Rechnernetze II

SoSe 2025

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

Stand: 8. April 2025

Rechnernetze II

SoSe 2025

0 Organisation

Zu meiner Person

- Studium der Informatik an der Techn. Univ. München
  - dort 1994 promoviert, 2001 habilitiert
- Seit 2004 Prof. für Betriebssysteme und verteilte Systeme an der Univ. Siegen
- **Forschung:** Sichere komponentenbasierte Systeme; parallele und verteilte Systeme
- Vorsitzender des Prüfungsausschusses Informatik

- **e-mail:** roland.wismueller@uni-siegen.de
- **Tel.:** 0271/740-4050
- **Büro:** H-B 8404

- **Sprechstunde:** Mo. 14:15-15:15 Uhr

Zur Fachgruppe „Betriebssysteme / verteilte Systeme“

Andreas Hoffmann

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

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

Felix Breitweiser

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

- Betriebssysteme
- Programmiersprachen
- Virtuelle Maschinen

Sven Jacobs

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

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

Lehrangebot

Vorlesungen/Praktika

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

Lehrangebot ...

Projektgruppen

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

Abschlussarbeiten (Bachelor, Master)

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

Seminare

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

Zur Vorlesung

Vorlesung (2 SWS)

- Di., 14:15 - 15:45 Uhr, H-C 6321

Übung (2 SWS)

- Mi., 10:15-11:45, Raum H-C 6321 bzw. H-A 4111 (für praktische Aufgaben)
- Übungsbeginn: Mi. 23.04., H-C 6321
- insgesamt ca. 5 praktische Aufgaben im H-A 4111
  - bitte Information auf der Webseite beachten
  - mit der Nutzung der Rechner akzeptieren Sie die Benutzerordnung (siehe Webseite)!
  - wegen Kennungen bitte ggf. noch **unverzüglich** für Vorlesung und Übung in unisono anmelden!

Zur Vorlesung ...

Information, Folien und Ankündigungen

- **Auf der Vorlesungs-Webseite:**

  `http://www.bs.informatik.uni-siegen.de/lehre/rn2`

- Ggf. Aktualisierungen/Ergänzungen kurz vor der Vorlesung
  - auf das Datum achten!
- Dort auch Übungsblätter und CCNA-Materialien
- Zugangsdaten für geschützte Bereiche:
  - werden in der Vorlesung bekanntgegeben!
- Es gibt auch einen Moodle Kurs mit Aufzeichnungen aus dem Jahr 2021(!), Selbsttests, etc.

Literatur

- Andrew S. Tanenbaum. *Computernetzwerke, 4. Auflage*. Pearson Studium, 2003.
- Larry L. Peterson, Bruce S. Davie. *Computernetze – Eine systemorientierte Einführung, 3. Auflage*. dpunkt.verlag, 2004.
- James F. Kurose, Keith W. Ross. *Computernetze*. Pearson Studium, 2002.
- Weitere Literaturhinweise im Verlauf der Vorlesung
- Skript: kein ausformuliertes Skript, aber Anmerkungen zu etlichen Folien in der 2-auf-1 Version
  - gedruckte Version: bitte beim Fachschaftsrat fragen!

Prüfung

- Mündliche Prüfung
  - Dauer: ca. 30-40 Minuten
- Prüfungstermine:
  - nach Vereinbarung
- Anmeldung:
  - Anmeldung in unisono
    - mindestens eine Woche vorher (besser deutlich früher)
  - danach Terminabsprache über das Sekretariat (Fr. Zetzsche)
    - H-B 8403, `bsvs.zetzsche@eti.uni-siegen.de`, Tel.: -4048
    - mindestens eine Woche vorher (besser deutlich früher)
  - Rücktritt bis 7 Tage vor Prüfungstemin möglich (über unisono)

Inhalt der Vorlesung

Ergänzungen / Vertiefungen zu Rechnernetze I:

- Wide Area Networks (WANs)
  - SONET, *Last Mile*
  - PPP, Frame Relay
- Netzwerk-Technik
  - schnelles Ethernet
  - drahtlose Netze
- Internetworking / IP
  - Routing: Tunnel, Multicast, Mobile IP, MPLS
  - Secure IP

Inhalt der Vorlesung ...

Ergänzungen / Vertiefungen zu Rechnernetze I: ...

- Überlastkontrolle und *Quality of Service*
  - Überlastvermeidung
  - *Quality of Service*
- Anwendungsprotokolle
  - Netzwerkmanagement, Multimedia
- Netzwerkprogrammierung
  - Sockets in C und Java
- Ausblicke
  - Netze für Cluster und Hochleistungsrechner
  - Netze für Realzeit- und Automatisierungssysteme
  - Drahtlose Sensornetze

Ziel der Vorlesung

- Ergänzung und Vertiefung von Rechnernetze I
- Verständnis moderner Netzwerktechniken
  - auch für Bewertung / Auswahl
- Praktische Erfahrungen in der Netzwerkprogrammierung

Rechnernetze II

SoSe 2025

1 *Wide Area Networks* (WANs)

Inhalt

- Einführung
- Etwas Theorie zur Signalübertragung
- Telefonnetz
- Protokolle für Punkt-zu-Punkt-Verbindungen: HDLC, PPP
- Protokolle für paketvermittelte WANs: *Frame Relay*, ATM
- *Last Mile*: ADSL, DOCSIS, GPON

- Tanenbaum, Kap. 1.5.2, 2.1, 2.5.1-2.5.4, 3.6
- Peterson, Kap. 2.3, 3.3
- Kurose, Ross, Kap. 5.8-5.10
- CCNA, Kap. 2, 3, 4, 6

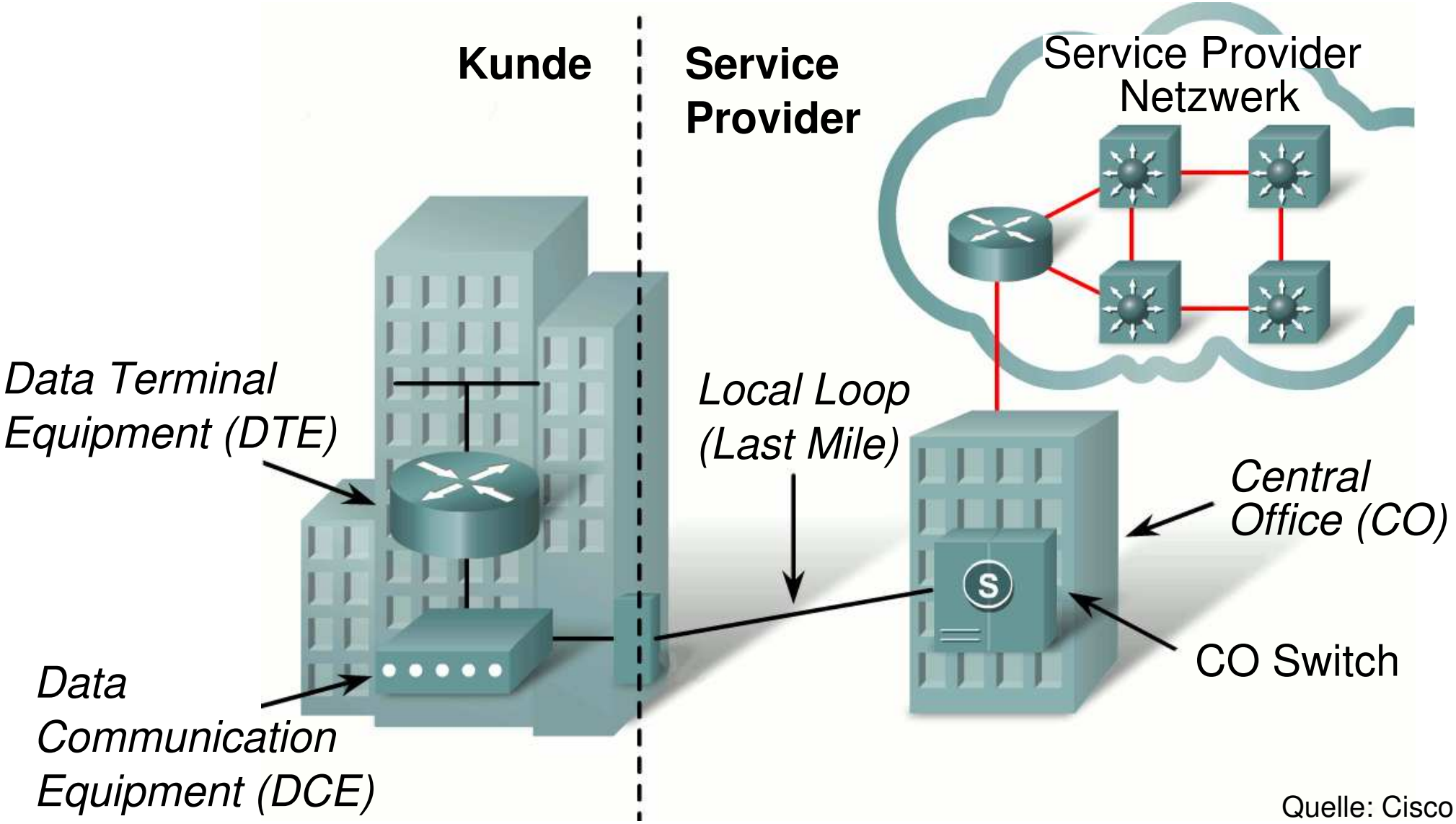
Charakteristika von WANs

- Verbinden Geräte (typ. Router) über größere geographischer Entfernung
- Nutzen Dienste von Kabelbetreibern (*Carrier*)
  - z.B. Telefonanbieter, Kabelfernseh-Anbieter, ...
- Nutzen verschiedene Typen serieller Verbingungen

Einsatz von WANs

- Kommunikation zwischen Firmenstandorten
- Kommunikation zwischen verschiedenen Firmen
- Entfernter Zugang für Firmenmitarbeiter
- Internet-Zugang für Haushalte
- ...

Typische Anbindung an ein WAN



Quelle: Cisco

WAN Protokolle

- WANs decken nur die OSI-Schichten 1 und 2 ab
- Typische Protokolle der Bitübertragungsschicht:
  - EIA/TIA-232 (RS-232): bis zu 64 kb/s, kurze Distanz
  - EIA/TIA-449/530 (RS-422): bis 2 Mb/s, längere Distanzen
  - HSSI (*High-Speed Serial Interface*): bis 52 Mb/s
  - V.35: ITU-T Standard, bis 2,048 Mb/s
- Typische Protokolle der Sicherungsschicht:
  - HDLC, PPP: für dedizierte Punkt-zu-Punkt-Verbindungen
  - ISDN: leitungsvermittelt
  - Frame Relay, X.25, ATM: virtuelle Leitungsvermittlung

Optionen für WAN-Verbindugnen

- Nutzung einer privaten Infrastruktur
  - dedizierte Verbindungen
    - eigene bzw. gemietete Leitungen (Standleitung)
  - vermittelte Verbindungen
    - leitungsvermittelt (Einwahlverbindung), z.B. ISDN
    - paketvermittelt (virtuelle Leitungsvermittlung), z.B. Frame Relay
- Nutzung des öffentlichen Internets
  - Zugang z.B. über DSL (☞ **1.6.1**) oder Kabelmodem
  - Einsatz von VPNs (☞ **4.1** und **RN_I, 5.9**)

Problem bei der Signalübertragung:

- Bandbreite der Leitungen ist begrenzt
  - höhere Frequenzen werden stark gedämpft
  - höchste nutzbare Frequenz abhängig von Leitungsart und -länge
  - Störungen durch Einstrahlung von aussen und Übersprechen
- Frage: Welche Übertragungsrate (bit/s) ist auf einer Leitung mit gegebener Grenzfrequenz (Bandbreite) möglich?
- Antworten liefern:
  - Nyquist-Theorem
  - Shannon'sches Theorem

Nyquist-Theorem (Abtasttheorem)

- Ein Signal mit Bandbreite H [Hz] kann mit $2 \cdot H$ (exakten) Abtastwerten pro Sekunde vollständig rekonstruiert werden
- Die maximal sinnvolle Abtastrate ist daher $2 \cdot H$ [1/s]
- Folgerung für Übertragung mit 1 Bit pro Abtastung:
  - maximale Datenübertragungsrate = $2 \cdot H$ [bit/s]
- Höhere Übertragungsraten sind möglich, wenn pro Abtastung mehr als 1 Bit gewonnen wird
  - im einfachsten Fall: unterscheide 2^n Spannungspegel für n Bits
  - Übertragungsrate ist dann begrenzt durch das **Rauschen** der Leitung

Shannon'sches Theorem

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

Zur Unterscheidung von Übertragungs- und Abtastrate

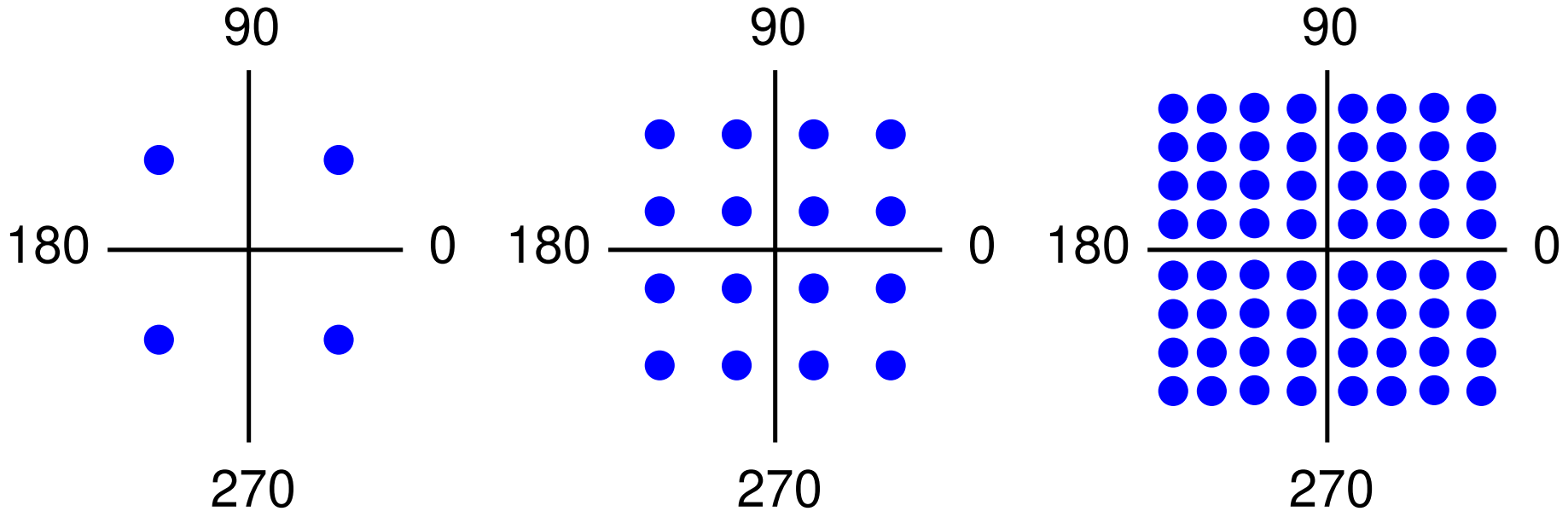
- Einheit **bit/s** für Übertragungsrate
- Einheit **Baud** (Zeichen/s) für Abtastrate

Gewinnung mehrerer Bits pro Abtastung

- Einfachster Fall: betrachte nur Spannungspegel (Amplitude)
  - z.B. Gigabit-Ethernet: 5 Spannungspegel (2 Bits + Redundanz)
- Vereinfachung durch Modulation von zwei Welleneigenschaften
  - Welleneigenschaften: Amplitude, Frequenz, Phase
- Typisch: Modulation von Amplitude und Phase
  - **QPSK** (*Quadrature Phase Shift Keying*)
  - **QAM** (*Quadrature Amplitude Modulation*)
- In der Praxis bis zu 15 Bit pro Abtastung möglich

Modulationsverfahren QPSK und QAM

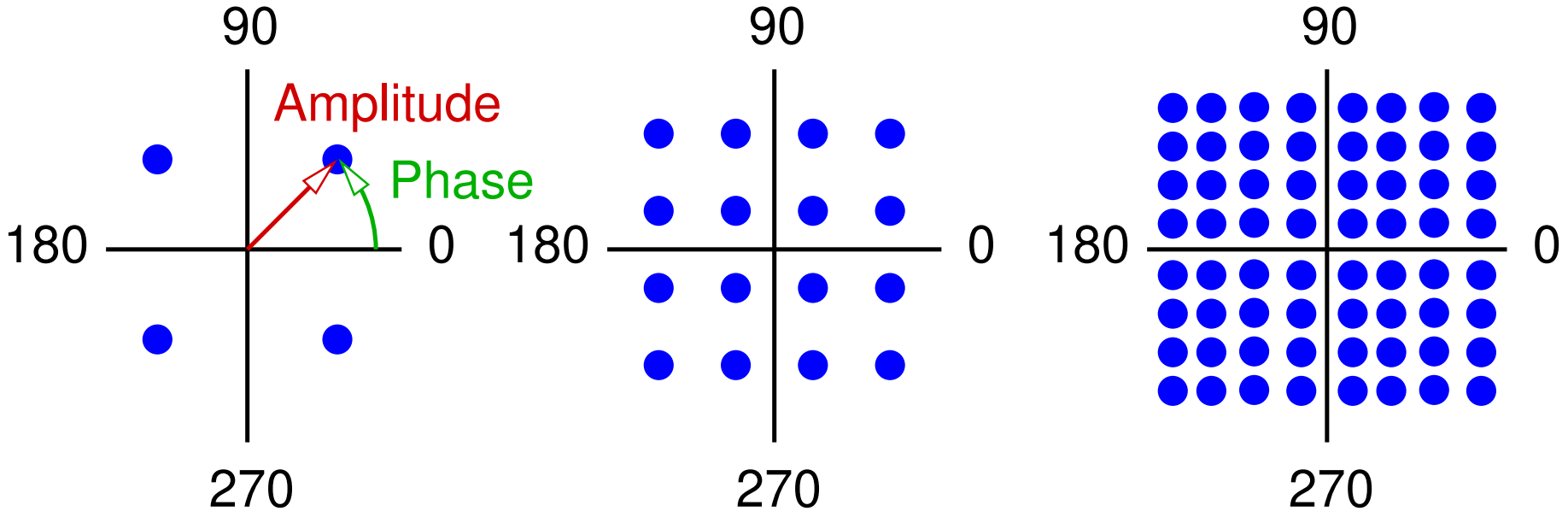
- Funktionsprinzip:
  - jeweils n Bits bestimmen **Amplitude** und **Phase** des Signals
- Beispiele:



QPSK (2 Bit) **16–QAM** (4 Bit) **64–QAM** (6 Bit)

Modulationsverfahren QPSK und QAM

- Funktionsprinzip:
  - jeweils n Bits bestimmen **Amplitude** und **Phase** des Signals
- Beispiele:

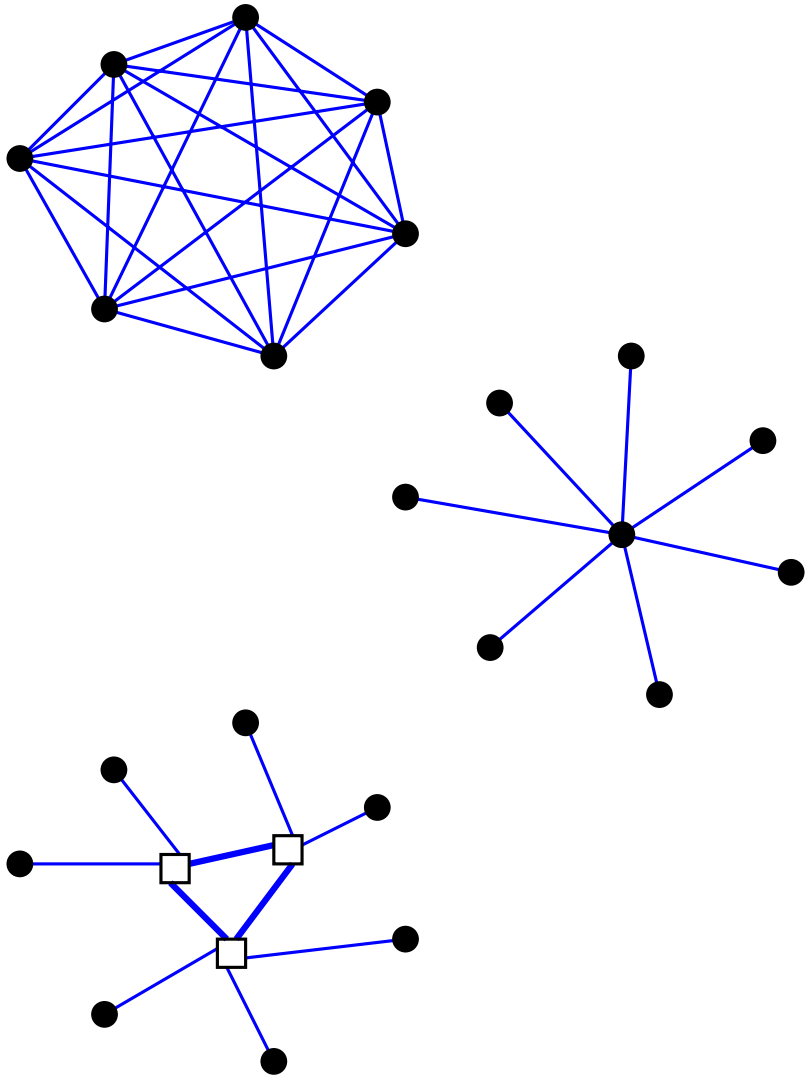


QPSK (2 Bit) **16–QAM** (4 Bit) **64–QAM** (6 Bit)

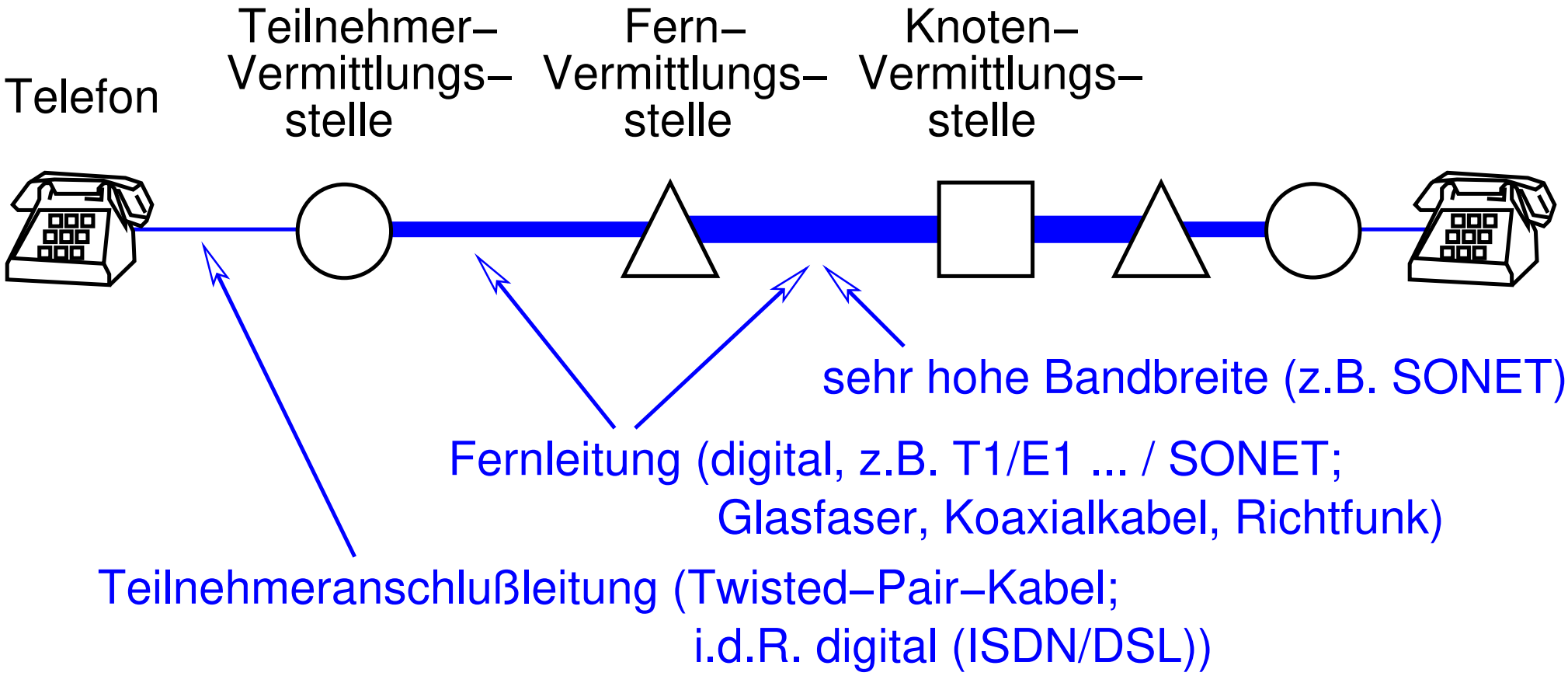
1.3 Telefonnetz

Struktur des Telefonnetzes:

- Zu Beginn: vollständige Vernetzung
  - mit wachsender Teilnehmerzahl unpraktikabel
- Bell (1878): erstes Vermittlungsamt
  - Stern-Topologie
- Danach: Vernetzung der Vermittlungen
  - Hierarchie von Vermittlungen



Typischer Leitungsweg bei mittlerer Entfernung:



- Sprachübertragung direkt digital (ISDN) oder über *Voice over IP* (VoIP, ☞ **7.2**) und DSL

Digitale Übertragung und Multiplexing

- PCM (*Pulse Code Modulation*): Sprachsignale werden im *Codec* digitalisiert:
  - 8000 Abtastungen/s (alle 125 μs)
  - 7 oder 8 Bit pro Abtastung
- Zusätzlich Übertragung von Steuerinformation (Signalisierung)
- Multiplexing mehrerer Gespräche auf eine Leitung
  - Zeitmultiplexing: byte- oder bitweise
- Beispiele: T1/E1, SONET

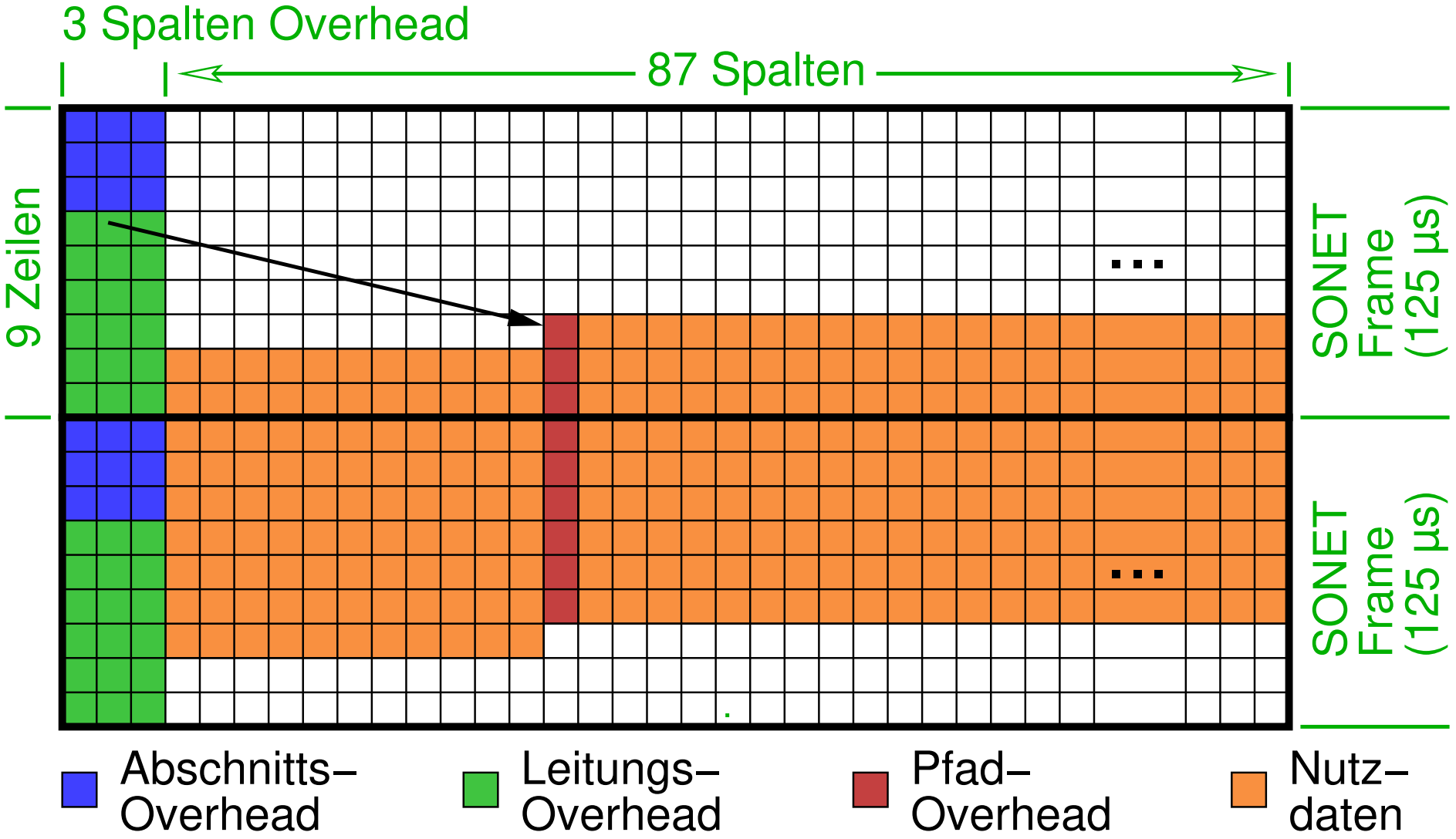
SONET (*Synchronous Optical Network*)

- Vorherrschender Standard für Fernübertragung auf Glasfaser
- Wichtige Eigenschaft: synchrones Netzwerk
  - Takte aller Teilnehmer sind genau synchonisiert
  - Daten kommen beim Empfänger in dem Zeitabstand an, in dem Sender sie geschickt hat
- Leitungsvermittelt
- Steuer- und Verwaltungsinformation werden in den Datenstrom eingestreut
- Im Folgenden: Framing, Multiplexing

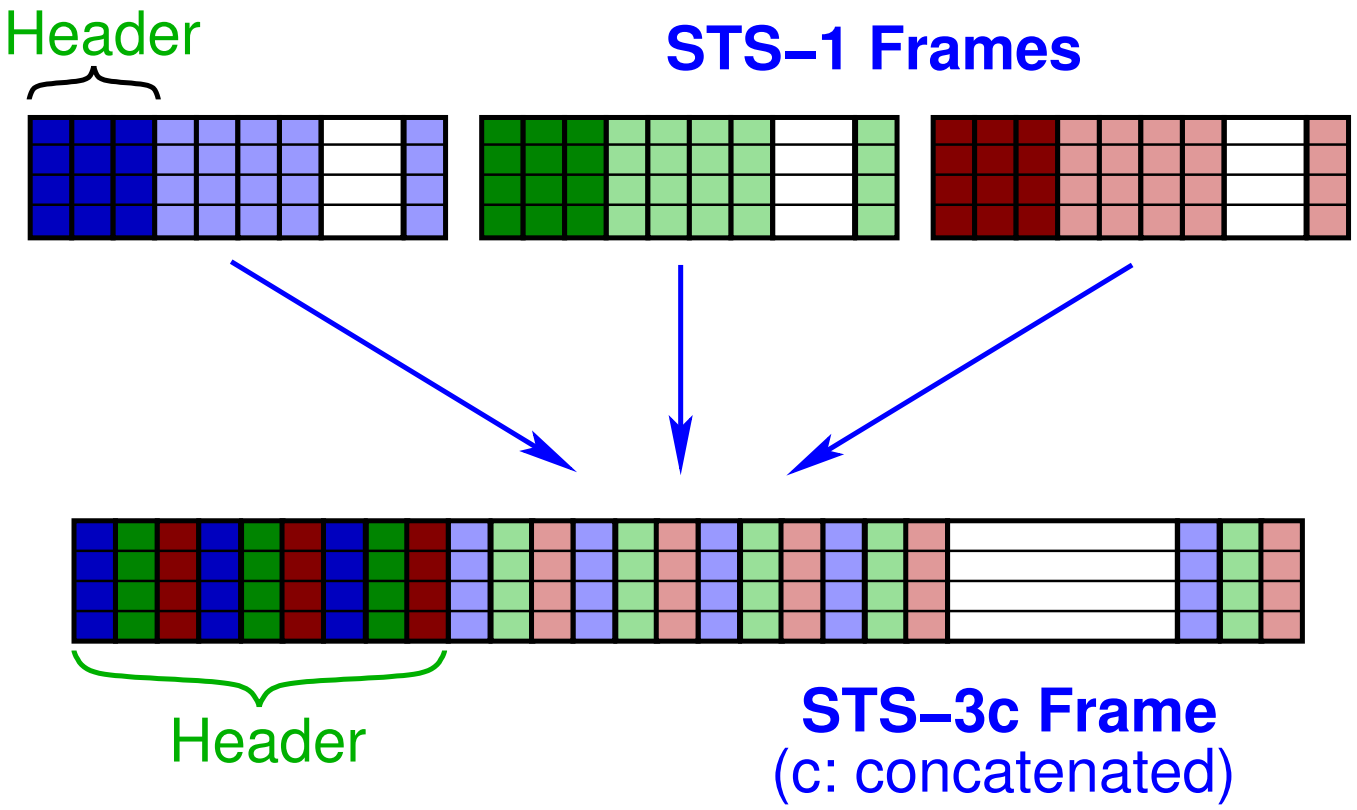
SONET Framing (STS-1: niedrigste Datenrate)

- Feste Framegröße: 810 Byte
- Alle 125 μs Übertragung eines Frames
  - permanent, d.h. ggf. Frames ohne Nutzdaten
  - damit: 51,84 MBit/s Datenrate
- Kein Bit- oder Bytestuffing
- Erkennung des Frame-Anfangs durch 2-Byte-Muster
  - wenn dieses alle 125μs (d.h. alle 810 Bytes) auftaucht, ist Empfänger synchronisiert
- Nutzdaten können an beliebiger Stelle des Frames beginnen

SONET: Aufbau eines STS-1 Frames



SONET: spaltenweises (= byteweises) Multiplexing



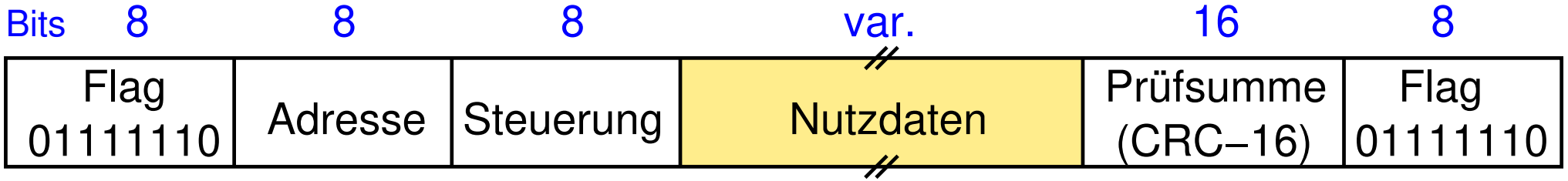
	Mb/s
STS–1	52
STS–3	155
STS–12	622
STS–48	2488
STS–192	9953
STS–768	39813
STS–3072	159252

- STS-x: elektrische, OC-x optische Übertragung

1.4.1 HDLC: *High Level Data-Link Control*

- Weit verbreitetes Schicht-2-Protokoll (ISO/IEC 13239:2002)
  - viele Variationen/Ableger: z.B. LAP (Teil von X.25)
- Eigenschaften:
  - bitorientert, Framing mit Bitstuffing
  - zuverlässige Übertragung (Übertragungsfehler, Reihenfolge)
  - *Sliding-Window*-Algorithmus mit Fenstergröße 7
    - akkumulative und negative ACKs
  - Flußkontrolle
- Drei verschiedene Frame-Typen:
  - *I-Frame*: zur Datenübertragung, mit Sequenznummer
  - *S-Frame*: Steuerung des Datenflusses
  - *U-Frame*: Steuer- und Datenframes ohne Sequenznummer

Frame-Format

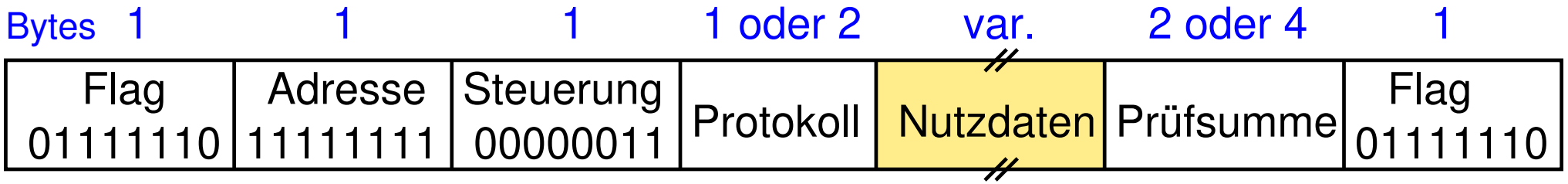


- **Adresse** zur Unterstützug von Punkt-zu-Multipunkt-Verbindungen
- **Steuerung**: enthält je nach Frame-Typ
  - Sequenz-Nummer des Frames
  - Sequenz-Nummer für (negative) Bestätigung
  - Kommando
- Der Datenteil kann beliebig lang sein
- Erweiterung (Cisco):
  - 16-Bit Feld nach Steuerung: übertragenes Schicht-3-Protokoll

1.4.2 PPP: Punkt-zu-Punkt Protokoll

- Protokoll der Sicherungsschicht im Internet
  - für Punkt-zu-Punkt-Verbindungen
    - z.B. Modemverbindung, Standleitung
  - oft auch PPP über Ethernet (PPPoE)
- Anforderungen / Aufgaben:
  - Unterstützung verschiedener Leitungsarten
    - seriell, parallel, synchron, asynchron, ...
  - Framing und Fehlererkennung
  - Unterstützung verschiedener Vermittlungsschicht-Protokolle
  - Aushandeln von Adressen der Vermittlungsschicht
  - Authentifizierung
- Nicht: Fehlerbehandlung, Reihenfolgeerhaltung, Flußkontrolle

PPP Frame-Format



- Basis: HDLC Frame-Format
- Eindeutige Framekennzeichnung durch *Byte-* oder *Bit-Stuffing*
- **Adresse** und **Steuerung** ungenutzt / für Erweiterungen
- **Protokoll** zum Demultiplexen empfangener Frames
  - an höhere Protokolle, z.B. IP, AppleTalk, DECnet, ...
  - an Teilprotokolle von PPP, z.B. LCP, NCP
- Max. Länge des Datenteils kann bei Verbindungsaufbau ausgehandelt werden (Default: 1500 Bytes)
- **Prüfsumme**: CRC, Länge wird ausgehandelt

LCP *Link Control Protocol*

- Für Initialisierung, „Wartung“ und Abschalten der Leitung
- Verbindungsaufbau:
  - Aushandeln der Leitungsoptionen
    - Initiator schlägt vor (*configure request*)
    - Partner nimmt an (*ack*) oder lehnt ab (*nak, reject*)
  - ggf. Authentifizierung
  - danach: Konfiguration der Vermittlungsschicht durch NCP
- Weitere spezielle LCP Frames für
  - Prüfen der Verbindung (*echo request / reply*)
  - Trennen der Verbindung (*terminate request / ack*)

Authentifizierung

- Optional, Aushandlung bei Verbindungsaufbau
  - einseitige und wechselseitige Authentifizierung möglich
- PAP (Password Authentication Protocol)
  - einmalige Übertragung von Nutzername und Passwort
  - im Klartext!
- CHAP (Challenge Handshake Authentication Protocol)
  - 3-Wege Handshake: *Challenge*, *Response*, (N)ACK
  - *Response* ist Hashwert über Passwort und *Challenge*
  - Authentifizierung kann jederzeit wiederholt werden
- Keine Verschlüsselung bzw. Authentifizierung der Daten!

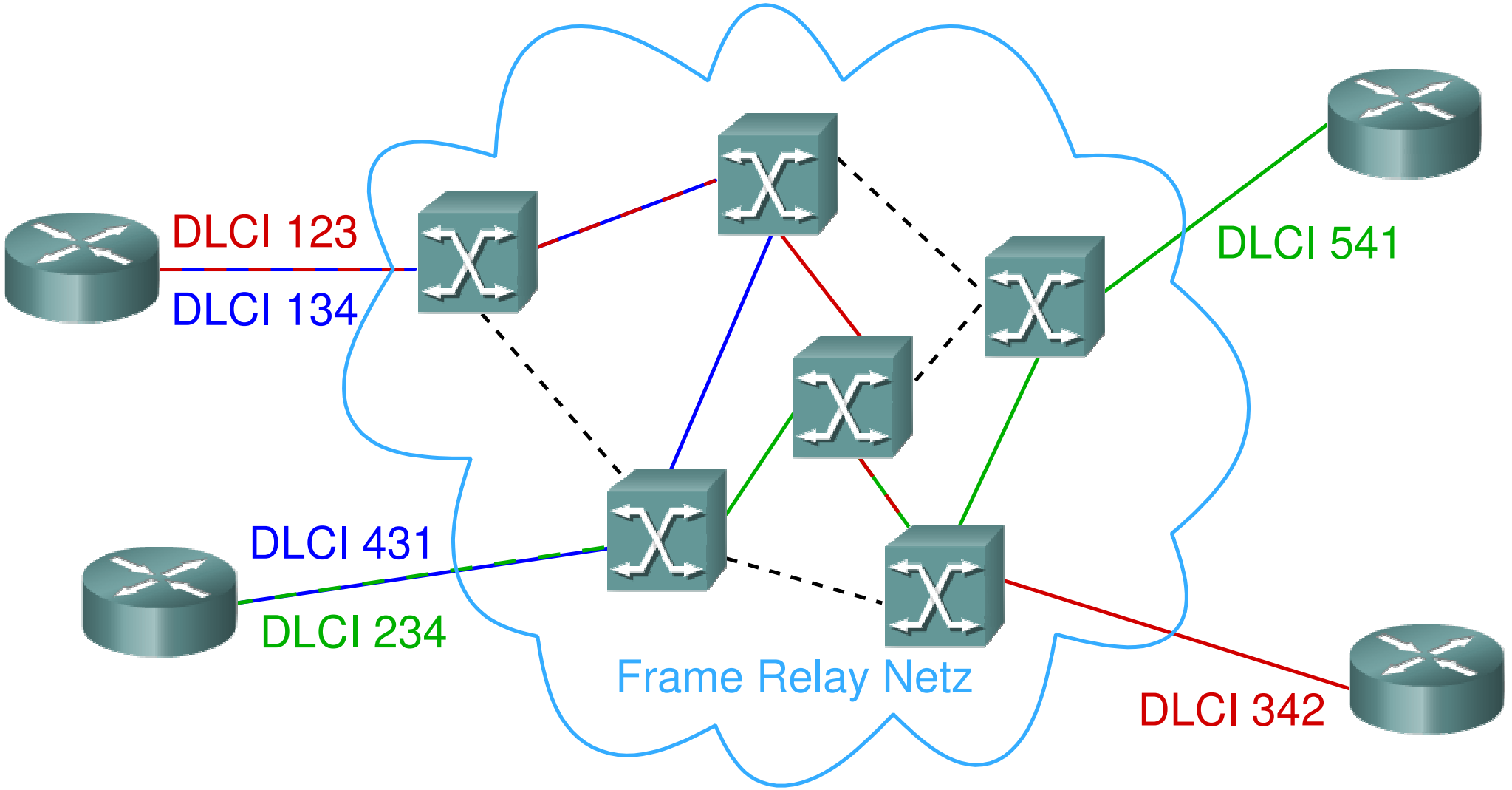
NCP *Network Control Protocol*

- Familie von Protokollen
  - spezifisch für jeweiliges Vermittlungsschicht-Protokoll
- NCP erst nach Verbindungsaufbau mit LCP verwendbar
- spezielles NCP für IP: IPCP (*IP Control Protocol*)
  - Ausgehandelt werden können u.a.:
    - IP-Adresse
    - DNS-Server
    - TCP/IP-Header-Kompression
- Nach Konfiguration mit NCP: PPP durch Vermittlungsschicht-Protokoll nutzbar

1.5.1 *Frame Relay*

- Paketvermittelte Übertragungstechnik für virtuelle Verbindungen
- Wurde von vielen Netzanbietern als Alternative zu Standleitungen angeboten
- Basiert auf dem älteren X.25-Protokoll, ursprünglich für ISDN entwickelt
- Eigenschaften:
  - unzuverlässig, keine Flußkontrolle, einfache Überlastkontrolle
  - *Switched* und *Permanent Virtual Circuits* (SVC, PVC)
  - nur lokal gültige Verbindungs-Identifikatoren
    - DLCI: *Data Link Connection Id*

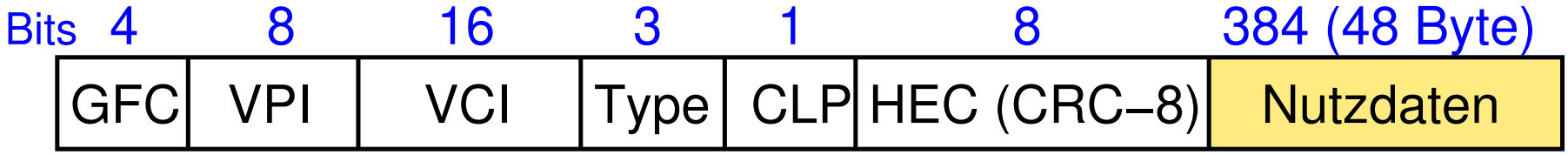
Virtuelle Verbindungen



1.5.2 ATM, *Asynchronous Transfer Mode*

- Entwickelt Anfang der 90'er Jahre
  - Ziel: Eignung für alle Arten digitaler Kommunikation (Telefonie, Video, Computernetze, ...)
- Verbindungsorientiert und paketvermittelt
  - Aufbau virtueller Verbindungen
- Zellen (≙ Frames) fester Länge
  - 53 Byte: 5 Byte Header, 48 Byte Nutzdaten
  - einfaches Forwarding in Hardware
  - *Quality-of-Service* Garantien vereinfacht
    - Leitung durch Zelle nur kurz belegt

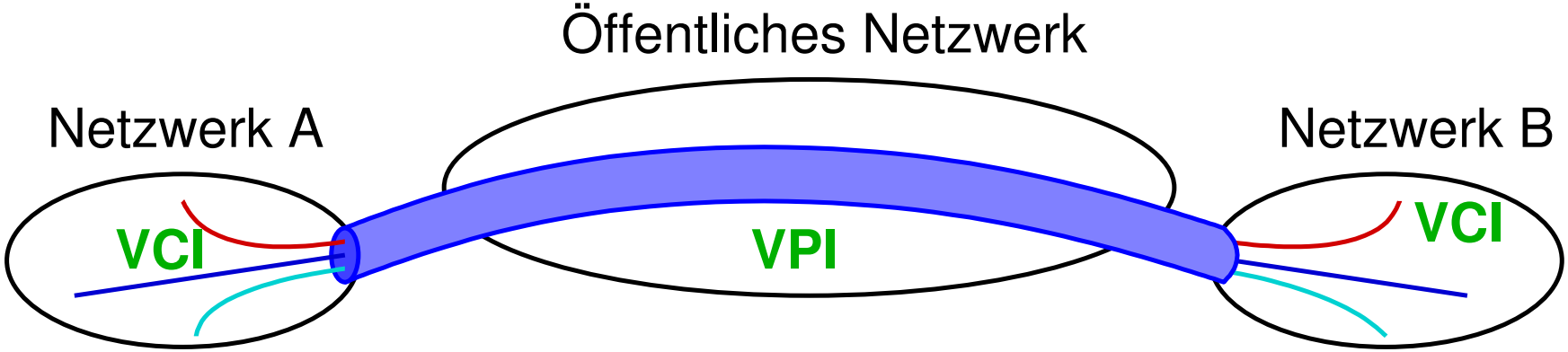
Zellenformat



- **GFC**: *Generic Flow Control* (meist ungenutzt)
- **VPI**: *Virtual Path Identifier*, **VCI**: *Virtual Circuit Identifier*
  - hierarchischer Bezeichner für virtuelle Verbindung
- **Type**: Steuer-/Benutzerdaten; bei Benutzerdaten:
  je ein Bit für Überlastkontrolle und Signalisierung
- **CLP**: *Cell Loss Priority* im Überlastfall
- **HEC**: *Header Error Check*: CRC-8 des Headers

VPI und VCI

- VPI zum Aufbau einer „Leitung“ durch das öffentliche Netz
- Innerhalb dieser Leitung werden durch VCI mehrere Verbindungen gemultiplext
- VCI zur Identifikation innerhalb der lokalen Netze



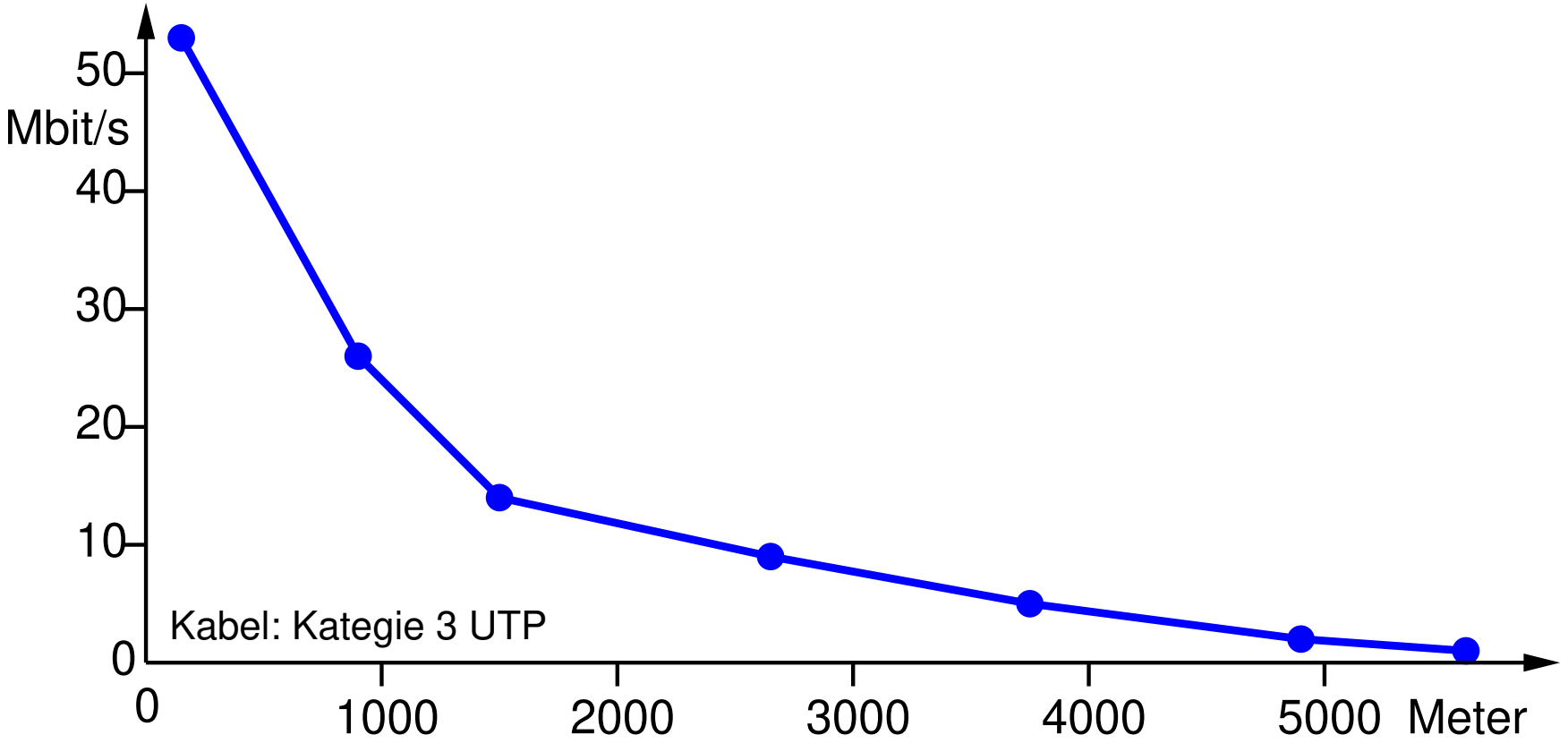
- Vgl. hierarchischer Aufbau von IP-Adressen

AAL: ATM Adaptionsschicht

- Schicht zur Anpassung von ATM an andere Dienste
  - mehrere Typen, je nach Anforderungen
- AAL 1: verbindungsorientiert, konstante Bitrate
  - z.B. unkomprimierte Sprache
- AAL 2: verbindungsorientiert, variable Bitrate, zeitsynchron
  - z.B. komprimiertes Audio / Video
- AAL 3/4: paketorientiert, variable Bitrate, keine Echtzeit
  - Hauptaufgabe: Pakete in Zellen zerlegen und zusammenbauen
  - z.B. für X.25, IP
- AAL 5: wie AAL 3/4, aber mit weniger Overhead

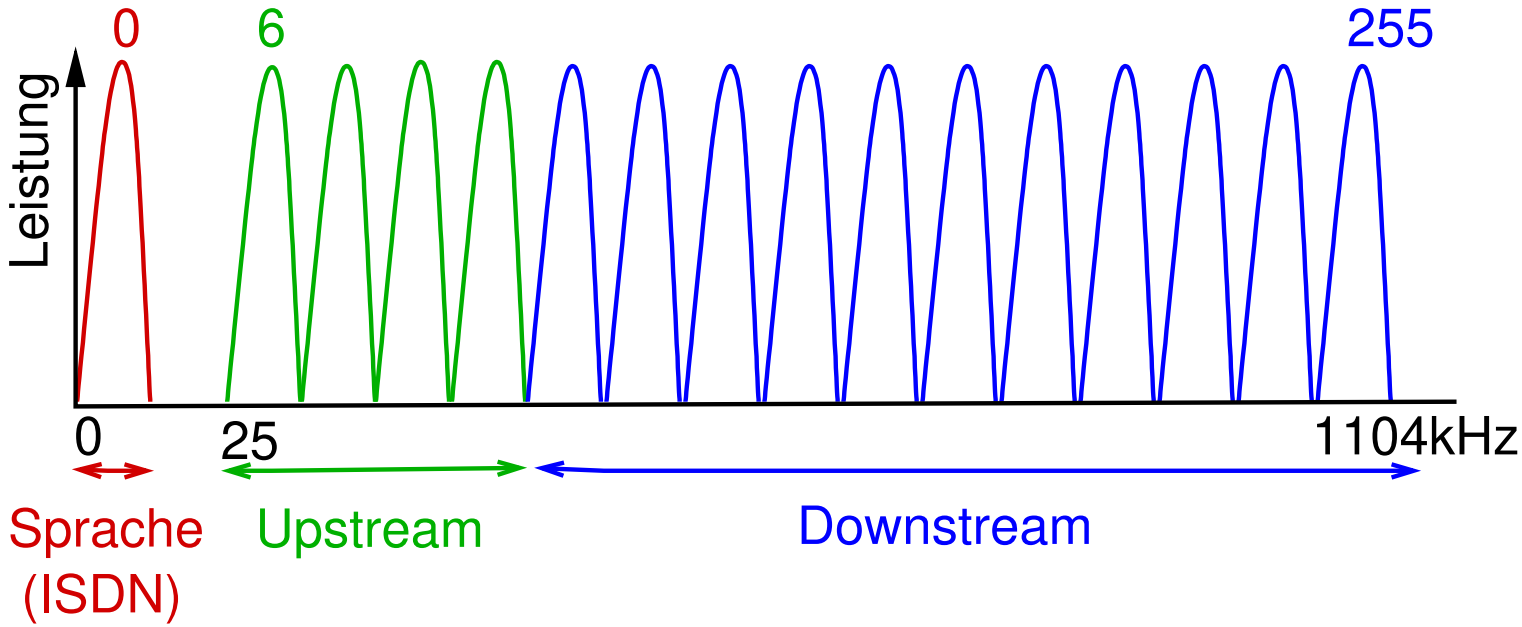
1.6.1 ADSL

- Ziel: Internet-Zugang über Telefon-Teilnehmeranschlußleitung
- Maximale Übertragungsrate abhängig von der Entfernung zur Teilnehmervermittlung:



Übertragungstechnik: DMT (Discrete MultiTone)

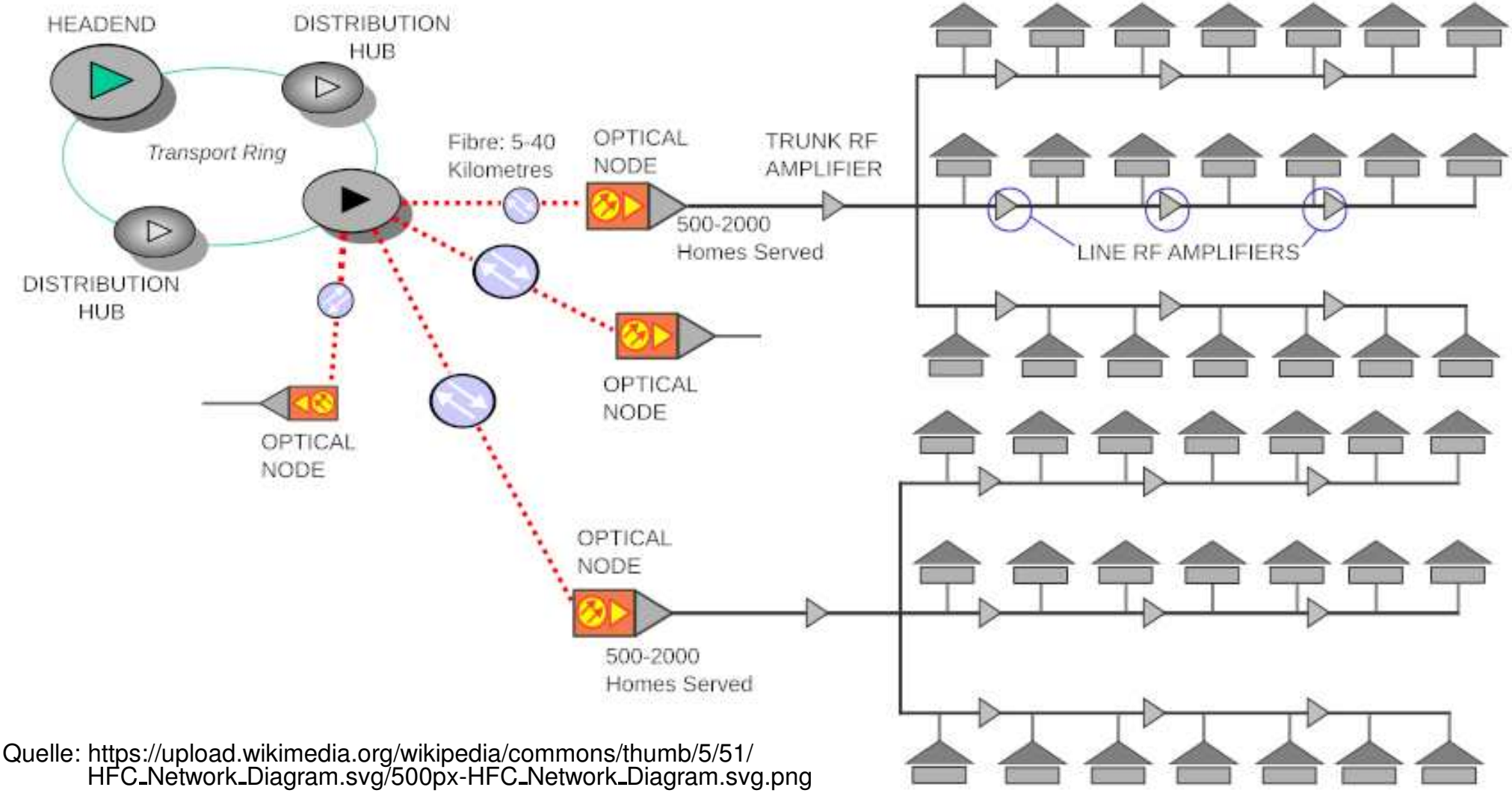
- Einteilung in 256 Frequenzkanäle, je 4 kHz breit:



- Aufteilung in Up- und Downstream flexibel
  - meist 80-90% für Empfangskanal
- Auf jedem Kanal: QAM (4000 Baud, max. 15 Bit pro Zeichen)
  - Kanäle werden beim Verbindungsaufbau ausgemessen

1.6.2 *Data Over Cable Service Interface Spec.* (DOCSIS)

- Breitbandanschluß über das Kabelfernseh-Netz



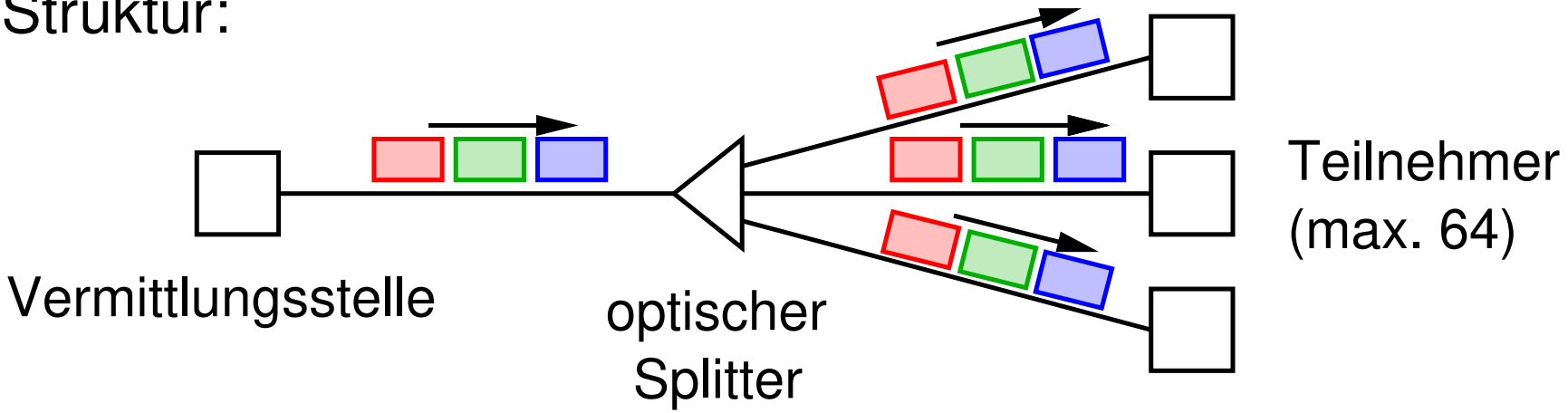
Quelle: https://upload.wikimedia.org/wikipedia/commons/thumb/5/51/HFC_Network_Diagram.svg/500px-HFC_Network_Diagram.svg.png

1.6.2 DOCSIS ...

- Datenraten (DOCSIS 3.1):
  - max. 10 Gb/s downstream, 1 Gb/s upstream
- Übertragung mit OFDM (ähnlich DMT, ☞ **3.1.1**)
  - downstream: 400-1800MHz, bis 3800 bzw. 7600 Unterkanäle
  - upstream: 5-400MHz, bis 1900 bzw. 3800 Unterkanäle
  - Trägerabstand der Unterkanäle: 50 bzw. 25 kHz
- Modulationsverfahren: QAM
  - bis 4096-QAM (12 Bit / Abtastung)
- Medienzugriffskontrolle (upstream):
  - FDMA / TDMA: Kabelnetz teilt Frequenz und Zeitschlitze zu
  - S-CDMA: Mischung aus TDMA und CDMA (☞ **3.1.1**)
- Datenüberragung erfolgt verschlüsselt (DOCSIS 3.0: 128 Bit AES)
  - downstream-Übertragung erfolgt als Broadcast

1.6.3 *Gigabit Passive Optical Network* (GPON)

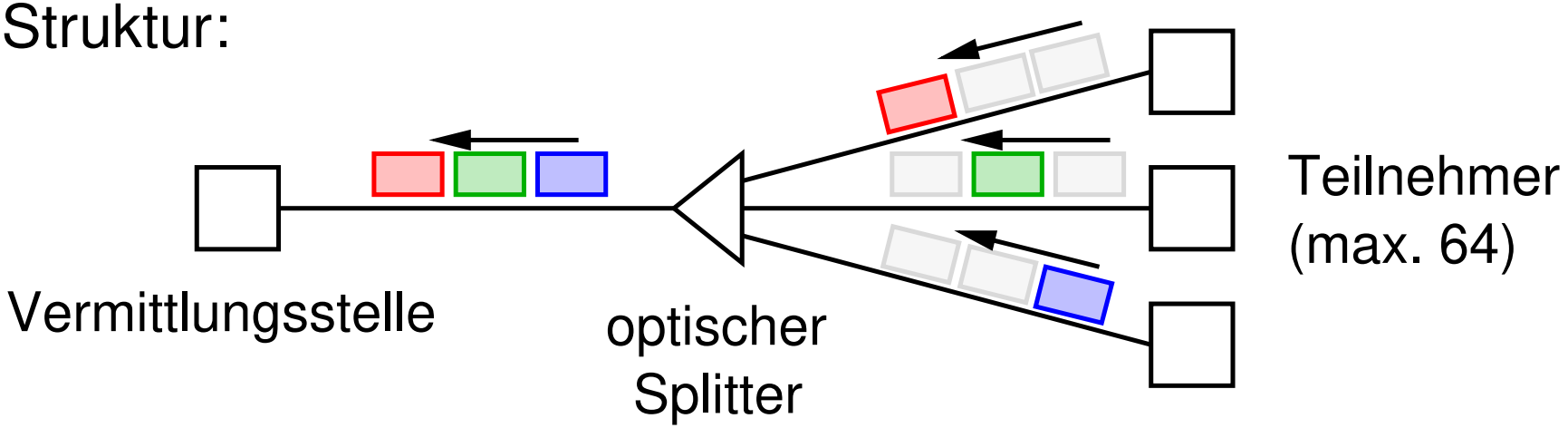
- ITU-T G.984 Standard für Glasfasernetze
- Struktur:



  - downstream: Broadcast
  - über eine Faser mit zwei verschiedenen Wellenlängen
- Übertragungsraten: 2.5 Gb/s downstream, 1.25 Gb/s upstream
- Übertragung eines GPON-Frames alle 125 μs

1.6.3 *Gigabit Passive Optical Network* (GPON)

- ITU-T G.984 Standard für Glasfasernetze
- Struktur:



  - downstream: Broadcast, upstream: TDMA
  - über eine Faser mit zwei verschiedenen Wellenlängen
- Übertragungsraten: 2.5 Gb/s downstream, 1.25 Gb/s upstream
- Übertragung eines GPON-Frames alle 125 μs

1.6.3 GPON ...

- GPON-Frame kann mehrere ATM-Zellen oder GEM-Frames enthalten
  - Header enthält u.a. Information zu TDMA-Zeitlsots
- GEM (*GPON Encapsulation Method*)
  - verbindungsorientiert
  - adressiert Teilnehmer über Port-IDs
  - fragmentiert Nutzdaten
  - kapselt u.a. E1/T1-Daten oder Ethernet-Frames
- Übertragung mit AES (128 Bit) verschlüsselt

- Neben GPON auch EPON (*Ethernet Passive Optical Network*)
  - überträgt Ethernet-Frames direkt

1.7 Zusammenfassung / Wiederholung

- Nyquist-Theorem
  - Signal mit Bandbreite H: max. $2 \cdot H$ Abtastungen / s
- Shannon'sches Theorem
  - Leitung mit Bandbreite H und Rauschabstand S/N: max. Datenübertragungsrate $H \cdot \log_2(1 + S/N)$
- bit/s versus Baud

- Telefonnetz (T1/E1, SONET)
  - synchrone Netze (garantierte, konstante Datenrate)
  - bit- bzw. byteweises Multiplexing
    - mehrere Datenströme über ein Kabel
  - taktbasiertes Framing (kein Bit-/Bytestuffing)

1.7 Zusammenfassung / Wiederholung ...

- PPP: Sicherungsschicht-Protokoll im Internet
  - unzuverlässig, keine Flußkontrolle
  - optionale Authentifizierung (PAP, CHAP)
  - Aushandlung von Parametern für Vermittlungsschicht
- *Frame Relay*: virtuelle Verbindungen
- ATM: Zellenvermittlung, virtuelle Verbindungen

- ADSL: 256 Kanäle á 4 kHz, jeweils mit QAM
- DOCSIS: OFDM, QAM, TDMA bzw. CDMA für upstream
- GPON: zwei Wellenlängen, TDMA für upstream

Rechnernetze II

SoSe 2025

2 Schnelles Ethernet

Inhalt

- 100 Mb/s
- 1 Gb/s
- 10 Gb/s

- Tanenbaum, Kap. **4.3.7-4.3.8**
- Kurose, Ross, Kap. 5.5

Zur Erinnerung: Klassisches Ethernet (IEEE 802.3)

- Gemeinsam genutztes Medium mit CSMA/CD
  - (logische) Bus-Topologie
  - maximale Leitungs- und minimale Framelänge wegen Kollisionserkennung
- Leitungscodierung: Manchester-Code
  - erfordert 10 MHz Bandbreite für 10 Mb/s
- Varianten (Kabeltypen):
  - 10Base5, 10Base2: Koaxialkabel als gemeinsamer Bus
  - 10BaseT: UTP Kat. 3 Kabel (Telefonleitung, max. 16 MHz)
    - nur in Verbindung mit Hubs / Switches
    - 2 Adernpaare für Duplex-Betrieb, max. 100 m Länge

100 MBit/s Ethernet (IEEE 802.3u)

- Ziel: keine wesentlichen Änderungen am Standard
  - Kompatibilität zu existierender Soft- und Hardware (Kabel)
- Rahmenformat, Schnittstellen (LLC) etc. unverändert
- Bitzeit auf 10 ns verkürzt
- Busverkabelung nicht mehr unterstützt
  - max. Kabellänge zu gering
  - Sternverkabelung hat sich schon bei 10 Mb/s durchgesetzt
- Verbindung nur über Hubs und/oder Switches
- Viele Varianten für verschiedene Kabelarten

