultiplex bei Glasfasern
- **Synchrones Zeitmultiplex**
  - Umschaltung zwischen den Kommunikationsbeziehungen in festen Zeitabständen
  - Vorteil: deterministisches Zeitverhalten (Echtzeit)
- **Statistisches Multiplexen**
  - Daten**pakete** der einzelnen Kommunikationsbeziehungen werden abwechselnd versendet
    - reihum oder nach Prioritäten (Dienstgüte)

1.4 Vermittlungsarten

★★

- **Leitungsvermittlung (*circuit switching*)**
  - für die Kommunikationspartner wird eine dedizierte Verbindung hergestellt
- **Speichervermittlung (*store and forward routing*)**
  - Daten werden von einer Vermittlungsstelle zur nächsten weitergegeben und vollständig gepuffert
- **Paketvermittlung (*packet switching*)**
  - Daten werden in Pakete zerteilt, Pakete werden unabhängig voneinander befördert
  - typisch für Rechnernetze
  - Varianten: Datagrammvermittlung, virtuelle Leitungsvermittlung (☞ **4.1**)

★★

Leitungsvermittlung

- Kommunikationspartner sind durch die geschaltete Leitung verbunden
- Beispiel: früheres Telefonnetz

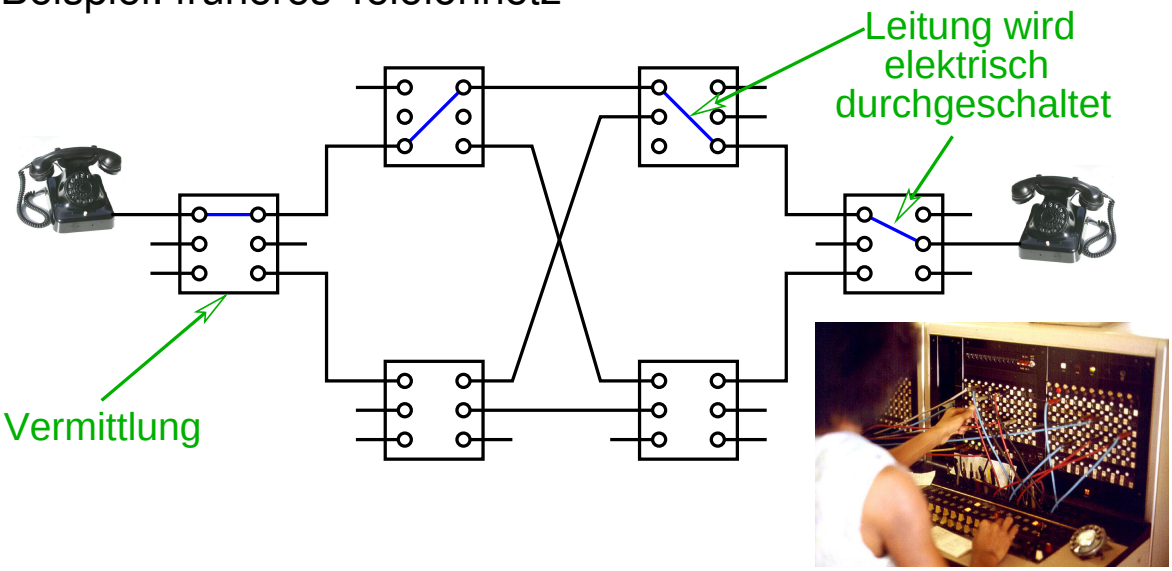
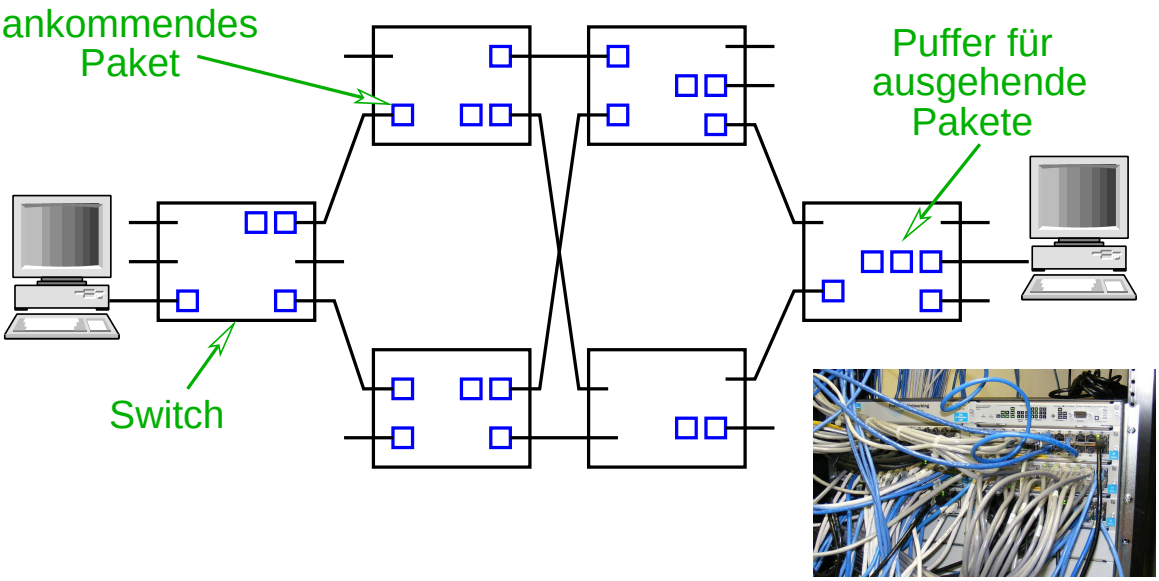


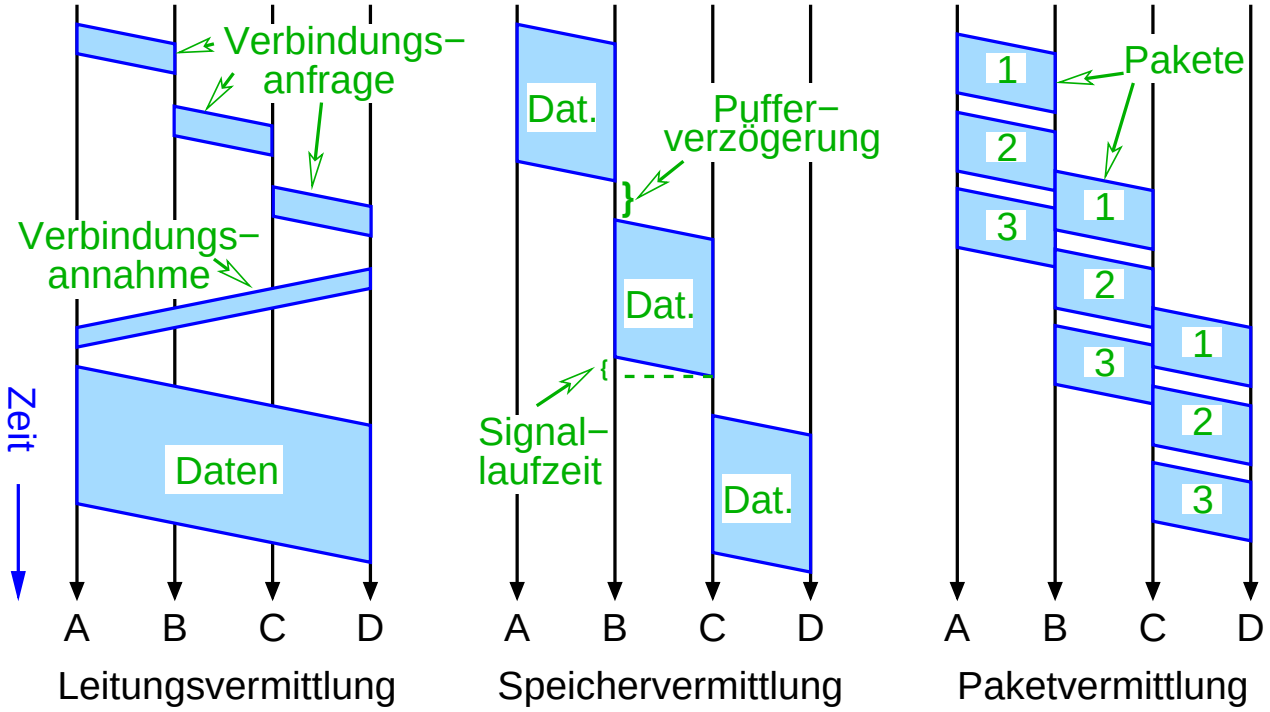
Photo by Joseph A. Carr, 1975

★★

Paketvermittlung

- Jeder Switch kann eine Anzahl von Paketen puffern
- Für jedes Paket kann der Weg unabhängig gewählt werden





★★

Zeitablauf der Datenübertragung

1.5 Anforderungen an Netze

- Unterstützung gemeinsamer Dienste
  - Netzwerk stellt Kanäle zwischen **Anwendungen** bereit
- Zuverlässigkeit
  - Bitfehler (z.B. durch elektrische Störungen)
  - Paketverlust (z.B. bei Pufferüberlauf)
  - Ausfall von Leitungen bzw. Vermittlungsknoten
  - Garantierte Paketreihenfolge?
- Sicherheit
  - Abhören von Daten, Manipluation von Daten, ...
- Leistung
  - Bandbreite, Latenz, Jitter

1.6 Leistungsparameter

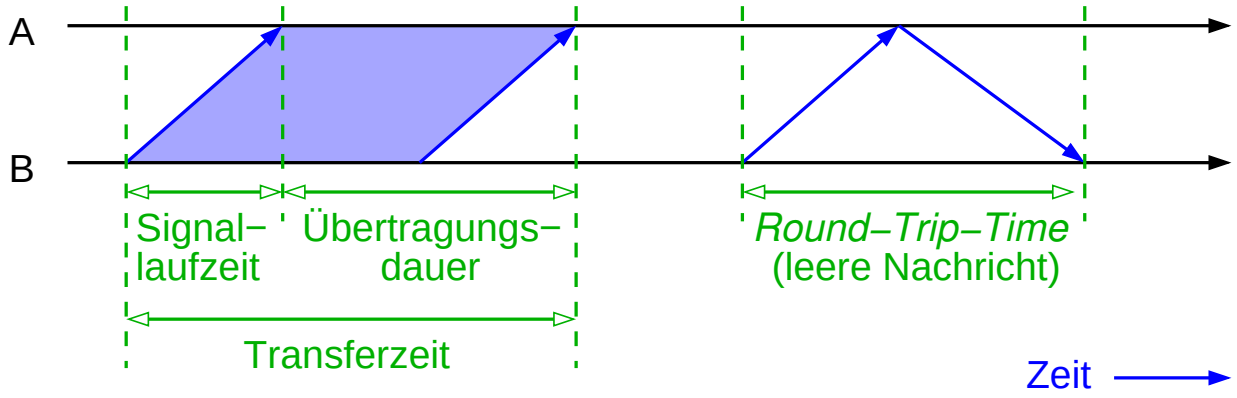
- **Übertragungsrate (Bandbreite)**
  - Übertragbares Datenvolumen pro Zeiteinheit
  - Maßeinheit: Bits pro Sekunde (**b/s** bzw. **bps**)
  - Vorsicht bei den Maßeinheiten:
    - 1 kb/s = 1000 Bits/Sekunde, 1 Mb/s = 1000 kb/s
    - 1 KB = 1024 Bytes, 1 MB = 1024 KB
      (nach NIST: KiB statt KB, MiB statt MB)
  - Unterscheidung:
    - Übertragungsrate der Leitung
    - Ende-zu-Ende Übertragungsrate (zw. Anwendungen)
- **Durchsatz**: tatsächlich erreichte Übertragungsrate
  - Durchsatz = Transfergröße / Transferzeit

1.6 Leistungsparameter ...

- **Transferzeit**
  - Zeit vom Beginn des Absendens einer Nachricht bis zu ihrem vollständigen Empfang
- ***Round-Trip-Time* (RTT)**
  - Zeit, um eine (leere) Nachricht von A nach B und wieder zurück zu schicken
- **Latenz**
  - Transferzeit einer **leeren** Nachricht
  - **Achtung**: der Begriff Latenz wird manchmal auch allgemein als Synonym für Transferzeit verwendet!

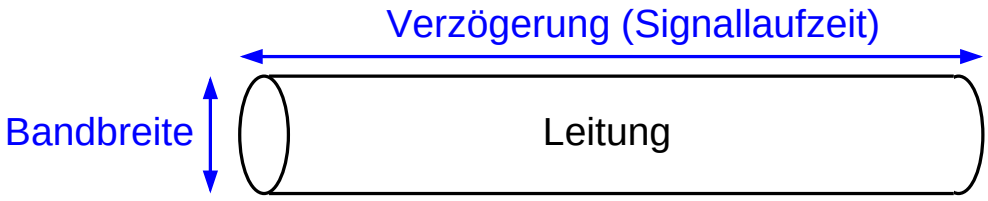
1.6 Leistungsparameter ...

- Bestandteile der Transferzeit:
  - Transferzeit = Signallaufzeit + Übertragungsdauer + Zeit für Pufferung in (Zwischen-)Knoten
  - **Signallaufzeit** = Entfernung / Lichtgeschwindigkeit
    - Lichtgeschwindigkeit im Kupferkabel $\approx 2 \cdot 10^8$ m/s
  - **Übertragungsdauer** = Nachrichtengröße / Übertragungsrate



1.6 Leistungsparameter ...

- Übertragungsrate vs. Signallaufzeit
  - Kurze Nachrichten: Signallaufzeit dominiert
  - Lange Nachrichten: Übertragungsrate dominiert
- **Verzögerungs-Bandbreiten-Produkt**
  - Gibt an, wieviele Bits sich in Übertragung („in der Leitung“) befinden



  - Z.B. Transatlantik-Kabel (3,2 Tb/s, Signallaufzeit 50 ms): $1{,}6 \cdot 10^{11}$ Bit $\approx$ 18,6 GB

1.6 Leistungsparameter ...

- **Jitter**
  - Varianz der Latenz einer Verbindung
  - Verursacht durch Pufferung und Konkurrenz um eine Verbindung
  - Folge: Datenpakete treffen in unregelmäßigen Abständen ein
  - Problem z.B. bei Audio-/Videoübertragung

1.7 Zusammenfassung

- Netz besteht aus Knoten und Verbindungen
  - Rekursiver Aufbau: Knoten verbinden Subnetze
- Paketweise Übertragung der Daten
- Jede Anwendung stellt andere Anforderungen an ein Netzwerk
- Leistungsparameter: Bandbreite und Latenz

Nächste Lektion:

- Netzwerkarchitektur: Schichten und Protokolle

Rechnernetze I

SoSe 2024

2 Protokolle und Protokollhierarchie

2 Protokolle und Protokollhierarchie ...

Inhalt

- Einführung
- Protokolle und Dienste
- Die OSI-Architektur
- Die Internet-Architektur

- Peterson, Kap. 1.3
- CCNA, Kap. 3

2.1 Einführung

Teilaufgaben bei der Kommunikation in Rechnernetzen

- Bestimmung eines Weges vom Sender zum Empfänger (**Routing**)
- Aufteilen der Daten in Pakete (wegen Multiplexing und Zwischenspeicherung), Zusammenbau beim Empfänger (in der richtigen Reihenfolge)
- Fehlerbehandlung: was, wenn ein Paket verlorengeht?
  - Quittierung der Pakete
  - nach Ablauf bestimmter Zeit: Sendung wiederholen
  - jetzt aber Behandlung von Kopien des Pakets nötig!
- **Flußkontrolle**
  - Empfänger an Sender: „nicht so schnell!“

2.1 Einführung ...

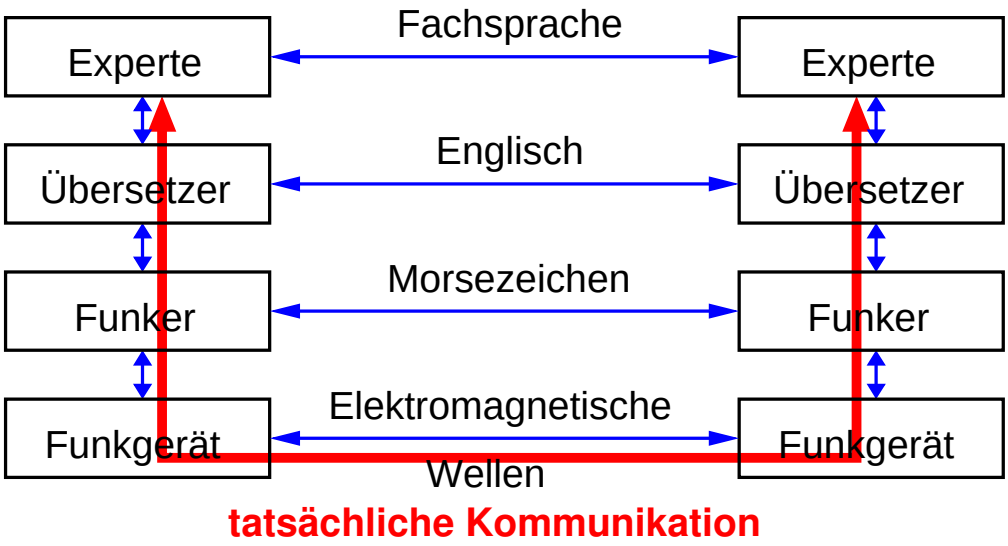
Teilaufgaben bei der Kommunikation in Rechnernetzen ...

- Behandlung verschiedener Datendarstellungen bei Sender und Empfänger (Formate, Byte-Reihenfolge...)
- Verschlüsselung der Daten?
- Bei Mehrfachzugriffs-Verbindungen: Regelung des Zugriffs auf das Übertragungsmedium
- Festlegung des Übertragungsmediums: Kabel / Funk, Stecker, Frequenzen, ...
- Kodierung und Format der Daten bei der Übertragung über dieses Medium (z.B. wie wird eine 1 bzw. 0 dargestellt?)
- ...

(Animierte Folie)

Ordnung ins Chaos: Schichten und Protokolle

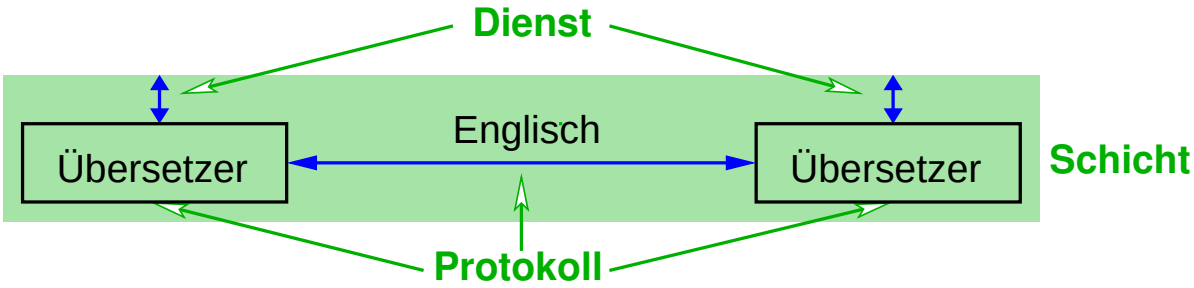
- Beispiel: zwei Experten unterhalten sich



(Animierte Folie)

Ordnung ins Chaos: Schichten und Protokolle

- Beispiel: zwei Experten unterhalten sich



2.2 Protokolle und Dienste

- Netzwerksysteme werden in Schichten organisiert

Anwendungsprogramme
Prozeß−zu−Prozeß−Kanäle
Host−zu−Host−Konnektivität
Hardware

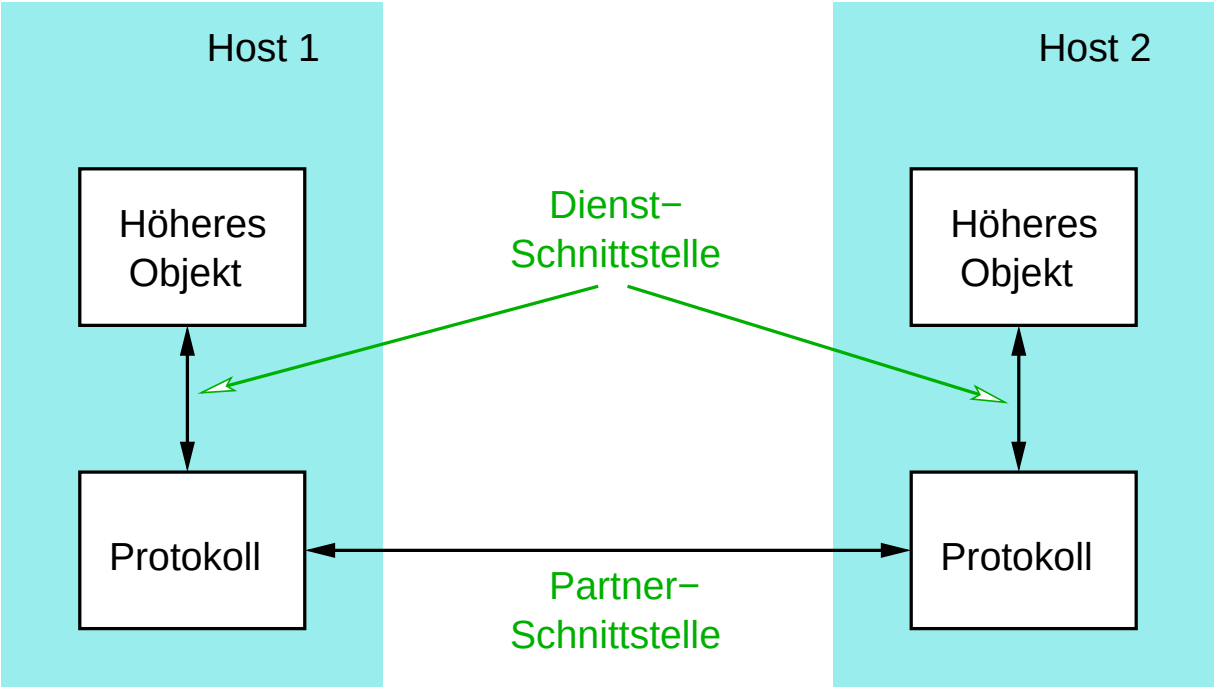
- Ziel der Schichtung:
  - jede Schicht definiert eine Abstraktionsebene
  - jede Schicht bietet eine definierte Schnittstelle
  - Implementierung der Schicht ist austauschbar

2.2 Protokolle und Dienste ...

Protokolle

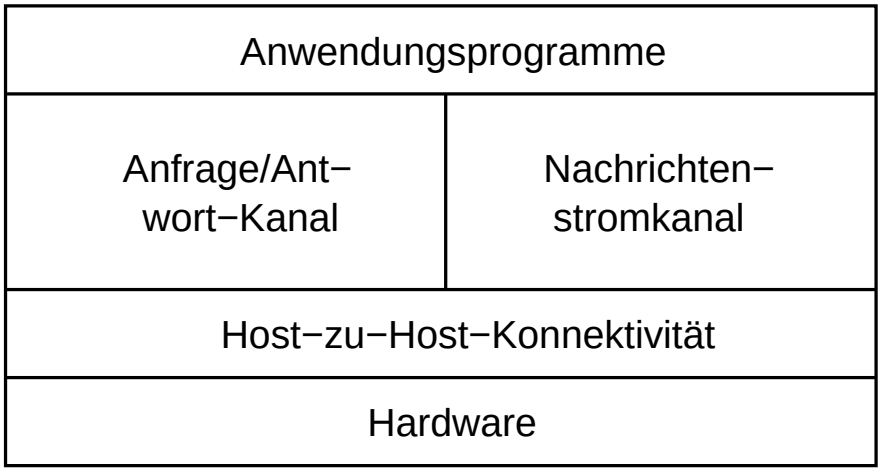
- Objekte, aus denen sich die Schichten zusammensetzen
- Bieten Objekten der höheren Ebene Kommunikationsdienste an
- Ein Protokoll bietet zwei Schnittstellen:
  - **Dienst-Schnittstelle (*Service interface*)**
    - für Nutzer der Dienste auf demselben Rechner
  - **Partner-Schnittstelle (*Peer-to-Peer Interface*)**
    - zu seinem Gegenstück auf dem anderen Rechner
- **Achtung**: Der Begriff Protokoll ist überladen:
  - Partner-Schnittstelle
  - Implementierung dieser Schnittstelle

Dienst-und Partnerschnittstellen



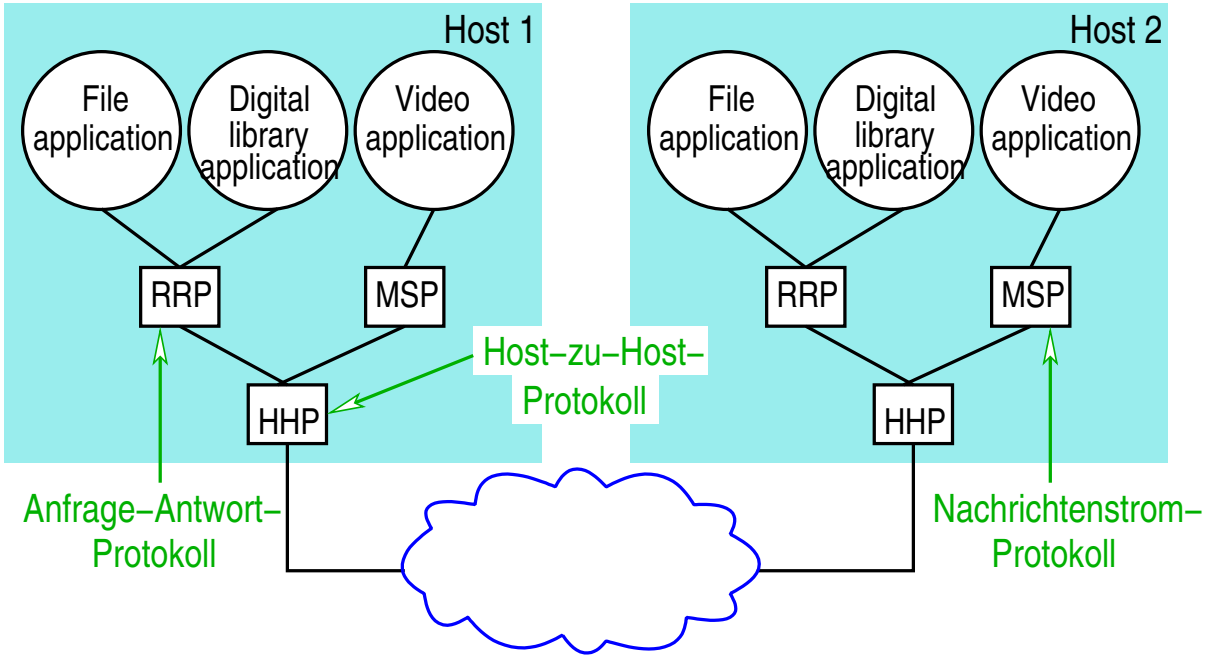
Protokollgraphen

- Häufig unterschiedliche Abstraktionen / Dienste in einer Schicht
  - Realisiert durch unterschiedliche Protokolle



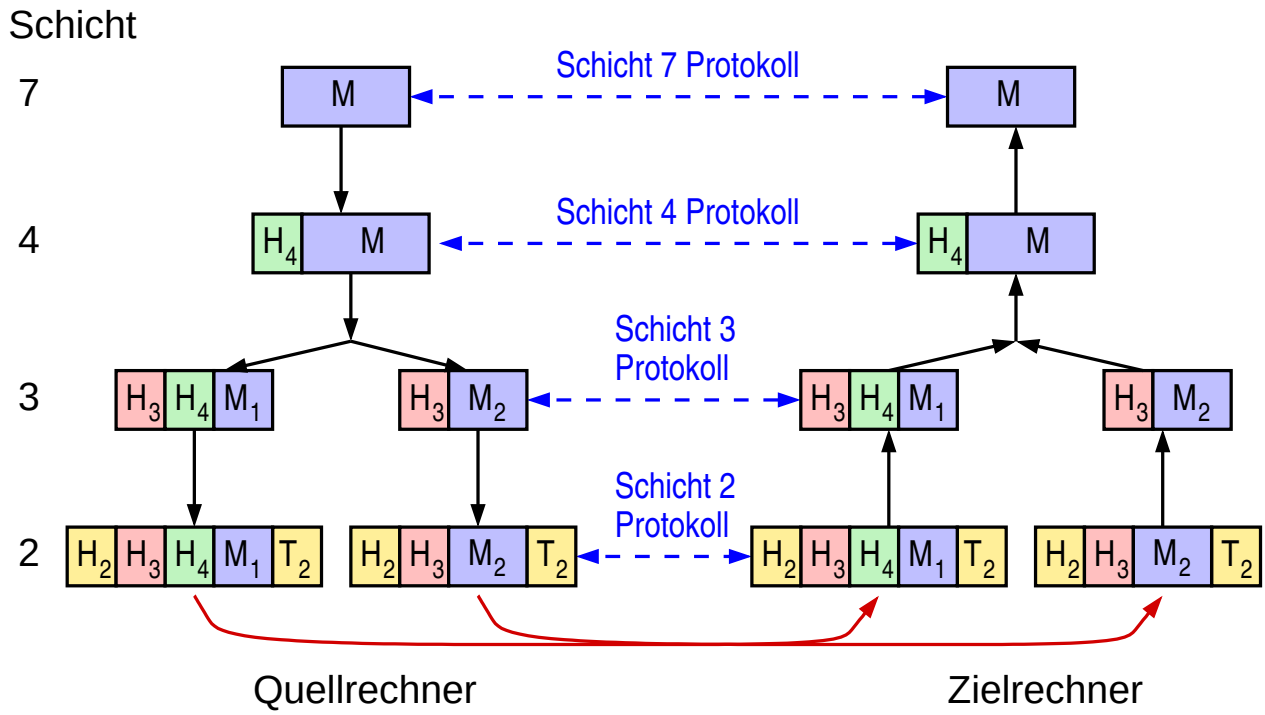
Protokollgraphen ...

➥ **Protokollgraph** stellt Abhängigkeiten zwischen Protokollen dar:



(Animierte Folie) ★★

Beispielhafter Informationsfluß zwischen den Schichten



Anmerkungen zu Folie 51:

Falls Sie sich am Ende der Vorlesung die Folie nochmals ansehen, ist im folgenden eine konkrete Situation beschrieben, die in der Internet-Architektur zu dem dargestellten Informationsfluss führen könnte:

- Eine Anwendung (Schicht 7) will eine 2 KByte große Nachricht per UDP an einen Zielprozess senden.
- UDP (Schicht 4) stellt den Daten einen UDP Header voran und übergibt sie zur Übertragung an IP.
- IP (Schicht 3) stellt fest, daß das UDP-Paket größer ist als die MTU des verwendeten Netzwerks (1500 Bytes) und damit fragmentiert werden muß (☞ **5.4**). Es teilt die Daten daher in zwei IP-Pakete auf.
- Die IP-Pakete (Fragmente) werden anschließend über Ethernet (Schicht 2) übertragen und beim Empfänger wieder zusammengebaut.

Die Schichten-Nummern entsprechen dabei denen des OSI-Schichtenmodells.

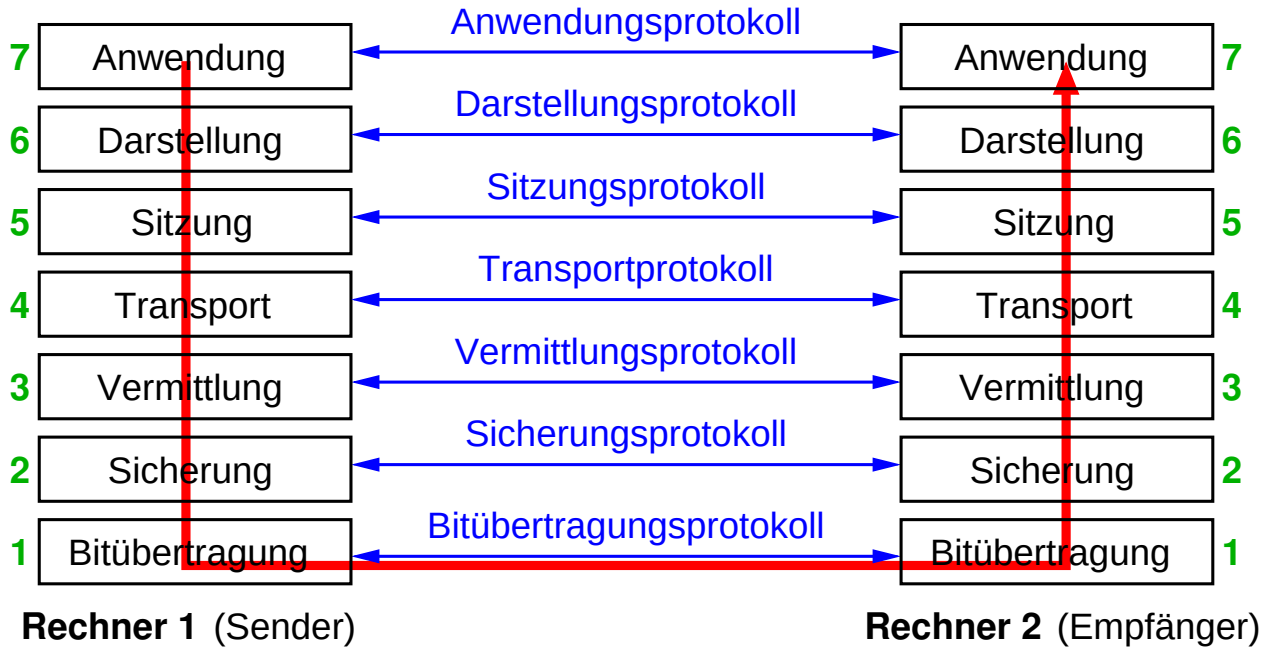
51-1

2.3 Die OSI-Architektur

★★★

Das ISO/OSI Referenzmodell

- OSI: *Open Systems Interconnection*



★

Vorbemerkung: Begriffe

- **PDU** (*Protocol Data Unit*)
  - Dateneinheit, die ein Protokoll überträgt
- **Segment**: PDU der Transportschicht
- **Paket**: PDU der Vermittlungsschicht
- **Frame**: PDU der Sicherungsschicht

★★

Schicht 1: Bitübertragungsschicht (*Physical Layer*)

- Übertragung einzelner „roher“ Bits
- Elektrische Spezifikation
  - Medium: Kabel, Glasfaser, Funk, Infrarot, ...
  - Spannungspegel, Frequenzen, Lichtwellenlänge, ...
  - Zeitverhalten
  - Codierung und Modulationsverfahren
  - Übertragung in nur eine oder beide Richtungen?
- Mechanische Spezifikation
  - Form / Art der Stecker und Kabel
  - Anzahl der Pins, ...

★★

Schicht 2: Sicherungsschicht (*Data Link Layer*)

- **Zugriffskontrolle (MAC, *Media Access Control*)**
  - physische Adressierung der Kommunikationspartner
  - regelt Zugriff auf das gemeinsam genutzte Medium
    - nur bei Mehrfachzugriffs-Verbindungen
- **LLC (*Logical Link Control*)**
  - sichert Datenübertragung auf einer Verbindung
    - Fehlerbehandlung, Flußkontrolle
  - Daten sind in Frames aufgeteilt (typ. $\sim$ 100-1000 Byte)
    - Frame durch Header und Trailer begrenzt
    - Trailer enthält Redundanzbits (z.B. Prüfsumme) zur Fehlererkennung bzw. –korrektur

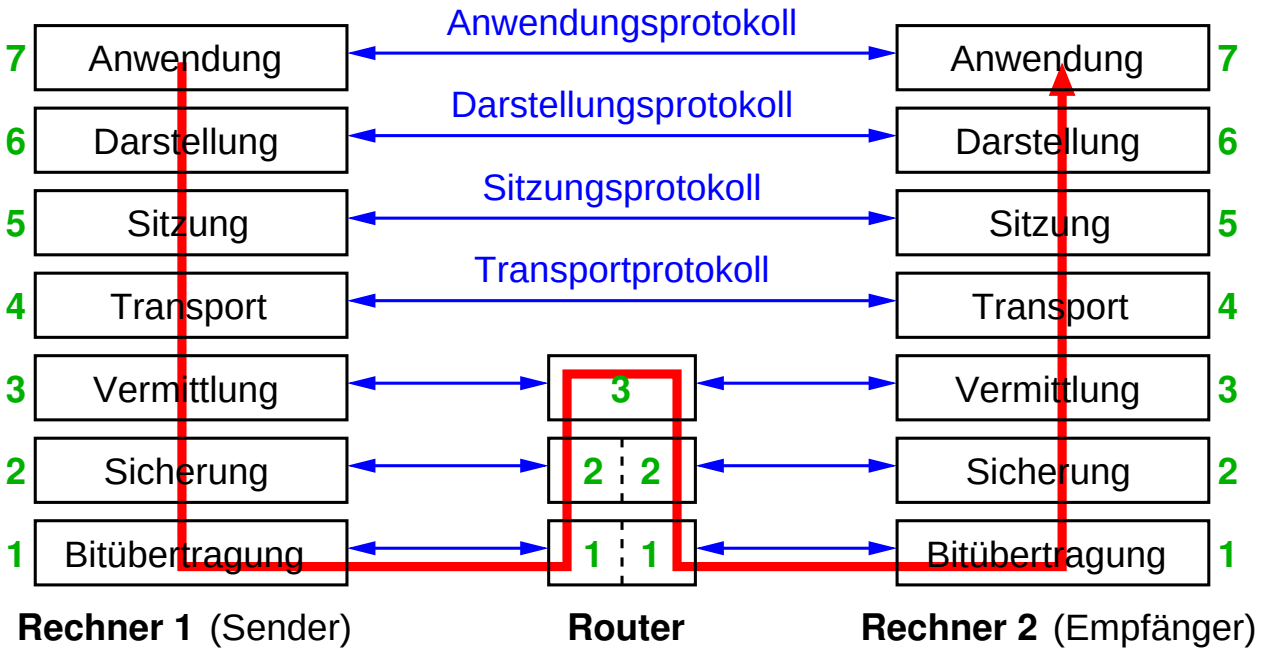
★★

Schicht 3: Vermittlungsschicht (*Network Layer*)

- Unterste Schicht, die Kommunikation zwischen nicht direkt verbundenen Netzwerk-Knoten ermöglicht
  - Host-zu-Host-Kommunikation
- Oberste Schicht der Netzwerk-Zwischenknoten
  - definiert Schnittstelle der Subnetze
- Definiert einheitliches Adressierungsschema (logische Adressen)
- Hauptaufgabe: **Routing** = Bestimmung eines Weges zwischen Sender und Empfänger
  - statisch, nur aufgrund der Verbindungstopologie
  - dynamisch, z.B. lastabhängig
- Beispiel: IP-Protokoll im Internet

★★

Netzwerk-Zwischenknoten (Router) im OSI-Modell



★★

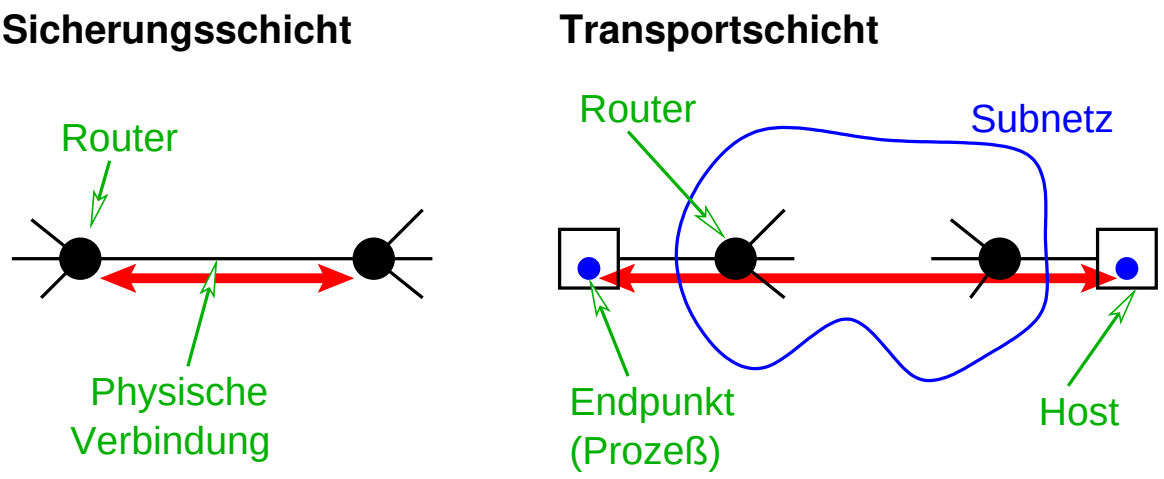
Schicht 4: Transportschicht (*Transport Layer*)

- Ermöglicht Kommunikation zwischen Endpunkten (Prozessen) auf verschiedenen Rechnern
  - Ende-zu-Ende-Kommunikation
- Stellt i.a. auch verbindungsorientierte Dienste bereit
  - Kommunikationspartner erhalten den Eindruck einer Leitungsvermittlung
    - selbst wenn untere Schichten paketorientiert arbeiten
- Aufgaben:
  - Adressierung der zu kontaktierenden Prozesse
  - Multiplexing von Kommunikationen
  - ggf. Auf- und Abbau von Verbindungen
- Beispiel: TCP-Protokoll im Internet

★★

Schicht 4: Transportschicht (*Transport Layer*) ...

- Sichert ggf. auch Datentransport zwischen Endpunkten
  - u.a. Fehlerbehandlung, Flußkontrolle
  - Abgrenzung der Schichten:



★★

Schicht 5: Sitzungsschicht (*Session Layer*)

- Dienste zur Verwaltung von Sitzungen, z.B.
  - Dialogsteuerung („wer darf wann senden?“)
  - atomare Aktionen („alles oder gar nichts“)
  - Synchronisierung (z.B. Weiterführung eines unterbrochenen Transfers)

Schicht 6: Darstellungsschicht (*Presentation Layer*)

- Unterste Schicht, die die Semantik der Daten kennt
- Konvertiert Datenformate und –darstellung
- Auch: Kompression, Verschlüsselung
- Schicht 5 und 6 heute i.a. in Anwendungsschicht integriert!

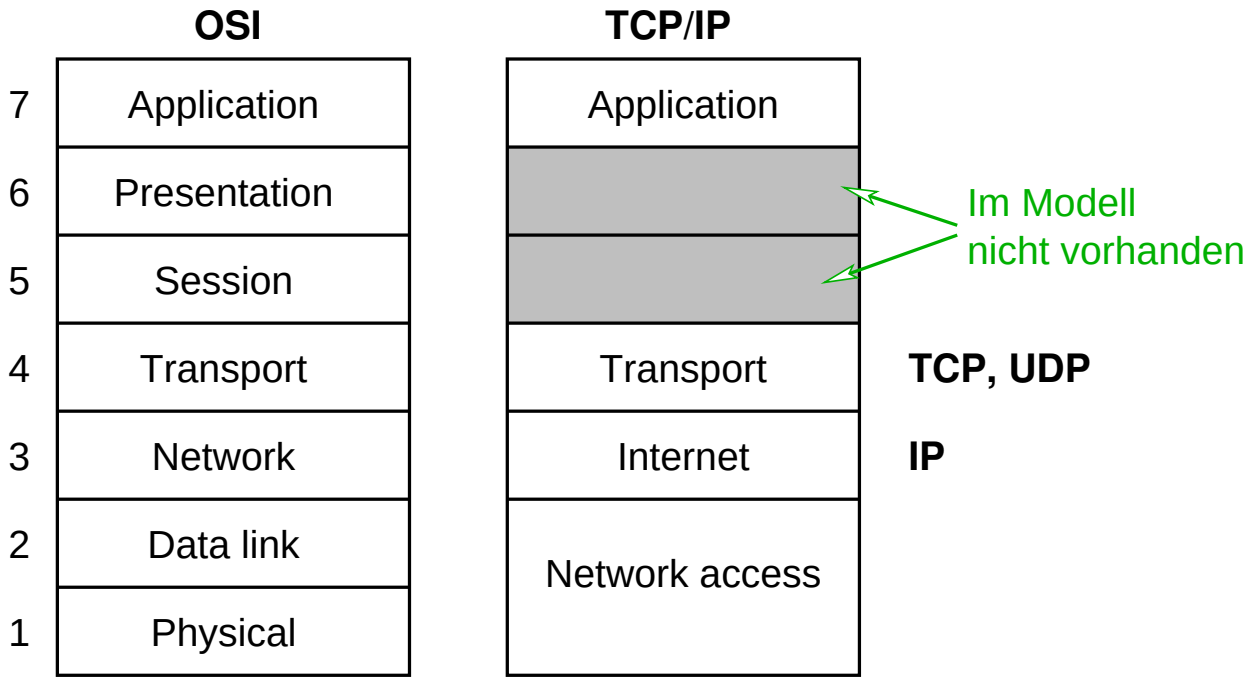
★★

Schicht 7: Anwendungsschicht (*Application Layer*)

- Spezialisierte Dienste und Protokolle für verschiedene Anwendungsbereiche
- Beispiele:
  - HTTP (*Hypertext Transport Protocol*)
    - zur Übertragung von Web-Seiten
  - SMTP (*Simple Mail Transport Protocol*)
    - zum Austausch von Email
  - SMB (*Server Message Block*) / NFS (*Network File System*)
    - Protokolle für Netzwerk-Dateisysteme
  - SSH (*Secure Shell*)
    - sicheres Protokoll zur Nutzung entfernter Rechner

★★

Die Internet-Architektur im Vergleich mit OSI

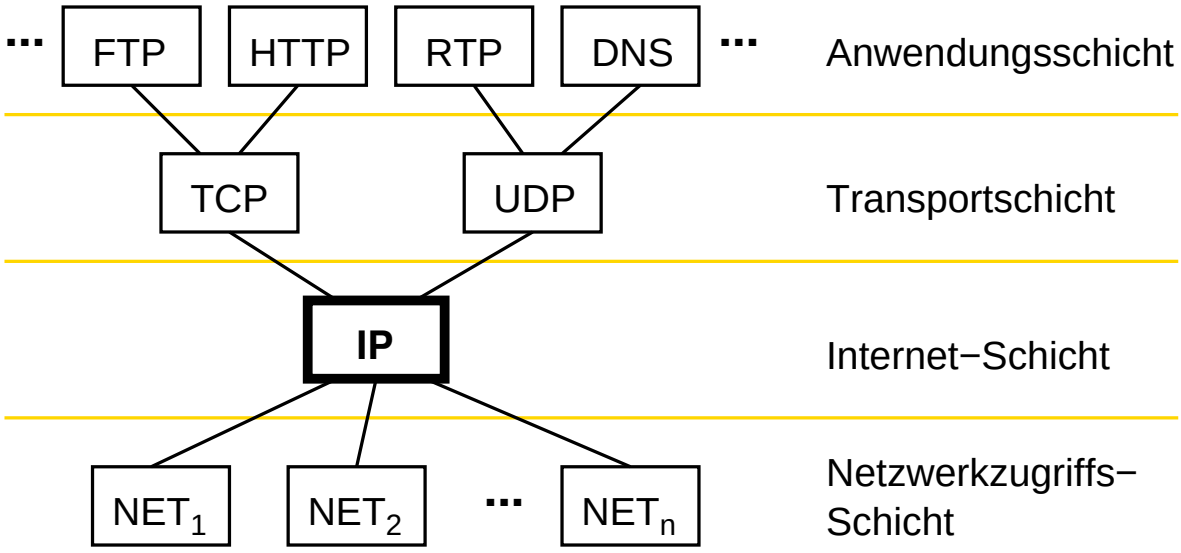


★★

Schichten der Internet-Architektur

- Netzwerk-Zugriffsschicht (*Network access*)
  - wird nicht von der Internet-Architektur spezifiziert
    - d.h. das IP-Protokoll kann auf beliebige Netzwerke aufgesetzt werden
- Internet-Schicht
  - ein zentrales Protokoll: **IP (*Internet Protocol*)**
    - verbindungslos, paketvermittelt, unzuverlässig
- Transportschicht
  - **TCP (*Transmission Control Protocol*)**
    - verbindungsorientiert, zuverlässig
  - **UDP (*User Datagram Protocol*)**
    - verbindungslos, unzuverlässig

Protokollgraph der Internet-Architekur



- Sanduhr-Modell: IP als zentrale Verbindung der höheren Protokolle und der Netzwerk-Zugriffsschicht

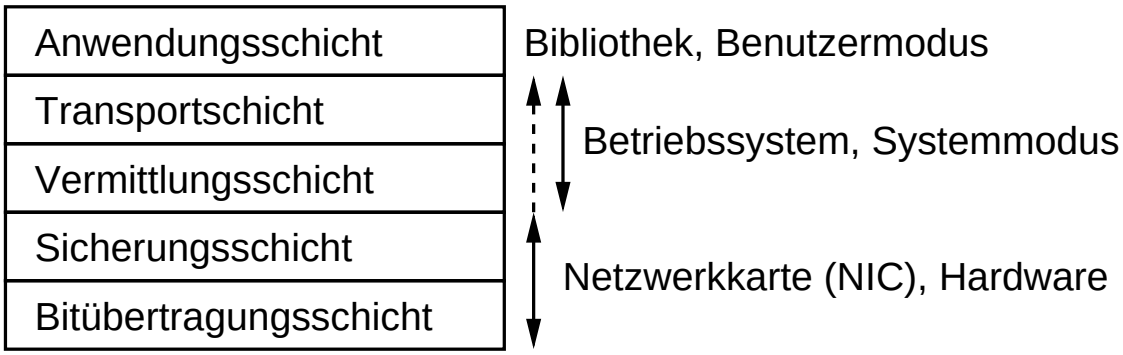
★★

Adressierung von Hosts im Internet

- Anwendungsschicht: **Hostname**
  - z.B. www.bs.informatik.uni-siegen.de
- Vermittlungsschicht: **IP-Adresse** (logische Adresse)
  - z.B. 141.99.179.6
- Sicherungsschicht: **MAC-Adresse** (physische Adresse)
  - z.B. 1a:68:25:f0:a3:d9

2.5 Implementierung von Protokollen

- Typische Implementierung der OSI-Schichten:



- Zur Vermeidung von Kopieroperationen: Nachrichten oft als Liste von Blöcken realisiert
  - erlaubt Hinzufügen / Entfernen von Headern ohne Kopie
  - Netzwerkkarten realisieren heute typisch direkten Speicherzugriff mit Scatter/Gather
- Z.T. auch Funktionen der Schicht 3/4 durch NIC realisiert

2.6 Zusammenfassung

- Schichten, Protokolle und Dienste
- ISO-OSI Referenzmodell
  - 7 Schichten: Bitübertragung, Sicherung, Vermittlung, Transport, Sitzung, Darstellung, Anwendung
- Internet Protokollarchitektur
  - Netzwerk-Zugriff, IP, TCP/UDP, Anwendung

Nächste Lektion:

- Direktverbindungsnetze
  - Codierung, Framing, Fehlererkennung und -korrektur
  - Medienzugriffssteuerung (MAC), Ethernet

Rechnernetze I

SoSe 2024

3 Direktverbindungsnetze

Inhalt

- Hardware-Bausteine: Knoten und Verbindungsleitungen
- Grundlagen zur Datenübertragung
- Modulation
- Codierung
- Framing
- Fehlererkennung und Fehlerkorrektur
- Medienzugriffssteuerung (MAC)
  - Allgemeines
  - Ethernet (CSMA-CD)

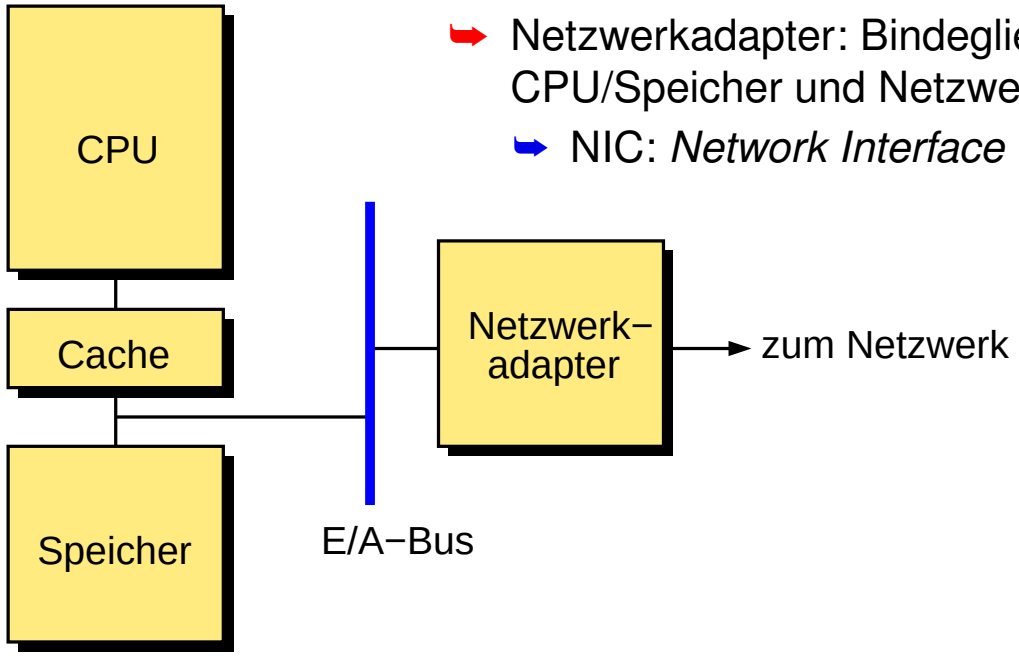
- Peterson, Kap. 2.1 – 2.6, 2.7.2
- CCNA, Kap. 4, 5.1

3.1 Hardwarebausteine

OSI: 1

Aufbau eines Knotens

- Netzwerkadapter: Bindeglied zwichen CPU/Speicher und Netzwerk-Schnittstelle
  - NIC: *Network Interface Controller*



Verbindungs„leitungen“

- Übertragen Signale als elektromagnetische Wellen
- Typische Attribute:
  - Frequenz- bzw. Wellenlängenbereich (Bandbreite)
  - Dämpfung (max. Kabellänge)
  - Richtung des Datenflusses
    - **Simplex**: nur in eine Richtung
    - **Vollduplex**: in beide Richtungen, gleichzeitig
    - **Halbduplex**: in beide Richtungen, abwechselnd
- Grundlegende Arten:
  - Kupferkabel
  - Glasfaserkabel (Lichtwellenleiter)
  - Drahtlose Verbindung (Funk, IR) (☞ **RN_II**)

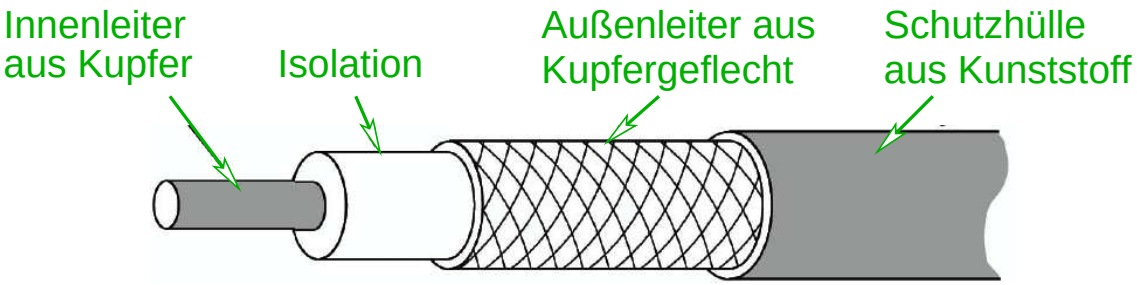
Anmerkungen zu Folie 71:

Die OSI-Schicht 1 hat eine gewisse Sonderrolle im Schichtenmodell, da es die einzige Schicht ist, die keine darunterliegende Schicht hat. Das bedeutet, dass die tatsächliche Kommunikation (im Gegensatz zu allen andern Schichten) auf derselben Schicht stattfindet. Daher müssen auf der Bitübertragungsschicht neben dem Dienst, den diese Schicht nach oben anbietet, auch die Eigenschaften der tatsächlichen Verbindung definiert werden.

Auf dieser Folie sind dabei nicht nur die reinen Eigenschaften der Verbindungsleitung genannt, sondern teilweise auch die Art ihrer Verwendung (z.B. bei „Frequenz- bzw. Wellenlängenbereich“ und bei „Richtung des Datenflusses“).

Kupferkabel: Koaxialkabel

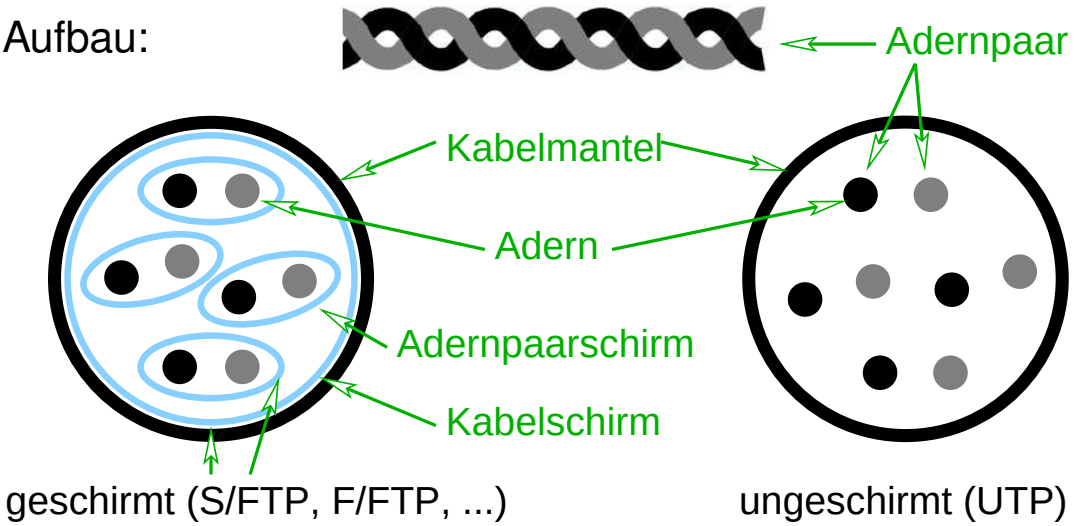
- Aufbau:



- Hohe Bandbreite, geringe Dämpfung, teuer
- Basisband-Kabel (direkte Übertragung, 1 Kanal, <500m)
  - Beispiele: Ethernet (10BASE-5, 10BASE-2)
- Breitband-Kabel (Modulation auf Träger, mehrere Kanäle, mehrere km)
  - Beispiel: Fernsehkabel

Kupferkabel: Twisted-Pair (verdrilltes) Kabel

- Aufbau:

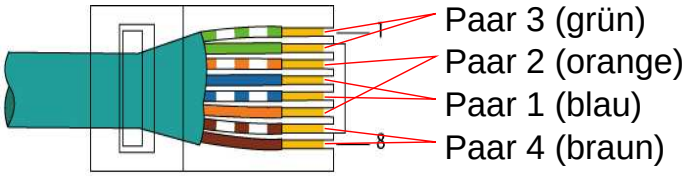


- Geringe Kosten, relativ gute Bandbreite
- Beispiel: Fast Ethernet (100BASE-TX)

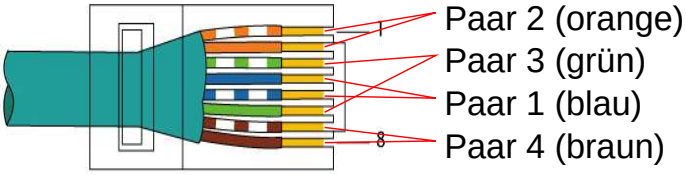
Beispiel: Fast Ethernet (100 Mb/s)

- Twisted-Pair Kabel (mind. Cat 5, UTP) mit 4 Adernpaaren
  - Paar 2: Signal vom Switch zum NIC
  - Paar 3: Signal vom NIC zum Switch

  } d.h. 2 Simplex-Leitungen
  - Paar 1 früher für analoges Telefon genutzt
- Stecker: RJ-45, zwei verschiedene Belegungen:
  - TIA-568A:



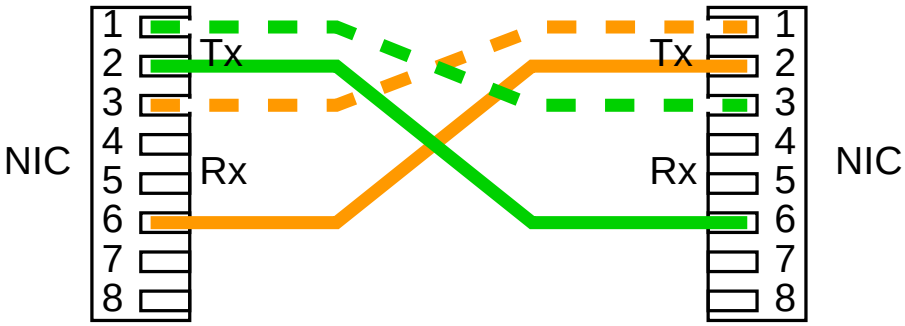
  - TIA-568B:



(Animierte Folie)

Straight-Through und *Cross-Over* Kabel

- Belegung der RJ-45 Buchsen:
  - NIC (PC, Router): Pin 1,2 = Transmit, Pin 3,6 = Receive
  - Switch/Hub: Pin 1,2 = Receive, Pin 3,6 = Transmit
- Straight-Through Kabel (beide Stecker nach 568A bzw. 568B)
- Cross-Over Kabel (ein Stecker 568A, ein Stecker 568B)



Anmerkungen zu Folie 75:

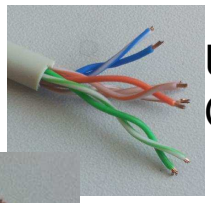
- Moderne Netzwerkkarten erkennen automatisch, wenn ein Cross-Over Kabel verwendet werden müsste und tauschen dann (nur auf einer Seite!) die Belegung intern.
- Ab 1 Gb/s Ethernet werden alle vier Adernpaare genutzt. Zudem wird auf jedem Adernpaar im Vollduplex-Betrieb übertragen.

