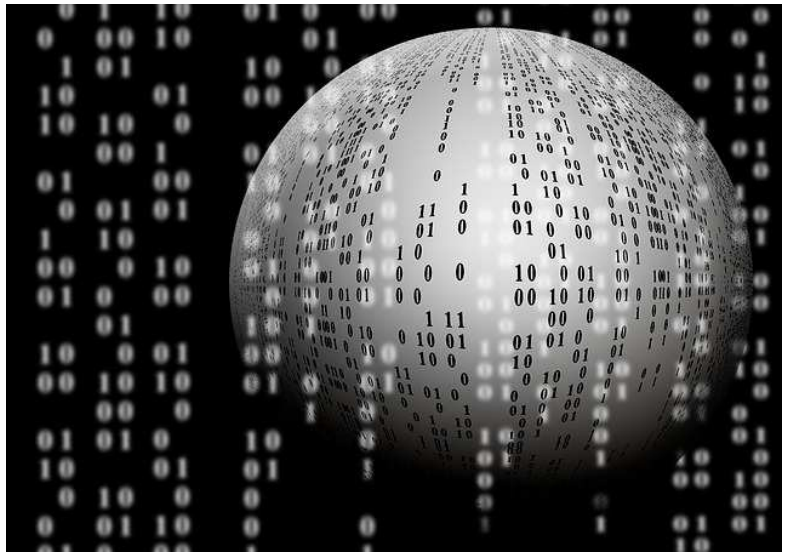
Kommunikationssysteme

Prof. Dr. Henning Pagnia

DHBW Mannheim



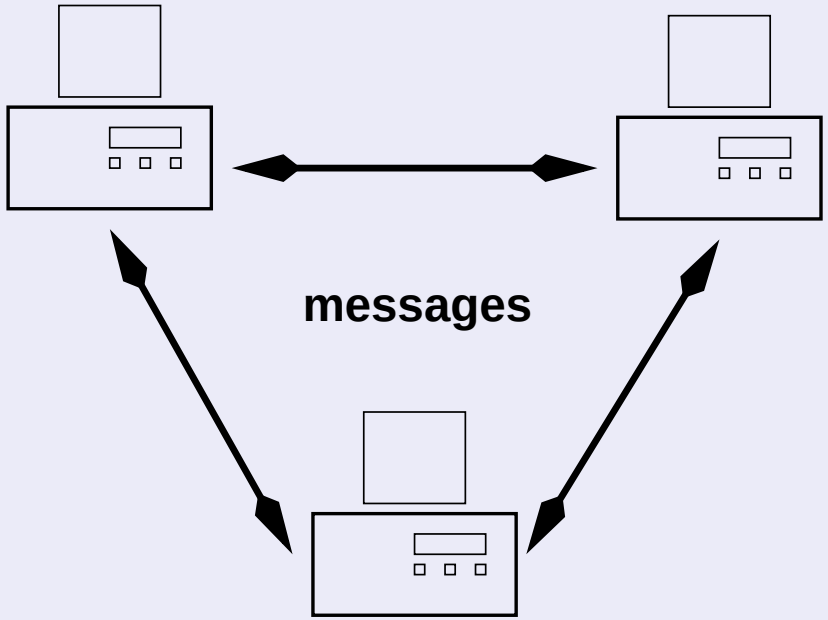
Wichtiges zur Vorlesung

Prof. Dr. Henning Pagnia

- Wirtschaftsinformatik
- Email: *pagnia@dhbw-mannheim.de*
- Telefon: *0621 / 4105-1131*
- Raum: *254 B*

Worum geht's?

Verbinden mehrerer Rechner



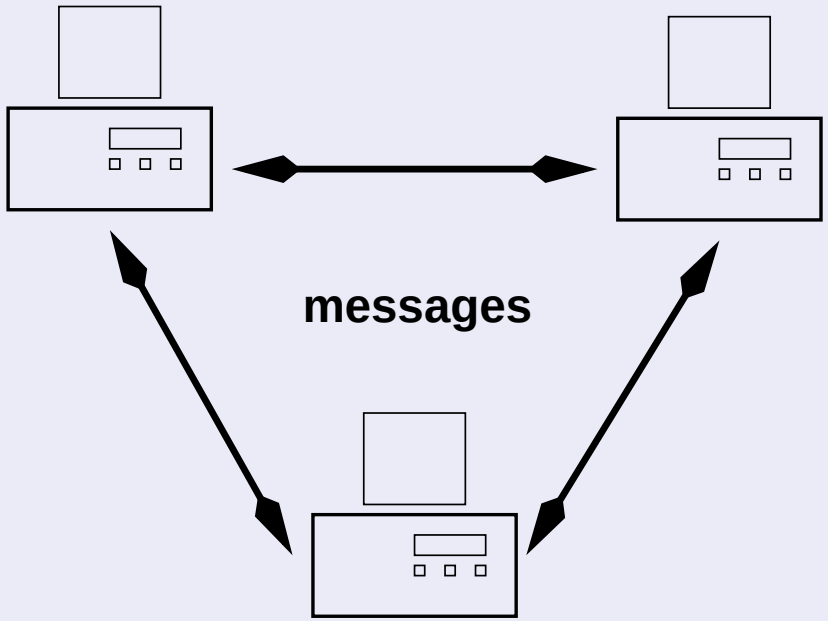
Wieso?

- Kommunikation
- verteilte Speichernutzung
- parallele Berechnungen
- Mehrfachnutzung von Diensten:
  - Backup
  - Zugang zu globalem Netz
  - Datenbank
  - ...

$\Rightarrow$ **Lokale Netzwerke entstehen!**

Worum geht's?

Verbinden mehrerer Rechner



Wieso?

- Kommunikation
- verteilte Speichernutzung
- parallele Berechnungen
- Mehrfachnutzung von Diensten:
  - Backup
  - Zugang zu globalem Netz
  - Datenbank
  - ...

$\Rightarrow$ **Lokale Netzwerke entstehen!**

Globale Netze

Anwendungsmöglichkeiten

- Dateitransfer
- E-Mail
- Chat
- Informationssysteme
- World-Wide Web
- Social Networks
- Streaming (Audio / Video)
- Live Streaming (Radio / TV)
- Telefon (VoIP)
- Videokonferenzen
- ... und vieles mehr

$\Rightarrow$ **Computer sind Kommunikationssysteme!**

Vernetzen! ... aber wie?

Verdrahtung

- Welche Kabel? ($\Rightarrow$ physikalische Eigenschaften?)
- Peripherie-Gerät: Netzwerkkarte
- Welche Topologie? ($\Rightarrow$ Zusatz-Hardware?)

Einbettung in das Betriebssystem

- mittels E/A-Interrupt-Konzept
- Zuordnung Nachricht zu Anwendung mittels spez. Adressen

Vernetzen! ... aber wie?

Verdrahtung

- Welche Kabel? ($\Rightarrow$ physikalische Eigenschaften?)
- Peripherie-Gerät: Netzwerkkarte
- Welche Topologie? ($\Rightarrow$ Zusatz-Hardware?)

Einbettung in das Betriebssystem

- mittels E/A-Interrupt-Konzept
- Zuordnung Nachricht zu Anwendung mittels spez. Adressen

Vernetzen! ... aber wie? (Forts.)

Benennung der Rechner

- mit Nummern?
- mit Namen? ($\Rightarrow$ Namensdienst notwendig)

Rechner müssen sich verstehen!

- Absprache mittels Protokoll (z. B. Signalkodierung, Header-Aufbau)
- Einsatz globaler Standards:
  - ISO/OSI-Schichtenmodell
  - TCP/IP-Stack

Vernetzen! ... aber wie? (Forts.)

Benennung der Rechner

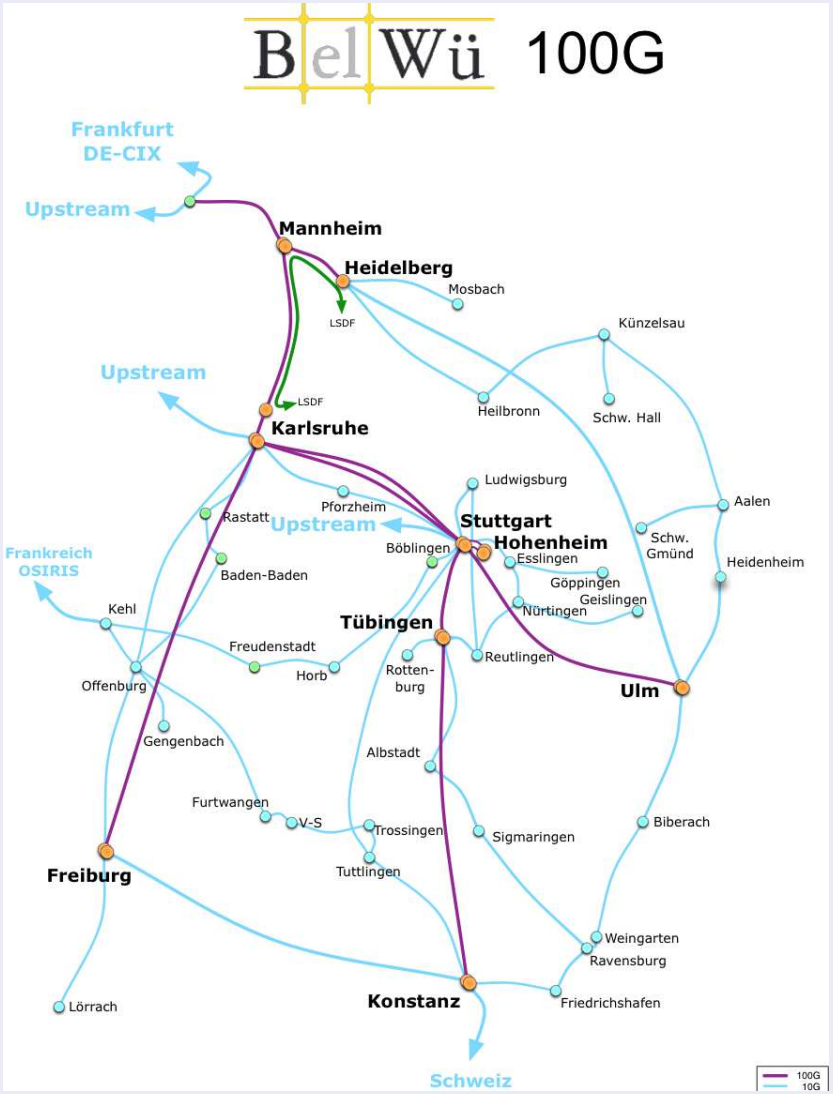
- mit Nummern?
- mit Namen? ($\Rightarrow$ Namensdienst notwendig)

Rechner müssen sich verstehen!

- Absprache mittels Protokoll (z. B. Signalkodierung, Header-Aufbau)
- Einsatz globaler Standards:
  - ISO/OSI-Schichtenmodell
  - TCP/IP-Stack

Vernetzen! ... aber wie? (Forts.)

Nachrichtenvermittlung



- Dabei:
  Keine vollständige Vermaschung aller Rechner möglich ...
- Konsequenz:
  Nachrichten müssen ggf. schrittweise weitergereicht werden
  $\Rightarrow$ *Routing*
- Problematisch:
  Das Netzverhalten ändert sich fortwährend.

(Bild: belwue.de; 100 G Ausbau; Stand 2014)

Themen der Vorlesung

(1) Hardware- und Software-Aspekte; Schichtenmodelle

(2) Die Bitebene (Schicht 1)

(3) Die Sicherungsschicht (Schicht 2)

(4) Die Vermittlungsschicht (Schicht 3)

(5) Die Transportschicht (Schicht 4)

(6) Die Verarbeitungsschicht (Schicht 7)

Literatur zur Vorlesung

- [TW2012] *Andrew S. Tanenbaum und David J. Wetherall:*
  Computernetzwerke
  5. aktualisierte Auflage, 2012
  Pearson Studium IT
  ISBN: 978-3868941371

- [PD2007] *Larry L. Peterson und Bruce S. Davie:*
  Computernetze: Eine systemorientierte Einführung
  4. Auflage, 2007.
  dpunkt-Verlag, Heidelberg.
  ISBN: 978-3898644914

- [Eck2018] *Claudia Eckert:*
  IT-Sicherheit
  11. Auflage, 2018.
  De Gruyter Oldenbourg.
  ISBN: 978-3110551587

Computernetzwerke: Die Hardware

Klassizifizierung:

Interprocessor distance	Processors located in same	Example
0.1 m	Circuit board	Data flow machine
1 m	System	Multicomputer
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1,000 km	Continent	
10,000 km	Planet	The Internet

Computernetzwerke: Die Hardware

Klassizifizierung:

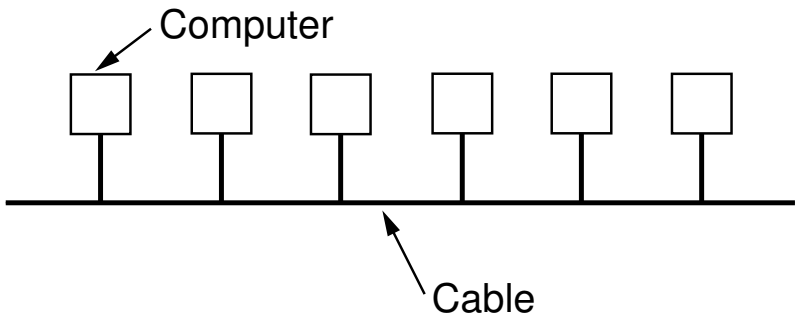
Interprocessor distance	Processors located in same	Example
0.1 m	Circuit board	Data flow machine
1 m	System	Multicomputer
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1,000 km	Continent	
10,000 km	Planet	The Internet

Übertragungstechniken:

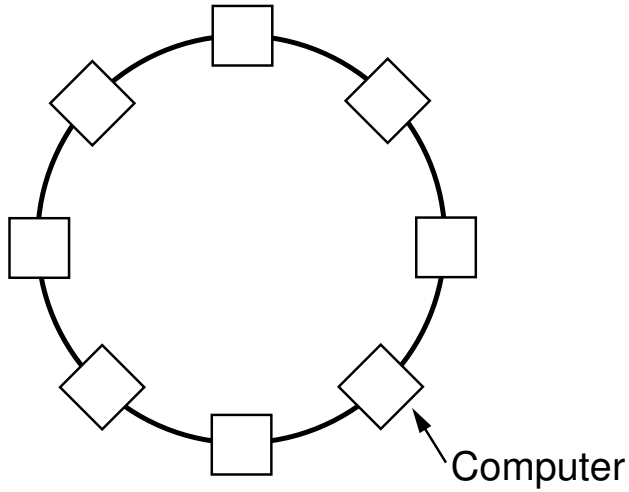
- **Broadcast**-Netzwerk:
  alle Rechner teilen sich den Kommunikationskanal
  - $\Rightarrow$ jeder erhält alle Pakete
  - $\Rightarrow$ *Kollisionen* sind möglich
- **Punkt-zu-Punkt**-Netzwerk:
  Rechner sind jeweils paarweise verbunden
  - $\Rightarrow$ nur Empfänger erhält Pakete
  - $\Rightarrow$ Ggf. muss *vermittelt* werden

Local Area Networks (LANs)

i. Allg. geringe Reichweite, Verwendung einfach strukturierter **Broadcast**-Topologien



(a)



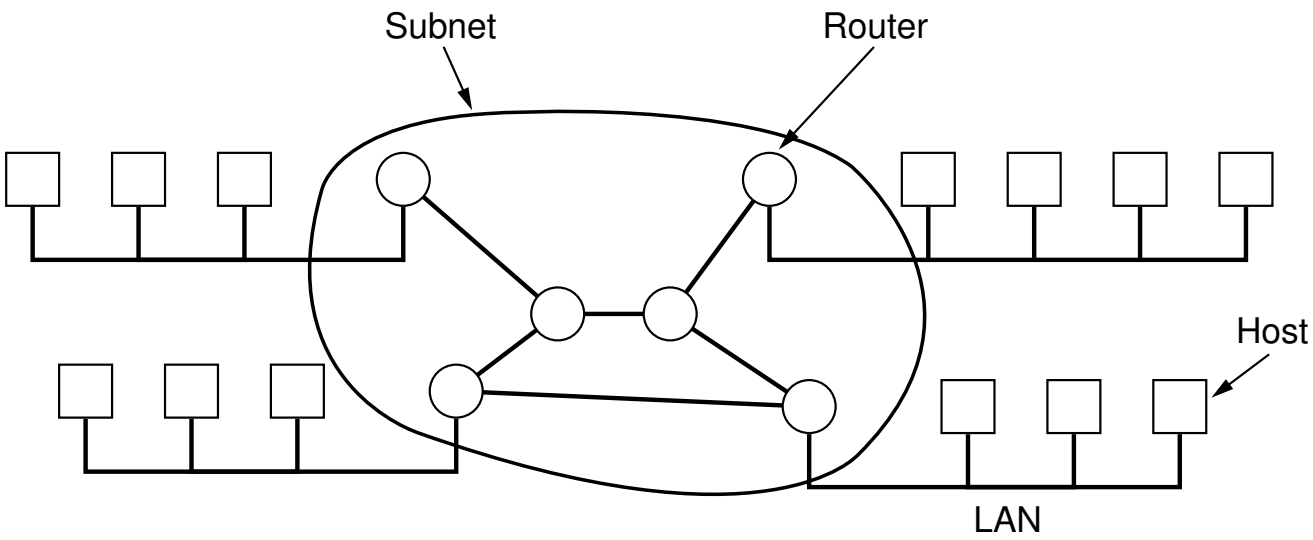
(b)

(a) Bus-basiert:
Alle Rechner hängen am gleichen Kabel (Bus) Was einer sendet, bekommen alle Wenn mehrere gleichzeitig senden, versteht keiner etwas
$\Rightarrow$ Kollision.

(b) Token-basiert:
Ein *Token* kreist ständig im Ring. Sender wartet bis er das Token hat. Er entnimmt es und stellt die Nachricht ein. Ist diese wieder bei ihm, ersetzt er sie durch das Token.

Wide Area Networks (WANs)

WANs verbinden LANs mit Hilfe von aktiven Verbindungselementen, sog. **Router**



Store-and-Forward-Prinzip

- puffern
- auspacken, auswerten, einpacken
- über den richtigen Weg weitersenden

Verbundnetze

Die bisher vorgestellten Topologien (Bus, Ring) gelten für homogene Netzwerke (z. B. Ethernet).
Sehr große Netze entstehen durch Verbinden von unterschiedlichen Netzwerktypen
$\Rightarrow$ *Internetworks* (Verbundnetze)

Beispiele

- Verbinden unterschiedlicher LANs (Bus- und Token-basierte)
  z. B. innerhalb einer Abteilung
- Großflächiges Verbinden lokaler LANs mittels eines WAN
  (z. B. geografisch verteilte Filialen)
- Weltweites Verbinden vieler WANs ($\Rightarrow$ Internet)

Intranet

bezeichnet das zur Verfügung stellen von Internet-Diensten *innerhalb* eines privaten Firmennetzes

Computernetzwerke: Geräte

Netzwerkkarte

- zur Anbindung eines Rechners
- Hardware-Adresse (MAC-Adresse)
- verarbeitet nur Nachrichten an diese Adresse

Computernetzwerke: Geräte (Forts.)

Repeater

- simples Gerät zur Signalauffrischung
- verbindet mehrere Teilsegmente eines LANs
- auf unterster Ebene des Protokollstacks

Computernetzwerke: Geräte (Forts.)

Repeater

- simples Gerät zur Signalauffrischung
- verbindet mehrere Teilsegmente eines LANs
- auf unterster Ebene des Protokollstacks

Hub

- besondere Art von Repeater mit mehreren Eingängen (*Ports*)
  $\Rightarrow$ sternförmige Verkabelung
- sendet jede eingehende Nachricht per Broadcast weiter
- *Uplink-Port*
- teilweise Dual-Speed-Technik

Computernetzwerke: Geräte (Forts.)

Bridge

- trennt LANs logisch auf
- intelligentes Weiterleiten von Nachrichten mittels Adress-Cache
  $\Rightarrow$ reduziert Netzverkehr
- selbstkonfigurierend, d. h. automatisches Löschen älterer Einträge
- Rechneradressierung über MAC-Adresse
- auf zweitunterster Schicht

Computernetzwerke: Geräte (Forts.)

Switch

- besondere Art von Bridge
- Port-zu-Port-Verbindungen
  $\Rightarrow$ jeweils volle Übertragungsrate pro Verbindung!
- kombinierbar mit Switches oder Hubs
- meist mit Management-Schnittstelle
  $\Rightarrow$ eigene IP-Adresse

Computernetzwerke: Geräte (Forts.)

Router

- verbindet (lokale) Netze
- Adresstabellen zur Wegewahl ($\Rightarrow$ Routing)
- im LAN an der Schnittstelle
- log. Strukturierung des LANs in Teilnetze
- oftmals Firewall-Funktion
- auf drittunterster Ebene des Protokollstacks

Computernetzwerke: Die Software

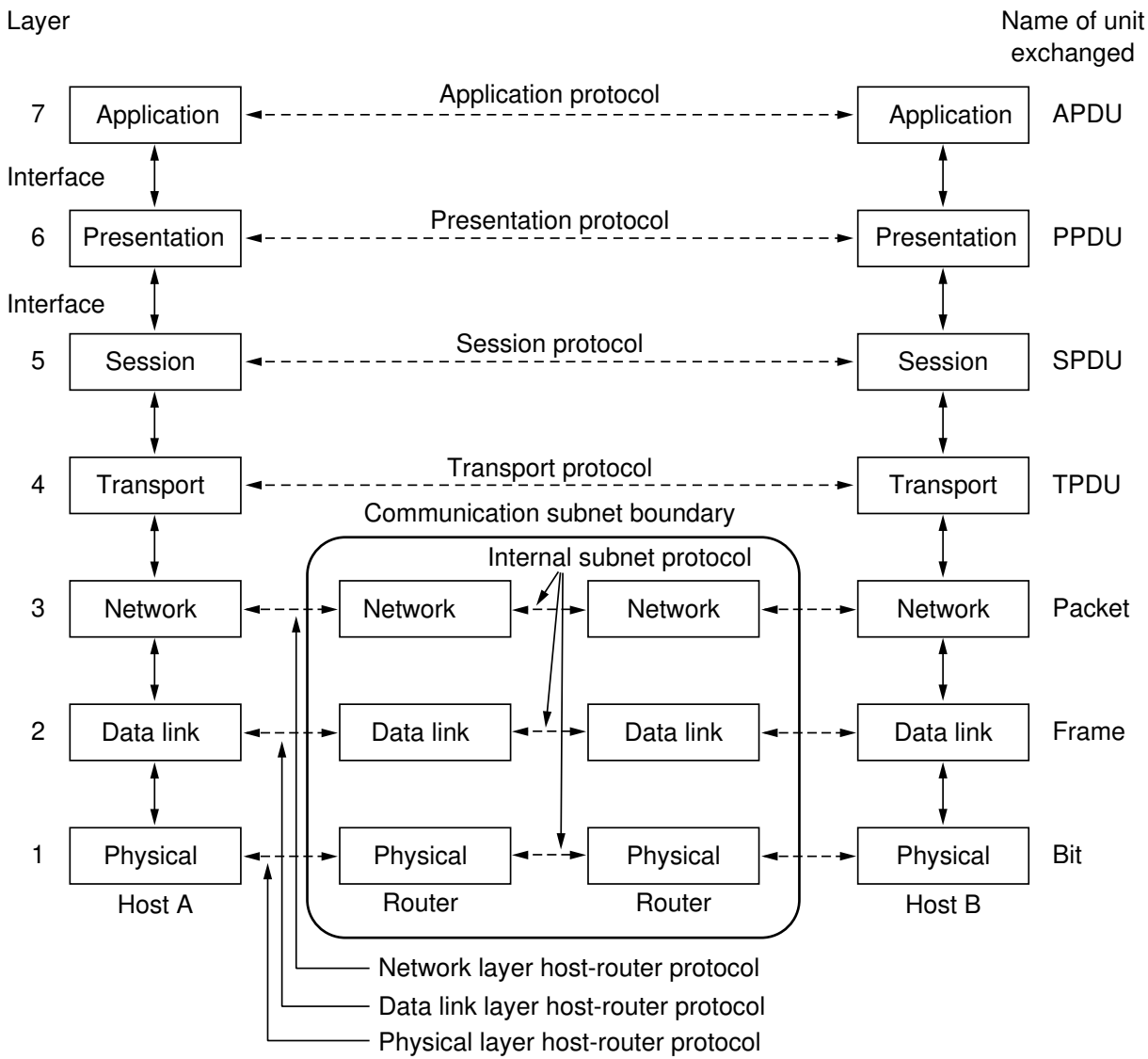
Schichtenmodelle

Jede Art von Software, die die Vernetzung von Computern ermöglicht, verwendet eine **Protokollhierarchie**, d. h. die Dienste, die das Netzwerk anbietet, werden in mehreren **Schichten** (*Layers*) angeordnet:

Begriffe

- **Protokolle:** *Wie* werden Nachrichten ausgetauscht?
- **Dienste:** *Was* bietet das Netz zur Kommunikation an?
- **Schnittstellen:** *Wie* können Clients Netzwerkdienste benutzen?

Das OSI-Referenzmodell



Achtung: zählt zum Allgemeinwissen!

Die OSI-Schichten

Layer 1: Bitübertragungsschicht (*physical layer*)

Betrifft die Übertragung von Bits in Form von Signalen. Sie beschäftigt sich mit mechanischen bzw. elektrischen Fragen.

Layer 2: Sicherungsschicht (*data link layer*)

Regelt u. a. den Zugriff auf ein gemeinsam benutztes Übertragungsmedium, teilt die Bits in *Data Frames* auf und stellt eine übertragungsfehlerfreie Kommunikation zur Verfügung.

Layer 3: Vermittlungsschicht (*network layer*)

Ist u. a. verantwortlich für das Routing.

Layer 4: Transportschicht (*transport layer*)

Bietet verbindungslose und -orientierte Dienste mit unterschiedlichen Zuverlässigkeitsgarantien an. Sie ist die Schnittstelle für Anwendungen.

Die OSI-Schichten (Forts.)

Layer 5: Sitzungsschicht (*session layer*)

Implementiert die Sitzungsfunktionalität, dazu gehören u. a. Dialogsteuerung, Synchronisation und Checkpunkte.

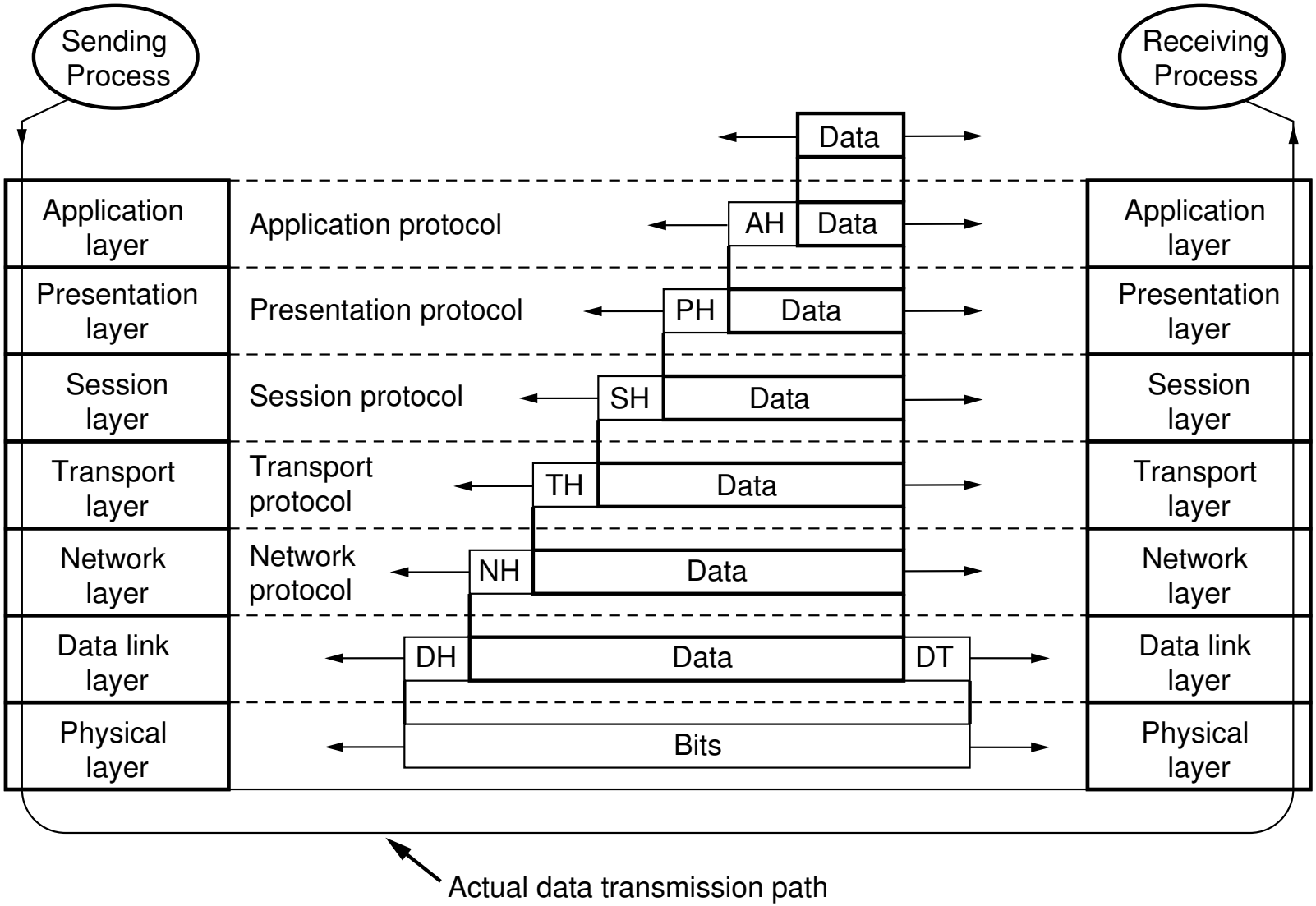
Layer 6: Darstellungsschicht (*presentation layer*)

Ermöglicht plattformunabhängigen Datenaustausch
($\Rightarrow$ Gleitkommadarstellung, Byte-Order).

Layer 7: Verarbeitungsschicht (*application layer*)

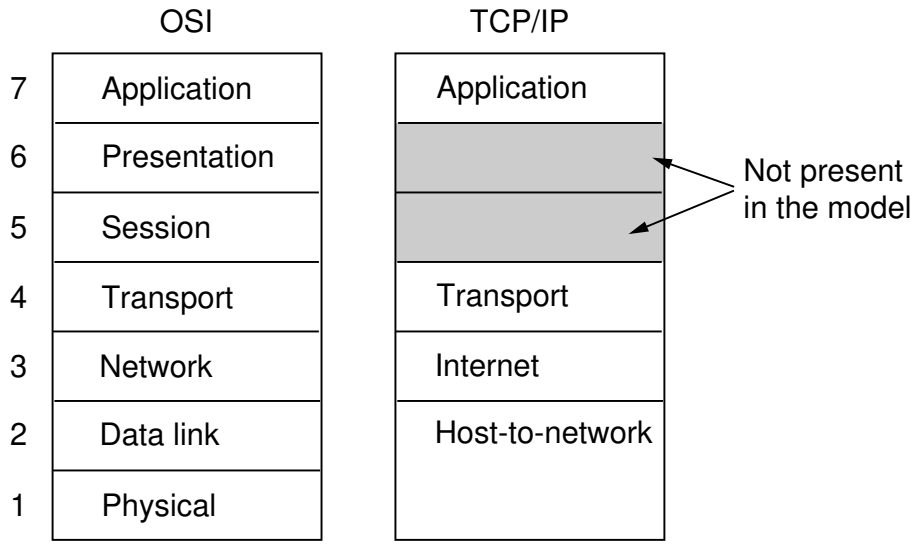
Implementiert, was dem Benutzer zur Verfügung steht
($\Rightarrow$ E-Mail, Telnet, WWW, ...).

Die OSI-Schichten (Forts.)



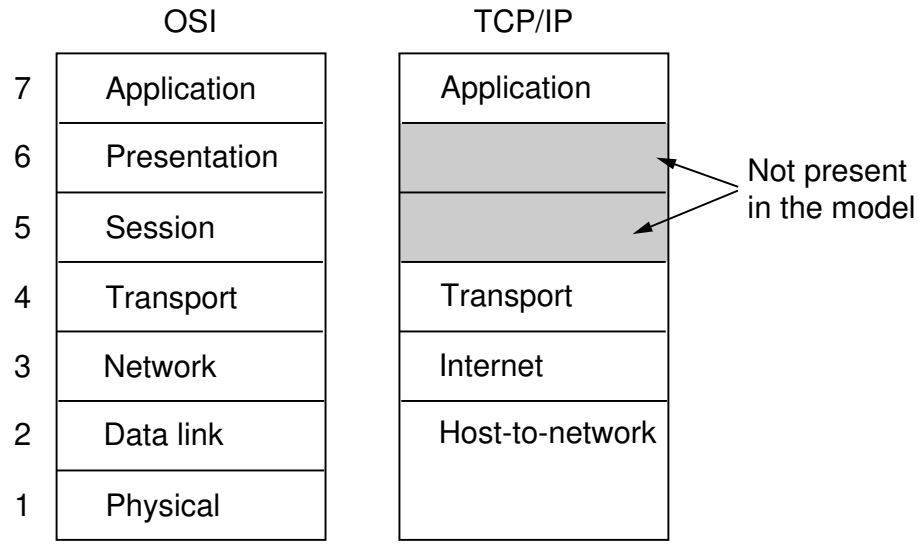
Das TCP/IP-Referenzmodell

Einfacher, aber nicht so sauber definiert wie das OSI-Modell:



Das TCP/IP-Referenzmodell

Einfacher, aber nicht so sauber definiert wie das OSI-Modell:



Nachteile:

(1) keine klare Unterscheidung zw. Diensten, Schnittstellen und Protokollen
$\Rightarrow$ schwierig, Schichten zu re-implementieren

(2) ominöse *Host-to-network* Schicht

Vorteile:

(1) war zur richtigen Zeit **verfügbar**

(2) schon früh in UNIX-Betriebssystemen implementiert

(3) **frei** benutzbar $\Rightarrow$ **De-facto Standard** für das Internet.

Schichtenmodell der Vorlesung

Wir verwenden einen Mix (vgl. Tanenbaum-Buch):

5	Verarbeitungsschicht	(*application layer*)
4	Transportschicht	(*transport layer*)
3	Vermittlungsschicht	(*network layer*)
2	Sicherungsschicht	(*data link layer*)
1	Bitübertragungsschicht	(*physical layer*)

In dieser Vorlesungsreihe werden wir diese fünf Schichten – von unten beginnend – besprechen.

Einordnung in unser Schichtenmodell

Layer 1: Bitübertragungsschicht (*physical layer*)

Betrifft die Übertragung von Bits in Form von Signalen. Sie beschäftigt sich mit mechanischen bzw. elektrischen Fragen.

Layer 2: Sicherungsschicht (*data link layer*)

Regelt u. a. den Zugriff auf ein gemeinsam benutztes Übertragungsmedium, teilt die Bits in *Data Frames* auf und stellt eine übertragungsfehlerfreie Kommunikation zur Verfügung.

Layer 3: Vermittlungsschicht (*network layer*)

Ist u. a. verantwortlich für das Routing.

Layer 4: Transportschicht (*transport layer*)

Bietet verbindungslose und -orientierte Dienste mit unterschiedlichen Zuverlässigkeitsgarantien an. Sie ist die Schnittstelle für Anwendungen.

Die Verarbeitungsschicht (*application layer*)

implementiert, was dem Benutzer zur Verfügung steht ($\Rightarrow$ E-Mail, Telnet, WWW, . . .).

Wichtige Begriffe

Bit (*binary digit*):

Maßeinheit für den Informationsgehalt. 1 Bit ist der Informationsgehalt, der in einer Auswahl aus zwei gleich wahrscheinlichen Möglichkeiten enthalten ist.

Speicherplatz:

Einheit: Byte
1PiB = 2^{10} TiB = 2^{20} GiB = 2^{30} MiB = 2^{40} KiB = 2^{50} Byte
(nach IEC 60027-2: pebi, tebi, gibi, mebi, kibi)

1 KB = 1000 Byte (leider nicht immer :–))

Datenrate:

Menge von Daten (in Bit), die über einen phys. Kanal innerhalb einer gegebenen Zeit übertragen werden kann (auch: *Übertragungsrate*, *Bitrate*.)
Einheit: **bps** (bits per second)
1 Gbps = 10^3 Mbps = 10^6 Kbps = 10^9 bps

Wichtige Begriffe (Forts.)

Durchsatz:

tatsächlich erreichte (Netto-)Datenrate
(ist je nach Schicht unterschiedlich $\Rightarrow$ Wieso?)

Latenz:

Zeit, die benötigt wird, um ein Bit zum Empfänger zu übertragen.