100BASE-T4

- Kann mit UTP Kat. 3 Kabeln arbeiten (Telefonkabel, max. 100m)
- Maßnahmen zur Erhöhung der Übertragungsrate:
  - andere Leitungscodierung: 8B6T statt Manchester
    - 8 Bits werden auf 6 Trits (ternäre Zeichen) abgebildet
    - Übertragung mit 3 Spannungspegeln
  - Verwendung aller 4 Adernpaare
    - 1 Paar zum Hub, 1 vom Hub, 2 umschaltbar
- Somit: max. 3 Adernpaare für eine Richtung
  - 3 Trits ($\hat{=}$ 4 Bits) pro Abtastung, 100 Mb/s bei 25 MBaud
- Zusätzlich 33 Mb/s Rückkanal (für Kollisionserkennung)

100BASE-T2

- 100BASE-T4 belegt alle Adernpaare
  - Nachteil bei Nutzung vorhandener Telefonkabel
- Daher: 100BASE-T2 (IEEE 802.3xy) später ergänzt
- Kommt mit 2 Paaren eines UTP Kat. 3 Kabels aus
- PAM 5x5 Codierung:
  - 4 Bit werden in zwei fünfwertige Signale codiert
  - Übertragung mit 5 Spannungspegeln auf jeder Leitung
  - ergibt 100 Mb/s bei 25 MBaud (halb- und vollduplex)
  - im Vollduplex-Modus: Echokompensation (*echo cancellation*)
- 100BASE-T2 hat sich (wie 100BASE-T4) nicht durchgesetzt

100BASE-TX

- Benötigt UTP Kat. 5 Kabel (100 MHz, max. 100 m)
  - je ein Adernpaar pro Richtung (vollduplex)
- 4B5B Leitungscodierung statt Manchester
  - ergibt 125 MHz Abtastrate (125 MBaud)

100BASE-FX

- Arbeitet mit Multimode-Glasfaser, max. 2 km
- Sonst wie 100BASE-TX

100 Mb/s Ethernet: Besonderheiten und Gemeinsamkeiten

- Verwendung „ungültiger“ Leitungscodes für Steuerzwecke
  - u.a. Markierung der Frame-Grenzen
- Maximale Netzgröße bei Verwendung von Hubs nur ca. 200 m
  - auch bei 100BASE-FX nur max. 272 m (Kollisionen!)
- *Autonegotiation*
  - Geschwindigkeit (10 Mb/s, 100 Mb/s) und Duplexmodus können ausgehandelt werden (Bitübertragungsschicht)
  - Ethernet-Karten senden im Leerlauf *Link Pulses* zur Prüfung der Leitung
    - Konfigurierungsinformation wird in diese Pulse eincodiert

1 Gb/s Ethernet (IEEE 802.3z)

- Ziel: Kompatibilität mit vorhandenen Standards
- Zwei Betriebsmodi für Netze mit Switches bzw. Hubs
- Vollduplex-Betrieb mit Switches
  - kein CSMA/CD nötig, da keine Kollisionen
  - Kabellänge nur durch Signalqualität beschränkt
- Halbduplex-Betrieb mit Hubs
  - Problem: Behandlung von Kollisionen
    - 64 Byte min. Framelänge $\Rightarrow$ max. Netzgröße 25 m!?
  - Lösung: Hardware stellt Framelänge $\geq$ 512 Byte sicher
    - *Carrier Extension*: Auffüllen des Frames
    - *Frame Bursting*: Zusammenfassen mehrerer Frames

1 Gb/s Ethernet: Varianten

- Glasfaser:
  - 1000BASE-SX: Multimode-Faser, 550 m
  - 1000BASE-LX: Mono- oder Multimode-Faser, max. 5000 m
  - 8B10B Codierung: jedes Byte wird mit 10 Bit codiert
    - max. 4 gleiche Bits nacheinander (Taktsynchronisation)
    - max. 6 Einsen / Nullen pro Wort (Gleichstromanteil)
- 1000BASE-CX: geschirmtes Twisted-Pair Kabel (STP), 25 m
- 1000BASE-T: UTP Kat. 5, 100 m
  - alle 4 Adernpaare genutzt
  - 2 Bit pro Zeichen (PAM 5, 5 Spannungspegel), 125 MBaud
  - Echokompensation für Vollduplex-Betrieb
  - keine Leitungscodierung ($\Rightarrow$ komplexe Taktsynchronisation)

10 Gb/s Ethernet (IEEE 802.3ae, ak, an, ap, aq)

- Nur noch Vollduplex-Betrieb mit Switches
- Sehr viele Varianten:
  - Glasfaser: 10GBASE-SR, 10GBASE-LR, 10GBASE-LRM, ...
  - SONET-Interoperabilität (OC-192): 10GBASE-SW, ...
  - Backplane (z.B. Blade-Server): 10GBASE-KX4, 10GBASE-KR
- 10GBASE-T: UTP Kat. 6a, 100m (mit Kat. 6: 50m)
  - Kat. 6a Kabel: bis 500 MHz
  - alle 4 Adernpaare genutzt
  - PAM16-Modulation mit 16 Spannungspegeln
  - 3 Bit pro Zeichen (1 Bit Redundanz), 833 MBaud/s
  - Echokompensation, keine Leitungscodierung

Flußkontrolle

- Wegen hoher Datenrate: neue Ethernet-Standards unterstützen einfache Flußkontrolle
- Empfänger sendet PAUSE-Frame an Sender
  - gekennzeichnet durch speziellen Typ / Zieladresse
  - 16-Bit Parameter gibt Länge der Pause an (in Einheiten von 512 Bitzeiten)
  - Sender stellt für diese Zeit die Übertragung ein
- Möglich nur im Vollduplex-Modus

Fast Ethernet: Fazit

- Ziel: Kompatibilität
  - Software, Verkabelung
- Verschiedene Realisierungen auf Bitübertragungsebene
- Heute vorherrschend:
  - 100BASE-TX, 1000BASE-T/SX/LX
  - 10 Gb/s vorwiegend im Core-Bereich
- 40 und 100 Gb/s Ethernet Standard (2010):
  - Glasfaser, Backplane, Kupferkabel (twinax, 4/10 Bit parallel)
  - 40 Gb/s auch über Cat. 8 Kabel (max. 30m)
- 200 und 400 Gb/s Ethernet Standard (2017)
  - Glasfaser, 4 bzw. 8 Wellenlängen (CWDM)

2.2 Zusammenfassung / Wiederholung

- Ziel: Kompatibilität mit Software / Verkabelung
- Geschwindigkeitserhöhung durch
  - andere Übertragungsmedien (höhere Grenzfrequenz)
  - Nutzung mehrerer Adernpaare (parallele Übertragung)
  - effizientere Codierungen
  - mehr Bits pro Abtastung
- Wegen Kollisionen: nur noch Verwendung von Switches
  - ab 10 Gb/s kein CSMA/CD mehr

Rechnernetze II

SoSe 2025

3 Drahtlose Netze

Inhalt

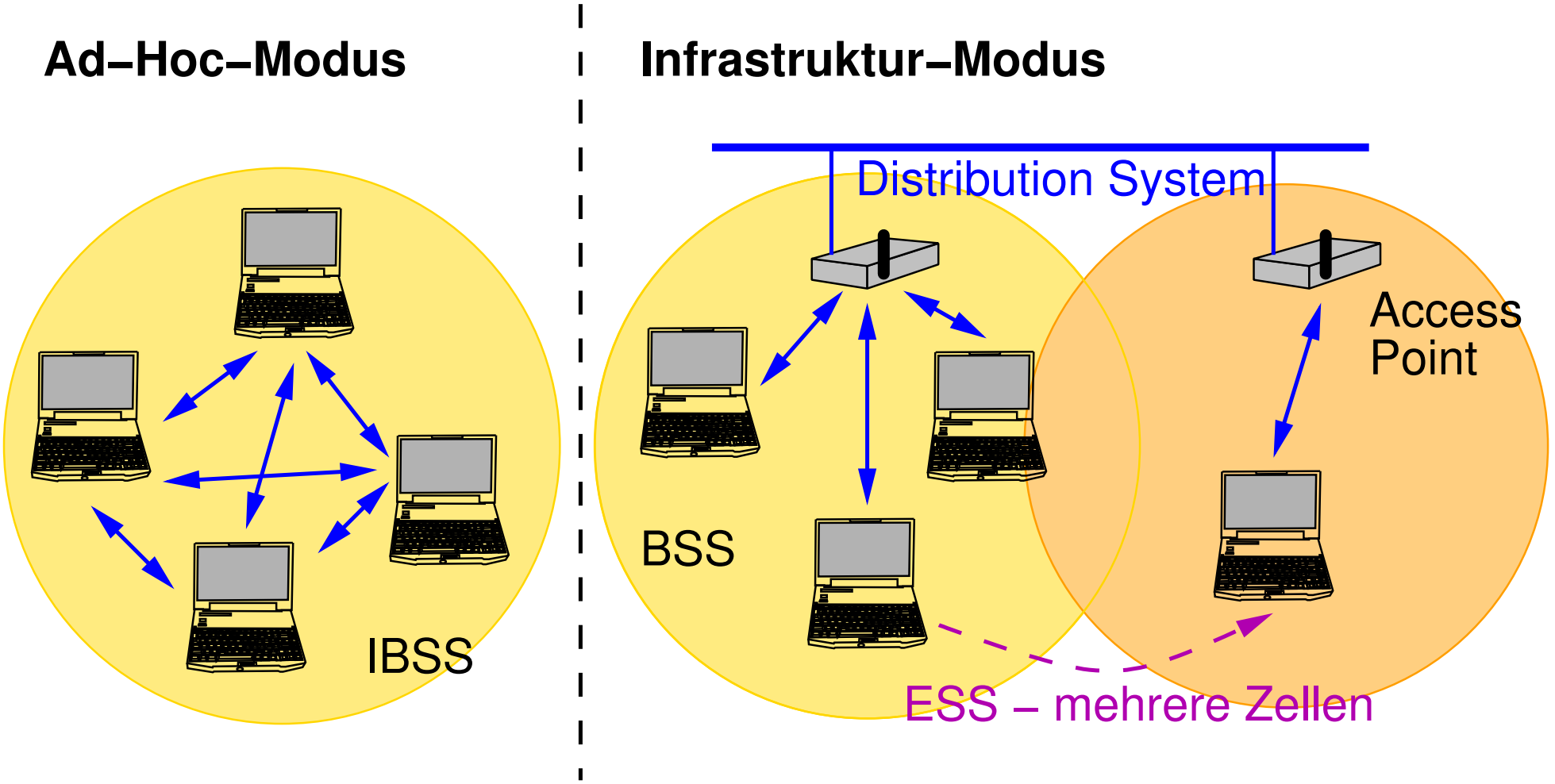
- WLAN (IEEE 802.11)
- Bluetooth (IEEE 802.15)

- Tanenbaum, Kap. 1.5.4, **4.4**, **4.6**
- Peterson, Kap. **2.8**
- Axel Sikora: Wireless LAN, Addison Wesley, 2001.
- Jörg Rech: Wireless LANs, 2. Auflage, Heise Verlag, 2006.
- Edgar Nett, Michael Mock, Martin Gergeleit: Das drahtlose Ethernet, Addison-Wesley, 2001.

Hintergrund

- Drahtlose Netzanbindung von mobilen Geräten
- Sicherungsschicht kompatibel zu Ethernet
- Unterstützung für zwei Betriebsmodi:
  - Ad-Hoc-Modus: Endgeräte kommunizieren direkt
    - IBSS (*Independent Basic Service Set*)
  - Infrastruktur-Modus: Kommunikation über *Access Point*
    - BSS (*Basic Service Set*): eine Funkzelle
    - ESS (*Extended Service Set*): mehrere Funkzellen, über ein anderes Netz (z.B. Ethernet oder auch WLAN) verbunden

WLAN-Betriebsmodi



802.11 Protokollstack

Höhere Schichten					
Logical Link Control (802.2, wie bei Ethernet)					
MAC–Teilschicht: CSMA/CA, MACAW					
802.11 IR / 2.4 GHz FHSS/DSSS 2 Mb/s	802.11a 5 GHz OFDM 54 Mb/s	802.11b 2.4 GHz HR–DSSS 11 Mb/s	802.11g 2.4 GHz OFDM 54 Mb/s	802.11n 2.4/5 GHz OFDM/MIMO – 600 Mb/s	802.11ac 5 GHz OFDM/MIMO – 1.69 Gb/s

Siche–
rungs–
schicht

Bitüber–
tragungs–
schicht

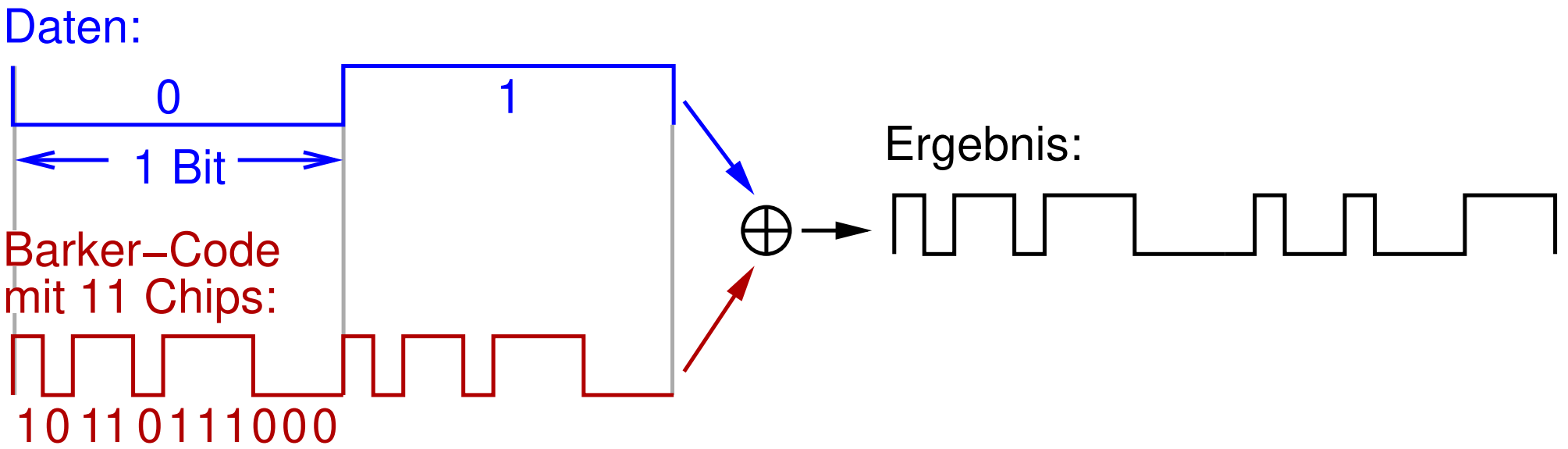
➥ Im Folgenden: Schwerpunkt auf 802.11b und 802.11g

Basis der Funkübertragung: Spreizbandtechnik

- Problem: 802.11 arbeitet in feigegebenen ISM-Bändern
  - ISM: Industrial, Scientific, Medical
  - 2,4 GHz und 5 GHz
- Maßnahme gegen Funkstörungen:
  - Übertragung in möglichst breitem Frequenzband
  - Störungen sind meist schmalbandig
- Techniken:
  - FHSS (*Frequency Hopping Spread Spectrum*)
    - viele Kanäle, Frequenz wechselt pseudozufällig
  - OFDM (*Orthogonal Frequency Division Multiplexing*)
    - im Prinzip ähnlich zu DMT (☞ **1.6.1**: ADSL)

Basis der Funkübertragung: Spreizbandtechnik ...

- Techniken ...:
  - DSSS (*Direct Sequence Spread Spectrum*)
    - Sendedaten werden mit (fester!) Pseudozufallsfolge höherer Bitrate XOR-verknüpft
    - Muster leicht aus verrauschtem Signal „herauszuhören“

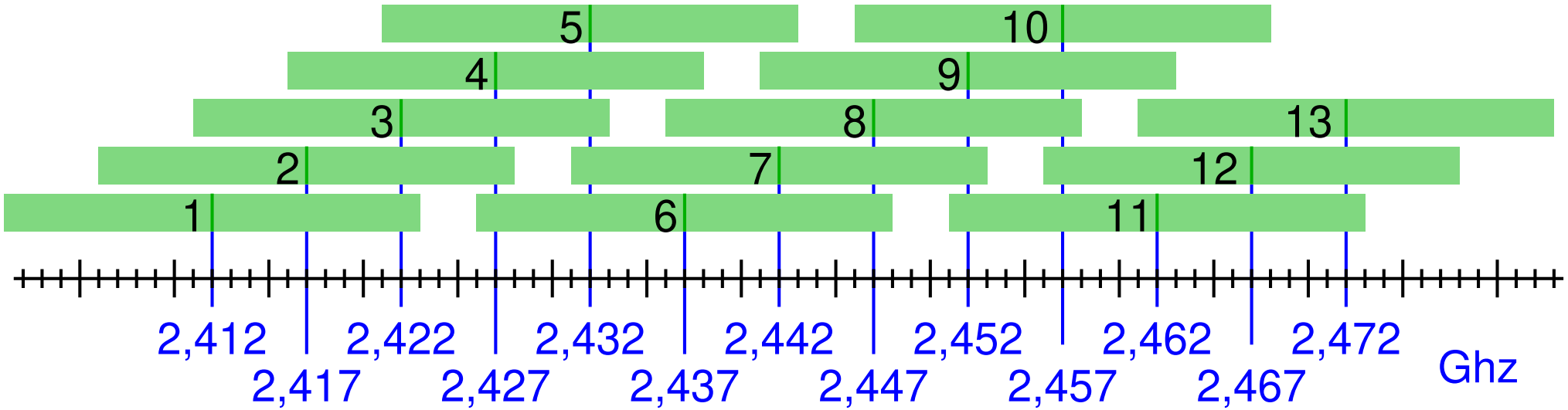


Basis der Funkübertragung: Spreizbandtechnik ...

- Techniken ...:
  - HR-DSSS (*High Rate DSSS*)
    - verkürzte Codelänge: 8 Chips
    - QPSK-artige Modulation
      - 4 Bit / Symbol (für 5.5 Mb/s)
      - 8 Bit / Symbol (für 11 Mb/s)
    - benötigt höheren Rauschabstand

Frequenzbänder im 2.4 GHz Band

- 13 Kanäle (Europa)
- Bandbreite eines Kanals bei 802.11b: 22 MHz
- Kanäle überlappen!
  - bei 802.11b max. 3 nicht überlappende Kanäle möglich



WLAN nach IEEE 802.11g

- Bruttodatenrate bis 54 Mbit/s (netto max. 50%)
- Verwendet OFDM wegen Mehrfachempfang durch Reflexionen
  - Problem verschärft sich bei höherer Bitrate
  - daher: parallele Übertragung auf mehreren (48) Unterkanälen
- Unterschiedliche Modulationsarten (z.B. QAM-16, QAM-64)
  - Symbolrate: 250 kHz
- Vorwärts-Fehlerkorrektur
  - Code-Rate 1/2, 2/3 oder 3/4 (Nutzdaten / Gesamtdaten)
- Zur Kompatibilität mit 802.11b:
  - 802.11g-Geräte unterstützen i.d.R. auch DSSS

Medienzugriffssteuerung (MAC)

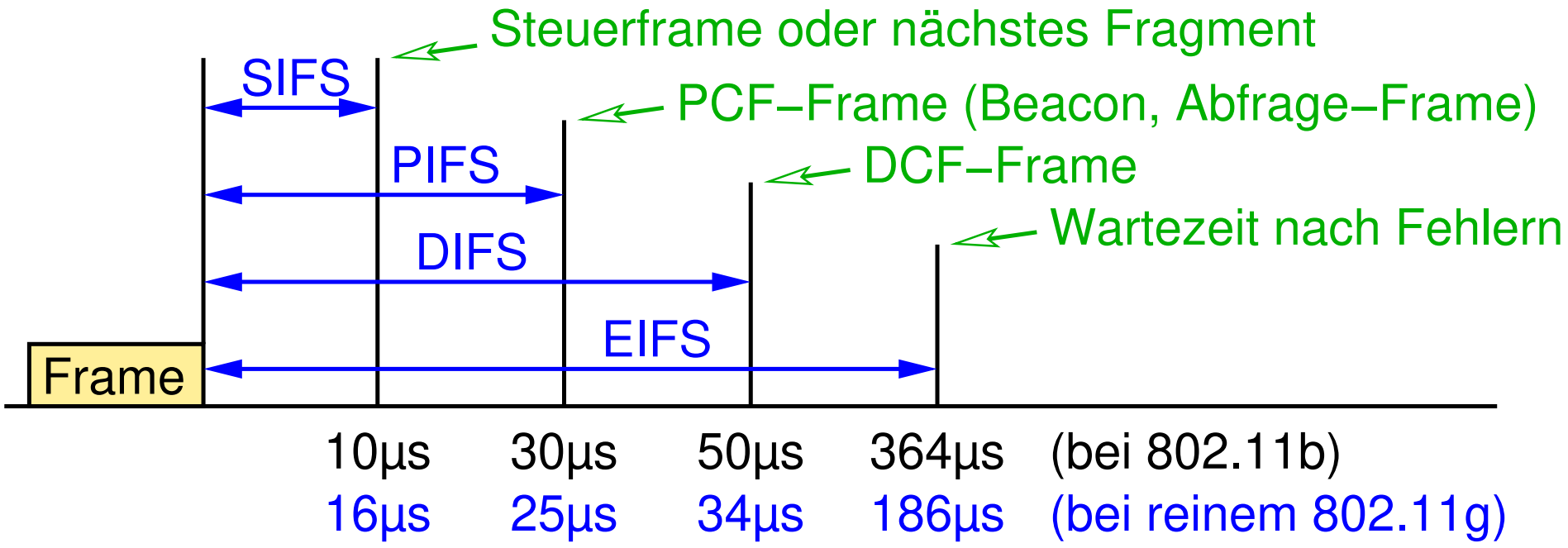
- CSMA/CD ist nicht möglich
  - Funkgeräte arbeiten im Halbduplex-Modus
  - während des Sendens kein Mithören möglich
  - nur Empfänger „erkennt“ Kollision (durch Prüfsumme)
- In IEEE 802.11 zwei Modi für Zugriffssteuerung:
  - DCF (*Distributed Coordination Function*)
    - dezentrales Verfahren (CSMA/CA, MACAW)
  - PCF (*Point Coordination Function*)
    - zentrale Steuerung durch den *Access Point*
  - beide Modi können gleichzeitig genutzt werden

DCF: CSMA/CA

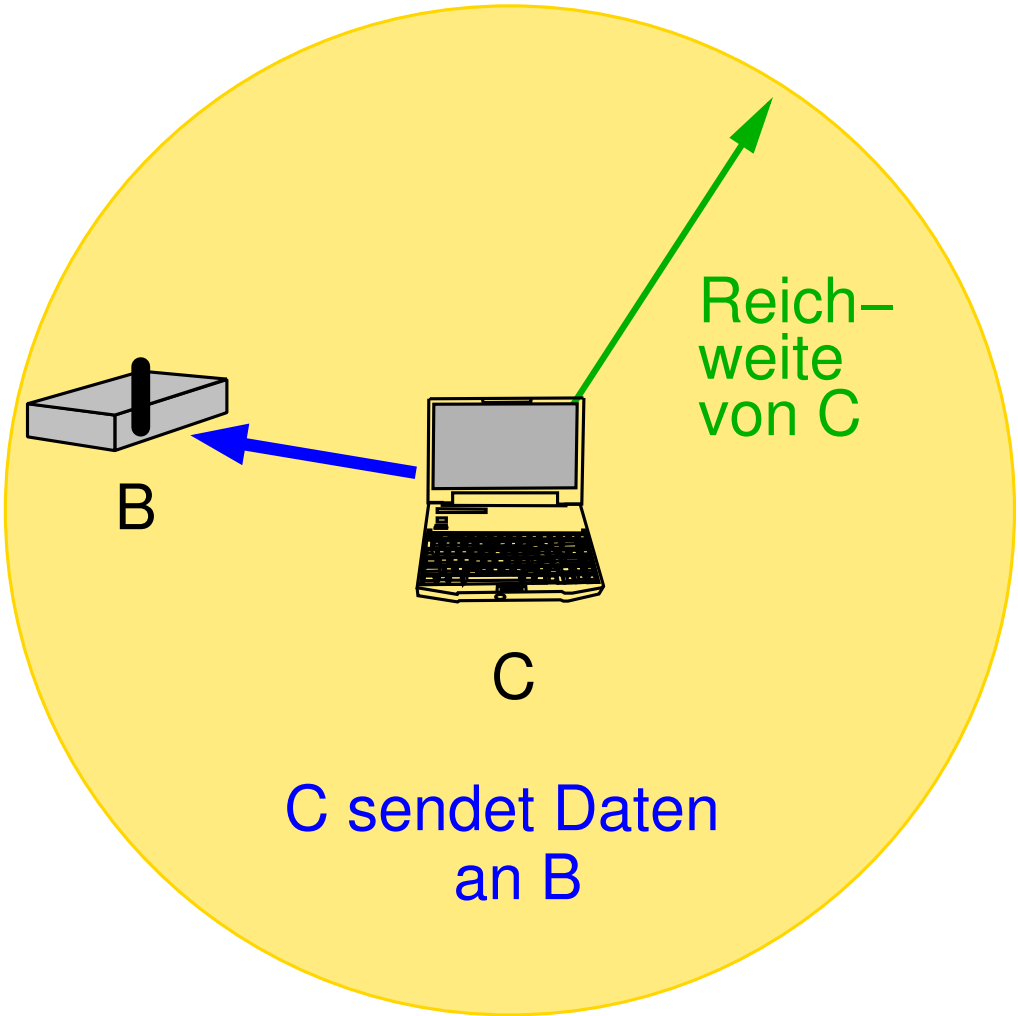
- *Carrier Sense Multiple Access / Collision Avoidance*
  - *Avoidance* heißt hier: **möglichst** vermeiden
  - Kollisionen sind aber immer noch möglich
- Vorgehen im Prinzip wie bei CSMA/CD:
  - Abhören des Mediums; senden, falls Medium frei
- Unterschiede:
  - keine Kollisionserkennung beim Senden
    - Empfänger muß jeden Frame bestätigen (ACK-Frame)
  - vor dem Senden muß das Netz immer mindestens für eine bestimmte Zeit abgehört und als frei erkannt werden:
    - IFS (*Interframe Spacing*)
  - Medium belegt: zufällige Wartezeit zur Kollisionsvermeidung

Interframe Spacing (IFS)

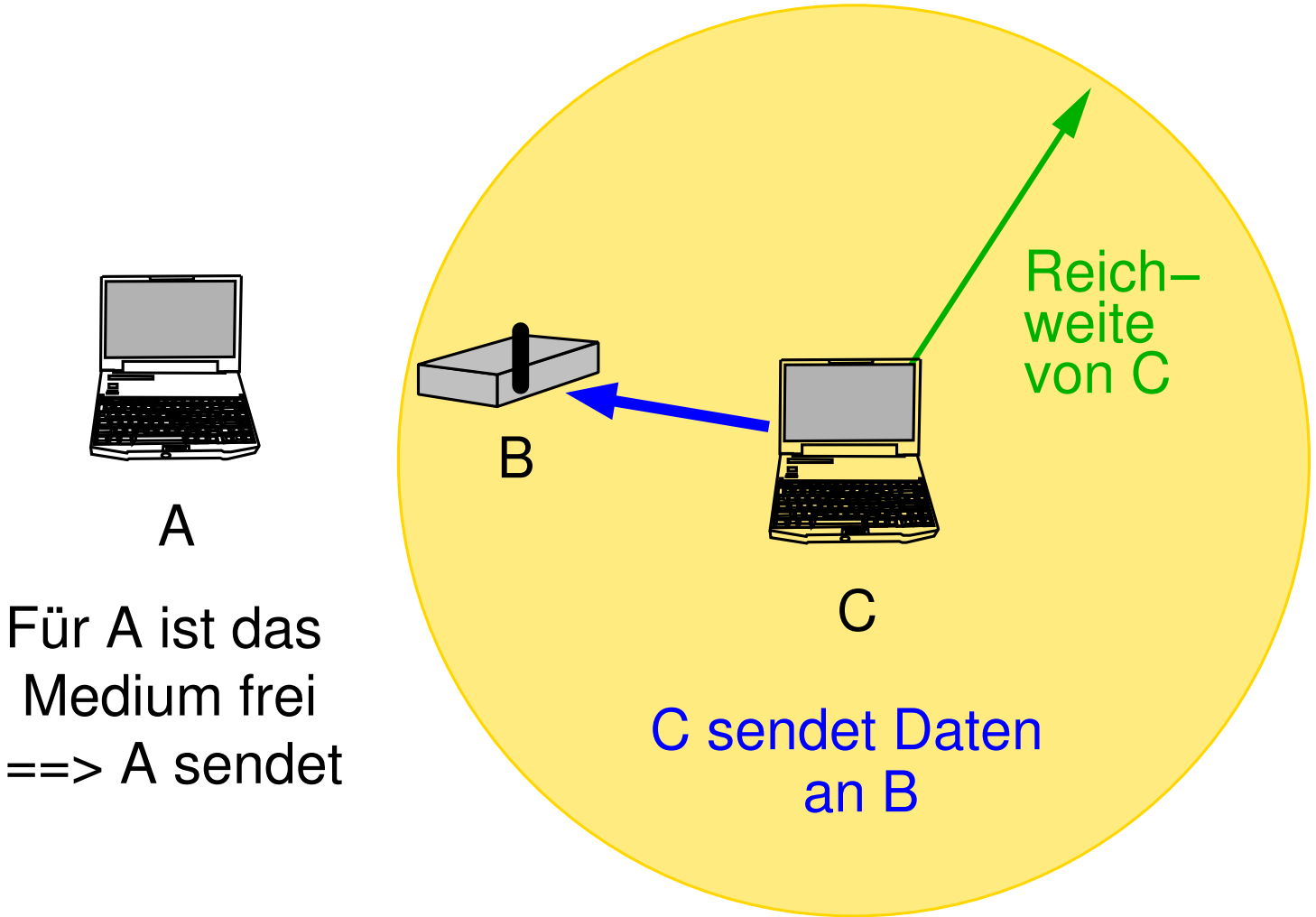
- Gibt an, wie lange eine Station das Medium mindestens als frei erkennen muß, bevor sie senden darf
- Unterschiedliche IFS-Zeiten für verschiedene Frame-Typen
  - damit: Realisierung unterschiedlicher Prioritäten



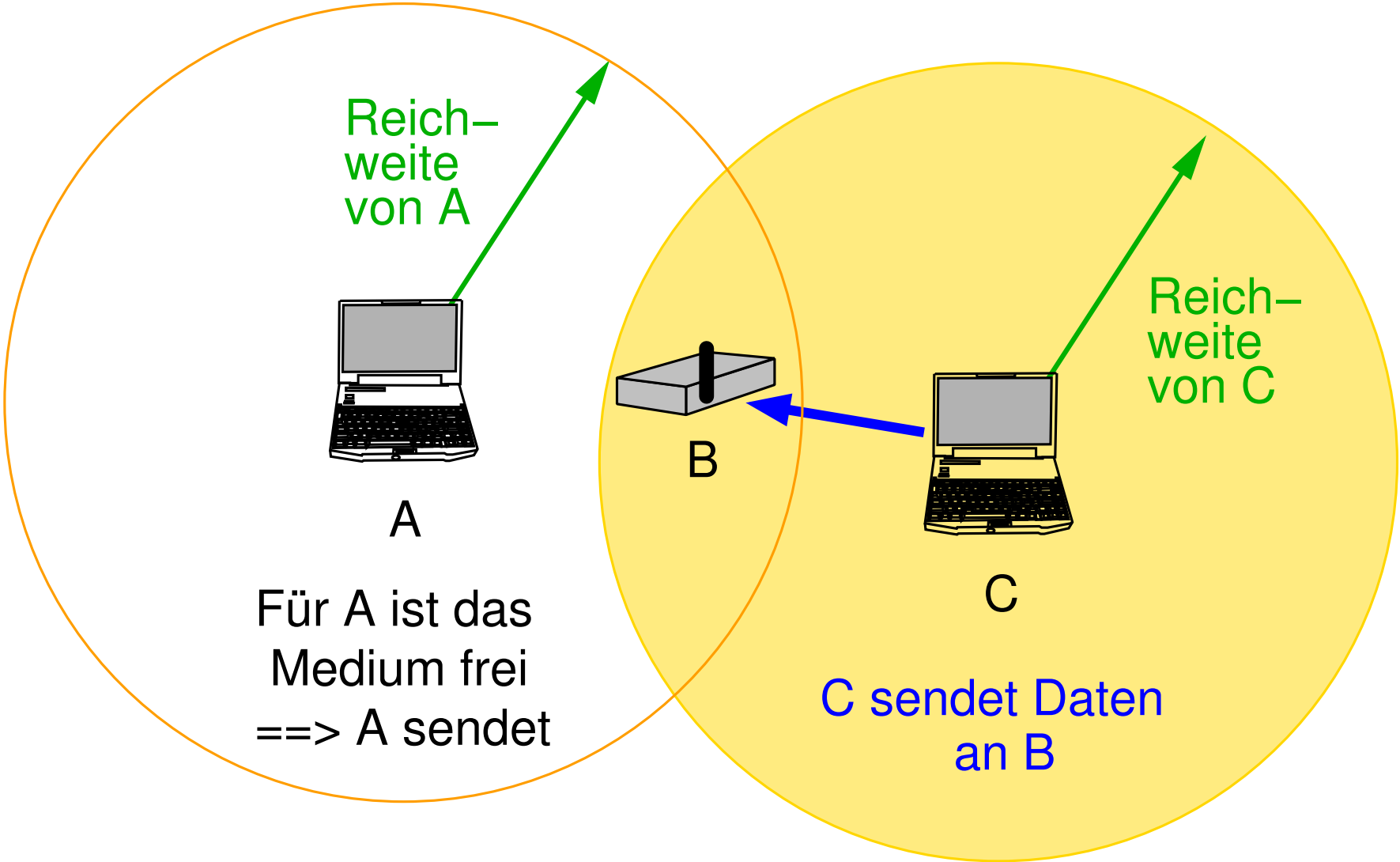
Das Hidden-Station-Problem



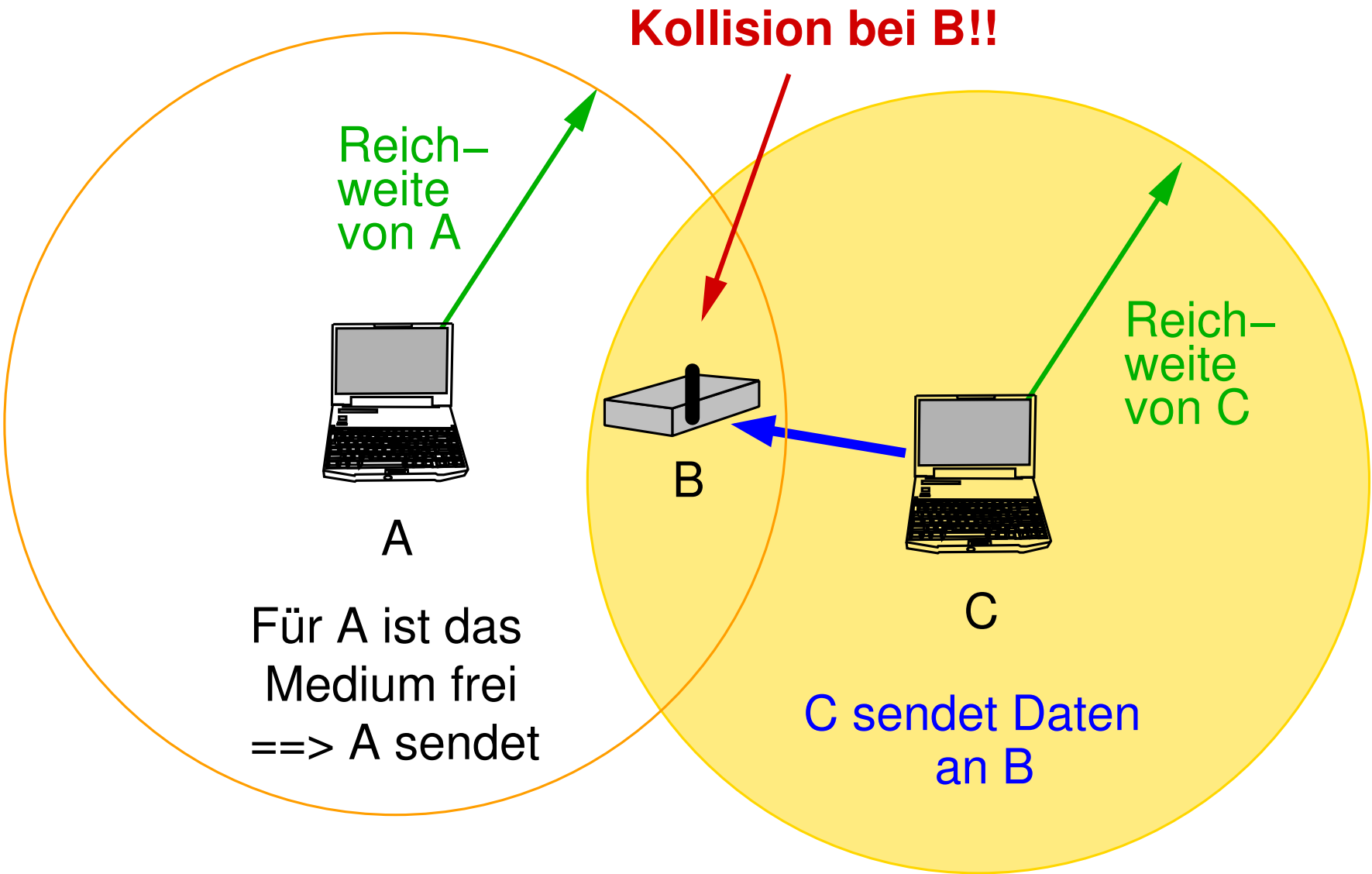
Das Hidden-Station-Problem



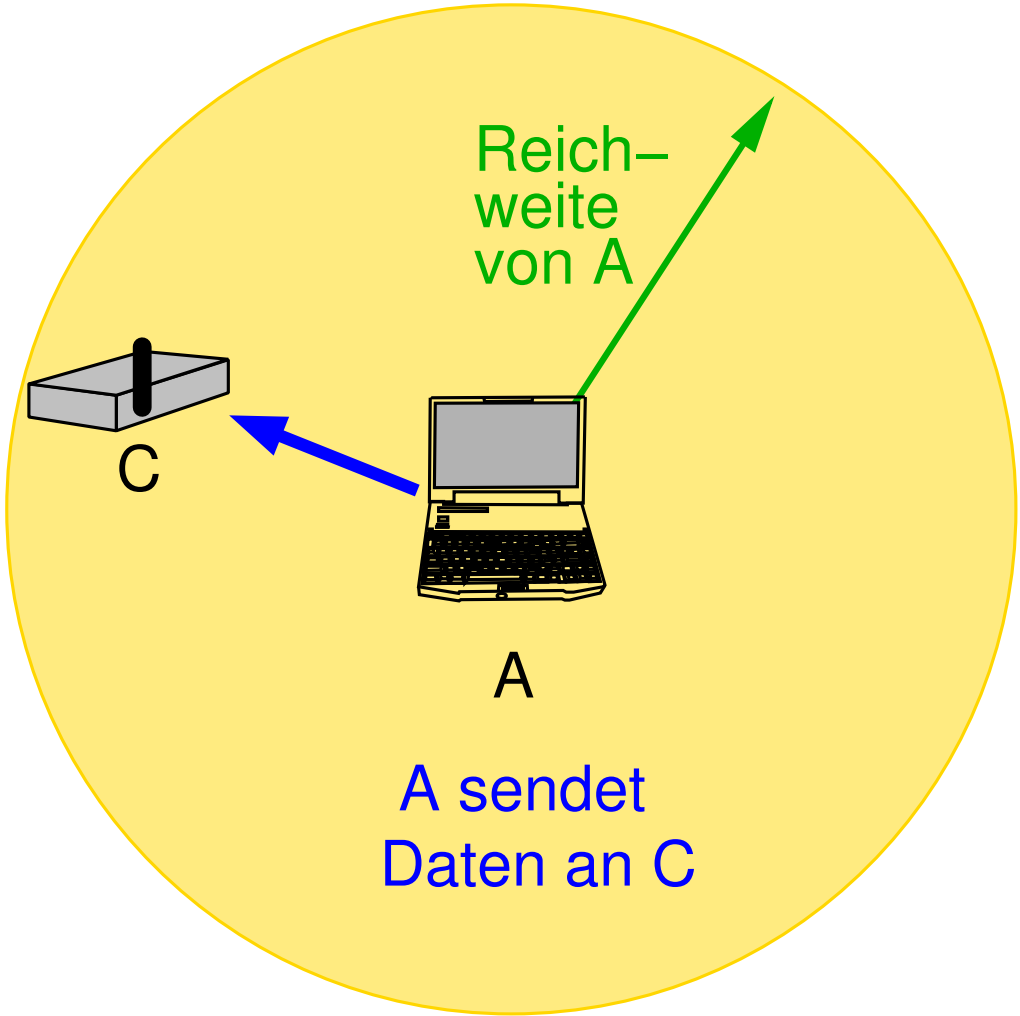
Das Hidden-Station-Problem



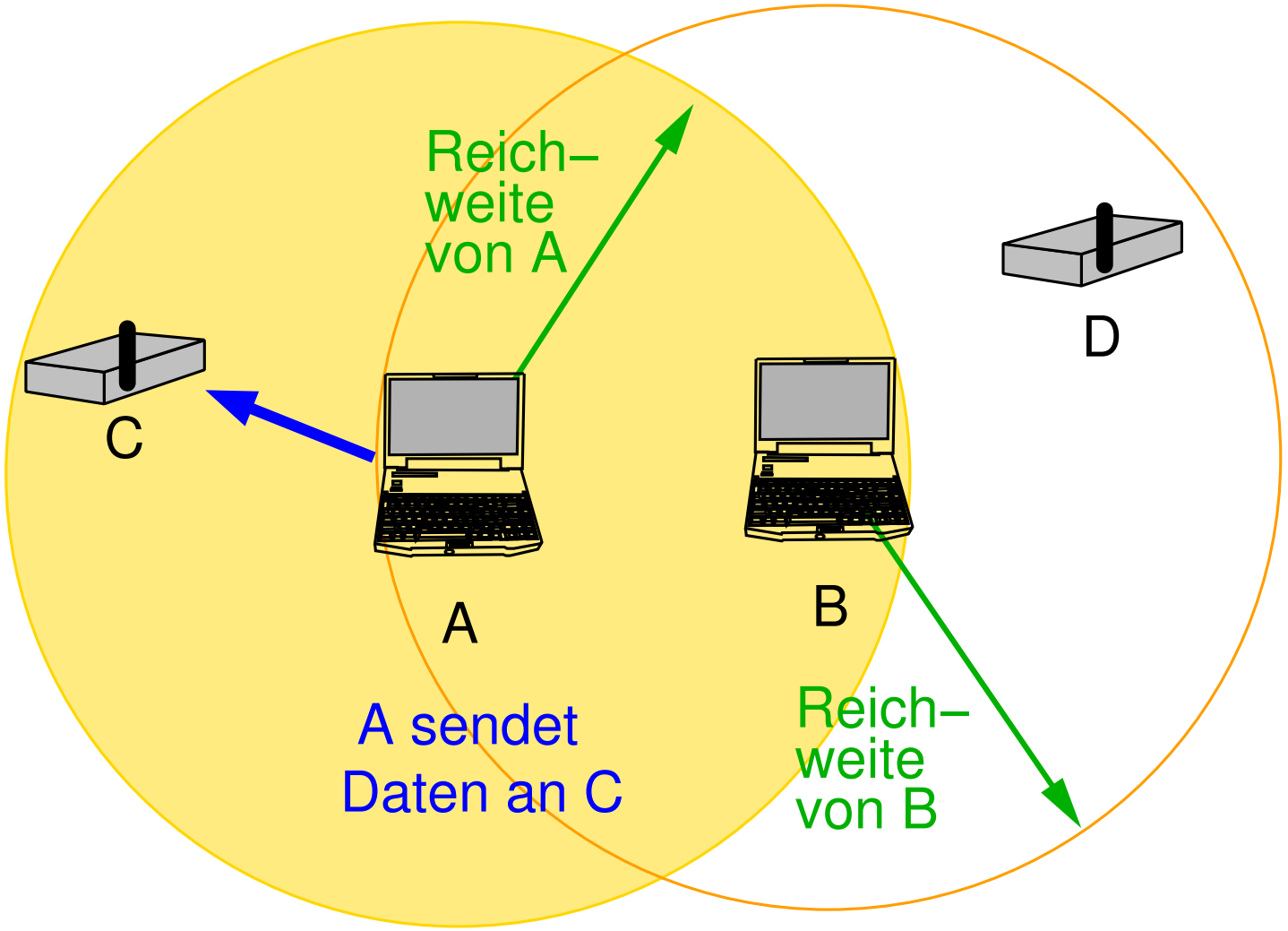
Das Hidden-Station-Problem



Das Exposed-Station-Problem



Das Exposed-Station-Problem



B will an D senden, glaubt aber, daß dies eine Kollision hervorruft

Das MACA - Protokoll (*Multiple Access, Collision Avoidance*)

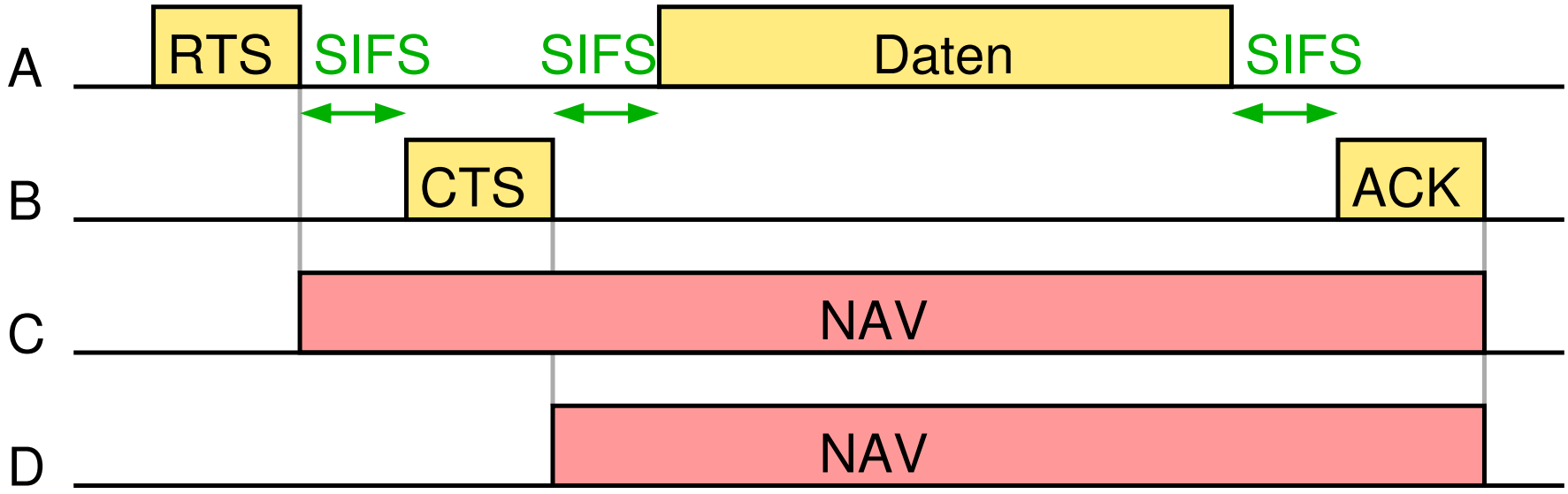
1. Sender sendet RTS (*Request To Send*) an Empfänger
2. Empfänger antwortet mit CTS (*Clear To Send*)
   - ➥ CTS-Frame enthält Dauer der Übertragung
3. Sender sendet Daten

- ➥ Wer RTS hört, sendet nicht, bis CTS übertragen sein sollte
  - ➥ Zeit ergibt sich aus Framelängen und Signallaufzeit
- ➥ Wer CTS hört, sendet nicht vor Ablauf der Übertragungsdauer
  - ➥ löst *Hidden Station* Problem
- ➥ Wer CTS nicht hört, kann gleichzeitig senden
  - ➥ löst *Exposed Station* Problem
- ➥ Wenn zwei RTS kollidieren, kommt kein CTS $\Rightarrow$ *Backoff*

MACAW - Erweiterung von MACA für WLAN

- Einführung von ACKs, um Neuübertragung durch Sicherungsschicht zu ermöglichen
  - schneller, da kürzere Timeouts als z.B. bei TCP
- Modifikationen gegenüber MACA:
  - Empfänger bestätigt Empfang der Daten mit ACK
  - Station, die RTS hört, darf nicht senden, bis ACK übertragen wurde
    - Übertragung könnte mit ACK kollidieren
  - auch RTS-Frame enthält Dauer der Übertragung
- 802.11 verwendet MACAW nur zum Versenden längerer Frames
  - in der Regel auch erst nach mehrfacher Kollision
  - für kurze Frames: immer einfaches CSMA/CA

MACAW - Beispiel

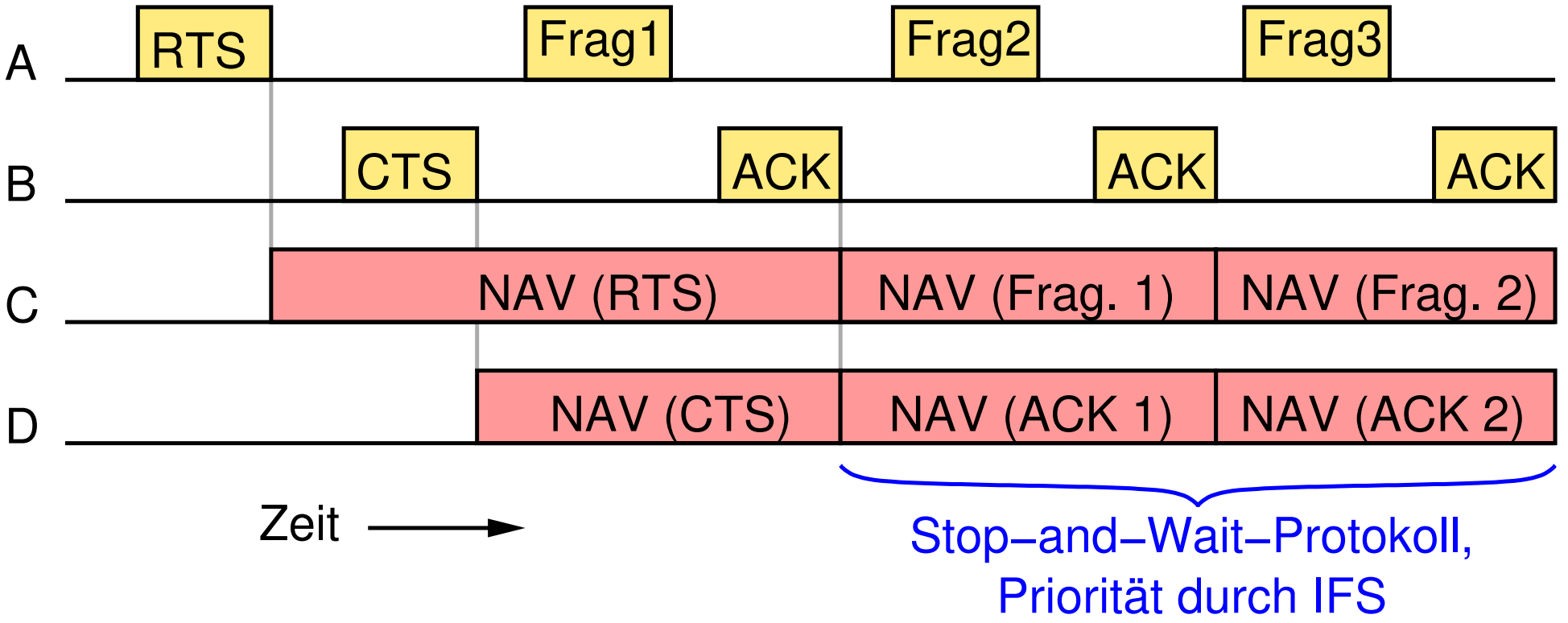


B und C sind in Reichweite von A, D nicht

NAV: *Network Allocation Vector* (Netz ist belegt, kein Senden)

Fragmentierung

- Frames können in mehreren Fragmenten übertragen werden
  - erhöht Effizienz bei hoher Bitfehlerrate

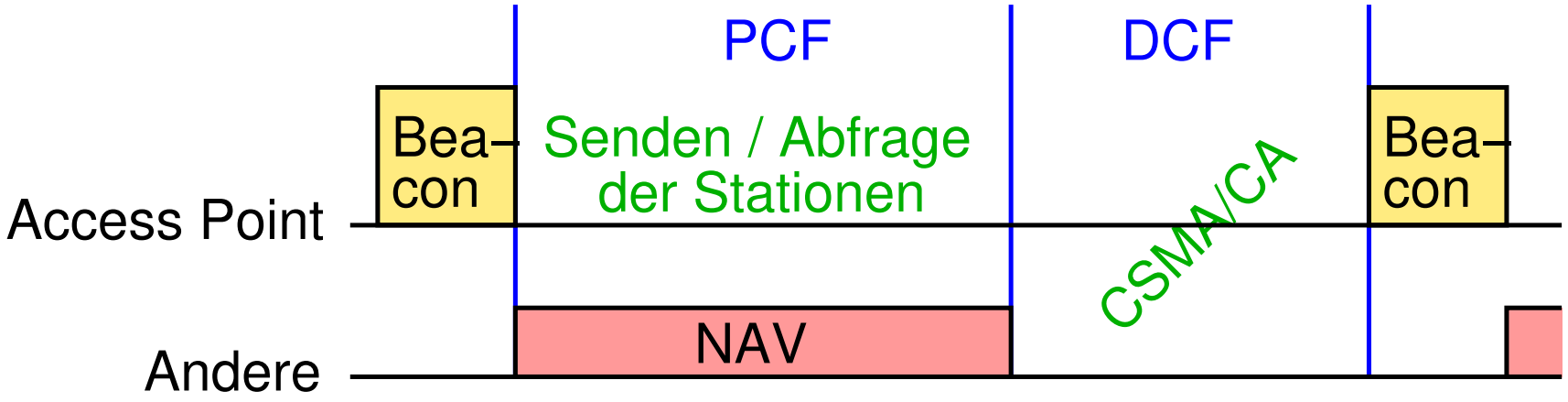


Koexistenz von 802.11b und 802.11g

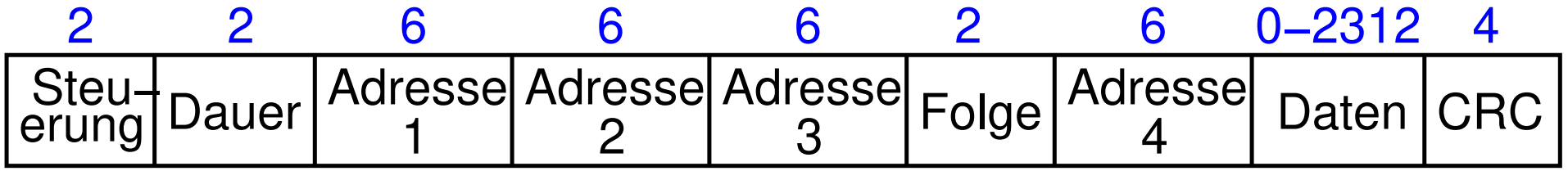
- Problem: 802.11b-Station erkennt nicht, daß 802.11g-Station sendet
- Lösung: *Protection*-Mechanismus
  - wird vom *Access-Point* aktiviert, wenn dieser eine 802.11b-Station erkennt
- Zwei Verfahren:
  - **CTS-to-Self:** 802.11g-Station sendet vor der eigentlichen Übertragung ein CTS mit DSSS, das das Medium reserviert
  - **RTS/CTS:** RTS, CTS und ACK werden mit DSSS übertragen, nur Datenframes werden mit OFDM gesendet
- Nachteil: Nutzdatenrate sinkt deutlich ($\sim$ 10-15 Mbit/s)

PCF: TDMA (*Time Division Multiple Access*)

- *Access Point* sendet regelmäßig *Beacon*-Frame als Broadcast
  - enthält verschiedene Systemparameter
  - kann Medium für bestimmte Zeit reservieren (über NAV)
    - in dieser Zeit: Stationen, die sich für PCF angemeldet haben, werden vom *Access Point* einzeln abgefragt
  - danach: normaler DCF-Betrieb bis zum nächsten *Beacon*



Frame-Format (für Daten-Frames)



- **Steuerung**: Frame-Typ, Frame von/an *Distribution System*, Verschlüsselung, *Power Management*, ...
- **Dauer**: für Belegung des Kanals über NAV
- **Adresse 1-4**: IEEE 802 MAC-Adressen
  - Quell- und Ziel-Rechner
  - BSS-ID bzw. Quell- und Ziel-*Access-Point*
- **Folge**: Numerierung von Fragmenten

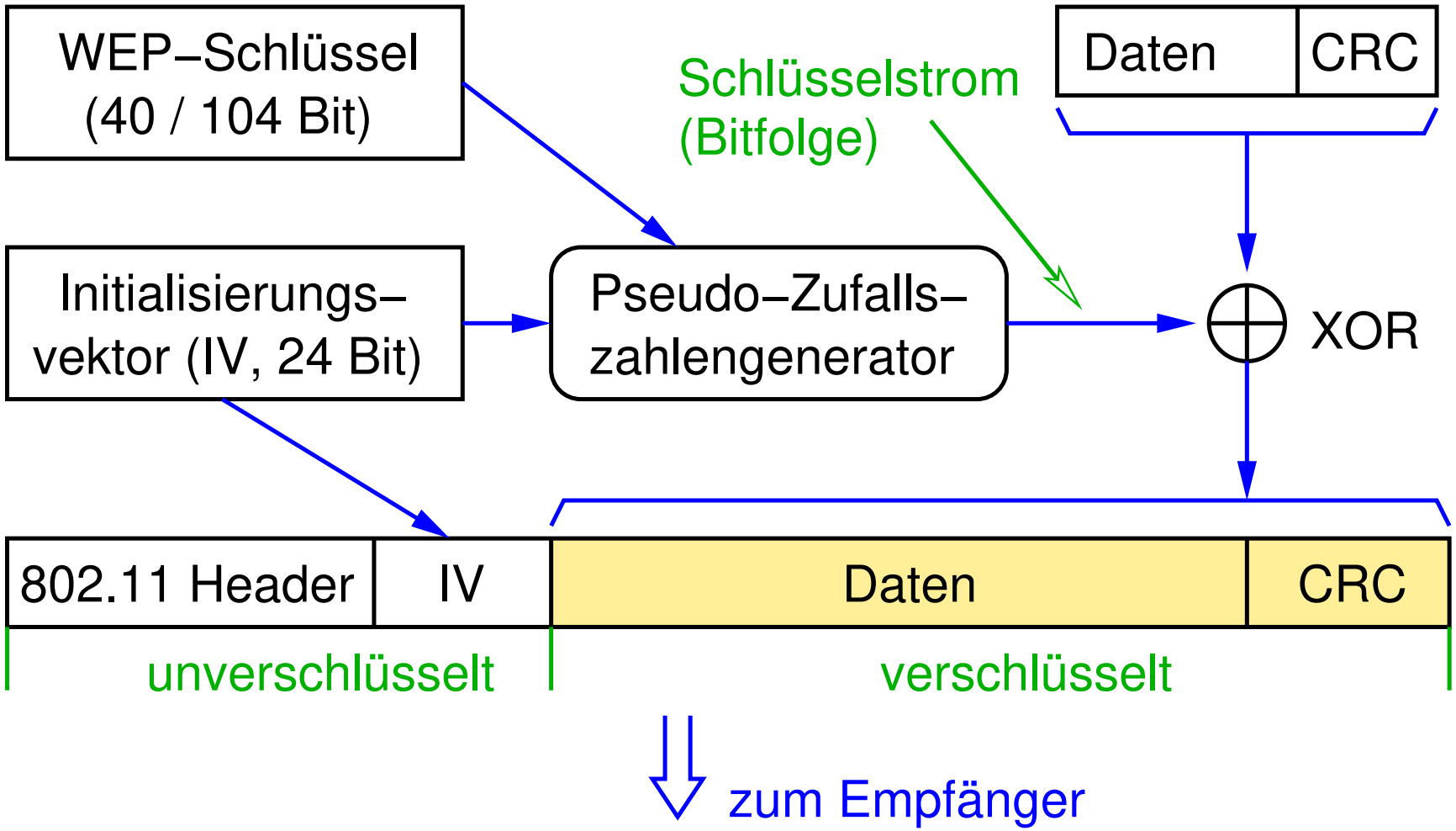
Sicherheitsmechanismen

- ESSID (*Extended Service Set IDentifier*): Name des Netzes
  - muß i.a. zum Anmelden an *Access Point* bekannt sein
  - wird i.d.R. vom *Access Point* im *Beacon*-Frame mitgesendet
  - viele WLAN-Karten akzeptieren auch „*any*“
- Authentifizierung über MAC-Adresse
  - Basisstation hat Liste der erlaubten MAC-Adressen
  - viele WLAN-Karten erlauben Änderung der MAC-Adresse!
- Verschlüsselung
  - WEP (*Wire Equivalent Privacy*, IEEE 802.11)
    - 40 (bzw. 104) Bit Schlüssel, veraltet
  - WPA und WPA2 (*Wi-Fi Protected Access*, IEEE 802.11i)
    - deutlich bessere Sicherheit als WEP

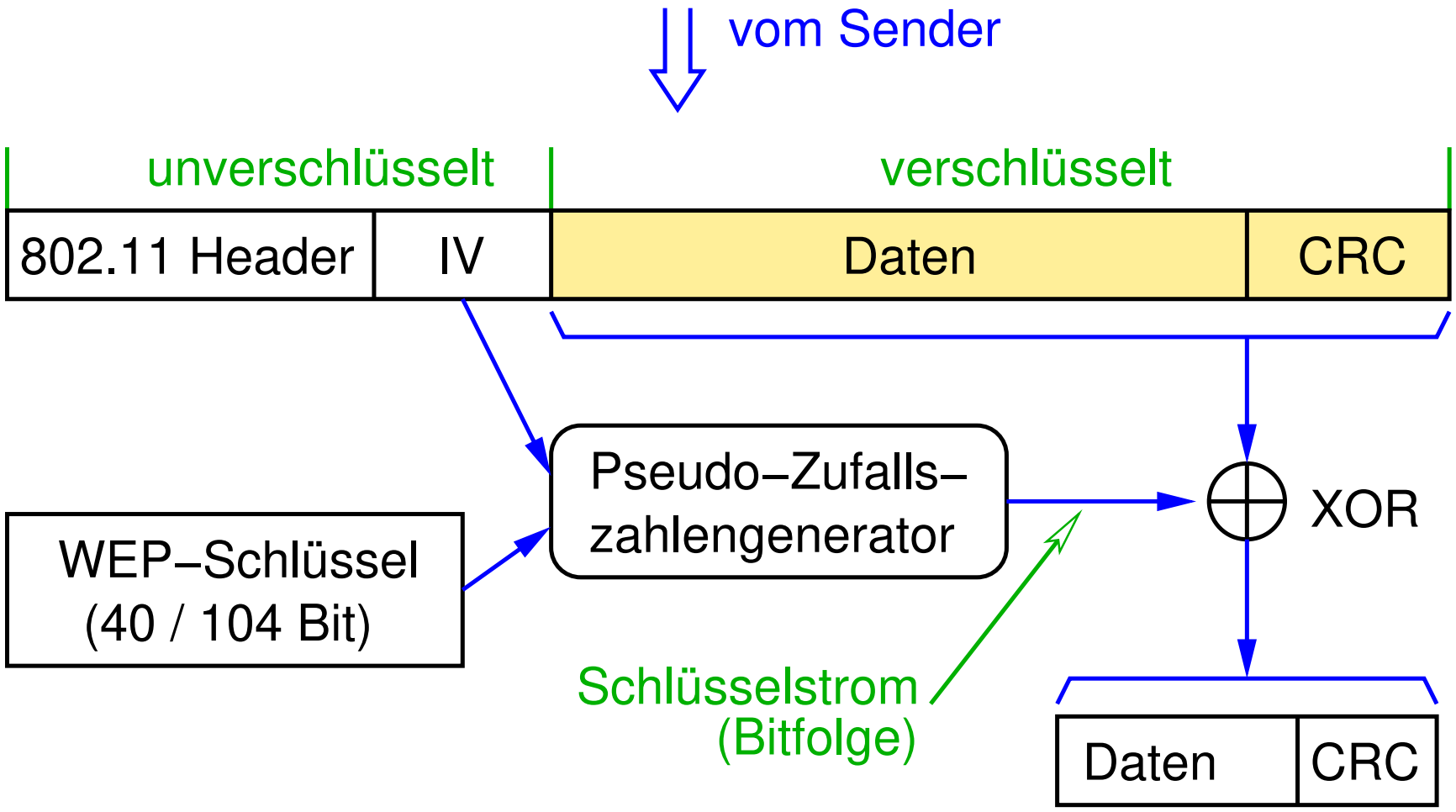
WEP: Funktionsweise

- Basis: symmetrische Verschlüsselung mit RC4 Stromchiffre
  - Daten werden mit Pseudozufalls-Bitfolge EXOR-verknüpft
  - Bitfolge kann aus Schlüssel und Initialisierungsvektor (IV) eindeutig bestimmt werden
  - Schlüssel (40 bzw. 104 Bit) muß allen Stationen bekannt sein
  - Initialisierungsvektor wird für jede Übertragung neu gewählt und (unverschlüsselt) mitübertragen
- Authentifizierung der Teilnehmer durch *Challenge-Response* - Protokoll

WEP-Verschlüsselung beim Sender



WEP-Entschlüsselung beim Empfänger



WEP: Schwachstellen

- Verschlüsselung ist angreifbar (Problem: Schlüsselerzeugung)
- Verschlüsselung erfolgt immer direkt mit WEP-Schlüssel
  - macht Schlüssel durch Kryptoanalyse angreifbar
- CRC ist bezüglich $\oplus$ linear $\Rightarrow$ Angreifer kann nach Manipulation der Daten verschlüsselten CRC neu berechnen
- IV ist zu kurz: wiederholt sich nach wenigen Stunden
  - wiederholte Verwendung desselben Schlüsselstroms
  - Schlüsselstrom kann durch Klartextangriff ermittelt werden
    - *Challenge-Response* - Protokoll bei Authentifizierung!
- WEP ist unsicher! $\Rightarrow$ WPA bzw. WPA2 verwenden!!!

