
Datenkommunikation

Wintersemester 2012/2013

Prof. Dr. Peter Mandl

Überblick

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Datenkommunikation

Grundlagen
von Rechnernetzen, Teil 1

Wintersemester 2012/2013

Einordnung

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

1. Referenzmodelle und Terminologie

- ISO/OSI-Referenzmodell
- TCP/IP-Referenzmodell
- Transportsystem
- Netzwerktopologien
- Internetworking

2. Bitübertragungsschicht

- Aufgaben, Begriffe und Definitionen
- Kodierung (Leitungskodierung)
- Laufzeit, Übertragungszeit, Transferzeit, Jitter
- Digitale Übertragung und PCM, Multiplexverfahren
- Datenübertragungsmedien und Verkabelung

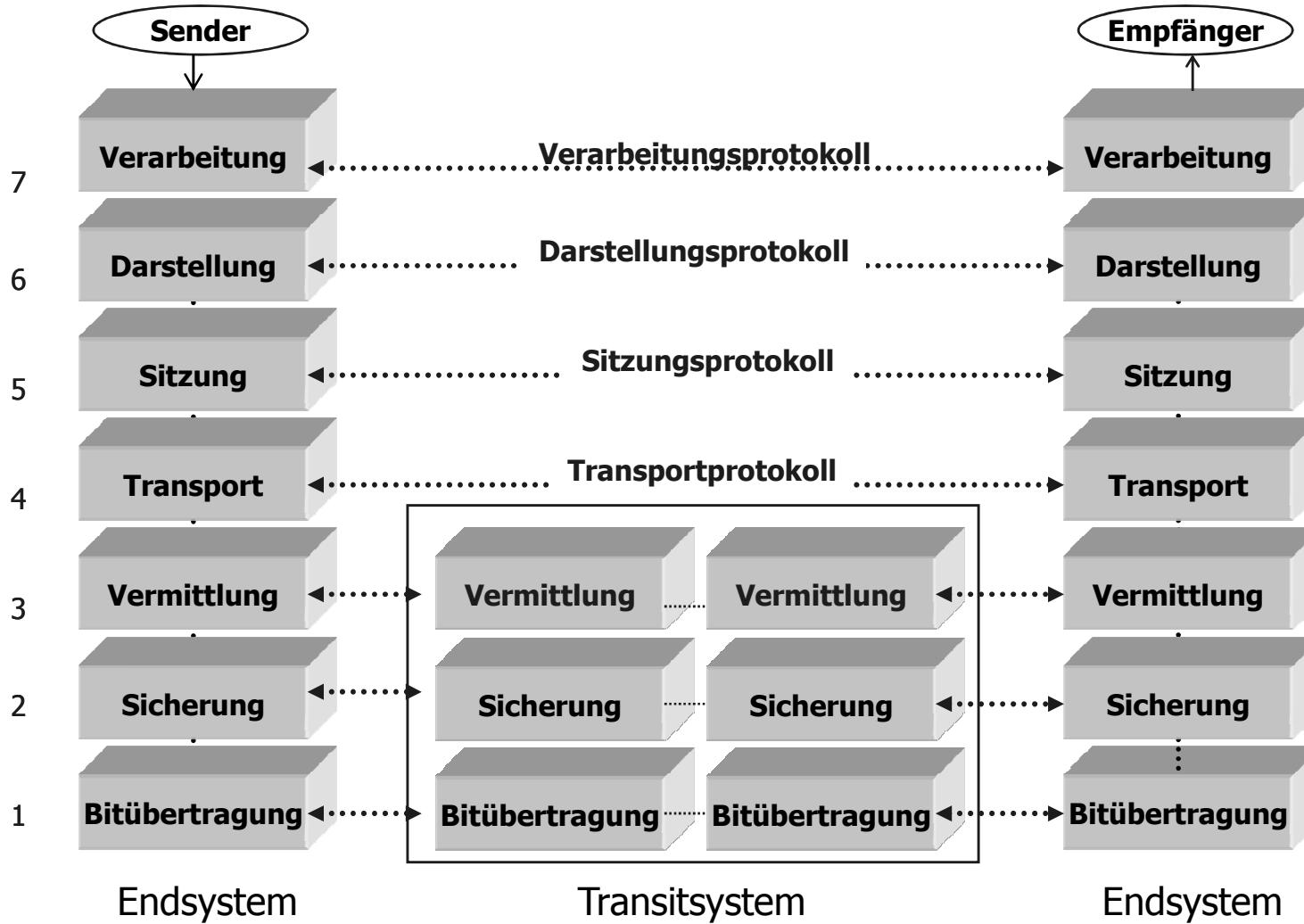
ISO/OSI-Referenzmodell

- Kommunikation zw. Rechnern in offenen, heterogenen Systemen wird beschrieben durch Referenzmodelle
- Vorteil: Offene allgemein verbindliche Vorstellung eines Kommunikationsvorgangs in Form eines Architekturmodells
- Beispiele:
 - ISO/OSI-Referenzmodell
 - OSI = Open System Interconnection
 - ISO = International Standardization Organization
 - TCP/IP-Referenzmodell
 - SNA-Modell (IBM)
 - TRANSDATA (Siemens)

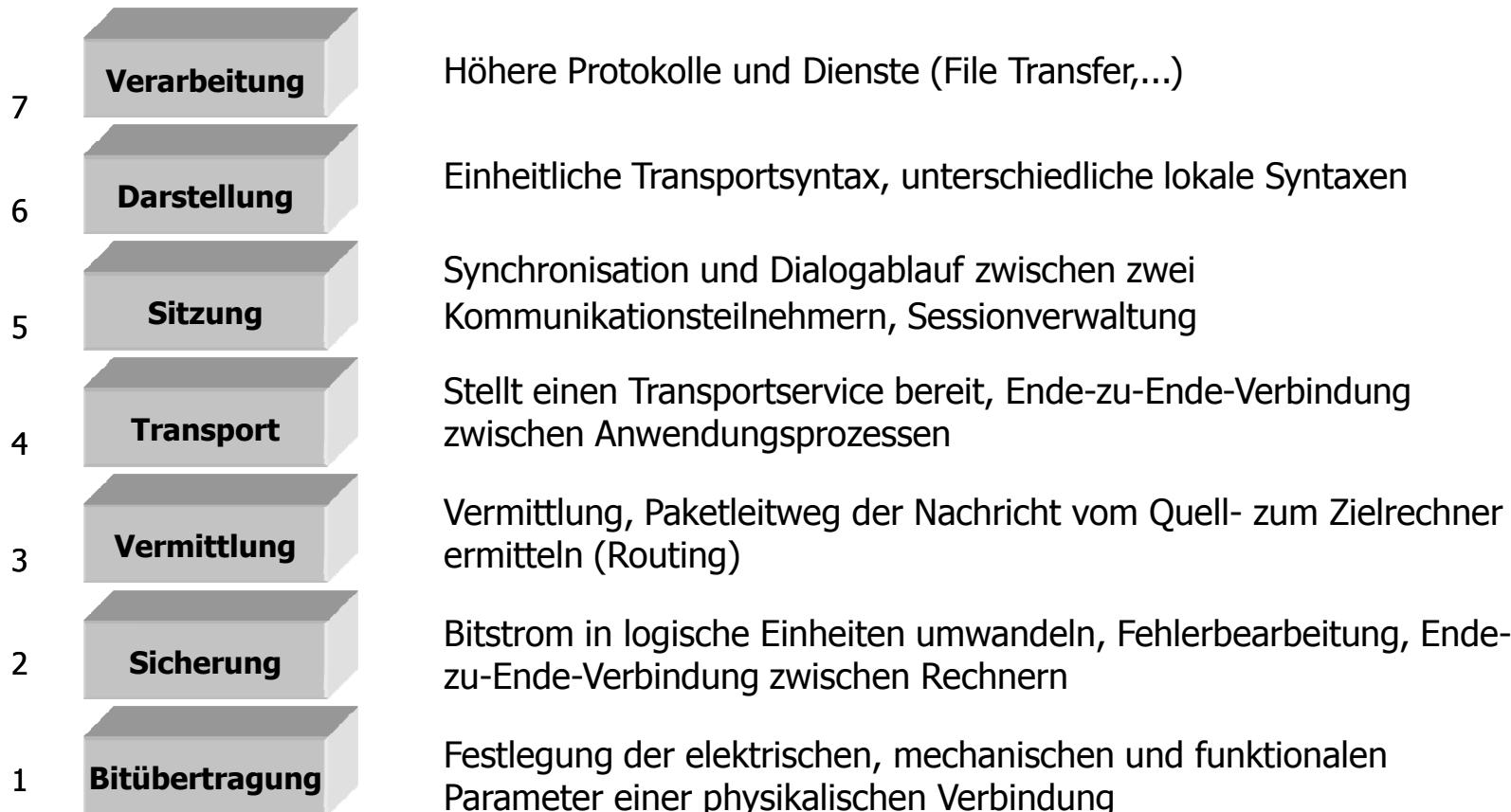
ISO/OSI-Referenzmodell: Schichtung

- Um ein Kommunikationsprotokoll überschaubar zu machen, zerlegt man es in **Schichten** (layer)
 - Beim ISO/OSI-Modell sind es 7 Schichten
- Das **Endsystem** umfasst alle sieben Schichten
- Das **Transitsystem** umfasst nur die unteren Schichten, z.B. die Schichten 1–3
 - unterschiedliche Teilstrecken miteinander verbinden
 - evtl. verschiedene Übertragungsmedien
 - evtl. verschiedene Netze

ISO/OSI-Referenzmodell: Schichtung



ISO/OSI-Referenzmodell: Aufgaben der Schichtung



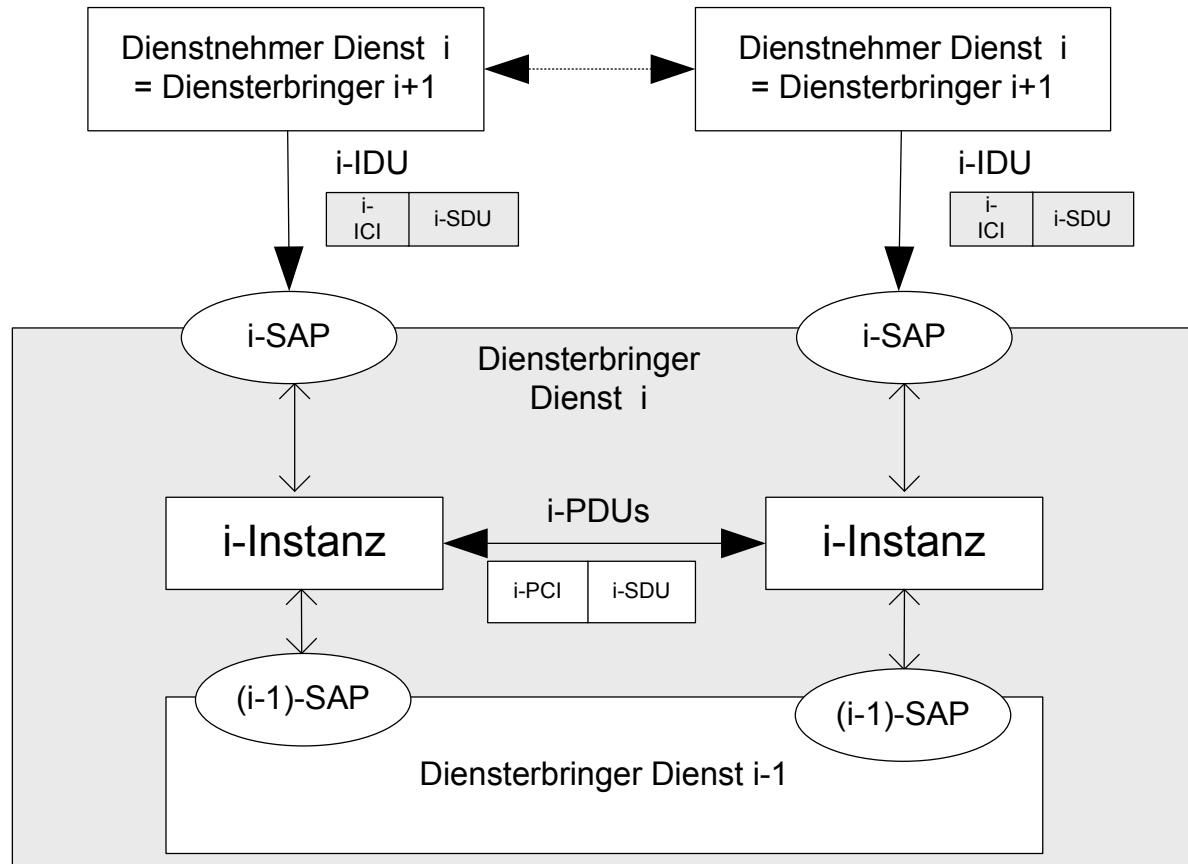
Protokolle und Dienste allgemein

- Protokolle sind **Verhaltensrichtlinien**, auf deren Grundlage sich Computersysteme untereinander „unterhalten“ und gegenseitig verstehen:
 - Bestimmte Spielregeln, an die sich Sender und Empfänger halten müssen,
 - Notwendig, damit die Übertragungswünsche der Netzteilnehmer nicht im Chaos enden
- Protokolle sind **Vorschriften und Konventionen** zur Regelung von
 - Verbindungsaufbau
 - Nachrichtenübermittlung und
 - Verbindungsabbau

Protokolle und Dienste allgemein

- Jede Schicht bietet der ihr jeweils übergeordneten Schicht Funktionen, sog. **Dienste** an
- Jede Schicht (bis auf die unterste) kann von der direkt darunter liegenden Schicht Dienste in Anspruch nehmen, **ohne ihre Implementierung zu kennen**
- Protokolle übernehmen verschiedene Aufgaben (je nach Schicht)
 - Verbindung aufbauen und Verbindung abbauen
 - Datenübertragung
 - Fehlererkennung und Fehlerbehebung
 - Staukontrolle (Congestion Control)
 - Flusskontrolle
 - ...

ISO/OSI-Referenzmodell: Hierarchische Dienststruktur

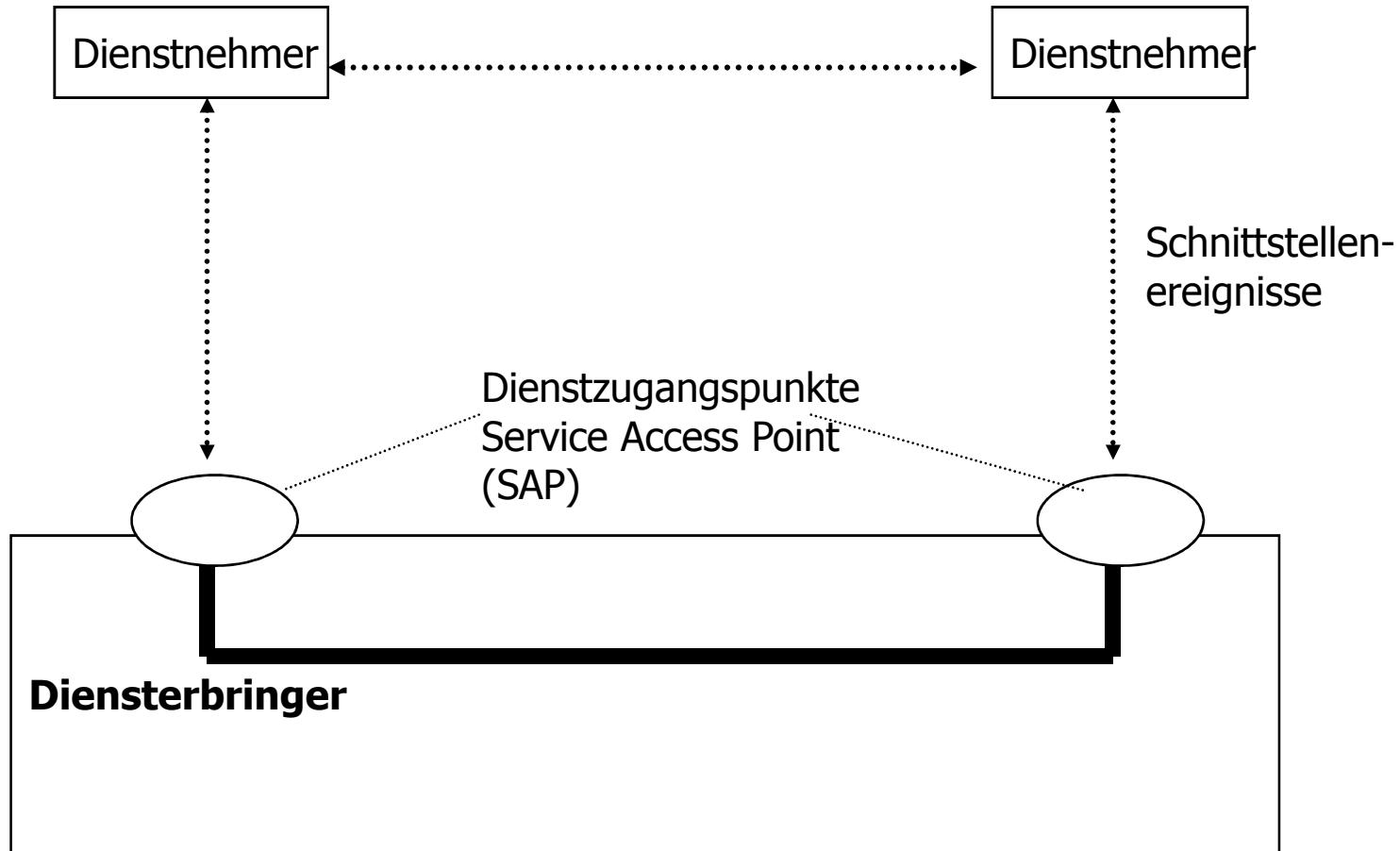


- Begriffe: IDU, SDU, SAP, ICI, SDU, PDU

ISO/OSI-Referenzmodell: Dienststruktur

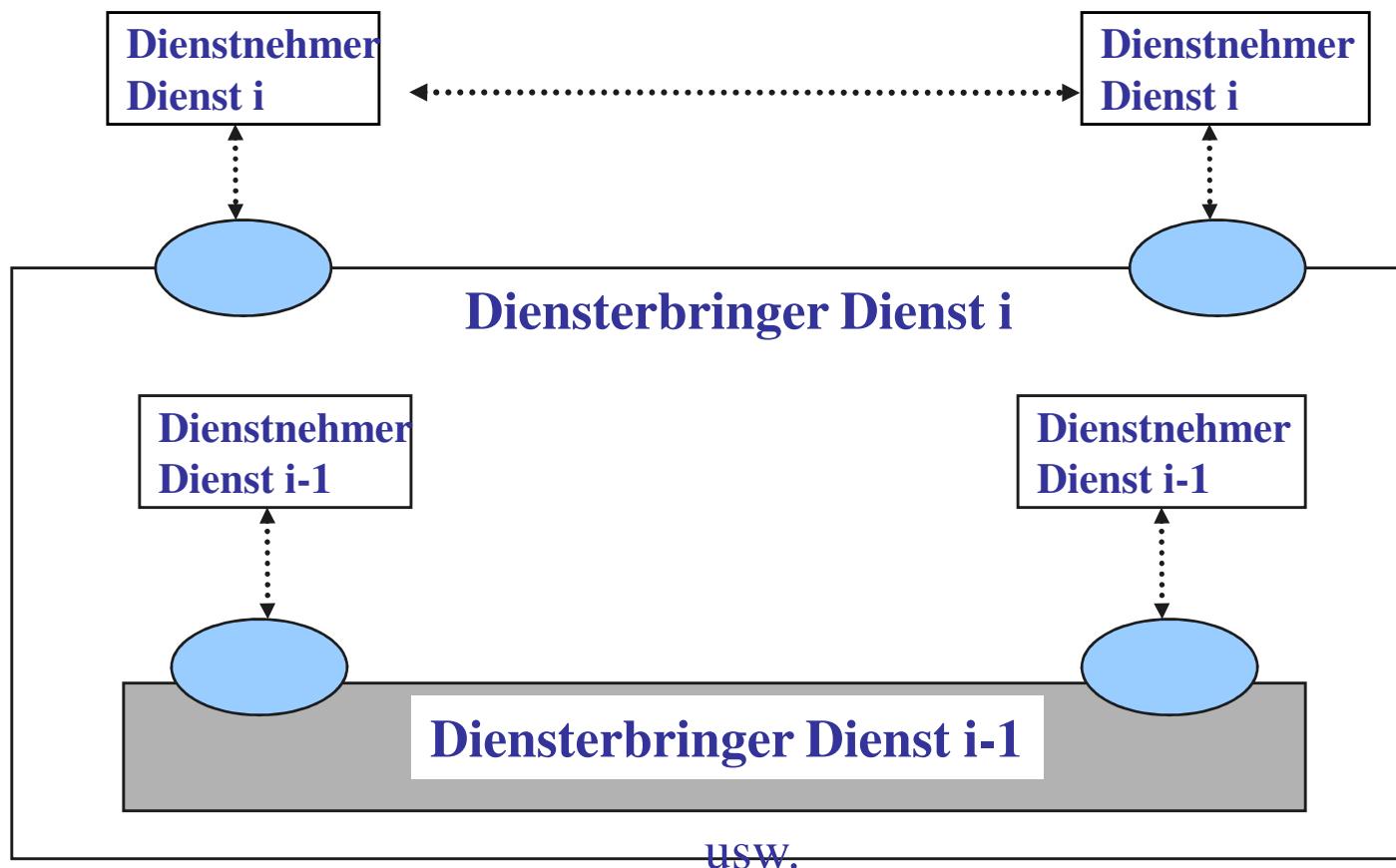
- Der **Dienstnehmer** in einer Schicht i nutzt zur Kommunikation einen **Diensterbringer** in (Dienstprovider) der wiederum die darunter liegende Schicht ($i-1$) nutzt
- Die Dienste werden über **Dienstzugangspunkte** (Service Access Points, SAP) bereitgestellt
- SAPs sind logische Schnittstellen, deren konkrete Realisierung z. B. in Form einer Funktionsbibliothek oder als eigener Systemprozess gegeben sein kann

ISO/OSI-Dienstmodell



ISO/OSI-Dienstmodell

Hierarchische Dienststruktur (Vgl. Gerdsen: Kommunikationssysteme 1
Theorie Entwurf Meßtechnik S. 17)



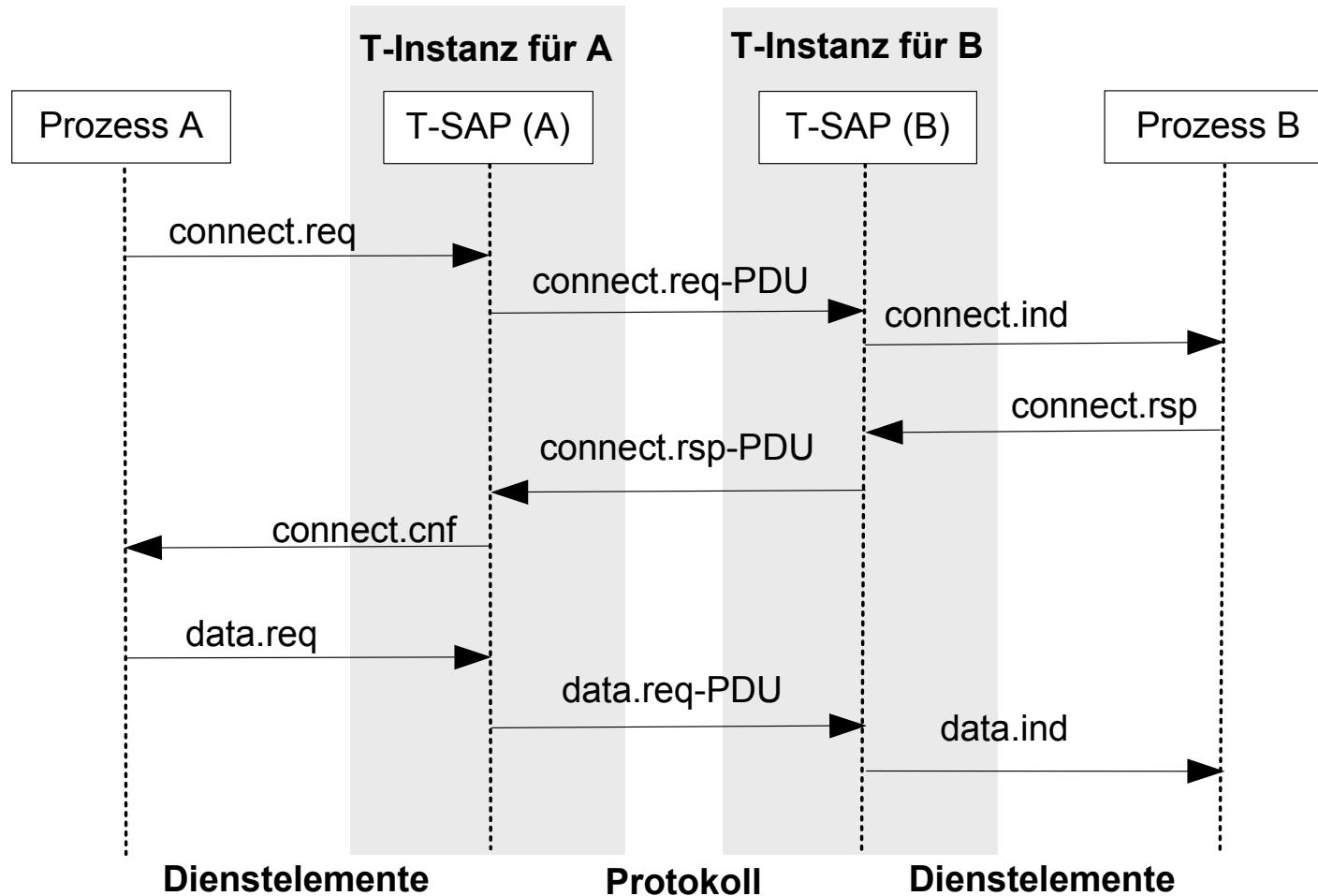
ISO/OSI-Referenzmodell: Instanzen

- Die Funktionen innerhalb einer Schicht werden von einer **Instanz** ausgeführt
- Eine Instanz erbringt die Dienstleistung, die ein Dienstnehmer von einem Dienstprovider erwartet
- Instanzen sind auf den kommunizierenden Systemen verteilt
- Instanzen der gleichen Schicht kommunizieren miteinander über Protokolle
- Zwei kommunizierende Instanzen werden als Partnerinstanzen bezeichnet

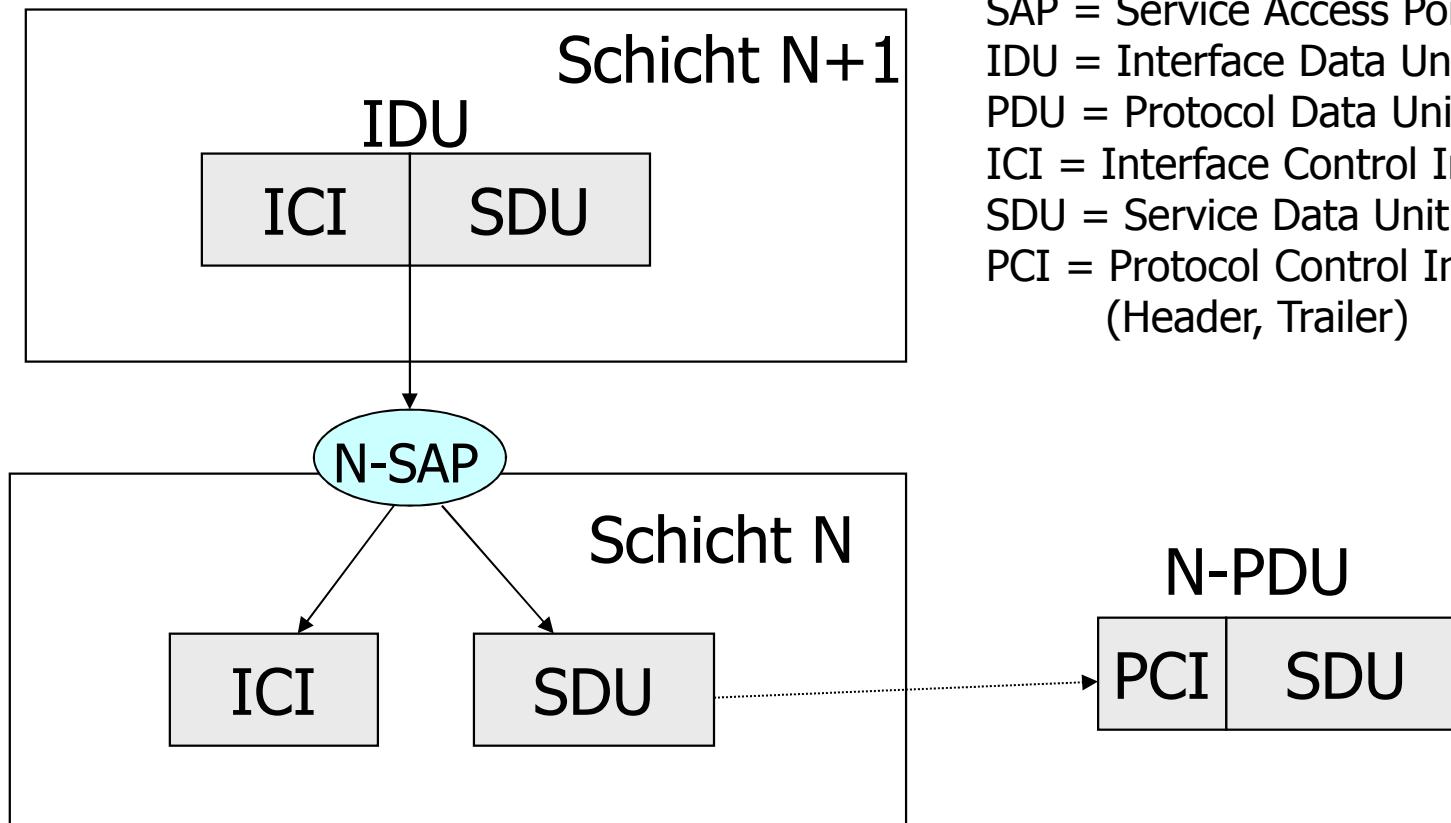
ISO/OSI-Referenzmodell: Dienstelemente

- Dienste (Abstraktion) im ISO/OSI-Modell
 - Beispiel: connect, disconnect, data (Datenübertragung)
- Dienstelemente/Dienstprimitive sind Operationen eines Dienstes
- Typische Dienstprimitive sind
 - Request
 - Indication
 - Confirmation
 - Response
 - Beispiel: connest.req, connect.ind, data.req

ISO/OSI-Referenzmodell: Dienste und Protokoll

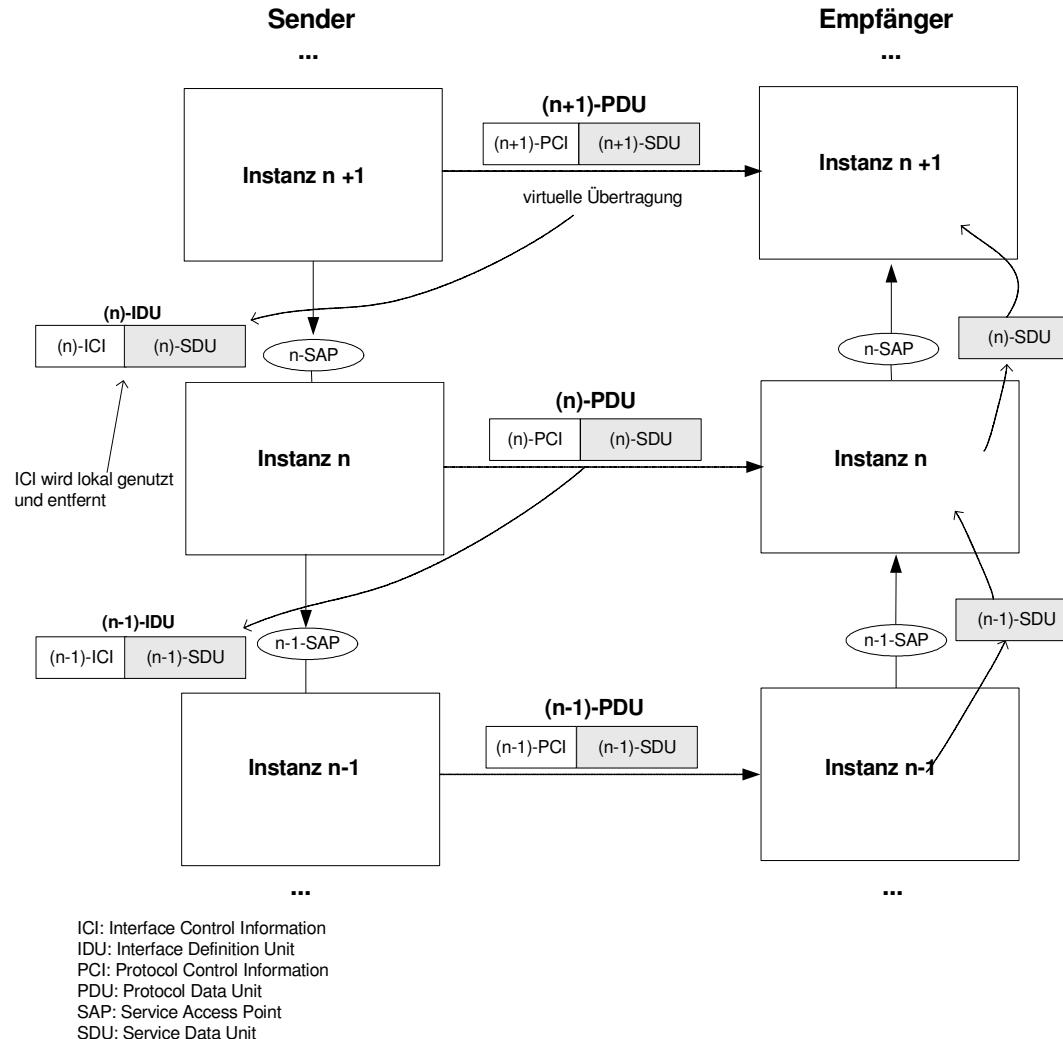


ISO/OSI-Referenzmodell: Begriffe



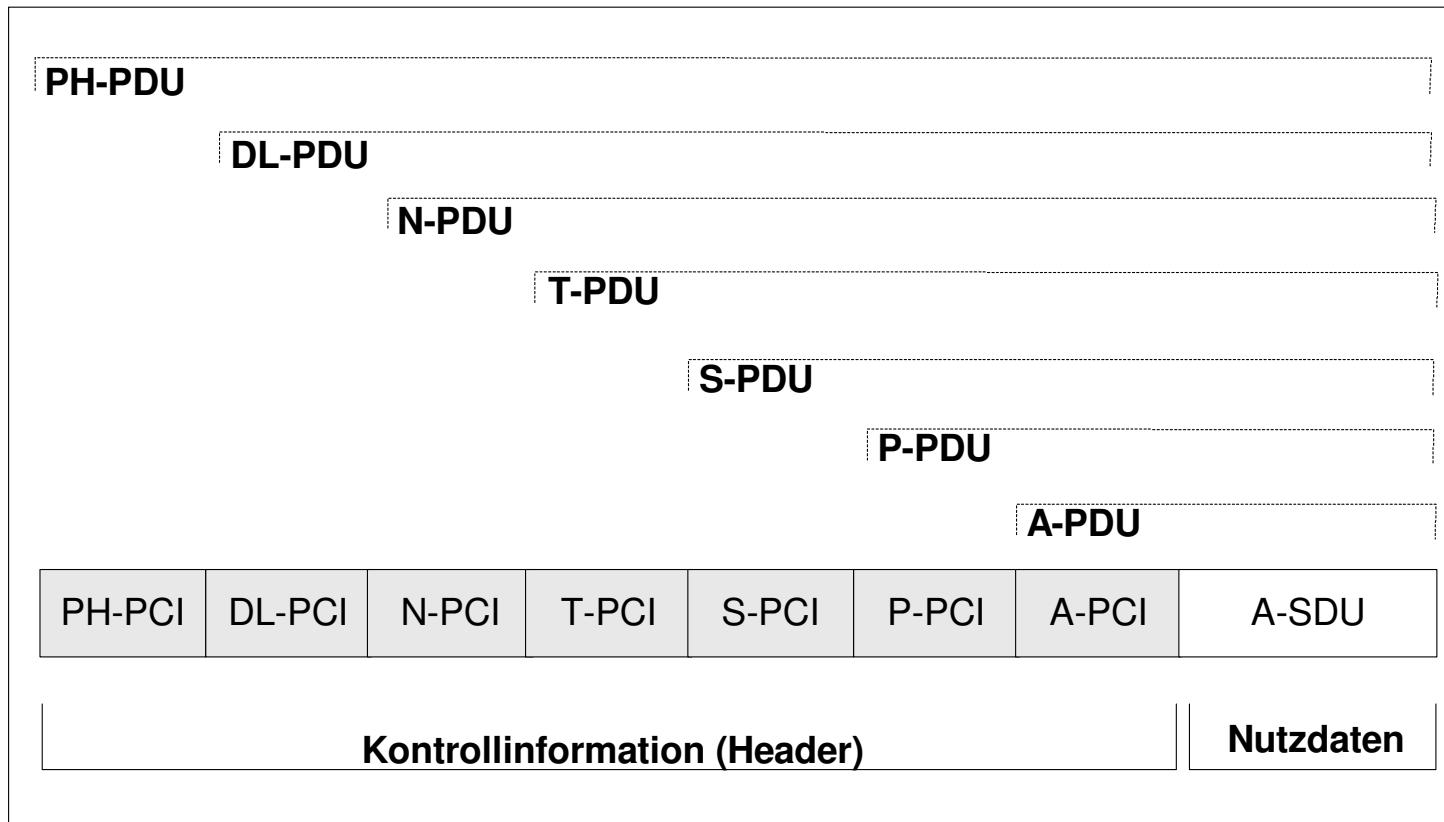
SAP = Service Access Point
IDU = Interface Data Unit
PDU = Protocol Data Unit
ICI = Interface Control Information
SDU = Service Data Unit
PCI = Protocol Control Information
(Header, Trailer)

ISO/OSI-Referenzmodell: Begriffe



ISO/OSI-Referenzmodell: Begriffe

A-PDU (PDU der Anwendungsschicht)



Transportsystem

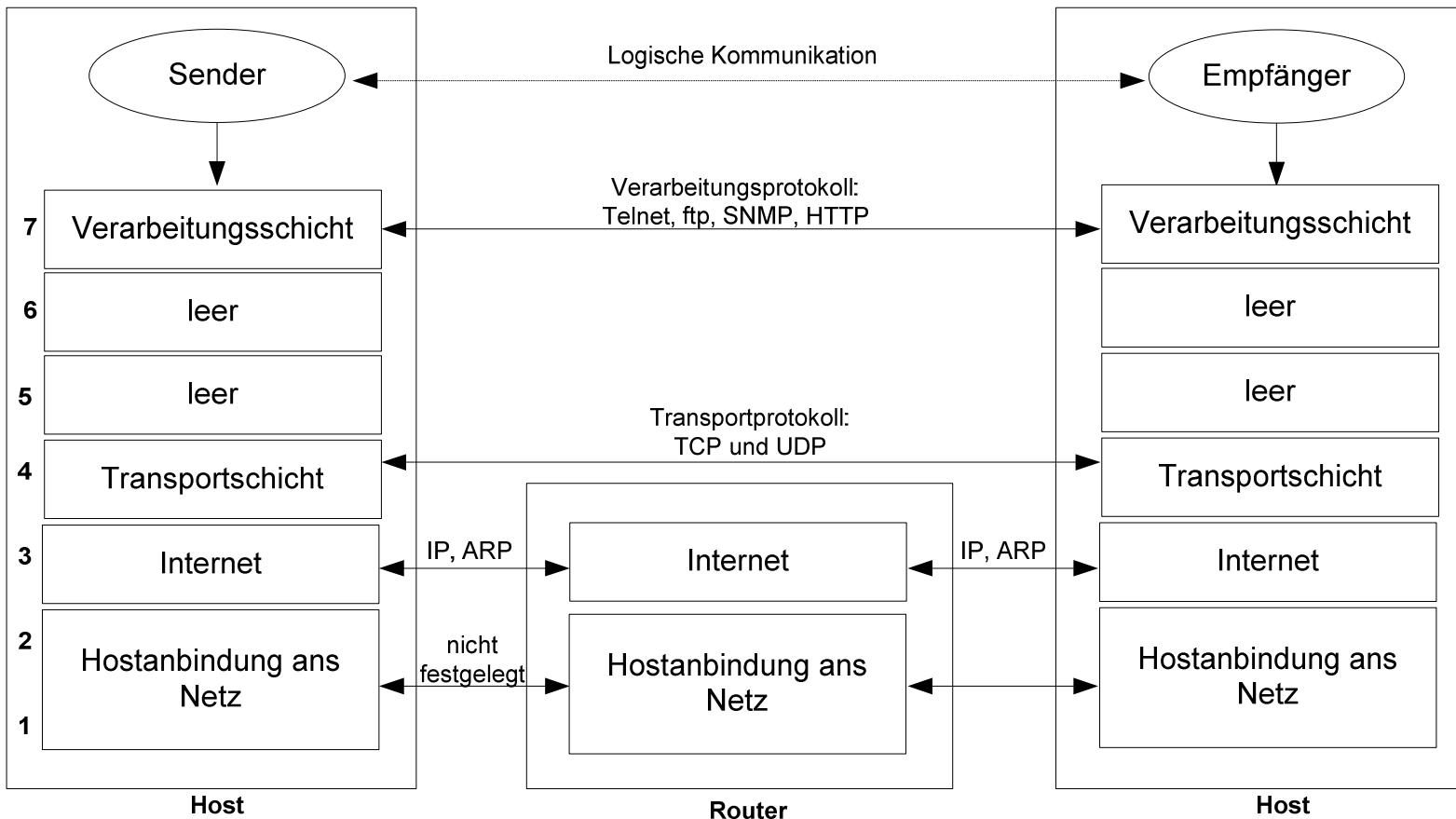
■ **Transportschicht (Überblick)**

- Stellt einen Transportservice bereit
- Ende-zu-Ende-Verbindung zwischen Anwendungsprozessen
- Gesicherte Transportverbindung



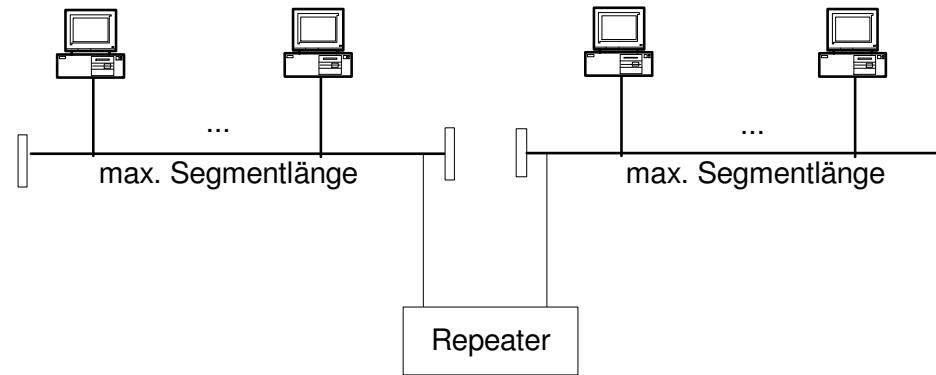
- Schicht 1-4 werden gemeinsam als **Transportsystem** bezeichnet
- Transportzugriffsschnittstelle ermöglicht Nutzung des Transportsystems
 - z.B. Dienst OSI TP4
 - Vergleichbar mit TCP in der TCP-Welt

TCP/IP-Referenzmodell

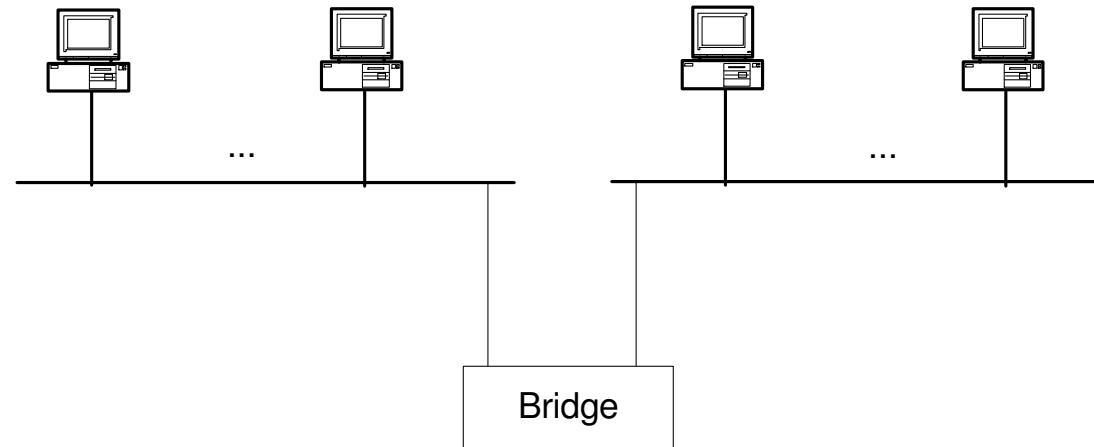


Internetworking: Repeater und Bridges

- Repeater

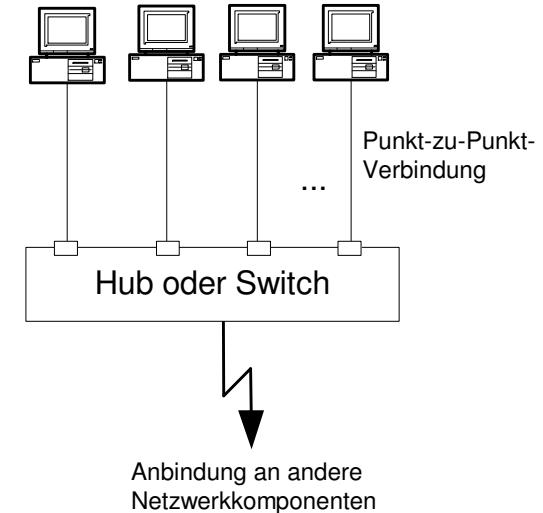


- Bridges

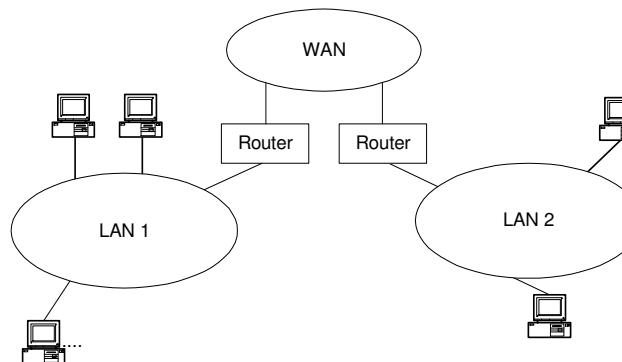


Internetworking: Hubs, Switches, Router, Gateways

- Hubs, Switches



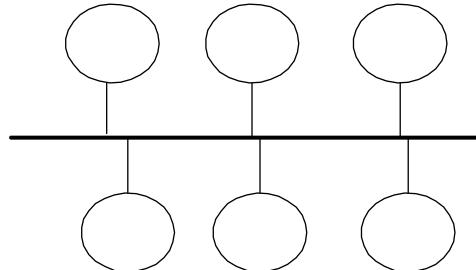
- Router



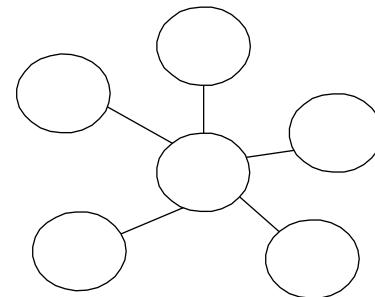
- Abgrenzung:

- Gateways (auf der Anwendungsebene)

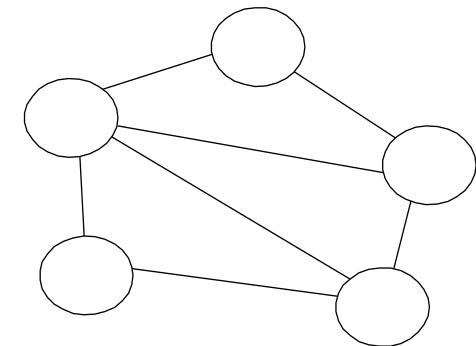
Netzwerktopologien



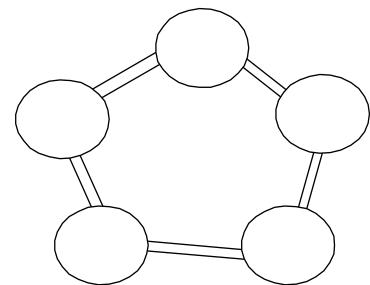
Busnetz



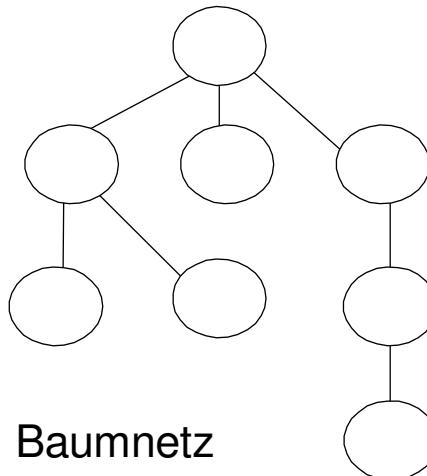
Sternnetz



Vermaschtes
Netz



Ringnetz



Baumnetz

Überblick

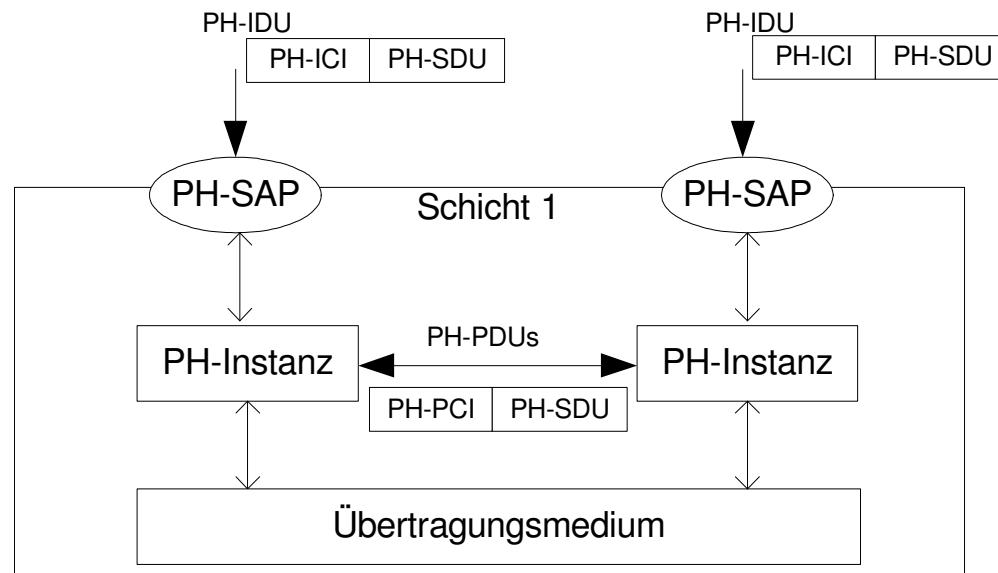
1. Referenzmodelle und Terminologie

2. Bitübertragungsschicht

- Aufgaben, Begriffe und Definitionen
- Kodierung (Leitungskodierung)
- Laufzeit, Übertragungszeit, Transferzeit, Jitter
- Digitale Übertragung und PCM, Multiplexverfahren
- Datenübertragungsmedien und Verkabelung

Aufgaben der Bitübertragungsschicht

- Bereitstellung der physikalischen Verbindung, Zugang zum Medium
 - Festlegung der elektrischen, mechanischen und funktionalen Parameter
 - Physikalische Bitdarstellung für das benutzte Übertragungsmedium
 - Übertragung von Bits und Bitgruppen



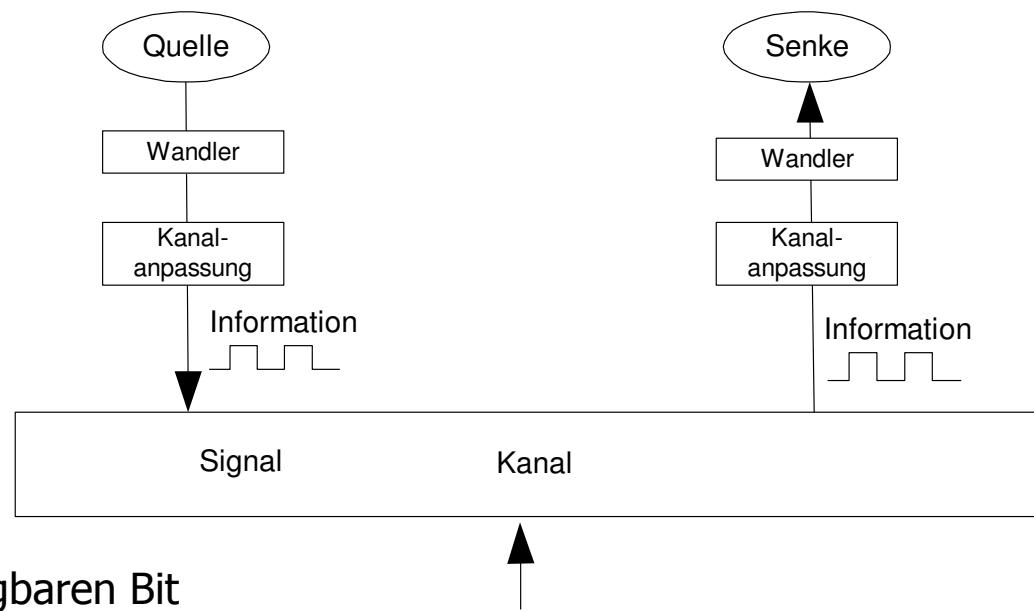
Bitrate, Schrittgeschwindigkeit und Bandbreite

■ Schrittgeschwindigkeit S

- Die Anzahl der Zustandsänderungen eines Signals pro Zeiteinheit
- Einheit: baud = bd = 1/s (Hz)

■ Bandbreite B

- Physikalische Eigenschaft des Mediums
- Einheit: Hz



■ Bitrate R

- Anzahl der übertragbaren Bit pro Zeiteinheit
- Gemessen in Bit/s

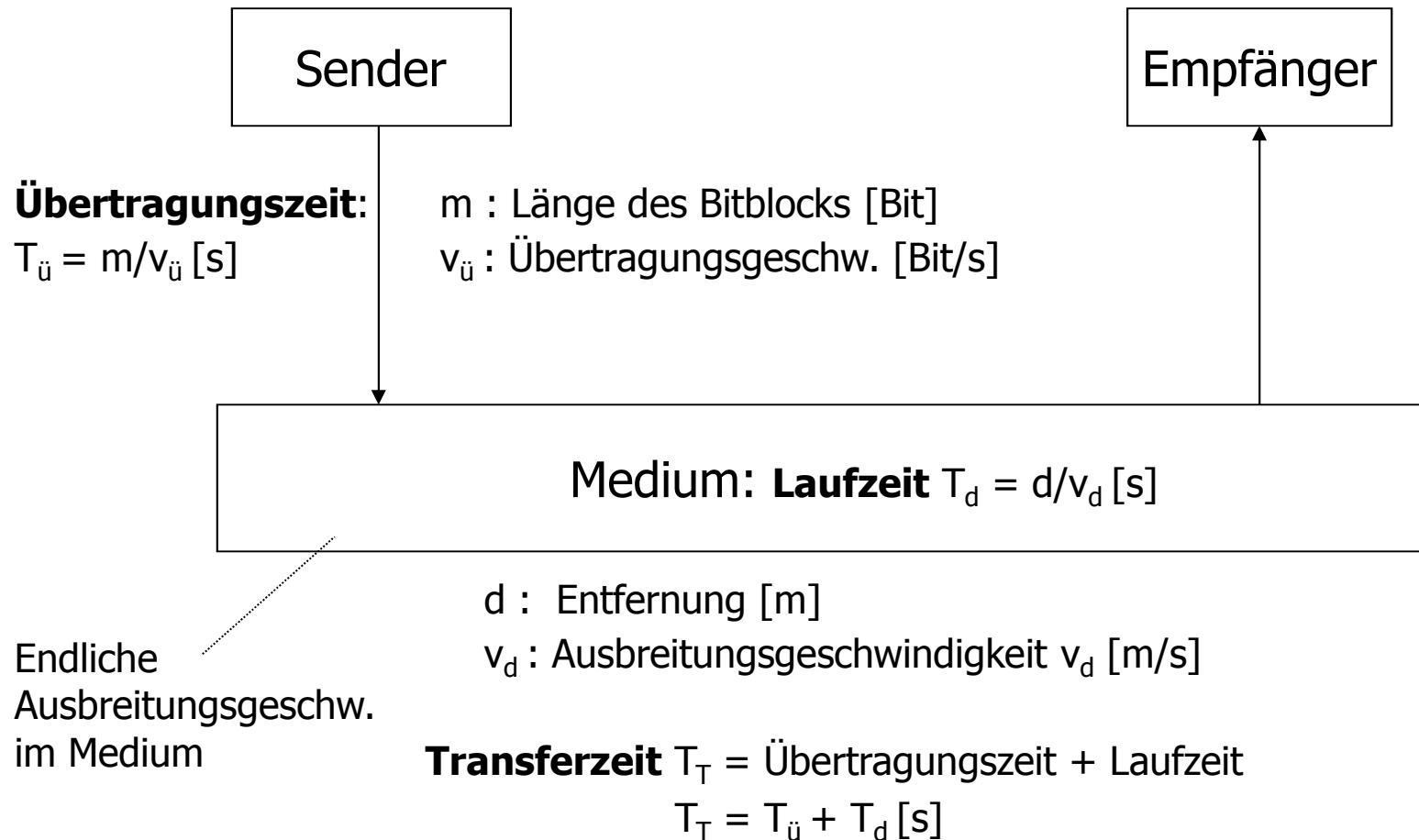
Schrittgeschwindigkeit

- Die **Schrittgeschwindigkeit S** (auch Schrittrate, Taktfrequenz) legt fest,
 - wie oft sich der Wert des Signals (z.B. die Spannung) pro Sekunde bzw. Zeiteinheit ändert
 - S wird gemessen in **baud, bd = 1/s (Hertz, Hz)**
- Eine Leitung mit b baud überträgt nicht unbedingt b Bit/s
 - Jeder Signalwert kann nicht immer nur ein Bit, sondern evtl. auch mehrere oder weniger Bits übertragen
 - **Nur** bei binärem Signal gilt: 1 baud entspricht 1 Bit/s

Übertragungsstörungen

- Übertragungsmedien sind nicht perfekt
- Störungen möglich
 - **Dämpfung** ist der Energieverlust, der bei der Verbreitung eines Signals entsteht; Angabe in Dezibel pro Kilometer [db/km]
 - Energieverlust abhängig von der Frequenz
 - Signal fällt bei terrestrischen Medien logarithmisch mit der Entfernung
 - **Laufzeitverzerrung**: Überholen und damit Mischen von Bits
 - **Rauschen**: Beeinträchtigung durch unerwünschte Energie aus anderen Quellen
 - Beispiel: Nebensprechen durch induktive Kopplung zwischen eng benachbarten Drähten

Laufzeit, Übertragungszeit, Transferzeit



Laufzeit, Übertragungszeit, Transferzeit

- Typische Ausbreitungsgeschwindigkeiten und dazu gehörige Laufzeiten vgl.: Gerdsen, P., Kommunikationssysteme

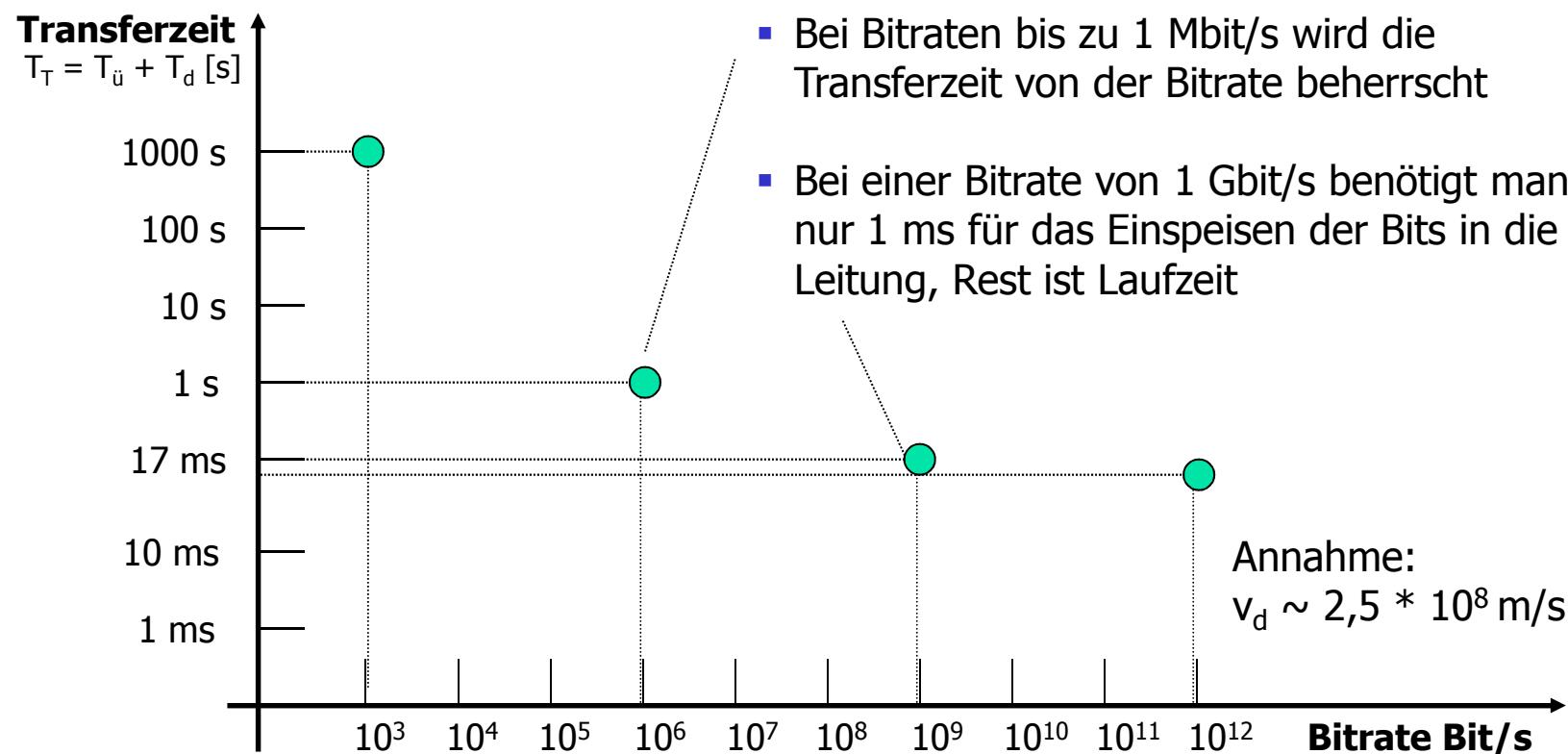
Medium	Ausbreitungsgeschw. V_d [m/s]	Laufzeit T_d [μs/km]
Funkkanal	$3 * 10^8$ (näherungsweise)	3,33
Freiraum-Infrarot	$3 * 10^8$ (näherungsweise)	3,33
Glasfaserleitung (Quarzglas)	$2 * 10^8$	5
Basisband-Koaxialkabel (50/75 Ohm)	$2,3 * 10^8$	4,33
Zweidrahtleitung (verdrillt)	$2,5 * 10^8$	4

Laufzeit und Übertragungszeit

- Bei niedrigen Übertragungsgeschwindigkeiten ist die Laufzeit gegenüber der Übertragungszeit vernachlässigbar: $T_{ü}$ ungefähr T_T
- Bei sehr großen Entfernungen (mehrere Tausend km) spielt die Laufzeit eine große Rolle

Übertragung bei großen Distanzen

- Übertragung einer Datei mit 1000000 Bit (10^6) über 4.000 Km ($4 * 10^6$ m)



Berechnungsbeispiele

■ Gegeben

- Transferzeit: $T_T = T_{\ddot{u}} + T_d$ [s] (bis zum Netzzugang des Empfängers)
- Übertragungszeit: $T_{\ddot{u}} = m/v_{\ddot{u}}$ [s] (im Medium, ohne sonstige Verzögerungen)
- Laufzeit: $T_d = d/v_d$ [s] (Übergabe der Signale ans Netz)
- Länge des Bitblocks $m = 10^6$ Bit
- Distanz $d = 4000$ km = $4 * 10^6$ m
- Ausbreitungsgeschwindigkeit $v_d \sim 2,5 * 10^8$ m/s (Kupfer)

■ Bitrate 10^3 Bit/s

- $T_{\ddot{u}} = 10^6$ Bit / 10^3 Bit/s = 10^3 s = **1000 s**
- $T_d = 4 * 10^6$ m / $2,5 * 10^8$ m/s = $1,6 * 10^{-2}$ s = $0,016$ s = **16 ms**
- $T_T = 1000 s + 16 ms$

■ Bitrate 10^9 Bit/s

- $T_{\ddot{u}} = 10^6$ Bit / 10^9 Bit/s = 10^{-3} s = **1 ms**
- $T_d = 16 ms$ (wie oben)
- $T_T = 1 ms + 16 ms = 17 ms$

Verzögerungsschwankung, Jitter

- Je nach Netzwerktechnologie unterliegt das Senden und Empfangen von aufeinanderfolgenden Paketen gewissen zeitlichen Schwankungen
- Als **Verzögerungsschwankung** bzw. **Jitter** (Flattern) bezeichnet man die Schwankung in der Verzögerung nacheinander empfangener Datenpakete
- Multimedia-Anwendungen (Audio- und Video) benötigen z.B. einen kontinuierlichen Datenfluss mit
 - minimaler Verzögerung und
 - minimaler Verzögerungsschwankung

Übertragungsverfahren (1)

■ **Analoge Übertragung des Bitstroms**

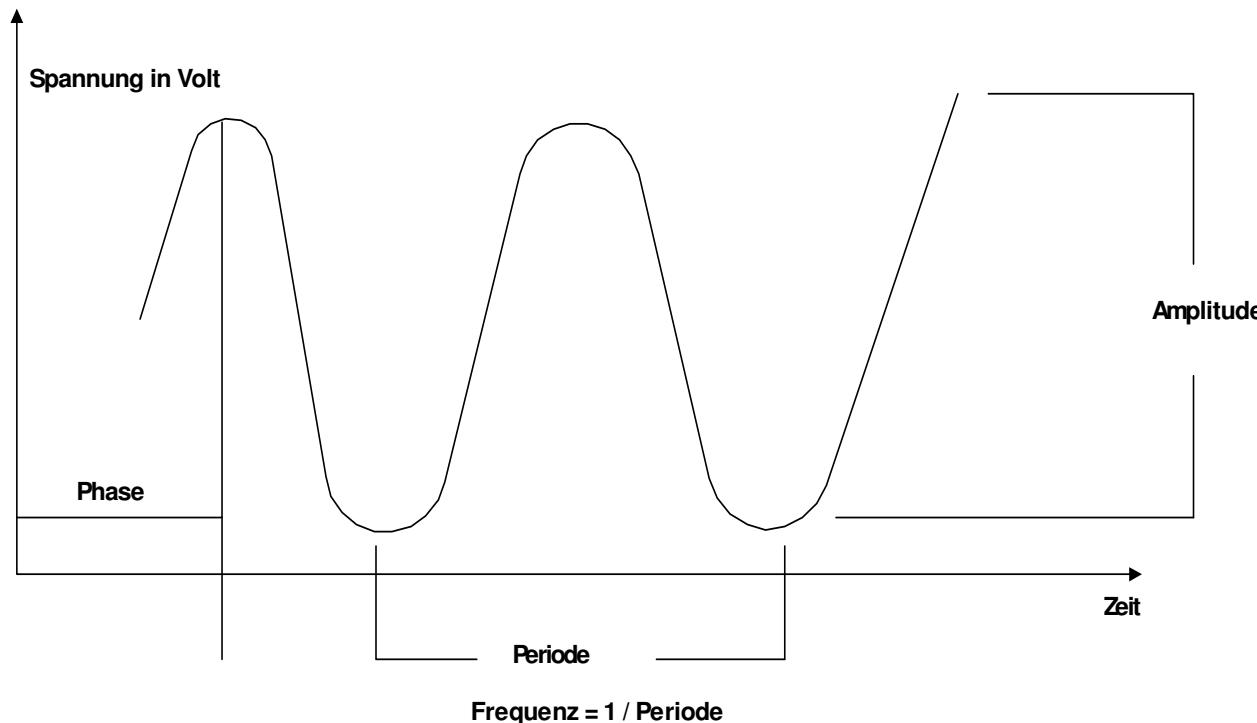
- Menge der kodierten Werte ist kontinuierlich, unendlicher Zeichenvorrat, Übertragung als elektrische Schwingung
- Digitale Signale müssen auf Trägersignal aufmoduliert werden
- Modulationsarten:
 - Frequenzmodulation (eine Schwingung = 1, zwei Schwingungen = 0)
 - Amplitudenmodulation (flache Schwingung = 0)
 - Phasenmodulation (Schwingung von unten nach oben = 1)
- Beispiel: V24

■ **Digitale Übertragung des Bitstroms**

- Endlicher Zeichenvorrat mit sprungartigem Übergang zwischen den digitalen Zeichen
- Beispiel: ISDN, ADSL (Asymmetric Digital Subscriber Line)

Übertragungsverfahren (2)

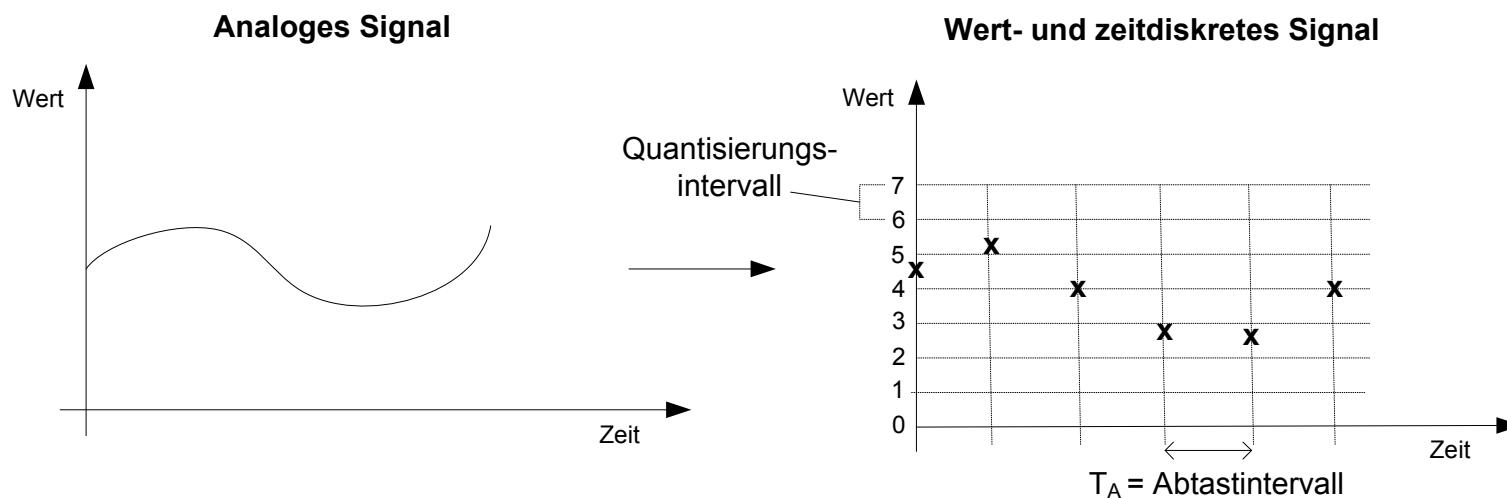
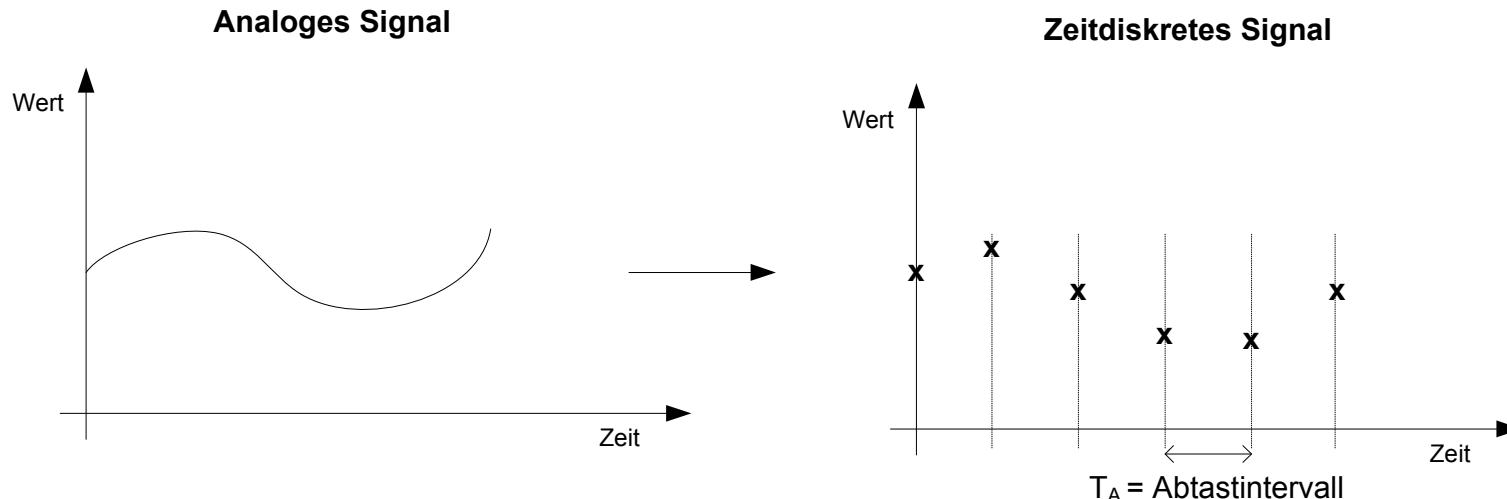
- Analoge Übertragung: Verlauf der Spannung als Sinuskurve
- Für die Kodierung der Zustände 1 und 0 kann genutzt werden: Variation der Amplitude, der Frequenz oder der Phase



Pulscodemodulation (PCM) (1)

- PCM bildet die Grundlage, um analoge Signale digital zu übertragen (im Zeitmultiplexverfahren)
- Analoges Signal ist zeit- und wertkontinuierlich
- Digitales Signal ist zeit- und wertdiskret
- Drei unterschiedliche Bearbeitungsschritte sind für die Umwandlung notwendig
 - Abtastung
 - Quantisierung
 - Kodierung

Pulscodemodulation (PCM) (2)



PCM: Abtastung

- Ziel der Abtastung
 - Zeitkontinuierliches Signal → zeitdiskretes Signal
- Basis: **Abtasttheorem von Shannon**
 - Periodische Abtastung benötigt minimale Abtastfrequenz f_A , um das Originalsignal fehlerfrei zu rekonstruieren
 - $1/T = f_A > 2 * B$ (B = Bandbreite)
- Analoger Fernsprechkanal: Bandbreite 3100 Hz zwischen 300 Hz und 3400 Hz
 - $f_A > 2 * 3100 \text{ Hz} (= 6200 \text{ Hz})$
 - Technisch wurde erhöhte Abtastfrequenz von 8000 Hz realisiert
 - Abtastperiode $T_A = 1/f_A = 125 \mu\text{s}$
 - Das analoge Sprachsignal wird somit alle $125 \mu\text{s}$ abgetastet

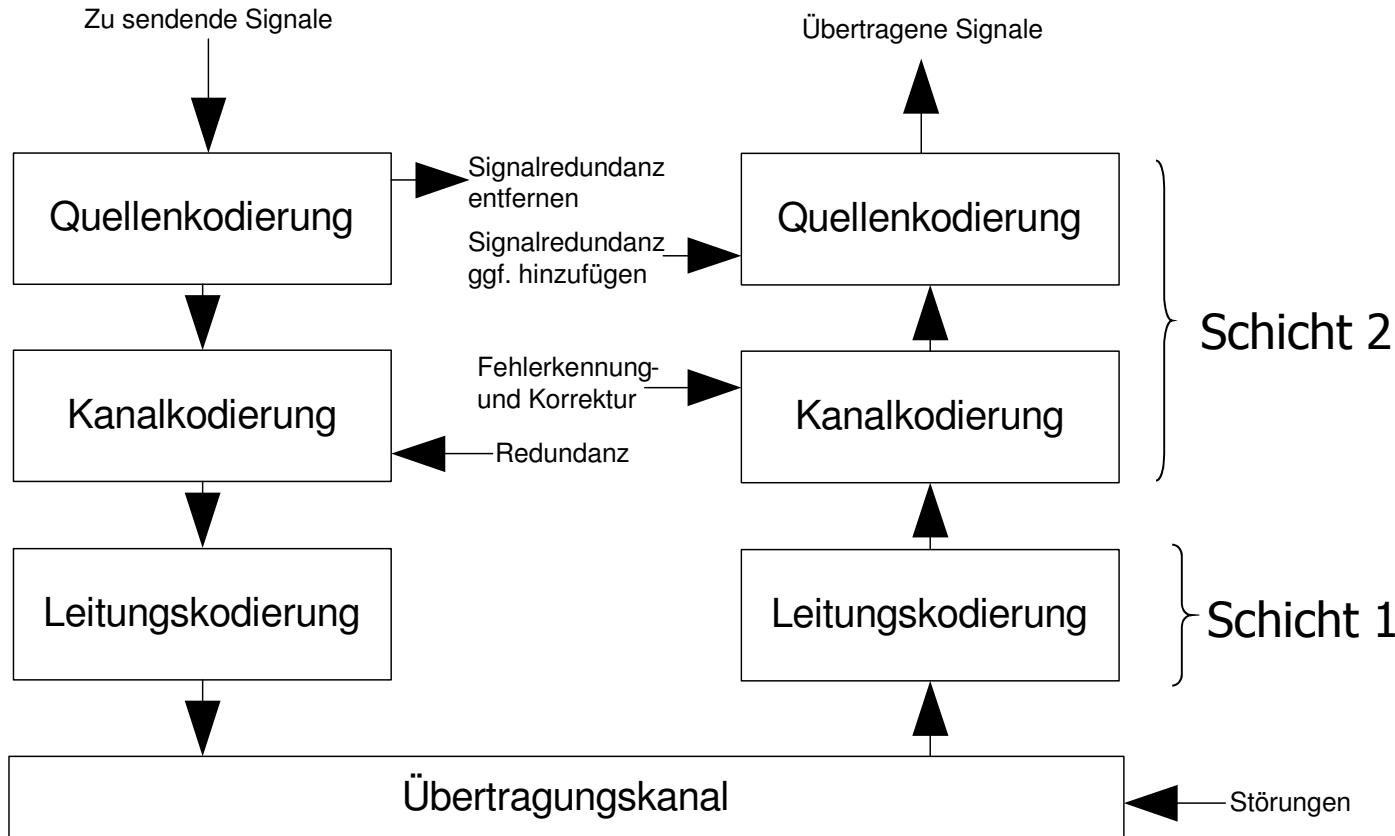
PCM: Quantisierung

- Die Quantisierung hat zum Ziel, aus einem wertkontinuierlichen Signal ein wertdiskretes Signal zu erzeugen
- Wertebereich des analogen Signals wird in eine endliche Anzahl sog. Quantisierungsintervalle gegliedert
- Jedes Intervall wird mit einem diskreten Wert belegt, d.h. allen analogen Signalen des Intervalls wird derselbe diskrete Wert zugewiesen
 - → Quantisierungsfehler, die sich beim Empfänger durch Rauschen bemerkbar machen (max. $\frac{1}{2}$ des Intervalls)
- Bei PCM-Technik: 256 Quantisierungsintervalle

PCM: Kodierung

- Bei der Kodierung werden den Quantisierungsintervallen sog. Codes zugewiesen
- Die PCM-Technik benötigt 8 Bits für die Kodierung (256 Intervalle müssen kodiert werden)
- Daraus ergibt sich ein digitaler Sprachkanal mit folgender Bitrate:
 - $f_A * 8 \text{ bit} = 8000 \text{ Hz} * 8 \text{ bit} = 64 \text{ kbit/s}$ ($\text{Hz} \rightarrow 1/\text{s}$)
- Einsatz z.B. bei B-Kanälen im ISDN (Schmalband)
- 64 kbit/s-Kanal stellt die Basiseinheit für die PCM-Übertragungshierarchie dar
 - PCM-30 ist dann: $30 * 64 \text{ kbit/s} + 2 * 64 \text{ kbit/s} = 2,048 \text{ Mbit/s}$
 - PCM-120: 8,448 Mbit/s, ...

Quellen-, Kanal- und Leitungskodierung

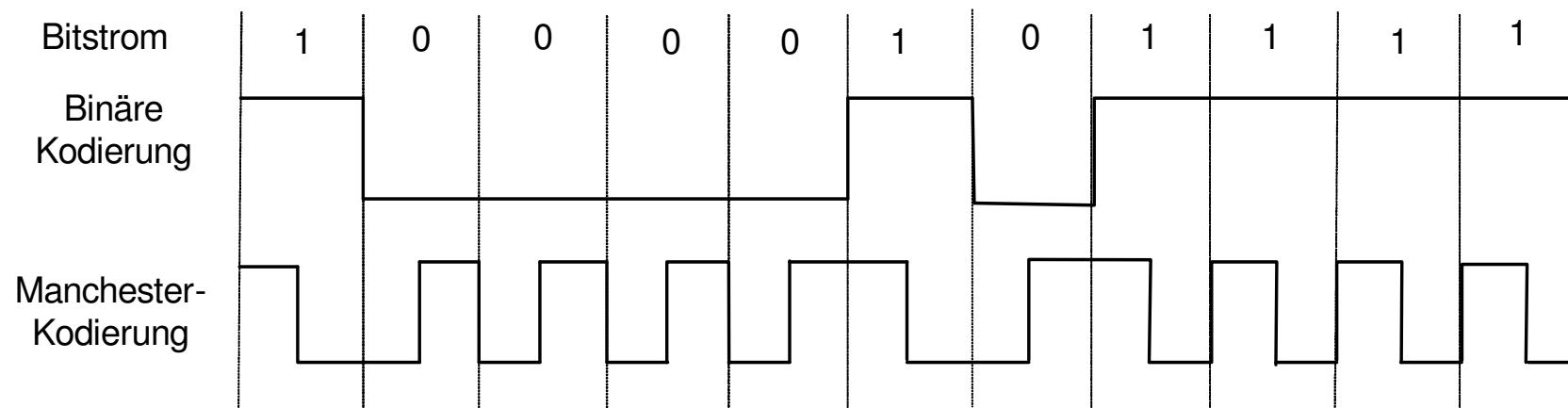


Leitungskodierung (1)

- Leitungskodierung legt fest, wie ein Signal physikalisch übertragen wird: Abbildung Bits → Signale
- Einfachste Form der Kodierung: Reine Bitkodierung
 - Zwei verschiedene Spannungen für 0 und 1
 - Aber: Bei langen Sequenzen von 0 oder 1 geht der Takt verloren, was zu Fehlinterpretationen beim Empfänger führt
 - Takt unterliegt immer kleinen Schwankungen
- Besser: Manchester-Kodierung (Bi-Phase-Mark)
 - Pegeländerung vor einer Bitdarstellung wird zur Taktrückgewinnung genutzt
- Weitere Verfahren
 - NRZI-Verfahren(Non-Return-to-Zero-Inverted)
 - MLT-3-Verfahren (Multi-Level 3)
 - AMI-Code, modifizierter AMI-Code (ISDN, S0-Bus)
 - 4B/5B-Kodierung, ..., 8B/10B-Kodierung (bei Gbit-Ethernet), ...

Leitungskodierung (2)

- **Manchester-Kodierung:** Pegeländerung vor einer Bitdarstellung wird zur Taktrückgewinnung genutzt
 - 1: von hoch auf niedrig
 - 0: umgekehrt



Leitungskodierung (3) - 4B/5B-Kodierung

- Manchester-Kodierung benötigt je Bit genau 2 Baud
 - Schlechte Effizienz: 50 %
- Reduzierung der erforderlichen Schrittgeschwindigkeit z.B. durch den 4B/5B-Kode
 - Unterbrechung von Langen 0- oder 1-Folgen durch Ergänzungsbits
 - 4 Bits der Daten werden zu 5-Bit-Kodes ergänzt
Effizienz: 80 % → 4 Bit mit 5 Baud übertragen

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

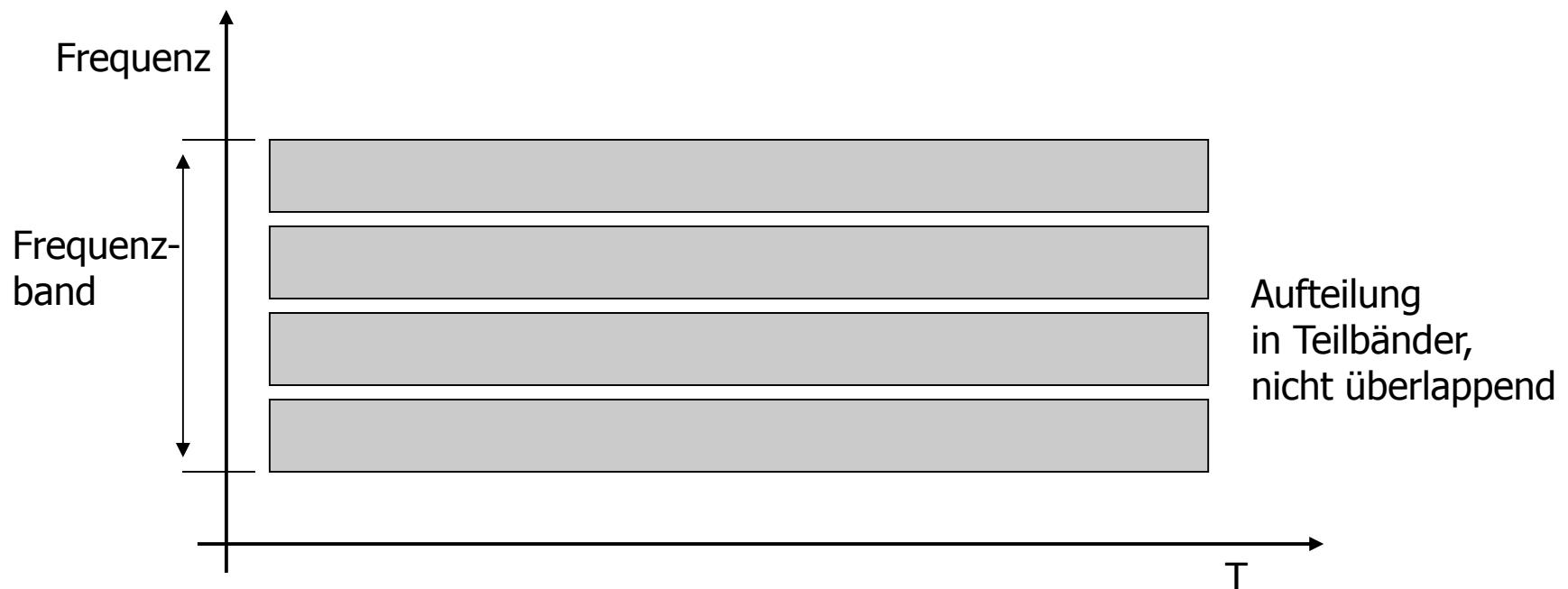
4-Bit-Daten	5-Bit-Code
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Mehrfachnutzung von Medien

- Übertragungskanal ist teuer
- Ausschöpfung der Übertragungskapazität über Multiplexierungsverfahren (Multiplexverfahren)
- Medien mit größerer Bandbreite können für mehrere Kanäle genutzt werden
- Man unterscheidet u.a:
 - Frequenzmultiplexverfahren
 - Zeitmultiplexverfahren

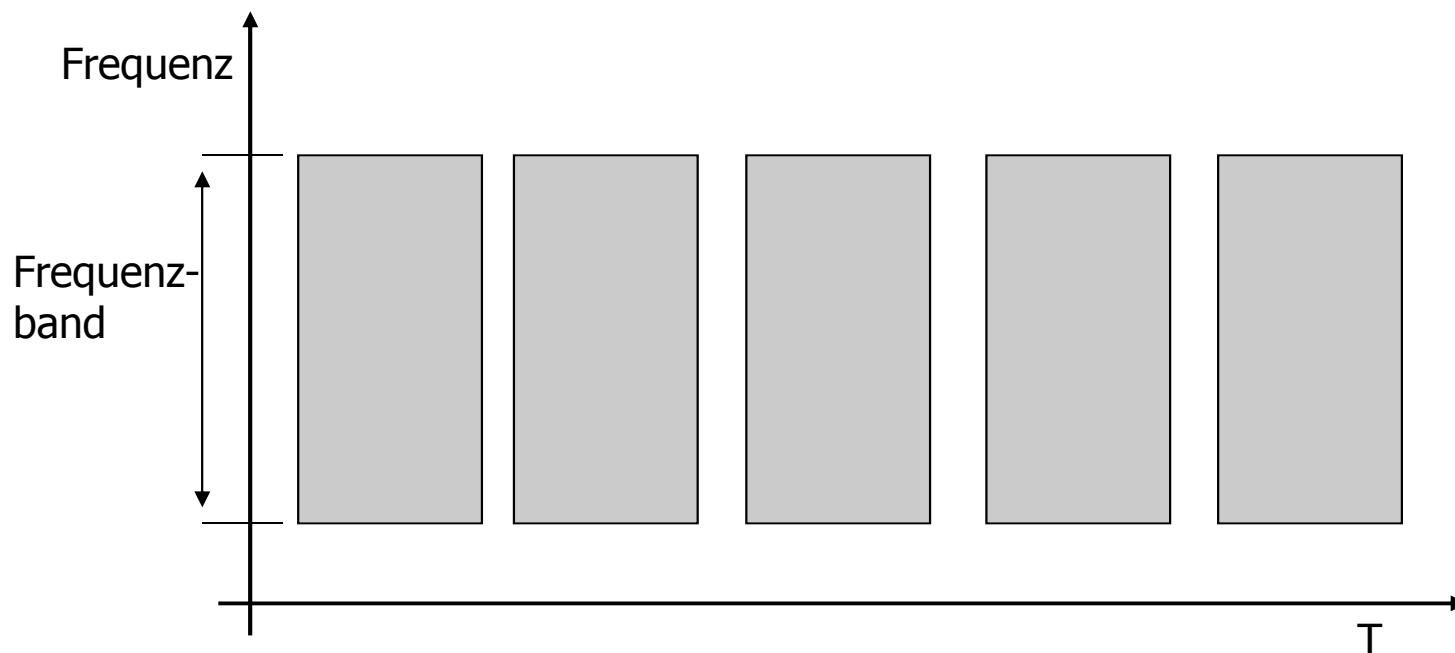
Frequenzmultiplexverfahren

- Beim Frequenzmultiplexverfahren (FDM = Frequency Division Multiplexing) wird das nutzbare Frequenzband des Mediums in mehrere Teilbänder aufgeteilt



Zeitmultiplexverfahren

- Beim Zeitmultiplexverfahren (TDM = Time Division Multiplex) wird das Medium für bestimmte Zeitperioden einem Quellen-Senken-Paar zugeordnet
- Übertragung in Zeitperioden, Umschaltung erforderlich



Datenübertragungsmedien: Überblick

Medium	Aufbau	Kanalkapazität
Twisted Pair	Verdrilltes 8-adriges Kupferkabelpaar (Telefonkabel)	Bis 10 Gbit/s
Koaxialkabel	Von zylindrischem Leiter umschlossener Kupferdrahtkern	~10Mbit/s
Glasfaserkabel	Von zylindrischem Glasmantel umschlossener Glaskern	Mehrere Gbit/s
Mobiler Datenfunk	Terrestrischer Zellularfunk (z. B. Funk-LAN)	Einige Mbit/s
Richtfunk	Terrestrisch, ortsfest	Bis 4 Mbit/s
Satellitenfunk	Geostationäre Nachrichtensatelliten	56 kbit/s und mehr

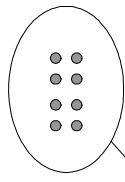
Datenübertragungsmedien: Typische Bitraten

Netze	Bandbreite (Bitrate)	Übertragung eines Lexikons mit 600 MB
Festnetztelefonie: Analog ISDN DSL	56 kbit/s 64 kbit/s 8 Mbit/s	25 Std. 25 Std. 10 Min.
Mobilkommunikation: GSM UMTS	9,6kbit/s 2 Mbit/s	146 Std. 42 Min.
Lokale Netze: Ethernet Fast-Ethernet Gigabit-Ethernet	10 Mbit/s 100 Mbit/s 1 Gbit/s	8 Min. 50 Sek. 5 Sek.
Weitverkehrsbandnetze: Breitband-ISDN (Auf Basis von ATM)	155 Mbit/s 622 Mbit/s	32 Sek. 8 Sek.
Optische Fernverbindungen: Pro Wellenlänge Pro Glasfaser	40 Gbit/s 1,6 Tbit/s	0,1 Sek. 0,003 Sek.