Übertragungszeit:

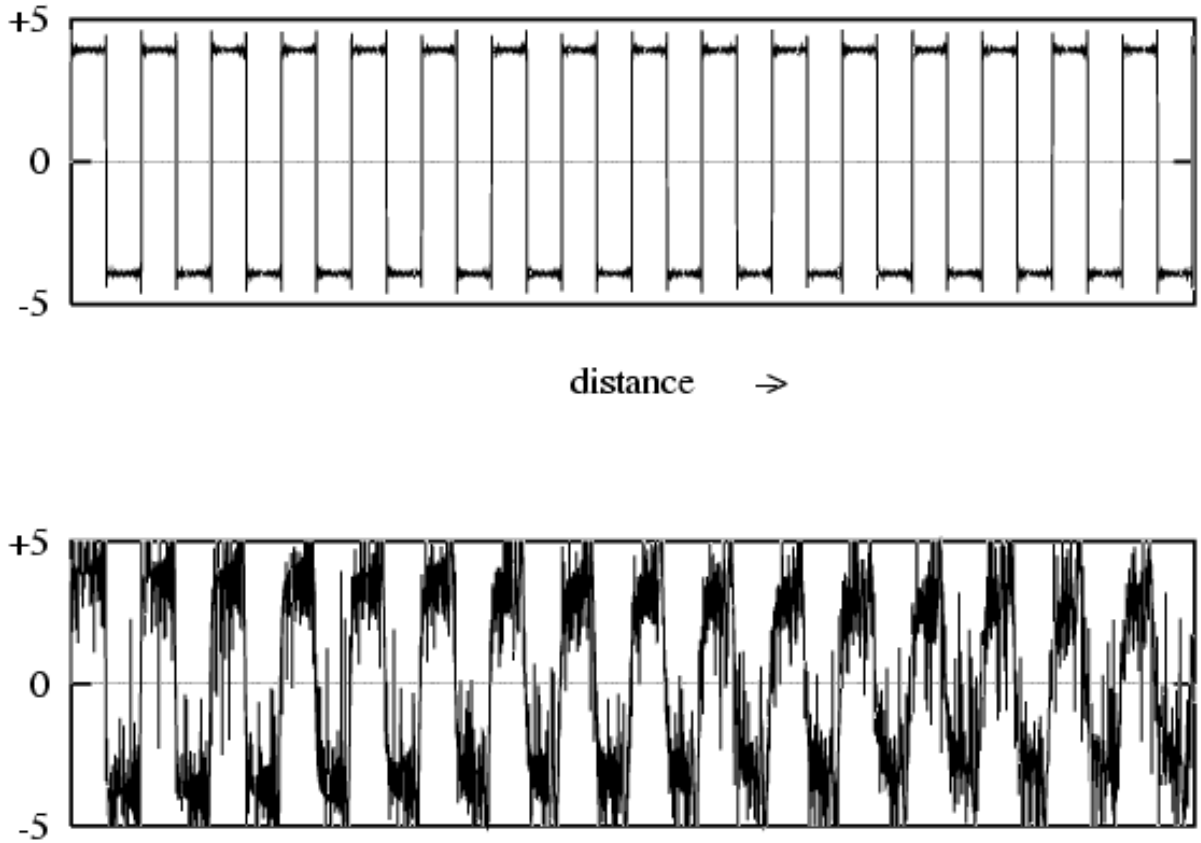
Zeit zur Übertragung einer vollständigen Nachricht (von z. B. 1 KiB).

Round-Trip-Time (RTT):

Zeit bis zu einer Nachricht eine Antwort eintrifft.

Übertragung von digitalen Signalen

- Information muss kodiert werden, z.B. -5 Volt für '0' und +5 Volt für '1'
- Leider gibt es bei der Übermittlung Schwierigkeiten:



Übertragung von digitalen Signalen (Forts.)

Beobachtung

- Mit wachsender Entfernung werden Signale schlechter und undeutlicher. Die Übertragungsqualität hängt u. a. davon ab, wie gut bestimmte Frequenzen vom Kanal übertragen werden (z. B. Bandbreitenbegrenzung durch Tiefpass-Filter).

Ursachen

- Dämpfung
- externe Störungen bzw. Rauschen
- kanalbedingte Verzerrungen, die entstehen, wenn bestimmte Frequenzen nicht oder nur gedämpft übertragen werden (z. B. materialbedingt).

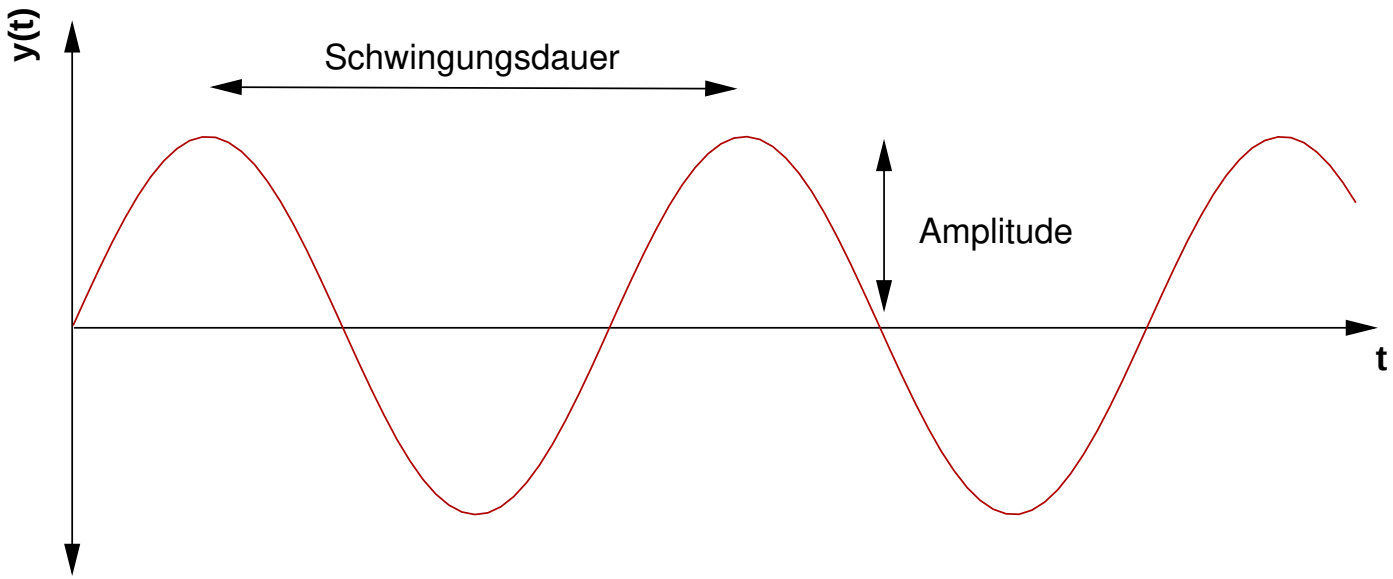
Folge

- Es gibt naturgegebene **Begrenzungen**, wie schnell wir Signale übertragen können.
  (Dies gilt gleichermaßen für kabelgebundene und drahtlose Kommunikation.)

Einschub: Periodische Funktionen

Kenngrößen

- Amplitude
- Schwingungsdauer (Periodendauer) T
- Frequenz $f = 1/T$



Definition *Bandbreite* B eines Kanals:
Spektrum aller für die Übertragung verwendeten Frequenzen (in Hz)

Signalbündelung

- Mehrere aufeinanderfolgende Bitwerte können über ein einziges nicht-binäres Signal kodieren werden.
- Beispiel mit vier Signalstufen:

'00' $\rightarrow$ 0 Volt	'01' $\rightarrow$ 2 Volt
'10' $\rightarrow$ 4 Volt	'11' $\rightarrow$ 6 Volt

- Eine solche Signalbündelung bezeichnet man als **Baud**.

Idee

Über eine 2 400 Baud Verbindung (z. B. ein Modem) kann man also mit einer Bitrate von 12 000 bps kommunizieren, wenn 32 unterschiedliche Signalstufen verwendet werden.

Frage: Lässt sich so die Datenrate beliebig erhöhen?

Maximale Übertragungsrate

Nyquist-Kriterium

Bei einem V-stufigen diskreten Signal und für die Bandbreite B ergibt sich nach Nyquist eine maximal mögliche Übertragungsrate Tr (in bps) von

$$Tr \leq 2\,B \cdot \log_2 V$$

Shannon-Kriterium

Laut **Shannon** gilt für einen verrauschten Kanal mit dem linearen Rauschabstand S/N eine weitere Begrenzung der Übertragungsrate Tr:

$$Tr \leq B \cdot \log_2(1 + \mathrm{S/N})$$

(S/N= *signal-to-noise ratio*, linear)

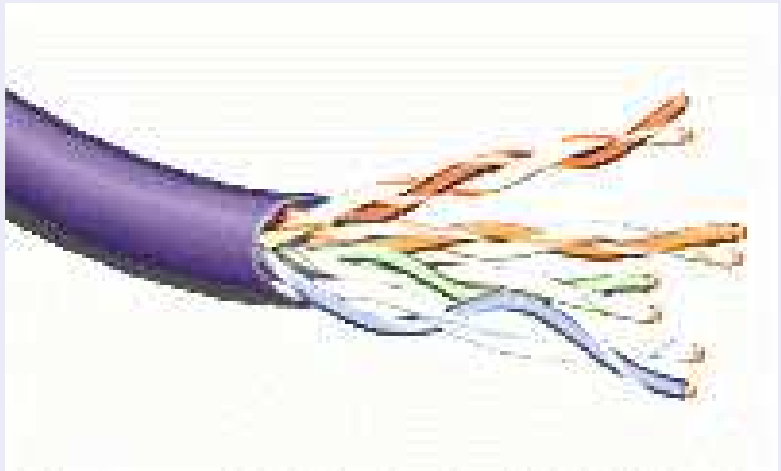
Hinweis:
Der Rauschabstand wird oft in dB angegeben. Es gilt die folgende Umrechnung:

$$[S/N] = 10^{[\text{Wert in dB}]/10}$$

Kabeltypen – Kupferkabel

Twisted Pair

- schon länger gebräuchlich z. B. als Telefonkabel
- besteht aus je zwei isolierten Kupferdrähten (ca. 1 mm dick)
- zur Verminderung des Antenneneffekts und Übersprechens spiralförmig umeinander gewickelt

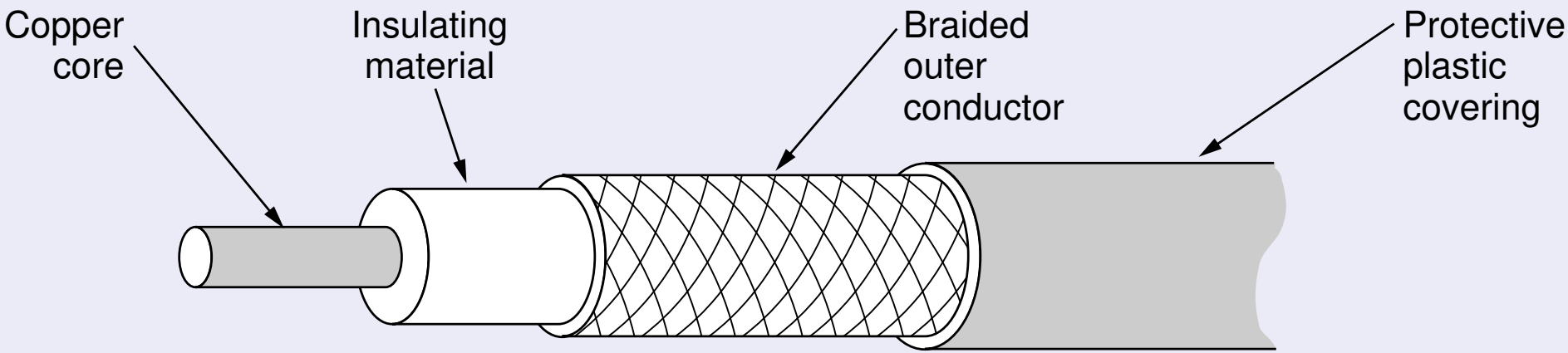


- Ethernet-Kabel: vier Adernpaare
- Varianten: Unshielded (UTP)
  und Shielded Twisted Pair (STP)
  $\Rightarrow$ einzeln und/oder gesamt geschirmt
- Kategorien: gebräuchlich sind
  - Cat-5 (bzw. Cat 5e) (UTP; 100 MHz Bandbreite; RJ-45; max. 1 Gbps)
  - Cat-6 (UTP; 250 MHz; RJ-45; max. 1 Gbps)
  - Cat 6a (STP; 500 MHz; RJ-45; 10 Gbps)
  - Cat 7 (STP; 600 MHz; GG45/Tera; 10 Gbps)
  - Cat 8 (STP; 1600 MHz; 40 Gbps)

Kabeltypen – Kupferkabel (Forts.)

Koaxialkabel

- bekannt vom Fernsehkabel (75 Ohm)

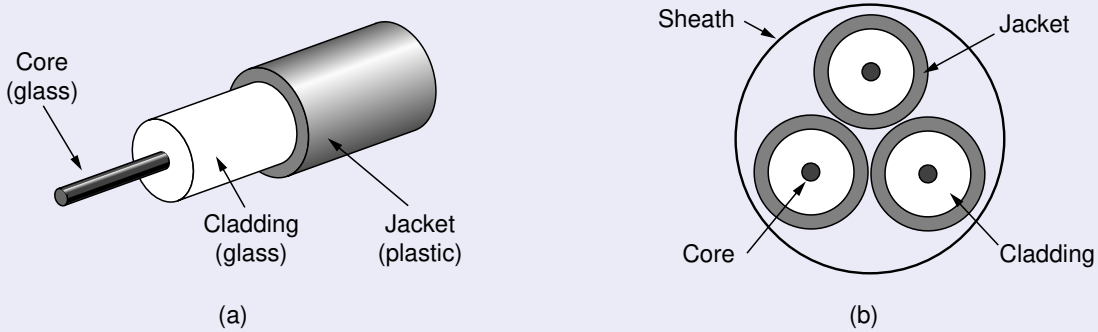


- Einsatz z. B. beim früheren Ethernet
- Koax erlaubt größere Bandbreiten und damit höhere Datenraten über größere Distanzen als Twisted Pair. Heute wird es weitestgehend durch Glasfaserkabel ersetzt.
- **Datenraten:** bis zu 2 Gbps (bei ca. 1 km)

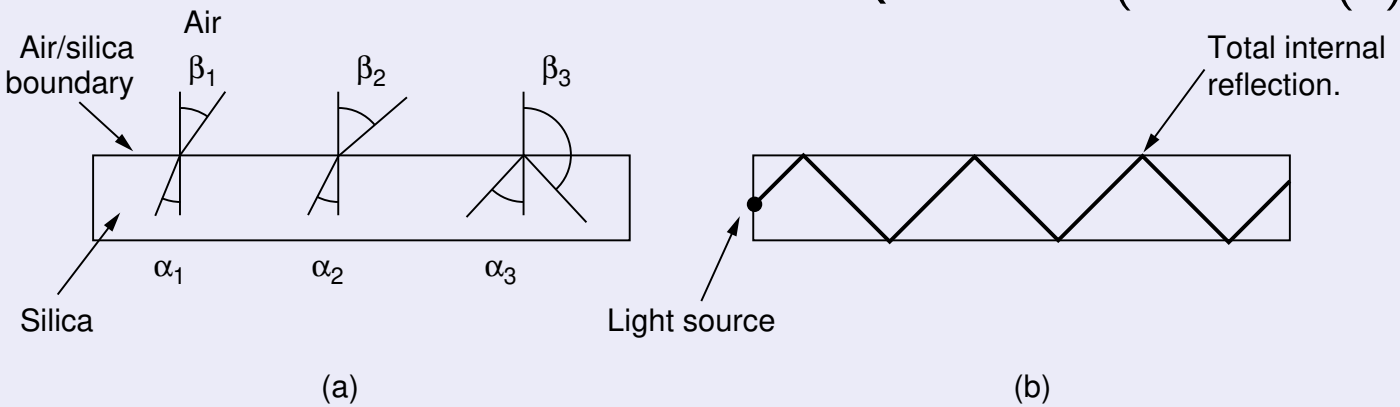
Kabeltypen (Forts.)

Glasfaser (*Fibre*)

- **Datenraten (pro Faser):**
  150 Tbps (theor.), 32 Tbps (AT&T: 580 km; Labor), 10 Gbps (Praxis)
- **Kabelaufbau:**



- Informationsübertragung mittels **optischer** Signale auch um Kurven
  $\Rightarrow$ Ausnutzen der *Totalreflektion* in einer Quarzfaser (im Bild: (b))



Kabeltypen (Forts.)

Vorteile von Glasfaser

- **hohe Datenraten**
  ⇒ Multimedia-Anwendungen
- **sehr geringe Dämpfung**
  ⇒ das Signal muss seltener verstärkt werden ⇒ weniger Repeater nötig
- **störungsunempfindlich**
  ⇒ keine Beeinflussung durch andere Kabel, elektromagnetische Wellen, usw.
- **leicht und billig**
  ⇒ Kupferkabel werden großflächig ersetzt

Apollo (Einsatzbeispiel):
Seekabelsystem im Atlantik zwischen Europa und Nordamerika.
Länge: 12 315 km,
Durchsatz: max. 3,2 Tbps.

Frage: Gibt es auch Nachteile?

Drahtlose Übertragung

Elektromagnetische Wellen

- bewegen sich mit Lichtgeschwindigkeit c.
  Im Vakuum gilt $c_0 = 299\,792\,458$ **m/s**

  Wir rechnen mit $c_0 \approx 3 \cdot 10^8$ **m/s** = 300 000 **km/s**

- haben eine bestimmte Frequenz f
  und eine dazugehörige Wellenlänge $\lambda = c/f$.

Beobachtungen:

- bei niedrigen Frequenzen nur vergleichsweise geringe Datenraten möglich
  $\Rightarrow$ erreichbare Datenraten i.Allg. bei Funk niedriger als im Kabel
- je größer die Wellenlänge, desto geringer die Dämpfung
  $\Rightarrow$ größere Entfernungen möglich

Drahtlose Übertragung (Forts.)

Vergleich drahtloser Techniken

System	max. Datenrate	Luftstrecke
IrDA	115 Kbps	Infrarot
- FIR	4 Mbps	Infrarot
Bluetooth	24 Mbps	Funk
WLAN		Funk
- IEEE 802.11b	11 Mbps	
- IEEE 802.11g	54 Mbps	
- IEEE 802.11n	600 Mbps	
- IEEE 802.11ac	1900 Mbps	
WIMAX	1000 Mbps	Funk
DECT	550 Kbps	Funk
GPRS/EDGE	384 Kbps	Funk
UMTS (3G)	2 Mbps	Funk
HSPA+ (3G+)	80 Mbps	Funk
LTE (4G)	350 Mbps	Funk
LTE-A	1200 Mbps	Funk

Modulationstechniken (Signalkodierung)

Vorhin hatten wir unsere Signale wie folgt kodiert:

'0'-Bit $\rightarrow$ –5 V und '1'-Bit $\rightarrow$ +5 V

Frage: Was ist an dieser Kodierung schlecht?

Einige Probleme:

- Robustheit gegenüber Störungen schlecht
  $\Rightarrow$ hohe Fehleranfälligkeit, mögliche Datenraten gering
- Wie kodiert man Funkstille? Verwendung des 0 V-Pegels bedeutet eine dritte Signalstufe; dies reduziert die max. Datenrate.
- zeitliche Synchronisation schwierig:
  $\Rightarrow$ wieviele Bits enthält eine längere Folge von identischen Bits?

Modulationstechniken (Signalkodierung)

Vorhin hatten wir unsere Signale wie folgt kodiert:

'0'-Bit $\rightarrow$ –5 V und '1'-Bit $\rightarrow$ +5 V

Frage: Was ist an dieser Kodierung schlecht?

Einige Probleme:

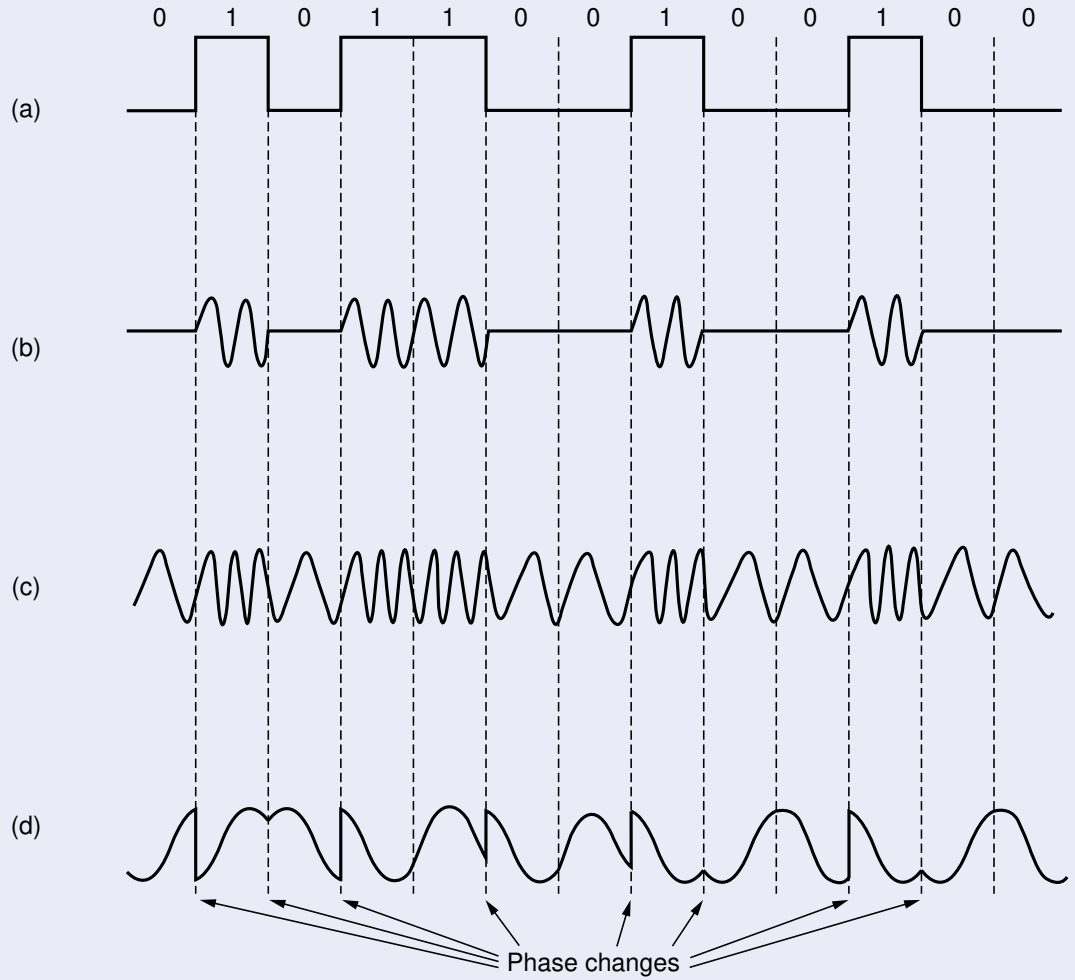
- Robustheit gegenüber Störungen schlecht
  $\Rightarrow$ hohe Fehleranfälligkeit, mögliche Datenraten gering
- Wie kodiert man Funkstille? Verwendung des 0 V-Pegels bedeutet eine dritte Signalstufe; dies reduziert die max. Datenrate.
- zeitliche Synchronisation schwierig:
  $\Rightarrow$ wieviele Bits enthält eine längere Folge von identischen Bits?

Modulationstechniken (Signalkodierung) (Forts.)

Verschiedene Verfahren

(Zu sendende Bits)

- **Amplitudenmodulation**
- **Frequenzmodulation**
- **Phasenmodulation**



Modulationstechniken (Signalkodierung) (Forts.)

Zur Abbildung (vorige Folie)

(b) **Amplitudenmodulation**
Kodierung durch Änderung der Amplitude des Signals:
'1' , falls sich die Amplitude ändert;
'0' , falls sie gleich bleibt.

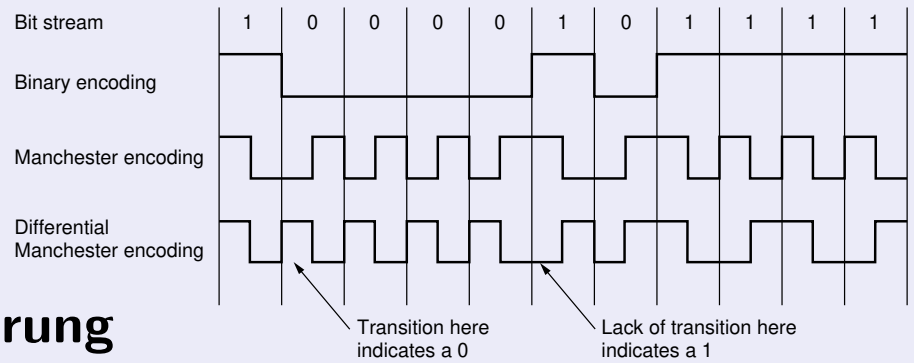
(c) **Frequenzmodulation**
Durch Überlagerung der Trägerfrequenz mit zwei unterschiedlichen Frequenzen zur Bit-Kodierung

(d) **Phasenmodulation**
Durch Kodierung der Bits mittels Änderung der Phase der Trägerwelle (Wechsel zwischen Sinus / Cosinus)

Modulationstechniken (Signalkodierung) (Forts.)

Einige weitere Signalkodierungsverfahren

- **Manchester-Kodierung** (z.B. Ethernet)
  Signalpegelsprung in jedem Bit:
  '1': H $\Rightarrow$ L; '0': L $\Rightarrow$ H
  (Ethernet: H=0.85 V; L=−0.85 V)
  doppelte Rohdatenrate benötigt



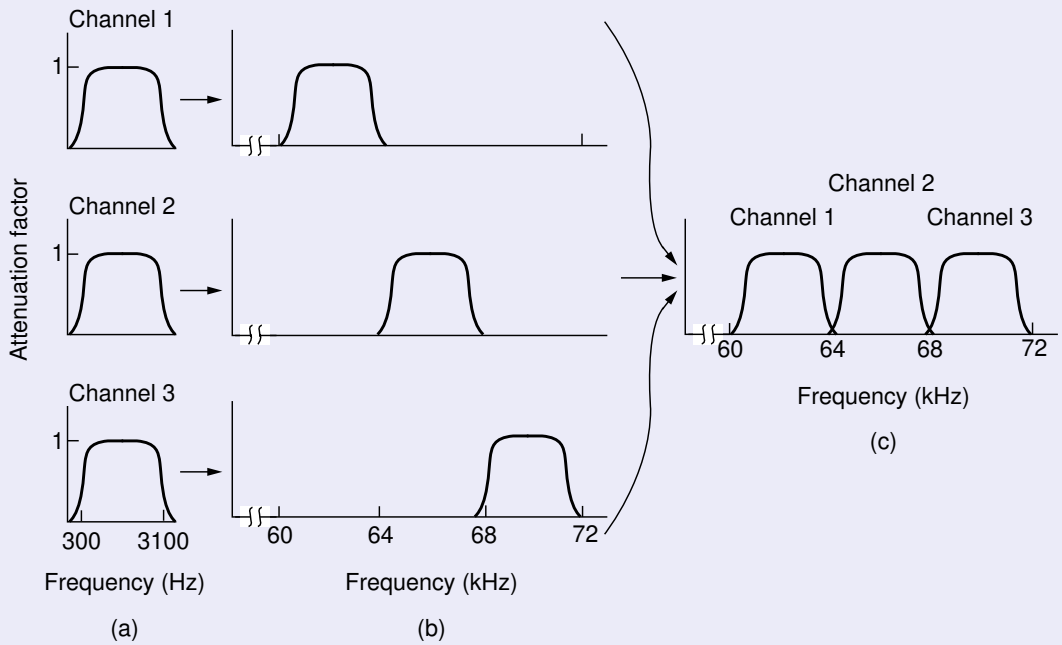
- **Differenzielle Manchester-Kodierung**
  Signalpegelsprung in jedem Bit,
  zusätzl. Signalpegelsprung nur vor '0'-Bit
  rauschbeständiger; komplexer
- **Quadratische Amplituden-Modulation** (QAM)
- **Pulse-Code Modulation** (PCM)
- u. v. m.

Multiplexverfahren

Problemstellung

Wie lässt sich ein Kanal mit großer Bandbreite für parallele Übertragungen nutzen?

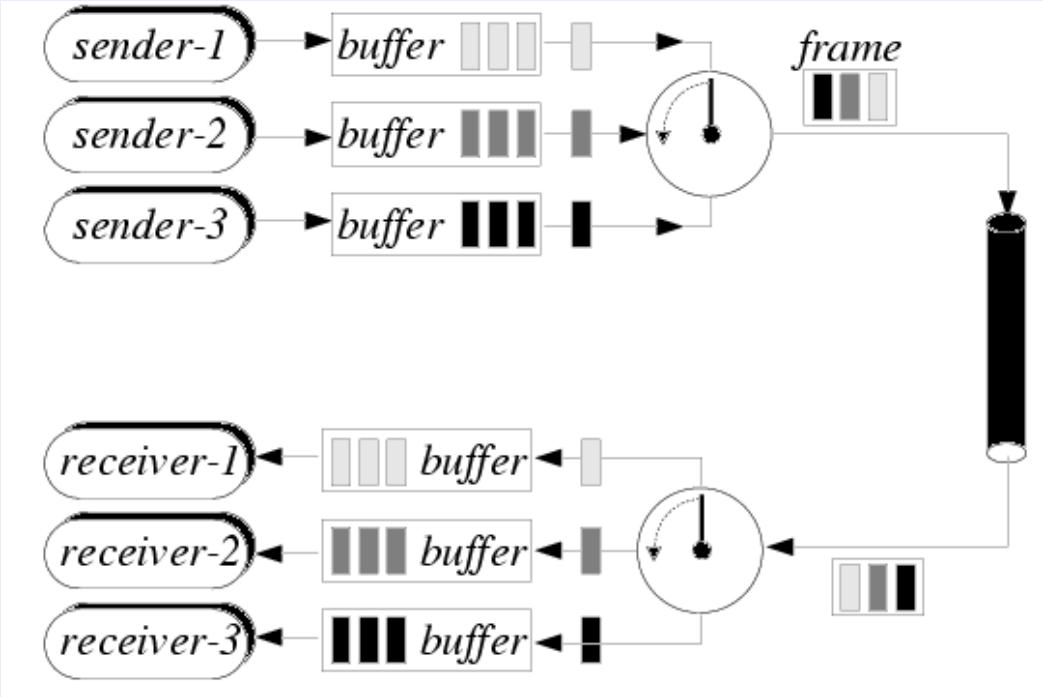
1. Lösung: Frequenzmultiplexing (FDM)



Multiplexverfahren (Forts.)

2. Lösung: Zeitmultiplexing (TDM)

Datenströme mehrerer Sender werden mittels Frames zu einem einzigen zusammengefasst, übertragen und danach wieder auseinanderdividiert und verteilt.



Vorteil: keine Analog-Bauteile notwendig $\Rightarrow$ billiger

Die Sicherungsschicht (*Data Link Layer*)

Aufgaben

- Bereitstellung grundlegender Übertragungsprotokolle
- Sicherstellung einer effizienten Übertragung durch
  - Kanalzugriffssteuerung ($\Rightarrow$ MAC-Teilschicht)
  - Maskierung von Übertragungsfehlern
  - Kompensation unterschiedlicher Geschwindigkeiten von Sender und Empfänger

Einordnung in unser Schichtenmodell

Layer 1: Bitübertragungsschicht (*physical layer*)

Betrifft die Übertragung von Bits in Form von Signalen. Sie beschäftigt sich mit mechanischen bzw. elektrischen Fragen.

Layer 2: Sicherungsschicht (*data link layer*)

Regelt u. a. den Zugriff auf ein gemeinsam benutztes Übertragungsmedium, teilt die Bits in *Data Frames* auf und stellt eine übertragungsfehlerfreie Kommunikation zur Verfügung.

Layer 3: Vermittlungsschicht (*network layer*)

Ist u. a. verantwortlich für das Routing.

Layer 4: Transportschicht (*transport layer*)

Bietet verbindungslose und -orientierte Dienste mit unterschiedlichen Zuverlässigkeitsgarantien an. Sie ist die Schnittstelle für Anwendungen.

Die Verarbeitungsschicht (*application layer*)

implementiert, was dem Benutzer zur Verfügung steht ($\Rightarrow$ E-Mail, Telnet, WWW, . . .).

Frame-Bildung und Fehlerbehandlung als Aufgaben der Sicherungsschicht

Problemstellung

- Die Bitübertragungsschicht macht es sich sehr leicht:

  Sie transportiert einfach nur Bits so schnell wie möglich und so gut es geht vom Sender zum Empfänger.
- Ab und zu geht dabei aber was schief:
  Es werden zu viele, zu wenige oder verfälschte Bits übertragen.
- Die Sicherungsschicht muss das korrigieren!
  $\Rightarrow$ Größere Gruppen von Bits werden hierfür zu *Frames* zusammengefasst. (**Wieso?**)

  $\Rightarrow$ **Frames** sind die Dateneinheit der Sicherungsschicht

Frage: Wie unterteilt man am besten einen Bitstrom in einzelne Frames?

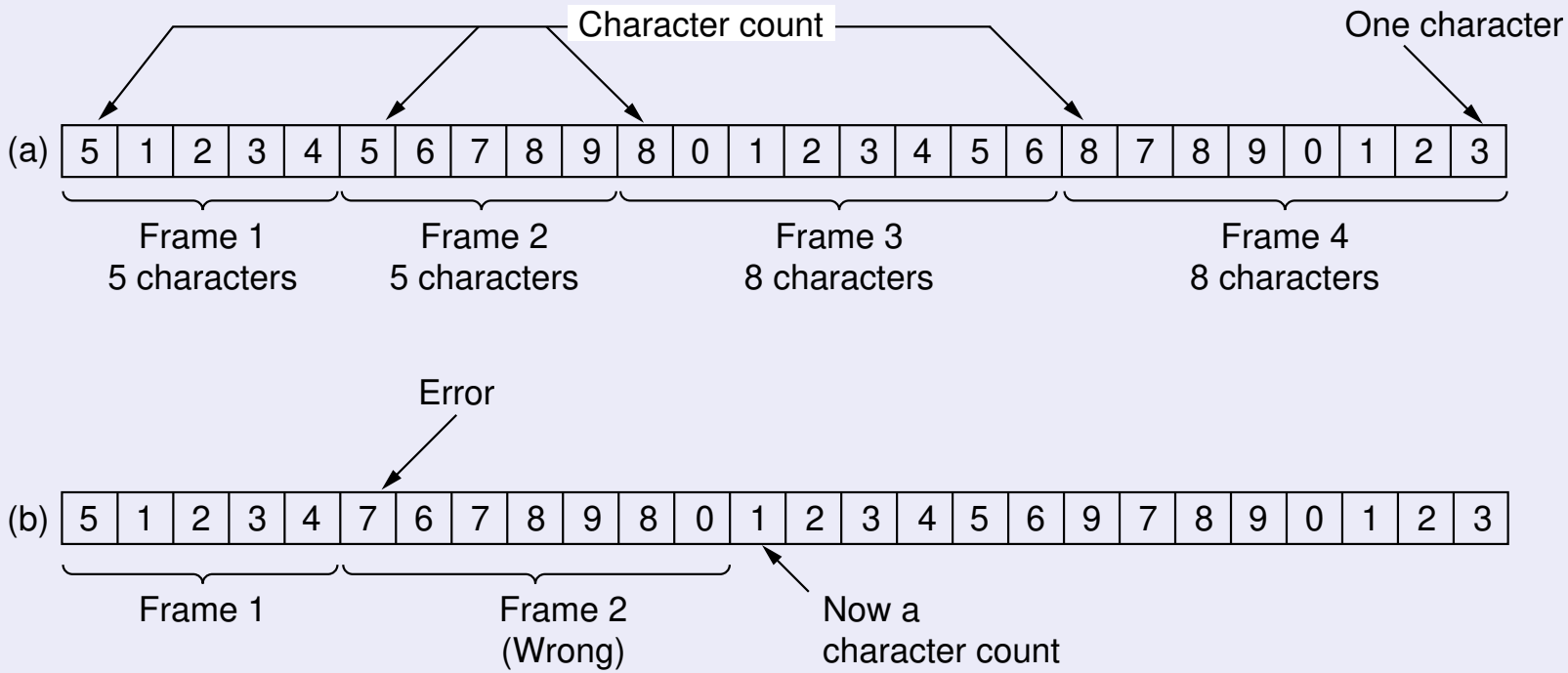
Bilden von Frames

Lösungsversuche

(1) Frames alle mit einheitlicher Länge (Beispiel: ATM-Zelle) ⇒ Effizienz?

(2) durch Pausen zwischen den Frames ⇒ Robustheit? Effizienz?

(3) durch Zählen der Zeichen pro Frame:



Bilden von Frames: Stuffing

Character-Stuffing

Idee: Innerhalb der Sicherungsschicht werden Anfang und Ende jedes Frames mit speziellen **Escape-Sequenzen** markiert:

`DLE STX` (*Data Link Escape, Start of TeXt*)
bzw. `DLE ETX` (*Data Link Escape, End of TeXt*).

Problem: Wie codiert man im Zeichenstrom auftretende `DLE`s???