IEEE 802.11i: verbesserte Sicherheitsstandards

- Bessere Verschlüsselung als WEP, sichere Integritätsprüfung
  - Ziel: schrittweiser Übergang unter Weiterverwendung vorhandener Hardware
  - daher Übergangslösung über Firmware-Update
    - ersetze WEP-Verschlüsselung durch TKIP
    - zusätzlich: Integritätsprüfung über Hash-Funktion
      - MIC: *Message Integrity Check*
  - endgültige Lösung (erfordert neue Hardware)
    - AES-CCMP (*Advanced Encryption Standard*)
- Verbesserte Authentifizierung (inkl. Schlüsselmanagement)
  - über Authentifizierungsserver (IEEE 802.1X, EAP)
  - oder über *Pre-Shared Key* (PSK)

WPA, WPA2: Quasi-Standard der Wi-Fi Alliance

- Die IEEE-Standardisierung dauerte zu lange ...
  - WPA entspricht (in etwa) Übergangslösung von IEEE 802.11i
  - WPA2 entspricht (in etwa) IEEE 802.11i
- Jeweils zwei Modi: *Personal* und *Enterprise*

WPA-Variante		**WPA**	**WPA2**
Personal-Mode	Authentifizierung	PSK	PSK
	Verschlüsselung	TKIP/MIC	AES-CCMP
Enterprise-Mode	Authentifizierung	802.1X/EAP	802.1X/EAP
	Verschlüsselung	TKIP/MIC	AES-CCMP

Authentifizierung mit 802.1X und EAP

- 802.1X: Authentifizierung über einen zentralen Server
  - RADIUS-Server (*Remote Authentication Dial-In User Service*)
  - Vorteil: zentrale Administration des Zugangs

- EAP: *Extensible Authentication Protocol* (RFC 2284)
  - zum Austausch der Authentifizierungsnachrichten

Ablauf von Authentifizierung und Schlüsselaustausch

- Client muß gegenüber Authentifizierungsserver seine Identität nachweisen
  - Challenge/Response, z.B. mit Paßwort oder X.509 Zertifikat
- Dabei gleichzeitig: Aushandlung eines Schlüssels
  - PMK: *Pairwise Master Key*
  - wird vom Server auch an *Access Point* geschickt
- Client und *Access Point* bilden aus PMK einen nur ihnen bekannten Schlüssel für diese Sitzung
  - PTK (*Pairwise Transient Key*), für Punkt-zu-Punkt-Kommunik.
- *Access Point* sendet an Client einen Gruppenschlüssel
  - GTK (*Group Transient Key*), verschlüsselt mit PTK
  - für Broadcast- und Multicast-Kommunikation

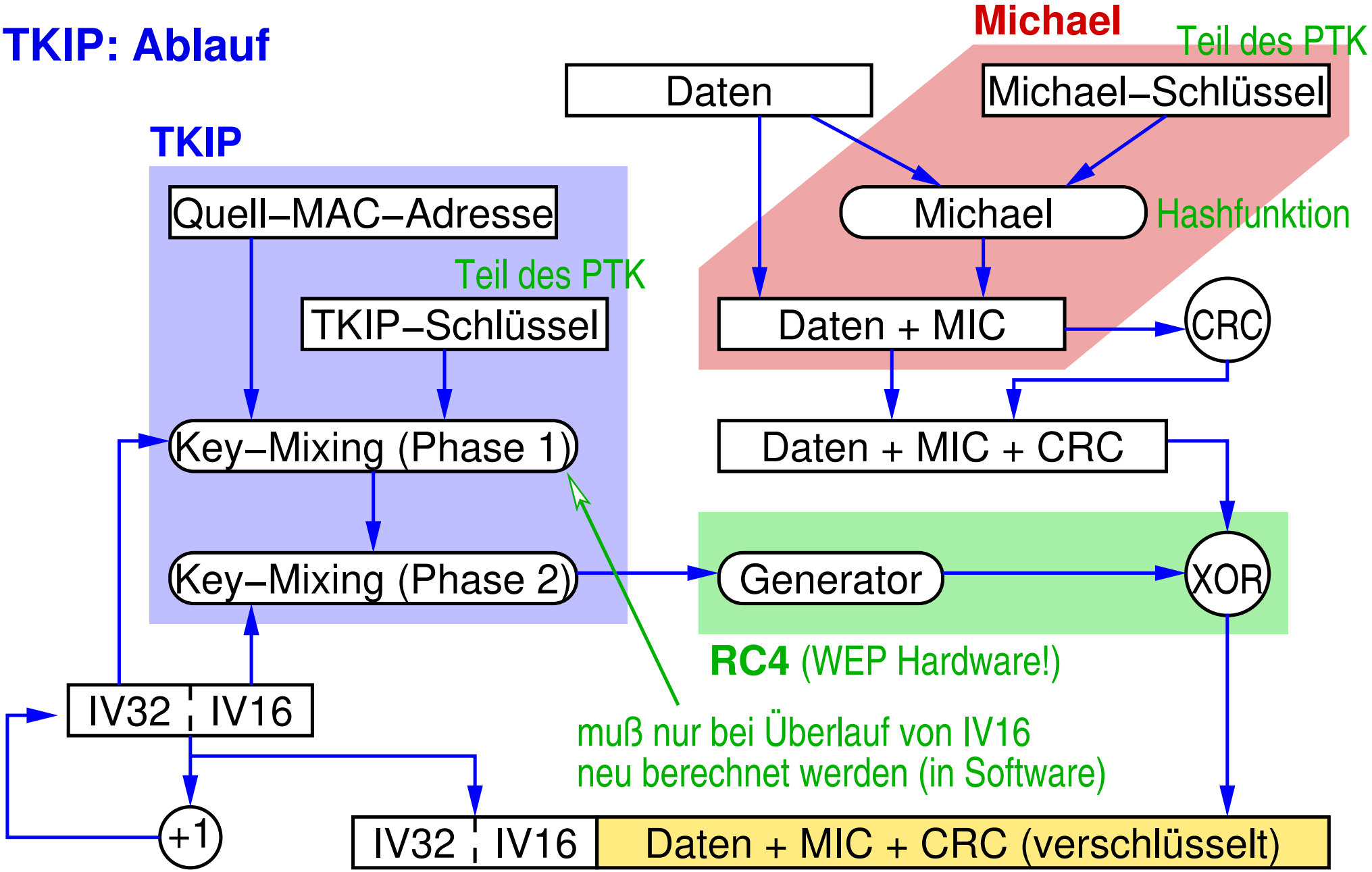
Authentifizierung mit PSK (*Pre-Shared Key*)

- PSK wird über Hashfunktion aus Passphrase und SSID gebildet
  - Passphrase wird auf allen Stationen manuell eingetragen
- PSK übernimmt die Rolle des PMK bei Auth. über 802.1X/EAP
  - d.h., Client und *Access Point* bilden aus PSK den PTK
    - unter Einbeziehung von MAC-Adresse und Zufallszahlen
- Nur, wenn Client und *Access Point* denselben PSK besitzen, erhalten sie denselben PTK und können kommunizieren
- PSK wird nicht für die Kommunikation verwendet
  - weniger Angriffspotential, um PSK zu ermitteln
  - trotzdem ist bei Kenntnis des PSK ein Entschlüsseln der Kommunikation anderer Clients möglich
    - Voraussetzung: Schlüsselaustauch wird abgehört

TKIP *Temporary Key Integrity Protocol*

- Übergangslösung für Verschlüsselung
  - Verwendung der WEP-Hardware mit neuer Software
- RC4 Verschlüsselung wie bei WEP
- Unterschiede:
  - Initialisierungsvektor (IV) mit 48 Bit
  - IV wird nach jedem Paket inkrementiert, Empfänger prüft Sequenz (Replayschutz)
  - 128 Bit langer TKIP-Schlüssel (Teil des PTK)
    - unterschiedliche Schlüssel für jeden Client
    - zusätzlich: Quell-MAC-Adresse fließt in RC4-Seed mit ein
  - Integritätsschutz (MIC): Hashfunktion mit Schlüssel (Michael)
    - getrennte Schlüssel je Übertragungsrichtung

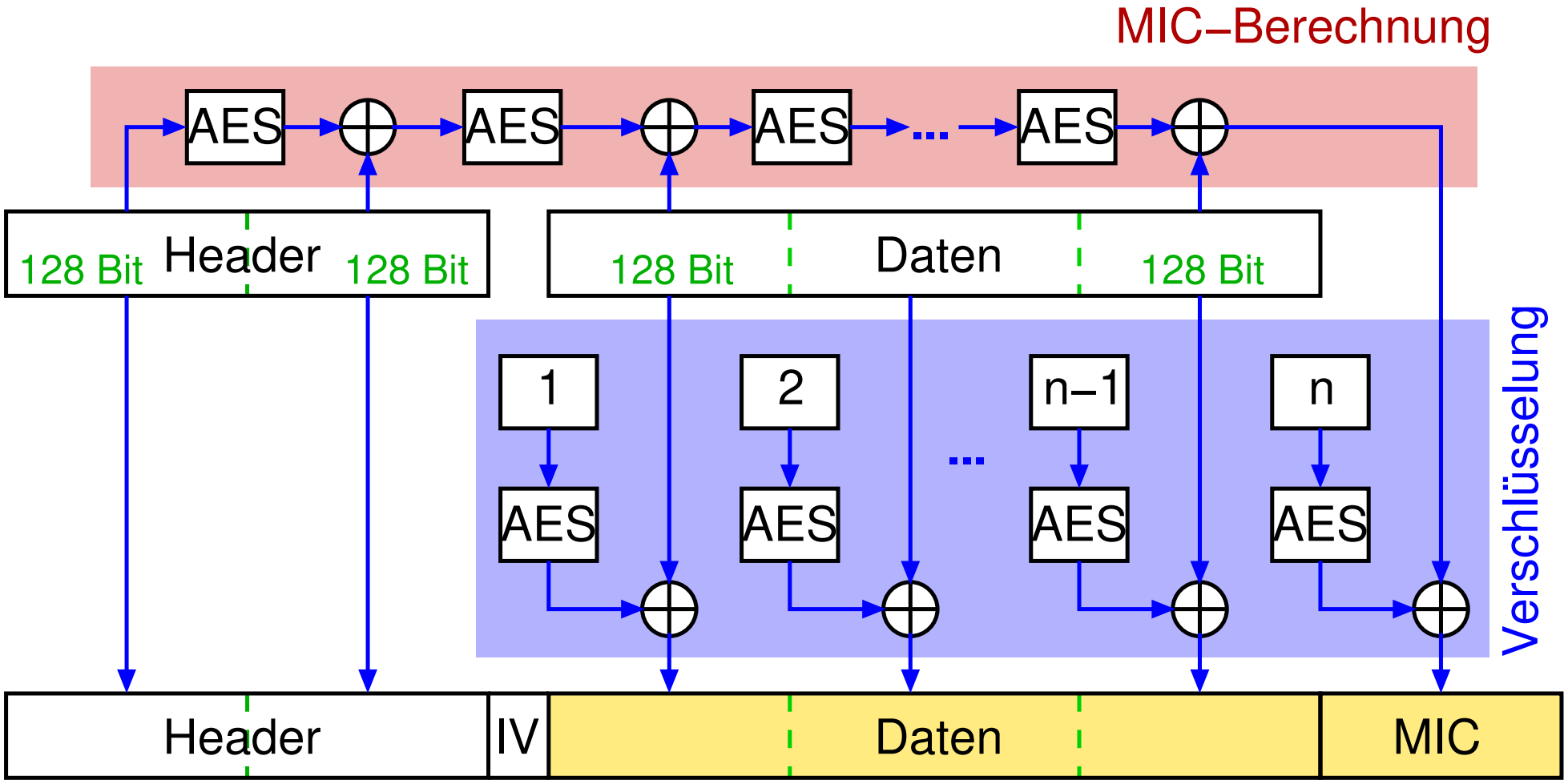
TKIP: Ablauf


Michael
Teil des PTK
Daten
Michael–Schlüssel
TKIP
Quell–MAC–Adresse
Michael
Hashfunktion
Teil des PTK
TKIP–Schlüssel
Daten + MIC
CRC
Key–Mixing (Phase 1)
Daten + MIC + CRC
Key–Mixing (Phase 2)
Generator
XOR
RC4 (WEP Hardware!)
IV32
IV16
muß nur bei Überlauf von IV16
neu berechnet werden (in Software)
+1
IV32
IV16
Daten + MIC + CRC (verschlüsselt)

AES-CCMP

- AES: vom NIST standardisiertes Verschlüsselungsverfahren
- AES-CCMP = AES *CTR/CBC-MAC Protocol*
  - AES im Zähler-Modus, MIC mittels *Cipher Block Chaining*
  - Integritätsprüfung (Datenteil + Teile des Headers) und Verschlüsselung (Datenteil + MIC)
    - ein gemeinsamer Schlüssel mit 128 Bit
  - benötigt neue Hardware
- 48 Bit Paketzähler mit Sequenzprüfung beim Empfänger
  - Sequenznummer geht mit Quell-MAC-Adresse in Verschlüsselung und Integritätsprüfung mit ein
  - Replayschutz

AES-CCMP: Ablauf (vereinfacht)



1 = Zähler

AES = AES–Verschlüsselung
(Quell–Mac und Sequenznummer IV gehen mit ein)

WPA / WPA2 / IEEE 802.11i: Fazit

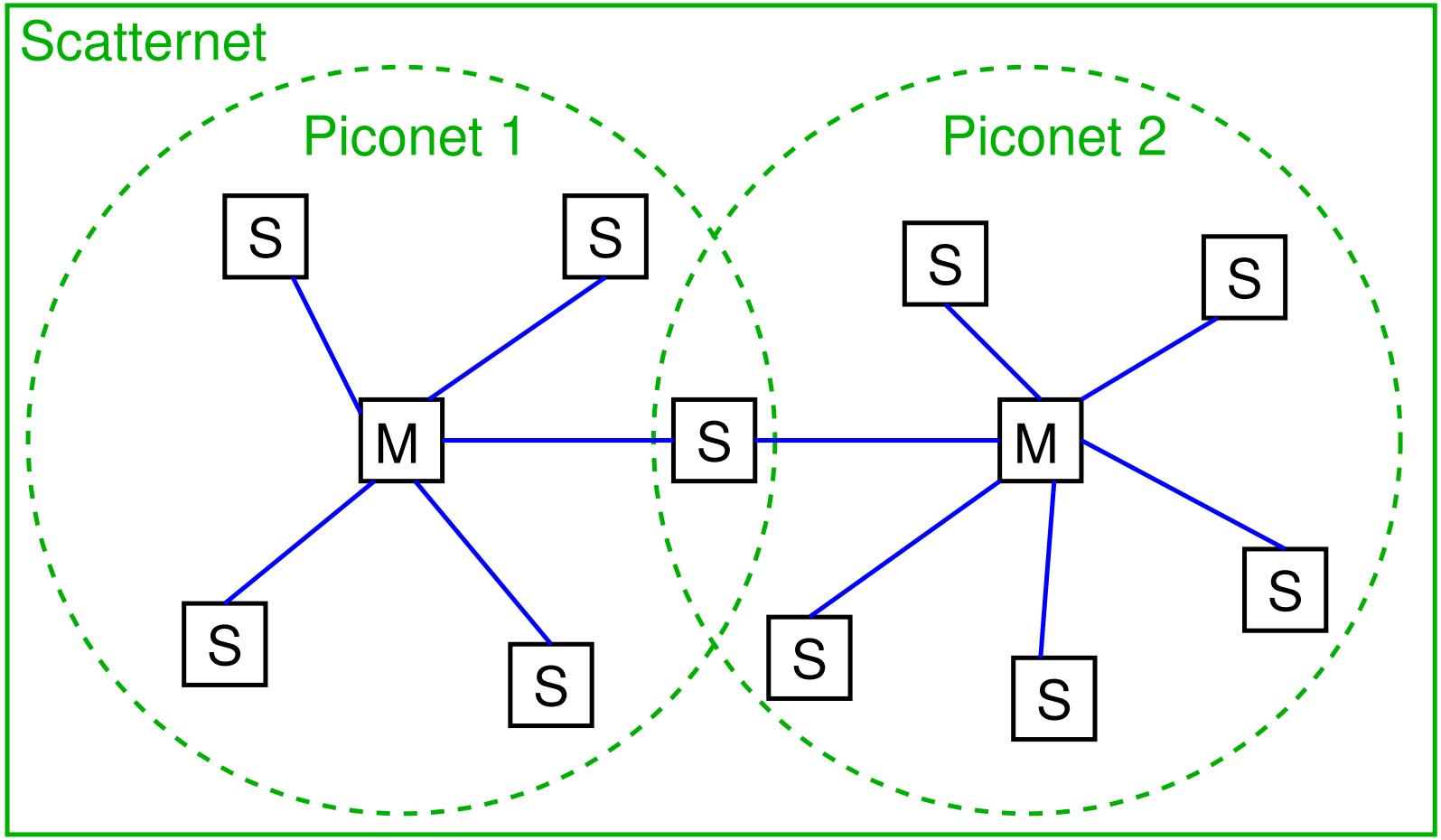
- AES-CCMP: Sicherheit nach Stand der Technik
- TKIP und Michael: Zwischenlösung für alte Hardware
  - bessere Verschlüsselung als WEP
    - paarweise geheime Schlüssel + Gruppenschlüssel, regelmäßiger Schlüsselwechsel, längerer IV
  - verbesserter Integritätsschutz (Hashwert mit Schlüssel)
  - Replayschutz (durch IV als Sequenznummer)
- PSK: für private / kleine WLANs
  - einfache Nutzung, aber Zugangsberechtigung nicht mehr ohne weiteres entziehbar
- IEEE 802.1X / EAP: für professionellen Einsatz
  - zentrale, flexible Benutzerverwaltung

3.2 Bluetooth

3.2.1 Bluetooth Classic

- Ursprüngliches Ziel: Verbindung von Mobiltelefonen mit anderen Geräten (PDA, ...)
  - geringer Stromverbrauch ist wesentlich
  - geringe Reichweite (10 m)
- Definition durch Gruppe mehrerer Unternehmen (1994 -)
  - untere Schichten in IEEE Standard 802.15 übernommen
- Bluetooth definiert Protokollstapel bis zur Anwendungsschicht
  - Zusammenarbeit der Geräte auf Anwendungsebene!
  - Profile für verschiedene Anwendungsbereiche
- benannt nach König Harald II. Blaatand (940-981)
  - vereinte Dänemark und Norwegen

Architektur eines Bluetooth-Netzes

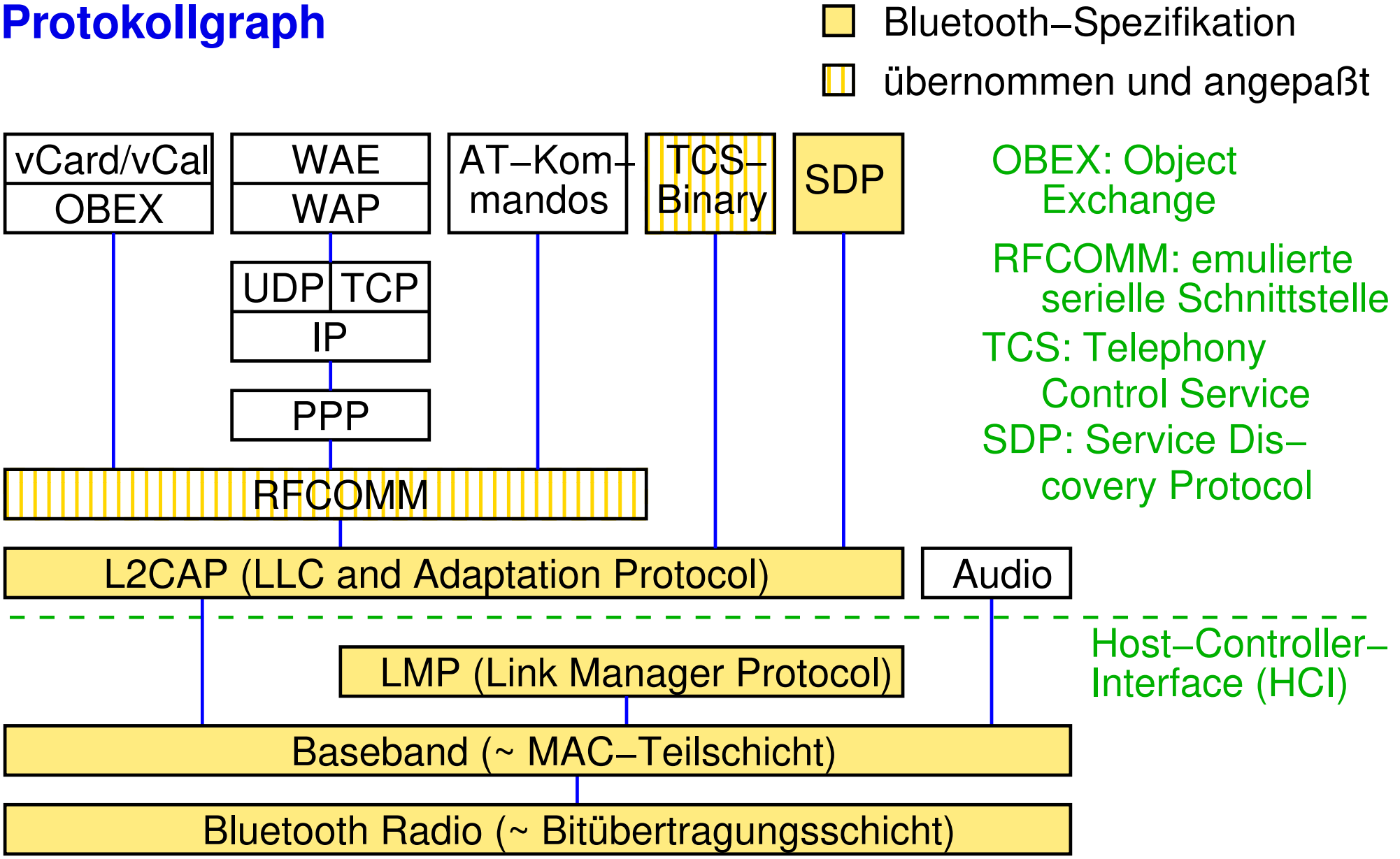


M: Master S: Slave

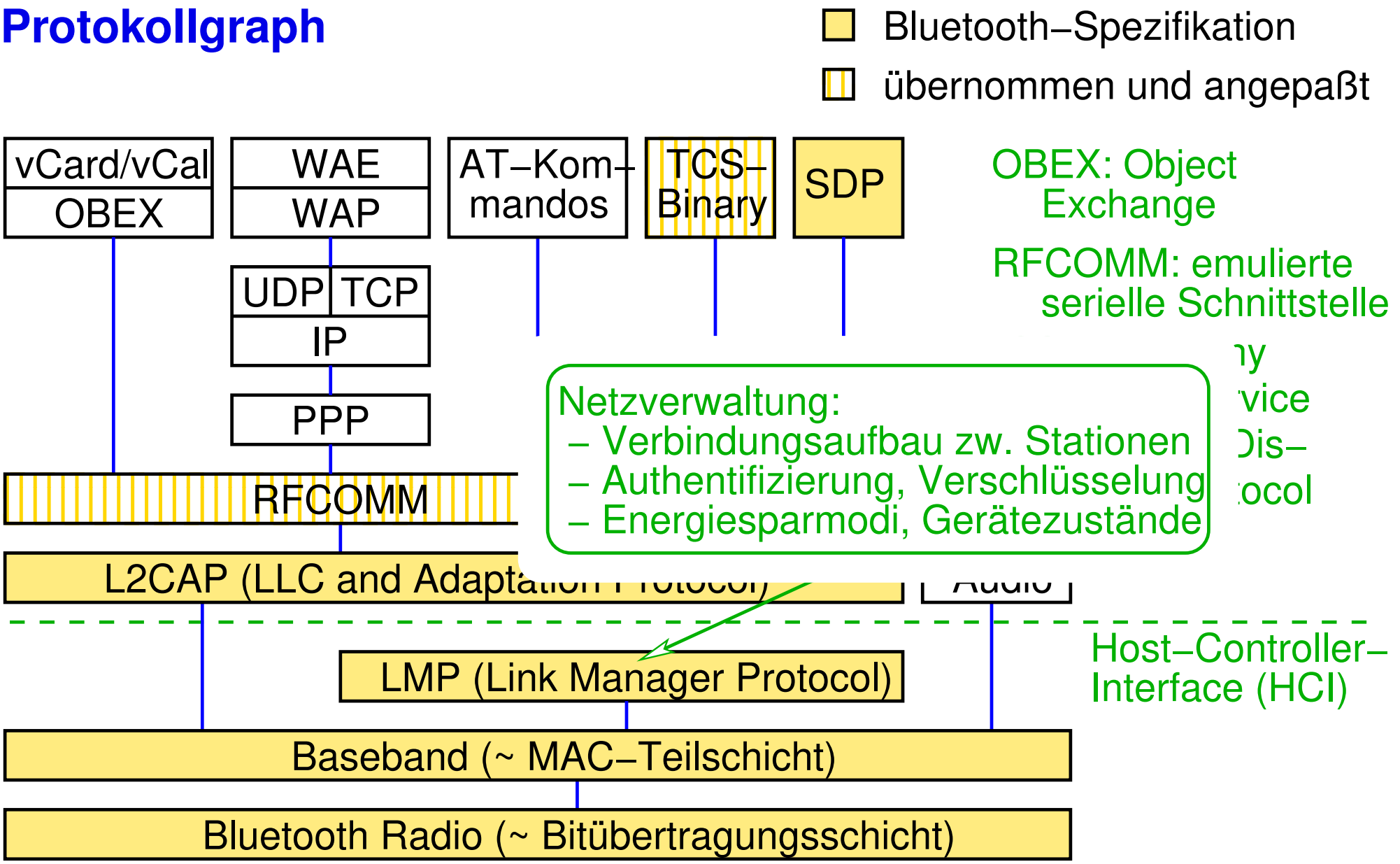
Architektur eines Bluetooth-Netzes ...

- Grundstruktur: Piconet
  - ein Master, bis zu 7 aktive Slaves
  - zusätzlich bis zu 255 „geparkte“ Slaves (Stromspar-Modus)
  - Medienzugang vollständig durch Master gesteuert (Zeitmultiplex)
- Mehrere Piconets können zu Scatternet verbunden werden
  - Verbindung über gemeinsamen Slave-Knoten als Bridge

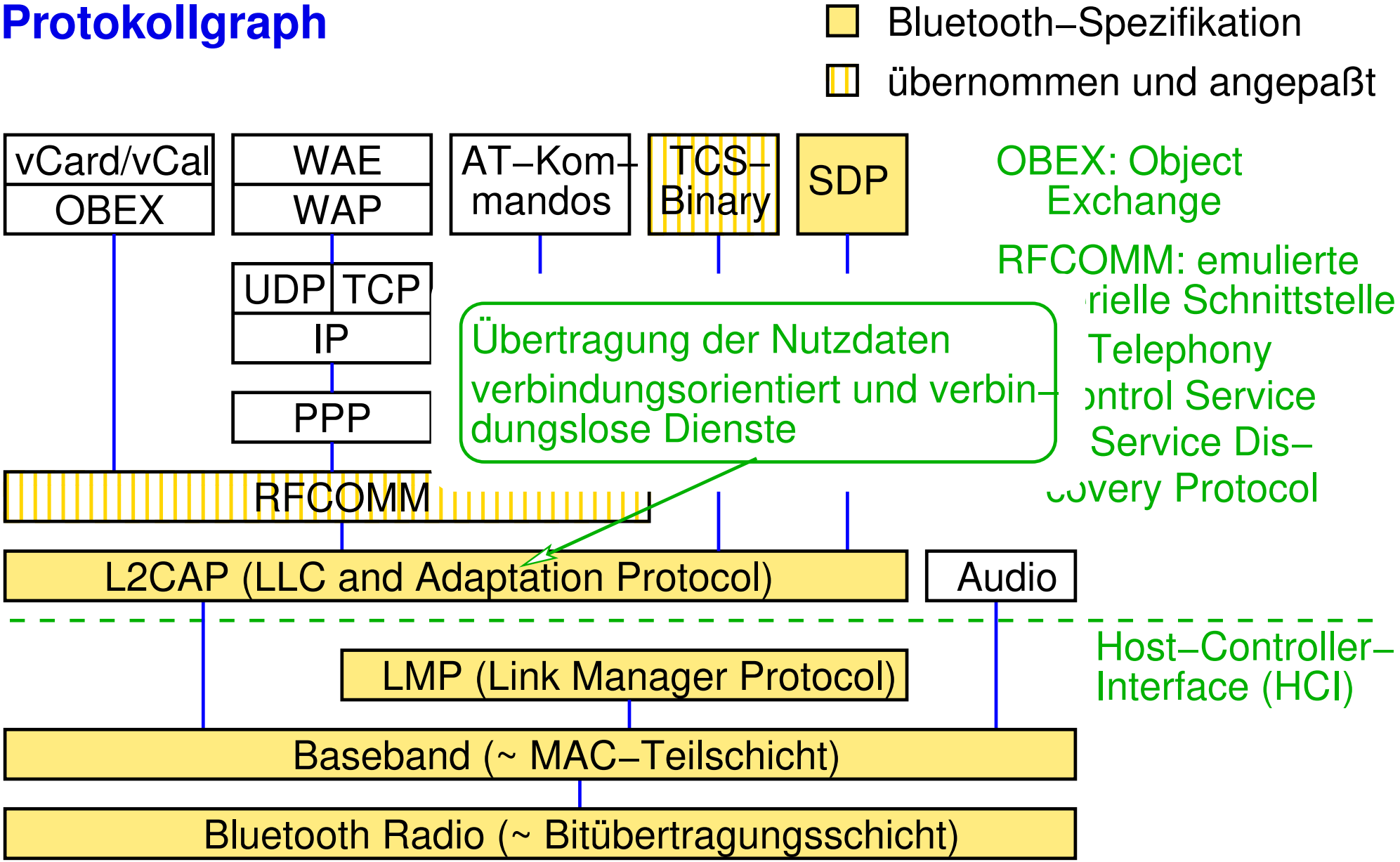
Protokollgraph



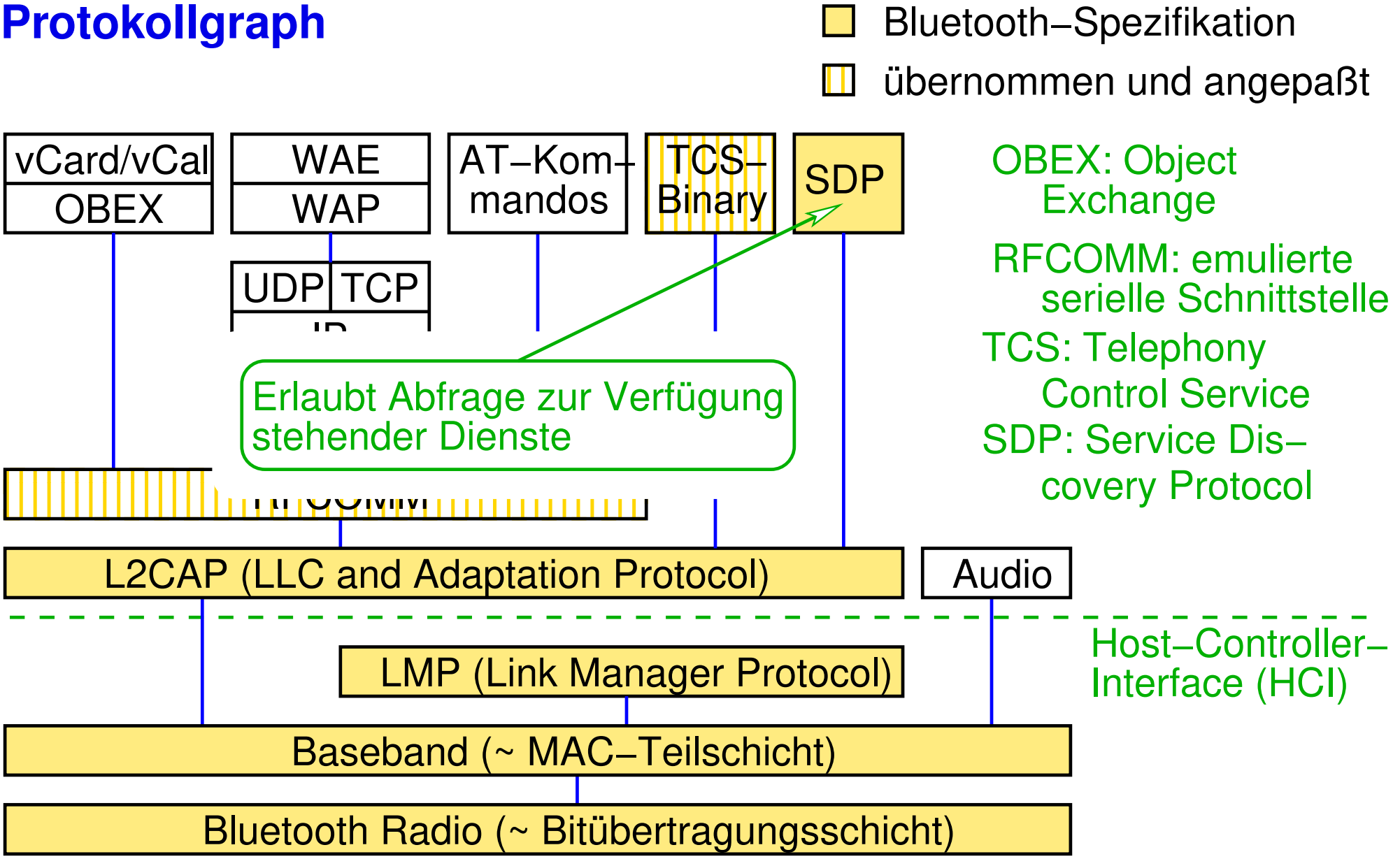
Protokollgraph



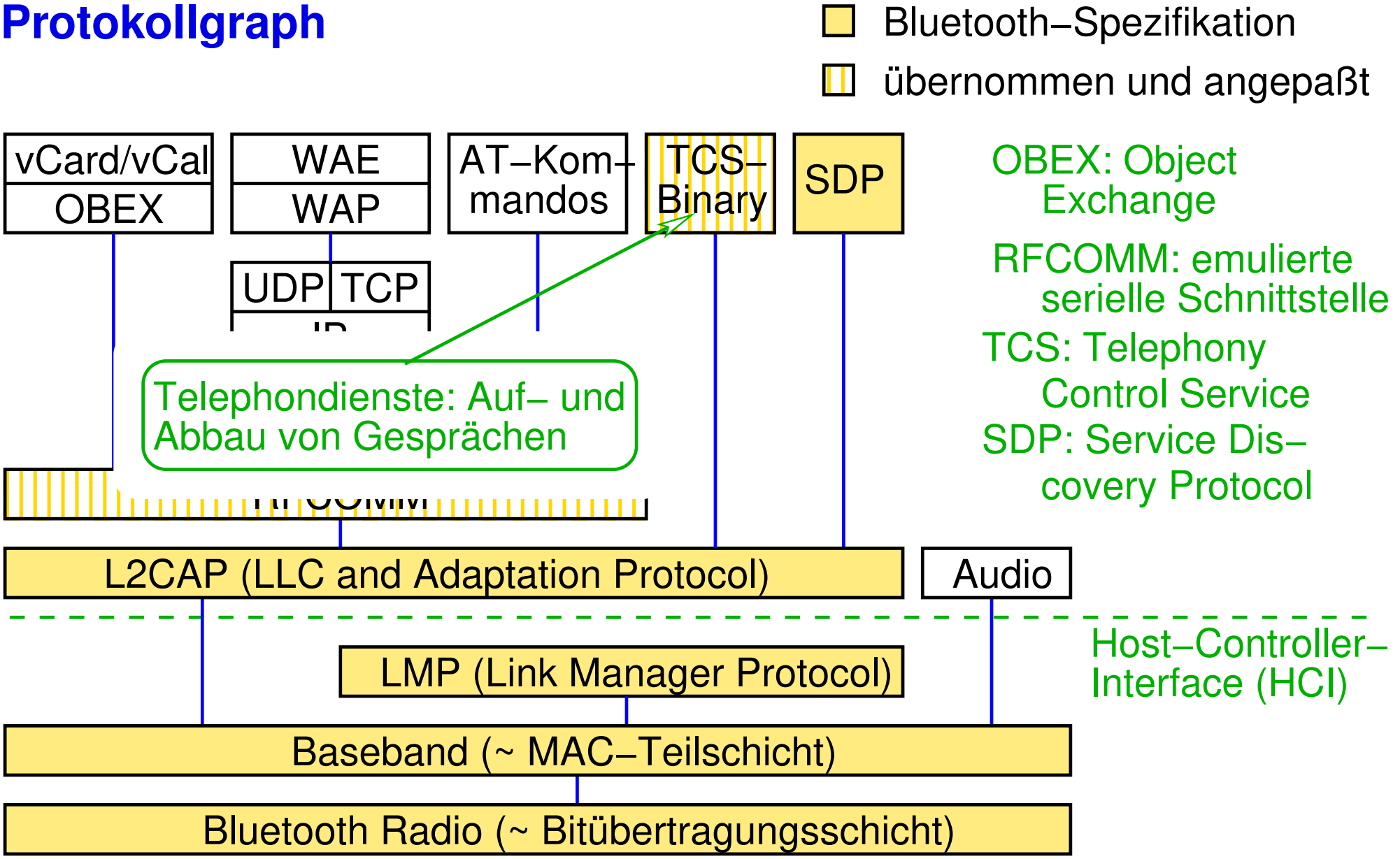
Protokollgraph



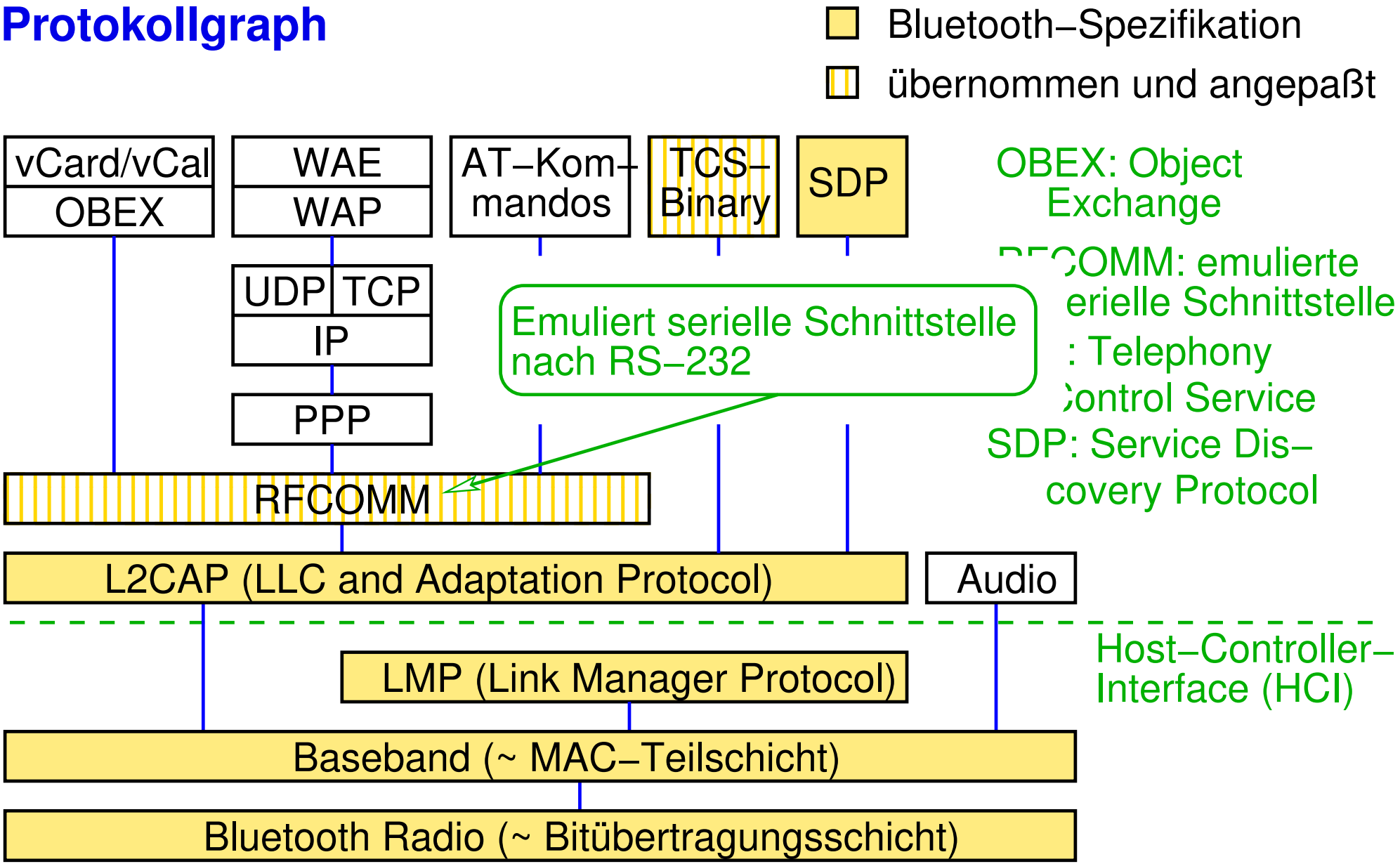
Protokollgraph



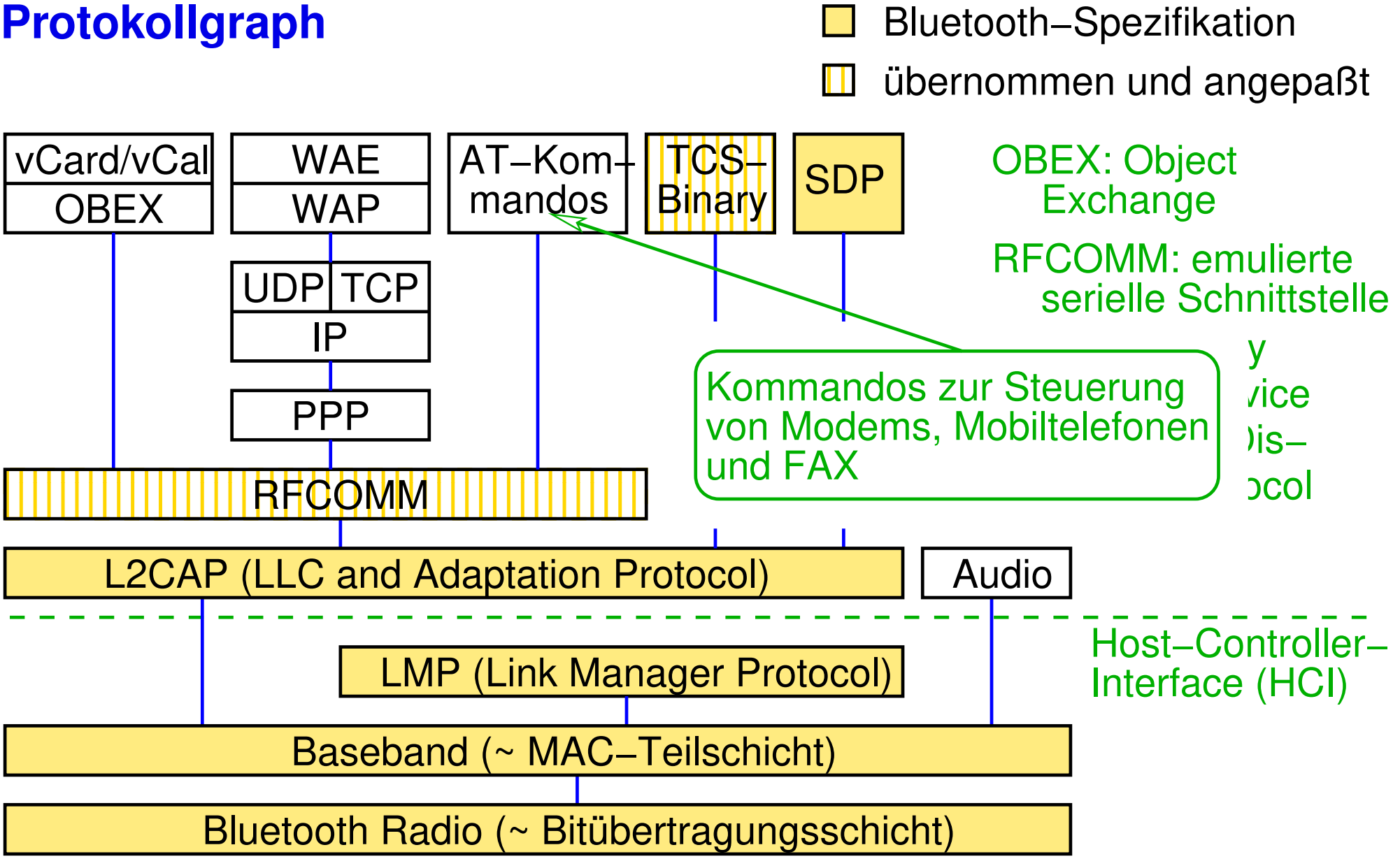
Protokollgraph



Protokollgraph



Protokollgraph



Funkschicht

- 2,4 GHz ISM Band
- 79 Kanäle á 1 MHz
- Frequenzsprungverfahren (FHSS)
  - 1600 Umschaltungen/s (alle 625 μs)
  - Sprungfolge wird vom Master vorgegeben
- Frequenzmodulation, Brutto-Datenrate 1 Mbit/s
- Auch 802.11b/g/n verwendet 2,4 GHz Band
  - gegenseitige Störungen!

Basisband-Schicht (MAC)

- Zeitmultiplex-Verfahren
  - Master beginnt Senden in geraden Zeitschlitzen
  - Slaves beginnen in ungeraden Zeitschlitzen
    - nur nach Erhalt eines Frames vom Master
  - Frames können 1, 3 oder 5 Zeitschlitze lang sein
    - 240 Bit Nutzdaten bei 1 Zeitschlitz
    - 2744 Bit bei 5 Zeitschlitzen
      - mehr als 5 * 240 Bit wegen Übergangszeit bei Frequenzwechsel und Frame-Header

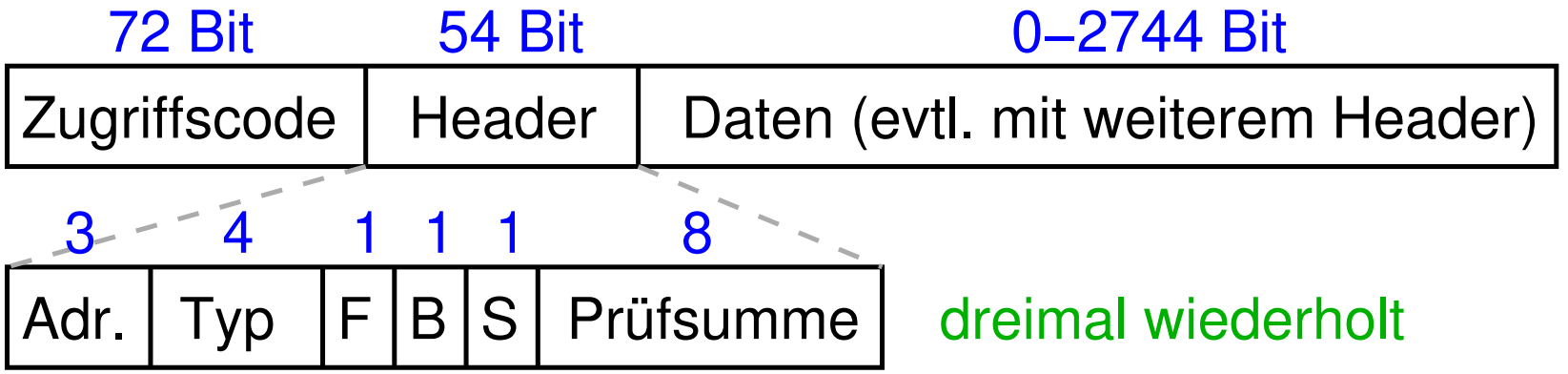
Basisband-Schicht (MAC)

- Übertragung über logische Kanäle (*Links*)
  - ACL (*Asynchronous Connectionless Link*)
    - paketvermittelte Daten, *best effort*
    - pro Slave max. 1 Link
  - SCO (*Synchronous Connection Oriented*)
    - für Echtzeitdaten (Telefonie)
    - feste Zeitschlitze für jede Richtung
    - Vorwärts-Fehlerkorrektur, keine Neuübertragung
      - Code-Raten 1/3, 2/3 und 3/3 (Nutz- / Gesamtdaten)
      - bei 1/3: Daten werden dreimal wiederholt, *Voting*
    - pro Slave max. 3 Links, 64000 Bit/s pro Link
    - Duplex-SCO-Link mit max. Redundanz lastet Netz aus!

L2CAP-Schicht

- *Logical Link Control Adaptation Protocol*
- Fragmentierung und Wiederzusammenbau von Paketen
  - Pakete bis 64 KB
- Multiplexen und Demultiplexen
  - Weitergabe von Paketen an höhere Protokolle
- Aushandlung / Verwaltung von Dienstgüte-Anforderungen
  - z.B. maximale Paketgröße

Frame-Format



- **Zugriffscode** identifiziert Master (d.h. Piconet)
- 3-Bit **Adresse** (7 Slaves + Broadcast durch Master)
- **Typ**: ACL, SCO, Polling, Fehlerkorrektur, Zeitschlitze, ...
- **F**: Flußkontrolle (Empfangspuffer ist voll)
- **B**: Bestätigung (ACK)
- **S**: Sequenzbit (*Stop-and-Wait*-Verfahren)

Sicherheit

- 3 Modi: keine Sicherheit, Sicherheit auf Diensteebene, Authentifizierung und Verschlüsselung auf Link-Ebene
- Bei erster Verbindungsaufnahme: *Pairing*
  - beide Geräte benötigen identische PIN (1-16 Bytes, fest installiert bzw. Benutzereingabe)
- Aus PIN werden Schlüssel berechnet: 8(!) - 128 Bit
- Authentifizierung und Verschlüsselung mit unterschiedlichen Chiffren (SAFER+ bzw. E0-Stromchiffre)
- Schwächen:
  - feste Geräteschlüssel möglich (für alle Verbindungen)
  - nur Geräte-, keine Benutzer-Authentifizierung
  - kein Replay-Schutz

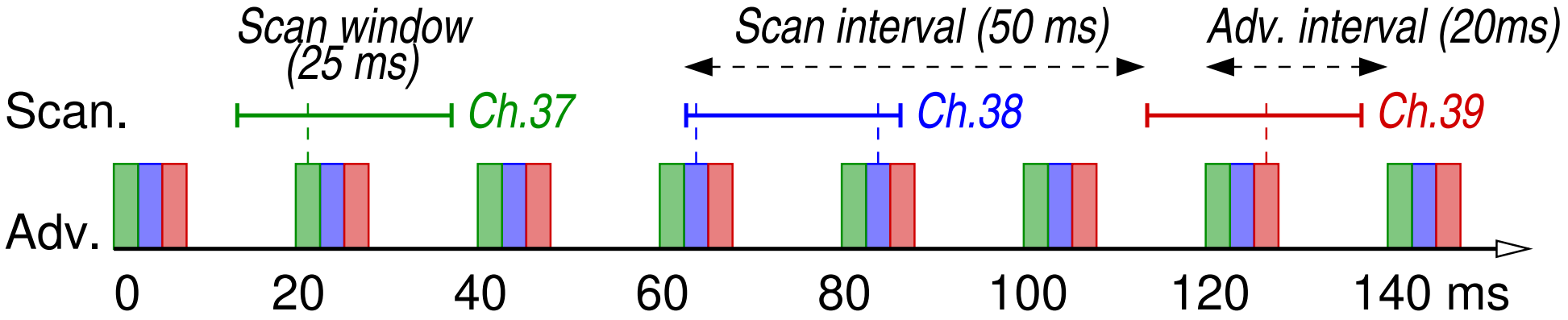
3.2.2 Bluetooth Smart (BT Low Energy, BT 4.x)

- Entwicklung seit 2001 durch Nokia, seit 2007 Bluetooth SIG
- Nicht kompatibel mit 2.x und 3.x, als Ergänzung
- Ziel: möglichst geringer Energieverbrauch, preisgünstig
- Kurze Nachrichten (max. 20 Byte), Datenrate max. 1 Mb/s
- Einfache Sterntopologie (keine Scatternets)
- Anwendungen z.B.:



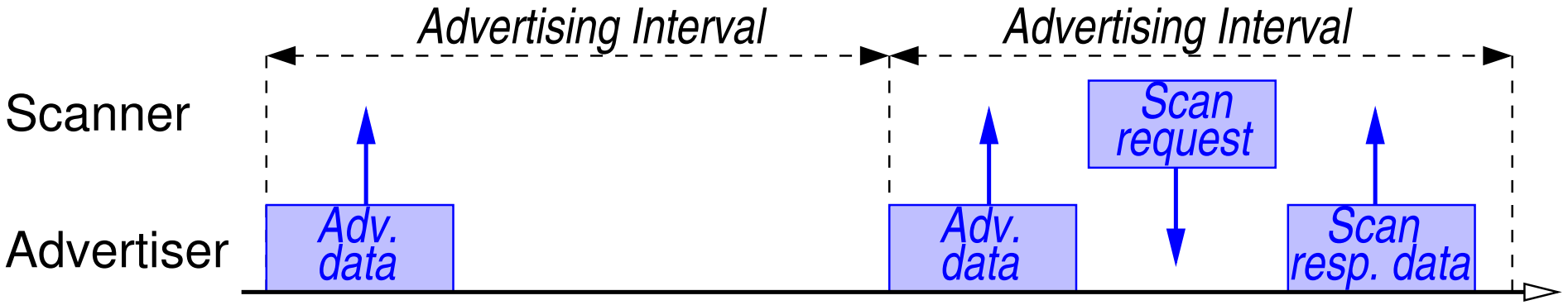

Sicherungsschicht

- Ziel: Funkgerät ist nur möglichst kurz eingeschaltet
  - Energieverbrauch: Empfangsbereitschaft ≈ Senden!
- *Advertising* und *Scanning*
  - Peripheriegerät (*Advertiser*) sendet periodisch Broadcasts
    - auf 3 reservierten Kanälen
    - Intervall: 20ms - 10,24s; mit oder ohne Nutzdaten / Adresse
  - *Scanner* hört Kanäle periodisch ab



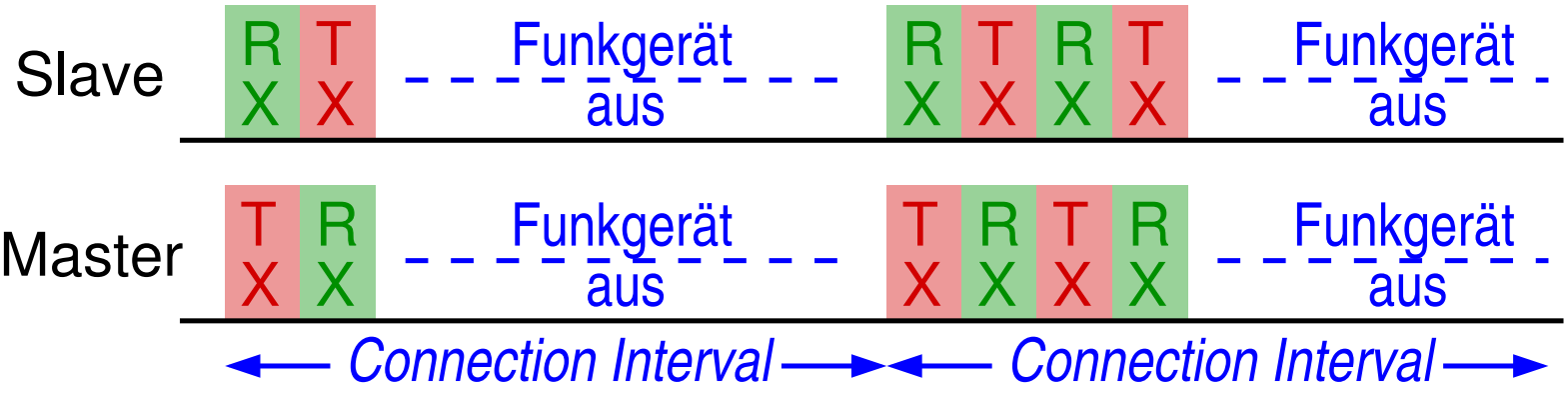
Sicherungsschicht ...

- Aktives Scannen
  - *Advertiser* bleibt nach Versenden der *Advertising*-Daten noch kurz empfangsbereit
  - *Scanner* kann so noch weitere Daten anfordern
  - aber: keine Übertragung von Nutzdaten zum *Advertiser*



Sicherungsschicht ...

- Verbindungsaufbau
  - erlaubt weiteren Datenaustausch, insbes. vom *Scanner* zum *Advertiser*
  - *Scanner* antwortet auf *Advertising*-Paket mit *Connection Request*, u.a. mit
    - Sprungfolge für *Frequency Hopping* (37 Kanäle)
    - *Connection Interval*: wann wird Funkgerät eingeschaltet?



  - Verschlüsselung möglich (bis 128 Bit AES)

Attribut-Protokoll und Attribut-Profil

- Server geben Attribute an Clients bekannt
  - Größe max. 20 Bytes
- Attribute werden über UUIDs (16 bzw. 128 Bit) identifiziert
- Operationen:
  - *Discover*/*Find*, Lesen, Schreiben, Benachrichtigung
- *Generic Attribute Profile*: höhere Abstraktion
  - Profil definiert Menge von *Services*
    - z.B. *Heart Rate Profile*: *Heart Rate* + *Dev. Info. Service*
  - *Service* enthält *Characteristics*
  - *Characteristic* enthält Wert und Metadaten (Eigenschaften, Beschreibung)

WLAN (IEEE 802.11)

- LLC-Teilschicht identisch zu Ethernet
- Ad-hoc und Infrastruktur-Modus
- Spreizbandtechnik
  - *Frequency Hopping*, *Orthogonal Frequency Division Multiplexing*, *Direct Sequence*
  - Ziel: Reduzierung der Störempfindlichkeit
- 802.11b: Überlappende Kanäle im 2,4 GHz ISM-Band
- Zwei MAC Varianten:
  - verteilte Kontrolle: CSMA/CA-Protokolle (MACAW)
  - zentrale Kontrolle: Zuteilung von Zeitschlitzen

WLAN (IEEE 802.11)

- *Hidden / Exposed Station* Probleme
- MACAW
  - MACA: RTS / CTS-Protokoll
    - Reservierung des Mediums für bestimmte Zeit (NAV)
  - MACAW: Einführung von Bestätigungsframes
- IFS zur Priorisierung von Frameklassen
- Sicherheit:
  - WEP: veraltet, kein ausreichender Schutz
  - WPA und v.a. WPA2 bieten gute Sicherheit
- Aktuell: IEEE 802.11n, MIMO-System, max. 600 Mbit/s; bzw. 802.11ac, MIMO-System, max. 1.69 Gbit/s pro Verbindung

Bluetooth

- Vernetzung mobiler Geräte (Handy, PDA), Kabelersatz
- Piconet: Master + max. 7 aktive Slaves
- definiert vollständigen Protokollstapel + Anwendungsprofile
- Funkschicht: 2,4 GHz ISM Band, Frequenzsprungverfahren
- MAC: Zeitmultiplex, zentral durch Master gesteuert
- Vorwärtsfehlerkorrektur, hohe Redundanz
- Sicherheit: optional, ausreichend, aber (theoretisch) angreifbar
- Bluetooth Smart: energiesparende Datenübertragung von „Sensorknoten“

Rechnernetze II

SoSe 2025

4 IP-Routing: Spezielle Aspekte

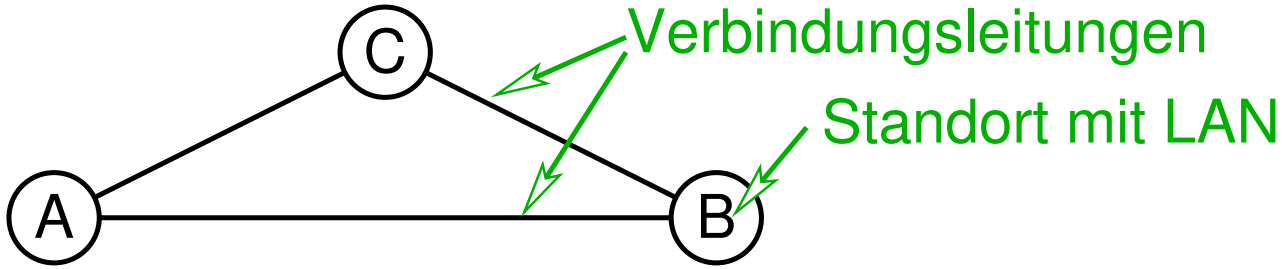
Inhalt

- Tunneling und VPN
- Multicast
- Mobile IP
- *Multiprotocol Label Switching*, MPLS

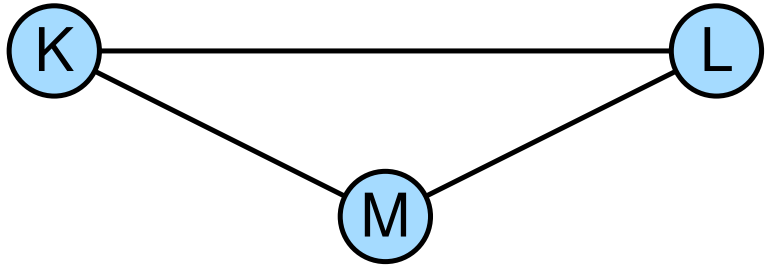
- Tanenbaum, Kap. 5.2.8, 5.2.9, 5.4.5, 5.6.2, 5.6.4
- Peterson, Kap. **4.2.5**, **4.3-4.3.4**, **4.4**, **4.5**
- J.F. Kurose, K.W. Ross: Computernetze. Pearson Studium, 2002. Kap. 4.8 (Multicast)
- CCNA, Kap. 5

4.1 Tunneling und virtuelle private Netze (VPN)

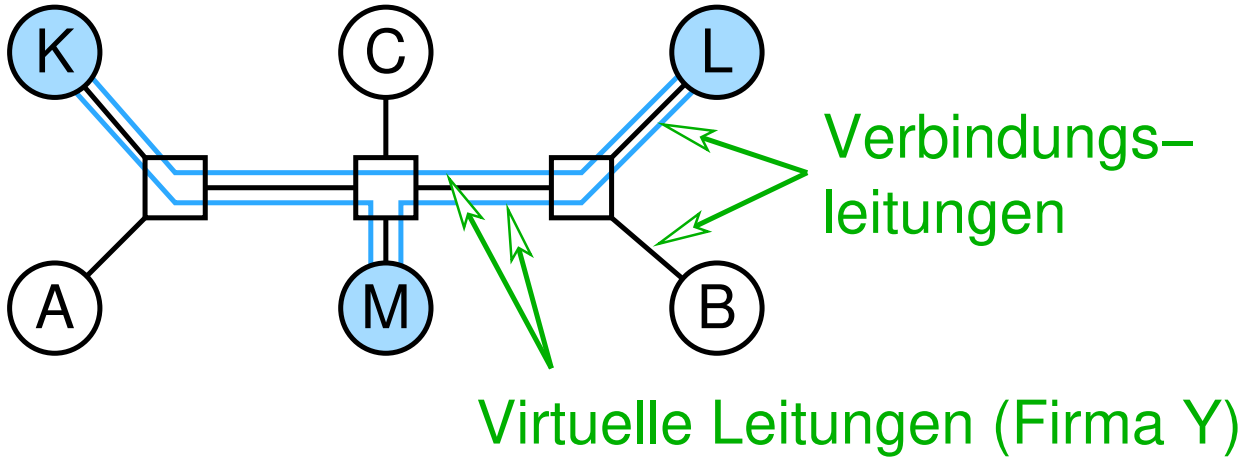
- Zwei private Netzwerke:
  - Firma X:



  - Firma Y:

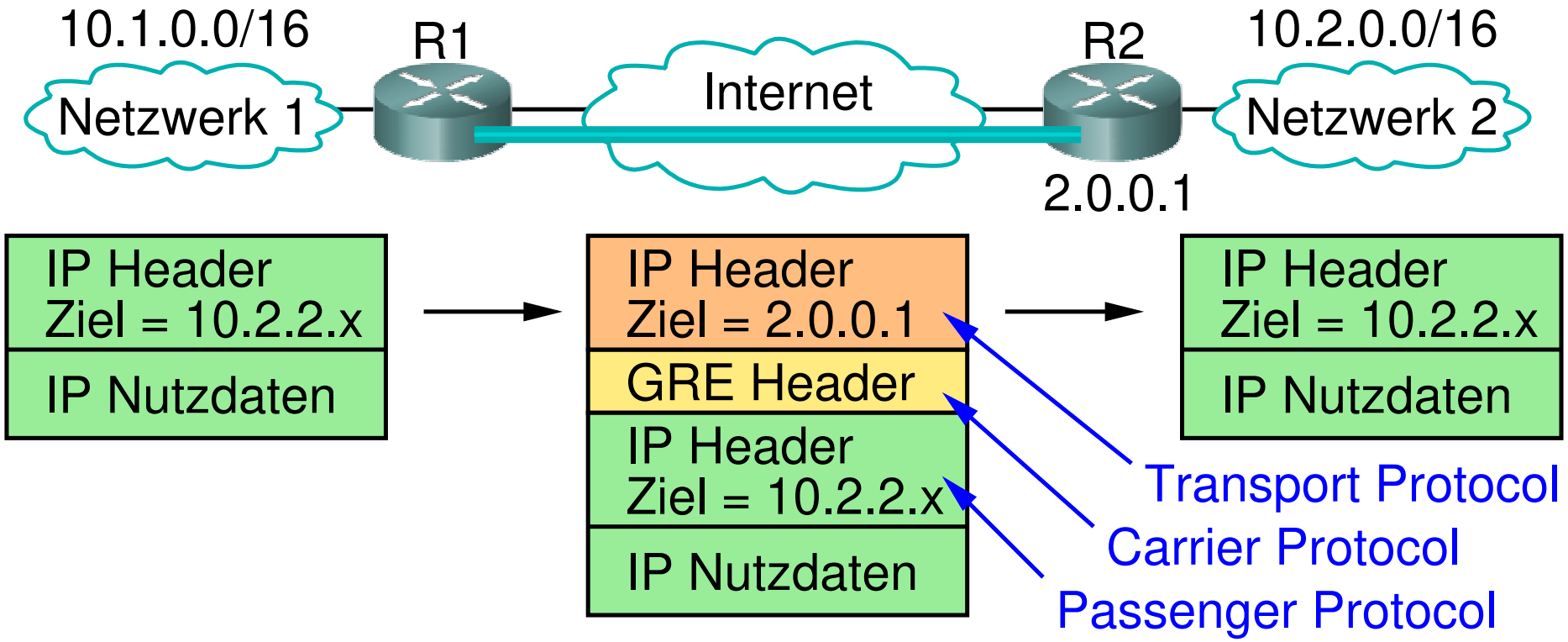


- Zwei VPNs:



4.1 Tunneling und virtuelle private Netze (VPN) ...

- Virtuelle Leitung simuliert eine Layer-2-Verbindung über ein Layer-3-Netz (z.B. Internet)
- Realisierung von virtuellen Leitungen: Tunnel



- Carrier-Protokoll kann ggf. auch fehlen
  - Aufgaben u.a. Multiplexing, Authentifizierung, Reihenfolge, ...

4.1 Tunneling und virtuelle private Netze (VPN) ...

- Einsatz von Tunneln:
  - spezielle Fähigkeiten von R1, R2
    - *Overlay*-Netze, z.B. MBone (Multicast Backbone)
  - Kopplung von nicht-IP-Netzen über das Internet
  - VPNs: Verschlüsselung und Authentifizierung im Tunnel
- Arten von VPNs
  - *Site-to-site* VPN: verbindet z.B. zwei Standorte
    - statisch
    - VPN für interne Hosts nicht sichtbar
    - Realisierung durch Router (VPN *Gateways*), typ. mit IPsec
  - *Remote-access* VPN
    - externer Client verbindet sich dynamisch mit VPN *Gateway*
    - Lösungen über TLS (eingeschränkt) bzw. IPsec

4.2 Multicast

- Multicast: ein Sender sendet an Gruppe von Empfängern
- Anwendungen, z.B.:
  - Multimedia-Streaming (Video- / Audioübertragung)
  - Telekonferenzen
  - Nachrichtenticker, z.B. Börsenkurse
- Einfachste Realisierung:
  - Unicast an jedes Gruppenmitglied
  - verschwendet Bandbreite auf gemeinsamen Verbindungen
- Ziel:
  - Multicast-Unterstützung durch Router
  - Paket auf jeder Verbindung nur einmal übertragen

4.2.1 Adressierung beim Multicast

- Explizite Angabe aller Empfänger skaliert nicht
- Daher: indirekte Adressierung über Multicast-Gruppen
  - Gruppe wird in IPv4 durch Adresse der Klasse D adressiert

    - Adreßbereich 224.0.0.0 - 239.255.255.255
  - In IPv6: Adressbereich FF00::/8
- Fragen:
  - Wahl der Multicast-Adresse?
  - dynamisches Ein- und Austreten möglich?
  - kennen sich die Gruppenmitglieder?
  - wie erfolgt das Routing?