Kabelarten und Verkabelung

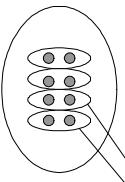
■ Kabelarten, Beispiel Twisted-Pair

Unshielded Twisted Pair
(UTP)



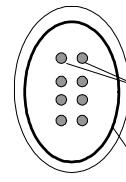
Kunststoffmantel

Shielded Twisted Pair
(STP)



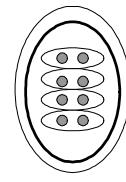
Adernschirm
(Shield)

Screened/Unshielded
Twisted Pair (S/UTP)



Kabelschirm (Screen)

Screened/Shielded Twisted
Pair (S/STP)



Adernpaar

Kat.5-Kabel (100 Mbit/s - 1 Gbit/s)



Kat.6-Kabel (1 – 10 Gbit/s)



Kat.7-Kabel (10 Gbit/s)



■ Kategorie 3

- zwei isolierte verdrillte Kabel
- gemeinsame Umhüllung für vier Adernpaare

■ Kategorie 5

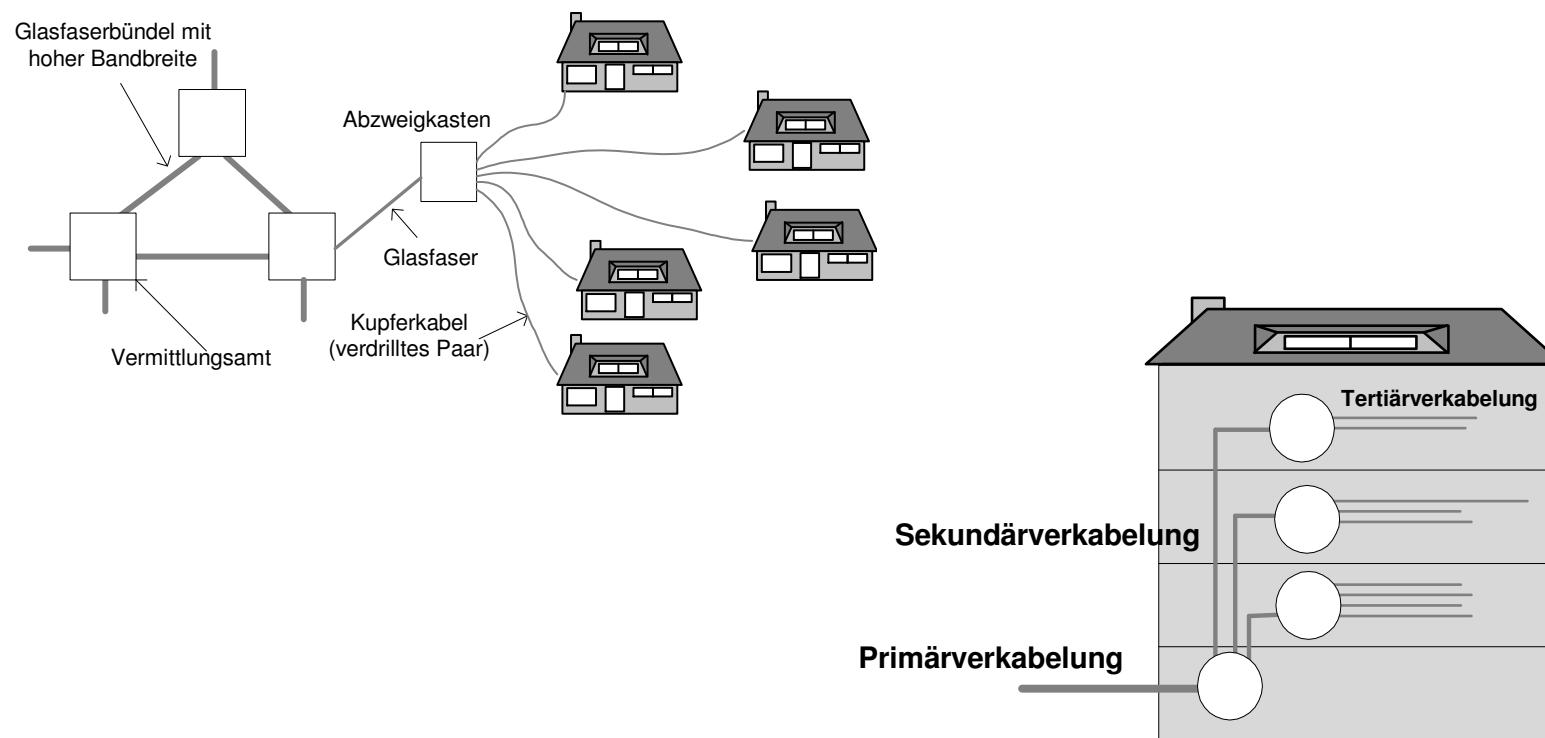
- mehr Windungen pro cm
- Umhüllung aus Teflon

■ Kategorie 6,7 → Kat 7 = SF/FTP (Foiled Twisted Pair → Geflecht + Folie)

- Adernpaare sind einzeln mit Silberfolie umwickelt

Verkabelung, strukturierte Verkabelung

- Verkabelung der letzten Meile
- Strukturierte Verkabelung in Gebäuden



Rückblick

1. Referenzmodelle und Terminologie

- ISO/OSI-Referenzmodell
- TCP/IP-Referenzmodell
- Transportsystem
- Netzwerktopologien
- Internetworking

2. Bitübertragungsschicht

- Aufgaben, Begriffe und Definitionen
- Kodierung (Leitungskodierung)
- Laufzeit, Übertragungszeit, Transferzeit, Jitter
- Digitale Übertragung und PCM, Multiplexverfahren
- Datenübertragungsmedien und Verkabelung

Datenkommunikation

Grundlagen
von Rechnernetzen, Teil 2

Wintersemester 2012/2013

Einordnung

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

1. Sicherungsschicht

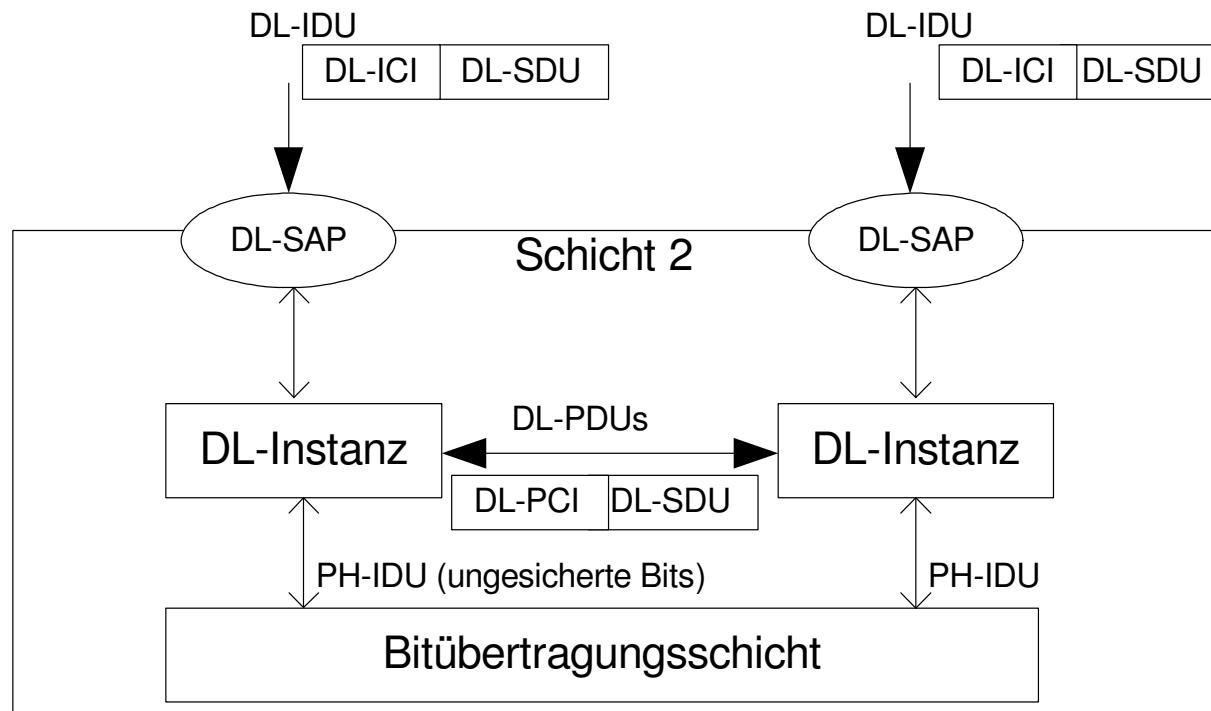
- Aufgaben
- XON/XOFF-Protokoll
- Kodierung (Quellen-, Kanalkodierung)

2. Buszugriffsverfahren und Ethernet

- Überblick
- CSMA-Protokolle
- Ethernet

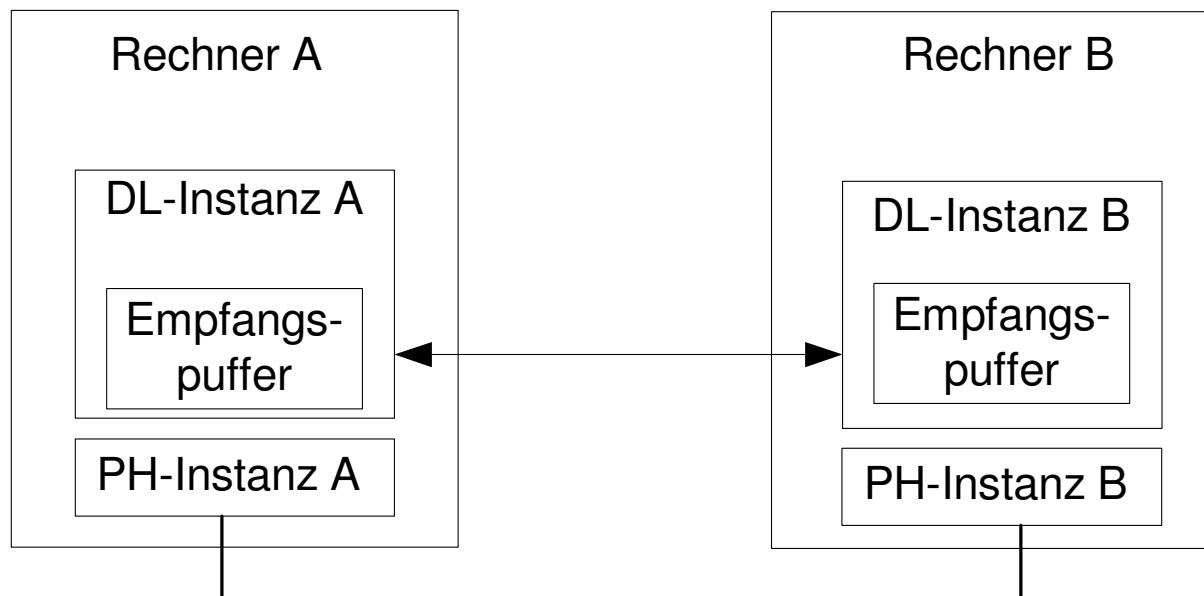
Aufgaben

- Gruppierung des übertragenen Bitstroms in logische Einheiten
- Fehlererkennung (Prüfsummen) und ggf. Fehlerkorrektur
- Ende-zu-Ende-Verbindung zwischen Rechnern/Knoten



DL-Instanzen

- Schicht-2-Instanzen verwalten üblicherweise Empfangspuffer und auch Sendepuffer

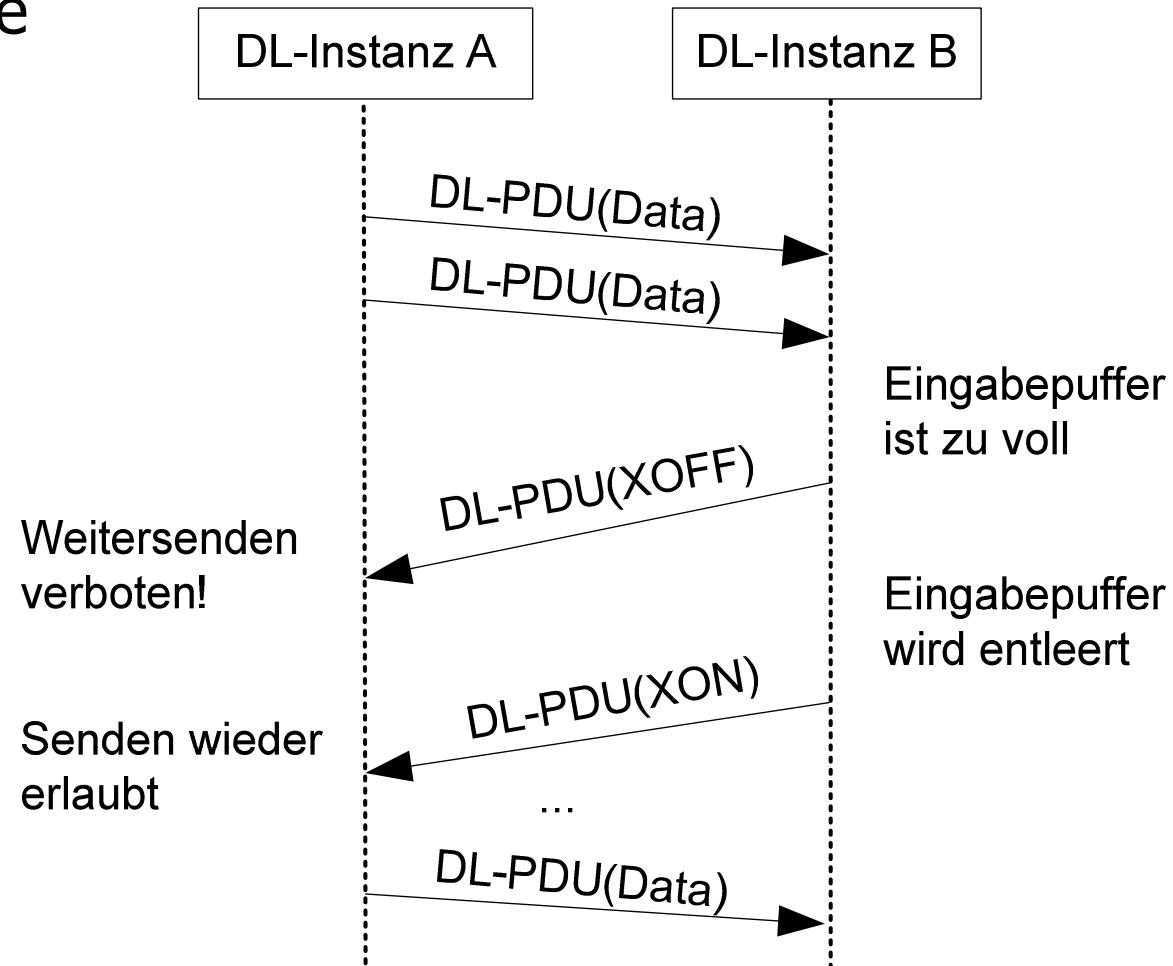


Typische Schicht-2-Protokolle

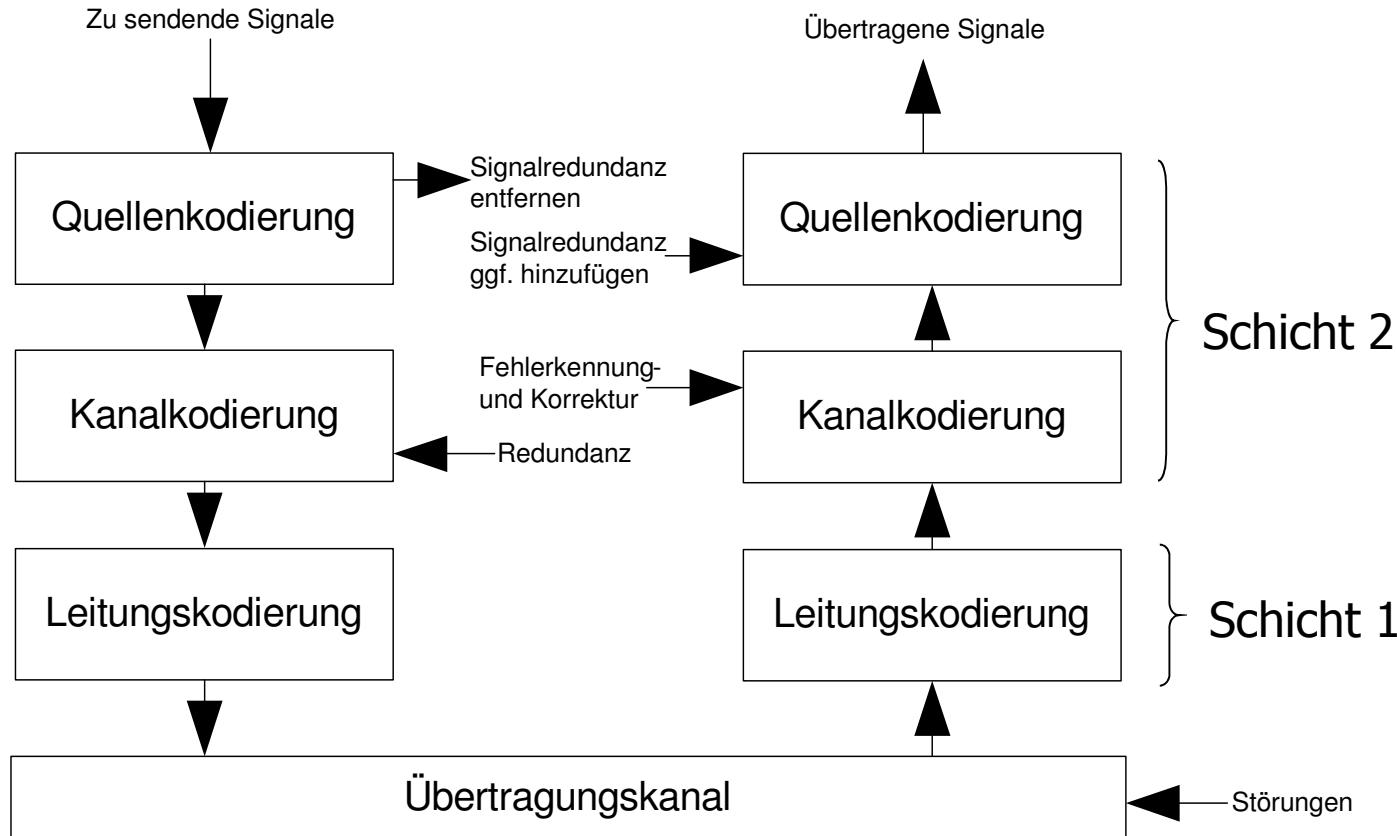
- HDLC
- SDLC
- BSC
- PPP

Beispiel: Flusskontrolle mit XON/XOFF-Protokoll

■ Flusskontrolle



Quellen- und Kanalkodierung



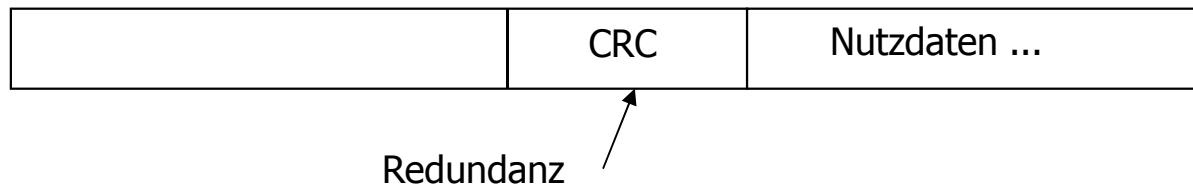
Quellenkodierung

- Aufgabe: Information mit möglichst geringer Bitrate übertragen
- Datenkomprimierung
 - Wichtig vor allem bei Audio- und Videoströmen
 - **Verlustbehaftete** Kompression
 - . Quellenkodierungstechniken
 - . Semantik des Bitstroms wird ausgewertet und für die Komprimierung genutzt
 - . Z.B. JPEG (Bilder), MP3 (Audio), MPEG (Video)
 - **Verlustfreie** Kompression
 - Entropiekodierungstechniken
 - . Manipulation des Bitstroms, ohne Betrachtung der Semantik
 - . Z.B. einfache Lauflängenkompression
 - . AAAAABBBCCDDDD → 4A3BC4D

Kanalkodierung

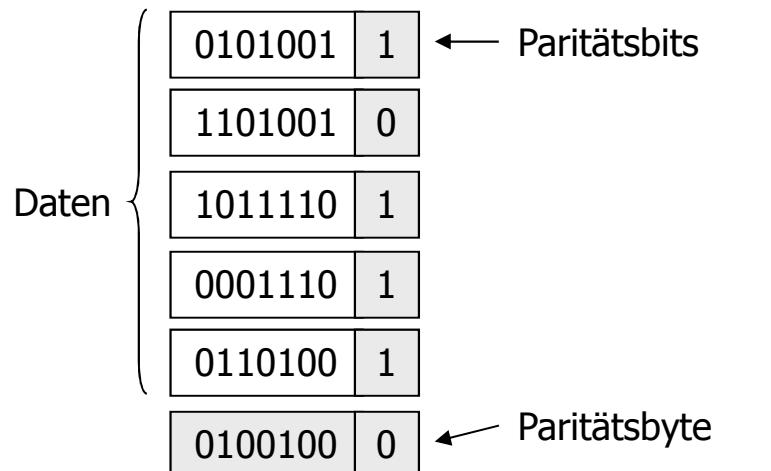
- Aufgabe: Übertragungsfehler durch Redundanz erkennen und behandeln
 - Fehlererkennende und fehlerkorrigierende Codes
 - Nutzung auch bei der Speicherung von Daten (z.B. Hamming-Code)
- Verfahren
 - Paritätsbits (einfache Paritätsbits und zweidimensionale Parität)
 - Prüfsummen (auch in IP für Header genutzt)
 - Zyklische Redundanzcodes (CRC = Cyclic Redundancy Check)

Beispiel:
Header eines Schicht-2-Protokolls (DL-PCI)



Kanalkodierung – Zweidimensionale Parität

- 7-Bit-Code wird um ein Paritätsbit ergänzt (even oder odd)
- Über alle Bytes einer Nachricht
- Even Parity = auf gerade 1-Bit-Anzahl erweitern
 - 0101100 → 01011001
- Ein zusätzliches Paritätsbyte für die gesamte Nachricht (in Schicht 2 auch Frame genannt)
- Beispiel: (5 Bytes im Frame) → 35 Nutzdaten-Bits, 13 Bits Redundanz



Es lässt sich zeigen:
Alle 1-, 2- und 3-Bit-
und die meisten 4-Bit-Fehler
werden erkannt!

Kanalkodierung – CRC (1)

- Gute Fehlererkennungsfähigkeit bei k redundanten Bits in einer n -Bit-Nachricht auch wenn $k \ll n$ (entschieden kleiner)
- Beispiel:
 - Ethernet nutzt CRC-Code: Bei 1500-Byte-Frame = 12000 Bit wird mit 32-Bit-langem CRC der Großteil der Fehler gefunden ($n = 12000$, $k = 32$)
- Wie wird es gemacht?
 - Senden und Empfangen von Nachrichten durch Austausch von „Polynomen“
 - Nachricht mit $n+1$ Bits wird durch ein Polynom vom Grad n repräsentiert
 - Bits der Nachricht werden als Koeffizienten in den Termen verwendet
 - Beispiel:
 - . Nachricht: 11011010
 - . $M(x) = x^7 + x^6 + x^4 + x^3 + x$
 - Divisor-Polynom $G(x)$ vom Grad k wird vereinbart → Auswahl wichtig für die Fehlererkennung
 - . Beispiel: $G(x) = x^3 + x^2 + 1$ ($k = 3$)
 - Gesendet werden bei einer Nachricht der Länge $n+1$ insgesamt $n+1+k$ Bits
 - Die redundante Nachricht wird als Polynom $T(x)$ bezeichnet
 - $T(x)$ muss durch $G(x)$ ohne Rest teilbar sein

Kanalkodierung – CRC (2)

- Grundlage: Modulo-2-Arithmetik
 - Polynom $B(x)$ ist durch Divisor-Polynom $G(x)$ teilbar, wenn $B(x)$ einen höheren Grad als $G(x)$ hat
 - Polynom $B(x)$ ist einmal durch Divisor-Polynom $G(x)$ teilbar, wenn $B(x)$ den gleichen Grad als $G(x)$ hat
 - Rest einer Division wird durch Subtraktion $B(x) - G(x)$ ermittelt
 - Subtraktion wird durch XOR-Operationen auf korrespondierende Koeffizientenpaare ermittelt
- Beispiel:
 - $B(x) = x^3 + x \rightarrow 1010$
 - $G(x) = x^3 + x^2 + 1 \rightarrow 1101$

$$\begin{array}{r} 1010 \\ 1101 \text{ XOR} \\ \hline 0111 = \text{Rest} \end{array}$$

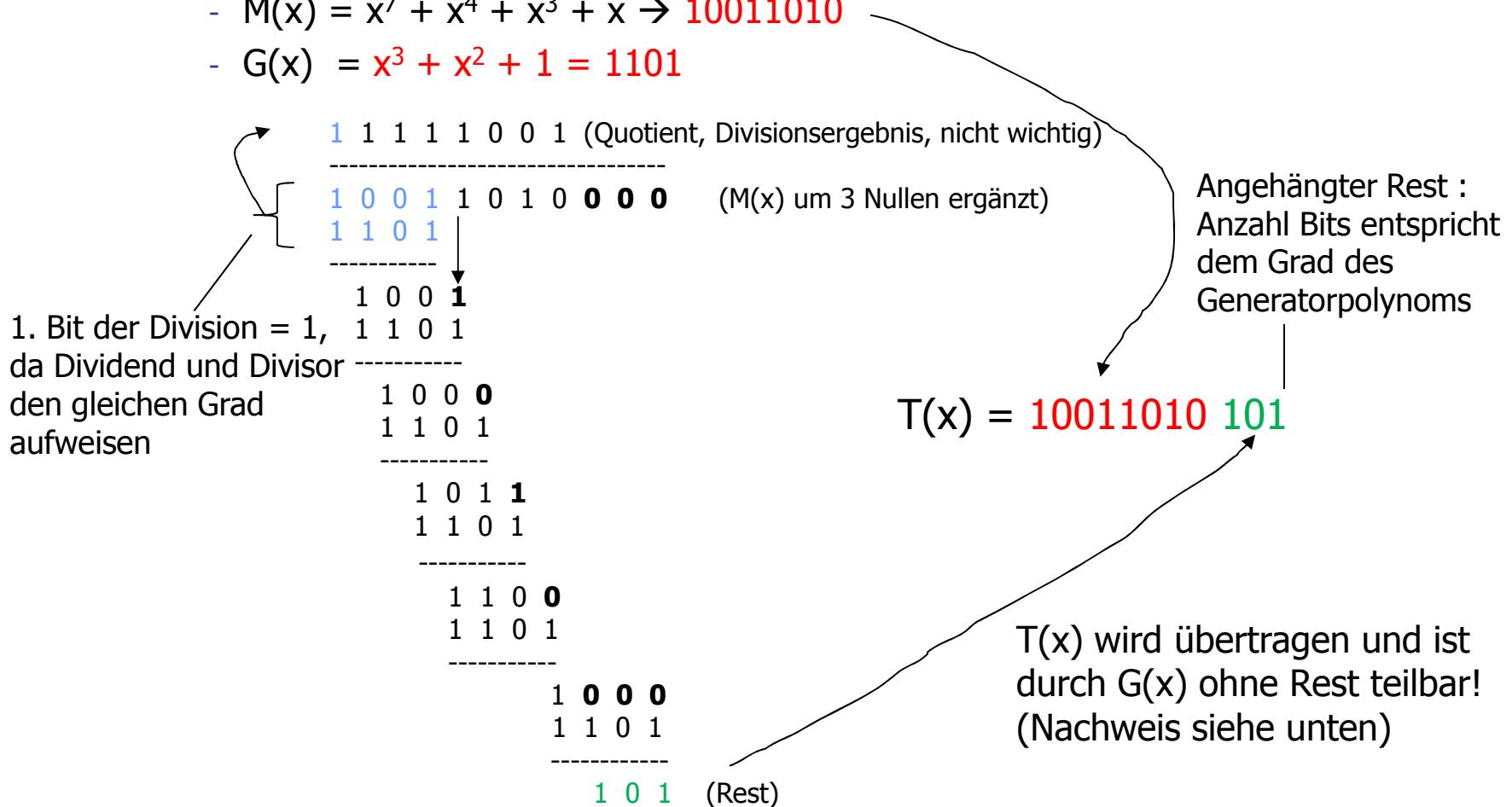
Kanalkodierung – CRC (3)

- Algorithmus:
 - $T(x) = M(x)$ ergänzt um k 0-Bits ($M(x)$ = Nachricht)
 - Dividiere $T(x)$ durch $G(x)$
 - Subtrahiere den Rest der Division von $T(x)$ → Ergebnis ist die um die Prüfsumme ergänzte Nachricht
 - Übertrage $T(x)$ an Empfänger
 - Empfänger teilt $T(x)$ durch $G(x)$ und muss bei Fehlerfreiheit 0 als Rest erhalten, sonst ist die Übertragung fehlerhaft
- Es lässt sich zeigen:
 - $T(x)$ ist durch $G(x)$ teilbar!
 - Denn es gilt für jede Division: Wenn man vom Dividenden den Rest abzieht, ist das Ergebnis durch den Divisor teilbar
 - Beispiel: $101 : 25$ ist 4 Rest 1 → $101 - 1 = 100$ (ist durch 25 teilbar)

Kanalkodierung – CRC (4)

■ Beispiel:

- $M(x) = x^7 + x^4 + x^3 + x \rightarrow 10011010$
- $G(x) = x^3 + x^2 + 1 = 1101$



Kanalkodierung – CRC (5)

- Sender und Empfänger müssen natürlich $G(x)$ kennen
- Ermittlung von $G(x)$ (CRC-Polynom = Generator-Polynome) so, dass die Wahrscheinlichkeit sehr gering ist, eine **falsche** Nachricht so zu teilen, dass der Rest 0 ist
- Wichtige CRC-Polynome:
 - CRC-CCITT wird im HDLC-Protokoll verwendet
 - . $x^{16} + x^{12} + x^5 + 1$
 - CRC-32 wird im Ethernet-Protokoll verwendet
 - . $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Ergänzung zum Verständnis: Modulo-2 Arithmetik
 - M = Daten, R = Divisionsrest, G = Divisor (Generatorpolynom), Q = Quotient
 - $M * 2^n / G = Q + R / G \Rightarrow M * 2^n + R / G = Q \quad (R/G + R/G = 0)!!$
 - Dies lässt sich zeigen, da in Modulo-2 die Addition gleicher Zahlen immer 0 ergibt

Übung

- Zeigen Sie, dass die Nachricht $T(X)$ von oben bei richtiger Übertragung durch das Generatorpolynom $G(X)$ ohne Rest teilbar ist
 - $T(x) = 10011010 \ 101$
 - $G(x) = x^3 + x^2 + 1 = 1101$
- $T(X) : G(x) = ?$

Überblick

1. Sicherungsschicht

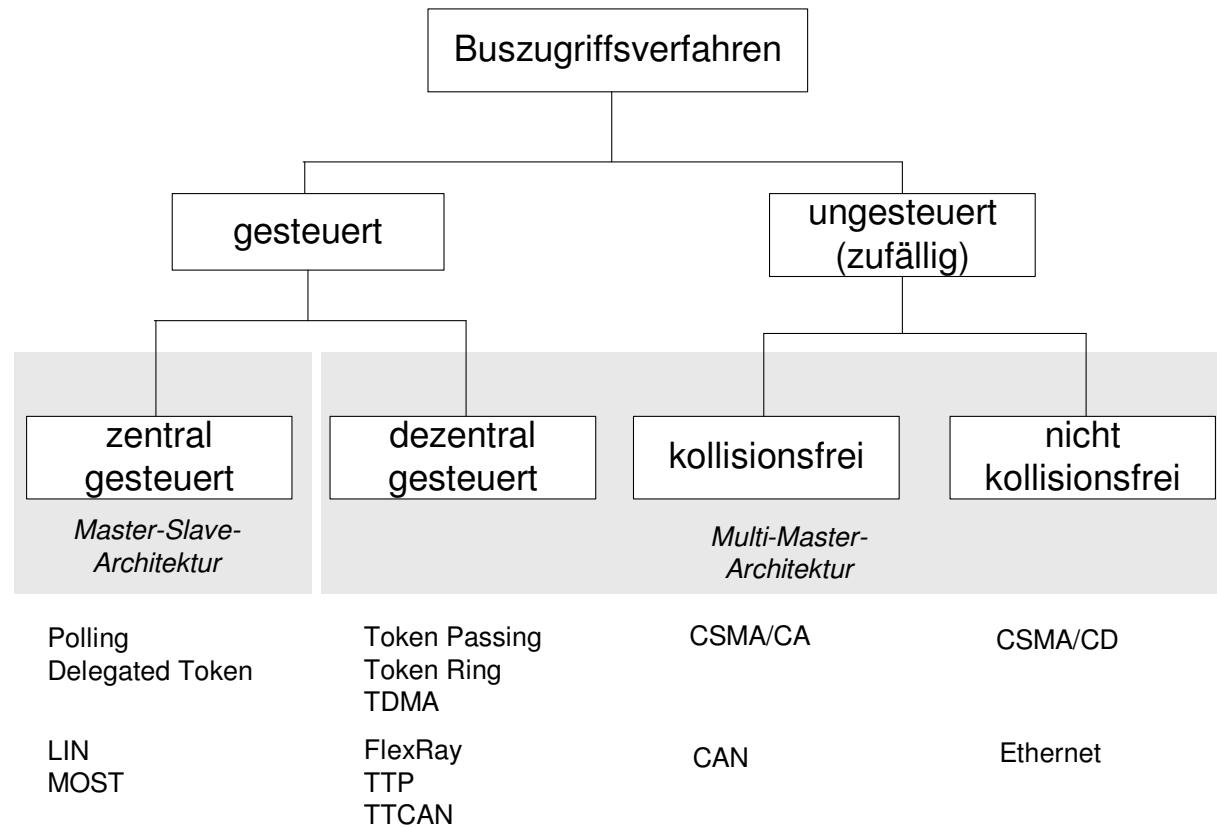
- Aufgaben
- XON/XOFF-Protokoll
- Kodierung (Quellen-, Kanalkodierung)

2. Buszugriffsverfahren und Ethernet

- Überblick
- CSMA-Protokolle
- Ethernet

Buszugriffsverfahren

- Bus als gemeinsam genutztes Medium
- Einteilung der Zugriffsverfahren



Zugriffsverfahren für Mehrfachzugriffskanäle

ALOHA und CSMA

■ **ALOHA** (Protokoll, das im ALOHAnet genutzt wurde)*

- Keine Prüfung des Kanals vor dem Senden
- Nicht so effektiv
- Varianten: slotted (feste Zuordnung von Zeitschlitten) und pure (beliebiges Senden)
- Pure ALOHA ist nicht kollisionsfrei

■ **CSMA**

- Prüfung des Kanals vor dem Senden
 - . Trägererkennungsprotokoll (Carrier Sense)
- Varianten: non-persistent (Kanal frei -> Senden) und p-persistent (Kanal frei → Senden mit WS p)
- CSMA ist nicht kollisionsfrei

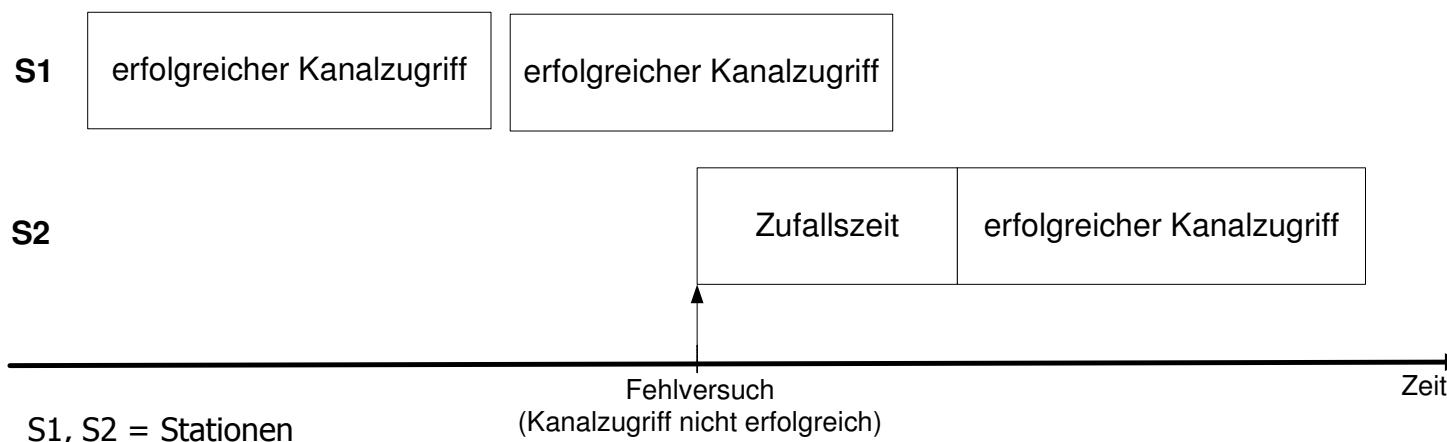
*) ALOHAnet = erstes Funk-basiertes Rechnernetz, nutzt ALOHA-Protokoll zur Verbindung der vielen Inseln um Hawaii

CSMA-Protokolle

■ Non-Persistent CSMA

- Kanal frei → Senden
- Kanal belegt → Zufällige Zeit warten, dann erneut versuchen

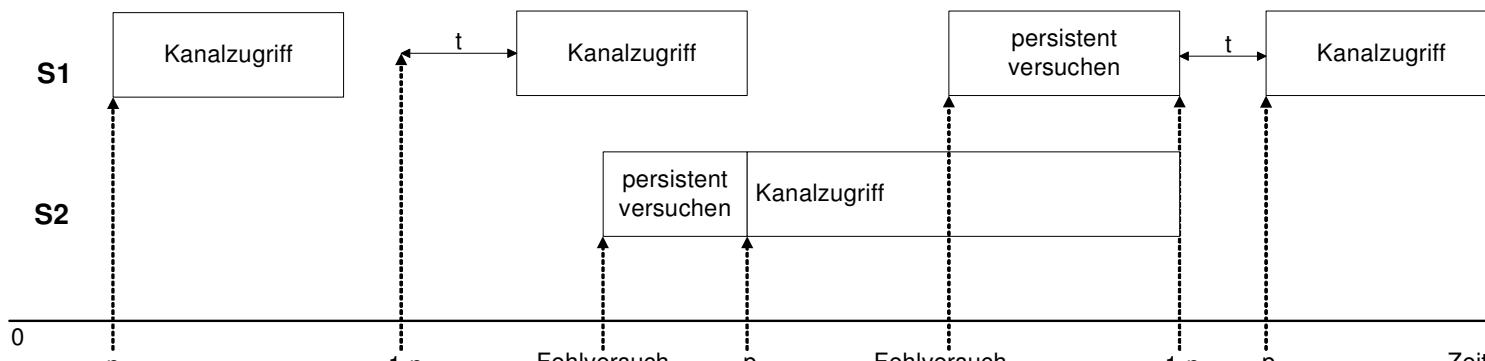
Non-persistent CSMA



CSMA-Protokolle

■ p-persistent CSMA

- Wenn Kanal frei ist, wird mit WS p gesendet und mit WS $1-p$ eine zufällige Zeit t gewartet und dann erneut gesendet
- Bei belegtem Kanal beobachtet Station zunächst den Kanal (siehe Station S2)



S1, S2 = Stationen

Ethernet

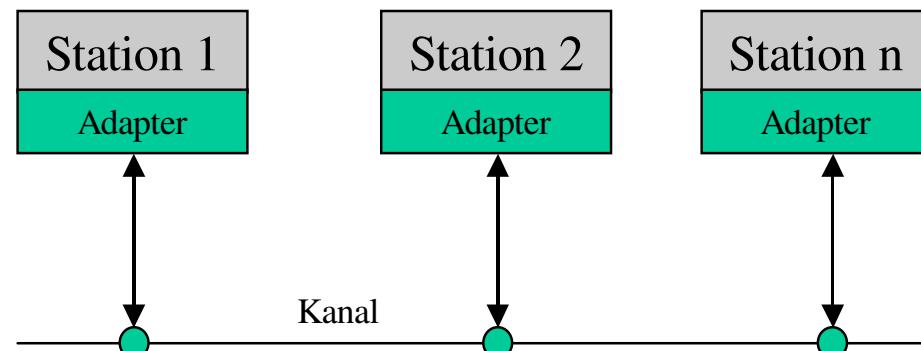
Überblick

- Ethernet wurde Anfang der 70er Jahre von **Bob Metcalfe** entwickelt und als **IEEE 802.3-Standard** bekannt
- Die Architektur basiert auf der Definition von Funktionen auf den beiden untersten Schichten des ISO/OSI-Referenzmodells für
 - die Festlegung der physikalischen Eigenschaften der benötigten Komponenten
 - die Zugriffsverfahren der Stationen auf das Netz und
 - den Aufbau der versendeten Nachrichten

Zugriffsverfahren

Mehrfachzugriffskanäle, Grundprinzip

- Kein zentraler Controller (Multi-Master-Architektur)
- Alle Stationen sind gleichberechtigt und entscheiden eigenständig
- Gesendete Signale pflanzen sich in beide Richtungen des Kanals fort
- Wettkampfverfahren erforderlich!
- Kollisionen möglich



Ethernet-Medienzugriffsverfahren

CSMA/CD, Grundprinzip (1)

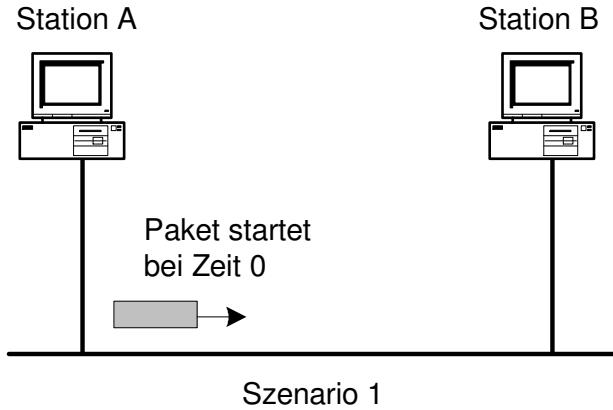
- CSMA/CD-Zugriffsverfahren auf das Medium (**dezentrale Steuerung**)
 - Medium wird von allen Stationen unabhängig abgehört, wenn Medium frei (keine Signalenergie) darf Station senden
 - **Kollision** möglich → Sendungen werden eingestellt
 - **Backoff:** Stationen warten eine bestimmte, zufällige Zeit → verhindert erneute Kollision
- Genaue Bezeichnung des Verfahrens: **1-persistent CSMA/CD mit exponentiellem Backoff:**
 - Bei freiem Medium wird sofort gesendet (1-persistent)
 - Bei Kollision wird zufällige Zeit gewartet (Rückzieher)
 - Nach jeder Kollision wird die Wartezeit bis zum 10. Versuch verdoppelt (binär exponentielles Wachstum)
 - Nach 16 Versuchen erfolgt Abbruch

Ethernet-Medienzugriffsverfahren

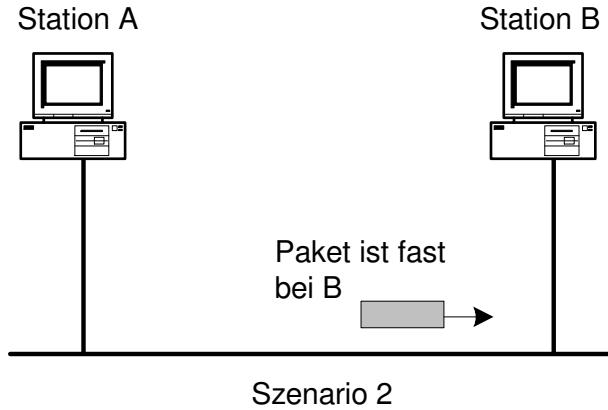
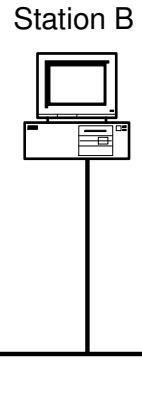
CSMA/CD, Grundprinzip (2)

- **MA** = „mehrfacher Zugriff“ von Rechnern auf ein Übertragungsmedium (Multiple Access)
- **CS** = „Befühlen des Mediums“: (Carrier Sense)
 - Sendewillige Station prüft, ob Kabel nicht gerade von einem anderen Rechner benutzt wird
 - Sendewillige Stationen hören den Bus ab und belegen ihn, wenn er frei ist (wenn keine andere Station bereits sendet)
- **CD** = Im **Kollisionsfall** Abbruch des Sendevorgangs und Wiederholung
- Stochastisches Verfahren
 - **für zeitkritische Anwendungen nicht geeignet**
 - **nicht deterministisch**

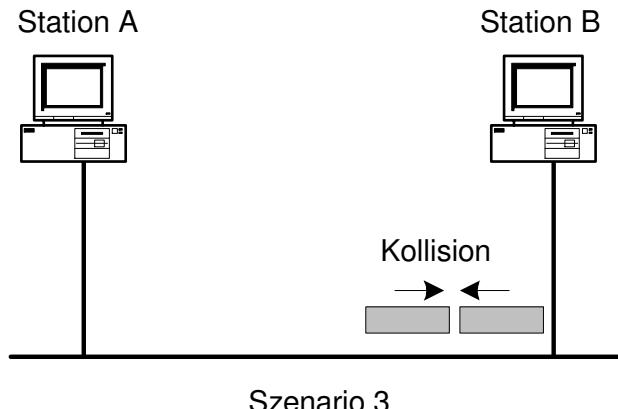
Kollisionen im Ethernet



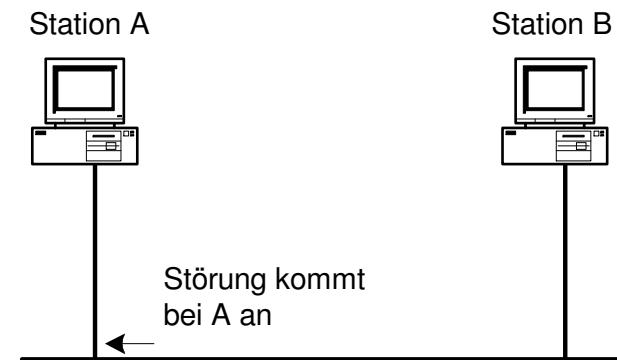
Szenario 1



Szenario 2

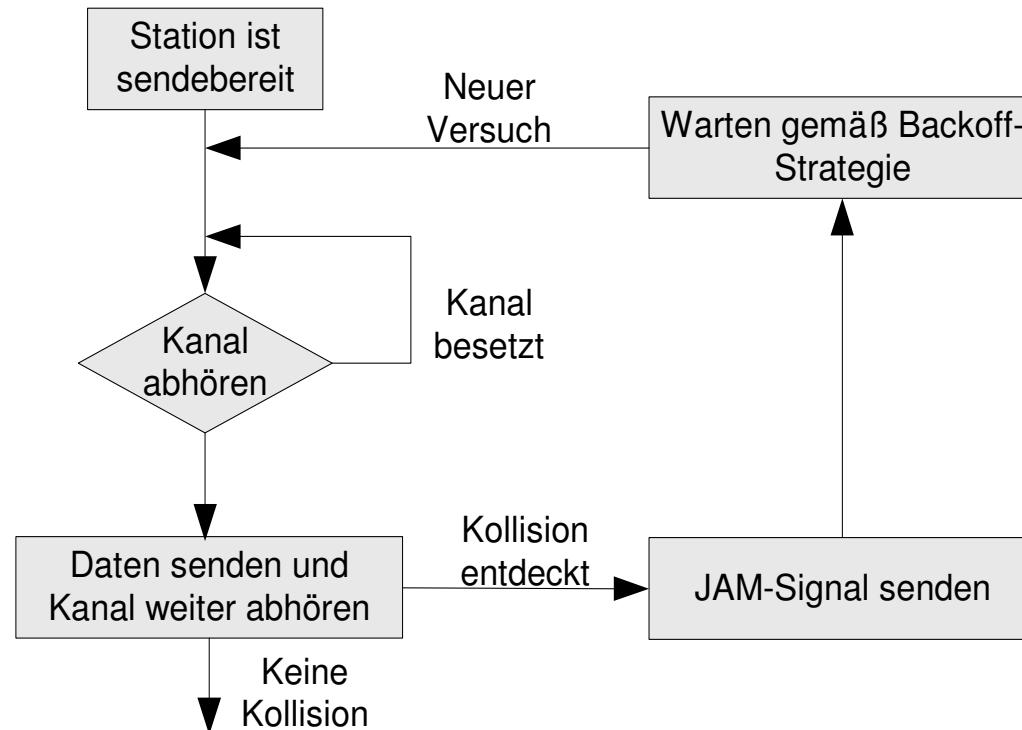


Szenario 3



Szenario 4

Ethernet: Ablauf

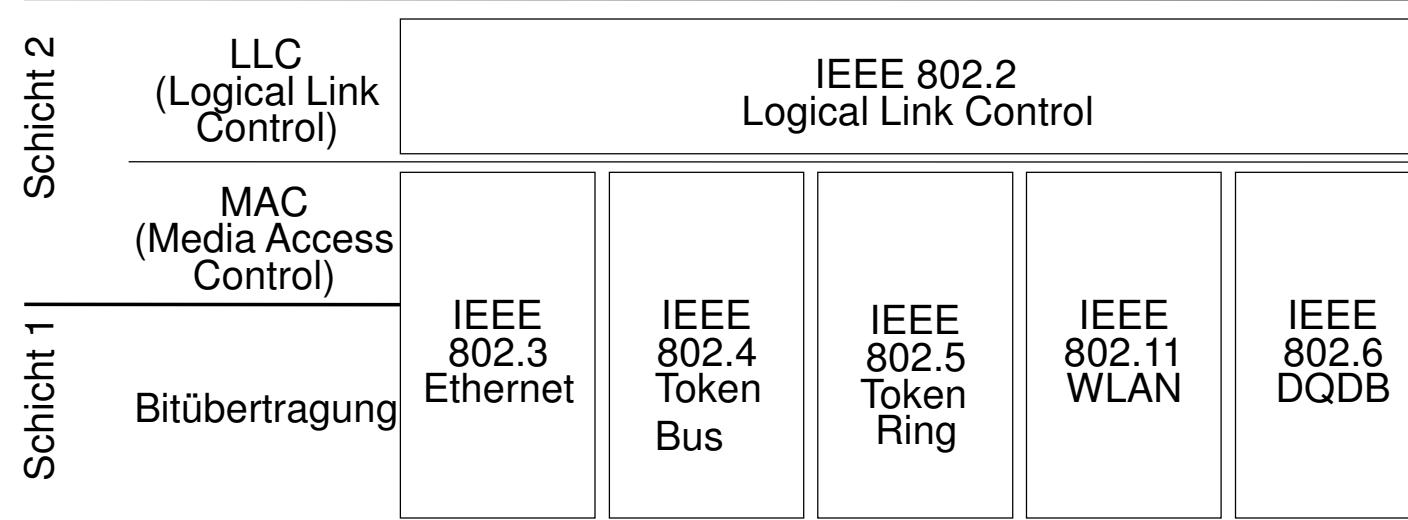


Backoff-Algorithmus:

- Algorithmus bestimmt nach einer Kollision eine Zeitspanne zum Warten, bevor sie einen neuen Sendevorschuss startet
- Die Zeitspanne ist ein Vielfaches von einem so genannten „Slot“, der z.B. 51,2 µs lang ist (je nach Ethernet-Typ, hier bei 10 Mbit/s)

Einordnung in den IEEE 802-Standard

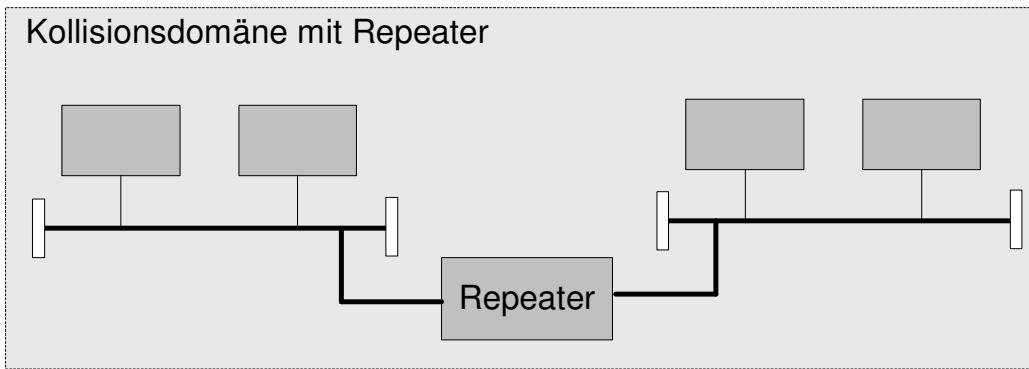
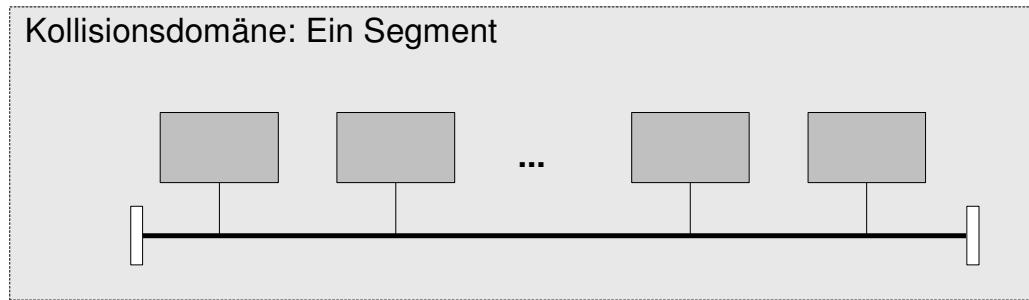
- Die Architektur basiert auf der Definition von Funktionen auf den beiden untersten Schichten des ISO/OSI-Referenzmodells für
 - die Festlegung der physikalischen Eigenschaften
 - die Zugriffsverfahren der benötigten Stationen auf das Netz
 - den Aufbau der versendeten Nachrichten
- Unterschiedliche Arbeitsgruppen der IEEE 802.3



Ethernet-Laufzeitbedingungen

- Mindestrahmenlänge erforderlich, um Kollisionen zu erkennen → Mindestrahmenlänge 64 Byte bei 10 Mbit/s
 - Damit der Sender bei maximalem Abstand zum Empfänger die Kollision noch erkennen kann
- Ethernet-Standard begrenzt die Entfernung zwischen zwei Knoten
- Signallaufzeit im Medium muss bedacht werden

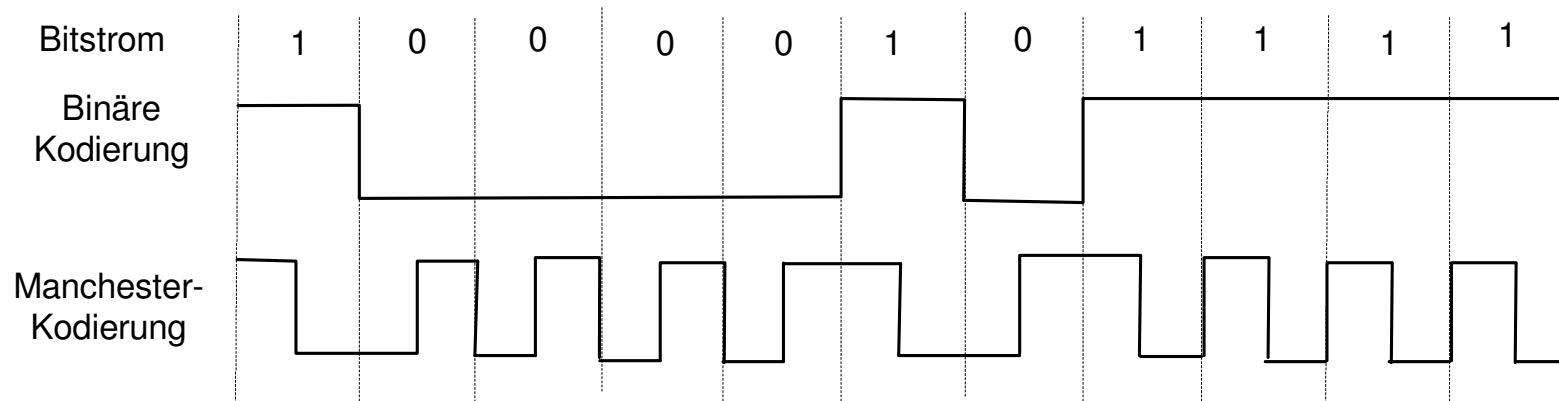
Kollisionsdomänen



Quelle: Stein (2004): Taschenbuch Rechnernetze und Internet,
Fachverlag Leipzig, S. 202

Basisbandübertragung und Manchesterkodierung

- Basisbandübertragung
 - Netzwerkadapter schiebt das digitale Signal direkt auf das Medium
 - Keine Verschiebung des Signals in ein anderes Frequenzband wie bei ADSL
- Leitungskodierung
 - Manchesterkodierung (ältere Ethernets) oder 4B5B-Kodierung (100 Mbit/s) bzw. 8B10B (1 Gbit/s) wird genutzt



Paketaufbau

■ Paketaufbau: Physikalischer Aufbau (MAC Frame)

- Die Struktur des Ethernet Pakets ist grundsätzlich für alle Übertragungsraten gleich: (vgl.:Riggert, 2001)

Präambel	7 Byte	Dient der Synchronisation der Station auf dem gemeinsamen Kabel
Start Frame Delimiter	1 Byte	SFD markiert den Anfang des Pakets
Zieladresse	6 Byte	Zur Identifikation des Empfängers: z. B. 00 00 0C 60 50 01 ₍₁₆₎
Quelladresse	6 Byte	HW-Adresse des Senders z. B. 00 06 7C 67 45 31 ₍₁₆₎
Pakettyp oder Längenfeld	2 Byte	IP 0800 (16) ARP 0806 ₍₁₆₎
Nutzdaten und Padding	0 Byte – 1500 Byte	Falls weniger als 46 Byte Nutzdaten, wird mit Füllbyte aufgefüllt (Padding)
Prüfsumme CRC	4 Byte	Cyclic Redundancy Check

- ARP=Address Resolution Protocol übersetzt die IP-Adresse eines Rechners in eine MAC-Adresse → siehe Internet-Protokolle

Einschub: Ethernet-Broadcast

- Broadcast wird von Ethernet unterstützt
- In IPv4: Limited Broadcast wird in Ethernet-LANs auf Ethernet-Broadcast abgebildet
- Nutzung der Zieladresse „FF FF FF FF FF FF FF₍₁₆₎“

Ethernet: Familie von LAN-Konzepten

- Ethernet ist eine **Familie** von LAN-Konzepten
- Gemeinsamkeiten:
 - Rahmenaufbau
 - Zugriffsverfahren (CSMA/CD, nicht mehr ab 10-Gbit-Ethernet)
- Topologie:
 - **Anfänglich:** Bustopologie mit Koaxialkabeln
 - **Danach:** Sterntopologie mit Twisted-Pair-Kabeln und Multiport-Repeater (Hubs)
 - **Heute:** Sterntopologie mit bidirektionalen, geschalteten Punkt-zu-Punkt-Verbindungen (Switches)
 - Vollduplex

Beispiele für Ethernet-Varianten

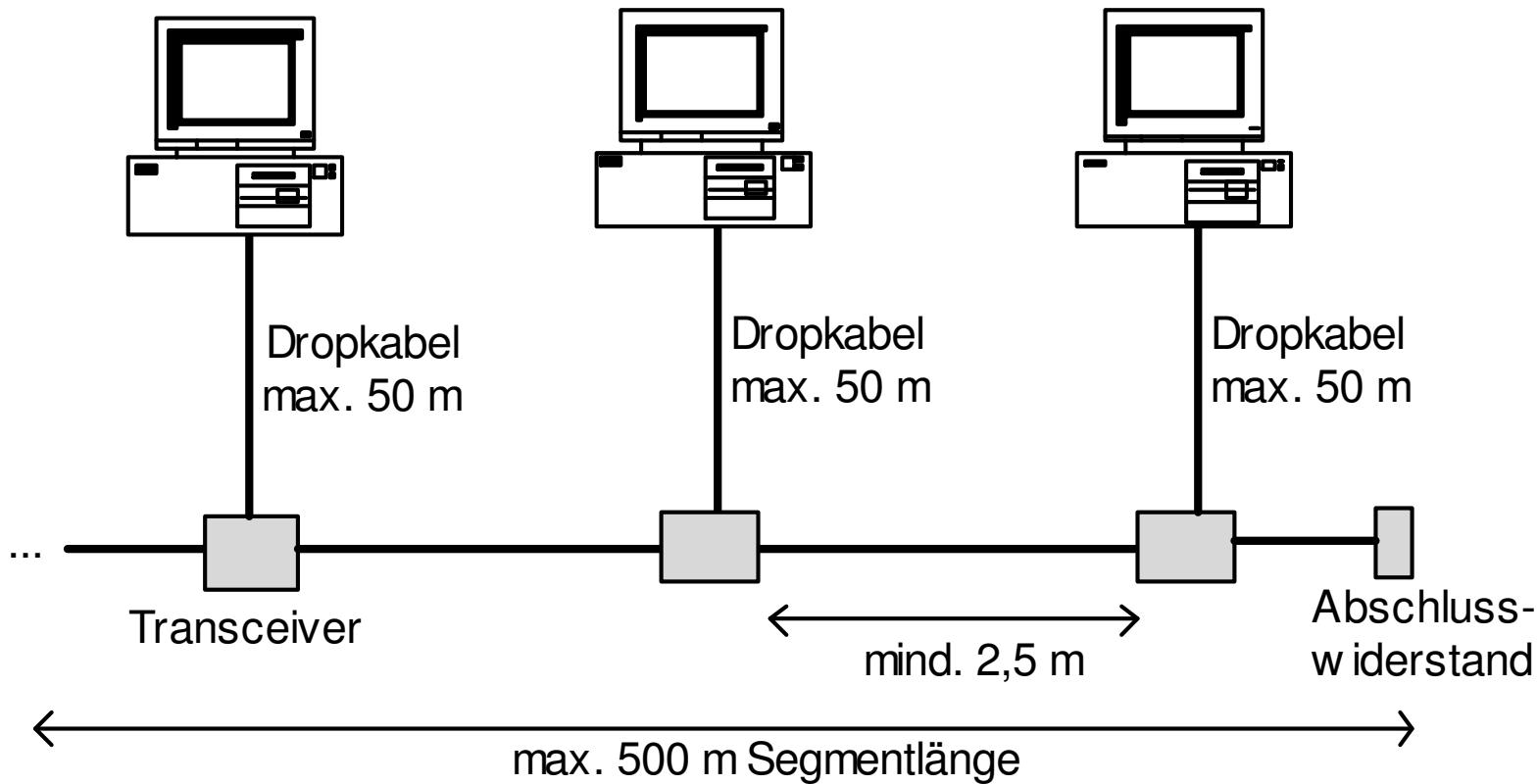
- 10Base5 (thicknet, Koaxialkabel)
- 10Base2 (cheapernet, Koaxialkabel)
- 10BaseT2 (2 Paare UTP Kat. 3,4 oder 5)
- 10BaseF (Glasfaser, 2 optische Fasern)
- 1000BaseT4 (4 Paare UTP Kat. 5 oder besser)
- 100BaseTX (2 Paare UTP Kat. 5 oder STP)

10Base5

Bustopologie

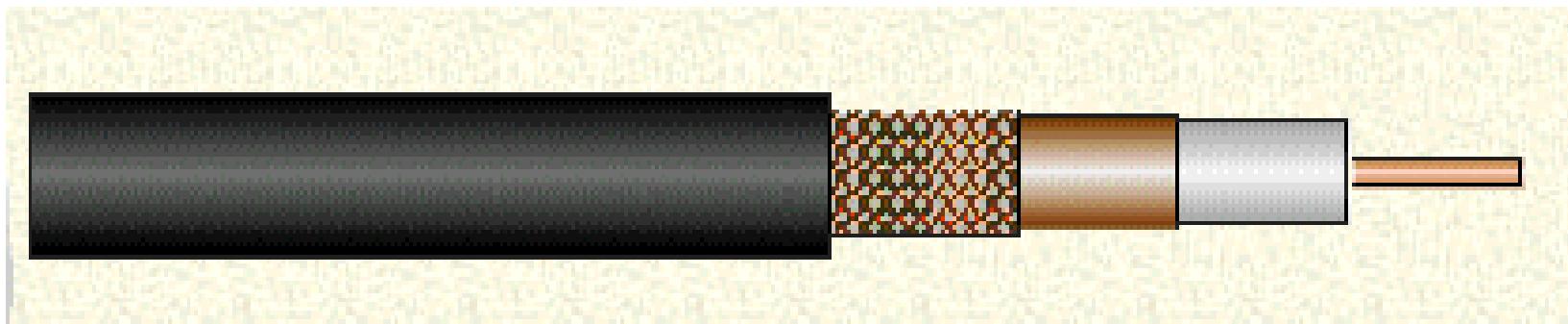
- Ausgangspunkt für Ethernet-Netzwerke („gelbes Kabel“)
- Alle Stationen sind über Transceiver an den Kanal gekoppelt
- Max. Segmentlänge: 500m
- Mindestabstand zwischen 2 Stationen: 2,5m
- Max. Anzahl der Stationen pro Segment: 100
- Max. Netzausdehnung: 2500 m (= 5 Segmente über 4 Repeater)
- Übertragungsgeschwindigkeit: 10 Mbit/s

10Base5 Bustopologie



10Base5

Verkabelung: Koaxialkabel



- 50 Ohm

10Base2

Bustopologie

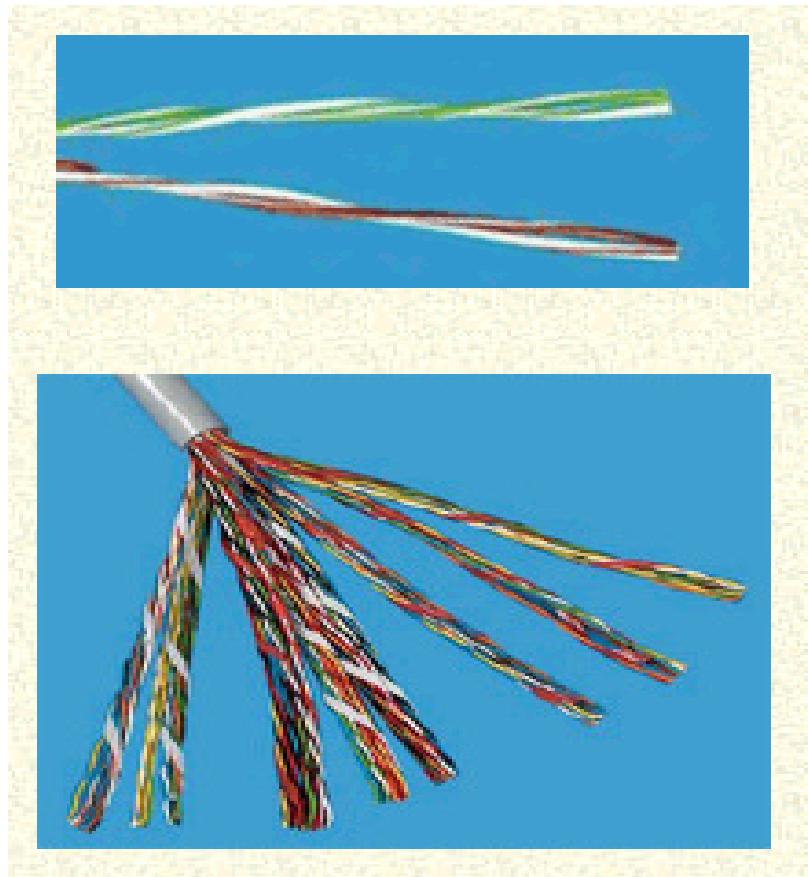
- Netze auf der Basis dieses Kabeltyps schließen die Stationen direkt an das Kabel gemäß BNC-Technik an
- BNC (Bayonet-Neill-Concelmann)-Stecker verbinden 2 Koaxialkabel
- Wegen Einsparung von Transceiver und Kabel preiswerter
- Max. Segmentlänge: 185m
- Mindestabstand zwischen 2 Stationen: 0,5m
- Max. Anzahl der Stationen pro Segment: 30
- Max. Netzausdehnung: 925 m (5 Segmente über 4 Repeater)
- Übertragungsgeschwindigkeit: 10 Mbit/s
- **Heute: Nicht mehr relevant!**

Fast Ethernet: 100BaseT,..., 100BaseFX

- Varianten: FX, TX, ...:
 - F → Glasfaserverkabelung
 - T → Twisted-Pair-Verkabelung
 - TX nutzt 2 Doppeladern (für Etagenverkabelung)
 - FX nutzt 2 Multimode-Fasern (Sekundärverkabelung)
- Alle Varianten verwenden eine **Sterntopologie**
- Zugriffsverfahren und Rahmenformat nach 802.3
- Segmentlänge: **100 m**
- Netzwerkausdehnung: 200 m, bei FX: 400 m
- Übertragungsgeschwindigkeit: **100 Mbit/s**
- **Vollduplex-Unterstützung → 200 Mbit/s**

Verkabelung für 10BaseT Twisted Pair

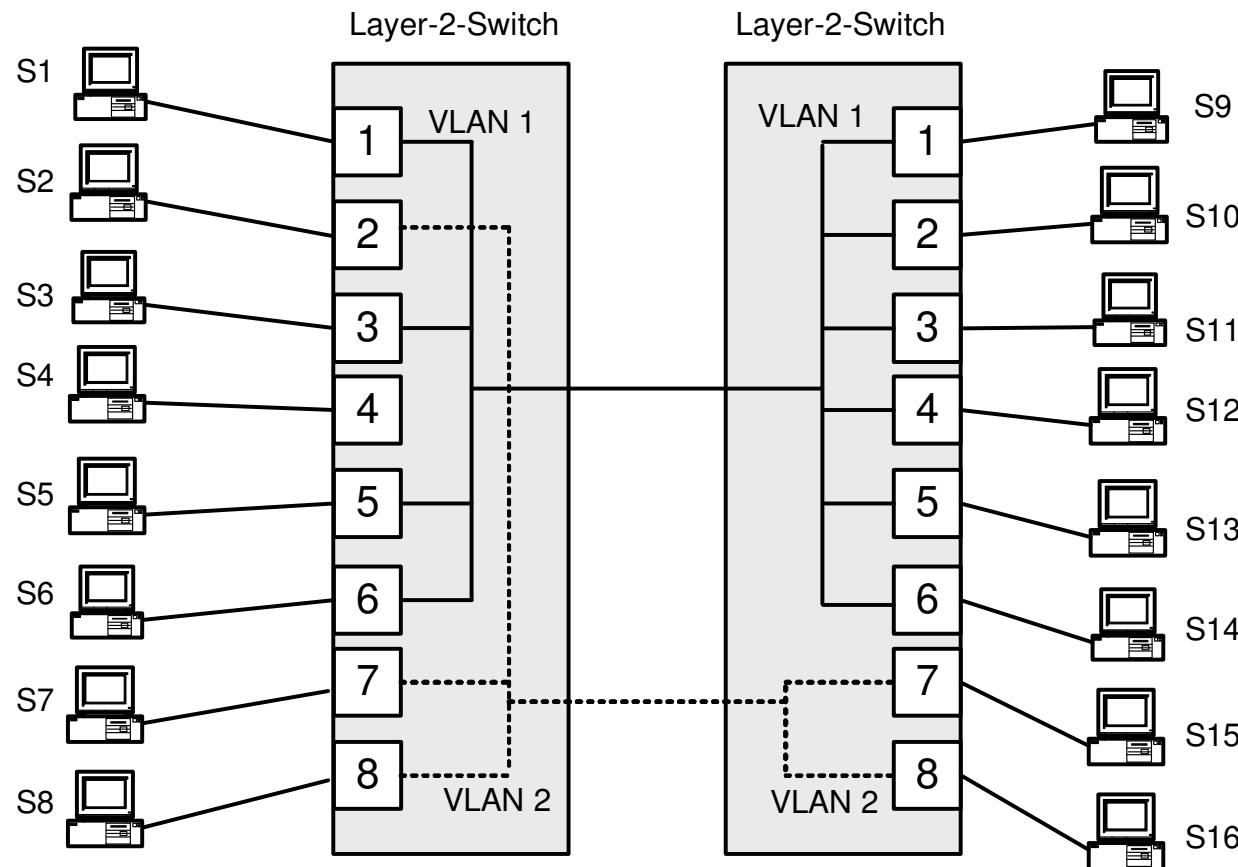
- Twisted Pair ist die generelle Bezeichnung für Kupferkabel mit einem oder mehreren verdrillten Leitungspaaren;
- Fast alle Dienste benötigen zur Signalübertragung 2 Paare (4 Adern):
 - 1 Paar für das Senden
 - 1 Paar für das Empfangen



Gigabit- und 10Gigabit-Ethernet Varianten

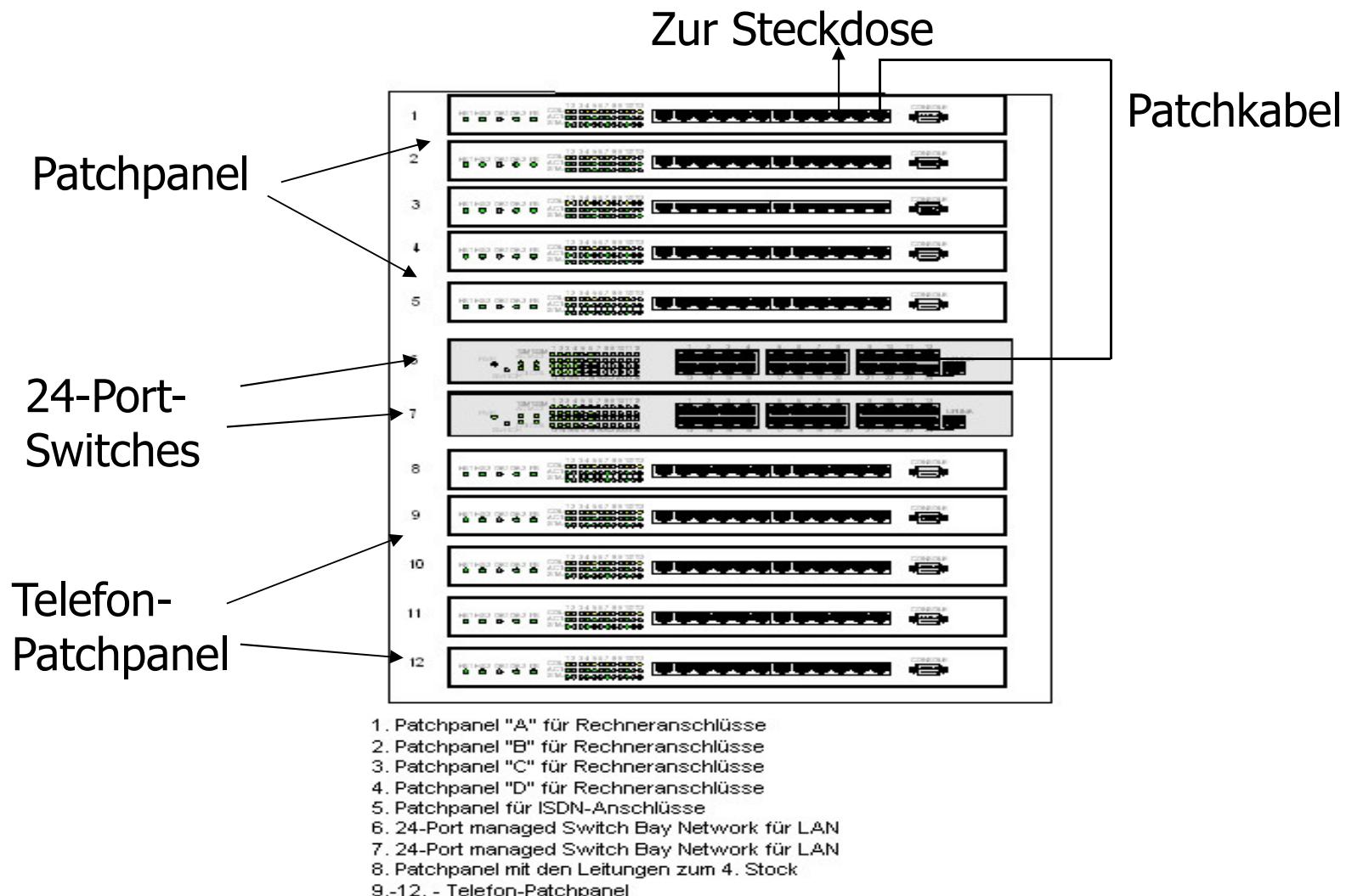
- 1000BaseT (4 Paare UTP Kat. 5, Distanz: 100 m)
- 1000BaseSX (Glasfaser, Distanz: 10 km)
- 10Gbase-LR (4 Paare UTP Kat. 5 oder besser)
- ...

Ethernet: LAN-Switching und VLAN



Strukturierte Verkabelung

Beispiel: Verteilerkasten



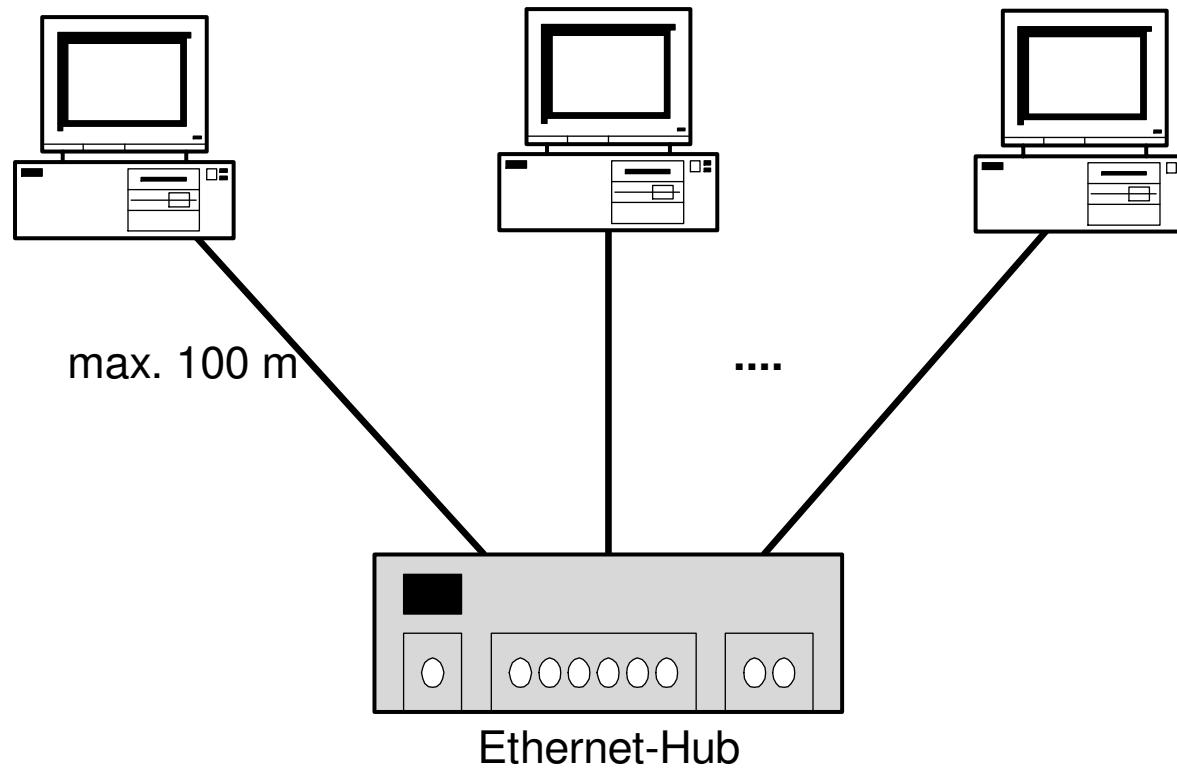
Switch

- Arbeitet auf der Schicht 2 und verbindet mehrere Segmente
- Exklusive Leitung je Port möglich:
 - Jeder Port ist eine eigene Kollisionsdomäne
 - Verzicht auf „shared Medium“
 - Keine Kollisionsbehandlung mehr erforderlich
 - Trotzdem noch CSMA/CD → eigentlich nicht mehr notwendig
- MAC-Schicht hat zusätzliche Flusssteuerung
 - Empfänger sendet Pausenrahmen zur Vermeidung von Pufferüberläufen im Switch
- Ein Switch kann in einem Ethernet-LAN verschiedene Gruppen schalten.
 - Z.B. können 100 Mbit/s-Segmente mit 10 Mbit/s-Segmente verbunden werden

Hub

- Verbindet mehrere Segmente eines LANs
- Besitzt mehrere Ports
- Kommt ein Paket an einem Port an, wird es an alle anderen Ports weitergeleitet
- Ein passiver Hub überträgt Daten von einem Port an alle anderen
- Ein intelligenter Hub beinhaltet Features, die es dem Administrator ermöglichen, den Verkehr des Hub zu überwachen und jeden Port im Hub zu konfigurieren
- Ein switching Hub liest die Zieladresse und gibt das Paket an den richtigen Port weiter

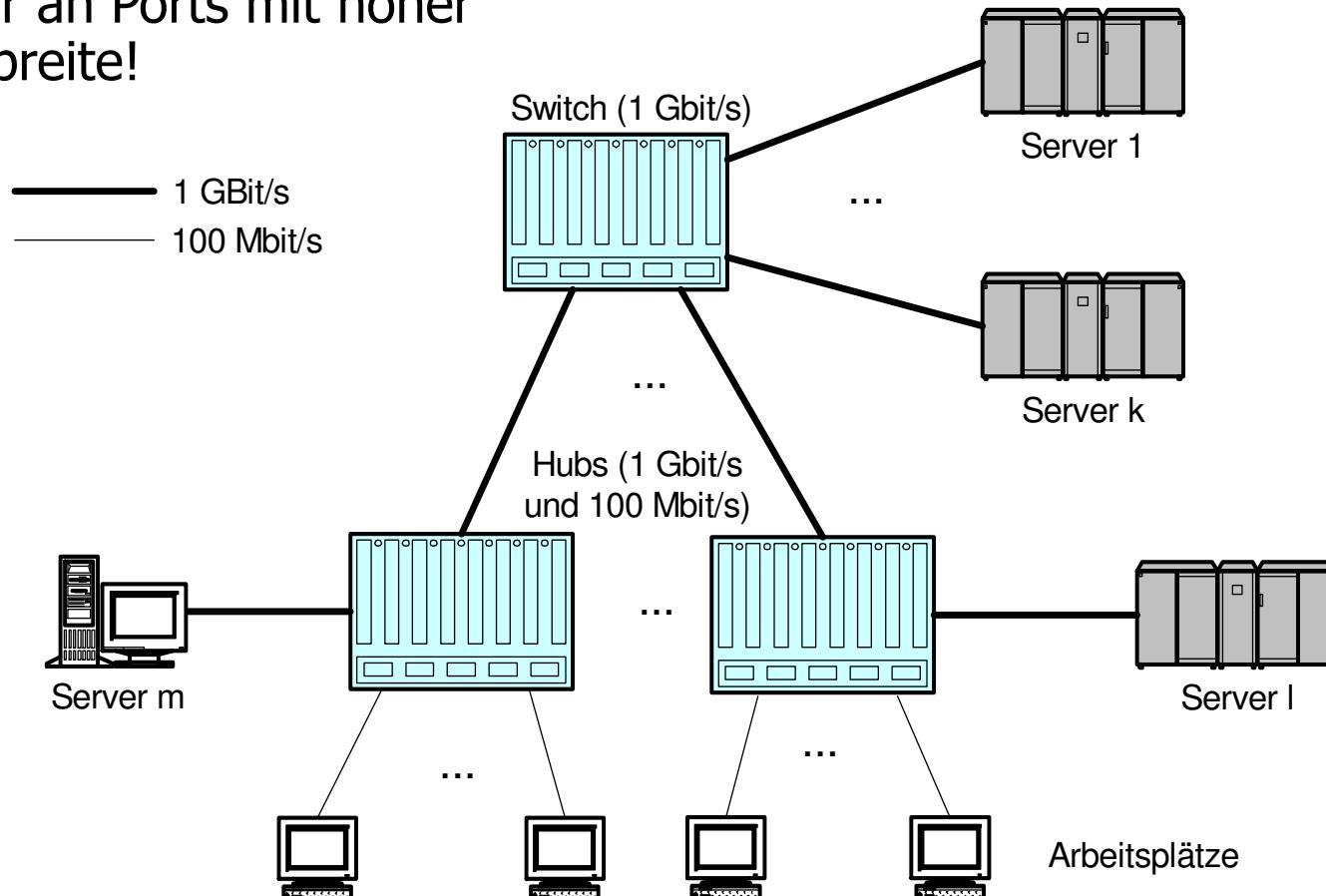
Hub



Switches und Hubs

Anwendungsbeispiel

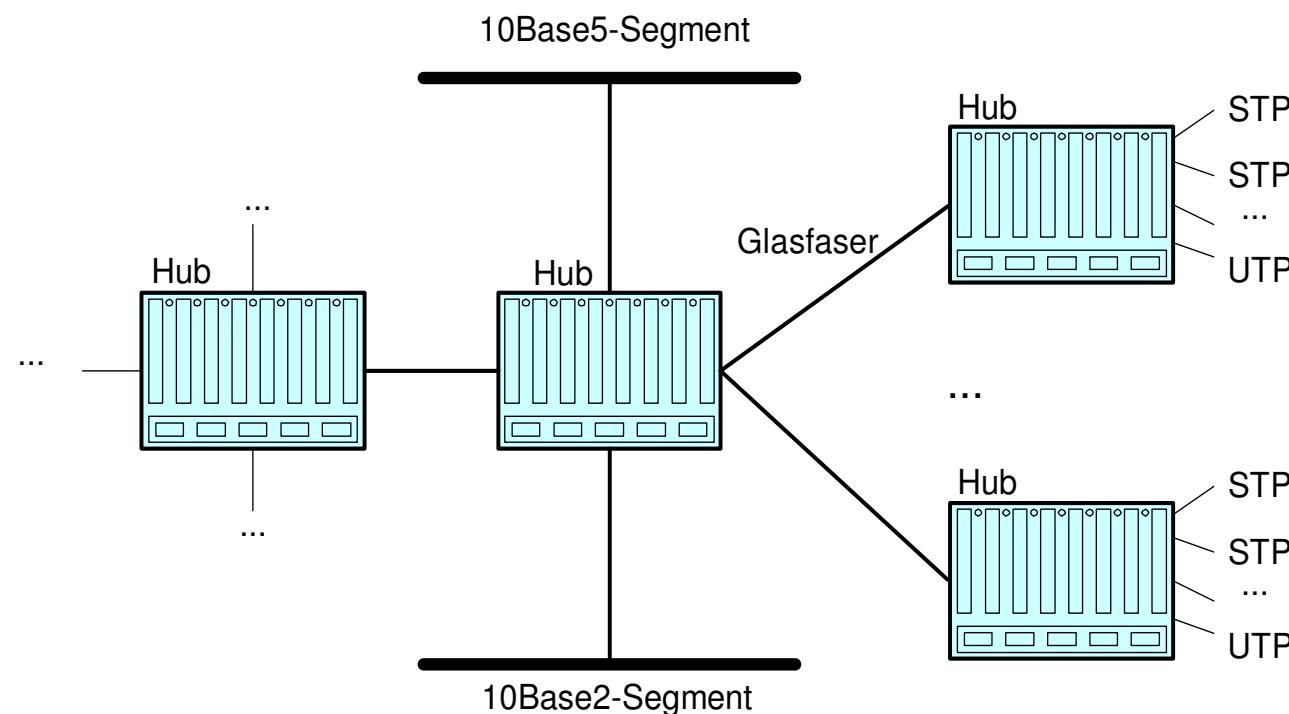
Server an Ports mit hoher Bandbreite!



Switches und Hubs

Hub als Sternkoppler, Beispiel

Sterntopologie



Rückblick: That's it!

1. Sicherungsschicht

- Aufgaben
- XON/XOFF-Protokoll
- Kodierung (Quellen-, Kanalkodierung)

2. Buszugriffsverfahren und Ethernet

- Überblick
- CSMA-Protokolle
- Ethernet

Datenkommunikation

Transportzugriff

Wintersemester 2012/2013

Einordnung

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

1. Konzepte

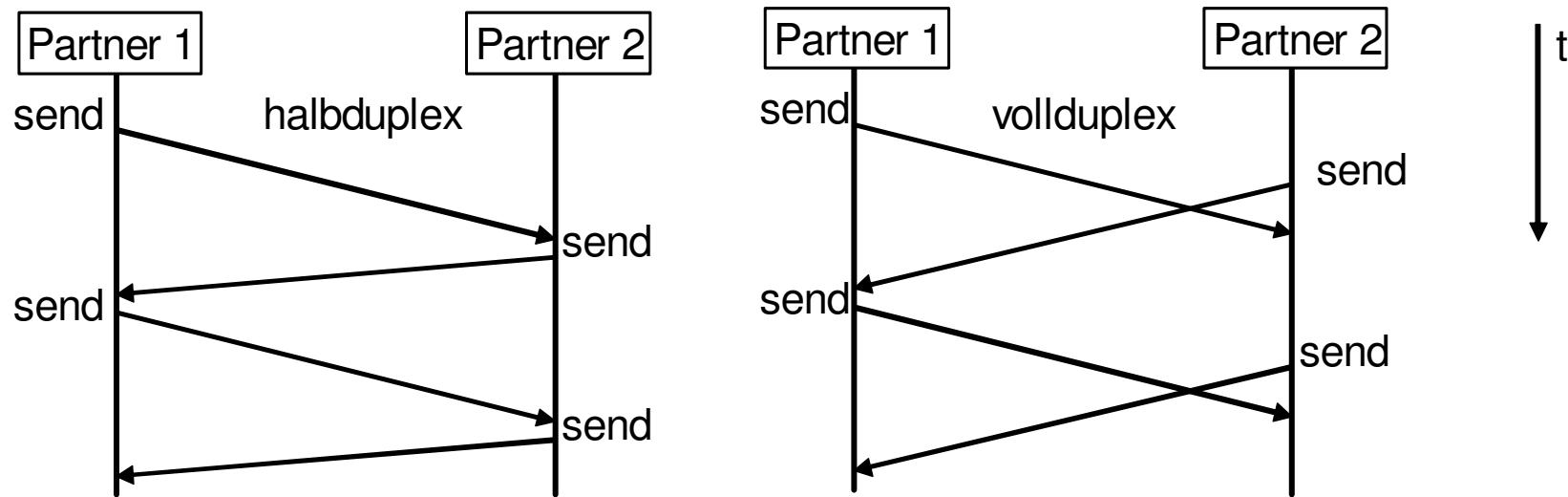
- Halbduplex, vollduplex, Empfängeradressierung
- Kommunikationsformen (synchron vs. asynchron)
- Ablauf der Kommunikation
- Fehlersemantiken

2. Socket-Programmierung

- TCP-Sockets
- Datagram-Sockets

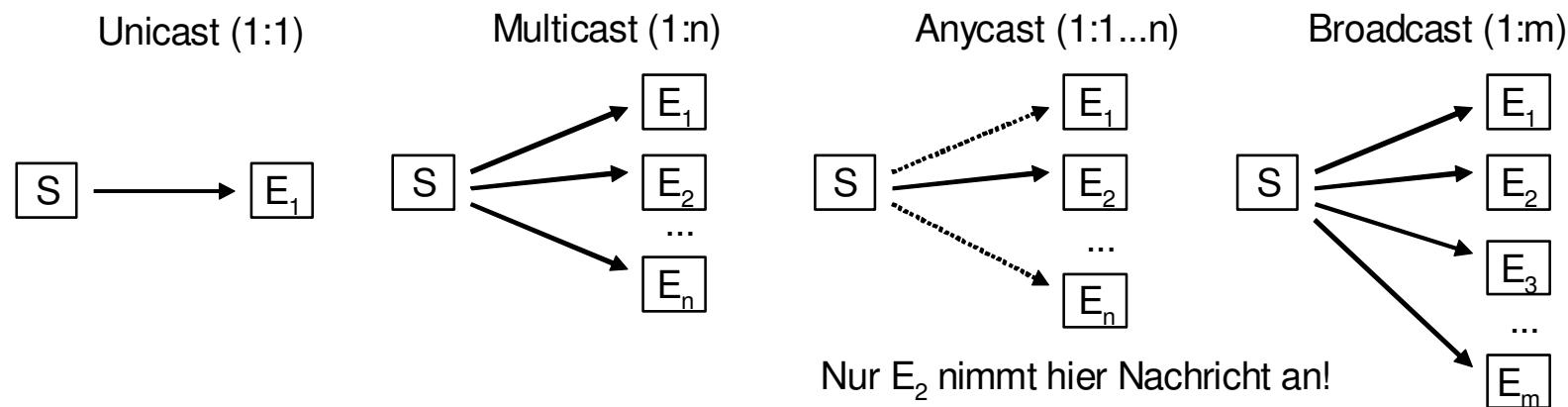
Halb- und vollduplex

- **Halbduplex:** Nur einer der Partner sendet zu einer Zeit
- **Vollduplex:** Beide Partner können unabhängig voneinander senden



Empfängeradressierung

- Unicast: Nur ein Empfänger wird adressiert
- Alle anderen Varianten adressieren mehrere Empfänger



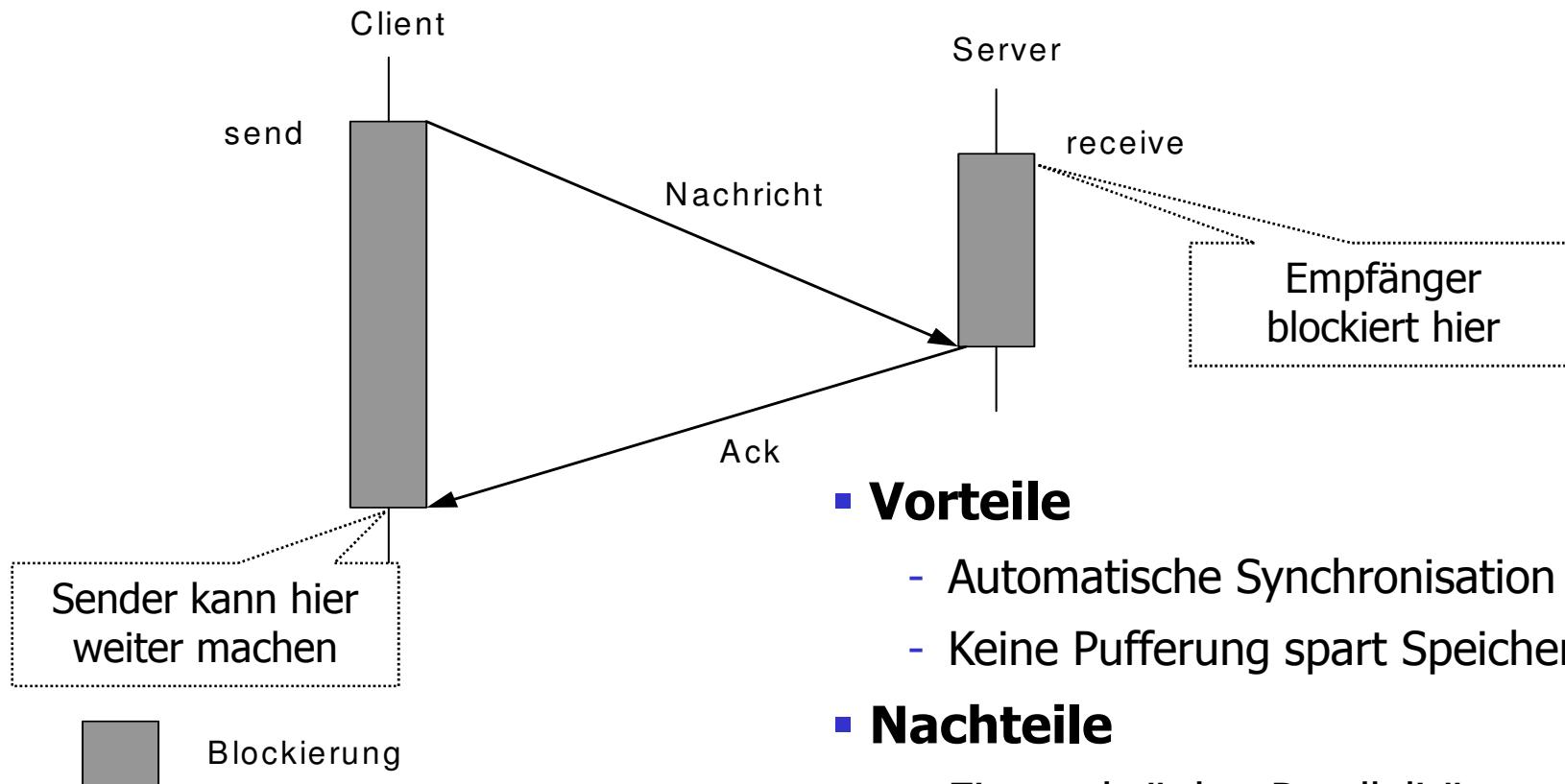
Blockierung

- Beim Nachrichtenaustausch unterscheidet man:
 - Synchrone Kommunikation
 - Asynchrone Kommunikation
- Synchron bedeutet:
 - **Blockierend**
 - Der Sender wartet, bis eine Methode send **mit einem Ergebnis** zurückkehrt
- Asynchron bedeutet:
 - **Nicht blockierend**
 - Der Sender kann weiter machen, wenn die Nachricht mit einer Methode send **in einen Transportpuffer** gelegt wurde