Bilden von Frames: Stuffing

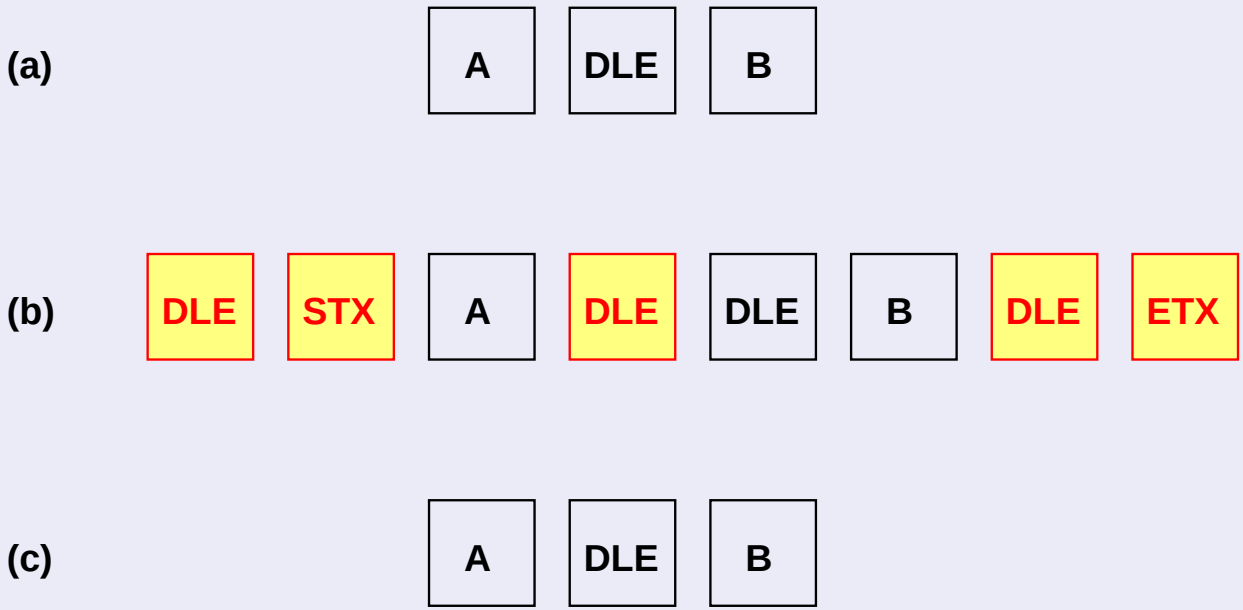
Character-Stuffing

Idee: Innerhalb der Sicherungsschicht werden Anfang und Ende jedes Frames mit speziellen **Escape-Sequenzen** markiert:

`DLE STX` (*Data Link Escape, Start of TeXt*)
bzw. `DLE ETX` (*Data Link Escape, End of TeXt*).

Problem: Wie codiert man im Zeichenstrom auftretende `DLE`s???

Lösung:



Nachteile: nur Zeichen-Frames; auf Zeichencode (ASCII) festgelegt.

Einschub: ASCII-Zeichencodes

"American Standard Code for Information Interchange"

0	nul	1	soh	2	stx	3	etx	4	eot	5	enq	6	ack	7	bel
8	bs	9	ht	10	nl	11	vt	12	np	13	cr	14	so	15	si
16	dle	17	dc1	18	dc2	19	dc3	20	dc4	21	nak	22	syn	23	etb
24	can	25	em	26	sub	27	esc	28	fs	29	gs	30	rs	31	us
32	sp	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	del

Bilden von Frames: Stuffing (Forts.)

Bit-Stuffing

Idee: Anfang und Ende des Frames werden mittels eines speziellen Bitmusters markiert (01111110).
Problem: Wie lassen sich im Datenstrom auftretende 01111110-Sequenzen codieren???
Lösung: Nach jeweils fünf '1' wird eine '0' eingefügt.

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 **0** 1 1 1 1 1 **0** 1 1 1 1 1 **0** 1 0 0 1 0

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Vorteile: beliebige Zahl von Bits pro Frame; Verwenden beliebiger Zeichencodes (ASCII, EBCDIC, ...).

Fehlererkennung und -korrektur

Wie erkennt der Empfänger Übertragungsfehler?

- Frames werden vom Sender als *Codewörter* geeignet codiert übertragen.
- Die Menge aller Codewörter heißt *Code*.

Definition: *Hamming-Distanz* zweier Codewörter

Die Hamming-Distanz zweier Codewörter ist die Anzahl der Bits, in denen sich die beiden unterscheiden. Beispiel:

$$\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \quad \oplus \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array} \quad \Rightarrow \texttt{Hamming-Distanz = 3}$$

Definition: *Hamming-Distanz* eines Codes

Die Hamming-Distanz eines Codes ist $\boldsymbol{d}$, wenn zwei beliebige Codewörter des Codes eine Hamming-Distanz von $\boldsymbol{d}$ oder größer besitzen.

Fehlererkennung und -korrektur (Forts.)

Zwei wichtige Sätze

(1) Um Bereiche mit e oder weniger Einzelbit-Fehlern sicher **erkennen** zu können, ist es notwendig und hinreichend, wenn der Code eine Hamming-Distanz von mindestens $e + 1$ besitzt.

Beispiel: Parity-Bit, $e =$???

(2) Um Bereiche mit k oder weniger Fehlern sicher **korrigieren** zu können, ist es notwendig und hinreichend, wenn der Code eine Hamming-Distanz von mindestens $2k + 1$ besitzt.

Beispiel: Konstruieren Sie einen Code mit 4 Codewörtern und einer Hamming-Distanz von 5. Wieviele Fehler können korrigiert werden?

Fehlerkorrektur

ist jedoch meistens zu aufwändig!
Daher: Fehlererkennung & wiederholte Übertragung (Vorauss.: Bestätigungen)

CRC – ein Fehlererkennungscode

CRC: Cyclic Redundancy Check – Herleitung

(1) Zu einem Bitstring $a =< a_{m-1}, a_{m-2}..., a_0 >,\ a_i \in \{0, 1\}$ wird das folgende Polynom vom Grad $m - 1$ konstruiert:

$$a(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_0x^0$$

Beispiel: Für <11010> konstruieren wir also $a(x) = x^4 + x^3 + x$.

(2) Nun brauchen wir ein Generatorpolynom des Grades k:

$$G(x) = g_kx^k + g_{k-1}x^{k-1} + ... + g_1x + g_0, \text{ mit } g_0, g_k \neq 0$$

Beispiel: Wir wählen $G(x) = x^3 + x^2 + 1$. ($\Rightarrow$ **Frage:** Welcher Grad?)

CRC – ein Fehlererkennungscode (Forts.)

CRC Herleitung (Forts.)

(3) Zu jedem $\boldsymbol{a} =< \boldsymbol{a_{m-1}}, \boldsymbol{a_{m-2}}..., \boldsymbol{a_0} >$ existiert ein Codewort $\boldsymbol{b} =< \boldsymbol{b_{n-1}}, \boldsymbol{b_{n-2}}..., \boldsymbol{b_0} >$ mit $\boldsymbol{n} = \boldsymbol{m} + \boldsymbol{k}$, so dass

$$\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}) \cdot \boldsymbol{G}(\boldsymbol{x}) \quad \text{in GF(2)}$$

Beispiel:

$$\begin{aligned} \boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x})\boldsymbol{G}(\boldsymbol{x}) &= (\boldsymbol{x}^4 + \boldsymbol{x}^3 + \boldsymbol{x})(\boldsymbol{x}^3 + \boldsymbol{x}^2 + 1) \\ &= \boldsymbol{x}^7 + 2\boldsymbol{x}^6 + \boldsymbol{x}^5 + 2\boldsymbol{x}^4 + 2\boldsymbol{x}^3 + \boldsymbol{x} \\ &= \boldsymbol{x}^7 + \boldsymbol{x}^5 + \boldsymbol{x} \end{aligned}$$

Das resultierende Codewort ist also $\boldsymbol{b}$ =<10100010>

CRC – ein Fehlererkennungscode (Forts.)

CRC Herleitung (Forts.)

(4) Der Empfänger erhält $b'(x)$ und dekodiert mittels Division durch $G(x)$ zu $a'(x)$ und $E(x)$:

$$b'(x) = a'(x)G(x) + E(x)$$

$$a' = < a'_{m-1}, a'_{m-2} \ldots, a'_0 >$$
$$E = < e_{k-1}, e_{k-2} \ldots, e_0 >$$

Beispiel: (a) ohne Fehler:
b' =<10100010>
$a'(x) = b'(x)/G(x)$, Rest $E(x)$
$\Rightarrow a = a'$ =<11010>, E =<000>

(b) mit Fehler:
b' =<**0**0100010>
$a'(x) = b'(x)/G(x)$, Rest $E(x)$
$\Rightarrow a'$ =<00111>$\neq a$, E =<001>

Nachteile

- Codewörter b enthalten nicht unmittelbar die Ursprungsdaten a
- Sender und Empfänger verwenden unterschiedliche Algorithmen
  $\Rightarrow$ bisherigen Algorithmus verbessern! (s. nächste Folie)

CRC – ein Fehlererkennungscode (Forts.)

CRC – der endgültige Algorithmus

Gegeben: Frame $\boldsymbol{F}$ der Länge $\boldsymbol{m}$ und $\boldsymbol{G}(\boldsymbol{x})$ vom Grad $\boldsymbol{r}$.

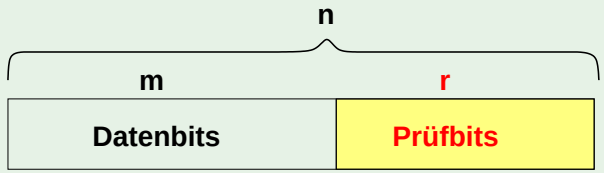
(1) Hänge $\boldsymbol{r}$ Nullbits an das Frame an $\Rightarrow \boldsymbol{F'}(\boldsymbol{x})$.

(2) Berechne den Rest $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{F'}(\boldsymbol{x}) / \boldsymbol{G}(\boldsymbol{x})$ (in GF(2))

(3) Übertrage $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{F'}(\boldsymbol{x}) + \boldsymbol{R}(\boldsymbol{x})$ (in GF(2))

Vorteile

- Sender und Empfänger müssen lediglich die Division implementieren
- leistungsfähige Implementierung in Hardware durch *Schieberegister*
- $\boldsymbol{T} \equiv \boldsymbol{F} \circ \boldsymbol{R}$, d. h. das gesendete Codewort enthält gesondert sowohl die Daten- als auch die Prüfbits.



CRC – ein Fehlererkennungscode (Forts.)

Generatorpolynom im IEEE 802 Standard (u. a. Ethernet und WLAN)

- $\boldsymbol{G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1}$
- Hamming-Distanz = 4 bei normaler Nachrichtenlänge
- Erkennt **Fehlerbündel** der Länge 32 oder kleiner, sowie ungerader Längen.
- Es gibt bessere Generatorpolynome vom Grad 32 (Hamming-Distanz = 6).

Beispiel: Berechnen Sie mit dem endgültigen Algorithmus:

$\boldsymbol{G(x) = x^4 + x + 1}$
$\boldsymbol{a} = 11010\,11011$

Welches $\boldsymbol{b}$ *wird gesendet?*

Flusskontrolle

Szenario

Ein schneller Quadcore-Desktop-Rechner, angebunden über 10 Gbps-Ethernet, dient als leistungsfähiger Video-Server. Wenn ich nun mit einem Raspi einen Film abrufe, kommt die kleine Kiste nicht ohne Weiteres nach!

Problem

Frames werden schneller gesendet als sie der Empfänger verarbeiten kann
$\Rightarrow$ in der Folge Datenverlust – selbst bei Pufferung!

Lösung

Die Übertragungsrate muss individuell angepasst werden:

Der Empfänger informiert den Sender inwieweit er die angekommenen Frames schon **verarbeitet** hat $\Rightarrow$ Flusskontrolle
Ist der Empfänger **langsam**, **senkt** der Sender die Übertragungsrate.
Wird der Empfänger **schneller**, kann die Übertragungsrate **erhöht** werden.

Hierzu brauchen wir Zwischenpuffer und Rückmeldungen (ACKs).

Flusskontrolle

Das wollen wir uns mal genauer ansehen:

Datenstrukturen (Prog.-Sprache C)

```
1
2  typedef enum{false, true} boolean;
3  typedef unsigned int seq_no;
4  typedef struct{char data[MAX_PKT]} packet;
5  typedef enum{dat, ack} frame_kind;
6  typedef enum{frame_arrival} event_type;
7  typedef struct{
8    frame_kind kind; /* Frame-Typ              */
9    seq_no seq;      /* Sequenznummer         */
10   seq_no ack;      /* Bestätigungsnummer    */
11   packet info;     /* Daten aus der
12                       Vermittlungsschicht */
13 } frame
```

Flusskontrolle (Forts.)

Erster Versuch – noch ohne Flusskontrolle:

```
1  sender1() {
2    frame s;
3    packet buf;
4
5    while(true) {
6      from_network_layer(&buf);
7      s.info = buf;
8      to_physical_layer(&s);
9    }
10 }
```

```
1  receiver1() {
2    frame r;
3    event_type evt;
4
5    while(true){
6            /* warte auf Frame */
7      wait_for_event(&evt);
8      from_physical_layer(&r);
9      to_network_layer(r.info);
10   }
11 }
```

Flusskontrolle (Forts.)

Zweiter Versuch:

```
1  sender2() {
2    frame s;
3    packet buf;
4    event_type event;
5
6    while(true){
7      from_network_layer(&buf);
8      s.info = buf;
9      to_physical_layer(&s);
10     wait_for_event(&event);
11   }
12 }
```

```
1  receiver2() {
2    frame r,s;
3    event_type event;
4
5    while(true){
6      wait_for_event(&event);
7        /* warte auf Frame */
8      from_physical_layer(&r);
9      to_network_layer(r.info);
10     to_physical_layer(&s);
11       /* sende ACK */
12   }
13 }
```

Frage: Wie funktioniert dieses Protokoll? Wie gut ist es?

Flusskontrolle (Forts.)

Das obige Protokoll funktioniert nur, wenn die Übertragung fehlerfrei ist. Was ist, wenn Frames verfälscht werden oder vollständig verloren gehen?

Probleme

- Ein Sender weiß nicht, ob ein Frame korrekt angekommen ist.
  Lösung: Der Empfänger schickt eine Bestätigung (ACK).
- Bestätigungen können ebenfalls verlorengehen!
  Lösung: Der Sender hat einen Timer; nach Auftreten eines Timeouts wiederholt er alle unbestätigten Frames.
- Der Empfänger erhält nun manche Frames mehrfach; er kann jedoch Duplikate nicht als solche erkennen.
  Lösung: Sequenznummern für Frames verwenden.
- Sequenznummern sind ein Header-Feld und daher nicht beliebig lang. Sie laufen irgendwann einmal über.
  Lösung: Zur Zeit genügen 0 und 1.
- Bei größeren RTTs ist das Protokolls nicht sehr effizient.
  Lösung: Puffer einbauen (später ...) .

Flusskontrolle (Forts.)

Dritter Versuch (Stop-and-Wait):

```
1  sender3() {
2    frame s;
3    packet buf;
4    event_type event;
5    seq_no next_no;
6
7    next_no = 0;
8    from_network_layer(&buf);
9    while(true){
10     s.info = buf;
11     s.seq = next_no;
12     to_physical_layer(&s);
13     start_timer(s.seq);
14     wait_for_event(&event);
15     if(event==frame_arrival){
16       from_network_layer(&buf);
17       next_no = 1 - next_no;
18     }
19   }
20 }
```

```
1  receiver3() {
2    frame r,s;
3    event_type evt;
4    seq_no expec_no;
5
6    expec_no = 0;
7    while(true){
8      wait_for_event(&evt);
9      if(evt==frame_arrival){
10       from_physical_layer(&r);
11       if(r.seq==expec_no){
12         to_network_layer(r.info);
13         expec_no=1 - expec_no;
14       }
15       to_physical_layer(&s);
16         /* sende ACK */
17     }
18   }
19 }
```

Leider immer noch langsam ...

Flusskontrolle (Forts.)

Von Simplex zu Duplex

Der Datenfluss beiden bisherigen Simplex-Protokollen verläuft nur in eine Richtung. Bei Verwendung eines **Duplex**-Protokolls können Frames in **beide Richtungen** transportiert werden.

Piggybacking

Wenn ohnehin Daten-Frames übertragen werden, dann können noch zu verschickende ACKs und NAKs einfach drangehängt werden.

Frage: Welche Vor- und Nachteile hat das?

Duplex-Protokoll mit 1-Bit Sequenznummern

Dritter Versuch (Stop-and-Wait; Duplex):

```
1  protocol4() {
2    /* Voraussetzung: 1. Frame wurde bereits gesendet. */
3    while(true) {
4      wait_for_event(&event);
5      if(event == frame_arrival){
6        from_physical_layer(&r);
7        if(r.seq == expec_no){
8          to_network_layer(&r.info);
9          expec_no = 1 - expec_no);
10       }
11       if(r.ack == next_no){
12         from_network_layer(&buf);
13         next_no = 1 - next_no;
14       }
15     }
16     s.info = buf;
17     s.seq = next_no;
18     s.ack = 1 - expec_no;     /* piggy backing */
19     to_physical_layer(&s);
20     start_timer(s.seq);
21   } /* while */
22 }
```

Flusskontrolle: Sliding-Window

Vorgehensweise

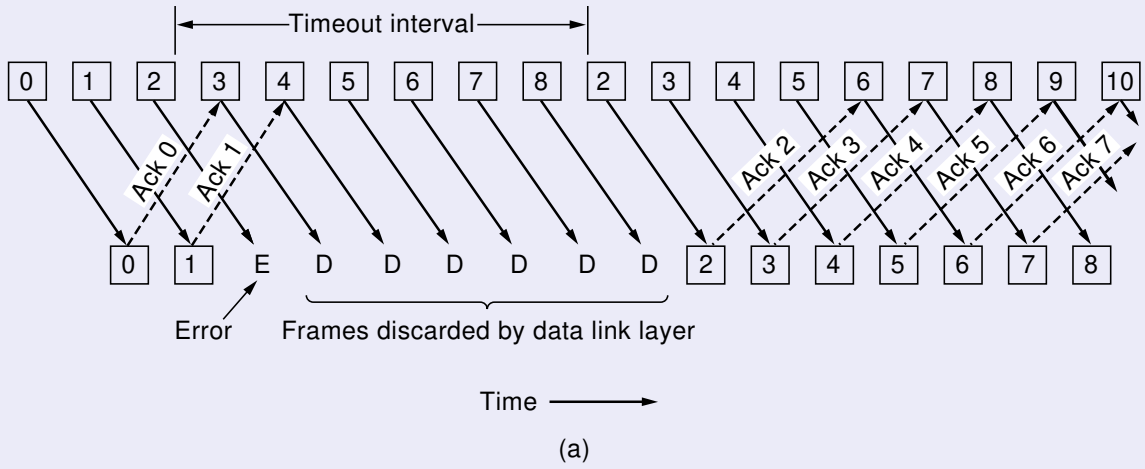
- Anstatt die Frames einzeln zu senden, darf der Sender eine größere Anzahl von aufeinanderfolgenden Frames (das so genannte **Sendefenster**) versenden. Frames bleiben immer so lange im Sendefenster gespeichert, bis ein passendes ACK eingetroffen ist.
- Der Empfänger besitzt analog ein **Empfangsfenster**, das eine bestimmte Anzahl an Frames speichern kann. Von hier holt die darüberliegende Schicht die Frames ab.
- Beschädigte Frames bleiben so lange im Empfangsfenster stehen, bis eine korrekte Version empfangen wurde.
- Frame Nr. k bleibt so lange im Empfangsfenster stehen bis Frame Nr. $k - 1$ verarbeitet wurde. **Wieso?**
- Neue Frames, die nicht mehr ins Empfangsfenster passen, werden kommentarlos zerstört.

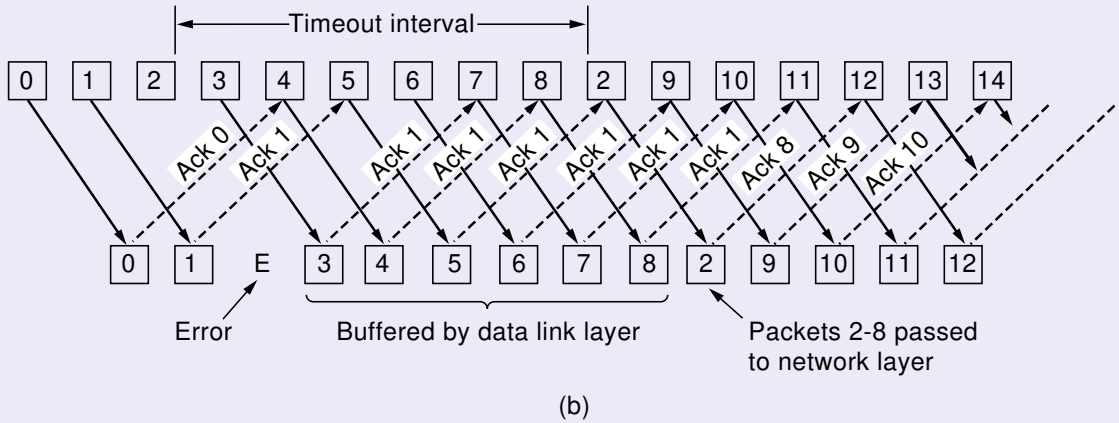
Frage: Welche Beziehung gibt es zwischen Fenstergröße und Übertragungszeit?

n-Bit Sliding-Window-Protokolle

Mögliche Auswirkung von Fehlern

(a) Großes Sendefenster, Empfangsfenstergröße 1





(b) Großes Sendefenster, großes Empfangsfenster

n-Bit Sliding-Window-Protokolle (Forts.)

Varianten des Protokolls:
Wie verhält sich der Empfänger, wenn ein beschädigter Frame ankommt?

Go-back *N*:

Der Empfänger verwirft alle inzwischen angekommenen neuen Frames. Der Sender muss die Übertragung dieses Frames und aller nachfolgenden wiederholen.

Größe des Sendesfensters bei n-Bit Sequenznummern:

$$2^n - 1 \text{ Frames}$$

Selective Repeat:

Der Empfänger verlangt, dass nur der beschädigte Frame nochmals gesendet wird (Achtung: Die Sicherungsschicht des Empfängers muss dennoch alle erhaltenen Frames in der korrekten Reihenfolge ausliefern!)

Größe des Sende- bzw. Empfangsfensters bei n-Bit Sequenznummern:

$$2^{n-1} \text{ Frames}$$

n-Bit Sliding-Window-Protokolle (Forts.)

Bewertung

Go-back *N* ist sehr einfach. Der Empfänger muss sich nichts weiter merken: Nach einem fehlerhaften Frame werden alle weiteren ignoriert.
Der Sender behält die Frames im Sendefenster bis sie bestätigt wurden.

(**Frage:** Was passiert, wenn das Sendefenster voll ist?)

Selective Repeat ist meistens effizienter, weil bereits übertragene korrekte Frames nicht verworfen werden. Allerdings braucht nun der Empfänger ein Empfangsfenster, und der Verwaltungsaufwand ist höher als bei Go-back-***N***.
(Achtung bei Folge-Time-Outs!)

Frage: Welche Einsatzgebiete sehen Sie jeweils?

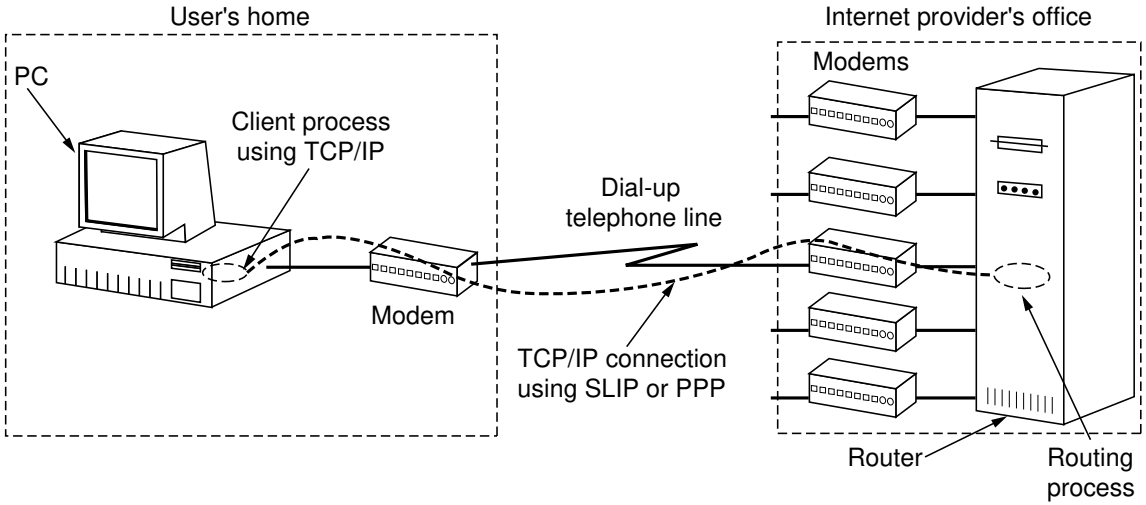
n-Bit Sliding-Window-Protokolle (Forts.)

Mögliche Optimierungen

- Verwendung von negativen Bestätigungen (NAK) bei fehlerhaft angekommenen Frames (CRC-Fehler), um Time-Out-Zeiten zu verkürzen.
- Empfänger wiederholt ein Frame, nachdem er mehrere identische ACKs erhalten hat, um Time-Out-Zeiten zu reduzieren. ($\Rightarrow$ Problem?)
- Verwendung von selektiven Bestätigungen (SACK), um unnötige Wiederholungen zu vermeiden (nur bei großem Empfangsfenster).

Beispiel: Sicherungsschicht bei Modemverbindungen

So ähnlich stellte sich früher eine einfache Internet-Verbindung vom heimischen PC über ein Analog-Modem zum Provider dar (max. 36 kbps):



Beispiel: Sicherungsschicht bei Modemverbindungen (Forts.)

Punkt-zu-Punkt

Im Internet wird unter Verwendung des Internet-Protokoll-Stacks eine Punkt-zu-Punkt Verbindung aufgebaut. Hierzu müssen die IP-Pakete aus der Vermittlungsschicht geeignet über die Einwählleitung transportiert werden.
$\Rightarrow$ Die IP-Pakete müssen so in Frames verpackt werden, dass sie am anderen Ende des Kanals nach dem Auspacken transparent an die Vermittlungsschicht weitergegeben werden können!

Mögliche Protokolle sind **SLIP** und **PPP**.

SLIP: Serial Line IP (seit 1984)

- Die Workstation sendet rohe IP-Pakete getrennt durch ein spezielles Flag-Byte (0xC0) über die Leitung $\Rightarrow$ Verwendung von Character-Stuffing.

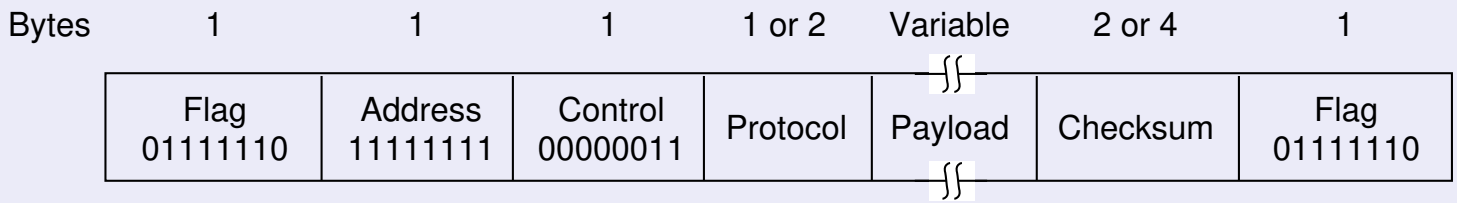
Es gab viele Probleme mit dem alten Protokoll

- SLIP unterstützt nur IP
  $\Rightarrow$ beide Parteien brauchen statische IP-Adressen.
  $\Rightarrow$ nicht einsetzbar in anderen Netzen, z. B. in Novell-Netzen.
- KEINE Fehlerbehandlung
  $\Rightarrow$ Sache der Transportschicht!
- Keine Authentisierung
  $\Rightarrow$ *Mit wem rede ich da eigentlich????*
- Kein zugelassener Standard
  $\Rightarrow$ unzählige inkompatible Varianten.

PPP: Point-to-Point Protocol

(Von der IETF; RFCs 1661–1663)

PPP Frame-Layout



Vorteile

- Saubere Einteilung in Frames durch Anfang- und Endekennung
- Zur Leitungskontrolle (Aufbau, Authentifikation, Testen, Verhandeln, Abbau) existiert ein eigenes Protokoll: *Link Control Protocol (LCP)*.
- Unterstützung zahlreicher *Layer-3*–Protokolle; nicht nur IP
- Unterstützung von dynamischen Netzwerkadressen

PPP: Point-to-Point Protocol (Forts.)

Beispielszenario

Beim Einwählvorgang beim Provider passierte Folgendes:

(1) Anrufen des Providers und Aufbau einer Modemverbindung zum Router

(2) Der PC schickt zunächst eine Reihe von LCP-Paketen (eingebettet in PPP-Frames), um die Art der gewünschten PPP-Verbindung auszuhandeln:
- maximale Frame-Größe
- Authentisierungsmethode (meistens Passwort)
- Überwachung der Leitungsqualität
- Header-Komprimierung

(3) Nun folgen die Aushandlungen der Vermittlungsschicht:
u.a. Beantragung einer dynamischen IP-Adresse beim Provider
$\Rightarrow$ Einsatz von DHCP (*Dynamic Host Configuration Protocol*)

Frage: Warum? Wie funktioniert das?

Medium Access Control: MAC-Sublayer

Um was geht es?

- Bis jetzt haben wir die Funktionalität der Sicherungsschicht und einige Protokolle für Punkt-zu-Punkt-Kommunikation näher betrachtet.
- Nun betrachten wir Netzwerke, die auf **Broadcast**-Kanälen basieren: Was eine Station sendet, können bzw. müssen sich alle anhören!

Problem

Bei einem solchen Kanal können aber mehrere Stationen **gleichzeitig** auf die Idee kommen, Frames zu versenden.
$\Rightarrow$ i. Allg. kommt es zu einer **Kollision**, bei der Frames sich überlagern und zerstören.

Lösung

- Der Kanal muss einer Station exklusiv zugeteilt werden
- Aber wie?
  Die zur Absprache notwendige Kommunikation muss ebenfalls über den Broadcast-Kanal abgewickelt werden!

Die MAC-Teilschicht (Forts.)

Es gibt drei alternative Strategien zur Kanalzuteilung:

- **Mit Reservierung:** Eine Station mit Sendewunsch macht vor dem Senden eine Reservierung ($\Rightarrow$ Wie denn?).
  Protokolle: Bit-Map, Binärer Countdown, ...

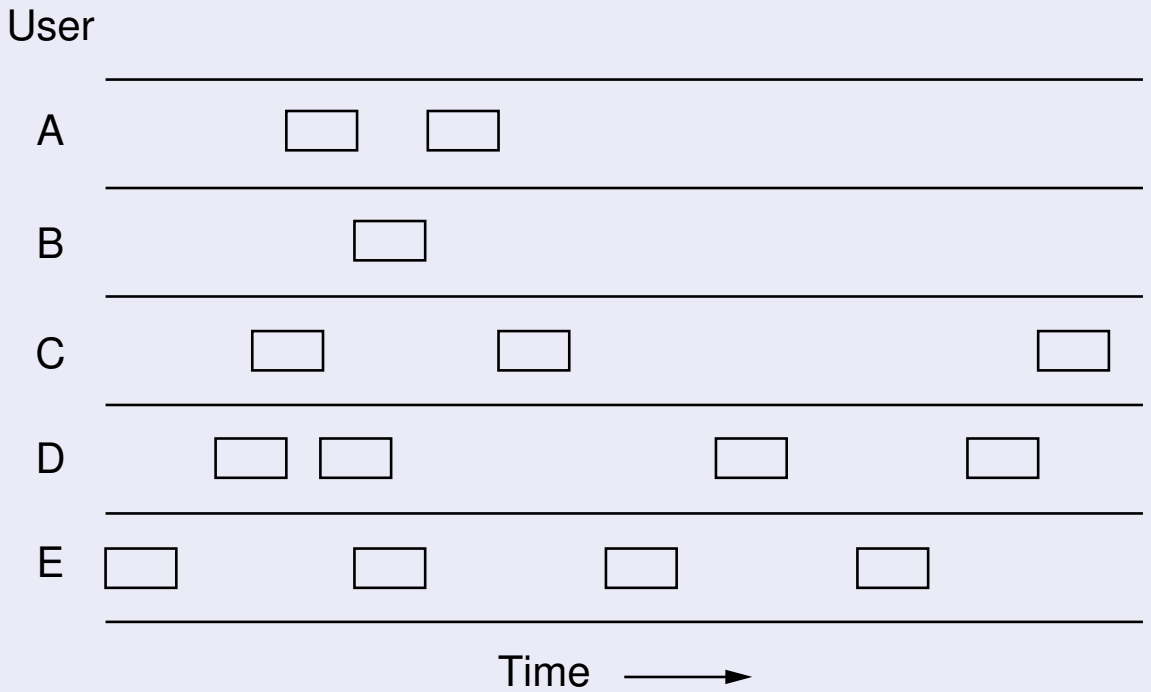
- **Round-Robin:** Der Kanal wird den Stationen in festgelegter Reihenfolge abwechselnd zugeteilt. Realisierung z. B. mittels **Token**.

- **Optimistisch:** Jede Station, die senden will, sendet einfach. Falls eine Kollision auftritt, dann muss eben was getan werden, um die Kollision beheben. Die Kollisionshäufigkeit sollte hier recht gering gehalten werden.
  Protokolle: ALOHA, CSMA, ...

Sehr optimistisch: ALOHA

Was ist, wenn wir nix tun?

Jeder, der ein Frame senden möchte, sendet einfach! Im Fall einer Kollision wird durch die Fehlerkontrolle das Frame (Timeout bzw. NAK) nochmals angefordert. (Protokoll der ersten Funknetze)



Frage: Wie gut funktioniert das?

Statistische Analyse

Voraussetzungen und Definitionen

- Wir betrachten ein Bus-Netzwerk mit unendlich vielen angeschlossenen Stationen.
- Alle Station senden unabhängig voneinander mit einer sehr geringen aber identischen Wahrscheinlichkeit gleichartige Frames.
- Die Zeit, die zur die Übertragung eines kompletten Frames benötigt wird, ist T_{frame} .
- Senden zwei oder mehr Stationen zeitlich überlappend Frames, tritt eine Kollision auf.

Statistische Analyse (Forts.)

Beobachtungen

- Die Sendeereignisse bilden einen sogenannten **Poisson-Prozess**.
- Die Zufallsvariable $\boldsymbol{X}$, die die Anzahl der Sendereignisse während einer festen Zeitspanne beschreibt, ist **Poisson-verteilt** mit Parameter $\boldsymbol{\lambda}$.
  $\Rightarrow \text{Prob}\,(\boldsymbol{X} = \boldsymbol{k}) = \frac{\lambda^k}{\boldsymbol{k}!}\, \boldsymbol{e}^{-\lambda}, \quad \boldsymbol{k} = 0, 1, 2, \ldots$
- $\boldsymbol{\lambda}$ entspricht der innerhalb der festgelegten Zeitspanne im Mittel gesendeten Frames.

- Beispiel: Poisson-Verteilung mit Parameter $\boldsymbol{\lambda} = 0.5$

Anzahl $\boldsymbol{k}$	0	1	2	3	...
Prob $(\boldsymbol{X} = \boldsymbol{k})$	0.607	0.303	0.076	0.0126	...

- Wozu ist die Poisson-Verteilung gut?
  - historisch: Verletzungen durch Huftritte pro Jahr
  - Telefongesellschaften: Anrufe pro Minute
  - Netzwerke: Nachrichtenpakete pro Sekunde
  - ...

Statistische Analyse (Forts.)

Beobachtungen für das ALOHA-Protokoll

- Die Ankünfte von Aufträgen der Vermittlungsschicht bilden einen Poisson-Prozess.

  $\Rightarrow$ Die Zufallsvariable
  "*Anzahl innerhalb von* $\boldsymbol{T_{frame}}$ **neu** *eingetroffener Frames*"
  ist Poisson-verteilt mit Parameter $\boldsymbol{N}$; sinnvollerweise gilt $0 < \boldsymbol{N} < 1$.

- Beim Senden der Frames kann es zu Kollisionen kommen, so dass (alte) Frames wiederholt werden müssen. Die gesamten Sendereignisse der Sicherungsschicht (neue und alte Frames) bilden ebenfalls einen Poisson-Prozess.

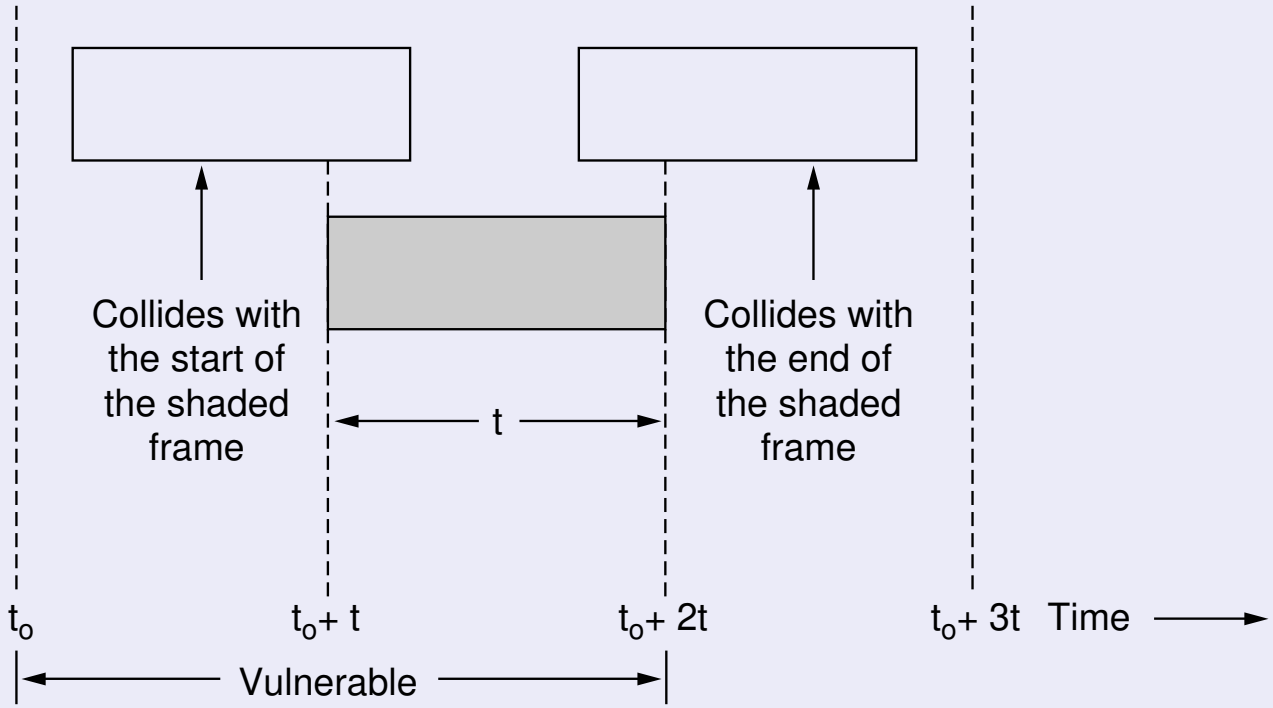
  $\Rightarrow$ Die Zufallsvariable
  "*Anzahl aller innerhalb von* $\boldsymbol{T_{frame}}$ *gesendeten Frames*"
  ist Poisson-verteilt mit Parameter $\boldsymbol{G}$, mit $\boldsymbol{G} \geq \boldsymbol{N}$.