75-1

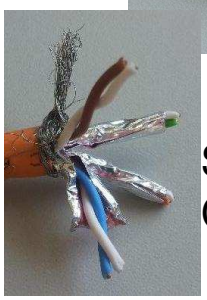
3.1 Hardwarebausteine ...

Twisted-Pair Kabelkategorien

- Cat. 3: bis 16 MHz, Telefon + 10 Mb/s Ethernet
- Cat. 5: 100 MHz, 100 Mb/s Ethernet
- Cat. 6: 250 MHz, 1 Gb/s Ethernet
- Cat. 6a: 500 MHz, 1 Gb/s Ethernet
- Cat. 7: 600 MHz, 10 Gb/s Ethernet
- Cat. 7a: 1 GHz, 10 Gb/s Ethernet
- Cat. 8: 2 GHz, 40 Gb/s Ethernet



UTP
Cat. 5

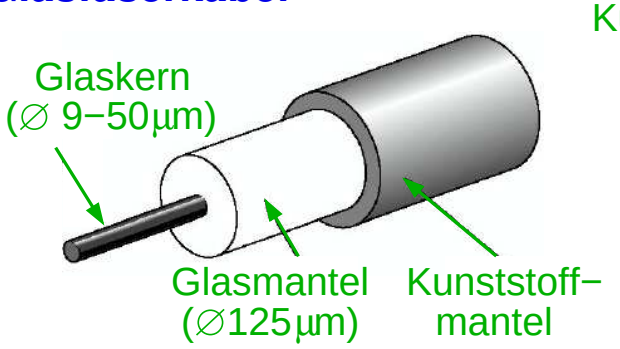


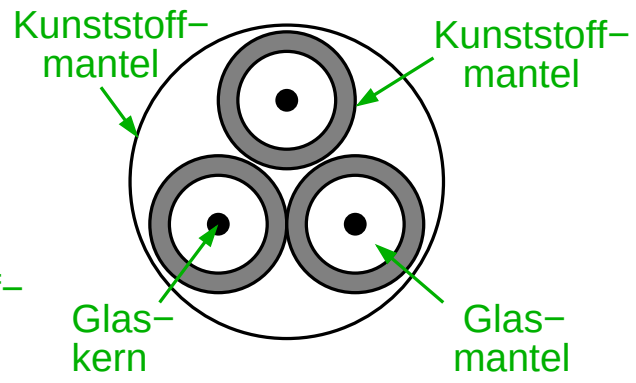
S/FTP
Cat. 7

- Unterschiede: Material, Verdrillung, Adernschirmung
- Maximale Länge bei Ethernet: 100 m

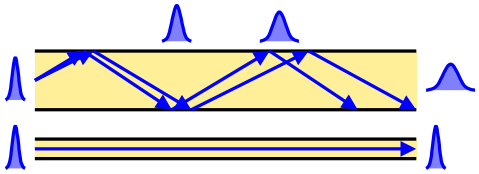
(Animierte Folie)

Glasfaserkabel





- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)



3.2 Grundlagen zur Datenübertragung

OSI: 1

Synchrone und asynchrone Übertragung

- Synchrone Übertragung:
  - serielle Übertragung eines **Bit**stroms
  - Bits müssen taktsynchron gesendet werden
  - Empfänger muss Sendetakt rekonstruieren können
  - z.B. Ethernet

- Asynchrone Übertragung:
  - serielle Übertragung eines Stroms von **Zeichen**
  - Zeichen kann zu beliebiger Zeit gesendet werden
  - Anfangsmarkierung durch Startbit
  - Empfänger muss Takt nur für ein einzelnes Zeichen halten
  - z.B. serielle Schnittstelle, V.24 / RS 232

Qualitätsmerkmale von Leitungen

- Grenzfrequenz / Bandbreite (in Hz)
  - bei Basisbandübertragung (ohne Modulation): höchste Frequenz, die die Leitung noch übertragen kann
  - bei Modulation auf ein Trägersignal (Breitbandkabel, Funk): Breite des verfügbaren Frequenzkanals
- Dämpfung (in dB)
  - welcher Teil der eingespeisten Leistung kommt am anderen Ende an?
  - logarithmisches Maß: 10 dB = Faktor 10, 20 dB = Faktor 100 ...
- Übersprechen (in dB)
  - bei Kabeln mit mehreren Adernpaaren: Einstrahlung von Leistung aus anderen Adernpaaren

Anmerkungen zu Folie 79:

- Die begrenzte Bandbreite bei Basisbandübertragung kommt dadurch zustande, daß die Leitung höhere Frequenzen stärker dämpft als niedrige. Insofern ist Definition einer scharf definierten Bandbreite eine Idealisierung.
- Dämpfung und Übersprechen sind abhängig von der Leitungslänge; sie stehgen mit zunehmender Länge der Leitung.
- Die Abschirmung einer Leitung dient der Reduktion der Dämpfung durch verminderte Abstrahlung elektromagnetischer Wellen; die Abschirmung der einzelnen Adernpaare reduziert das Übersprechen.

Maximal erreichbare Bitrate einer Leitung

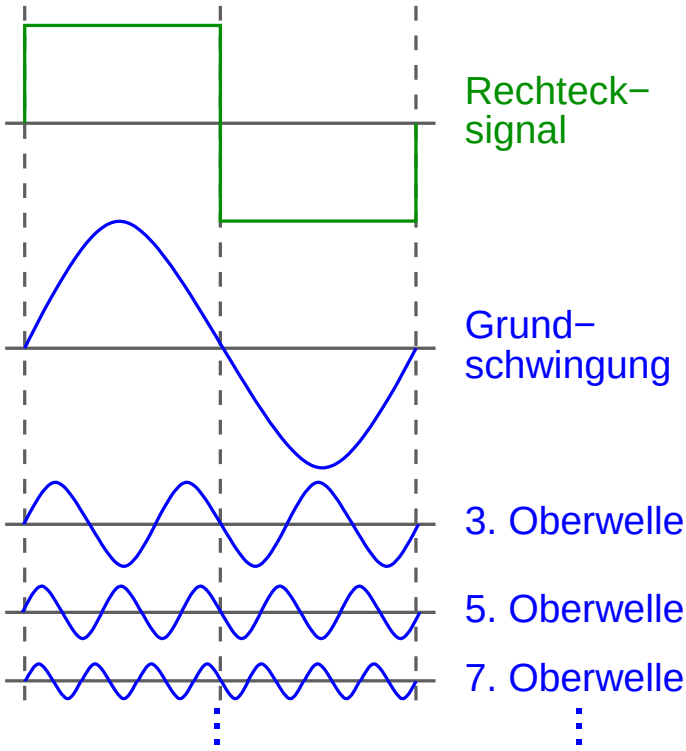
- Frage: Welche Übertragungsrate (bit/s) ist auf einer Leitung mit gegebener Grenzfrequenz (Bandbreite) möglich?
- Antworten liefern:
  - Nyquist-Theorem
  - Shannon'sches Theorem
- Hilfsmittel: Fourier-Analyse
  - jedes (periodische) Signal läßt sich als Summe von Sinusschwingungen darstellen

Beispiel zur Fourier-Analyse

- Rechtecksignal:

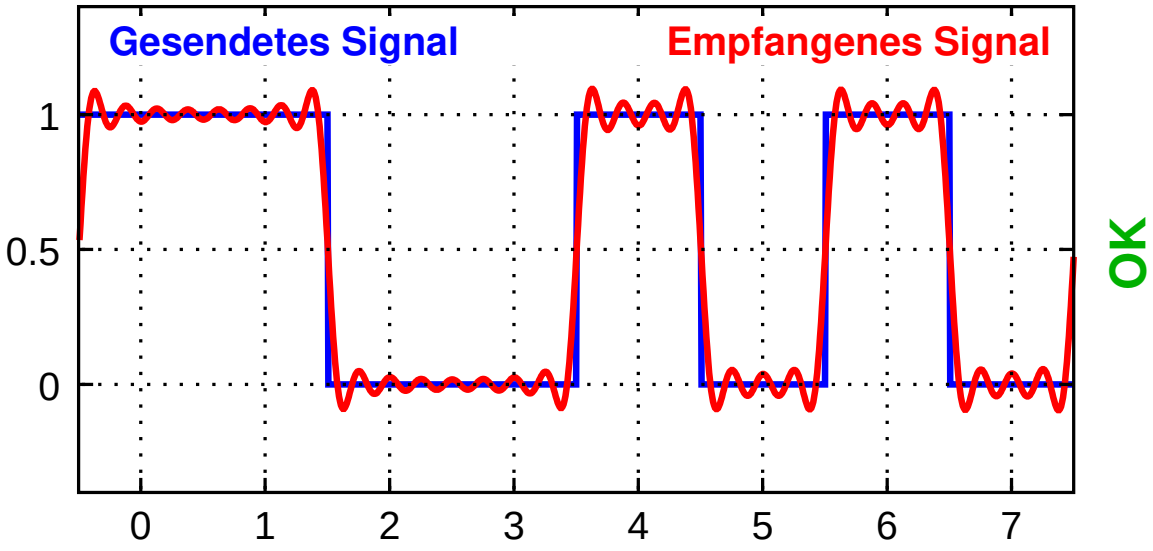
$$\sum_{k=1}^{\infty} \frac{4 \cdot \sin((2k-1)\omega t)}{(2k-1)\pi}$$

- Damit u.a. Auswirkungen der Grenzfrequenz einfach zu ermitteln
  - summiere nur die Oberwellen mit kleinerer Frequenz



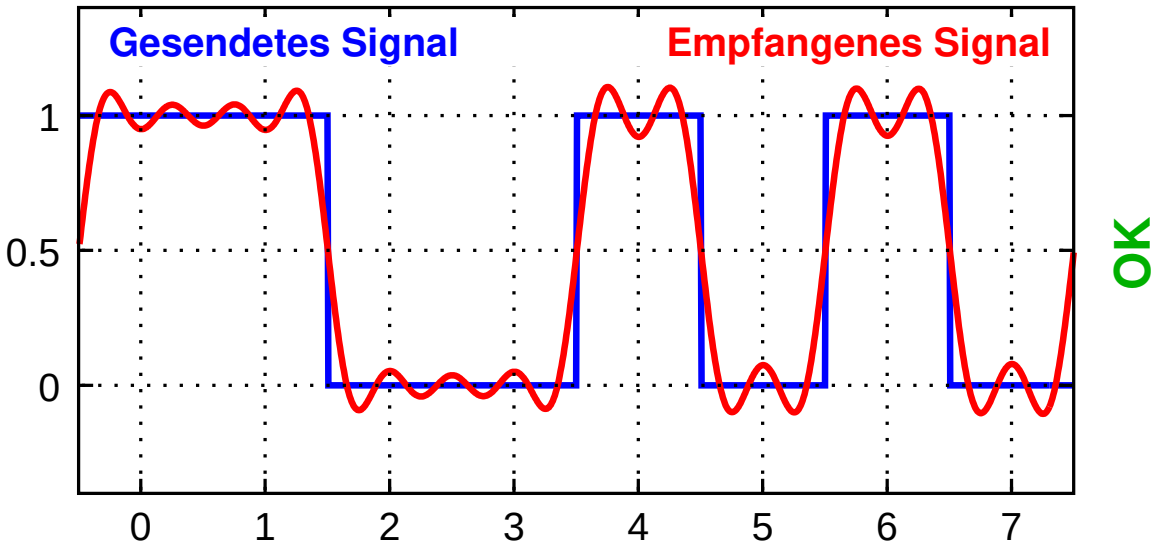
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes, NRZ-Codierung, 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 4 MHz



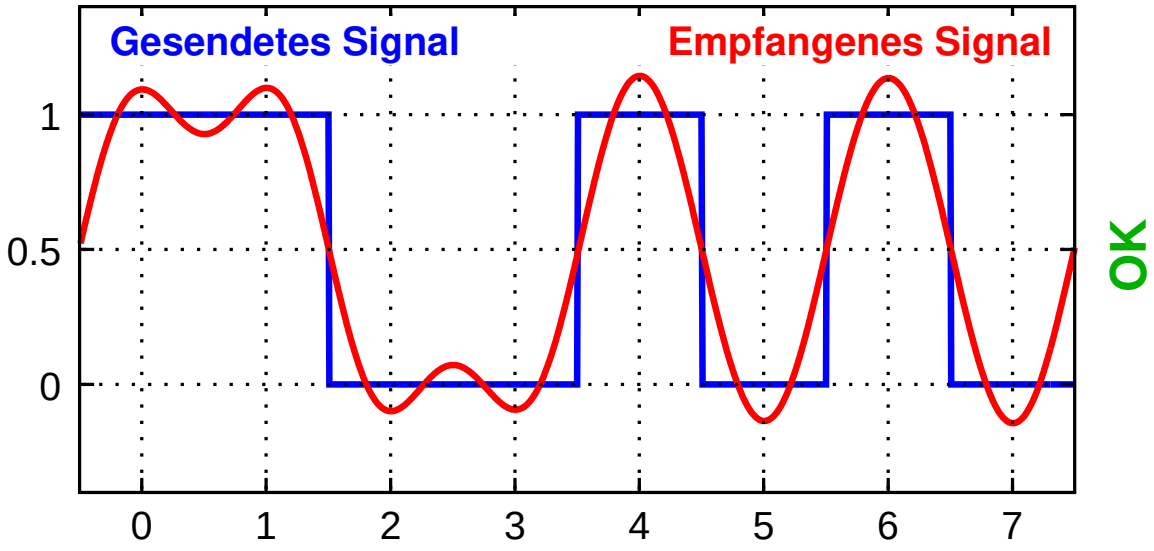
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes, NRZ-Codierung, 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 2 MHz



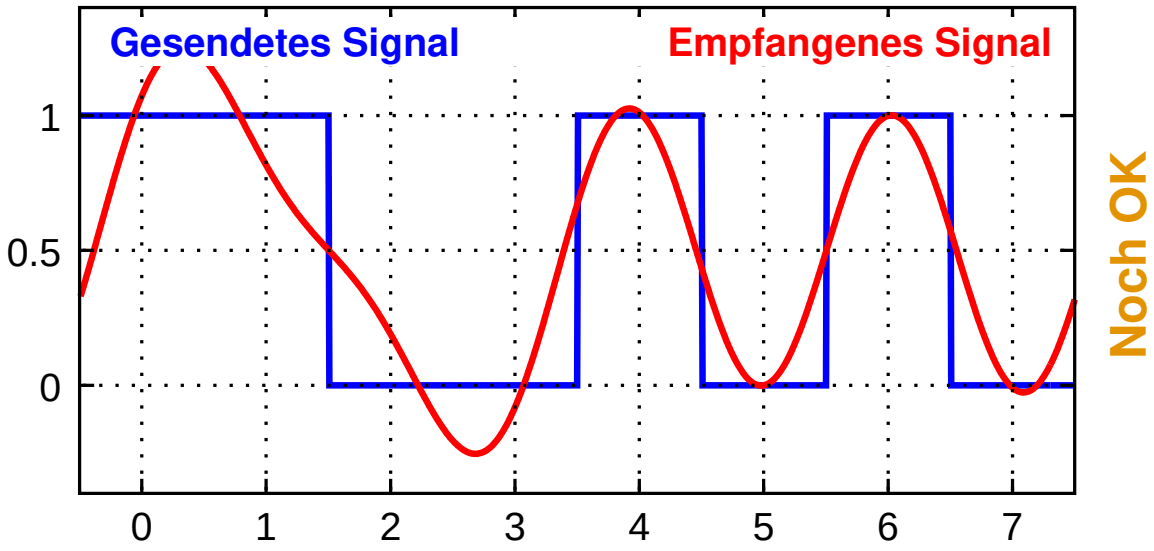
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes, NRZ-Codierung, 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 1 MHz



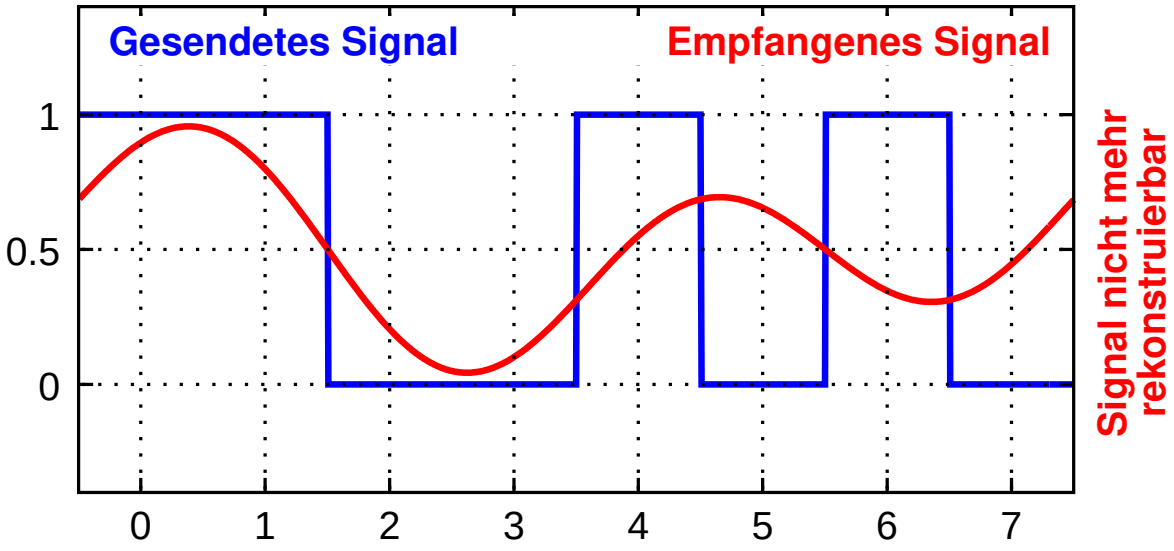
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes, NRZ-Codierung, 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 500 kHz



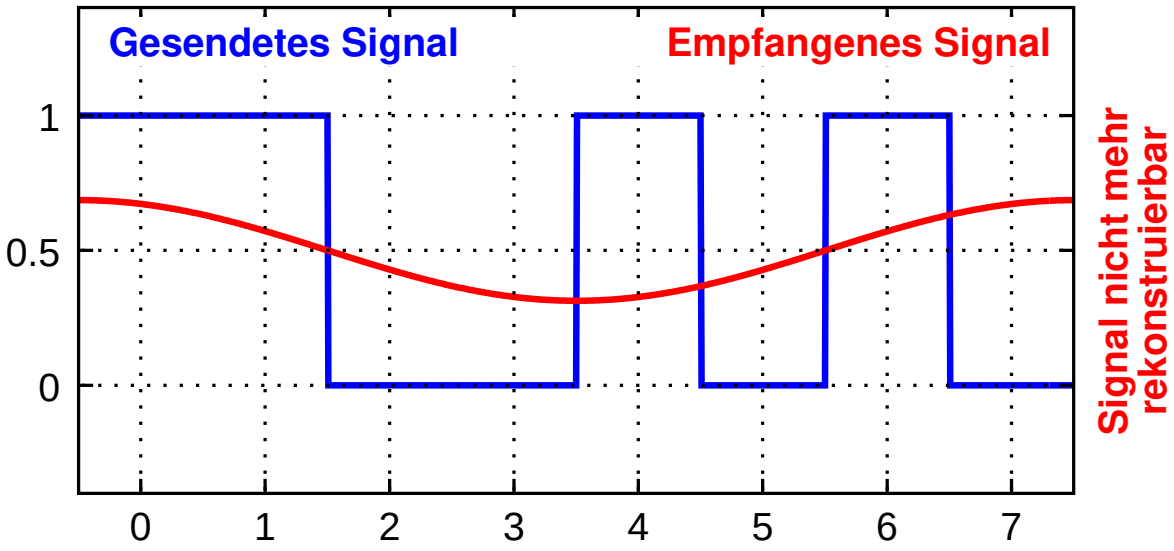
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes, NRZ-Codierung, 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 250 kHz



Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes, NRZ-Codierung, 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 125 kHz



Nyquist-Theorem (Abtasttheorem)

- Ein Signal mit Grenzfrequenz H [Hz] kann mit $2 \cdot H$ (exakten) Abtastwerten pro Sekunde vollständig rekonstruiert werden
- Die maximal sinnvolle Abtastrate ist daher $2 \cdot H$ [1/s]
- Folgerung für Übertragung mit 1 Bit pro Abtastung:
  - maximale Datenübertragungsrate = $2 \cdot H$ [b/s]
  - siehe Beispiel: 1 Mb/s erfordert 500 kHz Grenzfrequenz
- Höhere Übertragungsraten sind möglich, wenn pro Abtastung mehr als 1 Bit gewonnen wird (genauere Abtastung)
  - für n Bit pro Abtastung: 2^n Zustände (z.B. Spannungspegel)
  - Übertragungsrate ist dann begrenzt durch das **Rauschen**
    - z.B. Störeinstrahlung, thermisches Rauschen

Shannon'sches Theorem

- Max. Datenübertragungsrate = $H \cdot \log_2(1 + S/N)$
- S/N = **Rauschabstand** (**Signal/Rauschverhältnis**)
  - (Leistungs-)Verhältnis von Signalstärke zu Rauschen
    - beeinflusst u.a. durch Dämpfung und Übersprechen der Leitung
  - definiert maximale Genauigkeit der Abtastung

Zur Unterscheidung von Übertragungs- und Abtastrate

- Einheit **b/s** für Übertragungsrate
- Einheit **Baud** (Zeichen/s) für Abtastrate

Anmerkungen zu Folie 89:

Das Nyquist-Theorem besagt, dass wir maximal $2 \cdot H$ Abtastungen durchführen können. Wenn die maximale Datenübertragungsrate $H \cdot \log_2(1 + S/N)$ ist, erhalten wir dieser Abtastrate

$$\frac{H \cdot \log_2(1 + S/N)}{2 \cdot H} = \frac{\log_2(1 + S/N)}{2} = \log_2(\sqrt{1 + S/N})$$

Bits pro Abtastung.

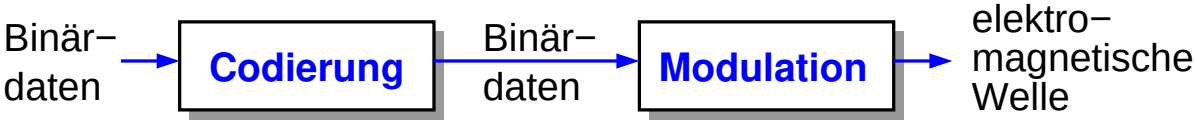
Die Wurzel kommt daher, dass das Signal-/Rauschverhältnis als Leistungsverhältnis angegeben wird, nicht als Spannungsverhältnis. Wenn die Signalspannung 4-mal so groß ist wie die Rauschspannung, können wir 4 Spannungsbereiche unterscheiden und damit 2 Bits an Information gewinnen. Das Signal-/Rauschverhältnis ist dabei 16 (4-fache Spannung $\Rightarrow$ 16-fache Leistung), und $\log_2(\sqrt{1 + 16}) = 2.0437...$ ist gerade etwas größer als 2.

3.3 Modulation

OSI: 1

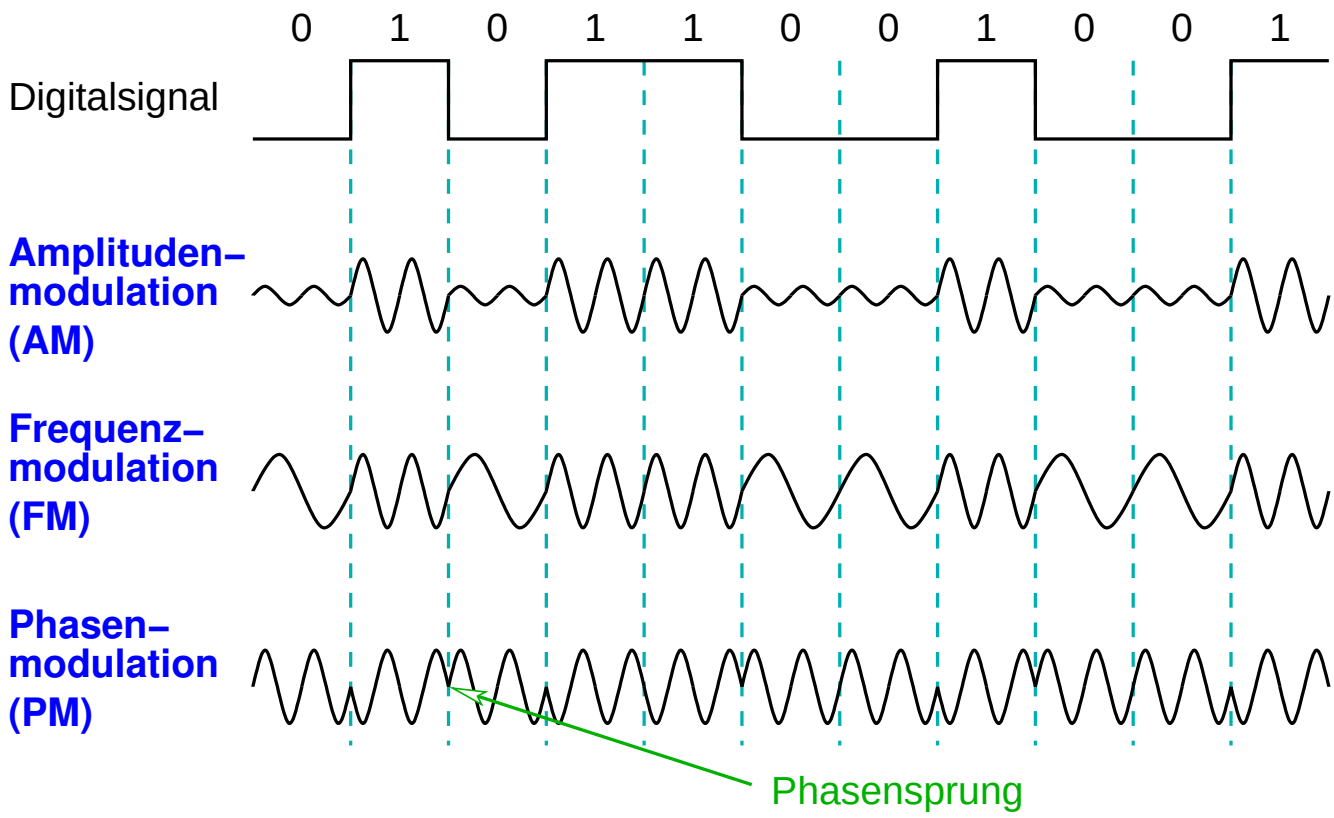
★★

- Zur Übertragung müssen Binärdaten (digitale Signale) in analoge elektrische Signale (elektromagnetische Wellen) umgesetzt werden
- Umsetzung in zwei Schritten:



- **Modulation:**
  - Variation von Frequenz, Amplitude und/oder Phase einer Welle
  - zur Überlagerung der (Träger-)Welle mit dem Nutzsignal
    - z.B. bei Funk, Modem, Breitbandkabel, ...
  - (entfällt bei Basisband-Übertragung)

3.3 Modulation ...

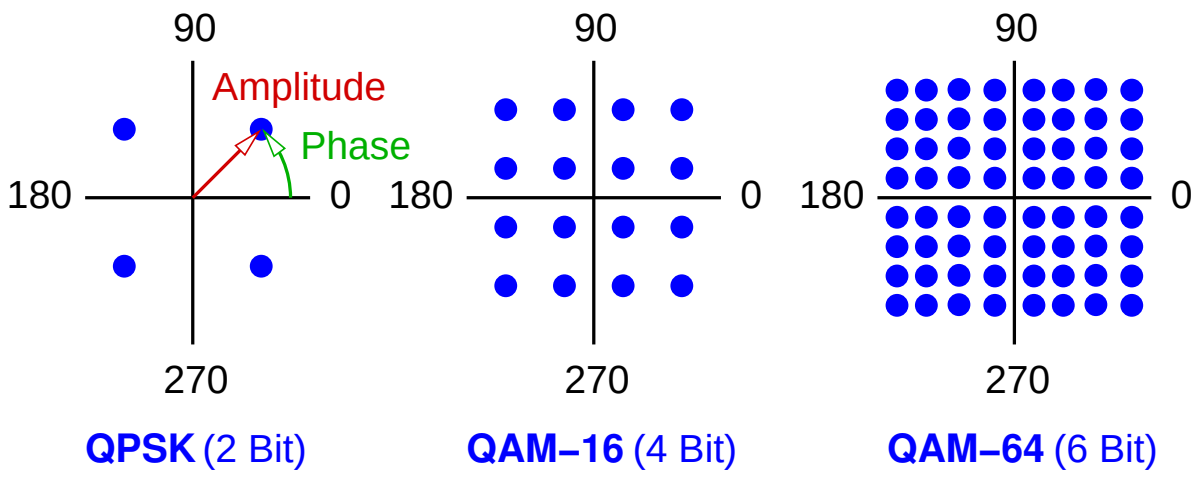


3.3 Modulation ...

(Animierte Folie)

Modulationsverfahren QPSK und QAM

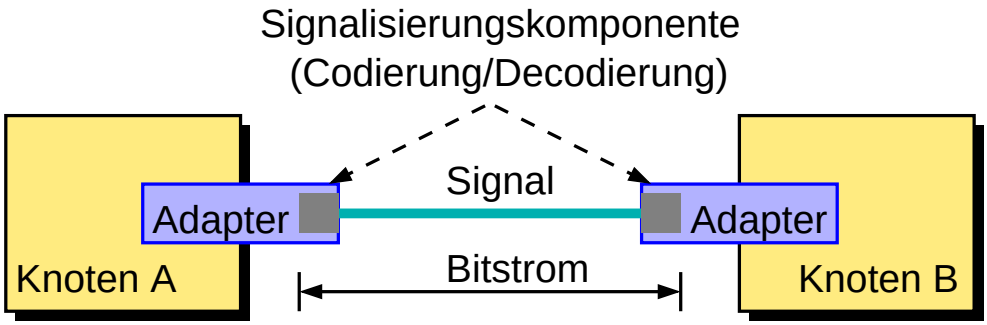
- Erleichtern die Gewinnung von mehreren Bits pro Abtastung
- Funktionsprinzip:
  - jeweils n Bits bestimmen **Amplitude** und **Phase** des Signals
- Beispiele:



3.4 Codierung

★★★

- Übertragung eines Bitstroms zwischen zwei Knoten:



- Einfachste Codierung:
  - ***Non-Return to Zero* (NRZ)**: **1** $\hat{=}$ *high*, **0** $\hat{=}$ *low*
- Probleme:
  - Festlegung der Spannungspegel für *high* und *low*
  - **Taktwiederherstellung** (**Synchronisation**)
    - wo ist die „Grenze“ zwischen zwei Bits?

Anmerkungen zu Folie 93:

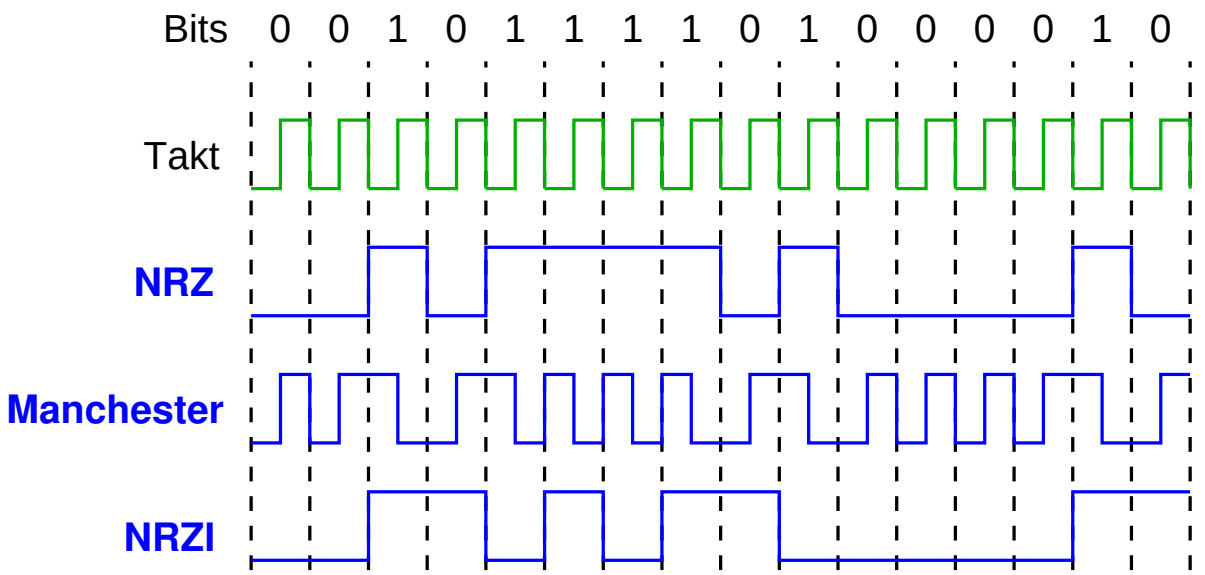
Man unterscheidet zwischen **Leitungscodierung** und **Kanalcodierung**. Wir betrachten hier zunächst ausschließlich die Leitungscodierung!

Die Kanalcodierung dient dazu, einen Bitstrom bei der Übertragung über gestörte Kanäle durch Hinzufügen von Redundanz gegen Übertragungsfehler zu schützen. Siehe dazu die Folien 109 ff.

3.4 Codierung ...

★

- Abhilfe: Codierungen mit Taktwiederherstellung



NRZI: *Non–Return to Zero Inverted*

3.4 Codierung ...

★★★

Manchester-Codierung

- Bitstrom wird mit Taktsignal EXOR-verknüpft
- Anwendung z.B. bei 10 Mb/s Ethernet
- Problem:
  - **Baudrate** (Rate, mit der das Signal abgetastet werden muß) ist doppelt so hoch wie die Bitrate
  - verschwendet Bandbreite

NRZI

- Signal wird bei jedem **1**-Bit invertiert
- Problem: keine Taktwiederherstellung bei aufeinanderfolgenden Nullen möglich

★★★

4B/5B-Codierung

- 4 Datenbits werden auf 5-Bit Codeworte so abgebildet, daß nie mehr als 3 aufeinanderfolgende Nullen übertragen werden müssen
  - jedes der 5-Bit Codeworte hat
    - höchstens eine Null am Anfang
    - höchstens zwei Nullen am Ende
  - Übertragung der Codeworte z.B. mit NRZI
  - Overhead nur noch 25%

- Bei schnellen Netzen (z.B. Fast Ethernet, GBit-Ethernet) oder auch schnellen Modems werden noch effizientere Verfahren zur Taktrückgewinnung eingesetzt

Anmerkungen zu Folie 96:

Die genaue Codetabelle der 4B/5B-Codierung ist:

Daten (4 Bit)	codierte Daten (5 Bit)
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111

Daten (4 Bit)	codierte Daten (5 Bit)
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Man erkennt, dass die Codierung auch die Länge von 1-Folgen begrenzt (maximal 8 aufeinanderfolgende 1-Bits).

Ziele der Codierung

- Taktrückgewinnung beim Empfänger
- Reduktion der benötigten elektrischen Bandbreite
  - Erhöhung der Bitrate (mehrere Bits pro Abtastung, z.B. 8B6T)
  - Reduktion der Stör-Ausstrahlung (z.B. MLT-3)
- Gleichspannungsfreiheit
  - Ziel: konstanter Mittelwert des Signals
  - erleichtert Übertragung und Erkennung von Low- und High-Pegel
  - bei 4B5B-Codierung nicht gegeben, daher z.B. 8B10B

} Leitungs-codierung

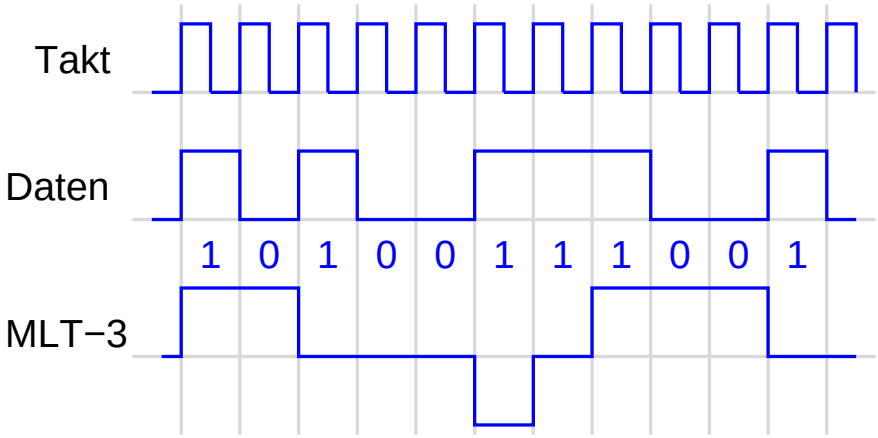
- Fehlerkorrektur — Kanalcodierung

Anmerkungen zu Folie 97:

- zu Bandbreitenreduktion:
  - Bei der 8B6T Codierung werden acht Bits auf sechs dreiwertige Signale abgebildet.
- Zu Gleichspannungsfreiheit:
  - Bei der 4B5B Codierung ist es nicht möglich, dass jedes Codewort genausoviele Nullen wie Einsen hat (da die Länge ungerade ist).
  - Bei der 8B10B Codierung haben die Codeworte entweder genausoviele Nullen und Einsen (also jeweils 5), oder 6 Nullen und 4 Einsen bzw. 4 Nullen und 6 Einsen. Im letzten Fall gibt es immer zwei alternative Codeworte, die abhängig davon ausgewählt werden, ob bisher mehr Einsen oder mehr Nullen im codierten Strom waren. Auf diese Weise ist die Differenz der Anzahl von Nullen und Einsen im codierten Strom höchstens ± 1.

Leitungscodierung beim Ethernet

- Ursprünglich: Manchester-Codierung, 20 MBaud bei 10 Mb/s
- Fast Ethernet: 4B5B-Codierung, 125 MBaud bei 100 Mb/s
  - anschließend MLT-3 Codierung (3 Spannungspegel)



  - damit Grundfrequenz nur noch maximal 31,25 MHz (statt 62,5 MHz mit NRZI)

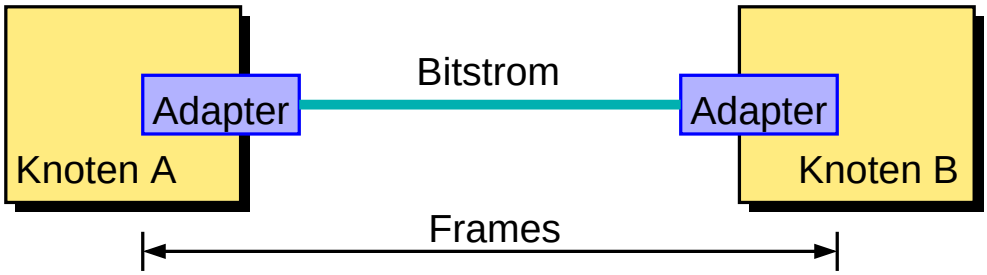
Anmerkungen zu Folie 98:

- Bei der MLT-3 Codierung wird immer die feste Folge 0, +, 0, -, 0, ... von Spannungspegeln durchlaufen: soll ein 1-Bit übertragen werden, wird zum nächsten Pegel der Folge gewechselt, bei einem 0-Bit bleibt der Pegel unverändert.

3.5 Framing

★★★

- Wir betrachten nun die Übertragung von Datenblöcken (**Frames**) zwischen Rechnern:



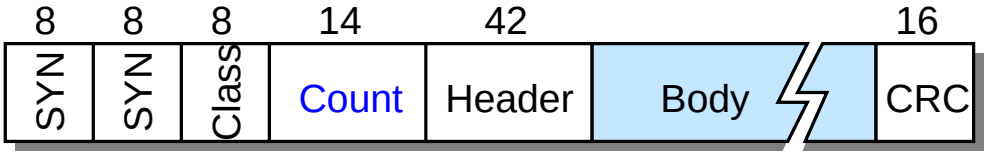
- Gründe für die Aufteilung von Daten in Frames:
  - einfaches Multiplexing verschiedener Kommunikationen
  - bei Fehler muss nur betroffener Frame neu übertragen werden
- Zentrale Aufgabe des Framings:
  - Erkennung, wo Frame im Bitstrom anfängt und wo er aufhört
  - dazu: Framegrenzen müssen im Bitstrom erkennbar sein

3.5 Framing ...

★★

Byte-Count Methode

- Frame-Header enthält Länge des Datenteils
- Beispiel: (Frame im DDCMP-Protokoll, DECNET)

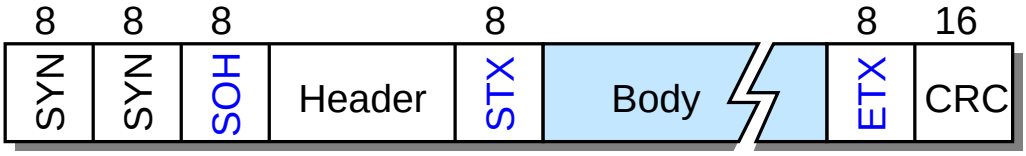


- Problem: was passiert, wenn die Länge fehlerhaft übertragen wird?
  - Frame-Ende wird nicht korrekt erkannt
  - SYN-Zeichen am Beginn jedes Frames, um (wahrscheinlichen!) Anfang des Folgeframes zu finden
- Verwendet u.a. beim ursprünglichen Ethernet

★★

Sentinel-Methode

- Frame-Ende wird durch spezielles Zeichen markiert
- Beispiel: (Frame im BISYNC-Protokoll, IBM)



- Problem: Das Endezeichen kann auch im Datenteil (Body) vorkommen
- Lösung: ***Byte-Stuffing***
  - ersetze ETX im Datenteil durch DLE ETX
  - ersetze DLE im Datenteil durch DLE DLE
  - verwendet u.a. bei PPP

★★

Sentinel-Methode ...

- Lösung: ***Bit-Stuffing***
  - Eindeutigkeit durch Einfügen von Bits in den Bitstrom erreicht
  - Beispiel: (HDLC-Protokoll)
    - Anfangs- und Endemarkierung ist 01111110_2
    - nach 5 aufeinanderfolgenden 1-Bits wird vom Sender ein 0-Bit in den Bitstrom eingeschoben
    - wenn Empfänger 5 aufeinanderfolgende 1-Bits gelesen hat:
      - nächstes Bit = 0: ignorieren, da eingeschoben
      - nächstes Bit = 1: sollte Endemarkierung sein (prüfe, ob die 0 folgt; falls nicht: Fehler)
- Lösung: Nutzung „ungültiger“ Codeworte
  - Anfang und Ende durch Codeworte markiert, die sonst nicht vorkommen (z.B. bei 4B/5B-Codierung)

Framing beim Ethernet

- Ein Ethernet-Frame enthält i.d.R. keine Längenangabe (☞ **3.1.6**)
- Ebenso wird kein Bit-/Bytestuffing verwendet
- Erkennung des Frame-Endes erfolgt auf OSI-Schicht 1!
  - 10 Mb/s Ethernet: Ausbleiben des Signalwechsels (Manchester-Codierung!)
  - Fast Ethernet: Ende durch Bitfolge 01101 00111 gekennzeichnet (ungültige 4B/5B Codeworte)

3.6 Fehlererkennung

OSI: 2

★★

- Ziel: Übertragungsfehler in Frames erkennen (und behandeln)
- Möglichkeiten zur Fehlerbehandlung:
  - Korrektur des Fehlers beim Empfänger
  - Verwerfen der fehlerhaften Frames, Neuübertragung durch das Sicherungsprotokoll (☞ **7.4**)
- Vorgehensweise: Hinzufügen von **Redundanzbits** (Prüfbits) zu jedem Frame
- Theoretischer Hintergrund: **Hamming-Distanz**
  - Hamming-Distanz d = Minimale Anzahl von Bits, in denen sich zwei Worte eines Codes unterscheiden
  - $d \geq f + 1 \quad \Rightarrow \quad f$ Einzelbitfehler erkennbar
  - $d \geq 2 \cdot f + 1 \quad \Rightarrow \quad f$ Einzelbitfehler korrigierbar
- Beispiel: Paritätsbit führt zu $d = 2$

(Animierte Folie)

Zweidimensionale Parität

- Erweiterung der einfachen Parität
- Beispiel: 6 Worte á 7 Bit
- Erkennt alle 1, 2, 3 sowie die meisten 4-Bit-Fehler
- Erlaubt auch die Korrektur von 1-Bit-Fehlern

		Paritäts-bits
Daten	0 1 0 1 0 0 1	1
	1 1 0 1 0 0 1	0
	1 0 1 1 1 1 0	1
	0 0 0 1 1 1 0	1
	0 1 1 0 1 0 0	1
	1 0 1 1 1 1 1	0
Paritäts-byte	1 1 1 1 0 1 1	0

Anmerkungen zu Folie 105:

Die zweidimensionale Parität hat keine praktische Bedeutung in Rechnernetzen, sondern dient hier nur zur Veranschaulichung, wie eine Fehlerkorrektur erreicht werden kann.

(Animierte Folie) ★★

CRC (*Cyclic Redundancy Check*)

- Ziel: hohe Warscheinlichkeit der Fehlererkennung mit möglichst wenig Prüfbits
- Basis des CRC-Verfahrens: Polynomdivision mit Modulo-2-Arithmetik (d.h. Add./Subtr. entspricht EXOR)
- Idee:
  - jede Nachricht *M* kann als Polynom $M(x)$ aufgefaßt werden, z.B.
    - *M* = 1 0 0 1 1 0 1 0 (7 6 5 4 3 2 1 0) (Bits 7, 4, 3, 1 sind 1)
    - $M(x) = x^7 + x^4 + x^3 + x^1$
  - wähle Generatorpolynom $C(x)$ vom Grad k
  - erweitere *M* um k Prüfbits zu Nachricht *P*, so daß $P(x)$ ohne Rest durch $C(x)$ teilbar ist

(Animierte Folie)

CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht *M*: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator *C*: 1101
- Divisionsrest *R* wird an die Nachricht *M* angefügt
- Versendete Nachricht *P*: 10011010101
- Diese Nachricht ist durch den Generator ohne Rest teilbar:
  - $R(x) = M(x) \mod C(x)$
    $\Rightarrow (M(x) - R(x)) \mod C(x) = 0$

```
1 0 0 1 1 0 1 0 0 0 0   ← Nachricht um 3 Bit erweitert
1 1 0 1                 ← Generator
---------
  1 0 0 1
  1 1 0 1
  -------
    1 0 0 0
    1 1 0 1
    -------
      1 0 1 1
      1 1 0 1
      -------
        1 1 0 0
        1 1 0 1
        -------
            1 0 0 0
            1 1 0 1
            -------
              1 0 1   ← Rest
```

CRC (*Cyclic Redundancy Check*) ...

- Wahl des Generatorpolynoms?
  - So, daß möglichst viele Fehler erkannt werden!
  - Beispiel für ein übliches CRC-Polynom:
    - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-16 erkennt:
    - alle Ein-und Zweibitfehler
    - alle Fehler mit ungerader Bitanzahl
    - alle Fehlerbündel mit Länge $\leq$ 16 Bit
- Gründe für den Einsatz von CRC:
  - Gute Fehlererkennung
  - Sehr effizient in Hardware realisierbar

3.7 Nachtrag: Kanalcodierung

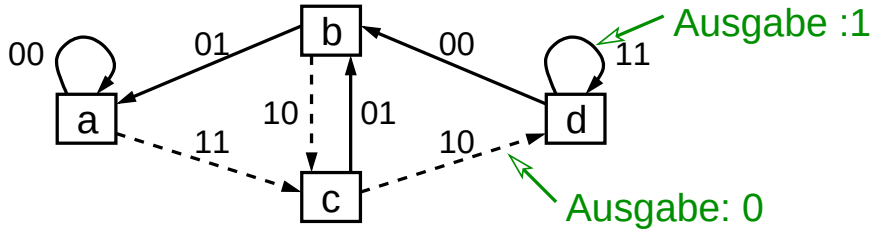
OSI: 1

- Ziel: Fehlerkorrektur bei der Bitübertragung (OSI-Schicht 1)
- *Linear Block Codes*
  - Codeworte fester Länge werden durch längere Codeworte mit Redundanz ersetzt
    - z.B. Anfügen eines Paritätsbits an jedes übertragene Byte
- *Convolutional Codes*
  - **Strom** von Eingabezeichen (bzw. -bits) wird durch Zustands-automat um Redundanzbits erweitert
    - Redundanzbit kann von allen bisherigen Eingabezeichen abhängen
  - beim Dekodieren wird der wahrscheinlichste **Pfad** durch die Zustände gesucht (Viterbi Decoder)

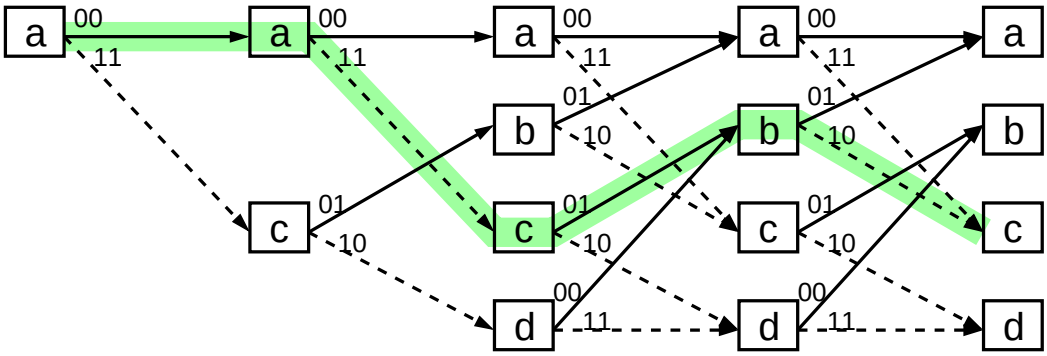
(Animierte Folie)

Convolutional Codes: Beispiel

- Automat zur Decodierung:



- „Abgewickelter“ Zustandsgraph (Trellis-Diagramm):



Empfangener Bitstrom: 00 11 01 10 –> 1 0 1 0

Anmerkungen zu Folie 110:

- Durchgezogene Übergänge bedeuten eine Dekodierung als 1-Bit, gestrichelte eine Dekodierung als 0-Bit.
- Der fehlerhaft empfangene Strom wird richtig dekodiert, da der Pfad mit den wenigsten notwendigen Änderungen verwendet wird.

3.8 Medienzugriffssteuerung (MAC)

OSI: 2

★★

- In vielen LANs:
  - Knoten greifen auf ein gemeinsames Medium zu
  - Zugriff muß geregelt werden, um **Kollisionen** zu vermeiden:
    - zu jeder Zeit darf nur jeweils ein Knoten senden

- Typische Vorgehensweisen:
  - statische Aufteilung
  - *Random Access* Verfahren
    - z.B. CSMA/CD (Ethernet)
  - kollisionsfreie Verfahren (für Echtzeit-Anwendungen)
    - z.B. Token-Ring

3.8.1 Statische Verfahren

- Feste Aufteilung des Mediums auf die Stationen ($\sim$ Multiplexing)
  - häufig aufgrund vorheriger Reservierung ($\rightarrow$ Dynamik)
- FDMA (*Frequency Division Multiple Access*): Zuteilung eines (unterschiedlichen) Frequenzbands
  - z.B. Kabelfernsehen, Mobilfunk
- TDMA (*Time* $\sim$): Zuteilung fester Zeitschlitze
  - häufig in Netzen für Automatisierungssysteme
- CDMA (*Code* $\sim$): gleichzeitiges Senden auf gespreiztem Frequenzband mit verschiedenen Codierungen
- SDMA (*Space* $\sim$): Fokussierung des Funkstrahls auf das Gebiet des jeweiligen Station
  - z.B. Mobilfunk-Netze

Anmerkungen zu Folie 112:

Literatur: Tanenbaum, Wetherall: Computernetzwerke, 5. Auflage, Pearson, 2012, Kap. 4

3.8.2 Dynamische Verfahren mit *Random Access*

- Unabhängige Stationen versuchen zu zufälligen Zeiten auf dasselbe Medium zu senden
  - gleichzeitiges Senden führt zu Kollision
- Unterschiedliche Voraussetzungen:
  - Trägerprüfung (sendet schon jemand?) möglich / nicht möglich
  - Sendezeitpunkt beliebig / nur zu Beginn fester Zeitscheiben
  - Kollision beobachtbar / nicht beobachtbar
- Führt zu vielen verschiedenen Verfahren
  - ohne Kollisioserkennung: z.B. Aloha, slotted Aloha, CSMA
  - mit Kollisioserkennung: CSMA/CD (☞ **3.8.5**)

Aloha

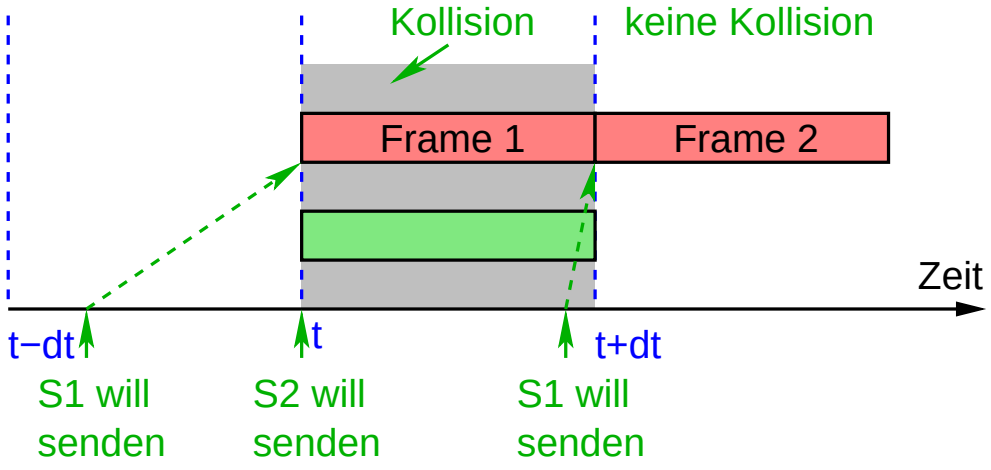
- Entwickelt in Hawaii in den 1970'er Jahren
  - Verbindung zum Hauptrechner über Inseln hinweg
- Station sendet zu beliebigem Zeitpunkt
  - keine Trägerprüfung, keine Zeitscheiben
- Empfänger quittiert den Empfang
  - keine Kollisionserkennung
- Ggf. Neuübertragung nach zufälliger Wartezeit
  - verhindert sofortige Neu-Kollision
- Bei fester Framegröße und konstanter mittlerer Framerate: Auslastung max. 18%
  - (diese Annahmen sind in der Praxis unrealistisch)

(Animierte Folie)

Slotted Aloha

- Wie Aloha, aber mit Zeitscheiben
  - benötigt Synchronisation der Stationen
- Verdopplung der max. Auslastung gegenüber Aloha

Mit Zeitscheiben:



Anmerkungen zu Folie 115:

In der Animation ist zu sehen, dass ohne Zeitscheiben eine Kollision auftritt, wenn S1 in einem der beiden Zeitintervalle $[t - dt, t]$ oder $[t, t + dt]$ senden will, da die Station in diesem Fall sofort sendet (dt ist die Dauer für das Senden eines Frames; hier sei angenommen, dass die Frames eine feste Länge haben).

Bei Verwendung von Zeitscheiben darf S1 immer erst zum Beginn der nächsten Zeitscheibe senden. In diesem Fall gibt es nur eine Kollision, wenn S1 im Zeitintervall $[t - dt, t]$ senden will. Bei einem Sendewunsch im Intervall $[t, t + dt]$ kann der Frame dagegen ohne Kollision übertragen werden.