Pufferung von Nachrichten

- Puffer für ankommende Nachrichten werden **in den Protokollinstanzen** (meist im Betriebssystemkern) verwaltet
- Die Instanzen **kopieren** die Nachrichten in den Adressraum der empfangenden Anwendungsprozesse
- Pufferspeicher müssen **verwaltet** werden (→ Overhead)
- Pufferspeicher **benötigen Adressraum** (Speicher)
- Pufferspeicher sind **begrenzt** (→ evtl. Verwerfen von Nachrichten, wenn sie voll sind)

Blockierung - Synchrone Kommunikation



■ Vorteile

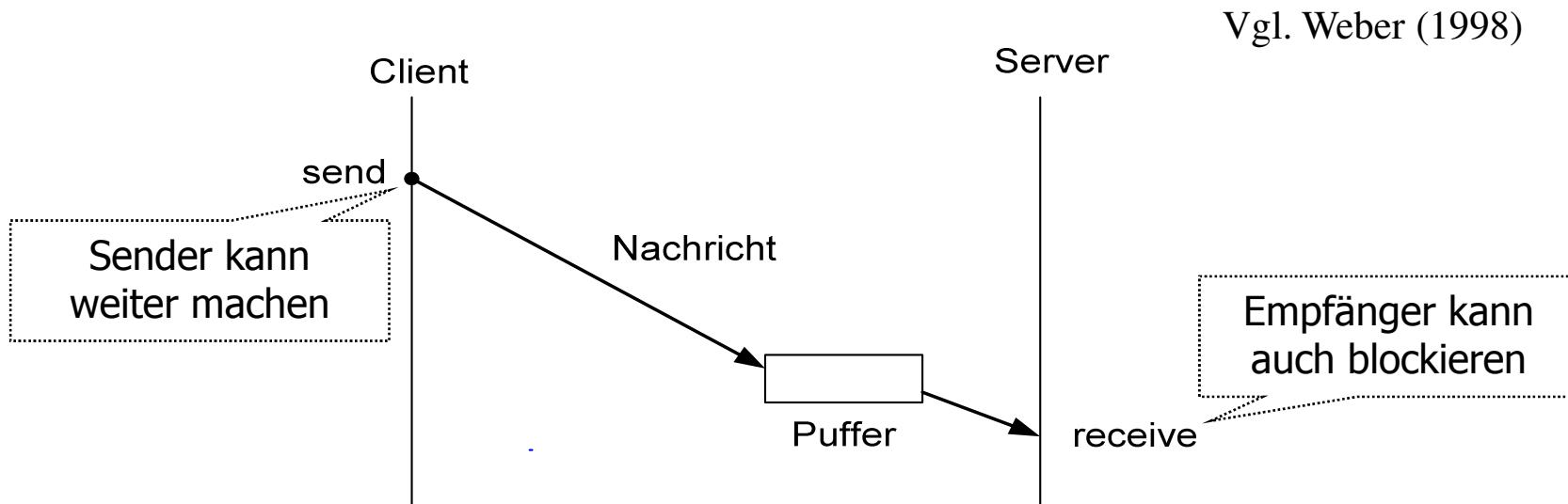
- Automatische Synchronisation
- Keine Pufferung spart Speicher

■ Nachteile

- Eingeschränkte Parallelität
- Evtl. langes Blockieren, wenn Empfänger keine Zeit hat

Vgl. Weber (1998)

Blockierung - Asynchrone Kommunikation



■ Vorteile

- Zeitliche Entkopplung
- Bessere Parallelarbeit möglich
- Ereignisgesteuerte Kommunikation möglich

■ Nachteile

- Zwischenpufferung notwendig
- Puffer voll führt trotzdem zum Blockieren wegen gesicherter Übertragung

Kommunikationsformen

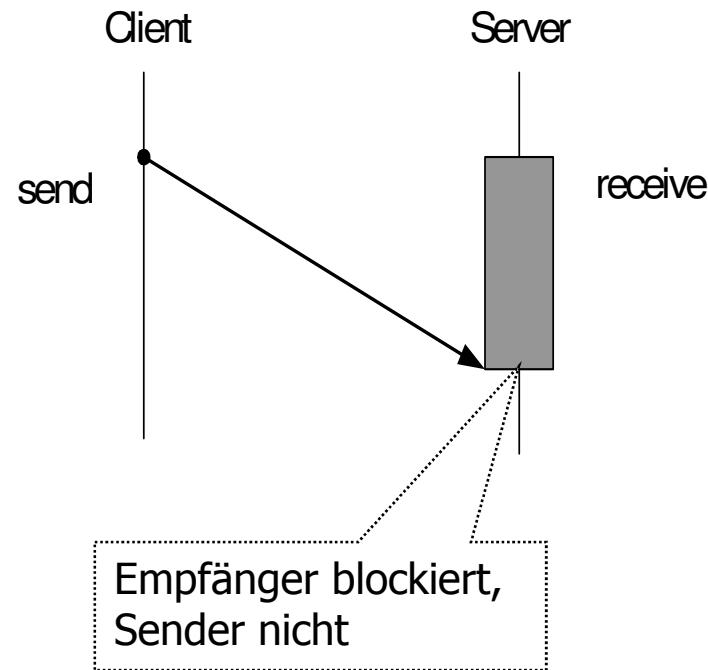
- Man unterscheidet weiterhin
 - **Meldungsorientierte** Kommunikation
 - Einwegnachrichten ohne Antwort
 - **Auftragsorientierte** Kommunikation
 - Request und Response (**mit Ergebnis**)
 - Entfernter Dienstauftrag

	asynchron	synchron
meldungsorientiert	Datagramm	Rendezvous
auftragsorientiert	asynchroner entfernter Dienstauftrag	synchroner entfernter Dienstauftrag

Vgl. Weber (1998)

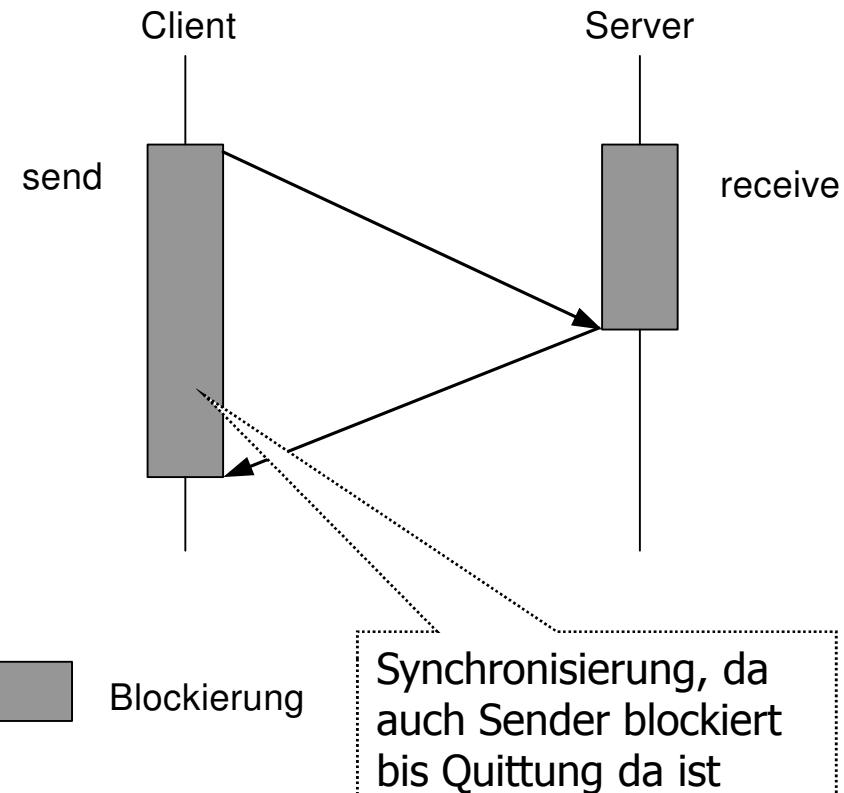
Meldungsorientiert Kommunikation

Datagramm



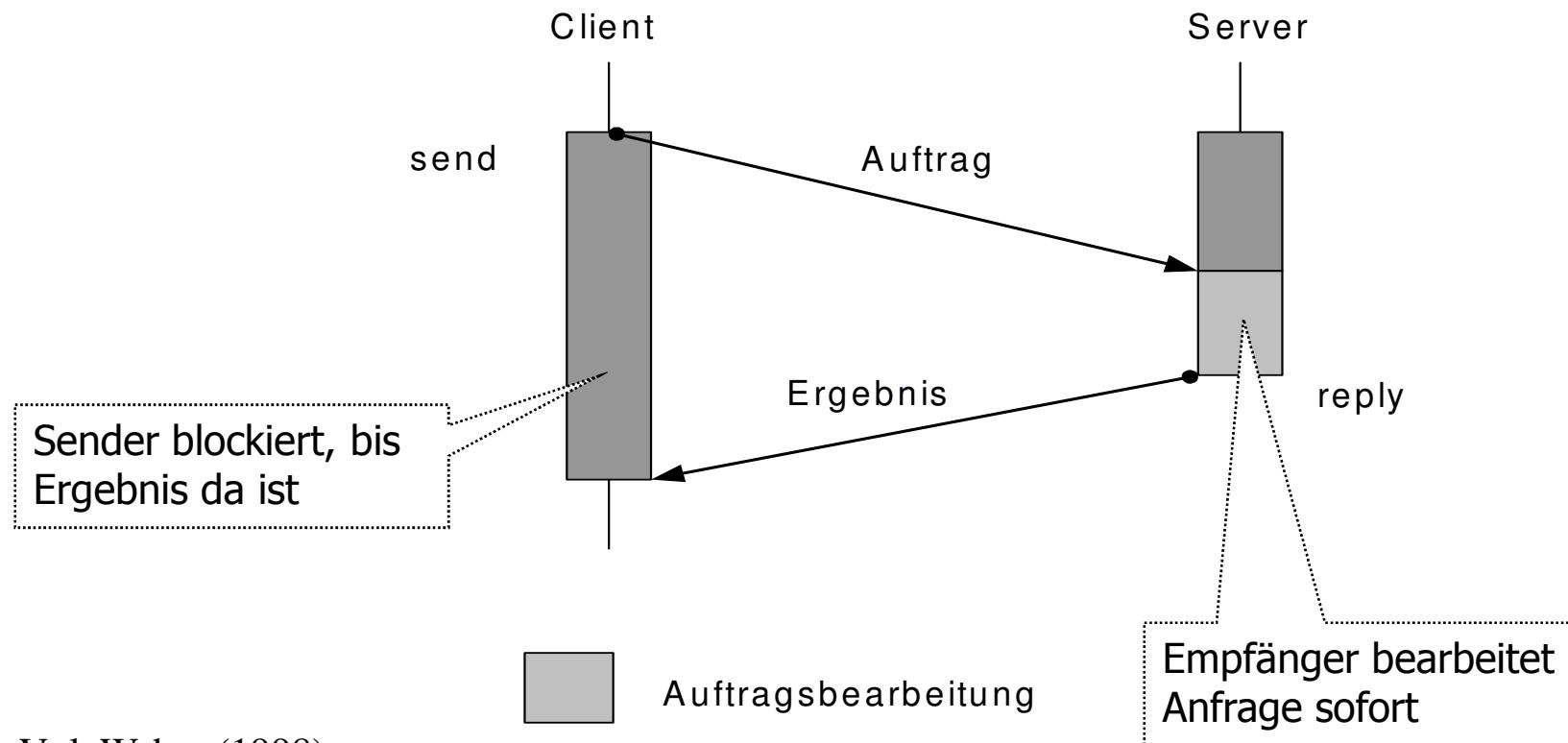
Vgl. Weber (1998)

Rendezvous



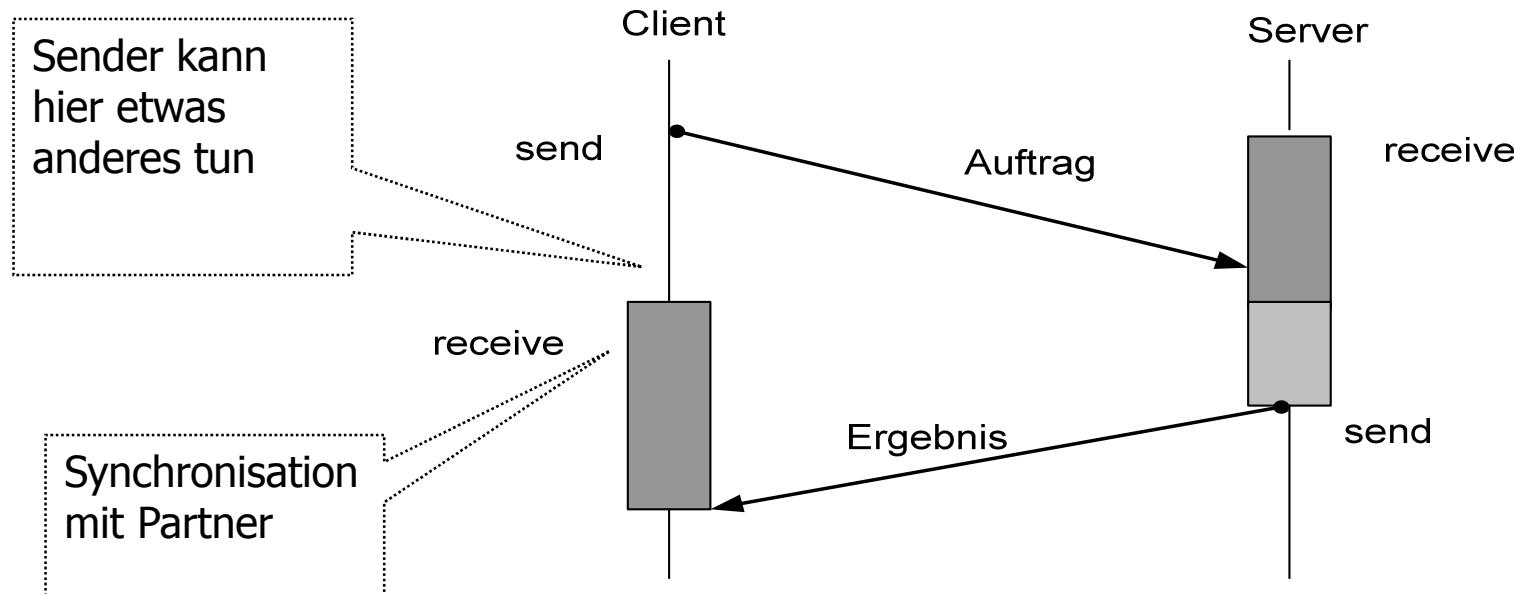
Auftragsorientierte Kommunikation

- Synchroner entfernter Dienstaufruf
 - Mit Rendezvous verwandt



Auftragsorientierte Kommunikation

- Asynchroner entfernter Dienstaufruf



Vgl. Weber (1998)

Diskussion

- Wie kann man als Entwickler einer Kommunikationsanwendung **Threads** als Parallelisierungstechnik einsetzen?

Fehlersemantiken (1)

- Es gibt viele **Fehlerursachen**
 - Netzwerkfehler
 - Sender fällt vor Empfang des Ergebnisses aus (→ verwaiste Aufträge, Orphans)
 - Empfänger fällt während der Bearbeitung eines Requests aus
 - ...
 - Ein Ausfall ist **zu jeder Zeit** möglich
 - Das Kommunikationssystem kann sich hier je nach Realisierung unterschiedlich verhalten
- **Verschiedene Fehlersemantiken** möglich

Fehlersemantiken

- Lokal: Alles fällt komplett aus oder es läuft
- Verteilt: Verschiedene Ausfallsituationen zu betrachten

Fehlerklasse	Fehlerarten			
	Fehlerfreier Ablauf	Nachrichtenverlust	Ausfall des Servers	Ausfall des Clients
Maybe	Ausführung: 1 Ergebnis: 1	Ausführung: 0/1 Ergebnis: 0	Ausführung: 0/1 Ergebnis: 0	Ausführung: 0/1 Ergebnis: 0/1
At-Least-Once	Ausführung: 1 Ergebnis: 1	Ausführung: >= 1 Ergebnis: >= 1	Ausführung: >= 0 Ergebnis: >= 0	Ausführung: >= 0 Ergebnis: 0
At-Most-Once	Ausführung: 1 Ergebnis: 1	Ausführung: 1 Ergebnis: 1	Ausführung: 0/1 Ergebnis: 0/1	Ausführung: 0/1 Ergebnis: 0
Exactly-Once	Ausführung: 1 Ergebnis: 1	Ausführung: 1 Ergebnis: 1	Ausführung: 1 Ergebnis: 1	Ausführung: 1 Ergebnis: 1

Nach Schill (2007)

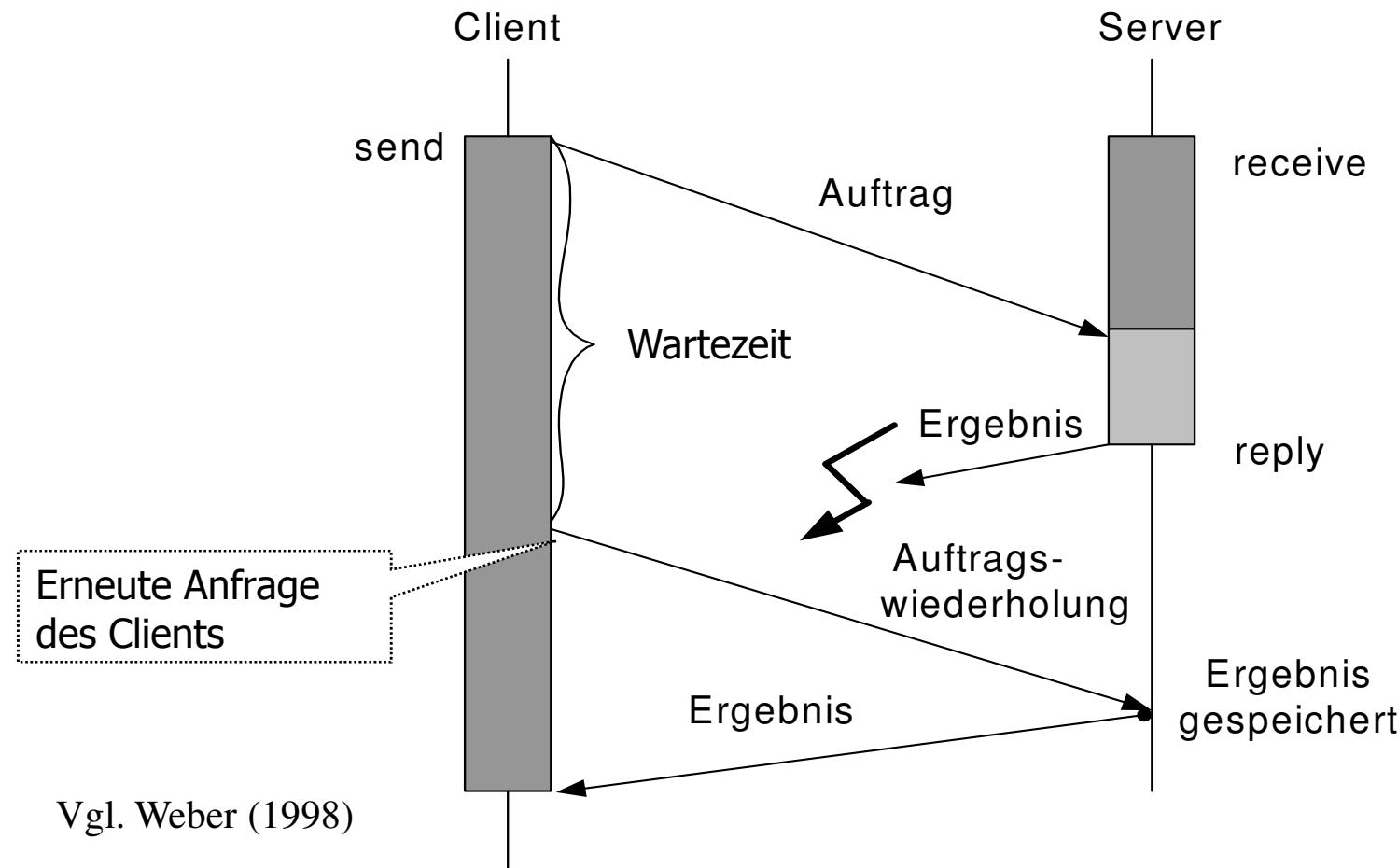
Wiederanlauf und Rücksetzmechanismen vorhanden → Transaktionen

Fehlersemantiken (3)

- Beispiel für eine Fehlersituation mit entsprechender Reaktion
 - Der Client erhält Antwort vom Server nicht und reagiert mit einem Timeout
 - Die Antwort des Servers ging verloren
 - Der Server speichert die Antwort
 - Der Client wiederholt den Auftrag nach dem Timeout
 - Der Server kann nun das gespeicherte Ergebnis senden
- Implementierungsaufwand!

Fehlersemantiken (4)

■ Möglicher Ablauf im Beispiel



Überblick

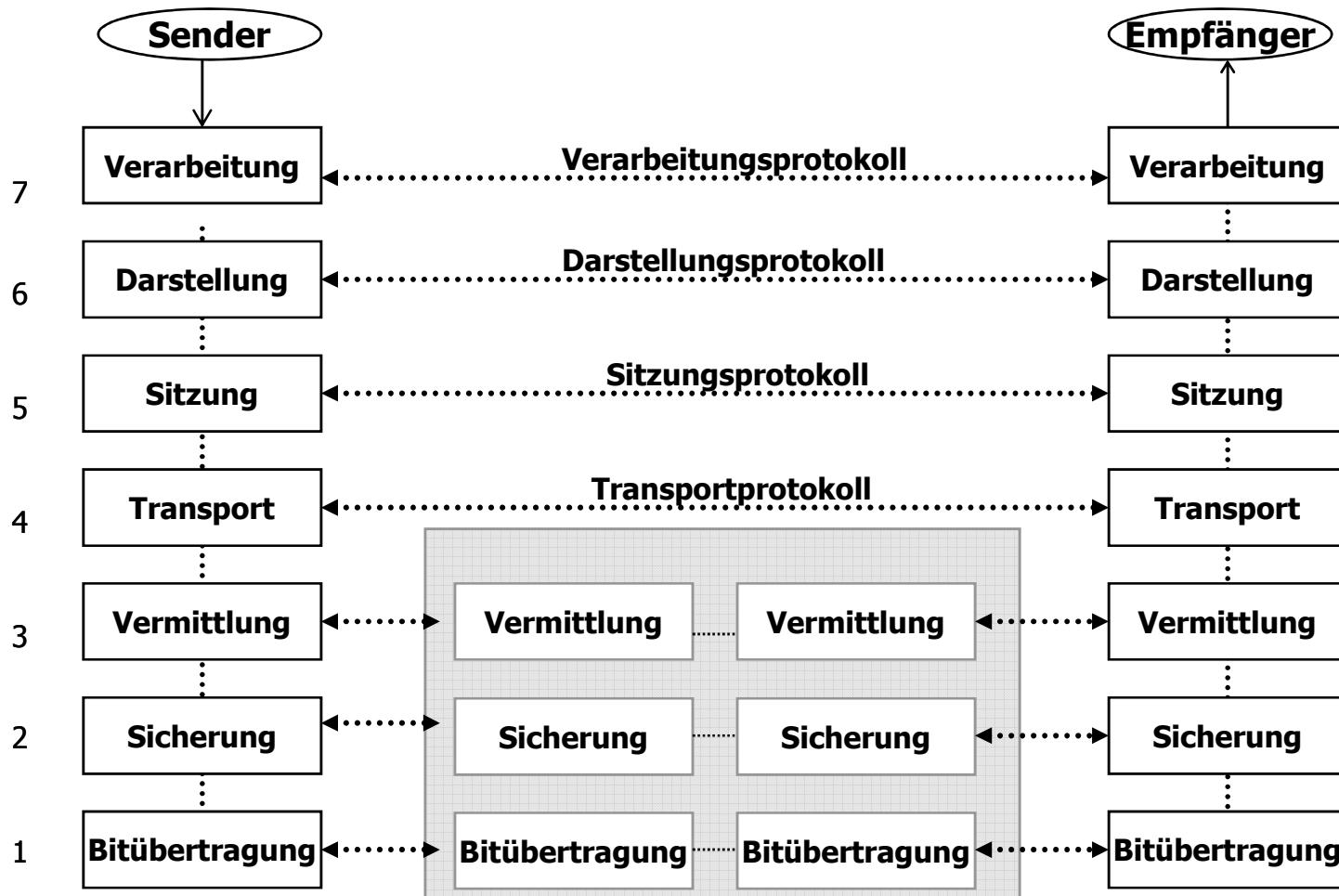
1. Konzepte

- Halbduplex, vollduplex, Empfängeradressierung
- Kommunikationsformen
- Ablauf der Kommunikation
- Fehlersemantiken

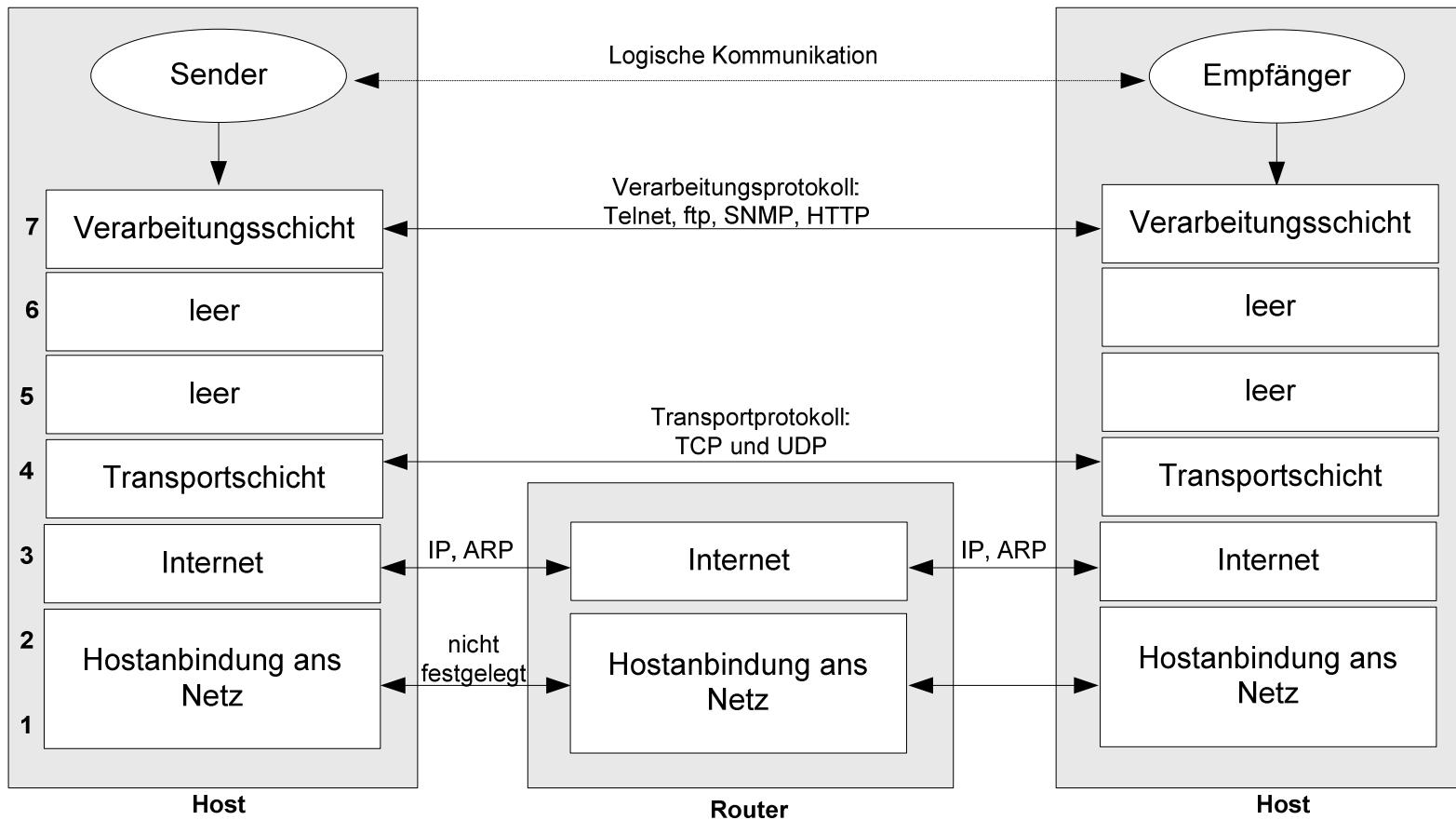
2. **Socket-Programmierung**

- Sockets: Programmiermodell, Systemeinbettung
- TCP-Sockets
- Datagram-Sockets

Wiederholung: ISO/OSI-Referenzmodell



Wiederholung: TCP/IP-Referenzmodell



Überblick über Sockets

- Die Socket-Schnittstelle ist eine API, mit der man Kommunikationsanwendungen entwickeln kann
- Die Socket-Schnittstelle ist eine **Transportzugriffs-schnittstelle**
- Sockets wurden in der Universität von Berkeley entwickelt (BSD-Version von UNIX), **erste Version 4.1cBSD-System** für die VAX aus dem Jahre 1982
- Die **Originalversion** der Socket-Schnittstelle stammt von Mitarbeitern der **Firma BBN** (ARPA-Projekt, 1981)
- Sockets sind heute der **De-facto-Standard**

Überblick über Sockets

- Die Socket API **unterstützt** vor allem **Client-Server-Anwendungen**, was aus dem Programmiermodell hervorgeht:
 - Aktiver Partner = Client
 - Passiver Partner = Server
- Sockets sind **Kommunikationsendpunkte** innerhalb der Applikationen, die in der Initialisierungsphase miteinander verbunden werden
- Dabei spielt es keine Rolle, auf welchen Rechnern die miteinander kommunizierenden Prozesse laufen

Protokollmechanismen

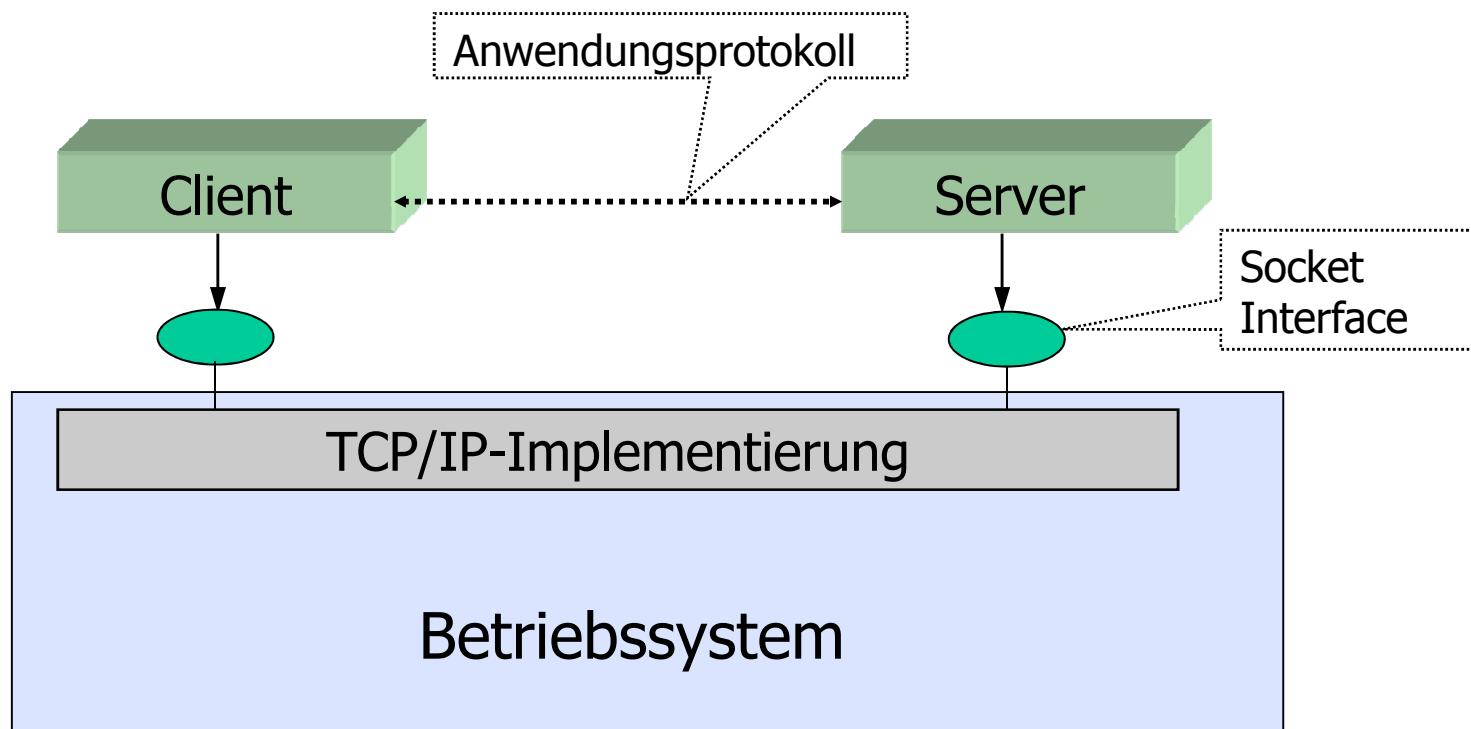
- Die TCP-Socket-Schnittstelle ist **stream-orientiert**:
Es wird ein Datenstrom eingerichtet
- Die UDP-Socket-Schnittstelle ist nachrichten-orientiert
- Es ist **verbindungsorientierte** (TCP-basierte) und **verbindungslose** (UDP-basierte) Kommunikation möglich
- Die Adressierung der Kommunikationspartner erfolgt über die Kommunikationsendpunkte über das Tupel (IP-Adresse, Portnummer)

Überblick über Verbindungsauflaufbau

- Aufbau der Kommunikationsbeziehung bei verbindungsorientierter Kommunikation:
 - Serveranwendung wartet auf ankommende Verbindungsauflaufbauwünsche ([listen](#)) an einem TCP-Port
 - Clientanwendung erzeugt eine [Connect.Request-PDU](#)
 - Serveranwendung nimmt den Verbindungswunsch entgegen
 - Serveranwendung beantwortet Verbindungswunsch mit einer [Connect.Response-PDU](#)

Überblick über Systemeinbettung

- Systemeinbettung von Sockets



Phasen der Kommunikation über Sockets

- Die Verwendung von Sockets in einer Applikation erfolgt programmietechnisch ist in **3 Phasen**:
 - Initialisierungsphase
 - Diese Phase ist für UDP symmetrisch, d.h. es gibt keine Unterscheidung zwischen Sender- und Empfänger-Sockets
 - Für TCP ist diese Phase asymmetrisch zwischen Client- und Server-Socket, d.h. ein Server-Socket wird anders initialisiert als ein Client-Socket
 - Lese- und Schreibphase
 - Sowohl für UDP als auch für TCP ist diese Phase symmetrisch, d.h. jeder Socket kann gleichermaßen senden und empfangen
 - Aufräumphase
 - Die benötigten Ressourcen werden freigegeben
 - Diese Phase ist sowohl für TCP als auch für UDP symmetrisch

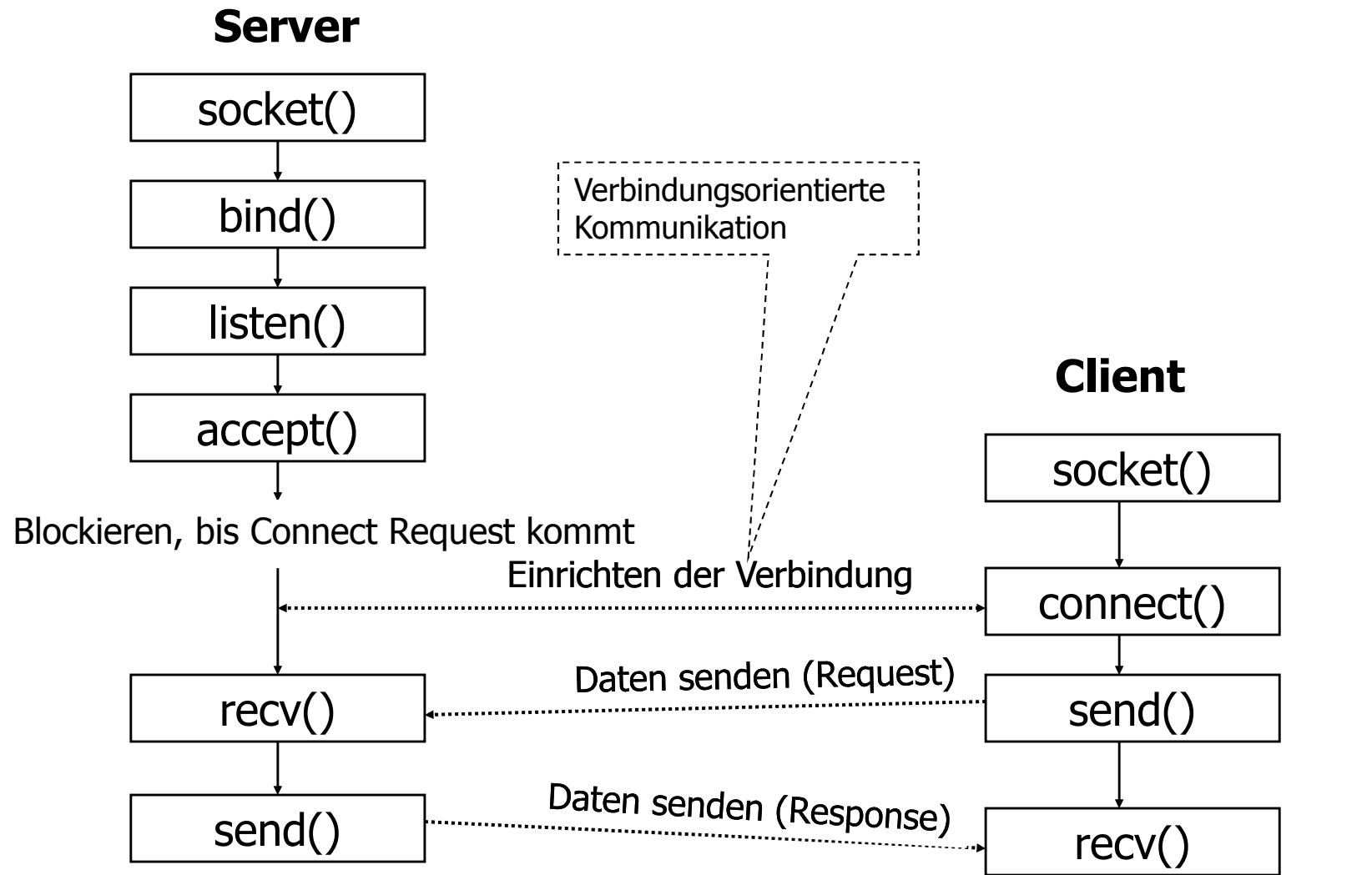
Socket-Programmierung in C

- C-Socket-Schnittstelle wird in nahezu jedem Betriebssystem in einer Funktionsbibliothek bereitgestellt
- Mit der Funktionsbibliothek lässt sich „relativ“ aufwändig programmieren
- Die meisten Socket-basierten Kommunikationsanwendungen sind heute in C programmiert
- Im Folgenden sind einige Ausschnitte aus einfachen Beispielanwendungen skizziert

Die wichtigsten C-Funktionen

- **socket()** - Initialisiert einen Socket
- **bind()** - Ordnet einem Socket eine lokale Adresse zu
- **connect()** - Baut eine Verbindung vom Client zum Server auf
- **listen()** - Setzt den Socket in einen passiven, d.h. auf ankommenden Verbindungswünsche wartenden Zustand
- **accept()** - Wird bei TCP-Verbindungen verwendet und gibt die nächste ankommende, aufgebaute Verbindung aus der Warteschlange zurück
- **close()** - Schließt eine Verbindung
- **recv()** - Liest Daten aus dem spezifizierten Socket und gibt die Anzahl der tatsächlich gelesenen Byte zurück
- **send()** - Sendet Daten über den spezifizierten Socket und gibt die Anzahl der tatsächlich gesendeten Byte zurück

Nutzung der C-Funktionen



C-Programmausschnitt für TCP-Sockets, Server

```
#include "inet.h"
#define SERV_TCP_PORT 5999
char *pname;...
main(int argc, char argv[])
{
    int sockfd, newsockfd, clilen, childpid;
    struct sockaddr_in cli_addr, serv_addr;
    pname = argv[0];
    socket(AF_INET, SOCK_STREAM, 0) < 0)
    // Fehlermeldung ausgeben...
    // Lokale Adresse binden, vorher Socket sockaddr
    // belegen mit IP-Adresse und Port
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr =
        htonl(INADDR_ANY);
    serv_addr.sin_port = htons(SERV_TCP_PORT);
    if (bind(sockfd, (struct sockaddr *)&serv_addr,
        sizeof(serv_addr)) < 0)
        // Fehlermeldung ausgeben
    listen(sockfd, 5);
    for (;;) {
        // Auf Verbindungsaufbauwunsch warten
        clilen = sizeof(cli_addr);
        newsockfd = accept(sockfd,
            (struct sockaddr *) &cli_addr, &clilen);
        if ((childpid = fork()) < 0)
            // Fehlermeldung ausgeben
        else if (childpid == 0) { // Sohnprozess
            close(sockfd);
            // Client-Request bearbeiten
            exit(0);
        }
        close(newsockfd); // Elternprozess
    }
}
```

Protokollfamilie

Sockettyp

Länge der Anfragequeue

Sohnprozess terminieren

Vgl. Stevens

C-Programmausschnitt für TCP-Sockets, Client

```
#include ...
#include "inet.h"
#define SERV_TCP_PORT 5999

// Serveradresse
#define SERV_HOST_ADDR "192.43.235.6"

char *pname;

main(int argc, char argv[])
{
    int sockfd;
    struct sockaddr_in serv_addr;
    pname = argv[0];

    // Serveradresse belegen...
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr =
        inet_addr(SERV_HOST_ADDR);
    serv_addr.sin_port = htons(SERV_TCP_PORT);

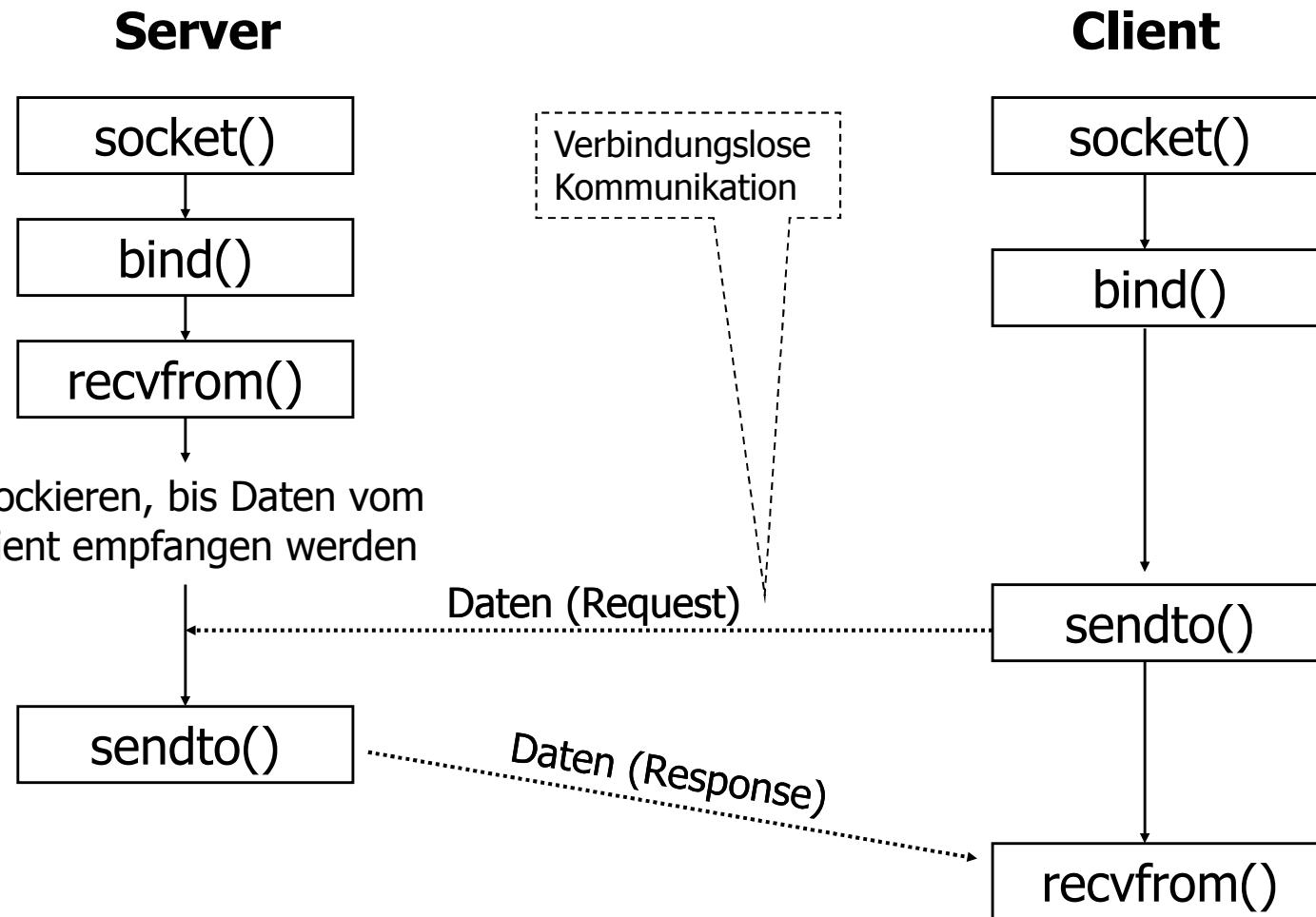
    // Socket öffnen
    if ((sockfd = socket(AF_INET,
        SOCK_STREAM, 0) < 0)
        // Fehlermeldung ausgeben

        // Verbindung zum Server aufbauen

        if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
            // Fehlermeldung ausgeben
            // Verarbeitung: Request senden ...

            close(sockfd);
            exit(0);
    }
}
```