  $\boldsymbol{G}$ heißt **Kanalbelastung**.

Statistische Analyse (Forts.)

Bestimmen des Durchsatzes

- Die Anzahl der während der festgelegten Zeitspanne (i. d. R. T_{frame}) erfolgreich übertragenen Frames heißt **Durchsatz** (des Protokolls).
- Für einen Frame, der gerade in Übertragung ist, gilt:
  Es erfolgt dann und nur dann keine Kollision, wenn während der **gefährlichen Zeitspanne** keine anderen Frames gesendet werden:



Statistische Analyse (Forts.)

Bestimmen des Durchsatzes (Forts.)

- Die Wahrscheinlichkeit, dass unser Frame unbeschädigt übertragen wird, entspricht also der Wahrscheinlichkeit, dass während der Zeitdauer von $2\boldsymbol{T_{frame}}$ genau 0 weitere Frames gesendet werden.
(Die Poisson-Annahme gilt auch für $2\boldsymbol{T_{frame}}$.)

$\Rightarrow$ Diese Wahrscheinlichkeit nennen wir im Folgenden $\boldsymbol{P_0}$ und sie beträgt $\boldsymbol{P_0 = e^{-2G}}$

- Der **Durchsatz** $\boldsymbol{S}$ des ALOHA-Protokolls berechnet sich somit zu

$$\boldsymbol{S = G \cdot P_0 = G \cdot e^{-2G}}$$

$$\Rightarrow \boldsymbol{S_{max}} = 0.184 \text{ für } \boldsymbol{G} = 0.5$$

Statistische Analyse (Forts.)

Bestimmen des Durchsatzes (Forts.)

- Die Wahrscheinlichkeit, dass unser Frame unbeschädigt übertragen wird, entspricht also der Wahrscheinlichkeit, dass während der Zeitdauer von $2\boldsymbol{T_{frame}}$ genau 0 weitere Frames gesendet werden.
(Die Poisson-Annahme gilt auch für $2\boldsymbol{T_{frame}}$.)

$\Rightarrow$ Diese Wahrscheinlichkeit nennen wir im Folgenden $\boldsymbol{P_0}$ und sie beträgt $\boldsymbol{P_0 = e^{-2G}}$

- Der **Durchsatz** $\boldsymbol{S}$ des ALOHA-Protokolls berechnet sich somit zu

$$\boldsymbol{S = G \cdot P_0 = G \cdot e^{-2G}}$$

$$\Rightarrow \boldsymbol{S_{max}} = 0.184 \text{ für } \boldsymbol{G} = 0.5$$

Slotted ALOHA

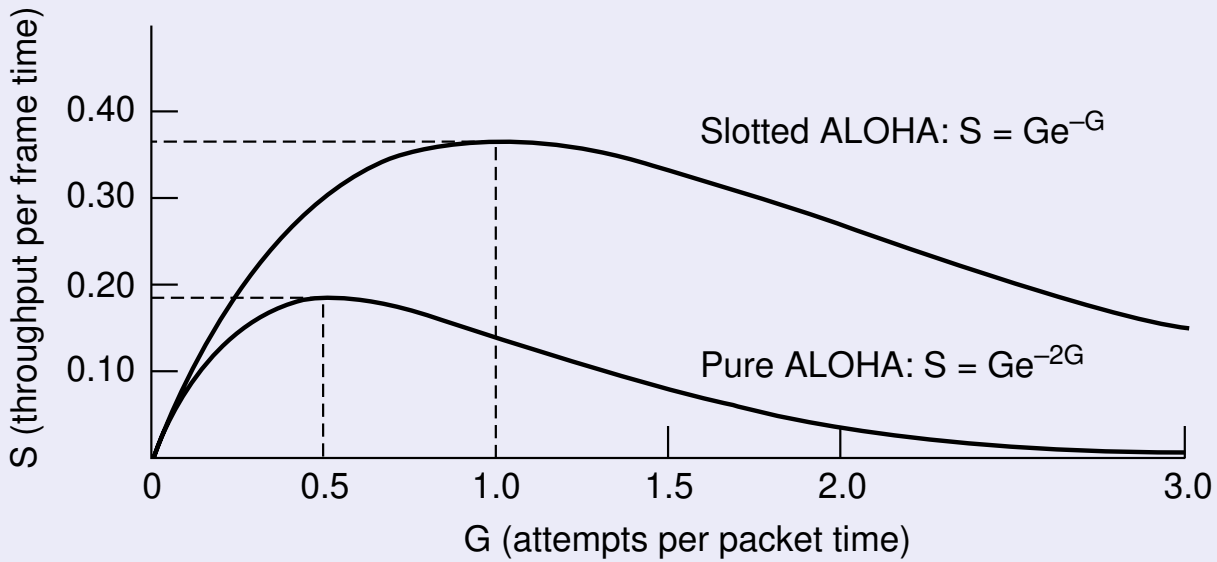
Verbesserung

Wir unterteilen den Kanal in Slots und legen fest, dass Übertragungen nur zu Beginn eines Slots starten dürfen.

$\Rightarrow$ Der Durchsatz ist nun $\boldsymbol{S = G \cdot e^{-G}}$ (**Frage:** Wieso?)

Analyse:

Für $\boldsymbol{G} = 0.5$ ergibt sich $\boldsymbol{S} = \frac{1}{2\sqrt{e}} = 0.303$ sowie $\boldsymbol{S_{max}} = 0.368$ für $\boldsymbol{G} = 1$.



Allerdings: Synchronisation der Stationen technisch aufwändiger!

CSMA (Carrier Sense Multiple Access)

Idee

Bevor ein Frame gesendet wird, lauscht der Sender am Kanal, ob dieser gerade frei ist. $\Rightarrow$ Viel besser als ALOHA!

Varianten

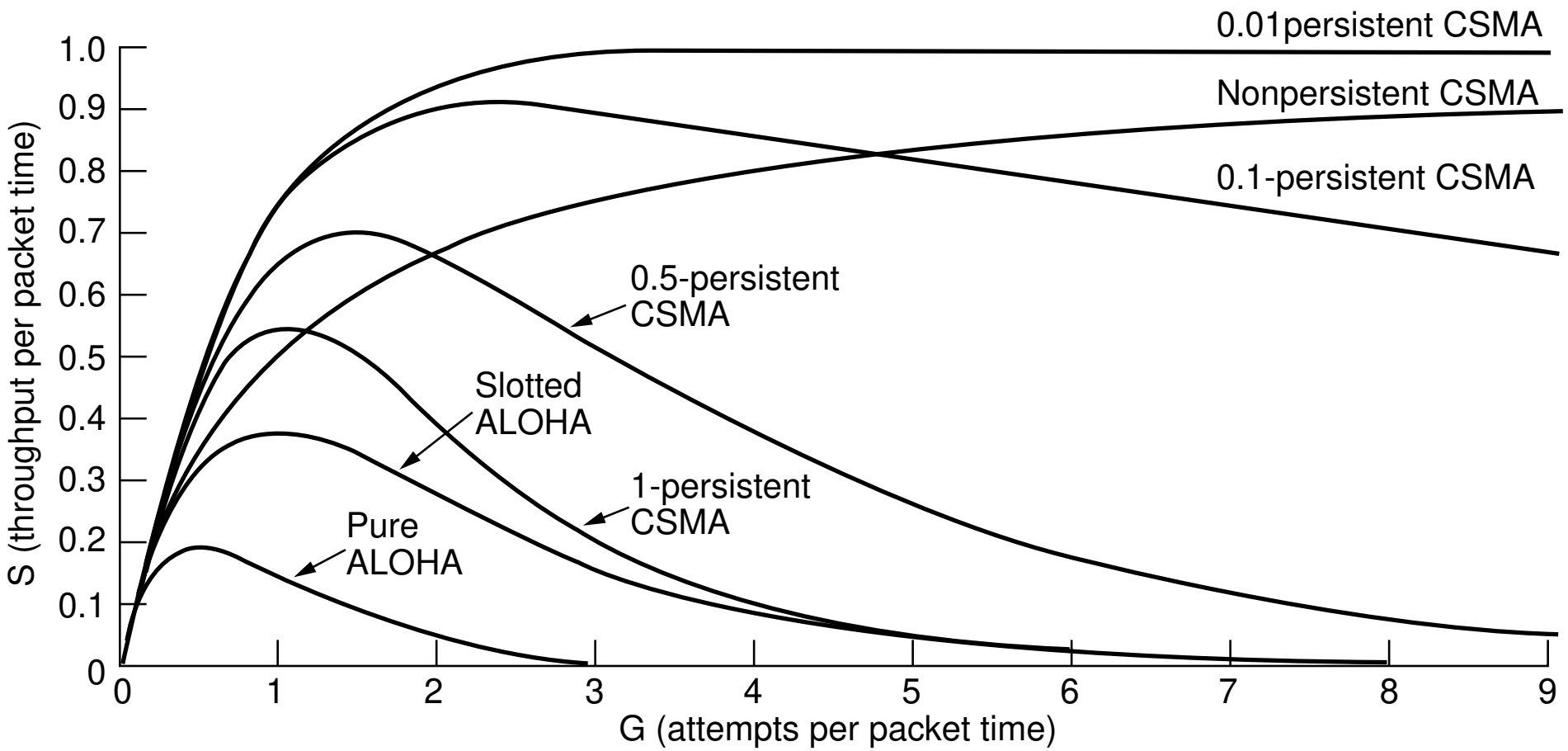
1-persistent:
Falls zu Beginn der gewünschten Übertragung der Kanal noch nicht frei ist, wartet der Sender bis er es ist und beginnt dann sofort mit dem Senden.

Nicht-persistent:
Wenn der Kanal gerade belegt ist, dann wartet der Sender eine zufällige Zeit und versucht es dann erneut.
(**Achtung!** Bei zu langen Wartezeiten fällt die Kanalauslastung.)

p-persistent:
Falls der Kanal im aktuellen *Slot* frei ist, sendet der Sender mit Wahrscheinlichkeit $\boldsymbol{p}$. Anderenfalls wartet er auf den nächsten Slot und prüft dann erneut.

(Was bedeutet hier $\boldsymbol{p} = 1$?)

Vergleich der bisherigen Protokolle



Frage: Ist also $p = 0$ am Besten?

CSMA/CD (mit Kollisionserkennung)

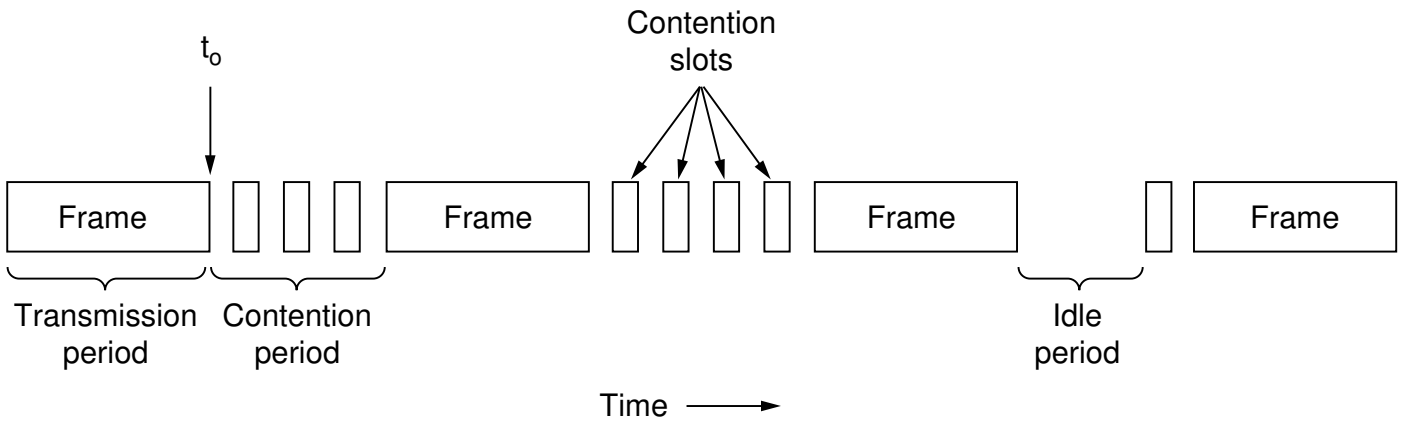
Verbessertes Protokoll

(1) Lausche am Kanal. Beginne mit der Übertragung, wenn der Kanal frei ist.

(2) Lausche auch während der Übertragung weiter am Kanal. Breche bei einer Kollision sofort das Senden ab. (Nur, wenn gleichzeitiges Senden und Empfangen möglich ist!)

(3) Warte eine zufällige Zeitdauer. $\Rightarrow$ Weiter mit Schritt (1).

Hieraus resultieren mögliche Kanalzustände:
Konkurrenz (*contention*), Übertragung (*transmission*) und Frei (*idle*):

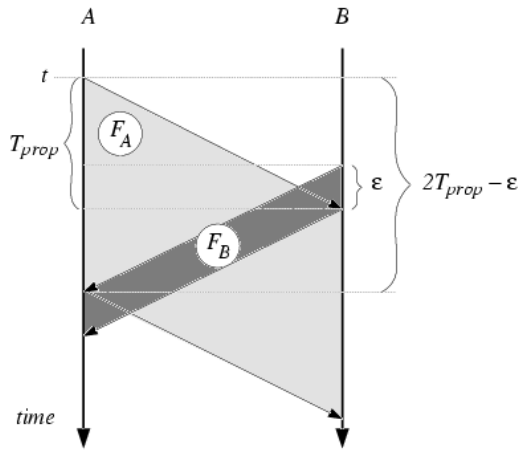


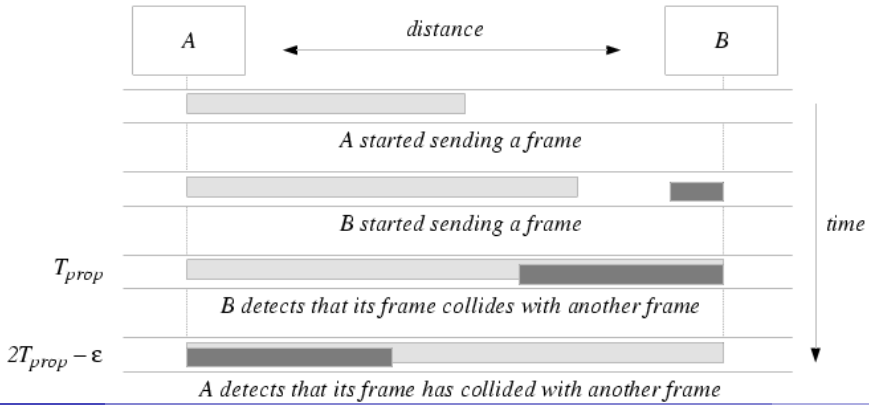
CSMA/CD (Forts.)

Länge der Konkurrenzphase

Ab wann kann während einer Übertragung keine Kollision mehr auftreten?

$\Rightarrow$ Antwort: doppelte Latenz

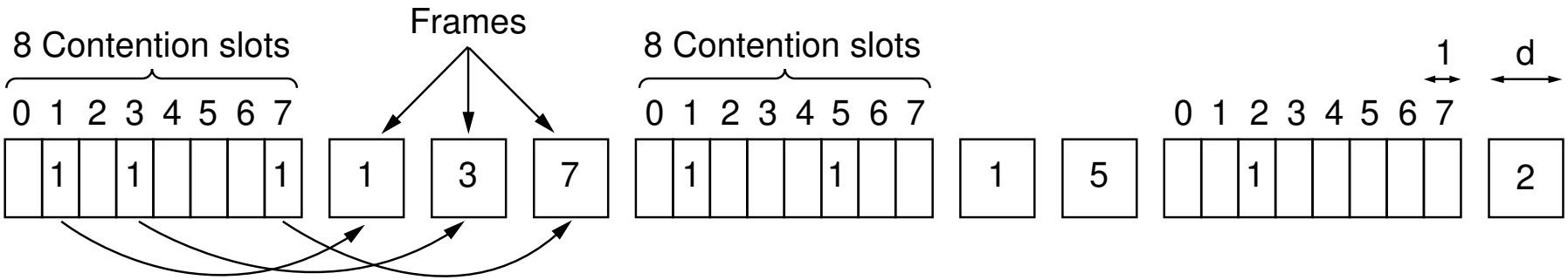




Bit-Map

Reservierendes Protokoll

- Alle Stationen haben eine eindeutige ID.
- Jede Konkurrenzphase wird in ***N*** Slots unterteilt.
- Wenn Station ***k*** ein Frame übertragen will, dann sendet sie eine "1" im ***k***-ten Slot.
- Von allen Stationen, die einen Sendewunsch geäußert haben, beginnt die Station mit der niedrigsten ID.

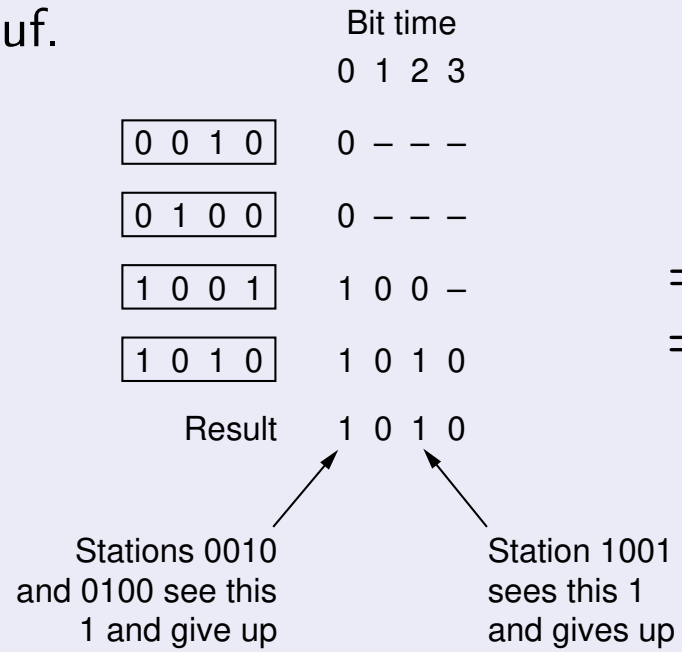


Nachteil: Pro Station muss man für die Konkurrenzphase 1 Bit spendieren.

Binärer Countdown

Protokoll

- Alle Stationen haben eine eindeutige ID $\Rightarrow$ Priorität.
- Jede Station, die senden will, überträgt ihre eigene Priorität, Bit für Bit, und lauscht dabei gleichzeitig am Kanal.
- Der Kanal führt eine koordinierte logische ODER-Verknüpfung aller gesendeten Prioritäten durch. Erkennt eine Station eine höhere Priorität als die eigene, dann gibt sie sofort auf.



$\Rightarrow$ **Vorteile?**
$\Rightarrow$ **Nachteile?**

- Beispiel: CAN-Bus im Kfz (11-Bit-IDs für Nachrichten, 0-Bit dominant)

Binärer Countdown (Forts.)

Faire Variante (Mok und Ward, 1979)

Die Prioritäten werden regelmäßig nach erfolgreichem Senden einer Station neu vergeben. Die Station, die gerade gesendet hat, bekommt die Priorität 0, die anderen rücken ggf. auf.

Beispiel:

Station	A	B	C	D	E	F	G	H
Priorität	0	1	2	3	4	5	6	7

Nun sendet Station ***E***. Welche Prioritäten haben die Stationen danach?

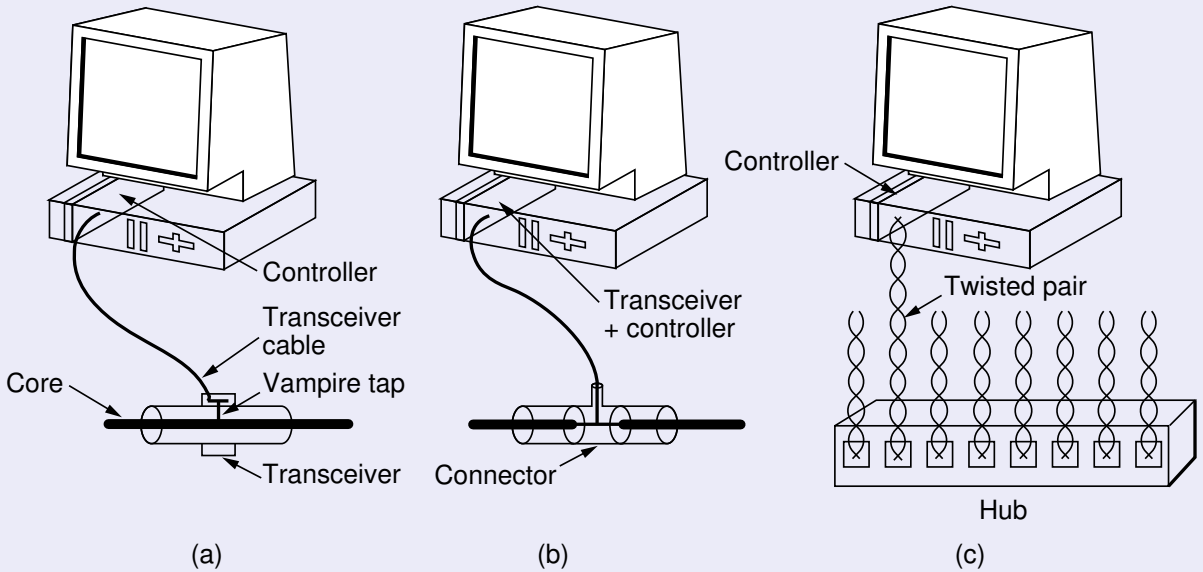
Protokolle mit eingeschränkter Konkurrenz

Beobachtung

- Kollisionsfreie Protokolle sind gut bei hohem Nachrichtenaufkommen. Aber: Vor **jeder** Übertragung müssen Stationen den Kanal reservieren!
- Optimistische Protokolle sind immer dann gut, wenn nicht zuviel gesendet wird. Dann kann eine Station oft sogar sofort senden.

IEEE 802.3: Ethernet (10 Mbps)

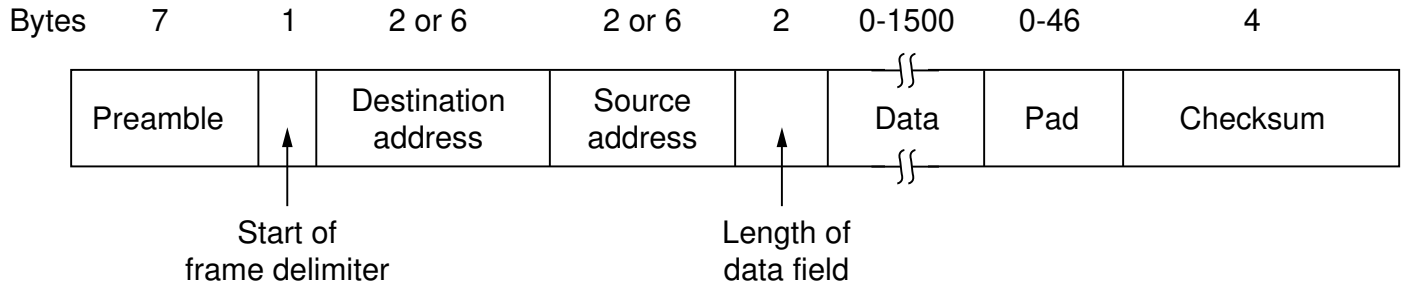
Kabeltypen



(a) 10Base5 **(b)** 10Base2 **(c)** 10Base-T

Name	Kabel	max. Distanz	Knoten/Segment
10Base5	dickes Koax	500 m	100
10Base2	dünnes Koax	200 m	30
10Base-T	Twisted Pair	100 m	1024
10Base-F	Glasfaser	2000 m	1024

Ethernet – Frame Layout (Forts.)



Header-Felder

Preamble: Zur Synchronisation der Uhren siebenmal 01010101
Frame Start-Byte: 01010111 um anzuzeigen, dass es nun losgeht.
Destination & Source address: (MAC-Adresse) i.d.R. 6 Bytes lang
Adresstypen:

- Normale Adresse, falls das höchstwertigste Bit '0'.
- Multicast-Adresse, falls das höchstwertigste Bit '1'.
- Broadcast-Adresse, falls alle Bits '1'.
- Unterscheidung in lokale und globale Adressen über das zweithöchste Bit. Ca. 7 Billionen eindeutige globale Adressen; Vergabe durch die IEEE.

Length: Länge des Datenfeldes: 0 Byte – 1.500 Byte
Pad: ggf. Auffüllen auf minimale Frame-Länge von 64 Byte
Checksum: CRC-Prüfsumme über das Datenfeld

Ethernet (Forts.)

Kanalzugriff

- CSMA /CD mit binären exponenziellem Backoff

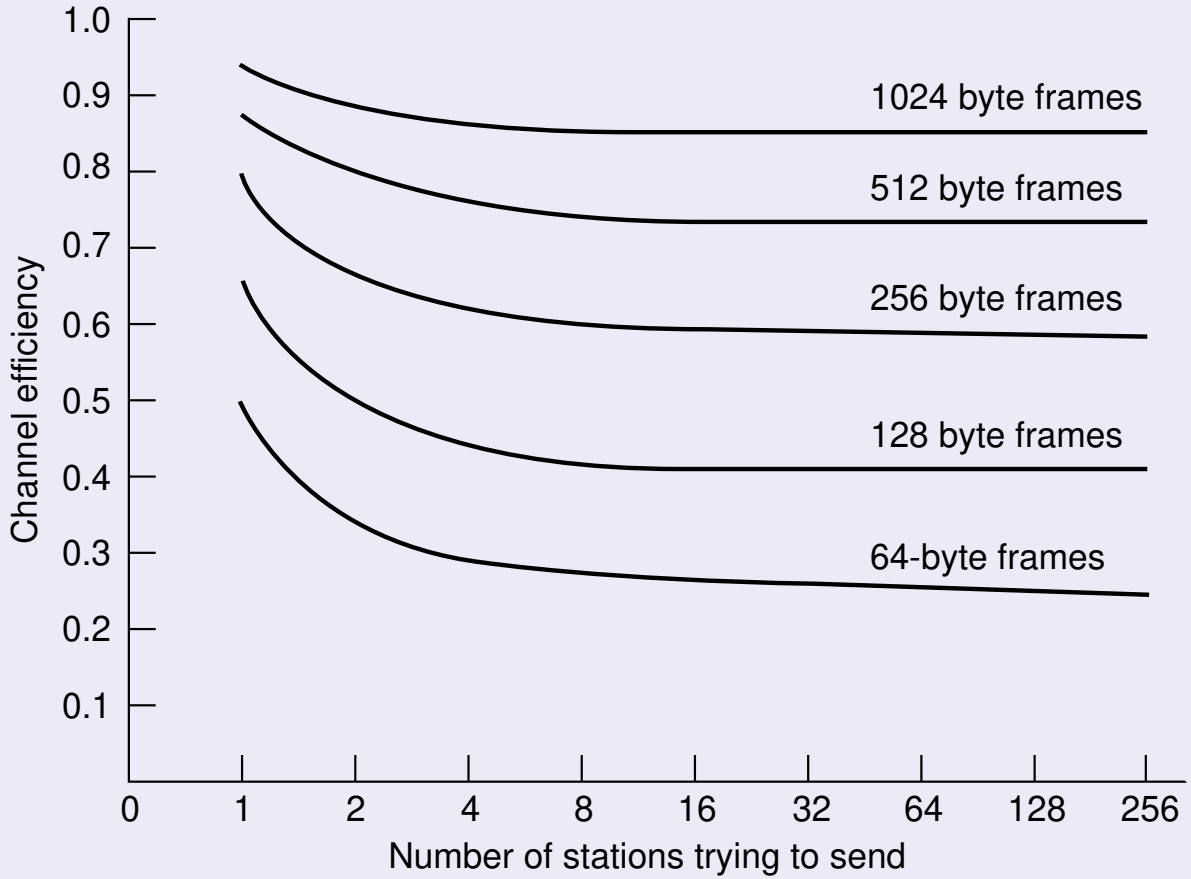
Kollisionsbehandlung: binärer exponenzieller Backoff

- Zufallsbedingte Wartezeit nach 1. Kollision: 0 oder 1 Slots
  (1 Slot dauert 512 Bits = 51,2 μs; dies entspricht 2τ.)
- nach 2. Kollision: 0, 1, 2 oder 3 Slots
- nach i. Kollision: $x \in \{0, ..., 2^i - 1\}$ Slots
- ab 10. Kollision: $x \in \{0, ...1023\}$ Slots
- nach 16. Kollision gibt der Sender auf:
  $\Rightarrow$ Fehlermeldung!
- Vorteile:
  - bei Kollision nur kurze Verzögerungen
  - dennoch rasche Kollisionsauflösung

Ethernet (Forts.)

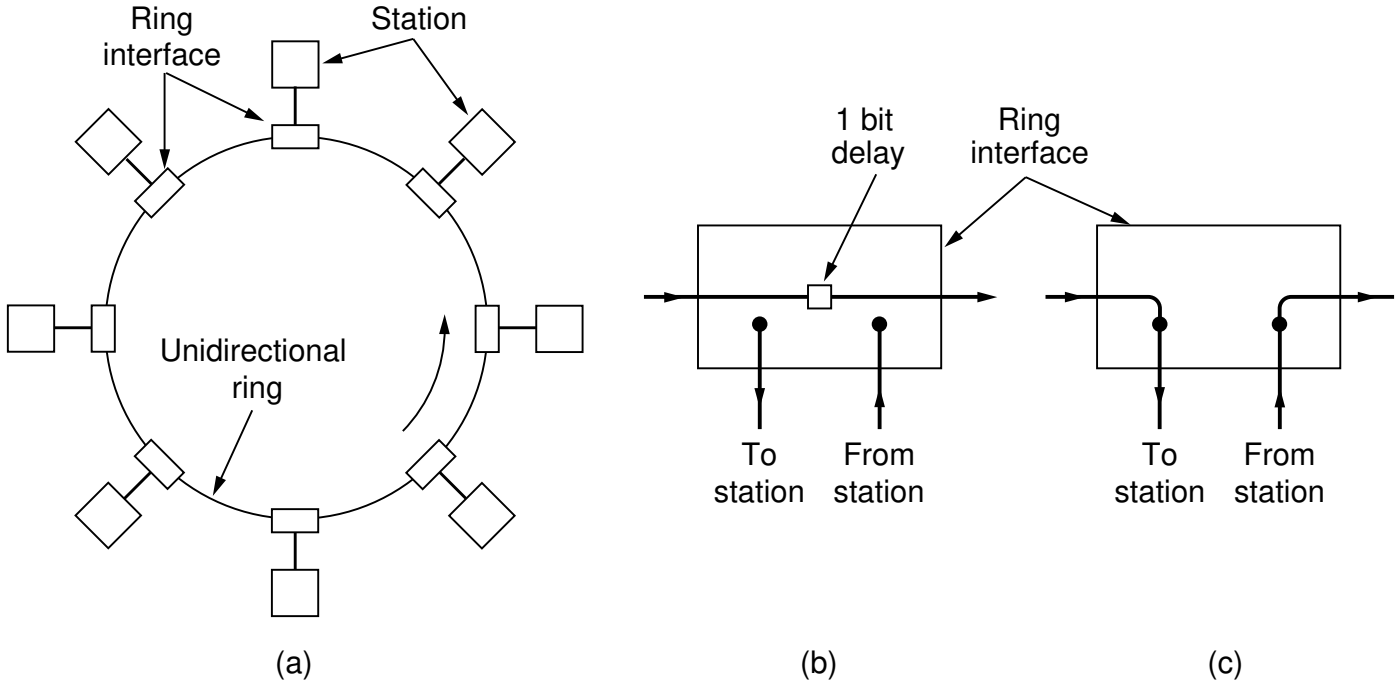
Kanalauslastung vs. Frame-Länge

(Formeln: s. Tanenbaum)



Frage: Frage: Warum ist das so?

IEEE 802.5: Token Ring



Protokoll

- Station mit dem Token sendet eine Nachricht statt dem Token.
- Die sendende Station nimmt ihre Nachricht selbst vom Ring. **($\Rightarrow$ Wieso?)**
- Bild (b): Listen-Modus; pro Station eine Verzögerung von 1 Bit
- Bild (c): Send-Modus
- Eine *Ring-Manager*-Station überwacht das Token.

IEEE 802.3u: Fast-Ethernet

Eigenschaften

- Datenrate: 100 Mbps
- Protokolle und Datenformate bleiben unverändert (und damit kompatibel).
- Bit-Zeit muss von 100 ns auf 10 ns gesenkt werden
  ($\Rightarrow$ Was bedeutet das?)

IEEE 802.3u: Fast-Ethernet

Eigenschaften

- Datenrate: 100 Mbps
- Protokolle und Datenformate bleiben unverändert (und damit kompatibel).
- Bit-Zeit muss von 100 ns auf 10 ns gesenkt werden
  ($\Rightarrow$ Was bedeutet das?)

$\Rightarrow$ Bessere Kabel notwendig:

	Kabeltyp	**Länge**	**Vorteile**
100Base-T4	Twisted Pair (4 Kabelpaare)	$\leq$ 100 m	vorhandene Kabel (UTP Kateg. 3)
100Base-TX	Twisted Pair (2 Kabelpaare)	$\leq$ 100 m	Vollduplex, preisgünstig
100Base-FX	Glasfaser	$\leq$ 2 km	Vollduplex, Distanz

Fast-Ethernet (Forts.)

Besonderheiten

- Alle Stationen werden sternförmig mittels eines **Hub** verkabelt.
  $\Rightarrow$ Verfügbare Bandbreite wird unter allen sendenden Stationen geteilt.
- Durch Einsatz eines **Switch** wird die für jede Station zur Verfügung stehende Bandbreite deutlich erhöht.
- An Dual-Speed-Hubs und Dual-Speed-Switches lassen sich gleichzeitig 10Base-T- und 100Base-T-Stationen betreiben.
  $\Rightarrow$ Unkomplizierte Aufrüstung!

Gigabit-Ethernet

Varianten

- 802.3ab: 1000Base-T (Kupferkabel, UTP Cat 5e oder Cat 6).
  Max. Segmentlänge: 100 m.
- 802.3z: Glasfaser, max. Faserlänge: bis 5km (1000Base-ZX bis 70 km).

Problem

Bei einer Datenrate von 1 Gbps ist Kollisionserkennung schwierig!
($\Rightarrow$ vgl. Abhängigkeiten von Framegröße und Kabellänge)

Lösung:
I. Allg. keine Kollisionen durch Verzicht auf Hubs und Einsatz von Switches.

Es geht noch viel schneller

10-Gigabit-Ethernet (10GbE), 40GbE, 100GbE , 200GBE und 400GbE.
800GbE und TbE in Vorbereitung ...

Gigabit-Ethernet

Varianten

- 802.3ab: 1000Base-T (Kupferkabel, UTP Cat 5e oder Cat 6).
  Max. Segmentlänge: 100 m.
- 802.3z: Glasfaser, max. Faserlänge: bis 5km (1000Base-ZX bis 70 km).

Problem

Bei einer Datenrate von 1 Gbps ist Kollisionserkennung schwierig!
($\Rightarrow$ vgl. Abhängigkeiten von Framegröße und Kabellänge)

Lösung:
I. Allg. keine Kollisionen durch Verzicht auf Hubs und Einsatz von Switches.

Es geht noch viel schneller

10-Gigabit-Ethernet (10GbE), 40GbE, 100GbE , 200GBE und 400GbE.
800GbE und TbE in Vorbereitung ...

Kapitelvorschau

Die Vermittlungsschicht (*Network Layer*)

- übernimmt das Routing, realisiert Ende-zu-Ende-Kommunikation, überträgt *Pakete*; im Internet durch **IP** realisiert

(Bild: X-Win Glasfasernetz des DFN)

Zu lösende Probleme:

- Sender und Empfänger brauchen eindeutige Bezeichner
- Irgendwer muss die Pakete weiterleiten

- Welche Route ist die beste?
- Welche Route ist *im Moment* die beste?

Einordnung in unser Schichtenmodell

Layer 1: Bitübertragungsschicht (*physical layer*)

Betrifft die Übertragung von Bits in Form von Signalen. Sie beschäftigt sich mit mechanischen bzw. elektrischen Fragen.

Layer 2: Sicherungsschicht (*data link layer*)

Regelt u. a. den Zugriff auf ein gemeinsam benutztes Übertragungsmedium, teilt die Bits in *Data Frames* auf und stellt eine übertragungsfehlerfreie Kommunikation zur Verfügung.

Layer 3: Vermittlungsschicht (*network layer*)

Ist u. a. verantwortlich für das Routing.

Layer 4: Transportschicht (*transport layer*)

Bietet verbindungslose und -orientierte Dienste mit unterschiedlichen Zuverlässigkeitsgarantien an. Sie ist die Schnittstelle für Anwendungen.

Die Verarbeitungsschicht (*application layer*)

implementiert, was dem Benutzer zur Verfügung steht ($\Rightarrow$ E-Mail, Telnet, WWW, . . .).

