Rechnernetze I

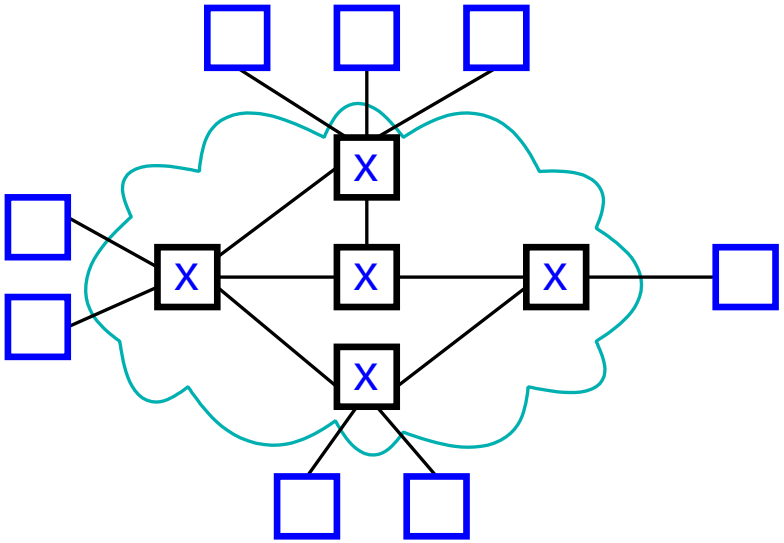
SoSe 2019

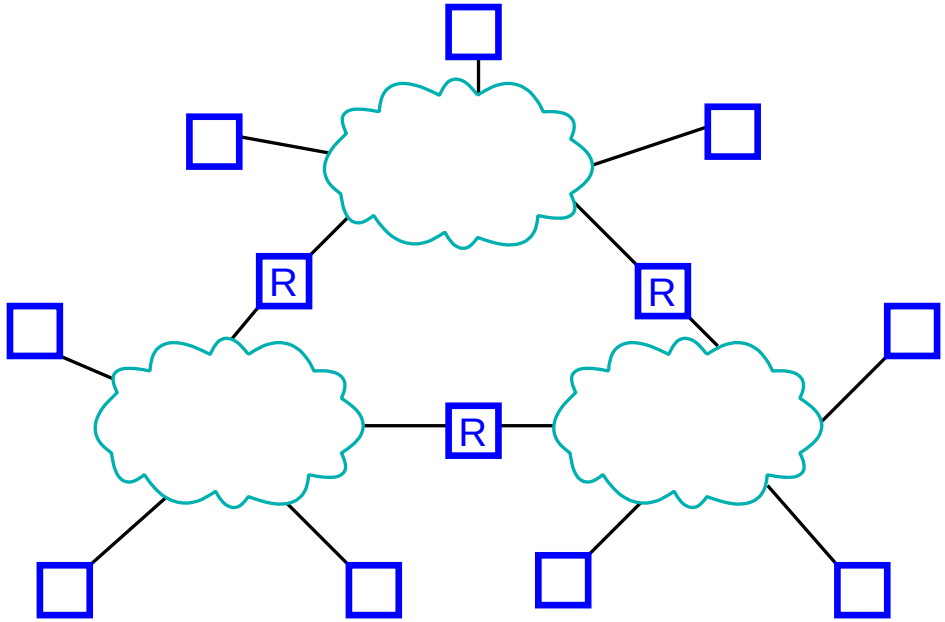
1 Einführung

Allgemeine Struktur eines Netzwerks

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

  oder
  - mehreren Netzwerken, verbunden durch ein oder mehrere Knoten

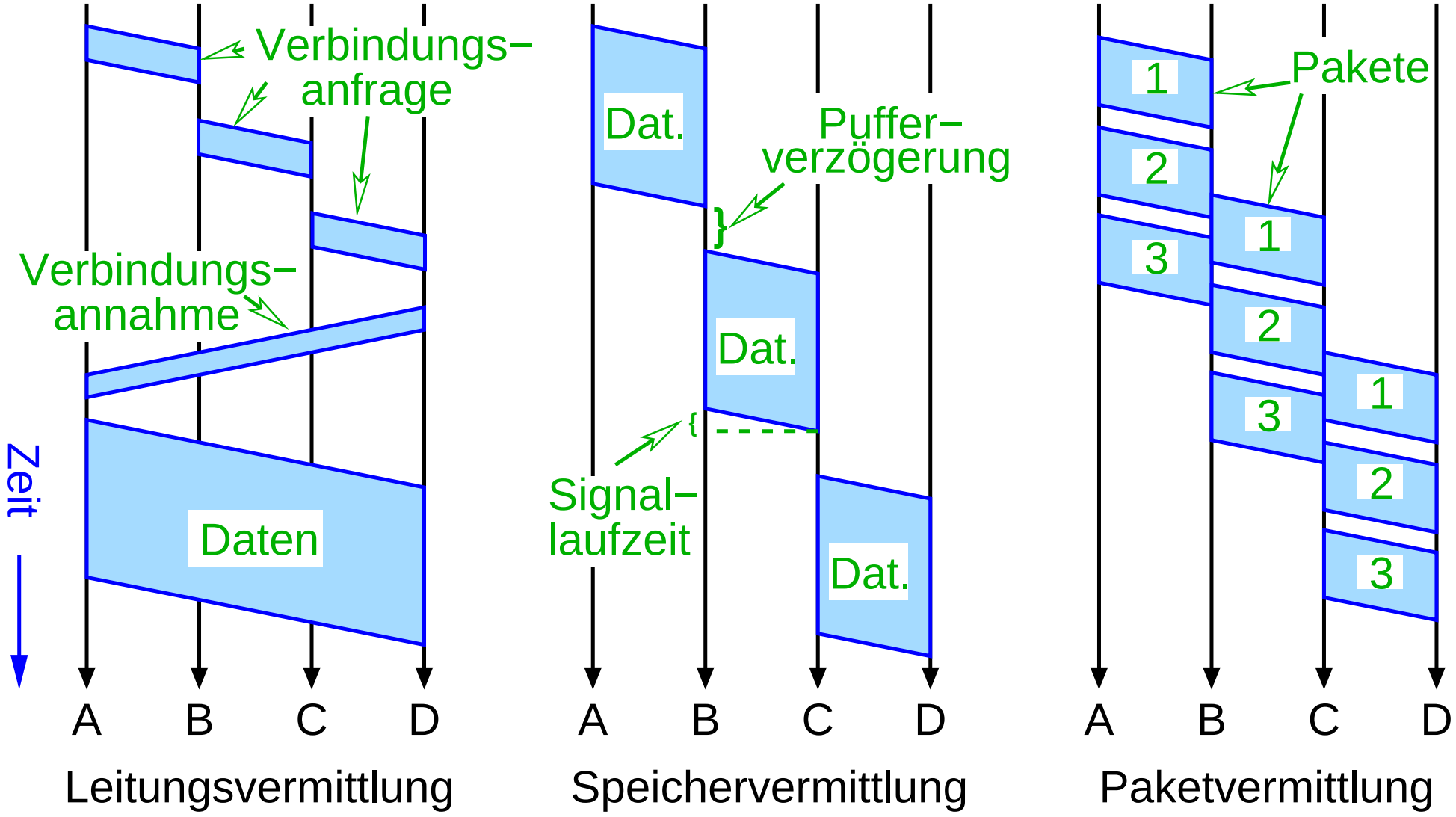




Wichtige Begriffe / Aufgaben

- **Adressierung**
  - physische Adresse: identifiziert Host weltweit eindeutig, keine Information über das Netz des Hosts
  - logische Adresse: identifiziert Netz und Host in diesem Netz
  - Verwendung numerischer Adressen
- Anzahl der Empfänger
  - **Unicast**: genau einer
  - **Broadcast**: alle
  - **Multicast**: mehrere bestimmte
- **Routing / Forwarding (Vermittlung / Weiterleitung)**
  - Weiterleitung der Daten zum Empfänger durch Zwischenknoten

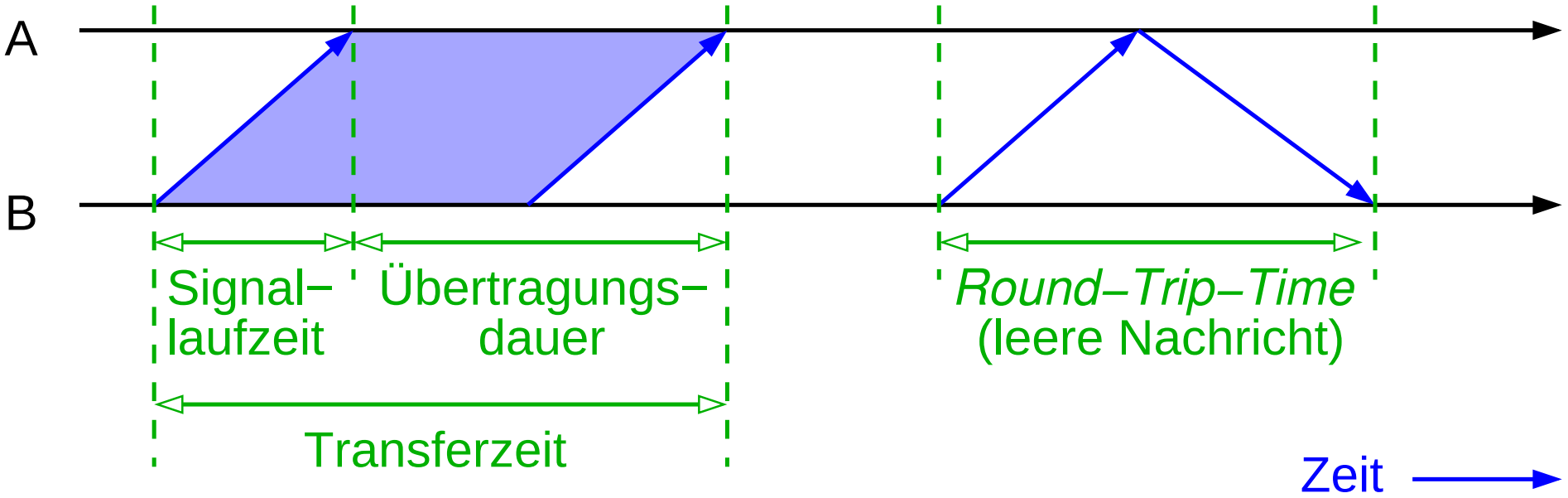
Zeitablauf der Datenübertragung



1.5 Leistungsparameter

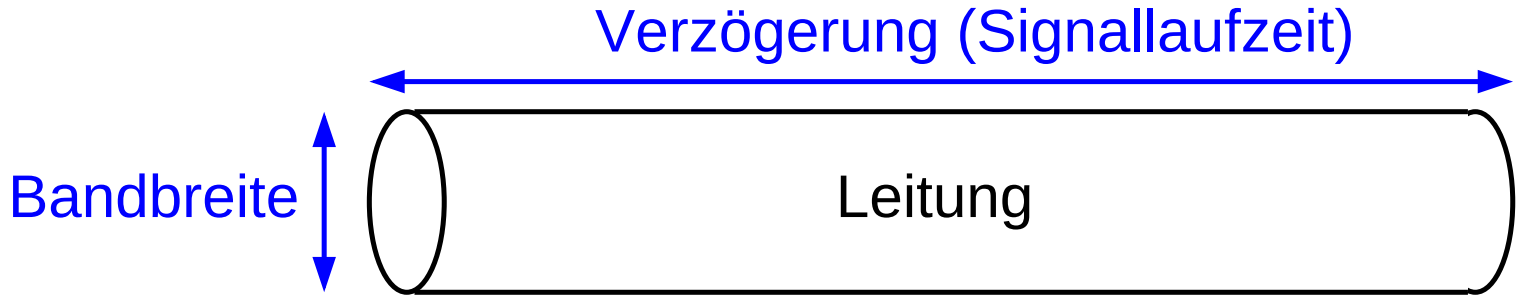
- **Bandbreite (Übertragungsrate)**
  - Übertragbares Datenvolumen pro Zeiteinheit
  - Maßeinheit: Bits pro Sekunde (**b/s** bzw. **bps**)
  - Vorsicht bei den Maßeinheiten:
    - 1 kb/s = 1000 Bits/Sekunde, 1 Mb/s = 1000 kb/s
    - 1 KB = 1024 Bytes, 1 MB = 1024 KB
      (nach NIST: KiB statt KB, MiB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