Systemcalls bei UDP-Sockets



C-Programmausschnitt für UDP-Sockets, Server (1)

```
#include „inet.h“;
#define SERV_UDP_PORT 5999
...
main(int argc, char argv[])
{
    int sockfd;
    struct sockaddr_in serv_addr, cli_addr;
    pname = argv[0];

    // UDP-Socket öffnen
    if (sockfd = socket(AF_INET, SOCK_DGRAM,0) < 0) {
        // Error handling
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(INADDR_ANY);
    serv_addr.sin_port = htons(SERV_UDP_PORT);

    // Binde lokale Adresse
    if (bind(socketfd, &serv_addr,...) ...
        dg_echo(sockfd, (struct sockaddr *) &cli_addr, sizeof(cli_addr));
}
```

C-Programmausschnitt für UDP-Sockets, Server (2)

```
#include <sys/types.h>
#include <sys/socket.h>
#define MAXMESG 2048

/* Jede Nachricht wird zum Client zurueckgesendet, echo */

dg_echo(int sockfd, sockaddr pcli_addr, int maxclilen)
{
    int n, clilen;
    char mesg[MAXMESG];

    for (;;) {
        clilen = maxclilen;
        n = recvfrom(sockfd, mesg, MAXMESG, 0, pcli_addr, &clilen);
        if (n < 0) {
            /* Error handling */
        }
        if (sendto(sockfd, mesg, n, 0, pcli_addr, clilen) != n) {
            /* Error handling */
        }
    }
}
```

C-Beispielprogramm für UDP-Sockets, Client (1)

```
#include „inet.h“; ...

main(int argc, char argv[])
{
    int sockfd;
    struct sockaddr_in serv_addr, cli_addr;
    pname = argv[0];
    bzero((char *) &serv_addr, sizeof(serv_addr()));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
    serv_addr.sin_port = htons(SERV_UDP_PORT);

    // UDP-Socket öffnen und lokale Adresse binden
    if (sockfd = socket(AF_INET, SOCK_DGRAM, 0) < 0) {
        /* Error handling */
        if (bind(socketfd, (struct sockaddr *) &cli_addr,...) ...
            dg_cli(stdin, sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
                /* Error handling */
                close(sockfd); exit(0);
    }
}
```

vgl. Stevens

C-Beispielprogramm für UDP-Sockets, Client (2)

```
#include <stdio.h>
#include <sys/socket.h>
#define MAXLINE 512

/* Nachricht ueber stdin einlesen, zum Server senden, wieder empfangen und auf
   stdout ausgeben */
dg_cli(FILE fd, int sockfd, struct sockaddr *pserv_addr, int servlen)
{
    int n;
    char sendline[MAXMESG]; recvline[MAXLINE+1];

    /* Nachricht von stdin einlesen */

    if (sendto(sockfd, sendline, n, 0, pserv_addr, servlen) != n) {
        /* Error handling */

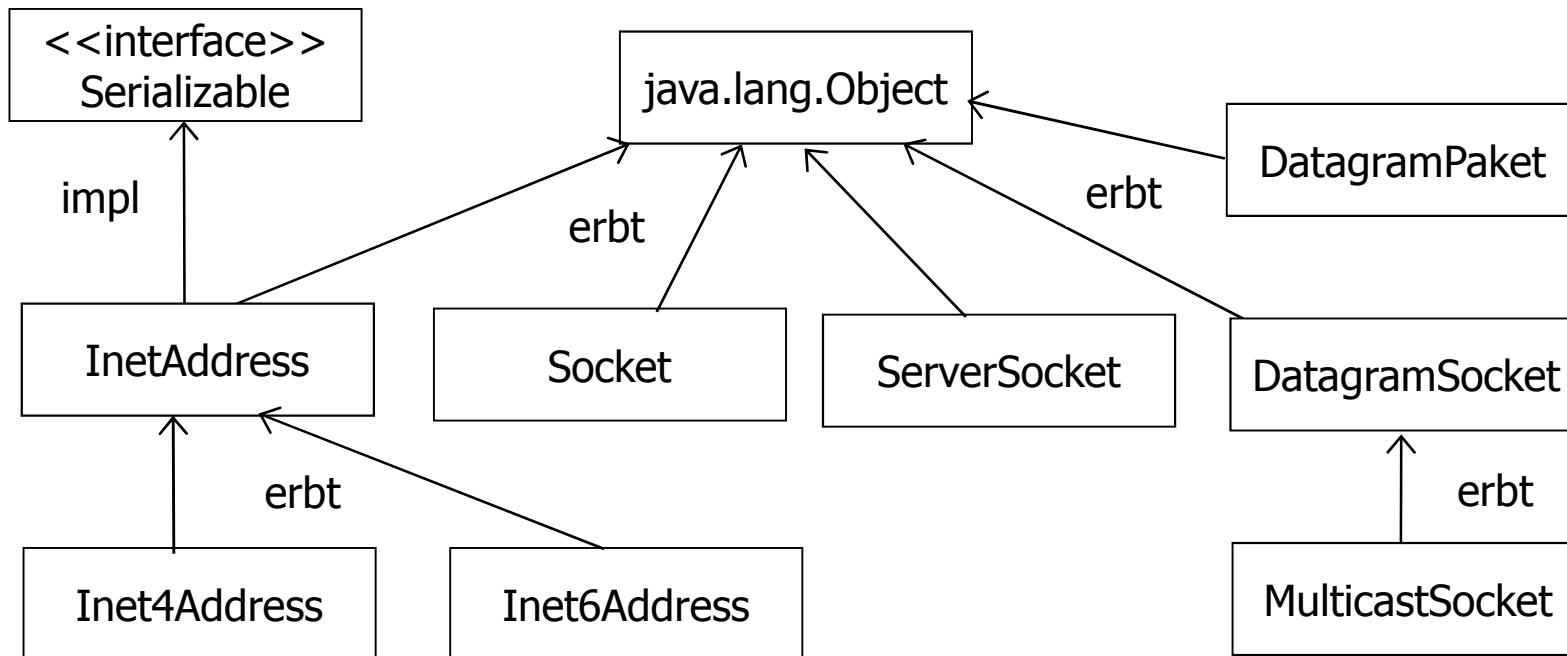
        n = recvfrom(sockfd, recvline, MAXLINE, 0, (struct sockaddr *) 0, (int *) 0);
        if (n < 0) {
            /* Error handling */

            /* Nachricht auf stdout ausgeben */
    }
}
```

Wichtige Java-Klassen für TCP-und UDP-Sockets

- In der Java-API ist das Package **java.net** für TCP-Sockets vorgesehen. Wichtige Klassen sind:
 - InetAddress
 - Socket (Client)
 - ServerSocket
- Das Package **java.net** enthält auch Klassen zur Bearbeitung von UDP-Sockets. Wichtige Klassen sind:
 - DatagramSocket
 - DatagramPaket

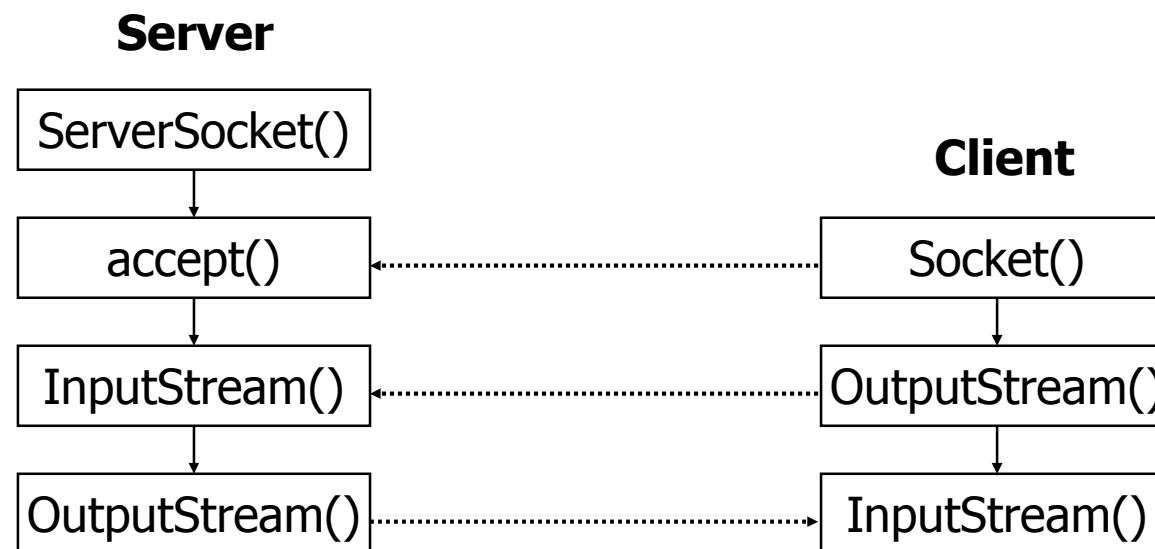
Package java.net, wichtige Klassen



Vgl. Java™ 2 Platform Standard Edition 1.4x, 1.5x, 1.6

Java-TCP-Sockets

- **java.net** stellt eine höherwertige Schnittstelle für Sockets zur Verfügung
 - objektorientierte Schnittstelle
 - Implementierungsdetails gut gekapselt
- Programmierung wird durch Input- und Output-Streams, die den Sockets zugeordnet werden, vereinfacht

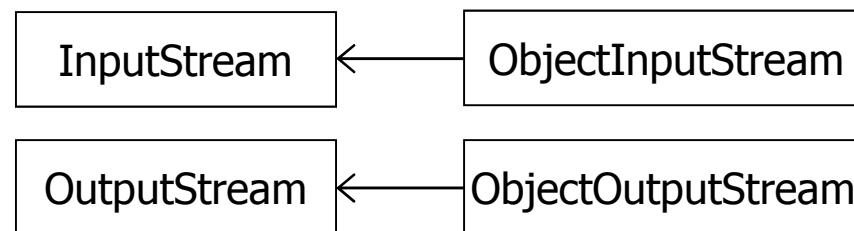


UDP-Sockets allgemein

- Paket-basierte Kommunikation im Gegensatz zu Streams-basiert
- Keine Empfangsgarantie
- Keine Ordnung der Pakete
- Duplikate sind möglich
- Vorteile:
 - Separate Nachrichten können gesendet werden
 - Ein Fehler in einer Nachricht wirkt sich nicht auf die folgenden Nachrichten aus
 - Es werden immer nur ganze Nachrichten empfangen

Weitere wichtige Klassen zur Socket-Programmierung

- Bei TCP-Sockets werden zum Lesen und Schreiben mit Objektströmen verwendet:
 - ObjectInputStream
 - ObjectOutputStream



Package `java.net`, `SocketExceptions`

`java.lang.Object`

`java.lang.Throwable`

`java.lang.Exception`

`java.io.IOException`

`java.net.ProtocolException`

`java.net.UnknownHostException`

`java.net.UnknownServiceException`

`java.net.SocketException`

`java.net.ConnectException`

`java.net.BindException`

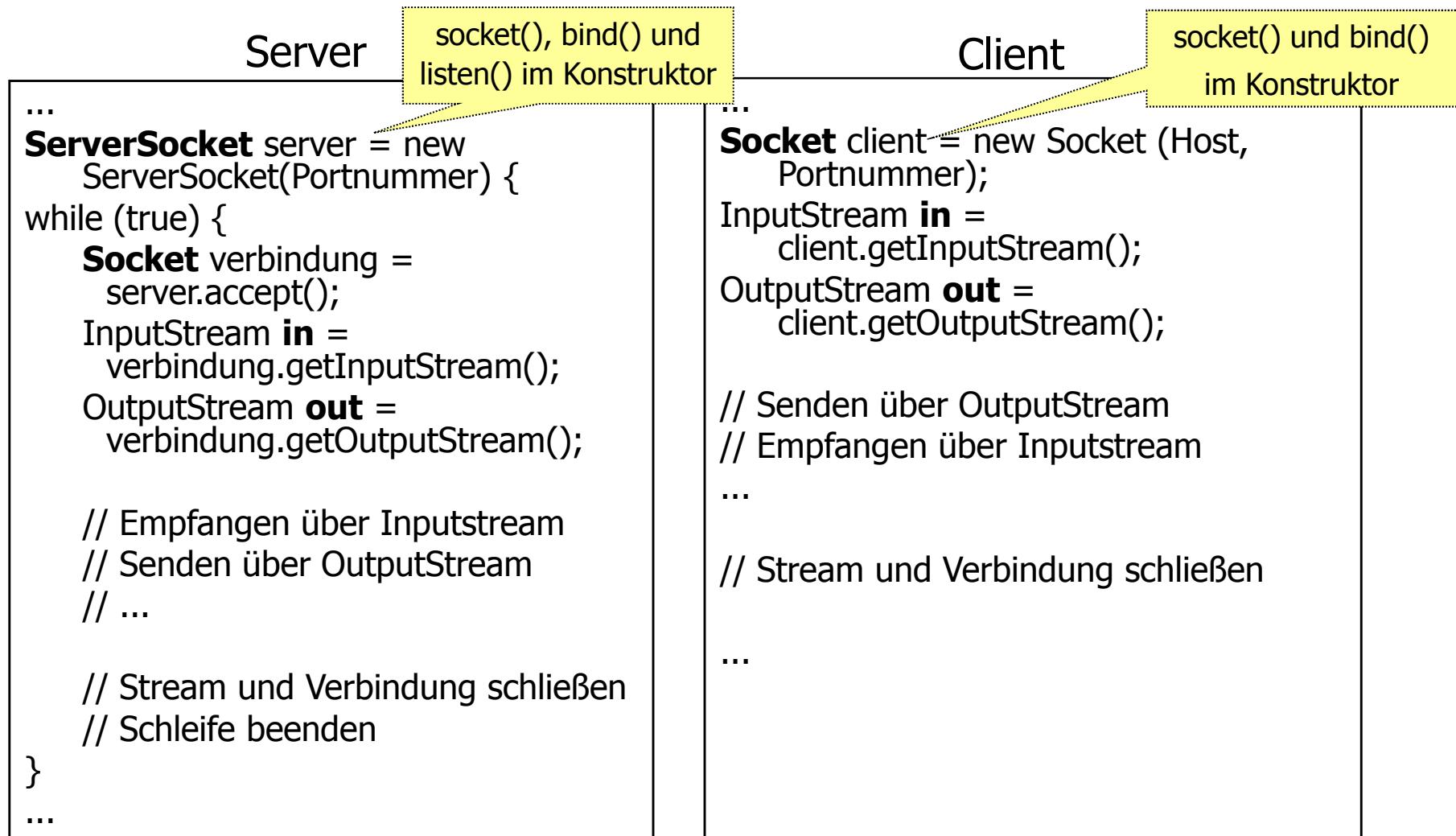
`java.net.NoRouteToHostException`

...

Beispielprogramm mit Java TCP-Sockets

- Der folgende Programmrahmen zeigt einen Server, der Client-Requests nur sequentiell abarbeiten kann
- Der Server übernimmt die Verbindung und arbeitet den Request ab
- Erst nach der Bearbeitung kann er wieder neue Requests empfangen
- Alternativ: Für die Bearbeitung nebenläufiger Requests kann man Java-Threads nutzen
 - Verbindungen werden über eigene Threads angenommen und bearbeitet

Beispielprogramm - Programmausschnitt



UDP-Sockets in Java (Datagram-Sockets)

- Sockets vom Typ **DatagramSocket** für Datagramme
 - analog TCP kann dem Konstruktor eine **Portnr. > 5000** vorgegeben werden
 - ohne spezifizierte Portnummer → **freie Portnummer** im Bereich 1024 – 5000
- **Methoden** der Klasse **DatagramSocket** für die nachrichtenorientierte Kommunikation
 - void **send**(DatagramPacket p) zum Senden und
 - void **receive**(DatagramPacket p) zum Empfangen von Datagrammen
 - receive blockiert den Aufrufer bis ein Datagramm eingeht

UDP-Sockets in Java

- Datagramme vom Typ **DatagramPacket** können instanziert werden:
 - zum **Empfangen**, gibt man dem Konstruktor ein Byte-Array mit, in das die zu empfangenden Daten eingetragen werden sollen.
 - zum **Versenden**, gibt man dem Konstruktor neben den Daten (Byte-Array) die **IP-Adresse** und den **Port** des Empfängers mit.
- **DatagramPacket** bietet ferner Methoden zum **Lesen** und **Schreiben** der Daten, der IP-Adresse und der Portnummer des Senders und des Empfängers
- → Vgl. Java™ 2 Platform Standard Edition für genaue Information zu den Methoden

Ablauf einer Kommunikation über einen UDP-Socket

- DatagramSocket erzeugen
- Zum Senden eines Datagramms:
 - Objekt vom Typ DatagramPacket erzeugen;
 - dabei die zu sendenden Daten und die Adresse des Empfängers angeben.
 - Das Datagramm mit der Methode send am DatagramSocket verschicken

Ablauf einer Kommunikation über einen UDP-Socket

- Zum Empfangen eines Datagramms:
 - Objekt vom Typ `DatagramPacket` erzeugen
 - Byte-Array angeben, das Platz für die zu empfangenden Daten bereitstellt
 - Daten mit `receive` am `DatagramSocket` empfangen
 - Die eigentlichen Daten und die Senderadresse mit Hilfe der Methoden `getData` bzw. `getAddress` am `DatagramPacket` auslesen
- `DatagramSocket` mit `close` schließen

Beispiel EchoServer (1)

```
package UDPEchoExample;
import java.net.*;
import java.io.*;

public class UDPEchoServer {
    protected DatagramSocket socket;
    public UDPEchoServer (int port) throws IOException
    {
        socket = new DatagramSocket (port);
    }
    public void execute () throws IOException
    {
        while (true) {
            DatagramPacket packet = receive();
            sendEcho (packet.getAddress (), packet.getPort (),
                      packet.getData (), packet.getLength ());
        }
    }
}
```

Beispiel EchoServer (2)

```
protected DatagramPacket receive () throws IOException {
    byte buffer[] = new byte[65535];
    DatagramPacket packet = new DatagramPacket (buffer, buffer.length);
    socket.receive (packet);
    System.out.println ("Received " + packet.getLength () + " bytes.");
    return packet;
}
protected void sendEcho (InetAddress address, int port, byte data[], int
length) throws IOException {
    DatagramPacket packet = new DatagramPacket (data, length, address, port);
    socket.send (packet);
    System.out.println („Response sent“);
}
public static void main (String args[]) throws IOException {
    if (args.length != 1)
        throw new RuntimeException ("Syntax: UDPEchoServer <port>");
    UDPEchoServer echo = new UDPEchoServer (Integer.parseInt (args[0]));
    echo.execute ();
}
```

Beispiel EchoClient (1)

```
package UDPEchoExample;
import java.net.*;
import java.io.*;
public class UDPEchoClient {
    protected DatagramSocket socket;
    protected DatagramPacket packet;
    public UDPEchoClient (String message, String host, int port) throws
        IOException {
        socket = new DatagramSocket ();
        packet = buildPacket (message, host, port);
        try {
            sendPacket ();
            receivePacket ();
        } finally {
            socket.close ();
        }
    }
    protected void sendPacket () throws IOException {
        socket.send (packet); ...
    }
}
```

Beispiel EchoClient (2)

```
protected void receivePacket () throws IOException {
    byte buffer[] = new byte[65535];
    DatagramPacket packet = new DatagramPacket (buffer, buffer.length);
    socket.receive (packet);
    ByteArrayInputStream byteI = new ByteArrayInputStream (packet.getData (),
        0, packet.getLength ());
    DataInputStream dataI = new DataInputStream (byteI);
    String result = dataI.readUTF ();
}
public static void main (String args[]) throws IOException {
    if (args.length != 3)
        throw new RuntimeException („EchoClient <host> <port> <message>”);
    while (true) {
        new UDPEchoClient (args[2], args[0], Integer.parseInt (args[1]));
        try {
            Thread.sleep (1000);
        } catch (InterruptedException ex) {...}
    }
}
```

Package java.net, MulticastSockets

- Die Klasse MulticastSockets dient zum Senden und Empfangen von IP-Multicast-Datagrammen
- Ein MulticastSocket ist ein UDP-DatagramSocket zur IP-basierten Gruppenkommunikation
 - Siehe IP-Adressen 224.* - 239.*
 - Methode **joinGroup(InetAddress groupAddr)** zum Anbinden an eine Gruppe
 - Methode **leaveGroup(InetAddress mcastaddr)** zum Verlassen der Gruppe
- Ein gesendetes Datagramm empfangen alle Gruppenmitglieder

Zusammenfassung: Sockets (1)

- Wie andere Programmiersprachen bietet Java Sockets für die TCP/IP- bzw. UDP/IP-Kommunikation an
- **Socketadressen** bestehen aus zwei Komponenten, der IP-Adresse bzw. dem logischen DNS-Namen des betreffenden Rechners und einer Portnummer
- Für TCP-Verbindungen gibt es **Server-Sockets und Client-Sockets**:
 - Wenn sich ein Client-Socket an einen Server-Socket anbindet, wird auf Server-Seite ein neuer Socket erzeugt, der den Endpunkt dieser Verbindung darstellt
 - Das eigentliche Senden und Empfangen von Daten geschieht über Schreib- und Leseströme, die man mit einem Socket assoziieren kann

Zusammenfassung: Sockets (2)

- Datagram-Sockets = Endpunkte zum Senden und Empfangen von Datagrammen
 - UDP ist verbindungslos
 - UDP ist nicht zuverlässig
 - UDP garantiert keine Reihenfolgetreue
- Das Anwendungsprogramm muss diese Dinge regeln
- Bei Nutzung von UDP-Sockets sollte man berücksichtigen:
 - 16 Bits für Paketlänge, also Datenlänge begrenzen, kein Stream
 - Vermeidung von Fragmentierung wichtig, wenn ein Fragment (IP) verloren geht, wird die ganze Nachricht verworfen

Überblick

1. Konzepte

- Halbduplex, vollduplex, Empfängeradressierung
- Kommunikationsformen
- Ablauf der Kommunikation

2. Socket-Programmierung

- Sockets: Programmiermodell, Systemeinbettung
- TCP-Sockets
- Datagram-Sockets

Datenkommunikation

Transportschicht Grundlagen

Wintersemester 2012/2013

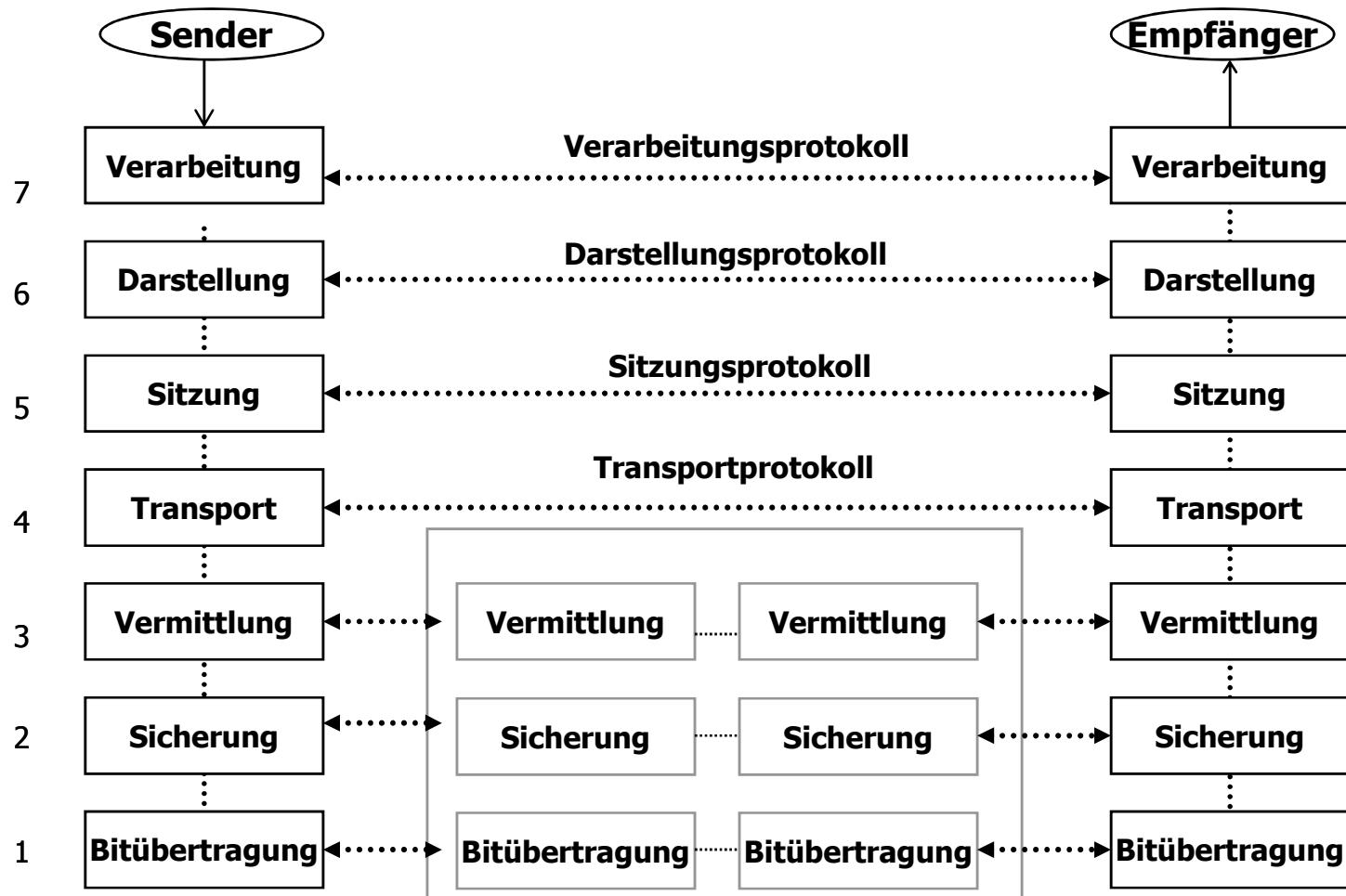
Überblick

1. Einführung in grundlegende Aspekte

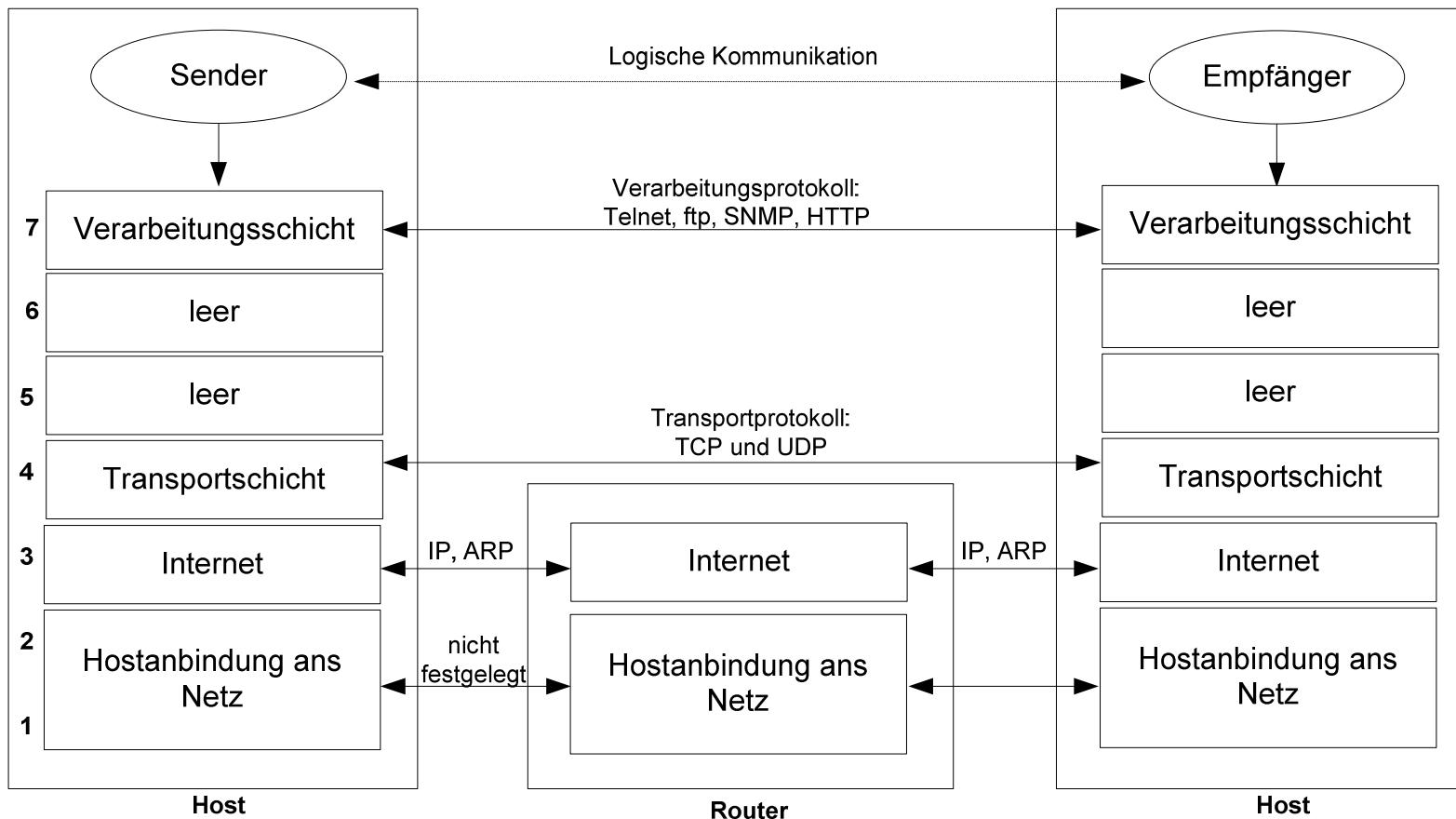
2. Transportorientierte Dienste

- Überblick und Grundlagen
- Protokollfunktionen der Transportschicht
- Verbindungsmanagement
 - Aufbau
 - Abbau
- Zuverlässiger Datentransfer
 - Quittierung
 - Übertragungswiederholung
 - Flusskontrolle
 - Staukontrolle

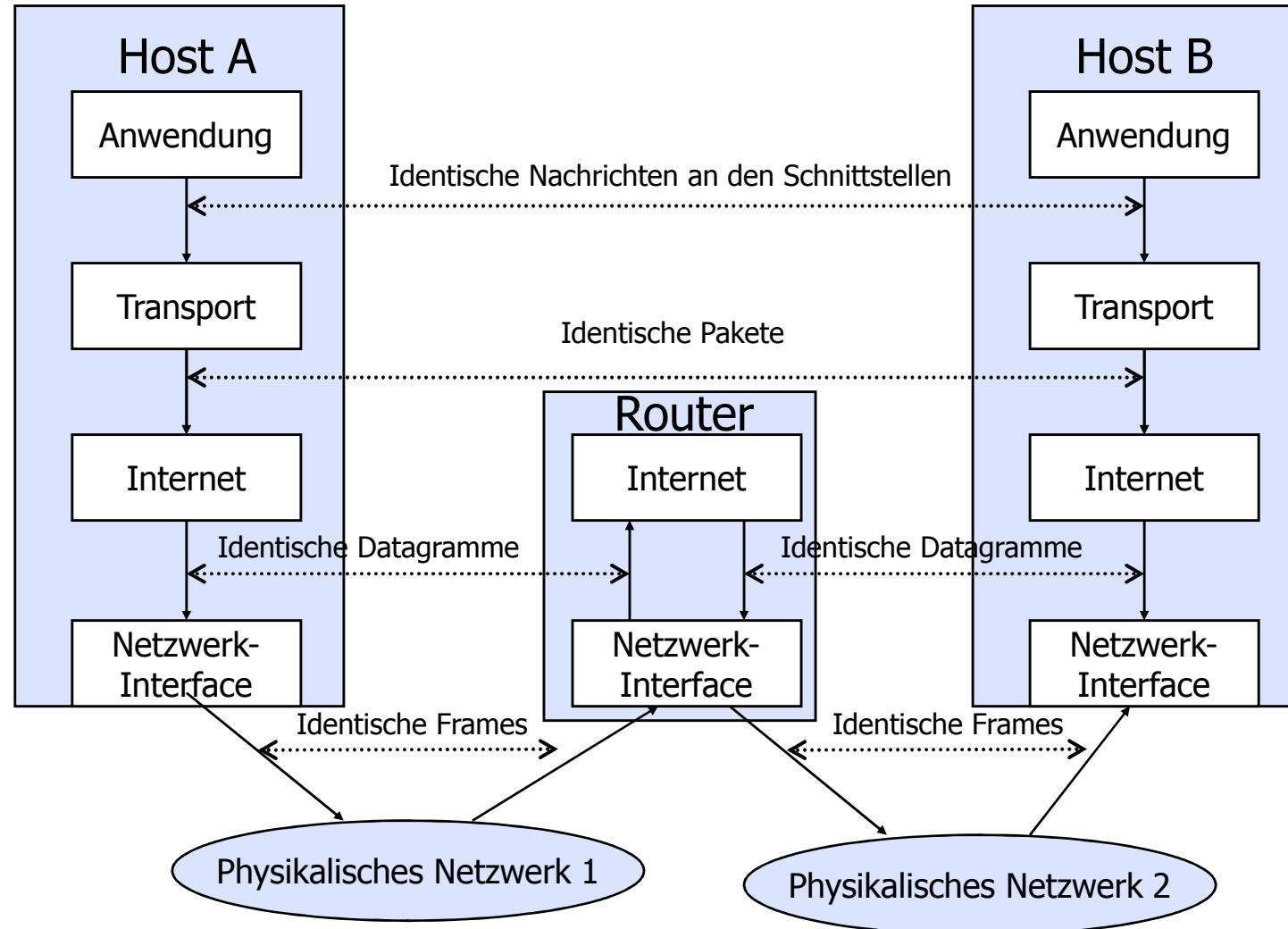
ISO/OSI-Referenzmodell



TCP/IP-Referenzmodell



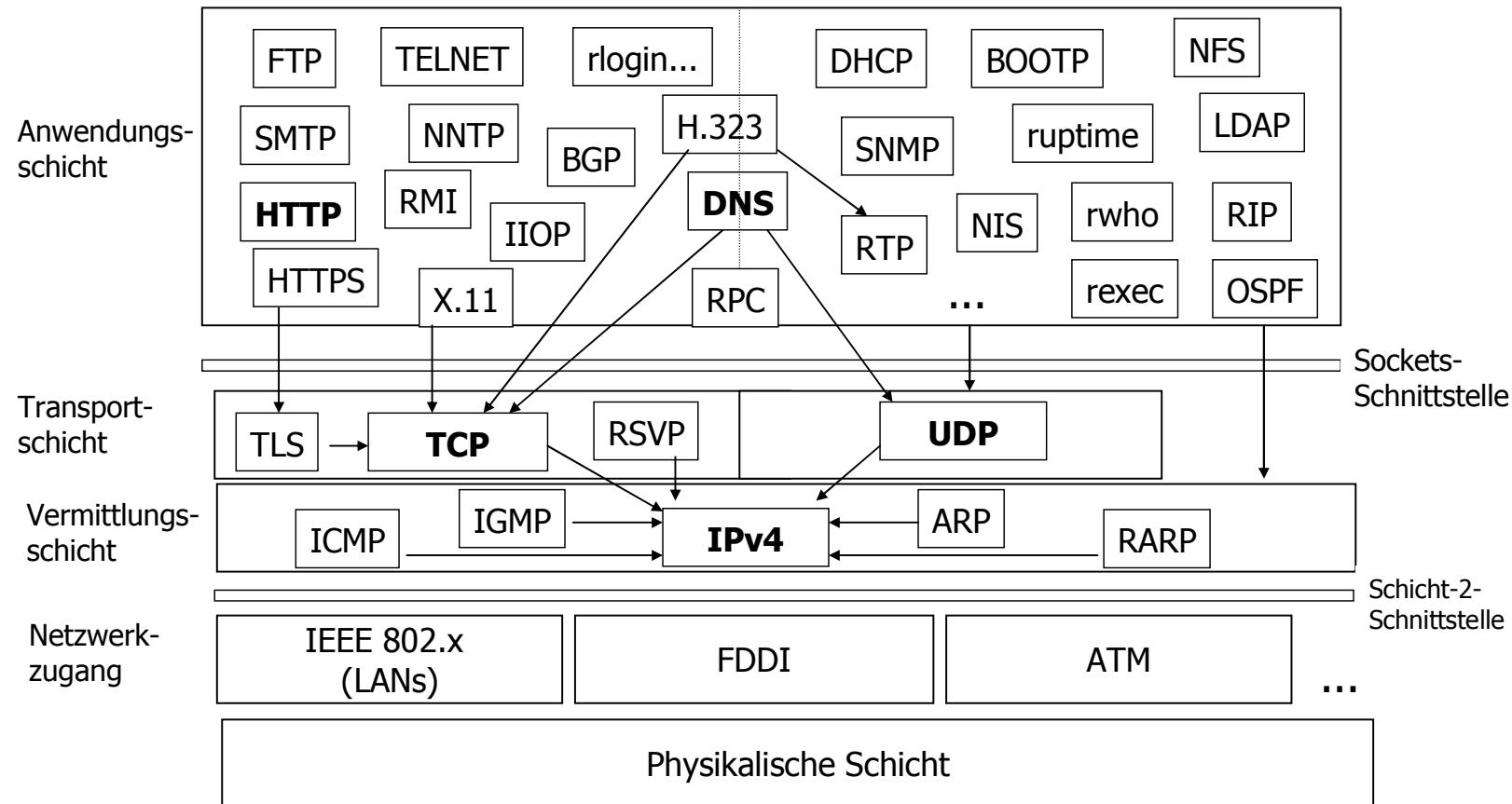
TCP/IP-Referenzmodell, Schichten und Gateways



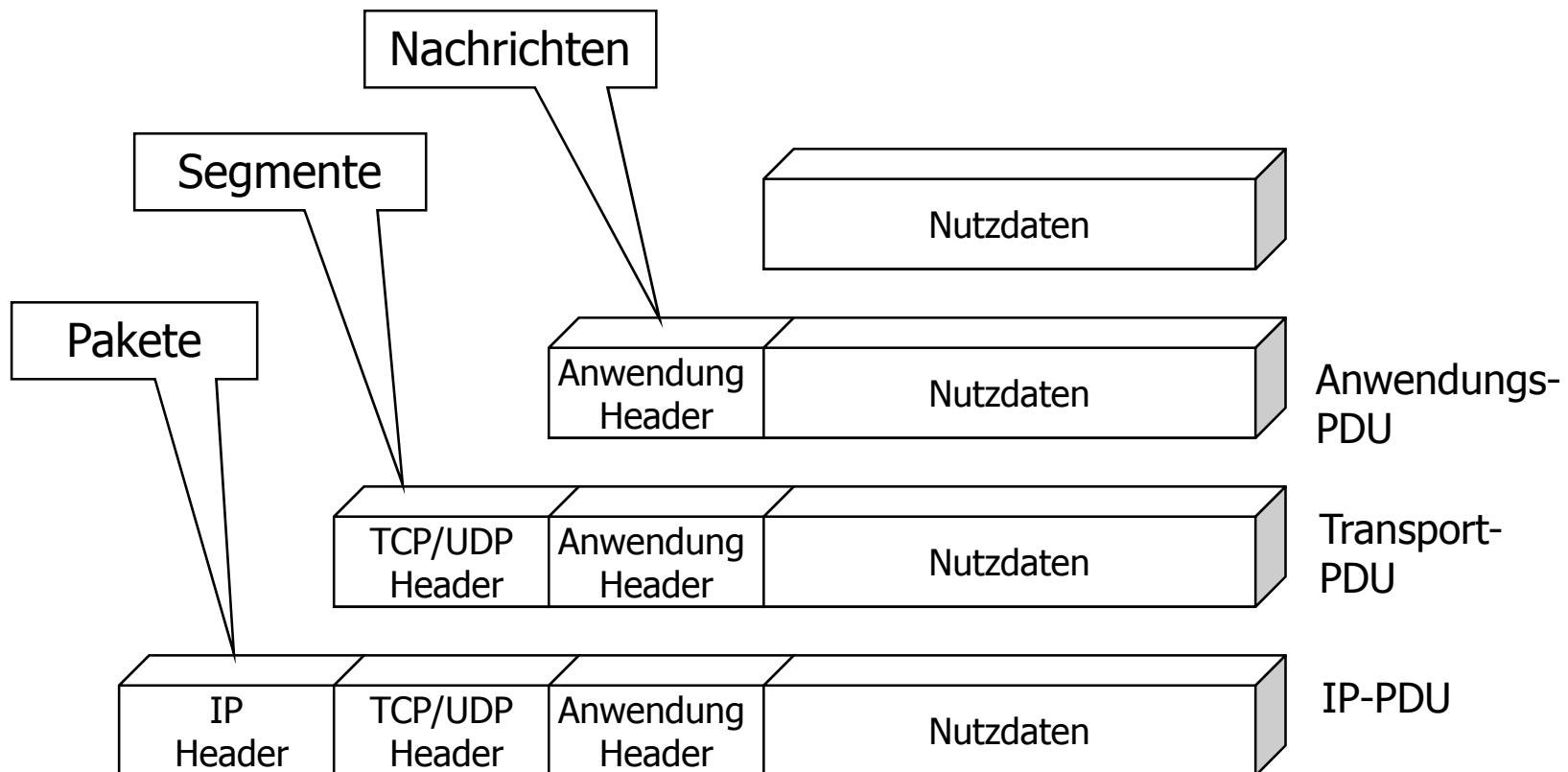
Allgemeine Protokollmechanismen

- Fehlerbehandlung
 - Reihenfolgegarantie, Sequenznummern
 - Quittierung
 - Zeitüberwachung
 - Fehlerkennung und -korrektur
 - Längenanpassung
 - Assemblierung und Deassemblierung
- Systemleistungsanpassung
 - Flusssteuerung
 - Überlaststeuerung
- Übertragungsleistungsanpassung
- ...

TCP/IP-Protokollfamilie



TCP/IP-Referenzmodell, Protokollkapselung



PDU = Protocol Data Unit

Überblick

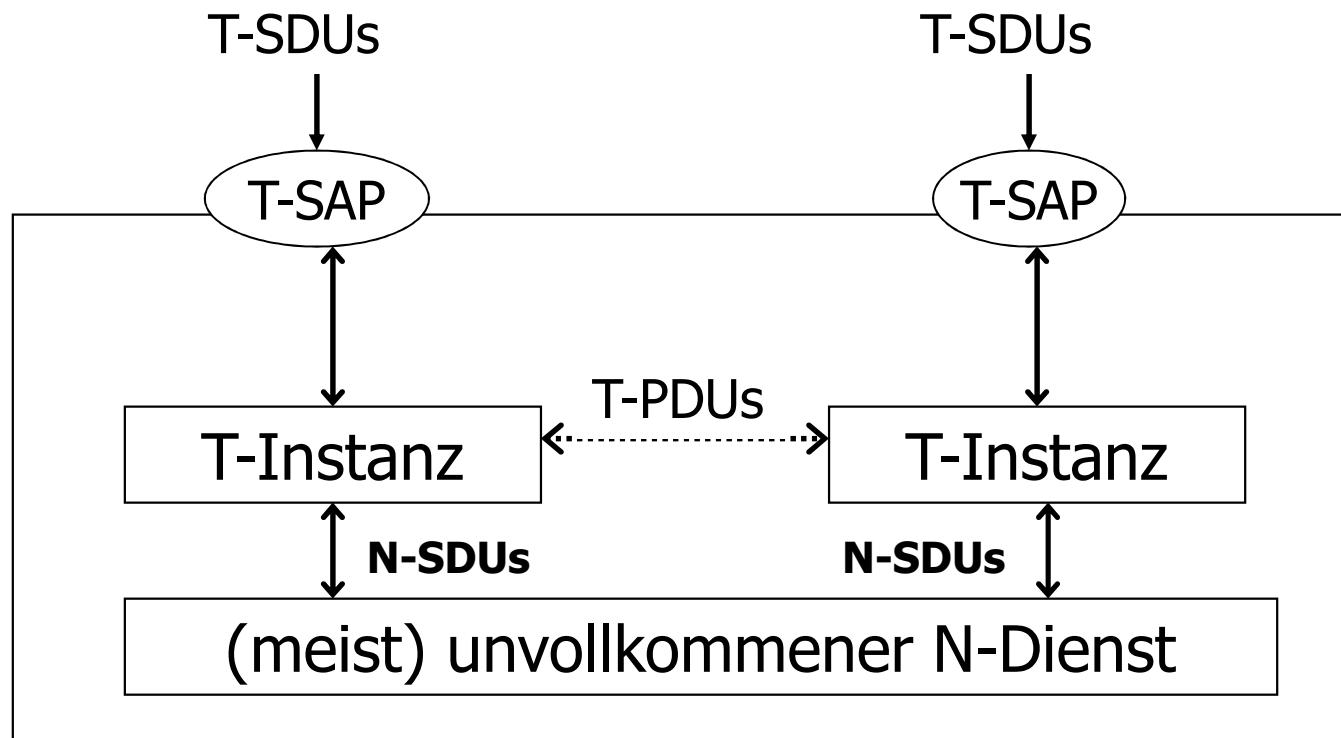
1. Einführung in grundlegende Aspekte

2. Transportorientierte Dienste

- Überblick und Grundlagen
- Protokollfunktionen der Transportschicht
- Verbindungsmanagement
 - Aufbau
 - Abbau
- Zuverlässiger Datentransfer
 - Quittierung
 - Übertragungswiederholung
 - Flusskontrolle
 - Staukontrolle

Dienste der Transportschicht

- Logischer Ende-zu-Ende-Transport



Verbindungen

- Man unterscheidet
 - **verbindungsorientierte** Transportdienste
 - **verbindungslose** Transportdienste

Verbindungsorientierte Transportdienste

- Eine Verbindung wird etabliert
- Gemeinsamer Kontext wird aufgebaut
- Geprägt von traditionellen Kommunikationsdiensten wie Telefonieren
- Hohe Zuverlässigkeit
- Fehlerfreie und reihenfolgerichtige Auslieferung der Daten beim Empfänger
- Verbindungsorientierte Protokolle sind komplexer
 - Warum?
- Wann braucht man Verbindungen?