Routing

Flooding

Jedes angekommene Paket wird über **alle** ausgehende Leitungen weitergesendet – jedoch nicht über die, auf der es angekommen ist.

Hauptproblem

Das Netz *ertrinkt* in Paketen. Lösungen:

- Nach jedem Weiterleiten eines Paketes wird dessen *Hop-Counter* erhöht; jedes Paket darf max. n-mal weitergeleitet werden, bevor es entfernt wird.
  $\Rightarrow$ Wahl von n?
- Jedes Paket wird von jedem Router maximal einmal weitergleitet.
  $\Rightarrow$ Pakete haben eindeutige Sequenznummern und Router merken sich die Nummern der Pakete, die sie schon mal weitergeleitet haben.

Einsatzbereich

Überall, wo besondere Robustheit gefordert ist (z. B. Ad-Hoc-Netze) oder wenn noch keine Informationen über das Netz vorliegen.

Routing

Flooding

Jedes angekommene Paket wird über **alle** ausgehende Leitungen weitergesendet – jedoch nicht über die, auf der es angekommen ist.

Hauptproblem

Das Netz *ertrinkt* in Paketen. Lösungen:

- Nach jedem Weiterleiten eines Paketes wird dessen *Hop-Counter* erhöht; jedes Paket darf max. n-mal weitergeleitet werden, bevor es entfernt wird.
  $\Rightarrow$ Wahl von n?
- Jedes Paket wird von jedem Router maximal einmal weitergleitet.
  $\Rightarrow$ Pakete haben eindeutige Sequenznummern und Router merken sich die Nummern der Pakete, die sie schon mal weitergeleitet haben.

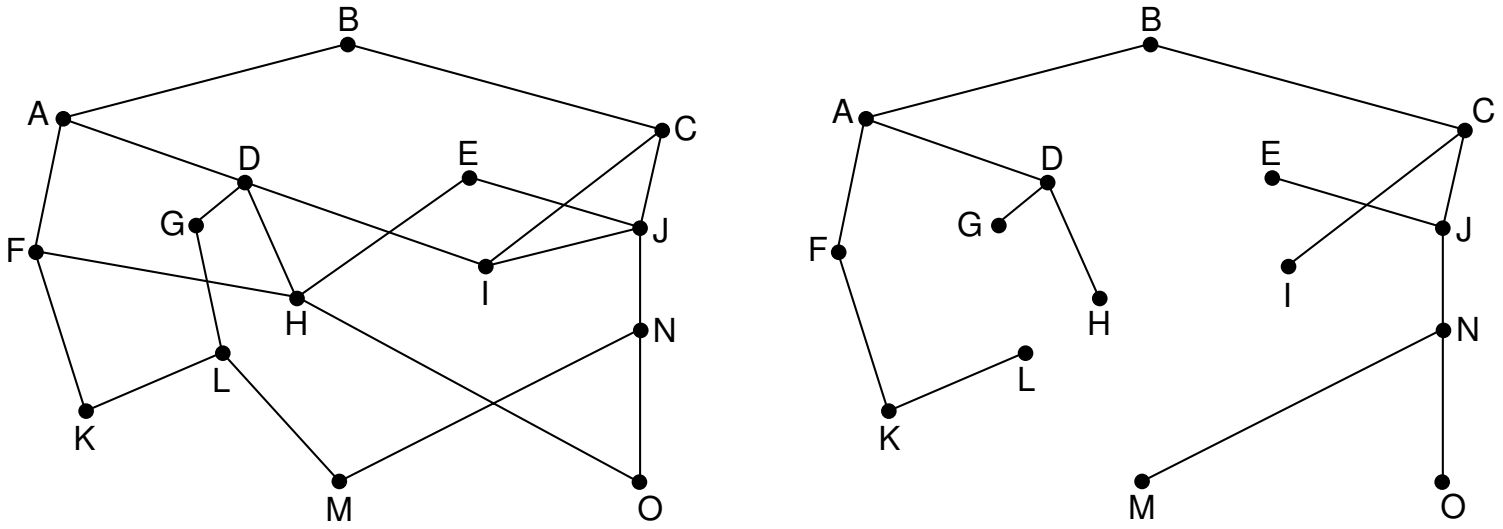
Einsatzbereich

Überall, wo besondere Robustheit gefordert ist (z. B. Ad-Hoc-Netze) oder wenn noch keine Informationen über das Netz vorliegen.

Routing (Forts.)

Beobachtung

Alle optimale Routen ausgehend von einer Station zu den restlichen Stationen spannen einen Baum auf ($\Rightarrow$ *minimaler Spannbaum*).



Idee

Die teilnehmenden Router müssen gemeinsam für jede Station diesen Spannbaum berechnen, bzw. approximieren ($\Rightarrow$ verteilter Algorithmus).

Shortest-Path-Routing

Dijkstra-Algorithmus

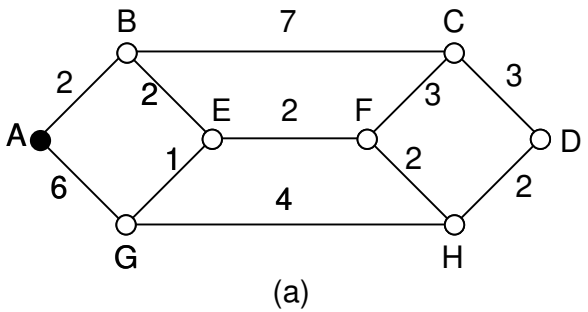
Ausgehend vom Startknoten als Wurzel wird der Spannbaum sukzessive konstruiert:

- Es werden für alle direkt erreichbaren Knoten die bisher bekannten geringsten Kosten ermittelt.
- Der (an den bisherigen Spannbaum angrenzende) Knoten, den man am kostengünstigsten erreichen kann, und die Kante, die mit minimalen Kosten zu ihm führt, werden in den Spannbaum aufgenommen.
- Dieser Knoten wird zum neuen Ausgangsknoten.

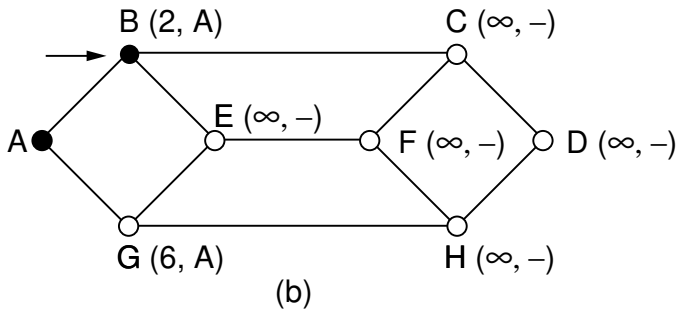
Wenn alle Knoten in den Baum aufgenommen wurden, ist der minimale Spannbaum fertig berechnet.

Shortest-Path-Routing (Forts.)

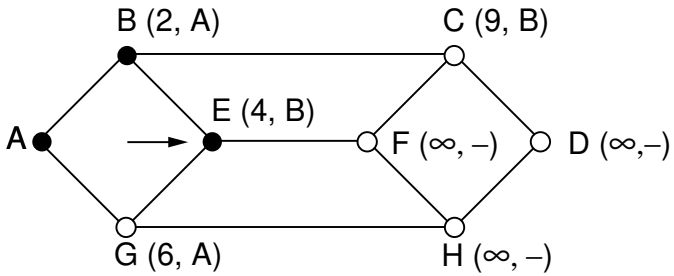
Beispiel:



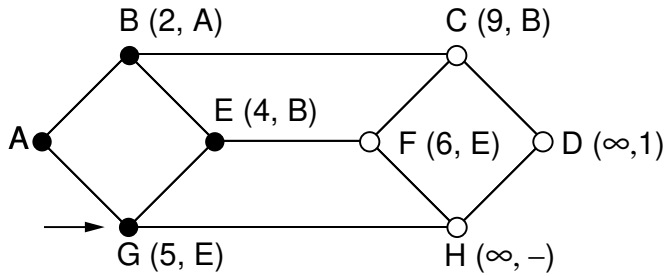
(a)



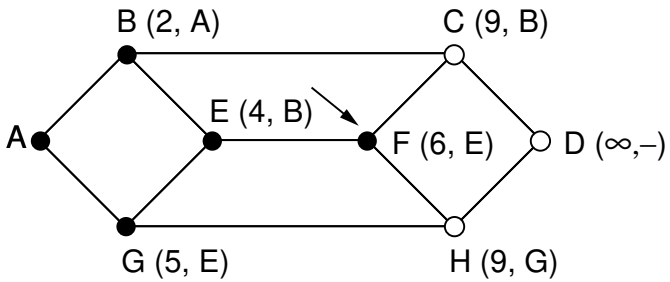
(b)



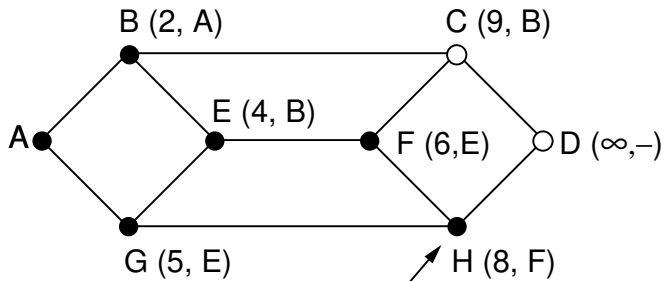
(c)



(d)



(e)



(f)

Und schließlich: C(9,B) und D(10,H).

Frage: Wie sieht die Routingtabelle aus?

Distance-Vector Routing (DVR)

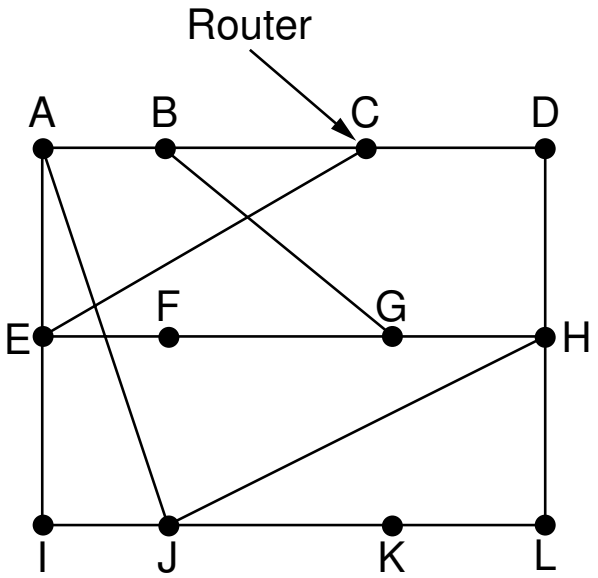
DVR wurde bis 1979 im ARPANET eingesetzt.

Protokoll

- Jeder Knoten veröffentlicht regelmäßig seine (geschätzten) Kosten zum Erreichen aller anderen Knoten (an seine Nachbarn).
- Jeder Knoten baut aus den erhaltenen Informationen, immer wieder einen neuen Distanzvektor zusammen, aus dem er die Routingtabelle ableitet.
- **Regel:**
  Sende das weiterzuleitende Paket an den Nachbarknoten, der verspricht, die Nachricht am kostengünstigsten zuzustellen (inkl. der Kosten um den Nachbarn zu erreichen.)

Distance-Vector Routing (Forts.)

Beispiel:



(a)

New estimated delay from J

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

JA delay is 8 | JI delay is 10 | JH delay is 12 | JK delay is 6

Vectors received from J's four neighbors

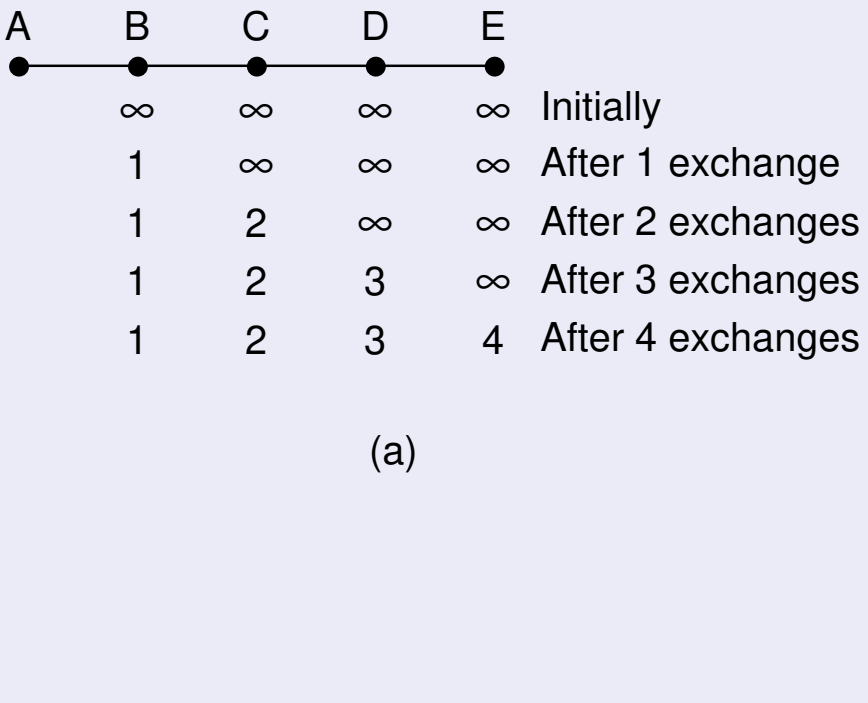
New routing table for J

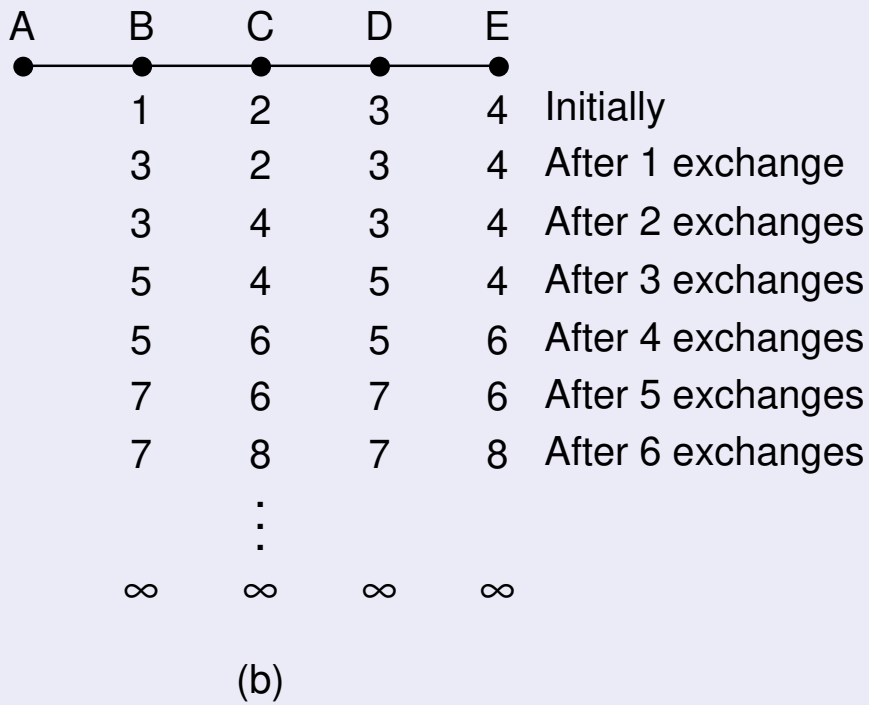
(b)

Distance-Vector Routing (Forts.)

Nachteile

- schlechte Konvergenz zum minimalen Spannbaum
- Änderungen verbreiten sich nur langsam
- das Count-to-Infinity-Problem:





(a) ***A*** wird neu eingegliedert

(b) ***A*** fällt aus

Link-State Routing Protocol (LSP)

LSP löste DVR als Routing-Algorithmus weitgehend ab.

Schnellere Konvergenz

- Jeder Router sendet **seine** Informationen das Netz betreffend an **alle**.
- Jeder Router berechnet damit seinen Spannbaum.
- Wie erreiche ich alle Router ohne ein bestehendes Routing-Verfahren?
  $\Rightarrow$ Flooding!

Vorgehensweise

(1) Finde alle Deine Nachbarn und deren Adressen heraus.
(2) Bestimme die Kosten zu Deinen Nachbarn.
(3) Broadcaste ein *Link-State Paket* mit diesen Infos.
(4) Empfange alle *Link-State Pakete*.
(5) Berechne den minimalen Spannbaum.

Link-State Routing (Forts.)

Problem 1

Wer sind meine Nachbarn?

Lösung

Sende auf allen vorhandenen Netzwerk-Interfaces ein “Hello”-Paket und warte auf die Antworten.

Link-State Routing (Forts.)

Problem 1

Wer sind meine Nachbarn?

Lösung

Sende auf allen vorhandenen Netzwerk-Interfaces ein “Hello”-Paket und warte auf die Antworten.

Problem 2

Wie bestimme ich die Kosten zu meinen Nachbarn?

Lösung

Sende jeweils einige “Hello”-Pakete und bestimme die mittlere RTT durch Messung.

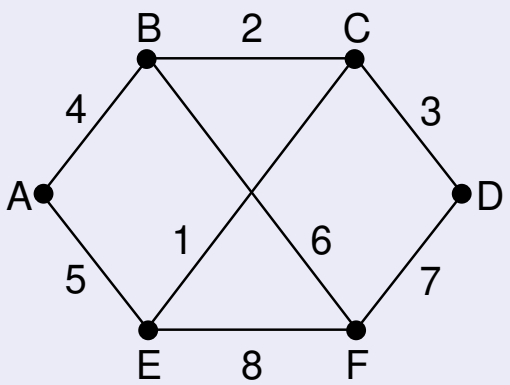
Link-State Routing (Forts.)

Problem 3

Welche Informationen sollen die *Link-State Pakete* enthalten?

Lösung

Jedes Paket enthält eine Sequenznummer, eine Lebensdauer und eine Liste mit den eigenen Messungen.



(a)

Link State Packets

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
E	8

(b)

Link-State-Routing (Forts.)

Problem 4

Wie werden die *Link-State Pakete* verteilt?

Lösung

Mittels Flooding ungefähr einmal pro Stunde:

- Jeder Router hält eine Liste (Router, SEQ-NO), sodass das Flooding gestoppt werden kann.
- Die Lebensdauer eines Paketes wird pro Sekunde und bei jedem Weitersenden um eins erniedrigt. Bei `Age = 0` $\Rightarrow$ Paket verwerfen!

Hierarchisches Routing

Problem der bisherigen Verfahren

Jeder Router kennt alle Router und alle Stationen.

$\Rightarrow$ Verfahren **skalieren nicht** mit wachsender Netzwerkgröße:

- Routingtabellen werden zu groß
  $\Rightarrow$ Speicherplatz, Suchzeiten
- Netzwerkzustand ist hochgradig dynamisch
  $\Rightarrow$ häufige Änderungen der Tabellen aller Routern

Hierarchisches Routing

Problem der bisherigen Verfahren

Jeder Router kennt alle Router und alle Stationen.
⇒ Verfahren **skalieren nicht** mit wachsender Netzwerkgröße:

- Routingtabellen werden zu groß
  ⇒ Speicherplatz, Suchzeiten
- Netzwerkzustand ist hochgradig dynamisch
  ⇒ häufige Änderungen der Tabellen aller Routern

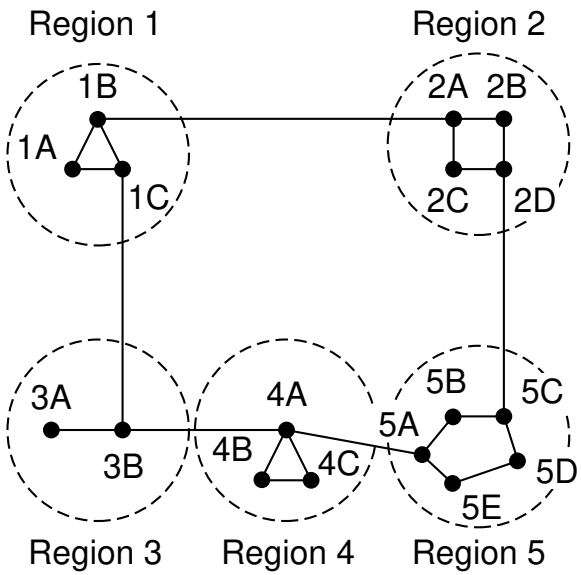
Lösungsidee

Das Gesamtnetz wird in mehrere Regionen untergliedert, die jeweils lokal für sich verwaltet werden.
Übergeordnet wird ein globales Verfahren verwendet.
Vorteile:

- Routingtabellen werden kleiner
- Änderungen im Netz bleiben lokal begrenzt.
- Regionen können individuelle Routing-Verfahren einsetzen

Hierarchisches Routing (Forts.)

Beispiel:



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

(a) Netz mit 5 Regionen

(b) flaches Routing

(c) zweistufiges Routing

Broadcast-Routing

Aufgabenstellung

Manchmal soll ein Paket an alle angeschlossenen Stationen gesendet werden (i.d.R. in LANs).

Lösungsideen

(1) Kopien des Pakets einzeln versenden $\Rightarrow$ aufwändig!

(2) Flooding $\Rightarrow$ i. Allg. Bandbreitenverschwendung!

(3) *Multi-Destination-Routing:*
An das Paket wird eine Empfängerliste angehängt, die jeder Router jeweils auswertet und dabei gegebenfalls aufsplittet.

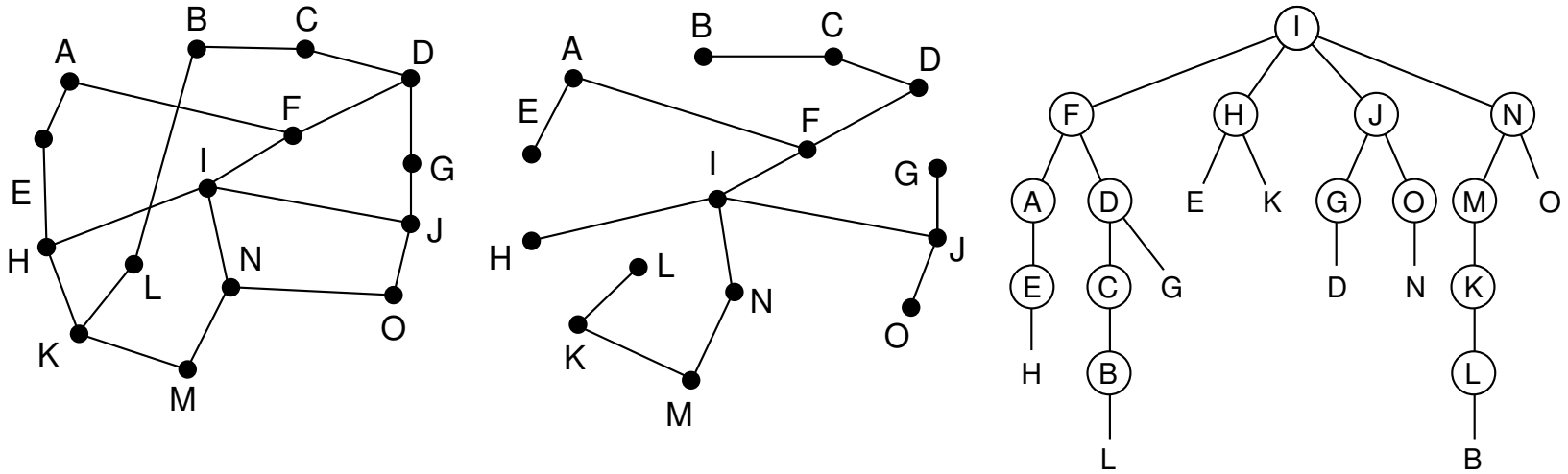
(4) Wir versenden über den Spannbaum.
Z.B.: Sender berechnet Spannbaum und sendet ihn mit dem Paket; Router werten diesen aus.

Broadcast-Routing (Forts.)

Reverse-Path-Forwarding

- Manchmal (z. B. DVR) kennen die Router keinen Spannbaum.
- Falls ein Broadcast-Paket vom Sender ***S*** auf der Leitung ankommt, über die es laut Routing-Tabelle auch an ***S*** gesendet würde, ist es vermutlich kein Duplikat.
  $\Rightarrow$ Weitersenden auf allen anderen Leitungen.

Beispiel:



(i) ***I*** erhält über ***F*** ein Broadcast-Paket von ***E*** $\Rightarrow$ weitersenden

(ii) ***I*** erhält über ***H*** ein Broadcast-Paket von ***E*** $\Rightarrow$ verwerfen

Multicast-Routing

Aufgabenstellung

Einige Pakete sollen an einen Teil aller Stationen gesendet werden (z. B. Radio, Videokonferenz).

Lösungsidee

(1) Die Stationen, welche die Pakete empfangen möchten, melden sich für eine bestimmte *Multicast*-Gruppe an.

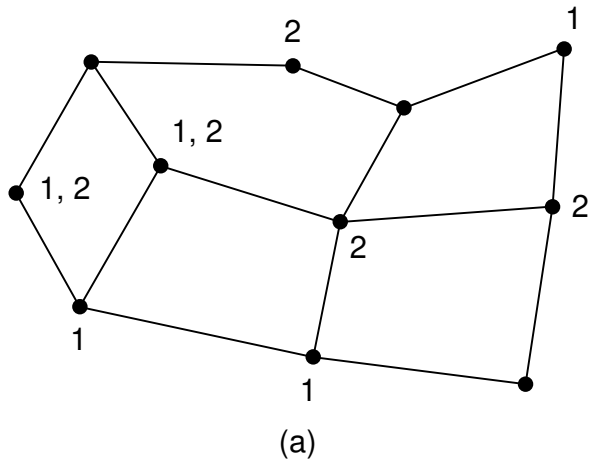
(2) Für das komplette Netz wird der Spannbaum berechnet.

(3) Der Spannbaum wird für die Multicast-Gruppe durch Senden entsprechender Nachrichten Stück für Stück "zurückgeschnitten" bis nur noch die Stationen enthalten sind, die Multicasts empfangen wollen oder Pakete weiterleiten müssen.

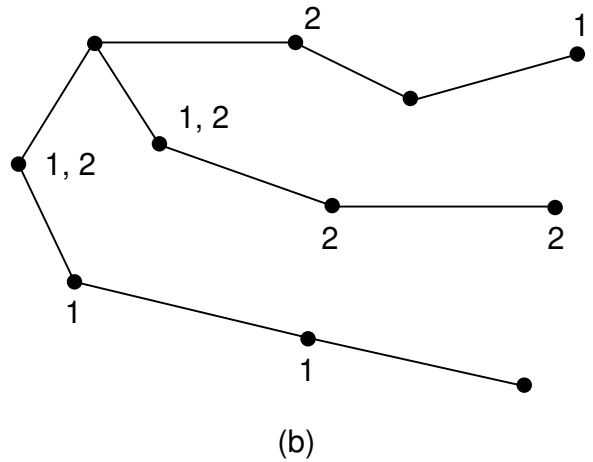
Multicast-Routing (Forts.)

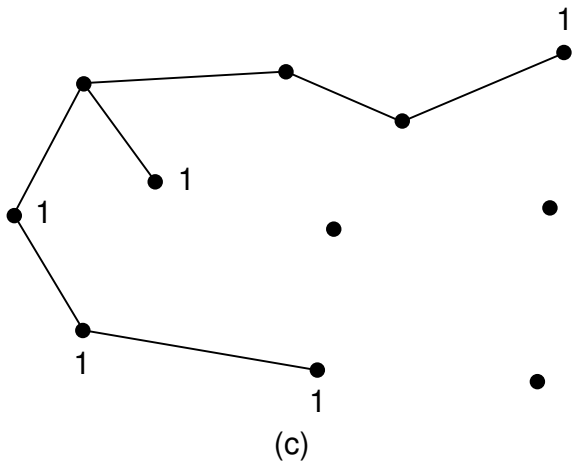
Beispiel:

(a) Netzwerk

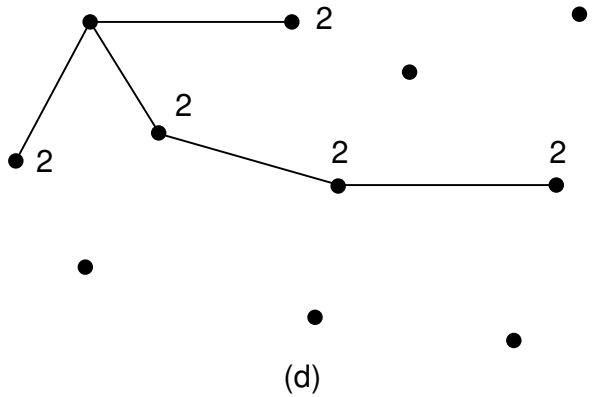


(b) vollständiger Spannbaum





(c) Multicast-Gruppe 1

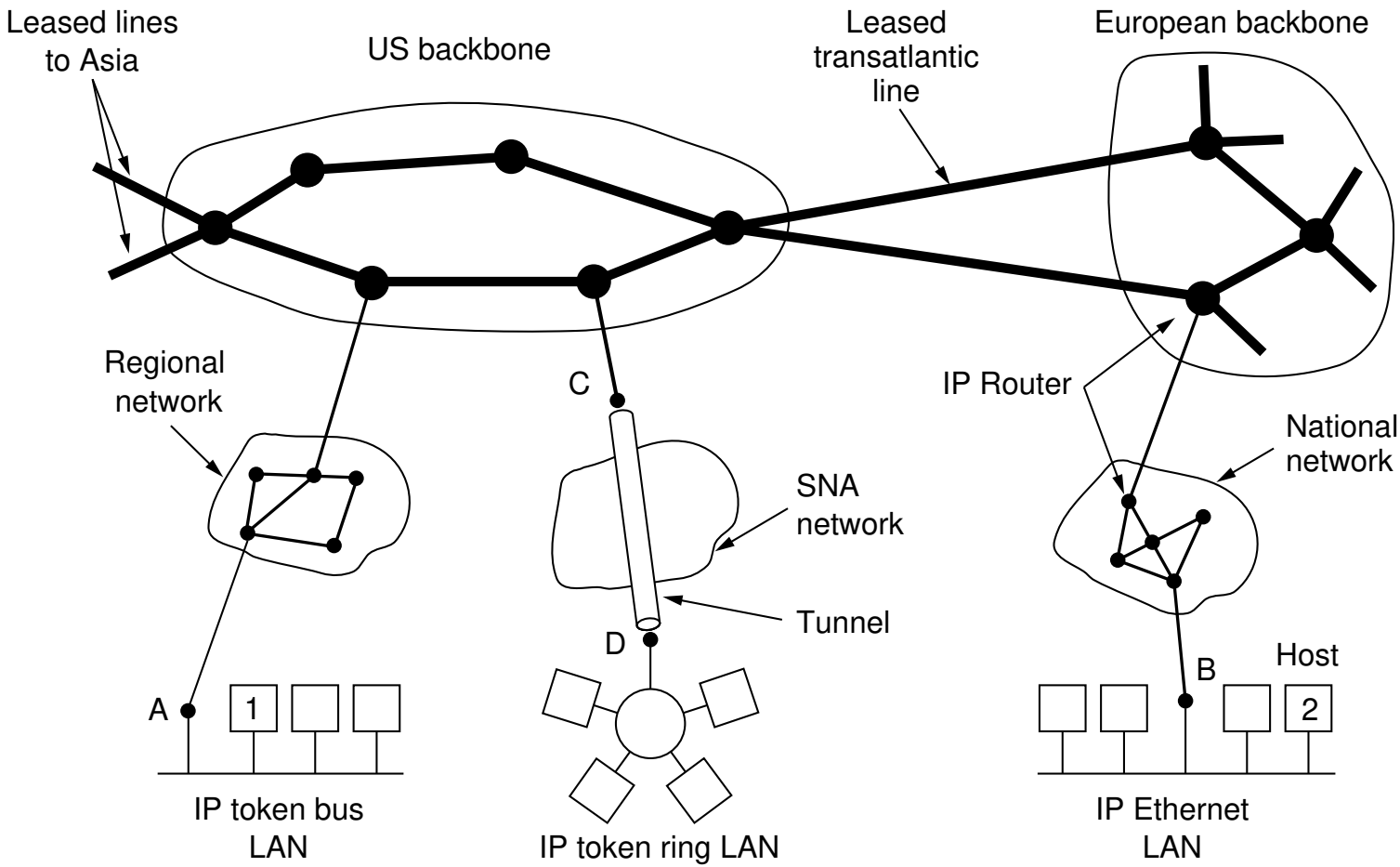


(d) Multicast-Gruppe 2

Das Internet Protocol (IP)

Das Internet

Das Internet ist ein Verbund autonomer Netze, die mittels **Backbone**-Netzen miteinander verschaltet sind. In allen Netzen wird IP "gesprochen".



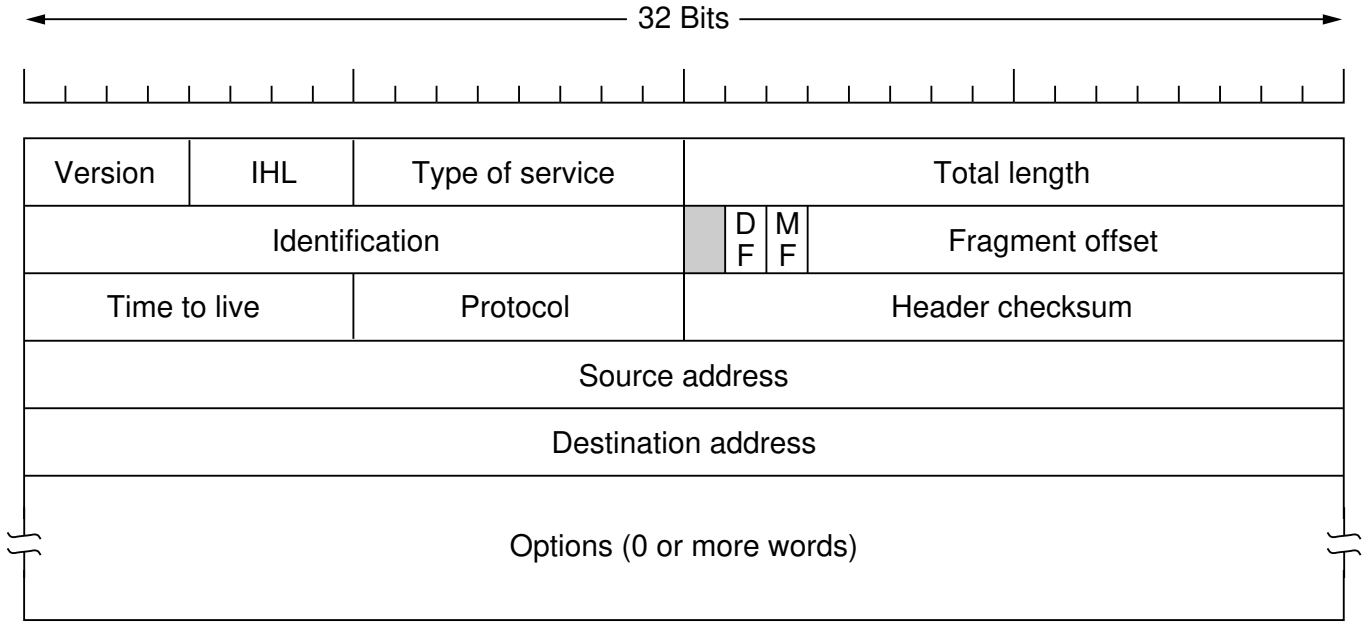
Der Internet-Protokoll-Stack

Application Layer
Transport Layer
Network Layer
Host-to-Network Layer

Ablauf

- Anwendungen verwenden die Dienste der Transportschicht, wobei sie zwischen verbindungslosen und verbindungsorientierten Diensten wählen können.
- Die Transportschicht zerlegt die Anwendungsdaten in *Datagramme*, die an die Vermittlungsschicht runtergereicht werden.
- Die Datagramme werden als Pakete durch das Internet "gerouted" und an die Host-to-Network-Schicht runtergereicht (und dort ggf. fragmentiert).

Der IP-Paket Header (IP v4)



IHL	Header-Länge (wg. Options-Feld)
TOS	"Type of Service", i.d.R. ignoriert
TLen	Paket-Länge inkl, Nutzdaten (max. 64 kB)
ID	Datagramm-ID
DF	gesetzt, falls Datagramm nicht fragmentierbar
MF	gesetzt, falls weitere Fragmente folgen
FOff	Position (Offset) des Fragments im Datagramm
TTL	max. Anzahl Hops oder Sekunden (max. 255 Sek.)
Prot	Globale IDs der Transportschicht
HCS	Header Checksum (Neuberechnen bei jedem Hop)

Der IP-Paket Header (IP v4) (Forts.)