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



1.5 Leistungsparameter ...

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



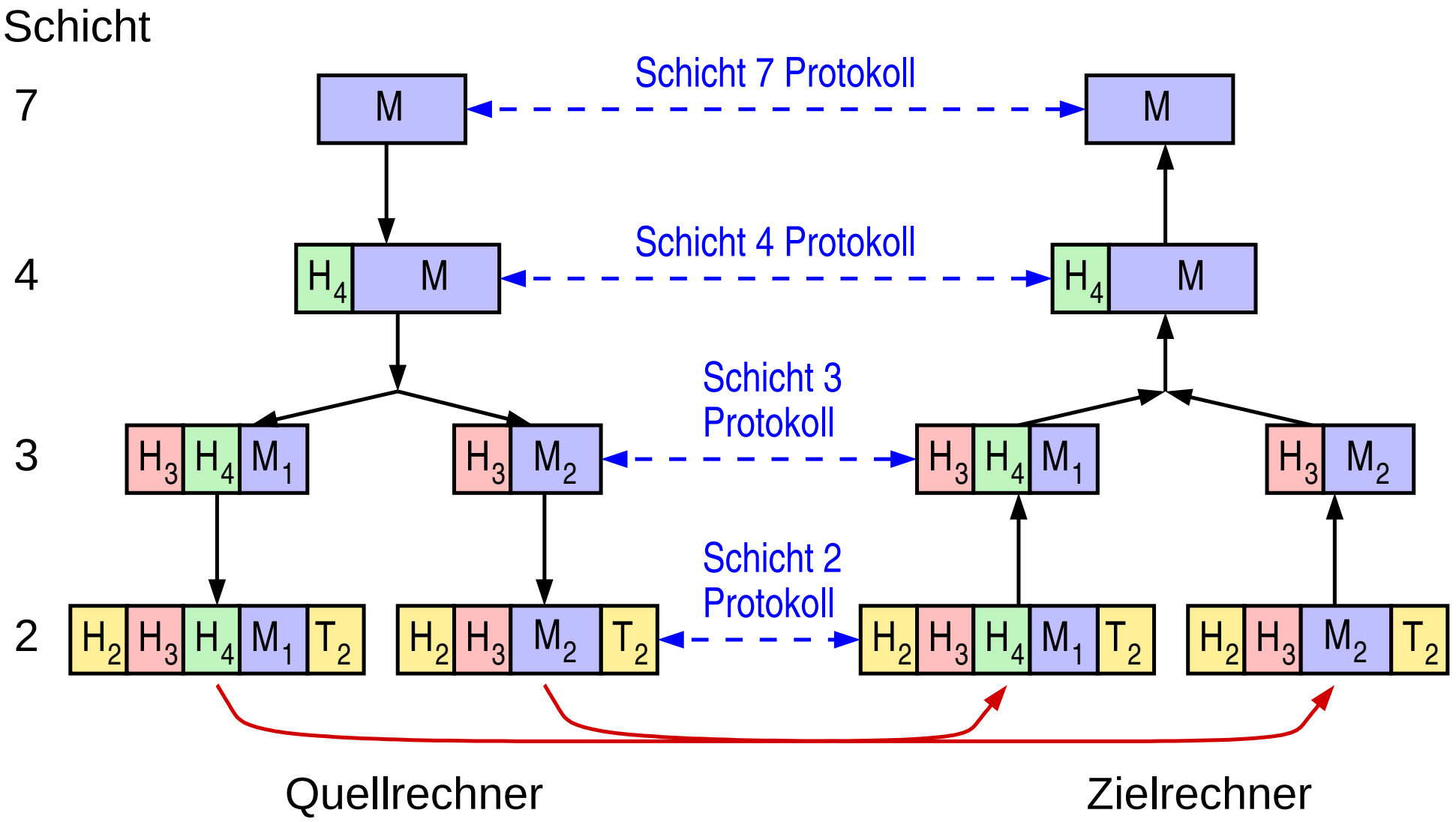
  - Z.B. Transatlantik-Kabel (3,2 Tb/s, Signallaufzeit 50 ms): $1{,}6 \cdot 10^{11}$ Bit $\approx$ 18,6 GB

Rechnernetze I

SoSe 2019

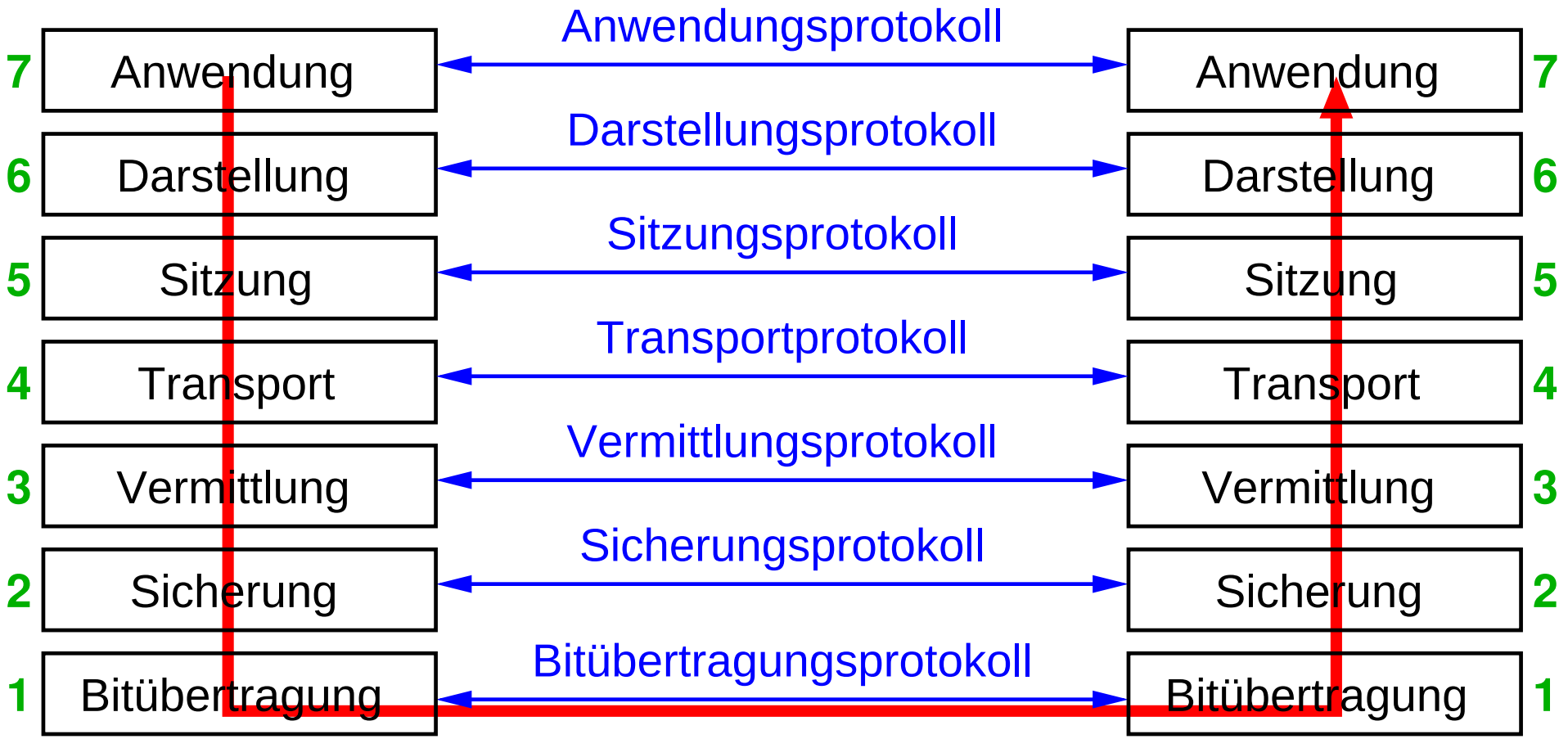
2 Protokolle und Protokollhierharchie

Beispielhafter Informationsfluß zwischen den Schichten



Das ISO/OSI Referenzmodell

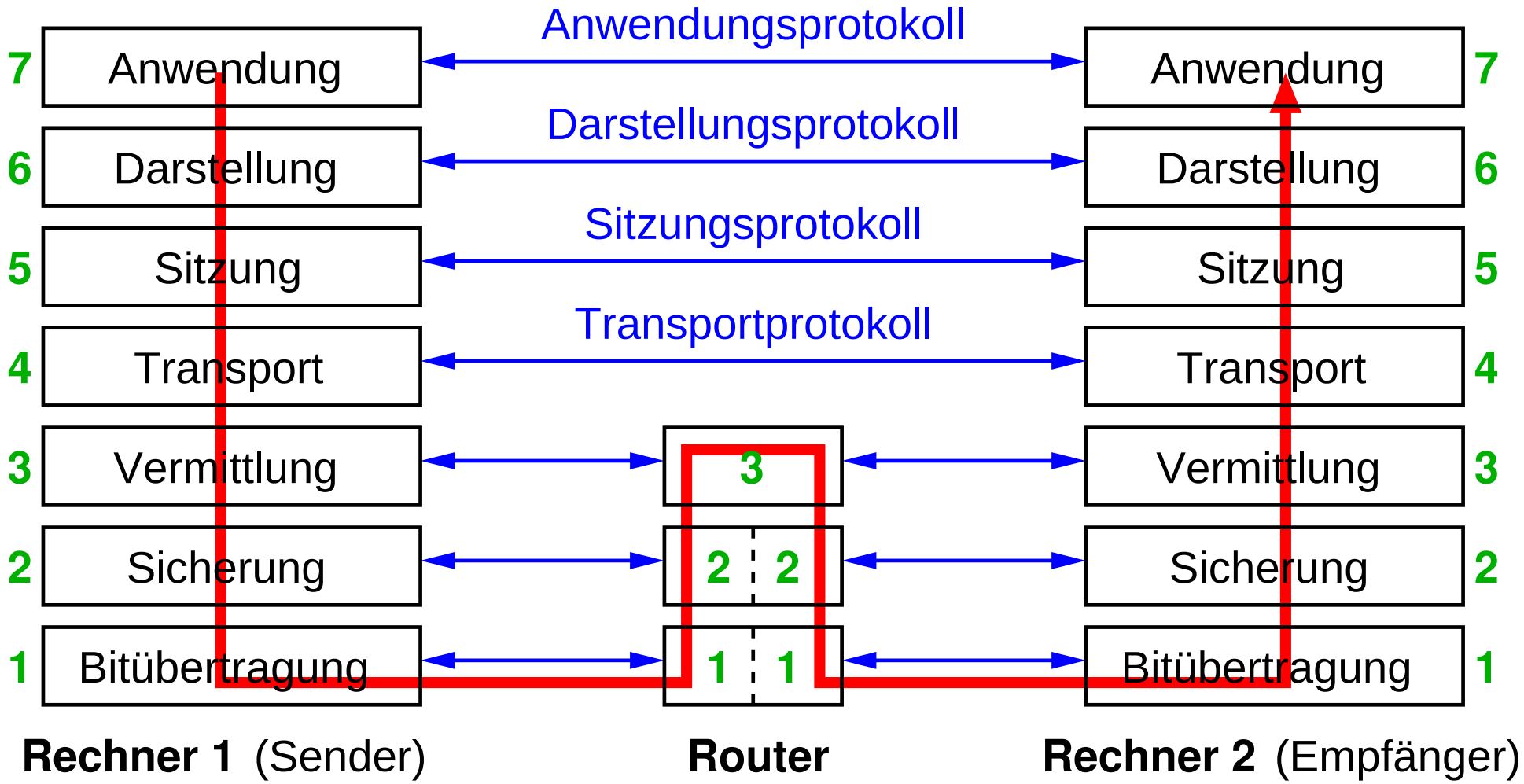
➥ OSI: *Open Systems Interconnection*



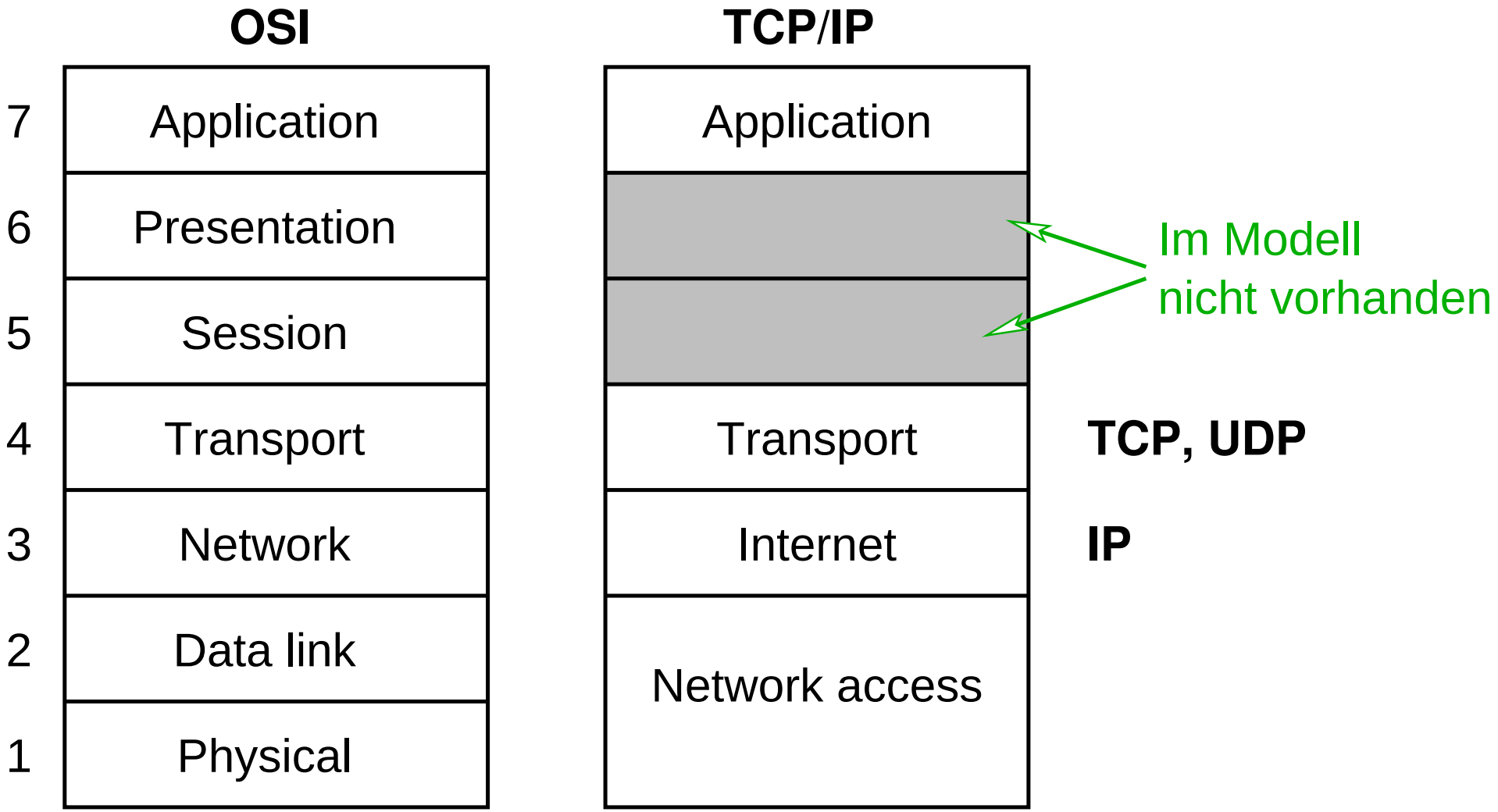
Vorbemerkung: Begriffe

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

Netzwerk-Zwischenknoten (Router) im OSI-Modell



Die Internet-Architektur im Vergleich mit OSI



Adressierung von Hosts im Internet

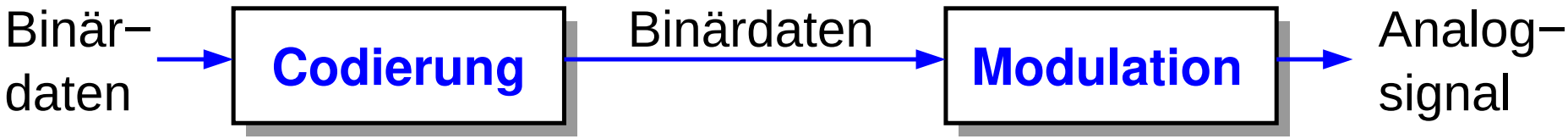
- Anwendungsschicht: **Hostname**
  - z.B. www.bs.informatik.uni-siegen.de
- Vermittlungsschicht: **IP-Adresse** (logische Adresse)
  - z.B. 141.99.179.6
- Sicherungsschicht: **MAC-Adresse** (physische Adresse)
  - z.B. 1a:68:25:f0:a3:d9