4.2.2 Management von Multicast-Gruppen

IGMP: *Internet Group Management Protocol* (IETF RFC 2236)

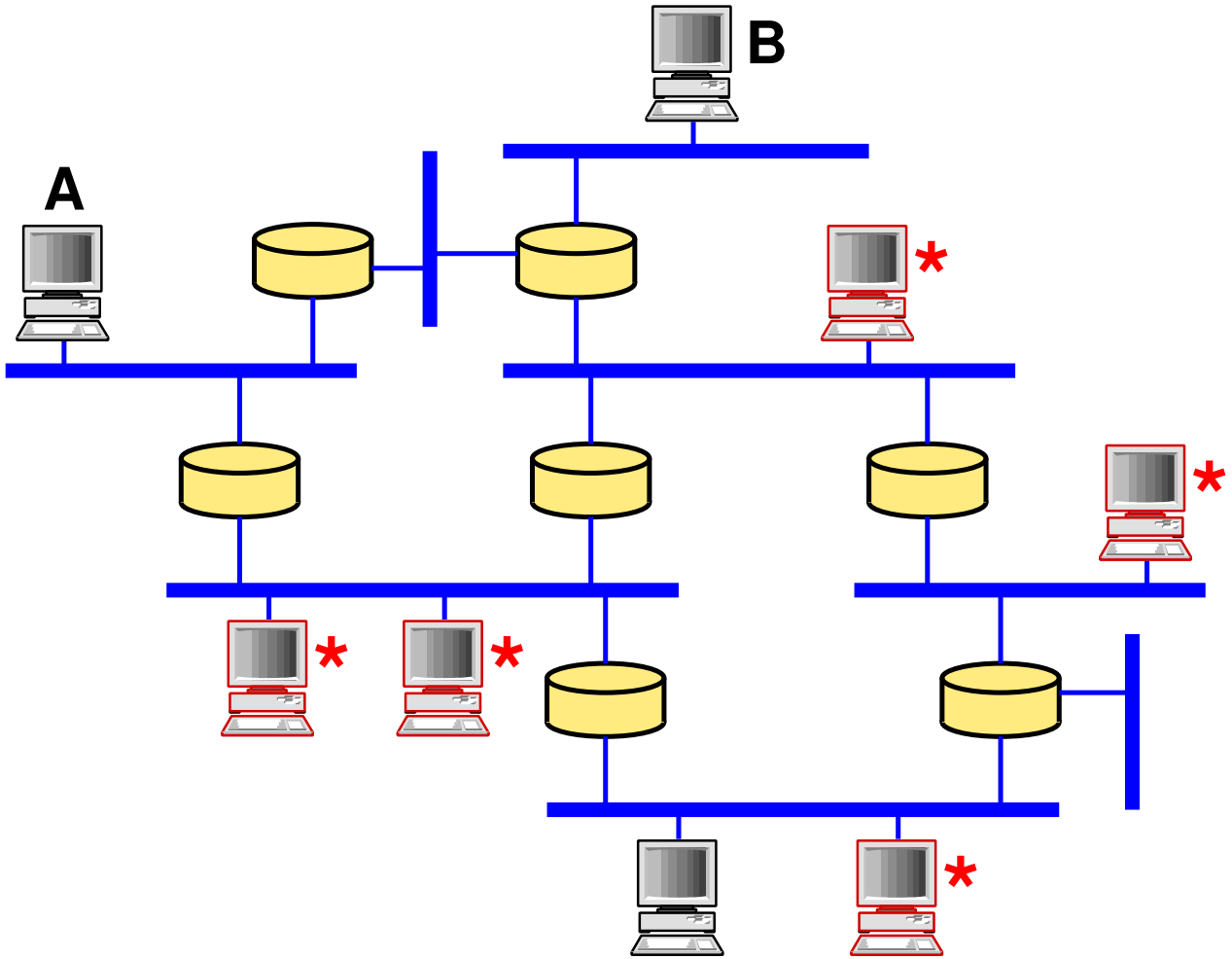
- Protokoll zwischen Host und lokalem Router
  - Informationsaustausch zwischen den Routern nur durch Routing-Protokolle
  - keine globale Information über Gruppenmitglieder!
- Nachrichtentypen:
  - *Membership_query*: Anfrage des Routers an lokales LAN
    - welche Gruppen haben Mitglieder im LAN?
    - ist ein Host Mitglied der angegebenen Gruppe?
    - versendet per LAN-Multicast
  - *Membership_report*: Host ist Mitglied der Gruppe
    - als Antwort auf *Membership_query* oder spontan
  - *Leave_group*: Host verlässt Gruppe (optional)

IGMP: Anmerkungen

- Wahl der Multicast-Adresse erfolgt **nicht** durch IGMP
- Router muß nur wissen, ob es im LAN einen Rechner in einer gegebenen Gruppe gibt
  - Pakete werden lokal mit LAN-Multicast geschickt
- Bei *Membership_query*: Feedback-Unterdrückung:
  - Host wartet vor Antwort zufällige Zeit
  - wenn Host Antwort im LAN sieht: eigene Antwort verwerfen
- Soft-State-Registrierung:
  - Registrierung hat nur bestimmte Lebensdauer
  - periodische *Membership_query*-Anfragen des Routers

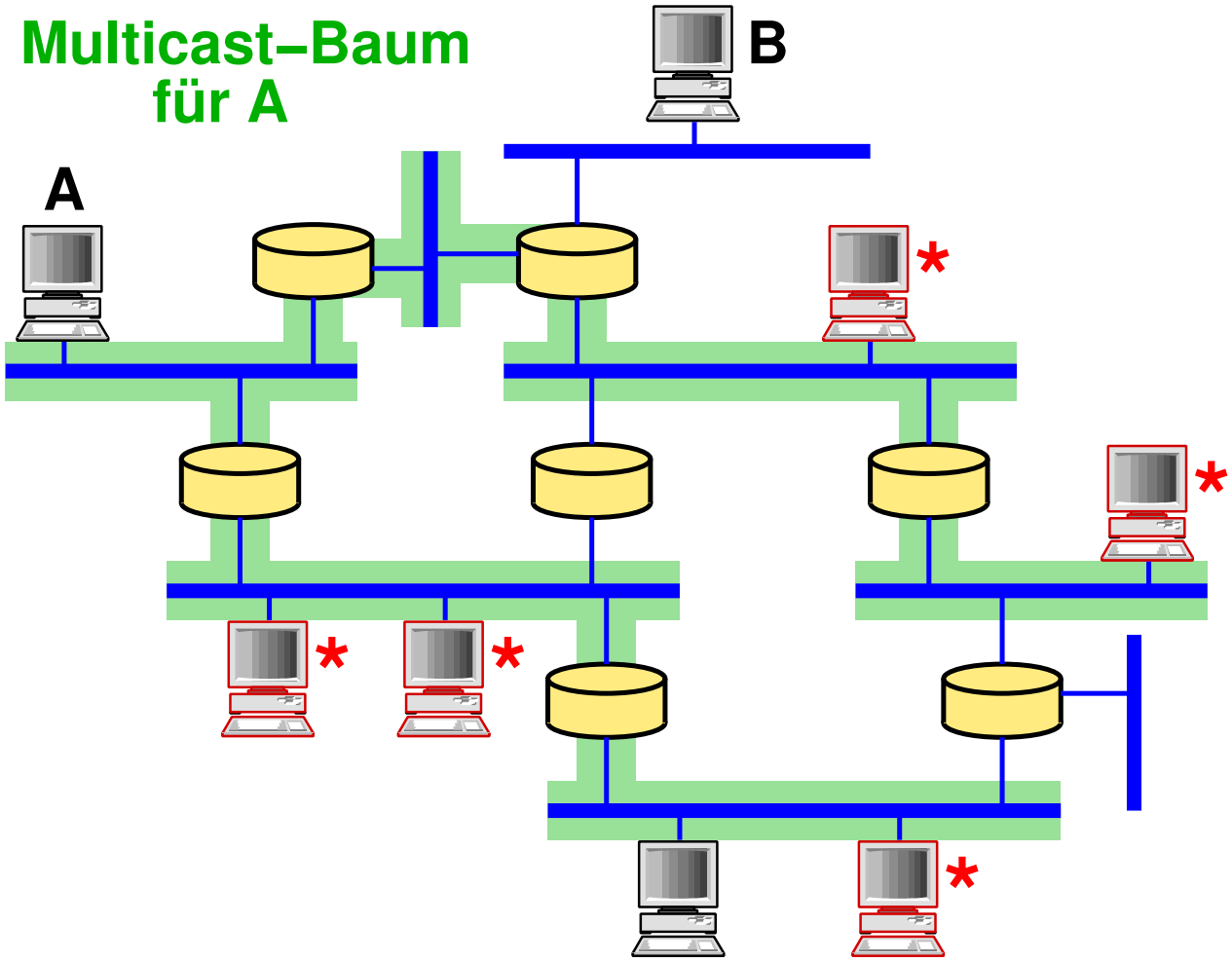
Beispiel-Netzwerk

- (Rot) markierte Rechner gehören zur Multicast-Gruppe



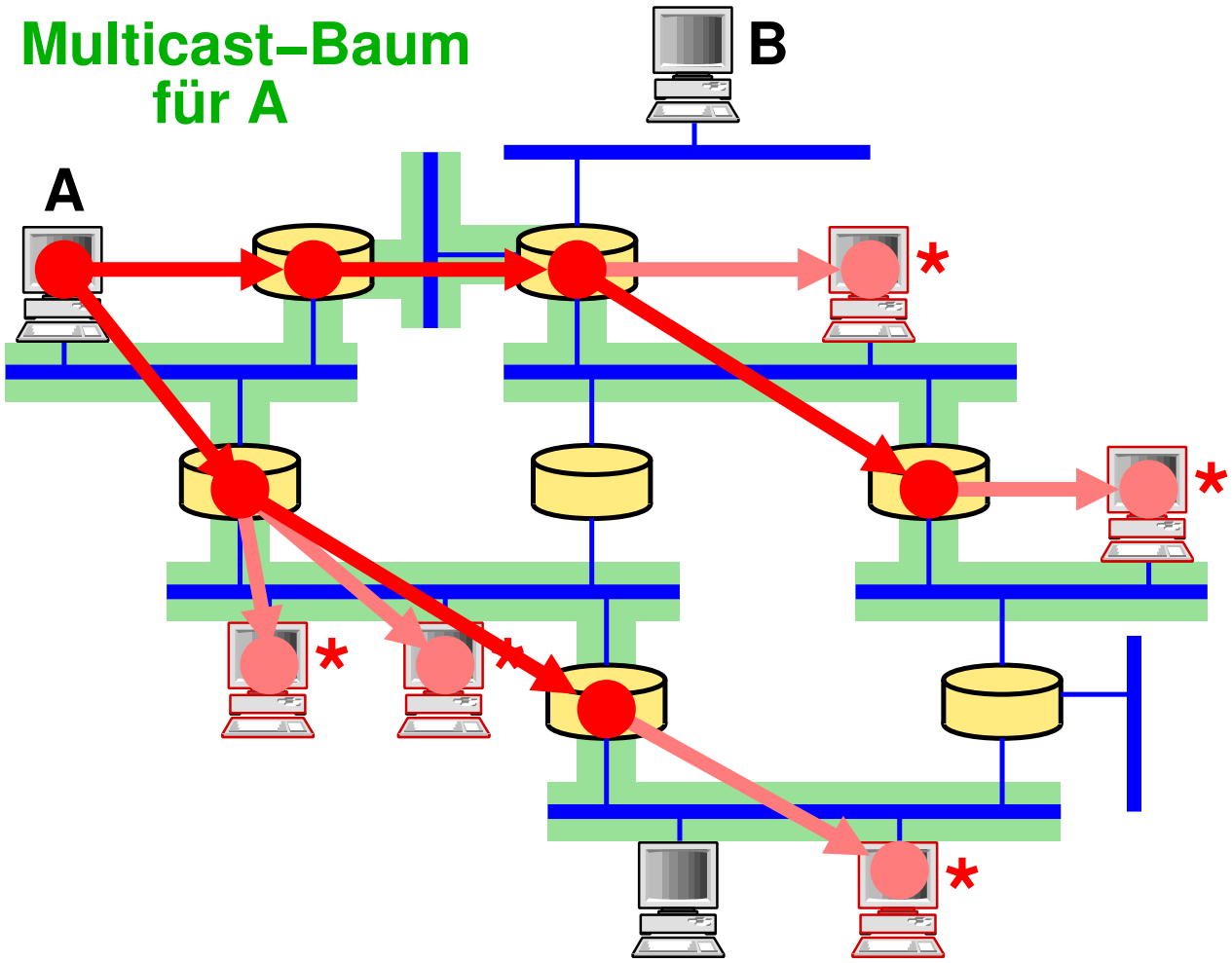
Beispiel-Netzwerk

- (Rot) markierte Rechner gehören zur Multicast-Gruppe
- A sendet: Nachrichten entlang eines aufspannenden Baums mit Wurzel A verteilen



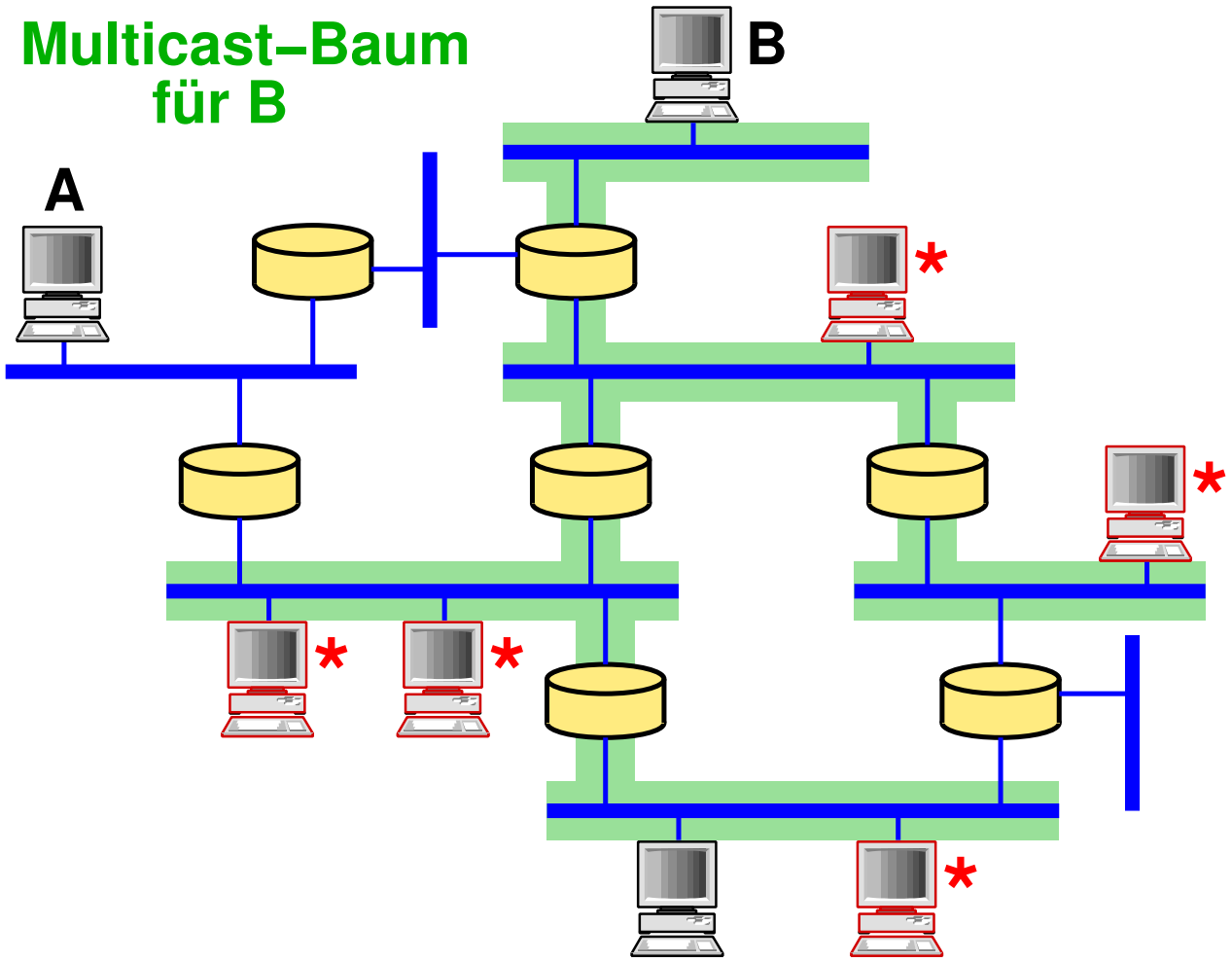
Beispiel-Netzwerk

- (Rot) markierte Rechner gehören zur Multicast-Gruppe
- A sendet: Nachrichten entlang eines aufspannenden Baums mit Wurzel A verteilen



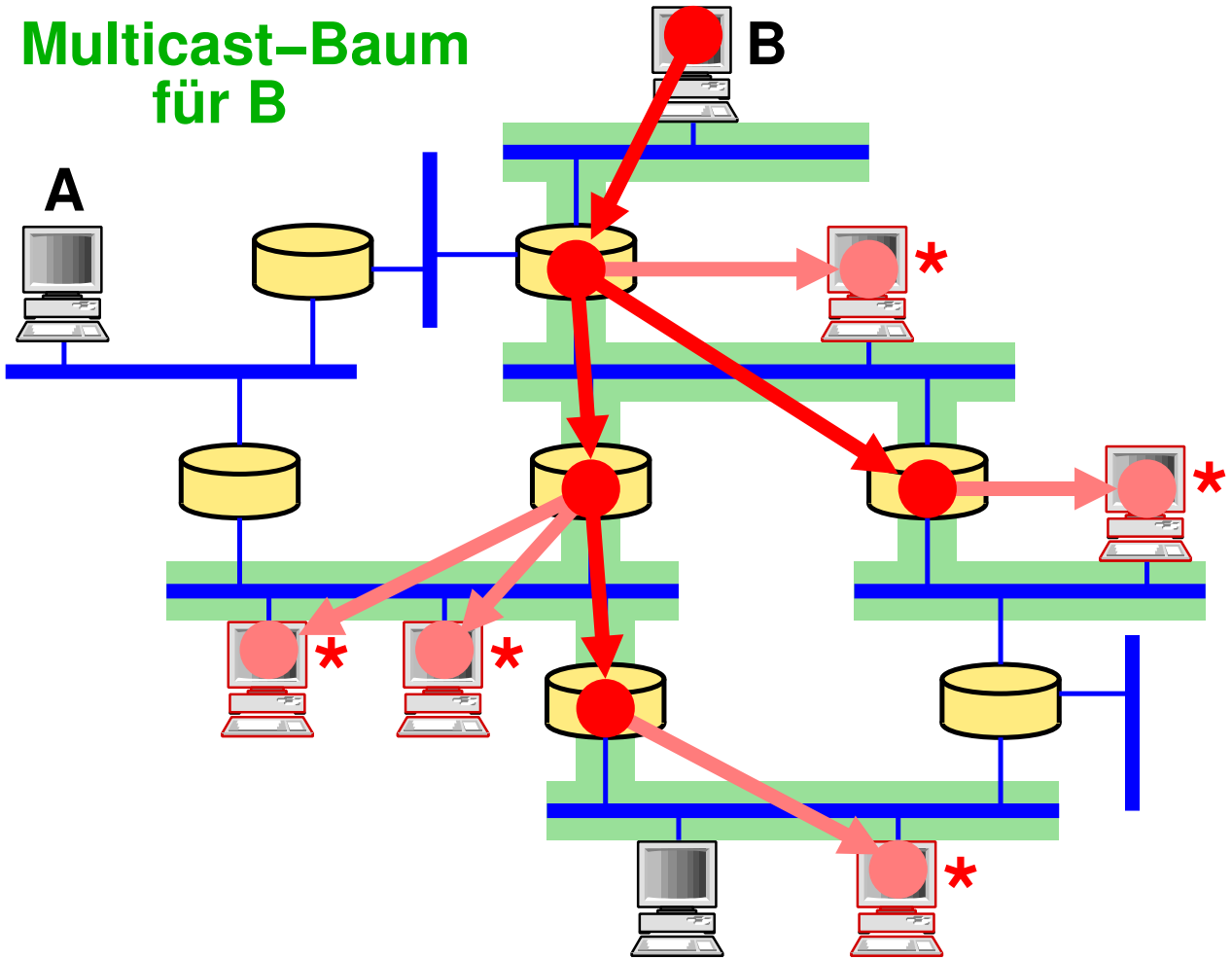
Beispiel-Netzwerk

- (Rot) markierte Rechner gehören zur Multicast-Gruppe
- B sendet: Nachrichten entlang eines aufspannenden Baums mit Wurzel B verteilen



Beispiel-Netzwerk

- (Rot) markierte Rechner gehören zur Multicast-Gruppe
- B sendet: Nachrichten entlang eines aufspannenden Baums mit Wurzel B verteilen



Link-State-Multicast Routing

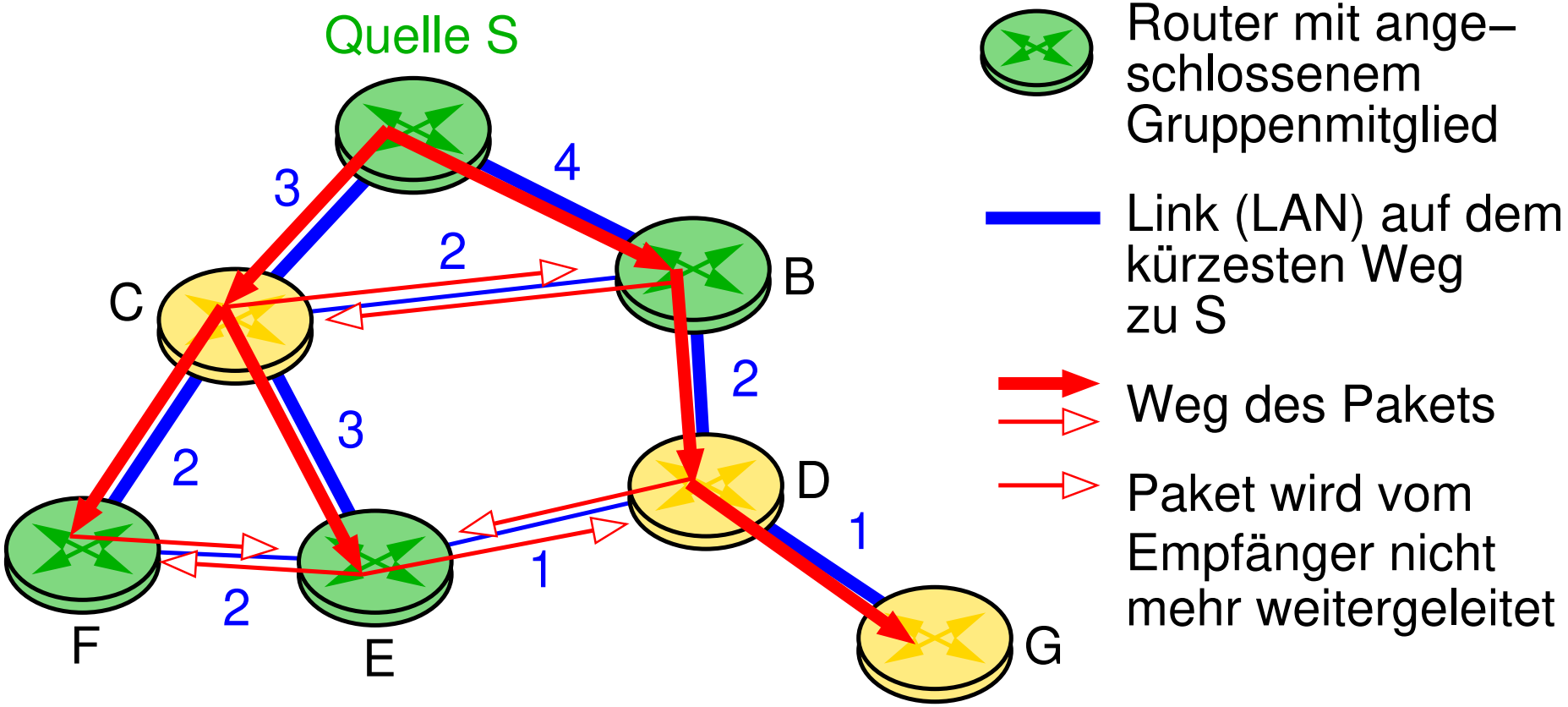
- Erinnerung:
  - Durch *Reliable Flooding* erhält jeder Router Information über das Gesamtnetz
  - Berechnung kürzester Wege durch Dijkstra-Algorithmus
- Für Multicast-Routing:
  - Link-State-Pakete geben für jedes LAN an, für welche Gruppen Mitglieder im LAN sind
  - jeder Router berechnet spannenden Baum mit kürzesten Wegen im Gesamtnetz
    - von jeder Quelle zu jeder Gruppe!

Distanzvektor-Multicast (IETF RFC 1075)

- Erinnerung: Distanzvektor-Routing
  - Router kennen globalen Netzwerkgraph nicht
  - jeder Router hält Tabelle mit Einträgen
    <Ziel, Kosten, nextHop>
  - Router tauschen *<Ziel, Kosten>* Nachrichten aus

- Grundprinzip: *Reverse Path Forwarding* (RPF)
  - wenn ein Router ein Paket von Quelle S über Link L erhält:
    - Paket an alle Links außer L weiterleiten (wie bei Flooding)
    - **aber nur**, wenn L der Link auf dem kürzesten Weg zu S ist (das Paket also vom *nextHop* in Richtung S kam)

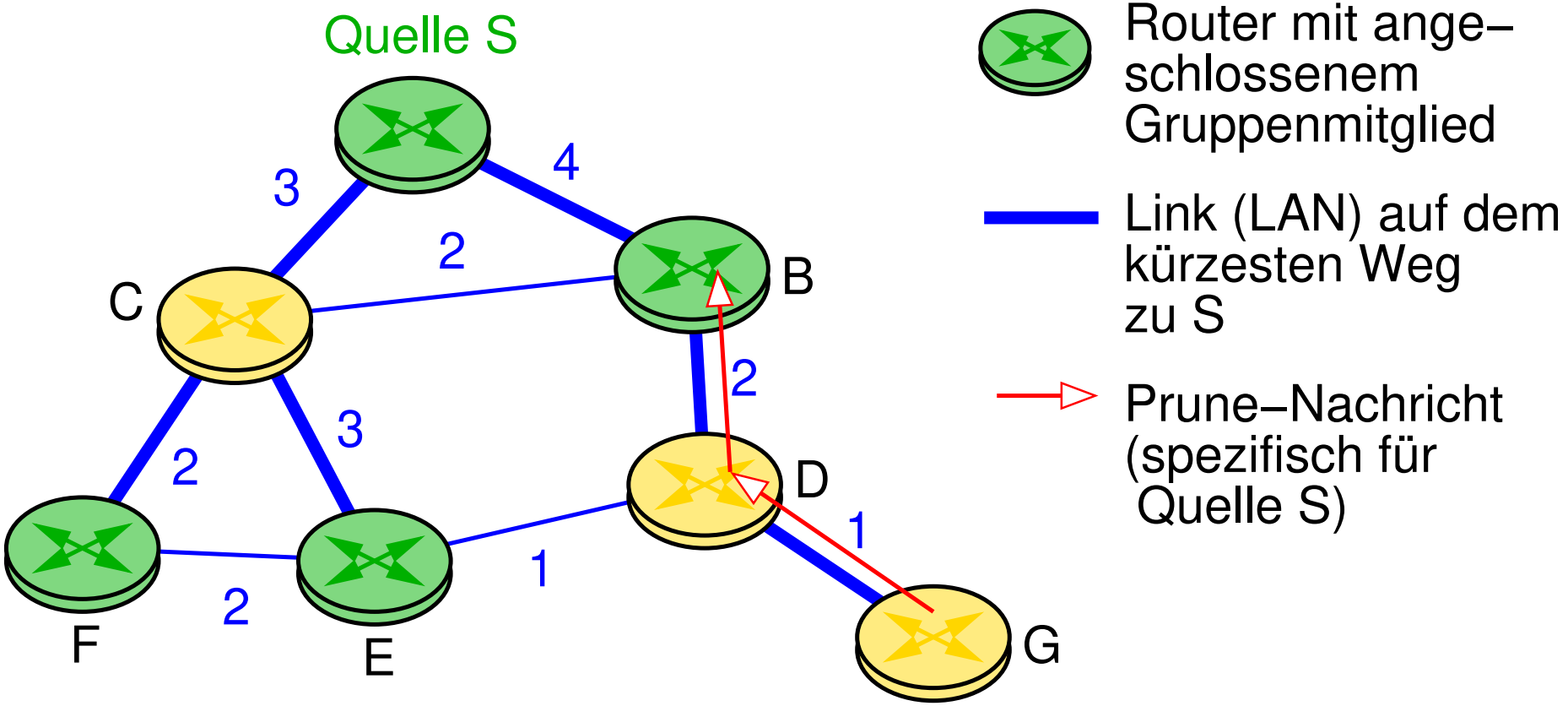
Beispiel zum *Reverse Path Forwardng*



Distanzvektor-Multicast ...

- *Pruning*
  - ein „Blatt“-Router, der Pakete empfängt, aber kein Gruppenmitglied im LAN hat (im Beispiel: G), sendet einen *Prune*-Nachricht an seinen Upstream-Router (im Beispiel: D)
  - ein Router, der von allen Downstream-Routern *Prune* empfangen hat, sendet *Prune* upstream weiter
  - Rückgängigmachen des *Pruning* durch *Timeout* oder explizite *Join*-Nachricht
- Verfeinerung: *Reverse Path Broadcast* (RPB)
  - wenn an ein LAN mehrere Router angeschlossen sind, sendet nur einer davon Multicast-Pakete in das LAN
- Einsatz in MBone (*Multicast Backbone*)

Beispiel zum *Pruning*



PIM: *Protocol Independent Multicast* (IETF RFC 2362)

- Problem bei RPF: Skalierbarkeit
  - Default-Verhalten: **jeder** Router erhält das Paket
  - meist aber nur wenige Router wirklich betroffen
- Bei PIM daher zwei Modi:
  - *dense*: Ansatz wie bei RPF (mit *Pruning*)
  - *sparse*: Router muß sich explizit registrieren
- Im Folgenden: *sparse*-Modus

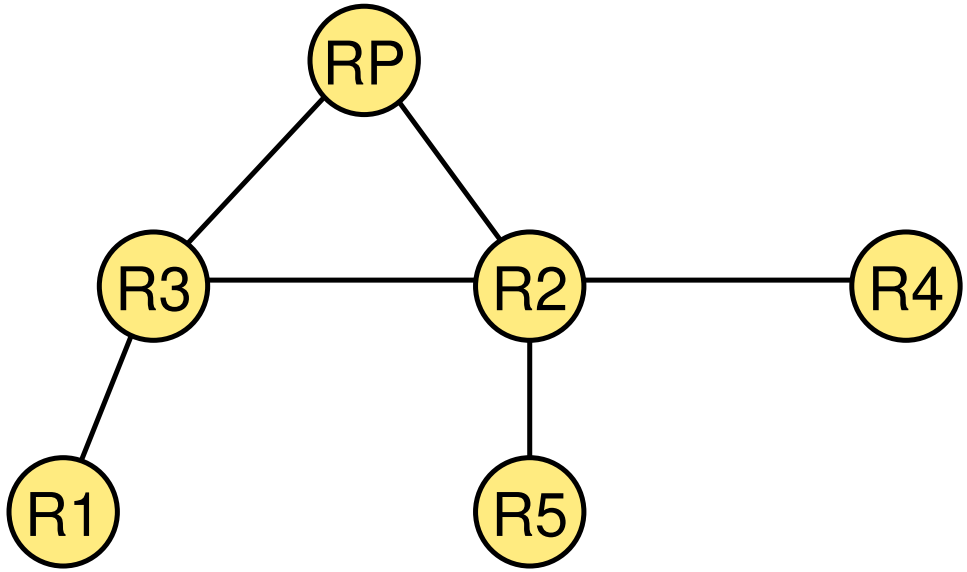
PIM: Aufbau des Multicast-Baums

- Für jede Gruppe wird ein spezieller Router (**Rendezvouspunkt**, RP) ausgewählt
- Router senden *Join* bzw. *Prune*-Nachrichten an RP, um sich zu registrieren bzw. abzumelden
- Durch den Weg der *Join*-Nachrichten wird ein Baum aufgebaut (mit RP als Wurzel)
  - unabhängig vom verwendeten Routing-Protokoll ($\Rightarrow$ **PI**M)
  - ein gemeinsamer Baum für alle Quellen

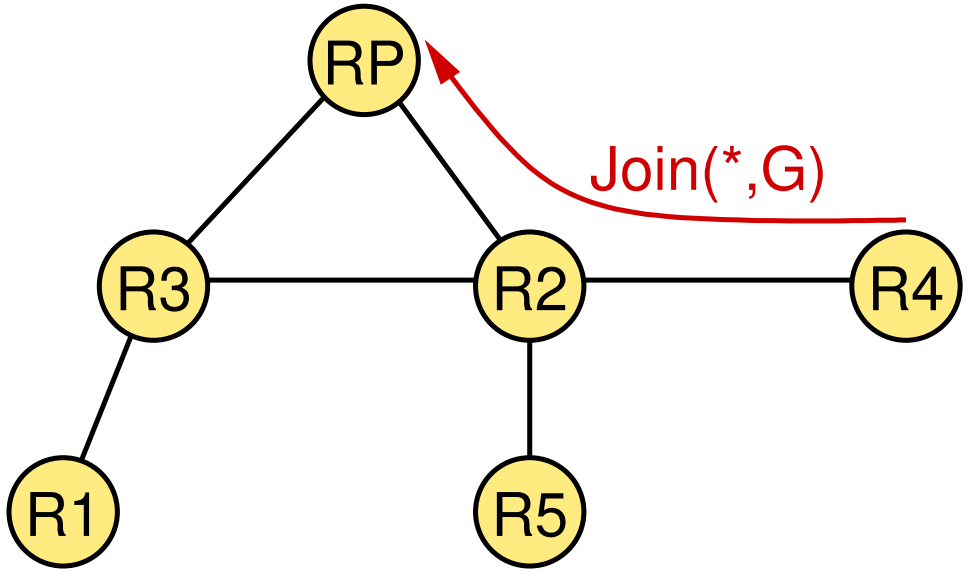
PIM: Routing eines Multicast-Pakets

- Ablauf beim Senden eines Multicast-Pakets:
  - Quelle sendet Paket über Tunnel an RP
  - RP sendet Paket über Baum an Multicast-Gruppe
- Optimierungen (bei entsprechendem Verkehrsaufkommen):
  1. RP sendet quellenspezifischen *Join* an Quelle
     - damit kennen dazwischenliegende Router den Pfad, kein IP-Tunnelling mehr notwendig
     - Pfad gilt nur für die im *Join* angegebene Quelle
  2. Empfänger senden quellenspezifischen *Join* an Quelle
     - Aufbau eines quellenspezifischen Baumes (mit Quelle als Wurzel)

PIM: Beispiel

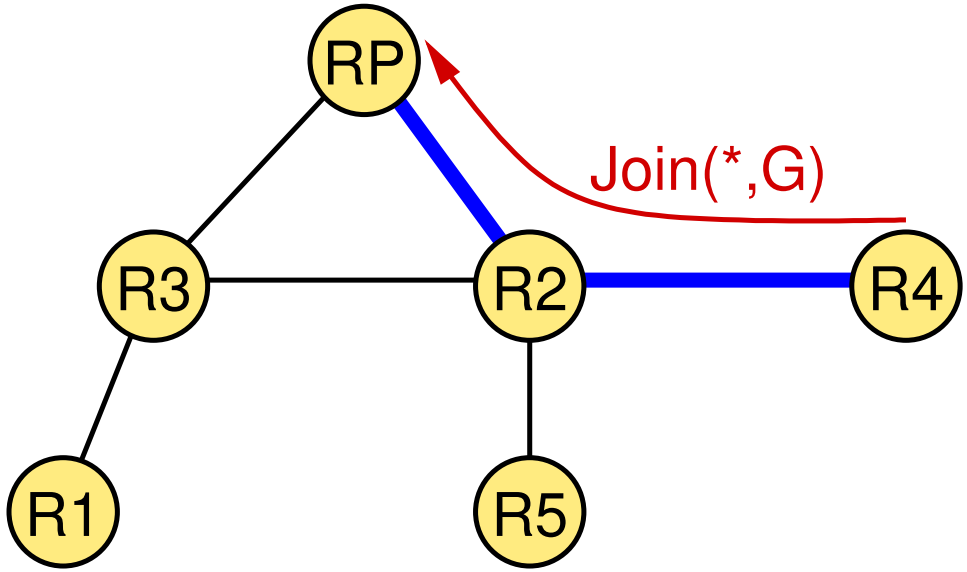


PIM: Beispiel



R4 sendet Join

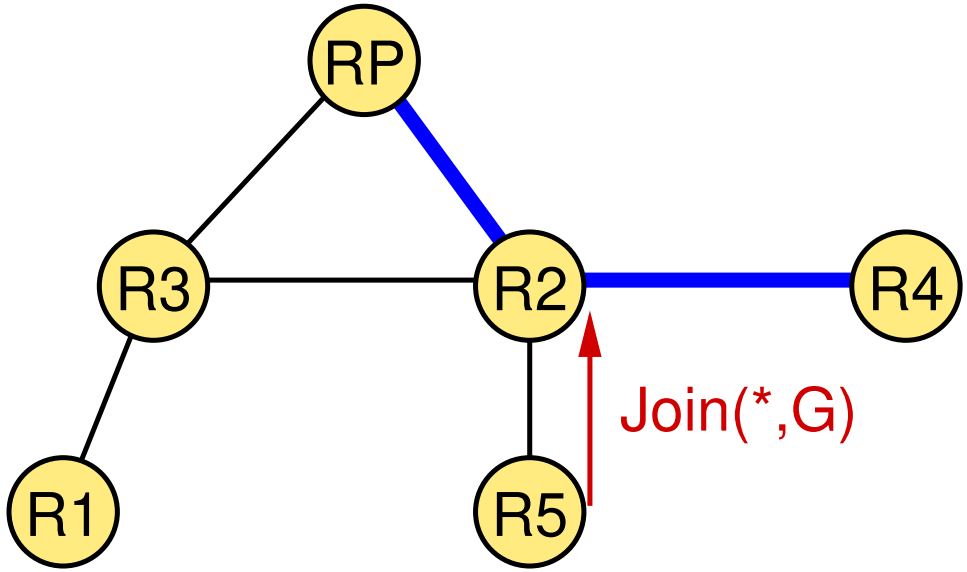
PIM: Beispiel



Pfad RP–R2–R4 wird in Baum aufgenommen

Gemeinsamer Multicast–baum

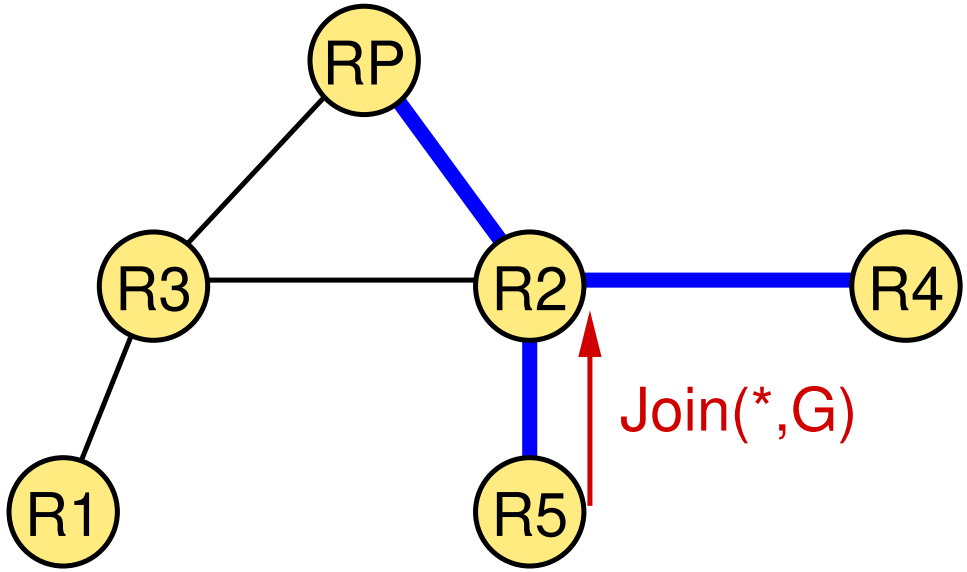
PIM: Beispiel



R5 sendet Join

Gemeinsamer Multicast–baum

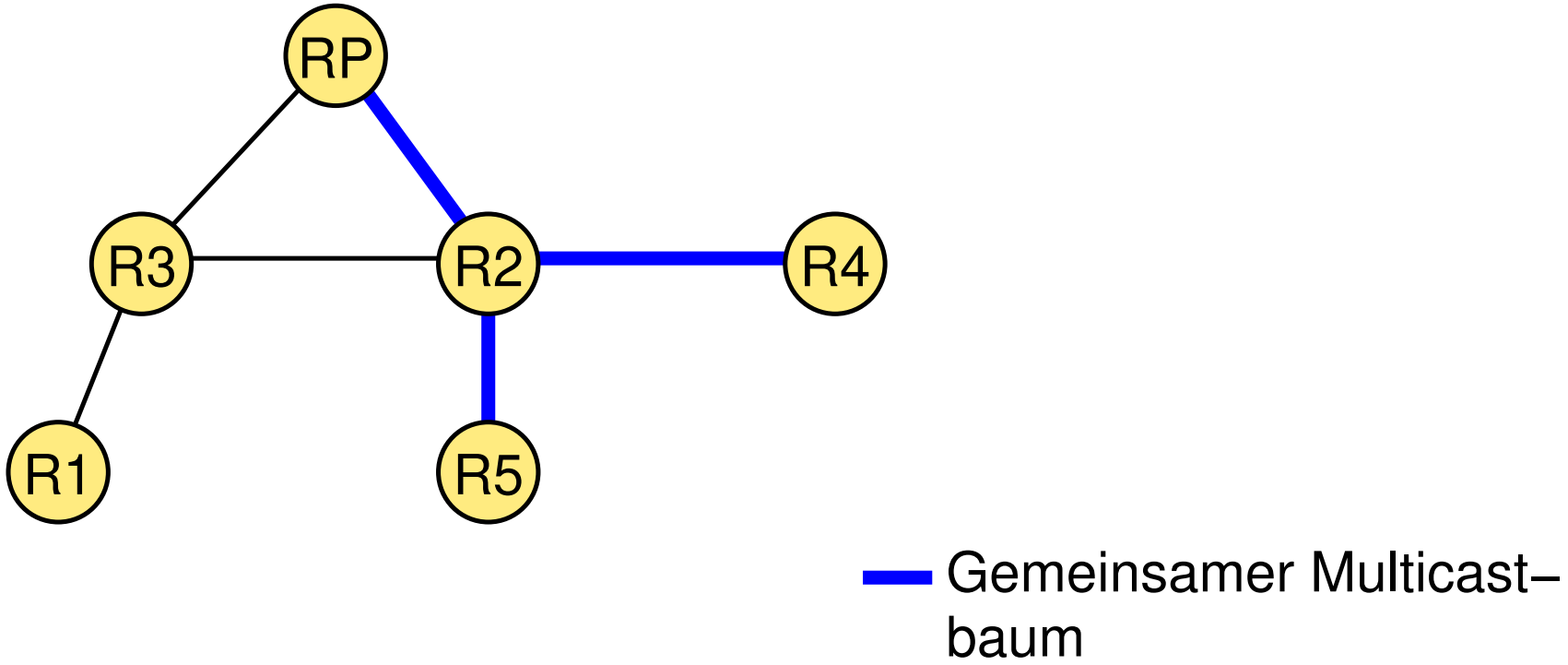
PIM: Beispiel



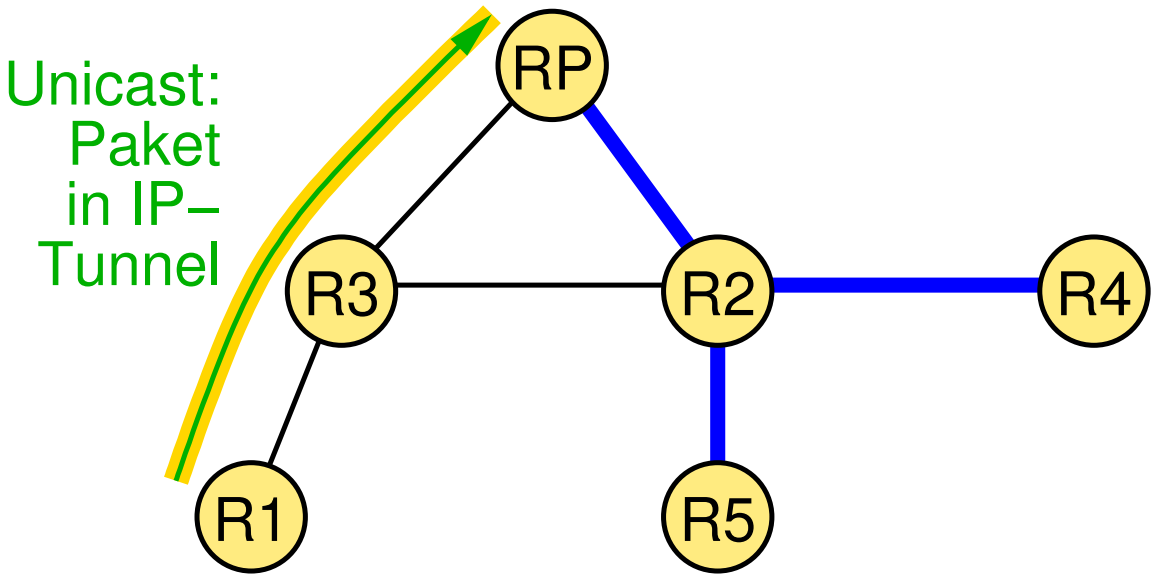
Pfad R2–R5 wird zu Baum hinzugefügt

Gemeinsamer Multicast–baum

PIM: Beispiel



PIM: Beispiel

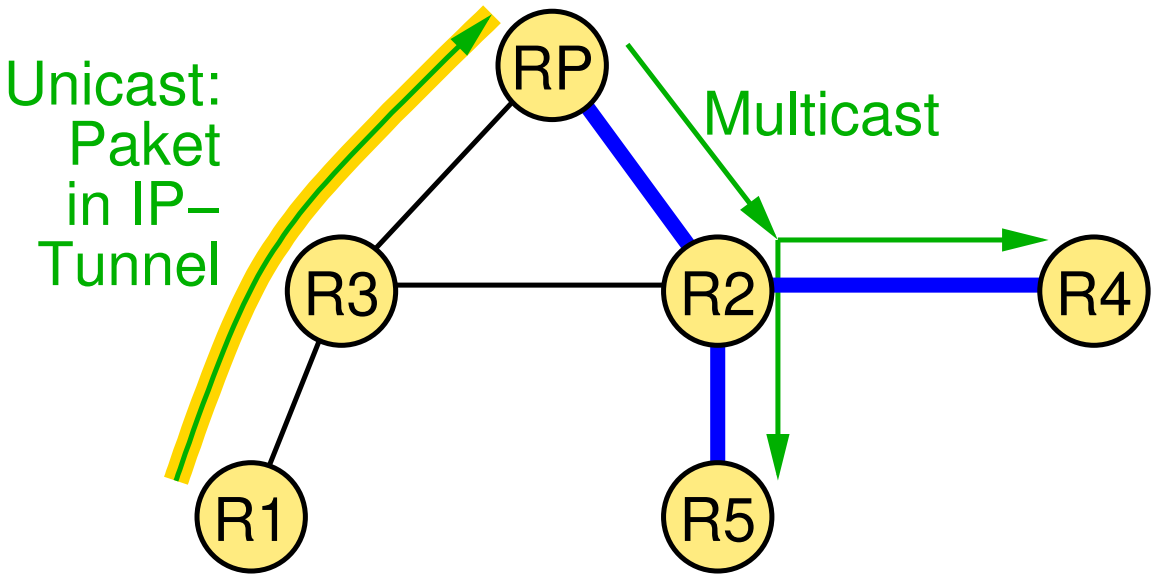


R1 sendet Paket an Gruppe:

a) R1 sendet Paket über Tunnel an RP

Gemeinsamer Multicast–baum

PIM: Beispiel



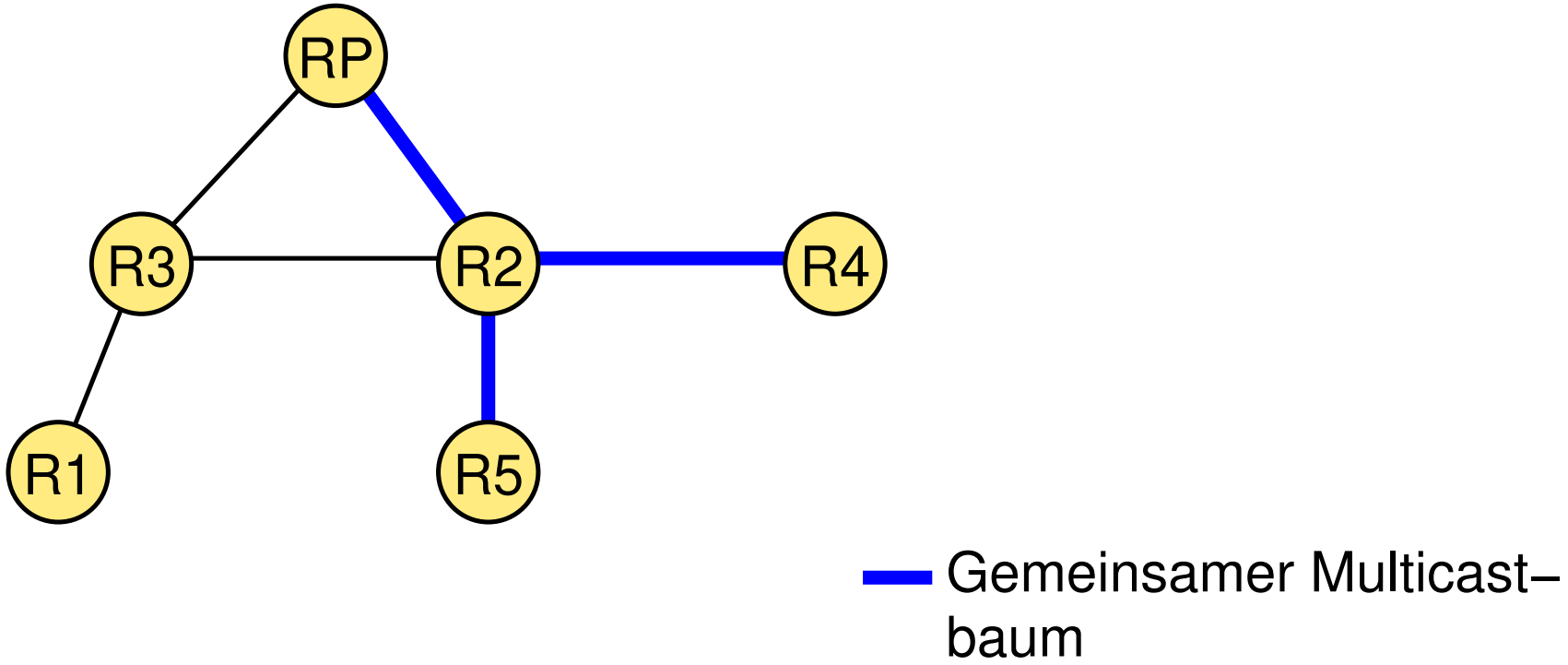
R1 sendet Paket an Gruppe:

a) R1 sendet Paket über Tunnel an RP

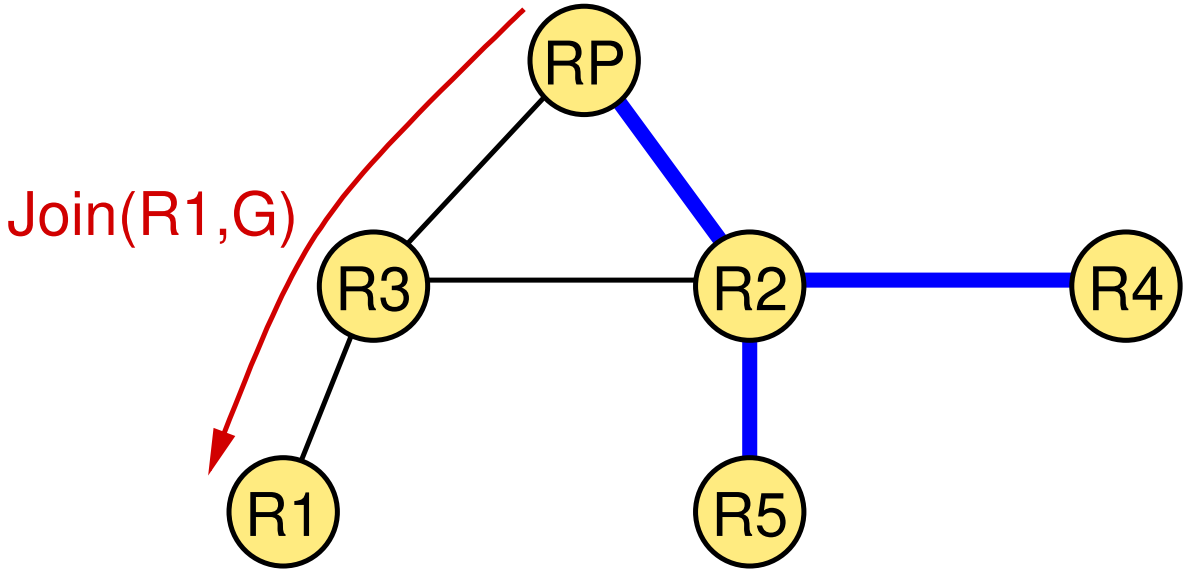
b) RP sendet Paket über Multicast

Gemeinsamer Multicast–baum

PIM: Beispiel



PIM: Beispiel

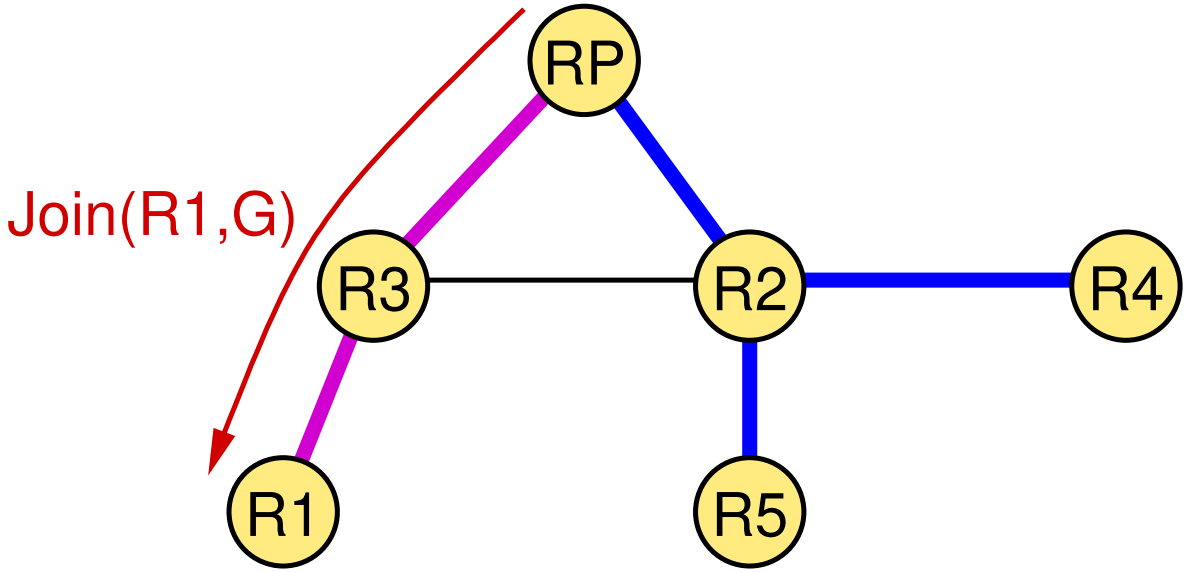


RP stellt hohes Paket–aufkommen von R1 fest

RP sendet quellenspezifi–schen Join an R1

Gemeinsamer Multicast–baum

PIM: Beispiel

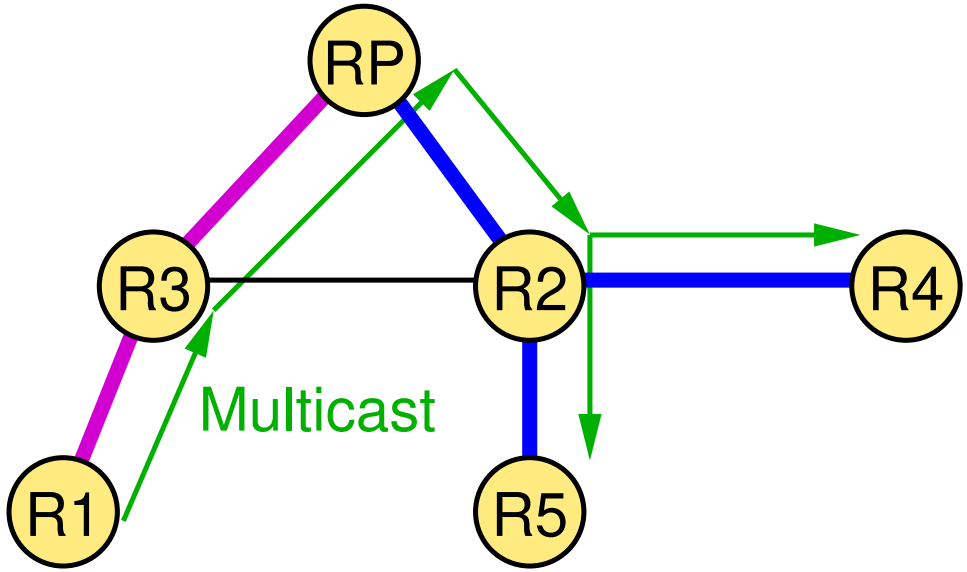


Pfad R1–RP wird zum Baum hinzugefügt

Gemeinsamer Multicast–baum

Quellenspezifischer Multi–castbaum für R1

PIM: Beispiel

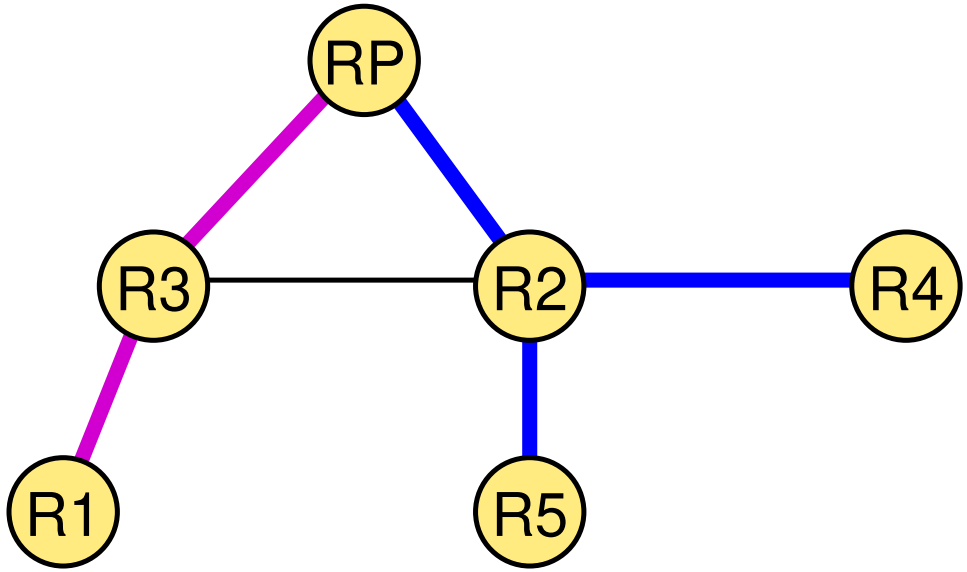


R1 sendet Paket über Multicast–Baum

Gemeinsamer Multicast–baum

Quellenspezifischer Multi–castbaum für R1

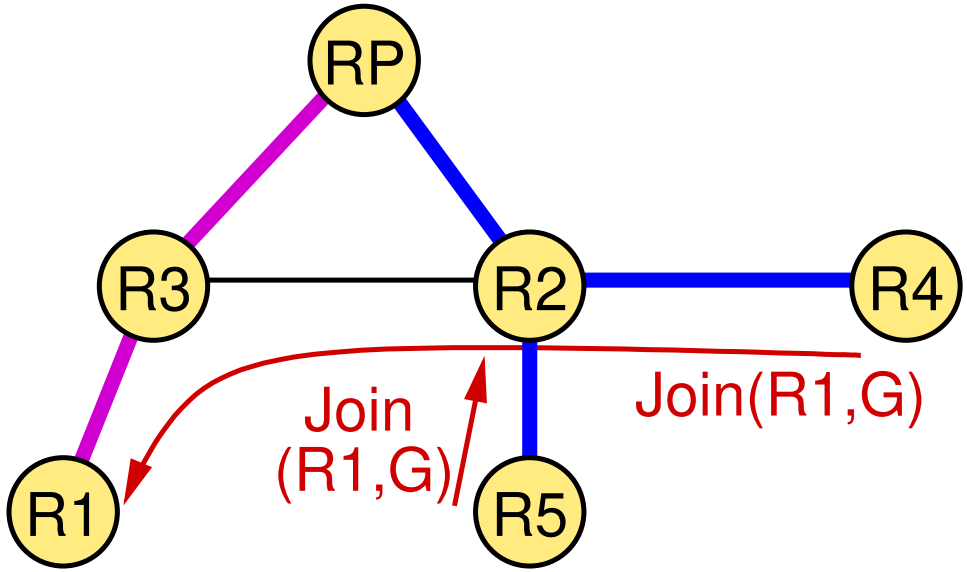
PIM: Beispiel



Gemeinsamer Multicast–baum

Quellenspezifischer Multi–castbaum für R1

PIM: Beispiel



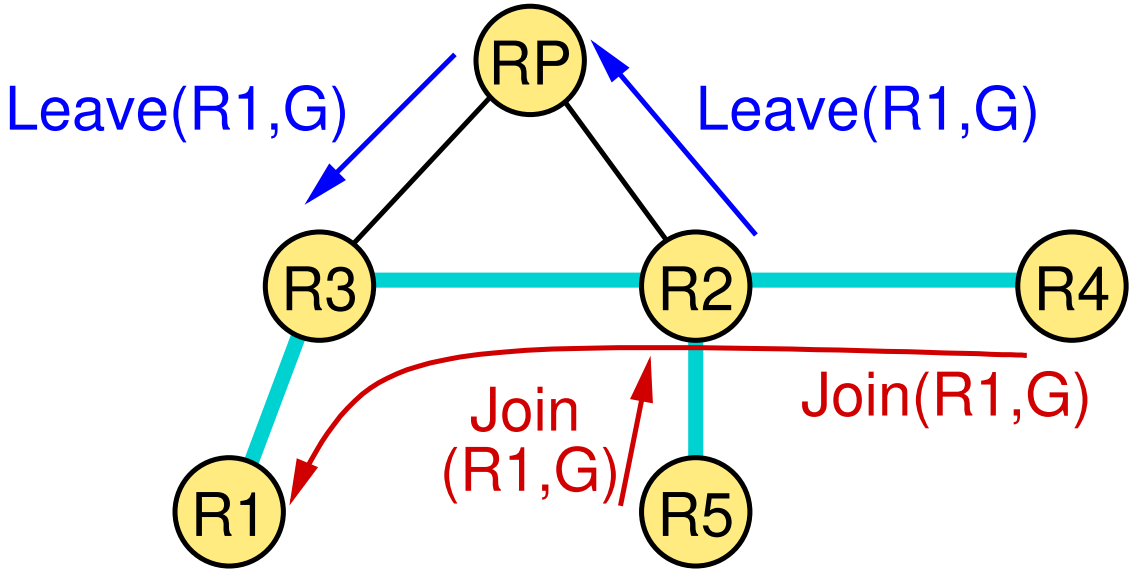
R4 und R5 stellen hohes Paketaufkommen von R1 fest

R4 und R5 senden quellen–spezifischen Join an R1

Gemeinsamer Multicast–baum

Quellenspezifischer Multi–castbaum für R1

PIM: Beispiel

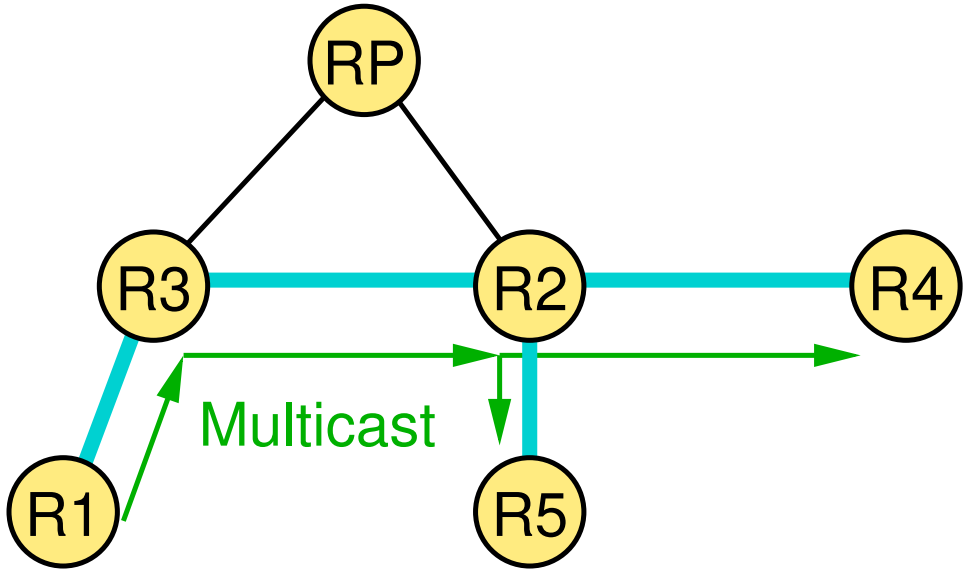


Pfade R1–R3–R2–R4 und R2–R5 werden in quellen–spezifische Baum von R1 aufgenommen

R2 und RP melden sich für Quelle R1 ab

Quellenspezifischer Multi–castbaum für R1

PIM: Beispiel



R1 sendet Paket über quellenspezifischen Baum

Quellenspezifischer Multi–castbaum für R1

4.2.4 Multicast: Zusammenfassung

Zwei Aspekte:

- Verwaltung von Multicast-Gruppen
  - An- und Abmelden von Teilnehmern (IGMP)
  - Wahl der Multicast-Adresse (durch *out-of-band*-Mechanismen)
- Multicast-Routing
  - Verteilung der Pakete über aufspannenden Baum
    - gemeinsamer Baum für alle Quellen
    - quellenspezifische Bäume
  - Erweiterung existierender Routing-Protokolle oder Protokoll-unabhängiger Multicast

Erinnerung: IP-Routing

- IP-Adressen sind aufgeteilt in
  - Netzadresse
  - Hostadresse (und ggf. Subnetz-Adresse)
- Router im Internet betrachten nur Netzadresse
  - Vorteil: bessere Skalierbarkeit
  - Problem: Host ist nur in „seinem“ Netz erreichbar
- Mobile Rechner (Laptops) werden in verschiedenen Netzen betrieben
  - neue IP-Adresse über DHCP ist nicht immer eine Lösung
    - bestehende Verbindungen werden unterbrochen

Ziele von *Mobile IP* (IETF RFC 3344)

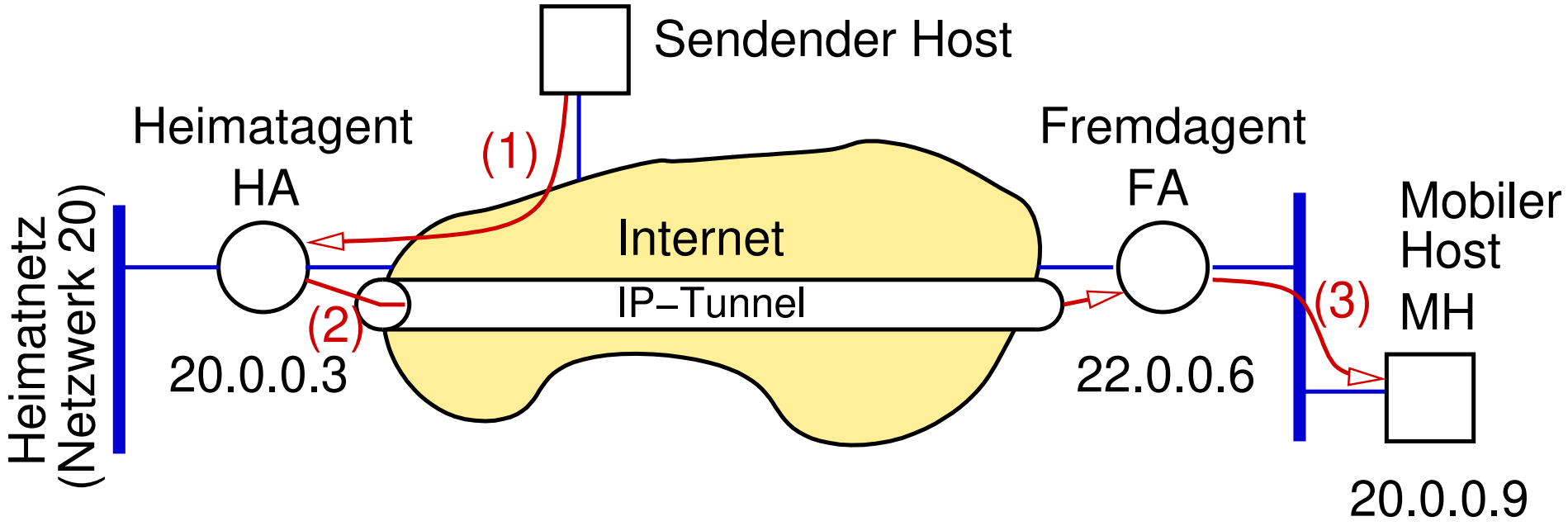
- Rechner kann (drahtloses) Netz wechseln („*Roaming*“)
  - ohne IP-Adresse zu wechseln
  - ohne Abbruch existierender Verbindungen
- Lösung darf keine Änderung
  - der über IP liegenden Software der mobilen Hosts
  - einer Vielzahl von Internet-Routern

  benötigen

Funktionsweise

- Ein bzw. zwei Router mit speziellen Fähigkeiten
  - **Heimatagent** (HA): im Heimatnetz des mobilen Hosts
    - permanente IP-Adresse (Heimatadresse) des mobilen Hosts liegt im Netz dieses Routers
  - **Fremdagent** (FA): im aktuellen Netz des MH
- HA und FA senden regelmäßig *Advertisements*
  - enthalten IP-Adresse des Routers
- Im Heimatnetz: mobiler Host (MH) erhält Adresse des HA
- Im Fremdnetz:
  - MH registriert sich bei FA, sendet Adresse des HA
  - FA sendet c/o-Adresse des MH (i.d.R. Adr. des FA) an HA

Routing eines Pakets an mobilen Host



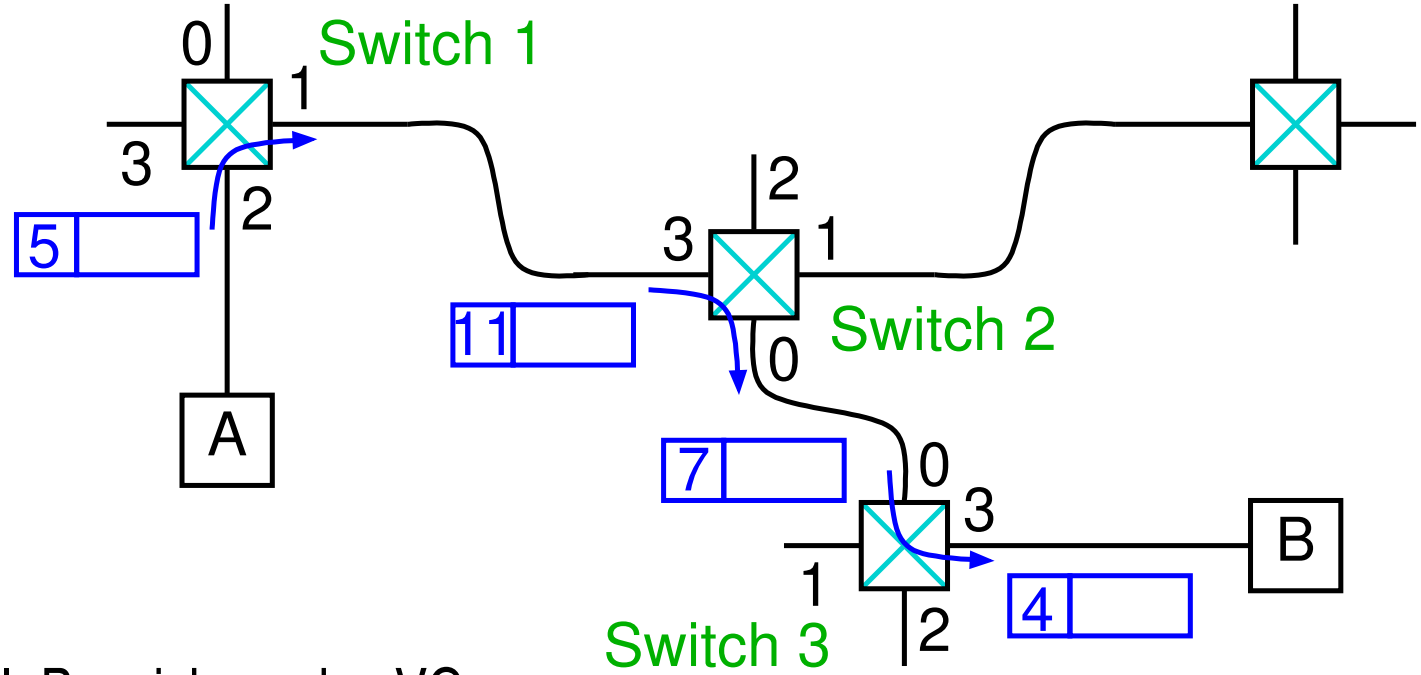
1. Host sendet an MH: Paket wird an HA geroutet
2. HA sendet Paket über IP-Tunnel an c/o-Adresse (d.h. FA)
3. FA sendet Paket an MH (über MAC-Adr. aus Registrierung)

Anmerkungen zum Routing

- Was, wenn das Paket nicht über HA ins Heimatnetz kommt?
  - z.B. Sender im Heimatnetz oder zweiter Router
  - Lösung: ***Proxy ARP***
    - HA sendet ARP-Paket (IP-Adr. MH, MAC-Adr. HA)
    - ohne Anfrage durch Host / Router: ***Gratuitous ARP***
- MH kann selbst die Funktion des FA übernehmen
- Optimierung: HA kann Sender anweisen, Folgepakete (über IP-Tunnel) direkt an FA zu senden (IPv6 *Binding-Update*)
  - falls sich MH weiterbewegt:
    - *Binding-Warning* durch FA, wenn Paket eintrifft
    - zusätzlich: begrenzte Lebenszeit (falls MH selbst FA ist)