Optionen

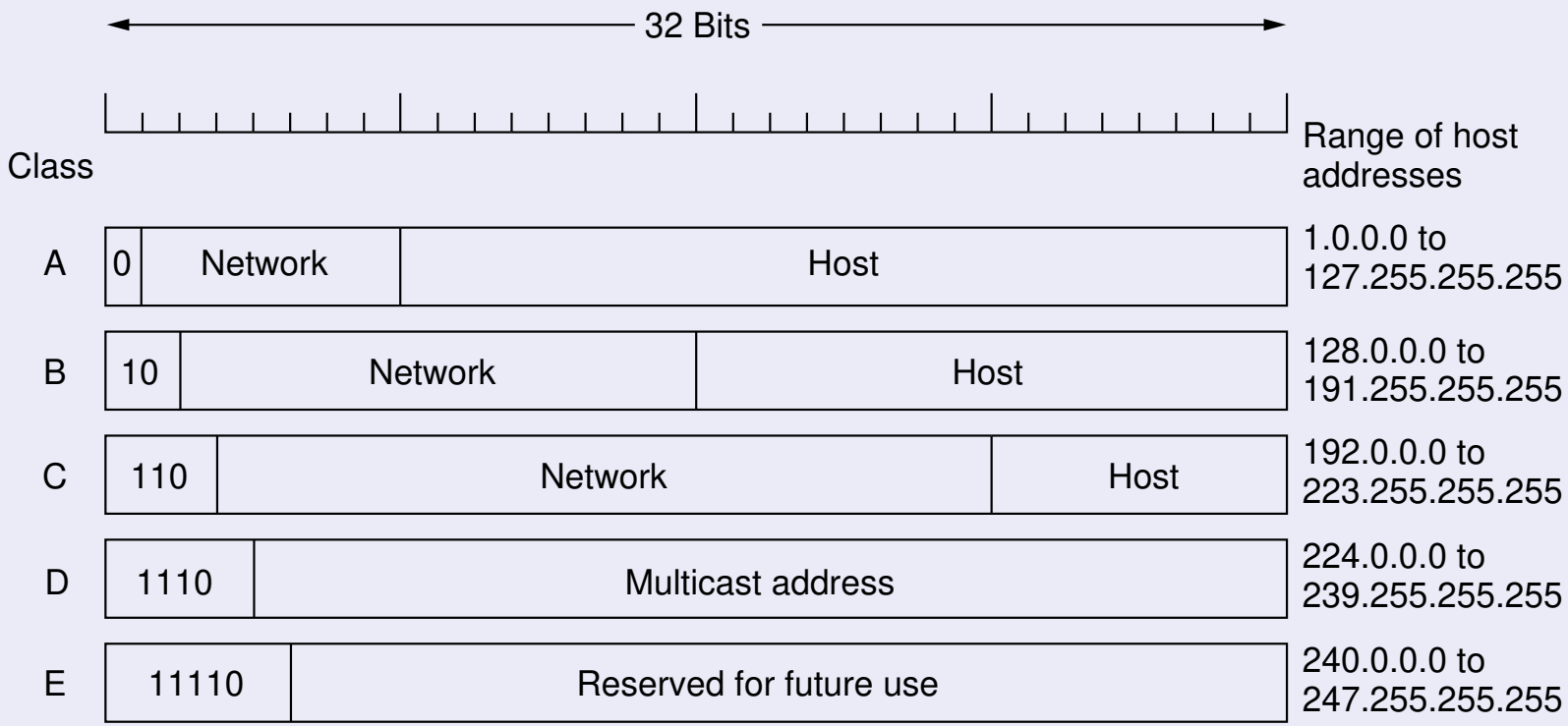
- *Security*
  Schutzwürdigkeit des Datagramms (wird nicht verwendet!)
- *Strict Source Routing*
  kompletter Routing-Pfad (Debugging, Sicherheit, mobile Rechner)
- *Loose Source Routing*
  Liste von Routern, die besucht werden sollen
- *Record Route*
  Jeder Router hängt seine IP-Adresse an (Debugging)
- *Timestamp*
  Jeder Router hängt seine IP-Adresse plus Zeitstempel an (Debugging)

Jedoch: Viele Optionen werden oft nicht unterstützt!

IP-Adressen

IP-Adressklassen

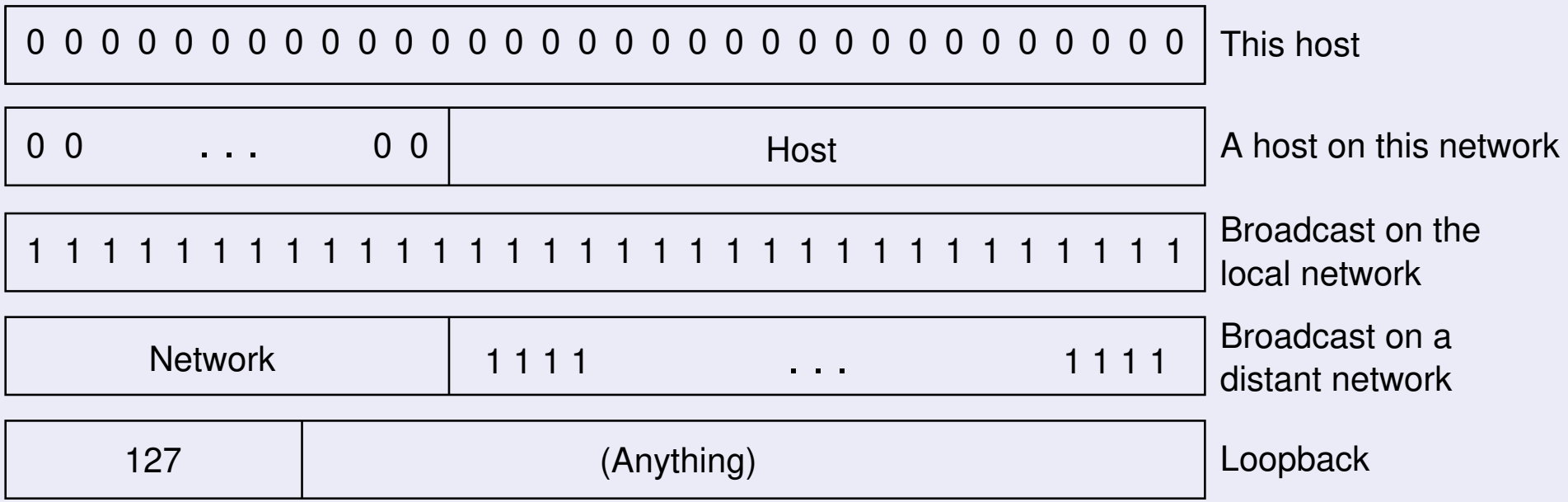
Durch die IP-Adressklassen A – C lassen sich verschieden große Netzwerke definieren.



Frage: Frage: Wieviele Klasse-A (-B, -C)-Adressen gibt es?

IP-Adressen (Forts.)

Besondere IP-Adressen (vgl. RFC 3330)

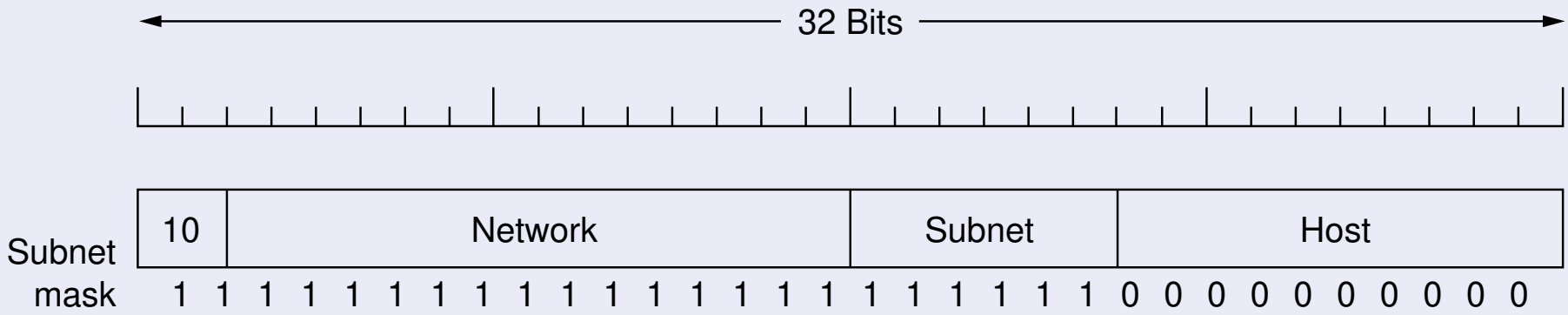


Außerdem gilt:
Die IP-Adressen `192.168.x.y` sowie `172.16.x.y` – `172.31.x.y` und `10.x.y.z` können nicht im Internet verwendet werden (vgl. RFC 1918). Sie dienen dem Aufbau privater Netze, die mittels *Masquerading* bzw. *Network Address (Port) Translation* an das Internet angebunden werden können.

IP-Adressen (Forts.)

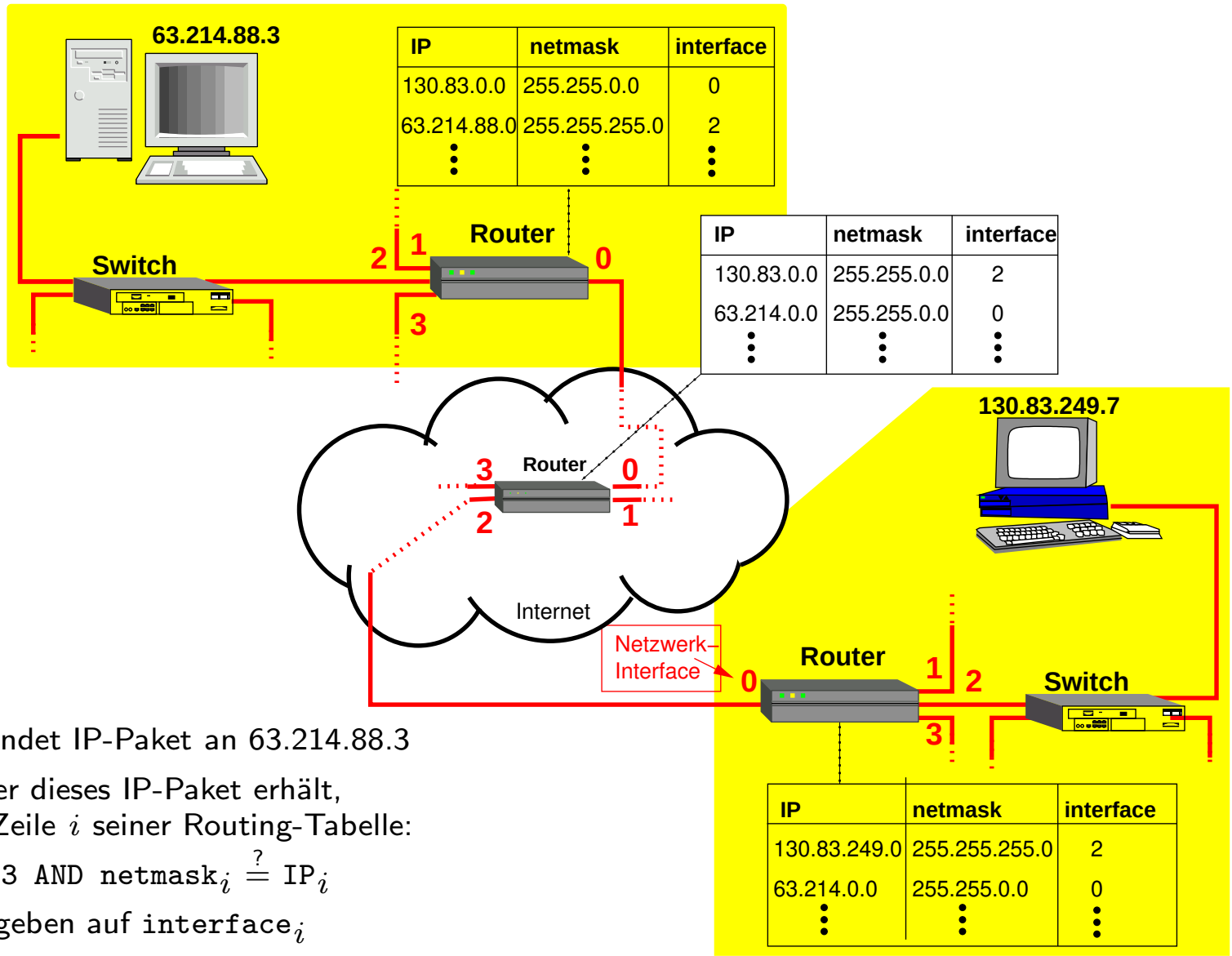
Subnet-Masking

Damit eine (größere) Organisation nicht für jedes neue LAN eine neue Netzadresse beantragen und bekanntgeben muss, bekommen i.d.R. alle Netze dieser Organisation dieselbe Netzwerkadresse. Die einzelnen LANs werden über die Teilnetzadresse unterschieden, die Rechner in einem Teilnetz über die Rechneradresse:



⇒ Es resultiert eine drei- oder mehrstufige Routing-Hierarchie!

CIDR: Classless Inter Domain Routing (schematisch)



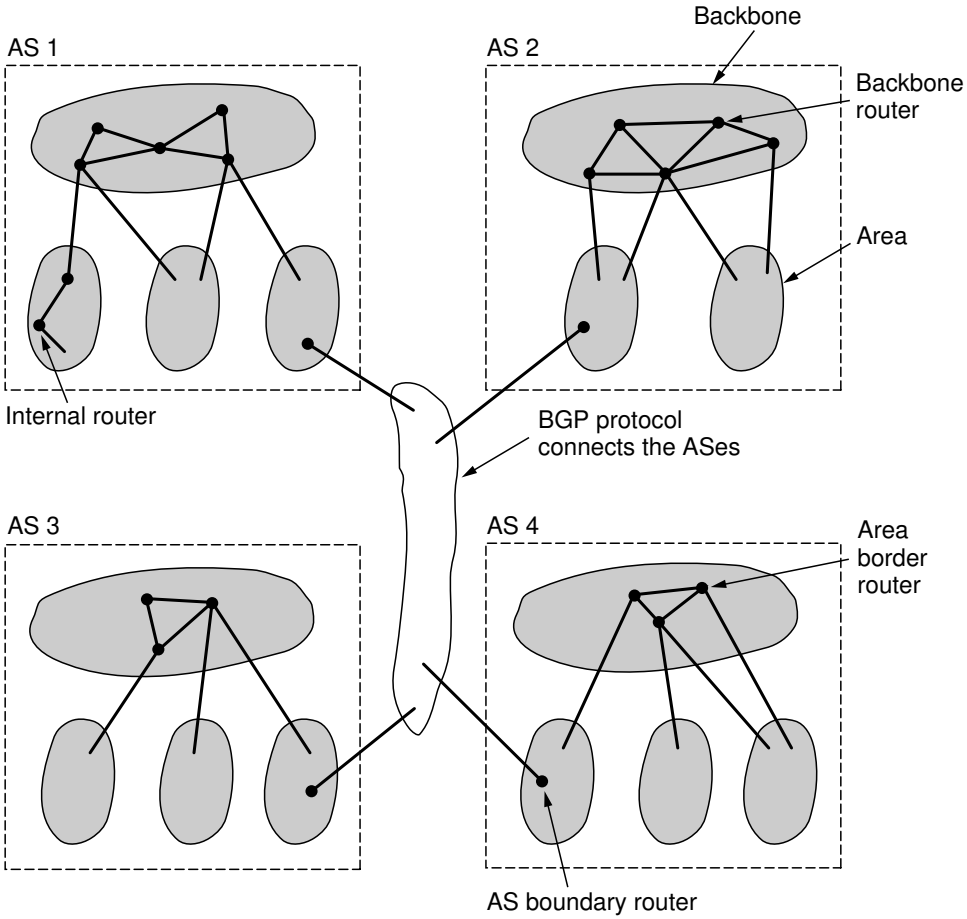
- 130.83.249.7 sendet IP-Paket an 63.214.88.3
- jeder Router, der dieses IP-Paket erhält, testet für jede Zeile i seiner Routing-Tabelle:

  $$\texttt{63.214.88.3 AND netmask}_i \stackrel{?}{=} \texttt{IP}_i$$

- bei Treffer: ausgeben auf $\texttt{interface}_i$
- man schreibt auch: 63.214.88.3/24 für netmask 255.255.255.0

IP-Routing

Hierarchisch, wobei die autonomen Teilnetze (AS_i) jeweils individuelle Routing-Verfahren einsetzen.



Interior Gateway Routing:
Möglichst optimales Routen von Paketen vom Sender zum Empfänger.
Meist OSPF bzw. RIP.

Exterior Gateway Routing:
Teilnetz-übergreifend, berücksichtigt politische Entscheidungen; z. B. dürfen manche Teilnetze nie in der Route enthalten sein.
Meist BGP.

OSPF: Interior-Gateway-Routing

Open Shortest Path First

OSPF ist ein Link-State-Protokoll (für Router) und hat das vormals weit verbreitete RIP (*Routing Information Protocol*, Distance-Vector-Routing) in weiten Teilen ersetzt.

Eigenschaften

- Open Standard (frei implementierbar)
- diverse Kostenmaße (# Hops, Zeitverzögerung, Geld, ...)
- anpassungsfähig an wechselnde Topologien, auch Lastverteilung
- unterschiedliches Routing für unterschiedliche Daten (Multimedia, ...)
- Unterstützung von hierarchischem Routing
- Sicherheit gegen die Verbreitung falscher Routing-Informationen
- Unterstützung von IP-Tunneln

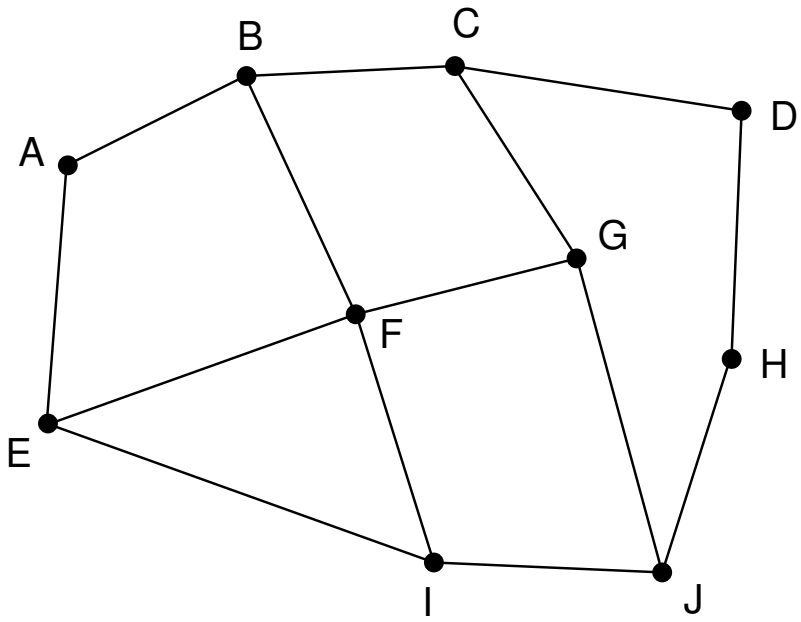
BGP: Exterior-Gateway-Routing

BGP (*Border Gateway Protocol*, RFC 1163)

- verbindet die *autonomen Systeme* (AS)
- Kommunikation über TCP-Port 179
- eigentlich ein DVR-Protokoll, d. h. jeder BGP-Router speichert Routing-Informationen zu allen anderen BGP-Routern
  $\Rightarrow$ Routingtabellen werden sehr groß (ca. 650 000 Einträge, Anfang 2017)!
- Vermeidung von *Count-To-Infinity* durch Mitführen des Routing-Pfads
- BGP-Router verwerfen Pakete mit erkennbaren Routingschleifen
- Routing-Regeln berücksichtigen die Länge des Routing-Pfads sowie strategische Aspekte
- Routing-Regeln werden i. Allg. manuell in die BGP-Router eingetragen:
  - "Route keine Pakete vom Pentagon via Iran"
  - "Route niemals Pakete von oder zu SAP via Oracle-eigene Router"
  - ...

BGP: Exterior-Gateway-Routing (Forts.)

Beispiel:



(a)

Information F receives
from its neighbors about D

From B: "I use BCD"
From G: "I use GCD"
From I: "I use IFGCD"
From E: "I use EFGCD"

(b)

BGP-Router ***F*** entscheidet sich zwischen den Routen F-BCD und F-GCD aufgrund der internen Routingregeln und einer internen Kostenfunktion ($\Rightarrow$ modifiziertes DVR).

ICMP

Internet Control Message Protocol

Kontrollnachrichten für

- Statusabfragen
  z. B. “ping” (Echo-Request zur Abfrage der Verbindungsqualität)
- Fehlermeldungen
  z. B. “destination unreachable” (type=3), “time exceeded”, ...
- weitere Detaillierung
  z. B. type=3, code=2 (Grund: nicht unterstütztes Protokoll)

ICMP-Pakete enthalten den Header des ursächlichen IP-Paketes; sie werden in normale IP-Datagramme verpackt.

Fragen

- Wie lassen sich ICMP-Pakete als solche erkennen?
- Was, wenn ein ICMP-Paket verloren geht?

ARP

IP-Pakete enthalten die IP-Adresse des Zielrechners, doch die Netzwerkkarte kennt nur die MAC-Adresse!

$\Rightarrow$ **Problem:** Woher weiß die Karte, ob sie eine Nachricht annehmen muss?

Address Resolution Protocol

(1) Station (z. B. Router) fragt im angeschlossenen LAN per Broadcast, ob einer die fragliche IP-Adresse besitzt.

(2) Jeder prüft, ob die IP-Adresse zu ihm gehört ($\Rightarrow$ Betriebssystem kennt IP-Adresse). Nur falls ja, antwortet er mit seiner MAC-Adresse.

(3) Station registriert und merkt sich die Antwort und verpackt das zu sendende IP-Paket in ein Frame mit der entsprechenden MAC-Adresse als Zieladresse.

Fragen

- Woran erkennt man eine ARP-Nachricht?
- Was, wenn kein Rechner antwortet?
- Was, wenn der falsche Rechner antwortet?

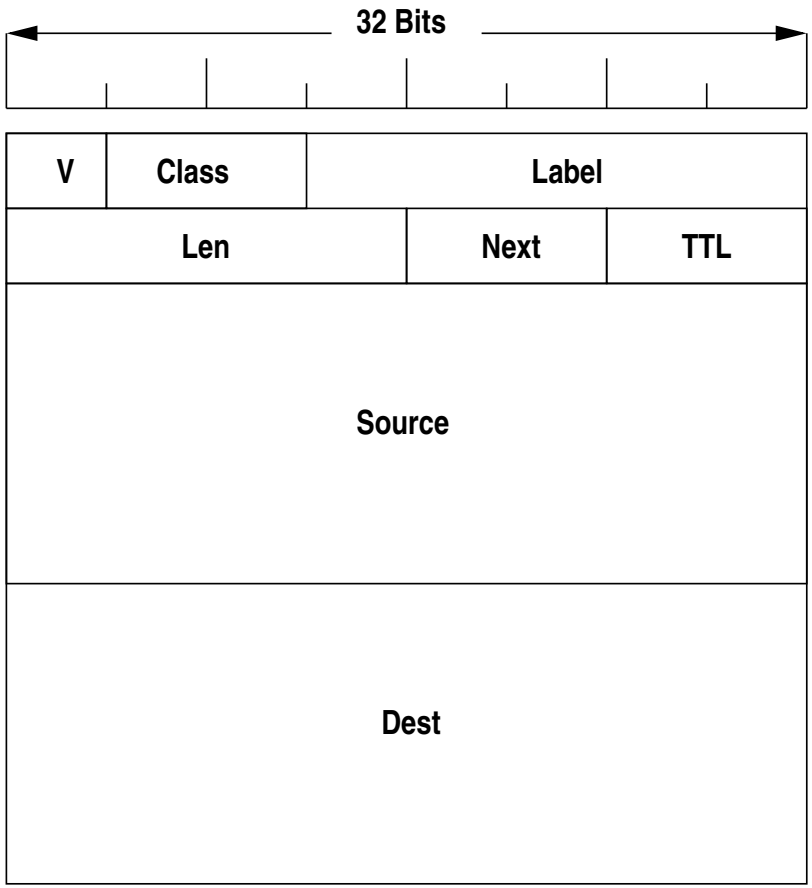
IPv6 (seit 1993)

Viele Ziele

- größerer Adressraum als bei IPv4
  ⇒ 16-Byte Adressen, z. B. `fb60:0:321:6533:fd5e:1965:2aa4:abba`
- einfacheres Protokoll, kleiner Header, ggf. Erweiterungs-Header
  ⇒ erweiterbar, schneller (z. B. keine Checksum)
- Jumbo-Pakete bis zu 4 GByte möglich (nur für spez. Anwendungen und Netze sinnvoll).
- kleinere Routing-Tabellen
- mehr Sicherheit
  ⇒ Authentisierung
  ⇒ Vertraulichkeit
  ⇒ IPSec (Punkt-zu-Punkt Sicherheit)
- QoS ⇒ Echtzeit-Anwendungen
- bessere Integration mobiler Rechner (Mobile-IPv6: feste IP-Adressen)
- **Derzeit noch besonders wichtig:** Dual-Stack-Betrieb (parallel zu IPv4)
  (Verbreitung 03/2019: ca. 27 % Zugriffe gemessen auf google.com via IPv6)

IPv6 (Forts.)

Paket Header:



V	IP-Versionsnummer
Class	Traffic Class; QoS Prioritäten
Label	Flow Label; ebenfalls QoS
Len	Nutzdatenlänge
Next	Typ des nächsten Erweiterungs-Header
TTL	Time-to-live / Hop Limit
Source	Quelladresse
Dest	Zieladresse

IPv6 (Forts.)

Adressformat nach RFC 5952

- Adressen werden in 8 Blöcke zu je 16 Bit mittels Doppelpunkten unterteilt.
- Hexadezimale Schreibweise, wobei alle in einer Adresse vorkommenden Buchstaben (`a`, `b`, `c`, `d`, `e`, `f`) immer klein geschrieben werden.
- Führende Nullen werden weggelassen. Der erste Bereich mit aufeinanderfolgenden "0000"-Blöcken "wird als "::" notiert.
- Beispiel: `2001:0db8:0000:0000:0001:0000:0000:0001`

  Die korrekte Schreibweise lautet

  `2001:db8::1:0:0:1`

IPv6 (Forts.)

Anycast

- Nachricht an eine bestimmte Gruppe von Rechnern in einem Netzwerk.
- Beispielsweise Anycast an alle Router:
  $\Rightarrow$ Nur der laut Routing-Tabelle am besten erreichbare Router antwortet.
- Anycasts ersetzen Broadcasts $\Rightarrow$ geringere Netzlast.

Multicast

- Gruppenkommunikation: Nachricht an eine Gruppe von Netzwerkknoten.
- Präfix und Multicast-Adressen sind in RFC 4291 beschrieben.

Stateless Address Autoconfiguration (SLAAC)

- Host wählt eigenständig eine *Link-Local*-Adresse (nur im LAN gültig): Aufbau der Adresse aus festem Präfix (`fe80:`) und MAC-Geräteadresse.
- Test auf Eindeutigkeit mittels Neighbor Solicitation Nachricht.
- Erzeugen einer globalen IPv6-Adresse mithilfe des Routers für Verbindungen ins Internet.

Einordnung in unser Schichtenmodell

Layer 1: Bitübertragungsschicht (*physical layer*)

Betrifft die Übertragung von Bits in Form von Signalen. Sie beschäftigt sich mit mechanischen bzw. elektrischen Fragen.

Layer 2: Sicherungsschicht (*data link layer*)

Regelt u. a. den Zugriff auf ein gemeinsam benutztes Übertragungsmedium, teilt die Bits in *Data Frames* auf und stellt eine übertragungsfehlerfreie Kommunikation zur Verfügung.

Layer 3: Vermittlungsschicht (*network layer*)

Ist u. a. verantwortlich für das Routing.

Layer 4: Transportschicht (*transport layer*)

Bietet verbindungslose und -orientierte Dienste mit unterschiedlichen Zuverlässigkeitsgarantien an. Sie ist die Schnittstelle für Anwendungen.

Die Verarbeitungsschicht (*application layer*)

implementiert, was dem Benutzer zur Verfügung steht ($\Rightarrow$ E-Mail, Telnet, WWW, . . .).

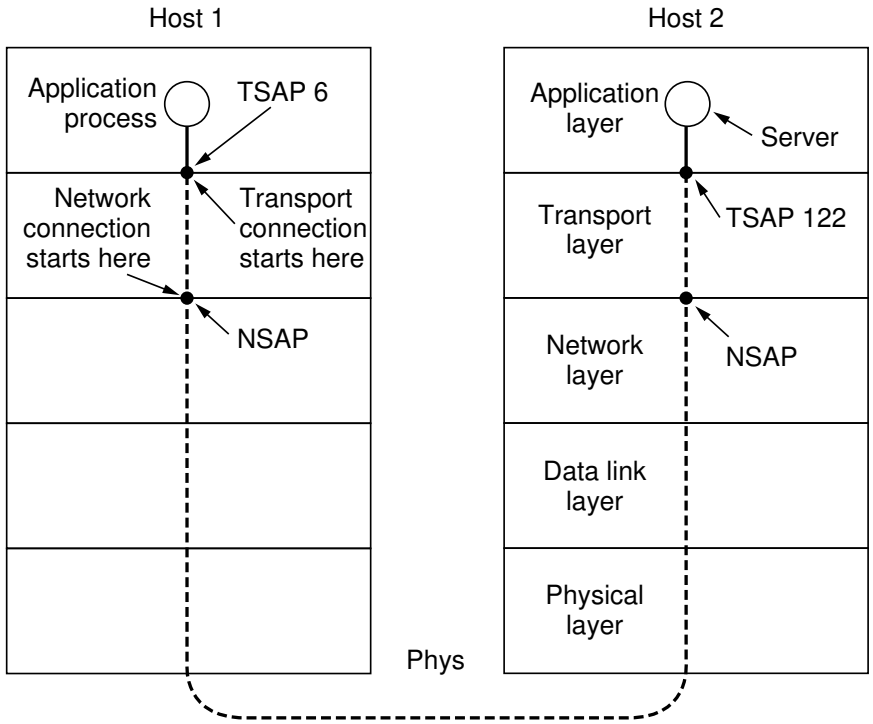
Transportschicht

Problem:
Die Vermittlungsschicht wird vom jeweiligen Netzbetreiber nach Eigeninteressen realisiert. Auf die angebotenen Dienste und Schnittstellen haben die Kunden i. d. R. wenig Einfluss.

Deshalb: Transportschicht

Zwischenschicht zwschen Vermittlungs- und Anwendungsschicht, liefert Anwendungen vom Netz unabhängige Schnittstellen ($\Rightarrow$ System-Calls).

Adressierung



TSAP

- Transport Service Access Point (allgemein)
- Kommunikation findet zwischen zwei Prozessen statt, wobei jedem Prozess ein Kommunikationsendpunkt (ein *Socket*) zugeordnet ist.
- Im Internet: Socket
  Socket-Adresse = (IP-Adresse; Port-Nummer)

Verbindungsorientierte Kommunikation

Erster Versuch zum Verbindungsaufbau

Wir senden eine *Connect*-Anforderung (connect request; CR) an die andere Seite. Wenn diese die Verbindung akzeptiert, sendet sie eine Bestätigung. Wenn keine Bestätigung ankommt, senden wir die Anforderung nochmal.

Was passiert bei Fehlern?

- 1. Fehlerfall: die 1. Anforderung ging verloren
  $\Rightarrow$ alles o.k.: nochmals senden, dann klappt's.
- 2. Fehlerfall: nur die Bestätigung ging verloren
  $\Rightarrow$ Weitere Verbindung wird aufgebaut $\Rightarrow$ unnötige Last!
- 3. Fehlerfall: die 1. Anforderung oder die Bestätigung wird sehr lange verzögert (im Netz können ja unverhersagbar lange Verzögerungen auftreten.)
  $\Rightarrow$ Wieder wird eine zweite Verbindung aufgebaut!

Verbindungsorientierte Kommunikation

Erster Versuch zum Verbindungsaufbau

Wir senden eine *Connect*-Anforderung (connect request; CR) an die andere Seite. Wenn diese die Verbindung akzeptiert, sendet sie eine Bestätigung. Wenn keine Bestätigung ankommt, senden wir die Anforderung nochmal.

Was passiert bei Fehlern?

- 1. Fehlerfall: die 1. Anforderung ging verloren
  $\Rightarrow$ alles o.k.: nochmals senden, dann klappt's.
- 2. Fehlerfall: nur die Bestätigung ging verloren
  $\Rightarrow$ Weitere Verbindung wird aufgebaut $\Rightarrow$ unnötige Last!
- 3. Fehlerfall: die 1. Anforderung oder die Bestätigung wird sehr lange verzögert (im Netz können ja unverhersagbar lange Verzögerungen auftreten.)
  $\Rightarrow$ Wieder wird eine zweite Verbindung aufgebaut!

Lösungsidee

Nachrichtenduplikate müssen erkannt werden $\Rightarrow$ Sequenznummern verwenden!

Verbindungsorientierte Kommunikation

Erster Versuch zum Verbindungsaufbau

Wir senden eine *Connect*-Anforderung (connect request; CR) an die andere Seite. Wenn diese die Verbindung akzeptiert, sendet sie eine Bestätigung. Wenn keine Bestätigung ankommt, senden wir die Anforderung nochmal.

Was passiert bei Fehlern?

- 1. Fehlerfall: die 1. Anforderung ging verloren
  $\Rightarrow$ alles o.k.: nochmals senden, dann klappt's.
- 2. Fehlerfall: nur die Bestätigung ging verloren
  $\Rightarrow$ Weitere Verbindung wird aufgebaut $\Rightarrow$ unnötige Last!
- 3. Fehlerfall: die 1. Anforderung oder die Bestätigung wird sehr lange verzögert (im Netz können ja unverhersagbar lange Verzögerungen auftreten.)
  $\Rightarrow$ Wieder wird eine zweite Verbindung aufgebaut!

Lösungsidee

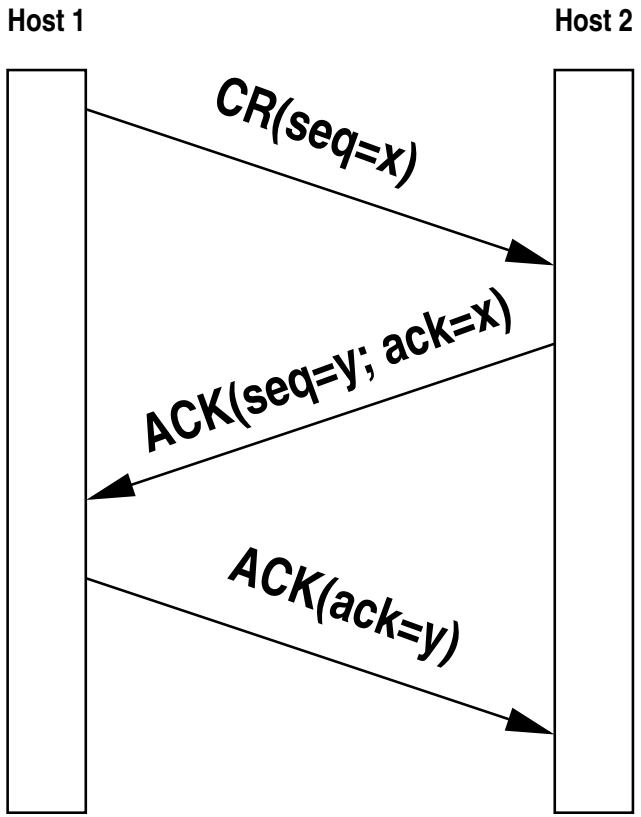
Nachrichtenduplikate müssen erkannt werden $\Rightarrow$ Sequenznummern verwenden!

Verbindungsorientierte Kommunikation (Forts.)

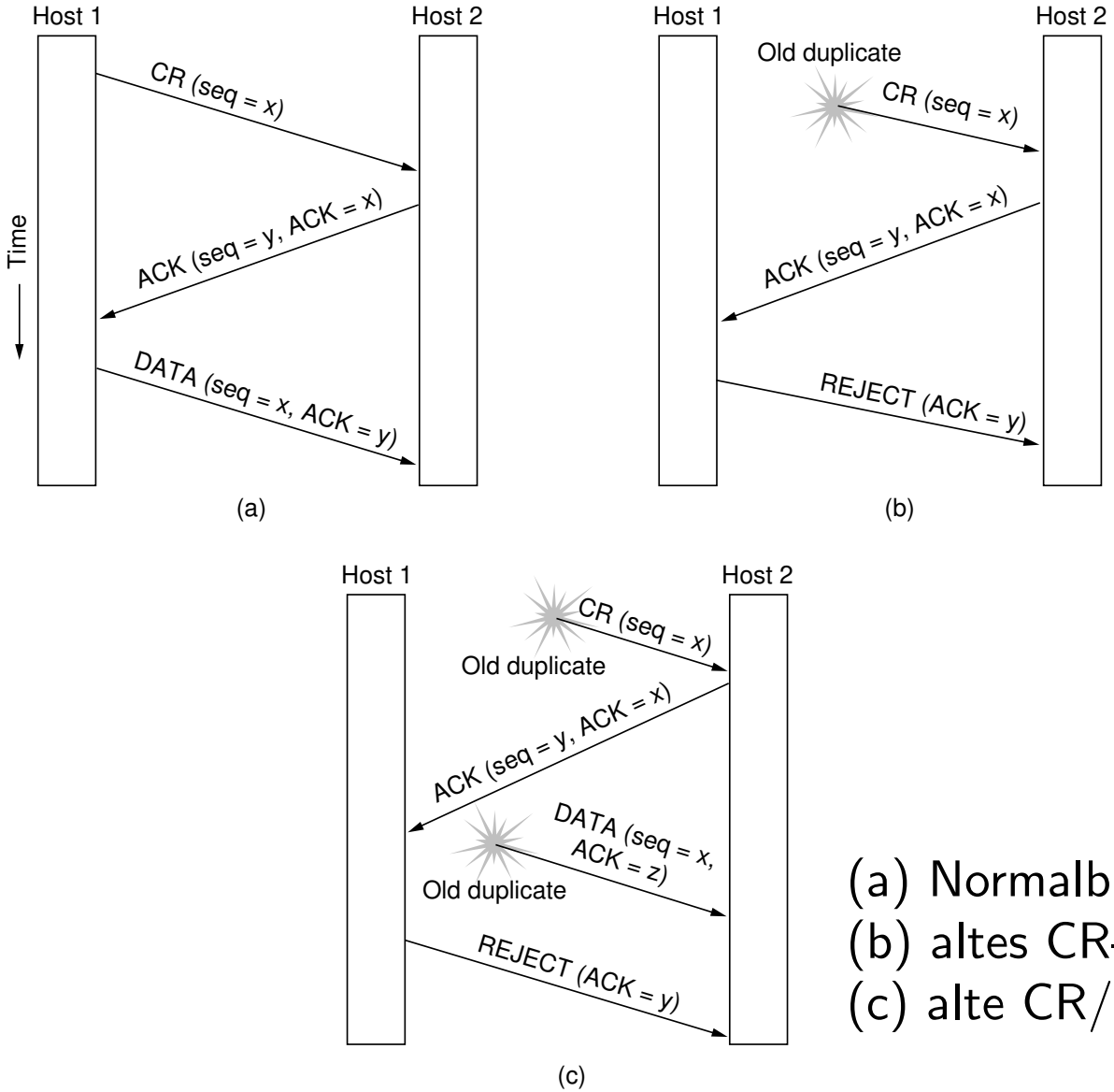
Fehlerfreier Verbindungsaufbau

Alle Nachrichten werden vom Sender bzw. Empfänger fortlaufend nummeriert. Sender und Empfänger teilen sich beim Verbindungsaufbau zunächst jeweils ihre initiale Sequenznummer (ISN) mit und sichern den Kommunikationsverlauf über *Bestätigungsnummern* ab.