115-1

3.8.2 Dynamische Verfahren mit *Random Access* ...

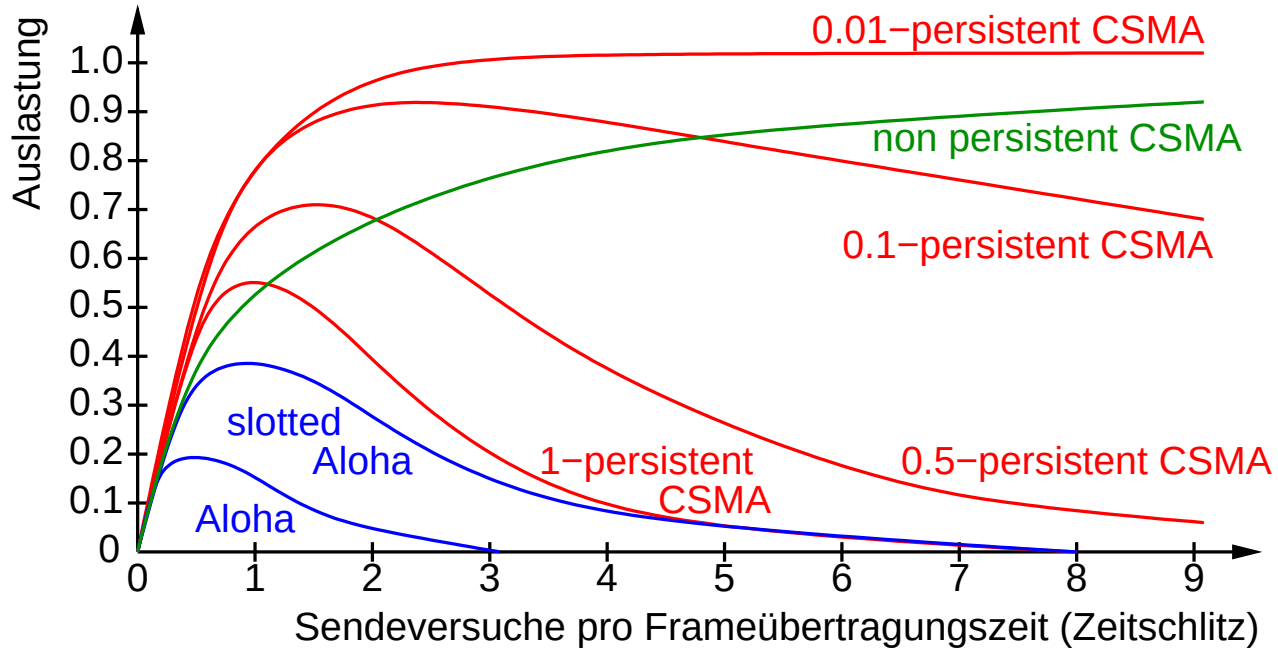
Beispiel: RFID-Tags

- Lesegerät sendet durchgehendes Signal aus
  - zur Energieversorgung der Tags
- Tags können Signal reflektieren oder absorbieren
  - Informationsübertragung an das Lesegerät
- Tags können nicht feststellen, ob ein anderes Tag „sendet“
- Vorgehensweise:
  - Lesegerät sendet Anfrage mit Zahl der Zeitscheiben; wiederholt Anfrage periodisch (für jede Zeitscheibe)
  - Tag wählt zufällige Zeitscheibe und sendet kurze Antwort
  - Lesegerät bestätigt, falls keine Kollision
  - nach Bestätigung sendet Tag seine ID

Carrier Sense Multiple Access (CSMA)

- Grundidee: höre das Medium vor dem Senden ab
  - falls frei: sende; falls belegt: sende, wenn wieder frei
- Verschiedene Designentscheidungen möglich:
  - sende immer, oder sende nur mit Wahrscheinlichkeit p (setzt Zeitscheiben voraus)
  - sende sofort, wenn Medium frei wird, oder warte zufällige Zeit
- Varianten
  - **1-persistent CSMA**: sende immer sofort mit $p = 1$
  - **nonpersistent CSMA**: falls frei, sende mit $p = 1$, falls nicht, warte zufällige Zeit
  - **p-persistent CSMA**: sende mit Wahrscheinlichkeit p, warte zufällige Zeit, falls andere Station sendet

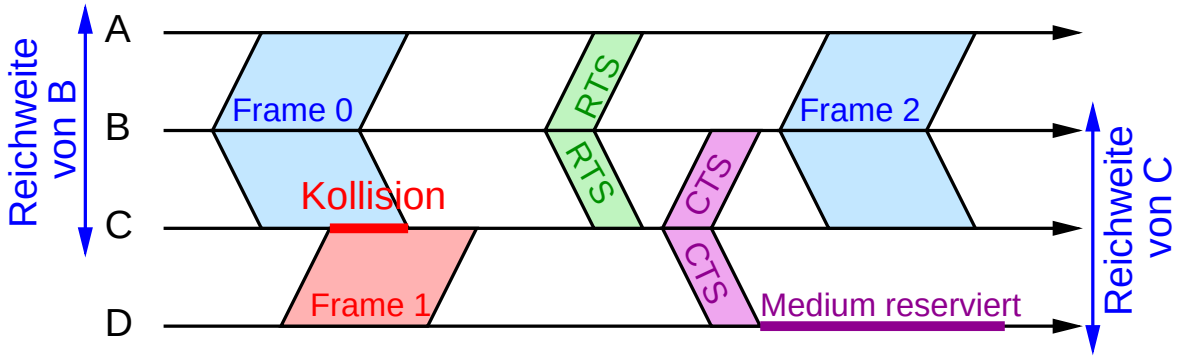
Vergleich



- Hier vorausgesetzt: keine Bursts, einheitliche Framelänge

CSMA bei drahtloser Übertragung

- Problem: Reichweite des Signals ist begrenzt
  - Station kann daher evtl. andere sendende Station nicht „hören“
  - Folge: erhöhte Zahl von Kollisionen (CSMA → Aloha)
- Lösung: Reservierung des Mediums durch den Empfänger
  - MACA-Protokoll mit RTS/CTS (*Request / Clear To Send*)



3.8.3 Kollisionsfreie dynamische Verfahren

Master/Slave

- eine Station im Netz ist Master
  - fordert andere Stationen (Slaves) zum Senden auf
- Slave darf nur nach Anforderung senden
- Verwendung z.B. bei Bluetooth

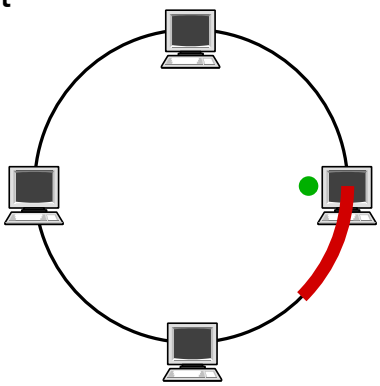
Tokenweitergabe

- Token = Erlaubnis zum Senden
- Token wird im Netz zyklisch weitergegeben
- Beispiel: Token-Ring

(Animierte Folie) ★★

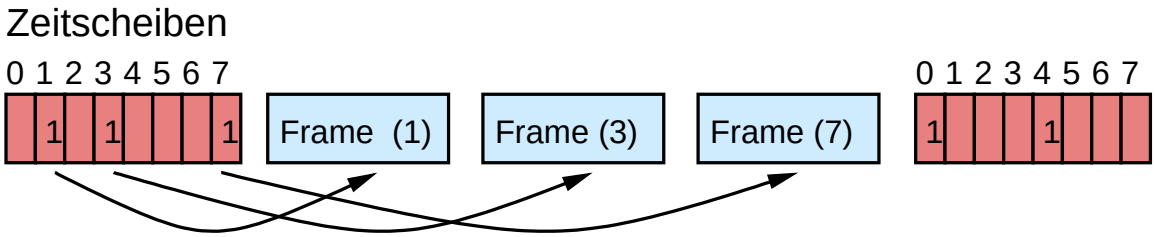
Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben



Bitmusterprotokoll

- Während der Konkurrenzphase werden alle Sendewünsche kollisionsfrei bekanntgegeben
- Jede Station erhält dazu eine eigene Zeitscheibe
  - Länge muß $\geq$ RTT/2 sein
  - Anzahl der Teilnehmer im Netz limitiert
- Im Anschluss kennen alle Stationen alle Sendewünsche
  - Abarbeitung in Reihenfolge der IDs



Binärer Countdown

- In Konkurrenzphase: Arbitrierung anhand der Sender-ID
- Gewinner kann anschließend kollisionsfrei senden
- Voraussetzungen:
  - Kanal bildet logisches ODER aller Signale
  - Bitzeit muss $\geq$ RTT/2 sein ($\Rightarrow$ geringe Übertragungsrate)
- Beispiel: CAN-Bus

Station 2	0 0 1 0	0	–	–	–
Station 4	0 1 0 0	0	–	–	–
Station 9	1 0 0 1	1	0	0	–
Station 10	1 0 1 0	1	0	1	0
	Ergebnis	1	0	1	0

Stationen 2 und 4 geben auf

Station 9 gibt auf

★★

Diskussion

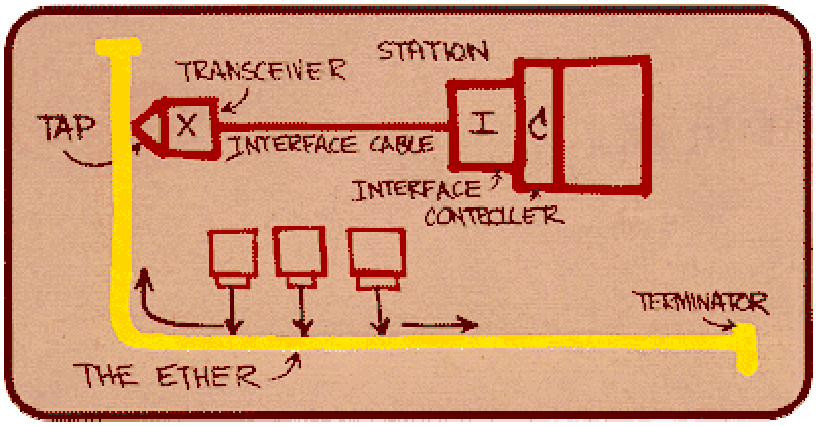
- Kollisionsfreie Verfahren gut geeignet für Echtzeit-Anwendungen
  - maximale Sendeverzögerung kann garantiert werden
- Beispiel Token-Ring: mit gegebener Token-Haltezeit THT kann jeder Knoten garantiert nach Ablauf der Zeit

  $$\text{TRT} \leq \text{Ringlatenz} + (\text{AnzahlKnoten} - 1) \cdot \text{THT}$$

  seinen Frame senden
- Mit CSMA-Verfahren sind keine Echtzeit-Garantien möglich
  - dafür kann ein Knoten bei unbelastetem Netz immer sofort senden

3.8.4 Ethernet

- Erfolgreichste LAN-Technologie der letzten Jahre
- Im Folgenden: Grundlagen, 10 Mb/s und 100 Mb/s Ethernet
- Ursprünglich Mehrfachzugriffsnetz: alle Rechner nutzen eine gemeinsame Verbindungsleitung



Bob Metcalfe
Xerox, 1976

- Heute: Punkt-zu-Punkt Verbindungen

Anmerkungen zu Folie 125:

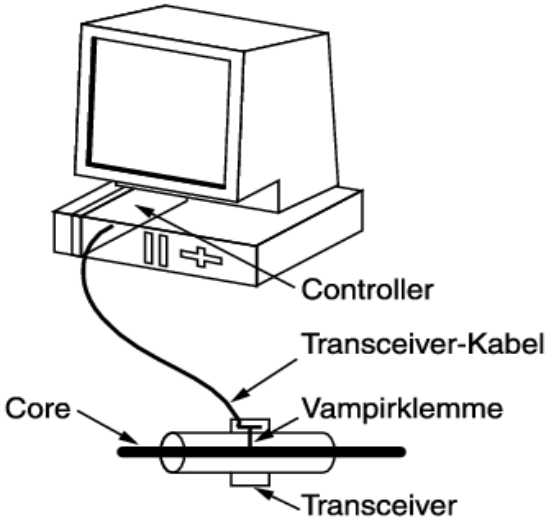
Heute gibt es viele verschiedene Ethernet-Standards, die durch *Autonegotiation*-Mechanismen häufig kompatibel zueinander sind. Die wichtigsten Standards sind:

- 10BASE-2, 10BASE-5: 10 Mb/s, Busverkabelung, Koaxialkabel, veraltet
- 10BASE-T: 10 Mb/s, Sternverkabelung, UTP, veraltet
- 100BASE-TX: 100 Mb/s, UTP-Kupferkabel, heute Standard
- 100BASE-FX: 100 Mb/s, Multimode-Glasfaserkabel
- 1000BASE-T: 1 Gb/s, UTP-Kupferkabel, heute Standard
- 1000BASE-SX, 1000BASE-LX: 1 Gb/s, Multi- bzw. Monomode-Glasfaser
- 10GBASE-T: 10 Gb/s, UTP-Kupferkabel
- 10GBASE-SR, 10GBASE-LR: 10 Gb/s, Multi- bzw. Monomode-Glasfaser, im Core-Bereich
- 40 und 100 Gb/s Standards exstieren seit Juni 2010
- 200 und 400 Gb/s Standards existieren seit 2017

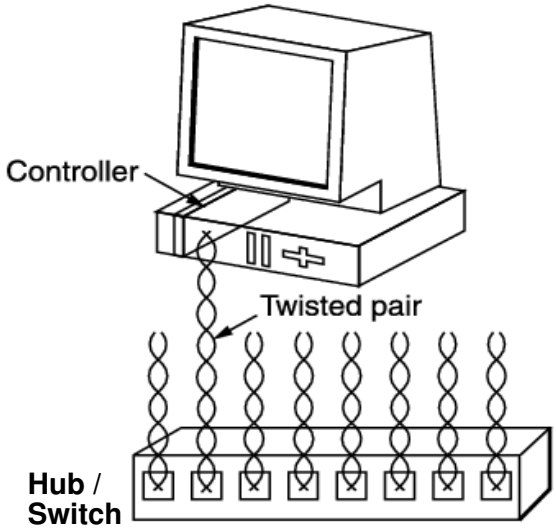
(Animierte Folie)

Verkabelung

10Base5 (max. 500m)



10BaseT / 100BaseTx (100m)

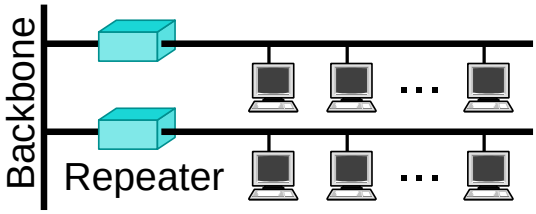


- **Hub**: „Verstärker“ und Verteiler (OSI-Schicht 1)
- **Switch**: Vermittlungsknoten (OSI-Schicht 2)

Physische Eigenschaften

10BASE-5 (10 Mb/s)

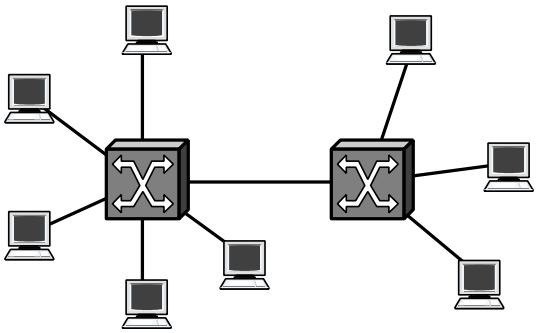
- Segmente aus Koaxialkabel, je max. 500m
- Segmente über **Repeater** (Hub mit 2 Ports) verbindbar



  - max. 4 Repeater zw. zwei Knoten erlaubt
- Manchester-Codierung

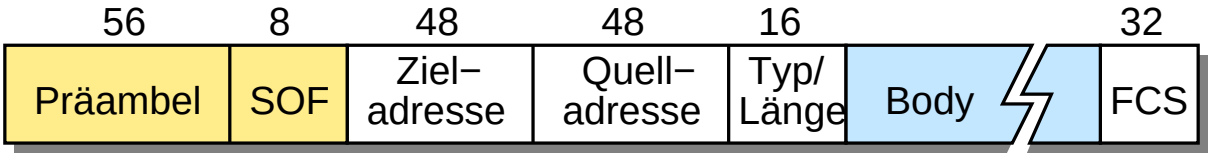
100BASE-TX (100 Mb/s)

- Twisted-Pair Kabel, je max. 100m
- Sternförmige Verkabelung mit Hubs / Switches



- 4B5B-Codierung

Frame-Format



- **Präambel/SOF**: Bitfolge 10101010 ... 10101011
  - zur Takt- und Frame-Synchronisation des Empfängers
  - letztes Byte (SOF, *Start of Frame*) markiert Frame-Anfang
- **Typ/Länge**:
  - Wert $<$ 1536 (0600_{16}): Framelänge
  - Wert $\geq$ 1536: *EtherType*, spezifiziert Protokoll im *Body*
- **FCS** (*Frame Check Sequence*): 32-Bit CRC Wert

Anmerkungen zu Folie 128:

- Im ursprünglichen Ethernet-Standard von Xerox war nur die Länge des Datenteils im Header angegeben. Bei Ethernet II (auch DIX Ethernet genannt) gibt das Typ/Länge-Feld nur den *EtherType* an, d.h. an welches Schicht-3 Protokoll der Empfänger die Nutzdaten übergeben soll. Bei der späteren Standardisierung durch die IEEE (Norm IEEE 802.3) wurde das Feld dann zunächst wieder nur als Längenfeld definiert, erst später wurde die beschriebene Doppelverwendung standardisiert.
- Wenn das Typ/Länge-Feld einen *EtherType* enthält, muß das Frameende auf OSI-Schicht 1 erkannt (Fehlen eines (Takt-)Signals bei 10 Mb/s Ethernet, „ungültiger“ 4B5B-Code bei 100 Mb/s Ethernet) und an Schicht 2 weitergemeldet werden.
- Das Schicht-3 Protokoll kann ausser über den *EtherType* auch noch über einen speziellen *Logical Link Control* (LLC) Header spezifiziert werden, der nach dem eigentlichen Ehternet-Header übertragen wird. In diesem Fall enthält das Typ/Länge-Feld dann die Frame-Länge.

Ethernet-Adressen (MAC-Adressen)

- Identifizieren die Netzwerkkarte
- 6 Byte (48 Bit) lang, weltweit eindeutig
- Schreibweise: byteweise hexadezimal, mit ':' oder '-' als Trennzeichen, z.B. 01:4A:3E:02:4C:FE
- jeder Hersteller erhält ein eindeutiges 24 Bit Präfix und vergibt eindeutige Suffixe
- Niedrigstwertiges Bit = 1: Multicast-Adresse
- Adresse ff:ff:ff:ff:ff:ff als Broadcast-Adresse
- Die Netzwerkkarte bestimmt, welche Frames sie empfängt

Anmerkungen zu Folie 129:

Bei Ethernet MAC-Adressen (nach IEEE 802.3) wird dabei das **niedrigstwertige** Byte zuerst notiert.

Ein Beispiel für eine Multicast-Adresse wäre daher: 01:00:0c:cc:cc:cc

Das „Multicast"-Bit ist damit Teil des Hersteller-Präfixes, ebenso wie ein weiteres spezielles Bit, das die Adresse als lokal bzw. weltweit eindeutig kennzeichnet.

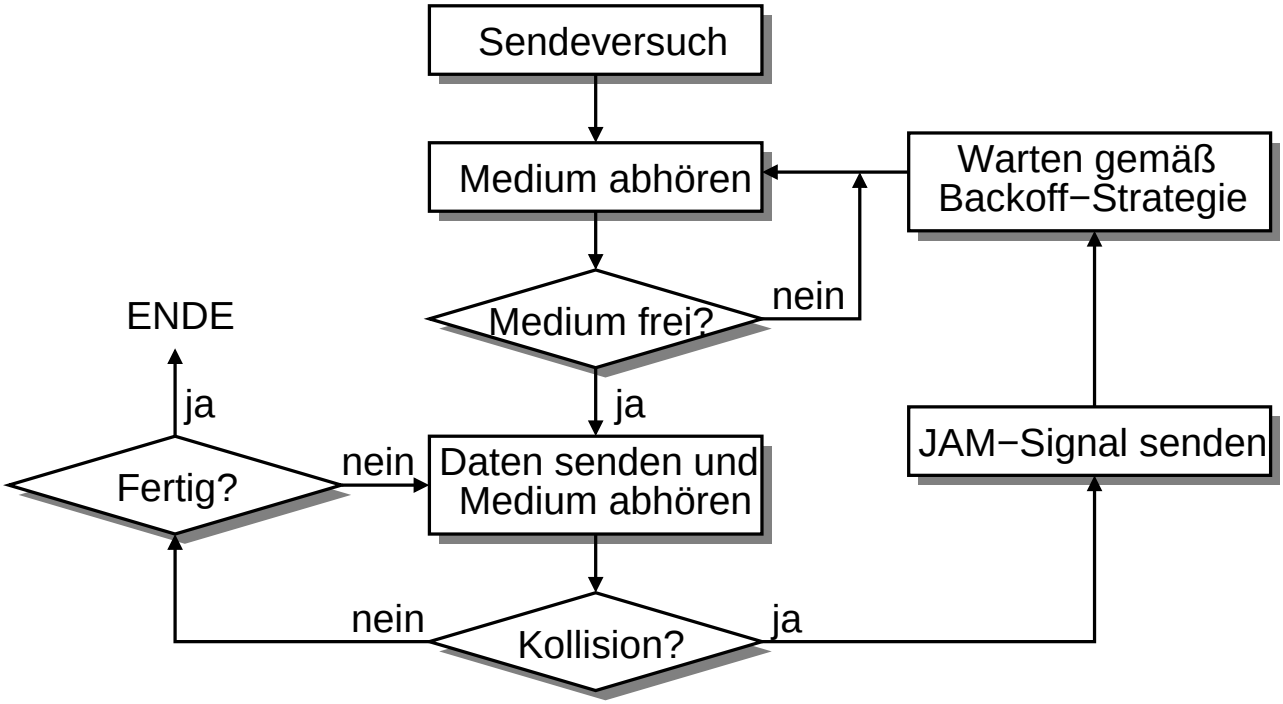
★★★

Begriffsdefinition

- **Zugangsprotokoll** zum gemeinsamen Übertragungsmedium beim Ethernet
  - *Carrier Sense Multiple Access*
    - jede Netzwerkkarte prüft zunächst, ob die Leitung frei ist, bevor sie einen Frame sendet
    - wenn die Leitung frei ist, sendet die Netzwerkkarte ihren Frame
  - *Collision Detection*
    - beim Senden erkennt der Sender Kollisionen mit Frames, die andere Netzwerkkarten evtl. gleichzeitig senden
    - bei Kollision: Abbruch des Sendens, nach einiger Zeit neuer Sendeversuch

★★★

CSMA/CD – Algorithmus



Anmerkungen zu Folie 131:

- Kollisionen werden durch Messung des Spannungspegels auf der Leitung erkannt. Falls mehrere Stationen gleichzeitig senden, überlagern (addieren) sich deren elektromagnetische Wellen auf der Leitung, wodurch sich der Spannungspegel (verglichen mit der Situation, daß nur eine Station sendet) erhöht.
- Das JAM-Signal dient dazu, die Kollision zu verlängern, damit auch alle anderen Stationen die Chance haben, die Kollision zu erkennen.

  Auch ein Hub, der eine Kollision erkennt (weil an mindestens zwei Eingängen ein Signal empfangen wird), gibt ein JAM-Signal an alle Ausgänge weiter.

3.8.5 CSMA/CD ...

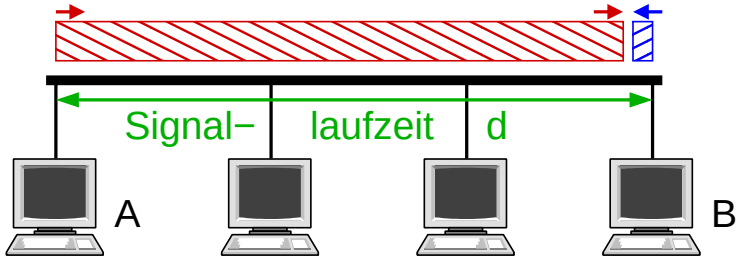
★★

Exponential Backoff-Strategie

- Aufgabe: Bestimmung der Wartezeit zwischen Kollision und erneutem Sendeversuch
- Ziel: erneute Kollision möglichst vermeiden
- Vorgehensweise:
  - c = Anzahl der bisherigen Kollisionen für aktuellen Frame
  - warte $s \cdot 51{,}2\ \mu$s (bei 10 Mb/s), wobei s wie folgt bestimmt wird:
    - falls $c \leq 10$: wähle $s \in \{ 0, 1, \ldots, 2^c\text{-}1 \}$ zufällig
    - falls $c \in [11, 16]$: wähle $s \in \{ 0, 1, \ldots, 1023 \}$ zufällig
    - falls $c = 17$: Abbruch mit Fehler
  - damit: neue Sendeversuche zeitlich entzerrt, kurze Wartezeit bei geringer Netzlast

(Animierte Folie) ★★

Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!
- $t = 2 \cdot d$: B's (Kollisions-)Frame kommt bei A an
  - wenn A zu diesem Zeitpunkt nicht mehr sendet, erkennt A keine Kollision

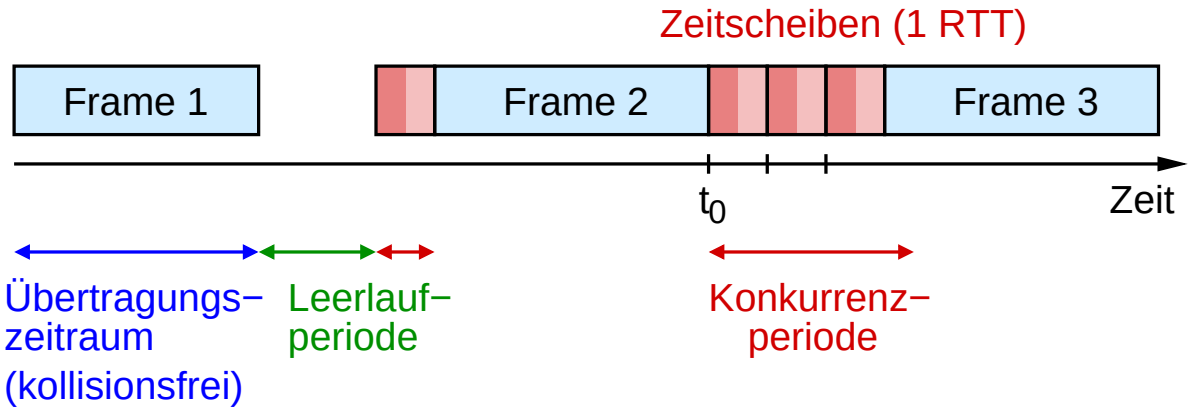
★★

Sichere Kollisionserkennung

- Um Kollisionen immer erkennen zu können, definiert Ethernet:
  - maximale RTT: 512 Bit-Zeiten
    - 51,2 μs bei 10 Mb/s, 5,12 μs bei 100 Mb/s
    - legt maximale Ausdehnung des Netzes fest, z.B. 200m bei 100BASE-TX mit Hubs
  - minimale Framelänge: 512 Bit (64 Byte), zzgl. Präambel
    - kleinere Frames werden vor dem Senden aufgefüllt
- Das stellt im Worst-Case Szenario sicher, daß Station A immer noch sendet, wenn B's Frame bei ihr ankommt
  - damit erkennt auch Station A die Kollision und kann ihren Frame wiederholen

Vorteil der Kollisionserkennung

- Kollisionen können nur innerhalb einer RTT nach Sendebeginn auftreten
  - bei Erkennung einer Kollision: Sendeabbruch
  - Rest der Frame-Übertragungszeit ist dann kollisionsfrei



Anmerkungen zu Folie 135:

- In der Graphik ist angenommen, daß die Frames deutlich länger sind als die minimale Framelänge von 64 Bytes.
- Nach dem Ende der Übertragung von Frame 1 will zunächst keine Station senden (Leerlaufperiode).
- Dann beginnt eine Station damit, Frame 2 zu senden.
  - Eine Kollision kann nur innerhalb der ersten halben RTT auftreten (d.h. beginnen, siehe Worst-Case-Szenario).
  - Bis sie erkannt wird, kann es eine RTT dauern.
  - Im Beispiel erkennt der Sender von Frame 2 nach Ablauf einer RTT keine Kollision und kann damit den Rest des Frames ohne Kollision versenden. CSMA/CD hat daher für längere Frames eine deutlich bessere Performance.
- Nach dem Ende von Frame 2 wollen mehrere Stationen senden. Der Sendebeginn kann dabei wegen der Signallaufzeit um RTT/2 variieren.
  - Im Beispiel erkennen die Sender die Kollision und stoppen das Senden.
  - Da der exponential Backoff-Algorithmus eine Wartezeit bestimmt, die ein Vielfaches der RTT ist, entstehen (näherungsweise) Zeitscheiben für die weiteren Sendeversuche.

3.9 Zusammenfassung

- Hardware: Knoten, Leitungen (Kupfer, Glasfaser, Funk)
- Codierung und Modulation
  - Umsetzen des Bitstroms in ein elektrisches Signal
  - wichtig: Taktwiederherstellung
- Framing: Erkennung des Anfangs / Endes eines Datenblocks
- Fehlererkennung (und -korrektur)
- Medienzugriffssteuerung (MAC)
  - Ethernet (CSMA-CD)
  - Token-Ring: garantierte maximale Sendeverzögerung

Nächste Lektion:

- Paketvermittlung, LAN-Switching

Rechnernetze I

SoSe 2024

4 LAN Switching

4 LAN Switching ...

Inhalt

- Weiterleitungstechniken
- Switching: Einführung
- Implementierung von Switches
- Lernende Switches
- *Spanning-Tree*-Algorithmus
- Virtuelle LANs

- Peterson, Kap. 3.1, 3.2, 3.4
- CCNA, Kap. 5.2

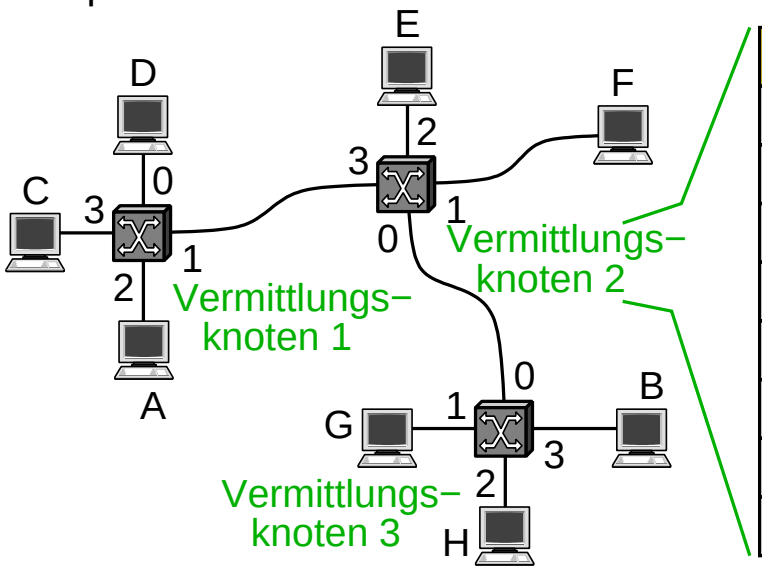
4.1 Weiterleitungstechniken

OSI: 2/3

★★

Weiterleitung von Datagrammen (verbindungslos)

- Jeder Vermittlungsknoten besitzt eine Weiterleitungstabelle
  - bildet Zieladresse auf Ausgangsport ab
- Beispiel:



Weiterleitungstabelle

Ziel	Port
A	3
B	0
C	3
D	3
E	2
F	1
G	0
H	0

★★

Weiterleitung von Datagrammen: Eigenschaften

- Knoten können jederzeit ein Paket an andere Knoten senden; Pakete können sofort weitergeleitet werden
  - kein Verbindungsaufbau
- Ein Knoten kann nicht feststellen, ob das Netz das Paket zustellen kann
- Pakete werden unabhängig voneinander weitergeleitet
- Ausfall einer Verbindung bzw. eines Vermittlungsknotens kann prinzipiell toleriert werden
  - Anpassung der Weiterleitungstabellen
- Eingesetzt z.B. im Internet (IP)
- (vgl. Kap. 1.4: Paketvermittlung)

★★

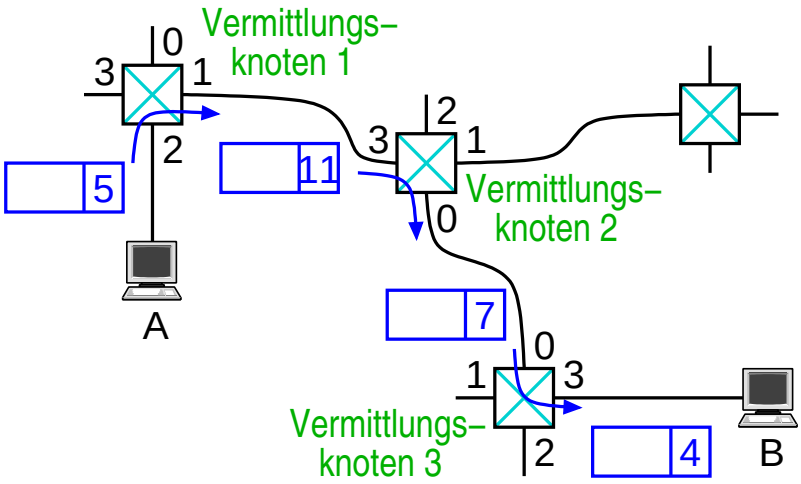
Virtuelle Leitungsvermittlung (verbindungsorientiert)

- Kommunikation in zwei Phasen:
  - Aufbau einer virtuellen Verbindung (***Virtual Circuit***, **VC**) vom Quell- zum Zielrechner
    - statisch (*Permanent VC*)
    - dynamisch (*Switched VC*)
  - Versenden der Pakete über VC:
    - Pakete enthalten Bezeichner des VC
    - alle Pakete nehmen denselben Weg
- VC-Bezeichner (VCI, *Virtual Circuit Identifier*) nur auf den einzelnen Leitungen eindeutig
  - Weiterleitungstabelle im Vermittlungsknoten bildet Eingangs-Port und -VCI auf Ausgangs-Port und -VCI ab

(Animierte Folie) ★★

Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

- Beispiel:
  A sendet an B



	Eingangsport	Eingangs–VCI	Ausgangsport	Ausgangs–VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

- Eingesetzt z.B. in Frame-Relay und MPLS (☞ **RN-II**)

Anmerkungen zu Folie 142:

Jeder Vermittlungsknoten hat seine eigene Weiterleitungstabelle, in der für alle möglichen Paare von Eingangsport und Eingangs-VCI der Ausgangsport und Ausgangs-VCI vermerkt ist. Auf der Folie ist der Übersicht halber nur jeweils eine relevante Zeile aus der Weiterleitungstabelle jedes Vermittlungsknotens gezeigt.

Eine globale Weiterleitungstabelle gibt es nicht!

★★

Vergleich der Weiterleitungstechniken

	Datagramm-Verm.	Virtuelle Leitungsv.
Verbindungsaufbau	nicht nötig	erforderlich
Adressierung	Pakete enthalten volle Sender- und Empfänger-Adresse	Pakete enthalten nur kurze VC-Bezeichner
Wegewahl	erfolgt unabhängig für jedes Paket	Weg wird bei Aufbau des VC festgelegt
Bei Ausfall eines Vermittlungsknotens	keine größeren Auswirkungen	alle VCs mit diesem Vermittlungsknoten sind unterbrochen
Dienstgütegarantien (QoS) und Überlastkontrolle	schwierig	Einfach, wenn vorab Resourcen für VC reserviert werden

★

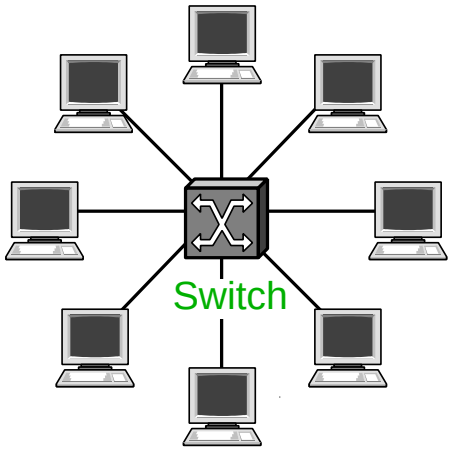
Begriffe

- **Switching / Forwarding (Weiterleitung):**
  - Weiterleiten v. Frames (Paketen) zum richtigen Ausgangsport
- **(LAN-)Switch / Bridge (Brücke):**
  - Vermittler im LAN (auf Ebene der Sicherungsschicht)
  - Bridge: Switch mit nur zwei Ports
- **Routing:**
  - (dynamischer) Aufbau von Tabellen zum Forwarding
  - Ziel: Finden von (guten/optimalen) Wegen zu anderen Netzen
- **Router:**
  - Knoten, der mit den Protokollen der Vermittlungsschicht Pakete weiterleitet
  - vereinigt Funktionalität von Routing und Forwarding

4.2 Switching: Einführung

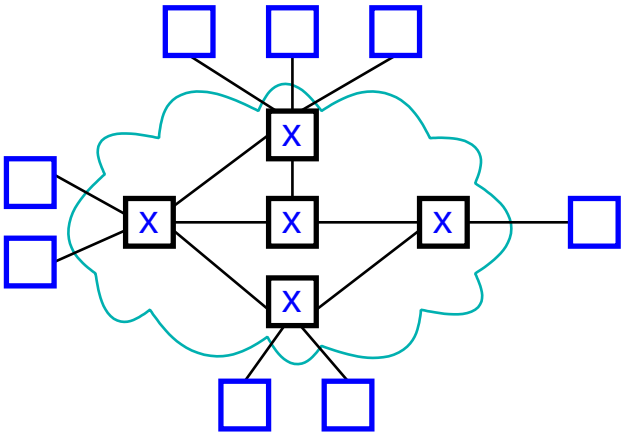
OSI: 2

- Motivation: Ersetzen von Mehrfachzugriffsverbindungen im LAN durch Punkt-zu-Punkt-Verbindungen
- Vermittler (Switch):
  - mehrere Ein-/Ausgänge
  - leitet Frames aufgrund der Zieladresse im Header weiter
- Führt zu Sterntopologie:



4.2 Switching: Einführung ...

- Vorteil gegenüber Bustopologie:
  - Kommunikation zwischen zwei Knoten wirkt sich nicht (notwendigerweise) auf andere Knoten aus
- Beschränkungen können durch Zusammenschalten mehrerer Switches überwunden werden:
  - begrenzte Anzahl von Ein-/Ausgängen pro Switch
  - Leitungslänge, geographische Ausdehnung



4.2 Switching: Einführung ...

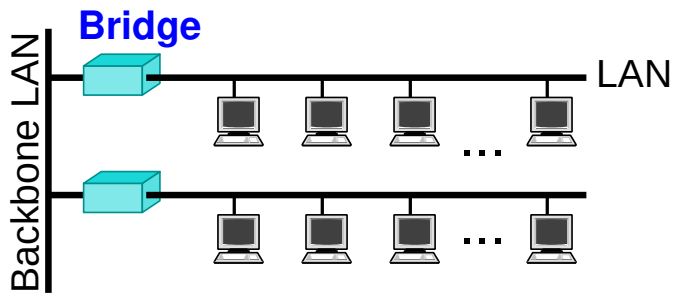
- Weitere Beschränkungen von Switches
  - Heterogenität
    - die verbundenen Netze müssen u.a. dasselbe Adressierungsschema haben
    - z.B. Ethernet und Token-Ring ist möglich, Ethernet und ATM nicht
  - Skalierbarkeit:
    - Broadcasts
    - *Spanning-Tree*-Algorithmus (☞ **4.5**)
  - Transparenz:
    - Latenz
    - Verlust von Frames bei Überlast

4.2 Switching: Einführung ...

★

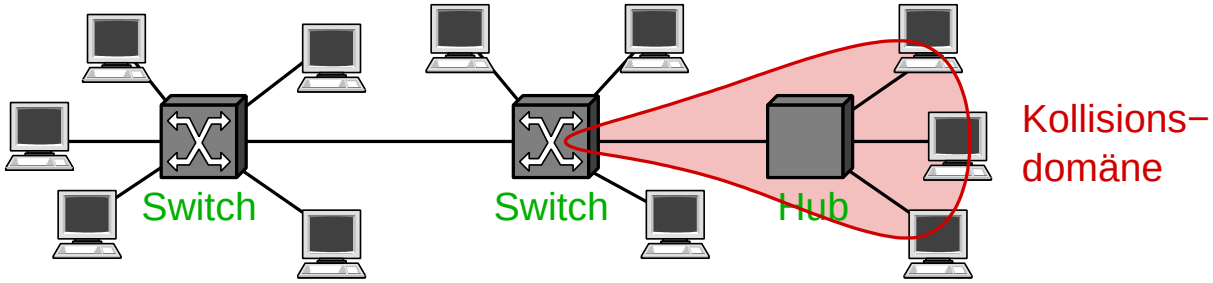
- **LAN Switch**: Vermittlungsknoten auf der Sicherungsschicht
  - kann Zieladresse im Sicherungsschicht-Header analysieren
  - gibt Frames nur an die Ports weiter, wo es notwendig ist

- **Bridge**: Switch mit 2 Ports
  - Einsatz bei 10Base5 zur Überwindung von physikalischen Beschränkungen (Leitungslänge, Anzahl Repeater, ...)



(Animierte Folie)

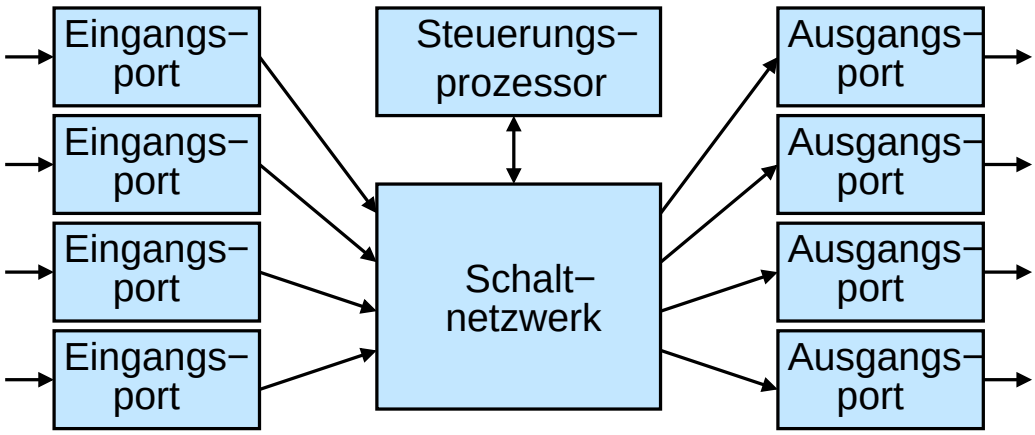
Ein (Ethernet-)LAN mit Switches



- Verwendung von Switches ist bei Ethernet ab 100 Mb/s üblich
  - ermöglicht Vollduplex-Betrieb
    - in diesem Fall treten keine Kollisionen mehr auf
    - aber: Switch muß ggf. Frames zwischenspeichern
- Möglich auch: gemischter Betrieb mit Switches und Hubs
  - Auswirkung von Kollisionen sind auf den Bereich des Hubs eingeschränkt (**Kollisionsdomäne**)

4.3 Implementierung von Switches

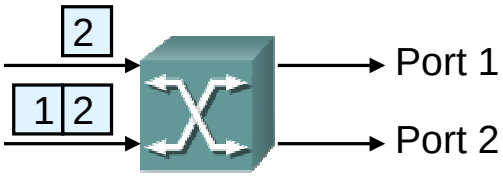
- Typischer Aufbau:



- Eingangsports analysieren Header, programmieren Schaltnetzwerk
- Pufferung in den (Ausgangs-)Ports und/oder im Schaltnetzwerk
- Steuerungsprozessor: u.a. Aufbau Weiterleitungstabelle, STP

Anmerkungen zu Folie 150:

- Eine Pufferung von Frames wird immer dann nötig, wenn mehrere Frames zur selben Zeit an denselben Ausgangsport gesendet werden müssen.
- Eine Pufferung in den Eingangsports kann dabei zum sog. *Head-of-Line-Blocking* führen:



  - In beiden Puffern steht vorne ein Frame an Port 2.
  - Wenn der obere Frame gerade an Port 2 versendet wird, wird bei einem FIFO-Puffer der Frame für Port 1 blockiert, obwohl er eigentlich gleichzeitig gesendet werden könnte.
- Daher verwenden Switches i.d.R. eine Pufferung beim Ausgangsport und/oder im Schaltnetzwerk

Literatur: Peterson, Davie, Kap. 3.4

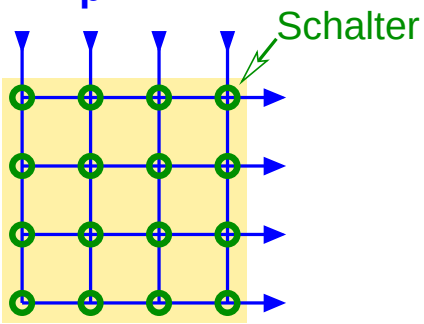
150-1

4.3 Implementierung von Switches ...

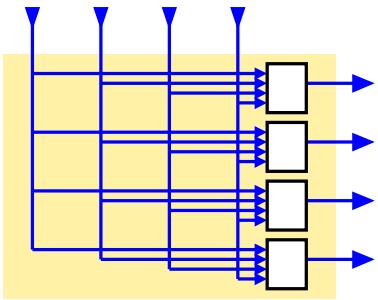
Realisierung des Schaltnetzwerks

- Gemeinsamer Bus / gemeinsamer Speicher
  - Bus- bzw. Speicherbandbreite begrenzt den Gesamtdurchsatz
- Crossbar
  - jeder Eingangsport kann an jeden Ausgangsport durchgeschaltet werden

Prinzip:



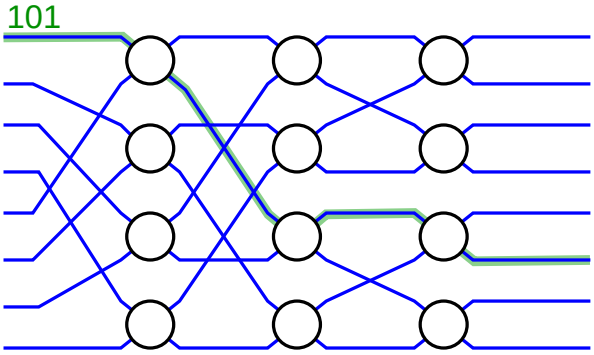
Mit Ausgangspufferung:



  - Problem: Speicherbandbreite der Ausgangspuffer

Realisierung des Schaltnetzwerks ...

- Eigenvermittelnde Schaltnetzwerke
  - Verwendung von 2x2-Schaltelementen, z.B. Banyan-Netzwerk:



  - Frames erhalten internen Header, der den Weg durch das Schaltnetz beschreibt
    - im Beipiel: 0 ≙ oberer Ausgang, 1 ≙ unterer Ausgang
  - Vorsortierung der Frames vermeidet interne Kollisionen

Queueing Strategien

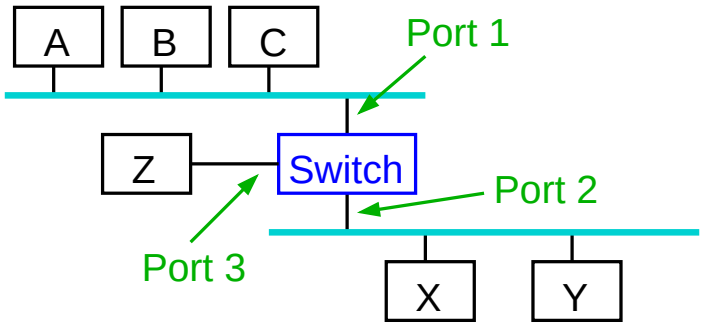
- FIFO-*Queueing* (mit *Tail Drop*)
  - Frames werden in FIFO-Reihenfolge übertragen
  - bei vollem Puffer: neu ankommender Frame wird verworfen
  - ggf. mehrere Warteschlangen mit unterschiedlichen Prioritäten
    - beim Ethernet im VLAN-Header (802.1Q) angegeben
- *Fair Queueing*
  - *Round-Robin*-Abarbeitung der Warteschlangen
  - wegen unterschiedlicher Framegrößen:
    - simuliere bitweises *Round-Robin*
  - ggf. auch mit Gewichtung der Warteschlangen
    - jede Warteschlange erhält einen bestimmten Bandbreiten-anteil

4.4 Lernende Switches

★★★

Automatisches Erstellen der Weiterleitungstabelle

- Beispiel:



Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	3

- Switch untersucht die Quelladresse jedes eingehenden Frames
  - falls nötig, Erzeugung bzw. Aktualisierung eines Tabelleneintrags
- Eintrag für eine Adresse wird gelöscht, wenn längere Zeit kein Frame mit dieser Quelladresse ankommt

4.4 Lernende Switches ...

★★★

Verhalten beim Eintreffen eines Frames

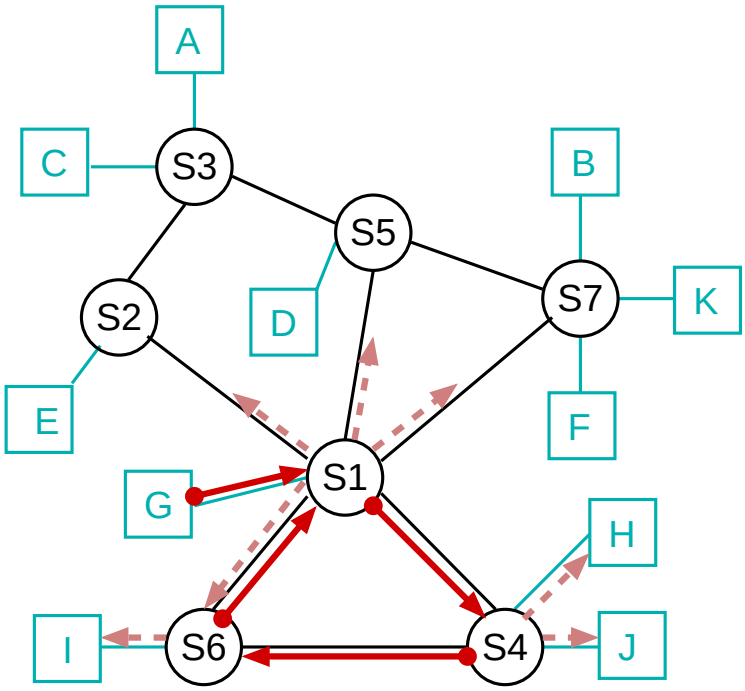
- Quell- und Ziel-Port identisch:
  - Frame verwerfen
- Quell- und Ziel-Port verschieden:
  - Frame an Ziel-Port weiterleiten
- Ziel-Port unbekannt:
  - weiterleiten an alle Ports (außer den Empfangsport)
- Broadcast-Frames werden immer an alle Ports geleitet
- Weiterleitungstabelle kann begrenzte Größe haben
  - Vollständigkeit ist nicht notwendig
  - Tabelle dient nur als Cache für aktive Knoten

4.5 *Spanning-Tree*-Algorithmus

(Animierte Folie) ★★★

Problem bei lernenden Switches: Zyklen

- Z.B. S1 – S4 – S6
- Frame von G mit unbekanntem Ziel läuft ewig im Kreis (Broadcast-Sturm)



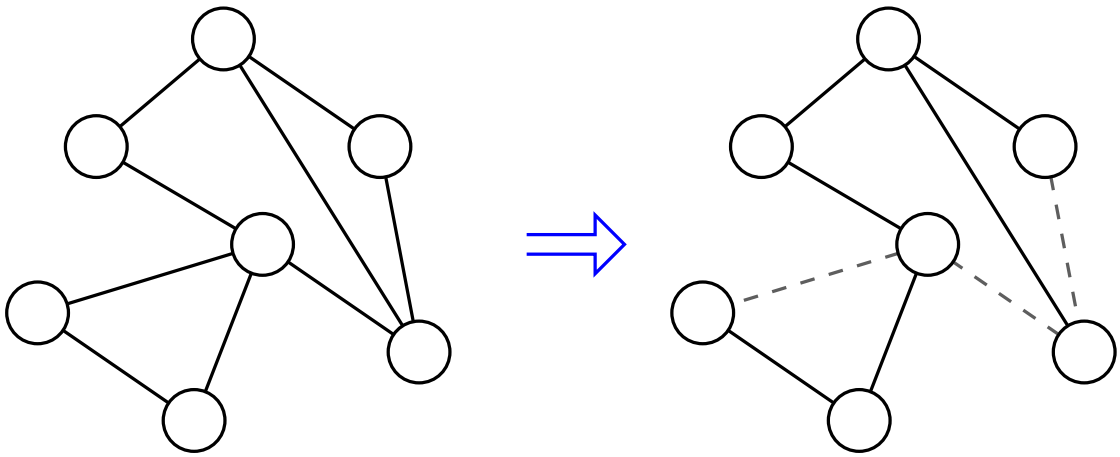
Abhilfe: *Spanning-Tree*-Algorithmus

- Reduziert das Netzwerk auf einen zyklenfreien Graphen (Baum)
- Einige Ports der Switches werden deaktiviert

4.5 *Spanning-Tree*-Algorithmus ...

Aufspannender Baum

- Zyklischer Graph und aufspannender Baum:



- der aufspannende Baum ist nicht eindeutig

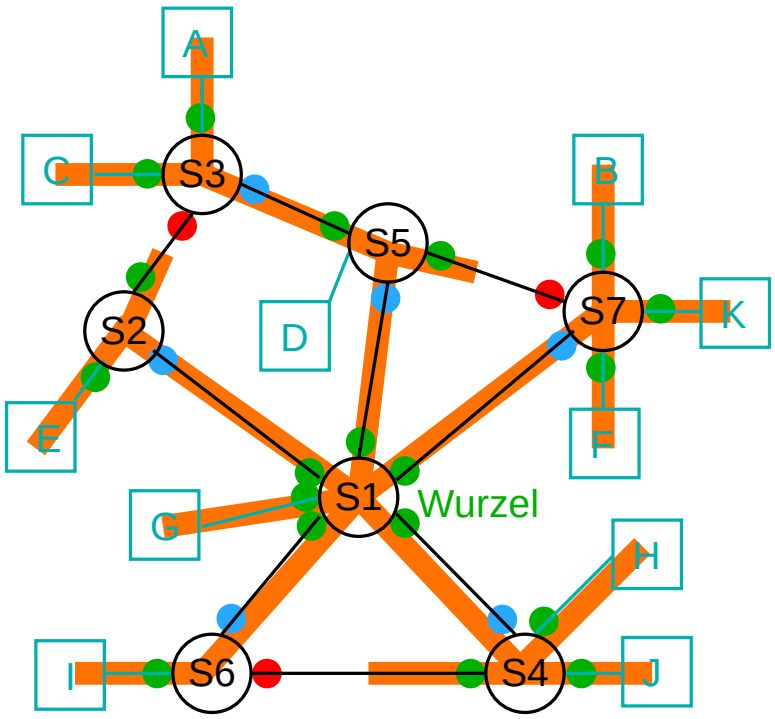
★★

Idee des Algorithmus

- Jeder Switch hat eine eindeutige Kennung (z.B. S1, S2)
- Wähle den Switch mit der kleinsten Kennung als Wurzel
- Jeder Switch bestimmt seinen ***Root Port***
  - der Port mit der geringsten Entfernung zur Wurzel
  - bei Gleichheit entscheidet Port-Priorität bzw. Port-Nummer
- Wähle für jedes LAN-Segment den Port des Switches mit der geringsten Entfernung zur Wurzel als ***Designated Port***
  - bei Gleichheit entscheidet die Switch-Kennung
- Alle anderen Ports sind ***Non-Designated Ports***
  - diese Ports sind blockiert (*blocking*) und leiten keine Frames weiter

(Animierte Folie) ★

Aufspannender Baum des Beispielnetzes



- Root Port
- Non-Designated Port
- Designated Port

Anmerkungen zu Folie 159:

- Die Kennung eines Switches besteht in der Praxis (bei Ethernet-Switches) aus einer (konfigurierbaren) 2-Byte Priorität und der MAC-Adresse des Switches (d.h., der kleinsten MAC-Adresse aller seiner Ports).

  Um VLANs zu unterstützen ist die Priorität nochmals in ein 4-Bit Prioritätsfeld und ein 12-Bit „Extended System ID“-Feld unterteilt, das die VLAN-ID enthält.

- Die Entfernung zur Wurzel wird über Pfadkosten bestimmt, die abhängig von der Bandbreite der Verbindungen sind.

- Es ist wichtig, daß ein Switch auch über den *Root Port* Frames weiterleiten darf, anderenfalls würde im Beispiel S3 keine Frames von A bzw. C mehr weiterleiten können.

- Bei Verbindungen wie z.B. zwischen S6 und S4 muss ein Port *Designated Port* bleiben, da statt einer direkten Verbindung auch ein Hub eingesetzt sein könnte, an dem weitere Hosts angeschlossen sind, die nach wie vor erreichbar bleiben müssen.

159-1

4.5 *Spanning-Tree*-Algorithmus ...

Spanning-Tree-Protokoll (STP)

- Die Knoten tauschen Konfigurations-Nachrichten aus:
  - Kennung des sendenden Switches
  - vermutete Wurzel-Kennung
  - eigene Entfernung zu dieser Wurzel
- Jeder Switch behält die beste Nachricht.
  Besser heißt dabei:
  - Wurzel-Kennung kleiner, oder
  - Wurzel-Kennung gleich, Entfernung kleiner, oder
  - Wurzel-Kennung und Entfernung gleich, Sender-Kennung kleiner
- Jeder Switch startet als Wurzel, bis dies widerlegt ist

Spanning-Tree-Protokoll (STP) ...

- Wenn ein Switch erfährt, daß er nicht die Wurzel ist, stellt er das Generieren von Nachrichten ein, leitet Nachrichten aber nach wie vor weiter
  - am Ende erzeugt nur noch die Wurzel Nachrichten
- Die Wurzel generiert weiter periodisch Konfigurations-Nachrichten
  - werden im Baum nur noch nach unten weitergegeben
  - automatische Rekonfiguration bei Ausfall der Wurzel
- Topologie-Änderungen werden von Switches zunächst an die Wurzel gesendet, diese gibt sie an alle weiter

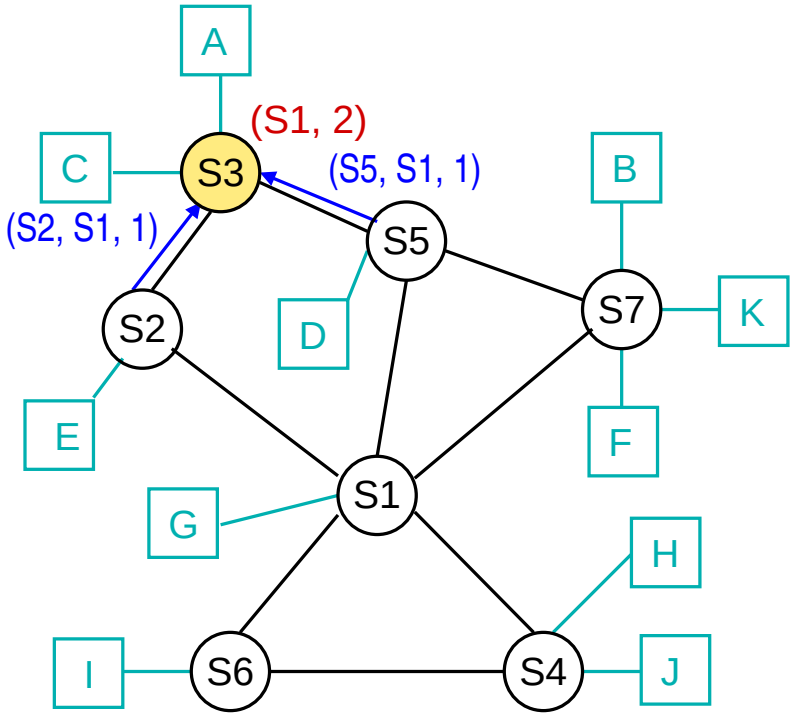
Anmerkungen zu Folie 161:

- Die Konfigurations-Nachrichten von STP werden i.a. als BPDUs (*Bridge Protocol Data Units*) bezeichnet.
- Wenn ein Switch nach Ablauf einer bestimmten Wartezeit keine Konfigurations-Nachricht erhält (sei es, weil die Wurzel ausgefallen ist oder auch ein anderer Switch), gibt sich der Switch wieder selbst als Wurzel aus und das *Spanning-Tree*-Protokoll startet von vorne.