Verbindungslose Transportdienste

- Verlust von Datenpaketen wird nicht bemerkt
- Verfälschung des Nutzdatenteils ist nicht unbedingt nachvollziehbar
- Reihenfolgezerstörung ist möglich
- Kein Zusammenhang bei aufeinanderfolgenden Dienstaufrufen
- T-PDUs enthalten die Adressinformation von Sender und Empfänger

Protokollfunktionen in Transportprotokollen

- Verbindungsmanagement und Adressierung
- Zuverlässiger Datentransfer
- Flusskontrolle
- Staukontrolle
- Multiplexierung (Multiplexing) und Demultiplexing
- Fragmentierung und Defragmentierung

Überblick

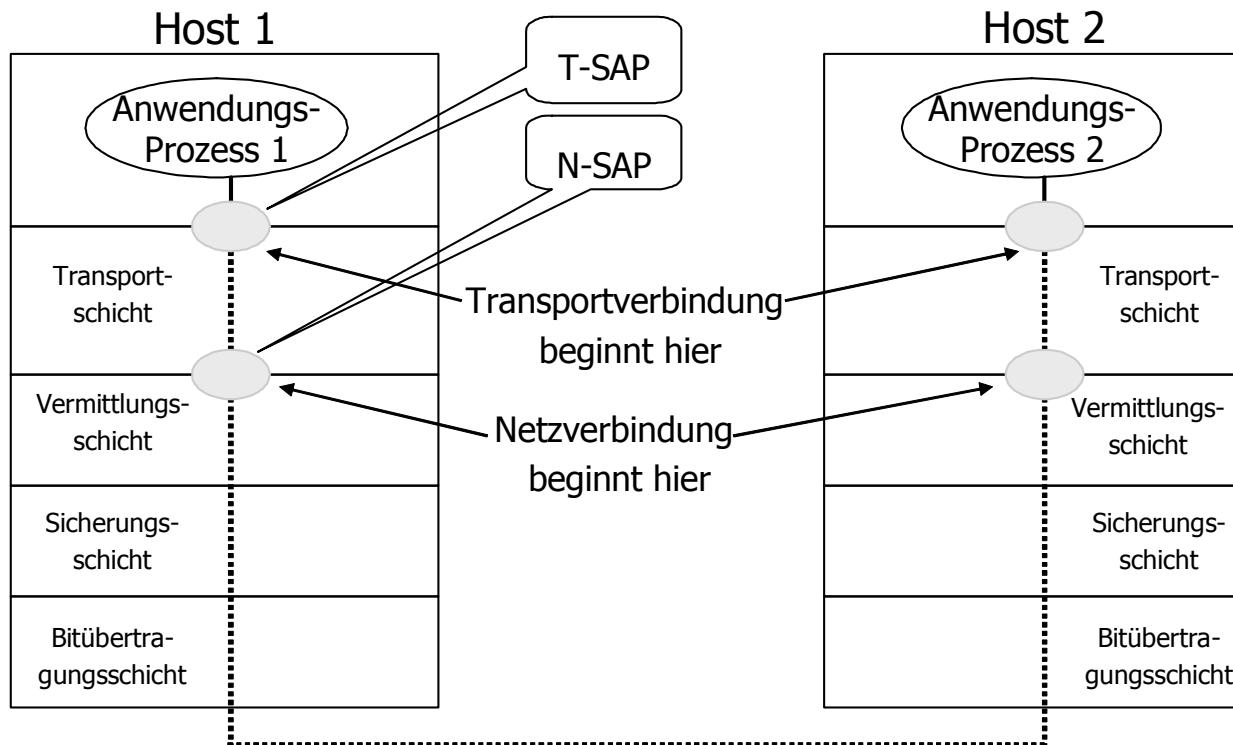
1. Einführung
2. Transportorientierte Dienste
 - Überblick und Grundlagen
 - Protokollfunktionen der Transportschicht
 - **Verbindungsmanagement**
 - Aufbau
 - Abbau
 - Zuverlässiger Datentransfer
 - Quittierung
 - Übertragungswiederholung
 - Flusskontrolle
 - Staukontrolle

Verbindungsmanagement und Adressierung

- Kommunizierende Anwendungsprozesse müssen sich kennen
 - Schicht 4: Transportadresse
 - T-SAP (Transport Service Access Point)
- Eine Transport-Instanz unterstützt in der Regel mehrere T-SAPs
- Transportadressen sind oft kryptisch, daher symbolische Adressen notwendig
 - Directory Service (Naming Service)

Verbindungsmanagement und Adressierung

T-SAP



T-SAP = Transport Service Access Point

N-SAP = Network Service Access Point

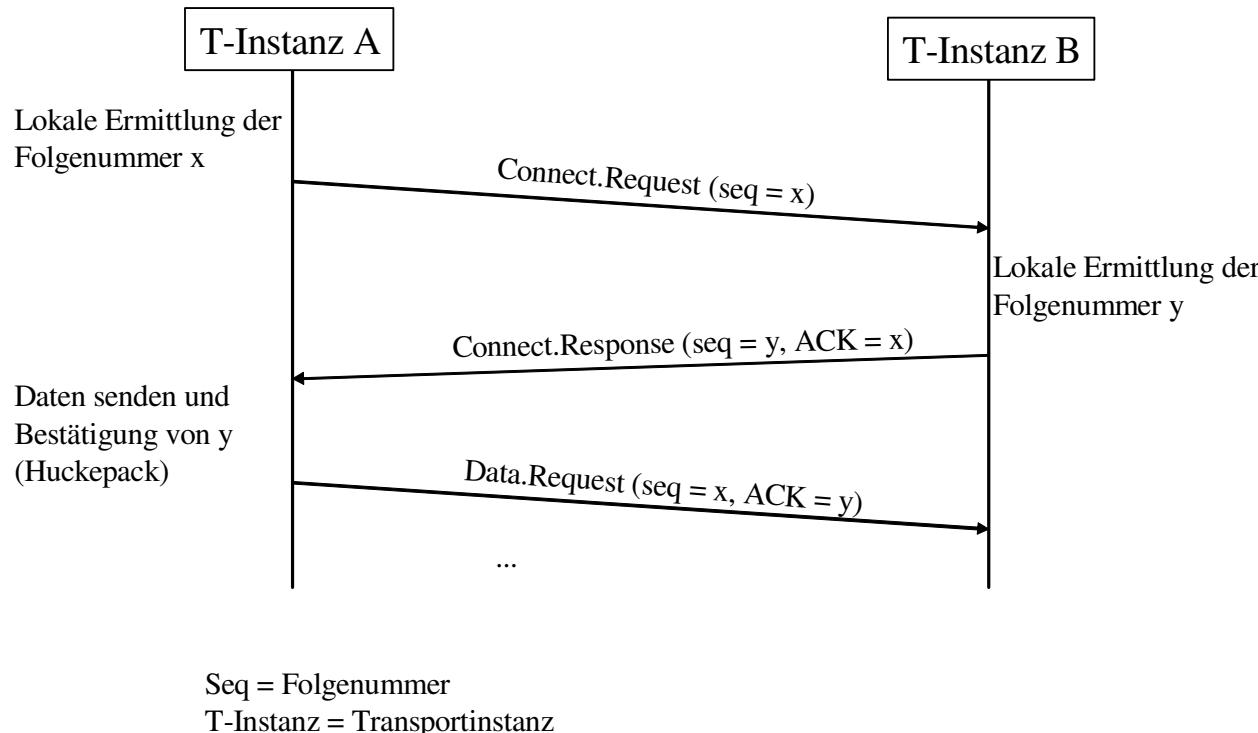
Verbindungsauftbau

- Einrichten von Connection End Points (CEP)
 - Kontextaufbau auf beiden Seiten
- **Zwei-Wege-Handshake-Protokolle**
- **Drei-Wege-Handshake-Protokolle**
- Vorsicht **Duplikate!**
 - Diverse Fehlerszenarien möglich
 - Mechanismus der **Folgenummern** kombiniert mit einer maximalen Paketlebensdauer
 - Folgenummern sind einfache Zähler

Verbindungsauftbau, Drei-Wege-Handshake

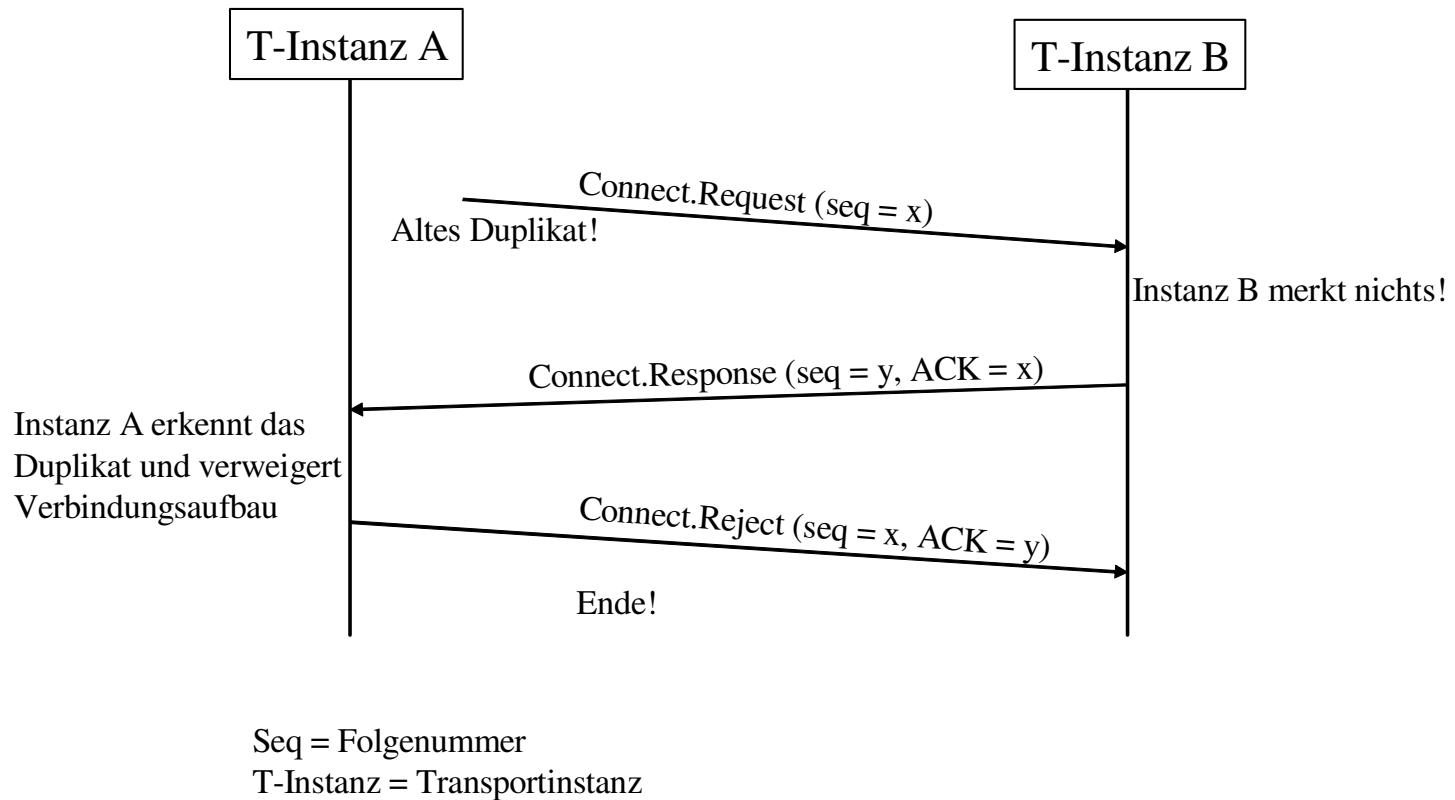
Normaler Protokollverlauf

- Instanz A und B suchen eigene Folgenummern x und y (seq) aus



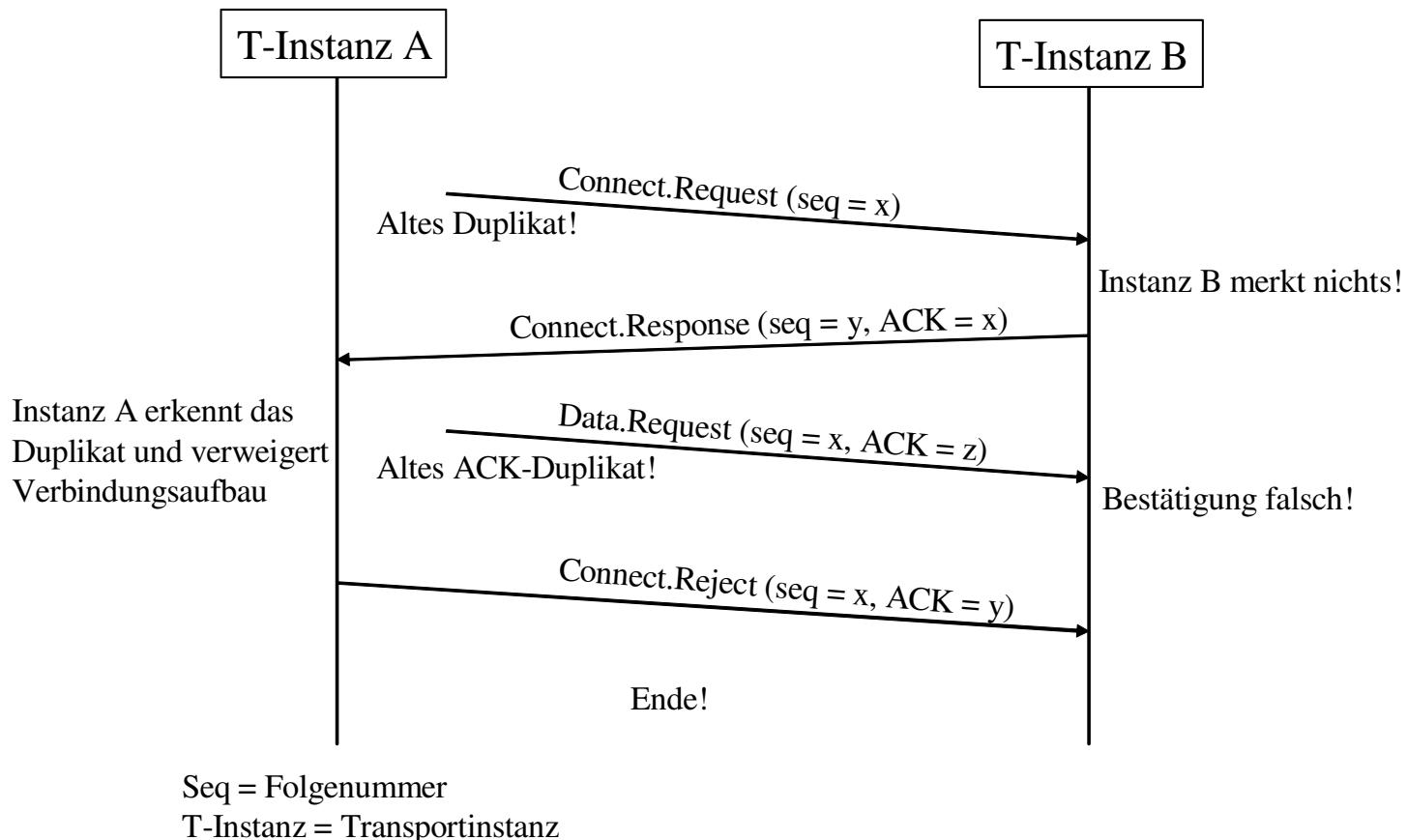
Verbindungsauftbau, Drei-Wege-Handshake

Altes CR-Duplikat taucht auf



Verbindungsauftau, Drei-Wege-Handshake

Duplikat von Connect-Request und Duplikat von ACK tauchen plötzlich auf

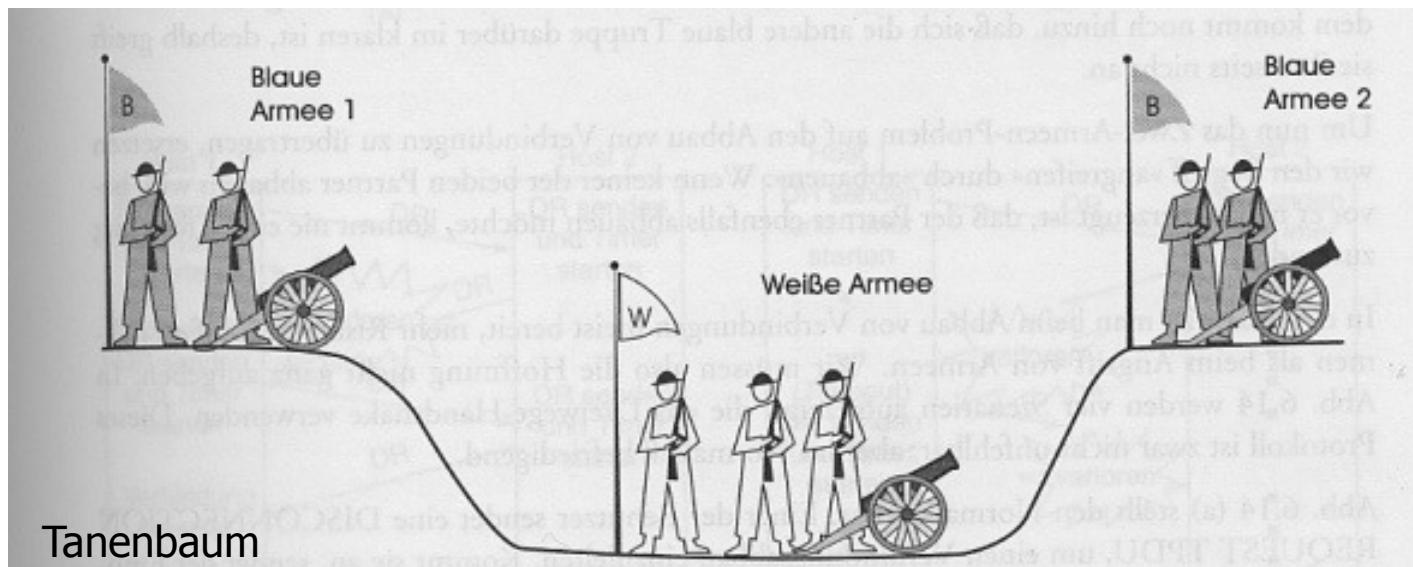


Verbindungsabbau

- Anforderung:
 - Beim Verbindungsabbau dürfen keine Nachrichten verloren gehen
- Datenverlust kann vorkommen, wenn
 - eine Seite einen Verbindungsabbau initiiert,
 - die andere aber vor Erhalt der Disconnect-Request-PDU noch eine Nachricht sendet
 - Diese **Nachricht ist verloren (Datenverlust)**
- Anspruchsvolles Verbindungsabbau-Protokoll notwendig:
 - Dreiwege-Handshake-Mechanismus wird auch hier genutzt
 - Beide Seiten bauen ihre „Senderichtung“ ab

Verbindungsabbau und das Zwei-Armeen-Problem

- Die Armee der Weißröcke lagert in einem Tal
- Auf beiden Anhöhen lagert ein Teil der Armee der Blauröcke
- Die Blauröcke können nur gemeinsam gewinnen und müssen ihren Angriff synchronisieren
- Unzuverlässiger Kommunikationskanal: Boten, die zu Fuß durch das Tal rennen müssen

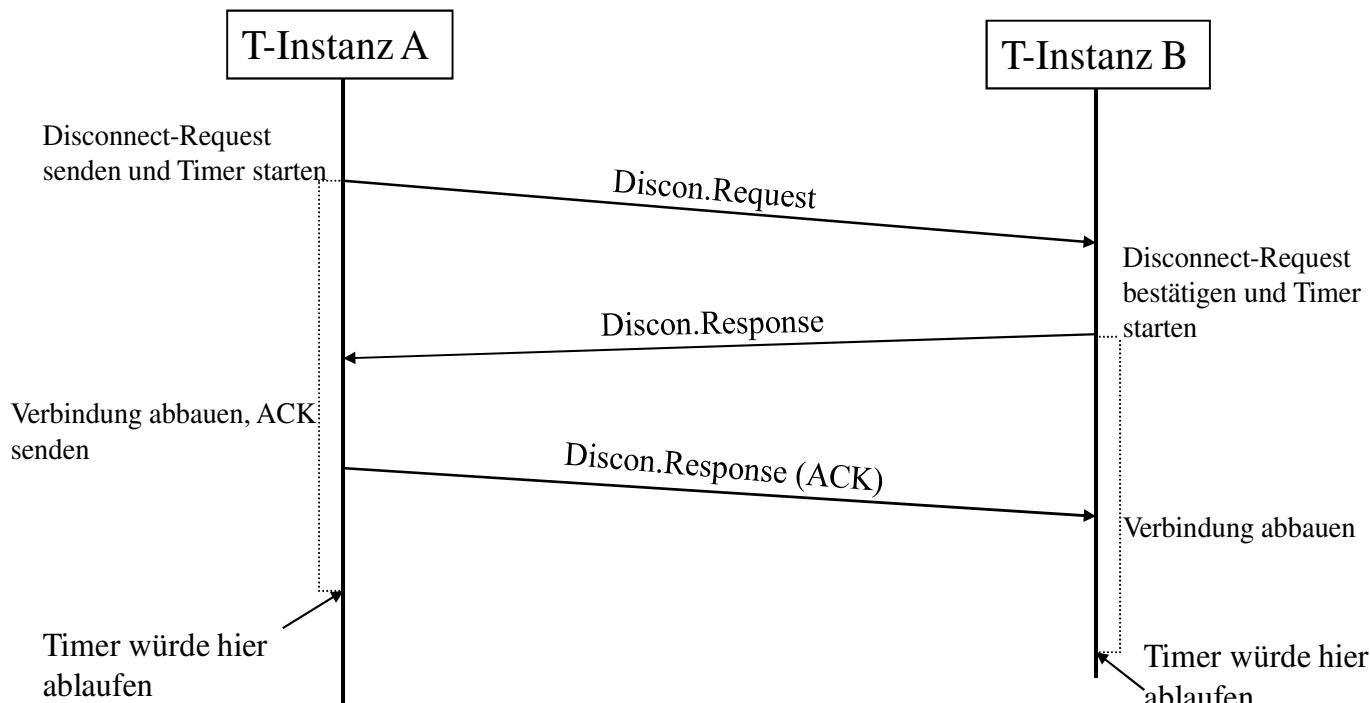


Verbindungsabbau, Timerüberwachung

- Kein Protokoll ist absolut zuverlässig
- Es wird immer eine Seite geben, die **unsicher** ist, ob die letzte Nachricht angekommen ist
- Übertragen auf den Verbindungsabbau:
 - Beim Dreiwege-Handshake kann **immer** ein Disconnect-Request oder **eine Bestätigung verloren gehen**
- Praktische Lösung: **Timerüberwachung** mit begrenzter Anzahl an Nachrichtenwiederholungen
- Nicht unfehlbar, aber doch ganz zufriedenstellend

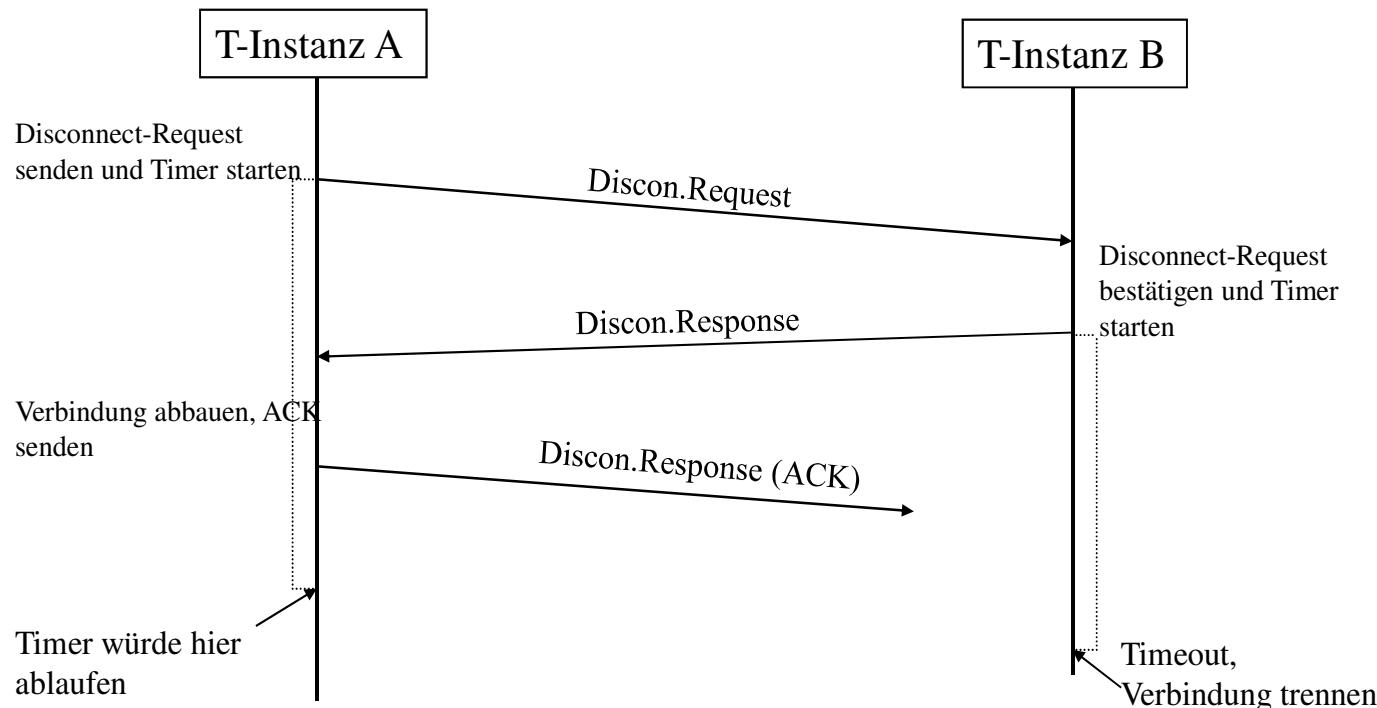
Timerüberwachung beim Verbindungsabbau

Szenario „Normaler Ablauf“



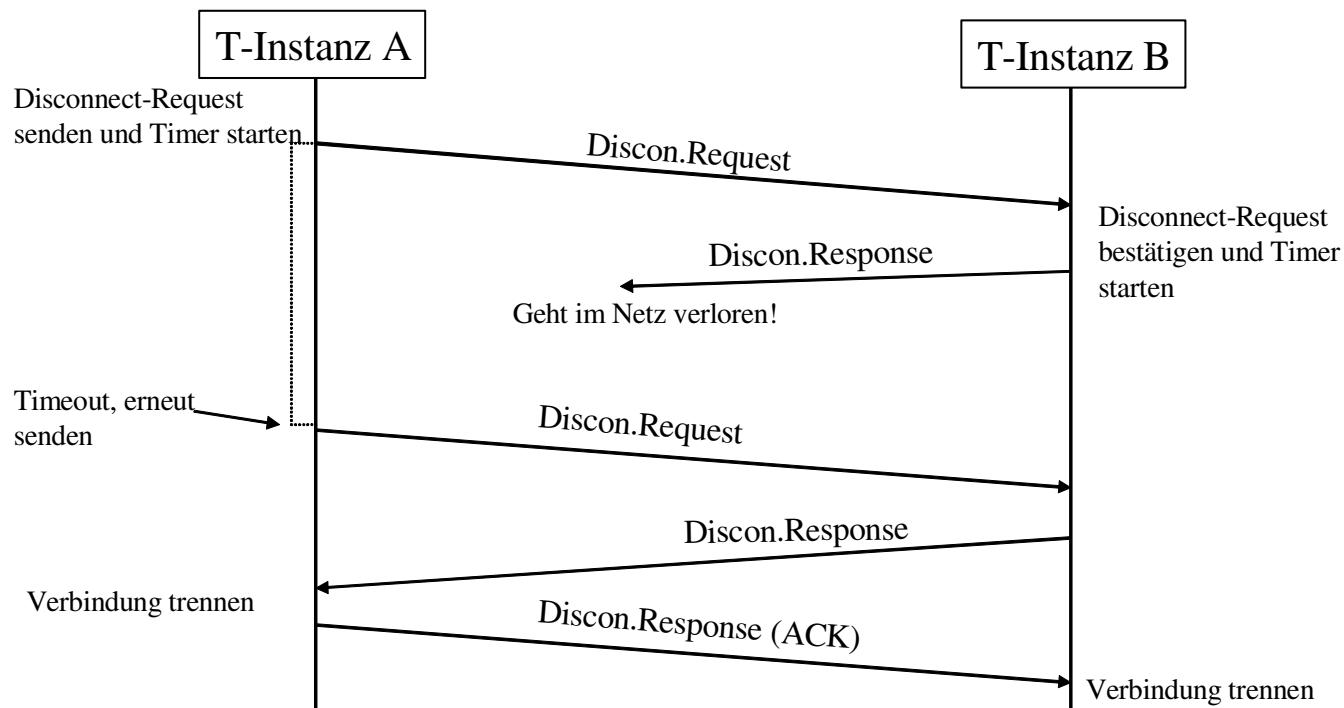
Timerüberwachung beim Verbindungsabbau

Szenario „Timer läuft ab“



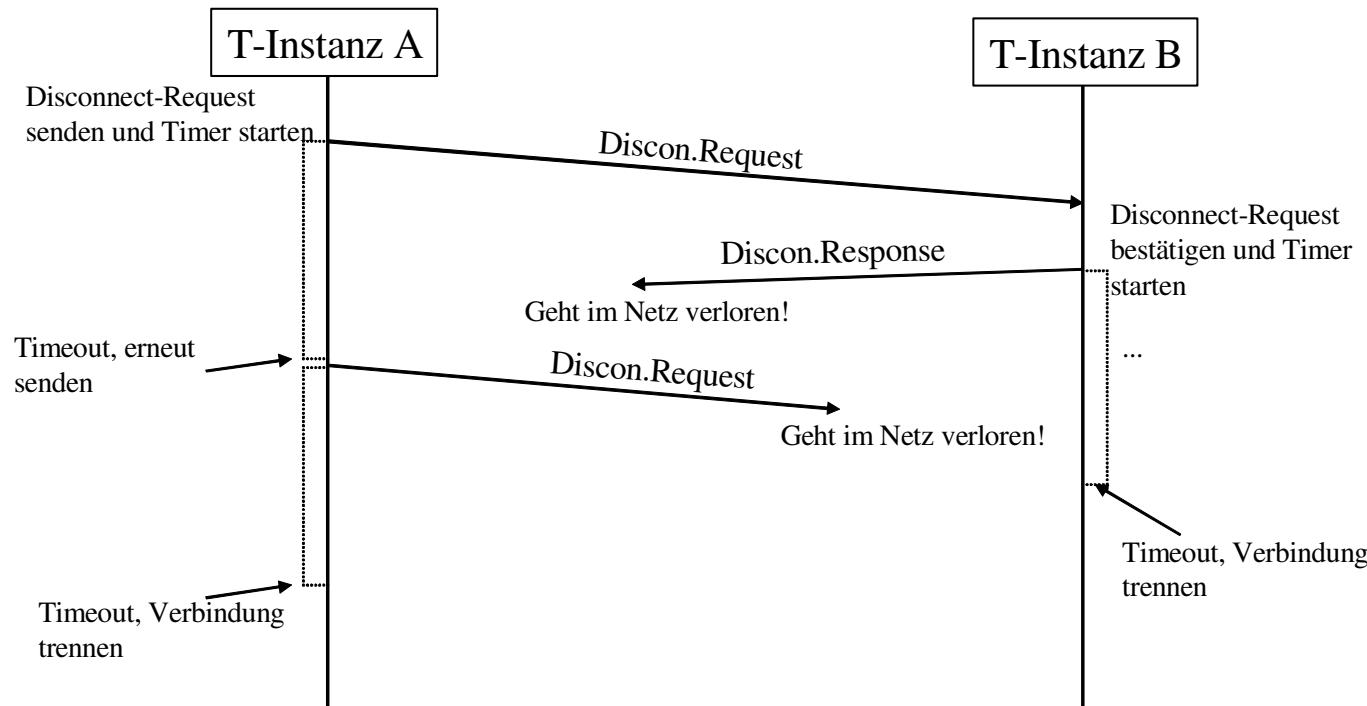
Timerüberwachung beim Verbindungsabbau

Szenario „Disconnect-Response geht verloren“



Timerüberwachung beim Verbindungsabbau

Szenario „Zwei Disconnect-PDUs gehen verloren“



Überblick

1. Einführung
2. Transportorientierte Dienste
 - Überblick und Grundlagen
 - Protokollfunktionen der Transportschicht
 - Verbindungsmanagement
 - Aufbau
 - Abbau
 - **Zuverlässiger Datentransfer**
 - Quittierung
 - Übertragungswiederholung
 - Flusskontrolle
 - Staukontrolle

Zuverlässiger Datentransfer

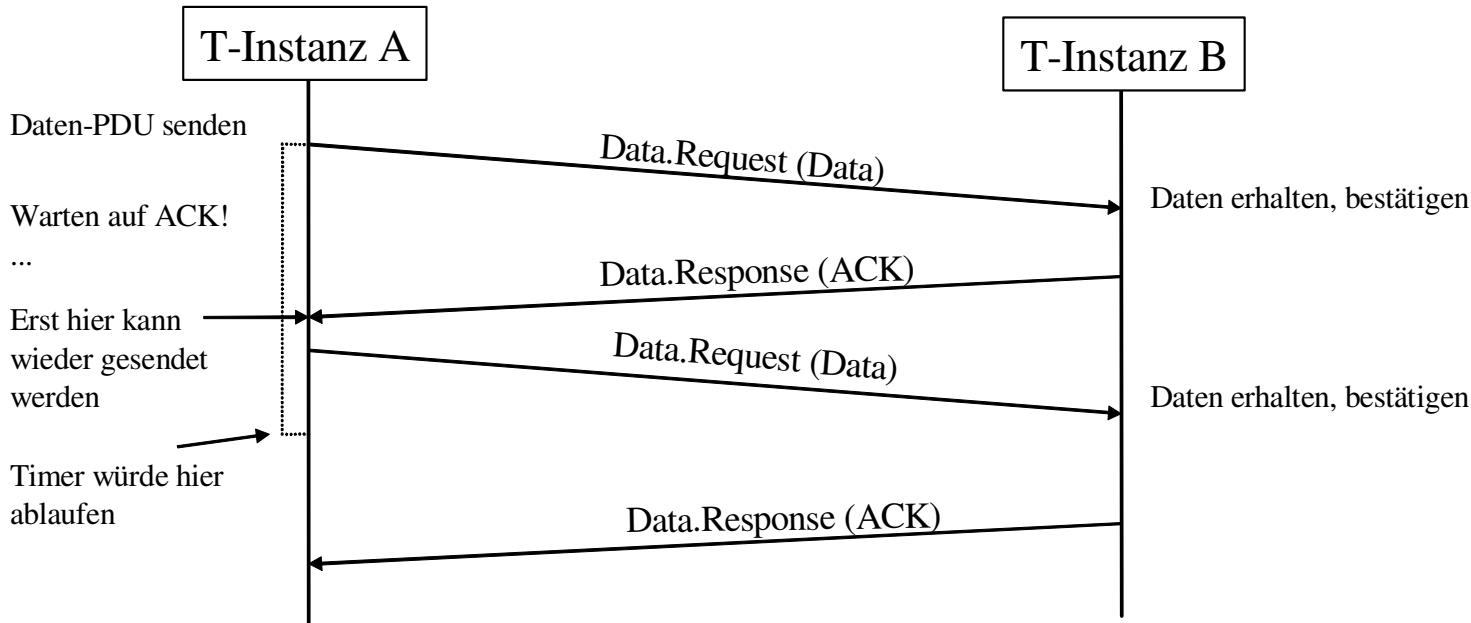
- Was heißt hier zuverlässige Datenübertragung?
 - Garantierte Ausführung ? → Nein!
 - Transaktionssicherheit → Nein!
 - Sicherstellen der Übertragung durch Quittierung und Übertragungswiederholung → **Ja!**
- Quittierungsvarianten
 - Positiv selektiv
 - Positiv kumulativ
 - Negativ selektiv
 - Kombination der Verfahren möglich
- Varianten der Übertragungswiederholung
 - Selektiv
 - Go-back-N

Zuverlässiger Datentransfer

Quittierungsvarianten

■ **Positiv selektives Quittierungsverfahren**

- Stop-and-Go (Stop-an-Wait): Eine Quittung pro gesendeter Nachricht
- Hoher Zusatzverkehr (viele ACK-PDUs)

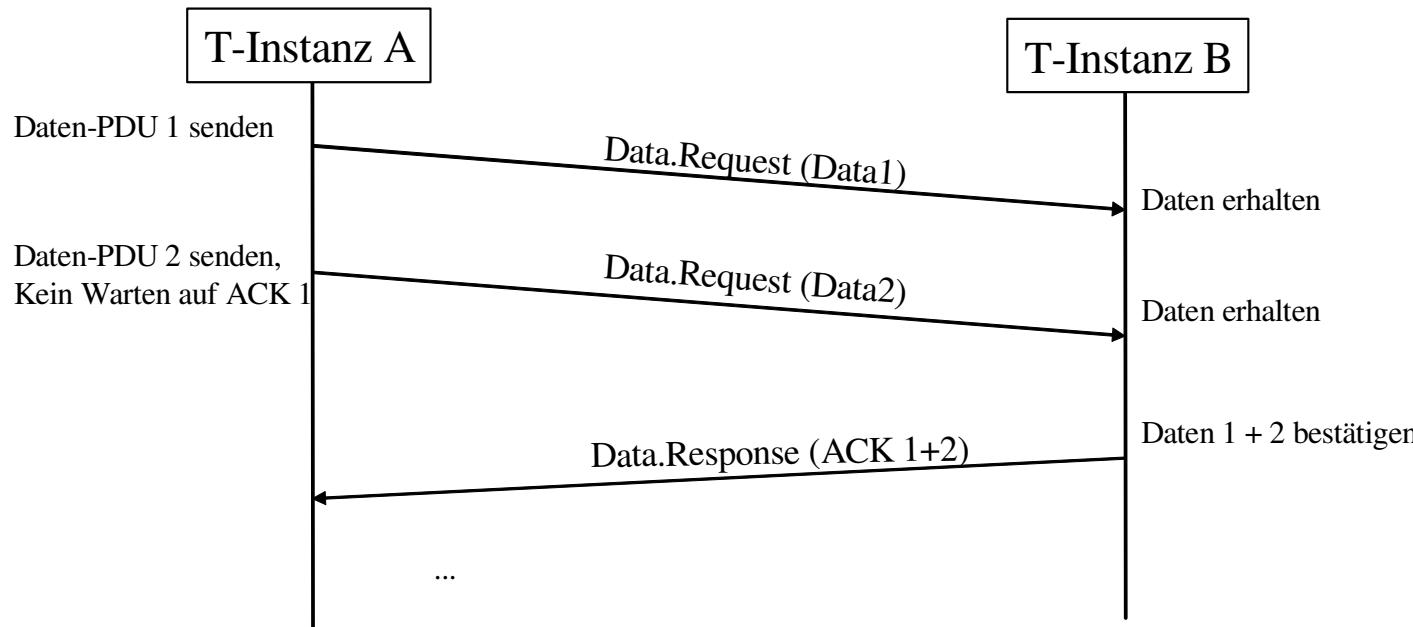


Zuverlässiger Datentransfer

Quittierungsvarianten

■ **Positiv kumulatives** Quittierungsverfahren

- Eine Quittung für mehrere Nachrichten
- Reduzierung der Netzlast
- Nachteil: Verspätete Information an den Sender über Datenverlust

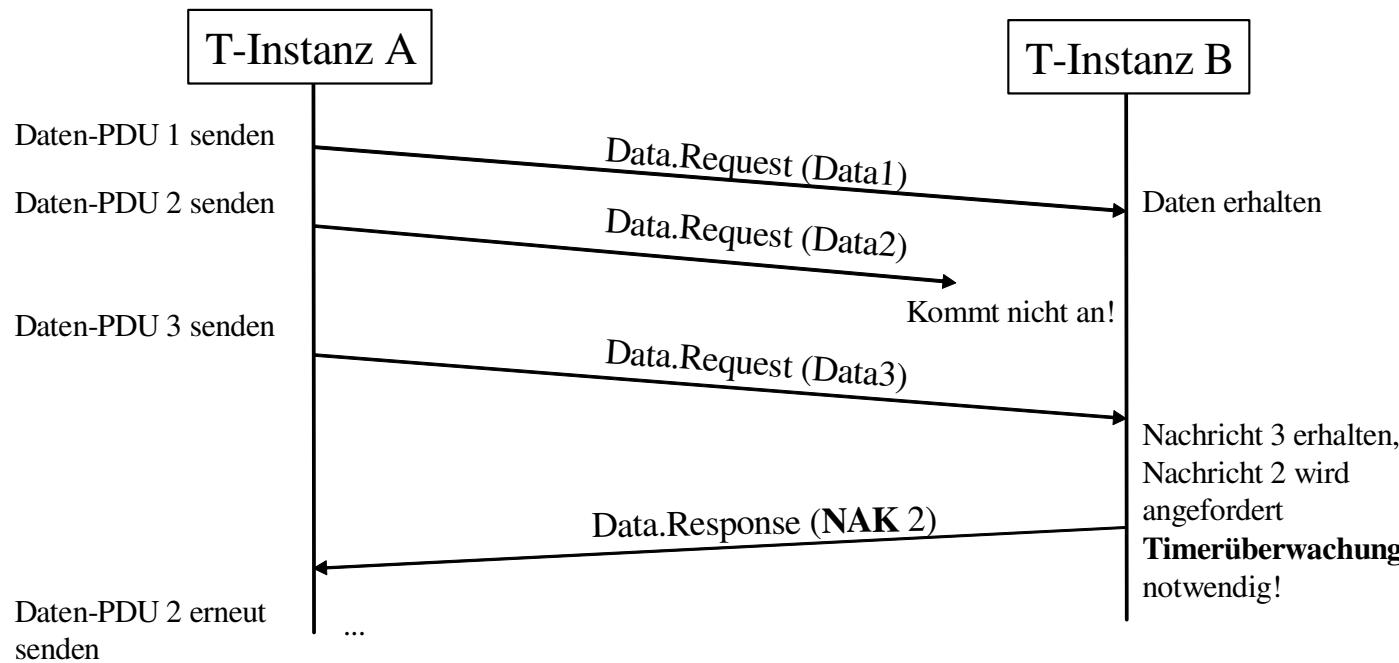


Zuverlässiger Datentransfer

Quittierungsvarianten

■ Negativ selektives Quittierungsverfahren

- Weitere Reduzierung der Netzlast
- Problem: Verlust von Quittungen und dessen Behandlung



Überblick

1. Einführung
2. Transportorientierte Dienste

- Überblick und Grundlagen
- Protokollfunktionen der Transportschicht
- Verbindungsmanagement
 - Aufbau
 - Abbau
- Zuverlässiger Datentransfer
 - Quittierung
 - **Übertragungswiederholung**
 - Flusskontrolle
 - Staukontrolle

Zuverlässiger Datentransfer

Übertragungswiederholung

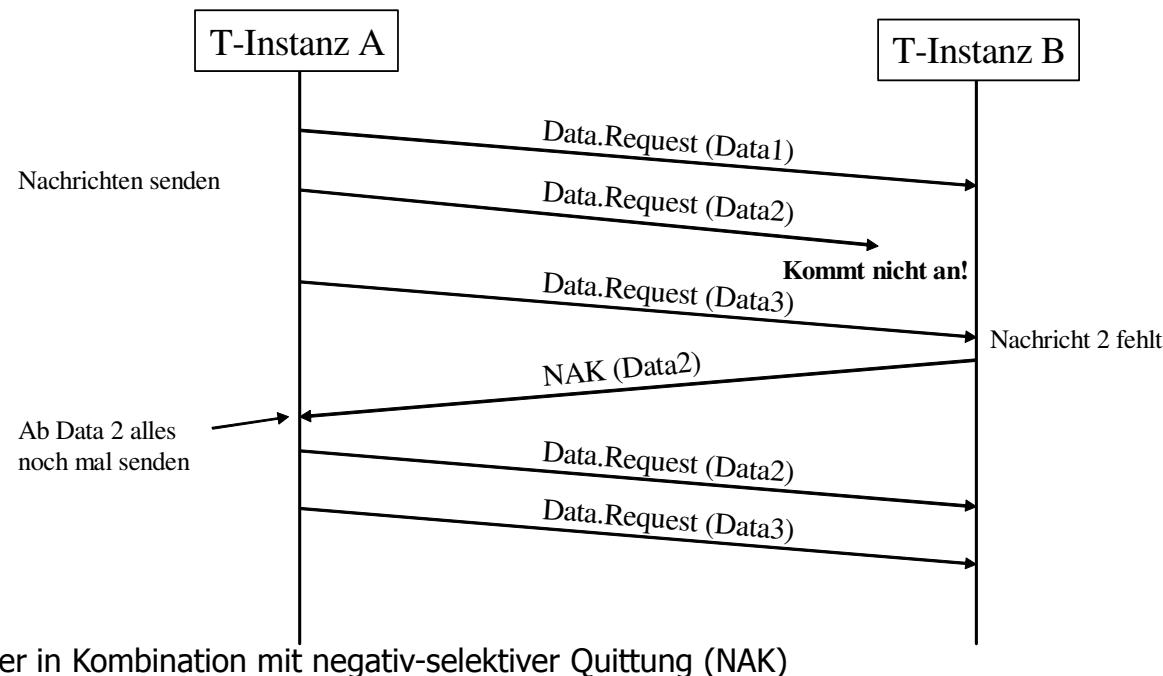
- Generell:
 - Sender muss Nachrichten über einen gewissen Zeitraum zur Übertragungswiederholung bereithalten
 - Man nennt diese Art von Protokollen auch ARQ-Protokolle
 - Automatic Repeat reQuest, = Automatische Wiederholungsanfrage
- **selektiv**
 - Nur die negativ quittierten Nachrichten werden wiederholt
 - Empfänger puffert die nachfolgenden Nachrichten, bis die fehlende Nachricht da ist
 - Reguläre Übertragung kann während der Wiederholung fortgesetzt werden
 - Nachteil: **Große Pufferkapazität** beim Empfänger

Zuverlässiger Datentransfer

Übertragungswiederholung

■ Go-Back-N

- Übertragungswiederholung der fehlerhaften Nachricht sowie aller nachfolgenden
- Die reguläre Übertragung wird unterbrochen
- Vorteil: **Geringe** Speicherkapazität beim Empfänger. **Warum?**



Überblick

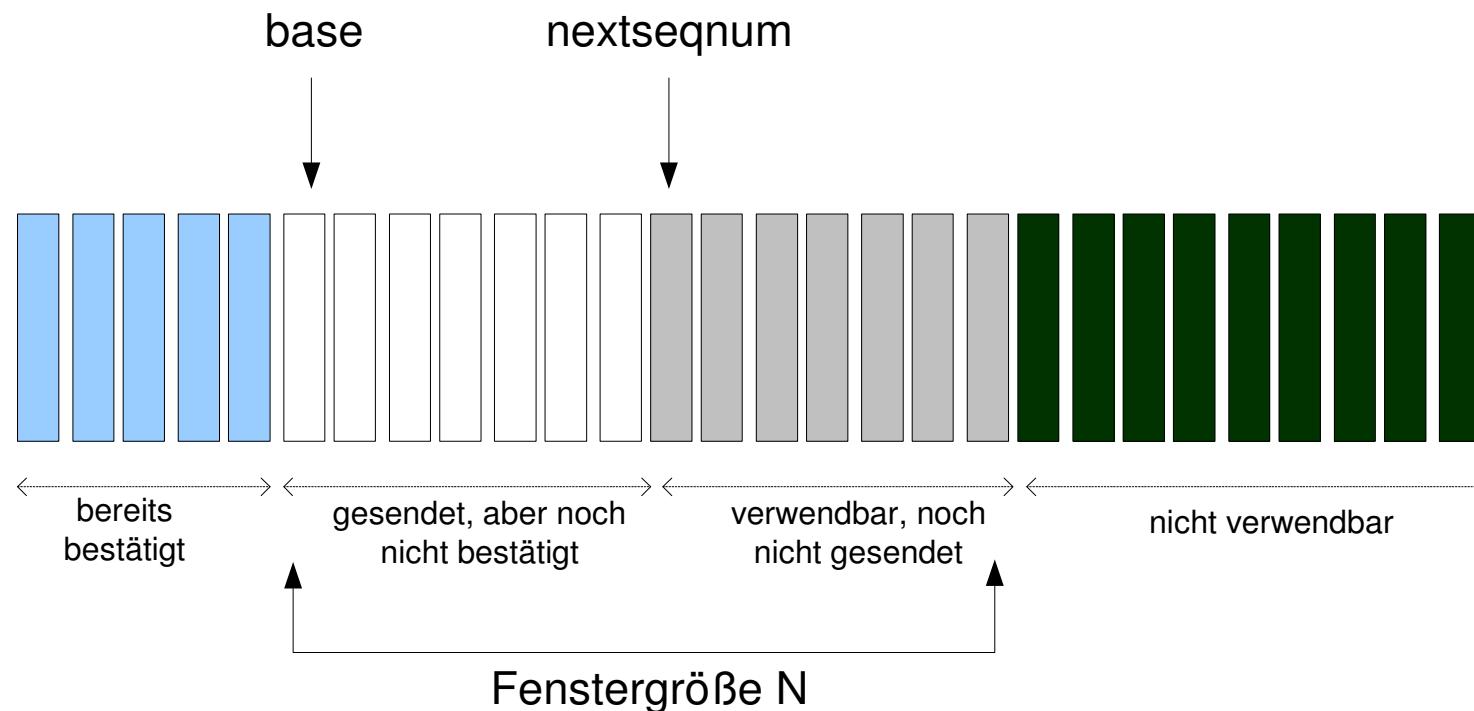
1. Einführung
2. Transportorientierte Dienste
 - Überblick und Grundlagen
 - Protokollfunktionen der Transportschicht
 - Verbindungsmanagement
 - Aufbau
 - Abbau
 - Zuverlässiger Datentransfer
 - Quittierung
 - Übertragungswiederholung
 - **Flusskontrolle**
 - Staukontrolle