Erinnerung: Virtuelle Leitungsvermittlung

- Kurze, link-spezifische Label statt langer Zieladresse



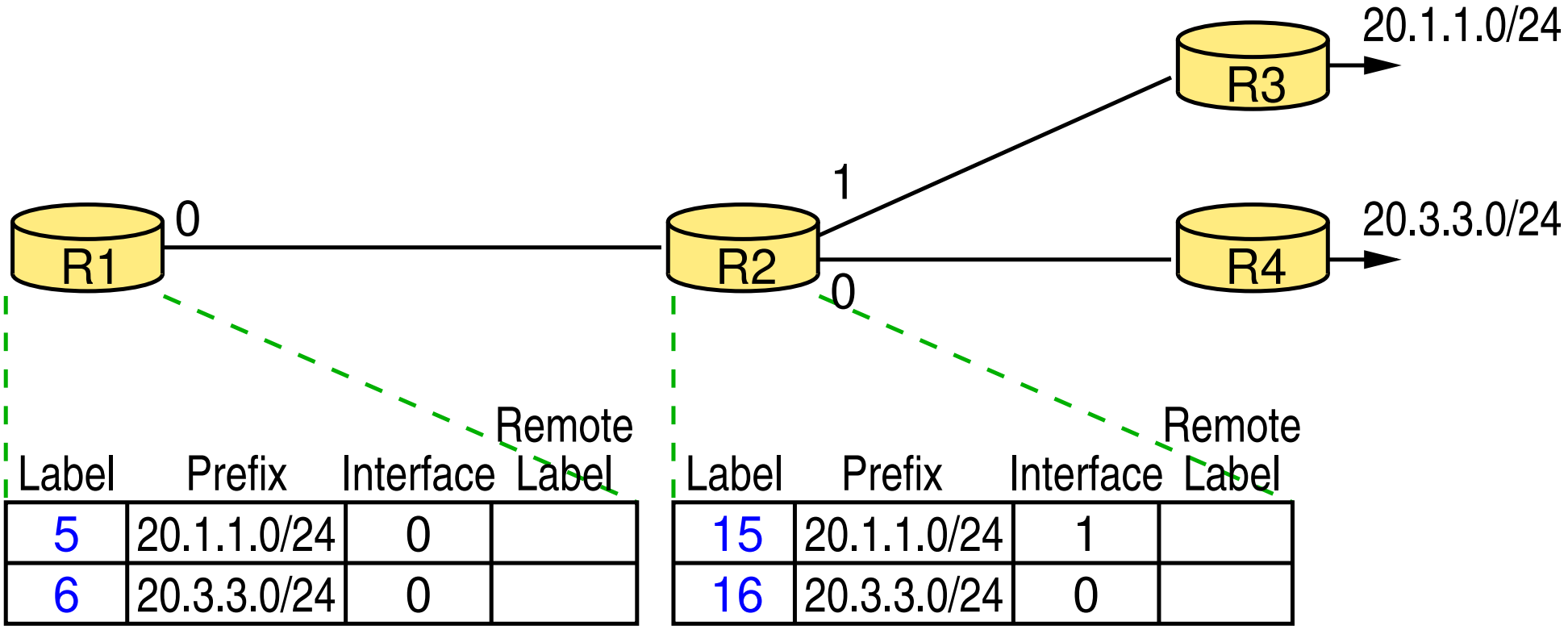
VCI: Bezeichner des VC

	Eingangsport	Eingangs–VCI	Ausgangsport	Ausgangs–VCI
Switch 1:	2	5	1	11
Switch 2:	3	11	0	7
Switch 3:	0	7	3	4

Ziel von MPLS (IETF RFC 3031)

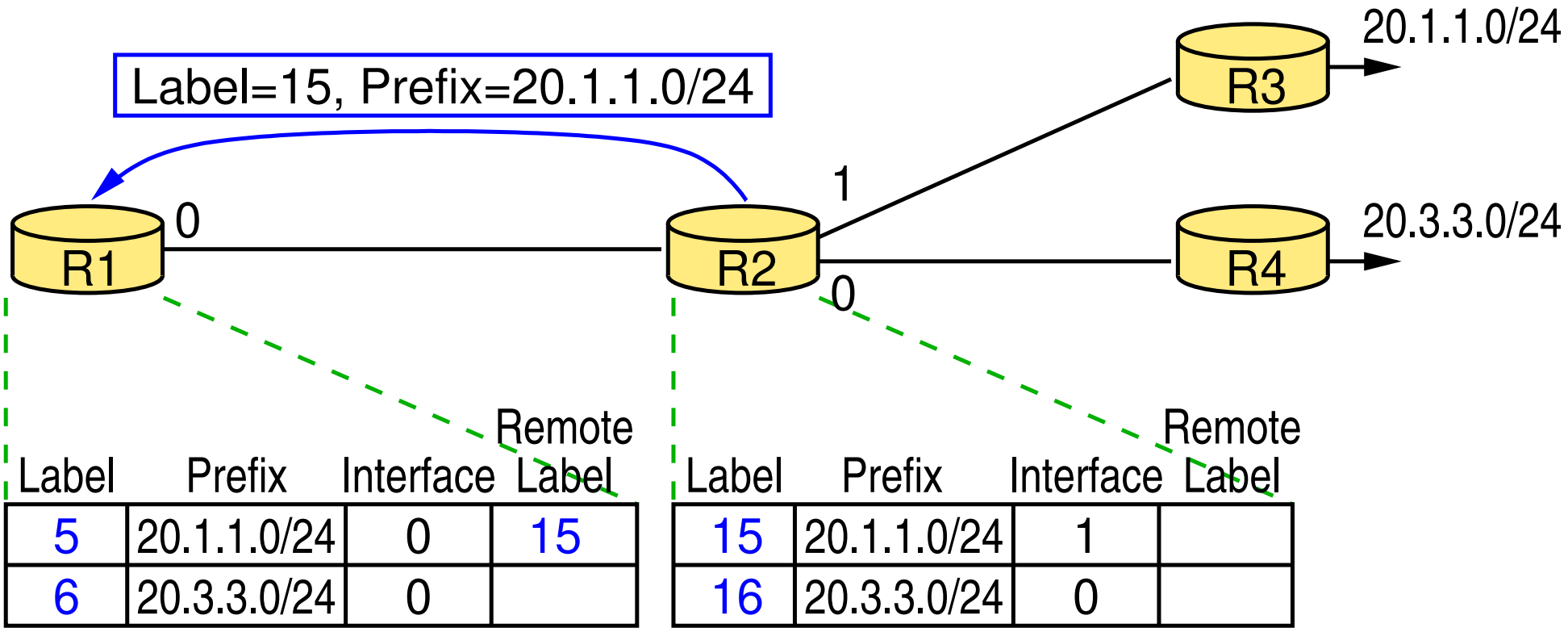
- Vorteile der virtuellen Leitungsvermittlung für IP nutzen
- Ursprüngliche Motivation: effizientere Weiterleitung
  - IP: Suche des längsten Präfixes (CIDR!) aufwendig
  - Label ist typischerweise Index in Weiterleitungstabelle
    - schnelle Weiterleitung, Hardware-Implementierung
- Einsatz von MPLS heute:
  - Weiterleitung von IP-Paketen entlang expliziter Routen
  - Realisierung von Tunneln und virtuellen privaten Netzen
  - IP-Unterstützung für Switches, deren Hardware keine IP-Pakete verarbeiten kann

Funktionsprinzip von MPLS



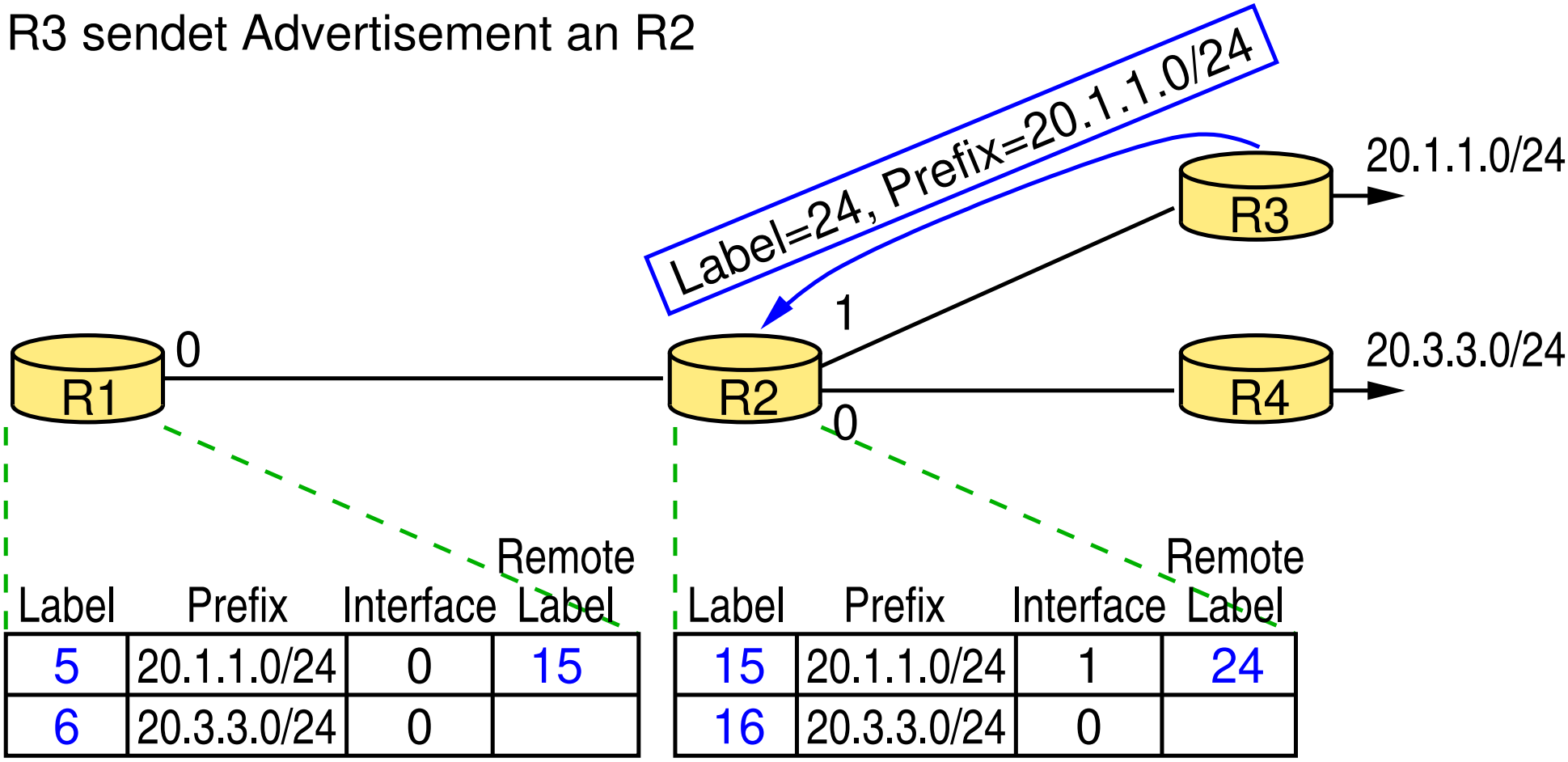
Funktionsprinzip von MPLS

R2 sendet Advertisement an R1



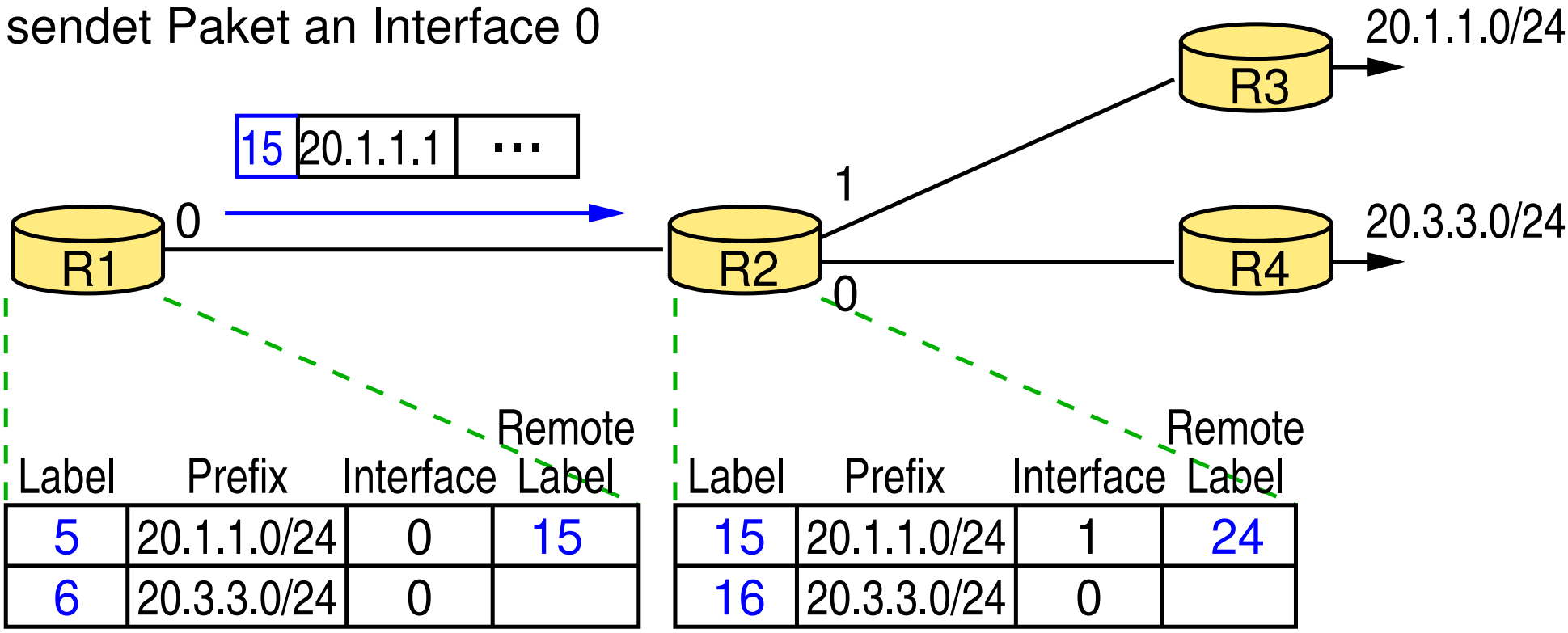
Funktionsprinzip von MPLS

R3 sendet Advertisement an R2



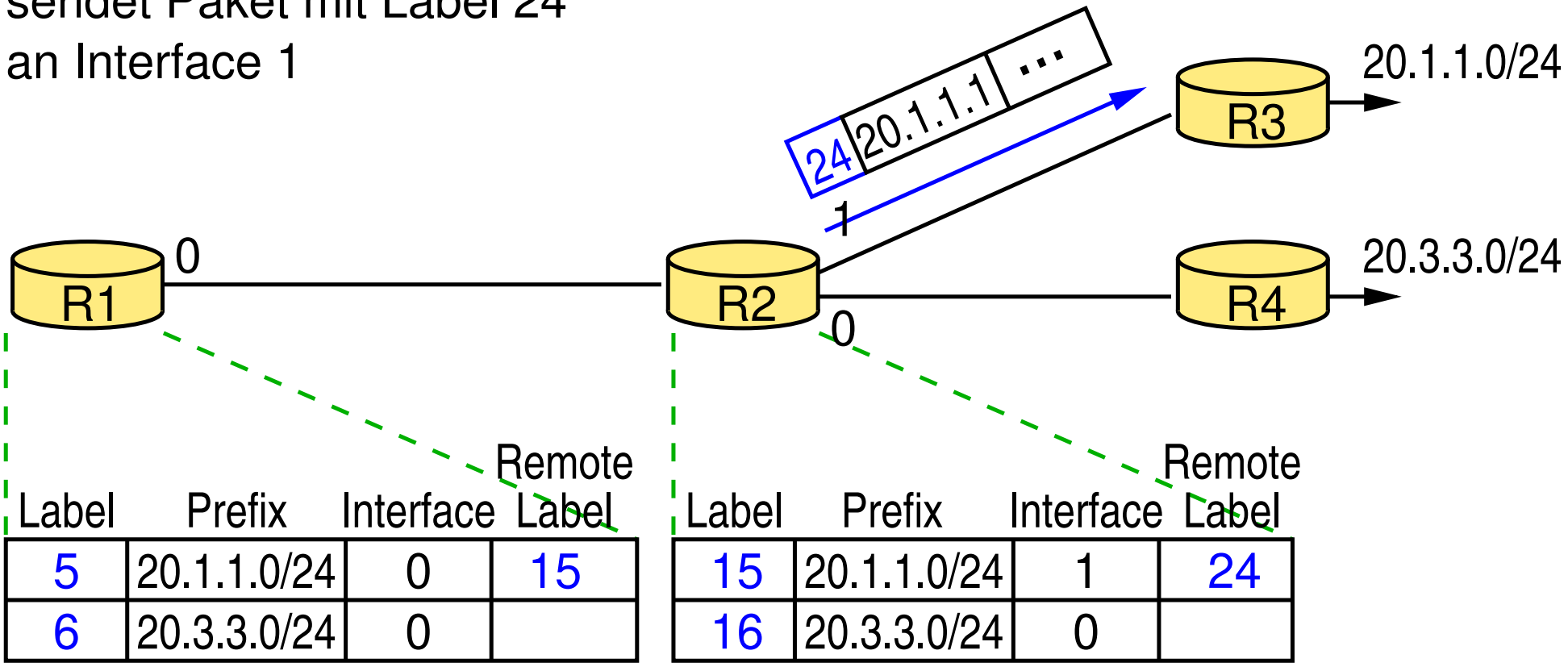
Funktionsprinzip von MPLS

R1 (Label Edge Router, LER)
erhält Paket, fügt Label an,
sendet Paket an Interface 0



Funktionsprinzip von MPLS

R2 betrachtet nur Label,
sendet Paket mit Label 24
an Interface 1



Label	Prefix	Interface	Remote Label
5	20.1.1.0/24	0	15
6	20.3.3.0/24	0	

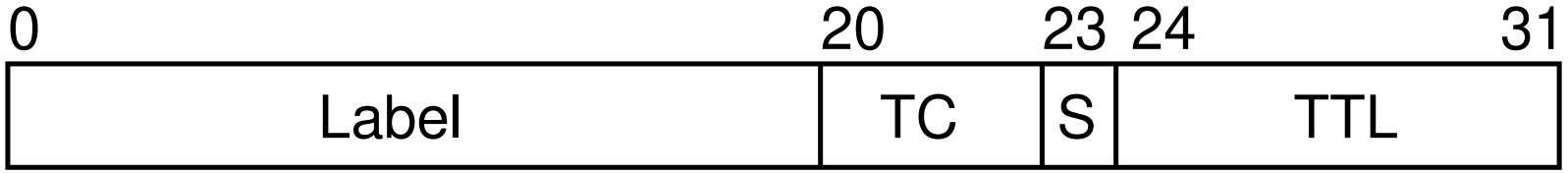
Label	Prefix	Interface	Remote Label
15	20.1.1.0/24	1	24
16	20.3.3.0/24	0	

Einfügen des Labels

- Bei den meisten Schicht-2-Protokollen (Ethernet, PPP, ...):
  - Einfügen zwischen Header von Schicht 2 und IP-Header:

  - MPLS ist „Schicht 2,5-Protokoll“

- Aufbau des MPLS-Headers (*MPLS Shim Header*):



  - **TC:** Traffic Class (für *Quality-of-Service*)
  - **S:** *Bottom of Stack*, kennzeichnet letztes Label

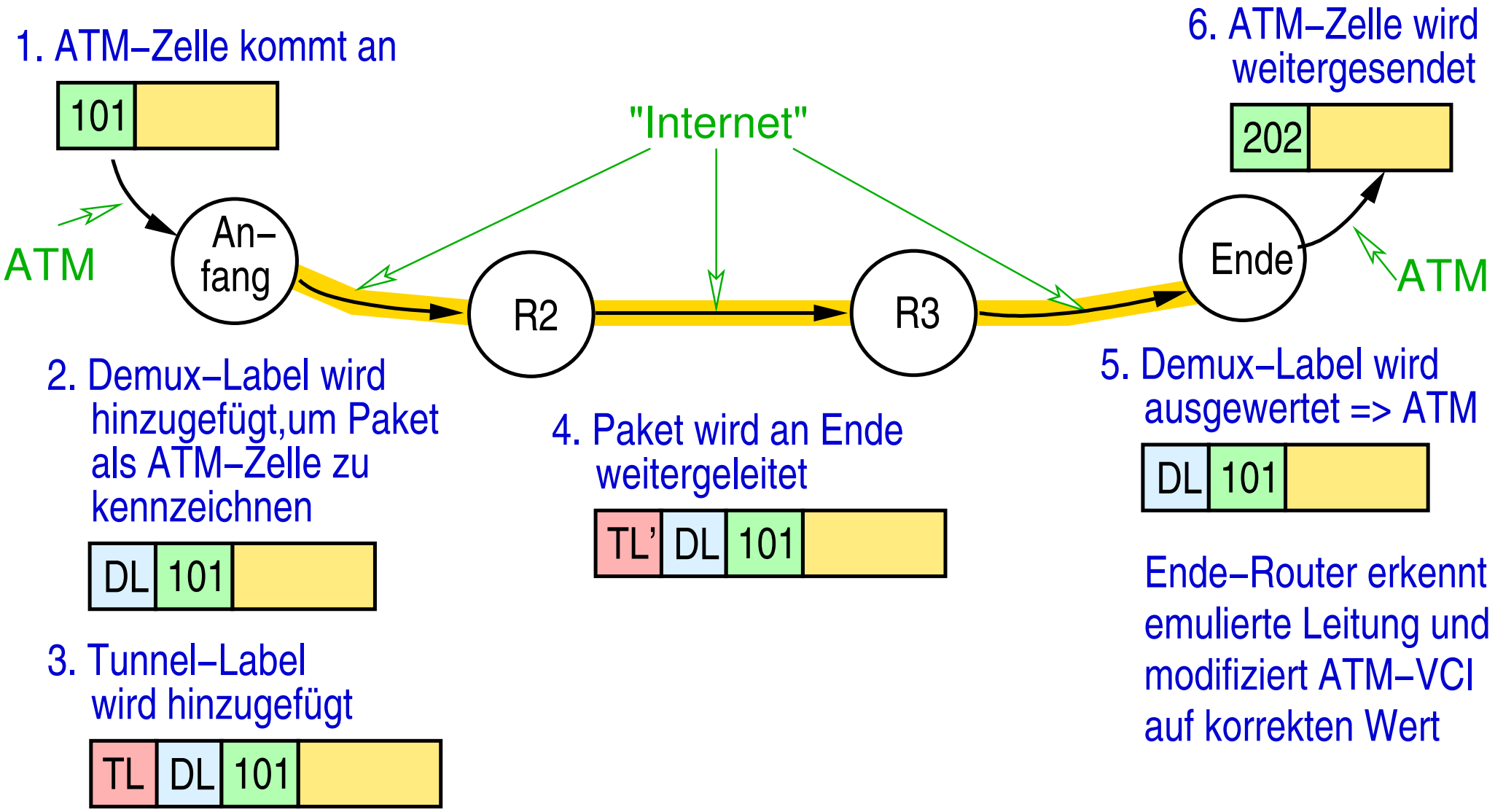
Explizite Routen

- MPLS ermöglicht Festlegung expliziter Pfade
  - analog zur virtuellen Leitungsvermittlung
  - Festlegung der Pfade z.B. über *Resource Reservation Protocol* (RSVP, siehe später: QoS)
    - RSVP-Nachrichten führen zur Reservierung von Puffer und Bandbreite auf dem ausgewählten Pfad
- Damit möglich z.B.:
  - quellenabhängige Routen
  - schnelles Rerouting bei Ausfall von Links
  - Dienstgütegarantien, z.B.:
    - Auswahl einer Route mit bestimmter Bandbreite
    - Nutzung der Route, auf der Ressourcen reserviert wurden

Tunnel und VPNs

- Prinzip wie bei IP-Tunnel:
  - Paket wird am Eingang des Tunnels mit MPLS-Label versehen, am Ausgang wird Label entfernt
- Vorteil gegenüber IP: Label ist kürzer als IP-Header
- Zum Demultiplexen am Tunnelende: weiteres MPLS-Label
  - letztes Label durch spezielles Bit gekennzeichnet
  - Tunnel damit für mehrere Verbindungen nutzbar
- Anwendung z.B.
  - Emulation von Schicht-2-Diensten, z.B. ATM über Internet
  - Realisierung von Schicht-3-VPNs
    - virtuelle, private IP-Netzwerke über Internet

Beispiel: Tunnelling von ATM-Zellen



Fazit

- MPLS kombiniert
  - label-basierte Weiterleitung der virtuellen Leitungsvermittlung

  mit
  - Routing- und Kontrollprotokollen von IP-Datagramm-Netzen
- Ergebnis:
  - Netzwerkklasse irgendwo zwischen leitungs- und datagrammvermittelnden Netzen

IP-Routing: Spezielle Aspekte

- IP Multicast
  - IGMP: Anmeldung und Abmeldung
    - Router erfährt, welche Gruppen im LAN vertreten sind
  - Link-State-Multicast
    - Berechnung spannender Bäume mit kürzesten Wegen
  - Distanzvektor-Multicast (*Reverse Path Multicast*)
    - Broadcast mit Zyklenvermeidung und *Pruning*
  - *Protocol Independent Multicast* (PIM), *Sparse Mode*
    - Wege der *Join*-Nachrichten ergeben Multicast-Baum
    - zunächst mit fester Wurzel (Rendezvous-Punkt)
    - Optimierung: quellenspezifische *Joins* bzw. Bäume

IP-Routing: Spezielle Aspekte ...

- Mobile IP
  - Heimatagent (HA) leitet Pakete über IP-Tunnel an Router des Fremdnetzes (oder mobilen Host (MH) selbst)
  - Proxy ARP: HA fängt Pakete an MH im lokalen Netz ab
- MPLS (*Multiprotocol Label Switching*)
  - Kombination von IP Datagramm-Vermittlung mit Weiterleitung aus virtueller Leitungsvermittlung
  - IP-Paket wird Label vorangestellt; Weiterleitung nur aufgrund des Labels
  - Einsatz: explizite Routen, Tunnels und VPN, ATM-Switches als *Label Switching Router*

Rechnernetze II

SoSe 2025

5 VPN, IP-Tunnel und IPsec

Inhalt

- Überblick
- Details
- Konfiguration und Verbindungsaufbau
- Zusammenfassung
- Tanenbaum, Kap. 5.6.8, 8.6.1
- Peterson, Kap. 4.3.5, 8.3.4
- Kurose/Ross, Kap. 4.7, 7.8
- William Stallings: *Cryptography and Network Security, 3rd Edition*, Prentice Hall, 2003, Kap. 16
- CCNA, Kap. 7

Vorbemerkung:
Welche Schicht sollte Sicherheit implementieren?

- Vermittlungsschicht

- Anwendungsschicht

Vorbemerkung: Welche Schicht sollte Sicherheit implementieren?

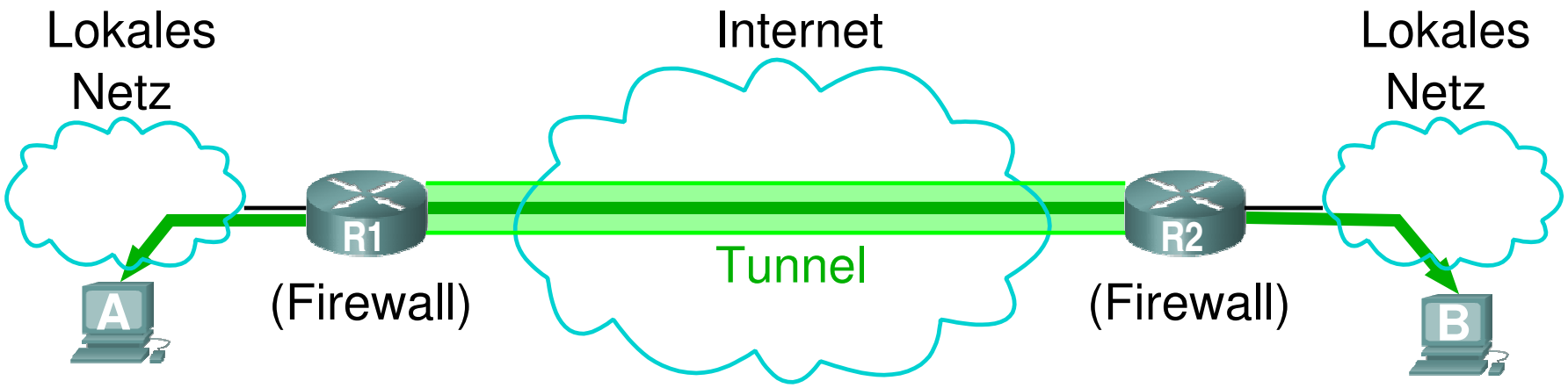
- Vermittlungsschicht
  - \+ Transparent für Anwendungen
  - \+ Sicherheitsvorgaben durch Administrator
  - \+ Erschwert Verkehrsflußanalysen
  - – Muß i.a. vom gesamten Netz unterstützt werden
  - – Abhören zw. Anwendung und Vermittlungsschicht möglich
- Anwendungsschicht
  - \+ Keine Anforderungen an Netzinfrastruktur
  - – Schlüsselverwaltung in Anwendungen problematisch
  - – Keine Sicherung gegen Verkehrsflußanalysen

IPsec (Secure IP)

- Ziel: Sichere Übertragung von IP-Paketen
- Unterstützung optional bei IPv4, vorgeschrieben bei IPv6
- Zwei Sicherheitsprotokolle
  - *Authentication Header* (AH) Protokoll
  - *Encapsulating Security Payload* (ESP) Protokoll
  - Kombination AH + ESP möglich
- Zwei Betriebsarten
  - Transport-Modus: Sicherheit für Protokolle über IP
    - keine Vertraulichkeit des IP-Headers
  - Tunnel-Modus: Sichere Verbindung zwischen zwei Routern

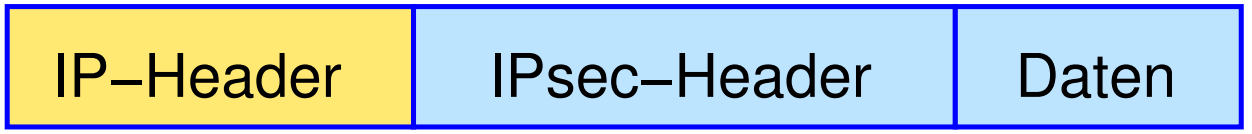
Tunnel-Modus (mit ESP)

- Ermöglicht die Realisierung sicherer VPNs



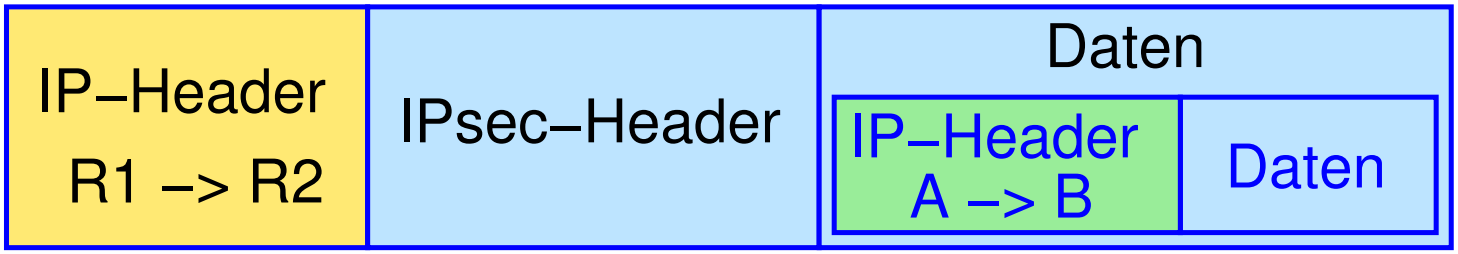
- Im Tunnel: gesamtes IP-Paket verschlüsselt und authentifiziert
  - Daten können nicht gelesen oder verändert werden
  - Quelle und Ziel können nicht ermittelt werden
    - Schutz vor Verkehrsflußanalyse

Aufbau eines IP-Pakets mit IPsec



- Weiterleiten der Pakete über „normales“ Internet möglich

Aufbau eines IP-Pakets im Tunnel-Modus von IPsec



- Sicherer Tunnel zwischen Router $R1$ und $R2$
  - Teilnehmer A und B müssen IPsec nicht implementieren

Security Association (SA, RFC 4301)

- Unidirektionale „Verbindung“
- Fasst Parameter für ein Sicherheitsprotokoll (AH oder ESP) zusammen, z.B.:
  - Betriebsart (Transport / Tunnel)
  - kryptographische Parameter
    - Chiffren, Schlüssel, Schlüssel-Lebensdauer, ...
  - aktuelle Paket-Sequenznummer (für Replay-Schutz)
  - Lebensdauer der SA
- SA eindeutig identifiziert durch IP-Adresse des Partners, Sicherheitsprotokoll und *Security Parameter Index* (SPI)
  - mehrere SAs pro Partner möglich

AH-Protokoll (RFC 4302, 4305, 8221)

- Authentifiziert das gesamte IP-Paket
  - IP-Header und Erweiterungs-Header
    (außer Felder, die sich bei Übertragung ändern)
  - AH-Header (außer Authentifizierungsdatenfeld)
  - Nutzdaten des IP-Pakets
- Aufbau des AH-Headers:

NextHeader	HeaderLen	Reserviert
Security Parameter Index (SPI)		
Sequenznummer		
Authentifizierungsdaten variabler Länge (= HMAC–Wert)		

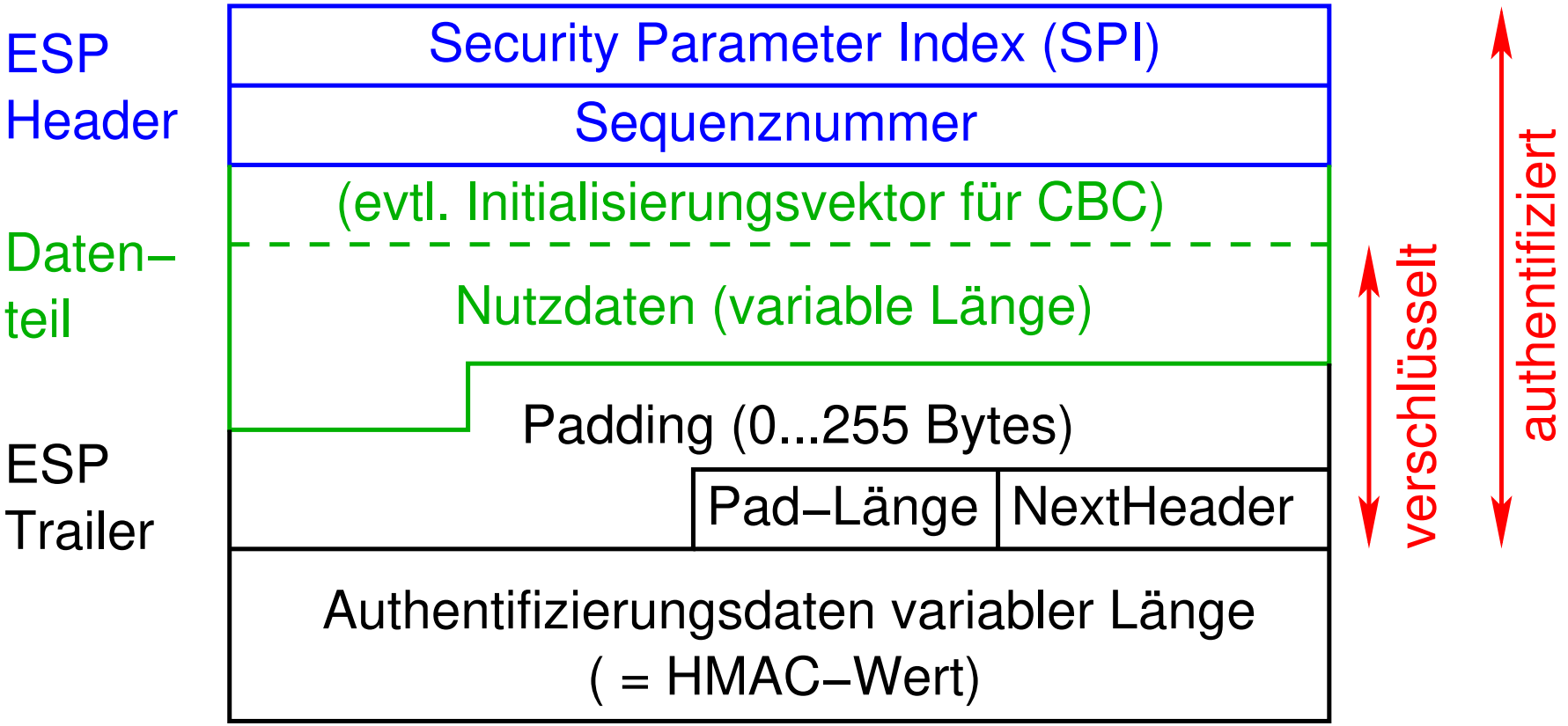
AH-Protokoll ...

- SPI: zur Idenitifizierung der *Security Association* (SA)
- Sequenznummer dient zur Abwehr von Replay
- HMAC = auf Hashfunktion basierender MAC (*Message Authentication Code*)
  - sichert Authentizität und Integrität
  - $HMAC(M, K) = H(K \oplus \texttt{opad} \parallel H(K \oplus \texttt{ipad} \parallel M))$
    $\texttt{ipad} = \mathbf{0011\ 0110\ ...\ 0011\ 0110_2}$
    $\texttt{opad} = \mathbf{0101\ 1010\ ...\ 0101\ 1010_2}$
- Verschiedene Hashfunktionen möglich
  - jede IPsec-Implementierung muß SHA2-256-128 unterstützen (SHA2 mit 256 Bit Schlüssellänge, Hashwert mit 128 Bit)
  - (derzeit muss auch noch SHA1-96 implementiert werden)

ESP-Protokoll (RFC 4303, 4305, 8221)

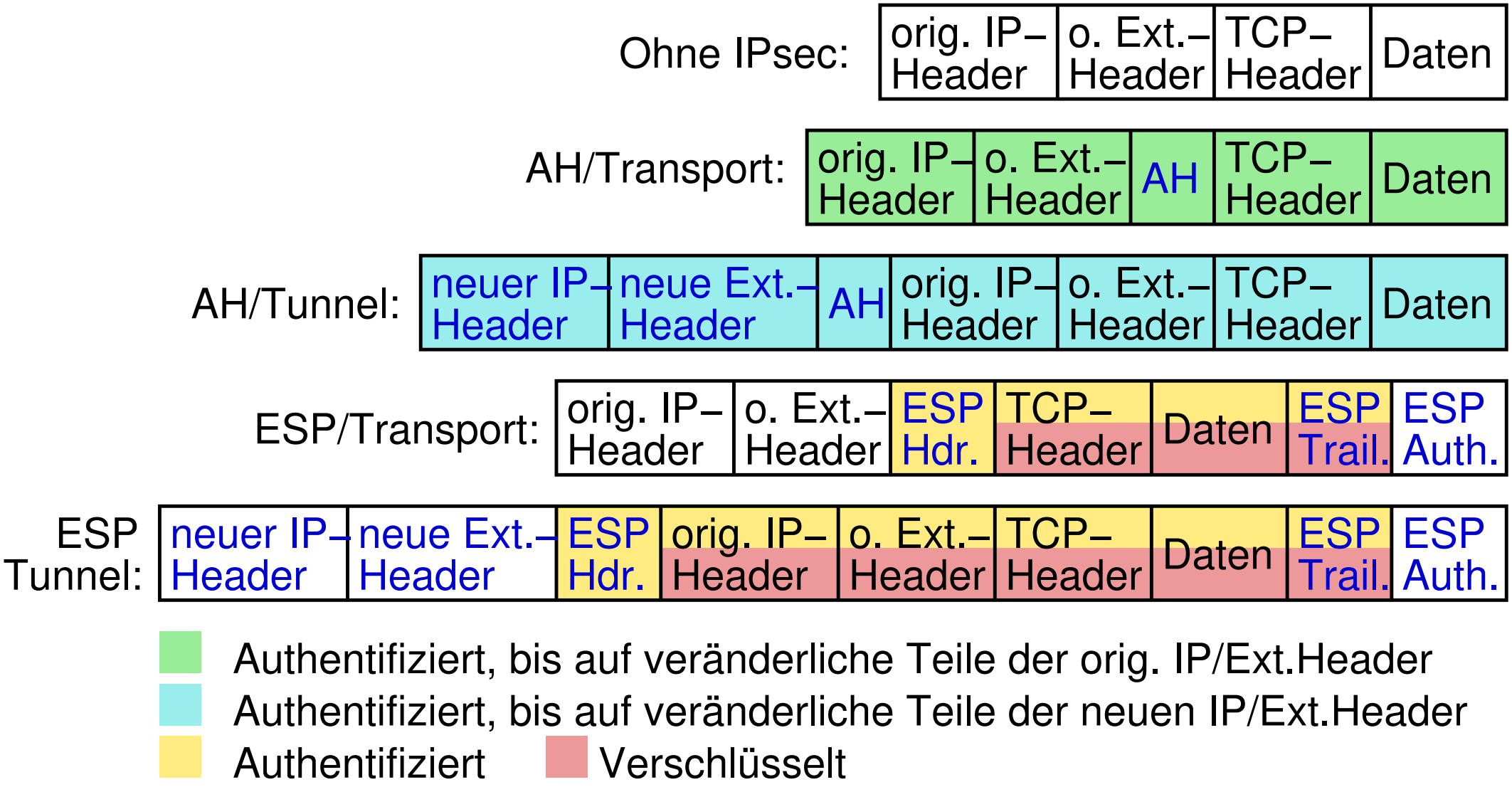
- Verschlüsselung und Authentifizierung des **Datenteils** eines IP-Pakets
  - Authentifizierung ist optional
- Verschiedene Verschlüsselungsverfahren möglich
- Jede IPsec-Implementierung muß unterstützen:
  - AES-CBC und AES-GCM-16 zum Verschlüsseln
    - AES: Blockchiffre, Schlüssellänge 128, 192 oder 256 Bit
    - CBC (*Cipher Block Chaining*): Chiffretexte der einzelnen Datenblöcke hängen voneinander ab
    - GCM (*Galois/Counter Mode*): Verschlüsselung und Integritätssicherung, sehr effizient realisierbar
  - HMAC-SHA2-256-128 u. HMAC-SHA1-96 zur Authentifizierung

Paketformat beim ESP-Protokoll



- Padding wegen Verwendung von Blockchiffren

Zusammenfassung: Paketformate (mit IPv6)



Zusammenfassung: Betriebsarten und Protokolle

Protokoll:	AH		ESP	
Betriebsart:	Transp.	Tunn.	Transp.	Tunn.
Verschlüssel. orig. IP-Header	−	−	−	+
Authentifiz. orig. IP-Header	+	+	−	(+)
Verschlüsselung Nutzdaten	−	−	+	+
Authentifizierung Nutzdaten	+	+	(+)	(+)
End-to-End Sicherheit	+	−	+	−

- Kombinationen (Verschachtelung) möglich!
  - z.B. AH im Transport-Modus über ESP Tunnel

Konfiguration von IPsec: Strategie-Datenbank

- Jeder IPsec-Knoten enthält eine Strategie-Datenbank (*Security Policy Database*, SPD)
- SPD legt für jede ein- und ausgehende Verbindung fest, wie Pakete zu behandeln sind:
  - unverschlüsselte Weiterleitung
  - Verwerfen
  - Anwendung von IPsec
    - Verschlüsselung und/oder Authentifizierung
    - Sicherheits-Parameter (Chiffren, Schlüssellänge / -lebensdauer, Tunnel- / Transportmodus, ...)

    $\Rightarrow$ Nutzung / Erzeugung einer *Security Association* (SA)
- Analog zu Filtertabellen in Firewalls

Beispieleintrag in Strategie-Datenbank

src: 192.168.2.1 - 192.168.2.10
dest: 192.168.3.1
ipsec-action: *esp* req *cipher* aescbc
integrity hmacsha1 *keylen* 128
expiry (seconds) 60
transport
ah req *integrity* hmacsha2 *keylen* 256
expiry (seconds) 60
tunnel

- Verschlüsselung/Authentifizierung der Daten durch ESP im Transport-Modus
- Zusätzlich Authentifizierung des Gesamtpakets durch AH im Tunnel-Modus

Verbindungsaufbau (Beispiel)

1. Paket soll gesendet werden
2. Nachsehen in Strategie-Datenbank: noch keine SA vorhanden
3. Erzeugen einer oder mehrerer SAs durch *Internet Security Association and Key Management Protocol* (ISAKMP)
   - wechselseitige Authentifizierung mit Schlüsselaustausch
     - *Internet Key Exchange* Protokoll (IKE, RFC 7296)
     - Authentifizierung z.B. über *Pre-shared Key*
   - Aushandeln der Sicherheitsattribute
4. Speichern der SA
   - Löschung nach Ablauf der Lebensdauer
5. Senden des Pakets

5.3 Zusammenfassung

- IPSec: Sicherheit auf der Vermittlungsschicht
- Zwei Protokolle:
  - AH: Authentizität für gesamtes IP-Paket
  - ESP: Authentizität und/oder Vertraulichkeit für Nutzdaten
    - Tunnel-Modus: auch Original-IP-Header verschlüsselt
- Basis für Kommunikation: *Security Association* (SA)
  - unidirektionale Verbindung, legt Sicherheitsparameter fest
  - Erzeugung der SA über ISAKMP / IKE
    - incl. Schlüsselaustausch und Partner-Authentifizierung

Bewertung

- Keine Verbindlichkeit (digitale Unterschrift)
- Schwierige Konfiguration (Strategie-Datenbank)
- IP ist verbindungsloses Protokoll
  - Integrität zunächst nur für einzelne Pakete garantiert
  - $\Rightarrow$ Zusatzmechanismen (Sequenznr., Anti-Replay-Fenster)
- IPsec erlaubt hostbasierte Schlüsselvergabe
  - selber Schlüssel für alle Verbindungen zw. zwei Rechnern
  - eröffnet Angriffsmöglichkeiten
    - nutze einen Rechner als Entschlüsselungsdienst
- IPsec derzeit v.a. für sichere Tunnels (VPNs) eingesetzt

Rechnernetze II

SoSe 2025

6 Überlastkontrolle und *Quality of Service*

Inhalt

- Erinnerung: Überlastkontrolle
- Überlastvermeidung
  - DECbit, RED, quellenbasierte Überlastvermeidung
- *Quality of Service*
  - Anforderungen, *IntServ*, *DiffServ*

- Peterson, Kap. 6.1-6.5
- Tanenbaum, Kap. 5.3, 5.4

6.1 Erinnerung: Überlastkontrolle

- **Überlast**: Zwischenknoten des Netzes (Router) verwerfen Pakete, da die Puffer voll sind
  - es kommen mehr Pakete an als weitergeleitet werden können
- Ziel der **Überlastkontrolle**: Überlastsituation erkennen und schnellstmöglich beenden
  - ansonsten Gefahr der Eskalation
- In TCP: *Additive Increase / Multiplicative Decrease*
  - mit jedem ankommenden ACK: Überlastfenster etwas erhöhen
  - bei Paketverlust: Überlastfenster halbieren
    - erkannt durch Timeout oder Duplikat-ACKs
  - Sendefenster bestimmt, wieviele Daten gesendet werden dürfen, bevor auf ein ACK gewartet werden muß
- Details: ☞ **RN-I, Kap. 7.6**

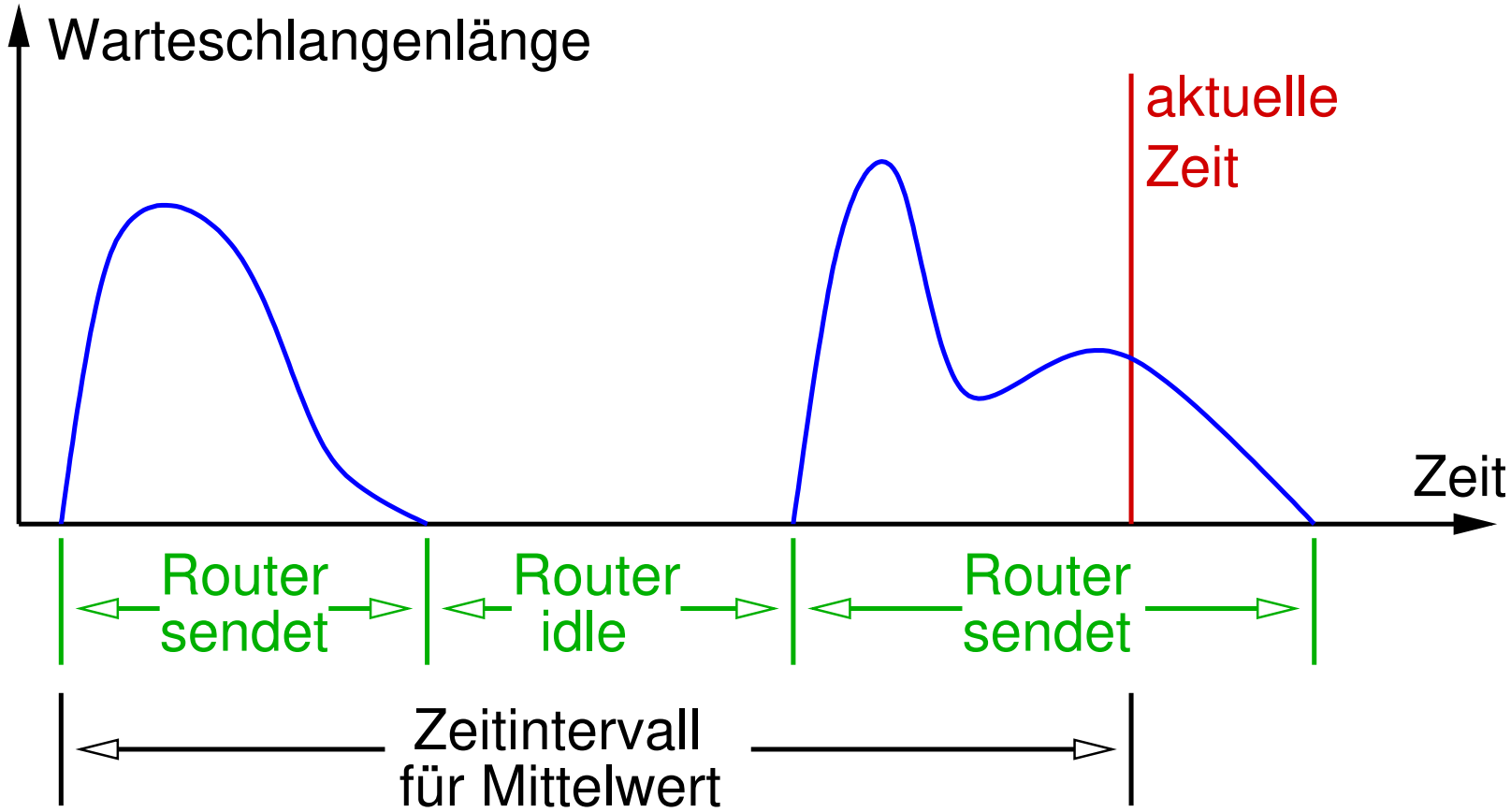
6.2 Überlastvermeidung

- Ziel der **Überlastvermeidung**: reduziere Senderate, **bevor** Überlast (d.h. Paketverlust) entsteht
  - d.h. bei ersten Anzeichen einer drohenden Überlast
- Alternativen zur Erkennung drohender Überlast:
  - Router-zentrisch
    - Router melden an Hosts, wenn sie die Senderate reduzieren sollen
    - Basis: mittlere Länge der Paket-Warteschlange im Router
    - Beispiele: DECbit, RED
  - Host-zentrisch (quellenbasierte Überlastvermeidung)
    - Hosts beobachten Anzeichen drohender Überlast selbst
    - z.B.: steigende Latenz, sinkender Durchsatz
    - Beispiel: TCP Vegas

6.2.1 DECbit

- Verwendet in DECs *Digital Network Architecture*
  - verbindungsloses Netz mit verbindungsorientiertem Transportprotokoll (analog zu TCP über IP)
- Router überwacht mittlere Länge der Warteschlange (Puffer)
- Bei Überschreiten eines Grenzwerts:
  - Router setzt ein Überlastbit im Paketheader
  - Empfänger kopiert dieses Überlastbit in sein ACK
  - Sender reduziert seine Senderate

Mittlere Warteschlangenlänge



- Überlastbit wird gesetzt, wenn Mittelwert ≥ 1

Reduktion der Senderate

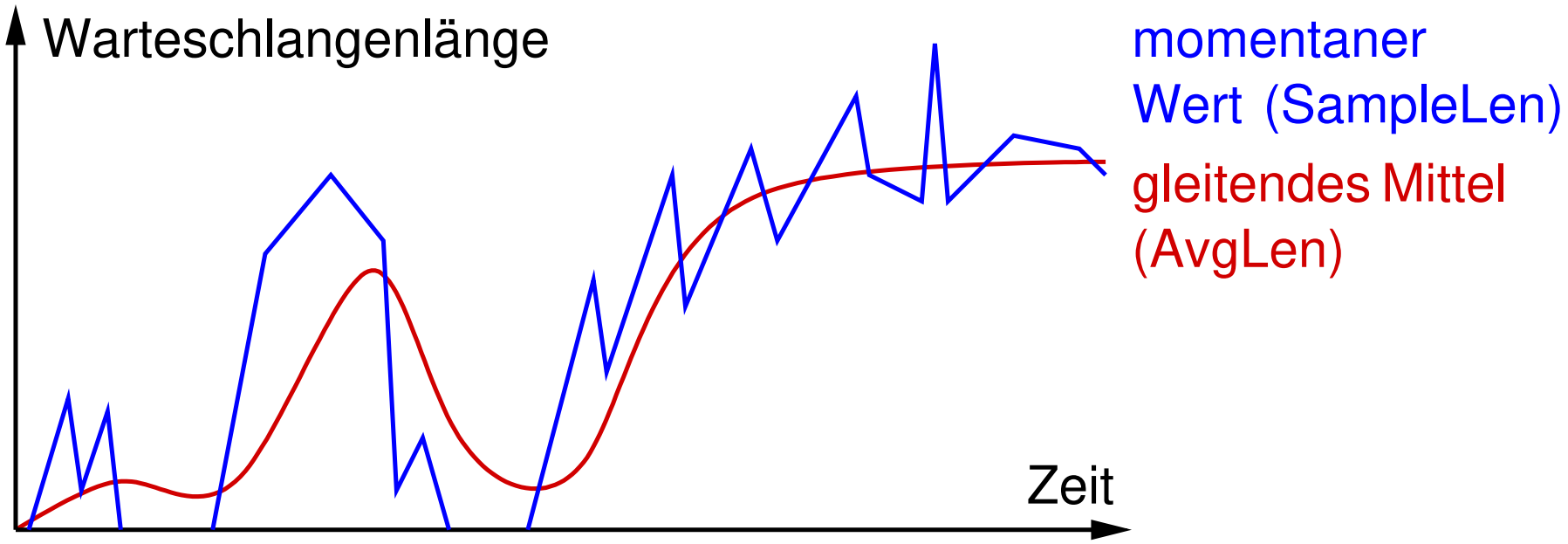
- Überlastfenster mit *Additive Increase / Multiplicative Decrease*
- Host bobachtet, wieviele Pakete im letzten Fenster zu ACKs mit gesetztem Überlastbit führten
  - mehr als 50%: Fenster auf 7/8 der Größe reduzieren
  - weniger als 50%: Fenstergröße um 1 erhöhen

6.2.2 *Random Early Detection* (RED)

- Floyd/Jacobson 1993, entwickelt für den Einsatz mit TCP
- Gemeinsamkeit mit DECbit
  - Router überwachen mittlere Warteschlangenlänge, Quellen passen Überlastfenster an
- Unterschiede:
  - kein explizites Feedback, sondern Verwerfen eines Pakets
    - Zusammenarbeit mit TCP-Überlastkontrolle: TCP halbiert bei Paketverlust das Überlastfenster!
    - Router verwirft also (einzelne) Pakete, bevor der Puffer wirklich voll ist (und alle verworfen werden müssten)
  - *Random Early Drop* statt ausschließlich *Tail Drop*
    - ab einer bestimmten Warteschlangenlänge, werden ankommende Pakete mit gewisser Wahrscheinlichkeit verworfen

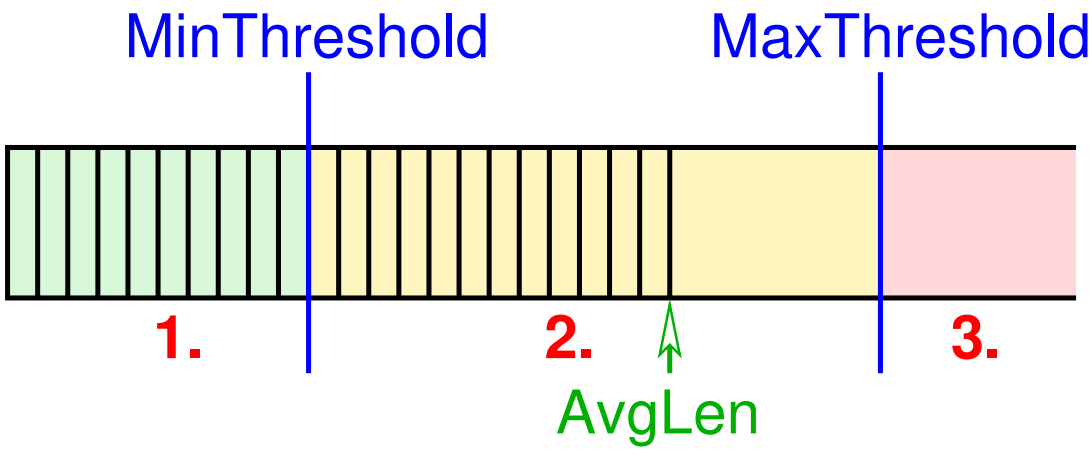
Gewichtetes, gleitendes Mittel der Warteschlangenlänge

- $\mathbf{AvgLen} = (1 - \mathbf{Weight}) \cdot \mathbf{AvgLen} + \mathbf{Weight} \cdot \mathbf{SampleLen}$
  - **SampleLen** z.B. immer bei Ankunft eines Pakets messen



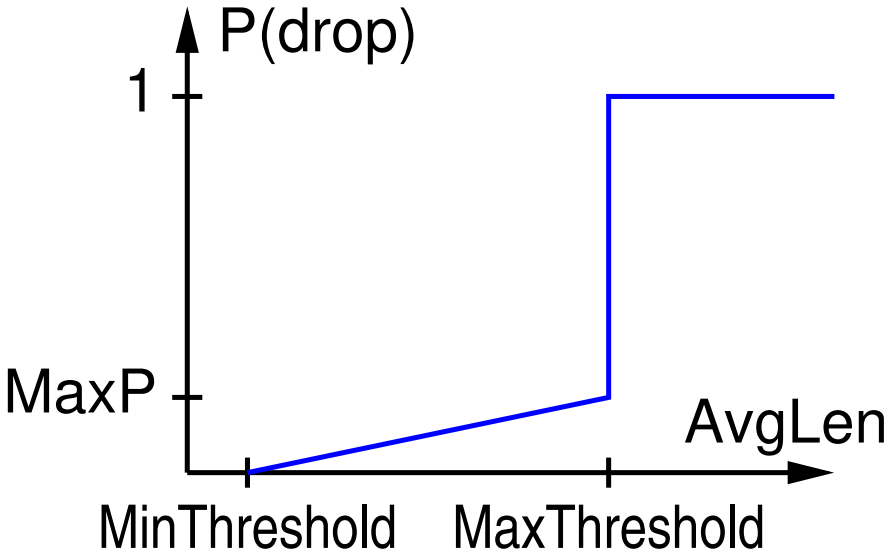
- Reaktion nur auf langlebige Überlast, nicht auf kurze *Bursts*
  - **Weight** so wählen, daß über ≥ 1 RTT gemittelt wird

Grenzwerte für mittlere Warteschlangenlänge



1. Falls **AvgLen** $\leq$ **MinThreshold**: stelle Paket in Warteschlange
2. Falls **MinThreshold** $<$ **AvgLen** $<$ **MaxThreshold**:
   - ➥ berechne Wahrscheinlichkeit **P**
   - ➥ verwirf ankommendes Paket mit Wahrscheinlichkeit **P**
3. Falls **AvgLen** $\geq$ **MaxThreshold**: verwirf ankommendes Paket

Berechnung der *Drop*-Wahrscheinlichkeit



$$\text{TempP} = \text{MaxP} \cdot \frac{\text{AvgLen} - \text{MinThreshold}}{\text{MaxThreshold} - \text{MinThreshold}}$$

$$P = \frac{\text{TempP}}{1 - \text{count} \cdot \text{TempP}}$$

- **count**: Anzahl der Pakete, die gepuffert (also nicht verworfen) wurden, seit **AvgLen** > **MinThreshold**
  - bewirkt gleichmäßige zeitliche Verteilung verworfener Pakete

Bemerkungen

- RED ist näherungsweise fair
  - Anzahl verworfener Pakete in etwa proportional zum Bandbreitenanteil eines Flusses
- Zur Wahl der Grenzwerte:
  - **MinThreshold** sollte groß genug sein, um Leitung auszulasten
  - oberhalb von **MaxThreshold** sollte noch genügend Speicherplatz für *Bursts* bleiben
  - Differenz größer als typische Erhöhung von **AvgLen** während einer RTT
    - im Internet typisch **MaxThreshold** = 2 · **MinThreshold**

6.2.3 Quellenbasierte Überlastvermeidung

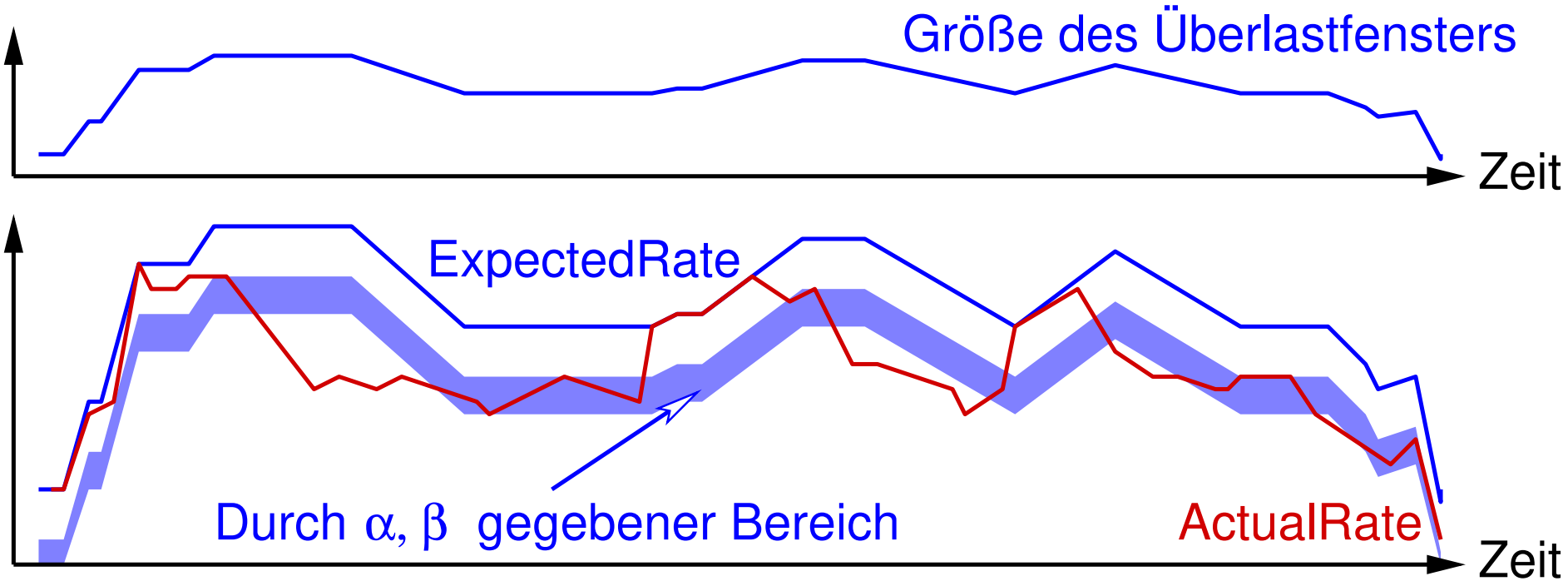
Wie kann eine Quelle drohende Überlast erkennen?

- Messung der RTT zwischen Paket und ACK
  - volle Warteschlangen in Routern bewirken höhere RTT
  - z.B.: verkleinere Überlastfenster auf 7/8, wenn gemessene RTT $\geq$ Mittelwert von bisheriger min. und max. RTT
- Vergleich von erreichtem mit erwartetem Durchsatz
  - erwarteter Durchsatz: Fenstergröße / RTT
    - pro RTT wird ein Fenster an Daten verschickt
    - maximal erreichbarer Durchsatz
  - bei hoher Auslastung:
    - bei größerem Fenster werden lediglich mehr Pakete in Routern gepuffert, aber nicht mehr übertragen

Überlastvermeidung in TCP Vegas

- Berechne **ExpectedRate** = **CongestionWindow** / **BaseRTT**
  - **BaseRTT**: Minimum der bisher gemessenen RTTs
- Berechne **ActualRate** einmal pro RTT
  - Datenmenge, die zwischen dem Abschicken eines Pakets und dem Empfang des zugehörigen ACK gesendet wurde
- Setze **Diff** = **ExpectedRate** - **ActualRate**
- Stelle Überlastfenster so ein, daß $\alpha \leq$ **Diff** $\leq \beta$:
  - falls **Diff** $< \alpha$: vergrößere Überlastfenster linear
  - falls **Diff** $> \beta$: verkleinere Überlastfenster linear
- α, β bestimmen min./max. zu belegenden Pufferplatz

Beispiel zu TCP Vegas

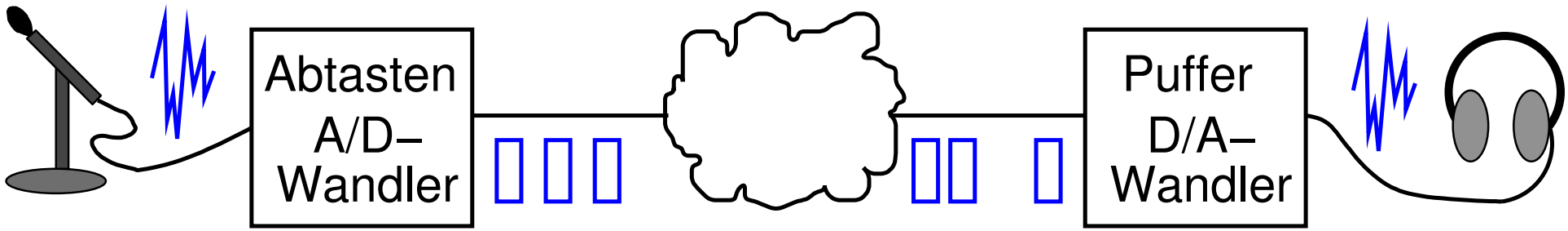


- **ActualRate** unterbalb des Bereichs: Gefahr, daß zu viele Pakete zwischengespeichert werden müssen
- **ActualRate** oberhalb: zu geringe Netzauslastung

6.3 *Quality of Service* (QoS)

- Paketvermittelnde Netze sollen neben traditionellen („elastischen“) Anwendungen auch Echtzeitanwendungen unterstützen, z.B.:
  - Audio- und Video, z.B. Videokonferenz
  - industrielle Steueraufgaben, z.B. Roboter
  - Datenbank-Aktualisierung über Nacht
- Dazu: Daten müssen **rechtzeitig** ankommen
- Netzwerk muß neben *Best Effort* bessere Dienste bieten
  - Zusicherung der zeitgerechten Übertragung
    - i.d.R. ohne Paketverlust und Neuübertragung
- **Dienstgüte** (***Quality of Service***)
  - verschiedene Dienstklassen mit verschiedenen Garantien

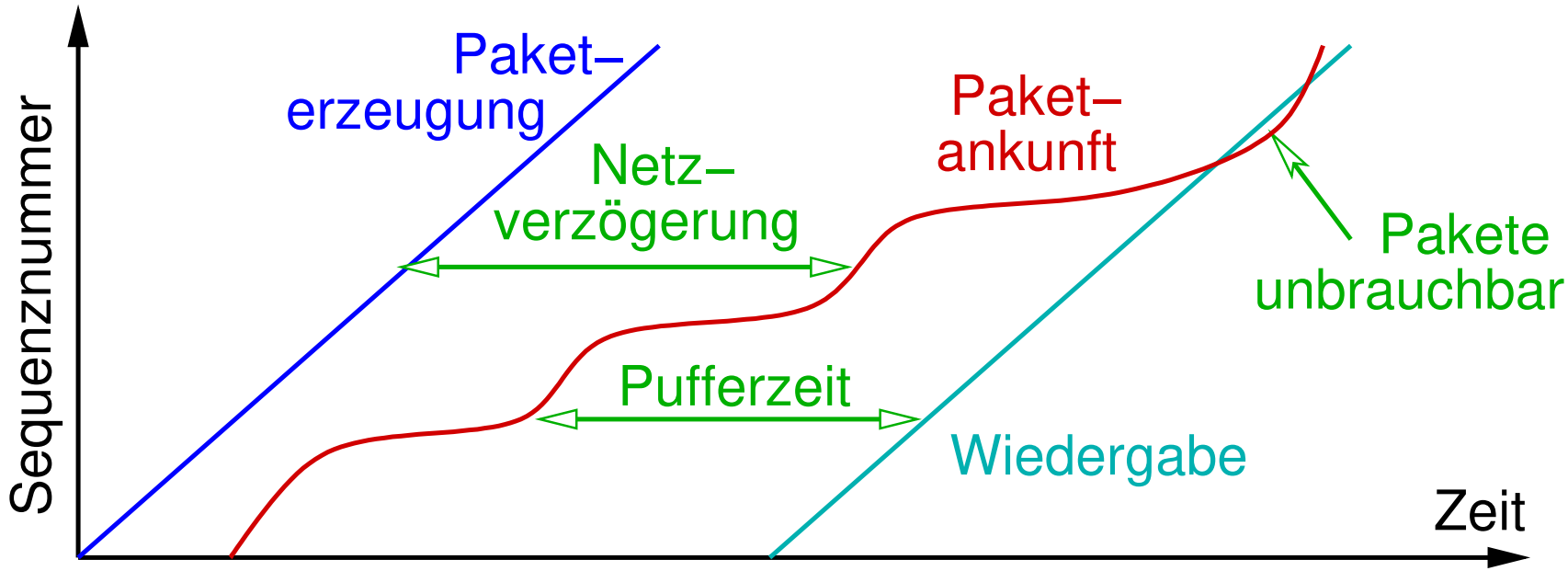
Beispiel: Echtzeit-Audioübertragung



- Sender tastet alle 125 μs ab, sendet regelmäßig Pakete
- Empfänger muß Signal mit der selben Rate wiedergeben
- Pakete treffen beim Empfänger unregelmäßig ein
  - zu spät kommende Pakete i.W. nutzlos!
  - auch verworfene und danach neu übertragene Pakete

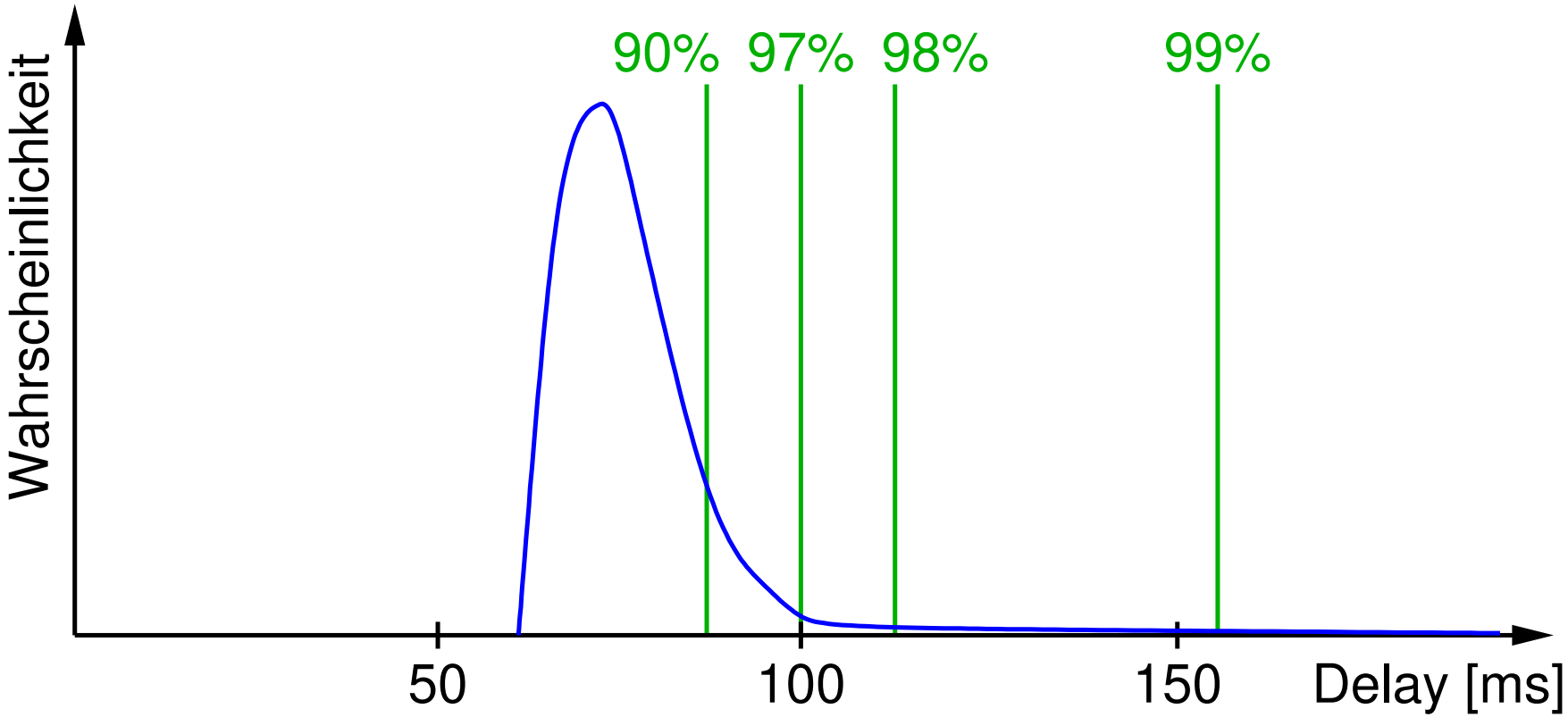
Beispiel: Echtzeit-Audioübertragung ...