(Animierte Folie)

Beispiel

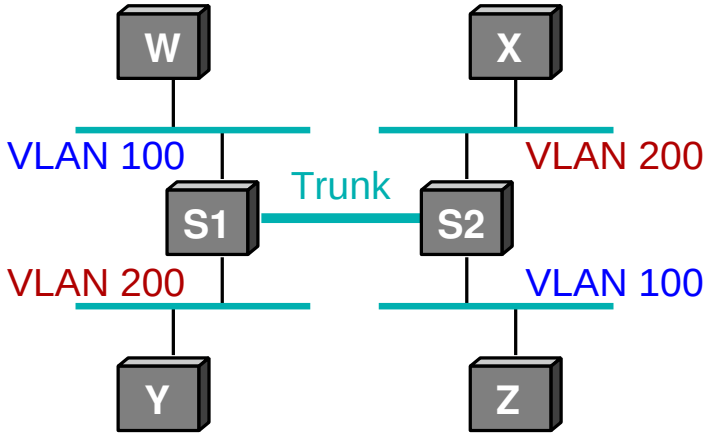
1. S2 → S3
2. S3: S2 ist Wurzel
3. S3 → S5
4. S2: S1 ist Wurzel
   S2 → S3
5. S5: S1 ist Wurzel
   S5 → S3
6. S3: S1 ist Wurzel
   S2, S5 näher an S1



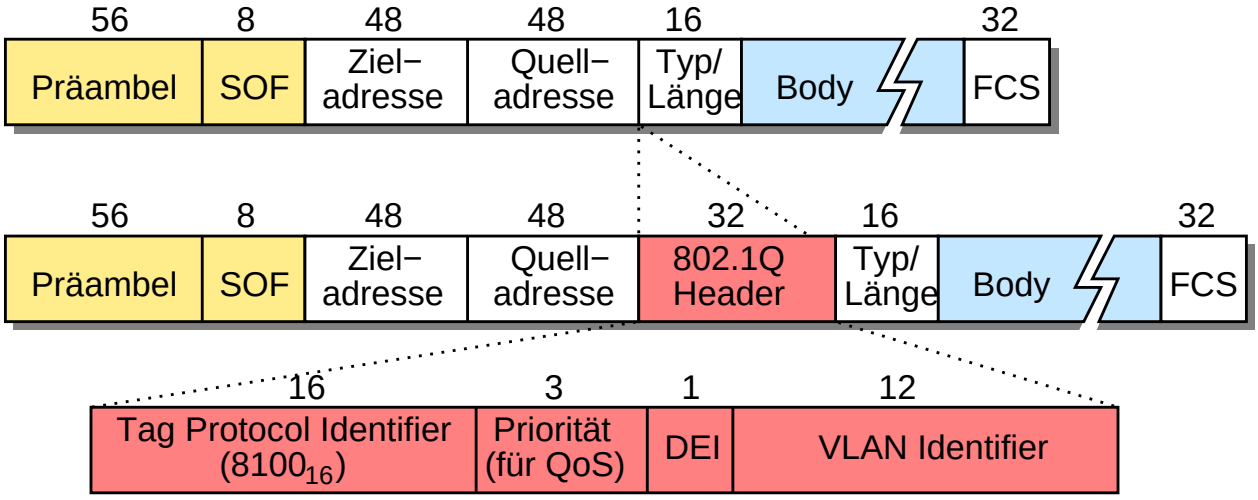
4.6 Virtuelle LANs (VLANs)

★★

- Ziele:
  - bessere Skalierung
  - höhere Sicherheit
- Jedes LAN erhält einen Bezeichner (VLAN-ID)
- Auf Trunk-Leitung: Switch fügt Header mit VLAN-ID ein bzw. entfernt ihn wieder
  - bei Ethernet: VLAN-ID wird **in** den Frame-Header eingefügt
- Frames werden nur an das LAN mit der korrekten VLAN-ID weitergeleitet
- LANs mit verschiedenen VLAN-IDs sind logisch getrennt
  - Kommunikation nur über Router möglich



Ethernet-Frame mit VLAN-Tag (IEEE 802.1Q)

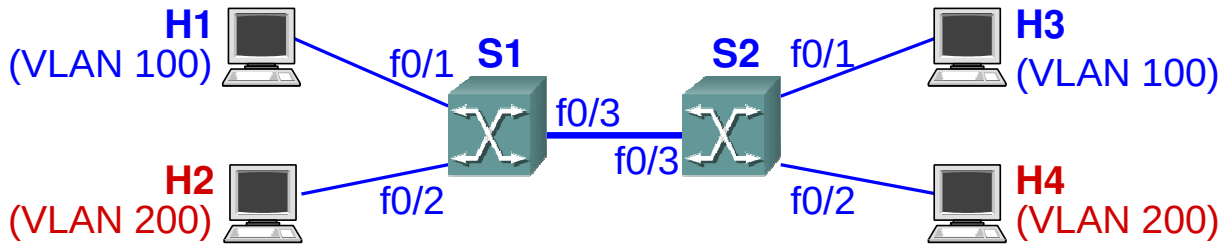


- *Tag Protocol Identifier* identifiziert „getagten“ Frame
- Prioritätsfeld erlaubt bevorzugte Weiterleitung im Switch
  - z.B. für Internet-Telefonie

Anmerkungen zu Folie 164:

- Der Wert 1 im Feld DEI (*Drop Eligible Indicator*) zeigt an, daß der Frame im Überlastfall bevorzugt verworfen werden kann.
- Vor 2011 hieß dieses Feld CFI (*Canonical Format Identifier*) und wurde für die Interoperabilität von Ethernet und Token Ring verwendet.

Beispiel-Konfiguration




```
interface f0/1
   switchport mode access
   switchport access vlan 100
interface f0/2
   switchport mode access
   switchport access vlan 200
interface f0/3
   switchport mode trunk native vlan 100
```

Sende / akzeptiere nur Frames ohne Tag.

Füge bei eingehendem Frame Tag 100 an; sende Frame nur, falls er Tag 100 hat.

Frames ohne Tag gehen ins VLAN 100.

4.7 Zusammenfassung

- Weiterleitungstechniken
  - Datagrammvermittlung, virtuelle Leitungsvermittlung
- LAN-Switches
  - lernen Weiterleitungstabellen selbst
  - zur Vermeidung von Zyklen: *Spanning Tree Protokoll*
  - erlauben die Realisierung von virtuellen LANs

Nächste Lektion:

- *Internetworking*
- Das Internet-Protokoll (IP)

Rechnernetze I

SoSe 2024

5 Internetworking

5 Internetworking ... OSI: 3

Inhalt

- IP
  - Grundlagen, Adressierung und Weiterleitung, Aufbau eines IP-Pakets, Fragmentierung / Reassembly
- ICMP
- Adreßübersetzung: APR, NDP
- Automatische IP-Konfiguration: DHCP
- NAT
- Tunneling
- Übergang von IPv4 auf IPv6

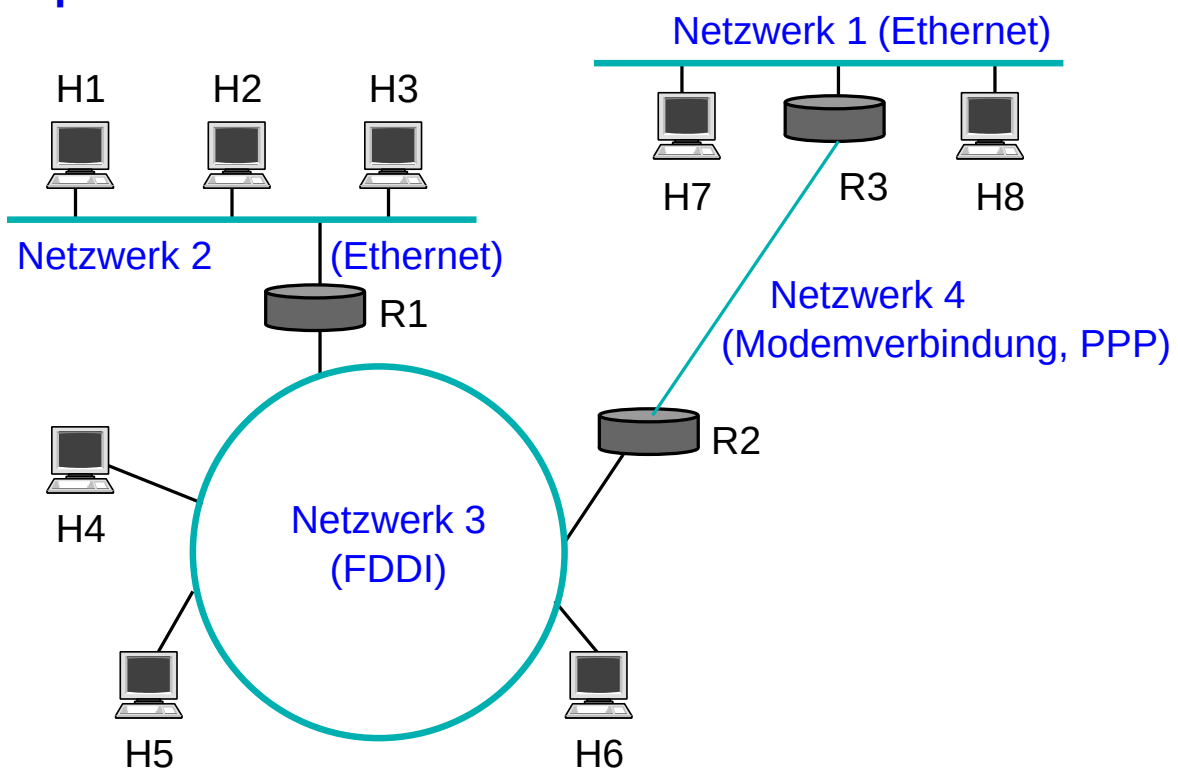
- Peterson, Kap. 4.1
- CCNA, Kap. 6, 7, 8, 5.3

5 Internetworking ...

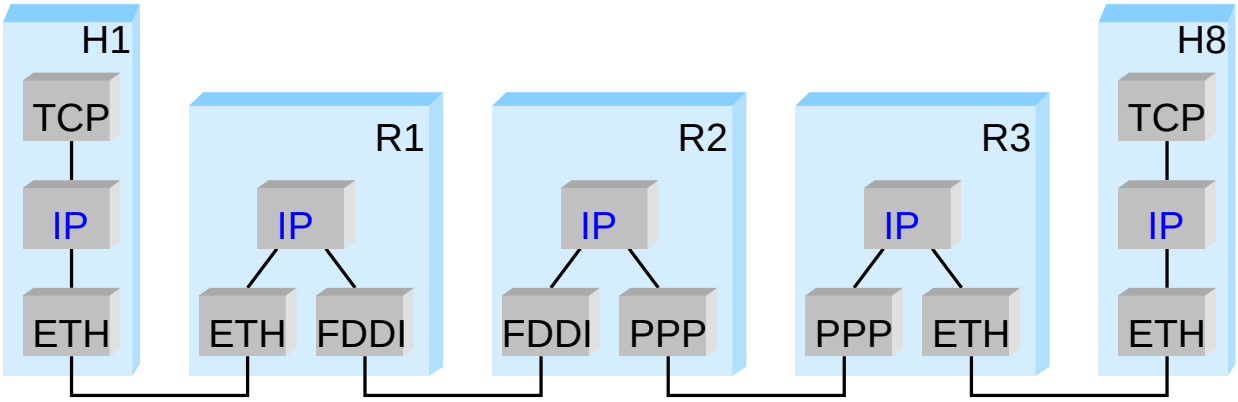
- Was ist ein Internetwork?
  - Zusammenschluß von einzelnen (physischen) Netzen über Router zu einem logischen Netz
    - Netz von Netzen
- Was macht ein Internetwork aus?
  - **Heterogenität**
    - Verbindung unterschiedlichster Netzwerktypen (auch zukünftiger!)
  - **Skalierung**
    - Integration von sehr vielen Rechnern und Netzen
    - Internet: $\geq$ 1 Milliarde Rechner
- Internetwork $\neq$ Internet

5 Internetworking ...

Beispiel für ein Internetwork



IP (*Internet Protocol*) als Internetwork-Protokoll



- Auf jedem Rechner und jedem Router läuft IP
  - IP kann auf unterschiedlichsten Netztechnologien aufsetzen

Anmerkungen zu Folie 171:

Beachten Sie, daß bei der Weiterleitung eines IP-Pakets (im Beispiel von H1 nach H8) **zwei** verschiedene Adressen notwendig sind. Wenn das Paket z.B. von H1 an R1 weitergegeben wird, ist das IP-Paket selbst an H8 adressiert, der Ethernet-Frame, in dem sich das Paket befindet, jedoch an R1.

Erst wenn R3 das Paket an H8 weitergibt, ist auch der Ethernet-Frame, in dem sich das Paket befindet, an H8 adressiert.

★★★

IP Dienstmodell (was bietet IP?)

- Adressierungsschema
- Datagramm-Zustellung

- Um Heterogenität und Skalierbarkeit zu unterstützen: kleinster gemeinsamer Nenner
  - IP bietet nur das, was mit jeder Netzwerktechnologie realisiert werden kann
    - „*run over everything*“
  - „*Best Effort*“-Modell:
    - IP „bemüht sich“, gibt aber keinerlei Garantien
    - Verlust, Duplikate, Vertauschung von Paketen möglich
    - höhere Schichten bieten bessere Dienste

★

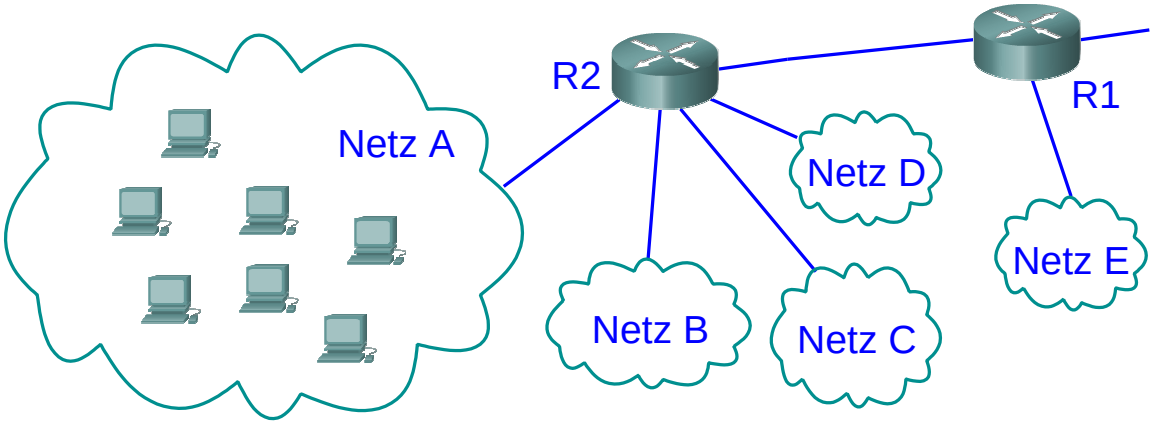
IP-Versionen: IPv4 und IPv6

- IPv4 ist seit 1980 standardisiert
- Motivation für IPv6: Wachstum des Internets
  - größerer Adreßraum für IP-Adressen notwendig
- Arbeit an IPv6 seit ca. 1991
  - längere IP-Adressen ⇒ neuer IP-Header ⇒ Anpassung aller IP-Software
  - daher: auch andere Probleme von IPv4 adressiert, u.a.
    - Unterstützung von Dienstgüte-Garantien
    - Sicherheit
    - automatische Konfiguration
    - erweitertes Routing (z.B. mobile Hosts)

(Animierte Folie)

Ziel: Effiziente Weiterleitung von IP-Paketen

- Typische Situation:



- Router müssen Pakete nur in das richtige Netz weiterleiten
  - Weiterleitungstabellen sollten nur Information über Netze enthalten, nicht über einzelne Hosts
  - Router muss aus IP-Adresse zugehöriges Netz bestimmen

Anmerkungen zu Folie 174:

Router R2 sollte in seiner Weiterleitungstabelle nur *einen* Eintrag für das gesamte Netz A haben müssen, statt für jeden einzelnen Host in diesem Netz. Analog hat dann auch R1 nur einen Eintrag für das Netz A.

In der Praxis kann das Netz A (z.B. ein großes Firmennetz) intern in meherere kleinere Netze (z.B. für die einzelnen Abteilungen) unterteilt sein, die über einen Router (R3 in Animationsschitt 2) verbunden sind. Man spricht dabei von *Subnetting*. Idealerweise sollten die Router ausserhalb von Netz A davon nichts wissen müssen, d.h., R2 und R1 haben nach wie vor nur einen Eintrag für das gesamte Netz A, während R3 natürlich für jedes Subnetz einen Weiterleitungstabellen-Eintrag besitzen muß.

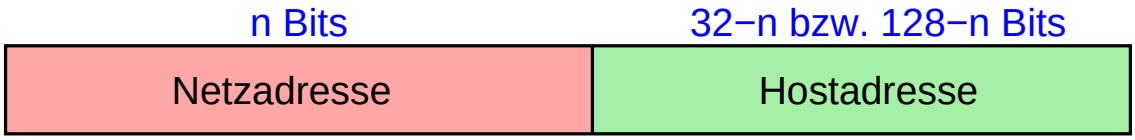
Wenn man noch etwas weiter denkt, könnten für R1 alle Netze, die über R2 erreichbar sind (also die Netze A, B, C, D) zu einem einzigen, größeren Netz ABCD zusammengefasst werden, so daß R1 (und alle anderen Router, die diese Netze erreichen können) nur noch einen einzigen Weiterleitungstabellen-Eintrag für diese vier Netze benötigen (Animationsschritt 3). An dieser Stelle spricht man von *Supernetting*.

5.2.1 Adressierung in IP

★★★

Aufgaben bei der Adressierung

- Identifikation von Hosts
  - durch numerische Adresse (IPv4: 32 Bit, IPv6: 128 Bit)
  - hierarchischer Aufbau:



- Identifikation von Netzen
  - durch Netzadresse und Präfixlänge n
  - in IPv4 ursprünglich:
    - n geht aus der Adresse eindeutig hervor (Adressklassen)
  - heute in IPv4 und IPv6: explizite Angabe von n (klassenlose Adressierung, CIDR)

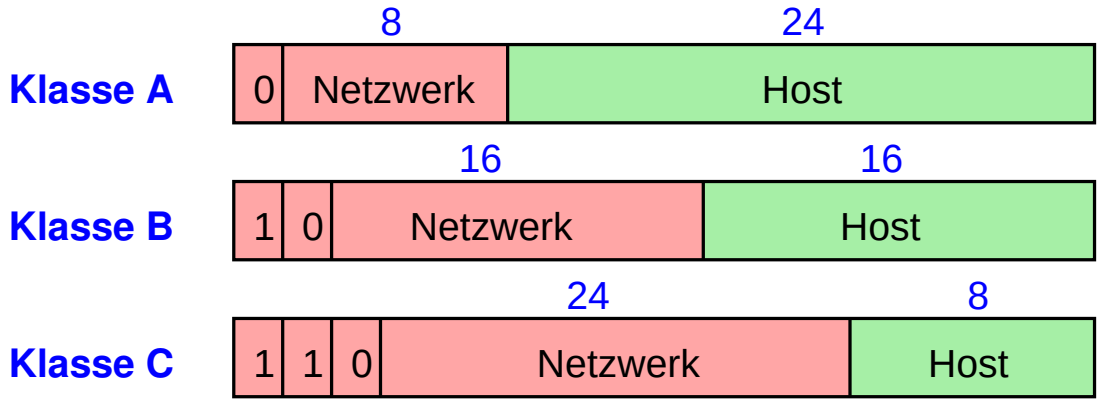
Anmerkungen zu Folie 175:

CIDR ist die Abkürzung für *Classless InterDomain Routing*. Die Motivation für die Einführung von CIDR in IPv4 und die Auswirkungen auf die IP-Weiterleitung bzw. das Routing sind im RFC 4632 beschrieben.

Netzadressen werden der Einfachheit halber wie normale IP-Adressen notiert und gespeichert. Per Konvention werden dabei alle Bits, die normalerweise die Hostadresse angeben, auf 0 gesetzt.

IPv4 Adressen

- Adressklassen:



- Schreibweise: byteweise dezimal, durch Punkt getrennt
  - z.B. 131.159.31.17
- Bei klassenloser Adressierung ggf. mit Angabe der Präfixlänge
  - z.B. 131.159.31.17/16 oder 131.159.30.0/24

Anmerkungen zu Folie 176:

- IPv4-Adressen werden byteweise dezimal aufgeschrieben, wobei die einzelnen Byte-Werte durch Punkte getrennt werden. Z.B.: 131.159.32.17
- Die klassenbasierte Adressierung erlaubt 128 Netzwerke der Klasse A mit jeweils max. 2^{24}-2 Hosts, 16384 Klasse-B-Netze mit jeweils max. 65534 Hosts sowie 2^{21} Klasse-C-Netze mit max. 254 Hosts.
- In jedem Netz sind immer zwei spezielle Werte für den Hostteil reserviert:
  - Sind alle Bits im Hostteil einer Zieladresse auf 1 gesetzt, z.B. in der Adresse 131.159.255.255 (Klasse B), so bedeutet dies einen Broadcast im angegebenen Netz (hier 131.159.0.0). Die Adresse 255.255.255.255 als Zieladresse bewirkt immer einen Broadcast im lokalen Netz.
  - Die Kombination „alle Bits auf 0“ ist immer reserviert, um die Netzadresse eindeutig bezeichnen zu können (z.B. 131.159.0.0)
- Ab 1993 wurde die klassenlose Adressierung eingeführt, bei der die Grenze zwischen Netzwerk- und Hostteil beliebig gesetzt werden kann. Da die Länge des Netzwerkteils damit nicht mehr eindeutig aus der Adresse hervorgeht, wird sie (z.B. bei Netzadressen oder auch bei Hostadressen, wenn das Netzwerk mit aus der Angabe hervorgehen soll) explizit mit angegeben, z.B. 131.159.0.0/23

IPv6 Adressen

- Typische Struktur:

48	16	64
Global Routing Prefix	Subnet ID	Interface ID

- Schreibweise:
  - 16-Bit-Teile hexadezimal, getrennt durch ':'
    - z.B. 47CD:0000:0000:0000:0000:1234:A456:0124
  - Kurzform
    - ohne führende Nullen, eine Nullfolge durch '::' ersetzt
    - z.B. 47CD::1234:A456:124
- Ggf. mit Angabe der Präfixlänge, z.B. 2000::/3

Adreßvergabe in IPv6

- Routing-Präfix: hierarchisch, i.W. nach Regionen
  - IANA vergibt Präfixe an *Regional Registries*, diese vergeben längere Präfixe an Provider, diese noch längere an Kunden
  - ermöglicht Aggregation von Routing-Information
- Interface-ID: manuell vergeben oder EUI-64 (∼ MAC-Adresse)
- Beispielstruktur einer IPv6-Adresse:

2001 : 06B8 : 0001 : 5270 : 021F : 9EFF : FEFC : 7AD0

- 2001 : 06B8 — Regional Registry (RIPE NCC)
- 06B8 — ISP
- 0001 — Site
- 5270 — Subnetz
- 021F : 9EFF : FEFC : 7AD0 — EUI−64 (MAC−Adr. 00:1F:9E:FC:7A:D0)

Anmerkungen zu Folie 178:

- Die IANA (*Internet Assigned Numbers Authority*) ist zuständig für die globale Koordinierung von Internet-Ressourcen, z.B. IP-Adressen. Sie vergibt Adressbereiche an die *Regional Registries*, z.B. ARIN für Nordamerika und RIPE NCC für Europa.
- Die im Beispiel gezeigte Struktur wurde im Wesentlichen in den Internet-Standards RFC 2073 und RFC 2374 so festgelegt. Später wurde im RFC 3587 diese Festlegung de facto in weiten Teilen rückgängig gemacht, da Implementierungen die Struktur nicht ausnutzen sollten. Geblieben ist die auf Folie 177 gezeigte Einteilung in „Global Routing Prefix“, „Subnet ID“ und „Interface ID“, wobei der Standard die Längen nicht verbindlich vorschreibt. Für IPv6-Adressen, die nicht mit 000 (binär) beginnen, ist die Länge der Interface-ID aber auf 64-Bit festgelegt (siehe dazu auch RFC 3513).
- EUI-64 ist ein von der IEEE standardisiertes Adressformat, das das EUI-48 Format für MAC-Adressen erweitert.
- Im Beispiel zu EUI-64 ist die MAC-Adresse 00:1F:9E:FC:7A:D0. Diese wird durch Einschieben von FFFE in der Mitte auf 64-Bit erweitert. Zusätzlich muss noch das „universal/local“ (U/L) Bit, das ist das Bit mit der Wertigkeit 2 im ersten Byte invertiert werden (zur Begründung: siehe Abschnitt 2.5.1 in RFC 4291), aus 00:1F wird daher 021F.

178-1

5.2.1 Adressierung in IP ...

Gültigkeitsbereiche von IP-Adressen

- **Global**
  - weltweit eindeutig
  - Pakete mit diesen Adressen werden im globalen Internet weitergeleitet
- **Link Local**
  - nur innerhalb eines physischen LANs eindeutig
  - Router leiten Pakete mit diesen Adressen nicht weiter
- **Private** (IPv4) bzw. **Unique Local** (IPv6)
  - nur innerhalb eines privaten Netzes eindeutig
    - z.B. eine Organisation
  - Pakete mit diesen Adressen werden nicht im globalen Internet weitergeleitet

Anmerkungen zu Folie 179:

- Da link-lokale Adressen nur innerhalb eines physischen LANs gültig sind, können Schnittstellen mit diesen Adressen nicht aus anderen Netzen aus angesprochen werden. Eine mögliche Anwendung sind z.B. Netzwerkdrucker, die typischerweise nur aus dem lokalen Netz heraus erreichber sein sollen.
- Da private IP-Adressen nicht weltweit eindeutig sind, kann ein privates IP-Netz nicht ohne Zusatzmaßnahmen (NAT, siehe 5.8) an das öffentliche Internet angeschlossen werden.
- Bei *Unique Local* Adressen beinhaltet das *Global Routing Prefix* eine 40-Bit lange Zufallszahl. Dadurch wird erreicht, daß trotz unkoordinierter und dezentraler Adreßvergabe die Netzwerkadresse sehr wahrscheinlich auch global eindeutig ist. Dadurch kann man einige Probleme privater Netze vermeiden (z.B., wenn zwei private Netze verbunden werden sollen). Siehe RFC 4193.

179-1

5.2.1 Adressierung in IP ...

Arten von IP-Adressen

- **Unicast**: Adresse für genau eine Netzwerk-Schnittstelle
  - aber: in IPv6 hat eine Schnittstelle i.d.R. mehrere Adressen
- **Multicast**: Adresse für eine Gruppe von Empfängern
- **Broadcast** (nur IPv4): alle Schnittstellen innerhalb eines Netzes
  - Adresse 255.255.255.255: Broadcast im lokalen Netz
- **Anycast**: nächstgelegene Schnittstelle aus einer Menge
  - mehrere Hosts mit identischer Adresse und gleicher Funktion
  - Router leiten Pakete zum nächstgelegenen Host weiter
  - Anwendung z.B. für DNS-Server

Spezielle Adreß-Bereiche in IP

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

Anmerkungen zu Folie 181:

- Die Adressbereiche wurden soweit möglich durch eine Adresse und die Präfixlänge angegeben, z.B.:
  - 169.254.0.0/16 = 169.254.0.0 - 169.254.255.255
  - 10.0.0.0/8 = 10.0.0.0 - 10.255.255.255
  - 172.16.0.0/12 = 172.16.0.0 - 172.31.255.255
  - 224.0.0.0/4 = 224.0.0.0 - 239.255.255.255
- Neben den in der Tabelle genannten gibt es noch etliche weitere Adressbereiche, die für spezielle Zwecke reserviert sind, u.a.:
  - 0.0.0.0/8 – U.a. für Default-Route in Routingtabellen
  - 192.0.2.0/24 – TEST-NET: nur für Beispiele und Dokumentation
- In IPv4 wird typischerweise die Adresse 127.0.0.1 als Loopback-Adresse verwendet. Eine Loopback-Adresse verweist immer auf den eigenen Host zurück.
- Die Loopback-Adresse ::1 in IPv6 lautet ausgeschrieben: 0000:0000:0000:0000:0000:0000:0000:0001
- Für Anycast gibt es keinen eigenen Adreß-Bereich. Stattdessen verwendet man normale Global Unicast Adressen, die in diesem Fall an mehr als eine Schnittstelle zugewiesen werden.

5.2.2 IP-Weiterleitung

★★

Grundlagen

- Jedes IP-Datagramm enthält IP-Adresse des Ziels
  - Netzadresse kennzeichnet das physisches Netz des Ziels
- Hosts mit gleicher Netzadresse kommunizieren direkt über ihr lokales Netz
- Jedes physische Netz, das Teil des Internets ist, ist mit mindestens einem **Router** verbunden
  - Router hat mehrere Netzwerk-Schnittstellen
    - jede Schnittstelle hat ihre eigene IP-Adresse
    - ***Gateway***: Schnittstelle eines Routers im lokalen Netz
- Aufgabe bei der Weiterleitung:
  - an welche Schnittstelle muß ein IP-Paket mit gegebener Zieladresse weitergeleitet werden?

5.2.2 IP-Weiterleitung ...

★★★

Routing-Tabelle (Weiterleitungstabelle)

- Weiterleitung wird durch Routing-Tabelle gesteuert
  - in Routern und auch in normalen Hosts
- Prinzipieller Aufbau der Tabelle:

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

- Netzadresse und Präfixlänge zusammen identifizieren ein Netz
  - bei IPv4 statt Präfixlänge ggf. auch Subnetzmaske (☞ **S. 187**)
- *Next Hop* = Router bzw. Schnittstelle, an den/die das Paket weitergegeben werden soll, falls Ziel im angegebenen Netz liegt

Anmerkungen zu Folie 183:

- In Tabelleneinträgen für lokale Netze, an die der Router/Host direkt angeschlossen ist, spezifiziert *Next Hop* die Schnittstelle, über die dieses lokale Netz erreichbar ist.

  In der Weiterleitungstabelle eines **Hosts** findet sich dabei immer ein Eintrag für das lokale Netz des Hosts. Dieser sorgt dafür, daß IP-Pakete, die an ein Ziel im lokalen Netz adressiert sind, über die entsprechende Schnittstelle direkt an das Ziel versendet werden und nicht an einen Router.

- In Tabelleneinträgen für Netze, an die der Router/Host nicht direkt angeschlossen ist, spezifiziert *Next Hop* typischerweise die IP-Adresse des Routers, an die Pakete für dieses Netz weitergeleitet werden sollen.

183-1

5.2.2 IP-Weiterleitung ...

★★★

Vorgehensweise bei der Weiterleitung

- Algorithmus:
  - suche Eintrag i mit größter *Präfixlänge*$_i$, für den gilt:
    - Zieladresse und *Netzadresse*$_i$ stimmen in den ersten *Präfixlänge*$_i$ Bits überein
    - d.h. Zieladresse liegt in dem durch *Netzadresse*$_i$ und *Präfixlänge*$_i$ gegebenen Netz
  - falls Eintrag gefunden: Weiterleiten an *NextHop*$_i$
  - sonst: Verwerfen des Pakets

- Typisch: zusätzlicher Tabellen-Eintrag für Default-Route

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

  - dieser Eintrag „paßt“ auf jede Zieladresse

Anmerkungen zu Folie 184:

Da jede Adresse in den ersten 0 Bits mit der Adresse 0.0.0.0 (IPv4) bzw. ::0 (IPv6) übereinstimmt, passt der Eintrag für die Default-Route auf jede Ziel-IP-Adresse. Da der Eintrag aber die kürzest kürzest mögliche Präfix-Länge hat (nämlich 0), wird er nur dann verwendet, wenn es keinen anderen passenden Eintrag in der Weiterleitungstabelle gibt.

184-1

5.2.2 IP-Weiterleitung ...

★★★

Beispiel

	Netzadresse	Subnetzmaske	Next Hop
0	0.0.0.0	0.0.0.0	10.0.0.1
1	141.99.0.0	255.255.0.0	10.1.0.1
2	141.99.179.0	255.255.255.0	Interface 1
3	141.99.128.0	255.255.192.0	10.3.4.1

- IP-Paket mit Ziel 141.99.178.6
  - 0: 141.99.178.6 AND 0.0.0.0 = **0.0.0.0** $(n = 0)$
  - 1: 141.99.178.6 AND 255.255.0.0 = **141.99.0.0** $(n = 16)$
  - 2: 141.99.178.6 AND 255.255.255.0 = **141.99.178.0**
  - 3: 141.99.178.6 AND 255.255.192.0 = **141.99.128.0** $(n = 18)$
- Weiterleitung an Router 10.3.4.1!

Anmerkungen zu Folie 185:

Der Router bestimmt für jeden Eintrag die UND-Verknüpfung der Ziel-IP-Adresse mit der jeweiligen Subnetzmaske. Stimmt das Ergebnis mit der Netzadresse des Eintrags überein, haben wir einen Treffer.

Am Ende wird der Treffer verwendet, der die größte Präfixlänge (oder gleichbedeutend, die numerische größte Subnetzmaske) besitzt.

Zum Eintrag 3:

- 178 = 10110010_2, 192 = 11000000_2.
- Damit 178 AND 192 = 10110010_2 AND 11000000_2 = 10000000_2 = 128.
- Die anderen Bytes sind problemlos, da für alle x:
  - x AND 255 = x
  - x AND 0 = 0

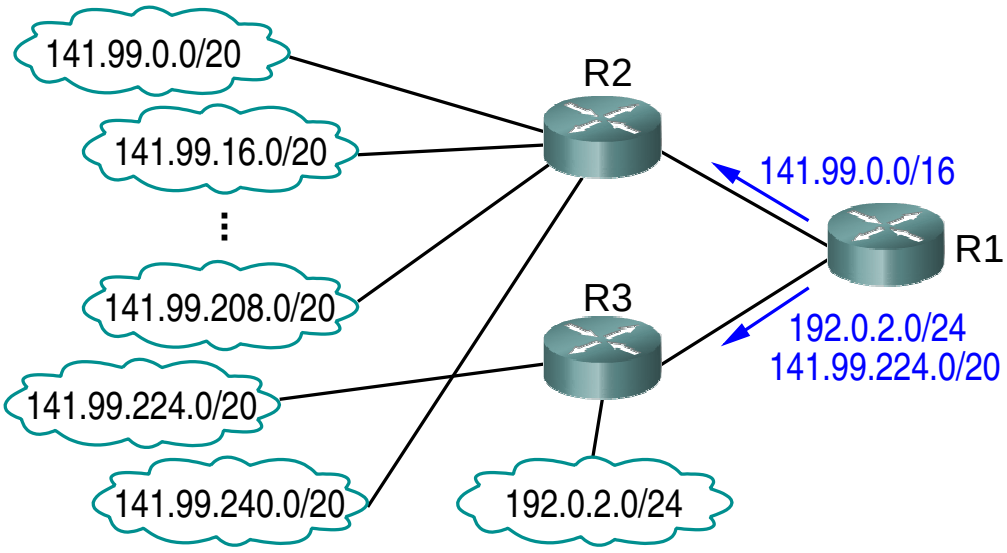
185-1

5.2.2 IP-Weiterleitung ...

(Animierte Folie)

Motivation für die Suche nach dem längsten Präfix

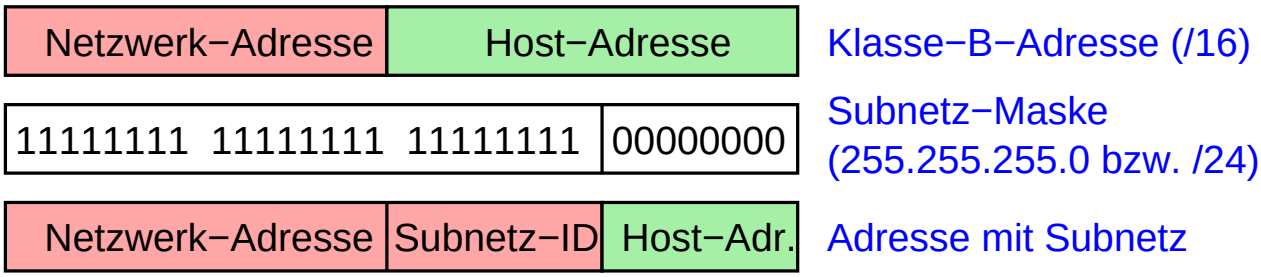
- Erlaubt überlappende Einträge in der Routing-Tabelle
- Damit auch Zusammenfassung „nicht zusammenhängender“ Adreßbereiche möglich:



★★★

Bildung von Subnetzen

- Motivation: Unterteilung eines großen Netzes (z.B. Firmennetz) in meherere kleinere Netze (z.B. Abteilungsnetze)
  - nach „aussen“ hin ist nur das Gesamtnetz sichtbar
- IPv4: ein Teil der Host-Bits wird für die Subnetz-ID „geborgt“



  - i.a. werden Subnetze unterschiedlicher Größe erzeugt
  - Subnetzmaske legt Präfixlänge für das Subnetz fest
- IPv6: Subnetz-ID in eigenem 16-Bit-Feld (☞ **S. 177**)

Anmerkungen zu Folie 187:

Man kann bei **IPv4** mehrere Arten des *Subnettings* unterscheiden:

- Beim einfachen *Subnetting* besitzen alle Subnetze eines Netzes dieselbe Subnetzmaske. Wenn die Subnetzmaske k Bits für die Subnetz-ID bereitstellt und m Bits für die Hostadresse verbleiben, wird das Netz also in 2^k Subnetze der Größe 2^m-2 unterteilt (zwei Werte sind für die (Sub-)Netz- und die Broadcast-Adresse reserviert).
- Beim hierarchischen Subnetting kann man ein Subnetz durch eine „längere“ Subnetzmaske weiter unterteilen. Ein Subnetz mit Subnetzmaske 255.255.255.0 kann z.B. mit einer Subnetzmaske 255.255.255.128 in zwei Sub-Subnetze unterteilt werden.
- Beim VLSM (*Variable Length Subnet Mask*) *Subnetting* wird ein Netz durch Subnetzmasken unterschiedlicher Länge in verschieden große Subnetze unterteilt.

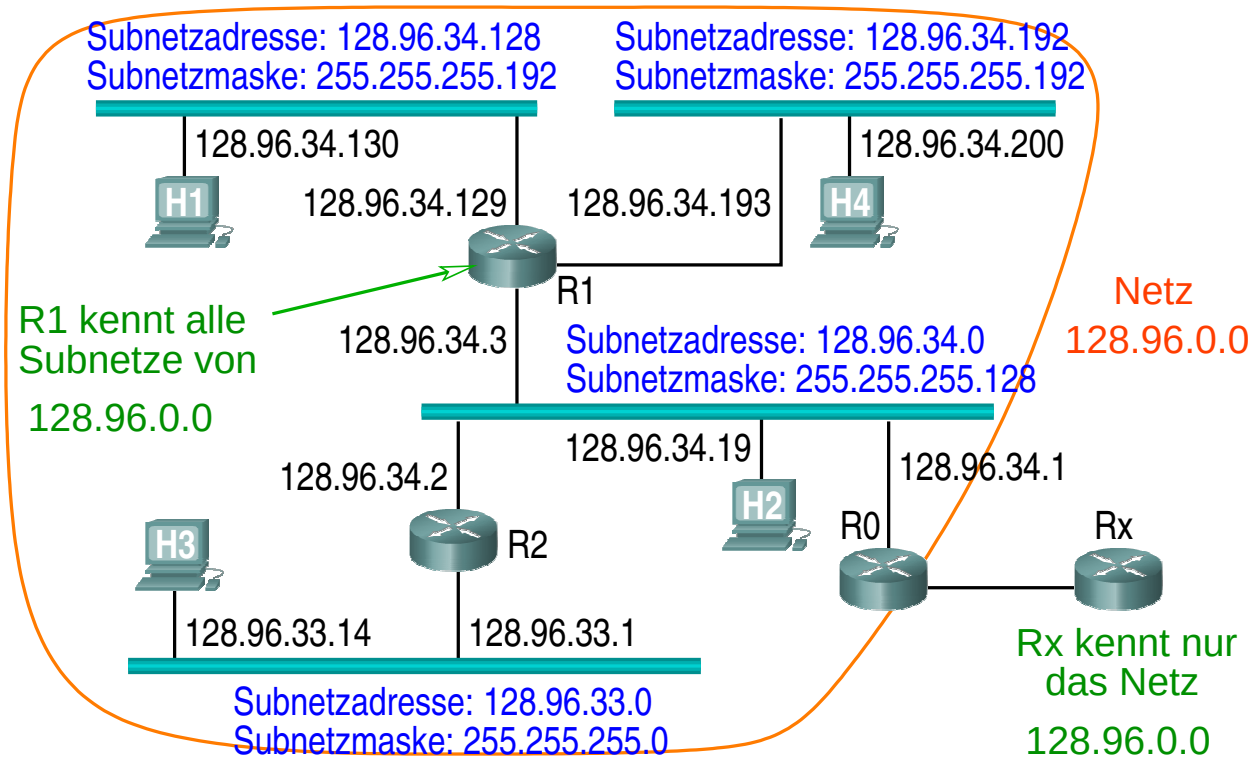
Im Endeffekt wird ein Subnetz aber immer wie auf Folie 175 beschrieben durch seine (Sub-)Netzadresse und die Präfixlänge identifiziert, wobei die Präfixlänge direkt aus der Subnetzmaske hervorgeht (Zahl der 1-Bits in der binären Subnetzmaske). Häufig wird daher auch bei IPv4 statt der (Sub-)Netzmaske einfach die Präfix-Länge angegeben. Z.B. wird ein Subnetz 128.96.34.0 mit Subnetzmaske 255.255.255.128 auch als 128.96.34.0/25 beschrieben.

Bei **IPv6** ist das Subnetting wesentlich einfacher, da man sich um die Größe des Subnetzes keine Gedanken machen muss (jedes Subnetz kann aufgrund der 64 Host-Bits über 10^{19} Hosts aufnehmen), so daß man die Subnetze einfach nur durchnumerieren muß. Die aufgrund der 16 Bit großen Subnetz-ID maximal mögliche Zahl von 65536 Subnetzen reicht in der Praxis ebenfalls immer aus.

5.2.3 Subnetting ...

(Animierte Folie)

Ein Netz mit Subnetzen (IPv4):



★★★

Bestimmung der (Sub-)Netzzugehörigkeit

- Wie bestimmt ein Host bzw. Router, ob eine IP-Adresse xyz in einem gegebenen (Sub-)Netz liegt?
- Gegeben: (Sub-)Netzadresse und Präfixlänge n
  - stimmen xyz und Netzadresse in den ersten n Bits überein?
  - (vgl. Weiterleitungsalgorithmus auf S. 184)
- Gegeben: (Sub-)Netzadresse und (Sub-)Netzmaske
  - gilt xyz AND Netzmaske $=$ Netzadresse ?
- Beispiel: 128.96.34.19 AND 255.255.255.128 $=$ 128.96.34.0

Erstellung von Subnetzen in IPv4

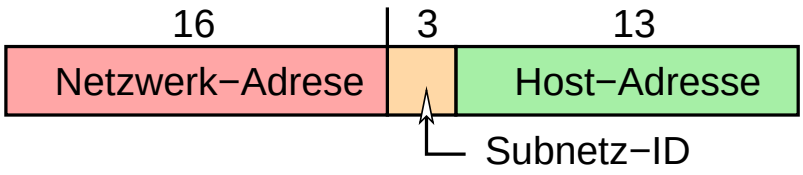
- Einschränkungen durch Realisierung:
  - Zahl der Hostadressen in einem Subnetz ist immer eine Zweierpotenz
    - bei Präfixlänge l des Subnetzes also 2^{32-l}
    - zwei Hostadressen sind reserviert:
      - $0...0_2$: Adresse des Netzwerks selbst
      - $1...1_2$: Broadcastadresse
    - damit $2^{32-l} - 2$ Hosts möglich
  - Subnetz mit 2^k Hostadressen kann nur an einer durch 2^k teilbaren Adresse beginnen
    - z.B. Adressbereich 141.99.179.64 - 141.99.179.191 ist nicht möglich
- Im folgenden: ursprüngliche Präfixlänge sei n

Mögliche Vorgehensweisen

- Einfaches Subnetting
  - feste Anzahl von k Bits des (bisherigen) Hostanteils ($32 - n$ Bits) wird für die Subnetz-ID „geborgt“
  - identische Subnetzmaske für alle Subnetze
  - ergibt 2^k Subnetze mit identischer Größe $2^{32-n-k} - 2$
- Hierarchisches Subnetting
  - bei Bedarf werden einzelne Subnetze weiter unterteilt
  - dabei entstehen längere Subnetzmasken
- VLSM (*Variable Length Subnet Mask*)
  - Subnetze werden von Anfang an entsprechend ihrer Größe durch Subnetzmasken unterschiedlicher Länge realisiert

Beispiel: Einfaches Subnetting

- Im Netzwerk 141.99.0.0/16 sollen 6 Subnetze realisiert werden
- Für 6 Subnetze müssen von den 16 Host-Bits 3 Bits „geborgt“ werden ($6 \leq 2^3$):

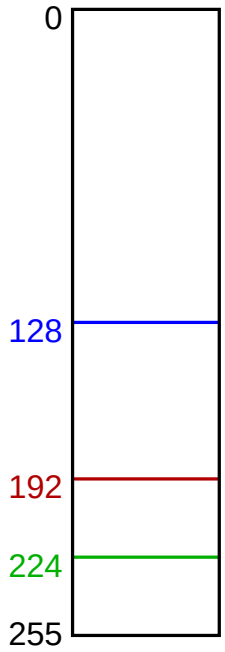


- Es entstehen 8 Subnetze für jeweils $2^{13} - 2 = 8190$ Hosts
  - 141.99.0.0 - 141.99.31.255; 141.99.32.0 - 141.99.63.255; ...; ...; 141.99.224.0 - 141.99.255.255
  - Subnetzmaske 255.255.224.0, Präfixlänge: 19
- Nachteil:
  - Subnetze unterschiedlicher Größe sind nicht möglich

(Animierte Folie)

Beispiel: Hierarchisches Subnetting

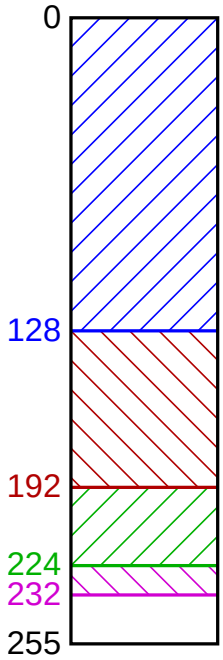
- Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- Durch Borgen eines Bits entstehen 2 Subnetze:
  - 1.2.3.0 - 1.2.3.127; 1.2.3.128 - 1.2.3.255
  - Netzmaske 255.255.255.128, Präfixlänge: 25
- Weitere Unterteilung des zweiten Subnetzes:
  - 1.2.3.128 - 1.2.3.191; 1.2.3.192 - 1.2.3.255
  - Netzmaske 255.255.255.192, Präfixlänge: 26
- Nochmal Unterteilung des zweiten Subnetzes:
  - 1.2.3.192 - 1.2.3.223; 1.2.3.224 - 1.2.3.255
  - Netzmaske 255.255.255.224, Präfixlänge: 27



(Animierte Folie)

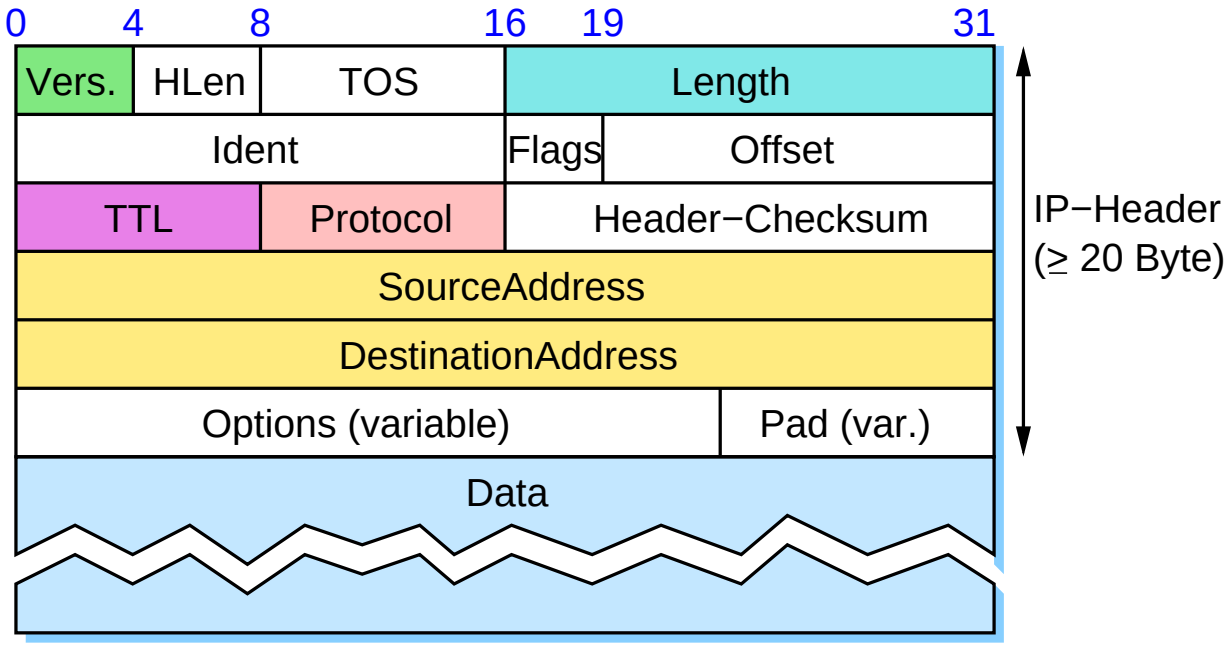
Beispiel: VLSM

- Im Netzwerk 1.2.3.0/24 sollen 4 Subnetze realisiert werden, die 100, 50, 25 und 5 Hosts aufnehmen können
- Teile möglichst kleine Subnetze in **absteigender** Reihenfolge der Größe zu:
  - 100 Hosts: Größe $128 = 2^7$, P.länge $32 - 7 = 25$
    - Netzadr. 1.2.3.0, N.maske 255.255.255.128
  - 50 Hosts: Größe $64 = 2^6$, P.länge $32 - 6 = 26$
    - Netzadr. 1.2.3.128, N.maske 255.255.255.192
  - 25 Hosts: Größe $32 = 2^5$, P.länge $32 - 5 = 27$
    - Netzadr. 1.2.3.192, N.maske 255.255.255.224
  - 5 Hosts: Größe $8 = 2^3$, P.länge $32 - 3 = 29$
    - Netzadr. 1.2.3.224, N.maske 255.255.255.248



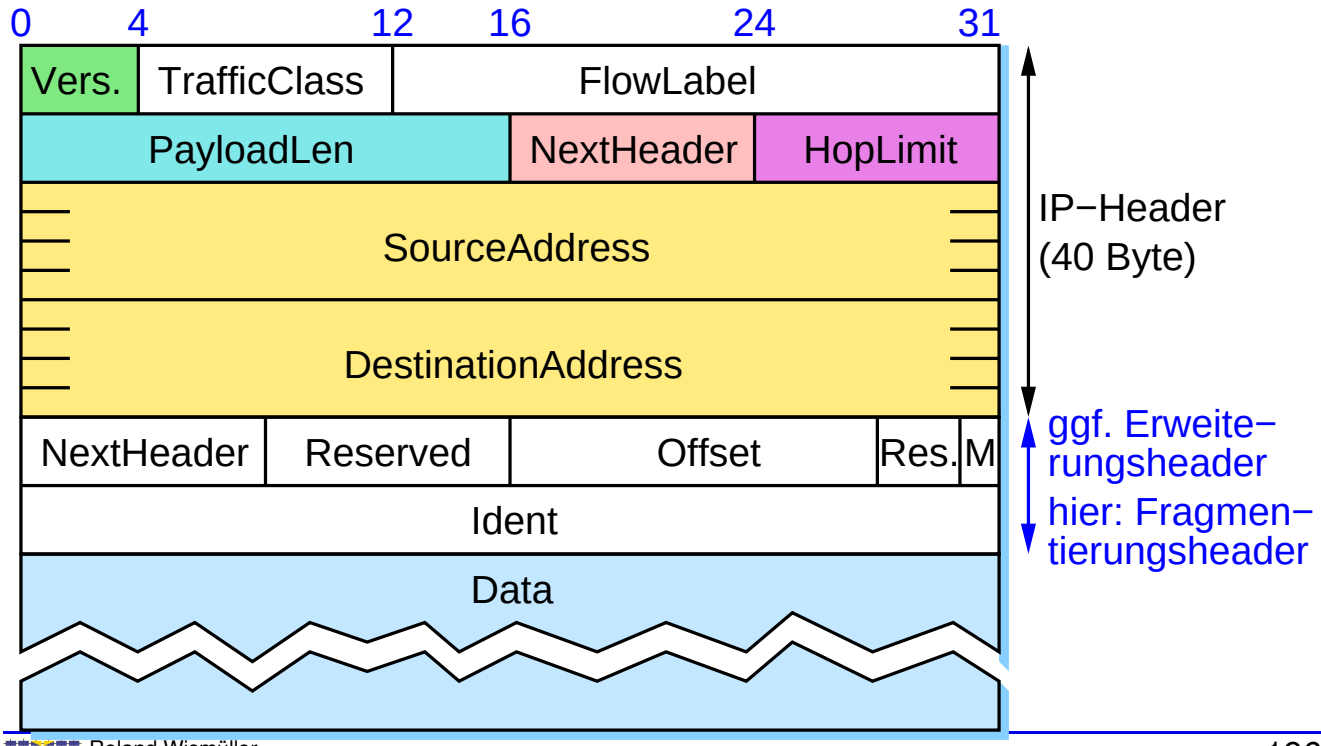
5.3 Aufbau eines IP-Pakets

IPv4



5.3 Aufbau eines IP-Pakets ...

IPv6



Anmerkungen zu Folie 196:

- Optionale Felder des IP-Headers sind bei IPv6 nicht mehr im IP-Header selbst gespeichert, sondern werden in Erweiterungsheader ausgelagert, die bei Bedarf auf den eigentlichen IP-Header folgen. Das Feld **NextHeader** gibt jeweils an, welcher Header als nächstes folgt, z.B. ein (weiterer) IP-Erweiterungsheader oder ein TCP-Header, falls der Datenteil des Pakets ein TCP-Segment enthält.
- Für die verschiedenen Erweiterungsheader ist eine verbindliche Reihenfolge festgelegt, um die Bearbeitung beim Empfänger zu vereinfachen.
- Erweiterungsheader sind (derzeit) definiert für:
  - Optionen für Teilstrecken
    - Datagramme mit $>$ 64 KB (für Hochleistungsrechner)
  - Optionen für Zielrechner
  - Routing-Optionen
    - Angabe eines Bereichs, über den Paket geleitet werden soll
  - Fragmentierung (☞ **5.4**)
  - Authentifizierung / Verschlüsselung (☞ **RN_II, Secure IP**)
  - Optionen für Mobilität (☞ **RN_II, Mobile IP**)

196-1

5.3 Aufbau eines IP-Pakets ...

★

Bedeutung der wichtigsten Felder

- **HLen**: Länge des Headers in 32-Bit Worten
- **TOS** / **TrafficClass** / **FlowLabel**: für *Quality of Service*
- **Length**: Gesamtlänge des Pakets inkl. IP-Header in Bytes
  PayloadLen: Länge des Pakets ohne Basis-IP-Header
  - maximal 65535 Bytes
- **Ident** / **Flags (M)** / **Offset**: für Fragmentierung / Reassembly
- **TTL** / **HopLimit**: zur Erkennung endlos kreisender Pakete
  - wird von jedem Router heruntergezählt, bei 0 wird das Paket verworfen
- **Protocol** / **NextHeader**: kennzeichnet das im Datenteil versendete Protokoll (z.B. TCP, UDP; für Demultiplexing)
  oder (bei IPv6) ggf. den Typ des folgenden Erweiterungsheaders

Anmerkungen zu Folie 197:

- Das Feld **Version** (Vers.) enthält die IP-Version (4 oder 6).
- Mit **TOS** bzw. **TrafficClass** kann ein IP-Paket einer bestimmten Verkehrsklasse zugeordnet werden, das **FlowLabel** ordnet das Paket einem bestimmten Datenfluß zu. Beide Angaben werden für *Quality of Service*-Mechanismen verwendet (☞ **RN_II**).
- Das Feld **Flags** bei IPv4 enthält neben dem M-Flag (*more fragments*), das für die Fragmentierung wichtig ist, zwei weitere Flags: eines ist reserviert und hat immer den Wert 0, das andere (DF, *don't fragment*) verbietet, daß das Paket von Routern weiter fragmentiert wird. Bei IPv6 ist dieses Bit nicht notwendig, da eine Weiterfragmentierung eines Pakets prinzipiell verboten ist (☞ **5.4**).
- **TTL** bedeutet eigentlich *Time-To-Live*. Das Feld beinhaltet aber auch bei IPv4 keine Zeitangabe, sondern einen Zähler und wurde daher bei IPv6 umbenannt.
- **Header-Checksum** ist eine einfache Prüfsumme über den IPv4-Header, die wegen der praktisch in allen Schicht-2-Protokollen vorhandenen CRC-Felder überflüssig ist.