Rechnernetze I

SoSe 2019

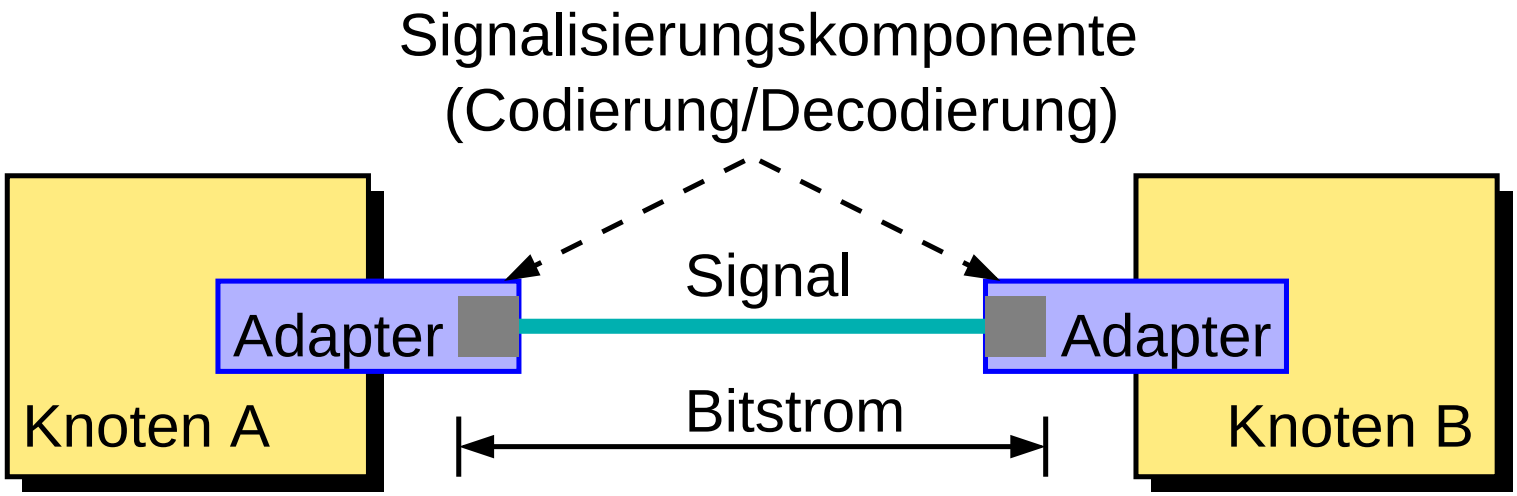
3 Direktverbindungsnetze

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



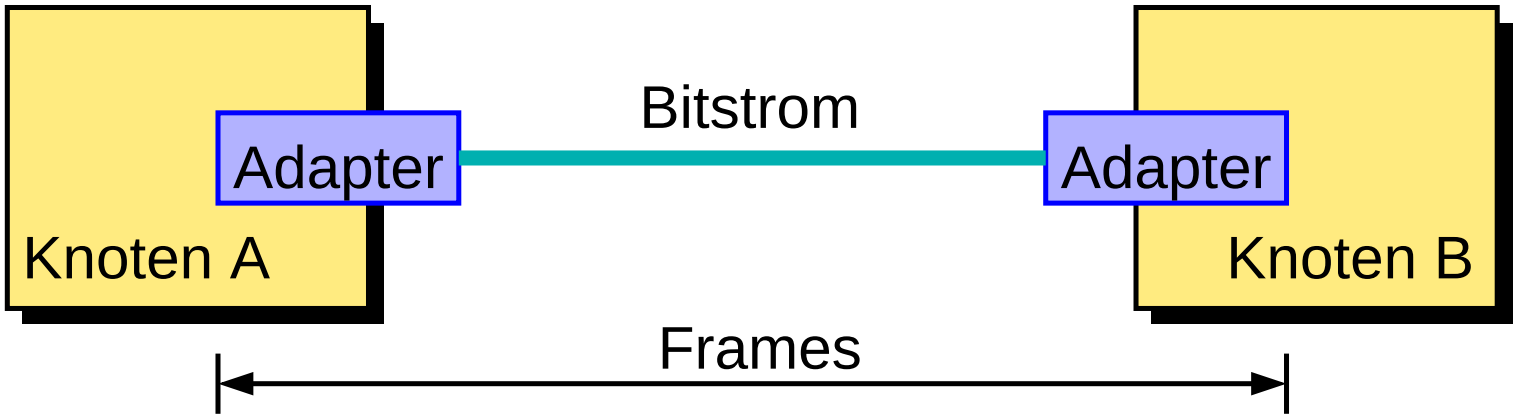
- **Modulation:**
  - Variation von Frequenz, Amplitude und/oder Phase einer Welle
  - zur Überlagerung der (Träger-)Welle mit dem Nutzsignal
    - z.B. bei Funk, Modem, Breitbandkabel, ...
  - (entfällt bei Basisband-Übertragung)

- Übertragung eines Bitstroms zwischen zwei Knoten:



- Einfachste Codierung:
  - ***Non-Return to Zero* (NRZ)**: **1** ≙ *high*, **0** ≙ *low*
- Probleme:
  - Festlegung der Schwelle zwischen *high* und *low*
  - **Taktwiederherstellung** (**Synchronisation**)
    - wo ist die „Grenze“ zwischen zwei Bits?

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



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

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

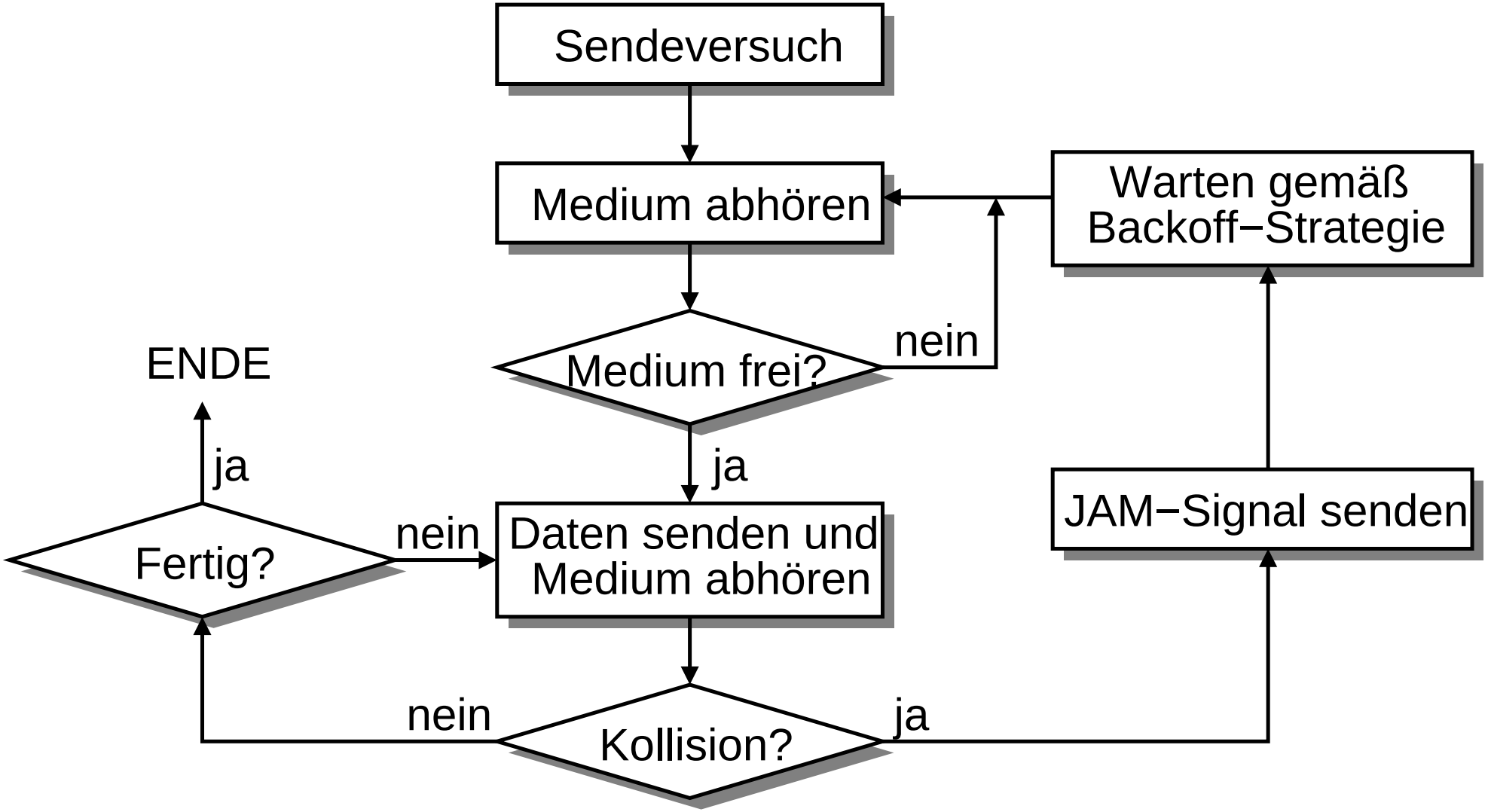
CRC (*Cyclic Redundancy Check*)

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

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

- Typische Verfahren:
  - CSMA/CD (Ethernet)
  - Tokenbasierte Verfahren (Token-Ring)
  - CSMA/CA (WLAN, CAN-Bus, ☞ **RN_II**)

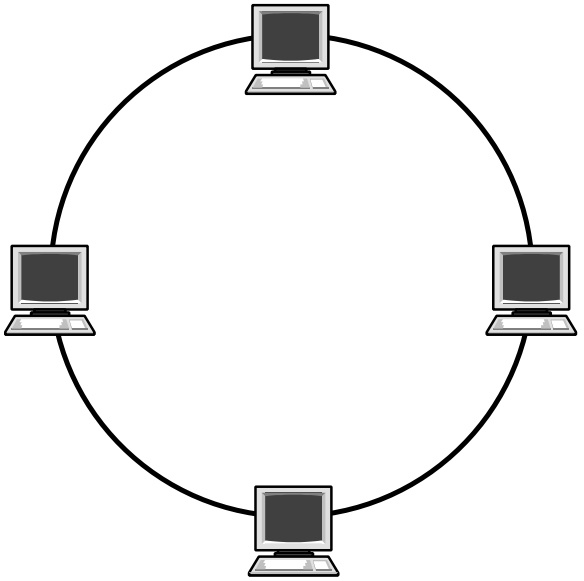
CSMA/CD – Algorithmus



Sichere Kollisionserkennung

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

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie



- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

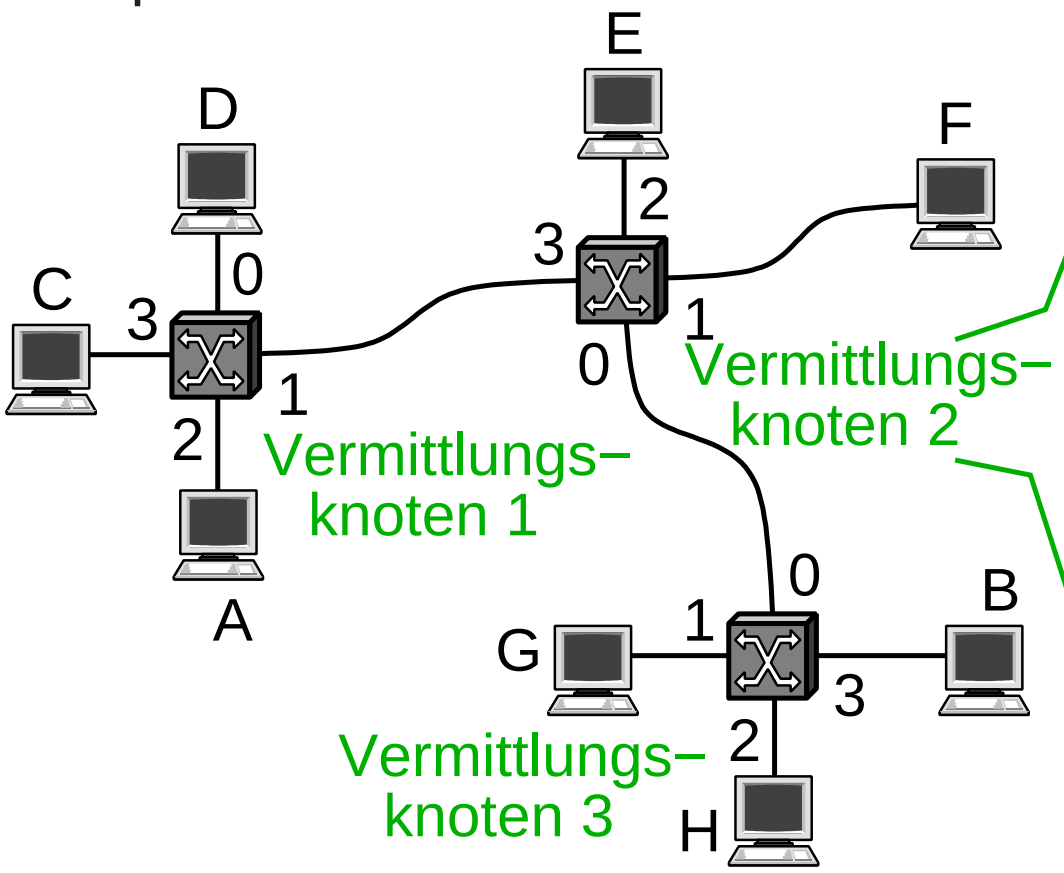
Rechnernetze I

SoSe 2019

4 LAN Switching

Weiterleitung von Datagrammen (verbindungslos)