- Einführung eines Wiedergabepuffers:



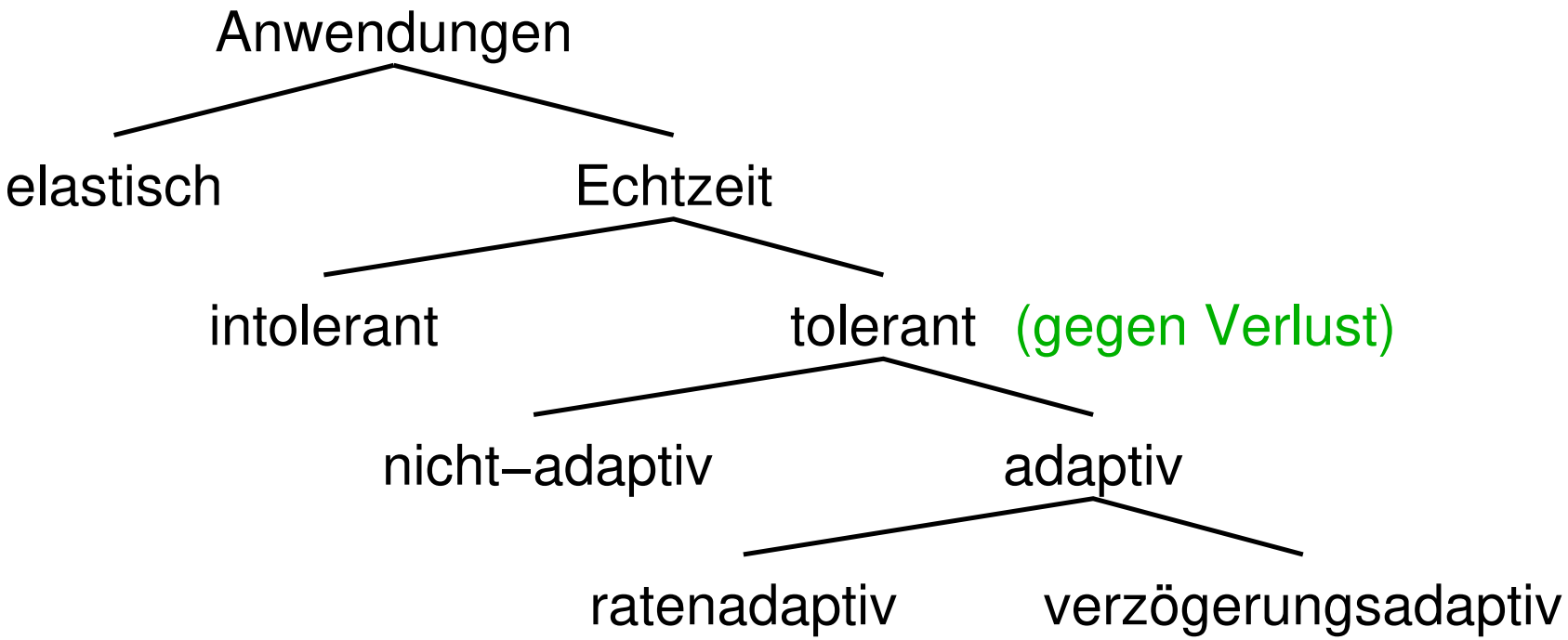
- Je größer der Puffer, desto toleranter gegen Verzögerung
- Verzögerung durch Puffer darf nicht zu groß werden
  - z.B. für Konversation max. ca. 300 ms

Typ. Verteilung der Verzögerungen einer Internet-Verbindung



- 97% aller Pakete kommen nach max. 100 ms an
- Einige Pakete brauchen aber auch mehr als 200 ms

Taxonomie von Anwendungen



- Ratenadaptiv: z.B. Anpassung der Auflösung bei Video
- Verzögerungsadaptiv: z.B. Anpassung Puffergröße bei Audio

Methoden zur Unterstützung von QoS

- Feingranulare Ansätze
  - stellen Dienstgüte für einzelne Datenflüsse bereit
  - im Internet: *Integrated Services*

- Grobgranulare Ansätze
  - stellen Dienstgüte nur für aggregierten Verkehr bereit
  - im Internet: *Differentiated Services*

Methoden zur Unterstützung von QoS

- Feingranulare Ansätze
  - stellen Dienstgüte für einzelne Datenflüsse bereit
  - im Internet: *Integrated Services*
  - benötigt viel „Intelligenz“ in den Routern
- Grobgranulare Ansätze
  - stellen Dienstgüte nur für aggregierten Verkehr bereit
  - im Internet: *Differentiated Services*
  - einfacher in den Routern, skalierbarer
  - keine wirklichen Ende-zu-Ende Garantien

6.3.1 *Integrated Services*

IntServ (IETF RFC 2205-2215)

- Motivation: Unterstützung von Multimedia-Streaming
- Basis: zusätzliche Dienstklassen
  - *Guaranteed Service*
    - für intolerante Anwendungen
    - Netz garantiert maximale Verzögerung
  - *Controlled Load*
    - für tolerante, adaptive Anwendungen
    - emuliert wenig belastetes Netz
- *Resource Reservation Protocol* (RSVP) erlaubt Reservierungen

Mechanismen zur Erbringung der Dienste

- *FlowSpecs*
  - Beschreibung des angeforderten Dienstes
  - Beschreibung der Datenflüsse
- Zugangskontrolle
  - kann der Dienst bereitgestellt werden?
- Ressourcen-Reservierung
  - Protokoll zwischen Netzwerk-Komponenten
- Paket-Scheduling
  - um Dienstgüte-Anforderungen in Routern zu erfüllen

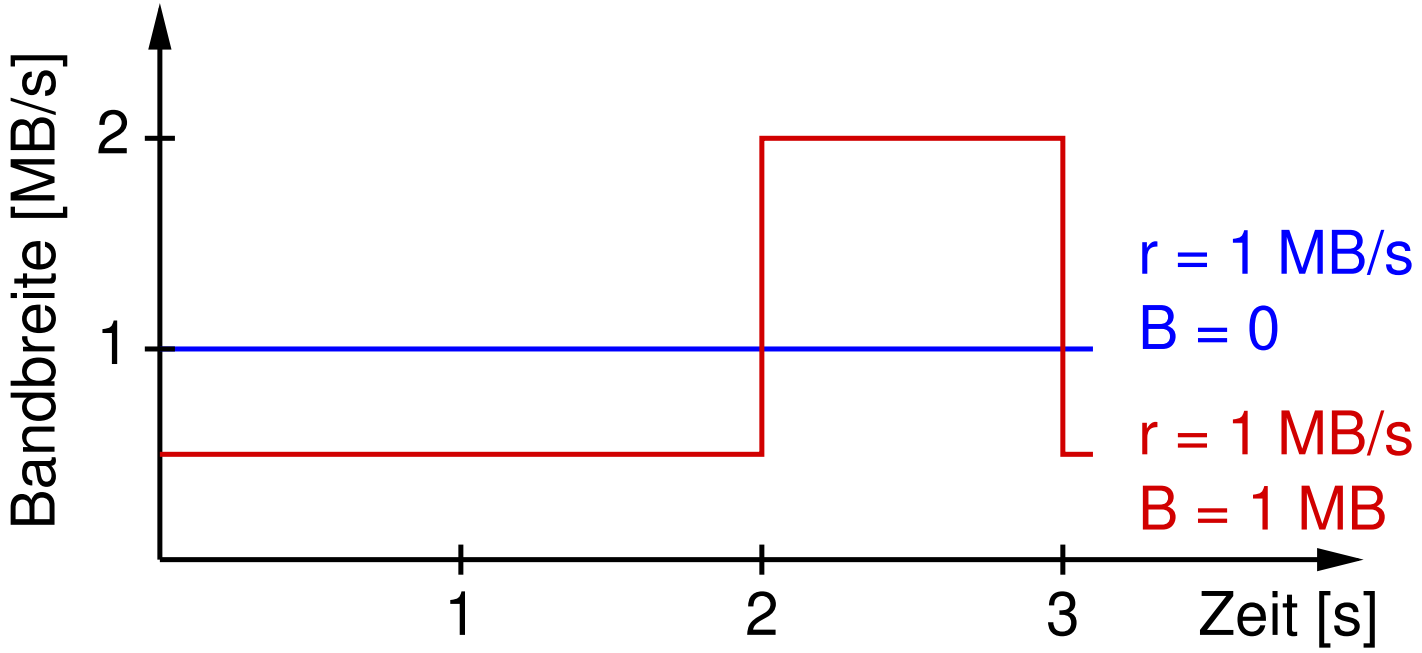
FlowSpec

- *RSpec* (*Request*): Beschreibung des angeforderten Dienstes
  - *Controlled Load Service* (CLS)
  - *Guaranteed Service* (GS) + Spezifikation der maximalen Verzögerung
- *TSpec* (*Traffic*): Beschreibung des Datenflusses
  - maximale mittlere Datenrate
  - maximale *Burst*-Größe (Varianz der Datenrate)
  - legt die zu reservierenden Resourcen in den Routern fest:
    - Bandbreite auf dem Link zum *Next Hop*
    - Pufferplatz

Einschub: *Token Bucket Filter*

- Wandelt Datenstrom so um, daß
  - die mittlere Datenrate höchstens r ist
  - *Bursts* mit höherer Rate auf eine Größe von B Bytes beschränkt sind
    - d.h. Router mit Datenrate r am Ausgang braucht höchstens B Bytes an Puffer
- Idee des Filters:
  - Sender benötigt für jedes gesendete Byte ein Token
  - Sender startet mit 0 Token,
    bekommt r Token pro Sekunde,
    kann maximal B Token akkumulieren

Einschub: *Token Bucket Filter ...*



- Beide Datenströme haben dieselbe mittlere Datenrate, aber unterschiedliche Bucket-Tiefe

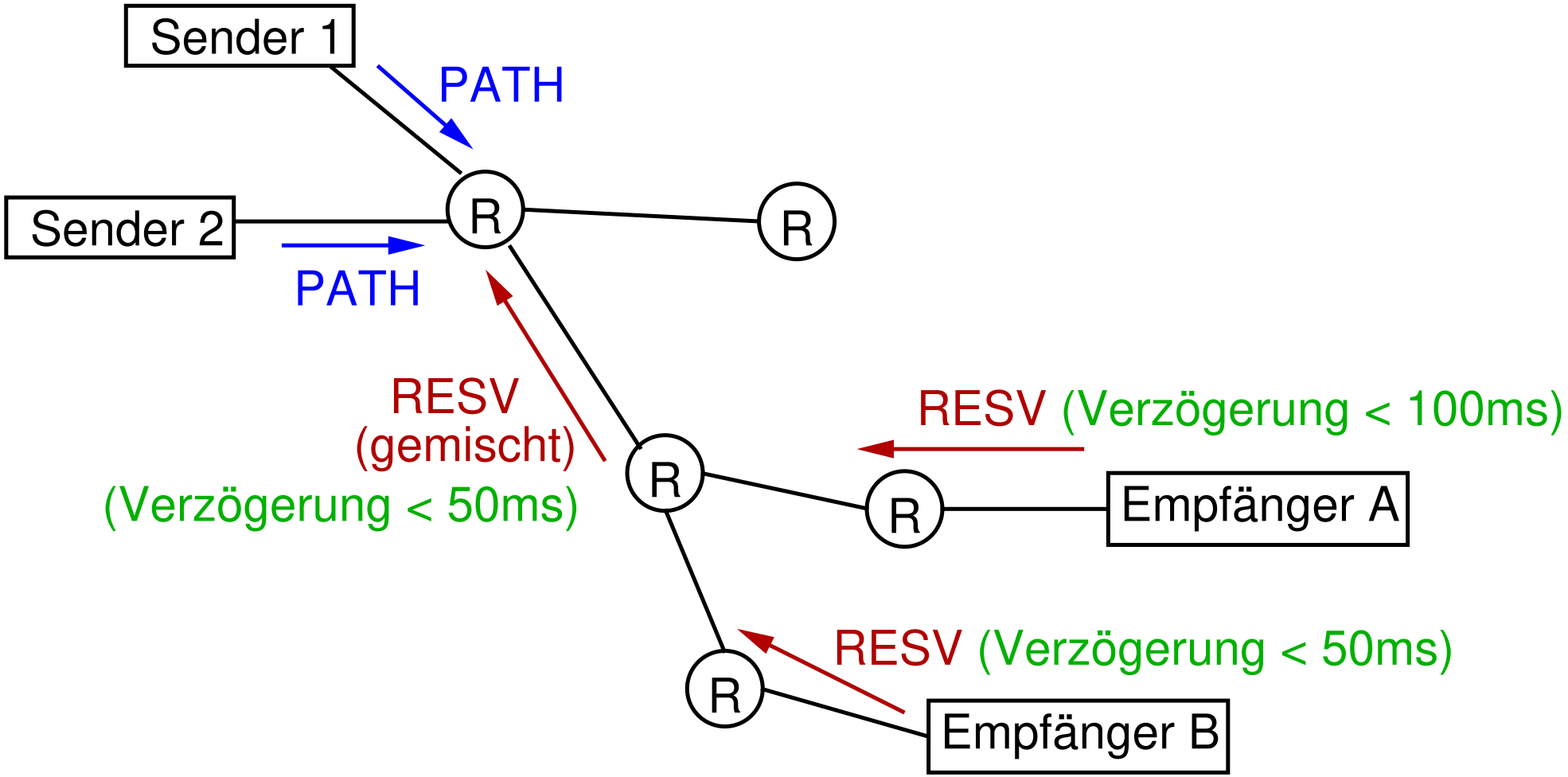
Zugangskontrolle

- Wenn neuer Datenfluß bestimmte Dienstgüte anfordert:
  - betrachte *RSpec* und *TSpec*
  - entscheide, ob Dienst bereitgestellt werden kann
    - bereits zugestandene Dienste dürfen nicht beeinträchtigt werden
- Entscheidung ggf. auf Basis von Benutzerklassen
- Nicht verwechseln mit *Policing*:
  - Prüfung, ob Datenstrom die *TSpec* einhält
  - regelverletzende Pakete werden verworfen bzw. als Wegwerfkandidaten markiert

RSVP Reservierungsprotokoll

- Deutlich verschieden von Signalisierungsprotokollen in verbindungsorientierten Netzen
  - Ziel: Robustheit verbindungsloser Netze erhalten
  - daher: *Soft State* mit periodischem Auffrischen ($\sim$ alle 30 s)
- RSVP unterstützt auch Multicast-Flüsse
  - Reservierung erfolgt durch Empfänger
- Sender schicht PATH-Nachricht mit *TSpec*
- Empfänger schickt RESV-Nachricht mit seinen Anforderungen (*RSpec*) entlang derselben Route zurück
  - Router teilen erforderliche Ressourcen zu
  - ggf. Zusammenfassung von Anforderungen möglich

RSVP Reservierungsprotokoll ...



Paketklassifizierung und Scheduling

- Klassifizierung: Zuordnung von Paketen zu Reservierungen
  - IPv4: über Quell-/Zieladresse, Quell-/Zielport, Protokoll
  - IPv6: über **FlowLabel**
- Einordnung in Klasse bestimmt Bearbeitung des Pakets in Router-Warteschlange (Scheduling):
  - bei GS: eine Warteschlange pro Fluß, *Weighted Fair Queueing* (WFQ)
    - *Fair Queueing* mit Gewichtung: jede Warteschlange erhält Bandbreitenanteil entsprechend ihrer Gewichtung
    - Ende-zu-Ende-Verzögerung leicht berechenbar
  - bei CLS: eine gemeinsame Warteschlange mit WFQ

Zur Skalierbarkeit

- Router müssen Information für einzelne Datenströme speichern
- Potentiell sehr viel Information:
  - z.B. Audioströme (64 Kb/s) über OC-48-Leitung (2.5 Gb/s): bis zu 39000 Flüsse!
- Für jeden Fluß muß Klassifizierung, *Policing* und Warteschlangenverwaltung durchgeführt werden
- Zusätzlich: Mechanismen, um große Reservierungen für lange Perioden zu vermeiden
- Wegen Skalierbarkeitsproblemen:
  - bisher keine breite Anwendung von *IntServ*

6.3.2 Differentiated Services

DiffServ (IETF RFC 2474/2475)

- Ziel: bessere Skalierbarkeit
- Einteilung des Verkehrs in wenige Verkehrsklassen
  - Zuteilung von Ressourcen an Verkehrsklassen statt einzelne Ströme
- Arbeitsaufteilung:
  - Router an der Peripherie (z.B. Eingangsrouter eines ISP)
    - Zuweisung einer Verkehrsklasse an Pakete
      - z.B. je nach Kunde eines ISP
    - Policing: Einhaltung eines Verkehrsprofils
  - innere Router:
    - Weiterleitung der Pakete entsprechend Verkehrsklasse

Kennzeichnung von Verkehrsklassen

- **TOS**-Feld in IPv4 bzw. **TrafficClass**-Feld in IPv6
  - wird vom Eingangsrouter gesetzt
  - z.B. aufgrund von Quell-/Ziel-Adresse, Quell-/Zielport
  - einfachster Fall: betrachte nur Quell-IP-Adresse
    - d.h. z.B. normaler und „Premium“ Internetzugang
- Feld legt in Routern das Weiterleitungsverhalten (*Per Hop Behavior*, PHB) fest
  - derzeit spezifiziert:
    - *Expedited Forwarding*, EF (RFC 2598)
    - *Assured Forwarding*, AF (RFC 2597)

Expedited Forwarding

- Ziel: Pakete mit EF-Kennzeichung werden mit
  - minimaler Verzögerung
  - geringstmöglichem Paketverlust

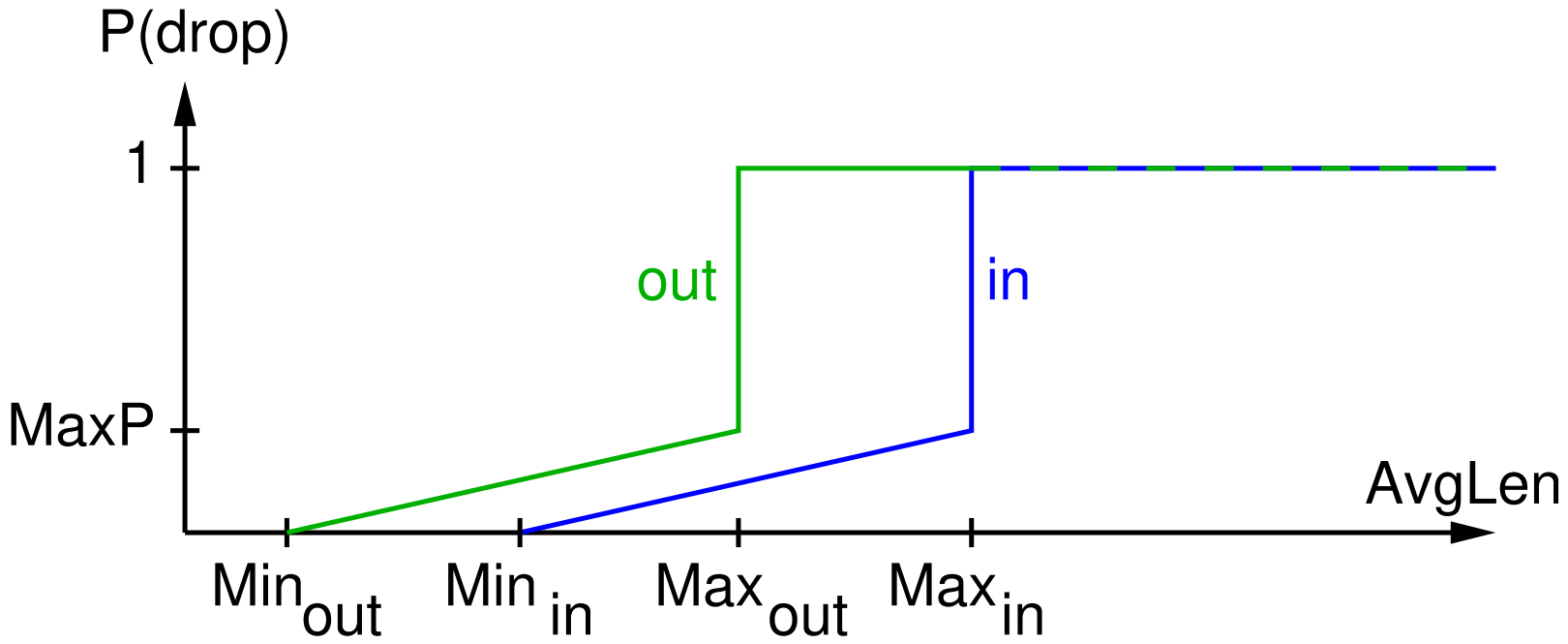
  weitergeleitet, z.B. für *Voice over IP* (VoIP)
- Router kann dies nur garantieren, wenn Ankunftsrate von EF-Paketen limitiert ist
  - Sicherstellung durch Router am Rand der Domäne
- Mögliche Implementierungen:
  - EF-Pakete erhalten strikte Priorität
  - WFQ mit entsprechend hoher Gewichtung für EF-Pakete

Assured Forwarding

- Ziel: gute Bandbreite bei geringer Verlustrate (z.B. für Video)
- 4 Klassen, jeweils mit Mindestanteil an Bandbreite und Pufferplatz
  - realisierbar durch mehrere Warteschlangen und WFQ mit entsprechenden Gewichtungen
  - Beispiel mit zwei Klassen (*Premium*, *Best Effort*):
    - Gewichtung 1 für *Premium*, 4 für *Best Effort* führt zu 20% reservierter Bandbreite für *Premium*
    - Verzögerung für *Premium* aber nicht notwendigerweise geringer als bei *Best Effort*
- Innerhalb jeder Klasse drei Drop-Prioritäten
  - für die Überlastvermeidung innerhalb der Klassen
  - realisierbar durch *Weighted RED*

Weighted RED

- Beispiel: RED mit zwei verschiedenen Drop-Wahrscheinlichkeiten



- RIO: *RED with In and Out*
- *In*: innerhalb der Verkehrsprofils, *Out*: außerhalb

6.3.3 Diskussion

- QoS im Internet motiviert durch Multimedia-Anwendungen
- *IntServ*
  - *Guaranteed Service* kann max. Verzögerung garantieren
  - getrennte Behandlung der einzelnen Flüsse
  - explizite, dynamische Ressourcenreservierung durch RSVP
  - Probleme: größere Änderungen in Routern, Skalierbarkeit
- *DiffServ*
  - Skalierbarer Ansatz
  - Klassifikation von Paketen am Rand des Netzwerks, Paketklasse bestimmt Behandlung in Routern
  - Frage: ausreichend für Anwendungen?

Überlastkontrolle

- Überlast: häufiger Paketverlust im Netz, da Pakete nicht weitergeleitet und auch nicht mehr gepuffert werden können
- Überlastkontrolle in TCP
  - *Additive Increase / Multiplicative Decrease*
    - verkleinere Fenster bei Überlast drastisch
  - Erweiterungen: *Slow Start*, *Fast Retransmit / Fast Recovery*

Überlastvermeidung

- Senderate reduzieren, bevor Überlast auftritt
- Erkennung drohender Überlast:
  - Router: mittlere Warteschlangenlänge
  - Host: Latenz bzw. Durchsatz

Überlastvermeidung ...

- DECbit
  - falls mittlere Warteschlangenlänge über Grenzwert:
    - Router setzt Überlastbit im Paketheader
  - Empfänger kopiert Überlastbit in ACK
  - Sender reduziert Überlastfenster, wenn die Mehrzahl der ACK's Überlastbit gesetzt hat
- RED (*Random Early Detection*)
  - für Zusammenarbeit mit TCP konzipiert
  - falls mittlere Warteschlangenlänge über Grenzwert:
    - Router verwirft zufällig einige Pakete
  - TCP reduziert bei Paketverlust das Sendefenster

Überlastvermeidung ...

- Quellenbasierte Überlastvermeidung
  - TCP Vegas: Vergleich von tatsächlichem Durchsatz und erwartetem (maximalem) Durchsatz
    - Differenz ist Maß für beanspruchten Pufferplatz
  - Sendefenster so regeln, daß immer (nur) eine kleine Zahl von Puffern belegt wird

QoS (*Quality of Service*)

- Unterstützung verschiedener Dienstklassen mit Garantien bzgl. Bandbreite, Latenz, Jitter, etc.
- Für Realzeit- bzw. Multimedia-Anwendungen

QoS (*Quality of Service*) ...

- *Integrated Services*: feingranularer Ansatz
  - fluß-spezifische Dienstgüte-Garantien
  - Mechanismen: *Flow Specs*, Zugangskontrolle, Reservierung, Paket-Scheduling
    - Spezifikation des Flusses über *Token Bucket*
  - zwei Dienste: *Guaranteed Service*, *Controlled Load*
  - Problem: Skalierbarkeit in den Routern
- *Differentiated Services*: grobgranularer Ansatz
  - Einteilung der Pakete in Verkehrsklassen
  - Zuteilung von Ressourcen an Verkehrsklassen

Rechnernetze II

SoSe 2025

7 Anwendungsprotokolle

Inhalt

- Netzwerkmanagement
- Multimedia-Anwendungen

- Peterson, Kap. 9.2.3, 9.3
- Tanenbaum, Kap. 6.4.3, 7.4.5
- Kurose/Ross, Kap. 8.1-8.4
- CCNA, Kap. 8

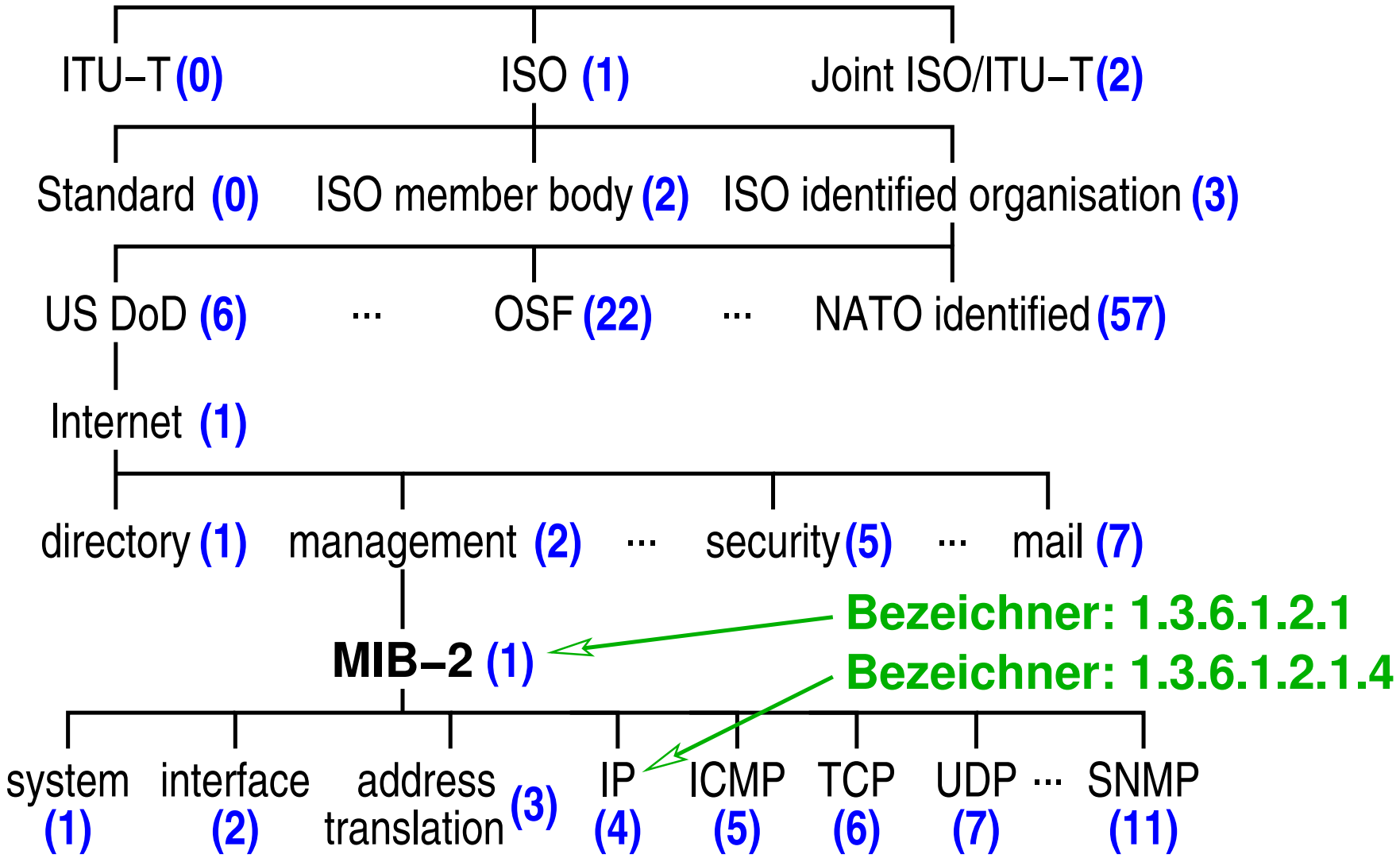
Aufgaben des Netzwerkmanagements

- Performance-Management
  - z.B. Überwachung des Verkehrs zur Ressourcenplanung
- Fehlermanagement
  - z.B. Ausfall einer Netzwerkkarte, Prüfsummenfehler, *Route-Flapping*, IP *Reassembly*-Fehler
- Konfigurationsmanagement
- Accounting-Management
  - z.B. Nutzungsquoten, Zugriffsrechte auf Ressourcen
- Sicherheits-Management
  - Einbruchserkennung, z.B. DoS, Port-Scans

Simple Network Management Protocol (SNMP) (RFC 1901 ff.)

- Überwachung des Netzes über das Netz
- Einfaches Anfrage-Antwort-Protokoll
  - Anfragen i.w.: GET und SET Operationen auf Objekten (Variablen)
- Setzt bevorzugt auf UDP auf
- Sicherheitsmechanismen erst ab Version 3 (SNMPv3)
- Sammlung aller Objekte: *Management Information Base* MIB
  - hierarchisch organisiert (Baumstruktur)
  - beschrieben in der Sprache SMI (Basis: ASN.1)
    - einfache Datentypen, Sequenzen (Tabellen), Strukturen
    - Objekte durch „dezimale Punktnotation“ benannt

ASN.1 Objektidentifizierungsbaum



MIB-Einträge (Beispiele)

- System: Systemname (HW, BS, ...), Hersteller, *Uptime*, Standort, Kontaktperson, verfügbare Dienste
- Interface: Hardware-Adresse, gesendete/empfangene Pakete
- Adreßübersetzung: ARP, insbesondere Übersetzungstabelle
- IP: Routing-Tabelle, Anzahl Datagramme, Statistiken zu Reassemblierung, verworfene Pakete
- TCP: Zahl passiver/aktiver *Open*-Operationen, Anzahl *Timeouts*, *Default-Timeout*, Liste offener TCP-Verbindungen
- UDP: Zahl gesendeter / empfangener / unzustellbarer Datagramme, Liste geöffneter UDP-Ports

Im Folgenden:

- RTP (*Realtime Transport Protocol*) und RTCP (*Realtime Transport Control Protocol*)
  - Übertragung von z.B. Audio- und Video-Strömen
  - RTCP: Steuerprotokoll zu RTP
- SIP (*Session Initiation Protocol*)
  - Aufbau von Sitzungen (Telefonie, Konferenz)

- Anmerkung: ITU-Empfehlung H.323 für Internet-Telefonie
  - umfaßt RTP, RTCP und Protokolle f. Registrierung (H.225), Anrufsignalisierung (Q.931) und Anrufsteuerung (H.245)
  - SIP ist IETF Alternative zu H.323-Steuerprotokollen

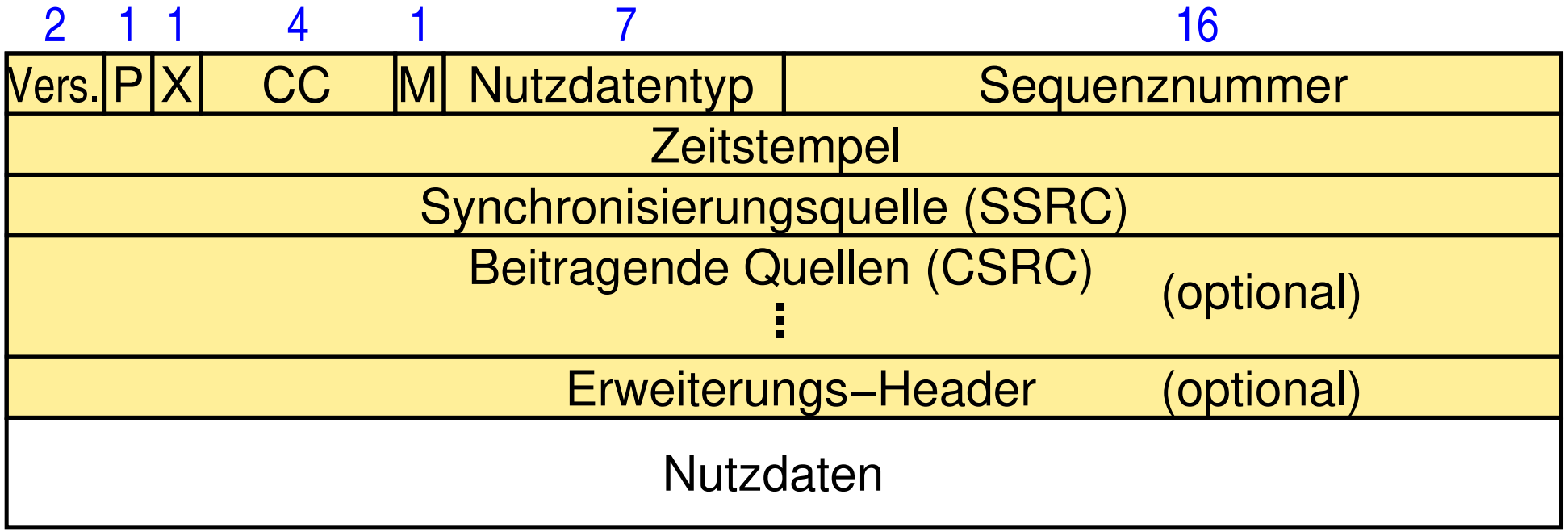
RTP *Realtime Transport Protocol* (RFC 3550)

- Allgemeines Transportprotokoll für Multimedia-Ströme
  - unabhängig vom Typ der Daten (Audio, Video, ...) und der Codiermethode (z.B. MPEG)
- Setzt i.d.R. auf UDP auf
  - typisch: Bibliothek oberhalb der Socket-Schnittstelle
- Unterstützt das Multiplexen mehrerer Ströme in einen Strom von UDP-Paketen
  - um Bandbreitenbedarf zu verringern
- Zwei Protokolle: RTP (Daten) und RTCP (Steuerung)
  - RTP: gerade Port-Adresse
  - RTCP: unmittelbar folgende Port-Adresse

RTP/RTCP: Anforderungen

- Pakete mit Zeitstempel (für korrekte Wiedergabe)
- Synchronisation verschiedener Ströme (z.B. Audio und Video)
- Empfänger soll Paketverlust erkennen können
  - keine Neuübertragung; Behandlung anwendungsspezifisch
- *Feedback* zum Sender für Überlastvermeidung
  - Statistiken über Paketverlust; Behandlung anwendungs-spezifisch, z.B. agressivere Kompression
- Kennzeichnung von Pakten (z.B. Frame-Grenzen bei Video)
- Geringer Paket-Overhead durch Header
  - i.d.R. kurze Pakete wegen Latenzzeit

Aufbau eines RTP-Pakets



Vers.: Versionsnummer (2)
P: Füllbit: Nutzdaten aufgefüllt, letztes Byte enthält Anzahl der Füllbytes
X: Erweiterungs–Header vorhanden
CC: Anzahl beitragender Quellen
M: Markierungsbit

Details zu RTP

- RTP definiert für jede Anwendungsklasse ein Profil und ein oder mehrere Formate für die Nutzdaten
  - Profil: z.B. Audio-, Video-Strom; Format: z.B. MPEG, H.261
  - legt fest, wie Marker / Zeitstempel zu interpretieren sind
- SSRC ist zufällig gewählte ID der Quelle (z.B. Kamera), die den RTP-Strom identifiziert
- CSRC nur benutzt, wenn mehrere Stöme durch Multiplex zusammengefaßt werden
  - SSRC identifiziert dann den Multiplexer
- Zeitstempel ist relative Zeit
  - Auflösung durch Profil definiert, je nach Abtastintervall

Steuerprotokoll RTCP

- Aufgaben:
  - Rückmeldung über Leistung der Anwendung / des Netzes
  - Synchronisation mehrerer RTP-Ströme eines Senders
  - anwenderfreundliche Senderidentifikation
- Pakettypen:
  - Empfängerberichte: Statistiken zu Zahl der Pakete, Paketverlust, geschätzter Jitter, ...
  - Senderberichte: Uhrzeit und zugehöriger RTP-Zeitstempel, Zahl bisher gesendete Pakete / Bytes
  - Quellenbeschreibungen: SSRC und kanonischer Name (user@host)

7.2.2 Sitzungs- und Anrufsteuerung

- Aufgaben u.a.:
  - Bereitstellung von Informationen zu einer Sitzung bzw. Verbindung (SDP: *Session Description Protocol*)
    - Name und Zweck der Sitzung
    - Anfangs- und Endezeit
    - Medientypen (Audio / Video)
    - Netzwerk-bezogene Informationen
      - Multicast Adresse, Transportprotokoll (z.B. RTP + Profil, UDP), Ports, Kodierung
  - Verbindungsaufname bei Internet-Telefonie (SIP: *Session Initiation Protocol*)

- (Hier nur Protokolle der IETF betrachtet)

SDP: *Session Description Protocol* (RFC 4566)

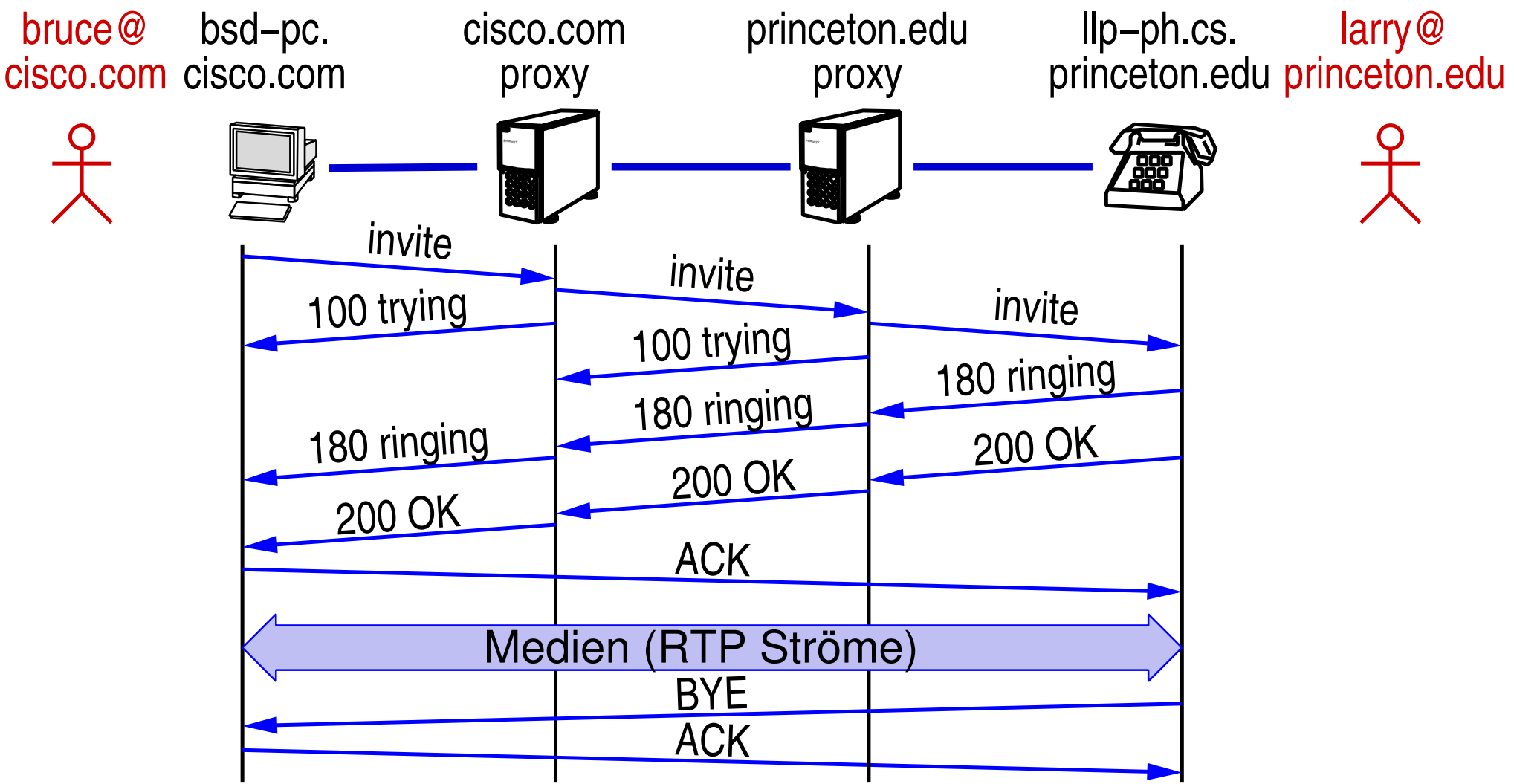
- Legt i.w. fest, wie Sitzungsinformationen codiert werden
  - ASCII-Codierung mit *Typ=Wert* Format

```
v=0                                                    SDP–Version
o=larry 2890844526 2890842807 IN IP4 10.0.1.5          Ursprung
s=Networking 101                                       Sitzungsname
i=A class on computer networking                       Sitzungsbeschreibung
u=http://www.cs.princeton.edu                          URI der Sitzung
e=larry@princeton.edu                                  Email–Adresse Kontaktperson
c=IN IP4 224.2.17.12/127                               IP–Multicast–Adresse der Sitzung
t=2873397496 2873404696                                Start– und Endzeit
m=audio 49170 RTP/AVP 0                                Medien (Typ, Port, Transportprotokoll, Format)
m=video 51372 RTP/AVP 31
m=application 32416 udp wp
```

SIP (*Session Initiation Protocol)* (RFC 3261)

- Anfrage/Antwort-Protokoll, ähnlich HTTP und SMTP
  - Verwendung von MIME für Nutzdaten
- Funktionen:
  - Bestimmung des Benutzerstandorts
    - Benutzer kann mehrere verschiedene Rechner nutzen
    - daher: Einführung eines Proxy-Servers, bei dem sich Benutzer mit aktuellem Standort (= Rechner) registriert
  - Feststellung, ob Benutzer an Sitzung teilnehmen kann/will
  - Aushandlung der Medien und Kodierungen
  - Einrichten von Sitzungsparametern (z.B. Port-Nummern)
  - Sitzungsmanagement, z.B. Rufweiterleitung, Änderung von Sitzungsparametern

Nachrichtenfluß einer SIP-Sitzung



Netzwerkmanagement

- Performance, Fehler, Konfiguration, Accounting, Sicherheit
- SNMP: Zugriff auf Objekte (Variablen) über das Netz
- MIB: Sammlung aller Objekte, hierarchisch strukturiert
  - System, Interface, ARP, IP, ICMP, TCP, UDP, ...

Multimedia-Protokolle

- H.323 Empfehlung: RTP/RTCP und Steuerprotokolle
- RTP: allgemeines Transportprotokoll, über UDP
  - Multiplexing von Datenströmen
  - Zeitstempel
  - Sequenznummern
  - Markierung von Paketen
- RTCP: Steuerprotokoll zu RTP
  - Sender- und Empfängerberichte
    - Paketverlust, Jitter
    - Zuordnung Zeitstempel $\leftrightarrow$ Reale Zeit
  - Quellenbeschreibungen

Rechnernetze II

SoSe 2025

8 Netzwerkprogrammierung

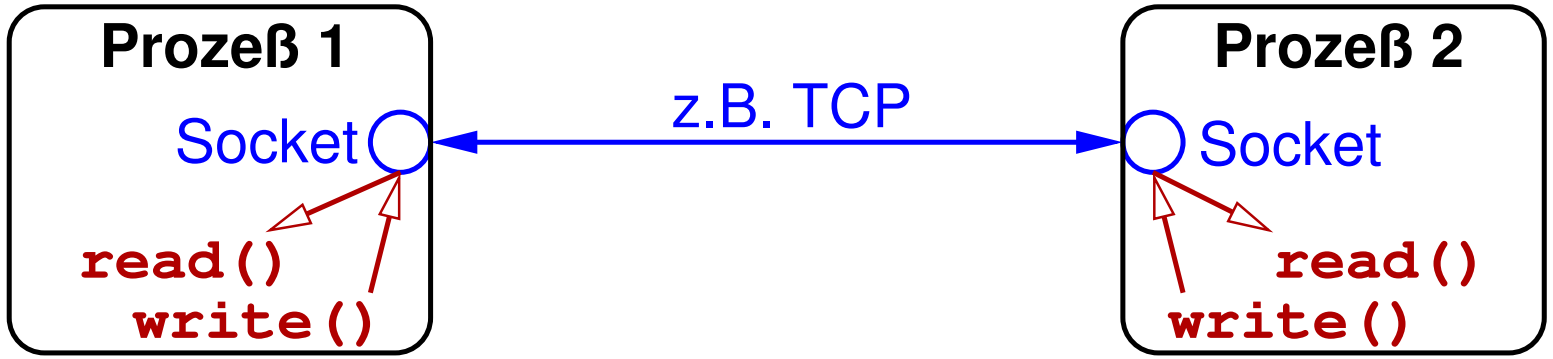
Inhalt

- Sockets
- Datagramm-Kommunikation (UDP)
- Strom-Kommunikation (TCP)
- Design von Server-Programmen

- W.R. Stevens: Programmieren von UNIX-Netzen, Hanser/Prentice Hall, 1992. Kap. 6 und 18.3
- T. Langner: Verteilte Anwendungen mit Java, Markt + Technik, 2002. Kap. 3

Socket-Schnittstelle

- API (*Application Programming Interface*) für die Interprozeß-Kommunikation
  - Prozesse auf demselben oder verschiedenen Rechnern
  - unabhängig vom Netzwerkprotokoll
  - eingeführt mit BSD 4.2 Unix (1981, *Berkeley Sockets*)
  - auch in Windows-Betriebssystemen verfügbar
- **Socket**: Abstraktion für Kommunikationsendpunkt



Erzeugung eines Sockets

- Systemaufruf `socket`
- `int fd = socket(int domain, int type, int protocol);`
  - `domain`: Kommunikations-Bereich
    - lokal, Internet, ...
  - `type`: Socket-Typ
    - Datenstrom, Datagramme, ...
  - `protocol`: zu verwendendes Protokoll
    - nötig, wenn Socket-Typ mehrere Protokolle unterstützt
    - in der Regel mit 0 besetzt
  - `fd`: Dateideskriptor des Sockets
    - bzw. -1 bei Fehler

Kommunikations-Bereiche

- Legen fest:
  - Kommunikation lokal oder über Netzwerk
  - verwendbare Kommunikationsprotokolle
    - genauere Auswahl über `type` und `protocol` Parameter
  - Aufbau von Namen bzw. Adressen
- Unterstützte Kommunikations-Bereiche:
  - `PF_UNIX`: *UNIX Domain*, rechnerlokale Kommunikation
  - `PF_INET`: *Internet Domain*, TCP, UDP, IP
  - etliche andere (IPv6, Novell, X.25, Appletalk, ...)

Die wichtigsten Socket-Typen

- `SOCK_STREAM`: *Stream Socket*
  - verbindungsorientierte, strombasierte Kommunikation
  - in *Internet Domain*: TCP

```
int fd = socket(PF_INET, SOCK_STREAM, 0);
```

- `SOCK_DGRAM`: *Datagram Socket*
  - verbindungslose Datagramm-Kommunikation
  - in *Internet Domain*: UDP

```
int fd = socket(PF_INET, SOCK_DGRAM, 0);
```

- `SOCK_RAW`: *Raw Socket*
  - verwendet Basisprotokoll des Kommunikations-Bereichs
  - in *Internet Domain*: IP

Binden an eine Adresse

- Systemaufruf `bind`
- `int rv = bind(int sockfd, struct sockaddr *addr, int addrlen);`
  - `sockfd`: Dateideskriptor des Sockets
  - `addr`: lokale Adresse
    - z.B. IP-Adresse und Port
  - `addrlen`: Länge der Adreß-Datenstruktur
  - `rv`: Rückgabewert, 0 = OK, -1 = Fehler

- **Hinweis:** Ab sofort werden nur noch *Internet Domain* Sockets betrachtet

Adreß-Datenstruktur für *Internet Domain* Sockets

- ```
  struct in_addr { unsigned long s_addr; };
  struct sockaddr_in { short sin_family;
                       u_short sin_port;
                       struct in_addr sin_addr; }
  ```

- Beispiel: binde Socket an Port 80

```
struct sockaddr_in addr;
memset(&addr, 0, sizeof(addr)); // alles auf 0
addr.sin_family = AF_INET;
addr.sin_port = htons(80);          // Port 80
addr.sin_addr.s_addr = htonl(INADDR_ANY);
if (bind(sockfd, (struct sockaddr *) &addr,
         sizeof(addr)) < 0) ...
```

- Hinweis: `sockaddr_in` ist quasi Unterklasse von `sockaddr`
```

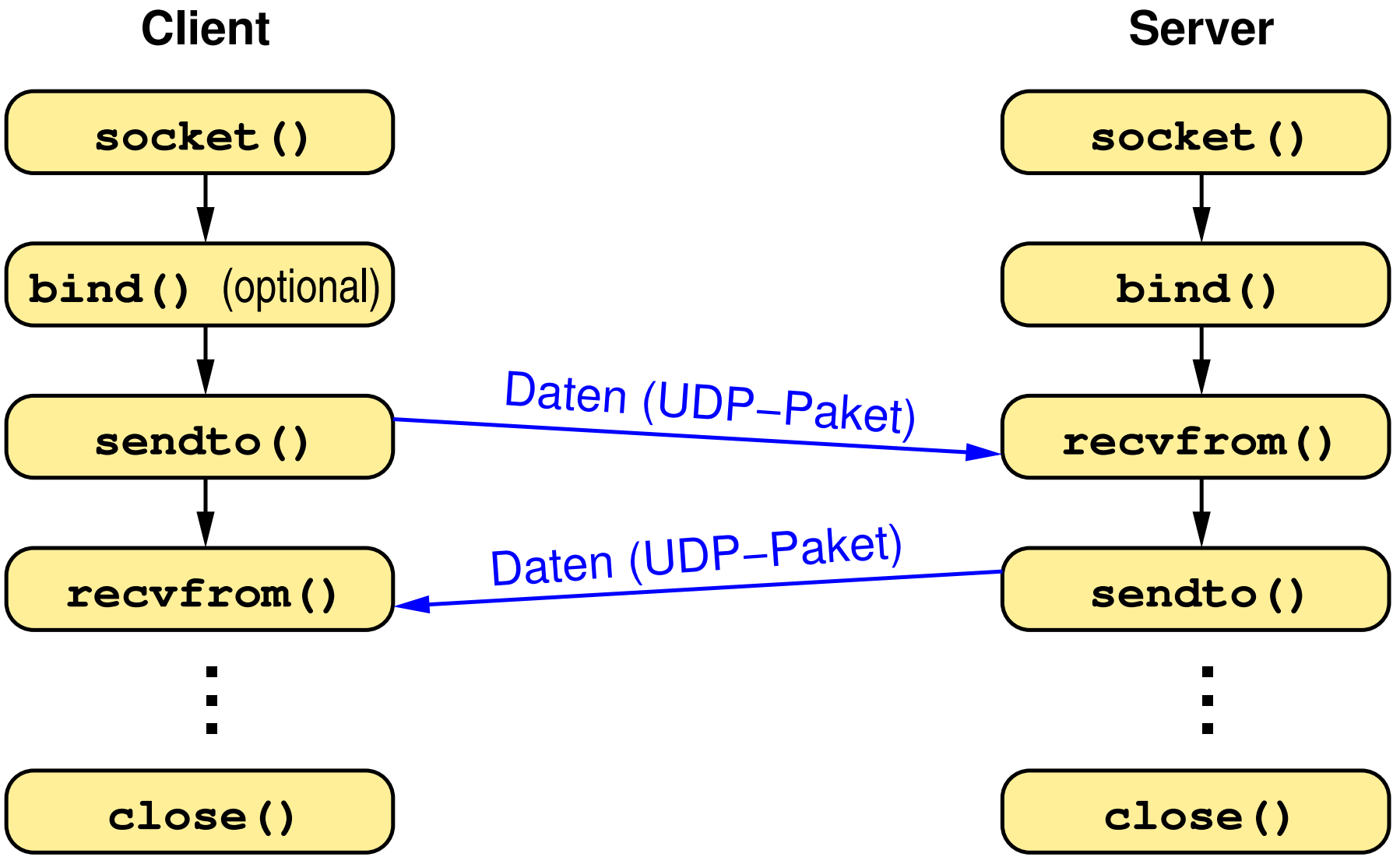
Binden an eine Adresse, Anmerkungen

- `bind` ist nur für Server-Sockets erforderlich
  - Client-Sockets wird automatisch ein Port zugewiesen
- `INADDR_ANY` bezeichnet beliebiges lokales Netzwerk-Interface
- Verwendung einer vorgegebenen IP-Adresse, z.B.:
  - `addr.sin_addr.s_addr = inet_addr("127.0.0.1");`
- DNS-*Lookup* ist bei Bedarf über die Bibliotheksfunktion `gethostbyname` möglich
- Die Funktionen `htons` bzw. `htonl` konvertieren vom Host- in das Netzwerk-Datenformat
  - entsprechend gibt es auch `ntohs` etc.

Schließen eines Sockets

- Systemaufruf `close`
- `close(int sockfd);`
  - `sockfd`: Dateideskriptor des Sockets
- Bei TCP-Sockets: Verbindungsabbau

8.2 Datagramm-Kommunikation (UDP)



Datagramm-Transfer

- Senden eines Datagramms:
  - `int sendto(int sockfd, char *msg, int len, int flags, struct sockaddr *addr, int addrlen);`
    - `msg`: zu sendende Nachricht (Länge: `len`)
    - `flags`: Optionen, normalerweise 0
    - `addr`, `addrlen`: Zieladresse
- Empfangen eines Datagramms:
  - `int recvfrom(int sockfd, char *msg, int len, int flags, struct sockaddr *addr, int *addrlenptr);`
    - `msg`: Empfangspuffer (Länge: `len`)
    - `addr`, `addrlenptr`: Rückgabe der Senderadresse

Beispiel-Code

- Siehe `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/gen/rn2/code/udp-c.zip`

Datagramm-Sockets in Java

- Klasse `DatagramSocket`
  - Konstruktoren:
    - `DatagramSocket()`
    - `DatagramSocket(int port)`
      - bindet Socket an `port` auf lokalem Rechner, für Server
  - Methoden:
    - `void send(DatagramPacket p)`
      - Senden eines Datagramms
    - `void receive(DatagramPacket p)`
      - Empfang eines Datagramms

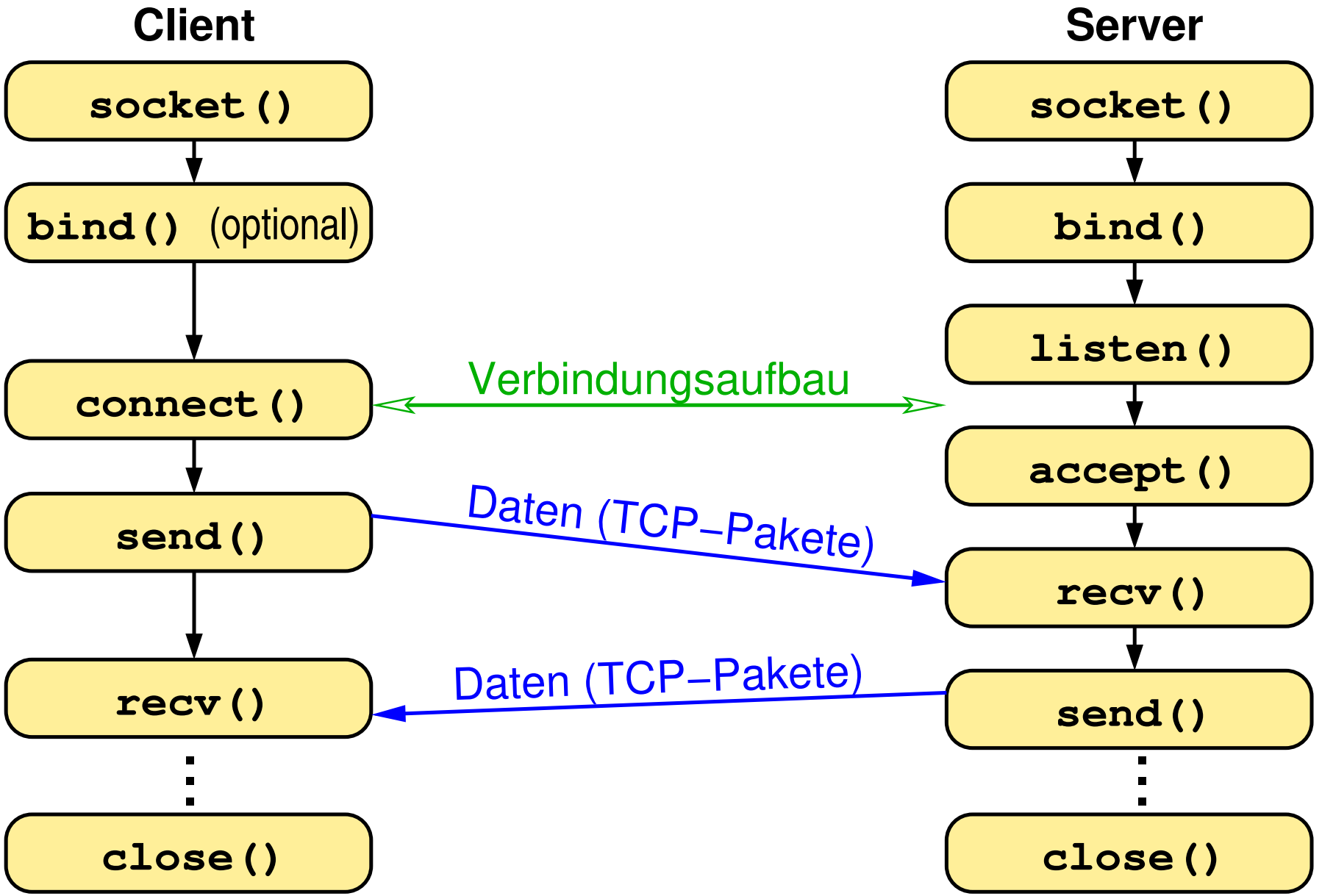
Datagramm-Sockets in Java ...

- Klasse `DatagramPacket`: Abstraktion eines UDP-Pakets
  - Konstruktoren:
    - `DatagramPacket(byte[] buf, int len, InetAddress addr, int port)`
      - für zu sendende Pakete
    - `DatagramPacket(byte[] buf, int len)`
      - für zu empfangende Pakete
  - Methoden:
    - `InetAddress getAddress()`: IP-Adresse des Pakets
    - `int getPort()`: Port-Nummer des Pakets

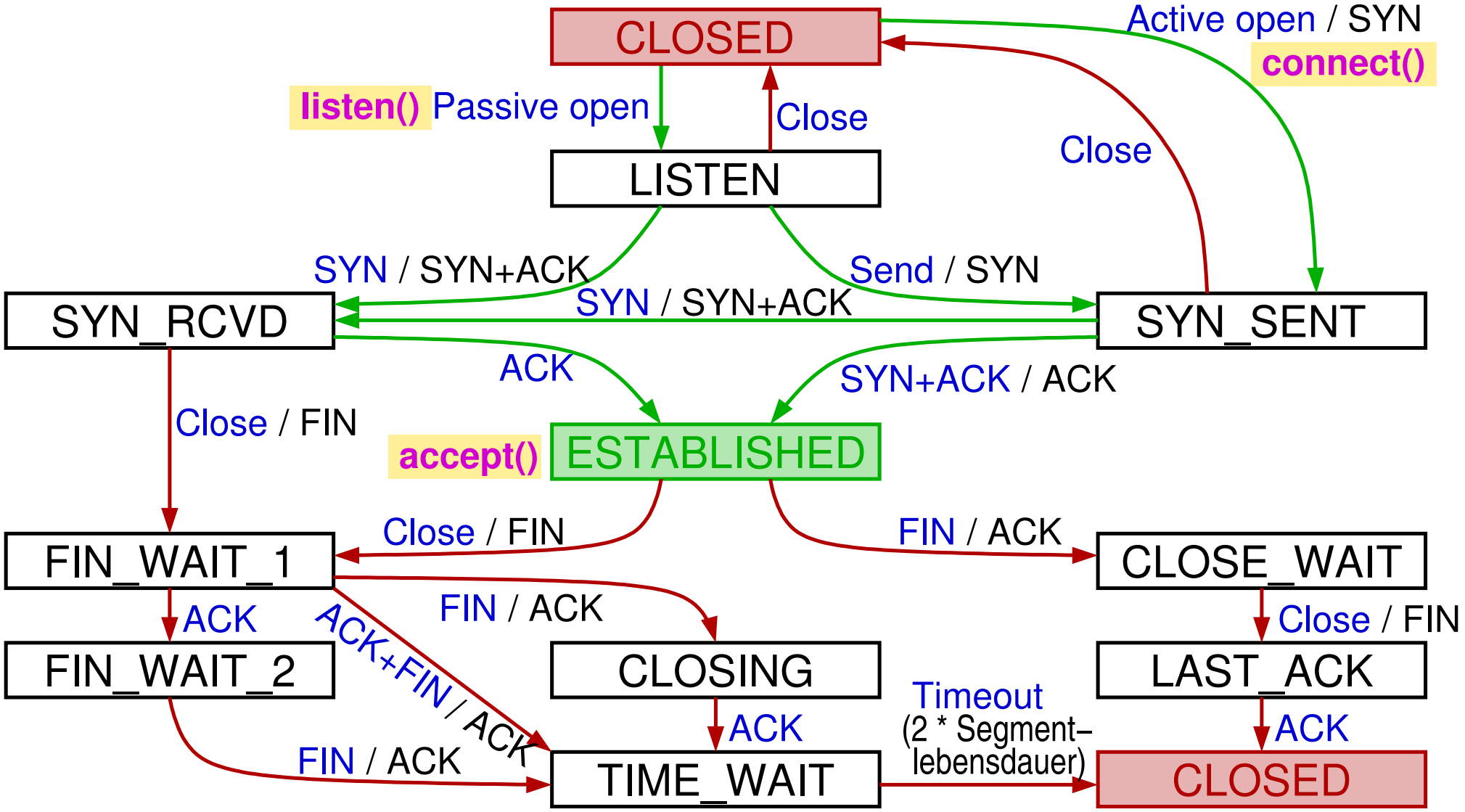
Datagramm-Sockets in Java ...

- Klasse `InetAddress`: IP-Adressen
  - statische Methoden:
    - `InetAddress getLocalHost()`
      - liefert `InetAddress`-Objekt für lokale IP-Adresse
    - `InetAddress getByName(String host)`
      - liefert `InetAddress`-Objekt für gegebenen Rechnernamen (oder IP-Adresse in String-Form)
    - `InetAddress getByAddress(byte[] addr)`
      - liefert `InetAddress`-Objekt für gegebene IP-Adresse

8.3 Strom-Kommunikation (TCP)



Erinnerung: Zustände einer TCP-Verbindung



Aktives Öffnen einer Verbindung (Client)

- Systemaufruf `connect`
- ```
  int rv = connect(int sockfd, struct sockaddr *addr,
                   int addrlen);
  ```
  - `addr`, `addrlen`: legt IP-Adresse und Port des Servers fest
  - `rv`: Rückgabewert, 0 = OK, -1 = Fehler
- Stellt Verbindung mit Socket des Servers her
  - blockiert, bis Verbindung zustandekommt
  ```

Passives Öffnen einer Verbindung (Server)

- Systemaufruf `listen`
- `int rv = listen(int sockfd, int backlog);`
  - `backlog`: legt fest, wieviele Verbindungswünsche gepuffert werden können
  - `rv`: Rückgabewert, 0 = OK, -1 = Fehler
- Teilt dem Betriebssystem mit, daß es Verbindungswünsche für diesen Socket entgegennehmen soll

Akzeptieren einer Verbindung (Server)

- Systemaufruf `accept`
- `int fd = accept(int sockfd, struct sockaddr *addr, int *addrlenptr);`
  - `addr`, `addrlenptr`: Rückgabe der Clientadresse
  - `fd`: **neuer** Socketdeskriptor zur Kommunikation mit diesem Client
- Bearbeitet ersten Verbindungswunsch aus Puffer
  - blockiert, wenn kein Verbindungswunsch vorliegt
- Server kann Verbindung erst nach `accept` zurückweisen (durch `close(fd)`)

Datentransfer

- Senden (Schreiben)
  - `int send(int sockfd, char *msg, int len, int flags);`
  - `int write(int sockfd, char *msg, int len);`
- Empfangen (Lesen)
  - `int recv(int sockfd, char *msg, int len, int flags);`
  - `int read(int sockfd, char *msg, int len);`
- `write` / `read` ist äquivalent zu `send` / `recv` mit `flags = 0`
- Ergebnis: geschriebene / gelesene Bytes bzw. -1 bei Fehler
  - Lesen blockiert, bis mindestens 1 Byte empfangen wurde
  - Ergebnis 0 bei `read`/`recv`: Verbindung v. Partner geschlossen
- Datendarstellung muß ggf. selbst konvertiert werden!
  - Funktionen `htons`, `ntohs`, `htonl`, ...

Datentransfer ...

- `send` / `recv` (und `write` / `read`) können zurückkehren, bevor die Daten komplett gesendet / empfangen wurden
- Daher: immer Verwendung in einer Schleife, z.B.:

```
char *msg = "HELO cs.tum.edu\n";
int len = strlen(msg);
int written = 0, res;
while (written < len) {
   res = write(fd, &msg[written], len-written);
   if ((res < 0) && (errno != EINTR)) {
      perror("write"); exit(1);
   }
   if (res >= 0) written += res;
}
```

Adreß-Information

- `int getsockname(int sockfd, struct sockaddr *addr, int *addrlenptr)`
  - lokale IP-Adresse und Port des Sockets
- `int getpeername(int sockfd, struct sockaddr *addr, int *addrlenptr)`
  - IP-Adresse und Port des Kommunikationspartners

Beispiel-Code

- Siehe `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/gen/rn2/code/tcp-c.zip`

Stream Sockets in Java

- Klasse `Socket`
  - Konstruktur: `Socket sock = new Socket();`
    - zur Erzeugung eines Client-Sockets
  - auch serverseitig verwendet (nach `accept()`)
- Serverseitig: Klasse `ServerSocket`
  - Konstrukturen:
    - `ServerSocket sock = new ServerSocket(int port);`
    - oder `new ServerSocket(int port, int backlog);`
    - Konstruktoren führen auch `bind()` und `listen()` aus
  - Methode `Socket accept()`
    - Client-Adresse durch `Socket`-Methoden zu erfragen

Stream Sockets in Java ...