Dreifacher Handshake:



Verbindungsaufbau bei Kommunikationsfehlern



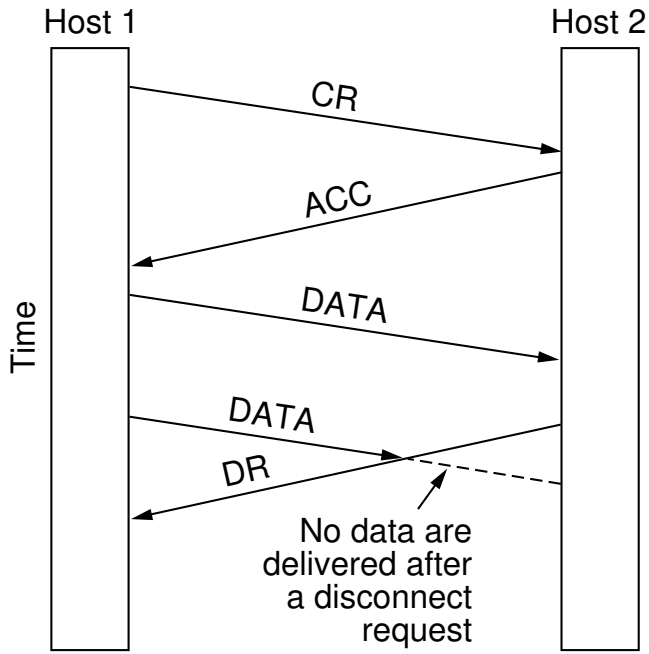
(a) Normalbetrieb
(b) altes CR-Duplikat
(c) alte CR/ACK-Duplikate

Frage: Darf man generell die 0 als ISN wählen?

Verbindungsabbau

Asymmetrisch

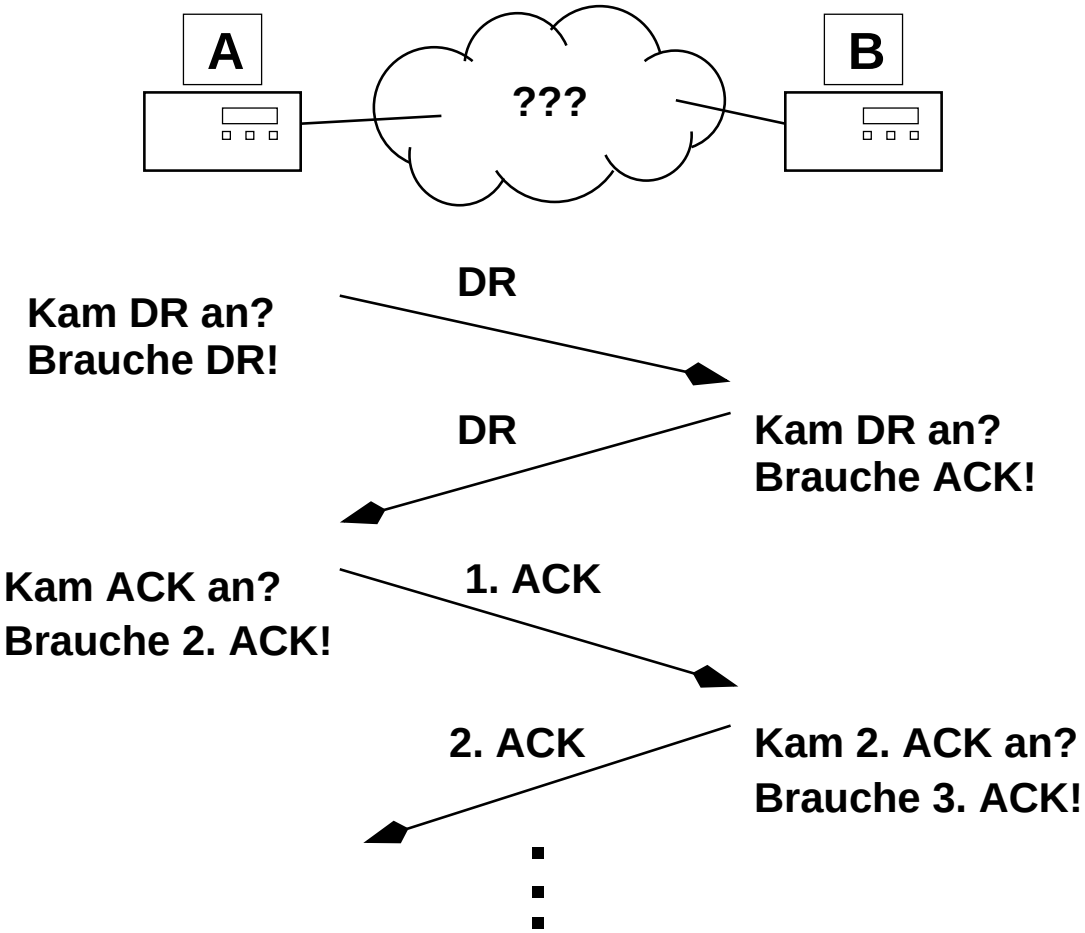
Nur eine der Parteien beendet durch Senden einer Disconnect-Anforderung (DR) die Verbindung.



(ACC = Connection Accepted)

Problematisch: Datenverlust möglich!

Symmetrischer Verbindungsabbau

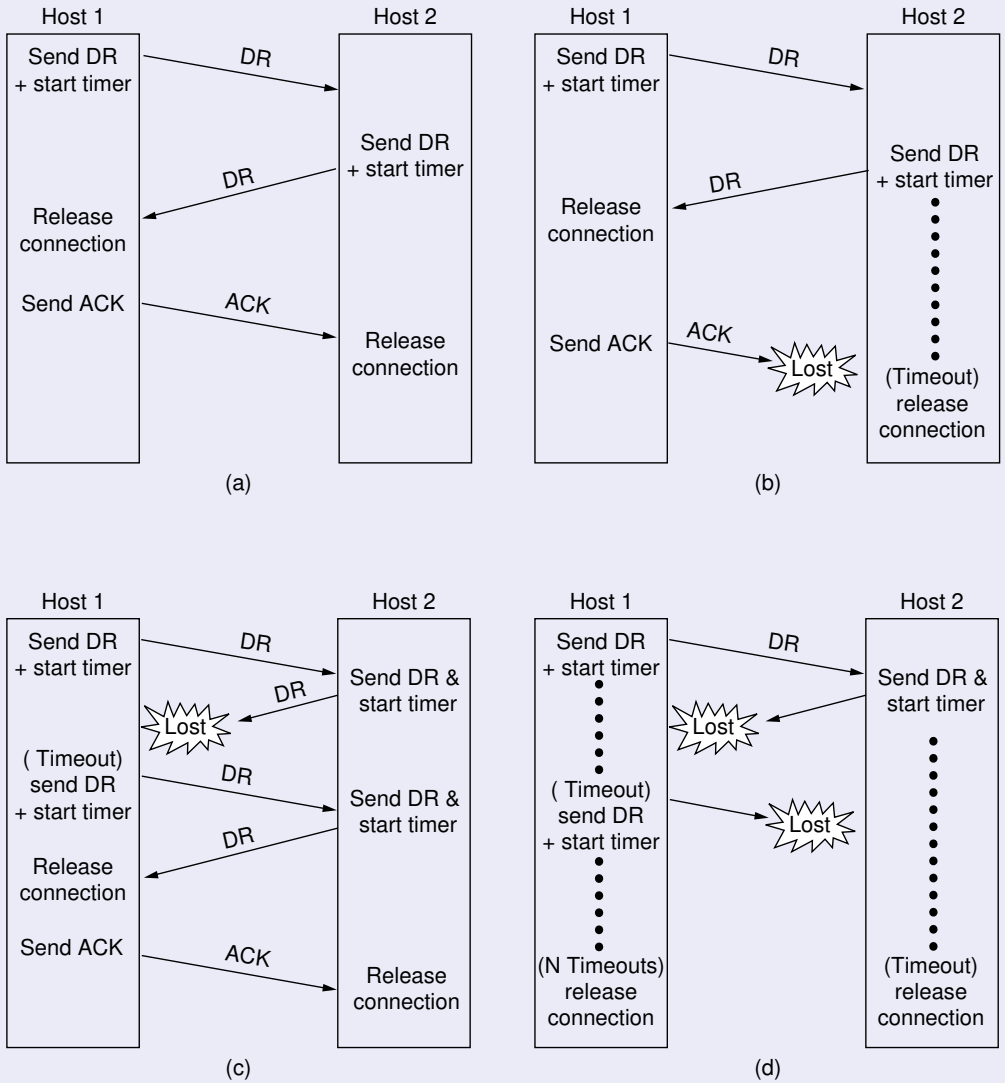


Problem

Falls Nachrichten auf unbestimmte Zeit verzögert werden können, gibt es (beweisbar) kein Protokoll, das garantiert, dass **immer beide** Parteien dem Abbau zustimmen. ($\Rightarrow$ *Distributed Consensus*)

Symmetrischer Verbindungsabbau (Forts.)

Pragmatische Lösung: Timeouts $\Rightarrow$ Problem gelöst???



Flusskontrolle und Pufferbereiche

Ein Speicherproblem

Server-Rechner können u. U. sehr viele Verbindungen gleichzeitig halten.
$\Rightarrow$ Sliding-Window-Protokolle brauchen zuviel Pufferspeicher!

Lösungsidee

- In unzuverlässigen Netzen muss der Sender noch nicht bestätigte TPDUs zwischenspeichern.
- Der Empfänger kann ankommende TPDUs ablehnen, wenn ihm der Pufferspeicher knapp wird.
- Der Sender muss aber auch in zuverlässigen Netzen TPDUs bis zu ihrer Bestätigung puffern! (Wieso?)

$\Rightarrow$ Wir vergeben die Pufferbereiche dynamisch, d. h. Sender und Empfänger müssen die Größe des Pufferbereichs aushandeln!

Puffer-Reservierung

Idee: Der Sender beantragt r Puffer, der Empfänger teilt ihm $g \leq r$ zu.
Falls $g < r$, kann der Empfänger ggf. später g bis zu r erhöhen.

	A	Message	B	Comments
1	$\longrightarrow$	<request 8 buffers>	$\longrightarrow$	A wants 8 buffers
2	$\longleftarrow$	<ack=15, buf=4>	$\longleftarrow$	B grants messages 0–3 only
3	$\longrightarrow$	<seq=0, data=m0>	$\longrightarrow$	A has 3 buffer left now
4	$\longrightarrow$	<seq=1, data=m1>	$\longrightarrow$	A has 2 buffer left now
5	$\longrightarrow$	<seq=2, data=m2>	$\times\longrightarrow$	Lost but A thinks it has 1 buffers left
6	$\longleftarrow$	<ack=1, buf=3>	$\longleftarrow$	B ACKs 0–1, permits 2–4 (m0 processed)
7	$\longrightarrow$	<seq=3, data=m3>	$\longrightarrow$	A has 1 buffer left
8	$\longrightarrow$	<seq=4, data=m4>	$\longrightarrow$	Afterwards A must block
9	$\longrightarrow$	<seq=2, data=m2>	$\longrightarrow$	A has timed out and retransmitted
10	$\longleftarrow$	<ack=4, buf=0>	$\longleftarrow$	Everything ACKed but A still blocked
11	$\longleftarrow$	<ack=4, buf=1>	$\longleftarrow$	1 buffer free (m1 processed)
12	$\longleftarrow$	<ack=4, buf=2>	$\longleftarrow$	B found a new buffer somewhere
13	$\longrightarrow$	<seq=5, data=m5>	$\longrightarrow$	A has 1 buffer left
14	$\longrightarrow$	<seq=6, data=m6>	$\longrightarrow$	A is blocked again
15	$\longleftarrow$	<ack=6, buf=0>	$\longleftarrow$	Everything ACKed, A still blocked
16	$\times\longleftarrow$	<ack=6, buf=5>	$\longleftarrow$	Potential deadlock!!!

Wie groß soll das Sendefenster sein?

Problem

Sender und Empfänger sind möglicherweise schneller als das Netz!

Lösung

Flusskontrolle findet beim Sender statt:
Der Sender muss immer dann stoppen, wenn das Sendefenster voll ist.

Bestimmung der Puffergröße

Sei c die mittlere Anzahl der pro Sekunde vom Netzwerk übertragenen TPDUs und r die mittlere RTT.

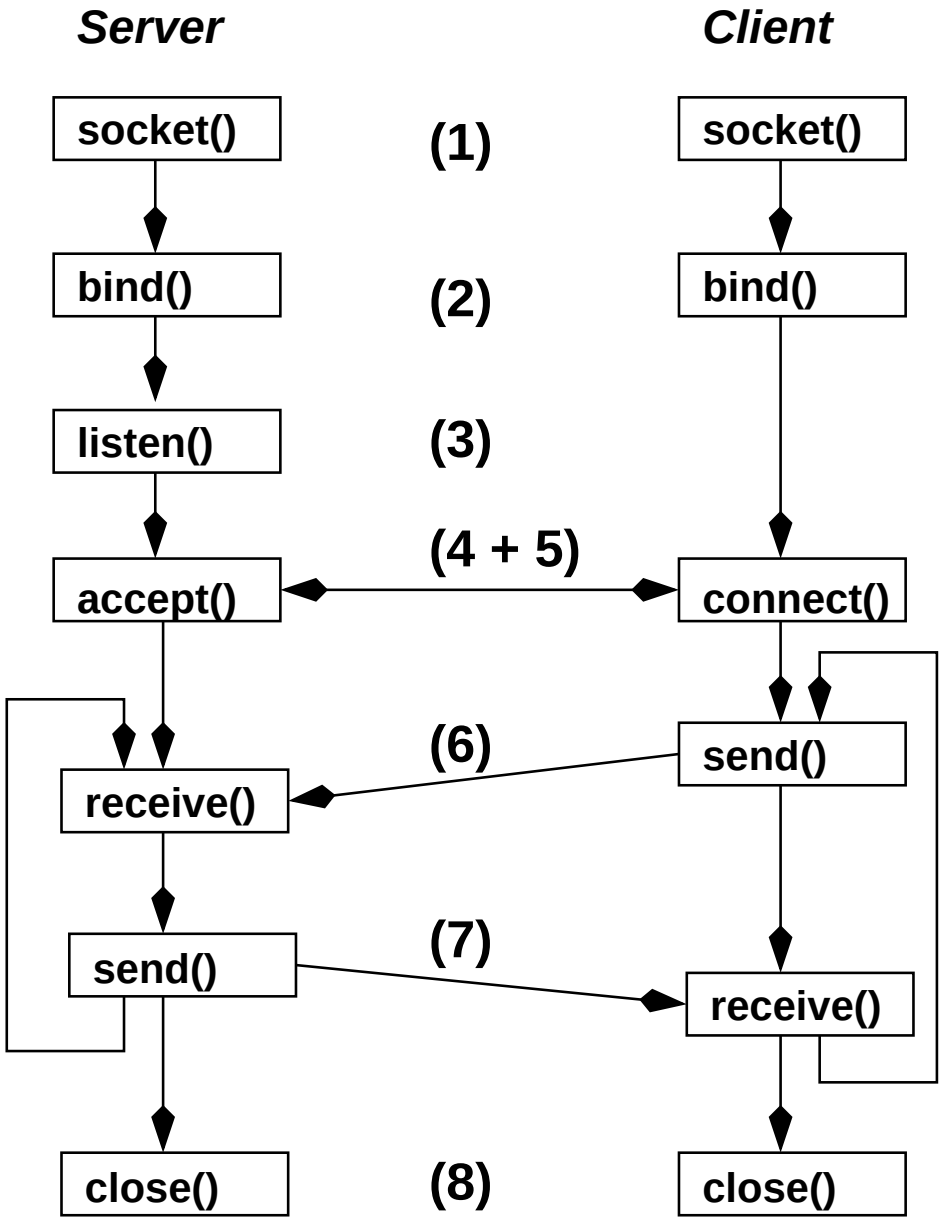
Der Sender benötigt im Normalfall im Puffer nur Platz für $c \cdot r$ TPDUs. (Wieso?)
Ein deutlich größerer Puffer wäre Verschwendung!

Die Transportschicht im Internet

TCP

- *Transmission Control Protocol* (TCP), RFC 793
  (z. B. : http://www.ibiblio.org/pub/docs/rfc/rfc793.txt)
- TCP stellt zuverlässige Ende-zu-Ende-Verbindungen zur Verfügung; Broadcasts und Multicasts werden nicht unterstützt.
- TPDUs heißen in TCP **Segmente**. Segmente haben einen Segment-Header und einen Nutzdatenteil. Ihre Gesamtlänge darf 65535 Bytes nicht übersteigen.
- Segmente werden **fragmentiert**, falls sie länger sind als von der Vermittlungsschicht erlaubt (MTU = Maximum Transfer Unit).
- Client und Server kommunizieren in TCP über *Sockets*.
  Sockets haben eine Adresse, die sich aus der IP-Adresse des Rechners und einer 16-Bit langen **Portnummer** zusammensetzt.
- Über TCP werden verbindungsorientierte Dienste angeboten, die Byte-Ströme verarbeiten können. $\Rightarrow$ Client-Server-Modell

Ablauf der Kommunikation



Zugriff auf Dienste

Frage

Woher weiß der Client, unter welcher Adresse der Server erreichbar ist?

Lösung

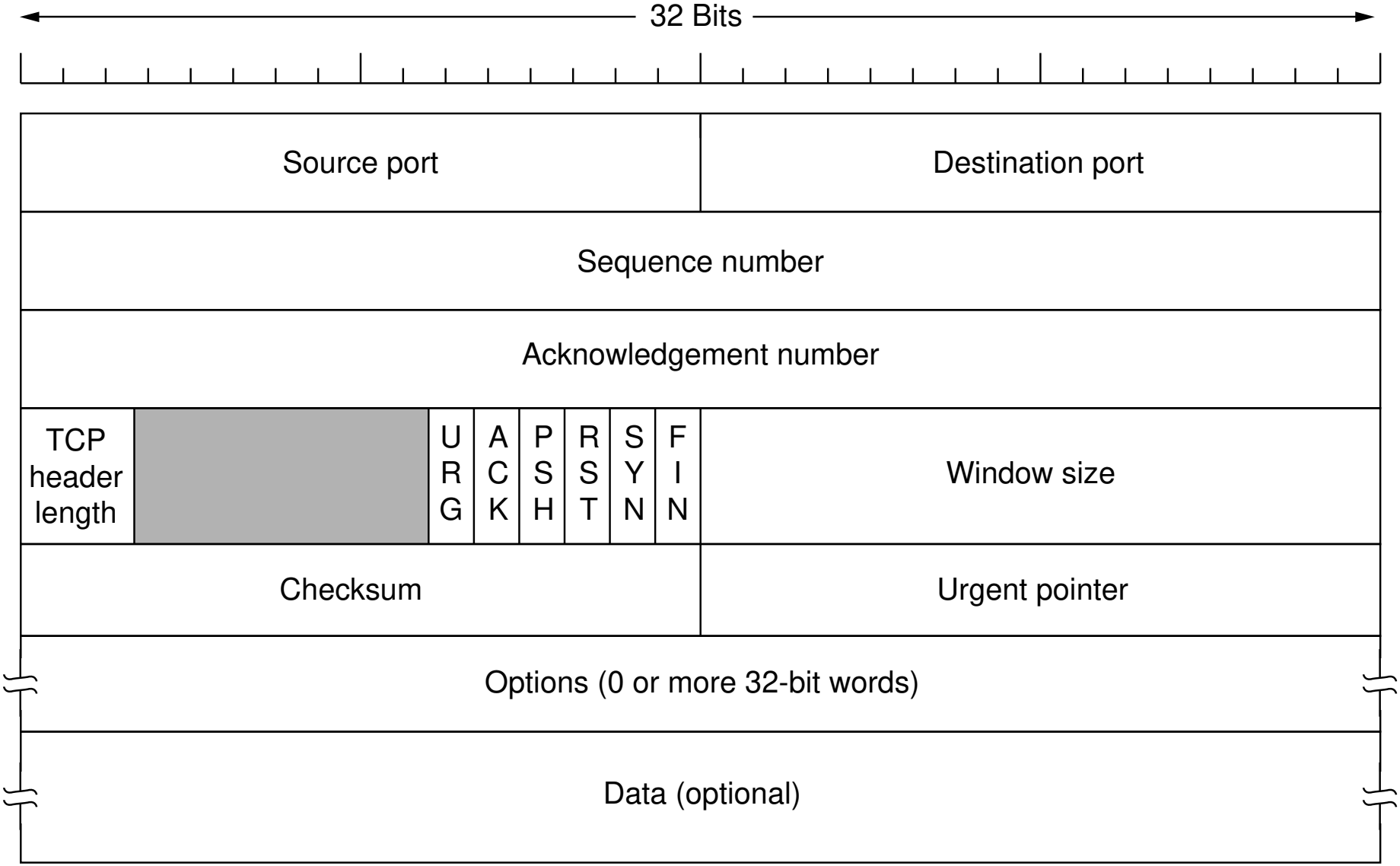
Standarddienste haben feste, allgemein bekannte Port-Nummern

- 0 – 1023: Well-known Ports (von der IANA verwaltet), nur System-Prozesse dürfen hier warten (auch *privilegierte Ports*):
- 1024 – 49151: Registrierte Ports, für Dienste, die üblicherweise auf bestimmten Ports laufen, z. B. Port 8080 als 2. Port für HTTP
- 49152 - 65535: Dynamic Ports / Private Ports, dienen zur Kommunikation zwischen den beiden beteiligten TCP-Schichten; werden nicht von Standarddiensten belegt und meistens von Clients verwendet

Auszug aus der Datei /etc/services :

```
...
ftp       21/tcp    # File Transfer [Control]
ssh       22/tcp    # SSH Remote Login Protocol
telnet    23/tcp    # Telnet
smtp      25/tcp    # Simple Mail Transfer
http      80/tcp    # World Wide Web HTTP
pop3     110/tcp    # Post Office Protocol - v3
https    443/tcp    # http over TLS/SSL
imaps    993/udp    # imap4 over TLS/SSL
...
```

Der TCP-Segment-Header



Frage: Wo stehen denn die IP-Adressen?

Der TCP-Segment-Header (Forts.)

Wichtige Flags

- Bestätigungen werden per Piggy-Backing versendet (ACK=1).
- Als Aufforderung zum Verbindungsaufbau schickt der Client ein Segment mit gesetztem SYN-Flag.
- Zum Verbindungsabbau wird ein Segment mit gesetztem FIN-Flag versendet.
- Zum Abweisen einer Verbindung wird ein Segment mit gesetztem RST-Flag gesendet.
- Das gesetzte URG-Flag (*urgent*), verlangt eine sofortige Bearbeitung (z. B. X-Window Mausbewegungen, Verbindungsabbruch, ...).

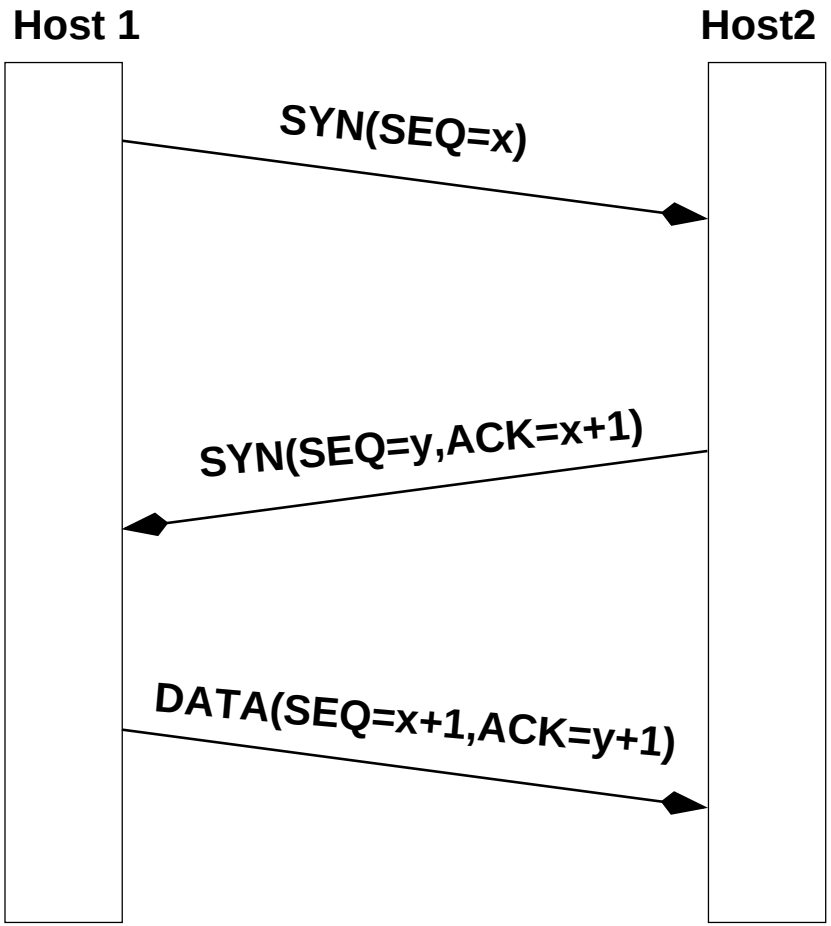
Pseudo-Header

Dem TCP-Header wird ein *Pseudo-Header* vorangestellt, der u.a. die IP-Adressen von Sender und Empfänger enthält und der von IP weiter verarbeitet wird.

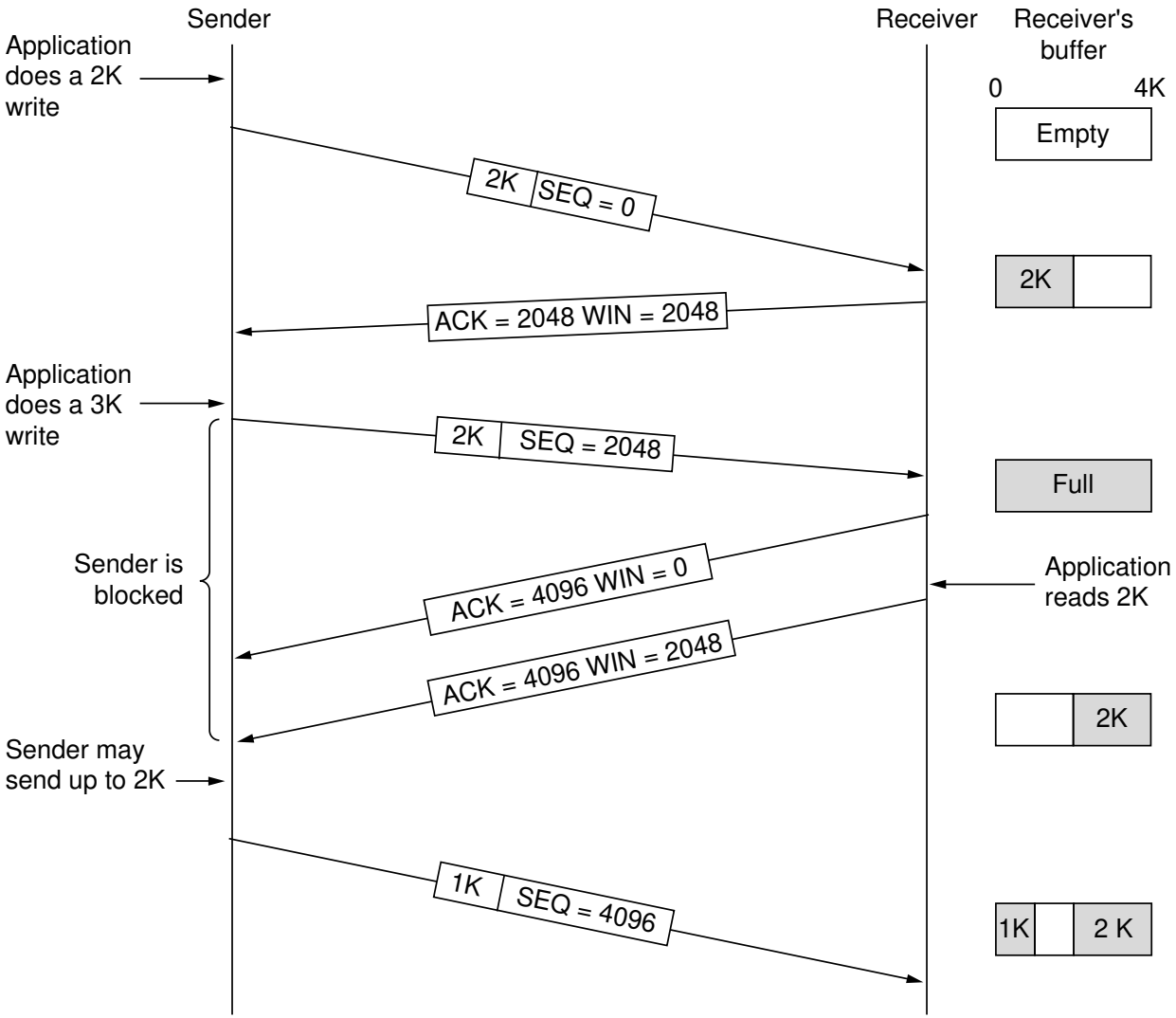
TCP (Forts.)

Verbindungsaufbau: TCP-Handshake

Zum Verbindungsaufbau werden die entsprechenden Flags im Header verwendet. Sequenznummern sind Byte-orientiert.



TCP: Fenstermanagement



Prinzip:

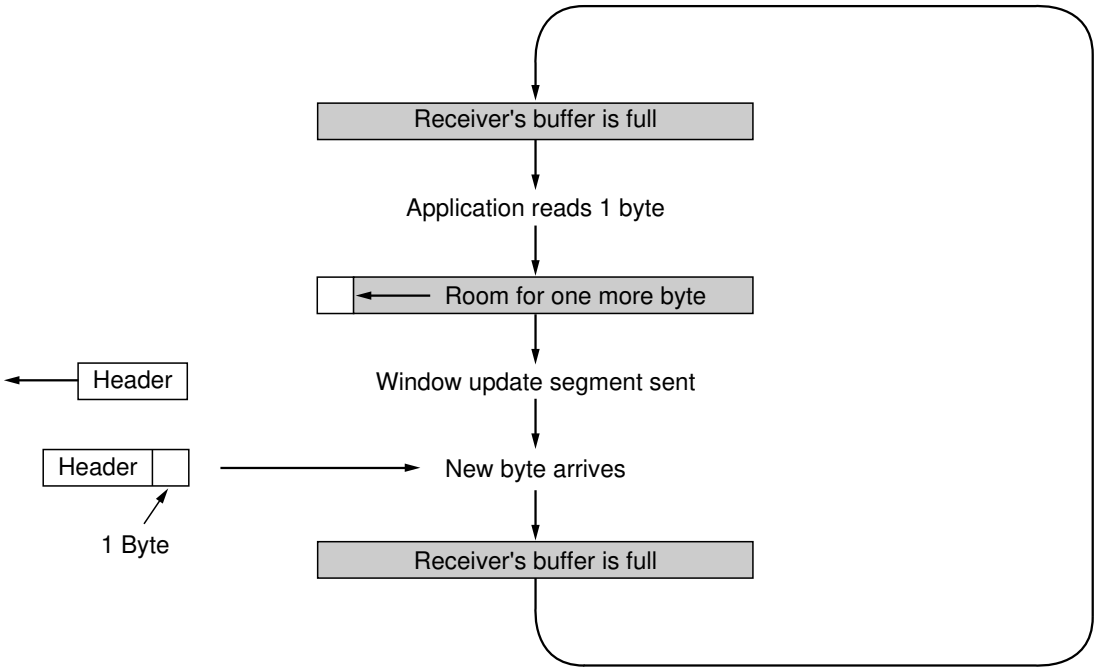
Jeder sendet jeweils

- Sequenznummer des nächsten Byte, das er erwartet
  $\Rightarrow$ *ACK no.*
- Anzahl Bytes, die der andere max. noch senden darf
  $\Rightarrow$ *Window size*

TCP: Fenstermanagement (Forts.)

Extremfall: "Silly-Window-Syndrome"

Eine (interaktive) Anwendung auf dem empfangenden Rechner liest immer nur ein Byte der erhaltenen Daten aus. Der Empfänger sendet jeweils eine Bestätigung und fordert den Sender auf, **ein** weiteres Byte zu senden!

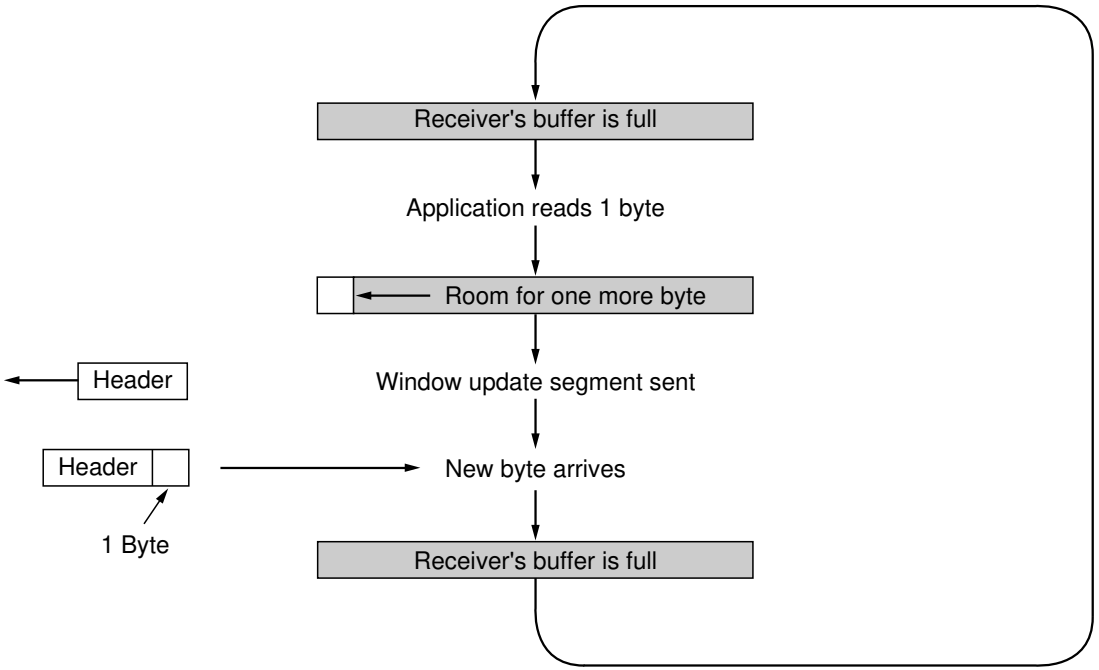


Abhilfe?

TCP: Fenstermanagement (Forts.)

Extremfall: "Silly-Window-Syndrome"

Eine (interaktive) Anwendung auf dem empfangenden Rechner liest immer nur ein Byte der erhaltenen Daten aus. Der Empfänger sendet jeweils eine Bestätigung und fordert den Sender auf, **ein** weiteres Byte zu senden!



Abhilfe?