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

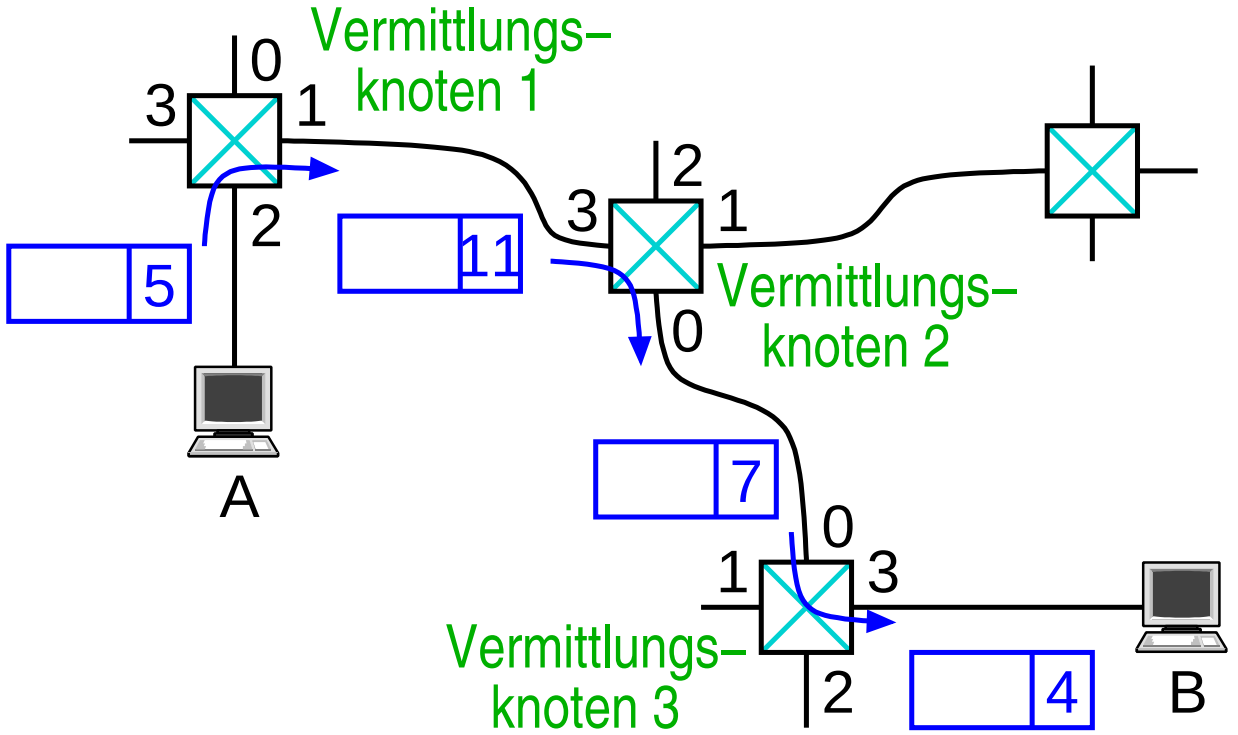


Weiterleitungstabelle

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

Virtuelle Leitungsvermittlung (verbindungsorientiert) ...

- Beispiel:
  A sendet an B

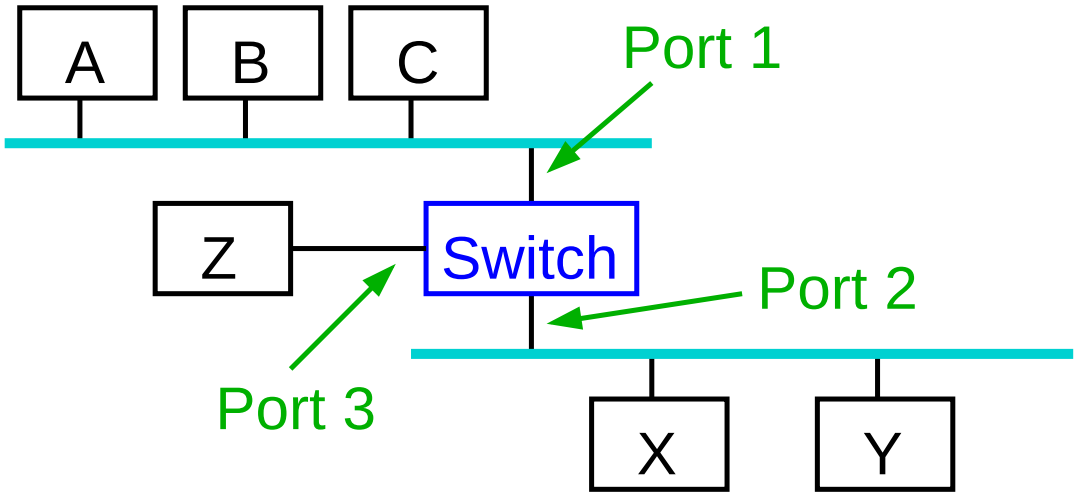


	Eingangsport	Eingangs–VCI	Ausgangsport	Ausgangs–VCI
Verm.kn. 1:	2	5	1	11
Verm.kn. 2:	3	11	0	7
Verm.kn. 3:	0	7	3	4

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

Automatisches Erstellen der Weiterleitungstabelle

- Beispiel:

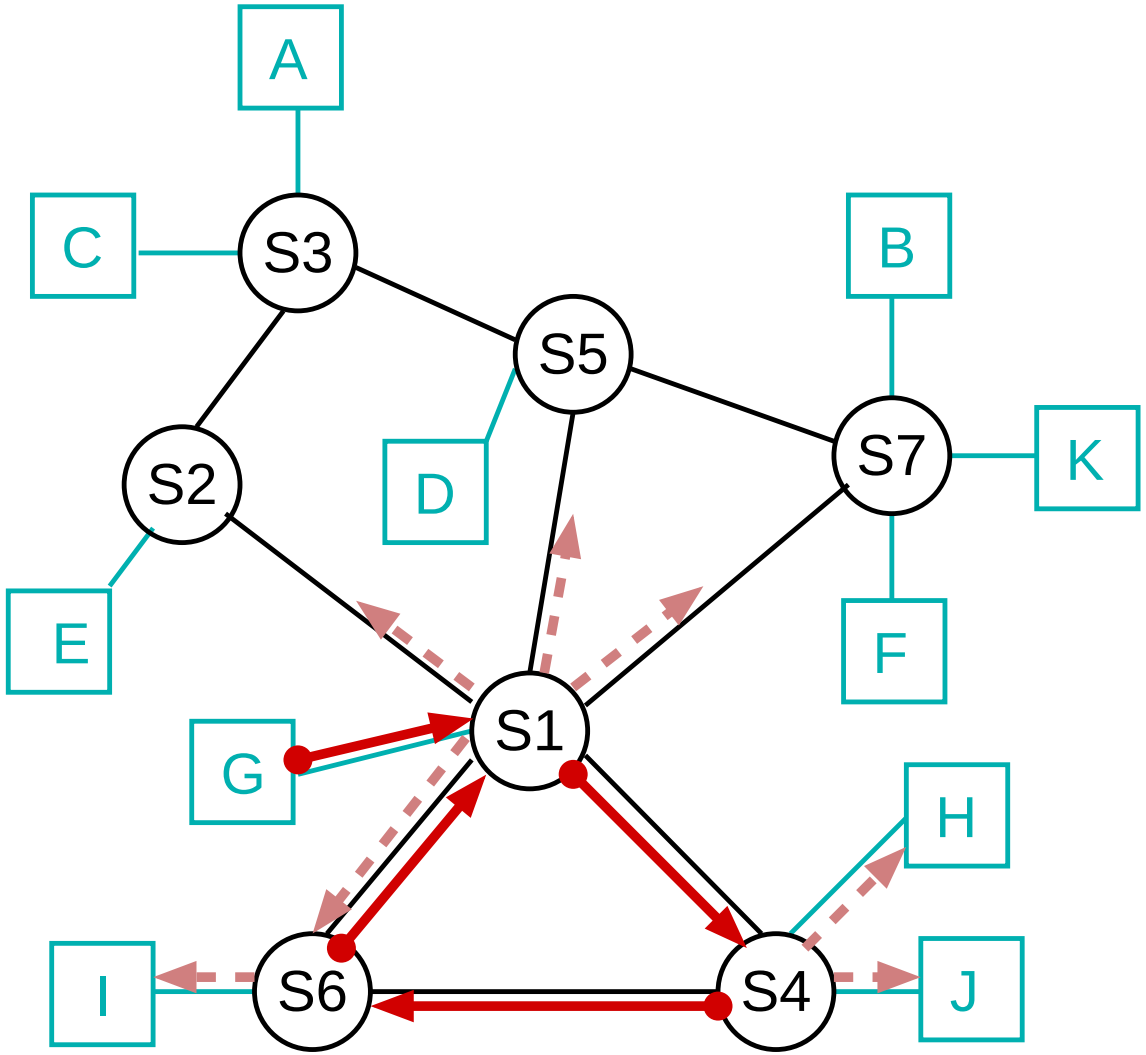


Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	3

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

Problem bei lernenden Switches: Zyklen

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



Abhilfe: *Spanning-Tree*-Algorithmus

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

Rechnernetze I

SoSe 2019

5 Internetworking

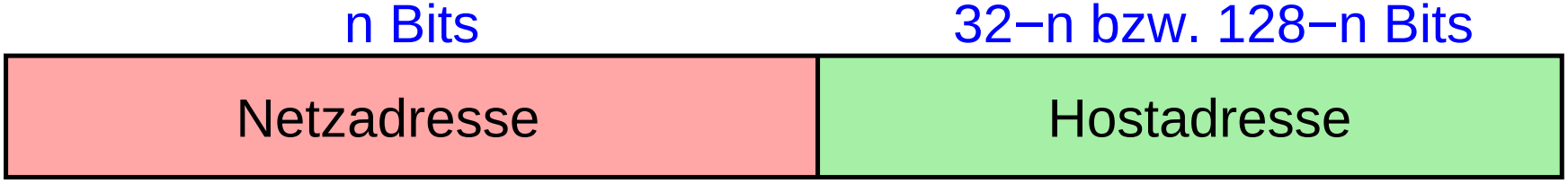
IP Dienstmodell (was bietet IP?)

- Adressierungsschema
- Datagramm-Zustellung

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

Aufgaben bei der Adressierung

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



- Identifikation von Netzen
  - durch Netzadresse (Host-Bits = 0) und Präfixlänge n
  - in IPv4 ursprünglich:
    - n geht aus IP-Adresse eindeutig hervor (Adressklassen)
  - heute in IPv4 und IPv6: explizite Angabe von n (klassenlose Adressierung, CIDR)

Routing-Tabelle (Weiterleitungstabelle)

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

$Netzadresse_1$	$Präfixlänge_1$	$Next\ Hop_1$
$Netzadresse_2$	$Präfixlänge_2$	$Next\ Hop_2$
...	...	...