- Verbinden eines Sockets (clientseitig)
  - über Methode `connect` der Klasse `Socket`:
    - ```
      byte[] b = { (byte)217, 72, (byte)195, 42 };
      InetAddress addr = InetAddress.getByAddress(b);
      sock.connect(new InetSocketAddress(addr, port));
      ```
    - bzw. mit DNS-Lookup, z.B.:
      ```
      sock.connect(new InetSocketAddress("www.web.de",
                                         80));
      ```
  - oder direkt über Konstruktor der Klasse `Socket`:
    - `Socket sock = new Socket(addr, port);`
    - `Socket sock = new Socket("www.web.de", 80);`
      ```

Stream Sockets in Java ...

- Weitere wichtige Methoden von `Socket`:
  - `void close()`
    - Schließen des Sockets (TCP-Verbindungsabbau)
  - `InetAddress getLocalAddress()` u. `int getLocalPort()`
    - liefern lokale IP-Adresse / Port
  - `InetAddress getInetAddress()` und `int getPort()`
    - liefern IP-Adresse / Port des Kommunikationspartners
  - `InputStream getInputStream()`
    - liefert Eingabestrom zum Lesen vom Socket
  - `OutputStream getOutputStream()`
    - liefert Ausgabestrom zum Schreiben auf den Socket

Stream Sockets in Java ...

- Lesen vom Socket (typisch):

```
BufferedReader in = new BufferedReader(
                        new InputStreamReader(
                            socket.getInputStream()
                    ));
String request = in.readLine();
```

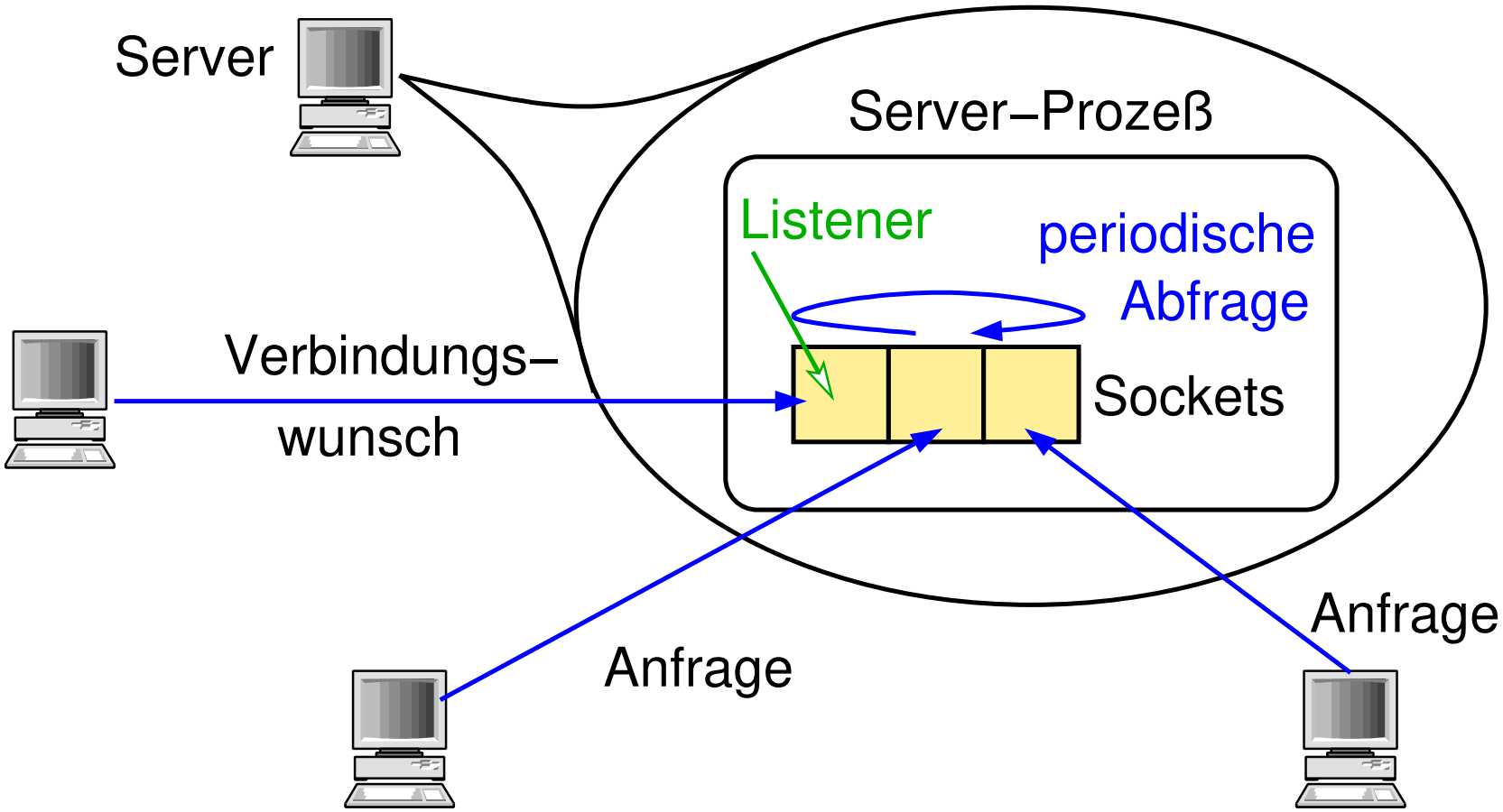
- Schreiben auf einen Socket (typisch):

```
PrintWriter out = new PrintWriter (
                        socket.getOutputStream()
                  );
out.println("This is the reply!");
out.flush();
```

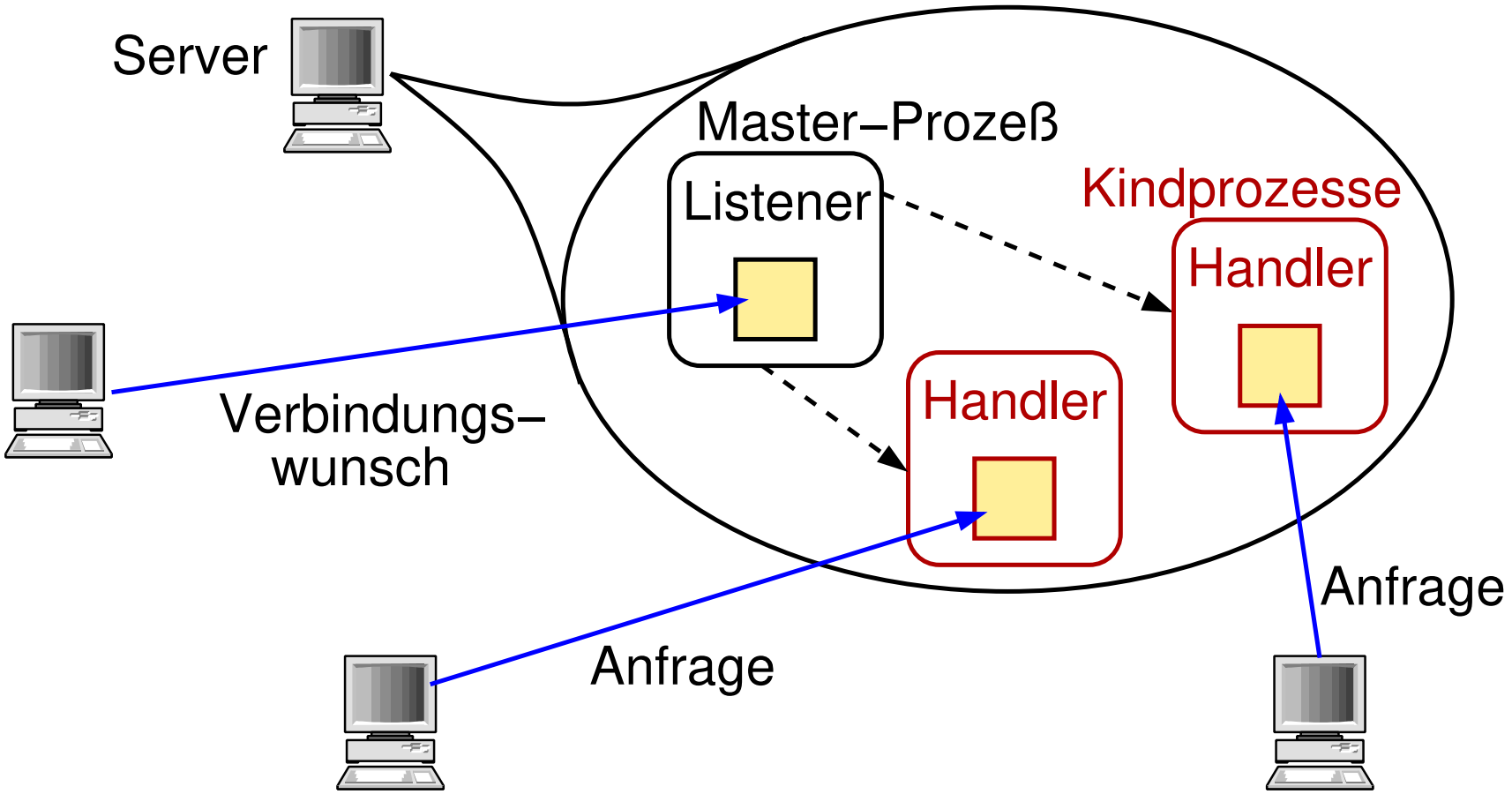
8.4 Design von Server-Programmen

- Server sollte mehrere Clients gleichzeitig bedienen können
  - d.h. Server muß gleichzeitig mehrere Sockets auf eingehende Nachrichten abfragen können
- Alternativen:
  - *Polling*: Sockets nacheinander nichtblockierend abfragen
    - durch Optionen des Sockets bzw. von `recv()`
  - für jeden Client einen neuen Prozeß erzeugen
    - UNIX: Systemaufruf `fork()` erzeugt Prozeßkopie
  - für jeden Client einen neuen Thread erzeugen
    - z.B. in Java
  - blockierendes Warten an einer Menge von Sockets
    - UNIX: Systemaufruf `select()` erlaubt Warten an einer Menge von Dateideskriptoren

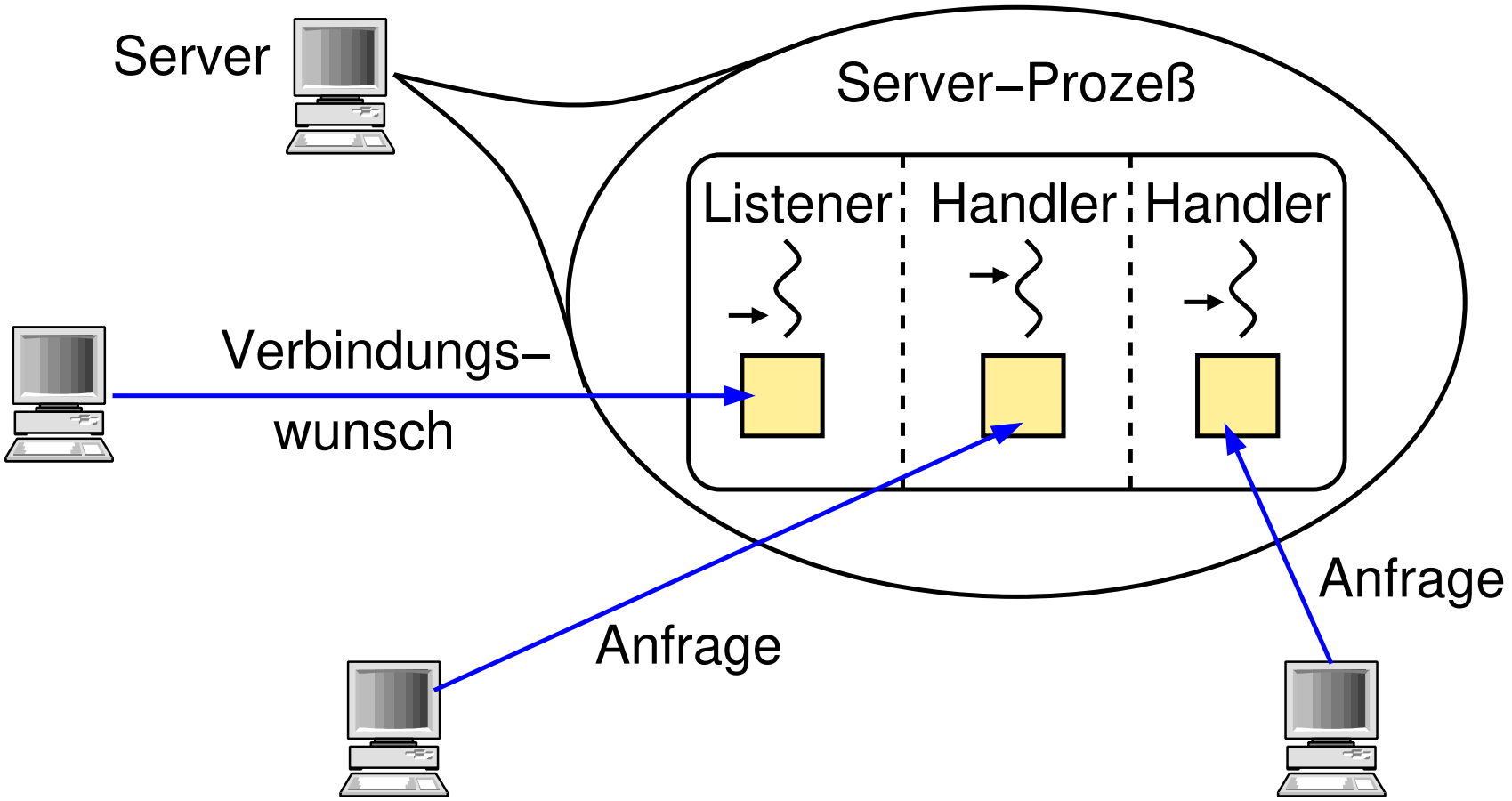
Polling Server



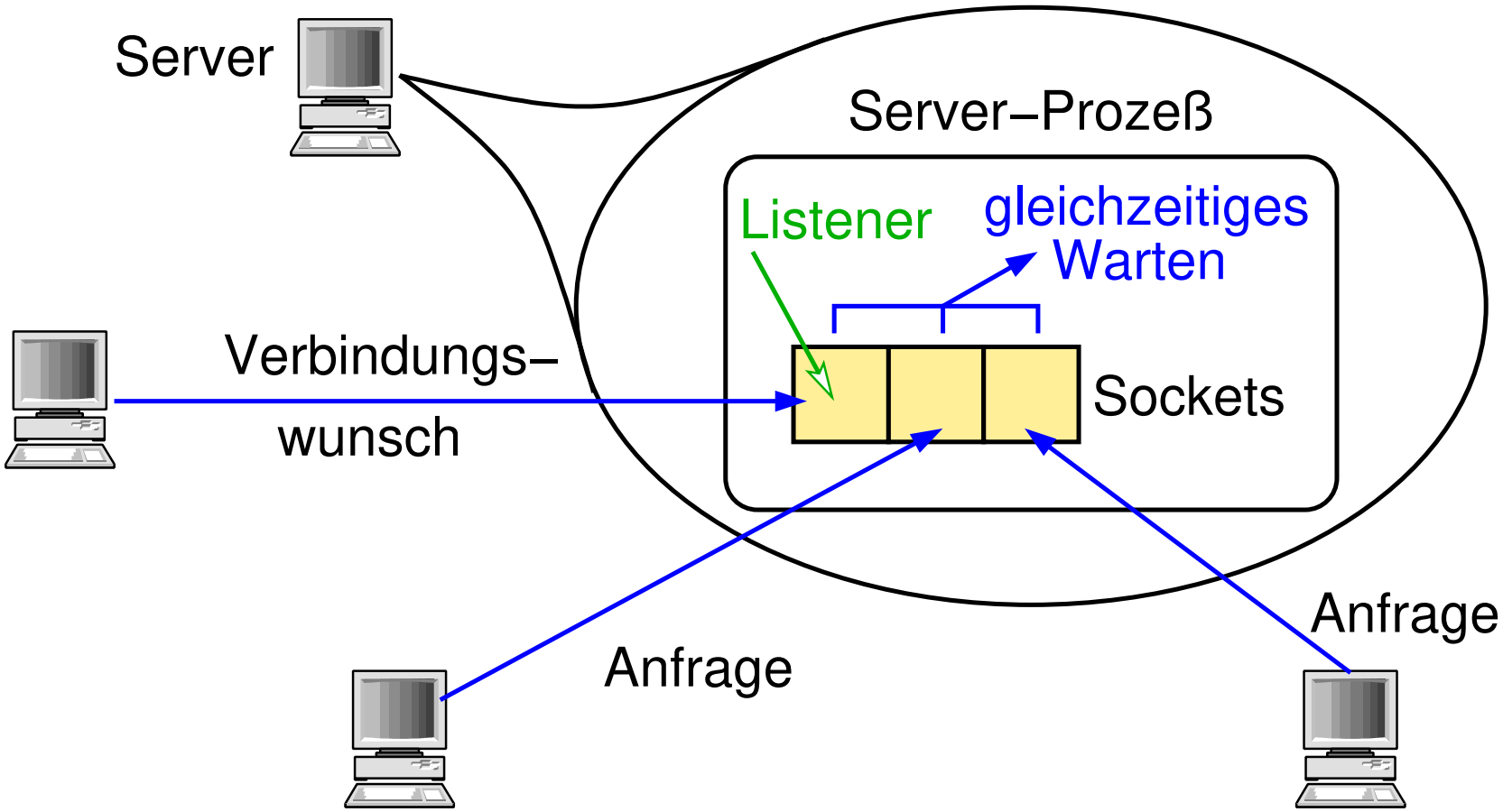
Prozeß pro Client



Thread pro Client



Select-basierter Server



Diskussion

- *Polling*: verschwendet CPU-Zeit
- Prozeß pro Client:
  - Problem: Ressourcenverbrauch bei vielen Clients
  - Schutz der einzelnen Server-Prozesse gegeneinander
- Thread pro Client:
  - weniger ressourcenintensiv als Prozeß pro Client
  - einfache Nutzung gemeinsamer Daten zw. den Threads
- Select-basierter Server:
  - benötigt keine zusätlichen Ressourcen
  - Aufträge werden rein sequentiell verarbeitet (keine Synchronisation, aber langer Auftrag blockiert alle folgenden)

Socket-Programmierung

- Socket: Abstraktion für Kommunikationsendpunkt
- Socket-API unabhängig von Netzwerk und Protokollen
- Datagramm- und *Stream* Sockets (mit IP: UDP / TCP)
- Datagramm-Kommunikation
  - `socket()`: Erzeugen eines Sockets
  - `bind()`: binden an IP-Adresse / Port (für Server)
  - `sendto()`: Senden eines Datagramms
  - `recvfrom()`: Empfang eines Datagramms
- Java-Schnittstelle für Datagramm-Sockets
  - Klasse `DatagramSocket`
  - Methoden `send()` und `receive()`

Socket-Programmierung ...

- *Stream-Sockets*
  - `listen()`: passives Öffnen des TCP-Ports durch Server
  - `connect()`: TCP-Verbindungsaufbau durch Client
  - `accept()`: Server erhält neuen Socket für akzeptierte TCP-Verbindung
  - `send()` oder `write()` zum Senden von Daten
  - `recv()` oder `read()` zum Empfangen
- Java-Schnittstelle für *Stream-Sockets*
  - Klasse `Socket`
    - Methoden `getInputStream()` und `getOutputStream()`
  - Klasse `ServerSocket`
    - Konstruktur erledigt `bind()` und `listen()`

Server-Design

- (Polling), Prozeß pro Client, Thread pro Client, Select-basiert
- Optimales Design abhängig vom Anwendungsfall

Rechnernetze II

SoSe 2025

9 Leistungssteigerung von Netzen

Inhalt

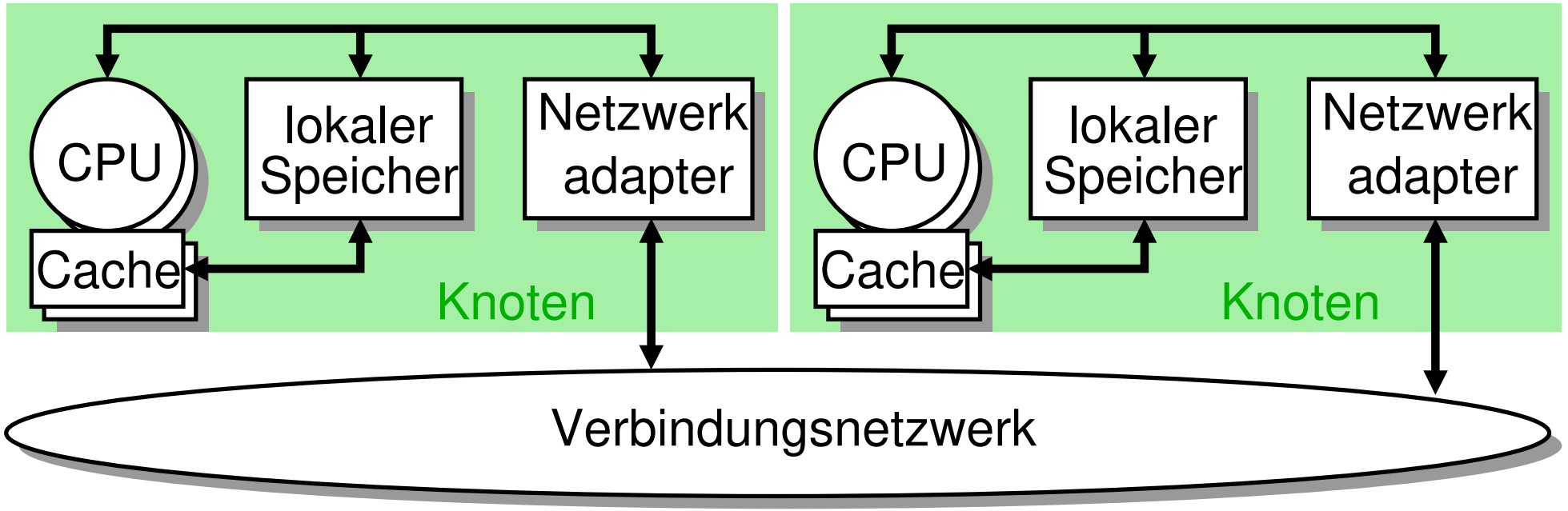
- Motivation: Hochleistungsrechner
- Maßnahmen zur Leistungssteigerung
- Beispiel: Infiniband

- Harald Richter: Verbindungsnetzwerke für parallele und verteilte Systeme. Spektrum Akademischer Verlag, 1997
- Weitere Angaben auf der WWW-Seite

Hochleistungsrechner

- Für rechenintensive Anwendungen
  - Klimasimulation, Genomanalyse, Arzneimittel-Design, ...
- Als Server
  - WWW (Suchmaschinen), Datenbanken, ...
- Multiprozessorsysteme: mehrere gekoppelte Prozessoren
  - mit gemeinsamem Speicher (SMP, ccNUMA)
  - mit verteiltem Speicher (MPP)
- Beispiel: Frontier (ORNL, USA, schnellster Rechner der Welt!)
  - 8,7 Millionen CPU-Kerne
  - $1{,}2 \cdot 10^{18}$ Flop/s (=1,2 EFlop/s) Rechenleistung

Multiprozessorsysteme mit verteiltem Speicher (MPPs)



- Sehr gut skalierbar (bis mehrere 100000 Knoten)
- Kommunikation/Synchronisation durch Nachrichtenaustausch
- Architektur entspricht Cluster von Rechnern

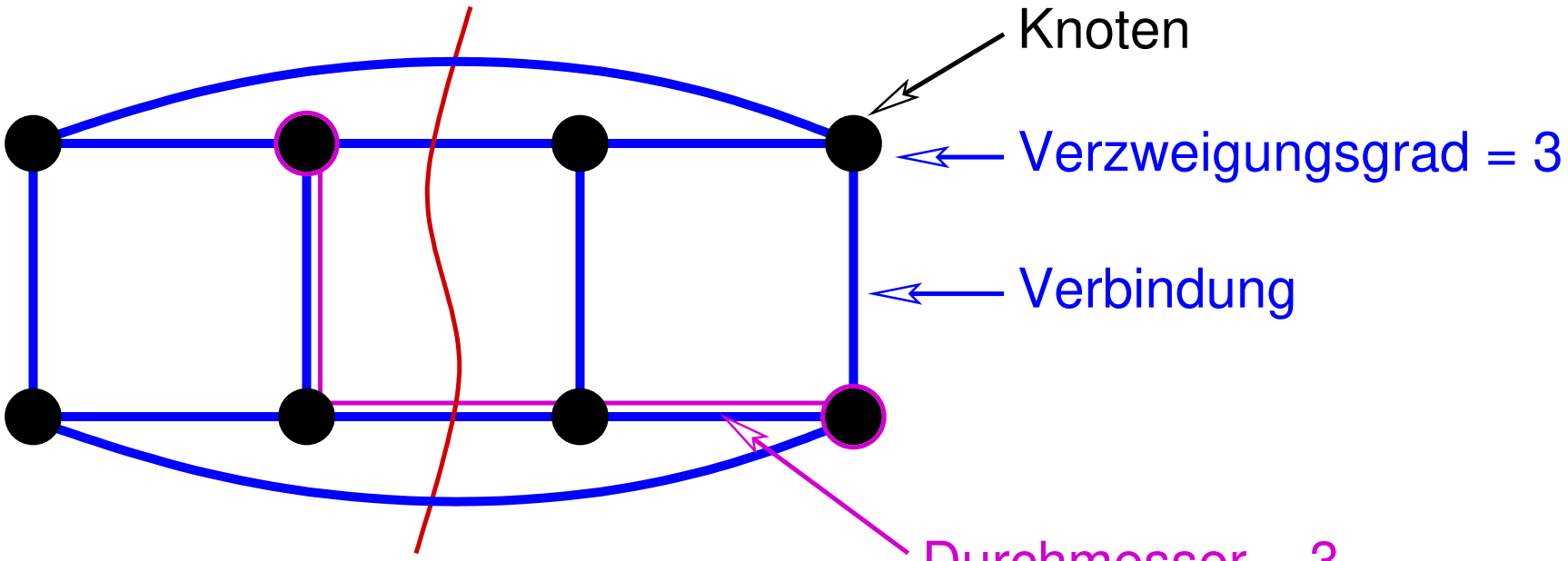
Programmierung von MPPs

- Parallele Prozesse (meist ein Prozeß pro Knoten)
  - jeder Prozeß bearbeitet ein Teilproblem
- Kooperation durch Nachrichtenaustausch
  - Punkt-zu-Punkt: *send*, *receive*
  - global: *broadcast* (1-zu-N), *reduce* (N-zu-1), ...
  - oft sehr große Nachrichten
- Wichtig für hohe Rechenleistung:
  - Kommunikationszeit $\ll$ Rechenzeit
  - Überlappung von Kommunikation und Berechnung
  - gleichzeitige Kommunikation vieler Knoten

Maßgebliche Eigenschaften der Verbindungsnetze

- Routing: i.a. mehrere mögliche Pfade
  - **Blockierungsfreiheit**: Kommunikation zweier Knoten blockiert andere Kommunikationen nicht
  - Ausfallsicherheit
- **Bisektionsbandbreite**: Gesamtbandbreite, wenn eine Hälfte der Knoten an die anderen sendet (*worst case*)
- **Durchmesser** des Netzes: Pfadlänge zwischen zwei maximal entfernten Knoten
- **Verbindungsgrad** eines Knotens: Anzahl der Links
- **Skalierbarkeit**: „Eigenschaften des Netzes bleiben bei Erweiterung erhalten“

Beispiel


Knoten
Verzweigungsgrad = 3
Verbindung
Durchmesser = 3
Bisektionsbandbreite =
4 * Bandbreite der Einzelverbindungen

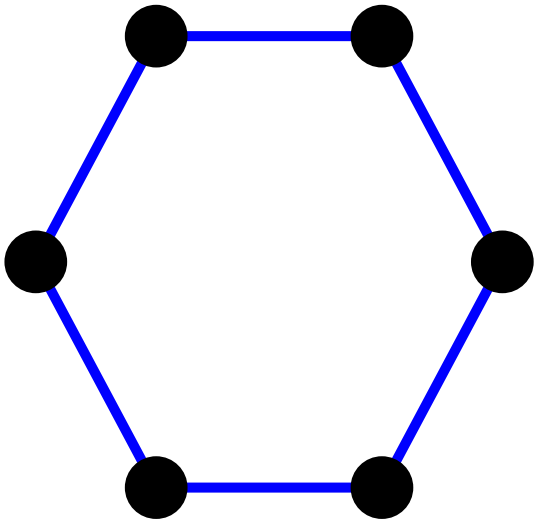
9.2 Maßnahmen zur Leistungssteigerung

- Ziele:
  - hohe Bisektionsbandbreite
  - niedrige Latenz
    - schnelle Weiterleitung in den Zwischenknoten
    - geringe Latenz durch Softwareschichten
- Mechanismen:
  - Netztopologien
  - Weiterleitungstechniken
  - Protokolle der Anwendungsschicht
    - Vermeidung von Kopiervorgängen
  - *Remote DMA* und *OS bypass*
    - Vermeidung von Betriebssystem-Einsprüngen

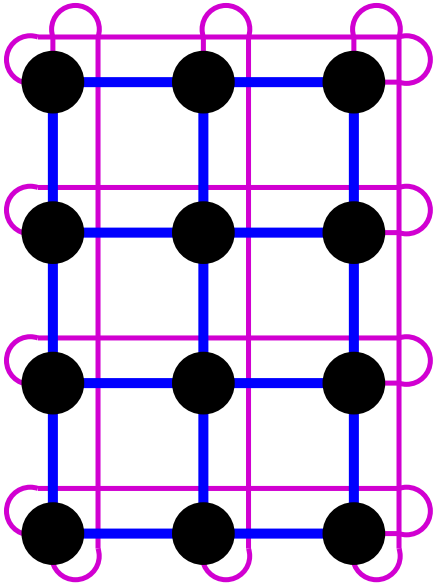
9.2.1 Netztopologien

- „Statische“ Verbindungsnetze
  - direkte Verbindungen zwischen Paaren von Knoten
  - Forwarding erfolgt durch in die Knoten eingebaute Switches
  - z.B. Ring, Gitter, Hyperwürfel
- Dynamische Verbindungsnetze
  - Knoten sind indirekt über einen oder mehrere Switches verbunden
  - z.B. Kreuzschienenverteiler, Clos-Netz

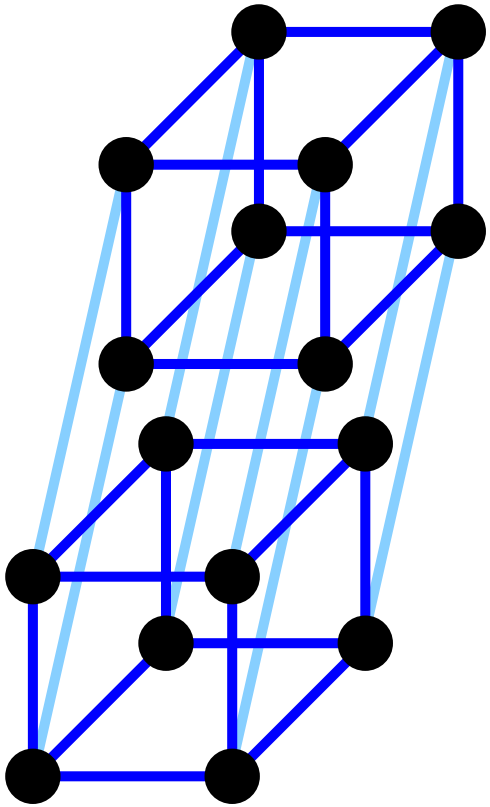
Statische Verbindungsnetze: Beispiele



Ring



Gitter / Torus

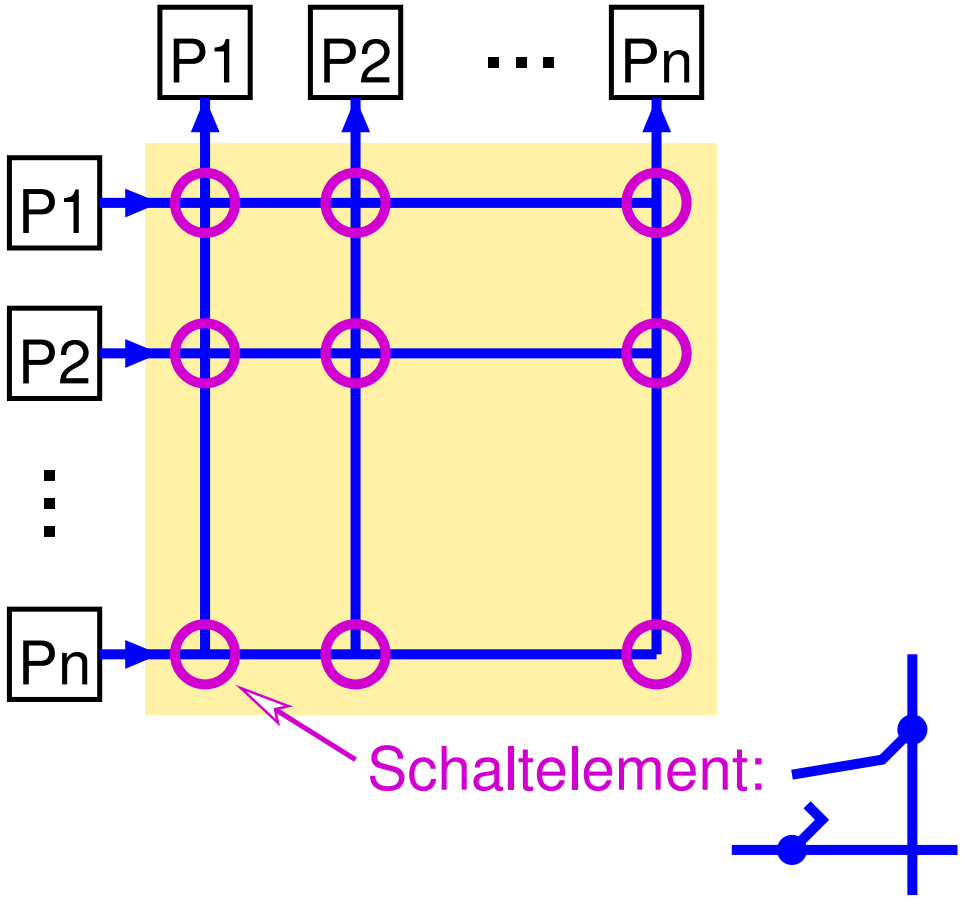


4D Hyperwürfel

	Ring	Gitter / Torus	4D Hyperwürfel
Durchmesser	O(N)	$O(\sqrt{N})$	O(log(N))
Verzweigungsgrad	2	4	O(log(N))
Bisektionsbandbr.	2 * Link	$O(\sqrt{N})$ * Link	O(N) * Link

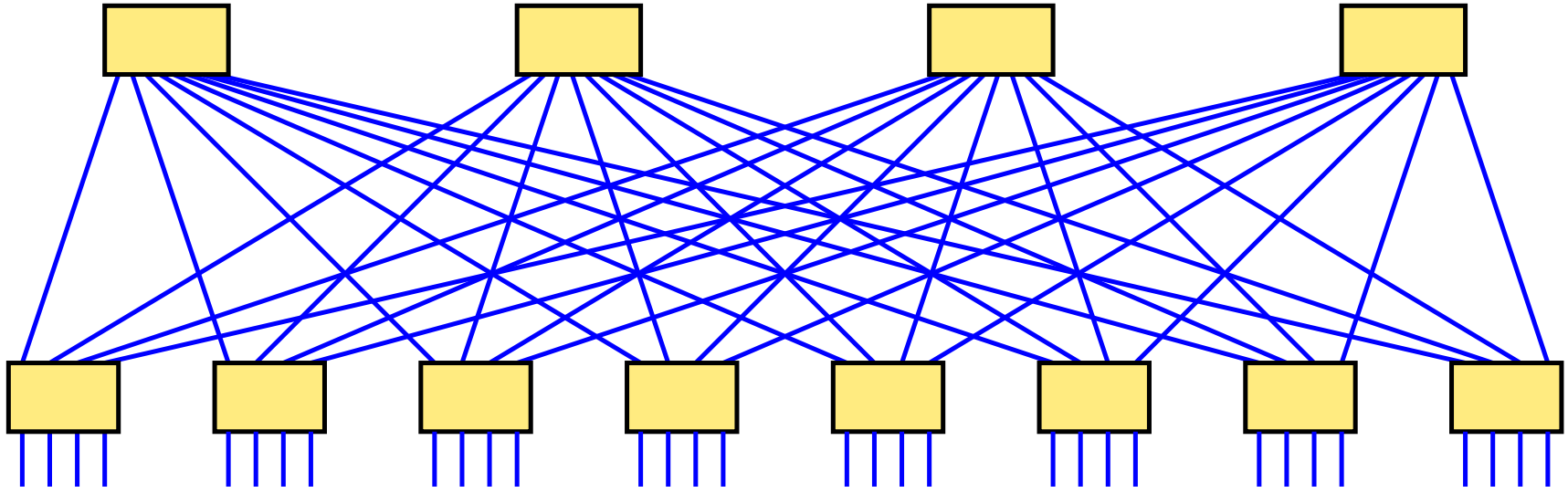
Dynamische Verbindungsnetze: Kreuzschienenverteiler (*Crossbar Switch*)

- Knoten hat getrennte Links pro Richtung
- Alle möglichen, disjunkten Knotenpaare können gleichzeitig verbunden werden
  - blockierungsfreies Netz
- Hoher Hardwareaufwand: n^2 Schaltelemente bei n Knoten
- I.a. nicht erweiterbar



Dynamische Verbindungsnetze: Clos-Netz

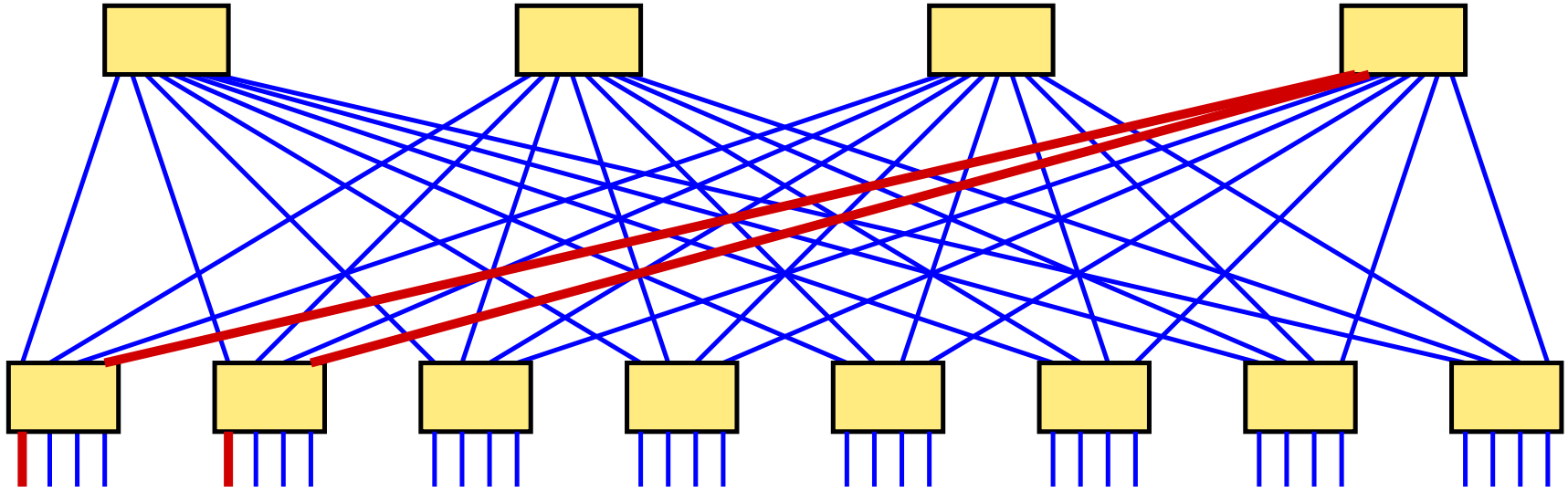
- Erweiterbares Netz auf Basis (kleiner) Kreuzschienenverteiler
  - maximal mögliche Bisektionsbandbreite
  - blockierungsfrei, wenn ex. Routen geändert werden dürfen
- Beispiel: Vernetzung 32 Knoten mit 8x8 Crossbars



32 Knoten

Dynamische Verbindungsnetze: Clos-Netz

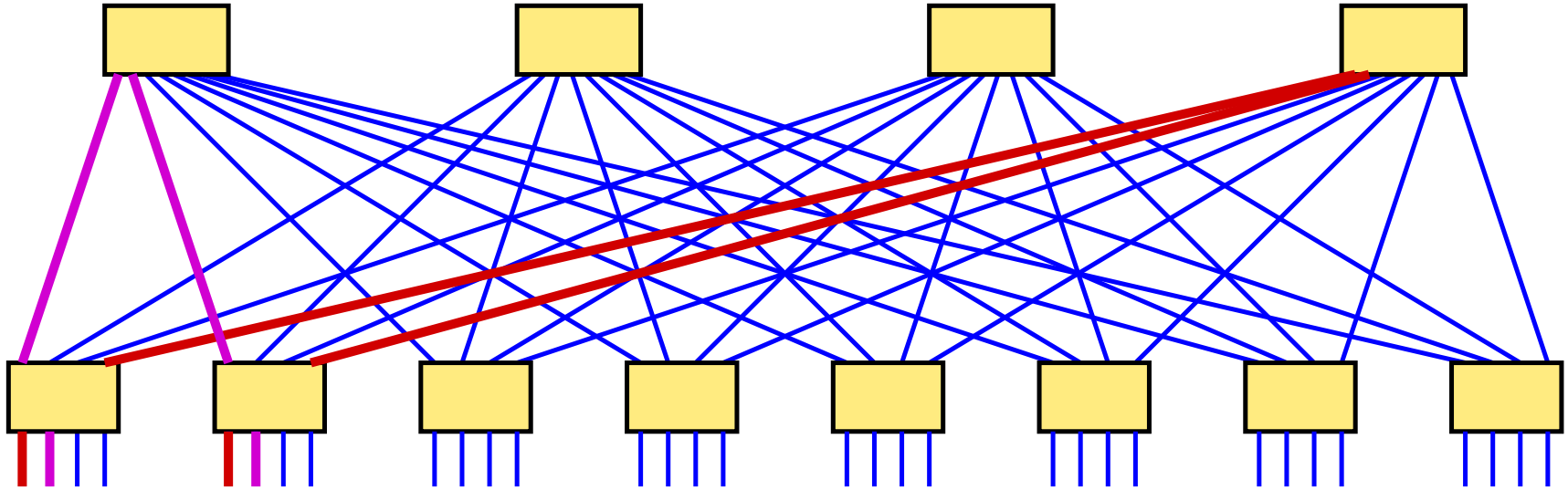
- Erweiterbares Netz auf Basis (kleiner) Kreuzschienenverteiler
  - maximal mögliche Bisektionsbandbreite
  - blockierungsfrei, wenn ex. Routen geändert werden dürfen
- Beispiel: Vernetzung 32 Knoten mit 8x8 Crossbars



32 Knoten

Dynamische Verbindungsnetze: Clos-Netz

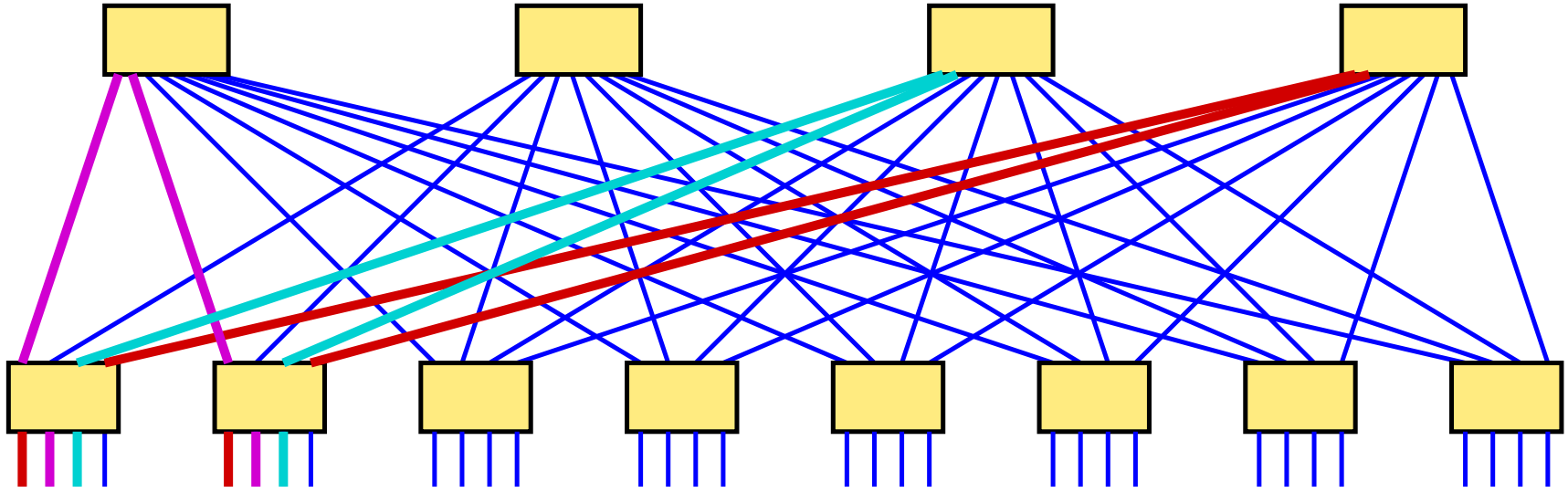
- Erweiterbares Netz auf Basis (kleiner) Kreuzschienenverteiler
  - maximal mögliche Bisektionsbandbreite
  - blockierungsfrei, wenn ex. Routen geändert werden dürfen
- Beispiel: Vernetzung 32 Knoten mit 8x8 Crossbars



32 Knoten

Dynamische Verbindungsnetze: Clos-Netz

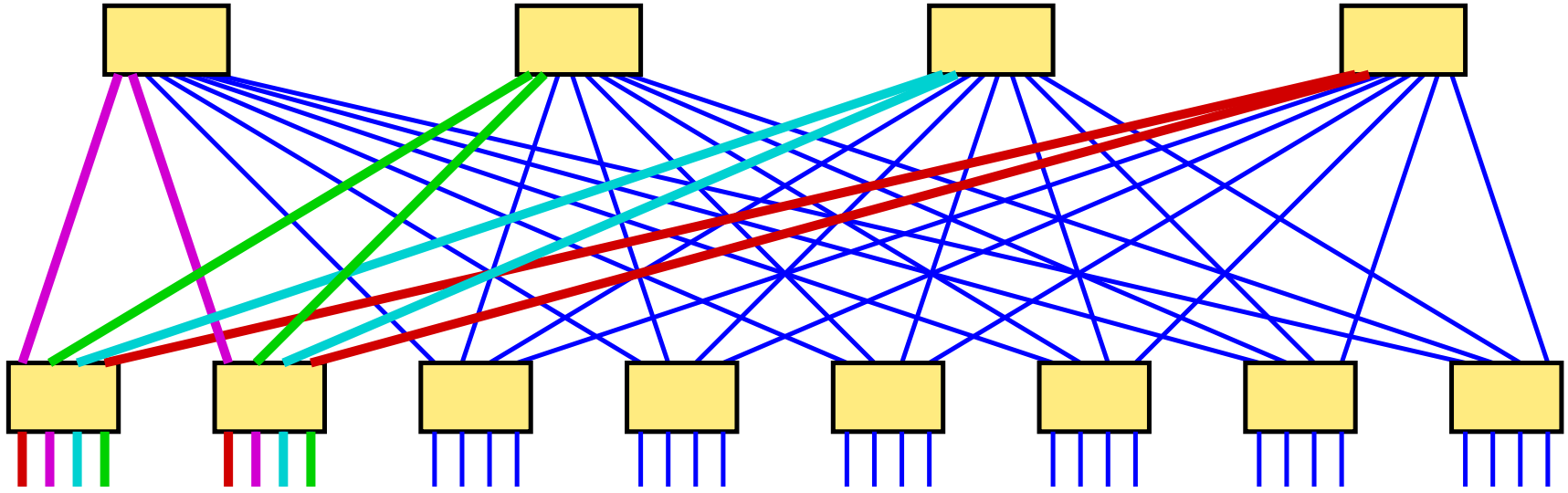
- Erweiterbares Netz auf Basis (kleiner) Kreuzschienenverteiler
  - maximal mögliche Bisektionsbandbreite
  - blockierungsfrei, wenn ex. Routen geändert werden dürfen
- Beispiel: Vernetzung 32 Knoten mit 8x8 Crossbars



32 Knoten

Dynamische Verbindungsnetze: Clos-Netz

- Erweiterbares Netz auf Basis (kleiner) Kreuzschienenverteiler
  - maximal mögliche Bisektionsbandbreite
  - blockierungsfrei, wenn ex. Routen geändert werden dürfen
- Beispiel: Vernetzung 32 Knoten mit 8x8 Crossbars



32 Knoten

9.2.2 Weiterleitungstechniken

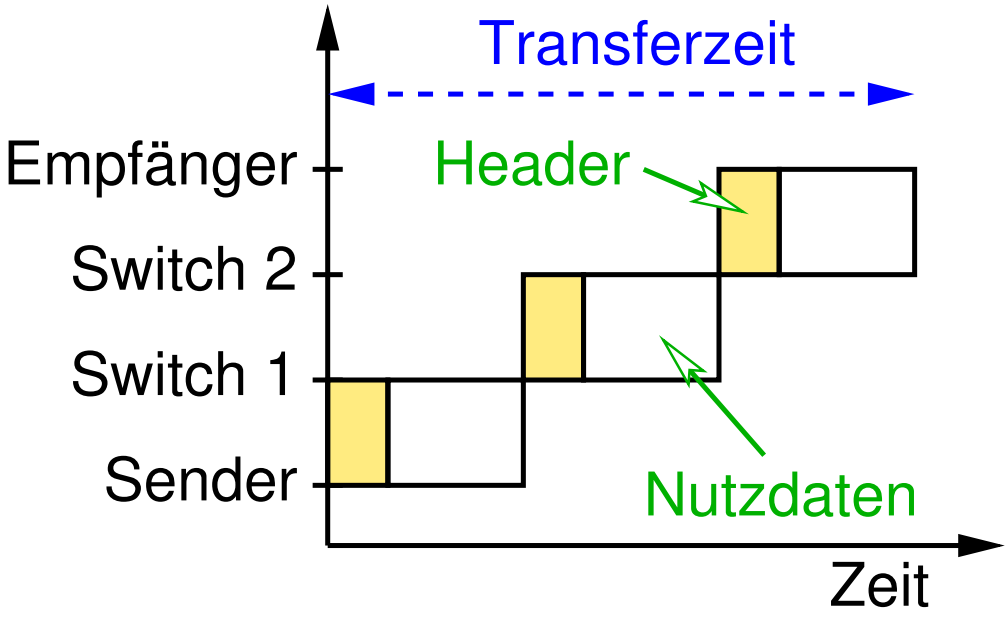
- Ziel: Schnelle Weiterleitung in den Zwischenknoten
- Randbedingung: Netze mit hoher Bandbreite und großen Paketen
  - hoher Bedarf an Pufferplatz in den Switches
  - daher ggf. Flusskontrolle zwischen Switches
- Randbedingung: mehrere mögliche Wege im Netz
  - schnelle (einfache) Routing-Verfahren erforderlich
- Trend: Zentralisierung der Steuerung des Netzverkehrs
  - *Software Defined Networking*

9.2.2 Weiterleitungstechniken ...

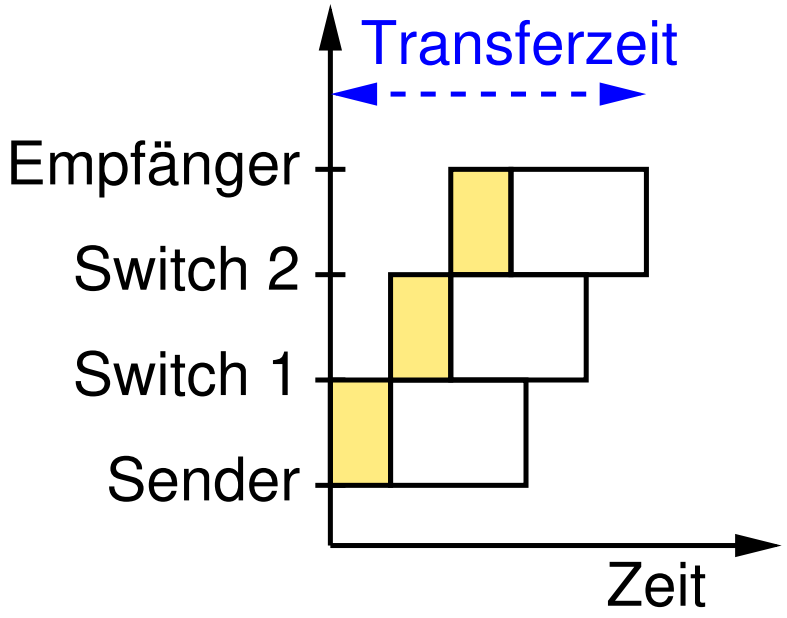
- ***Store-and-Forward Routing***
  - Switch empfängt Paket vollständig, analysiert Header, gibt Paket an entsprechenden Ausgangsport weiter
  - Puffer für mindestens ein Paket notwendig
  - paketorientierte Flußkontrolle
  - Problem: hohe Latenz
- ***Virtual-Cut-Through Routing***
  - Switch schaltet Weg zum passenden Ausgangsport bereits nach Empfang des Headers durch
    - benötigt schnelle Forwarding-Logik
  - falls Ausgangsport belegt: Puffern des gesamten Pakets
  - paketorientierte Flußkontrolle

Vergleich von *Store-and-Forward* und *Virtual-Cut-Through*

Store–and–Forward–Routing



Virtual–Cut–Through–Routing



- Anmerkung: auch schnelle Ethernet-Switches verwenden heute *Virtual-Cut-Through*

9.2.2 Weiterleitungstechniken ...

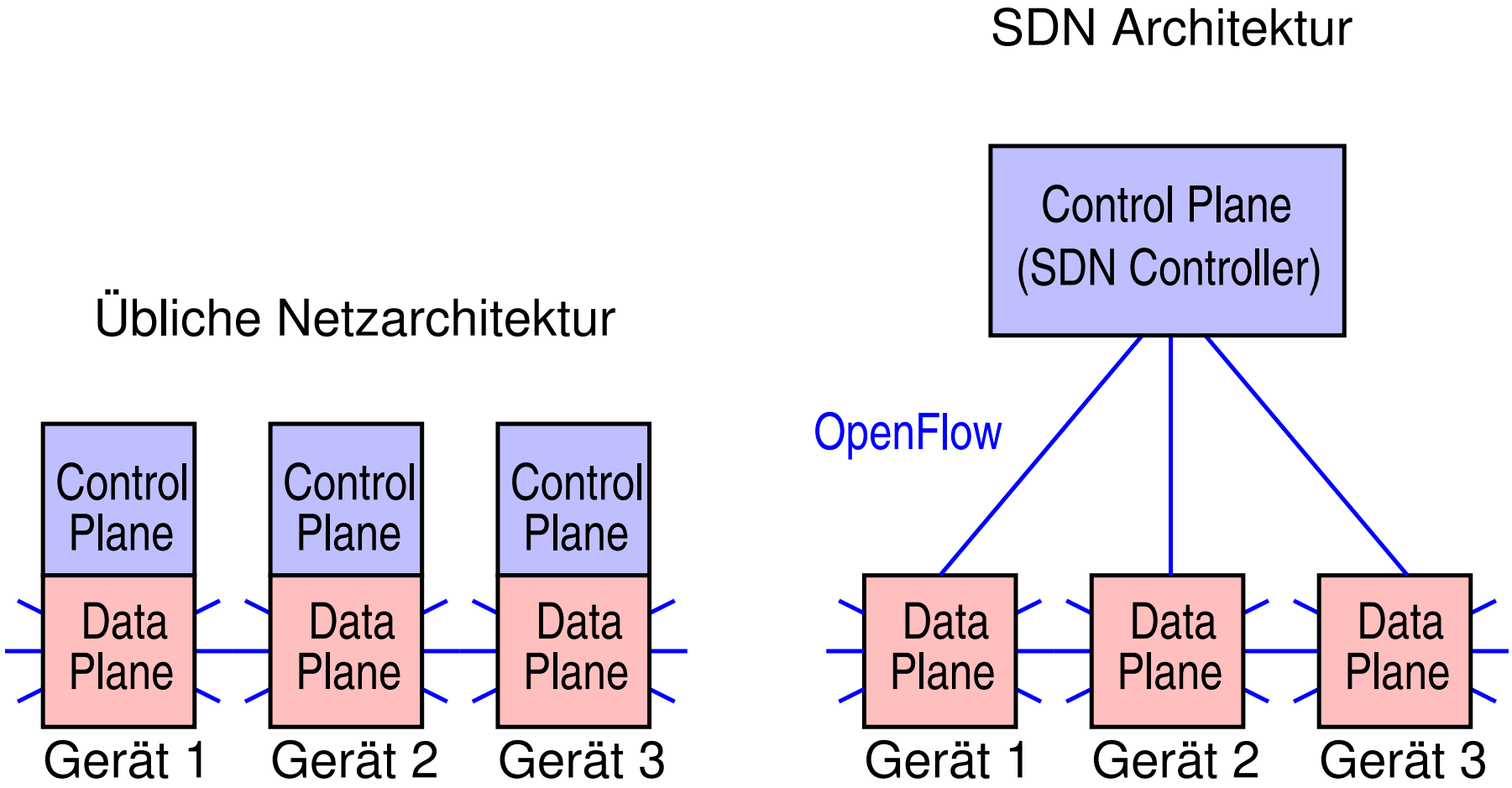
- ***Wormhole-Routing***
  - Weiterleitung wie bei *Virtual-Cut-Through*
  - feingranulare Flußkontrolle auf Bitübertragungsebene
    - Flits (*flow control digits*), typ. 1-2 Bytes
  - in Switches nur Puffer für wenige Flits (abhängig von RTT)
  - bei belegtem Ausgangsport: Rückstau des Pakets
    - Flits enthalten keine Zieladresse $\Rightarrow$ Links bleiben blockiert
  - Vorteil: geringe Latenz, wenig Pufferbedarf
  - Nachteil: (größere) Gefahr von Deadlocks
    - Vermeidung durch entsprechende Routing-Algorithmen

Software Defined Networking (SDN)

- Schichtenaufbau eines Weiterleitungsknotens:
  - *Data Plane*: regelbasierte, tabellengesteuerte Weiterleitung von Frames/Paketen
  - *Control Plane*: Erstellung der Regeln bzw. Weiterleitungstabellen (z.B. durch Routing-Protokolle)
- Grundideen von SDN:
  - Trennung von *Data Plane* und *Control Plane*
  - Zentralisierung der *Control Plane*
    - Kommunikation zwischen zentralem Controller und Weiterleitungsknoten über das Netzwerk
    - Protokoll z.B. OpenFlow (über TLS / TCP)

Software Defined Networking (SDN) ...

- Vergleich der Architekturen:

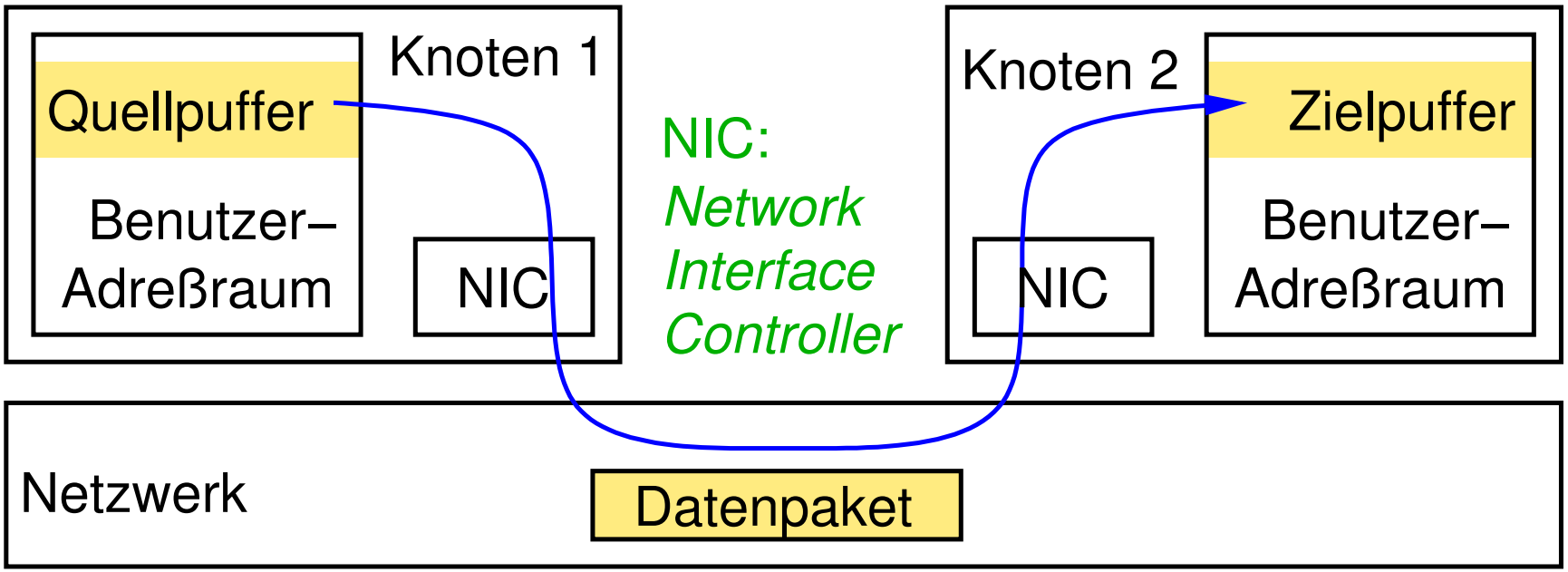


Vorteile von SDN:

- Flexiblere Weiterleitungsregeln
  - bei OpenFlow u.a. basierend auf:
    - Quell-/Ziel-MAC, VLAN-ID, VLAN Priorität
    - MPLS Label
    - Quell-/Ziel-IP-Adresse, Flow-Label, Protokoll
    - Quell-/Ziel-Port
  - Weiterleitung kann flussbasiert erfolgen
  - damit insbes. bessere QoS Unterstützung
- Intelligentere Controller möglich
  - z.B. können Weiterleitungsregeln automatisch aufgrund vorgegebener Policies erstellt werden

9.2.3 Protokolle der Anwendungsschicht

- Nachrichtenaustausch auf Anwendungsebene:
  - Kopieren von Daten aus dem Adreßraum des Senders in den Adreßraum des Empfängers

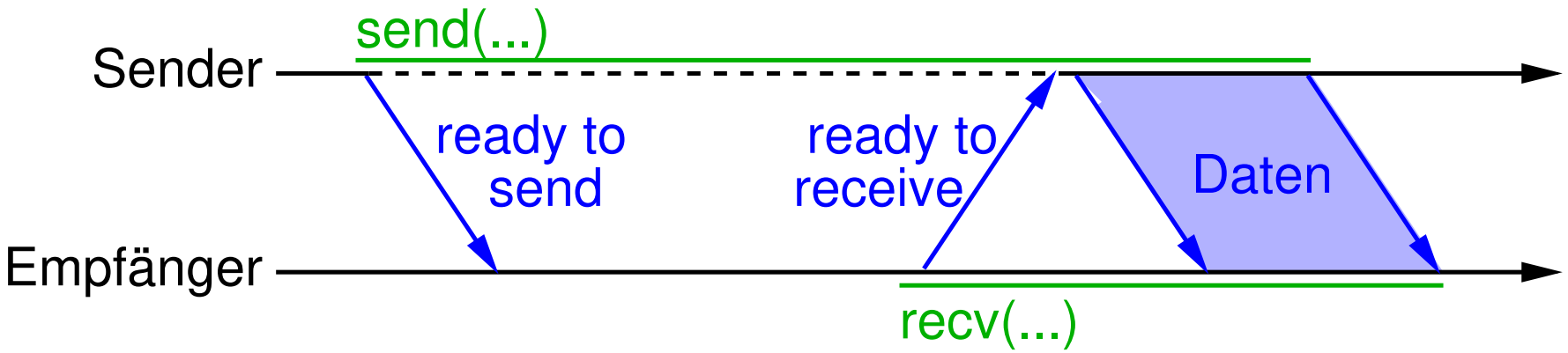


  - Problem: Adresse des Zielpuffers erst bekannt, wenn Empfänger bereits auf Nachricht wartet

Asynchrones, optimistisches Protokoll

- Senderprozeß verschickt Nachricht ohne Wissen über den Zustand des Empfängerprozesses
- Falls Empfangsoperation noch nicht gestartet: Kommunikations-bibliothek muß Nachricht zwischenspeichern
  - Probleme:
    - Speicherplatzbedarf
    - Zusätzliche Kopie bei Empfangsoperation
- Daher nur für kurze Nachrichten eingesetzt
  - lange Nachrichten mit synchronem Protokoll (∼ RTS/CTS) (oder Zusicherung des Senders, daß Empfänger bereit ist)

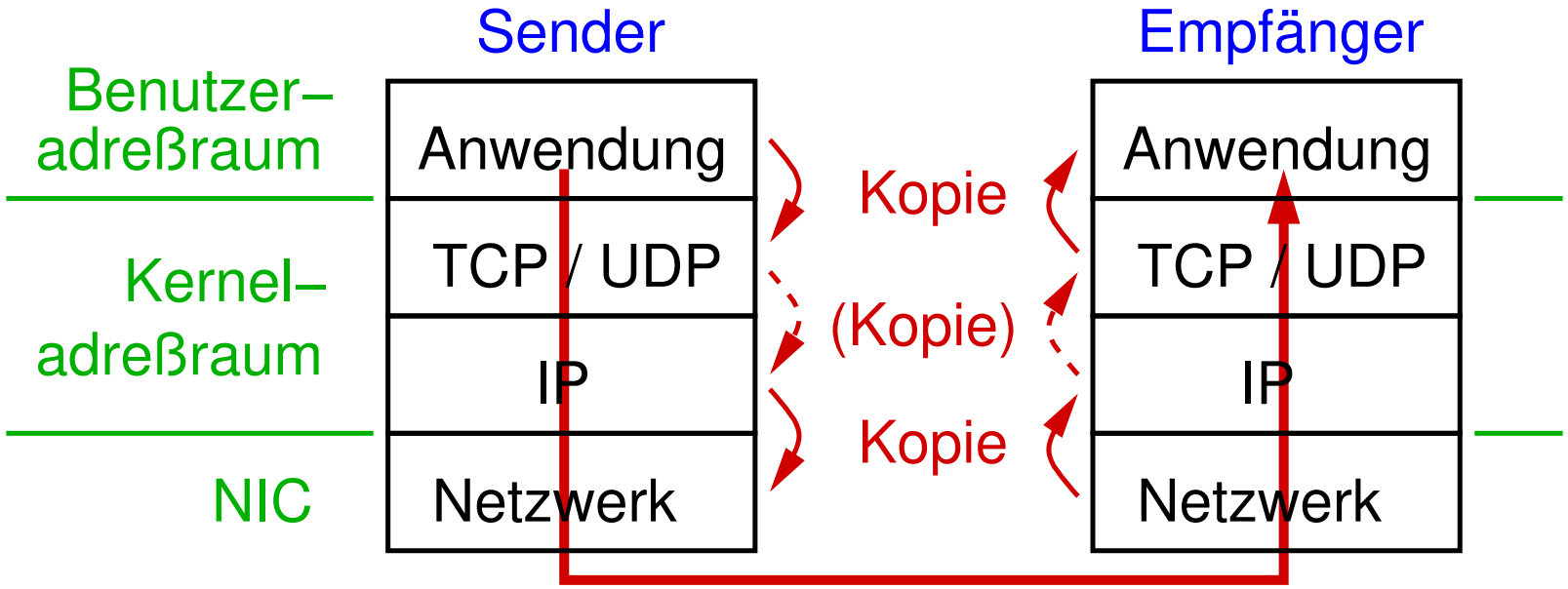
Synchrones Protokoll



- Sender wird blockiert, bis Empfang gestartet wurde
- *ready-to-send*-Nachricht enthält Länge der Daten
- Vorteil:
  - kein zusätzlicher Pufferspeicher notwendig
  - Vermeidung einer Kopieroperation beim Empfänger
- Nachteil: evtl. höhere Latenz und Blockierung des Senders

9.2.4 *Remote DMA* und *OS Bypass*

- Kommunikation in geschichteten Systemen (hier IP):