Bei Sender **und** Empfänger Puffer ein

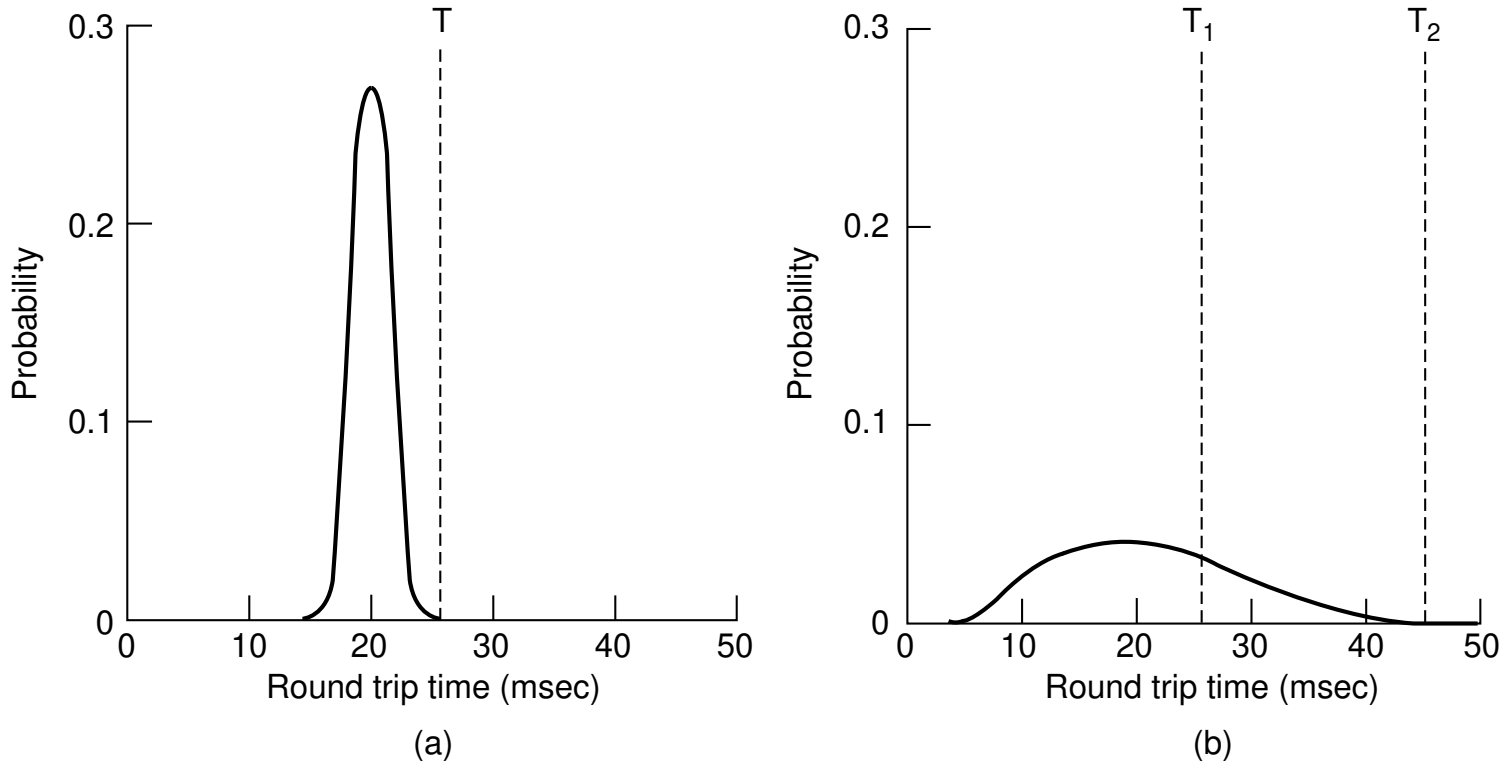
⇒ Wann soll gesendet werden?

⇒ Klappt das immer?

Timeouts

Wahl der Timeout-Zeit

Ist die Zeit zu lang, werden Übertragungsverluste erst sehr spät erkannt.
Ist sie zu kurz, kommt es zu unnötigen Wiederholungsübertragungen.



RTT und Timeout-Wahl(T)
(a) in Schicht 2 (b) in Schicht 4

Timeouts (Forts.)

Lösung (nach Jacobson)

Wir legen die momentan besten Schätzungen (RTT) und den gerade gemessenen Wert für die Round-Trip-Time (RTT_m) sowie die Schätzung der Standardabweichung (D) zugrunde.
Die Schätzwerte werden ständig aktualisiert:

$$RTT_{neu} = \alpha \cdot RTT + (1 - \alpha) RTT_m$$

(Glättungsfaktor $0 < \alpha < 1$, z. B. $\alpha = 7/8$)

$$D_{neu} = \alpha' \cdot D + (1 - \alpha')|RTT - RTT_m|$$

Die TCP-Timeout-Zeit berechnet sich damit zu:

$$Timeout = RTT_{neu} + 4 \cdot D_{neu}$$

$\Rightarrow$ Je größer die Streuung der Messwerte, desto länger bis zum Timeout.

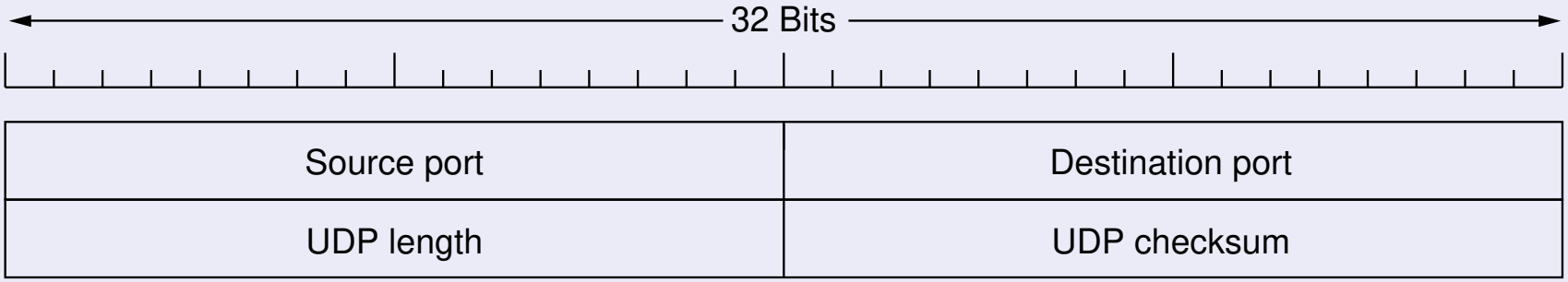
Alternative (nach Karn)

Bei jedem aufgetretenen Timeout wird die Timeout-Zeit verdoppelt.

UDP

User Datagram Protocol (UDP), RFC 768

- UDP-Header



- verbindungslose Kommunikation
- Übertragung von Datagrammen:
  senden gekapselter roher IP-Pakete
- Pseudo-Headers (wie TCP)
- Anwendung: Versenden kurzer Nachrichten ohne Zustellungsgarantie (zumeist in lokalen Netzen), z. B. Routing, NTP, RPC, ...

Kapitelvorschau

Die Verarbeitungsschicht

- verwendet die Schnittstellen der Transportschicht
- implementiert die Benutzerschnittstelle (und alles was noch fehlt ...)

Dienste der Verarbeitungsschicht

- **Domain Name System**
- Telnet
- E-Mail
- **World-Wide-Web**
- **Datenkomprimierung**
- **Sichere Kommunikation**
- Remote Procedure Call
- Multimedia
- ...

Einordnung in unser Schichtenmodell

Layer 1: Bitübertragungsschicht (*physical layer*)

Betrifft die Übertragung von Bits in Form von Signalen. Sie beschäftigt sich mit mechanischen bzw. elektrischen Fragen.

Layer 2: Sicherungsschicht (*data link layer*)

Regelt u. a. den Zugriff auf ein gemeinsam benutztes Übertragungsmedium, teilt die Bits in *Data Frames* auf und stellt eine übertragungsfehlerfreie Kommunikation zur Verfügung.

Layer 3: Vermittlungsschicht (*network layer*)

Ist u. a. verantwortlich für das Routing.

Layer 4: Transportschicht (*transport layer*)

Bietet verbindungslose und -orientierte Dienste mit unterschiedlichen Zuverlässigkeitsgarantien an. Sie ist die Schnittstelle für Anwendungen.

Die Verarbeitungsschicht (*application layer*)

implementiert, was dem Benutzer zur Verfügung steht ($\Rightarrow$ E-Mail, Telnet, WWW, ...).

Domain Name System (DNS)

Wieso DNS?

- Weil IP-Adressen für Menschen schwer zu merken sind (z. B. 141.72.70.16)
  $\Rightarrow$ Rechnername + Domain-Name (muss insgesamt eindeutig sein)
  $\Rightarrow$ IP-Adressen dürfen sich ruhig ändern
- Rechner werden oft nach Funktion benannt (z. B. **www**.dhbw-mannheim.de)
- keine 1:1-Abbildung!

Wie verwaltet man das?

- lokale Tabelle, z. B. UNIX: `/etc/hosts` ($\Rightarrow$ skaliert nicht im Internet!)
- verteilte Datenbank: viele Rechner verwalten jeweils einen Teil

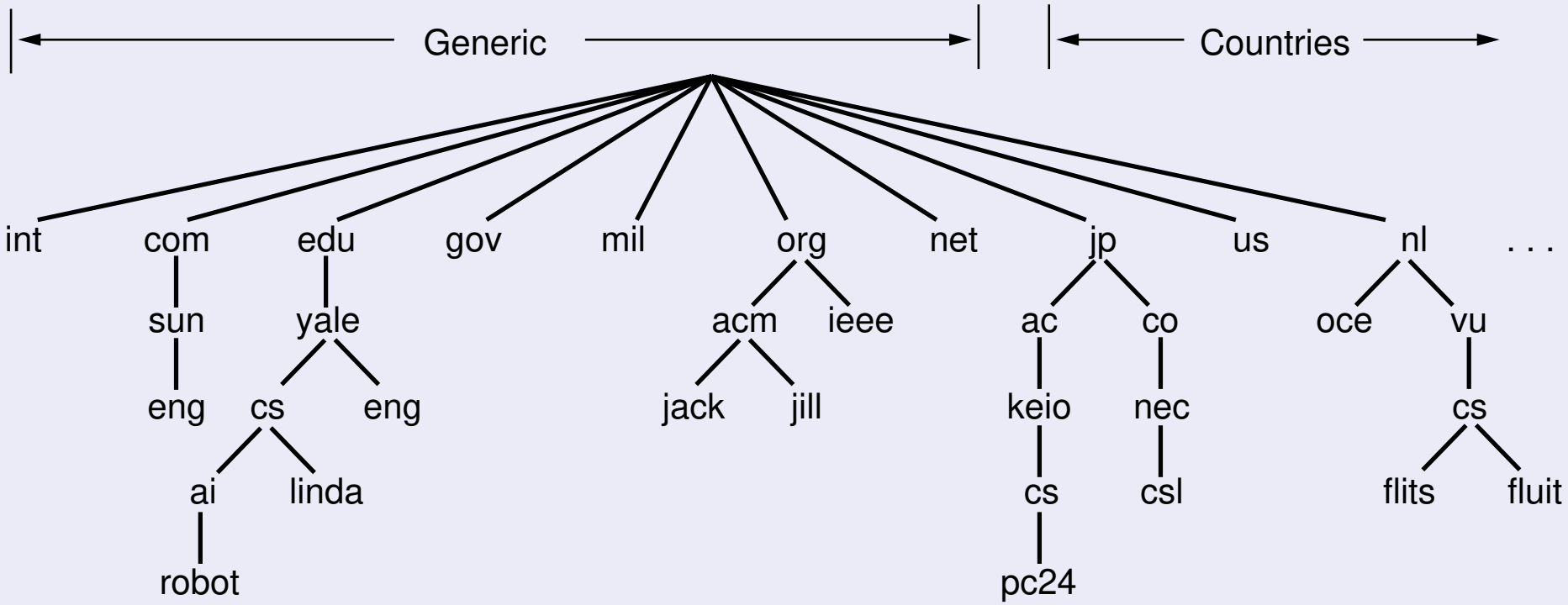
DNS-Server

- leisten die Umsetzung *Namensbezeichnung* $\Leftrightarrow$ *IP Adresse*.
- verwalten oft auch weitere Informationen über Rechner und Dienste
- speichern Ergebnisse für begrenzte Zeit zwischen ($\Rightarrow$ DNS-Cache)

DNS (Forts.)

Domainnamen

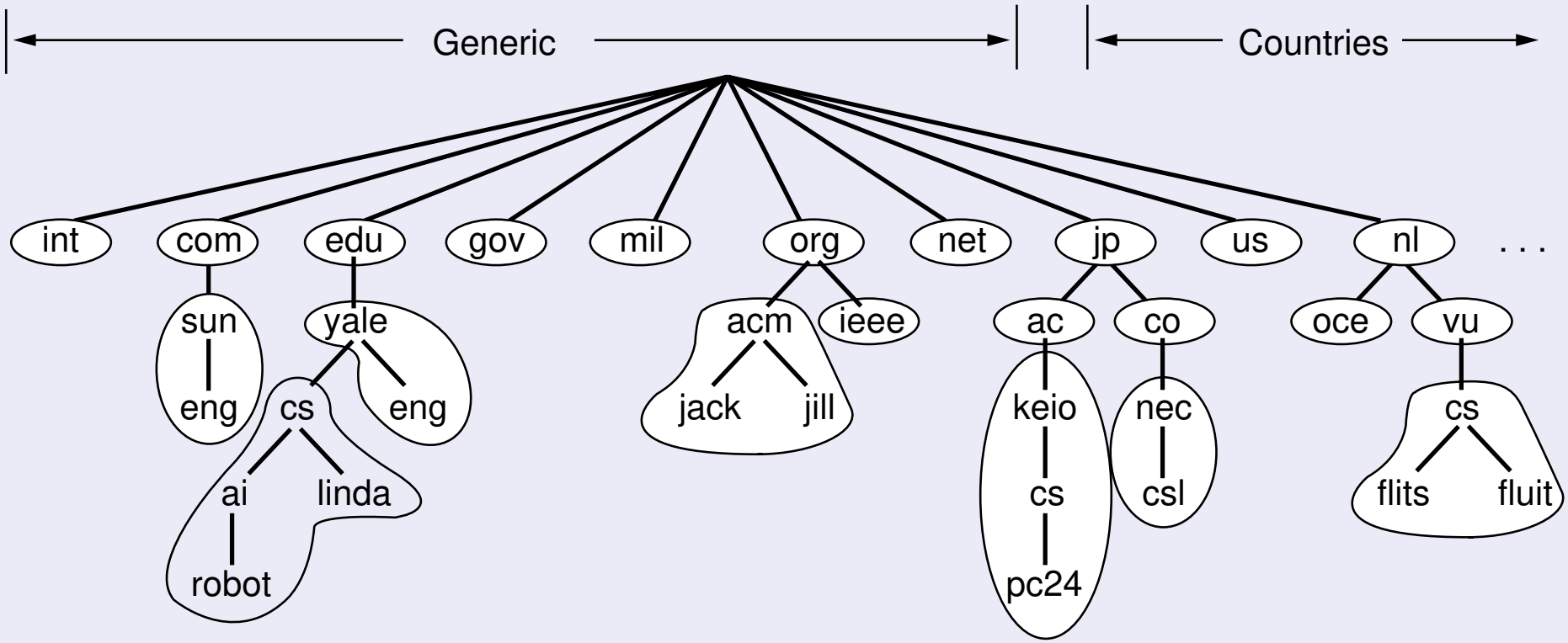
Es sind generische (.com .org ...) sowie länderspezifische (.de .fr ...) Top-Level-Domains (TLDs) definiert, unter denen jeweils bestimmte Organisationen verwaltet werden.



DNS (Forts.)

Wie werden die Server platziert?

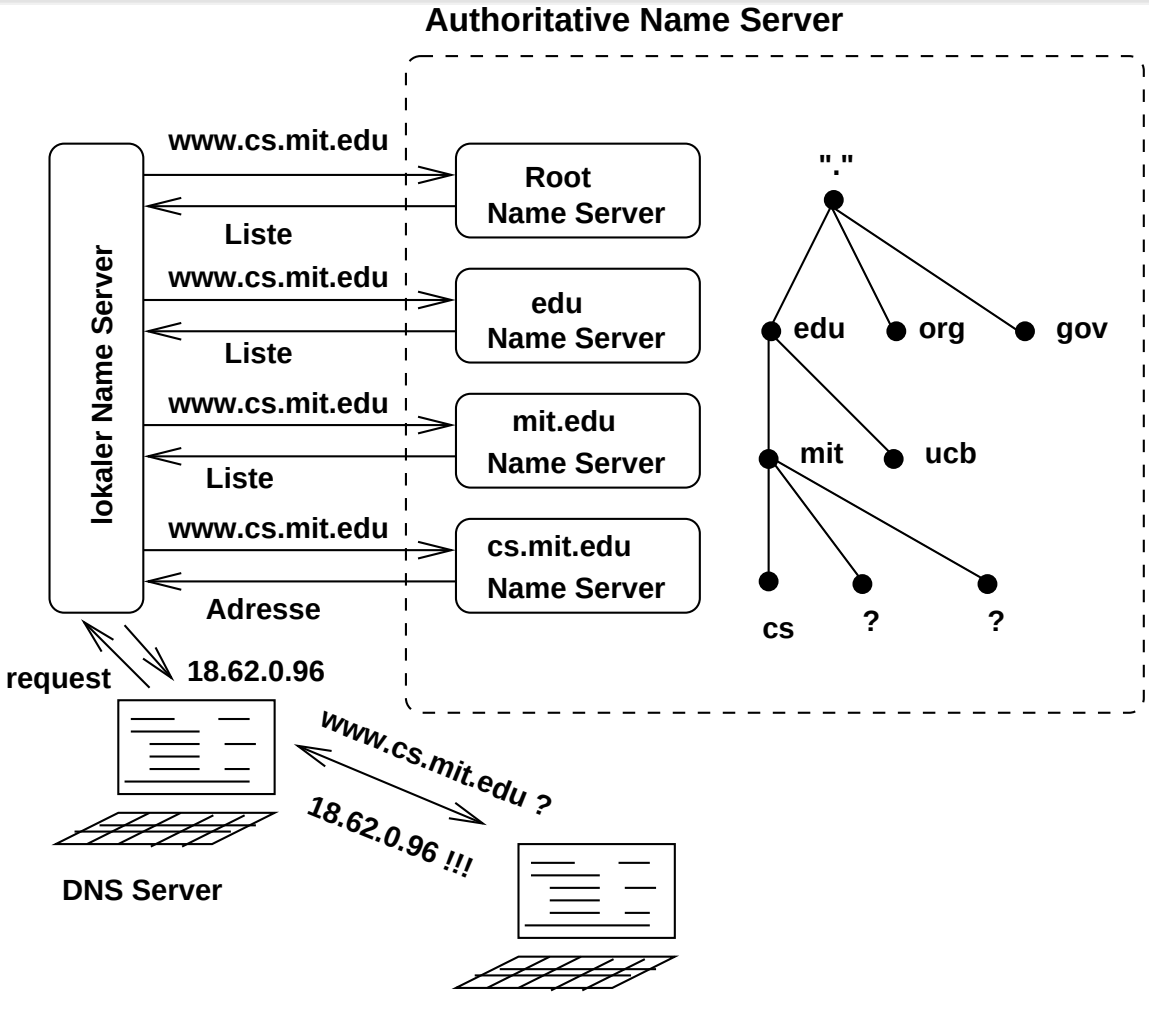
Der gesamte Namensraum ist in disjunkte **Zonen** unterteilt. Für jede Zone existieren i.d.R. mehrere gleichberechtigte DNS-Server.
Es gibt weltweit z.Zt. 13 *Root Nameserver*.



DNS (Forts.)

Wie werden Domainnamen aufgelöst?

Entweder iterativ (s. Bild) – oder rekursiv.



Das World-Wide-Web (WWW)

Die *"grafische Oberfläche des Internet"*

- Client/Server Prinzip
- Personen oder Organisationen stellen **Dokumente** zum Abruf zur Verfügung
- Dokumente sind in HTML (*Hypertext Markup Language*) beschrieben.

- Dokumente bestehen aus Text, Multimedia-Informationen (i. Allg. Bilder) und/oder **Hyperlinks** auf andere Dokumente bestehen
- Multimedia-Informationen werden als Verweise auf entsprechende Dateien eingebunden und automatisch vom Browser nachgeladen und angezeigt
- Das Auswählen eines Hyperlinks führt dazu, dass der Browser das entsprechende Dokument lädt und anzeigt

- Client und Server verwenden das *Hypertext Transfer Protocol* (HTTP)
- HTTP basiert auf TCP

WWW: Dokumentenbezeichner

Eine URL (*Uniform Resource Locator*) ist ein global eindeutiger Dokumentenbezeichner. Bestandteile einer URL sind:

Zugriffsprotokoll

- http (normalerweise)
- ftp (File Transfer)
- file (lokale Datei)
- news (News Group)
- gopher (Gopher)
- mailto (Senden von E-Mails)
- telnet (Remote-Login)
- ...

Rechnername

DNS-Name des HTTP-Servers; ggf. die Portnummer (i. d. R. Port 80)

Dateiname

Pfad- und Dateiname relativ zum "Server-Root-Verzeichnis"

Beispiel:

`http://cruncher.dhbw-mannheim.de:80/~pagnia/index.html`

HTTP

HTTP-Befehle

- GET: lesen eines Dokuments

  ```
  GET /fachrichtungen/wirtschaft.html
  ```

- in Verbindung mit "Option" `HTTP/1.0` bzw. `HTTP/1.1`:
  $\Rightarrow$ Dokument wird im MIME-Format unter Angabe des Typs zurückgeliefert
- HEAD: lesen des Dokumenten-Headers
- POST: senden von Informationen an den Server
- ferner gibt es noch: PUT, DELETE sowie LINK und UNLINK

Zustandslose Server

In HTTP sind Server zustandslos, sodass der Client für jeden Zugriff eine neue Verbindung etablieren und jeweils den vollständigen Dokumentennamen angeben muss. (**Frage:** Welche Vorteile hat das?)

Datenkomprimierung

Vorteil

Viele Daten lassen sich durch Komprimierung wesentlich reduzieren
$\Rightarrow$ höhere Übertragungsraten möglich!

Nachteil

Das Komprimieren erfordert Rechenzeit.
Bei manchen Verfahren benötigen Komprimieren und Dekomprimieren unterschiedlich viel Rechenzeit.

Prinzipielle Verfahren

- Entropiekodierung:
  Der Bitstrom wird – ohne Berücksichtung seiner Bedeutung – mit generischen Verfahren behandelt ($\Rightarrow$ verlustfrei).
- Quellkodierung:
  Der Bitstrom wird – unter Berücksichtigung der Semantik – mit speziellen Verfahren behandelt ($\Rightarrow$ meist verlustbehaftet).

Datenkomprimierung (Forts.)

Entropiekodierung

- **Lauflängenkodierung:**
  Sequenzen aufeinanderfolgender gleicher Bitfolgen werden als Wiederholungen kodiert,
  z. B. “AAAAAAAAAA” $\Rightarrow$ “10A”.

- **Statistische Kodierung:**
  Zeichen(-folgen), die häufig auftreten, bekommen eine kurze Kodierung, seltene erhalten eine längere Kodierung, z. B. Morsekode, Huffman-Kodierung, Lempel-Ziv-Kodierung, ...

Huffman-Kodierung

Eigenschaften

- optimale verlustfreie Komprimierung
- Präfix-Code

Vorgehen

(1) Bestimmen der Symbolhäufigkeiten der Eingabedaten
(2) Berechnen des Huffman-Baumes
(3) Erzeugen der Code-Liste aus dem Huffman-Baum
(4) Generieren der Ausgabedaten

Huffman-Kodierung (Forts.)

Algorithmus zum Aufstellen des Baums

(i) Erzeuge pro Symbol einen Knoten mit dessen Häufigkeit als Wert.

(ii) Fasse die beiden Knoten mit den geringsten Werten zu einem neuen Knoten zusammen; der Wert dieses Knotens ergibt sich durch Addition der beiden Häufigkeiten.

(iii) Wiederhole Schritt (ii) solange bis der Baum nur noch eine Wurzel besitzt.

(iv) Traversiere den Baum beginnend bei der Wurzel und markiere die Kanten mit Nullen und Einsen.

(v) Leite die Kodierungen der Symbole aus den Kantenmarkierungen ab.

Huffman-Kodierung: Beispiel

Eingabedaten

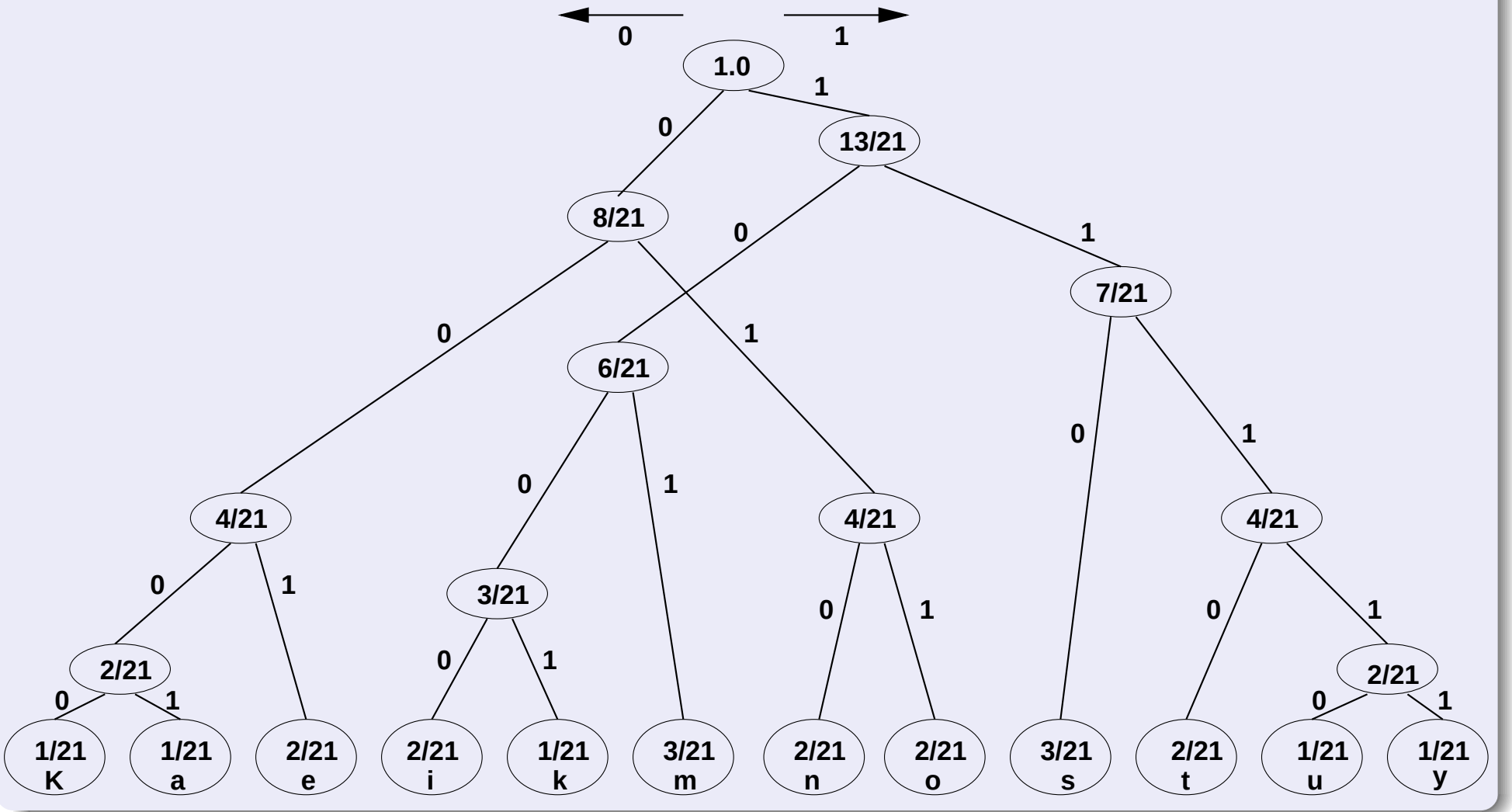
`Kommunikationssysteme` (Länge: 21 Buchstaben)

Symbolhäufigkeiten (hier: Symbole = Buchstaben)

Symbol	absolute Häufigkeit	relative Häufigkeit
K	1	0.047
a	1	0.047
e	2	0.094
i	2	0.094
k	1	0.047
m	3	0.141
n	2	0.094
o	2	0.094
s	3	0.141
t	2	0.094
u	1	0.047
y	1	0.047

Huffman-Kodierung: Beispiel (Forts.)

Berechnen des Huffman-Baums und Anbringen der Markierungen



Huffman-Kodierung: Beispiel (Forts.)

Erzeugen der Code-Liste

Symbol	Kodierung
K	0000
a	0001
e	001
i	1000
k	1001
m	101
n	010
o	011
s	110
t	1110
u	11110
y	11111

Huffman-Kodierung: Beispiel (Forts.)

Erzeugen der Code-Liste

Symbol	Kodierung
K	0000
a	0001
e	001
i	1000
k	1001
m	101
n	010
o	011
s	110
t	1110
u	11110
y	11111

Generieren der Ausgabedaten

000001110110111100101000
100100011110100011010110
110111111110111000110100 1
(74 Bit)

⇒ **Wie groß ist die Ersparnis?**

⇒ **Wie wird dekodiert?**

Quellkodierung

Methoden

- **CLUT-Kodierung** (Color LookUp Table)**:**
  Für Bilder: Farbwerte werden in einer vorangestellten Tabelle aufgeführt und über ihren Index kodiert.

- **Differenzialkodierung:**
  Datenblöcke werden durch ihren Unterschied zum jeweiligen Vorgängerblock kodiert. (Funktioniert nur mit bestimmten Daten)

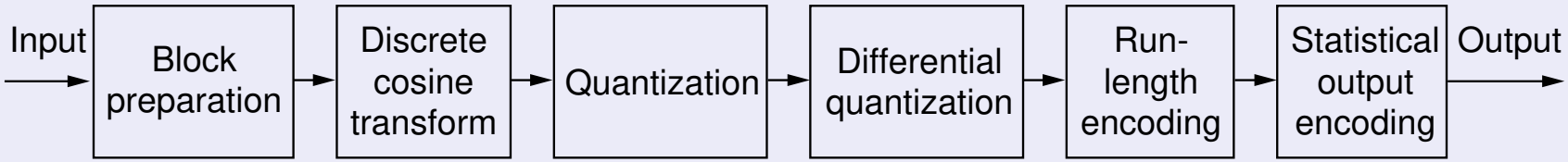
- **Transformationskodierung:**
  Transformation der Datenblöcke in einen anderen Zustandsraum
  (z. B. Fourier-Transformation)
  Verfahren: Diskrete-Cosinus-Transformation (DCT), ...

- **Vektorquantisierung:**
  Zweidimensionale Bilder werden in Quadrate gerastert. Die Referenzquadrate werden in einer Tabelle aufgeführt und ähnliche Quadrate werden über den Tabellenindex kodiert.

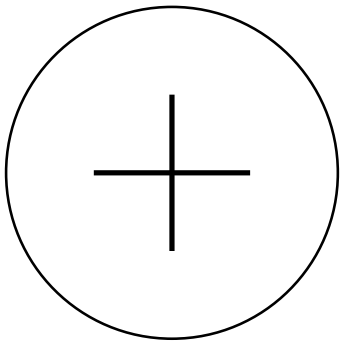
Quellkodierung (Forts.)

JPEG

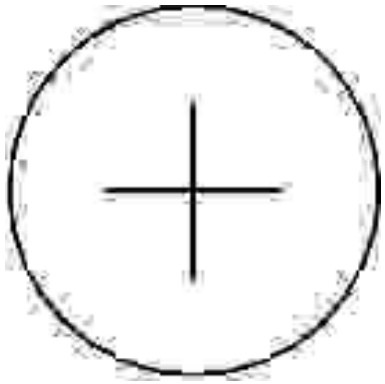
- *Joint Photographic Expert Group*
- Norm zur Komprimierung von Bildern, insb. Fotos
- vier unterschiedliche Betriebsmodi,
  z. B. verlustbehafteter sequenzieller Modus:



Beispiel:



Original-Bild (TIFF, 55812 Bytes)



Komprimiert (JPEG, 1429 Bytes)

Datensicherheit

Idee

Daten, die im Internet übermittelt werden, können prinzipiell von Angreifern gelesen und sogar verändert werden. Deshalb sollten wichtige Daten **verschlüsselt** werden.

Ziele der Verschlüsselung

- Schutz der Vertraulichkeit der Daten
- Schutz der Integrität der Daten
- Sicheres Ermitteln des Verfassers der Daten

Prinzipielle Verfahren

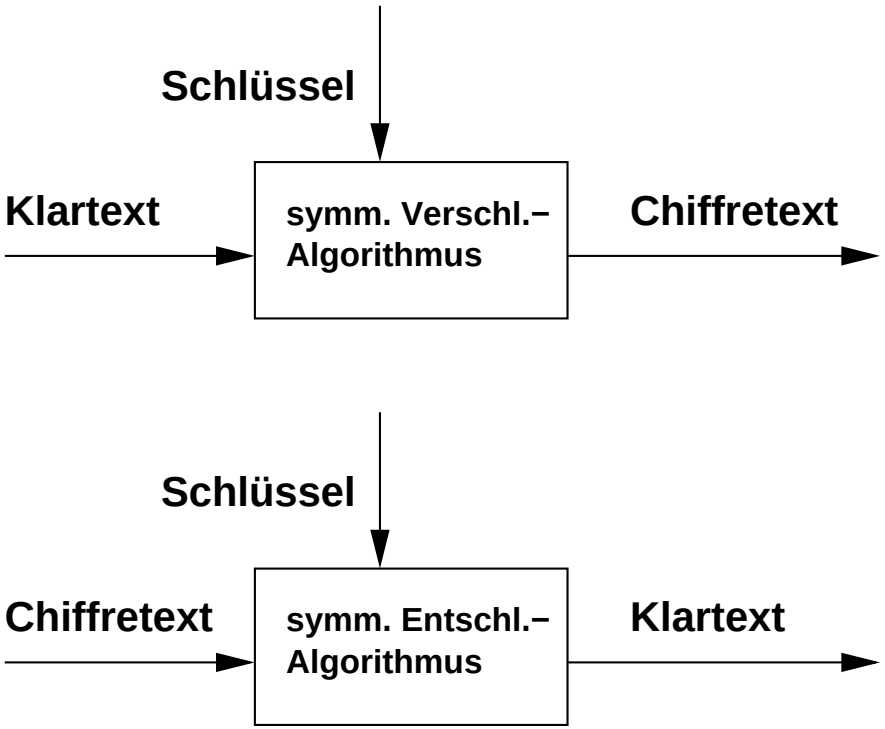
- Symmetrische Verschlüsselung
- Asymmetrische Verschlüsselung

(Def. Datenschutz: Schutz personenbezogener Daten)

Symmetrische Verschlüsselung

Idee

Es gibt einen öffentlich bekannten Verschlüsselungsalgorithmus, der als Eingabeparameter den zu verschlüsselnden *Klartext* sowie einen *Schlüssel* verwendet und der nun daraus den *Chiffretext* berechnet. Mit Hilfe des Entschlüsselungsalgorithmus und **demselben** Schlüssel kann der Empfänger der Nachricht aus dem Chiffretext den Klartext zurückgewinnen.



Symmetrische Verschlüsselung (Forts.)

Sicherheit

Für sichere Verschlüsselungsverfahren existiert keine Möglichkeit, ohne Kenntnis des Schlüssels den Chiffretext korrekt zu entschlüsseln.
Einem Angreifer bleibt als einzige Möglichkeit ein *Brute-Force-Angriff*, d. h. das systematische Durchprobieren *aller* möglichen Schlüssel.
Durch Verwendung ausreichend langer zufälliger Schlüssel kann dieser Angriff nahezu unmöglich gemacht werden.

Verfahren:

– AES (128–256 Bit)

– Threefish (128 – 1024 Bit)

– 3DES (112 Bit, unsicher)

– DES (56 Bit, unsicher!)

– ...

Probleme:

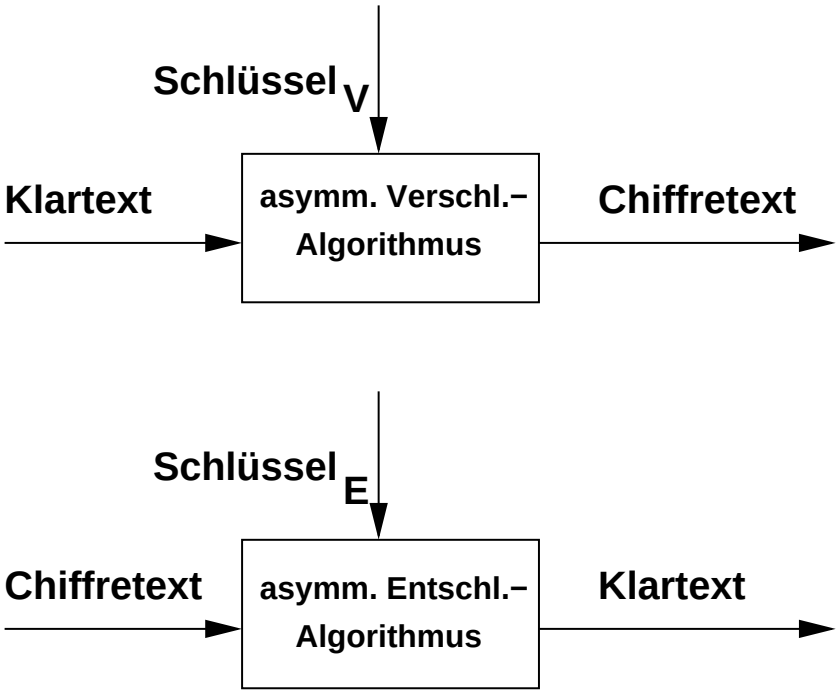
(1) Sender **und** Empfänger müssen den Schlüssel kennen.

(2) Kenne ich wirklich den Sender?

Asymmetrische Verschlüsselung

Idee

Zum Verschlüsseln und Entschlüsseln werden zwei unterschiedliche Schlüssel verwendet. Der Verschlüsselungsschlüssel ***V*** wird öffentlich bekannt gegeben, der Entschlüsselungsschlüssel ***E*** bleibt geheim. Da asymmetrische Verfahren deutlich rechenintensiver und damit langsamer sind als symmetrische werden sie i. Allg. ausschließlich zum Schlüsseltausch für symmetrische Verfahren und zum Schutz der Datenintegrität sowie dem sicheren Ermitteln des Urhebers eingesetzt.



Verfahren:

- Diffie-Helmann (Schlüsseltausch)
- RSA
- ElGamal
- ...

(Schlüssellängen mind. 2048)

TLS / SSL

Transport Layer Security

- Sicherheitsprotokoll für TCP-basierte Dienste, z. B. HTTP $\Rightarrow$ HTTPS
  – diese erhalten neue Portnummern (z. B. 443 statt 80)
- implementiert Serverauthentisierung, Datenintegrität sowie Vertraulichkeit der Kommunikation
- verwendet Serverzertifikate
  - ausgestellt von sog. CAs (vertrauenswürdige Zertifizierungsstellen)
  - enthalten u. a. den Servernamen und den Public Key des Servers

TLS / SSL (Forts.)

Prinzipieller Ablauf einer TLS-Sitzung