- Netzadresse und Präfixlänge zusammen identifizieren ein Netz
  - bei IPv4 statt Präfixlänge ggf. auch Subnetzmaske (☞ **S. 145**)
- *Next Hop* = Router bzw. Schnittstelle, an den/die das Paket weitergegeben werden soll, falls Ziel im angegebenen Netz liegt

Vorgehensweise bei der Weiterleitung

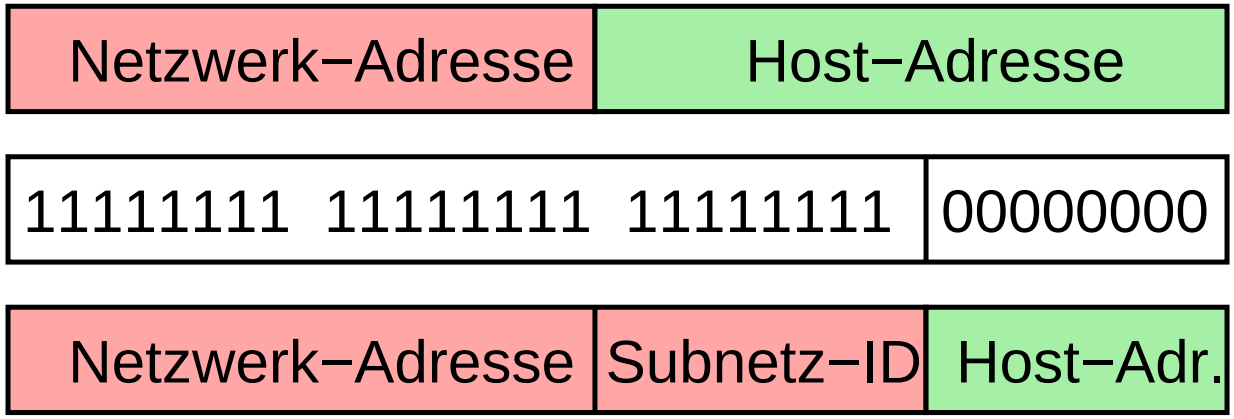
- Algorithmus:
  - Suche Eintrag i mit größter *Präfixlänge*$_i$, für den gilt:
    - Zieladresse und *Netzadresse*$_i$ stimmen in den ersten *Präfixlänge*$_i$ Bits überein
    - d.h. Zieladresse liegt in dem durch *Netzadresse*$_i$ und *Präfixlänge*$_i$ gegebenen Netz
  - Falls Eintrag gefunden: Weiterleiten an *NextHop*$_i$
  - Sonst: Verwerfen des Pakets
- Typisch: zusätzlicher Tabellen-Eintrag für Default-Route

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

  - dieser Eintrag „paßt“ auf jede Zieladresse

Bildung von Subnetzen

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



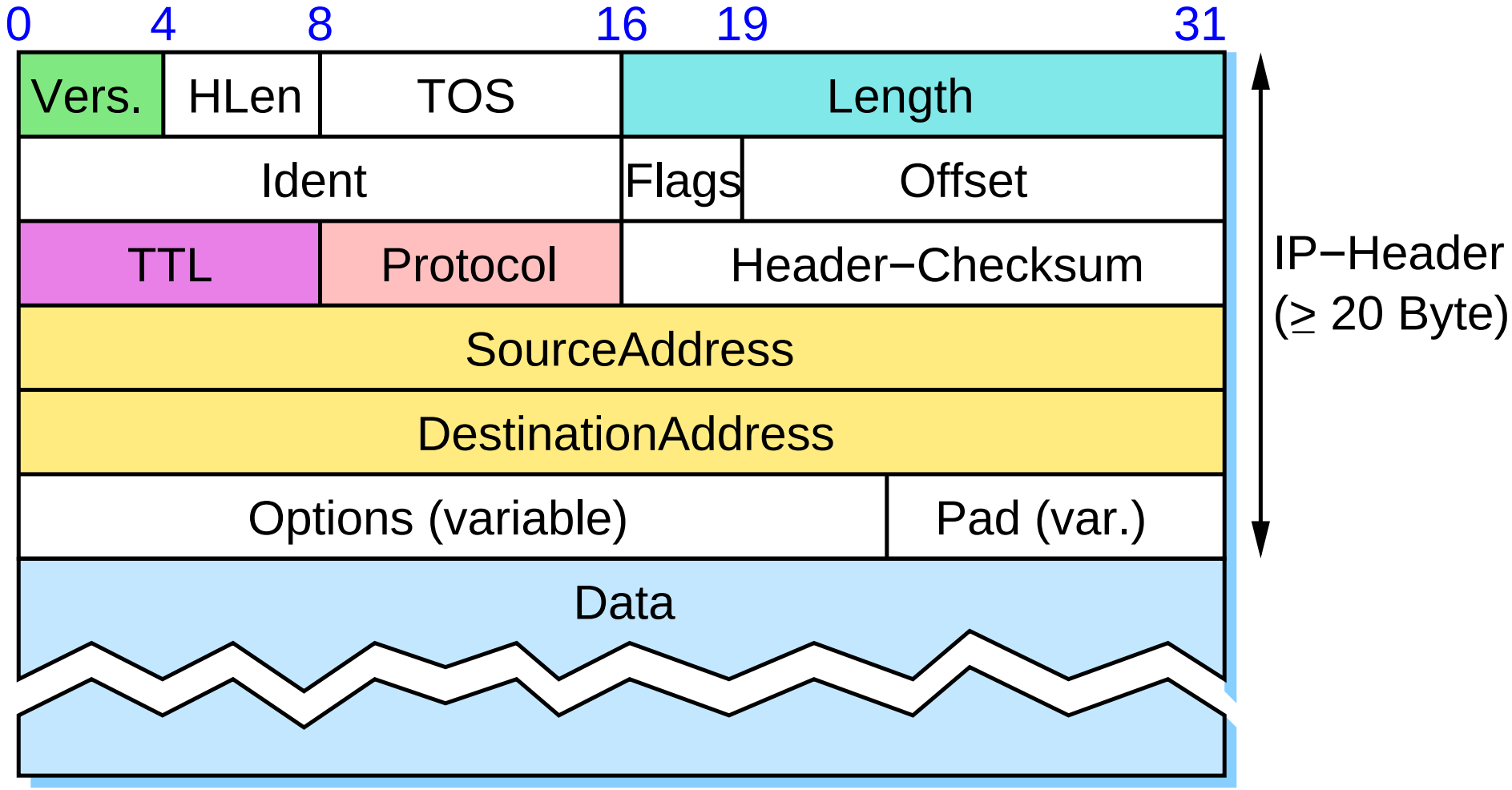
  - i.a. werden Subnetze unterschiedlicher Größe erzeugt
  - Subnetzmaske legt Präfixlänge für das Subnetz fest
- IPv6: Subnetz-ID in eigenem 16-Bit-Feld (☞ **S. 136**)

Bestimmung der (Sub-)Netzzugehörigkeit

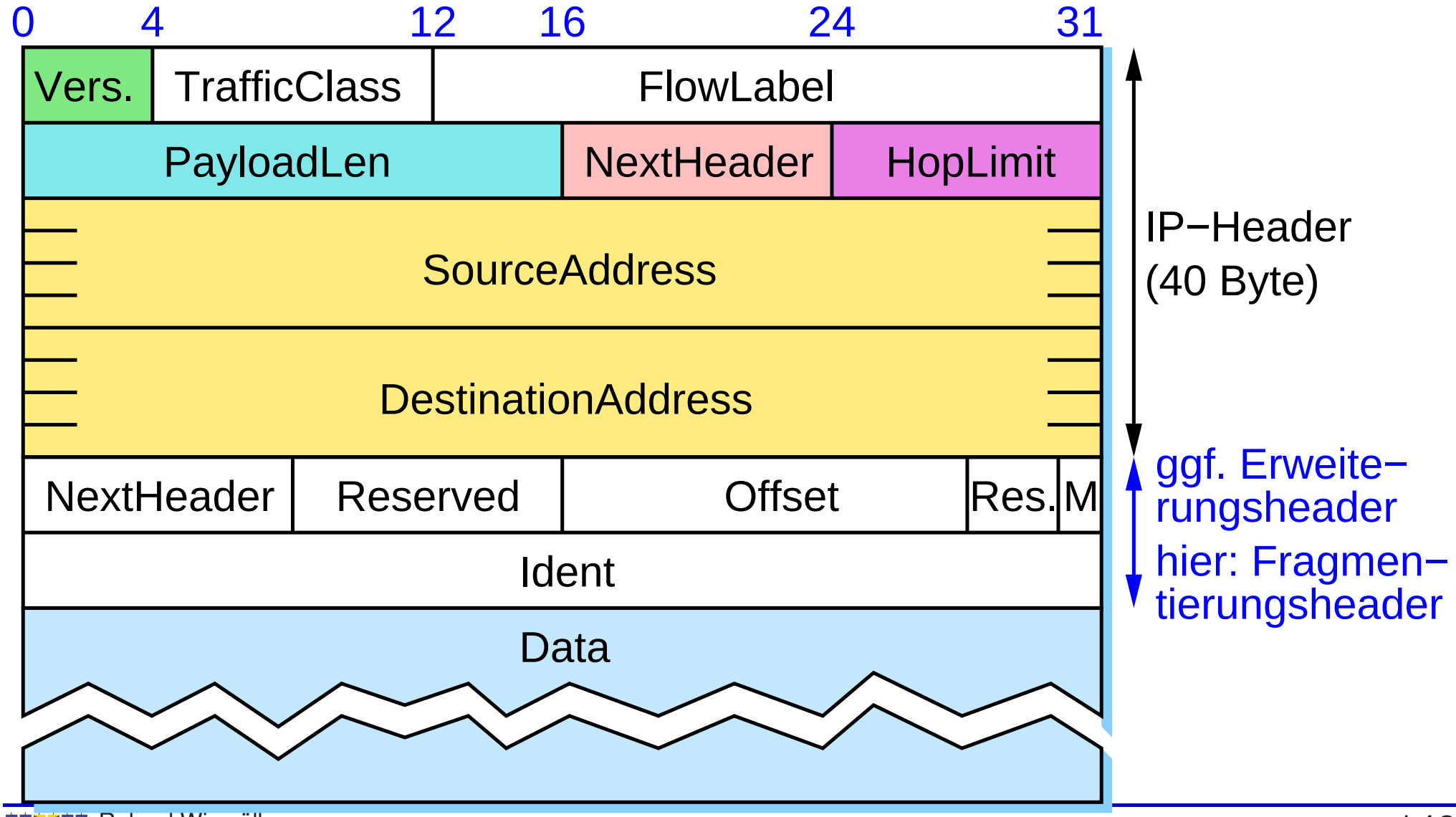
- Wie bestimmt ein Host bzw. Router, ob eine IP-Adresse xyz in einem gegebenen (Sub-)Netz liegt?
- Gegeben: (Sub-)Netzadresse und Präfixlänge n
  - stimmen xyz und Netzadresse in den ersten n Bits überein?
  - (vgl. Weiterleitungsalgorithmus auf S. 143)
- Gegeben: (Sub-)Netzadresse und (Sub-)Netzmaske
  - gilt xyz AND Netzmaske $=$ Netzadresse ?

- Beispiel: 128.96.34.19 AND 255.255.255.128 $=$ 128.96.34.0

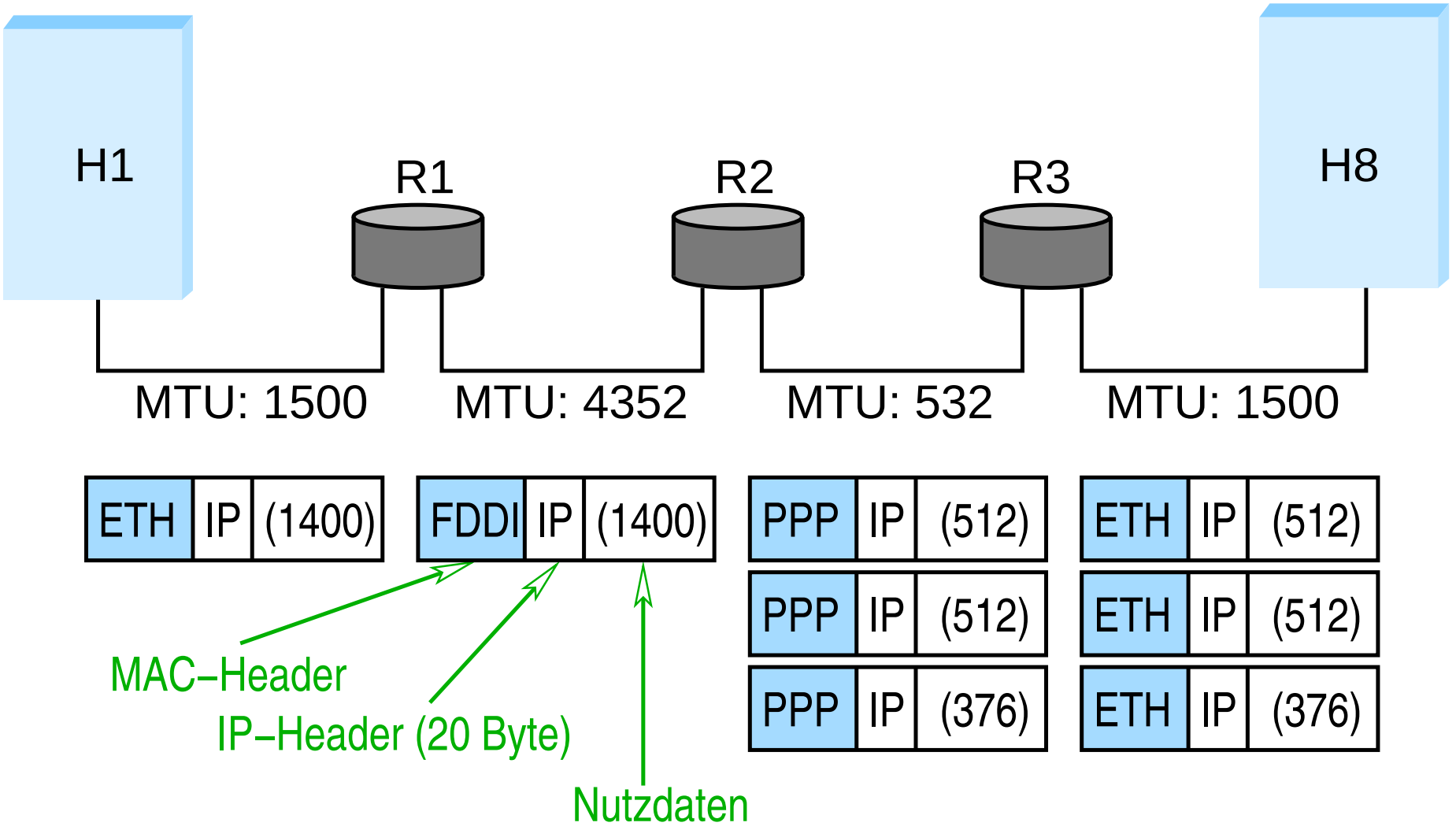
IPv4


0
4
8
16
19
31
Vers.
HLen
TOS
Length
Ident
Flags
Offset
TTL
Protocol
Header−Checksum
SourceAddress
DestinationAddress
Options (variable)
Pad (var.)
Data
IP−Header
(≥ 20 Byte)

IPv6


0
4
12
16
24
31
Vers.
TrafficClass
FlowLabel
PayloadLen
NextHeader
HopLimit
SourceAddress
DestinationAddress
NextHeader
Reserved
Offset
Res.
M
Ident
Data
IP−Header
(40 Byte)
ggf. Erweite−
rungsheader
hier: Fragmen−
tierungsheader