197-1

5.4 IP: Fragmentierung/Reassembly

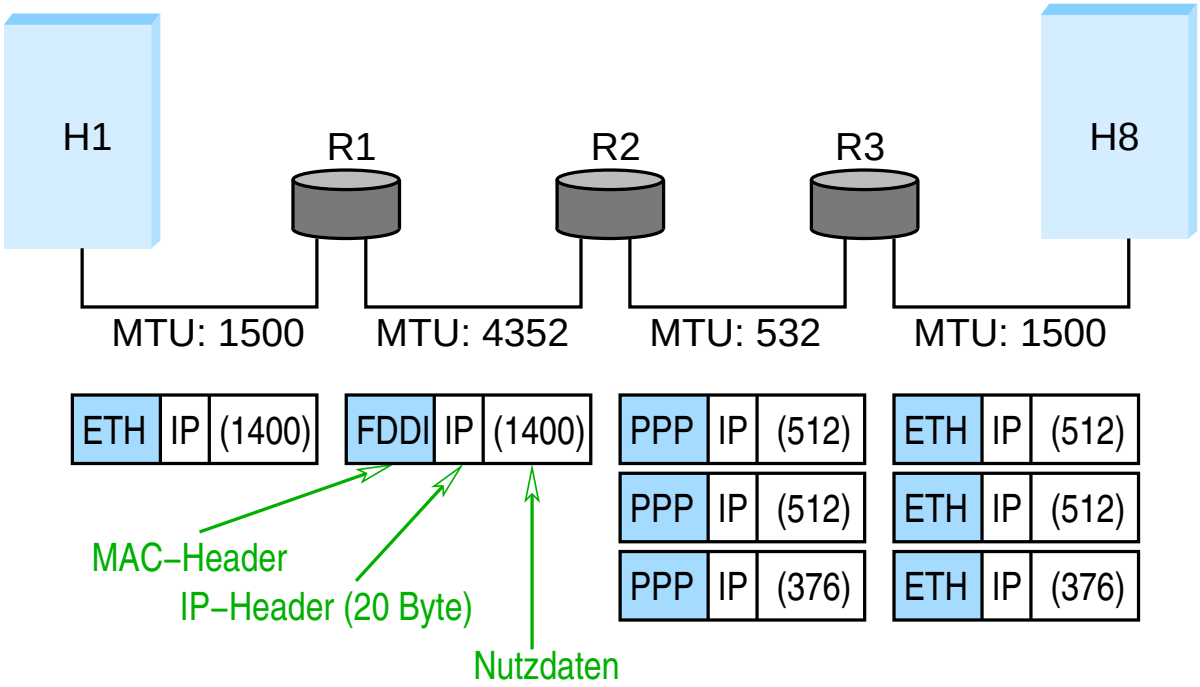
Problem für IP

- Jedes lokale Netzwerk definiert eine (unterschiedliche) maximale Framegröße (**MTU**: ***Maximum Transmission Unit***)

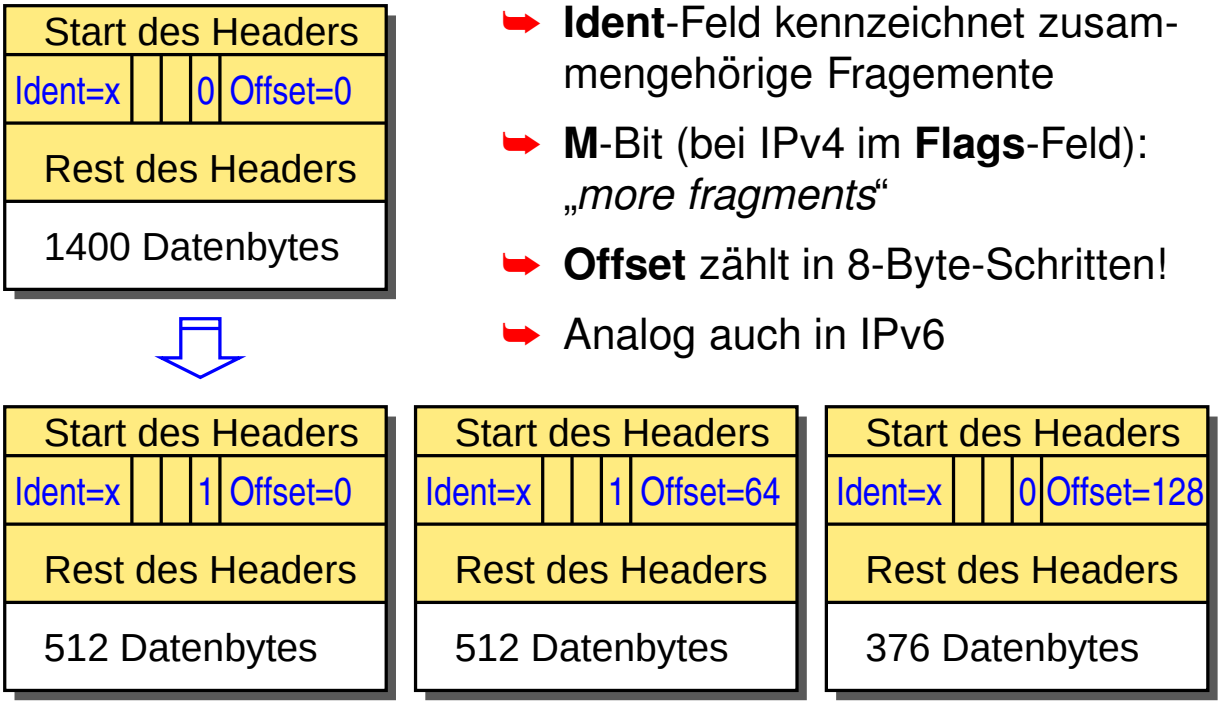
Netzwerk	MTU [Byte]
Ethernet	1500
FDDI	4352
4 Mbits/sec Token-Ring (IEEE 802.5)	4464
16 Mbits/sec Token-Ring (IBM)	17914

- Alternativen für IP:
  - max. Paketgröße = minimale MTU (welcher Netze??)
  - Möglichkeit der Fragmentierung von IP-Paketen

Beispiel zur Fragmentierung (IPv4)



Fragmentierung im Detail (IPv4)



- **Ident**-Feld kennzeichnet zusammengehörige Fragemente
- **M**-Bit (bei IPv4 im **Flags**-Feld): „*more fragments*“
- **Offset** zählt in 8-Byte-Schritten!
- Analog auch in IPv6

Anmerkungen zu Folie 200:

Bei IPv6 wird die Fragmentierungsinformation in einem eigenen Erweiterungs-Header gespeichert:

0	8	16	29	31
NextHeader	Reserved	**Offset**	Res.	**M**
Ident				

200-1

5.4 IP: Fragmentierung/Reassembly ...

★★

Fragmentierung im Detail ...

- Fragmentierung geschieht bei Bedarf im Sender bzw. bei IPv4 auch in den Routern
- Jedes Fragment ist ein eigenständiges IP-Datagramm
  - IPv4: ein Fragment kann ggf. nochmals fragmentiert werden
- Empfänger baut aller Fragmente wieder zusammen
  - falls ein Fragment nicht ankommt, werden alle anderen zugehörigen Fragmente verworfen
- Bei IPv6 und meist auch bei IPv4: „*Path MTU Discovery*“
  - Fragmentierung im Router verboten (IPv4: **DF**-Flag im Header) $\Rightarrow$ ggf. Fehlermeldung an Sender (über ICMP, siehe später)
  - Sender kann minimale MTU ermitteln

5.5 ICMP

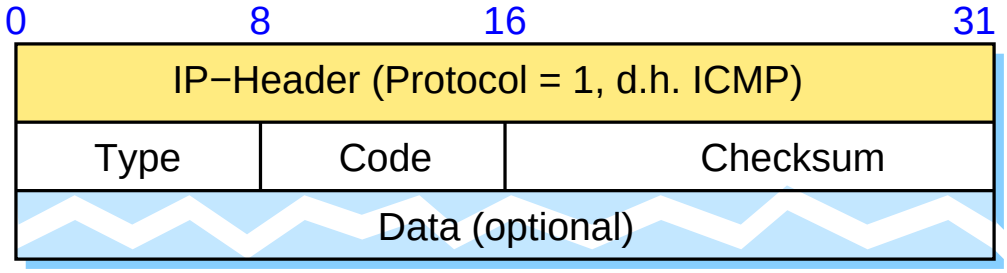
★★

ICMP: *Internet Control Message Protocol*

- Datagramme für Fehler- und Verwaltungsmeldungen:
  - Ziel nicht erreichbar
  - Reassembly fehlgeschlagen
  - Fragmentierung nicht erlaubt, aber erforderlich
  - TTL wurde 0
  - *Redirect*: besserer Router für das Ziel
  - *Echo Request / Reply*: z.B. für `ping` und `traceroute`
  - *Router Solicitation / Advertisement* (nur IPv6): Suche nach / Bekanntgabe von lokalen Routern
  - *Neighbor Solicitation / Advertisement* (nur IPv6): Adreßübersetzung (siehe später)
  - ...

5.5 ICMP ...

Aufbau eines ICMP-Pakets (siehe auch Wireshark-Aufzeichung)

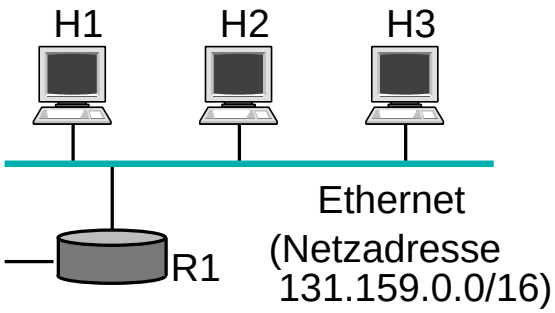


- *Echo* / *Echo Reply* (Typ = 8 / 0):
  - Data: Identifikator + Sequenznummer
- Ziel nicht erreichbar (Typ = 3):
  - Code: z.B. 0 ≙ Netz, 1 ≙ Host, 3 ≙ Port nicht erreichbar
  - Data: IP-Header + erste 64 Datenbytes des unzustellbaren Pakets

5.6 Adressübersetzung

Motivation

- IP-Weiterleitung bringt ein Paket in das richtige LAN
- Wie funktioniert IP-Kommunikation **innerhalb** eines LANs?
- Beispiel:
  - R1 empfängt Paket für Rechner 131.159.32.12
  - R1 muß Paket über Ethernet an diesen Rechner weiterleiten
  - Woher weiß R1 die MAC-Adresse des Rechners mit IP-Adresse 131.159.32.12?



- Anmerkung: dasselbe Problem tritt auch auf, wenn z.B. H1 ein Paket an IP-Adresse 131.159.32.12 senden will

5.6 Adressübersetzung ...

Motivation ...

- Problem: Umsetzung von IP-Adressen auf MAC-Adressen im lokalen Netz
  - allgemein: auf Sicherungsschicht-Adresse
- Lösungs-Alternativen:
  - MAC-Adresse in IP-Adresse kodieren?
    - bei IPv4 nicht realisierbar (Ethernet: 48 Bit MAC-Adresse!)
  - Manuell verwaltete Tabellen?
    - Verwaltungsaufwand!
  - Automatisches (dynamisches) Erstellen der Tabellen!

★★★

ARP: *Address Resolution Protocol* (IPv4)

- Annahme: ein Rechner H will ein Paket an IP-Adresse *xyz* senden, *xyz* ist im lokalen Netz
- H sucht in seinem ARP-Cache nach der zu *xyz* gehörigen MAC-Adresse
- Falls gefunden: Paket an diese MAC-Adresse senden
- Sonst:
  - H sendet Anfrage (*ARP-Request*) per Broadcast in das LAN: „wer hat IP-Adresse *xyz*?“
  - Der betroffene Rechner sendet Antwort (*ARP Reply*) mit seiner IP- und MAC-Adresse zurück
  - H trägt Zuordnung in sein ARP-Cache ein
    - automatische Löschung nach bestimmter Zeit ohne Nutzung

Anmerkungen zu Folie 206:

Jede APR-Nachricht beinhaltet vier Adressfelder (siehe RFC 826):

- *Source Hardware Address* (SHA): die MAC-Adresse des Senders (bei einer ARP-Antwort steht hier die gesuchte MAC-Adresse),
- *Source Protocol Address* (SPA): die IP-Adresse des Senders,
- *Target Hardware Address* (THA): die MAC-Adresse des Empfängers (bei einer ARP-Anfrage mit 00:00:00:00:00:00 belegt),
- *Target Protocol Address* (TPA): die IP-Adresse des Empfängers.

RFC 826 spezifiziert folgendes Verhalten beim Empfang eines ARP-Pakets (unabnhängig davon, ob es sich um eine ARP-Anfrage oder ARP-Antwort handelt):

- wenn SPA im ARP-Cache steht, wird die MAC-Adresse im ARP-Cache auf SHA aktualisiert,
- wenn der Empfänger die IP-Adresse in TPA besitzt, trägt er die Zuordnung (SPA, SHA) in seinen ARP-Cache ein,

In gängigen Betriebssystemen wird offenbar teilweise ein anderes Verhalten implementiert.

Aufbau eines ARP-Pakets (siehe auch Wireshark-Aufzeichung)

0	8	16 … 31
Hardwareadresstyp		Protokolladresstyp
HA−Länge	PA−Länge	Operation
Sender−Hardware−Adresse (z.B. MAC−Adresse)		
Sender−HW−Ad. (MAC−A.)		Sender−Protokoll−Adr. (IP)
Sender−Protokoll−Adr. (IP)		Ziel−HW−Ad. (MAC−A.)
Ziel−Hardware−Adresse (z.B. MAC−Adresse)		
Ziel−Protokoll−Adressre (z.B. IP−Adresse)		

- ARP wird direkt über das Layer-2-Protokoll übertragen
- Typ und Länge der Sender- und Zieladressen sind frei wählbar
- Operation: *request* (1), *reply* (2)

Spezielle Verwendungen von ARP

- ARP *probe*
  - zur Prüfung von Konflikten nach Konfiguration der IP-Adresse
  - Host sendet ARP-Anfrage nach seiner gewählten IP-Adresse
- *Gratuitous* APR
  - Host sendet ARP-Anfrage mit seiner eigenen IP-Adresse als Sender- und Zieladresse
  - alle anderen Hosts aktualisieren ihren ARP-Cache
  - z.B. beim Booten oder bei Umschaltung auf Backup-Server
- *Proxy* ARP
  - Router sendet ARP-Antwort für Anfrage nach einem Host in einem andern Netz
  - anfragender Host schickt IP-Paket dann an Router

Anmerkungen zu Folie 208:

- Der RFC 5227 schreibt vor, daß beim ARP *probe* die Sender-IP-Adresse 0.0.0.0 sein muß, damit andere Hosts nicht eine (ggf. falsche) Zuordnung in Ihren ARP-Cache eintragen.
- Das *Gratuitous* APR Paket ist typischerweise das erste Paket, das ein Rechner nach dem Booten (d.h., sobald er eine IP-Adresse hat) versendet. Durch dieses Paket lernt dann übrigens auch ein Ethernet-Switch sofort, an welchem Port sich der Rechner befindet.
- Proxy-ARP kann z.B. eingesetzt werden, wenn ein Netz in mehrere Subnetze geteilt ist, die einzelnen Hosts dies aber nicht wissen (sollen). Der Router erkennt dann anhand der Ziel-IP-Adresse, dass diese in einem anderen Subnetz liegt und antwortet mit seiner eigenen MAC-Adresse.

208-1

5.6 Adressübersetzung ...

NDP: *Neighbor Discovery Protocol* (IPv6)

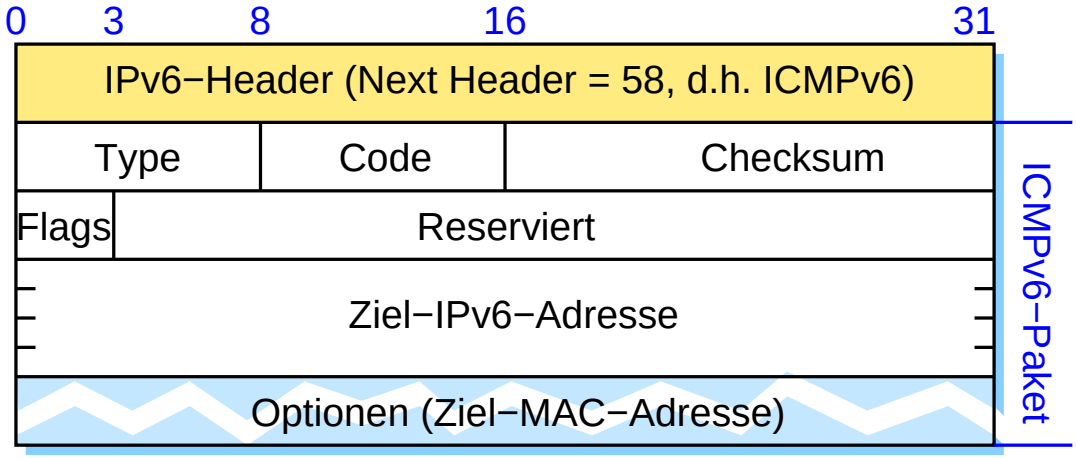
- NDP ist Teilprotokoll von ICMPv6
  - *Neighbor Solicitation/Advertisement*
- Funktionsweise der Adreßumsetzung analog zu ARP
- Unterschiede:
  - Nutzung von ICMP-Paketen statt ARP-Protokoll
    - Anfrage: *Neighbor Solicitation*
    - Antwort: *Neighbor Advertisement*
  - Anfrage nicht per Broadcast sondern per Multicast
    - an die zugehörige *Solicited Nodes* Multicast-Gruppe
    - Gruppe wird aus letzten 24 Bits der Ziel-IP-Adresse bestimmt

Anmerkungen zu Folie 209:

Die *Neighbor Solicitation* wird an die IPv6 Multicast-Adresse FF02:0:0:0:0:1:FFxx:xxxx gesendet, wobei xx:xxxx die letzten 24 Bits der Ziel-IP-Adresse sind. Diese IP-Multicast-Adresse wird fest auf die MAC-Multicast-Adresse 3333:FFxx:xxxx abgebildet, so daß die Anfrage als Ethernet-Frame mit dieser Zieladresse versendet wird. Dadurch wird erreicht, daß die Anfrage nicht von allen, sondern nur von einigen möglicherweise betroffenen Ethernet-Karten empfangen wird und entsprechend nicht von allen Rechnern bearbeitet werden muß.

5.6 Adressübersetzung ...

Aufbau eines NDP-Pakets (siehe auch Wireshark-Aufzeichung)



- *Neighbor Solicitation* (Type = 135)
  - Multicast an ff02::1:ff00:0/104 + letzte 24 Bits der Zieladresse
- *Neighbor Advertisement* (Type = 136)
  - Multicast an alle IPv6 Hosts (ff02::1)
  - Options-Feld beinhaltet MAC-Adresse des Ziels

Anmerkungen zu Folie 210:

- Das Feld „Code“ ist bei *Neighbor Solicitation* und *Neighbor Advertisement* auf 0 gesetzt.
- Das Feld „Flags“ wird nur beim *Neighbor Advertisement* genutzt:
  - Das erste Bit (**R**), wenn die antwortende Station ein Router ist.
  - Das zweite Bit (**S**) zeigt an,daß das *Neighbor Advertisement* die Antwort auf eine ***Unicast***-*Neighbor-Solicitation* ist.
  - Das dritte Bit (**O**) wird gesetzt, wenn der Empfänger den Eintrag im *Neighbor Cache* aktualisieren soll.

210-1

5.7 Automatische IP-Konfiguration

★★

DHCP: *Dynamic Host Configuration Protocol*

- Automatisiert u.a. die Vergabe von IP-Adressen (IPv4 und IPv6)
- Vorgehensweise:
  1. Rechner sendet Broadcast-Anfrage (DHCPDISCOVER)
     - Verbreitung nur im lokalen Netz
  2. DHCP-Server sendet DHCPOFFER: Angebot für IP-Adresse
     - Zuordnung statisch oder dynamisch, als Schlüssel dient MAC-Adresse des Rechners
     - ggf. auch weitere Optionen (Hostname, DNS-Server, ...)
  3. Rechner fordert angebotene Adresse vom DHCP-Server an (DHCPREQUEST)
  4. DHCP-Server bestätigt (DHCPACK)
- IP-Adresse wird nur für eine bestimmte Zeit „gemietet“
  - periodische Wiederholung der Schritte 3 und 4

Anmerkungen zu Folie 211:

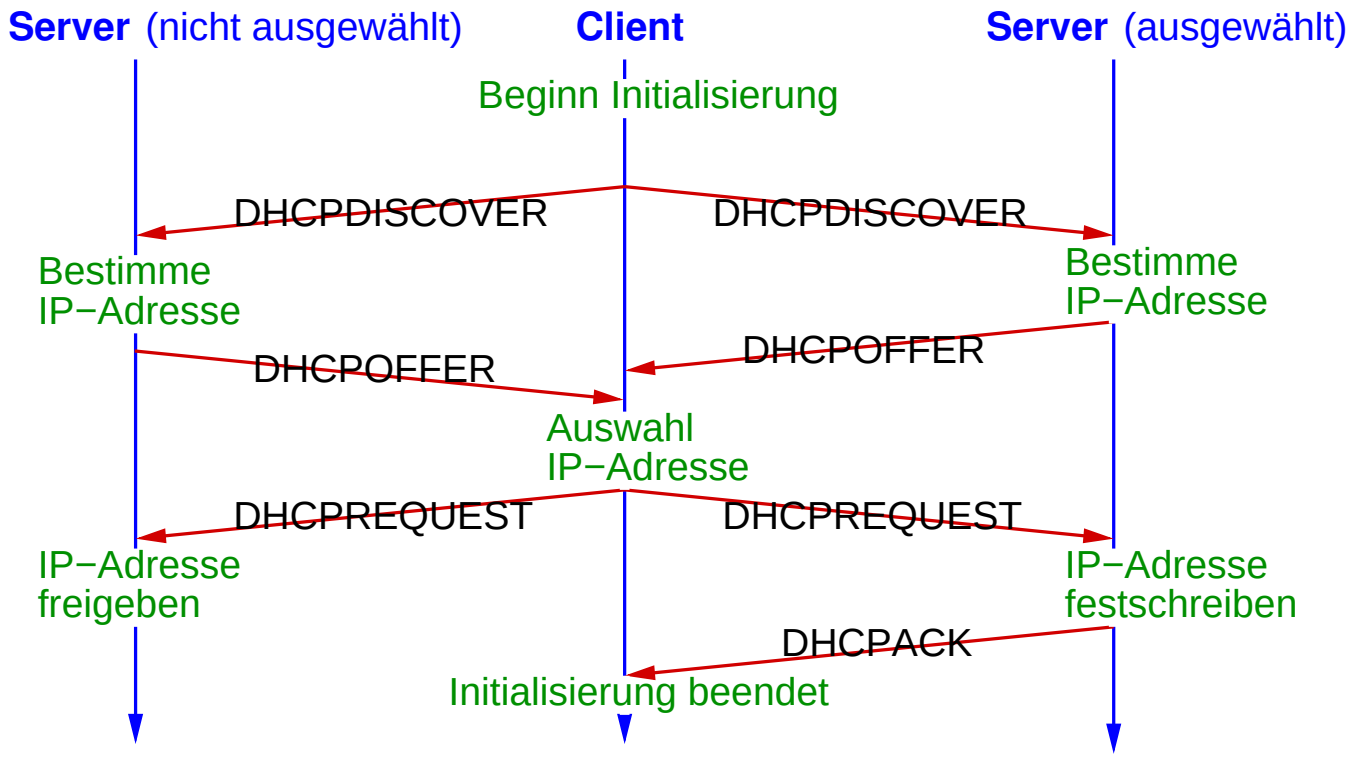
Das DHCP-Protokoll unterstützt, daß es mehrere DHCP-Server in einem Netz geben kann. DHCPDISCOVER wird von allen Servern empfangen, die auch alle antworten. Der Client kann sich dann eine Antwort aussuchen. Der DHCPREQUEST wird ebenfalls als Broadcast versendet. Er enthält zusätlich die IP-Adresse des Servers, dessen DHCPOFFER der Client annehmen will. Der DHCPREQUEST zeigt damit allen anderen Servern an, daß der Client deren Angebot nicht annimmt.

Zusätzlich zum DHCPREQUEST gibt es auch noch DHCPINFORM Pakete, mit denen ein Client, der schon eine (statisch konfigurierte) IP-Adresse hat, weitere Konfigurationsparameter vom DHCP-Server anfordern kann.

DHCP sieht auch vor, daß ein sog. DHCP-Relay Anfragen in ein anderes Netz weitergeben kann (d.h., der DHCP-Server kann sich dann auch in einem anderen Netz befinden).

5.7 Automatische IP-Konfiguration ...

Ablauf



Aufbau eines DHCP-Pakets (siehe auch Wireshark-Aufzeichung)

- Felder u.a. für:
  - Pakettyp (*request*, *reply*)
  - angefragte Client-IP-Adresse
    - im DHCPREQUEST
  - angebotene Client-IP-Adresse
    - in DHCPOFFER und DHCPACK
  - Hardware-Adresse des Clients
    - i.a. MAC-Adresse
  - Name einer Boot-Datei
    - wird vom Server an Client per SFTP übertragen
  - Optionen
    - z.B. Anfrage / Zuweisung von Subnetzmaske, Hostname, Default-Gateway, DNS-Server, NTP-Server

★

Zustandslose IPv6 Autokonfiguration

- Vorgehensweise eines Hosts:
  1. bilde *link-local* Adresse
     - Präfix FE80::/10 und z.B. EUI-64 oder Zufallszahl
  2. prüfe Adresse mit NDP: *Neighbor Solicitation*
     - falls keine Antwort: Adresse im LAN eindeutig
  3. warte auf *Router Advertisement*
     - sende ggf. explizit *Router Solicitation* wg. Wartezeit
  4. *Router Advertisement* teilt mit:
     - Adresse und/oder andere Optionen über DHCP holen?
     - ggf. globales Routing-Präfix und Präfixlänge
  5. bilde Adresse aus Präfix und z.B. EUI-64 oder Zufallszahl
  6. prüfe Adresse mit NDP: *Neighbor Solicitation*
- Adresse ist i.d.R. nur begrenzte Zeit gültig

Anmerkungen zu Folie 214:

- Die Vorgehensweise bei der zustandslosen IPv6 Autokonfiguration (Stateless Address Autoconfiguration, SLAAC) ist in RFC 4862, Abschnitt 4 gut beschrieben.
- Bei der Prüfung auf Eindeutigkeit der gewählten Adresse wird das *Neighbor Solicitation* Paket an die gewählte Adresse gesendet. Falls es einen anderen Host mit dieser Adresse gibt, wird er mit seiner MAC-Adresse in einem *Neighbor Advertisement* Paket antworten. Zusätzlich muß der Host auch prüfen, ob er ein *Neighbor Solicitation* Paket an die von ihm gewählte Adresse empfängt. Das kann passieren, wenn ein anderer Host gleichzeitig dieselbe Adresse konfiguriert.
- Falls im lokalen Netz kein Router vorhanden ist, wird die Autokonfiguration mit Schritt 3 beendet. Dann hat der Host (genauer: das zu betreffende Interface) nur eine *link-local* Adresse, was in diesem Fall aber ausreicht.
- Der Schritt 5 findet nur wie angegeben statt, wenn das *Router Advertisement* mitteilt, daß kein DHCP benutzt werden soll. Ansonsten wird statt oder zusätzlich zu Schritt 5 noch eine DHCP-Konfiguration wie auf S. 211 durchgeführt.

214-1

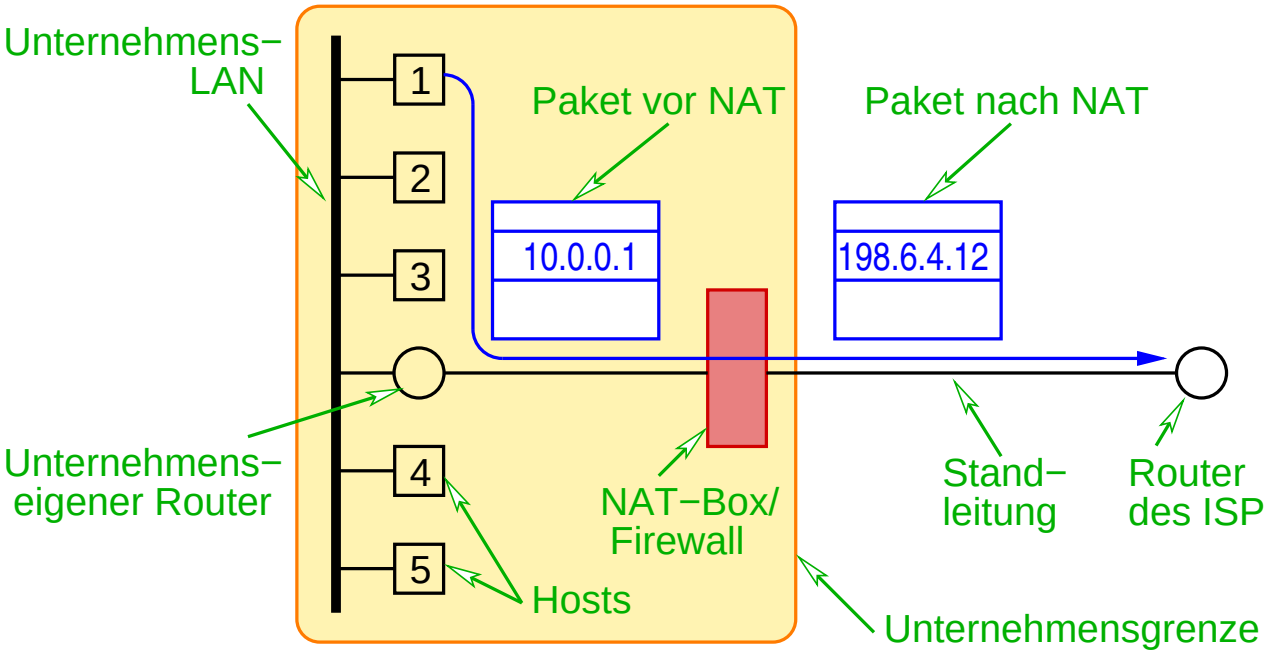
5.8 *Network Address Translation*

★★

NAT, speziell NAPT (*Network Address Port Translation*)

- Ziel: Einsparung von IP-Adressen
  - mehrere Rechner eines Netzes werden auf dieselbe, nach außen sichtbare IP-Adresse gemultiplext
- Prinzip:
  - jeder Rechner des Netzes bekommt **private** IP-Adresse
  - drei reservierte Adreßbereiche:
    - 10.0.0.0/8 (10.0.0.0 - 10.255.255.255)
    - 172.16.0.0/12 (172.16.0.0 - 172.31.255.255)
    - 192.168.0.0/16 (192.168.0.0 - 192.168.255.255)
  - das gesamte Netz erhält **eine** „echte“ IP-Adresse
  - ausgehende Pakete durchqueren NAT-Box
    - NAT-Box ersetzt private IP-Adresse durch echte

Prinzipielle Funktionsweise von NAT



★★

Problem: Zustellung von Antwort-Paketen!?

- Alle eingehende Pakete sind an dieselbe Adresse gerichtet
  - woher weiß die NAT-Box, an wen das Paket gehen soll?
- Lösung (schmutzig!):
  - praktisch der gesamte IP-Verkehr ist UDP oder TCP
  - UDP- und TCP-Header enthalten Felder zur Adressierung von Quell- und Zielprozess (Quell-/Ziel-Port)
  - NAT-Box verwendet Quell-Port-Feld des UDP/TCP-Headers zum Speichern eines Demultiplex-Schlüssels
    - bei ausgehendem Paket: speichere Quell-IP-Adresse und Quell-Port (⇒ Schlüssel)
    - ersetze IP-Adresse, ersetze Quell-Port durch Schlüssel
    - Antwortpaket enthält Schlüssel als Ziel-Port
    - ersetze Ziel-Adresse/Port durch gespeicherte Adresse/Port

Anmerkungen zu Folie 217:

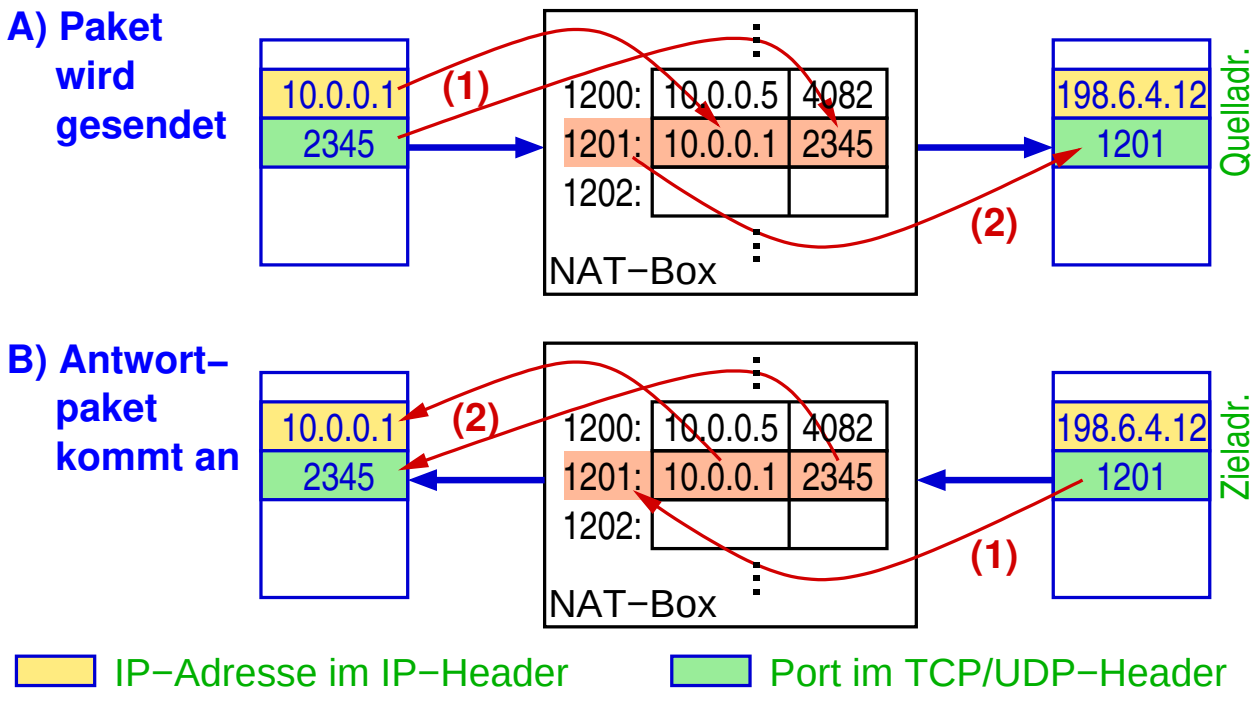
Was in diesem Abschnitt beschrieben (und in den meisten NAT-Boxen eingesetzt) wird, ist NA**P**T (*Network Address **Port** Translation*). Dabei wird ein ganzes Netz auf eine einzige IP-Adresse abgebildet.

Prinzipiell kann auch reines NAT genutzt werden, wobei dem Netz dann aber eine **Menge** von öffentlichen IP-Adressen zugewiesen wird. Die NAT-Box ersetzt dann die private IP-Adresse in ausgehenden Paketen durch eine dieser öffentlichen Adressen. Antwort-Pakete können dann zugestellt werden, wenn im lokalen Netz (in einem Zeitraum) höchstens soviele Rechner aktiv sind, wie öffentliche IP-Adressen vorhanden sind. Die NAT-Box hält dazu einen Cache mit den Zuordnungen zwischen privater und öffentlicher IP-Adresse mit einer begrenzten Zahl an Einträgen, wobei jeder Eintrag nach einer gewissen Zeit der Inaktivität gelöscht wird. Die öffentliche IP-Adresse wird damit wieder frei für andere Rechner des Netzes.

5.8 *Network Address Translation ...*

★★

Genaue Funktionsweise von NAT



5.8 *Network Address Translation ...*

Kritik:

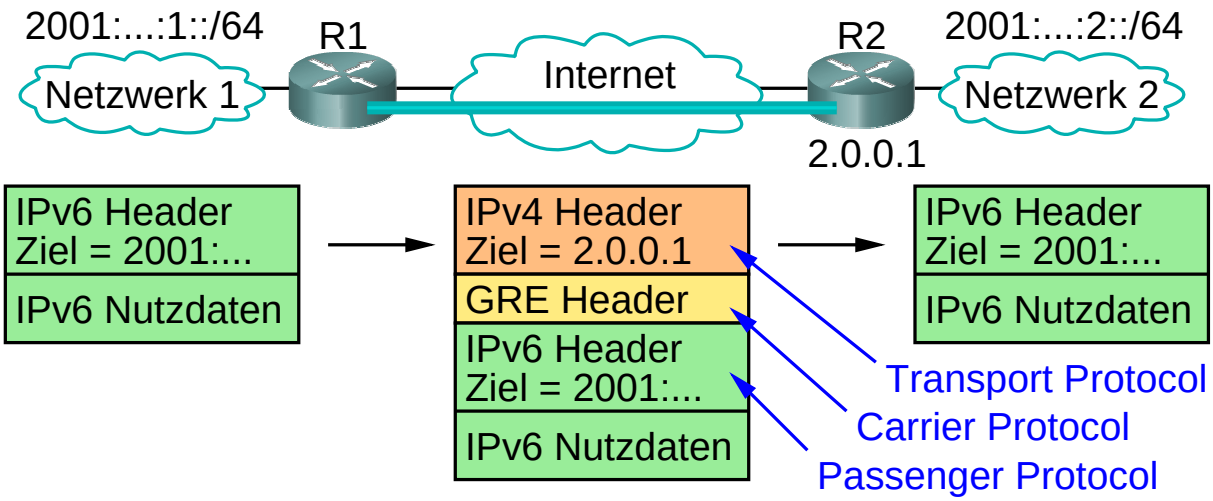
- IP-Adressen nicht mehr weltweit eindeutig
- Macht aus Internet quasi verbindungsorientiertes Netz
  - Ausfall der NAT-Box: Zerstörung aller Kommunikationsbeziehungen (TCP und UDP)!
- NAT (Schicht 3) macht Annahmen über Schicht 4!
  - neue TCP/UDP-Version, andere Protokolle über IP !?
- Nutzdaten können IP-Adressen enthalten (z.B. FTP, H.323)

Vorteile:

- Einsparung von IP-Adressen
- Sicherheit: Hosts sind von außen nicht erreichbar, Struktur des lokalen Netzes nach außen verborgen

5.9 Tunneling

- Übertragung eines Protokolls über ein Protokoll derselben oder einer höheren OSI-Schicht
- z.B. Übertragung von IPv6-Paketen über IPv4



5.9 Tunneling ...

- Aufgaben des *Carrier*-Protokolls:
  - z.B. Multiplexing, Reihenfolgeerhaltung, Authentifizierung, ...
  - ggf. kann das Carrier-Protokoll auch fehlen
- Einsatz von Tunneln:
  - Kommunikation zwischen Routern mit speziellen Fähigkeiten
    - z.B. Multicast, IPv6
  - Kopplung von nicht-IP Netzen (z.B. Ethernet) über das Internet
  - VPNs: Authentifizierung und Verschlüsselung im Tunnel
  - „Durchtunneln“ von Firewalls
    - oft HTTP als Transportprotokoll

Anmerkungen zu Folie 221:

- GRE (*Generic Routing Encapsulation*) ist ein einfaches, unsicheres Tunnel-Protokoll, das von CISCO entwickelt wurde.
- Bei dem von Microsoft verwendeten Teredo-Tunneling zur Übertragung von IPv6-Paketen über IPv4-Netze übernimmt UDP die Aufgabe des Carrier-Protokolls.

  Die Verwendung von UDP ermöglicht dabei die Übertragung der Pakete durch NAT-Boxen hindurch.

5.10 Übergang von IPv4 auf IPv6

Verschiedene Ansätze

- Dual-Stack-Ansatz
  - Netzknoten implementieren sowohl IPv6 als auch IPv4
  - Fähigkeiten des Zielknotens über DNS zu ermitteln
    - IPv6-Adresse für IPv6-fähigen Knoten, IPv4-Adresse sonst
  - Router müssen ggf. IP-Header der Pakete anpassen
    - wenn zwischenliegende Router nur IPv4 unterstützen
    - Informationsverlust möglich (z.B. FlowLabel nur in IPv6)
- Tunneling
  - Aufbau eines IPv4-Tunnels zwischen IPv6-Knoten
  - d.h. IPv6-Pakete werden in IPv4-Pakete eingepackt
- Adreßübersetzung analog zu NAT (6to4)

5.10 Übergang von IPv4 auf IPv6 ...

Stand

- Praktisch alle neuen Features von IPv6 inzwischen auch in IPv4 verfügbar
  - Hauptmotivation: größerer Adreßbereich
- Moderne Betriebssysteme beinhalten IPv4 und IPv6
  - z.B. Linux, Windows (ab XP)
- Bis Juni 2006: Internet-Testbett 6bone
- IPv6 inzwischen im „Produktionseinsatz“
  - einige Dienste nur noch mit IPv6 angeboten
- Zeitrahmen für die Ablösung von IPv4 aber unklar

5.11 Zusammenfassung

- Internetwork: Netz von Netzen
- IP-Protokoll: Best-Effort, run over everything
  - Hierarchische Adressen: Routing nur zwischen Netzen
  - Bessere Skalierbarkeit durch CIDR und Subnetze
- Hilfsprotokolle: ICMP, ARP, NDP, DHCP
- NAT: Umsetzung privater auf globale IP-Aressen

Nächste Lektion:

- Routing

Rechnernetze I

SoSe 2024

6 Routing

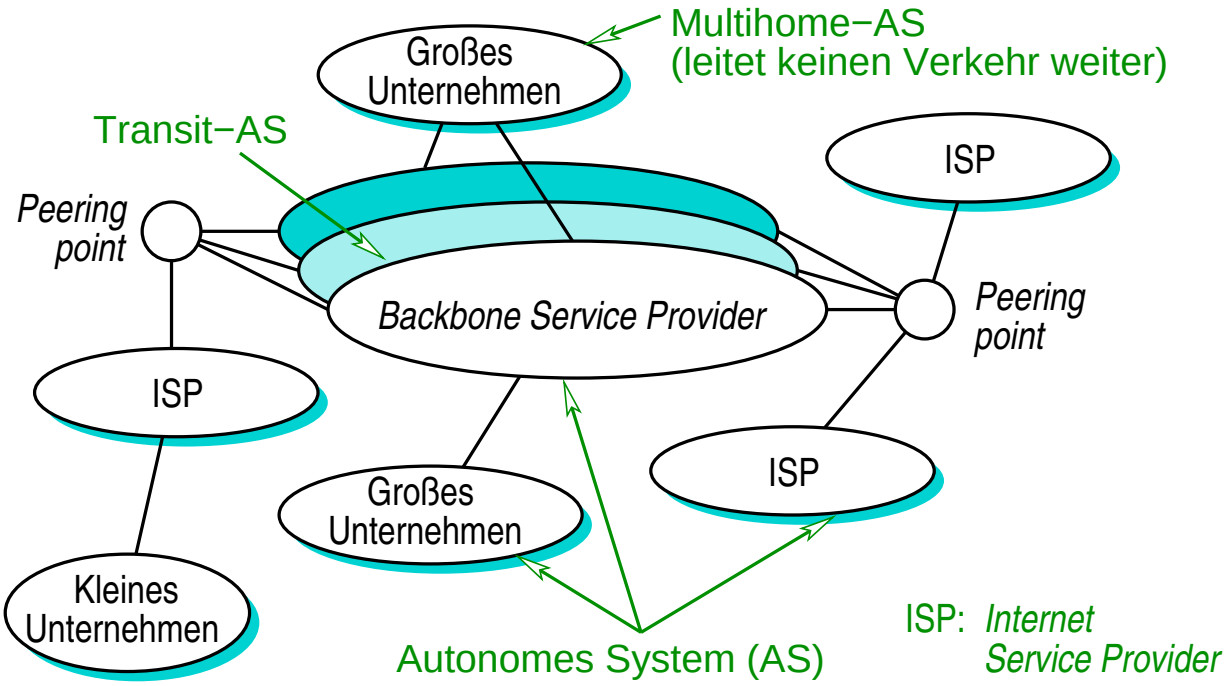
Inhalt

- Einführung
- Routing innerhalb einer Domain
  - *Distance Vector Routing* (RIP, EIGRP)
  - *Link State Routing* (OSPF)
- Interdomain-Routing
  - *Border Gateway Protocol* (BGP)

- Peterson, Kap. 4.2.1 – 4.2.4

6.1 Einführung

Heutige Struktur des Internet



★

Routing im Internet

- Heute über 92.000 autonome Systeme (AS)
  - Backbones, Provider, Endbenutzer
  - Netzwerk-Adressen werden an AS zugewiesen
- Probleme: Skalierbarkeit, heterogene Administration
- Hierarchischer Ansatz: ***Routing Domains***
  - Routing innerhalb eines administrativen Bereichs (z.B. Campus, Unternehmen, Provider)
    - *Interior Gateway Protocols* (IGP)
      - z.B. RIP, OSPF, EIGRP
  - Interdomain Routing (zwischen Teilnetzen des Internet)
    - *Exterior Gateway Protocols* (EGP)
      - z.B. BGP (*Border Gateway Protocol*)

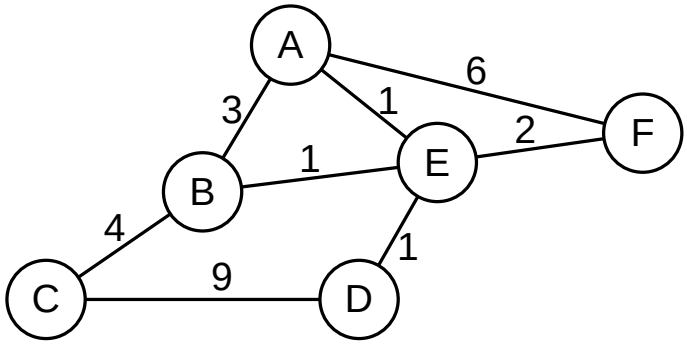
6.2 Routing innerhalb eines Bereichs

OSI: 3

★★

Routing als Graph-Problem

- Knoten:
  - Router
  - (Hosts)
  - (Netzwerke)
- Kanten: Verbindungen
  - mit Kosten (Metrik)
    (symmetrisch oder asymmetrisch)



- Aufgabe des Routings:
  - finde Pfade mit geringsten Kosten zwischen allen Paaren von Knoten

Anmerkungen zu Folie 229:

- Im Graphen werden üblicherweise nur die Router mit den Verbindungen (d.h. Netzen) zwischen ihnen modelliert. An vielen Router sind i.d.R. natürlich noch ein oder mehrere lokale Netze angeschlossen, die im Graphen aber nicht dargestellt sind.
- Zur Bestimmung der Kosten einer Verbindung können typischerweise die Bandbreite, die Latenz und die Auslastung der Verbindung herangezogen werden. Man hat durch Erfahrung gelernt, daß der Wertebereich dabei relativ eng sein muß, damit der billigste Link nicht völlig überlastet wird. Die Bandbreite ist dabei wichtiger als die Latenz, die Auslastung sollte nur bei mittlerer bis hoher Last in die Kosten eingehen.

  Im einfachsten Fall können die Kosten auch für jede Verbindung gleich gewählt werden (i.d.R. als 1).

229-1

6.2 Routing innerhalb eines Bereichs ...

Statisches Routing

- Pfade werden manuell bestimmt und in Tabellen eingetragen
- Probleme:
  - Ausfall von Knoten / Verbindungen
  - neue Knoten / Verbindungen
  - dynamische Änderung der Verbindungskosten (Last)

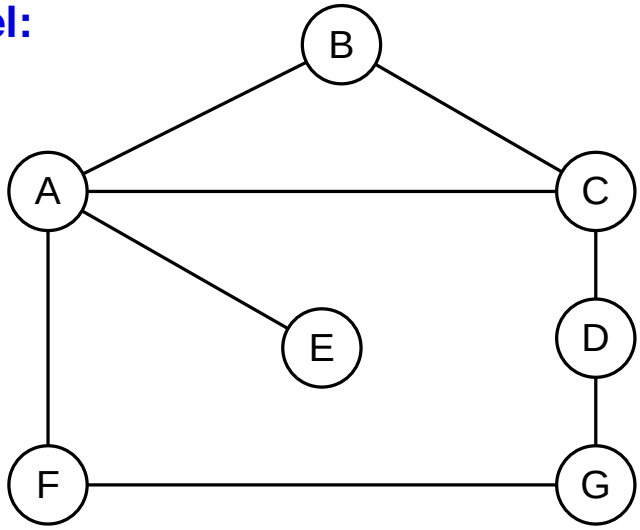
Dynamisches Routing

- Verteilte Algorithmen zum Aufbau der Tabellen
- Anforderungen:
  - schnelle Konvergenz / Skalierbarkeit
  - einfache Administration

6.2.1 *Distance Vector Routing*

- Router kennen nur Distanz und „Richtung“ (*Next Hop*) zum Ziel
  - nur diese Informationen werden (mit Nachbarn) ausgetauscht
- Verteilter Algorithmus zur Erstellung der Routing-Tabellen
  - typisch: Bellman-Ford-Algorithmus

Beispiel:

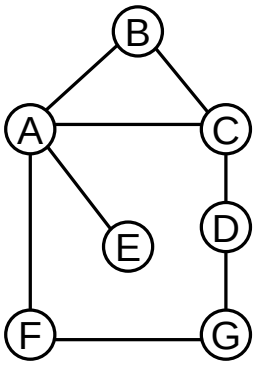


Vereinfachung:
alle Link−Kosten
seien 1

6.2.1 *Distance Vector Routing ...*

Beispiel: ...

- Initial besitzen die Router Information über die folgenden Distanzvektoren (verteilt!)



	Distanz zu Knoten						
Information bei	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

6.2.1 *Distance Vector Routing ...*

★

Beispiel: ...

- Initiale Routing-Tabelle von A:
  - A kennt nur seine direkten Nachbarn

Ziel	Kosten	NextHop
B	1	B
C	1	C
E	1	E
F	1	F

6.2.1 *Distance Vector Routing ...*

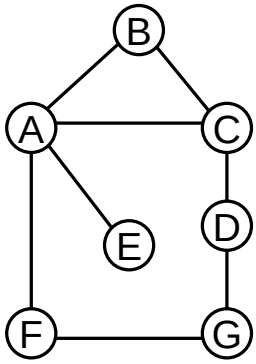
★★★

Vorgehensweise

- Nachrichtenaustausch zur Erstellung der Routing-Tabellen:
  - Router senden ihren Distanzvektor an alle direkten Nachbarn
  - bessere Routen werden in den eigenen Distanzvektor (und die Routing-Tabelle) übernommen
    - aber auch schlechtere vom Next Hop
- Nach mehreren Runden des Nachrichtenaustauschs konvergieren die Distanzvektoren ... jedenfalls meistens !!
- Der Nachrichtenaustausch erfolgt
  - periodisch (typ. alle 30 s)
  - bei Änderung des Distanzvektors eines Knotens
    - z.B. Ausfall einer Verbindung, neue Verbindung
  - auf Anfrage (z.B. bei Neustart eines Routers)

Beispiel ...

- Distanzvektoren nach dem Konvergieren des Verfahrens:

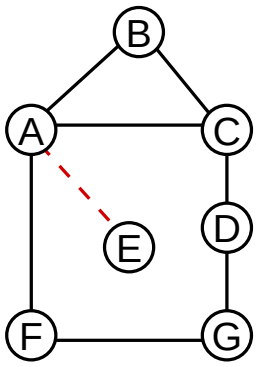


Information bei	Distanz zu Knoten						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

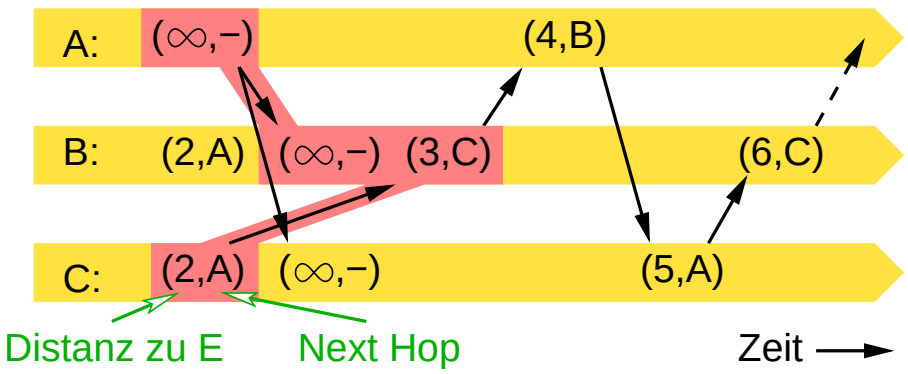
(Animierte Folie) ★

Problem des Algorithmus: *Count-to-Infinity*

- Beispiel: A erkennt Ausfall der Verbindung zu E



Zeitliche Entwicklung der Distanz−Vektoren zu E:



- Keine wirklich gute, allgemeine Lösung des Problems
  - pragmatisch: beschränke ∞ auf den Wert 16

Anmerkungen zu Folie 236:

- Neben dem *Count-to-Infinity*-Problem können beim Distanzvektor-Routing auch Routing-Schleifen entstehen. Im Beispiel entsteht während des Hochzählens eine Routing-Schleife (C→A→B→C), in der Pakete kreisen, bis sie wegen TTL=0 verworfen werden.
- Weitere Maßnahmen, um die genannten Probleme zu verhindern, sind:
  - *Split Horizon*: Information, die über eine bestimmte Schnittstelle gelernt wurde, wird nicht wieder an dieselbe Schnittstelle weitergegeben. Dies kann Routing-Schleifen verhindern, sofern das Netzwerk zyklenfrei ist (was im Beispiel nicht der Fall ist).
  - *Route Poisoning*: Nicht mehr verfügbare Routen werden im Distanzvektor nicht gelöscht, sondern mit einer Metrik von 16 (d.h. ∞) weitergegeben. Im Beispiel auf der Folie wird dies bereits so gemacht, kann in diesem Fall das Problem aber nicht verhindern.
  - *Holddown*-Timer: wenn eine Route ausfällt, wird diese zunächst nur markiert. Bis der Timer abläuft (typ. 180s), werden empfangene Aktualisierungen für die Route ignoriert, die die gleiche oder eine schlechtere Metrik haben wie die Metrik vor dem Ausfall. Im Beispiel würde B die falsche Aktualisierung von C ignorieren, wenn sie vor Ablauf des Timers eintrifft.

236-1

6.2.1 *Distance Vector Routing ...*

★

RIP (*Routing Information Protocol*)

- Einfaches *Distance Vector Routing* Protokoll
- Internet-Standard
- Alle Link-Kosten sind 1, d.h. Distanz = *Hop Count*
- Drei Versionen:
  - RIPv1: leitet keine Präfixlänge weiter
    - Subnetting nur möglich, wenn alle Subnetze des klassenbehafteten Netzes dieselbe Größe haben
    - d.h. Subnetzmaske ist global
  - RIPv2: ermöglicht klassenloses Routing
  - RIPng: unterstützt IPv6

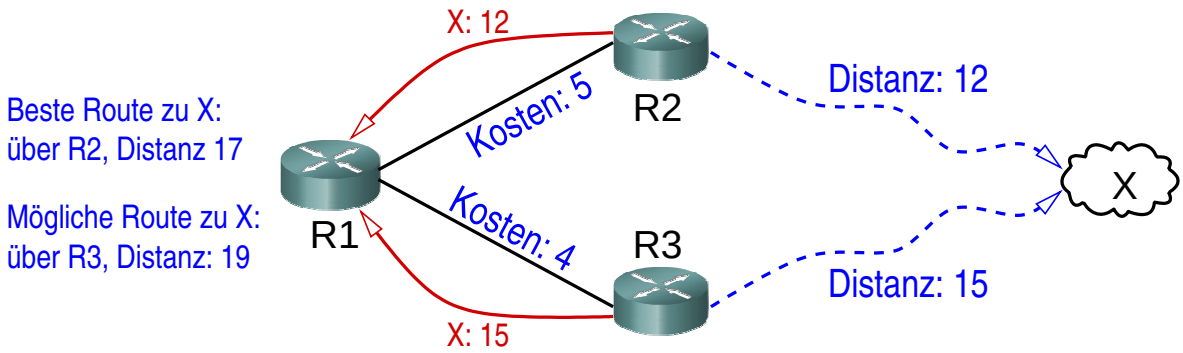
Anmerkungen zu Folie 237:

- RIP-Pakete werden mittels UDP übertragen (in RIPv1 als Broadcast, in späteren Versionen als Multicast).
- RIPv1 ist im Detail im RFC 1058 spezifiziert, RIPv2 im RFC 2453, RIPng im RFC 2080.

6.2.1 *Distance Vector Routing ...*

EIGRP (*Enhanced Interior Gateway Routing Protocol*)

- Erweitertes *Distance Vector Routing* Protokoll
- Cisco-proprietär, seit 2013 offener Internet-Standard
- Link-Kosten berücksichtigen Bandbreite und Latenz
- Unterstützt IPv4, IPv6 und andere Schicht-3-Protokolle
- Updates nur bei Änderungen, kein *Count-to-Infinity*
- Behält alle Routen, nicht nur die beste



Anmerkungen zu Folie 238:

- Im Gegensatz zu RIP behält EIGRP von den eingehenden Distanzvektoren nicht nur den besten, der zum kürzesten Weg zum Ziel gehört, sonden alle, die zu garantiert zyklenfreien Wegen zum Ziel gehören. Um die Zyklenfreiheit zu garantieren, vergleicht EIGRP beim Empfang eines Distanzvektor in einem Router R die darin gemeldete Distanz des Senders S zum Ziel mit der (aktuell bekannten) eigenen Distanz zum Ziel. Hat S eine kleinere Distanz, kann sein bester Weg nicht über R gehen, da die Link-Kosten nicht negativ werden können.

  Wenn im gezeichneten Beispiel R1 den Distanzvektor von R2 erhält (rote Nachricht mit Beschriftung „X: 12“), trägt er als beste Route zu X die über R2 ein, mit einer Distanz von 12+5=17. Beim Erhalt des Distanzvektors von R3 („X: 15“) stellt R1 fest, daß die von R3 gemeldete Distanz 15 kleiner als seine eigene (17) ist, und der Weg von R3 nach X daher nicht über R1 führen kann. R3 ist damit ein möglicher Nachfolger (*feasible successor*) von R1 auf dem Weg zu X. Bei Ausfall der Verbindung zwischen R1 und R2 kann R1 daher die Route in der Weiterleitungstabelle sofort so abändern, daß R3 der *Next Hop* wird. Er muß dazu nicht auf Routing-Nachrichten anderer Route warten.

- Um den Ausfall von Verbindungen zu erkennen, sendet EIGRP regelmäßige *Hello*-Pakete an alle Nachbarn.

- EIGRP-Pakete werden über ein eigenes, zuverlässiges Transportprotokoll (*Reliable Transport Protocol*, RTP) übertragen, das auf IP aufsetzt.

- Den *Informational* RFC zu EIGRP finden Sie unter der URL `https://tools.ietf.org/html/draft-savage-eigrp-00`.

6.2.2 *Link State Routing*

★★★

- Grund„problem“ beim *Distance Vector Routing*:
  - Router haben ausschließlich lokale Information
- ***Link State Routing:***
  - Router erhalten Information über die Struktur des gesamten Netzwerks
- Vorgehensweise:
  - Kennenlernen der direkten Nachbarn
    - einschließlich der Link-Kosten
  - Versenden von Link State Paketen an **alle** anderen Router (***Reliable Flooding***)
  - Berechnung der kürzesten Wege mit Dijkstra-Algorithmus

6.2.2 *Link State Routing* ...