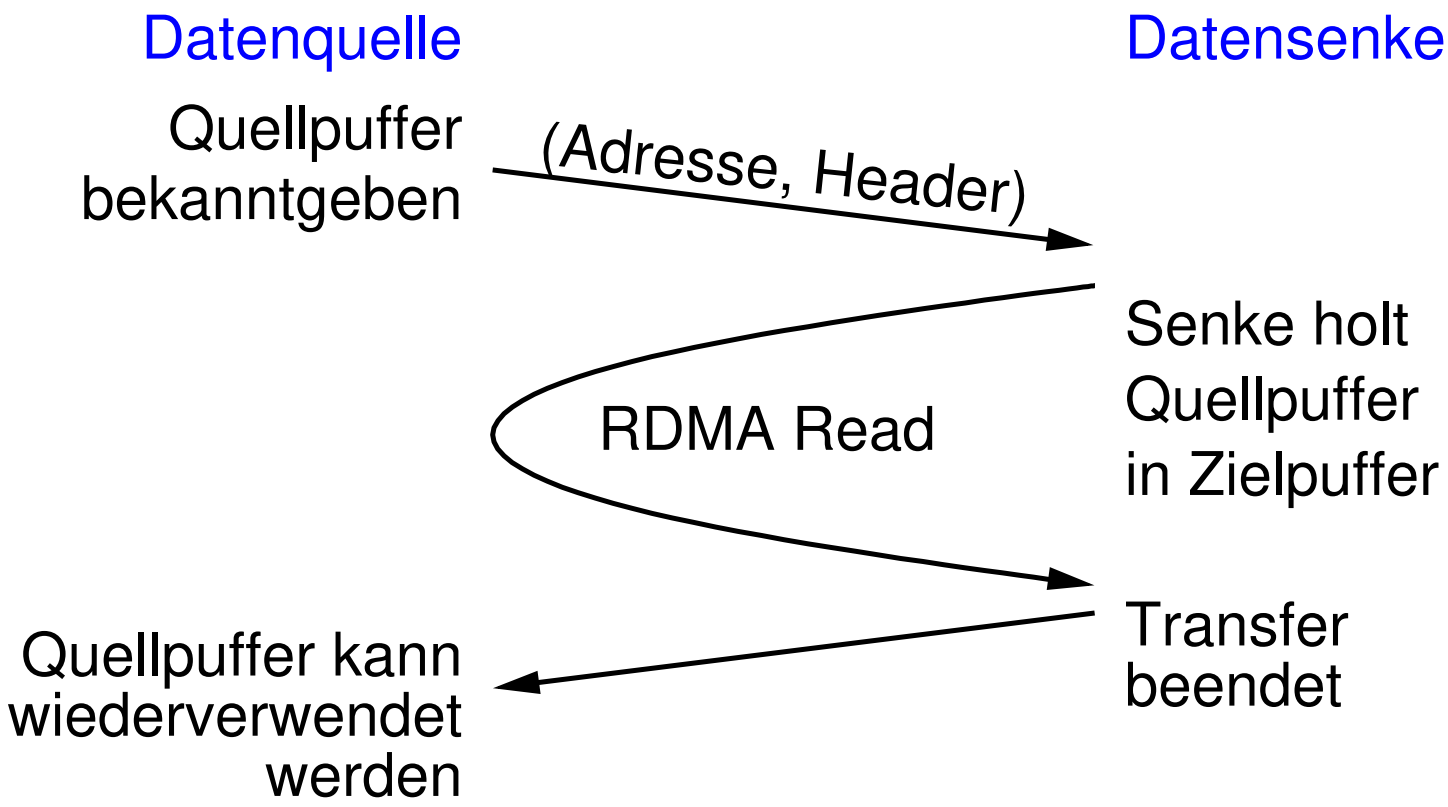


- Kopieroperationen limitieren Bandbreite, erhöhen Latenz, belasten CPU
- Ziel: Kopieren möglichst vermeiden („*zero copy*“)

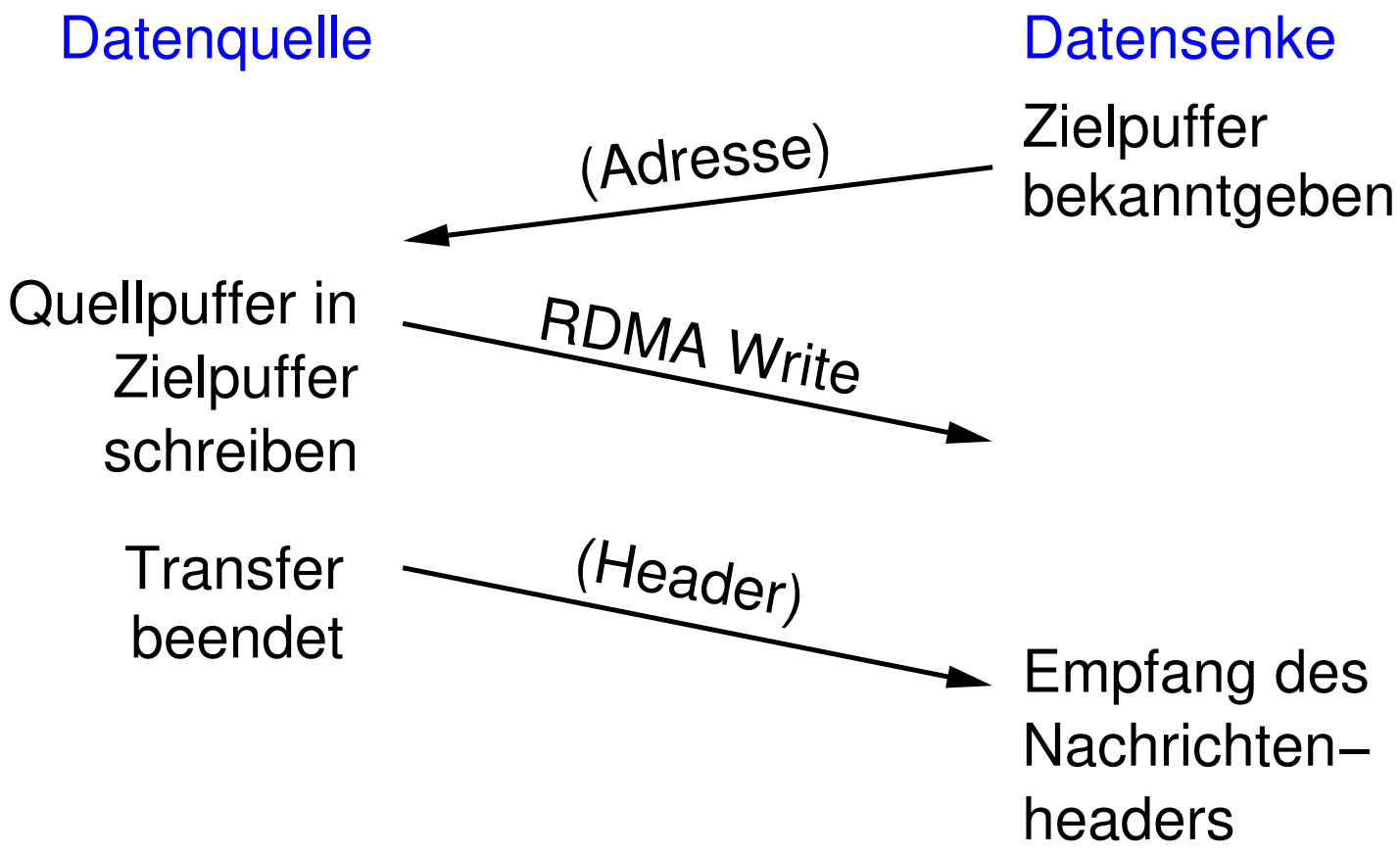
Remote DMA

- DMA (*Direct Memory Access*)
  - DMA-Controller führt selbständig Datentransfers zwischen zwei Speicherbereichen durch (parallel zu CPU-Aktivität)
  - Auftrag durch die CPU: Quell- und Zieladresse, Länge
- *Remote DMA*
  - DMA-Transfer zwischen (Benutzer-)Speicherbereichen auf **verschiedenen** Rechnern über ein Netz
  - Operationen: RDMA Read, RDMA Write
    - Protokolle werden vollständig durch NIC realisiert
  - Betriebssystem muß Transfer nur noch anstoßen

Nachrichtenübertragung mit RDMA Read



Nachrichtenübertragung mit RDMA Write



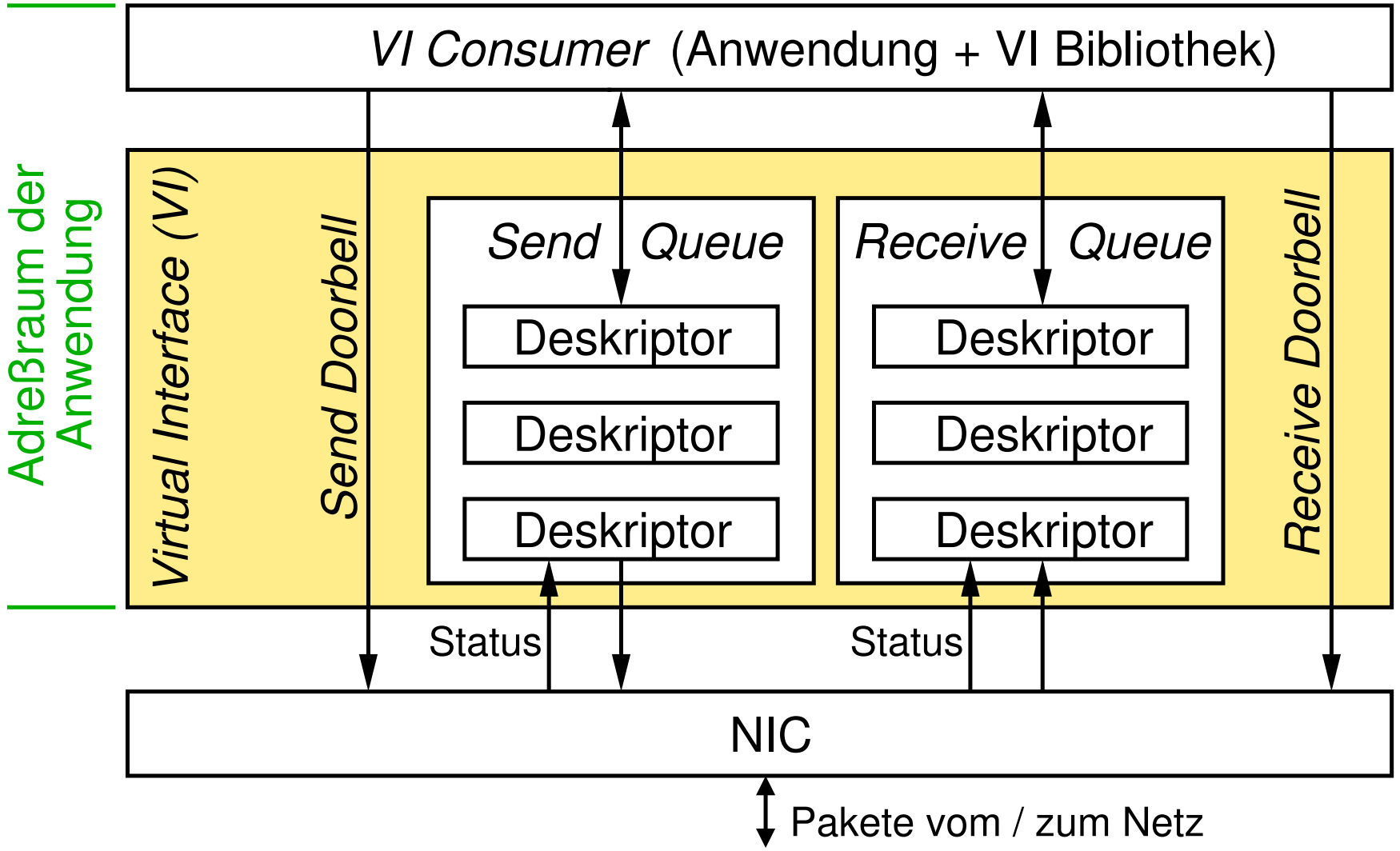
VIA: *Virtual Interface Architecture*

- Spezifikation von Intel, Compaq und Microsoft (1997)
- Ziel: Betriebssystem (BS) vollständig aus kritischem Pfad der Kommunikation entfernen
- Trennung von Steuerung und Daten
  - BS nur noch zum Aufsetzen von Verbindungen nötig (verbindungsorientiertes Protokoll)
  - Kommunikation ohne Einbeziehung des BS möglich
- Datenaustausch wird vollständig durch NIC realisiert, insbesondere Multiplexing und Demultiplexing
  - jeder Prozeß bekommt eigene virtuelle Schnittstelle zum NIC

VIA Operationen

- Initialisierung und Terminierung (über BS)
  - erzeugen der virtuellen Schnittstelle
- Registrierung und Deregistrierung von Puffern (über BS)
  - Puffer werden im physischen Adreßraum fixiert
- Verbindungsauf- und -abbau (über BS)
  - BS programmiert NIC und erzeugt Zugriffsschlüssel
- Datentransfer (direkt über virtuelle Schnittstelle)
  - Senden, Empfangen, RDMA Write, RDMA Read (optional)

VIA: Aufbau der virtuellen Schnittstelle



VIA: Ablauf eines Datentransfers

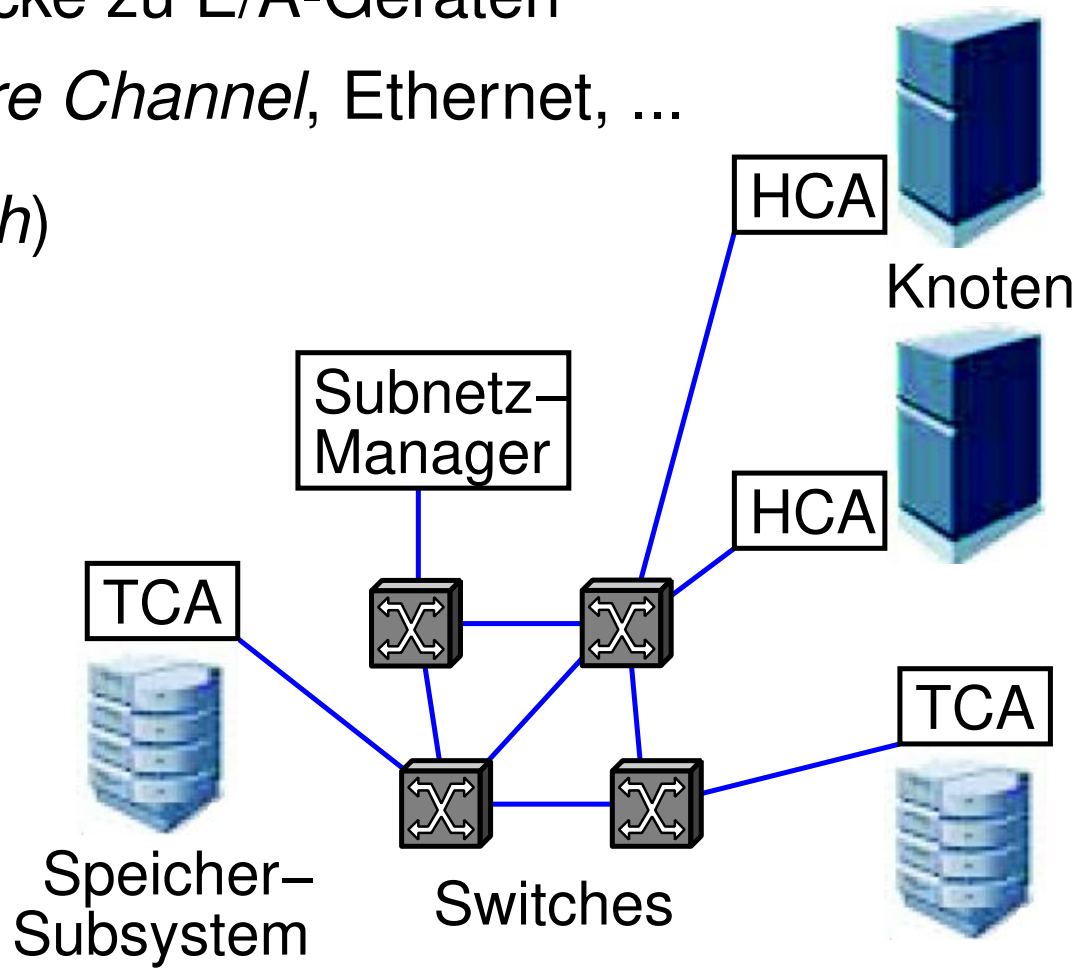
- Senden:
  - Deskriptor erzeugen (Adresse und Länge des Sendepuffers) und in *Send Queue* einreihen
  - *Send Doorbell* läuten
  - Statusbit im Deskriptor zeigt Ende der Operation an, Polling durch Anwendung (oder: blockierender BS-Aufruf)
- Empfangen: analog
  - eingehende Pakete ohne passenden Deskriptor werden verworfen
- RDMA Write / Read:
  - Initiator gibt auch Pufferadresse im anderen Knoten an

9.3 Beispiel: Infiniband

- Ziel: Hochgeschwindigkeitsnetz für Cluster
  - Kommunikation zwischen den Knoten
  - Anbindung von E/A-Geräten (Alternative zu E/A Bussen)
- Leistungsdaten:
  - Bandbreite bis zu 300 Gb/s
  - Hardware-Latenz pro Switch: ca. 100ns
  - zwischen Anwendungen: Latenz 1,32 μs, Durchsatz 952 MB/s (mit 8 Gb/s Link)
- Infiniband umfasst die OSI-Schichten 1-4
  - Subnetze beliebiger Topologie, ggf. über Router verbunden
    - häufig: *Fat Tree*, Clos-Netzwerke, 3D-Torus, ...
- Unter den 500 schnellsten Rechnern der Welt: 40% mit Infiniband (45% mit 10G Ethernet)

Komponenten eines Infiniband-Subnetzes

- *Host Channel Adapter*: Netzwerkkarte
- *Target Channel Adapter*: Brücke zu E/A-Geräten
  - Schnittstelle zu SCSI, *Fibre Channel*, Ethernet, ...
- Switches (*Virtual-Cut-Through*)
- Subnetz Manager
  - zentrale (ggf. redundante) Komponente
  - Konfiguration der Switches (Weiterleitungstabellen!)
  - Netzwerkmonitoring



Adressierung

- LID (*Local ID*): 16-Bit Adresse innerhalb des Subnetzes
  - wird vom Subnetz Manager zugewiesen
- GUID (*Globally Unique ID*): weltweit eindeutige 64-Bit Adresse
  - analog zur MAC-Adresse bei Ethernet
- GID (*Global ID*): gültige IPv6 Adresse
  - z.B. gebildet aus Subnetz-Präfix und GUID

Paketformat

- Lokaler und globaler Header (Schicht 2 bzw. 3), ggf. weitere
- Zwei CRCs: für Ende-zu-Ende und *Hop-by-Hop* Fehlererkennung
- Wählbare MTU: 256 B, 1 KB, 2 KB, 4 KB

Bitübertragungsschicht

- Unterschiedliche Datenraten
  - SDR (2 Gb/s), DDR (4 Gb/s), QDR (8 Gb/s), FDR (14 Gb/s), EDR (25 Gb/s)
- Verbindungsleitungen: Kupfer oder Glasfaser
  - ein Kabelpaar (bzw. eine Faser) pro Übertragungsrichtung
  - Bündelung von 4 bzw. 12 Leitungen möglich (4x / 12x Link)
  - max. Länge ca. 10-17m (Kupfer), 125m-10km (Glasfaser)
- *Auto-negotiation* für Datenrate und Linkbreite
- Codierung:
  - 8B10B bei SDR, DDR und QDR; 64B66B bei FDR und EDR
  - Kontrollcodes für Framing, Auto-negotiation, Training, ...

Sicherungsschicht

- Physische Links unterteilt in 2-16 virtuelle Verbindungen (*Virtual Lanes*, VLs)
  - VLs besitzen unterschiedliche Prioritäten (für QoS)
  - jeweils eigene Warteschlange, *Weighted RR* Scheduling
  - VL 15 (höchste Priorität) reserviert für Netzwerkmanagement
- Innerhalb jeder VL (außer VL 15)
  - paketweise Flußkontrolle
  - Überlastkontrolle (ähnlich DECbit, optional)

Transportschicht

- Verbindungslose und verbindungsorientierte Dienste
  - jeweils unzuverlässig und zuverlässig
  - zusätzlich *Raw*-Modus zum Transport von z.B. IPv6 Paketen
  - Paketierung und Sicherungsprotokoll in Hardware realisiert
- Verbindungsorientierte Dienste unterstützen Fehlertoleranz
  - Aufbau von zwei alternativen Verbindungen möglich
    - automatisches Umschalten im Fehlerfall
  - Anhalten („Leerlaufen lassen“) der Warteschlagen möglich
    - Verbindung wird dann durch Software umkonfiguriert
- Neben Send/Receive auch RDMA (Read/Write/Atomics)
  - jeweils mit *OS-Bypass* (ähnlich zu VIA)

9.4 Fazit / Zusammenfassung

- Optimierung von (Bisektions-)Bandbreite und Latenz
  - Kosten (oft) sekundär
- Heute Vernetzung i.d.R. mit *Crossbar*-Switches
  - blockierungsfrei
  - *Virtual-Cut-Through* bzw. *Wormhole*-Routing
    - feingranulare Flußkontrolle auf Bitübertragungsebene
  - bei höherer Knotenzahl: z.B. Clos-Netze
- Trend: *Software Defined Networking*
- Zur Reduktion der Latenz:
  - Einsatz von RDMA („*zero copy*“)
  - Betriebssystem-*Bypass*, z.B. VIA

Rechnernetze II

SoSe 2025

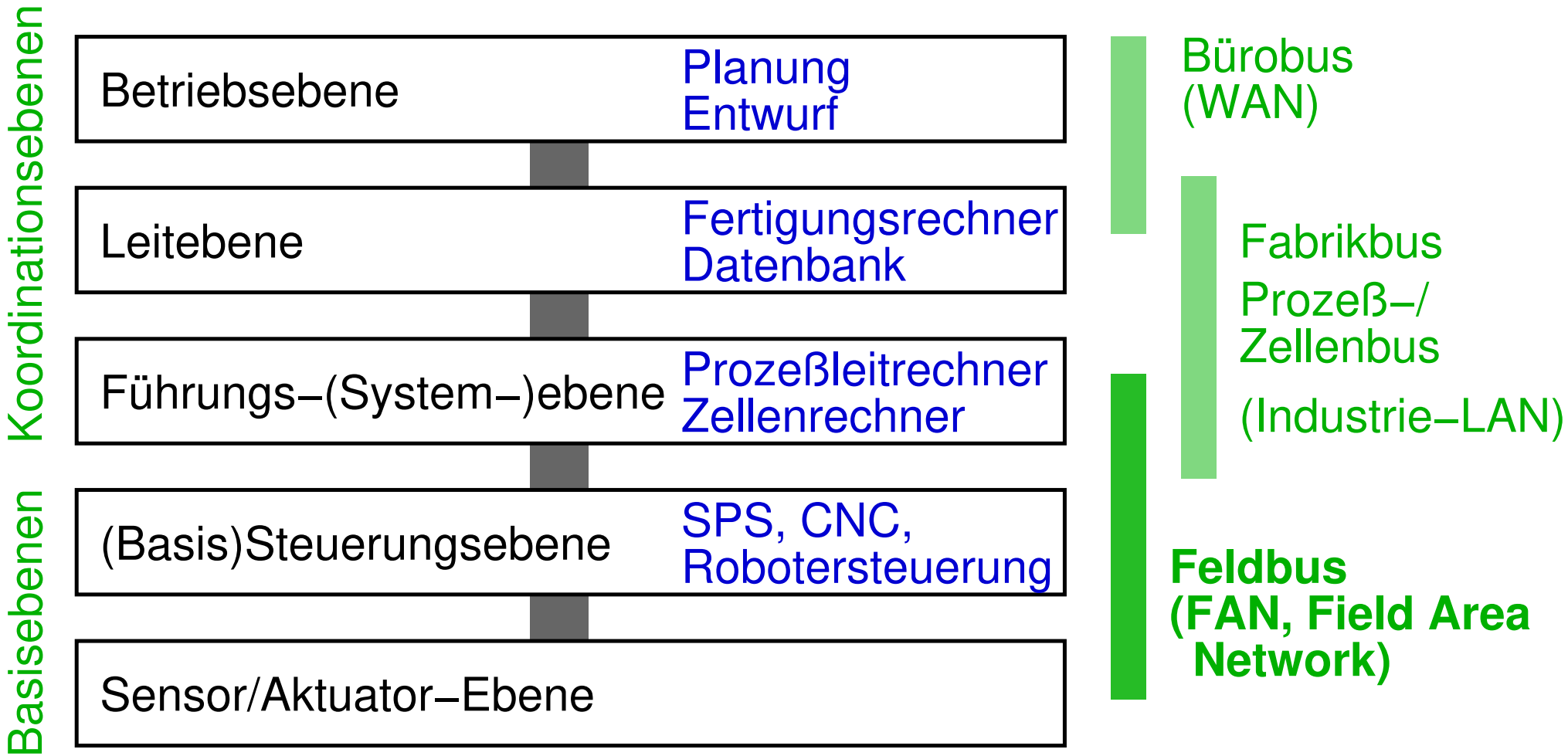
10 Netze für Automatisierungssysteme

Inhalt

- Einführung
- Typische Merkmale von Feldbussen
- CAN
- Echtzeit-Ethernet

- Gerhard Schnell: Bussysteme in der Automatisierungs- und Prozesstechnik, Vieweg Verlag.
- Vorlesung Prof. Varchmin, TU Braunschweig
  https://docplayer.org/12583965-Industrielle-kommunikation-mit-feldbussen.html

Hierarchieebenen in der Automatisierung



Einige spezielle Anforderungen an Feldbusse

- Zuverlässigkeit, geringe Störempfindlichkeit
  - Störungen durch elektrische Maschinen
- Einfache, kostengünstige Vernetzung
  - günstige Verkabelung und Endgeräte
- Übertragung von Prozeßdaten (Sensoren/Aktoren)
  - viele Teilnehmer, kleine Informationsmengen
  - zyklische Datenerfassung
  - Echtzeitanforderungen: konstante und vorhersagbare Abtastintervalle
- Sicherheitsanforderungen (z.B. Explosionsschutz)

Echtzeit

- Information muß zu bestimmtem Zeitpunkt vorliegen
- Abstufungen:
  - weiche Echtzeit: Nutzen der Information sinkt nach der Deadline stetig
  - harte Echtzeit: Nutzen nach der Deadline ist sofort Null
    - Spezialfall: obere und untere Schranke
- Relevante Parameter im Netzwerk-Bereich:
  - Verzögerung
  - Jitter: Variation der Ankunftszeit
  - Determinismus des Medienzugangs

Echtzeitklassen nach IAONA

Echtzeit-Klasse	1	2	3	4
Anforderung	gering	mittel	hoch	extrem hoch
typische Ebene	Leit-ebene	Führungs-ebene	Steuerungs-ebene	Sensor/ Aktor-Ebene
Nutzdaten max.	500 KB	500 B	32(-200) B	20 B
Verzögerung max.	5 s	500 ms	5 ms	0,5 ms
Jitter max.	$>$ 1 ms	0,1 - 3 ms	10 - 400 μs	0,5 - 15 μs

➥ IAONA: Industrial Automation Open Networking Alliance

10.2 Typische Merkmale von Feldbussen

- Busstruktur
  - geringer Verkabelungsaufwand bei hoher Teilnehmerzahl
- Deterministische Medienzugangskontrolle:
  - meist zeitgesteuert
    - zentrale Zuteilung (Master/Slave)
    - dezentrale Zuteilung (Token-Bus bzw. -Ring, TDMA)
  - manchmal prioritätsgesteuert
    - (vollständige) Kollisionsvermeidung durch CSMA/CA
- Kurze Frames (**Telegramme**), oft nur wenige (8) Bytes
- Sehr zuverlässige Fehlererkennung

10.2 Typische Merkmale von Feldbussen ...

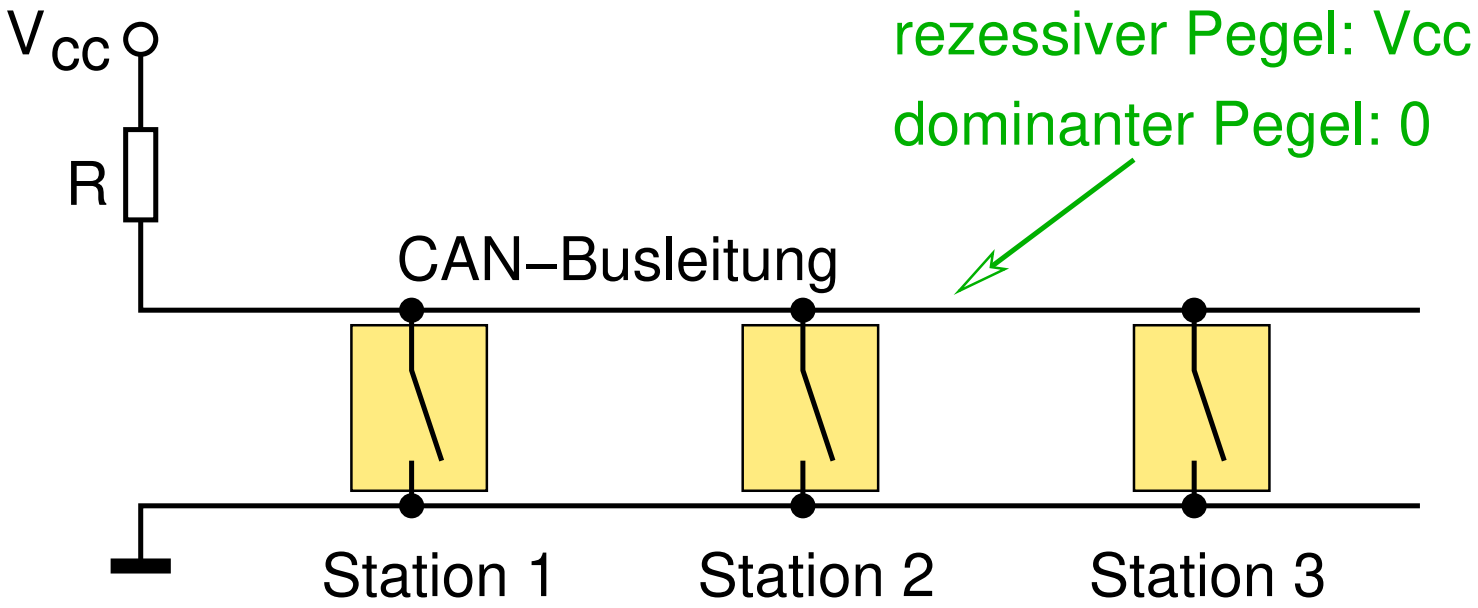
- Oft verschiedene Bandbreiten, abhängig von Leitungslänge
- Meist geschirmte Zweidrahtleitung, teilw. auch Lichtleiter
- Bei Sensor/Aktor-Bussen oft Zweidrahtleitung incl. Spannungsversorgung
- Meist nur OSI-Schichten 1, 2 und 7 implementiert
- Anwendungsprotokolle realisieren häufig Zugriff auf Objektverzeichnisse
  - Prozeßobjekte, z.B. Temperaturwerte, Schaltzustände, ...

Übersicht

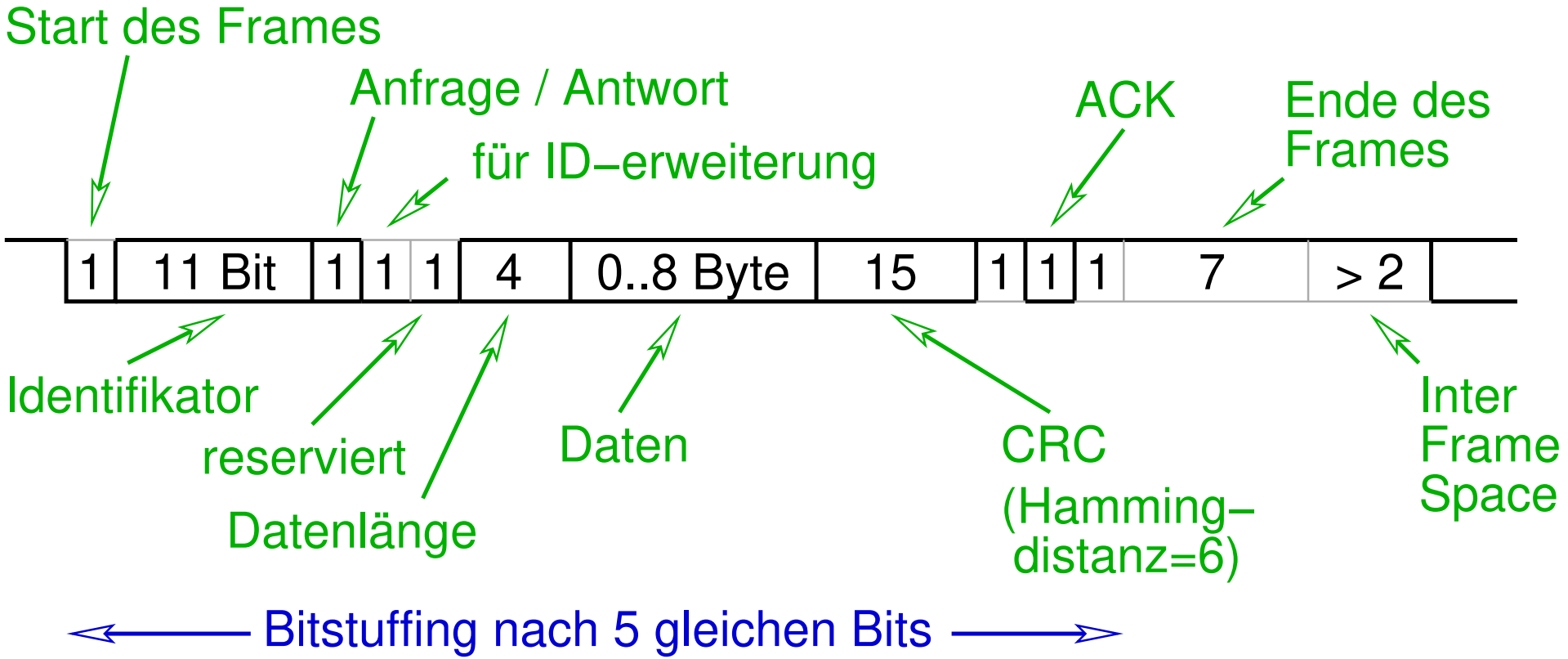
- CAN: *Controller Area Network*
- Ursprünglich für Vernetzung in Fahrzeugen entwickelt, inzwischen auch im industriellen Bereich
- Adressierung erfolgt nachrichtenorientiert
  - Nachrichtentyp statt Sender-/Empfänger-Adresse
  - Geräte reagieren auf bestimmte Nachrichtentypen
- Bustopologie, keine Begrenzung der Teilnehmerzahl
- Medium: i.d.R. verdrillte Zweidrahtleitung
- Übertragungsraten:
  - 20 kb/s (max. 1000 m) bis 1 Mb/s (max. 40 m)
  - jedes Bit füllt die Leitung vollständig aus!

Bitübertragung: rezessive und dominante Pegel

- Falls zwei Stationen gleichzeitig unterschiedliche Pegel an den Bus legen:
  - dominanter Pegel (0-Bit) setzt sich durch
- Prinzip-Bild:



Frameformat

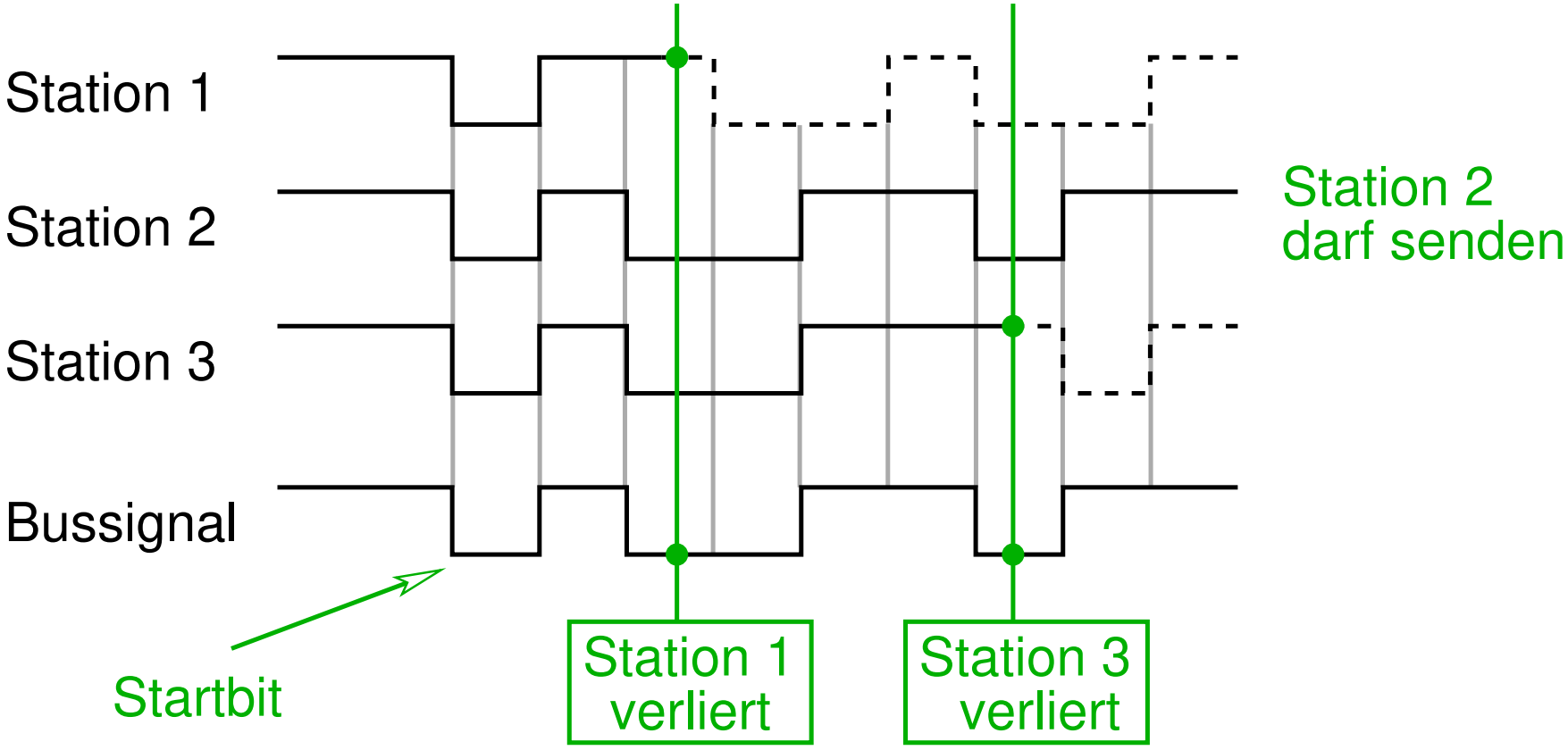


- ACK-Bit wird mit rezessivem Pegel gesendet
- Empfänger setzt (während d. Übertragung) dominanten Pegel

Medienzugriffsverfahren

- Prioritätengesteuerte Arbitrierung
  - CSMA/CA, vollständige Vermeidung von Kollisionen
- Sender erkennen, wenn Bus unbenutzt ist
  - Bus liegt mehr als 5 Takte auf rezessivem Pegel
- Arbitrierung durch Mithören beim Senden des Identifikators und des Anfrage/Antwort-Bits
  - bitweiser Vergleich von gesendeten Daten und Buspegel
  - bei Abweichung: Senden sofort einstellen
  - Identifikator mit der ersten 0 gewinnt Arbitrierung
    - d.h. kleinster Identifikator hat höchste Priorität
    - bzw. bei gleichem Identifikator: Antwort hat Priorität

Beispiel für die Busarbitrierung



10.4 Echtzeit-Ethernet

- Ziel: durchgehende Verwendung von Ethernet in allen Ebenen
  - d.h. Koexistenz von Echtzeit und Nicht-Echtzeit-Verkehr
- Definition verschiedener Erweiterungen zur Echtzeitfähigkeit
  - Basis: *switched* Ethernet, vollduplex ohne CSMA/CD
- Problem: zeitliche Verzögerung von Frames in den Switches
- Lösungsansätze:
  - zeitgesteuerte Verfahren mit globalem Schedule
    - für periodische, zeitkritische Frames
  - prioritätsgesteuerte Verfahren (Tags nach 802.1Q)
- Beispiele:
  - *Time Triggered Ethernet*
  - *Time Sensitive Networking*

10.4.1 *Time Triggered Ethernet*

- Standardisiert durch Society of Automotive Engineers (SAE)
- Realisiert durch zuzätzliche Schicht zwischen MAC und LLC
  - zeitliche Steuerung, incl. Uhrensynchronisation
  - Fehlertoleranz (durch Nutzung redundanter Wege)
- Drei Kommunikationsdienste:
  - *Time-triggered* Frames: periodisch
    - Knoten und Switches haben globalen TDMA-Schedule
  - *Rate-constrained* Frames: nicht periodisch, mit Prioritäten
    - Realisierung durch priorisierte Warteschlangen
    - Switches kennen minimale Zeit zwischen zwei Frames
  - *Best-effort* Frames: normale Ethernet-Frames

10.4.1 *Time Triggered Ethernet* ...

- Echtzeit-Dienste sind verbindungsorientiert (*one-to-many*)
  - Adressierung über Multicast MAC-Adressen
  - festes 32-Bit Feld + 16 Bit *Virtual Link Identifier*
- Mechanismen zur Konfliktauflösung bei der Weiterleitung:
  - *Shuffling*: keine Behandlung
  - *Timely Block*:
    - Einführung eines *Guarding Windows* vor TDMA-Slot
    - Länge: Zeit für maximal langen Frame
  - *Preemption*: Abbruch des übertragenen Frames
    - Problem: nicht von Übertragungsfehler unterscheidbar

10.4.2 *Time Sensitive Networking*

- Von der IEEE standardisierte Ergänzungen zu Ethernet (802.1)
- Basis: Prioritäten nach 802.1Q
  - für Prioritäten können unterschiedliche TDMA-Zeitfenster vorgesehen werden
  - damit ratenbeschränkte und synchrone Frames möglich
- Erweiterungen u.a. für:
  - Zeitsynchronisation (802.1AS)
  - Zeitgesteuertes Scheduling (802.1Qbv)
    - ermöglicht u.a. *time-triggered* Frames
  - *Frame Preemption* (802.1Qbu)
  - Frame Replikation zur Fehlertoleranz (802.1CB)
  - Wegewahl und Reservierung (802.1Qca)

10.5 Zusammenfassung / Wiederholung

- Hierarchieebenen in der Automatisierung:
  - unterschiedliche Anforderungen an Netze
- Feldbusse:
  - Leitrechner, Steuerungen, Sensoren/Aktoren
  - Zuverlässigkeit, Kosten, Echtzeitanforderungen
- Merkmale und Dienste der Netze werden durch Anwendungen und Einsatzumgebung bestimmt
- Verkabelungskosten $\Rightarrow$ meist Busstruktur
- Umgebung $\Rightarrow$ zuverlässige Fehlererkennung notwendig
- Anwendungen $\Rightarrow$ Schichten 3-6 fehlen
  - Schicht 7: Objektverzeichnisse

10.5 Zusammenfassung / Wiederholung ...

- Anwendungen $\Rightarrow$ Echtzeitfähigkeit
  - Übertragungsdauer und Jitter berenzt
  - typisch: zyklische Übertragung mit fester Zykluszeit
- Medienzugriffssteuerung (MAC):
  - deterministisch
    - Zeitmultiplex (TDMA), Master/Slave, Token-Ring
  - nichtdeterministisch
    - z.B. CSMA/CA mit Prioritäten (CAN)
      - kleinste ID gewinnt
      - Voraussetzung: Bitzeit $>$ RTT

Rechnernetze II

SoSe 2025

11 Drahtlose Sensornetze

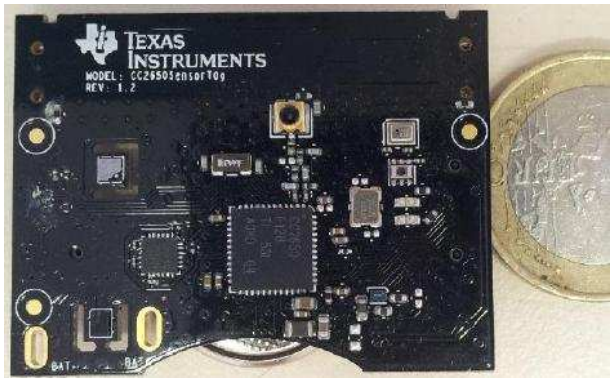
Inhalt

- Einführung
- MAC-Protokolle
- Routing

- Holger Karl, Andreas Willig: *Protocols and Architectures for Wireless Sensor Networks*, Wiley, 2005
- Koen Langendoen:
  Medium Access Control in Wireless Sensor Networks
- Pei Huang *et al*: *The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey*, IEEE Communications Surveys & Tutorials, Vol. 15, No. 1, 2013, S. 101-120

Drahtlose Sensornetze (WSN)

- Ziel: Überwachung ausgedehnter Gebiete / Strukturen
  - Umweltmonitoring, Landwirtschaft, intelligente Gebäude, Strukturüberwachung von Bauwerken, Patienten-Monitoring, Industrie, Logistik, ...
- Batteriebetriebene Sensorknoten mit Sensorik, CPU, Radio
- Vernetzung ohne Infrastruktur
  - ad-hoc Netzwerk, selbstorganisierend
- Typische Eigenschaften von WSNs:
  - viele Knoten, hohe Knotendichte
  - beschränkte Ressourcen (Energie, CPU-Leistung, Reichweite)
  - Dynamik (Ausfälle, mobile Knoten)



Kommunikation in WSNs

- Eigenschaften der Funkgeräte (Radios)
  - geringe Sendeleistung $\Rightarrow$ i.A. multi-hop Kommunikation
  - Stromverbrauch für Senden und Empfang etwa gleich
    - zur Energieeinsparung: Radio abschalten
    - Umschalten zwischen den Modi kostet Energie
- Kommunikationsstruktur
  - häufig eine Senke (mit Anbindung an LAN/WAN)
  - Kommunikationsformen: *flooding*, *convergecast*, *local gossip*
    - periodisch oder ereignisgetrieben
    - bei *convergecast*: Datenaggregation ist wichtig
  - Adressierung über Ort / Eigenschaften der Knoten
    - *data centric network* / *data centric routing*

Zielsetzung

- Funkgeräte sollen möglichst oft ausgeschaltet sein
- Gründe für überflüssigen Energieverbrauch:
  - ***Idle listening***: Funkgerät ist eingeschaltet, obwohl niemand eine Nachricht sendet
  - ***Overhearing***: Knoten empfängt eine Nachricht, die er nicht weitergeben / verarbeiten muß
  - ***Kollisionen***: insbes. durch *Hidden Station* Problem
    - RTS/CTS bei kleinen Datenmengen nicht sinnvoll
  - ***Verkehrs-Fluktuationen***: zeitlich und räumlich
    - führt ggf. zu *Overprovisioning*
  - ***Protokoll-Overhead***: MAC-Header, Steuernachrichten
    - daher ausgeklügelte Protokolle oft nicht verwendbar

Allgemeine Ansätze für MAC-Protokolle

- *Random Access* (*Contention based*)
  - Knoten können jederzeit versuchen, das Medium zu nutzen
  - verschiedene Varianten von CSMA
- *Fixed Assignment*
  - den Knoten werden statisch exklusive Ressourcen zugeteilt:
    - Zeitschlitze (*Time Division Multiple Access*, TDMA)
    - Frequenzen (*Frequency Division Multiple Access*, FDMA)
    - Codes (*Code Division Multiple Access*, CDMA)
    - Raumgebiet (*Space Division Multiple Access*, SDMA)
- *Demand Assignment*
  - Zuteilung der Ressourcen (Zeit, Frequenz) erfolgt dynamisch
    - z.B. *Token Ring*

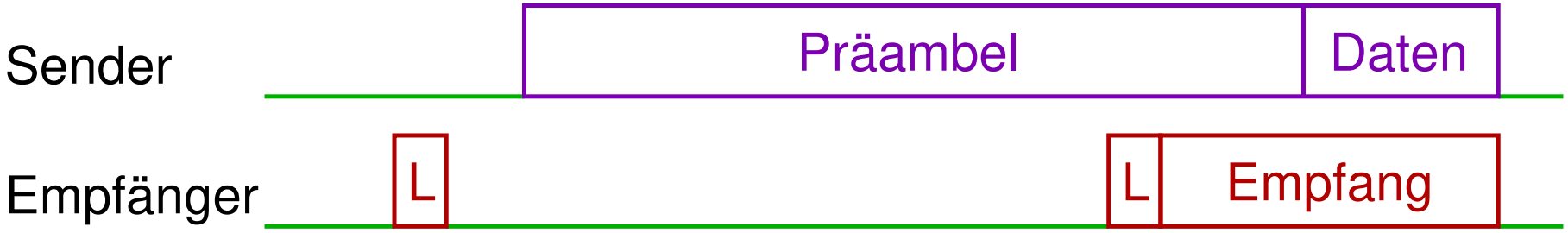
Grundprinzip energiesparender MAC-Protokolle

- Funkgeräte werden periodisch für kurze Zeit eingeschaltet, um nach eintreffenden Nachrichten zu lauschen (*Duty Cycle*)

- Problem: Senden nur möglich, wenn Empfänger aktiv ist
- Lösungsansätze:
  - asynchrone Protokolle: Sender hat kein a-priori Wissen, wann Empfänger aktiv ist
  - synchrone Protokolle: Sender weiß, wann Empfänger aktiv ist
  - Frame-basierte Protokolle: *Listen*-Periode wird zur Kollisionsvermeidung in Zeitschlitze unterteilt

Asynchrone Protokolle: B-MAC

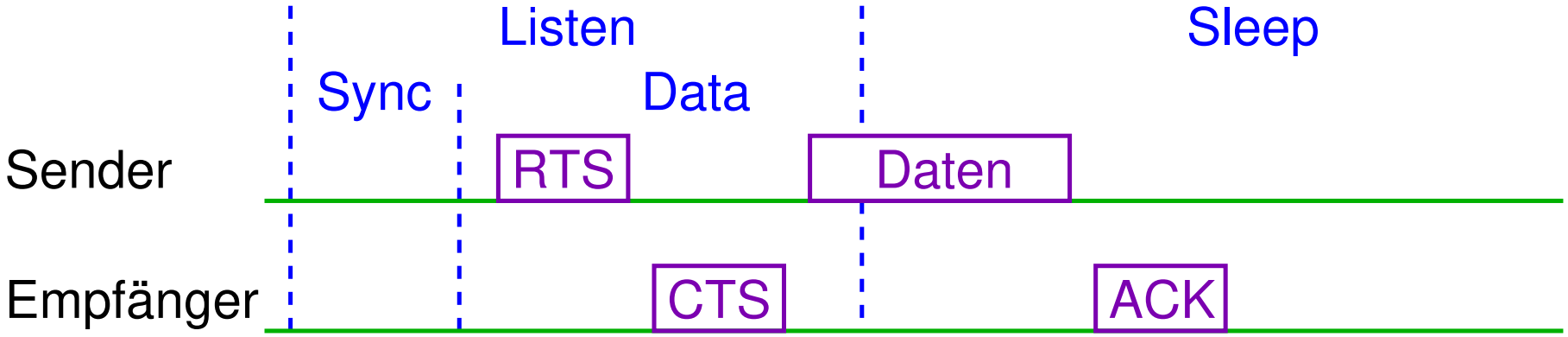
- Idee: Sender sendet vor dem Paket eine lange Präambel
  - wenn Empfänger Präambel hört, bleibt er wach



- *Low Power Listening*: Beim Abhören des Mediums (auch für CSMA) wird nur Signalstärke ausgewertet
- Optimierungen:
  - Folge kurzer Präambeln mit Zieladresse: kein *Overhearing*
  - zusätzlich: Zeitdauer bis zur Datenübertragung in Präambel
    - kein *Idle Listening*
- Nachteil: Sender verbraucht viel Energie

Synchrone Protokolle: S-MAC

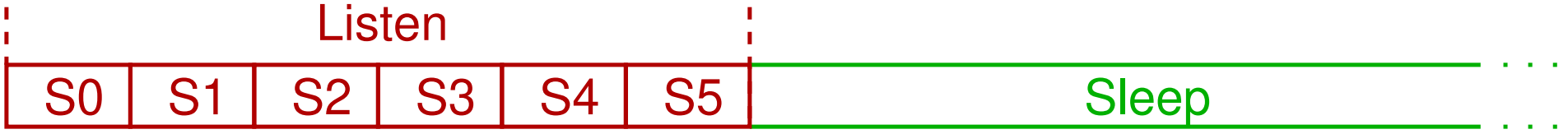
- Idee: Knoten synchronisieren ihre *Listen*- und *Sleep*-Zeiten
  - nicht global, sondern in räumlichen Clustern
  - Knoten kann ggf. in zwei Clustern sein
- Zusätzlich RTS/CTS zur Kollisionsvermeidung



- Variante: T-MAC
  - adaptive *Listen*-Periode, wird bei Aktivität verlängert
- Nachteil: Sychronisationsaufwand, nur wenige Hops pro Periode

Frame-basierte Protokolle: L-MAC

- Idee: Kollisionsvermeidung durch Einführung von Zeitschlitzen



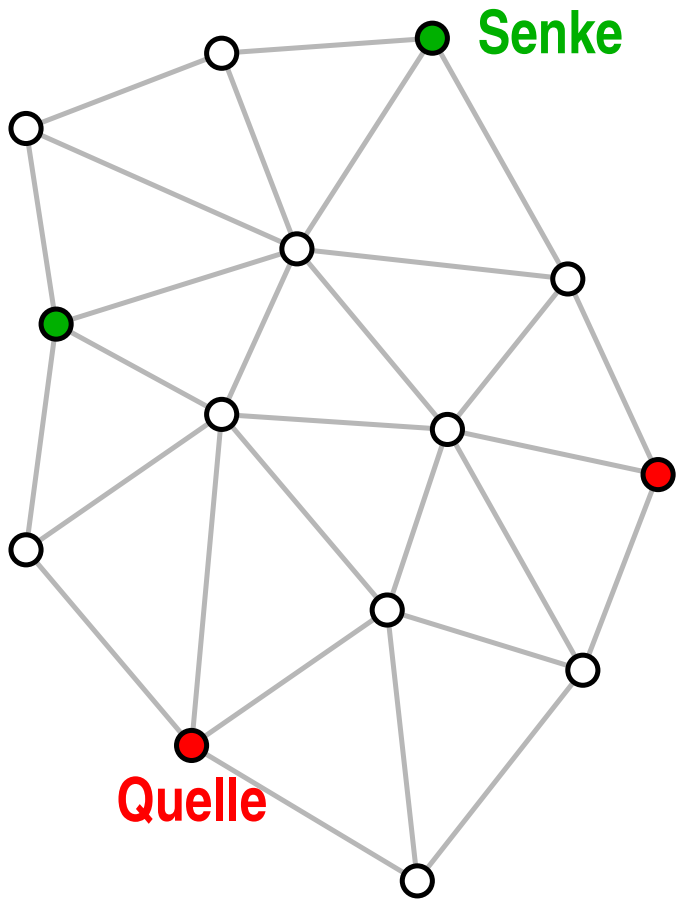
- Knoten senden nur in „ihrem“ Zeitschlitz
  - Header (in jeder Periode, zur Synchronisation)
    - enthält Bitmaske der durch Nachbarn belegten Zeitschlitze
  - ggf. gefolgt von Nutzdaten
- Ermittlung freier Zeitschlitze:
  - Oder-Verknüpfung aller empfangener Bitmasken
  - wähle freien Zeitschlitz zufällig, Wiederholung bei Kollision
- Nachteil: Overhead im Header, begrenzte Zahl an Nachbarn

Einige Aspekte

- Kommunikationsform / Routing-Schema:
  - *unicast*: Punkt-zu-Punkt
  - *broadcast* / *convergecast*: Baumstruktur
  - *geocast*: geographisches Routing
    - Adressierung über den Ort, Ortsinformation zum Routing
- Energieeffizienz
  - Minimierung der Energie pro Paket (bzw. Bit)
  - Maximierung der Lebensdauer des Netzes
  - Verbleibende Restenergie in den Batterien
  - Abwägung zu andern Metriken (Verzögerung, Zuverlässigkeit)
- *Multipath* Routing: Erhöhung der Zuverlässigkeit
- Mobile Knoten

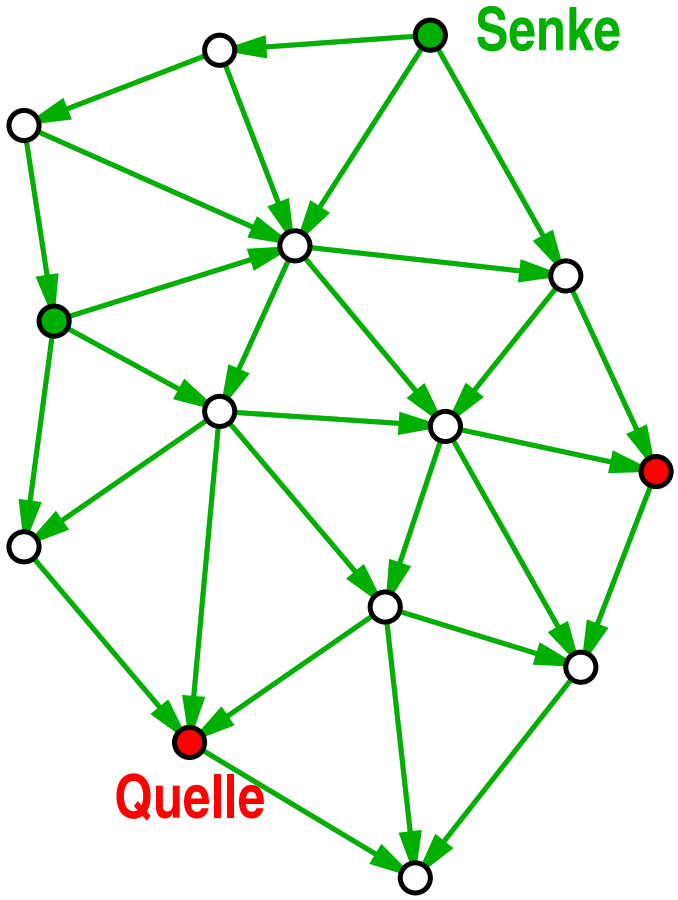
Beispiel: Datenzentrisches Routing mit *Directed Diffusion*

- Senken fordern periodische Informationen von Quellen an
- Vier Schritte:



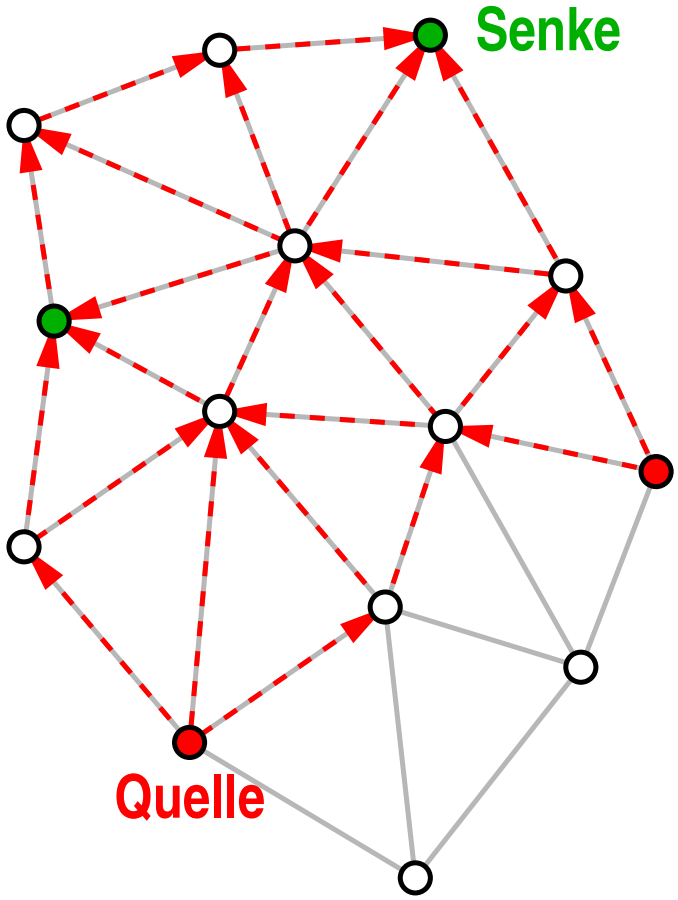
Beispiel: Datenzentrisches Routing mit *Directed Diffusion*

- Senken fordern periodische Informationen von Quellen an
- Vier Schritte:
  - *Interest propagation*
    - Anforderung an alle Knoten verteilen (z.B Flooding)



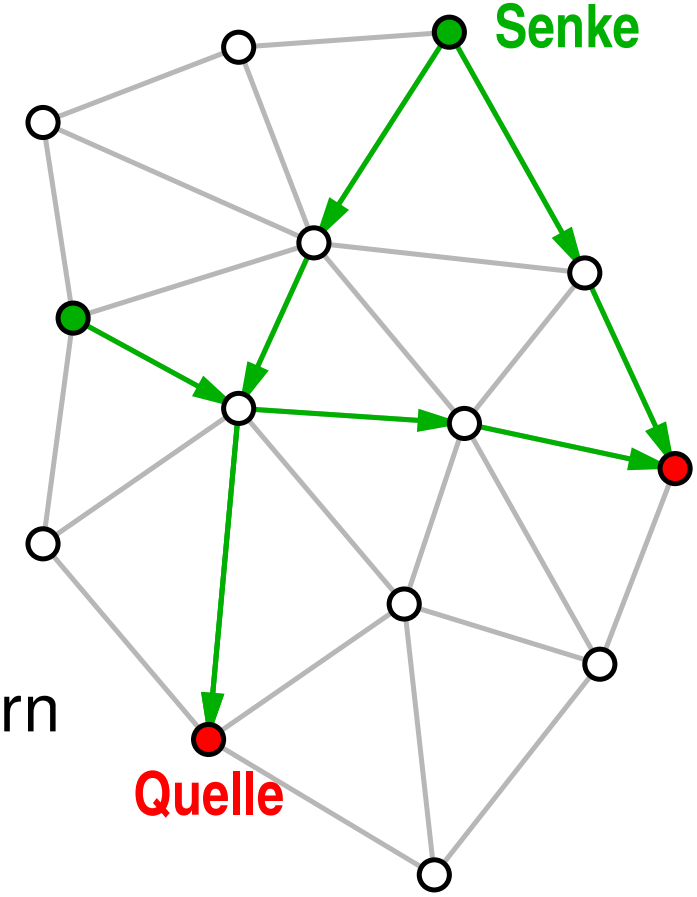
Beispiel: Datenzentrisches Routing mit *Directed Diffusion*

- Senken fordern periodische Informationen von Quellen an
- Vier Schritte:
  - *Interest propagation*
    - Anforderung an alle Knoten verteilen (z.B Flooding)
  - *Gradient setup / Exploratory data*
    - Daten mit geringer Rate entlang aller Pfade



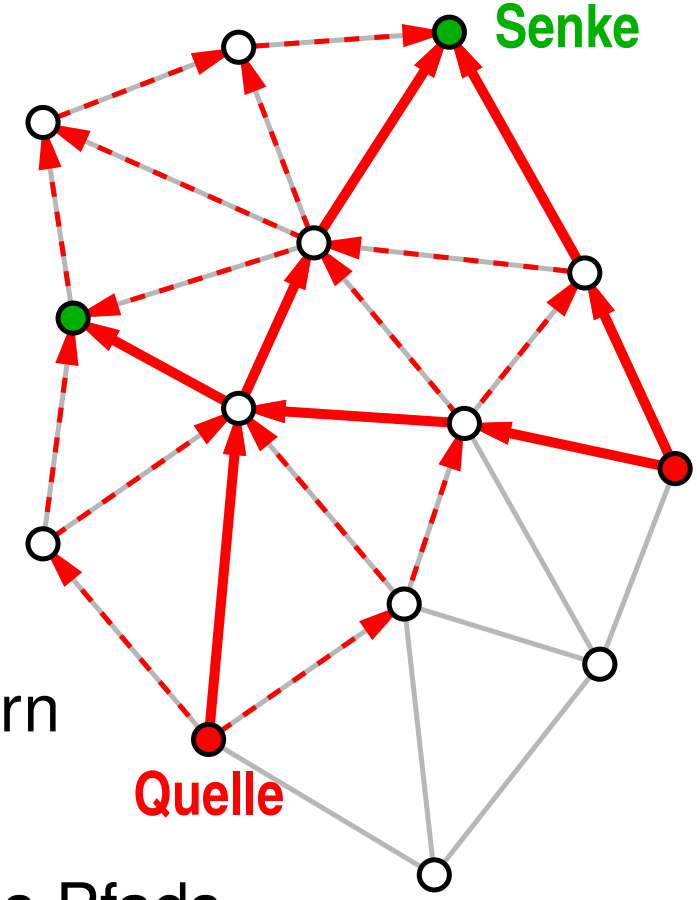
Beispiel: Datenzentrisches Routing mit *Directed Diffusion*

- Senken fordern periodische Informationen von Quellen an
- Vier Schritte:
  - *Interest propagation*
    - Anforderung an alle Knoten verteilen (z.B Flooding)
  - *Gradient setup / Exploratory data*
    - Daten mit geringer Rate entlang aller Pfade
  - *Reinforcement*
    - Empfänger wählt beste(n) Nachbarn



Beispiel: Datenzentrisches Routing mit *Directed Diffusion*

- Senken fordern periodische Informationen von Quellen an
- Vier Schritte:
  - *Interest propagation*
    - Anforderung an alle Knoten verteilen (z.B Flooding)
  - *Gradient setup / Exploratory data*
    - Daten mit geringer Rate entlang aller Pfade
  - *Reinforcement*
    - Empfänger wählt beste(n) Nachbarn
  - *Data delivery*
    - Daten mit hoher Rate entlang eines Pfads



11.4 Zusammenfassung

- Sensorknoten mit beschränkter Energie und Rechenleistung
- Selbstorganisierende Vernetzung
- Typisch: Kommunikation in Baumstruktur
- Adressierung über Ort bzw. Eigenschaften der Knoten
- Energiesparende MAC-Protokolle
  - Vermeidung von *Idle listening*, *Overhearing* und Kollisionen
  - Randbedingungen: Verkehrs-Fluktuationen, Protokoll-Overhead
  - Grundidee: *Low Duty Cycle*
- Routing
  - Berücksichtigung der Energie bei der Routenauswahl
  - *Data centric routing*

Rechnernetze II

SoSe 2025

12 Zusammenfassung, wichtige Themen

1. *Wide Area Networks* (WANs)

- Zusammenhang Übertragungsrate / Bandbreite
  - Fourier-Analyse
  - **Nyquist-Theorem** (Abtasttheorem)
  - **Shannon'sches Theorem** $B = H \cdot \log_2(1 + S/N)$
- Telefonnetz
  - synchone Netze (8000 Abtastungen/s), Multiplexing
- Modulationsverfahren mit mehreren Bits pro Baud
  - Änderung der Phase und Amplitude
- PPP: Zweck, Aufgaben
- *Frame Relay*, ATM: virtuelle Leitungsvermittlung
- ADSL: *Discrete MultiTone*, mehrere Frequenzkanäle

2. Schnelles Ethernet

- **Techniken zur Bandbreitenerhöhung**
  - 8B6T, 4B5B, 8B10B etc. statt Manchester-Codierung
  - mehrere Adernpaare
- Switches und **Vollduplexbetrieb** (keine Kollisionen)
  - keine max. Leitungslänge / min. Paketgröße

3. Drahtlose Netze

- **Spreizbandtechniken** (Motivation, FHSS, OFDM, DSSS)
- **MAC im WLAN**: DCF, PCF
  - DCF: kein CSMA/CD möglich, **CSMA/CA, MACAW**
    - **Funktionsweise, *Hidden*/*Exposed Station*-Problem**
  - **PCF**: Zeitmultiplex (TDMA)
- **WLAN-Sicherheit**
  - Funktionsweise und Schwächen von WEP
  - Verbesserungen durch WPA und WPA2
- **Bluetooth**: Funkschicht (FHSS), MAC (TDMA), Übertragungssicherung, Sicherheit

4. IP-Routing: Spezielle Aspekte

- **Multicast**
  - Multicast-Adressen
  - **IGMP**: Gruppenmanagement
  - Routing: (gemeinsame oder quellenspezifische) spannende Bäume
    - *Link-State-Routing*: Gesamtnetz bekannt
    - Distanzvektor-Routing: ***Reverse Path Broadcast*** / ***Multicast*** (*Flooding* mit *Pruning*)
    - **PIM**: explizite *Join*-Nachrichten bauen Multicast-Baum auf

4. IP-Routing: Spezielle Aspekte ...

- Mobile IP
  - Weiterleitung durch Heimatagen über IP-Tunnel
- **MPLS: virtuelle Leitungsvermittlung**
  - *Label-Edge-Router* fügt Label an Paket an,
    restliche Router betreiben Label-basierte Weiterleitung

5. VPN, IP-Tunnel und IPsec

- **Secure IP**
  - Vor-/Nachteile der Sicherheit auf Vermittlungsschicht
  - **AH und ESP-Protokoll, Transport- und Tunnel-Modus**
    - was wird jeweils verschlüsselt, authentifiziert?
  - ***Security Association***, Konfiguration

6. Überlastkontrolle und Ressourcenzuteilung

- **Überlastkontrolle**
  - **Basismechanismus „*Additive Increase / Multiplicative Decrease*“**
  - Erweiterungen „*Slow Start*“, „*Fast Retransmit / Fast Recovery*“

6. Überlastkontrolle und Ressourcenzuteilung ...

- **Überlastvermeidung**
  - **Router-zentrisch: mittlere Warteschlangenlänge**
    - DECbit: Warnbit im Header
    - **RED: Verwerfen einiger zufälliger Pakete**
  - **Host-zentrisch: Latenz, Durchsatz**
    - TCP Vegas: Vergleich erwarteter und erreichter Durchsatz
      - Ziel: belege konstante, kleine Zahl von Puffern im Router
- *Quality of Service*
  - Dienstklassen mit verschiedenen Garantien (Bandbreite, Latenz, Jitter)

6. Überlastkontrolle und Ressourcenzuteilung ...

- ***Quality of Service***
  - **feingranular**: Datenflüsse einzeln betrachten
    - Mechanismen: *Flow Specs* (geforderter Dienst, Datenfluß), Zugangskontrolle, Ressourcen-Reservierung, Paket-Scheduling
    - Beispiel *IntServ*: GS, CLS
    - *Token-Bucket*-Filter, RSVP, WFQ
  - **grobgranular**: nur Klassen von Datenflüssen betrachten
    - Paketklassifizierung an der Peripherie, fest reservierte Ressourcen
    - Beispiel *DiffServ*: EF, AF (4 Klassen á 3 Prioritäten)
  - Skalierbarkeit vs. echte Garantien

7. Anwendungsprotokolle

- Netzwerkmanagement
  - Aufgaben
  - SNMP: Lesen/Schreiben von Objekten, MIB: Aufbau, Inhalt
- **Multimedia-Anwendungen**
  - RTP/RTCP: Synchronisation (Zeitstempel), Multiplexing, Sequenznummern, Sender-*Feedback*

8. Netzwerkprogrammierung

- Sockets: Netzunabhängiges API für Kommunikation
  - *Stream* und *Datagram Sockets*
  - `bind()`, `connect()`, `listen()`, `accept()`
- Server-Design: Prozeß / Thread pro Client, `select`

9. Netze für Cluster und Hochleistungsrechner

- Bisektionsbandbreite, Verbindungsgrad, Durchmesser
- Netztopologien, *Crossbar*, Clos-Netz (Idee)
- *Virtual-Cut-Through*-Routing, *Remote* DMA, *OS Bypass*

10. Netze für Automatisierungssysteme

- Anforderungen, Echtzeit, Merkmale von Feldbussen
- MAC für Echtzeit-Aufgaben: *Token Passing*, *Master-Slave*, CSMA/CA (prioritätengesteuerte Arbitierung)

11. Drahtlose Sensornetze

- Spezielle Anforderungen: Enegieeffizienz, *data centric*