Beispiel zur Fragmentierung (IPv4)



ICMP: *Internet Control Message Protocol*

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

ARP: *Address Resolution Protocol* (IPv4)

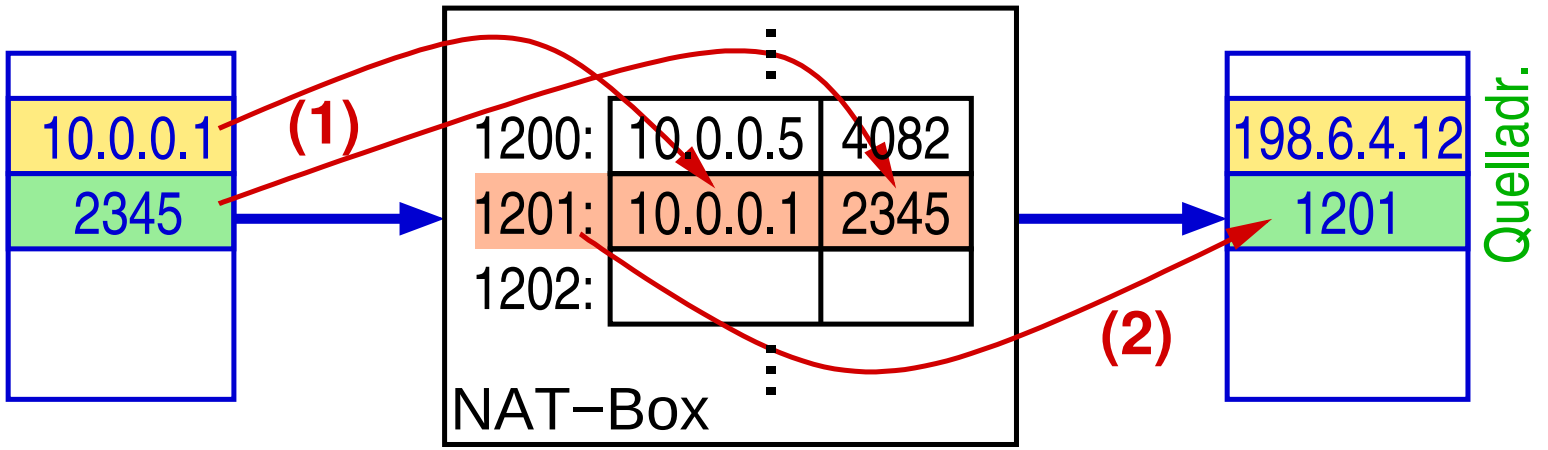
- Annahme: ein Rechner H will ein Paket an IP-Adresse *xyz* senden, *xyz* ist im lokalen Netz
- H sucht in seinem ARP-Cache nach der zu *xyz* gehörigen MAC-Adresse
- Falls gefunden: Paket an diese MAC-Adresse senden
- Sonst:
  - H sendet Anfrage (*ARP-Request*) per Broadcast in das LAN: „wer hat IP-Adresse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

DHCP: *Dynamic Host Configuration Protocol*

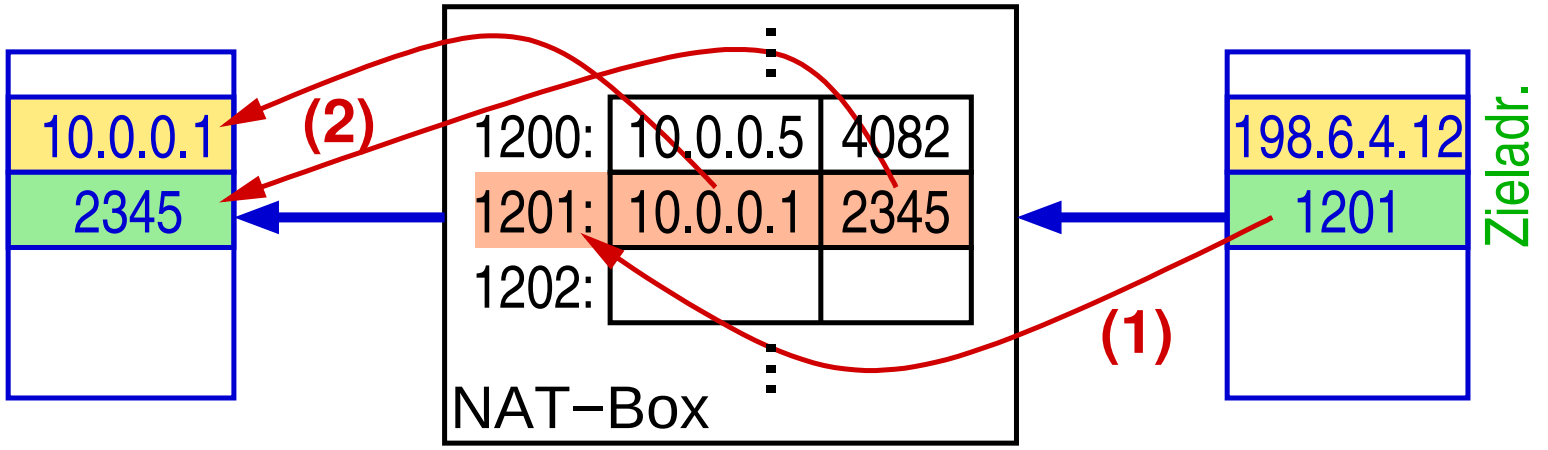
- Automatisiert u.a. die Vergabe von IP-Adressen (IPv4 und IPv6)
- Vorgehensweise:
  1. Rechner sendet Broadcast-Anfrage (DHCPDISCOVER)
     - Verbreitung nur im lokalen Netz
  2. DHCP-Server sendet DHCPOFFER: Angebot für IP-Adresse
     - Zuordnung statisch oder dynamisch, als Schlüssel dient MAC-Adresse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

Genaue Funktionsweise von NAT

A) Paket wird gesendet



B) Antwort−paket kommt an



IP−Adresse im IP−Header Port im TCP/UDP−Header

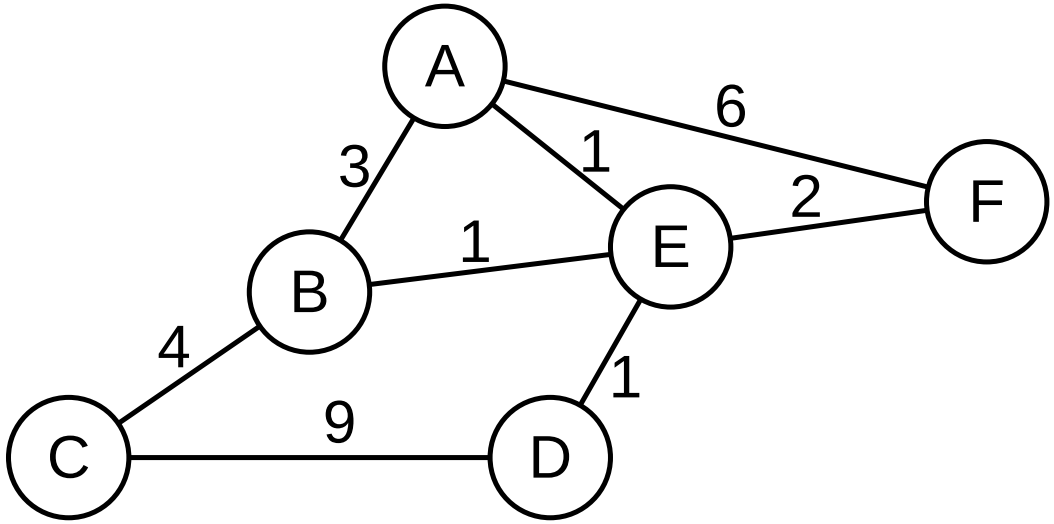
Rechnernetze I

SoSe 2019

6 Routing

Routing als Graph-Problem

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



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

Vorgehensweise

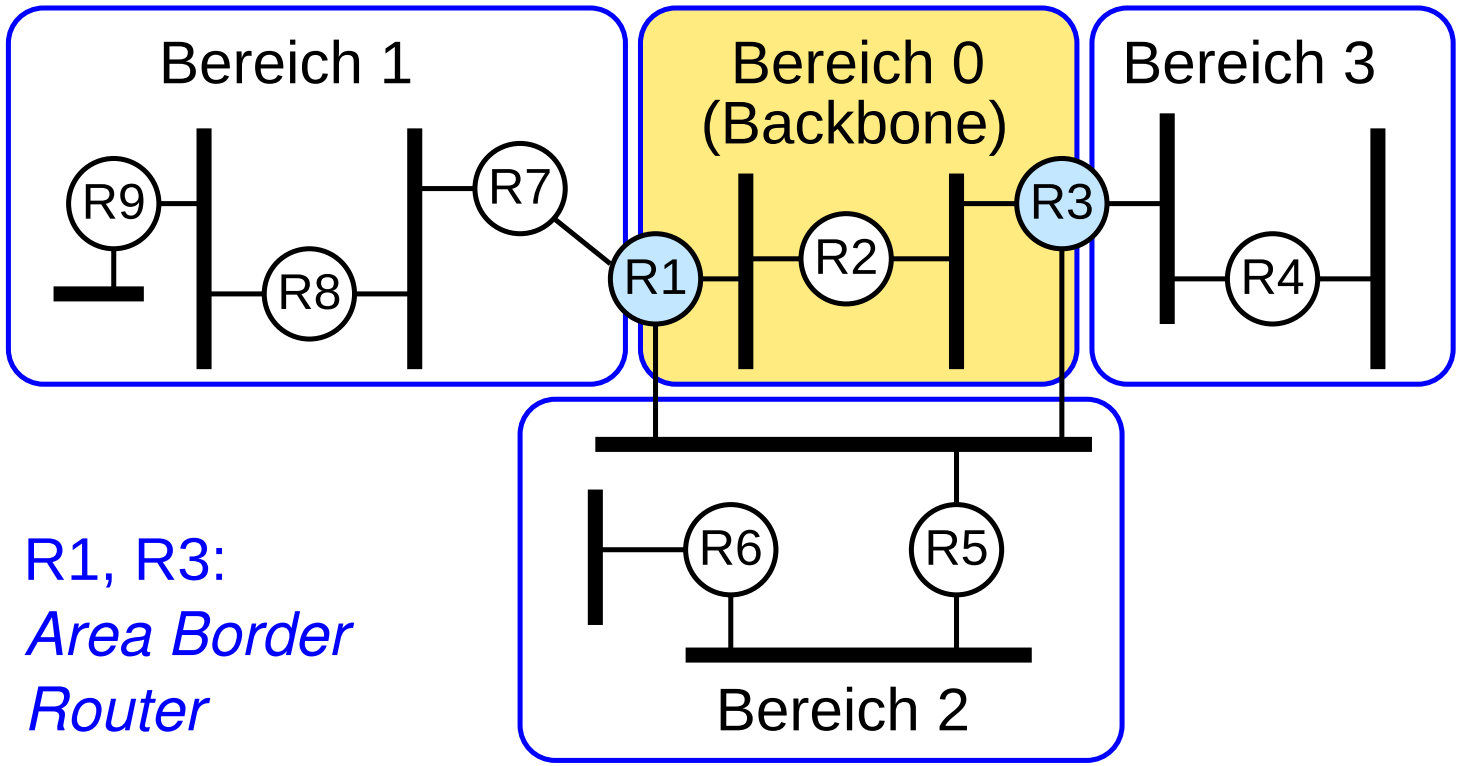
- Nachrichtenaustausch zur Erstellung der Routing-Tabellen:
  - Router senden ihren Distanzvektor an alle direkten Nachbarn
  - bessere Routen werden in den eigenen Distanzvektor (und die Routing-Tabelle) übernommen
    - aber auch schlechtere vom Next Hop
- Nach mehreren Runden des Nachrichtenaustauschs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

6.2.2 *Link State Routing*

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

Multi-Area OSPF

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



Routing mit BGP

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

Rechnernetze I

SoSe 2019

7 Ende-zu-Ende Protokolle

7.1 Ports: Adressierung von Prozessen

- Wie identifiziert man Prozesse?
  - **Nicht** durch die Prozeß-ID des Betriebssystems:
    - systemabhängig, „zufällig“
  - **Sondern**: indirekte Adresierung über Ports
    - 16-Bit Nummer

- Woher weiß ein Prozeß die Port-Nummer des Partners?
  - „*well known ports*“ (i.d.R. 0...1023) für Systemdienste
    - z.B.: 80 = Web-Server, 25 = Mail-Server
    - Analogie: Tel. 112 = Feuerwehr
  - Server kennt die Port-Nummer des Clients aus UDP- bzw. TCP-Header der Anfrage