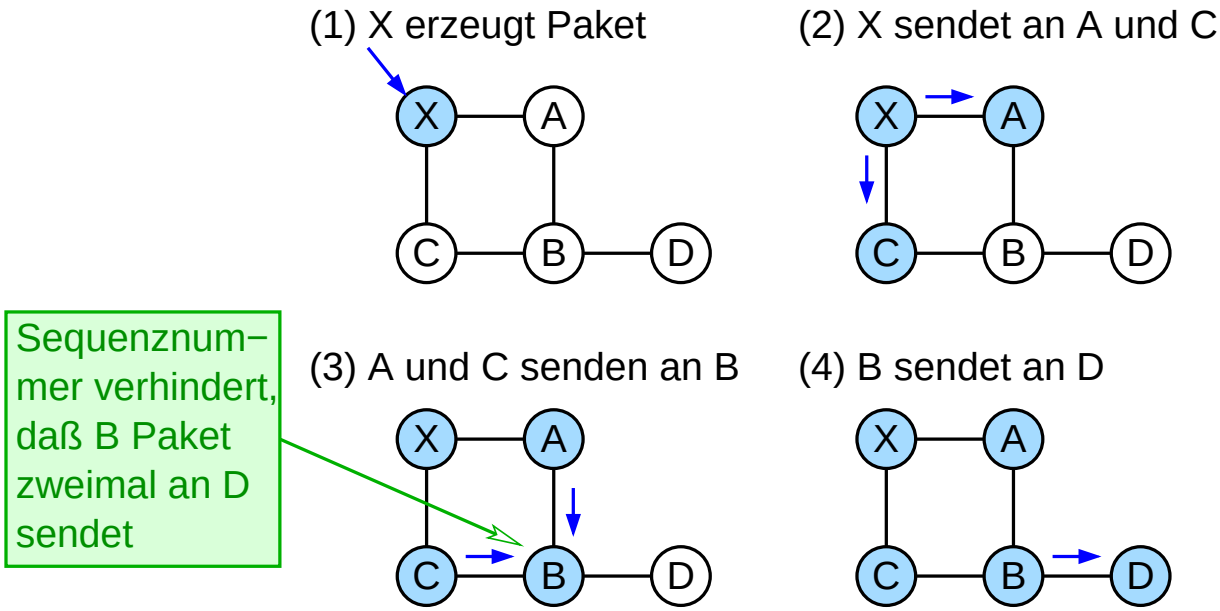
★

Link State Pakete

- Inhalt eines Link State Pakets:
  - ID des erzeugenden Routers
  - Liste der direkten Nachbarn mit Link-Kosten
  - Sequenznummer
    - Paket nur weitergeleitet, wenn die Sequenznummer größer als die des letzten weitergeleiteten Pakets ist
  - *Time-to-Live* (TTL)
    - jeder Router dekrementiert TTL
    - bei TTL = 0 wird das Paket gelöscht
- Versenden der Link State Pakete
  - Periodisch ($\sim$ Stunden) oder bei Topologie-Änderungen

Reliable Flooding

- Flooding mit ACK und ggf. wiederholter Übertragung



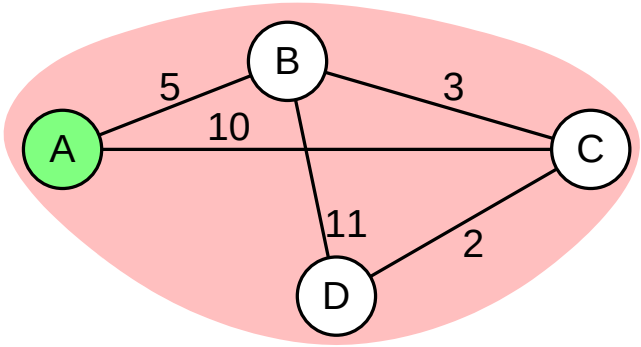
Dijkstra-Algorithmus zur Bestimmung kürzester Wege

- Eingabe: N: Knotenmenge, $l(i,j)$: Link-Kosten, s: Startknoten
- Ausgabe: $C(n)$: Pfad-Kosten von s zu n
- Algorithmus:

$M = \{s\}$
Für alle $n \in N - \{s\}$: $C(n) = l(s,n)$
Solange $M \neq N$:
  Wähle $w \in N - M$ so, daß $C(w)$ minimal ist
  $M = M \cup \{w\}$
  Für alle $n \in N - M$:
    $C(n) = \min(\, C(n),\, C(w) + l(w,n)\,)$

(Animierte Folie)

Beispiel



Schritt	w	M	C(B)	C(C)	C(D)
Initial		{A}	5	10	∞
1	B	{A,B}	5	8	16
2	C	{A,B,C}	5	8	10
3	D	{A,B,C,D}	5	8	10

Anmerkungen zu Folie 243:

Um neben der kürzesten Entfernung auch den nächsten Knoten auf dem Weg zum Ziel (*Next Hop*) zu erhalten, kann der Dijkstra-Algorithmus einfach erweitert werden. Die notwendige Erweiterung ist hier nicht dargestellt, sondern bleibt Ihnen als Übung überlassen.

6.2.2 *Link State Routing* ...

OSPF (*Open Shortest Path First*)

- Weit verbreitetes *Link State Routing* Protokoll
- Internet-Standard
- Link-Kosten vom Protokoll nicht festgelegt
  - in der Praxis meist Link-Bandbreite verwendet
- Versionen:
  - OSPFv2: für IPv4
  - OSPFv3: unterstützt IPv6

6.2.2 *Link State Routing* ...

OSPF (*Open Shortest Path First*) ...

- Wichtige Pakettypen:
  - *Hello*-Pakete: bauen Nachbarschaftsbeziehungen (Adjazenzen) auf und testen diese
  - *Link State Update*-Pakete enthalten *Link State Advertisements* (LSAs)
    - LSA beschreibt Verbindung zwischen zwei Routern
- In Mehrfachzugriffsnetzen:
  - Router wählen einen „designierten“ Router (DR) und einen Backup DR (BDR)
    - bei Ausfall DR: BDR wird neuer DR, Neuwahl BDR
  - jeder Router hat (nur) DR als Nachbarn
    - LSAs werden nur an DR gesendet und von diesem verteilt
    - verhindert exzessives Flooding im Netz

Anmerkungen zu Folie 245:

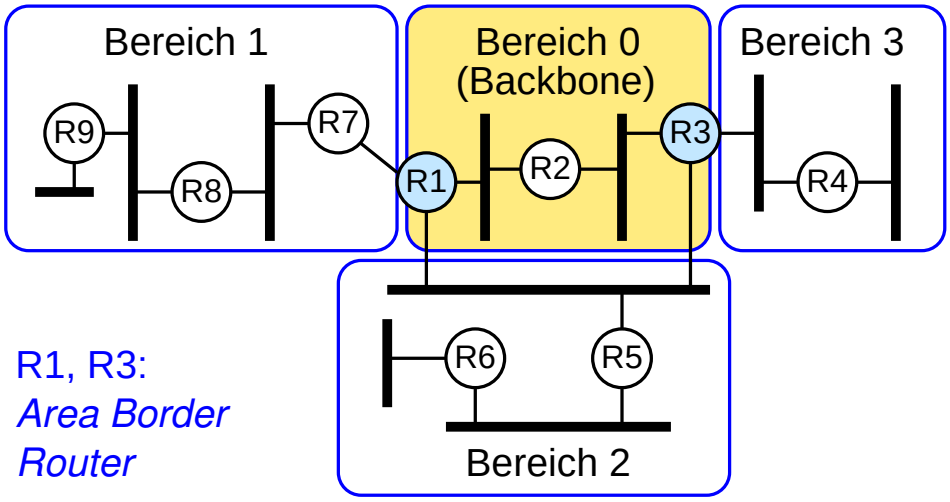
- OSPF besitzt fünf verschiedene Pakettypen, die in ein IP-Paket gekapselt und per Multicast versendet werden:
  1. *Hello*-Pakete dienen dazu, die Verbindung zu benachbarten Routern zu testen und in Mehrfachzugriffsnetzen den DR/BDR zu wählen.
  2. *Database Description*-Pakete werden verwendet, um den OSPF-Datenbestand eines Routers mit seinem Nachbarn zu synchronisieren, wenn die Verbindung zu diesem Nachbarn hergestellt wird.
  3. *Link State Request*-Pakete dienen zur expliziten Anforderung von Link State Information.
  4. *Link State Update*-Pakete enthalten ein oder mehrere *Link State Advertisements* (LSA). Ein LSA beschreibt dabei eine Verbindung zwischen zwei Routern.
  5. *Link State Acknowledgements* dienen zur Bestätigung beim *Reliable Flooding*.
- OSPFv2 ist im Detail im RFC 2328 spezifiziert, OSPFv3 im RFC 5340.

245-1

6.2.2 *Link State Routing* ...

Multi-Area OSPF

- Für große *Routing-Domains*:
  - OSPF erlaubt Einführung einer weiteren Hierarchieebene: **Routing-Bereiche** (***Areas***)
- Beispiel:



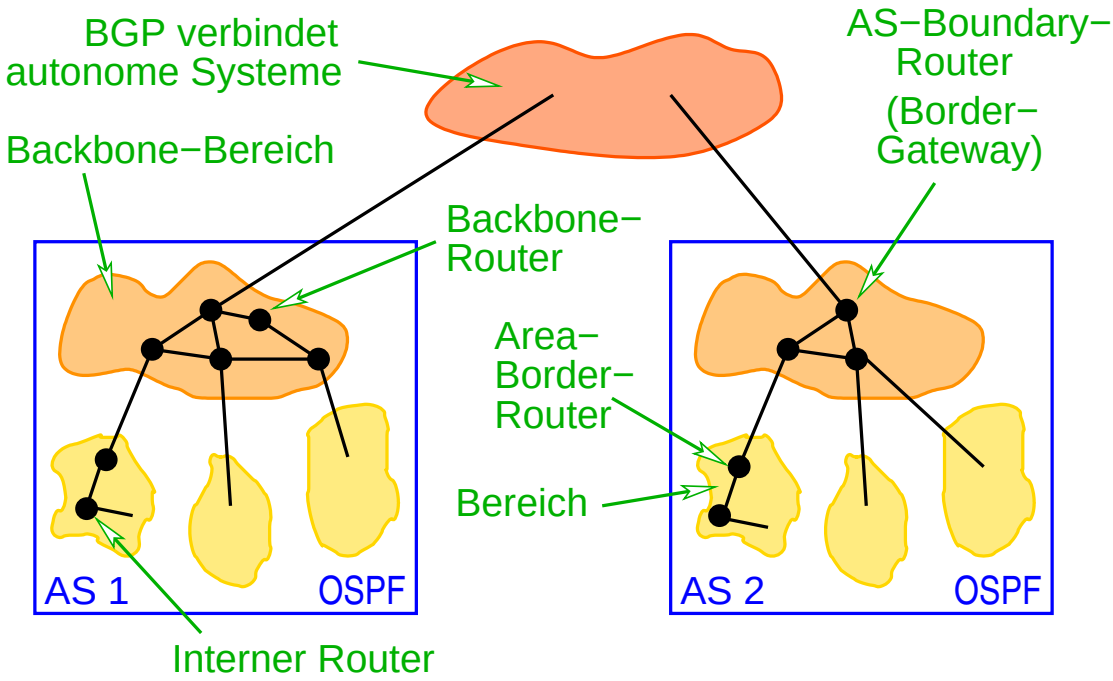
Multi-Area OSPF ...

- Innerhalb jedes Bereichs: *Flooding* von Link-State-Paketen
- *Area Border Router* (ABR) geben diese nicht weiter
- Stattdessen: ABR sendet zusammengefaßte Information
  - nur ein Link-State-Paket für den gesamten Bereich
  - ABR spiegelt vor, daß alle Hosts in seinem Bereich **direkt** mit ihm verbunden sind
- Pakete zwischen Bereichen immer über ABR geleitet
  - bei mehreren ABR: automatische Auswahl über die Link-Kosten
- Damit: bessere Skalierbarkeit
  - kleinere Graphen in den einzelnen Routing-Bereichen
  - weniger Neuberechnungen der kürzesten Wege
  - evtl. aber suboptimale Routen

Anmerkungen zu Folie 247:

- OSPF verwendet fünf verschiedene Arten von Link State Advertisements:
  - Typ 1: enthält Information zu allen Router-Schnittstellen eines Bereichs, wird nur innerhalb des Bereichs weitergegeben.
  - Typ 2: wird vom DR an alle Router eines Mehrfachzugriffsnetzes gesendet, Weitergabe nur innerhalb eines Bereichs.
  - Typ 3: Zusammengefasste Information zur Erreichbarkeit von Netzen aus anderem Bereich.
  - Typ 4: Information zur Erreichbarkeit des AS Boundary Routers.
  - Typ 5: Information zur Erreichbarkeit von Netzen ausserhalb des AS (vom AS Boundary Router erzeugt).
- Suboptimale Routen entstehen zum einen durch die von Multi-Area OSPF erzwungene Topologie mit einem zentralen Backbone-Bereich, zum anderen durch die Implementierung des OSPF Protokolls. Details dazu finden Sie im Internet-Draft `http://tools.ietf.org/html/draft-ietf-ospf-shortcut-abr-02`

Routing-Hierarchie mit Multi-Area OSPF



6.3 Interdomain Routing

★★

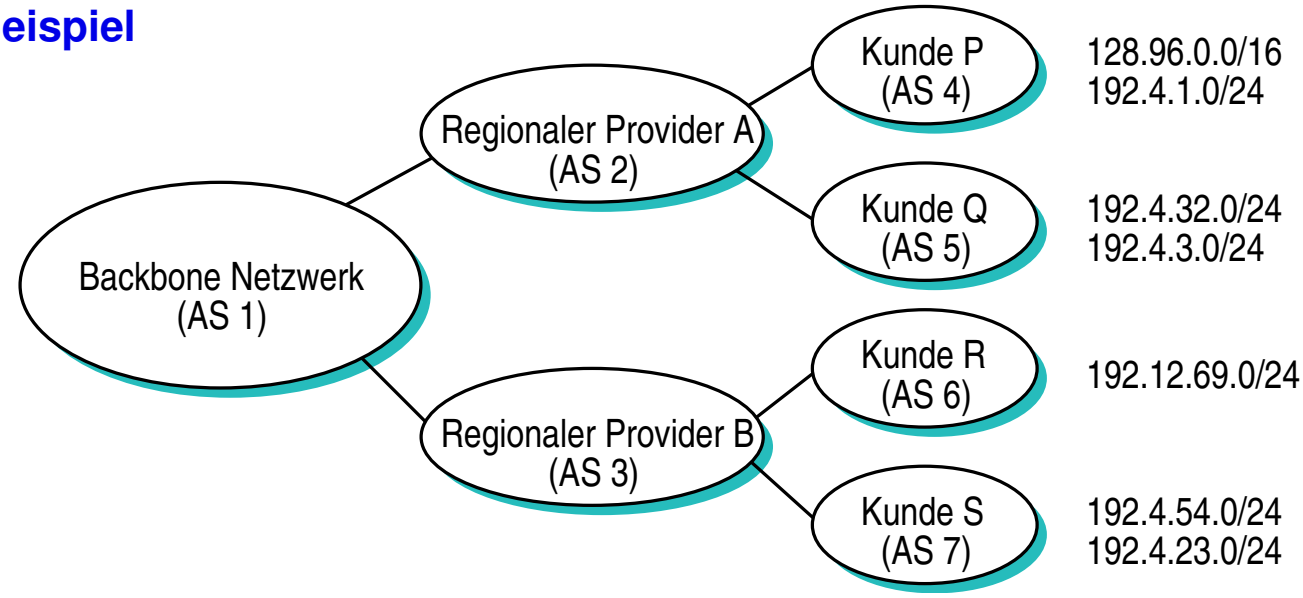
Routing innerhalb und außerhalb von Domains

- Innerhalb einer Domain: RIP, OSPF
  - Bestimmung optimaler Routen
- Zwischen Domains: ***Border Gateway Protocol* (BGP)**
  - Autonome Systeme ⇒ keine gemeinsame Metrik
  - Routen werden „politisch“ bestimmt
    - „benutze Provider B für Adressen xyz“
    - „benutze Pfad über möglichst wenige AS“
  - Wichtigstes Ziel: Erreichbarkeit
    - „gute“ Routen sind sekundär

Routing mit BGP

- Jedes Autonome System (AS) hat
  - ein oder mehrere *Border Router*
    - Verbindung zu anderen AS
  - einen BGP Sprecher, der bekanntgibt:
    - lokales Netzwerk
    - über dieses AS erreichbare Netzwerke
      (nur, wenn das AS Pakete an andere AS weiterleitet)
- Bekanntgegeben werden vollständige Pfade
  - zur Vermeidung von zyklischen Routen

Beispiel



- AS 2 gibt (u.a.) bekannt: „ich kann AS 4, AS 5 direkt erreichen“
  - Netze 128.96.0.0/16, 192.4.1.0/24, 192.4.32.0/24, 192.4.3.0/24
- AS 1 gibt (u.a.) bekannt: „ich kann AS 4, AS 5 über AS 2 erreichen“

Anmerkungen zu Folie 251:

Etwas exakter:

- der BGP-Sprecher von AS2 gibt u.a. bekannt:
  - „über den Pfad AS2, AS4 kann ich folgende Netze erreichen: 128.96.0.0/16, 192.4.1.0/24“
  - „über den Pfad AS2, AS5 kann ich folgende Netze erreichen: 192.4.32.0/24, 192.4.3.0/24“
- der BGP-Sprecher von AS1 gibt u.a. bekannt:
  - „über den Pfad AS1, AS2, AS4 kann ich folgende Netze erreichen: 128.96.0.0/16, 192.4.1.0/24“
  - „über den Pfad AS1, AS2, AS5 kann ich folgende Netze erreichen: 192.4.32.0/24, 192.4.3.0/24“

Das bekanntgebende AS ist also immer Teil des Pfades.

251-1

6.4 Zusammenfassung

- Routing „im kleinen“ (innerhalb einer Domain):
  - Suche optimale Pfade
  - *Distance Vector Routing* (nur mit lokaler Information)
  - *Link State Routing* (globale Information, *reliable Flooding*)
- Routing „im großen“ (zwischen Domains)
  - *Border Gateway Protocol*
    - Bekanntgabe der Erreichbarkeit
    - Routenwahl ist „politische“ Entscheidung

Nächste Lektion:

- Ende-zu-Ende Prokotolle: UDP, TCP

Rechnernetze I

SoSe 2024

7 Ende-zu-Ende Protokolle

7 Ende-zu-Ende Protokolle ... OSI: 4

Inhalt

- Ports: Adressierung von Prozessen
- UDP
- TCP (Bytestrom, Paketformat)
- Sicherung der Übertragung
- Übertragungssicherung in TCP
- Überlastkontrolle
- TCP Verbindungsauf- und abbau

- Peterson, Kap. 5.1, 5.2.1 – 5.2.4, 5.2.6
- CCNA, Kap. 9

7 Ende-zu-Ende Protokolle ...

★★★

Einordnung

- **Protokolle der Vermittlungsschicht:**
  - Kommunikation zwischen **Rechnern**
  - Adressierung der Rechner
  - IP: *Best Effort*, d.h. keine Garantien
- **Protokolle der Transportschicht:**
  - Kommunikation zwischen **Prozessen**
  - Adressierung von Prozessen auf einem Rechner
  - ggf. Garantien: Zustellung, Reihenfolge, ...
    - Sicherung der Übertragung notwendig
  - zwei Internet-Protokolle: UDP und TCP

7.1 Ports: Adressierung von Prozessen

★★

- Wie identifiziert man Prozesse?
  - **Nicht** durch die Prozeß-ID des Betriebssystems:
    - systemabhängig, „zufällig“
  - **Sondern**: indirekte Adresierung über Ports
    - 16-Bit Nummer
- Woher weiß ein Prozeß die Port-Nummer des Partners?
  - „*well known ports*“ (i.d.R. 0...1023) für Systemdienste
    - z.B.: 80 = Web-Server, 25 = Mail-Server
    - Analogie: Tel. 112 = Feuerwehr
  - Server kennt die Port-Nummer des Clients aus UDP- bzw. TCP-Header der Anfrage

Anmerkungen zu Folie 256:

- Aus Sicherheitsgründen sind die Port-Nummern 0...1023 in den gängigen Betriebssystemen privilegiert, d.h., sie können nur von Prozessen benutzt werden, die mit Administrator-Rechten gestartet wurden.
- Clients können beliebige Port-Nummern ab 1024 benutzen. Sie können die Nummer entweder selbst wählen (sofern sie auf dem Rechner noch unbenutzt ist) oder sich vom Betriebssystem eine freie Port-Nummer zuteilen lassen.

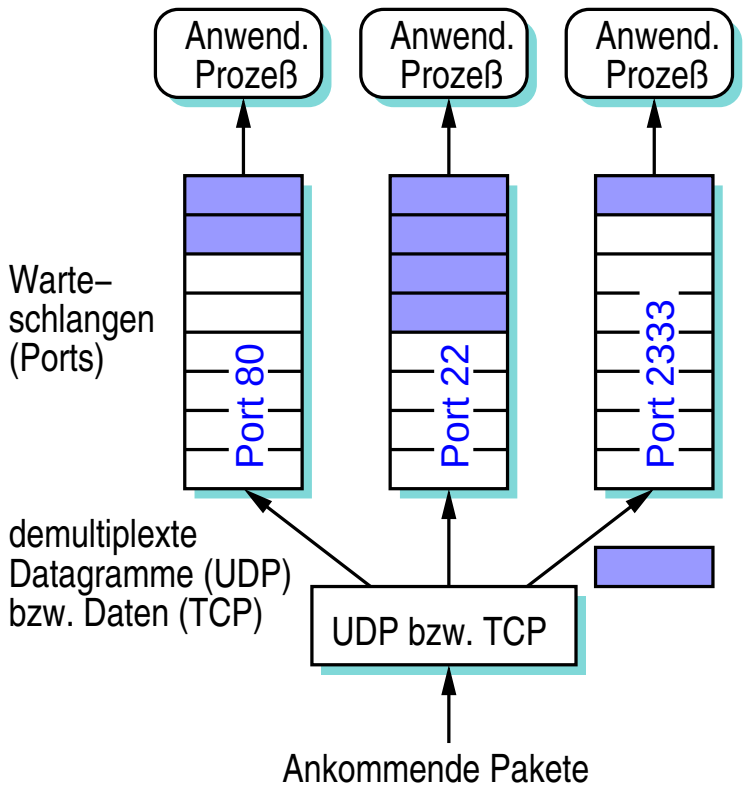
  Häufig kann der Bereich der Port-Nummern, die für Clients verwendet werden können, in der Konfiguration des Betriebssystems eingeschränkt werden. Dies ist notwendig, wenn Firewalls nur bestimmte Bereiche von Port-Nummern erlauben (siehe Abschnitt 10.6).

256-1

7.1 Ports: Adressierung von Prozessen ...

Port-Demultiplexing

- Ports sind typischerweise durch Warteschlangen realisiert
- Bei voller Warteschlange: UDP bzw. TCP verwirft das Paket



Anmerkungen zu Folie 257:

Das Port-Multiplexing unterscheidet sich bei UDP und TCP etwas:

- Bei **UDP** werden unterscheidbare Datagramme in die Warteschlange eingereiht. Der Anwendungsprozess erhält beim Auslesen der Warteschlange jeweils ein Datagramm.
- Bei **TCP** ist die Warteschlange als Folge von Bytes, d.h. als Byte-Strom realisiert (siehe Folie 261). Beim Eintreffen eines TCP-Segments werden die Nutzdaten des Segments in den Byte-Strom eingefügt. Der Anwendungsprozess kann jedes Mal eine völlig beliebige Zahl von Bytes aus dem Strom lesen (solange dieser nicht leer wird).

257-1

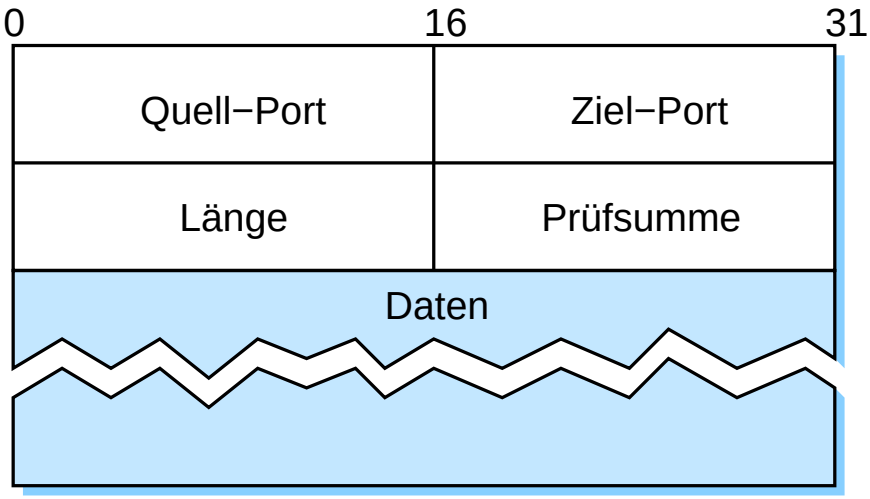
7.2 UDP

★★★

UDP: *User Datagram Protocol*

- Dienstemodell von UDP:
  - Übertragung von Datagrammen zwischen Prozessen
    - unzuverlässiger Dienst
- „Mehrwert“ im Vergleich zu IP:
  - Kommunikation zwischen Prozessen
    - ein Prozess wird identifiziert durch das Paar (Host-IP-Adresse, Port-Nummer)
    - UDP übernimmt das Demultiplexen (☞ **7.1**)
      - d.h. Zustellung an Zielprozess auf dem Zielrechner
  - Prüfsumme über Header und Nutzdaten

Aufbau eines UDP-Pakets



- (Das UDP-Paket ist im Nutzdatenteil eines IP-Pakets!)

7.3 TCP

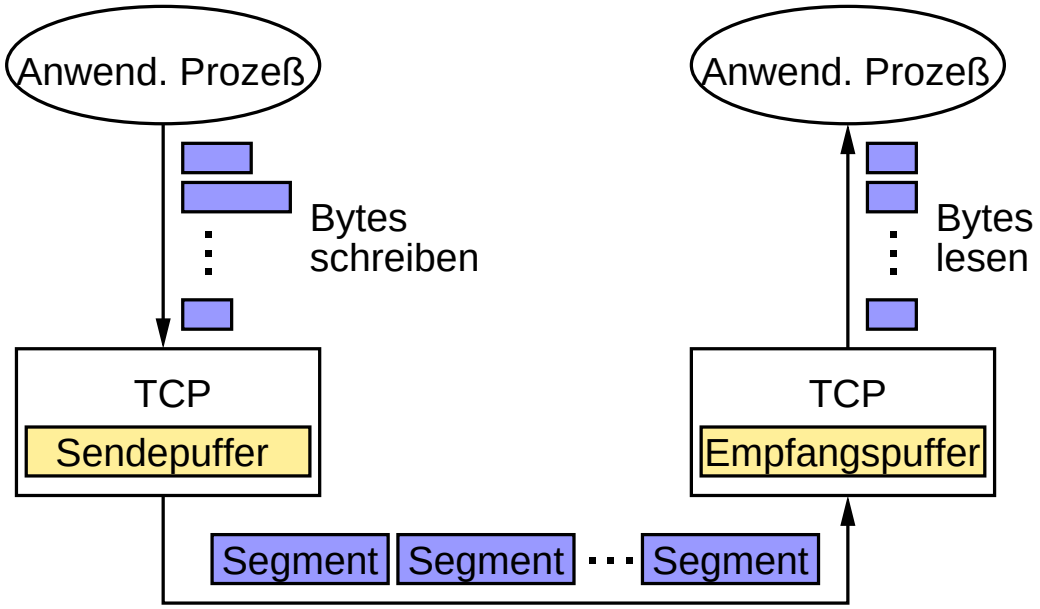
★★★

TCP: *Transmission Control Protocol*

- Dienstemodell von TCP:
  - verbindungsorientierte, zuverlässige Übertragung von Datenströmen zwischen Prozessen
- Meist verwendetes Internet-Protokoll
  - befreit Anwendungen von Sicherung der Übertragung
- TCP realisiert:
  - Port-Demultiplexing (☞ **7.1**)
  - Sicherung der Übertragung, Reihenfolgeerhaltung
  - Flußkontrolle
  - Überlastkontrolle

Bytestrom-Übertragung

- TCP überträgt Daten (Byteströme) segmentweise

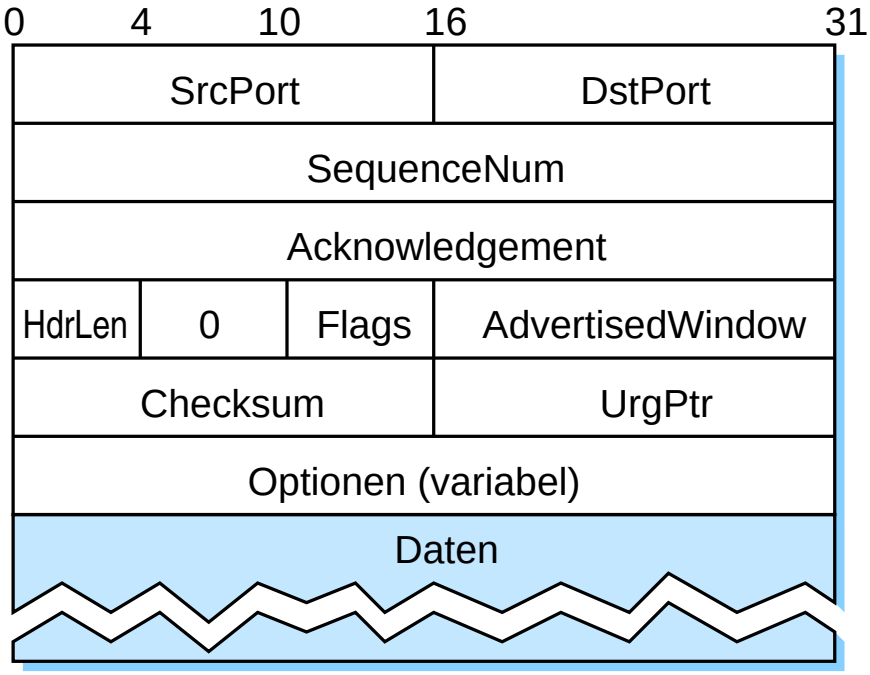


Übertragung von Segmenten

Wann wird ein Segment gesendet?

- Wenn die maximale Segmentgröße erreicht ist
  - ***Maximum Segment Size*** (**MSS**)
  - i.d.R. an maximale Frame-Größe (**MTU**, ***Maximum Transmission Unit***) des lokalen Netzes angepaßt:
    - MSS = MTU – Größe(TCP-Header) – Größe(IP-Header)
    - verhindert, daß das Segment von IP sofort wieder fragmentiert werden muß (☞ **5.4**)
- Wenn der Sender es ausdrücklich fordert
  - *Push*-Operation (*Flush*)
- Nach Ablauf eines periodischen Timers

Aufbau eines TCP Segments



- (Das TCP-Segment ist im Nutzdatenteil eines IP-Pakets!)

★★

Aufbau eines TCP Segments ...

- **SequenceNum, Acknowledgement, AdvertisedWindow:**
  - für *Sliding-Window*-Algorithmus (siehe später, **7.5**)
- **Flags**:
  - SYN Verbindungsaufbau
  - FIN Verbindungsabbau
  - ACK **Acknowledgement**-Feld ist gültig
  - URG Dringende Daten (*out of band data*)
    UrgPtr zeigt Länge der dringenden Daten an
  - PSH Anwendung hat *Push*-Operation ausgeführt
  - RST Abbruch der Verbindung (nach Fehler)

  Es können mehrere Flags gleichzeitig gesetzt sein

7.4 Sicherung der Übertragung

OSI: 4, 2

★★

Problem:

- Bei der Übertragung eines Segments bzw. Frames können Fehler auftreten
  - Empfänger kann Fehler erkennen, aber i.a. nicht korrigieren
  - Segmente bzw. Frames können auch ganz verloren gehen
    - z.B. durch überlasteten Router bzw. Switch
    - oder bei Verlust der Frame-Synchronisation (☞ **3.5**)

- Segmente bzw. Frames* müssen deshalb ggf. neu übertragen werden

* Zur Vereinfachung wird im Folgenden nur der Begriff „Frame“ verwendet!

Anmerkungen zu Folie 265:

Das Problem der Übertragungssicherung tritt sowohl auf der OSI-Schicht 4 als auch auf Schicht 2 auf, wenn ein zuverlässiger Dienst angeboten wird.

Die Tatsache, daß im Fehlerfall nur ein Teil der Daten (nämlich nur die betroffenen Segmente bzw. Frames) neu übertragen werden muß, ist eine wichtige Motivation dafür, daß die Daten zur Übertragung in kleinere Einheiten (Segmente bzw. Frames) aufgeteilt werden. Eine weitere Motivation ist die Möglichkeit des Multiplexings mehrerer Datenströme über eine gemeinsame Leitung (vgl. Abschnitt 3.5).

★★★

Basismechanismen zur Lösung:

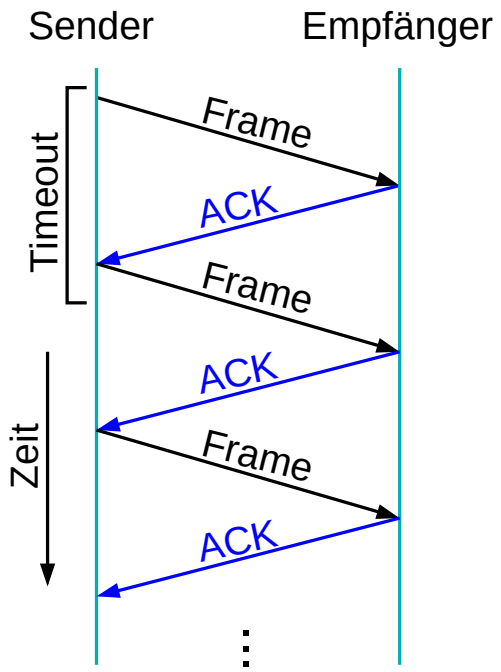
- **Bestätigungen** (***Acknowledgements***, ACK)
  - spezielle Kontrollinformationen, die an Sender zurückgesandt werden
  - bei Duplex-Verbindung (wie z.B. bei TCP) auch **Huckepack-verfahren** (***Piggyback***):
    - Bestätigung wird im Header eines normalen Frames übertragen
- Senderseitige Zwischenspeicherung unbestätigter Frames
- ***Timeouts***
  - wenn nach einer bestimmten Zeit kein ACK eintrifft, überträgt der Sender den Frame erneut

7.4.1 *Stop-and-Wait*-Algorithmus

★★★

Ablauf bei fehlerfreier Übertragung

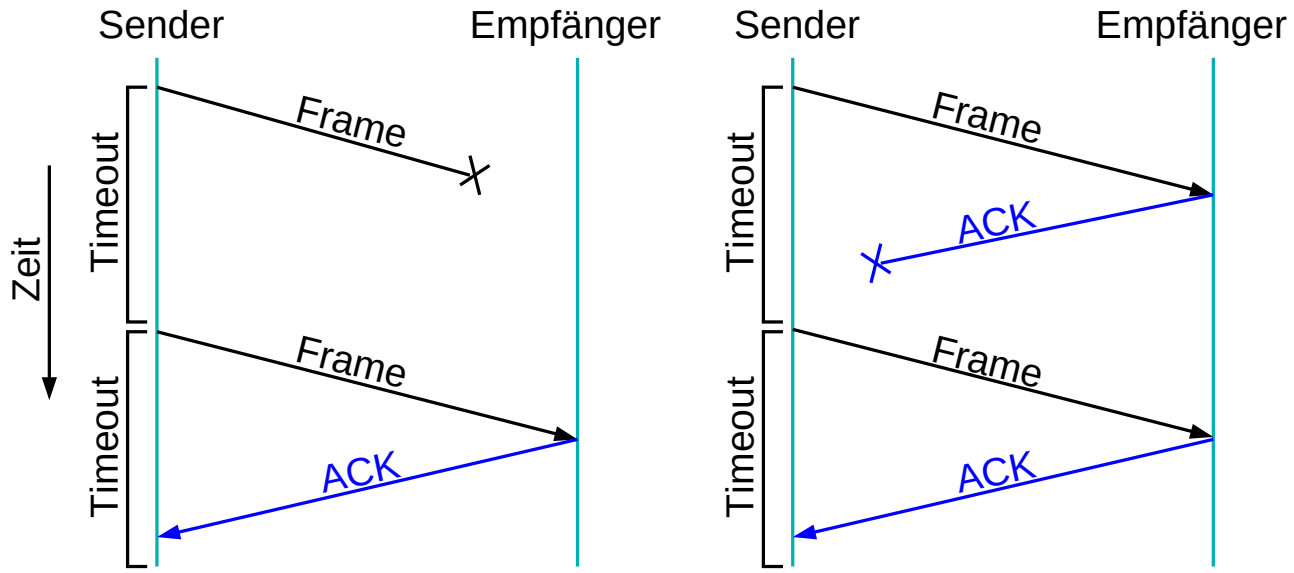
- Sender wartet nach der Übertragung eines Frames, bis ACK eintrifft
- Erst danach wird der nächste Frame gesendet



(Animierte Folie) ★★★

Ablauf bei Übertragungsfehler

- Falls ACK nicht innerhalb der Timout-Zeit eintrifft:
  - Wiederholung des gesendeten Frames

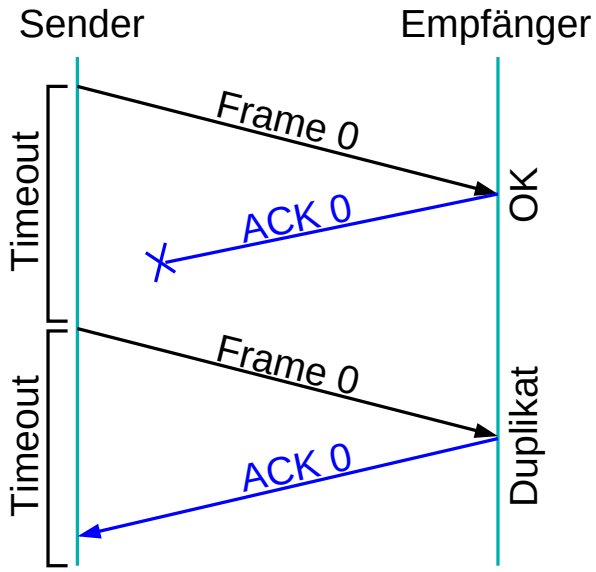


★★★

Was passiert, wenn ACK verloren geht oder zu spät eintrifft?

- Der Empfänger erhält den Frame mehrfach
- Er muß dies erkennen können!

- Daher: Frames und ACKs erhalten eine **Sequenznummer**
- Bei Stop-and-Wait reicht eine 1 Bit lange Sequenznummer
  - d.h. abwechselnd 0 und 1
  - Voraussetzung: Leitung vertauscht keine Frames

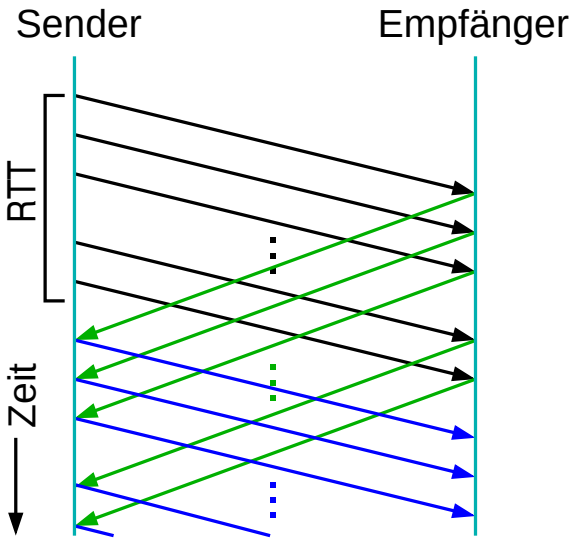


7.4.2 *Sliding-Window*-Algorithmus

(Animierte Folie) ★★★

Motivation

- Problem bei *Stop-and-Wait*:
  - Leitung wird nicht ausgelastet, da nur ein Frame pro RTT übertragen werden kann
- Um Leitung auszulasten:
  - Sender sollte die Datenmenge senden, die dem Verzögerungs(RTT)-Bandbreiten-Produkt entspricht, bevor er auf das erste ACK wartet
  - dann mit jedem ACK einen neuen Frame senden

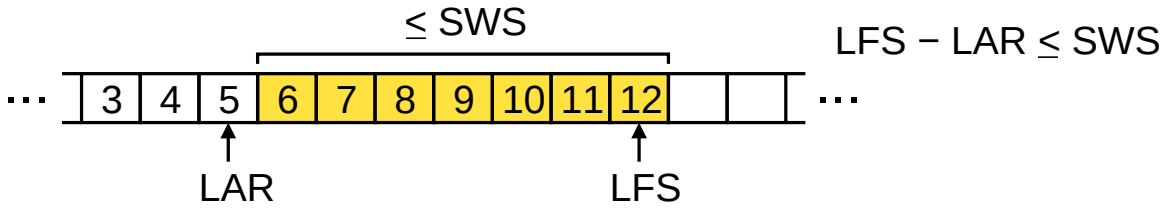


7.4.2 *Sliding-Window*-Algorithmus ...

(Animierte Folie) ★

Funktionsweise

- Jeder Frame erhält eine Sequenznummer
- Der **Sender** besitzt ein „Schiebefenster“ (*Sliding Window*):



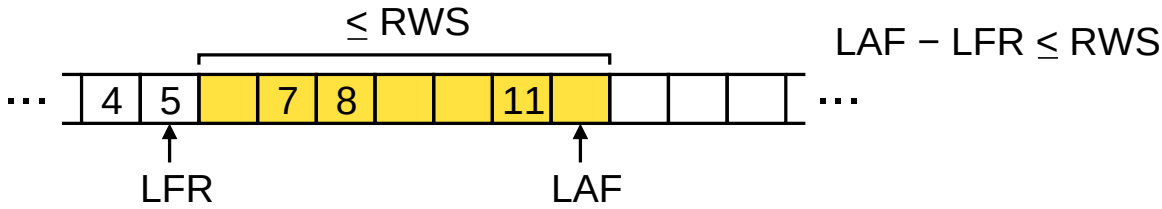
  - Jeder Eintrag steht für einen gesendeten Frame
  - LAR: *Last Acknowledgement Received*
    - bis zu diesem Frame (incl.) wurden alle quittiert
  - LFS: *Last Frame Sent*
  - SWS: *Sender Window Size*
    - max. SWS Frames werden ohne ACK abgeschickt

7.4.2 *Sliding-Window*-Algorithmus ...

(Animierte Folie)

Funktionsweise ...

- Der **Empfänger** hat ebenfalls ein *Sliding Window*:



  - Jeder Eintrag steht für einen empfangenen Frame
  - LFR: *Last Frame Received*
    - alle Frames n mit $n \leq$ LFR wurden korrekt empfangen und quittiert
  - LAF: *Largest Acceptable Frame*
    - Frame n wird nur akzeptiert, wenn LFR $< n \leq$ LAF
  - RWS: *Receiver Window Size*
    - Anzahl der Pufferplätze beim Empfänger

7.4.2 *Sliding-Window*-Algorithmus ...

Quittierung von Frames

- **Akkumulatives Acknowledgement**:
  - ACK für Frame n gilt auch für alle Frames $\leq n$
- Zusätzlich **negative Acknowledgements** möglich:
  - Wenn Frame n empfangen wird,
    aber Frame m mit $m < n$ noch aussteht,
    wird für Frame m ein NACK geschickt
- Alternative: **selektives Acknowledgement**:
  - ACK für Frame n gilt nur für diesen Frame

Problem in der Praxis

- Begrenzte Anzahl von Bits für die Sequenznummer im Frame-Header
  - z.B. bei 3 Bits nur Nummern 0 ... 7 möglich
- Reicht ein endlicher Bereich an Sequenznummern aus?
  - ja, abhängig von SWS und RWS:
    - falls RWS = 1: NSeqNum $\geq$ SWS + 1
    - falls RWS = SWS: NSeqNum $\geq 2 \cdot$ SWS

    (NSeqNum = Anzahl von Sequenznummern)
  - aber nur, wenn die Reihenfolge der Frames bei der Übertragung nicht verändert werden kann!

7.5 Übertragungssicherung in TCP

OSI: 4

- TCP nutzt den *Sliding-Window*-Algorithmus
- Prinzipiell wie in **7.4.2** vorgestellt, aber Unterschiede:
  - Sequenznummer zählt Bytes, nicht Segmente
  - TCP benötigt Verbindungsaufbau und -abbau
    - Austausch der *Sliding-Window* Parameter
  - Netzwerk (IP) kann Pakete umordnen
    - TCP toleriert bis zu 120 Sekunden alte Pakete
  - Keine feste Fenstergröße
    - Sendefenstergröße angepasst an Puffer des Empfängers bzw. Lastsituation im Netz
  - RTT ist nicht konstant, sondern ändert sich laufend
    - Timeout muß adaptiv sein

★★

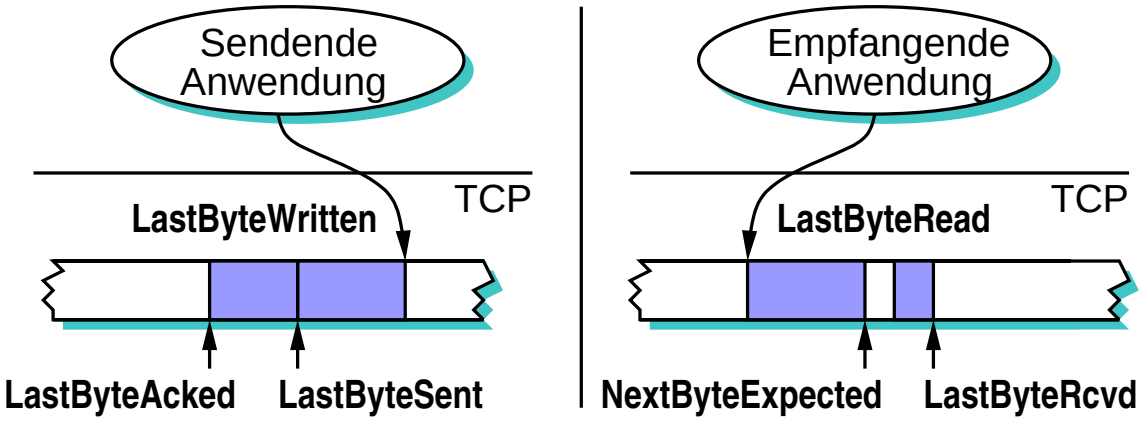
Aufgaben des Sliding-Window-Algorithmus in TCP

- Zuverlässige Übertragung
- Sicherstellung der richtigen Reihenfolge der Segmente
  - TCP gibt Segmente nur dann an obere Schicht weiter, wenn alle vorherigen Segmente bestätigt wurden
- Flußkontrolle
  - keine feste Sendefenstergröße
  - Empfänger teilt dem Sender den freien Pufferplatz mit (**AdvertisedWindow**)
  - Sender passt Sendefenstergröße entsprechend an
- Überlastkontrolle
  - Sendefenstergröße wird dynamisch an Netzlast angepasst

★

Zuverlässige und geordnete Übertragung

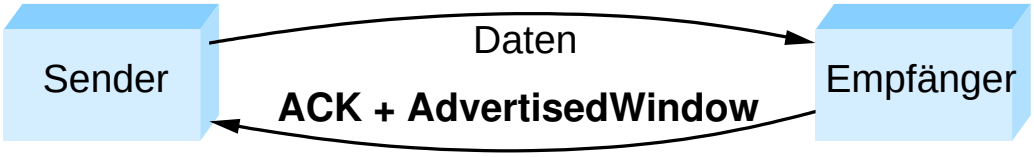
- Algorithmus arbeitet auf Byte-Ebene
  - Sequenznummern werden um die Anzahl gesendeter bzw. empfangener Bytes erhöht



★★

Flußkontrolle

- Empfänger teilt Sender die Größe des freien Puffers mit:



  - **AdvertisedWindow** $=$ **MaxRcvBuffer** $-$ **(LastByteRcvd** $-$ **LastByteRead)**
- Sender muß sicherstellen, daß jederzeit gilt:
  - **LastByteSent** $-$ **LastByteAcked** $\leq$ **AdvertisedWindow**
- Differenz: Datenmenge, die der Sender noch senden kann
- Sendende Anwendung wird blockiert, wenn Daten (**y** Bytes) nicht mehr in Sendepuffer passen, d.h. wenn
  - **LastByteWritten** $-$ **LastByteAcked** $+$ **y** $>$ **MaxSendBuffer**

Anmerkungen zu Folie 278:

Die Datenmenge, die der Sender noch senden kann, ist
EffectiveWindow $=$ **AdvertisedWindow** $-$ **(LastByteSent** $-$ **LastByteAcked)**

Sequenznummern-Überlauf

- Erinnerung an **7.4.2**: endlicher Seqenznummernbereich nur möglich, wenn Netzwerk die Reihenfolge erhält
- TCP-Header: 32-Bit Feld für Sequenznummern
- Pakete können bis zu 120 Sekunden alt werden

Bandbreite	Zeit bis zum Überlauf
10 MBit/s (Ethernet)	57 Minuten
100 MBit/s (FDDI)	6 Minuten
155 MBit/s (OC−3)	4 Minuten
1,2 GBit/s (OC−24)	28 Sekunden
9,95 GBit/s (OC−192)	3,4 Sekunden

- ⇒ TCP-Erweiterung: Zeitstempel als Überlaufschutz

Größe des AdvertisedWindow

- TCP-Header sieht 16-Bit vor, d.h. max. 64 KBytes
- Nötige Sendefenster-Größe, um Kanal gefüllt zu halten, bei RTT = 100 ms (z.B. Transatlantik-Verbindung):

Bandbreite	RTT * Bandbreite
10 MBit/s (Ethernet)	122 KByte
100 MBit/s (FDDI)	1,2 MByte
155 MBit/s (OC−3)	1,8 MByte
1,2 GBit/s (OC−24)	14,8 MByte
9,95 GBit/s (OC−192)	119 MByte

- ⇒ TCP-Erweiterung: Festlegung eines Skalierungsfaktors für **AdvertisedWindow** beim Verbindungsaufbau

Adaptive Neuübertragung

- Timeout für Neuübertragung muß abhängig von RTT gewählt werden
- Im Internet: RTT ist unterschiedlich und veränderlich
- Daher: adaptive Bestimmung des Timeouts nötig
  - ursprünglich:
    - Messung der durchschnittlichen RTT (Zeit zwischen Senden eines Segments und Ankunft des ACK)
    - Timeout = 2 · durchschnittliche RTT
  - Problem:
    - Varianz der RTT-Meßwerte nicht berücksichtigt
    - bei hoher Varianz sollte der Timeout deutlich über dem Mittelwert liegen

Adaptive Neuübertragung: Jacobson/Karels-Algorithmus

- Berechne gleitenden Mittelwert und (approximierte) Standardabweichung der RTT:
  - $\textbf{Deviation} = \delta \cdot |\textbf{SampleRTT} - \textbf{EstimatedRTT}| + (1-\delta) \cdot \textbf{Deviation}$
  - $\textbf{EstimatedRTT} = \delta \cdot \textbf{SampleRTT} + (1-\delta) \cdot \textbf{EstimatedRTT}$
- Berücksichtige Standardabweichung bei Timeout-Berechnung:
  - $\textbf{TimeOut} = \mu \cdot \textbf{EstimatedRTT} + \Phi \cdot \textbf{Deviation}$
- Typisch: μ =1, Φ = 4, δ = 0,125

7.6 Überlastkontrolle

OSI: 4, 2

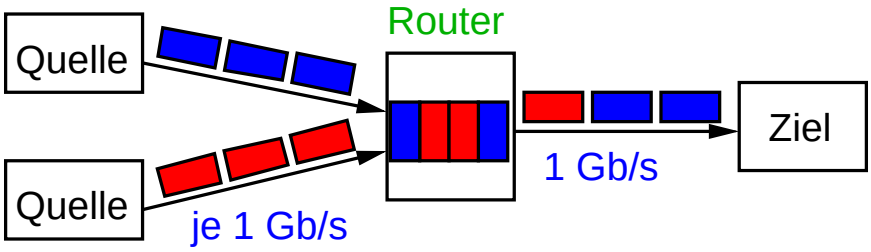
★★★

Was bedeutet Überlast?

- Pakete konkurrieren um Bandbreite einer Verbindung
- Bei unzureichender Bandbreite:
  - Puffern der Pakete im Router
- Bei Pufferüberlauf:
  - Pakete verwerfen
- Ein Netzwerk mit häufigem Pufferüberlauf heißt **überlastet** (*congested*)

7.6 Überlastkontrolle ...

Beispiel einer Überlastsituation



- Sender können das Problem nicht direkt erkennen
- Adaptives Routing löst das Problem nicht, trotz schlechter Link-Metrik für überlastete Leitung
  - verschiebt Problem nur an andere Stelle
  - Umleitung ist nicht immer möglich (evtl. nur ein Weg)
  - im Internet wegen Komplexität derzeit utopisch

7.6 Überlastkontrolle ...

★★

Überlastkontrolle

- Erkennen und möglichst schnelles Beenden der Überlast
  - z.B. einige Sender mit hoher Datenrate stoppen
  - in der Regel aber Fairness gewünscht

Überlastvermeidung

- Erkennen von drohenden Überlastsituationen und Vermeidung der Überlast (☞ **Rechnernetze II**)

Abgrenzung

- **Flußkontrolle** verhindert, daß **ein Sender** seinen **Empfänger** überlastet
- **Überlastkontrolle** (bzw. **-vermeidung**) verhindert, daß **mehrere Sender** einen Teil des **Netzwerks** (= Zwischenknoten) überlasten

7.6.1 Überlastkontrolle in TCP

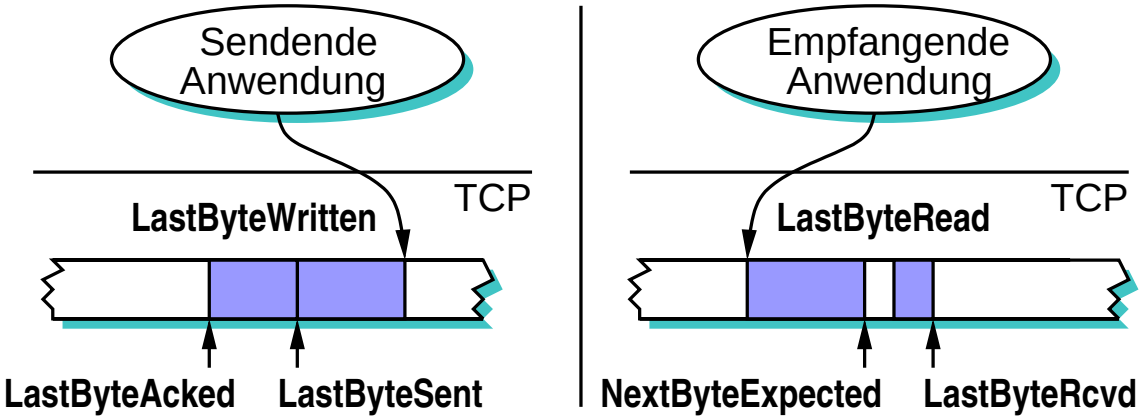
★

- Einführung Ende der 1980'er Jahre (8 Jahre nach Einführung von TCP) zur Behebung akuter Überlastprobleme
  - Überlast ⇒ Paketverlust ⇒ Neuübertragung ⇒ noch mehr Überlast!
- Idee:
  - jeder Sender bestimmt, für wieviele Pakete (Segmente) Platz im Netzwerk ist
  - wenn Netz „gefüllt" ist:
    - Ankunft eines ACKs ⇒ Senden eines neuen Pakets
    - „selbsttaktend"
- Problem: Bestimmung der (momentanen) Kapazität
  - dauernder Auf- und Abbau anderer Verbindungen

7.6.1 Überlastkontrolle in TCP ...

Erinnerung: Flußkontrolle mit *Sliding-Window*-Algorithmus

- Empfänger sendet in ACKs **AdvertisedWindow** = **MaxRcvBuffer** − **(LastByteRcvd** − **LastByteRead)**
- Sender darf dann noch maximal so viele Bytes senden: **EffectiveWindow** = **AdvertisedWindow** − **(LastByteSent** − **LastByteAcked)**



7.6.1 Überlastkontrolle in TCP ...

Erweiterung des *Sliding-Window*-Algorithmus

- Einführung eines **CongestionWindow**
  - Sender kann noch so viele Bytes senden, ohne Netzwerk zu überlasten
- Neue Berechnung für **EffectiveWindow**
  - **MaxWindow** = **MIN (CongestionWindow, AdvertisedWindow)**
  - **EffectiveWindow** = **MaxWindow** − **(LastByteSent** − **LastByteAcked)**
- Damit: weder Empfänger noch Netzwerk überlastet
- Frage: Bestimmung des **CongestionWindow**?

7.6.1 Überlastkontrolle in TCP ...

Bestimmung des CongestionWindow

- Hosts bestimmen **CongestionWindow** durch Beobachtung des Paketverlusts
- Basismechanismus:
  - *Additive Increase / Multiplicative Decrease*
- Erweiterungen:
  - *Slow Start*
  - *Fast Retransmit / Fast Recovery*

7.6.2 *Additive Increase / Multiplicative Decrease*

★

Vorgehensweise

- **CongestionWindow** sollte
  - groß sein ohne / bei wenig Überlast
  - klein sein bei viel Überlast
- Überlast wird erkannt durch Paketverlust
- Bei Empfang eines ACK:
  - **Increment** $=$ **MSS** $\cdot$ **(MSS** $/$ **CongestionWindow)**
  - **CongestionWindow** $+=$ **Increment**

  im Mittel: Erhöhung um MSS Bytes pro RTT
  (MSS = *Maximum Segment Size* von TCP)
- Bei Timeout: **CongestionWindow** halbieren
  - höchstens, bis MSS erreicht ist

Anmerkungen zu Folie 290:

Die Grundidee beim *Additive Increase / Multiplicative Decrease* ist folgende:

- Wenn das Überlastfenster vollständig übertragen wurde, vergrößere das Fenster um ein Paket (additiv)
- Wenn ein Paket verloren geht, halbiere das Fenster (multiplikativ)

Da in TCP die Fenstergröße aber nicht in Paketen (bzw. Segmenten), sondern in Bytes gemessen wird, wird das **Increment** gemäß folgender Überlegung berechnet:

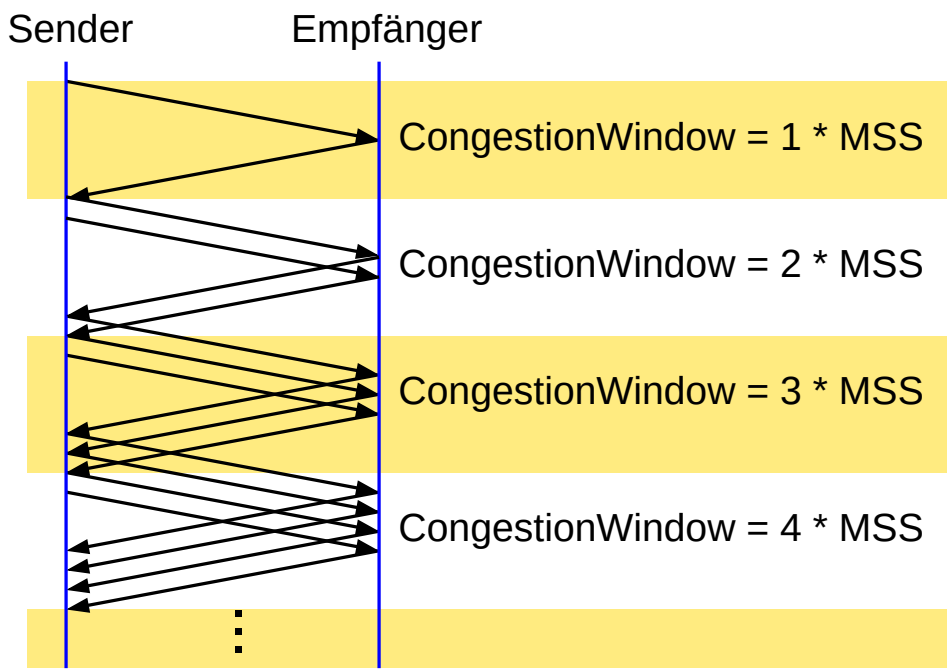
- Pro RTT wird immer ein **CongestionWindow** an Daten versendet.
- Im Mittel soll daher das **CongestionWindow** um ein (maximal grosses) Segment, also 1 **MSS**, pro RTT erhöht werden.
- Wenn maximal grosse Segmente versendet werden, bestätigt somit jedes ACK den Anteil **MSS** / **CongestionWindow** der Gesamtdatenmenge.
- Also wird das Überlastfenster mit jedem ACK um diesen Anteil der **MSS** erhöht, d.h. um **(MSS / CongestionWindow) · MSS**.