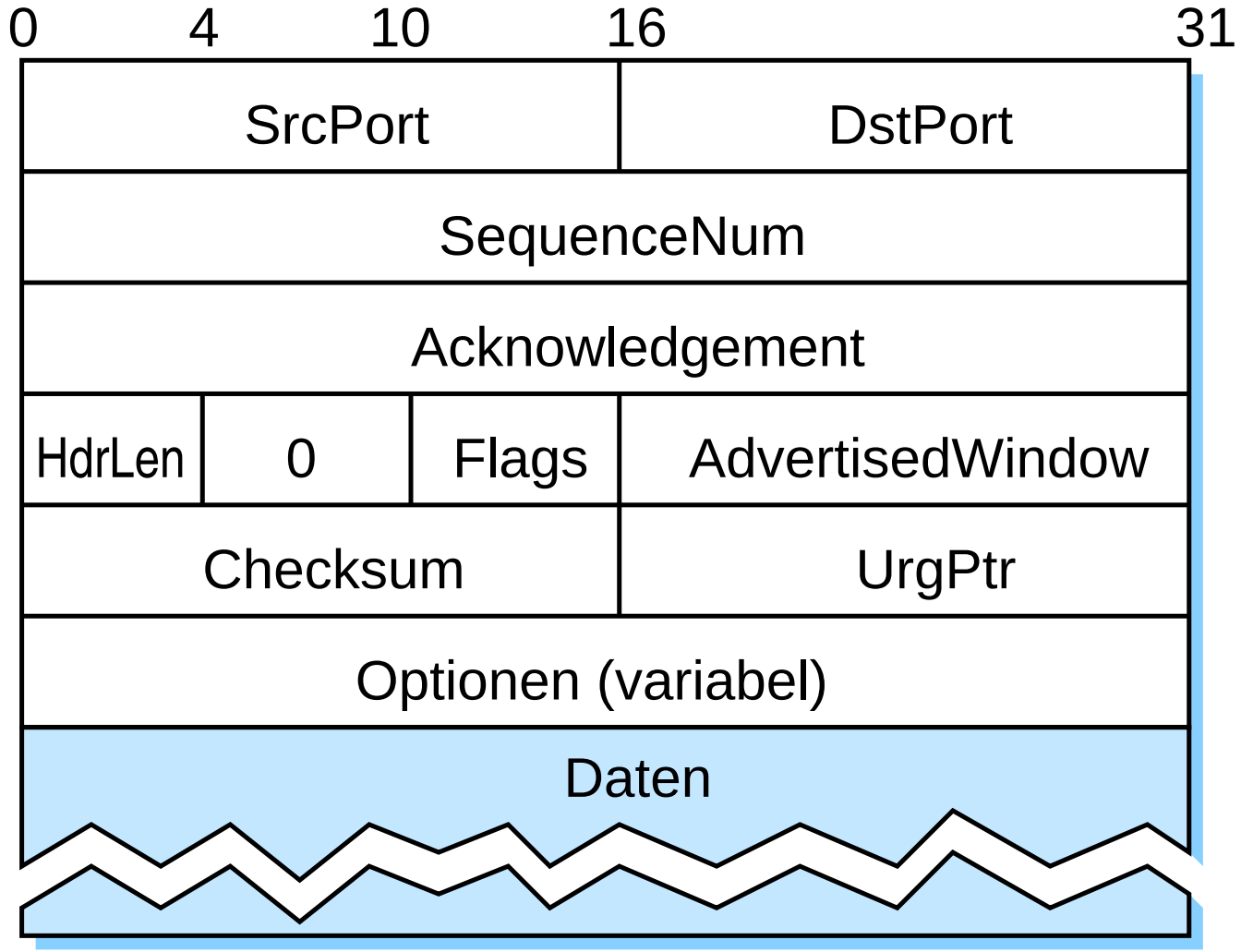
UDP: *User Datagram Protocol*

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

TCP: *Transmission Control Protocol*

- Dienstemodell von TCP:
  - zuverlässige Übertragung von Daten**strömen** zw. Prozessen
    - verbindungsorientiert
- Meist verwendetes Internet-Protokoll
  - befreit Anwendungen von Sicherung der Übertragung
- TCP realisiert:
  - Port-Demultiplexing (☞ **7.1**)
  - Vollduplex-Verbindungen
  - Flußkontrolle
  - Überlastkontrolle

Aufbau eines TCP Segments



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

7.3.3 Verbindungsauf- und -abbau

Verbindungsaufbau

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

Verbindungsabbau

- Symmetrisch:
  - beide Seiten müssen die Verbindung schließen

Problem:

- Bei der Übertragung eines Segments bzw. Frames können Fehler auftreten
  - Empfänger kann Fehler erkennen, aber i.a. nicht korrigieren
  - Segmente bzw. Frames können auch ganz verloren gehen
    - z.B. durch überlasteten Router bzw. Switch
    - oder bei Verlust der Frame-Synchronisation (☞ **3.4**)
- Segmente bzw. Frames* müssen deshalb ggf. neu übertragen werden

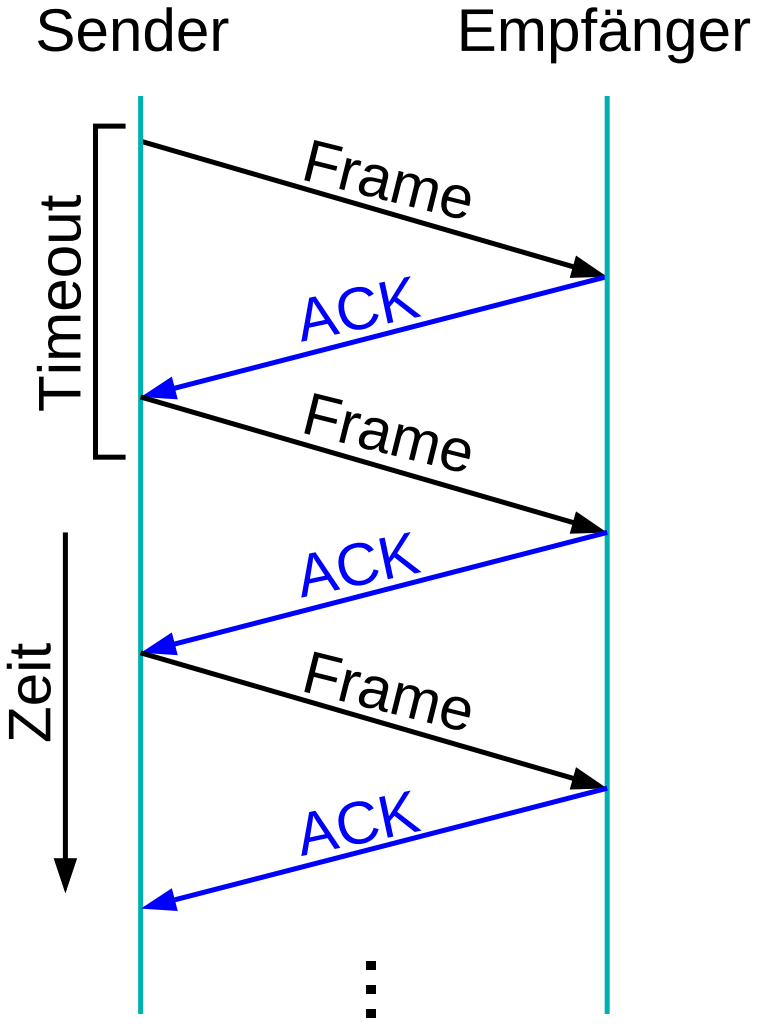
* Zur Vereinfachung wird im Folgenden nur der Begriff „Frame“ verwendet!

Basismechanismen zur Lösung:

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

Ablauf bei fehlerfreier Übertragung

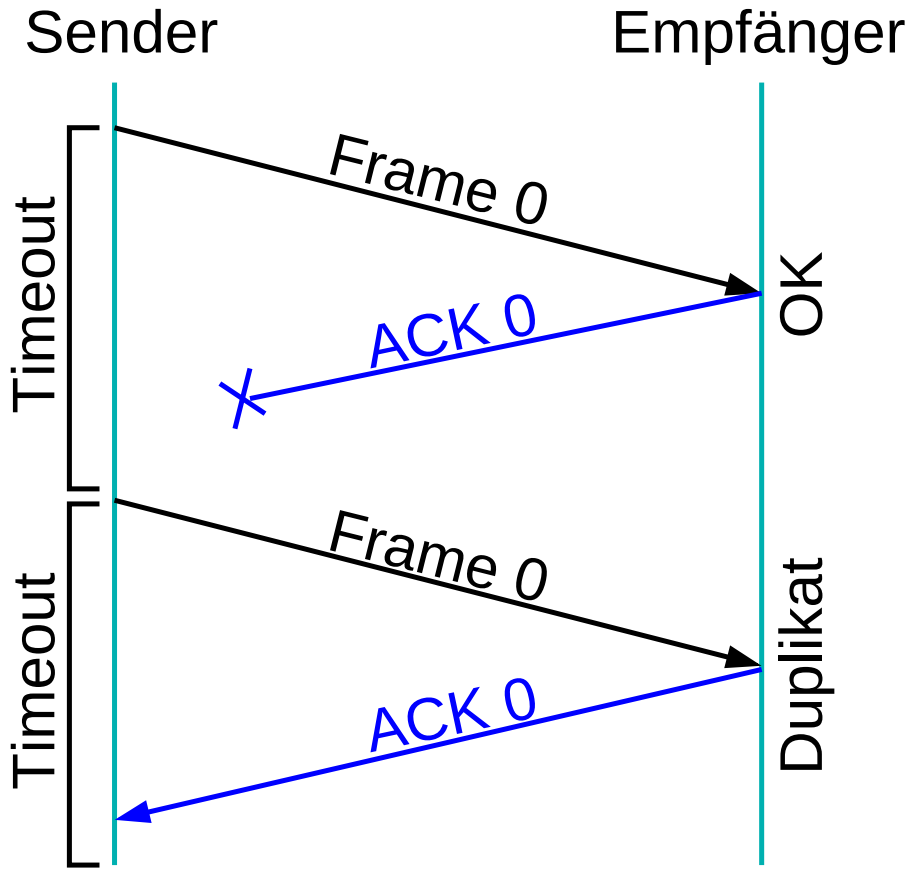
- Sender wartet nach der Übertragung eines Frames, bis ACK eintrifft
- Erst danach wird der nächste Frame gesendet



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

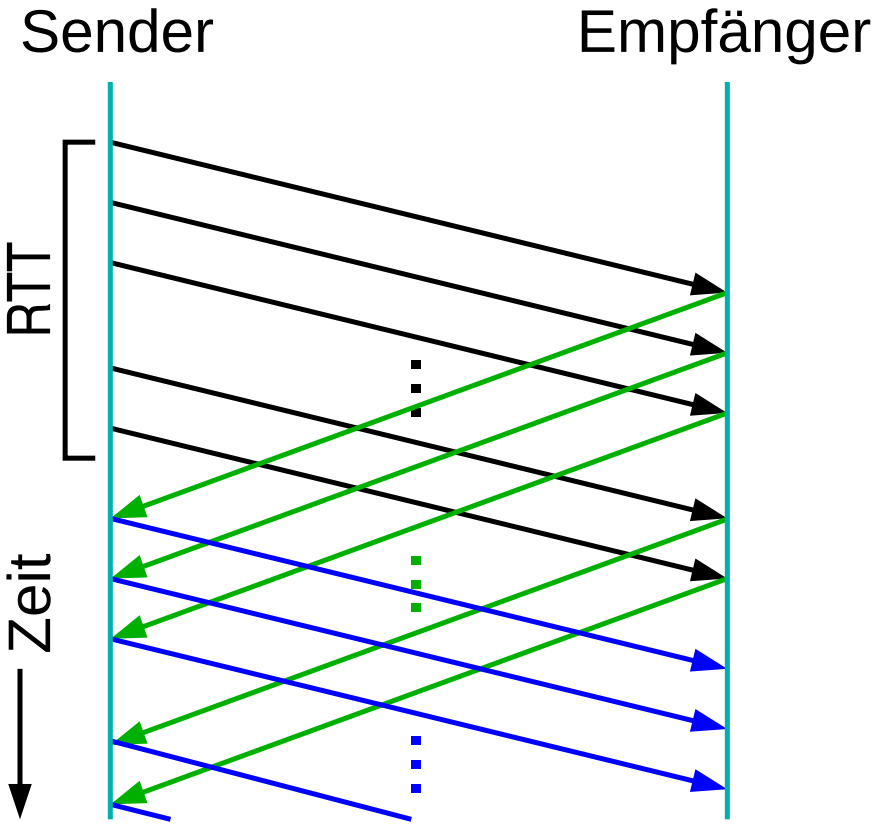
- Der Empfänger erhält den Frame mehrfach
- Er muß dies erkennen können!

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



Motivation

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

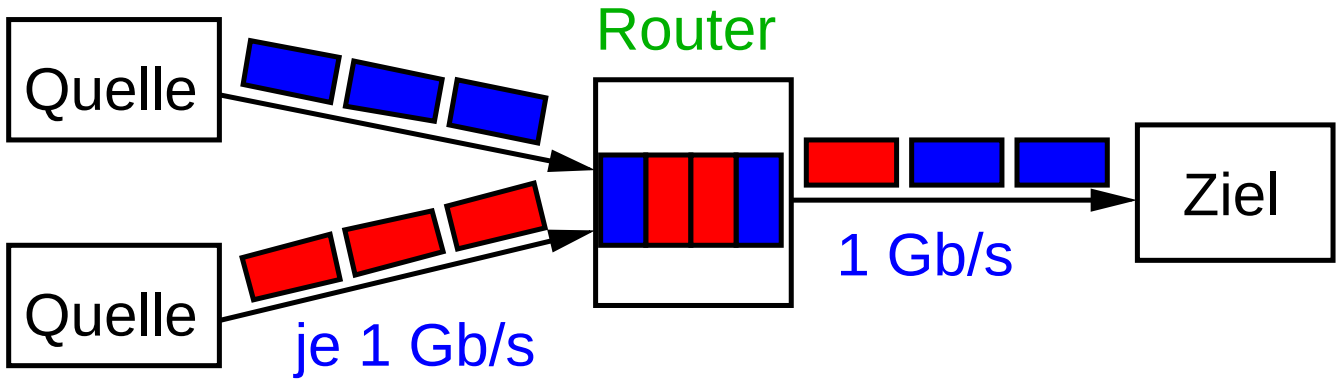


Aufgaben des Sliding-Window-Algorithmus in TCP

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

Überlastkontrolle

- **Flußkontrolle** verhindert, daß **ein Sender** seinen **Empfänger** überlastet
- **Überlastkontrolle** verhindert, daß **mehrere Sender** einen Teil des **Netzwerks** überlasten (durch Konkurrenz um Bandbreite):