290-1

7.6.2 *Additive Increase / Multiplicative Decrease ...*

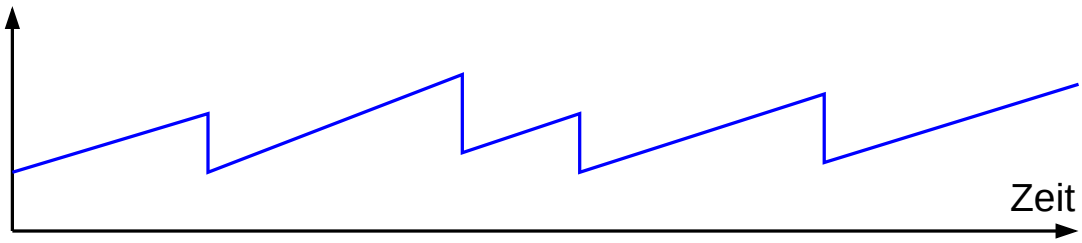
★

Additive Increase



7.6.2 *Additive Increase / Multiplicative Decrease ...*

Typischer Zeitverlauf des CongestionWindow



- Vorsichtige Erhöhung bei erfolgreicher Übertragung, drastische Reduzierung bei Erkennung einer Überlast
  - ausschlaggebend für Stabilität bei hoher Überlast
- Wichtig: gut angepaßte Timeout-Werte
  - Jacobson/Karels Algorithmus

7.6.3 *Slow Start*

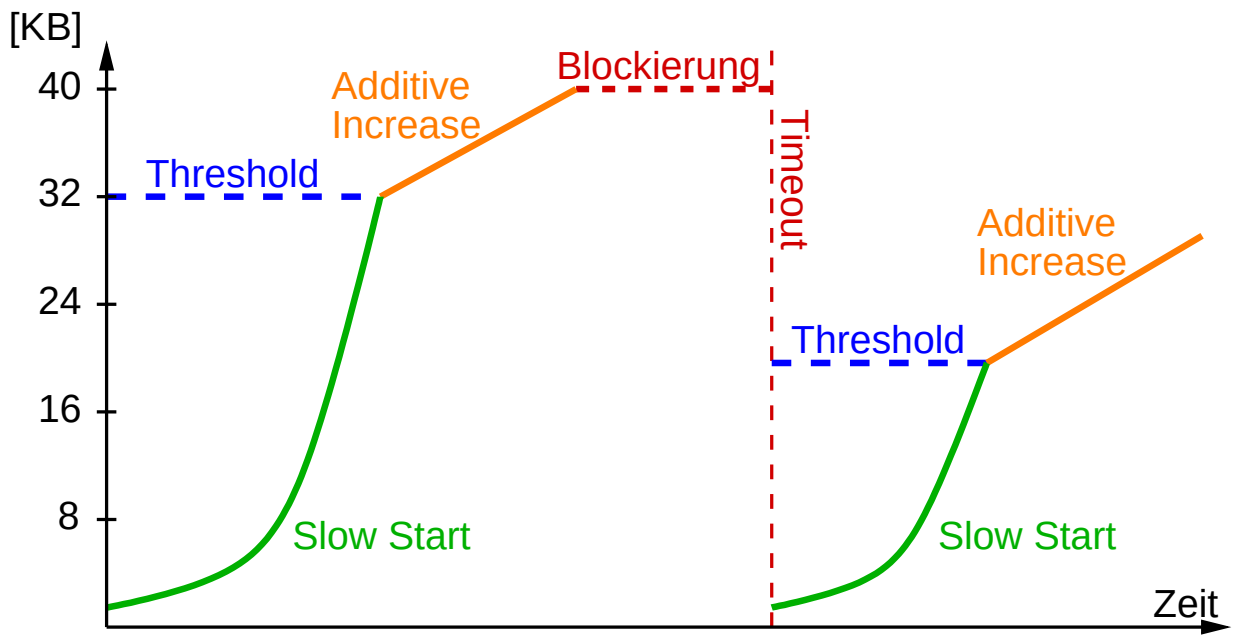
Hintergrund

- Verhalten des ürsprünglichen TCP beim Start (bzw. bei Wiederanlauf nach Timeout):
  - sende **AdvertisedWindow** an Daten (ohne auf ACKs zu warten)
  - d.h. Start mit maximalem **CongestionWindow**
- Zu aggressiv, kann zu hoher Überlast führen
- Andererseits: Start mit **CongestionWindow** = MSS und *Additive Increase* dauert zu lange
- Daher Mittelweg:
  - Start mit **CongestionWindow** = MSS
  - Verdopplung bis zum ersten Timeout

Verhalten bei Timeout

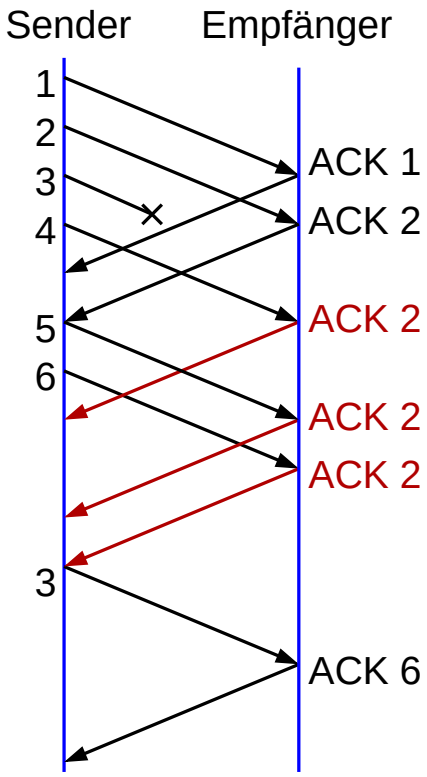
- *Slow Start* wird auch verwendet, wenn eine Verbindung bis zu einem Timeout blockiert:
  - Paket X geht verloren
  - Sendefenster ist ausgeschöpft, keine weiteren Pakete
  - nach Timeout: X wird neu übertragen, ein kumulatives ACK öffnet Sendefenster wieder
- In diesem Fall beim Timeout:
  - **CongestionThreshold** = **CongestionWindow** / 2
  - *Slow Start*, bis **CongestionThreshold** erreicht ist, danach *Additive Increase*

Typischer Verlauf des CongestionWindow



7.6.4 *Fast Retransmit / Fast Recovery*

- Lange Timeouts führen oft zum Blockieren der Verbindung
- Idee: Paketverlust kann auch durch Duplikat-ACKs erkannt werden
- Nach dem dritten Duplikat-ACK:
  - Paket erneut übertragen
  - **CongestionWindow** halbieren ohne *Slow Start*
- *Slow Start* nur noch am Anfang und bei wirklichem Timeout



7.7 TCP Verbindungsauf- und -abbau

★★

Verbindungsaufbau

- Asymmetrisch:
  - Client (rufender Teilnehmer): **aktives Öffnen**
    - sende Verbindungswunsch zum Server
  - Server (gerufener Teilnehmer): **passives Öffnen**
    - warte auf eingehende Verbindungswünsche
    - akzeptiere ggf. einen Verbindungswunsch

Verbindungsabbau

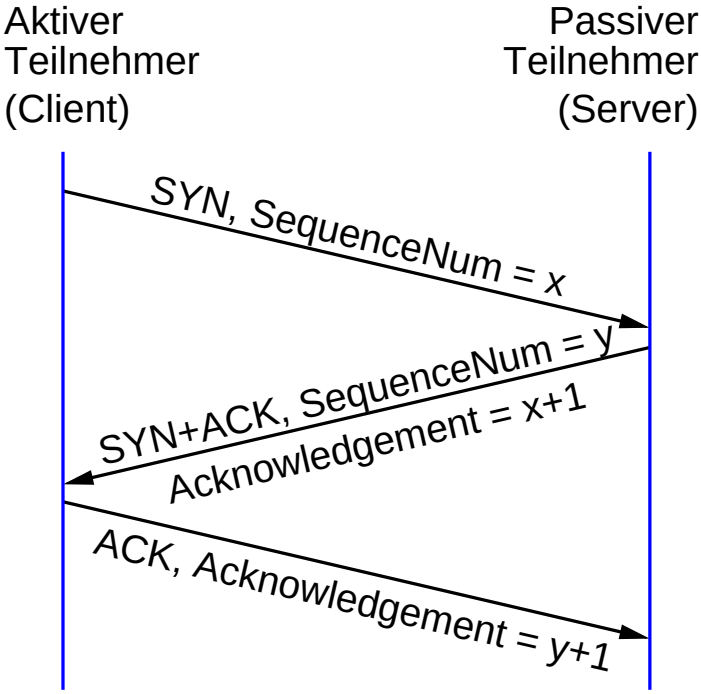
- Symmetrisch:
  - beide Seiten müssen die Verbindung schließen

Zum Begriff der TCP-Verbindung

- Das Tupel (Quell-IP-Adresse, Quell-Port, Ziel-IP-Adresse, Ziel-Port) kennzeichnet eine TCP-Verbindung eindeutig
  - Nutzung als Demultiplex-Schlüssel
- Nach Abbau einer Verbindung und Wiederaufbau mit denselben IP-Adressen und Port-Nummern:
  - neue **Inkarnation** derselben Verbindung

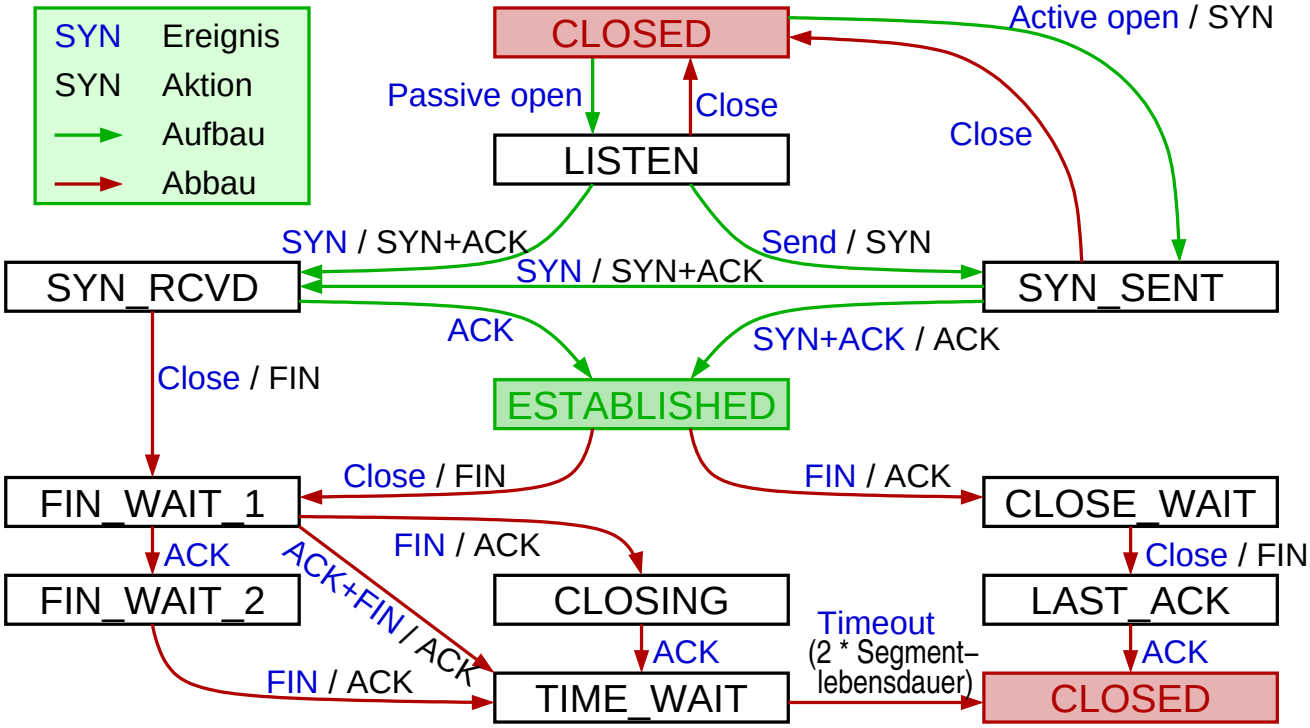
★★

Verbindungsaufbau: *Three-Way Handshake*



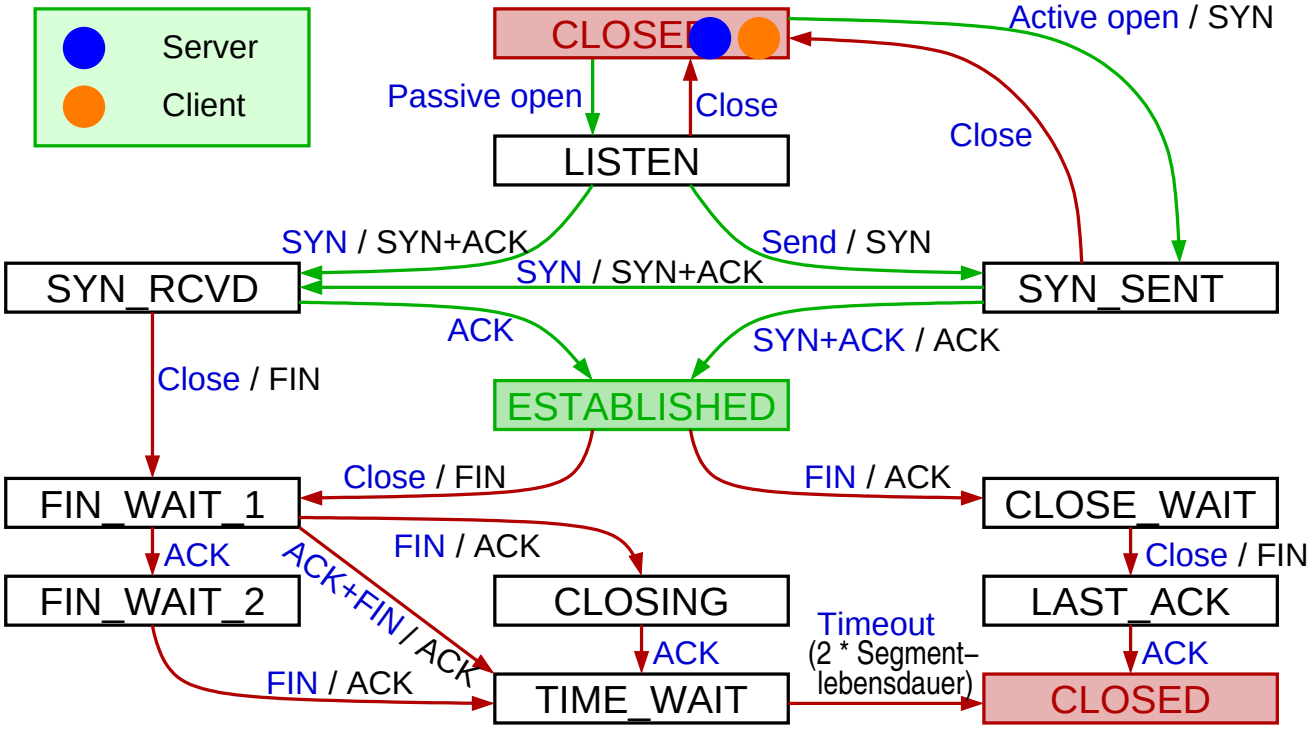
- Austausch von Sequenznummern
- „Zufälliger“ Startwert
  - jede Inkarnation nimmt andere Nummern
- Acknowledgement: nächste erwartete Sequenznummer

Zustände einer TCP-Verbindung



(Animierte Folie)

Zustände einer TCP-Verbindung ...



Anmerkungen zu Folie 301:

Im Beispiel kann der Client seinen Port nach dem Abbau der Verbindung längere Zeit nicht wiederverwenden (Zustand TIME_WAIT). Damit wird verhindert, daß zu schnell eine neue Inkarnation derselben Verbindung aufgebaut wird, da dies zu Problemen führen könnte, wenn noch Segmente aus der alten Inkarnation unterwegs sind.

301-1

7.8 Zusammenfassung

- Ende-zu-Ende Protokolle: Kommunikation zwischen Prozessen
- UDP: unzuverlässige Übertragung von Datagrammen
- TCP: zuverlässige Übertragung von Byte-Strömen
  - Verbindungsaufbau
- Sicherung der Übertragung allgemein
  - *Stop-and-Wait*, *Sliding-Window*
- Übertragungssicherung in TCP (inkl. Fluß- und Überlastkontrolle)
  - *Sliding-Window*-Algorithms, adaptive Neuübertragung

Nächste Lektion:

- Datendarstellung

Rechnernetze I

SoSe 2024

8 Datendarstellung

8 Datendarstellung ... OSI: 6

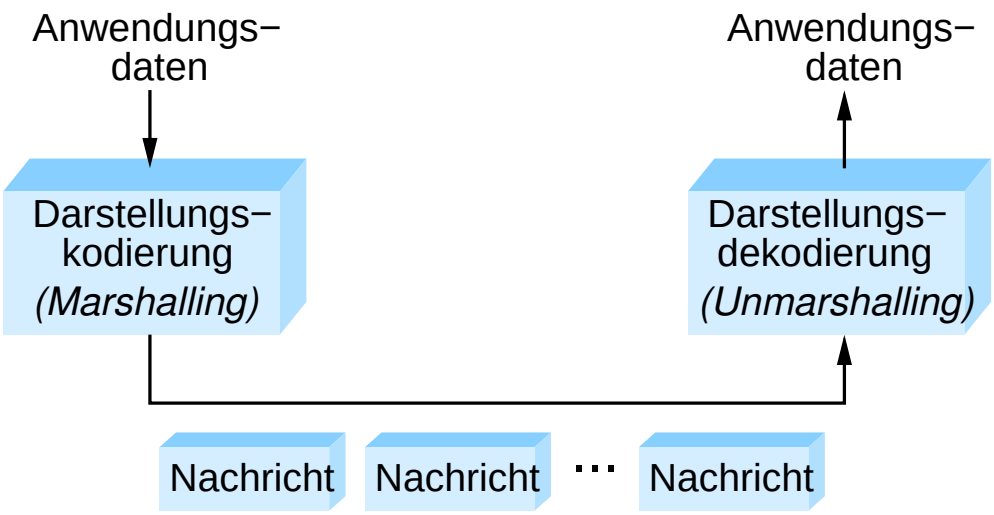
Inhalt

- *Marshalling*, Designalternativen

- Peterson, Kap. 7.1

8.1 Marshalling

- Daten der Anwendungen müssen in eine für die Übertragung geeignete Form umgewandelt werden:
  - **Darstellungsformatierung**
  - ***Marshalling / Unmarshalling***



8.1 Marshalling ...

Problem: Heterogenität

- Rechner haben unterschiedliche Datendarstellung
- Z.B. einfache 32-Bit Ganzzahl
  - 34.677.374 = $0211227E_{16}$ (Hexadezimal)

Big–endian
(z.B. Sparc)

(02_{16})	(11_{16})	(22_{16})	$(7E_{16})$
00000010	00010001	00100010	01111110

Little–endian
(z.B. x86)

$(7E_{16})$	(22_{16})	(11_{16})	(02_{16})
01111110	00100010	00010001	00000010
Adresse x	x+1	x+2	x+3

★

Unterschiede in der Datendarstellung

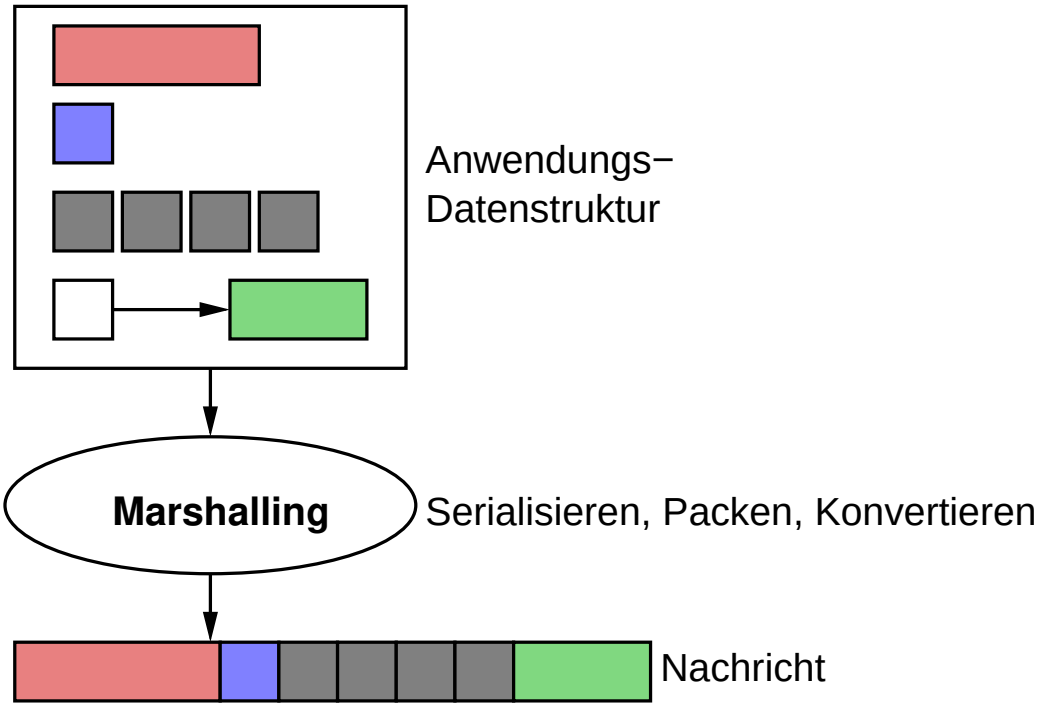
- Ganzzahlen:
  - Größe (16, 32, 64 Bit)
  - Byte-Reihenfolge: *Big Endian* vs. *Little Endian*
- Gleitkommazahlen
  - IEEE 754 vs. proprietäre Formate
  - Größe (bes. bei *extended precision*: 80, 96, 128 Bit)
  - Byte-Reihenfolge
- Datenstrukturen:
  - *Alignment* der Komponenten im Speicher
    - z.B. darf eine 16-Bit Zahl an einer ungeraden Adresse beginnen?

★★

Datentypen beim *Marshalling*: drei Ebenen

- **Basistypen**: int, double, Boolean, char, ...
  - Formate und Byte-Reihenfolge konvertieren
- **Flache Datentypen**: Strukturen, Arrays
  - Füllbytes für *Alignment* entfernen (**packen**) bzw. einfügen (**auspacken**)
- **Komplexe Datentypen (mit Zeigern)**: Listen, Bäume, ...
  - **Serialisierung** der Datenstruktur notwendig
    - Zeiger sind Speicheradressen,
      sie können nicht übertragen werden
  - Empfänger **deserialisiert** wieder

Marshalling veranschaulicht



Konvertierungsstrategien

- **Kanonisches Netzwerk-Datenformat**
  - Sender konvertiert in Netzwerk-Datenformat
  - Empfänger konvertiert dann in sein Format
- ***Receiver-Makes-Right***
  - Sender sendet in seinem eigenen Format
    - serialisiert lediglich
  - Empfänger konvertiert in sein Format, falls nötig
- Diskussion:
  - *Receiver-Makes-Right* ist N-mal-N-Lösung
    - jeder Empfänger muß alle Formate kennen
  - Meist sind die Formate aber identisch (N ist klein)

Datentyp-Kennzeichnung (Tags)

- **Tag**: Information, die hilft, die Nachricht zu dekodieren, z.B.:
  - Typ-Tag
  - Längen-Tag (etwa für Arrays)
  - Architektur-Tag (für *Receiver-Makes-Right*)
- Beispiel: 32-Bit-Ganzzahl mit Tags:

Typ = int	Länge = 4	Wert = 417892

- Oft kann man auch ohne Tags arbeiten:
  - Sender und Empfänger wissen, was übertragen wird
  - Darstellungsformatierung dann aber Ende-zu-Ende

8.2 Zusammenfassung

- Daten werden in Rechnern unterschiedlich dargestellt
- Bei der Übertragung ist eine Anpassung erforderlich
  - 3 Schritte: Serialisieren, Packen, Konvertieren
- Realisierungs-Alternativen:
  - kanonisches Format vs. *Receiver-Makes-Right*
  - mit / ohne Datentyp-Kennzeichnung

Nächste Lektion:

- Anwendungsprotokolle

Rechnernetze I

SoSe 2024

9 Anwendungsprotokolle

9 Anwendungsprotokolle ... OSI: 7

Inhalt

- SMTP (*Simple Mail Transport Protocol*)
- HTTP (*Hypertext Transport Protocol*)
- DNS (*Domain Name Service*)

- Peterson, Kap. 9.1, 9.2.1 – 9.2.2
- CCNA, Kap. 10

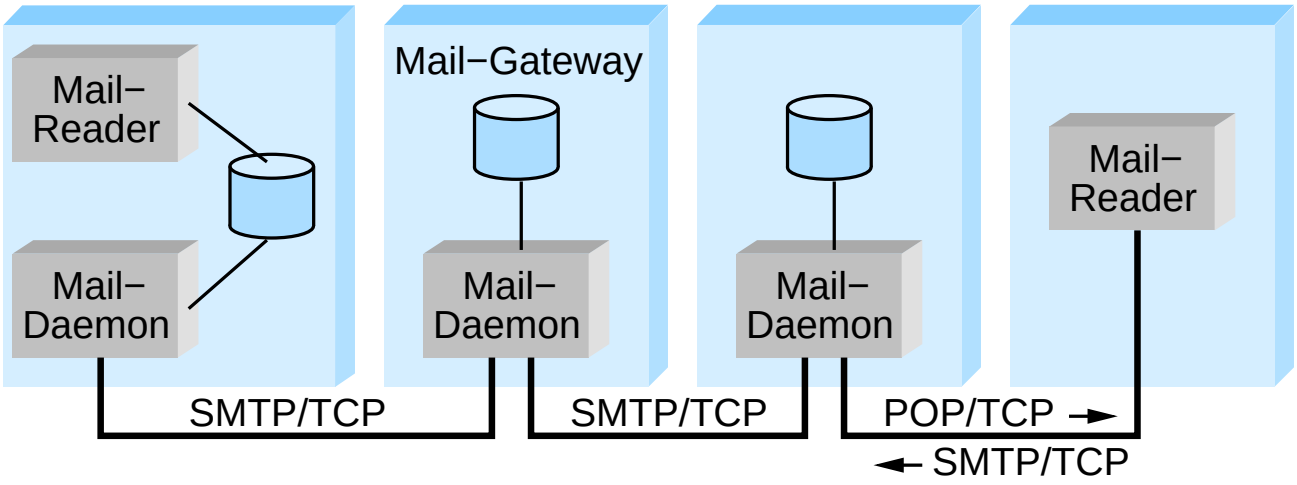
9.1 SMTP

Protokolle für Emails

- RFC 822 und MIME: Format einer Email
  - Header (ASCII)
  - Rumpf (ASCII)
    - binäre Daten (z.B. Bilder) werden durch MIME in ASCII codiert
- **SMTP (*Simple Mail Transport Protocol*)**
  - regelt Weitergabe der Emails zwischen Rechnern
  - textbasiertes Protokoll
    - vermeidet Darstellungsprobleme
    - erleichtert Test und Debugging
- **POP** (*Post Office Prot.*) / **IMAP** (*Internet Message Access Prot.*)
  - Herunterladen der Emails von einem entfernten Server

9.1 SMTP ...

Transport von Emails über Mail-Gateways



- Eigene Speichervermittung, da
  - Zielrechner beim Sender nicht immer bekannt
  - Zielrechner nicht immer erreichbar

Anmerkungen zu Folie 316:

Der Mail-Reader (MUA, *Mail User Agent*) bekommt eingehende E-Mails entweder über einen *Mail Delivery Agent* (MDA) vom lokalen Mail-Daemon (MTA, *Mail Transfer Agent*) zugestellt, oder kann sie über Protokolle wir POP und IMAP von einem entfernten Mail-Server abholen.

316-1

9.1 SMTP ...

Beispiel zum Aufbau einer Email

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----417CAWVF...
From: Alice Smith <Alice@cicso.com>
To: Bob@cs.Princeton.edu
Subject: proposed material
Date: Mon, 07 Sep 1988 19:45:19 -0400

----417CAWVFNCVKRZKAZCFHKWFHQ
Content-Type: text/plain; charset=us-asci
Content-Transfer-Encoding: 7bit

Bob, here is the jpeg image I promised.
Alice

----417CAWVFNCVKRZKAZCFHKWFHQ
Content-Type: image/jpeg; name="beach.jpg"
Content-Transfer-Encoding: base64
```

Header

Rumpf

... ASCII–codiertes JPEG–Bild ...

Beispiel eines SMTP Dialogs

(Client, *Server*)

```
HELO cs.princeton.edu
250 Hello daemon@mail.cs.princeton.edu [128.12.169.24]
MAIL FROM: <Alice@cs.princeton.edu>
250 OK
RCPT TO: <Bob@cisco.com>
250 OK
RCPT TO: <Tom@cisco.com>
550 No such user here
DATA
354 Start mail input; end with <CRLF>.<CRLF>
Blah blah blah ...
... etc. etc. etc.
.
250 OK
QUIT
221 Closing connection
```

siehe vorige Folie

9.2 HTTP

★★

- HTTP ist wie SMTP textbasiert
- Kommandos (u.a.):

GET <URL>	Anfrage nach einem Dokument
HEAD <URL>	Anfrage nach Metainformation
POST <URL>	Senden von Information an den Server
PUT <URL>	Speichern eines Dokuments
DELETE <URL>	Löschen eines Dokuments

- Beispiel-Anfrage:

```
GET index.html HTTP/1.1
Host: www.cs.princeton.edu
```

9.2 HTTP ...

- Ergebniscodes:

1xx	Informativ	Anfrage erhalten
2xx	Erfolg	Anfrage empfangen u. akzeptiert
3xx	Weiterleitung	Weitere Aktionen notwendig
4xx	Client-Fehler	Syntaxfehler, nicht erfüllbar, ...
5xx	Server-Fehler	Anfrage OK, Problem im Server

- Beispiel-Antwort:

```
HTTP/1.1 301 Moved Permanently
Location: http://www.cs.princeton.edu/cs/index.html
```


9.2 HTTP ...

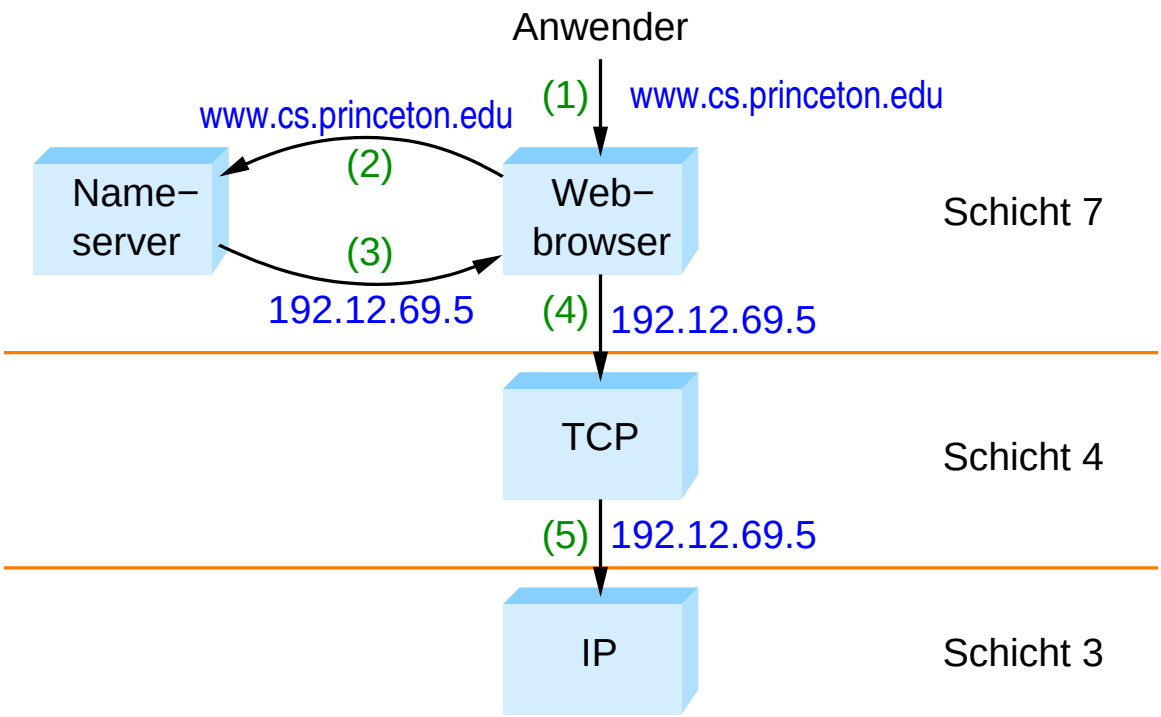
- In HTTP 1.0:
  - neue TCP-Verbindung für jedes Seiten-Element
    - z.B. Frame, eingebettetes Bild, ...
- Ab HTTP 1.1: persistente Verbindungen möglich
  - mehrere Anfragen / Antworten über dieselbe TCPVerbindung
  - deutlich effizienter (Verbindungsaufbau)
  - Problem: Server muß viele Verbindungen offenhalten
    - wie lange? $\Rightarrow$ Timeout-Mechanismus notwendig
- Einsatz von Caches (Proxy-Server)
  - nahe beim Client, speichert häufig besuchte Seiten

9.3 DNS

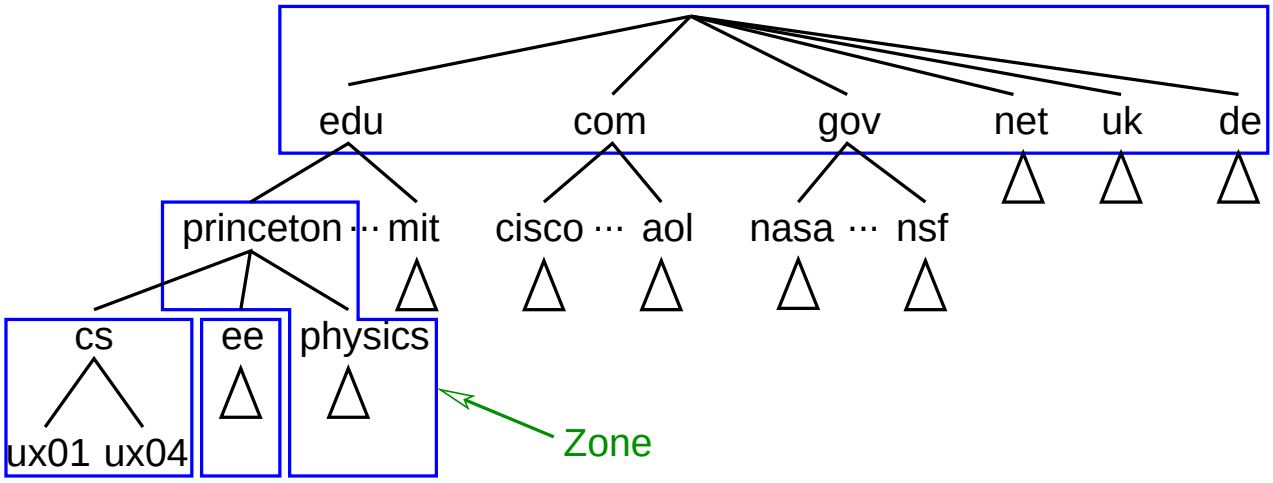
- Das Internet-Protokoll (IP) arbeitet mit numerischen Adressen
  - einheitliches Format (Länge)
  - identifizieren das Netz des Hosts (logische Adressen)
    - wichtig für das Routing von Paketen
  - nicht benutzerfreundlich
- Benutzer verwenden (Host-/Rechner-)Namen:
  - benutzerfreundlich
  - unterschiedliche Länge
  - nicht für das Routing nutzbar
- Daher: Umsetzung von Namen auf IP-Adressen:
  DNS (*Domain Name Service*)

9.3 DNS ...

Ablauf der Umsetzung von Namen in Adressen

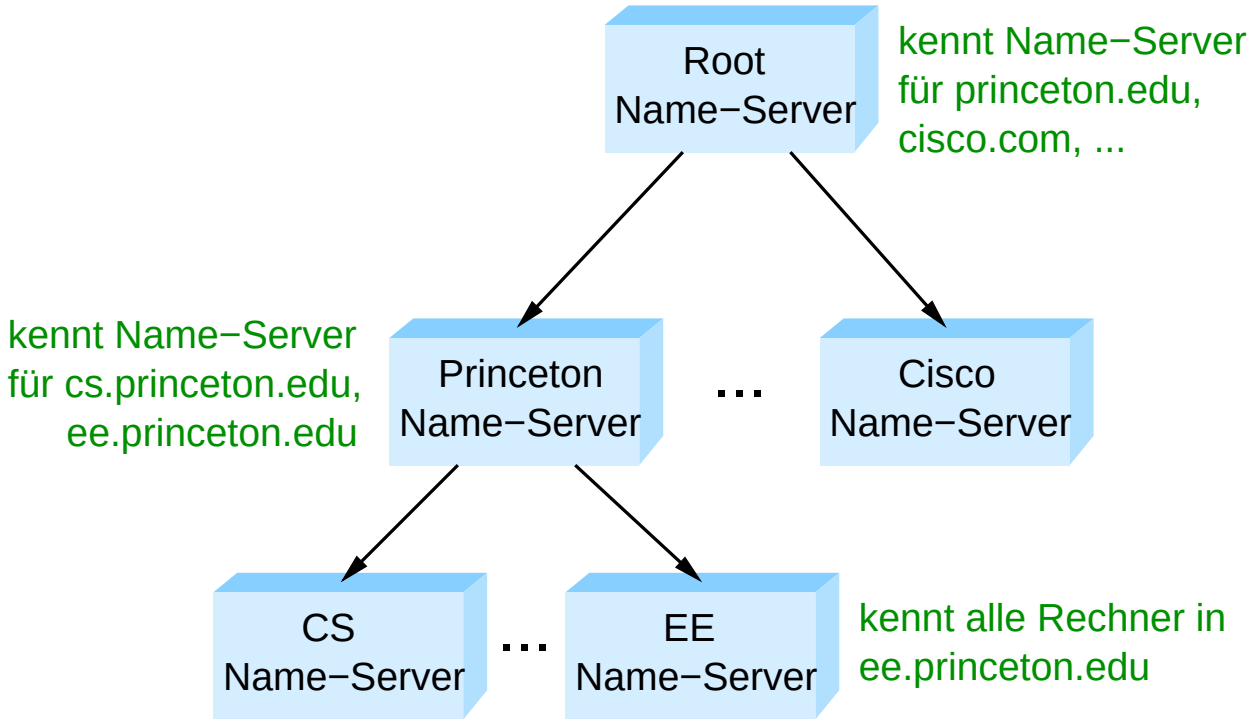


DNS realisiert einen hierarchischen Namensraum



- Für jede Zone ist ein Name-Server zuständig
  - Hierarchie von Name-Servern

Hierarchie von Name-Servern bei DNS



★★

Hierarchische Namensauflösung mit DNS

- Jeder Rechner kennt nur seinen lokalen Name-Server
- Lokale Name-Server
  - lösen lokale Namen auf
  - kennen Root-Name-Server (über Konfigurationsdatei)
- Server und Hosts führen Cache mit bereits aufgelösten Namens-Adreß-Paaren
  - begrenzte Lebensdauer der Einträge
- Alternativen, falls Zuordnung nicht im Cache:
  - Nachfrage bei anderem Servern (*recursive query*)
  - antworte mit Adresse eines anderen Servers, erneute Anfrage des Clients (*nonrecursive query*)

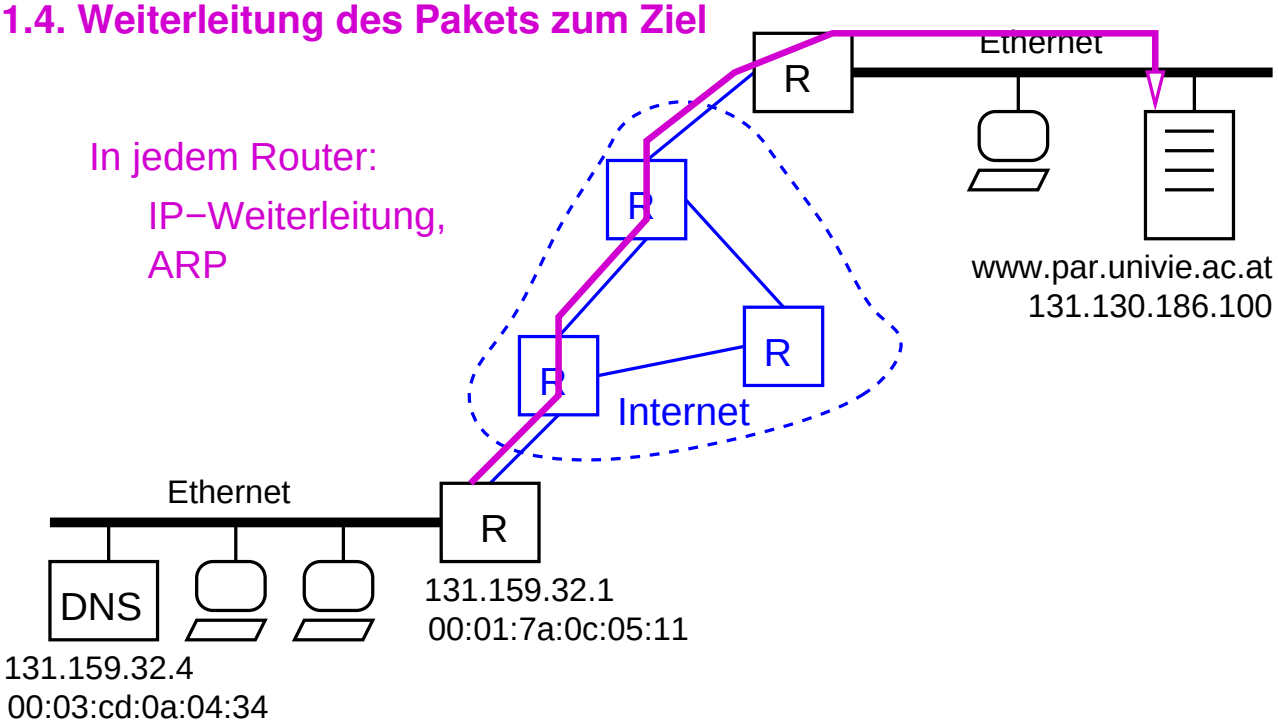
9.4 Zusammenfassung

(Animierte Folie)

★★

Beispiel: Ablauf einer Web-Server-Anfrage

1.4. Weiterleitung des Pakets zum Ziel



9.4 Zusammenfassung ...

- SMTP: textbasiert, speichervermittelnd
- HTTP: ebenfalls textbasiert
  - wie viele andere Anwendungsprotokolle im Internet
- DNS: Umsetzung von Rechnernamen auf IP-Adressen
  - Hierarchischer Namensraum + Server-Hierarchie

Nächste Lektion:

- Netzwerksicherheit

Rechnernetze I

SoSe 2024

10 Netzwerksicherheit

Inhalt

- Sicherheitsanforderungen
- Sicherheitsprobleme der Internet-Protokolle
- Kryptographische Grundlagen
- Sicherheitsmechanismen für Protokolle
- Beispiele sicherer Protokolle
- Firewalls

- Peterson, Kap. 8.1, 8.2, 8.3.1, 8.3.3, 8.4
- CCNA, Kap. 11.2

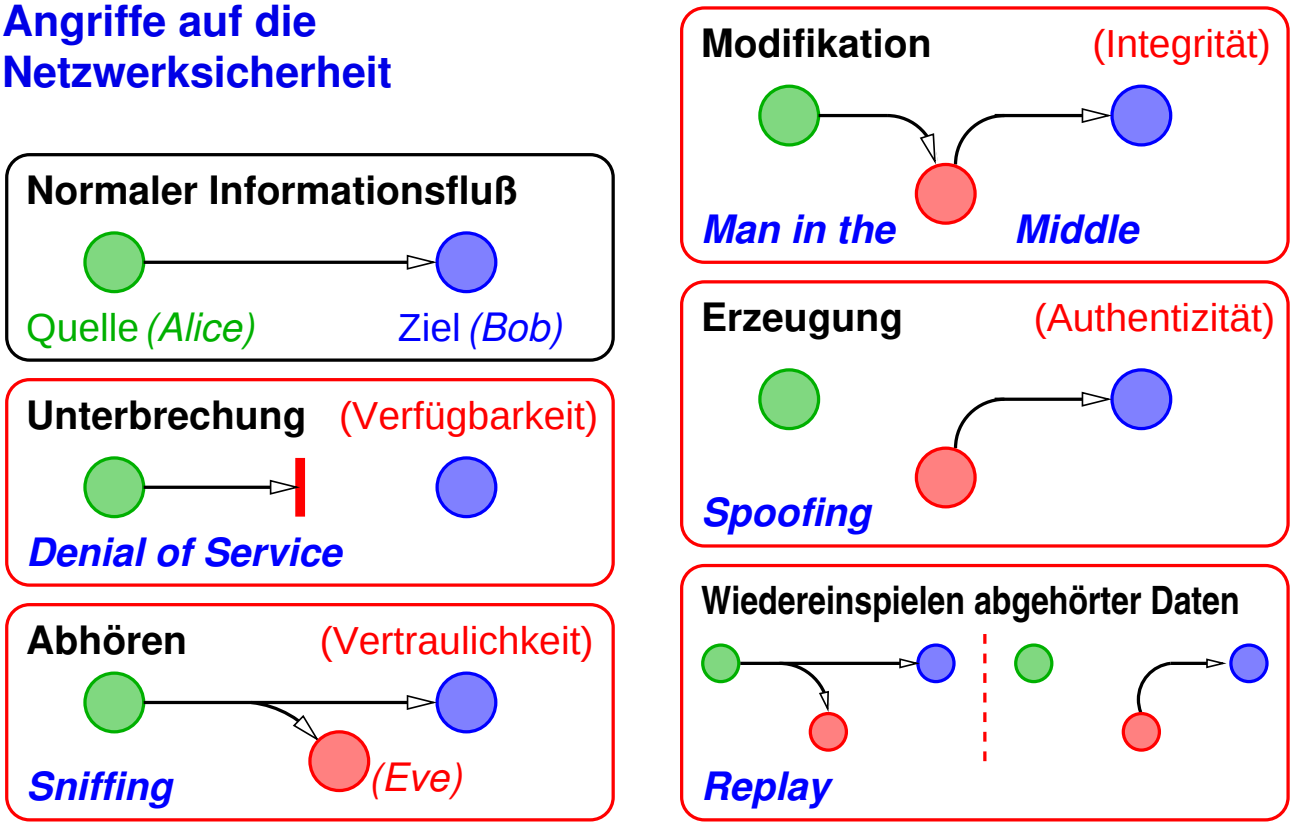
10.1 Sicherheitsanforderungen

- In Netzwerken wird persönliche / wertvolle / vertrauliche Information übermittelt
  - Information sollte nur Berechtigen bekannt werden!
  - Authentizität der Information?
- Wachsende Bedeutung der Netzwerksicherheit wegen
  - steigender Vernetzung
    - höheres Angriffspotential
  - neuer Einsatzgebiete
    - z.B. e-Business: elektronische Zahlung / Verträge

Allgemeine Sicherheitsanforderungen

- **(Informations-)Vertraulichkeit (*confidentiality*)**
  - Schutz vor unautorisierter Informationsgewinnung
- **(Daten-)Integrität (*integrity*)**
  - Schutz vor unautorisierter Veränderung von Daten
- **(Nachrichten-)Authentizität (*message authenticity*)**
  - Urheber der Daten kann korrekt identifiziert werden
- **Verbindlichkeit (*nonrepudiation*)**
  - Handlungen können nicht abgestritten werden
- **Verfügbarkeit (*availability*)** von Diensten
- **Anonymität** der Kommunikationspartner

Angriffe auf die Netzwerksicherheit



10.1 Sicherheitsanforderungen ...

★★★

Konkret: Alice sendet eine Nachricht an Bob

- **Vertraulichkeit**: niemand außer Alice und Bob erfahren den Inhalt der Nachricht
- **Integrität**: Bob kann sich (nach entsprechender Prüfung!) sicher sein, daß die Nachricht während der Übertragung nicht (absichtlich) verfälscht wurde
- **Authentizität**: Bob kann sich (nach entsprechender Prüfung!) sicher sein, daß die Nachricht von Alice gesendet wurde
- **Verbindlichkeit**: Alice kann nicht bestreiten, die Nachricht verfaßt zu haben
  D.h. Bob kann Dritten gegenüber **beweisen**, daß die Nachricht von Alice gesendet wurde
- Im Folgenden: Beschränkung auf diese vier Anforderungen

10.2 Sicherheitsprobleme des Internets

Ein Problem des IP-Protokolls: IP-Spoofing

- Viele IP-basierte Protokolle vertrau(t)en der Absenderadresse
  - z.B. UNIX-Dienste rsh, rcp, rlogin: (*)
    - Festlegung von *Trusted Hosts*
    - Zugriff von *Trusted Host* aus auch ohne Paßwort
- Aber: Angreifer kann IP-Pakete mit beliebiger (falscher) Absenderadresse versenden
  - z.B. um vorzutäuschen, ein *Trusted Host* zu sein
- **Problem**: fehlende Authentifizierung der Pakete in IPv4

(*) Inzwischen nicht mehr in Verwendung!

Ein Problem des IP-Protokolls: IP-Spoofing ...

- IP-Spoofing ist Basis vieler anderer Angriffe
- Gegenmaßnahmen:
  - nicht auf Senderadresse vertrauen
  - Router-Konfiguration: *Source Address Validation*
    - Prüfen, ob Paket mit angegebener Senderadresse aus dem jeweiligen Subnetz kommen kann
  - IPsec (neuer Internet-Standard):
    sichere Authentifizierung des Senders

Anmerkungen zu Folie 336:

Ein Grundproblem der Internet-(Un)sicherheit ist die Tatsache, daß der Absenderadresse von IP-Paketen nicht vertraut werden kann (der Sender kann die Quell-IP-Adresse Aufwand beliebig angeben, dazu ist praktisch kein Aufwand notwendig).

Daher ist es wichtig, in Protokollen nicht auf die Authentizität der Absenderadresse zu vertrauen.

Eine weitere Maßnahme, um das Problem des *IP-Spoofings* zu adressieren, ist die Verwendung von *Source-Address-Validation* in Routern. Dabei prüft der Router, ob ein IP-Paket, das er über eine bestimmte Schnittstelle S empfangen hat, tatsächlich aus diesem Netz kommen kann. Zur Überprüfung bestimmt der Router über die Routing-Tabelle, über welche Schnittstelle S' er ein Paket an den Rechner mit der angegebenen **Quell**adresse senden würde. Nur wenn S' mit S übereinstimmt, leitet der Router das Paket weiter. Empfängt der Router beispielsweise über eine Schnittstelle, die direkt an ein LAN mit der Netzadresse 141.99.0.0/16 angeschlossen ist, ein Paket mit Quelladresse 131.159.79.3, so weiß er, daß dieses Paket eine falsche Quelladresse besitzt und verwirft es.

Eine sichere Authentifizierung von IP-Paketen ist mit *Secure IP* (IPsec) möglich.

Meist wird heute die Sicherheit aber nicht auf Ebene von IP realisiert, sondern in den darüberliegenden (Anwendungs-)Protokollen.

Ein Problem durch Programmierfehler: *Ping of death*

- Fehler in der Implementierung des `ping`-Kommandos unter Windows 95:
  - `ping -l 65510 my.computer.de` sendet ein fragmentiertes IP-Paket der Länge 65538
- Fehler in (alten Versionen) fast aller Betriebssysteme:
  - Pufferüberlauf im Betriebssystemkern beim Zusammenbau des Pakets
  - Absturz des Systems, *Reboot*, ...
- **Problem**: fehlende Validierung der Eingabe

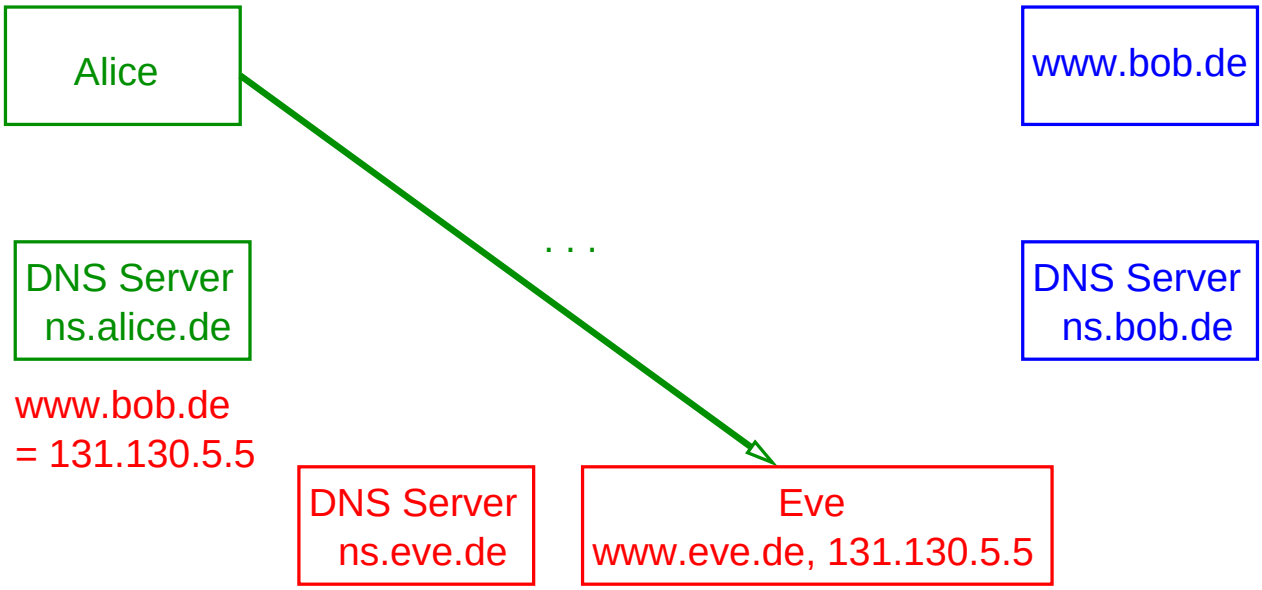
Ein Problem des DNS: DNS-Spoofing

- Angreifer kann falsche Zuordnung zwischen Hostnamen und IP-Adresse in DNS-Servern installieren
  - Zugriffe auf diesen Host werden z.B. auf Rechner des Angreifers umgeleitet (= *Man-in-the-Middle* Attacke)
  - z.B. gefälschte Web-Sites, Ausspionieren von Kreditkarteninfo, Paßworten, ...
- **Problem:** keine Authentizierung
- Schadensbegrenzung: keine *recursive queries* zulassen
  - nur DNS-Cache eines Rechners kann infiziert werden
- Lösungen: TSIG, DNSSEC (IETF Standards)

(Animierte Folie)

Angriff auf DNS: DNS-*Spoofing*

3. Alice kontaktiert Eve und glaubt es wäre Bob!

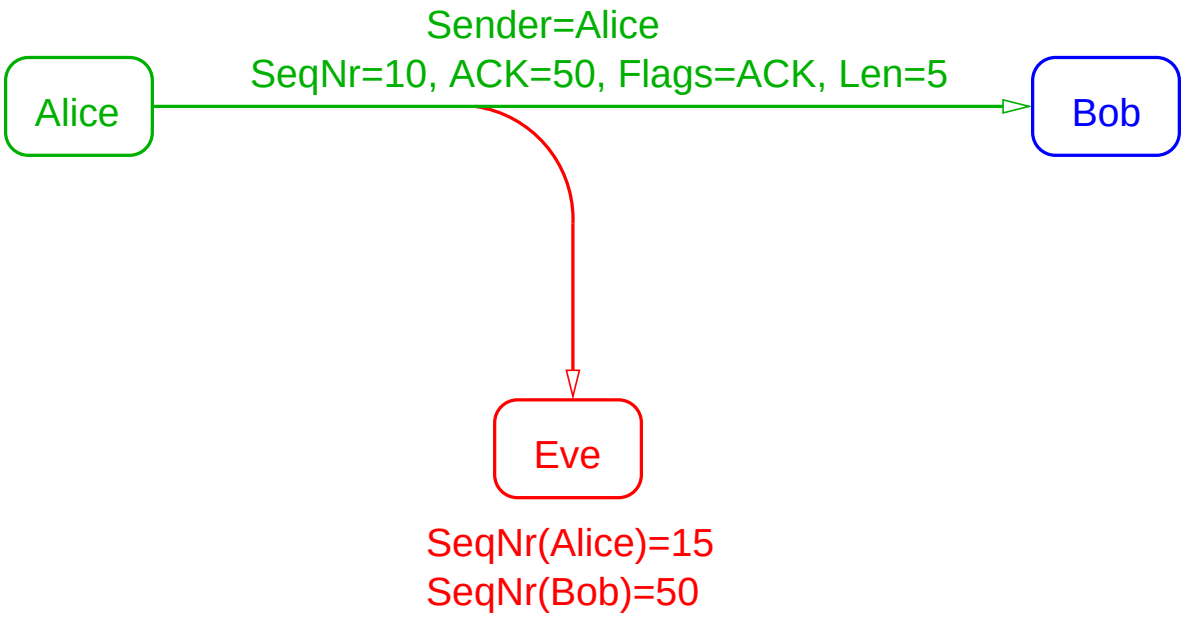


Ein Problem des TCP: Hijacking

- „Feindliche Übernahme“ einer offenen TCP-Verbindung
  - z.B. **nach erfolgter Authentifizierung von Alice**!
- Angriff:
  1. Senden eines gefälschten RST-Pakets (Verbindungsabbruch) an Alice
  2. Senden gefälschter Pakete an Bob, mit geschätzten oder abgehörten Sequenznummern
- Setzt i.d.R. Abhörmöglichkeit (z.B. Zugang zu lokalem Ethernet) voraus
  - Sequenznummern zählen die übertragenen Bytes

(Animierte Folie)

Ein Problem des TCP: Hijacking ...



Weitere problematische Protokolle

- UDP
  - **Problem**: fehlende Authentifizierung
  - „Fälschen“ von UDP-Paketen viel einfacher als bei TCP
  - Einige wichtige Dienste (DNS, NFS) basieren auf UDP
- ICMP
  - **Problem**: fehlende Authentifizierung
  - erlaubt Abbrechen, Behindern und Umleiten von Verbindungen
- ARP
  - **Problem**: keine Authentizierung
  - ARP ist zustandslos, akzeptiert Antwort von beliebigem Sender
  - Angreifer im lokalen Netz kann durch falsche ARP-Antwort Pakete zu sich umleiten (ARP Spoofing)

Weitere problematische Protokolle ...

- NFS (*Network File System*)
  - **Problem**: Authentifizierung nur über Hostname/User-ID
  - **Problem**: ungeschützte Übertragung von Dateiinhalten und -handles
- NIS (*Network Information System*)
  - Zentrale Paßwortdatei für Rechner eines Netzes
  - **Problem**: keine Authentifizierung des Servers
  - **Problem**: Zugriffskontrolle nur durch Domänen-Namen

Weitere problematische Protokolle ...

- `rsh`, `rcp`, `rlogin`, `telnet`, FTP (*File Transfer Protocol*)
  - **Problem**: Authentifizierung z.T. nur über Hostname/User-ID
  - **Problem**: ungeschützte Übertragung (incl. Paßworte!)
- HTTP (*HyperText Transport Protocol*)
  - **Problem**: keine Authentifizierung des Servers
  - **Problem**: ungeschützte Übertragung (auch Paßworte!)
- SMTP (*Simple Mail Transport Protocol*)
  - **Problem**: keine Authentifizierung des Absenders
  - **Problem**: Übertragung / Zwischenspeicherung im Klartext

★★

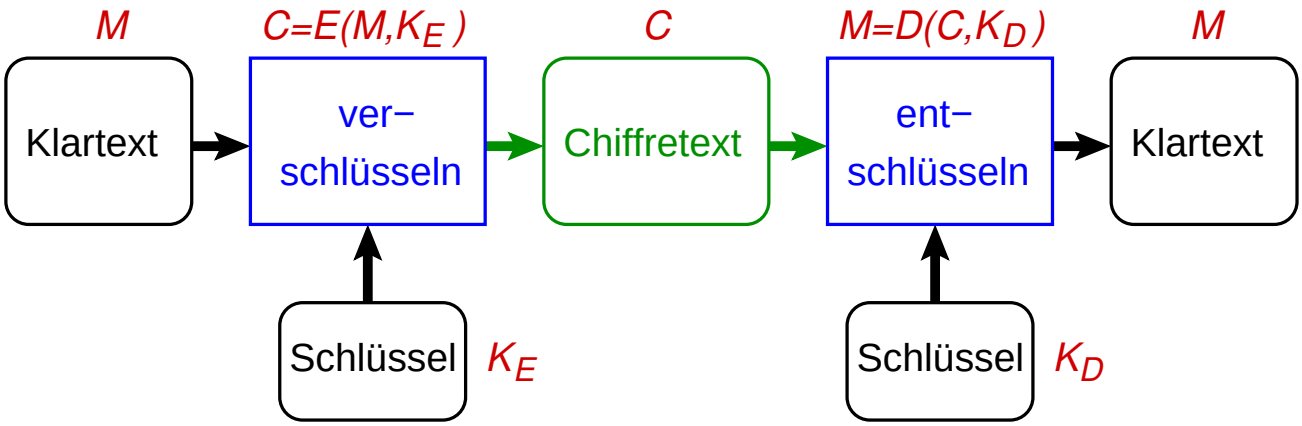
Fazit

- Die Standard-Internet-Protokolle (u.a. IP, TCP, DNS, ARP, NFS, HTTP, SMTP) erfüllen **keine** der in 10.1 genannten Sicherheitsanforderungen
- Hauptprobleme:
  - öffentliche Netze prinzipiell abhörbar
  - fehlende / unzureichende Authentifizierung
- Abhilfe:
  - sichere Protokolle in der Anwendungsschicht:
    - SSL/TLS (HTTPS, FTPS), S/MIME, PGP, SSH, ...
  - sicheres IP-Protokoll (IPsec, siehe Rechnernetze II)
- Basis: kryptographische Verfahren

10.3 Kryptographische Grundlagen

★

Grundprinzip der Verschlüsselung:



- Symmetrische Verschlüsselungsverfahren
  - $K_E = K_D = K$ = gemeinsamer geheimer Schlüssel
- Asymmetrische Verschlüsselungsverfahren
  - K_E = öffentlicher, K_D = privater Schlüssel

★

Anforderungen an Verschlüsselungsverfahren:

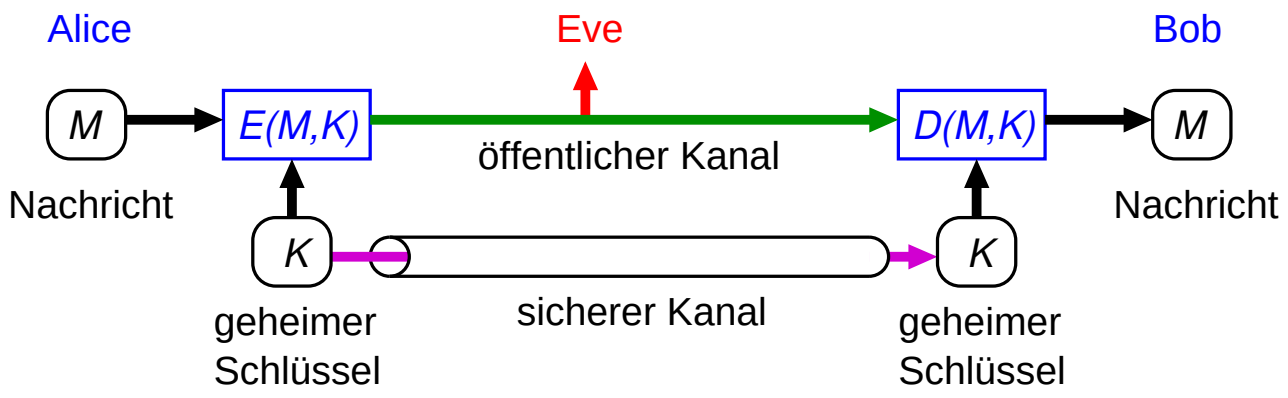
- Nur der Besitzer des geheimen bzw. privaten Schlüssels kann den Chiffretext entschlüsseln
- Sicherheit basiert nicht auf Geheimhaltung der Algorithmen

Mögliche Angriffe:

- Klartext-Angriff: Klartext + Chiffretext $\Rightarrow$ Schlüssel
- Im Idealfall: alle Schlüssel müssen durchprobiert werden
  - Schlüssel müssen lang genug sein!
- Bei asymmetrischen Verfahren auch effizientere Angriffe
  - Berechnung von K_D aus K_E ($\Rightarrow$ längere Schlüssel nötig)

★★★

10.3.1 Symmetrische Verschlüsselung



- Symmetrische Verschlüsselung ist sehr effizient realisierbar
- Schlüssel sind relativ kurz (heute typisch 128-256 Bit)
- Problem: Austausch des Schlüssels K

10.3.1 Symmetrische Verschlüsselung ...

Beispiele symmetrischer Verschlüsselungsverfahren:

- **DES**: veraltet, Schlüssel nur 56 Bit lang
- **Triple-DES**: veraltet, dreifache Anwendung von DES
  - effektive Schlüssellänge: 112 Bit
- **AES**: Nachfolger von DES, 128-256 Bit Schlüssel
  - vom amerikanischen NIST standardisiert
  - in praktisch allen sicheren Protokollen verwendet / unterstützt
- **IDEA**: 128 Bit Schlüssel
  - freies Verfahren, benutzt z.B. in PGP

10.3.1 Symmetrische Verschlüsselung ...

- Zwei verschiedene Arten symmetrischer Verfahren:
  - **Blockchiffren** ver-/entschlüsseln Blöcke von Zeichen mit einer fest vorgegebenen Größe (z.B. 128 Bit bei AES)
    - zu kleine Blöcke müssen aufgefüllt werden (*Padding*)
    - für Nachrichten beliebiger Größe: zusätzliche Betriebsmodi für den Umgang mit einer Folge von Blöcken
  - **Stromchiffren** ver-/entschlüsseln in einem Zeichenstrom jedes Zeichen einzeln
    - typisch: EXOR-Verknüpfung mit Pseudozufallsfolge
- Grundoperationen symmetrischer Chiffren:
  - Substitution: ersetze Bitgruppen systematisch durch andere Bitgruppen
  - Permutation: vertausche Bitgruppen nach festgelegtem Schema

Anmerkungen zu Folie 350:

- Für eine Stromchiffre wird ein kryptographischer Pseudozufallszahlengenerator (PRNG) benötigt, der den symmetrischen Schlüssel als Initialisierungswert verwendet. Dieser lässt sich durch spezielle Betriebsarten einer Blockchiffre realisieren. Die Ausgabe des PRNGs wird EXOR mit dem Datenstrom verknüpft.

  Zum Entschlüsseln verknüpft man den verschlüsselten Datenstrom nochmals mit der Ausgabe des PRNGs (mit dem selben symmetrischen Schlüssel als Initialisierungswert). Da $M \oplus X \oplus X = M$ erhält man dabei wieder den Klartext.

- Verwendet man eine **echte** Zufalls-Bitfolge als Schlüssel, kann man mathematisch beweisen, daß ohne Kenntnis des Schlüsselstroms der Chiffre-Text nicht entschlüsselt werden kann, da er nich von einer Zufalls-Bitfolge zu unterscheiden ist. Der Nachteil dieser sog. Vernam-Chiffre (*One Time Pad*) ist, daß der Schlüssel dieselbe Länge hat wie der zu übertragende Klartext, und daß er nur einmal verwendet werden darf.

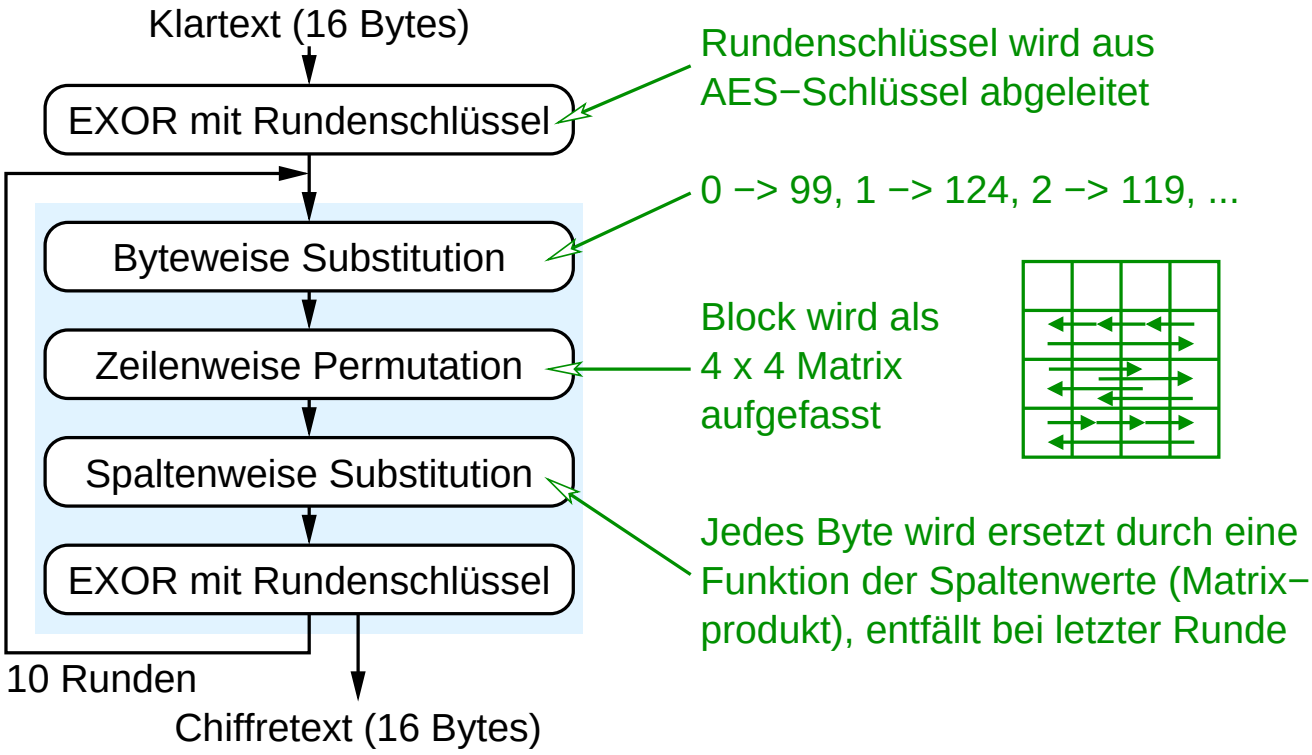
- Ein einfaches Beispiel für die Substitution ist die klassische Caesar-Chiffre, die jeden Buchstaben durch den im Alphabet 3 Plätzen weiter hinten liegenden Buchstaben ersetzt, z.B.:

  Klartext: `JULIUS CAESAR IMPERATOR`
  Chiffretext: `MXOLXV FDHVDU LPSHUDWRU`

- Ein einfaches Beispiel für eine Permutation wäre es, erst die Zeichen an den geraden Positionen und dann die an den ungeraden Positionen zu nehmen, z.B.:

```
Klartext:    JULIUS CAESAR IMPERATOR
             J L U _ A S R I P R T R
              U I S C E A _ M E A O
Chiffretext: JLU ASRIPRTRUISCEA MEAO
```

Grobstrutkur von AES (Verschlüsselung)

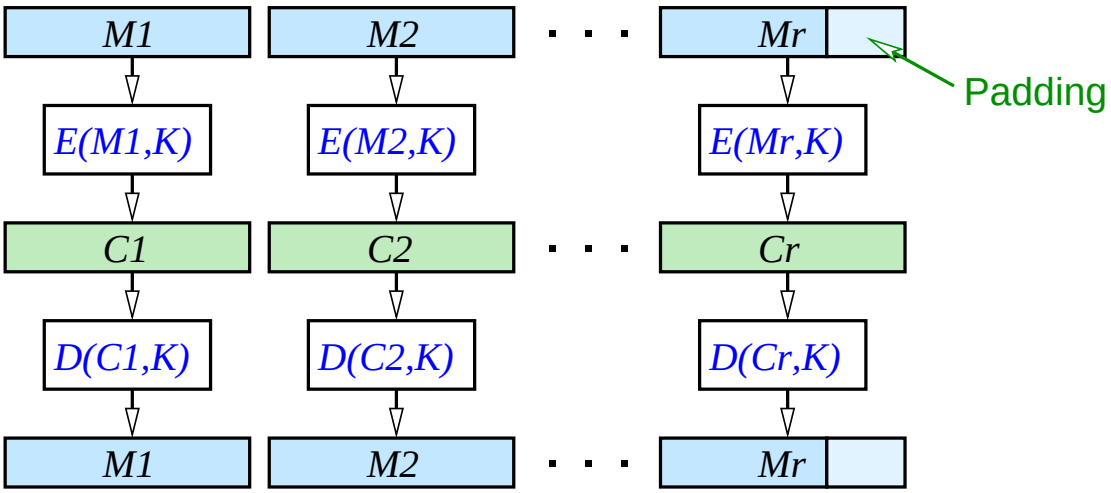


Anmerkungen zu Folie 351:

- Zur Entschlüsselung werden jeweils die inversen Substitutionen und Permutationen in umgekehrter Reihenfolge angewendet.
- Eine detaillierte Beschreibung von AES finden Sie in William Stallings: „*Cryptography and Network Security - Principles and Practices*“, Prentice Hall, 2003.

Betriebsarten von Blockchiffren: *Electronic Code Book* (ECB)

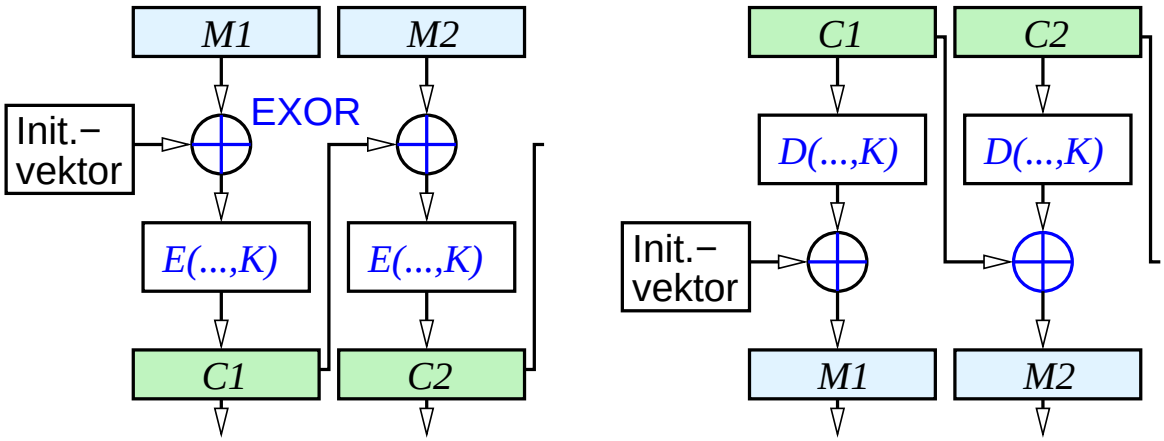
- Klartext wird in Blöcke (z.B. 128 Bit) aufgeteilt, ggf. mit Padding
- Blöcke werden unabhängig voneinander ver-/entschlüsselt



- Kein Schutz vor Löschung / Wiedereinspielung von Blöcken

Betriebsarten von Blockchiffren: *Cipher Block Chaining* (CBC)

Verschlüsselung **Entschlüsselung**



- Gleiche Klartextblöcke ⇒ verschiedene Chiffretextblöcke
- Fehlerfortpflanzung: Vorteil und Nachteil

Anmerkungen zu Folie 353:

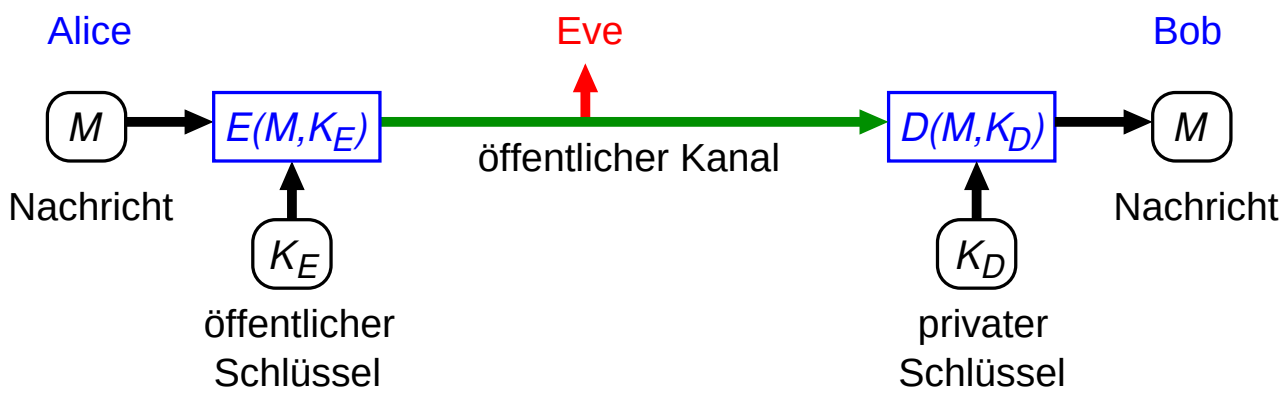
Es gibt weitere Betriebsmodi, insbesondere *Cipher Feedback Mode* (CFB), *Output Feedback Mode* (OFB) und *Counter Mode* (CTR), mit deren Hilfe aus einer Blockchiffre eine Stromchiffre gemacht werden kann.

WPA-2 beim WLAN verwendet z.B. AES im *Counter Mode* zur Verschlüsselung.

10.3 Kryptographische Grundlagen ...

★★★

10.3.2 Asymmetrische Verschlüsselung



- Bob berechnet K_E aus K_D und veröffentlicht K_E
  - Problem: Authentizität von K_E
- Weniger effizient als symmetrische Verfahren
- Längere Schlüssel nötig (heute typisch 2048-4096 Bit)

Basis asymmetrischer Verfahren:

- Einwegfunktionen (*on-way functions*):
  - Berechnung von $y = f(x)$ einfach
  - Berechnung von $x = f^{-1}(y)$ praktisch unmöglich
- Beispiele:
  - diskreter Logarithmus: $f(x) = a^x \bmod p$, p prim
    - Verwendung z.B. im Diffie-Hellman-Schlüsselaustausch
    - *Elliptic Curve Cryptography*: verwendet „Elliptische Kurven über endlichen Körpern“ als algebraische Struktur
      - erlaubt deutlich kürzere Schlüssel
  - Multiplikation großer Primzahlen vs. Faktorisierung
    - Verwendung z.B. in RSA
- Schwierigkeit der Berechnung der Umkehrfunktion nicht bewiesen

Diffie-Hellman-Schlüsselaustausch

- Frage: Wie können Alice und Bob über einen öffentlichen Kanal einen gemeinsamen geheimen Schlüssel K aushandeln?
- Gegeben sind öffentliche Elemente
  - q: Primzahl
  - a: primitive Wurzel von q ($a^n \bmod q$ durchläuft $1 \ldots q-1$)

Alice		Bob
wähle $X_A < q$		wähle $X_B < q$
berechne $Y_A = a^{X_A} \bmod q$	$\xrightarrow{Y_A}$	
	$\xleftarrow{Y_B}$	berechne $Y_B = a^{X_B} \bmod q$
berechne $K = (Y_B)^{X_A} \bmod q$		berechne $K = (Y_A)^{X_B} \bmod q$

- Problem: keine Authentifizierung!

RSA (Rivest, Shamir, Adleman)

- Schlüsselgenerierung
  - wähle große Primzahlen p und q, berechne **Modul** $n = p \cdot q$
    - Euler'sche Zahl $\varphi(n) = (p-1) \cdot (q-1)$
  - wähle e mit $1 < e < n$ und $ggT(\varphi(n), e) = 1$
  - wähle d so, daß $e \cdot d$ mod $\varphi(n) = 1$
  - öffentlicher Schlüssel $K_E = (e, n)$
  - privater Schlüssel $K_D = (d, n)$
- Verschlüsseln und Entschlüsseln
  - Klartextblock M als binärcodierte Zahl auffassen: $M < n$
  - Verschlüsseln: $C = E(M, K_E) = M^e$ mod n
  - Entschlüsseln: $M = D(C, K_D) = C^d$ mod n

Anmerkungen zu Folie 357:

- Die Wahl der Primzahlen erfolgt typischerweise so, daß zunächst eine Zufallszahl bestimmt wird, die anschließend durch (probabilistische) Primzahltests, z.B. Rabin-Milner, überprüft wird. Ggf. wird eine neue Zufallszahl gewählt.
- Die Euler'sche Zahl $\varphi(n)$ ist die Anzahl der zu n teilerfremden Zahlen im Intervall $[1, n[$
  - für eine Primzahl p gilt $\varphi(p) = p - 1$
  - für das Produkt zweier Primzahlen p und q gilt $\varphi(p \cdot q) = (p-1) \cdot (q-1)$
- Die Berechnung von d (als multiplikative Inverse von e) erfolgt über einen erweiterten Euklidischen Algorithmus. Dieser testet gleichzeitig, ob $ggT(\varphi(n), e) = 1$.

RSA: Beispiel zum Nachrechnen

- Schlüsselerzeugung
  - $p = 3,\ q = 11$
  - $n = p \cdot q = 33,\ \varphi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$
  - $e = 3$, damit $ggT(\varphi(n), e) = ggT(20, 3) = 1$
  - Wähle d so, daß $e \cdot d \equiv 1$ mod $\varphi(n),\ 3 \cdot d \equiv 1$ mod $20,\ d = 7$
  - Öffentlicher Schlüssel $K_E = (3, 33)$, privater $K_D = (7, 33)$
- Verschlüsseln und Entschlüsseln
  - Klartextnachricht $M = 5$
  - $C = E(M, K_E) = 5^3$ mod $33 = 125$ mod $33 = 26$
  - $D(C, K_E) = 26^7$ mod $33 = 8031810176$ mod $33 = 5 = M$

Anmerkungen zu Folie 358 (zur Korrektheit des RSA-Verfahrens):

Mathematische Grundlagen:

- Definition der Restklassenarithmetik:
  - $a \equiv b \bmod n \quad \Leftrightarrow \quad a \bmod n = b \quad \Leftrightarrow \quad \exists k : a = b + k \cdot n$
- Satz von Euler:
  - $ggT(M, n) = 1 \quad \Rightarrow \quad M^{\varphi(n)} \equiv 1 \bmod n$

Zu zeigen: $D(E(M, K_E), K_D) = M$

Beweis

- Nach Definition von E und D: $D(E(M, K_E), K_D) = M^{e \cdot d} \bmod n$
- Da $e \cdot d \equiv 1 \bmod \varphi(n)$, gilt $\exists k : e \cdot d = k \cdot \varphi(n) + 1$
  - damit: $M^{e \cdot d} \bmod n = M^{k \cdot \varphi(n)+1} \bmod n = M \cdot M^{k \cdot \varphi(n)} \bmod n$
- Falls $ggT(M, n) = 1$ gilt mit dem Satz von Euler:
  - $M \cdot M^{k \cdot \varphi(n)} \bmod n = M$

- Sonst: M ist entweder ein Vielfaches von p oder ein Vielfaches von q (beides zusammen geht nicht, da $M < n = p \cdot q$)
  - falls M ein Vielfaches von p ist:
    - dann $ggT(M, q) = 1$ und $\exists i : M = i \cdot p$
    - mit Satz von Euler: $M^{\varphi(q)} \equiv 1 \bmod q$
    - damit auch $M^{k \cdot \varphi(p) \cdot \varphi(q)} = M^{k \cdot \varphi(n)} \equiv 1 \bmod q$
    - also $\exists k : M^{k \cdot \varphi(n)} = 1 + l \cdot q$
    - damit ist $M \cdot M^{k \cdot \varphi(n)} = M \cdot (1 + l \cdot q) = M + M \cdot l \cdot q = M + i \cdot p \cdot l \cdot q = M + i \cdot l \cdot n$
    - das bedeutet $M \cdot M^{k \cdot \varphi(n)} \bmod n = M$
  - falls M ein Vielfaches von q ist: analog

358-2

10.3 Kryptographische Grundlagen ...

★★★

10.3.3 Kryptographische Hashfunktionen (*Message Digest*)

- Analog einer normalen Hashfunktion:
  - Nachricht wird auf einen Wert fester Größe abgebildet
- Zusätzliche Eigenschaft: **Kollisionsresistenz**
  - zu Nachricht *x* kann (in vernünftiger Zeit) keine andere Nachricht *y* mit gleichem Hashwert gefunden werden
- Einsatz zur Sicherung der Integrität
  - „kryptographische Prüfsumme“
- Beispiele
  - MD5 (*Message Digest, Version 5*): 128 Bit Hashwert, unsicher
  - SHA-1 (*Secure Hash Algorithm 1*): 160 Bit Hashwert, unsicher
  - SHA-2 / SHA-3: 224 - 512 Bit Hashwert

Anmerkungen zu Folie 359:

Genauer unterscheidet man zwischen „schwacher“ und „starker“ Kollisionsresistenz.

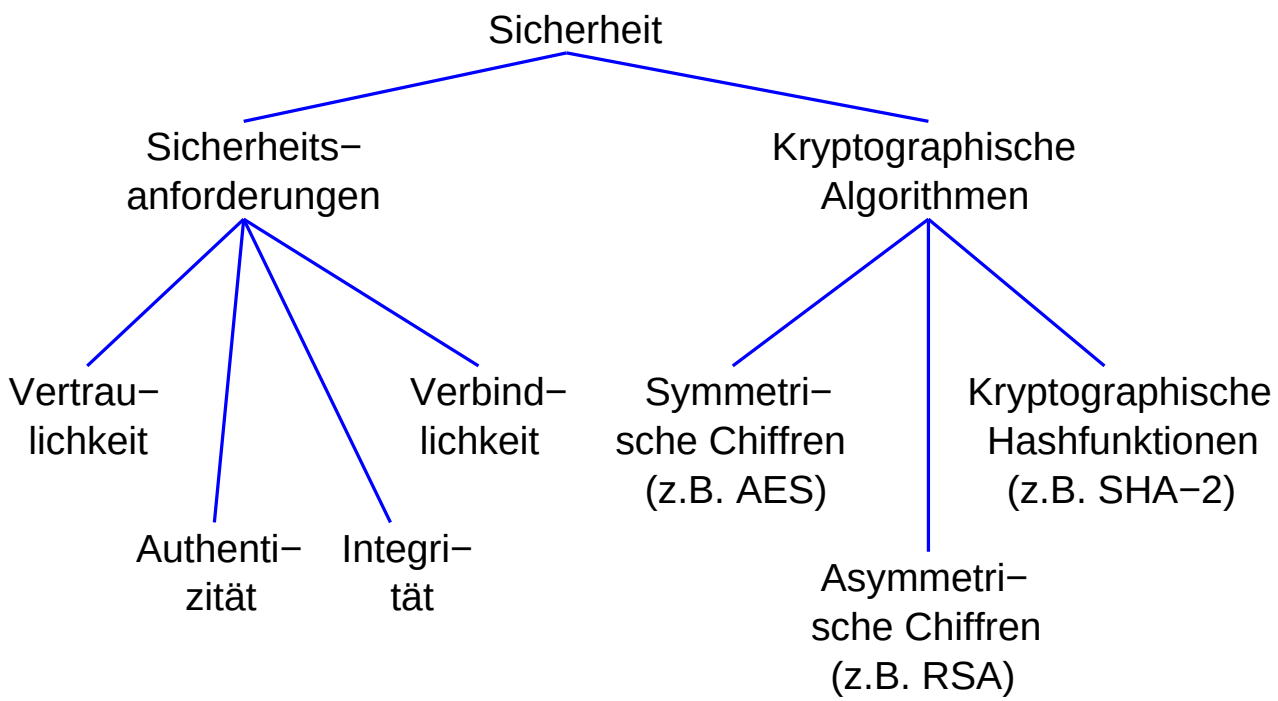
- Die hier gegebene Definition ist die für die schwache Kollisionsresistenz.
- Starke Kollisionsresistenz ist gegeben, wenn es (praktisch) unmöglich ist, zwei Nachrichten x und y zu finden, die denselben Hashwert produzieren.
- Um starke Kollisionsresistenz zu erreichen, sind in der Regel doppelt so lange Hashwerte erforderlich (siehe „Geburtstagsparadoxon“).

359-1

10.3 Kryptographische Grundlagen ...

★★★

10.3.4 Zusammenfassung



10.3.4 Zusammenfassung ...

★★★

Was leistet die reine Verschlüsselung von Nachrichten?

- Vertraulichkeit: **ja**
- Integrität: **bedingt**
  - nur, wenn Klartext genügend Redundanz aufweist
  - ⇒ Verwendung von *Message Digests*
- Nachrichtenauthentizität:
  - **nein** bei asymmetrischen Verfahren: K_E öffentlich!
  - **bedingt** bei symmetrischer Verschlüsselung
    - nur mit gesicherter Integrität und Schutz vor *Replay*
- Verbindlichkeit: **nein**
- Schutz vor Replay: **nein**
  - ⇒ Transaktionszähler im Klartext + Integrität sichern

10.4 Sicherheitsmechanismen

★

- Kryptographische Algorithmen sind nur Bausteine für die Netzwerksicherheit
- Zusätzlich benötigt: Mechanismen und Protokolle
- Einige Sicherheitsaufgaben:
  - Authentifizierung
    - von Kommunikationspartnern
      - „wer ist mein Gegenüber?“
    - von Nachrichten
      - „stammt die Nachricht wirklich vom Absender?“
  - Sicherung der Integrität von Nachrichten
  - Verbindlichkeit
  - Verteilung öffentlicher Schlüssel

★

Partner-Authentifizierung

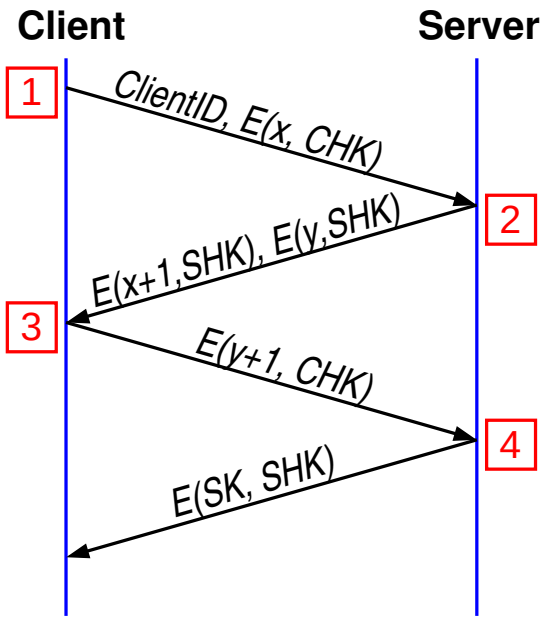
- Kommunikationspartner identifizieren sich gegenseitig
  - Beispiel: File-Server
    - Server verlangt ID des Clients zur Prüfung der Schreib-/Leserechte
    - Client verlangt ID des Servers zum Lesen/Schreiben sensibler Daten
- Manchmal auch nur einseitige Authentifizierung
  - Beispiel: WWW-Server
    - Client verlangt ID des Servers zur Übertragung wichtiger / vertraulicher Daten

★★

Partner-Authentifizierung mit Drei-Wege-Handshake

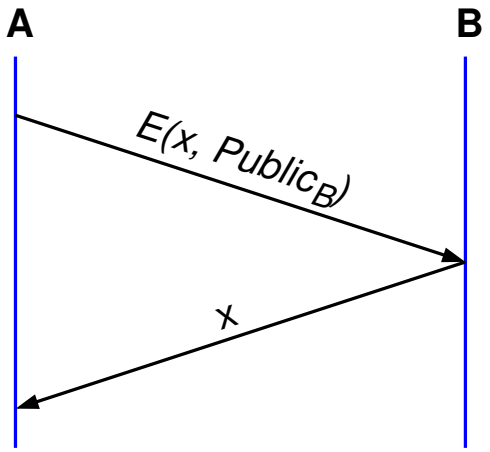
- Server kennt Schlüssel des Clients (z.B. Paßwort bei login)

1. Client sendet *ClientID* und verschlüsselte Zufallszahl x (CHK: *Client Handshake Key*)
2. Server sucht den zu *ClientID* gehörigen Schlüssel *SHK*, sendet *x+1* und Zufallszahl *y*
3. Server ist authentifiziert (*x+1*)
4. Client ist authentifiziert (*y+1*), Server sendet *Session Key SK* für weitere Kommunikation



★★

Partner-Authentifizierung über asymmetrische Chiffre



- Einseitige Authentifizierung von *B*
  - ggf. authentifiziert sich *A* ebenso (≈ 3-Wege-Handshake)
- $Public_B$ nicht zum Verschlüsseln verwenden!

★★

Sicherung der Nachrichtenintegrität und -authentizität

- Integrität: Kein Dritter soll Nachricht verfälschen können
  - setzt sinnvollerweise Nachrichten-Authentizität voraus
- Bei Übertragung mit symmetrischer Verschlüsselung:
  - kryptographischen Hashwert *H(M)* an Klartext *M* anfügen und verschlüsseln
  - bei Modifikation des Chiffretexts paßt die Nachricht nicht mehr zum Hashwert
  - kein Angreifer kann neuen Hashwert berechnen / verschlüsseln
  - Nachrichten-Authentizität (bis auf Replay) durch symmetrische Chiffre sichergestellt
    - Replay-Schutz: Transaktionszähler / Zeitstempel in *M*

★★

Sicherung der Nachrichtenintegrität und -authentizität ...

- Bei asymmetrischer Verschlüsselung:
  - Hash-Wert allein nützt nichts, da Nachrichten-Authentizität nicht sichergestellt ist
- Bei unverschlüsselter Übertragung (oft sind Daten nicht vertraulich, aber ihre Integrität wichtig):
  - Hash-Wert stellt Integrität nicht sicher, da jeder nach einer Modifikation der Nachricht den neuen Hash-Wert berechnen kann
- Lösungen:
  - kryptographischer Hashwert mit geheimem Schlüssel
  - digitale Signatur

★★★

Hashwert mit geheimem Schlüssel

- Einbeziehen eines (gemeinsamen) geheimen Schlüssels *K* in den Hashwert:
  - füge *H(M + K)* an Nachricht M an (*+* = Konkatenation)
- Sichert auch Nachrichten-Authentizität (bis auf Replay)
  - kein Dritter kann *H(M + K)* korrekt berechnen
  - Replay-Schutz: Transaktionszähler / Zeitstempel in *M*
- Sichert nicht Verbindlichkeit
  - Empfänger kann *H(M + K)* berechnen
- Beispiel: *HMAC-SHA-256*

Anmerkungen zu Folie 368:

Tatsächlich wird der Schlüssel nicht einfach an die Nachricht angefügt, sondern komplexer mit dieser verknüpft, um Kryptoanalyse zu erschweren:

$$HMAC(M, K) = H(K \oplus \texttt{opad} + H(K \oplus \texttt{ipad} + M))$$

$\texttt{ipad} = 0011\ 0110 \ldots 0011\ 0110_2$
$\texttt{opad} = 0101\ 1010 \ldots 0101\ 1010_2$

Dabei ist $\oplus$ die bitweise Exklusiv-Oder-Verknüpfung und $+$ die Konkatenation.

368-1

10.4 Sicherheitsmechanismen ...

★★★

Digitale Signatur mit asymmetrischer Chiffre

- Sender *A* sendet *M* und *E(M, Private$_A$)* an Empfänger *B*
- *B* entschlüsselt mit *Public$_A$* und prüft, ob Ergebnis gleich *M* ist
- Problem: asymmetrische Verschlüsselung ist langsam
- Daher: Kombination mit kryptographisher Hashfunktion
  - digitale Signatur von *A* auf *M* dann: *E(H(M), Private$_A$)*
- Digitale Signatur sichert Integrität, Nachrichten-Authentizität (bis auf Replay) und Verbindlichkeit
  - nur *A* besitzt *Private$_A$*
  - Replay-Schutz: Transaktionszähler in *M*

Anmerkungen zu Folie 369:

Bei der Signatur mit Hilfe einer kryptographishen Hashfunktion ist die exakte Vorgehensweise wie folgt:

- Alice sendet M und $S := E(H(M), Private_A)$ an Bob
- Bob entschlüsselt S und erhält einen „Soll"-Hashwert $H' := D(S, Public_A)$
- Bob berechnet nun selbst den Hashwert über die Nachricht M:
  $H := H(M)$
- Nun vergleicht Bob H' mit H. Sind beide gleich, akzeptiert er die Signatur.

Dieses Signaturschema ist nur sicher, wenn die verwendete Hashfunktion kollisionsresistent ist. Sonst könnte ein Angreifer zu dem Hashwert
$H' := D(S, Public_A)$ einer Signatur S eine neue Nachricht M' berechnen, die dieselbe Signatur hat, und somit behaupten, Alice hätte M' signiert.

369-1

10.4 Sicherheitsmechanismen ...

Verteilung öffentlicher Schlüssel

- Problem: Übertragung des öffentlichen Schlüssels $Public_A$ von A zu B
- Woher weiß B, daß $Public_A$ authentisch ist?
  - zur Authentifizierung bräuchte B den Schlüssel von A ...
- Lösungen:
  - Übertragung über andere Medien (persönlich, Post, ...)
  - Zertifikate (*Certificates*)

★★★

Zertifikat

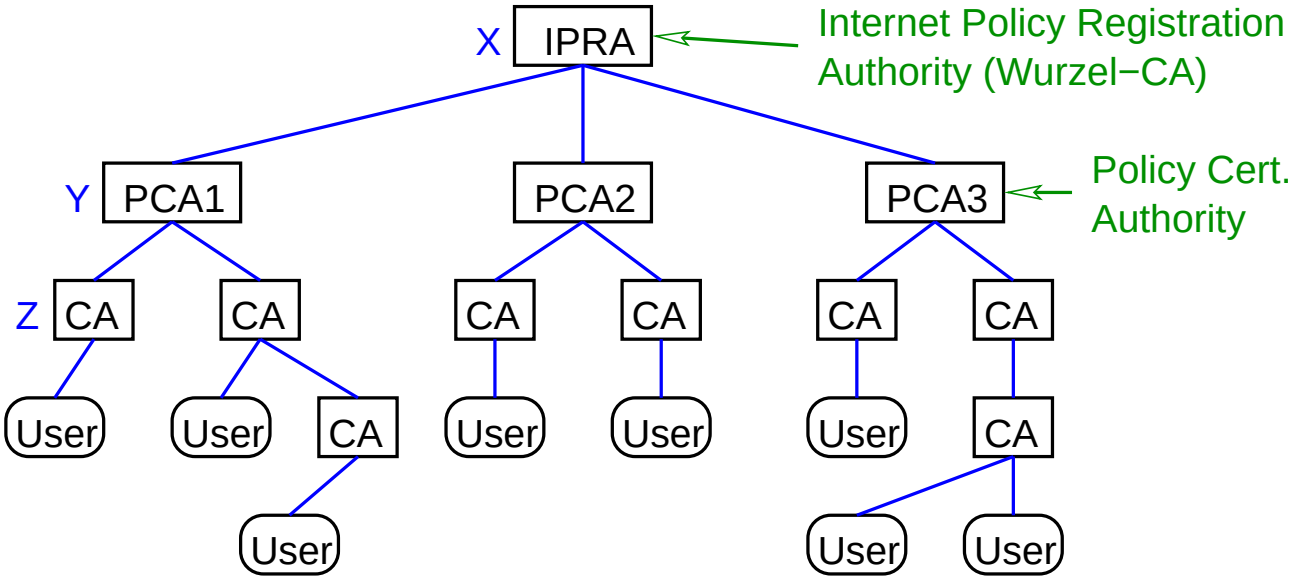
> *Ich bestätige, daß der in diesem Dokument stehende öffentliche Schlüssel dem angegebenen Eigentümer gehört.*
>
> Gezeichnet: $\mathcal{CA}$

- Die Zertifizierungsstelle (CA, *Certification Authority*) beglaubigt die Zuordnung zwischen einem öffentlichem Schlüssel und seinem Besitzer
  - durch digitale Signatur
- Nur noch der öffentliche Schlüssel der CA muß separat veröffentlicht werden

★

Zertifizierungshierarchie (z.B. bei HTTPS)

- Vertrauenskette: *X* zertifiziert, daß Schlüssel von *Y* authentisch ist, *Y* zertifiziert Schlüssel von *Z*, ...



X.509 Zertifikate

- X.509: wichtiger Standard für Zertifikate
- Komponenten des Zertifikats:
  - Name der Person/Institution oder eines Rechners
    - ggf. auch Email-Adresse oder Domain-Name
  - öffentlicher Schlüssel der Person/Institution bzw. des Rechners
  - Name der CA
  - Ablaufdatum des Zertifikats (optional)
  - digitale Signatur der CA
    - über alle obigen Felder

Invalidierung von Zertifikaten

- Zertifikate können beliebig kopiert und verbreitet werden
- Identität wird durch ein Zertifikat nur in Verbindung mit dem Besitz des privaten Schlüssels belegt
- Falls privater Schlüssel ausgespäht wurde:
  - Widerruf des Zertifikats nötig
- Einfache Möglichkeit:
  - *Certificate Revocation List* (CRL)
    Liste widerrufener Zertifikate, signiert von CA
  - Ablaufdatum begrenzt Länge der Liste

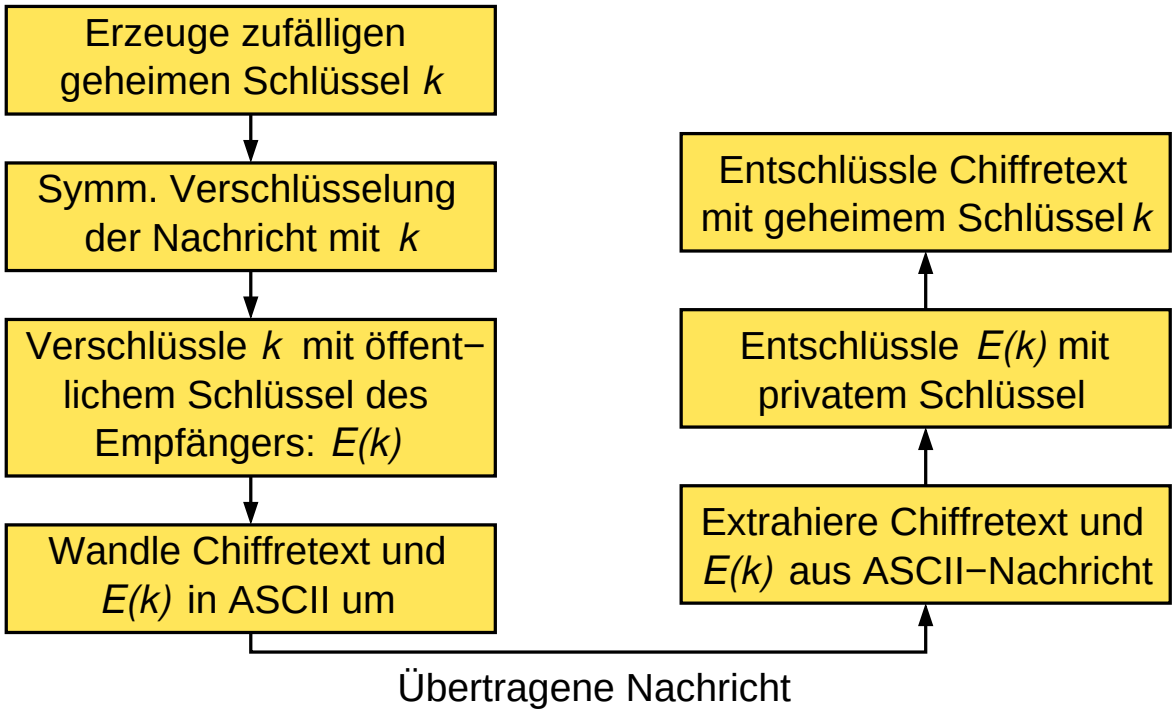
10.5 Beispiele sicherer Protokolle

PGP (Pretty Good Privacy)

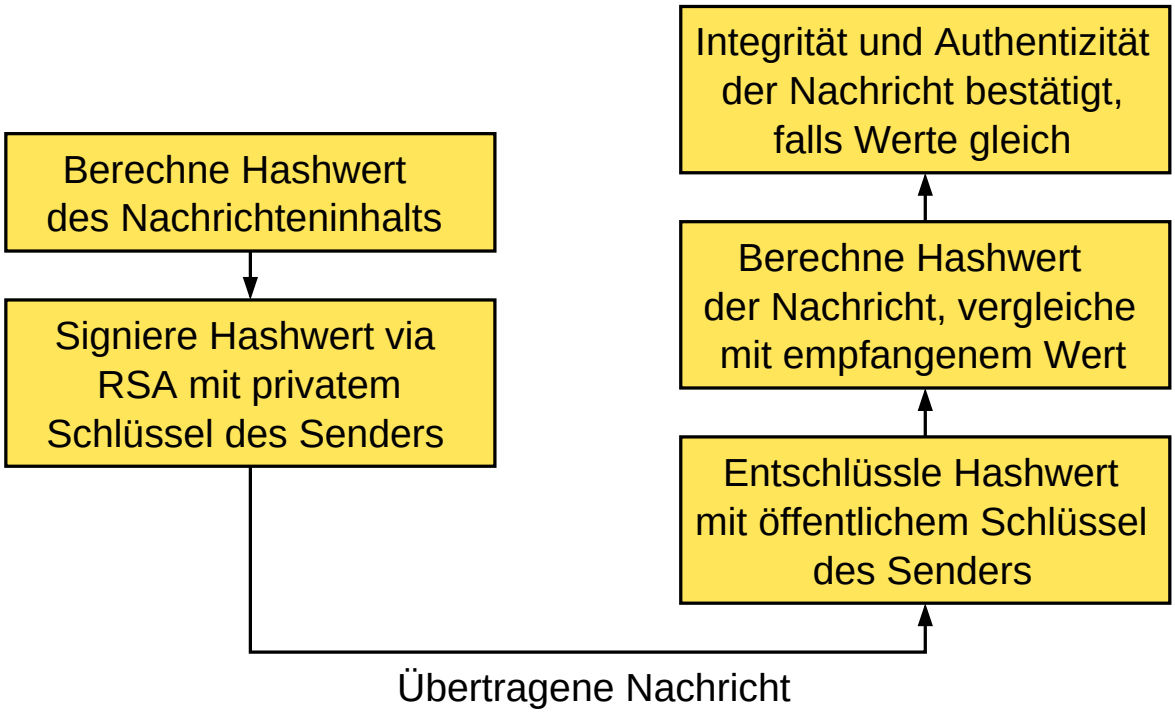
- Realisiert Vertraulichkeit, Integrität, Authentifizierung und Verbindlichkeit für Email
- Mechanismen: Verschlüsselung und digitale Signatur
  - einzeln oder kombiniert verwendbar
- Keine Zertifizierungsstellen bzw. –hierarchie
  - PGP-Benutzer zertifizieren die öffentlichen Schlüssel gegenseitig
    - mehrere Zertifikate möglich (höheres Vertrauen)
  - Vertrauensstufe des Schlüssels wird bei Email-Empfang angezeigt

10.5 Beispiele sicherer Protokolle ...

PGP: Verschlüsselte Übertragung von Emails



PGP: Signierung von Emails



★★

TLS (*Transport Layer Security*)

- Motivation: Sicherheit im WWW, z.B. für Kreditkartenzahlung
  - Vertraulichkeit (der Kreditkarteninformation)
  - Authentizität (des WWW-Servers)
  - Integrität (der Bestelldaten)
  - (Verbindlichkeit wird von TLS nicht gewährleistet)
- TLS ist ein Internet-Standard der IETF
  - Basis: ältere Realisierung SSL (*Secure Socket Layer*)
- TLS ist die Grundlage vieler sicherer Protokolle im WWW:
  - z.B. HTTPS, FTPS, ...
  - realisiert durch eine zusätzliche Schicht

Anmerkungen zu Folie 378:

SFTP ist eine weitere sichere Variante des Dateitransfer-Protokolls FTP, das nicht TLS, sondern das SSH-Protokoll als Basis für die Sicherheit verwendet.

10.5 Beispiele sicherer Protokolle ...

TLS: sichere Transportschicht

Anwendung (z.B. HTTP)
Sichere Transportschicht (TLS)
TCP
IP
Netzwerk

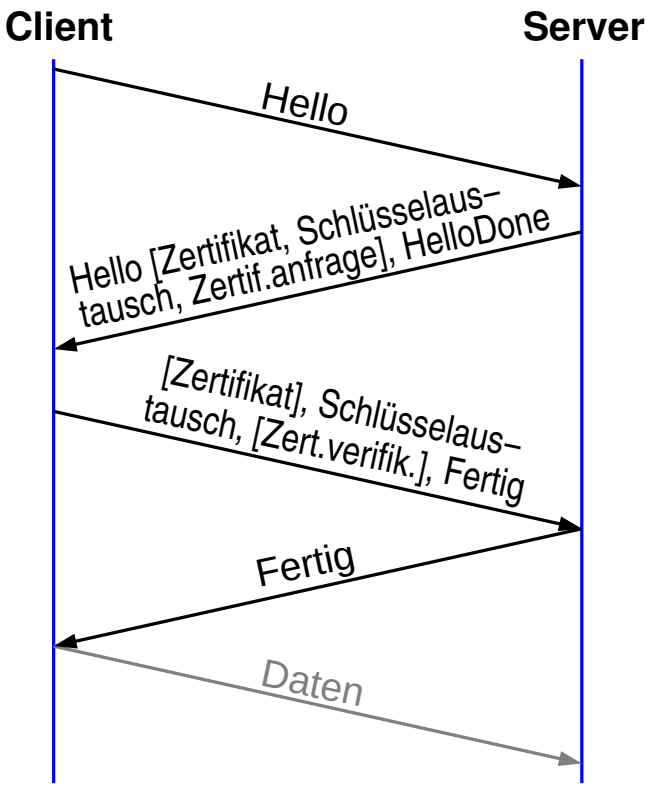
- Vorteil: unveränderte Anwendungsprotokolle
- Spezielle Ports, z.B. 443 für HTTPS
  - TLS gibt Daten von TCP an HTTP-Protokoll weiter (bzw. umgekehrt)

Wichtige TLS Teil-Protokolle:

- Handshake-Protokoll
  - beim Verbindungsaufbau
  - Aushandeln der kryptographischen Parameter:
    - Verfahren, Schlüssellänge, Sitzungsschlüssel, Zertifikate, Kompression
- Record-Protokoll
  - für die eigentlichen Daten
  - Fragmentierung, Kompression, Message Digests, Verschlüsselung, Transport (TCP)

TLS Handshake-Protokoll

- Bis zu 12 Nachrichten
- Aushandeln der kryptographischen Parameter notwendigerweise unverschlüsselt
- Man-in-the-Middle kann schwache Verschlüsselung aushandeln
- Anwendungen müssen auf Mindestanforderungen bestehen, ggf. Verbindungsabbruch



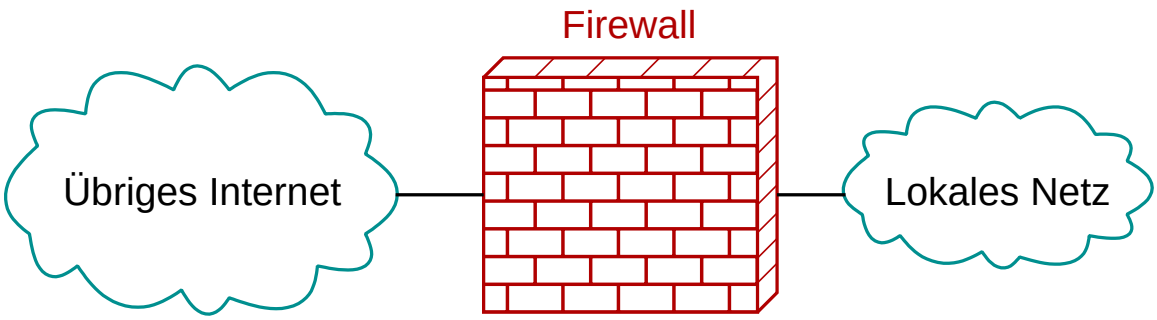
Anmerkungen zu Folie 381:

- Die Hello-Nachrichten dienen zum Aushandeln von Parametern:
  - Version, Zufallszahl, SessionID, Verfahren für Schlüsselaustausch, Verschlüsselung und Kompression, ...
- Für den Schlüsselaustausch können unterschiedliche Verfahren verwendet werden, die dann auch Anzahl und Inhalt der Nachrichten bestimmen.
- Ein Client-Zertifikat wird nur auf Anfrage durch Server übertragen
- Die Zertifikatsverifikations-Nachricht enthält die digitale Signatur aller bisher gesendeten und empfangenen Nachrichten
- Der Server verifiziert sein Zertifikat beim Schlüsselaustausch durch eine Signatur über die Zufallszahlen aus den Hello-Nachrichten und die Server-Schlüsselaustausch-Parameter

381-1

10.6 Firewalls

★★



- **Firewall**: Router mit Filterfunktion
  - kann bestimmte Pakete ausfiltern (verwerfen) und somit Zugriff auf bestimmte Hosts / Dienste unterbinden
  - wäre i.W. überflüssig, wenn alle Dienste sicher wären!
- Zwei Typen:
  - Filter-basierte Firewalls
  - Proxy-basierte Firewalls

★★

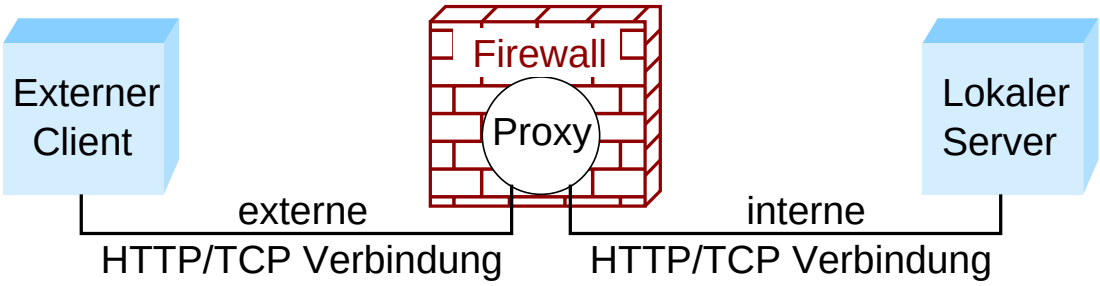
Filter-basierte Firewalls

- Filtern nur aufgrund von Quell- und Ziel-IP-Adressen, Quell- und Ziel-Ports, sowie übertragenem Protokoll
- Filterregeln z.B.
  - `deny tcp 192.12.0.0/16 host 128.7.6.5 eq 80`
  - `permit tcp any host 128.7.6.5 eq 25`
- Frage: alles erlaubt, was nicht verboten ist, oder umgekehrt?
- Statische oder dynamische Regeln
  - z.B. FTP: neue Ports für jede übertragene Datei
- „Level-4-Switch“: Firewall kennt Transport-Protokolle

★

Proxy-basierte Firewalls

- Proxy: Mittler zwischen Client und Server
  - für Client: Proxy ist Server, für Server: Proxy ist Client



- Proxy arbeitet auf Anwendungsschicht
  - kann auf der Basis des Nachrichteninhalts filtern
  - z.B. HTTP-Anfragen nach bestimmten Seiten nur von speziellen Hosts akzeptieren

10.6 Firewalls ...

Grenzen von Firewalls

- Kein Schutz interner Benutzer untereinander
- Nur begrenzter Schutz gegen mobilen Code (z.B. Email Wurm)
- Schutz von Teilen eines Netzes schwierig
- Angreifer kann sich in privilegiertes Netz „einschleichen“
  - z.B. bei drahtlosen Netzen
- Filterung über Sender-IP-Adresse/Port ist unsicher

Vorteil von Firewalls

- Umsetzung einer Sicherheitsstrategie an zentraler Stelle

10.7 Zusammenfassung

- Sicherheitsanforderungen:
  - Vertraulichkeit, Integrität, Authentizität,Verbindlichkeit
  - Verfügbarkeit, Anonymität, ...
- IP, TCP, UDP erfüllen keine Sicherheitsanforderungen
  - Vertraulichkeit, Integrität, Authentizität
- Kryptographische Verfahren:
  - symmetrische und asymmetrische Chiffren
  - Kryptographische Hashes (Message Digest)
- Sicherheitsmechanismen
  - Authentifizierung (Kommunikationspartner, Nachrichten)
  - Integrität: Hashwerte mit Schlüssel, digitale Signatur
  - Verteilung öffentlicher Schlüssel: Zertifikate

10.7 Zusammenfassung ...

- Sichere Protokolle, z.B. PGP, TLS (HTTPS), IPsec
- Firewalls

Fortsetzung:

- Rechnernetze-Praktikum (WiSe)
  - Aufbau von Netzen, Routing und Switching
  - PO 2012: B.Sc., Vertiefungspraktikum, 5 LP
  - FPO 2021: B.Sc., Grundlagenpraktikum, 6 LP
- Rechnernetze II (SoSe)
  - weitere Netzwerktechnologien (Fast Ethernet, WLAN, ...)
  - Vertiefung (Routing, QoS, IPsec, ...)
  - PO 2012: B.Sc., Wahlmodul, 5 LP
  - FPO 2021: M.Sc., Kernmodul, 6 LP