  - bei unzureichender Bandbreite: Puffern der Pakete im Router
  - bei Pufferüberlauf: Router muß Pakete verwerfen
- Ein Netzwerk mit (häufigem) Pufferüberlauf heißt **überlastet** (*congested*)

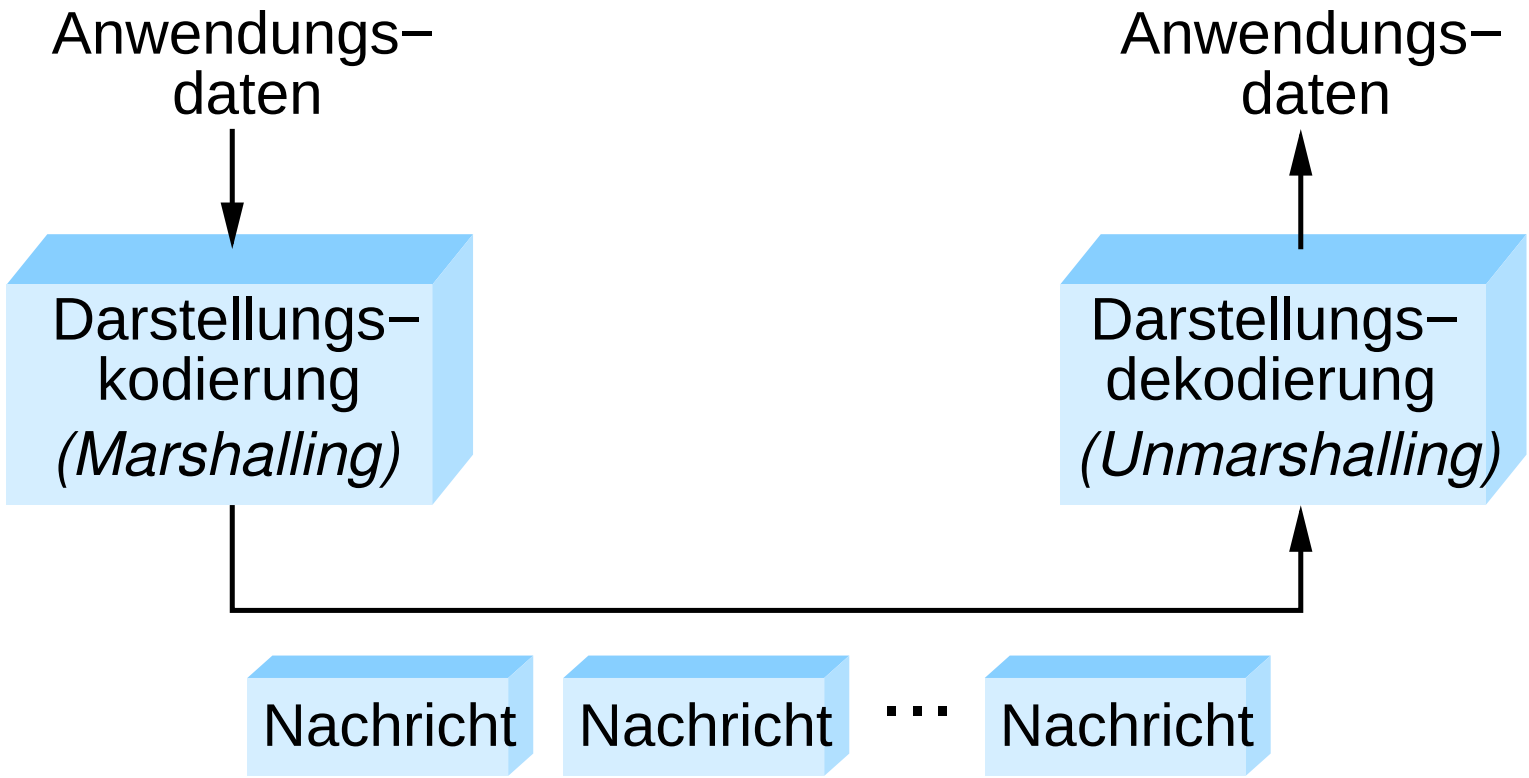
Rechnernetze I

SoSe 2019

8 Datendarstellung

8.1 Marshalling

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



Rechnernetze I

SoSe 2019

9 Anwendungsprotokolle

Protokolle für Emails

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

9.2 HTTP

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

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

- Beispiel-Anfrage:

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

Hierarchische Namensauflösung mit DNS

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

Rechnernetze I

SoSe 2019

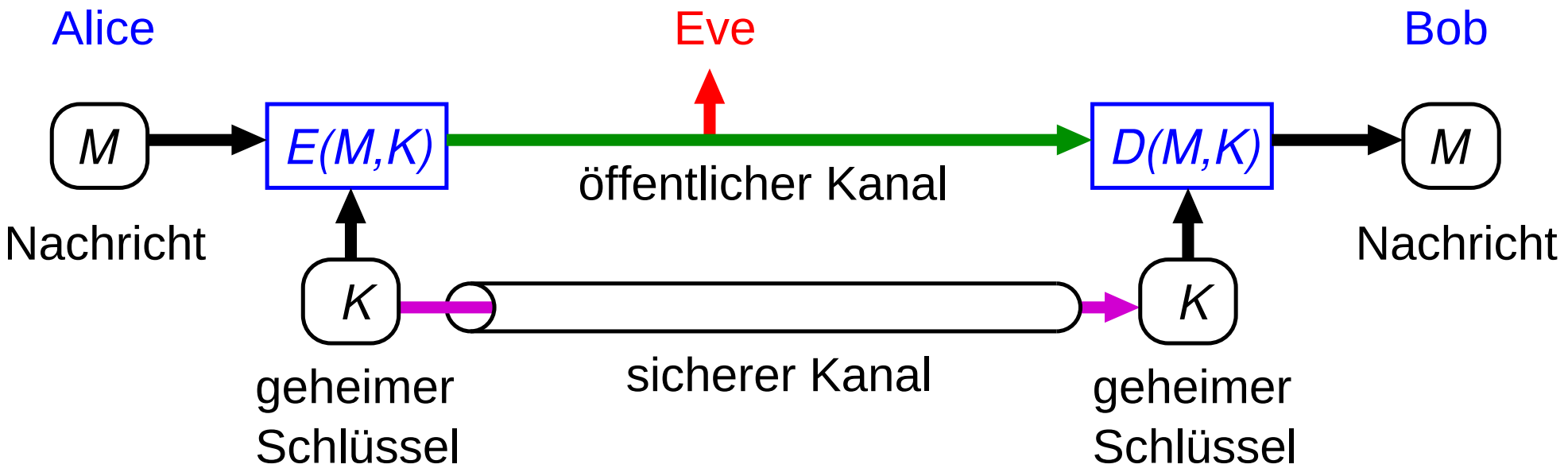
10 Netzwerksicherheit

Konkret: Alice sendet eine Nachricht an Bob

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

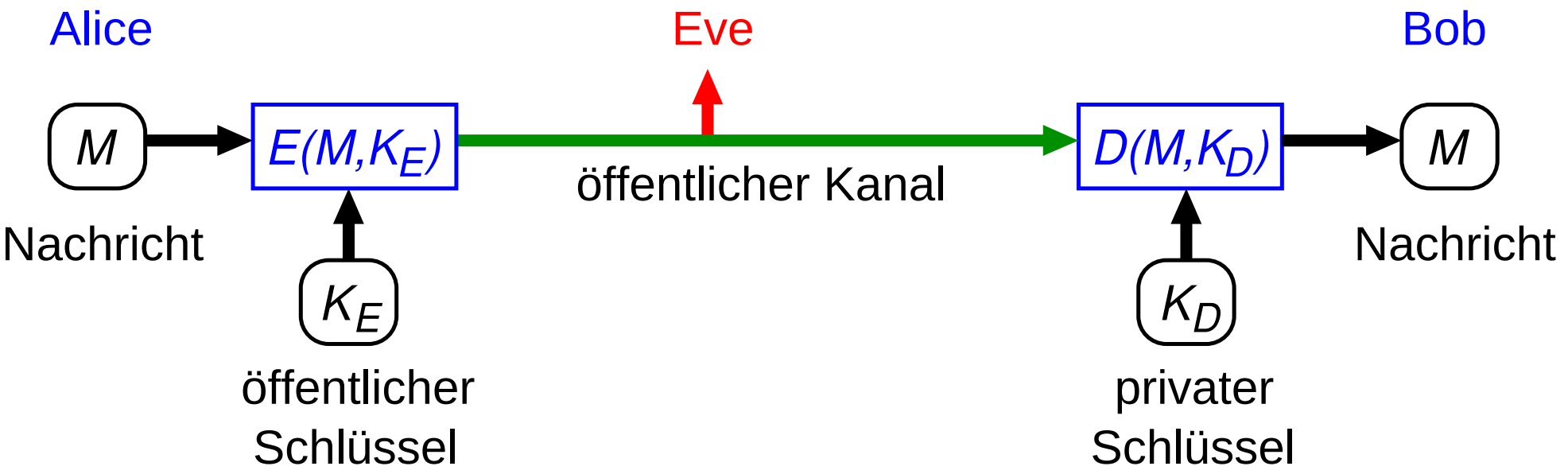
- Im Folgenden: Beschränkung auf diese vier Anforderungen

Symmetrische Verschlüsselung:



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

Asymmetrische Verschlüsselung:



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

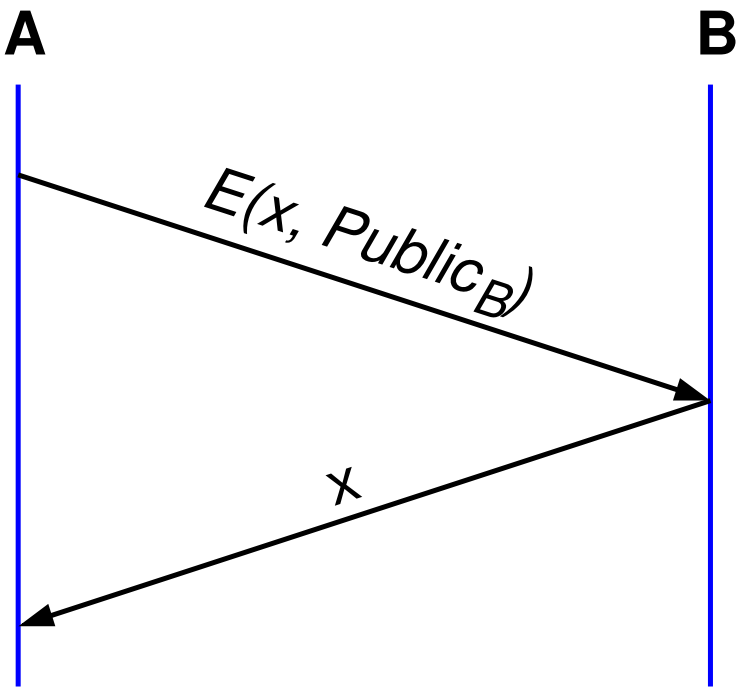
Kryptographische Hashes (*Message Digest*)

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

Was leistet die reine Verschlüsselung von Nachrichten?

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

Partner-Authentifizierung über asymmetrische Chiffre



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

Sicherung der Nachrichtenintegrität und -authentizität

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

Sicherung der Nachrichtenintegrität und -authentizität ...

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

Digitale Signatur mit asymmetrischer Chiffre

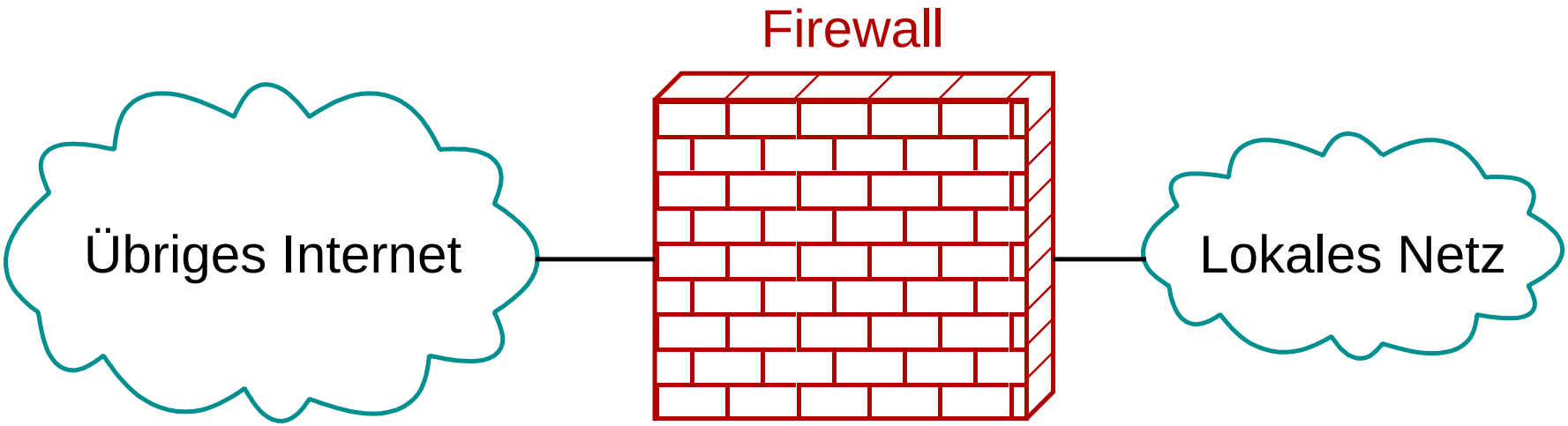
- Sender *A* sendet *M* und $E(M, Private_A)$ an Empfänger *B*
- *B* entschlüsselt mit $Public_A$ und prüft, ob Ergebnis gleich *M* ist
- Problem: asymmetrische Verschlüsselung ist langsam
- Daher: Kombination mit kryptographisher Hashfunktion
  - digitale Signatur von *A* auf *M* dann: $E(H(M), Private_A)$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

Zertifikat

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

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

10.6 Firewalls



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