Flusskontrolle

- Steuerung des Datenflusses
- Vermeidet **Überlastung des Empfängers**
- Traditionelle Verfahren sind:
 - **Stop-and-Go (Stop-and-Wait)**
 - Einfachstes Verfahren
 - Kopplung von Fluss- und Fehlerkontrolle
 - Nächste Nachricht wird erst nach der Quittierung gesendet
 - **Fensterbasierte Flusskontrolle**
 - Empfänger vergibt sog. **Sendekredit**, also eine max. Menge an Nachrichten oder Bytes, die unquittiert an ihn gesendet werden dürfen
 - Empfänger kann den Sendekredit durch **positive Quittungen** erhöhen
 - Vorteil: Kontinuierlicher Datenfluss und höherer Durchsatz als bei Stop-and-Go möglich

Flusskontrolle

- Sliding-Window-Protokoll: Vier Intervalle
- Bestätigung (ACK) führt zum Weiterrücken des Zeigers *base* und des Fensters



Quelle: Kurose

Überblick

1. Einführung
2. Transportorientierte Dienste
 - Überblick und Grundlagen
 - Protokollfunktionen der Transportschicht
 - Verbindungsmanagement
 - Aufbau
 - Abbau
 - Zuverlässiger Datentransfer
 - Quittierung
 - Übertragungswiederholung
 - Flusskontrolle
 - **Staukontrolle**

Staukontrolle (Überlastkontrolle, Congestion Control)

- Nicht mit Flusskontrolle verwechseln
- Durch Staukontrolle sollen **Verstopfungen** bzw. Überlastungen im Netz **vermieden** werden
- Staukontrolle in der Transportschicht durch **Ende-zu-Ende-Steuerung** zwischen Endsystemen
- Staukontrolle ist ein Mechanismus mit **netzglobalen Auswirkungen**
- Beispiel später: Slow-Start-Verfahren bei TCP

Rückblick

1. Einführung
2. Transportorientierte Dienste
 - Überblick und Grundlagen
 - Protokollfunktionen der Transportschicht
 - Verbindungsmanagement
 - Aufbau
 - Abbau
 - Zuverlässiger Datentransfer
 - Quittierung
 - Übertragungswiederholung
 - Flusskontrolle
 - Staukontrolle

Datenkommunikation

Transportschicht TCP (1)

Wintersemester 2012/2013

Einordnung

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

- 1. Einordnung und Aufgaben des Protokolls**
2. Der TCP-Header
3. Verbindungsauflauf- und abbau
4. Datenübertragung

Zusammenfassung zur Transportschicht

- Logischer Ende-zu-Ende-Transport
- Verbindungslos versus verbindungsorientiert
- Verbindungsmanagement und Adressierung
 - Verbindungsaufbau
 - Verbindungsabbau
- Zuverlässiger Datentransfer
 - Quittierungsverfahren
 - Übertragungswiederholung
- Flusskontrolle
- Staukontrolle

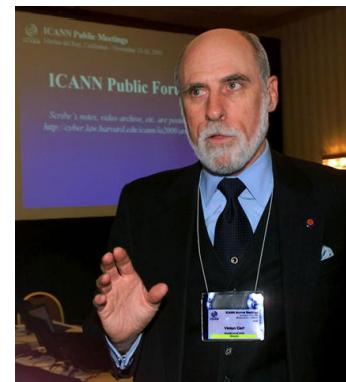
Überblick

- **Einordnung und Aufgaben des Protokolls**
- Der TCP-Header
- Verbindungsauf- und abbau
- Datenübertragung

Robert E. Kahn



Vinton G. Cerf

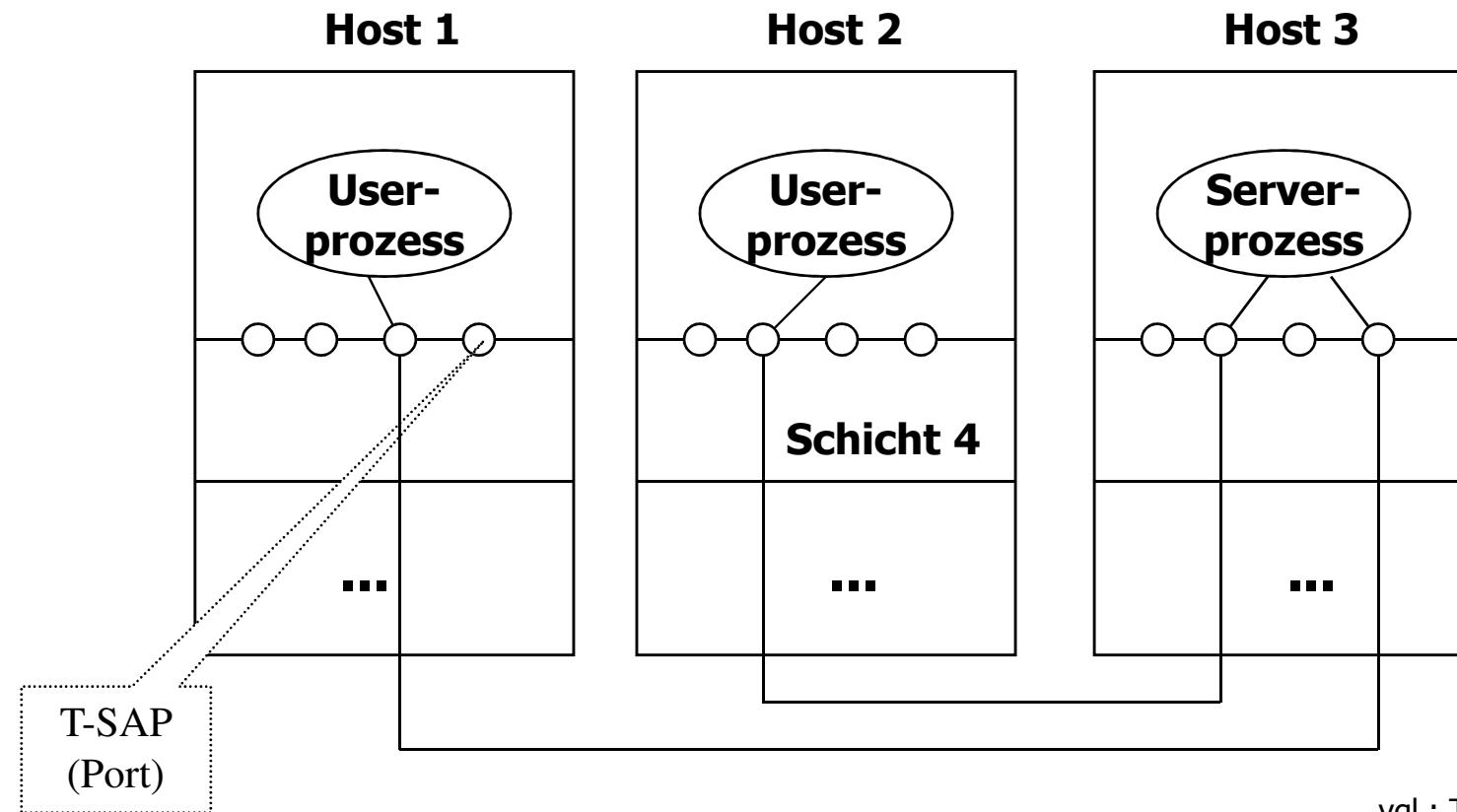


Geschichte:

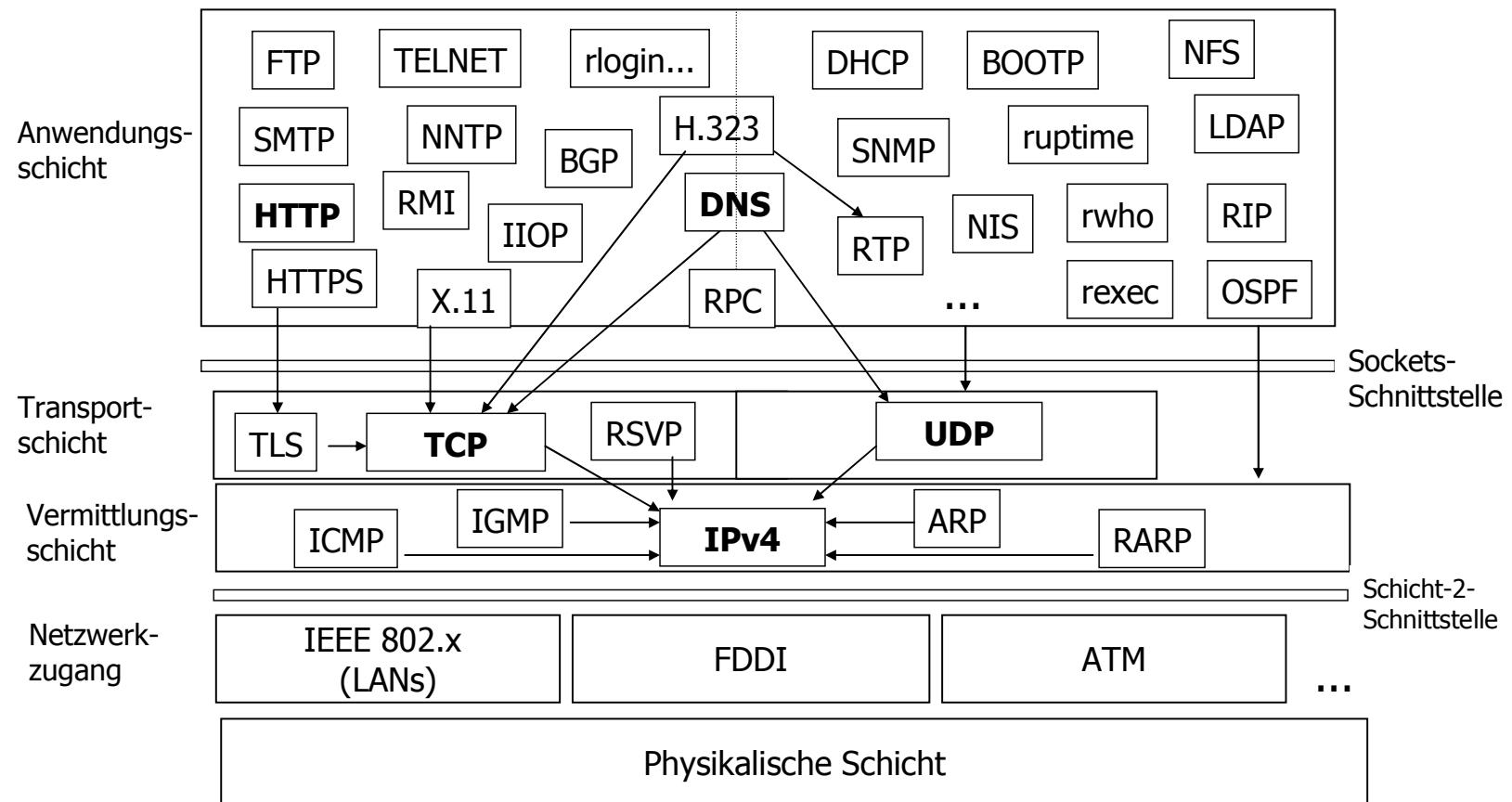
- TCP wurde von Robert E. Kahn und Vinton G. Cerf ab 1972 entwickelt (mehrere Jahre)
- Erste Standardisierung von TCP 1981 im RFC 793

Einordnung und Aufgaben

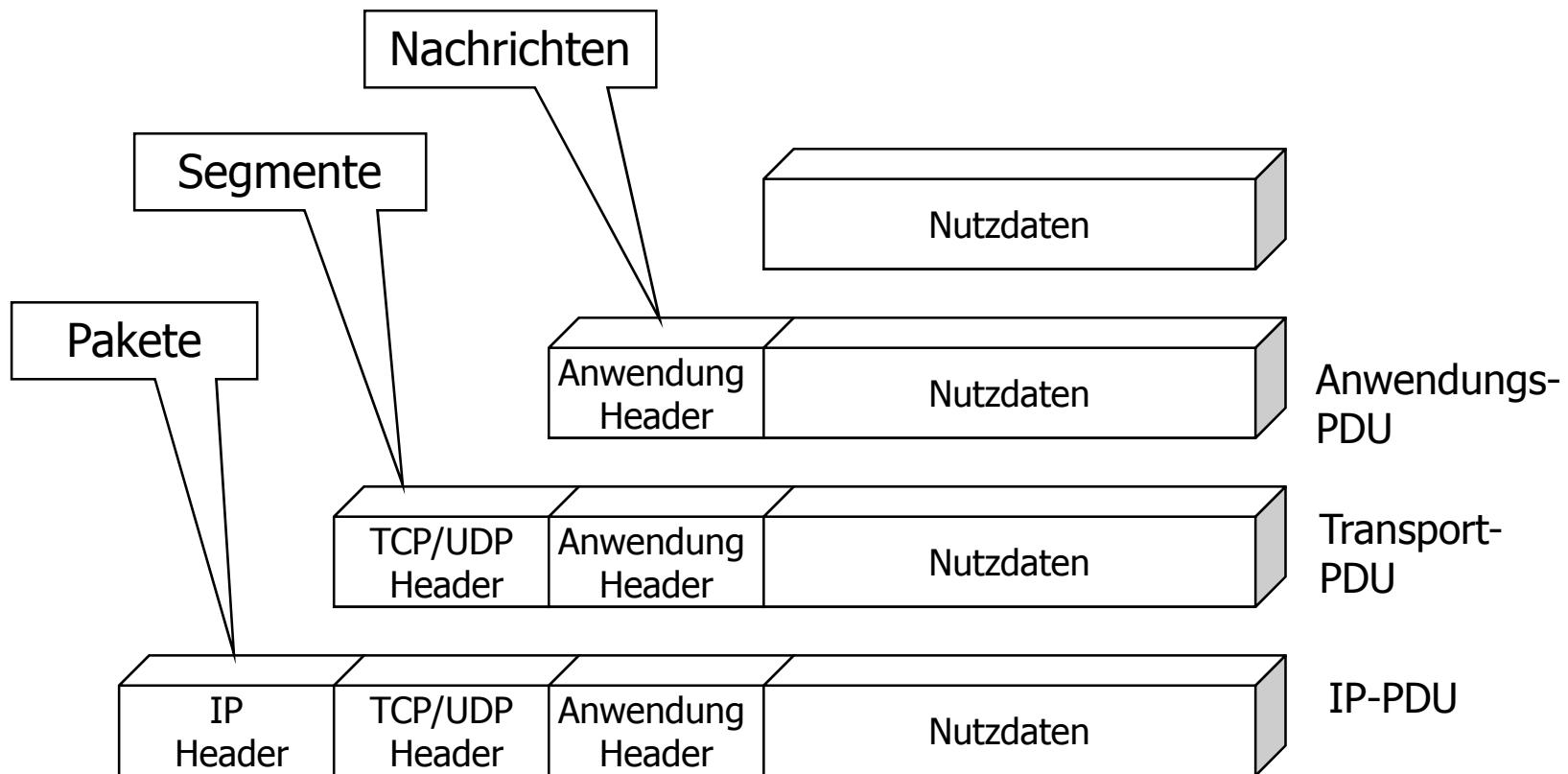
- TCP ermöglicht eine **Ende-zu-Ende Beziehung** zwischen kommunizierenden Anwendungsinstanzen



Erinnerung: TCP/IP-Protokollfamilie



Erinnerung: TCP/IP-Referenzmodell, Protokollkapselung



PDU = Protocol Data Unit

Einordnung und Aufgaben

- TCP ist weit **verbreitet**
 - Hersteller haben TCP/IP als **Industrienorm** akzeptiert
 - Genormt in RFC 793, aktualisiert in RFC 1122, 1349,...
- TCP ist **offen**, nicht proprietär
 - also nicht an einen Hersteller gebunden
- TCP ist **skalierbar**
 - Ein bestehendes Netz auf TCP/IP-Basis kann relativ einfach um weitere Rechner erweitert werden; die neuen Rechner benötigen nur IP-Adressen
- TCP/IP ist die **Grundlage des Internets**
 - TCP/IP ist Bestandteil von UNIX
 - Das Internet basiert auf TCP/IP

Einordnung und Aufgaben

- TCP ermöglicht die Kommunikation
 - über **vollduplex-fähige, virtuelle Verbindungen** zwischen Anwendungen
- **Stream-orientierte** Kommunikation im Unterschied zur blockorientierten Übertragung (siehe OSI TP4)
- TCP bietet eine **zuverlässige Ende-zu-Ende-Verbindung** zwischen Stationen
- TCP verwendet IP als Grundlage des Pakettransports

Aufgaben im Detail

- Schaffung einer gesicherten Ende-zu-Ende-Verbindung auf Basis von IP
- Reihenfolgegarantie
- Garantierte Auslieferung
- Staukontrolle
- Flusskontrolle (Vermeidung von Überschwemmungen langsamer Empfänger)
- Multiplexing und Demultiplexing der IP-Verbindung
- Fragmentierung und Defragmentierung der Nachrichten (Segmente!)

Einordnung und Aufgaben

- **Maßnahmen zur Sicherung** der Übertragung:
 - Drei-Wege-Handshake für Verbindungsmanagement
 - Prüfsumme
 - Quittierung (ACK-PDU): Positiv-kumulativ und implizites NAK
 - Zeitüberwachung für jedes Segment (Timer) und Nachrichtenwiederholung
 - Go-Back-N für die Nachrichtenwiederholung
 - Sequenznummern = Folgenummern für die Reihenfolgeüberwachung
 - Sliding Windows Prinzip zur Flusskontrolle
 - Slow-Start-Verfahren zur Staukontrolle

Dienste, Ports und Adressierung

- Anwendungsprozess kommuniziert über eine Adresse, die als **Socket** bezeichnet wird
 - Tupel der Form (IP-Adresse, TCP-Portnummer)
- **Well-known Ports**
 - 16-Bit-Integer
 - Es gibt hier eine Reihe von sog. well-known Ports für reservierte Services (Portnr. ≤ 1024)
 - Jeder Dienst hat eine eigene Portnummer (siehe Datei /etc/services unter Unix)
- Ein Anwendungsprozess kann auch mehrere Verbindungen unterhalten
- Client-Server-Prinzip leicht realisierbar:
 - Server wartet an einem Port auf Connect-Requests

Dienste, Ports und Adressierung

- Eine Verbindung wird durch ein Paar von Endpunkten identifiziert (**Socket Pair**)
 - Dadurch ist es möglich, dass **ein TCP-Port** auf einem Host für **viele Verbindungen** genutzt werden kann
 - Beispiel:
 - HTTP-Port 80 wird für viele Verbindungen eines HTTP-Servers verwendet. TCP-Verbindungen aus Sicht des HTTP-Servers sind:
 - ((195.214.80.76, **80**) (196.210.80.10,6000))
 - ((195.214.80.76, **80**) (197.200.80.11,6001))
 - ...
- wobei 195.214.80.76 die IP-Adresse des Servers ist.

Einschub: Vorgaben für Portnutzung

- ICANN-Vorgabe für die Ports:
 - Well Known Ports (0 – 1023)
 - Registered Ports (1024 – 49151)
 - Ports, die Hersteller für Anwendungen reservieren können
 - Dynamic and/or Private Ports (49152 – 65535)
 - Private Ports, beliebig vergeben
 - Siehe Datei [http://www.iana.org/assignments/port-numbers!](http://www.iana.org/assignments/port-numbers)

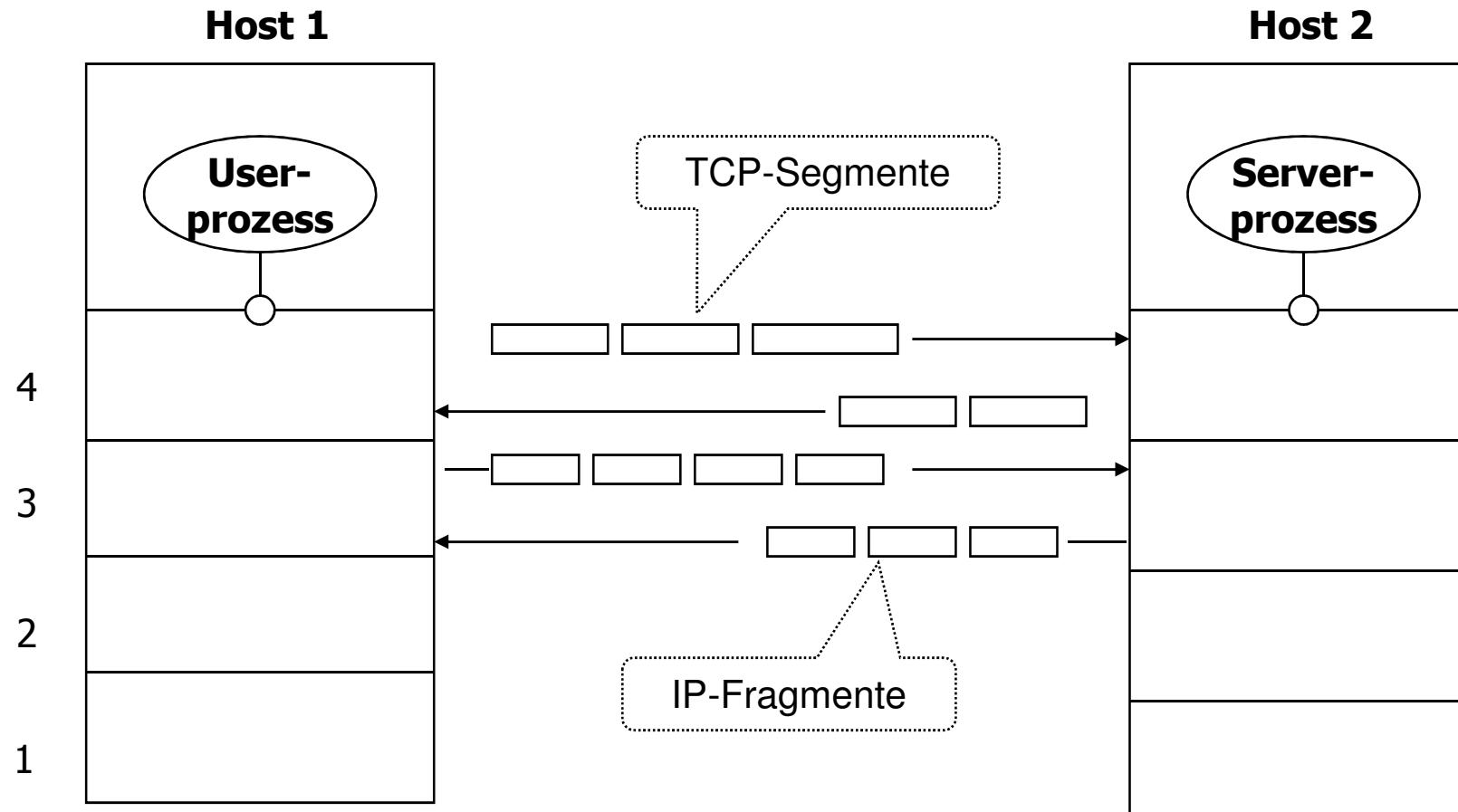
Beispiele für Dienste und Ports

TCP-Portnummer	Protokoll, Service
23	Telnet – Remote Login
20,21	ftp – File Transfer Protocol
25	SMTP – Simple Mail Transfer Protocol
80	HTTP

Überblick

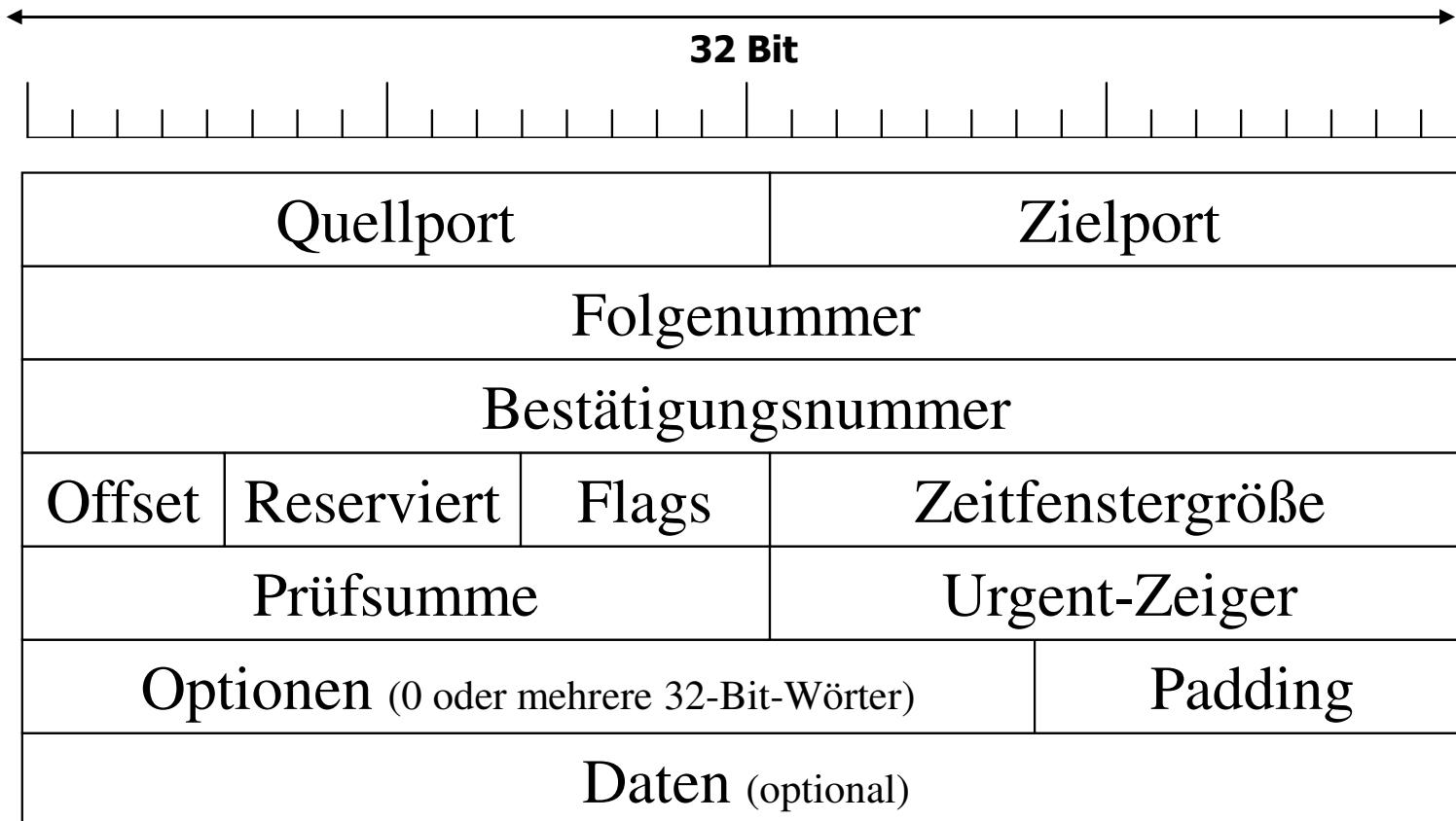
1. Einordnung und Aufgaben des Protokolls
- 2. Der TCP-Header**
3. Verbindungsauflauf- und abbau
4. Datenübertragung

TCP-Segmente



- Byte-Strom wird in TCP-Segmente gepackt
- TCP-Segmentierung \neq IP-Fragmentierung

TCP-Header (PCI, Protocol Control Information)



TCP-Segmente

- TCP sieht den Datenstrom als eine Sequenz von Octets (Bytes) und unterteilt diese zur Übertragung in **Segmente**
- Ein Segment besteht aus einem mind. 20 Bytes langen Header
- Es bleiben max. $65.535 - 20$ (TCP-Header), also **65.515 Bytes** für die Daten in einem Segment
- **Maximum Segment Size (MSS)** kann optional ausgehandelt werden
 - MSS-Option
 - Nicht verwechseln mit Window-Größe für Flusskontrolle, verlängerbar über WSOpt-Option, bis 1 GB

TCP-Header

- **Quell- und Zielport**
 - Portnummer des Anwendungsprogramms des Senders und des Empfängers
- **Folgenummer**
 - Nächstes Byte innerhalb des TCP-Streams ($\text{mod } 2^{32}$)
- **Bestätigungsnummer**
 - Gibt das als nächstes erwartete Byte im TCP-Strom an und bestätigt damit den Empfang der vorhergehenden Bytes
- **Offset**
 - Gibt die Länge des TCP-Headers in 32-Bit-Worten an
- **Reserviert**
 - Hat noch keine Verwendung

TCP-Header

- **Flags:** Hier handelt es sich um Kontroll-Bits mit unterschiedlicher Bedeutung:

Flag	Bedeutung
URG	Urgent-Zeiger-Feld ist gefüllt
ACK	Bestätigung (z.B. bei Verbindungsaufbau genutzt), d.h. die ACK-Nummer hat einen gültigen Wert
PSH	Zeigt Push-Daten an, Daten dürfen beim Empfänger nicht zwischengespeichert werden, sondern sind sofort an den Empfängerprozess weiter zu leiten
RST	Dient zum <ul style="list-style-type: none">- Rücksetzen der Verbindung (sinnvoll z.B. bei Absturz eines Hosts)- Abweisen eines Verbindungsaufbauwunsches- Abweisen eines ungültigen Segments
SYN	Wird genutzt beim Verbindungsaufbau
FIN	Wird genutzt beim Verbindungsabbau

TCP-Header

- **Zeitfenstergröße**
 - Erlaubt dem Empfänger, mit ACK dem Sender den vorhandenen Pufferplatz in Byte zum Empfang der Daten mitzuteilen
- **Urgent-Zeiger**
 - beschreibt die Position (Byteversatz von der aktuellen Folgenummer ab) an der dringliche Daten vorgefunden
 - Diese Daten werden vorrangig behandelt
 - Selten genutzt!

TCP-Pseudoheader

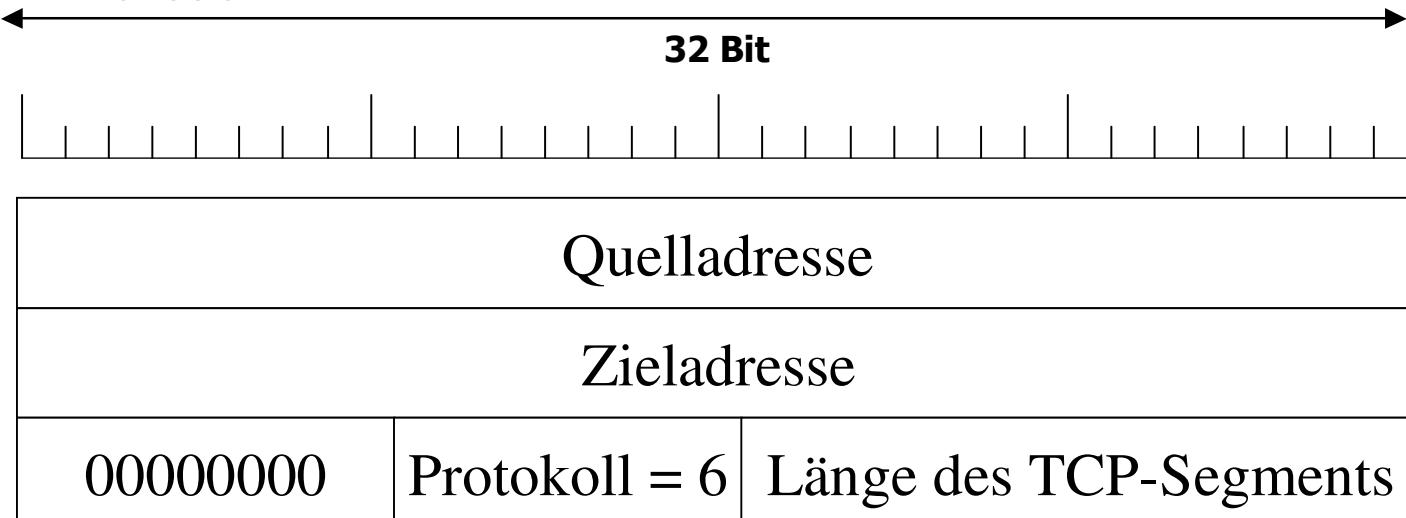
- **Prüfsumme – Berechnung**
 - Verifiziert das TCP-Segment (Header + Daten) inkl. eines Pseudoheaders auf Basis eines einfachen Prüfsummenalgorithmus:
 - Prüfsumme im Header auf Null setzen
 - Nutzdaten ggf. auf gerade Byteanzahl mit einem Nullbyte ergänzen
 - Summe über alle 16-Bit-Wörter der ganzen Nachricht inkl. TCP-Header und sog. Pseudo-Header bilden
 - Danach Bildung des Einer-Kompliments aus der Summe ergibt die Prüfsumme

TCP-Pseudoheader

- **Pseudoheader:**

- Wird vor der Berechnung der Prüfsumme an das TCP-Segment (vor den TCP-Header) gehängt , aber nicht mit übertragen

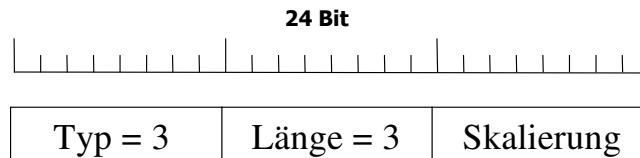
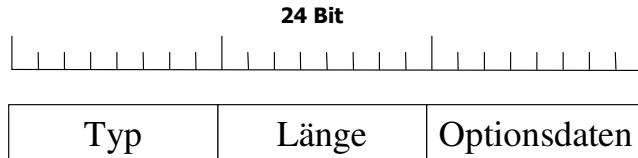
- Aufbau:



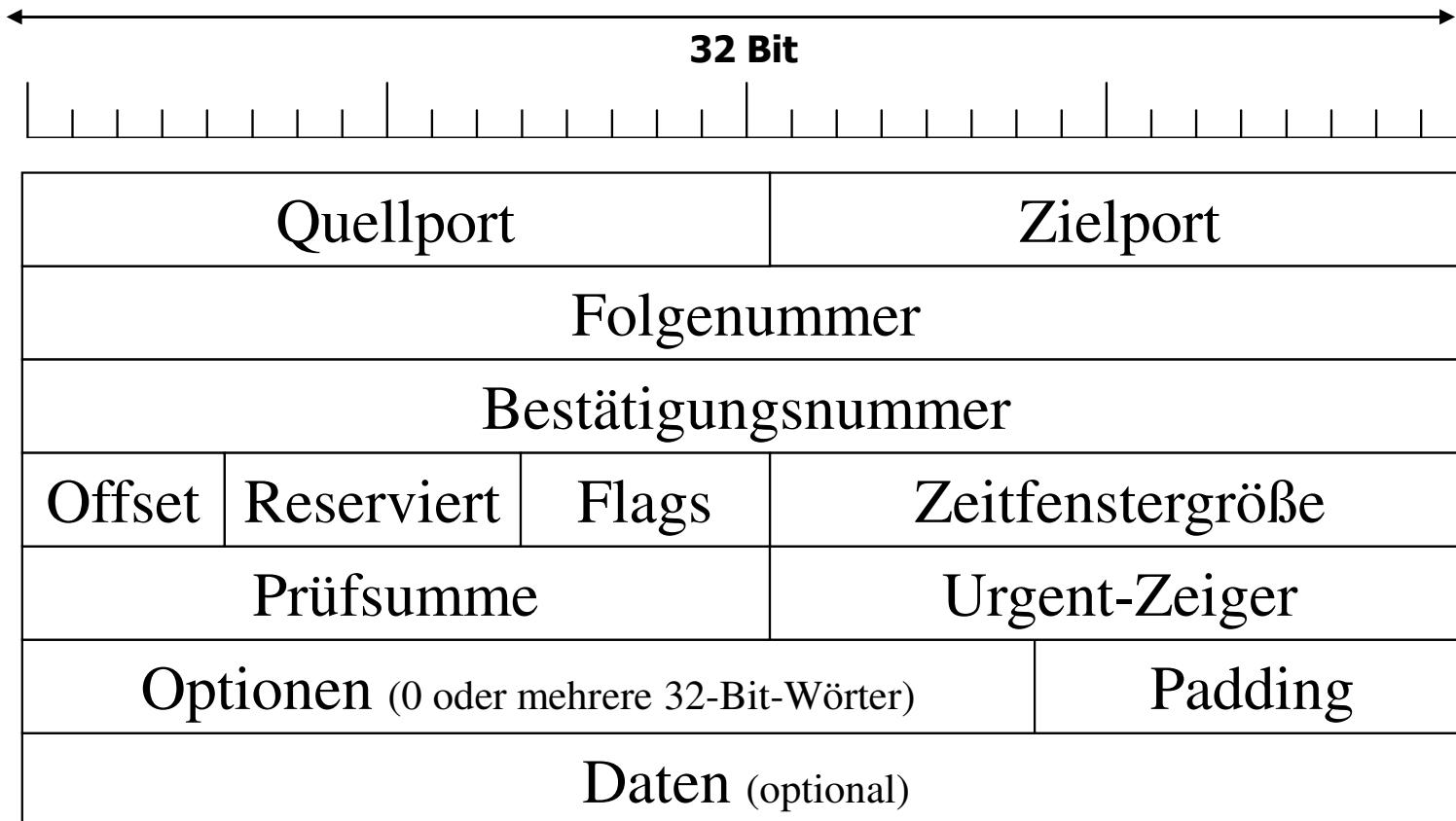
TCP-Header

■ Optionen

- Wurden früher relativ selten verwendet
- Bestehen entweder aus einem einzelnen Byte (Optionsnummer) oder haben eine variable Form
- Optionen sind z.B.:
 - MSS: Maximum Segment Size der Verbindung einstellen
 - *SACKOK*: Selektive Wiederholungen anstelle von *go back n* einstellen
 - *WSOPT (Windows-Scale-Option)*: Maximale Fenstergröße verlängern (2^{30} Byte max. möglich)



TCP-Header (PCI, Protocol Control Information)



Überblick

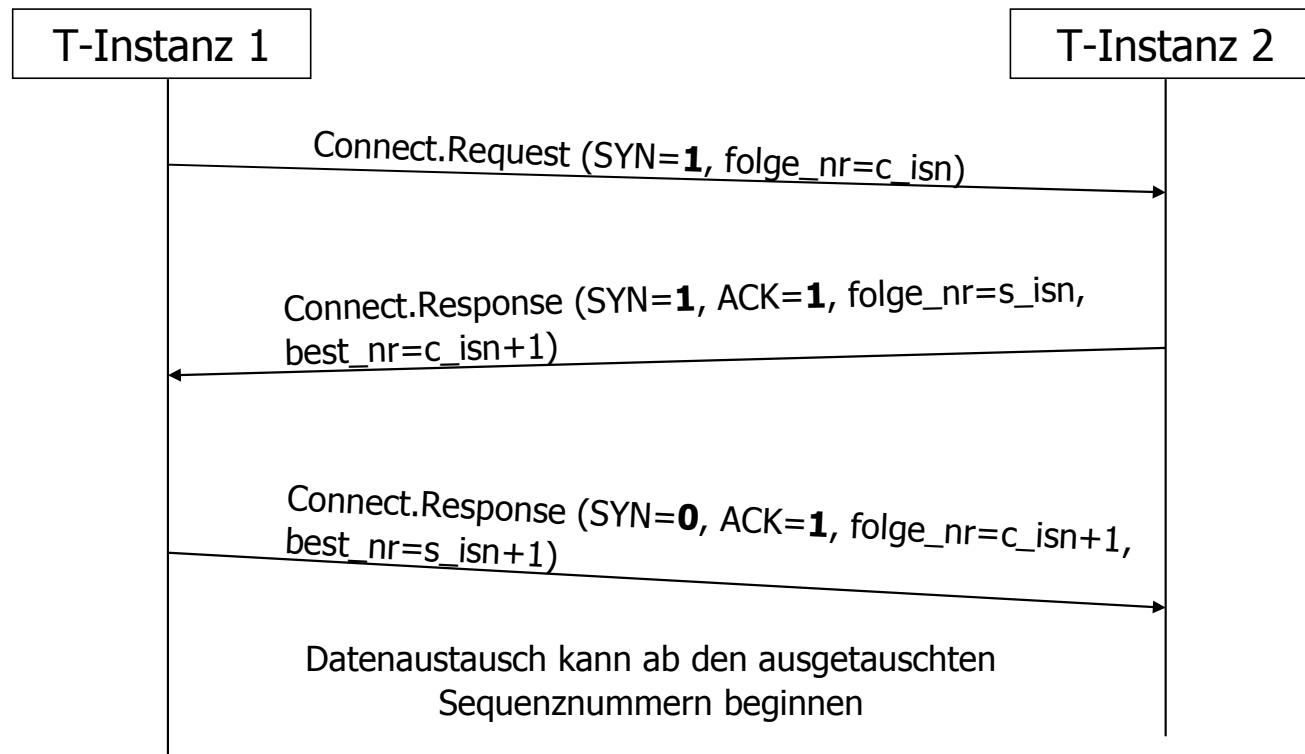
1. Einordnung und Aufgaben des Protokolls
2. Der TCP-Header
- 3. Verbindungsauflauf- und abbau**
4. Datenübertragung

Verbindungsauftbau: Parameter aushandeln und Drei-Wege-Handshake

- Beim Verbindungsauftbau werden die **MSS** und die **Sequenznummern** ausgehandelt und ggf. weitere Einstellungen (Optionen) ausgehandelt
- Verwendung des Drei-Wege-Handshake-Mechanismus
 - Initiale Sequenznummern werden berechnet und ausgetauscht
- Kollision beim Verbindungsauftbau ist möglich:
 - Zwei Hosts versuchen gleichzeitig eine Verbindung mit gleichen Parametern zueinander aufzubauen
 - Es wird nur eine TCP-Verbindung aufgebaut

Verbindungsauftbau: Protokoll

- Normaler Ablauf



c_isn = Initial Sequence Number des Clients (Instanz 1)
s_isn = Initial Sequence Number des Servers (Instanz 2)

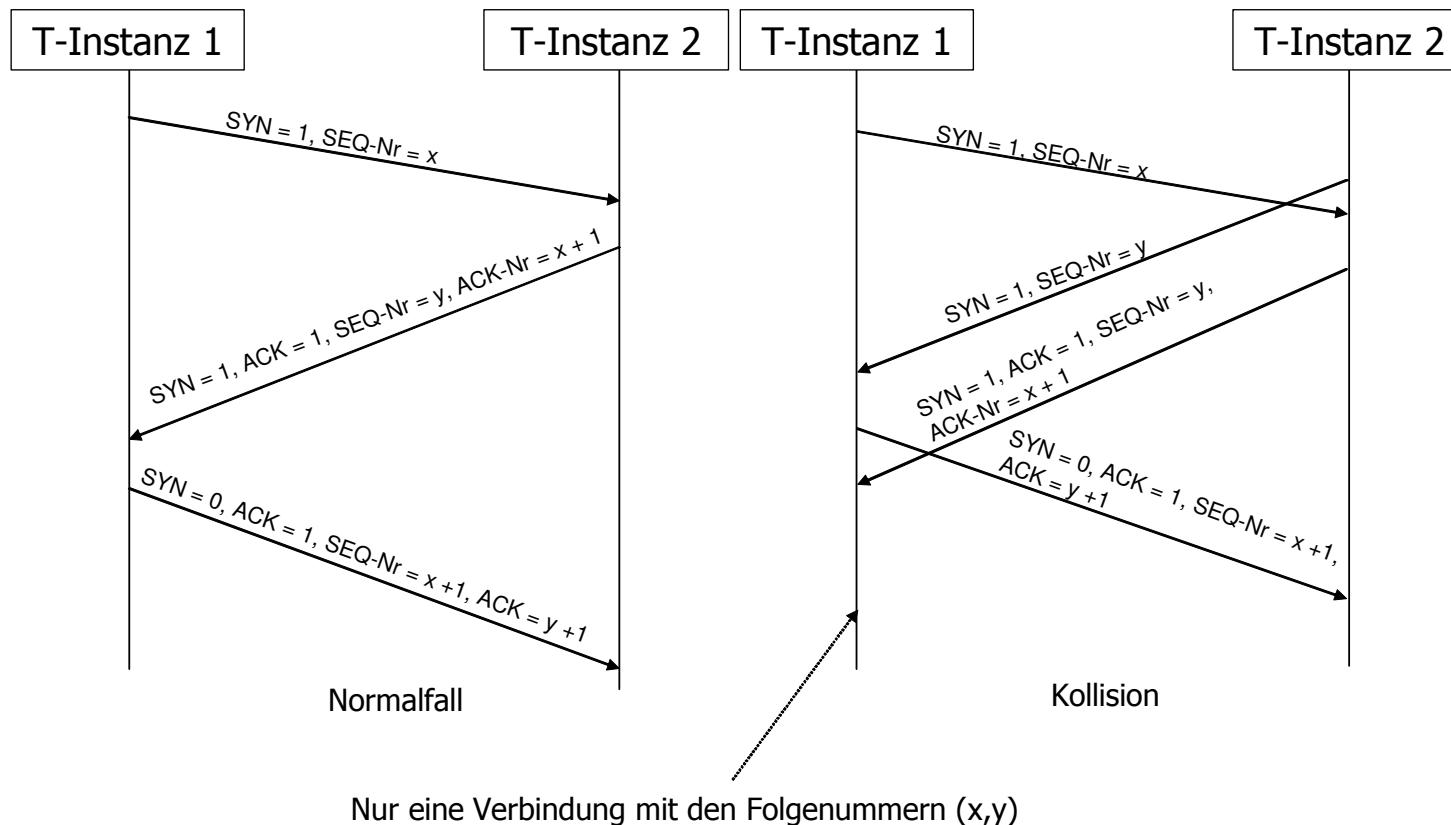
Verbindungsauftbau

Fehlerszenarien

- TCP muss mehrere Fehlersituationen richtig bearbeiten
 - Gleichzeitiger Verbindungsauftbauversuch beider Partner darf nur zu einer Verbindung führen
 - Alte Duplikate von TCP-Segmenten werden beim Verbindungsauftbau empfangen
 - Reset der Verbindung (RST-Bit) und erneuter Aufbau
 - Halb-offene Verbindung erkennen
 - Mit Reset (RST-Bit) abbauen
 - ...

Verbindungsauftbau Kollisionsfall

- Normalfall und Kollision (gleichzeitiger Verbindungsauftbau) im Vergleich



Verbindungsaufbau Kollision der Sequenznummern (1)

■ Problem:

- Erneuter, schneller Verbindungsauftakt nach Crash könnte zu Sequenznummern-Kollision führen
- Ein noch altes TCP-Segment könnte bei neuer „Inkarnation“ der Verbindung (gleiche Adressparameter) ankommen und nicht erkannt werden
- Das muss verhindert werden

Verbindungsaufbau

Kollision der Sequenznummern (2)

▪ Lösung zur Synchronisation der Sequenznummern:

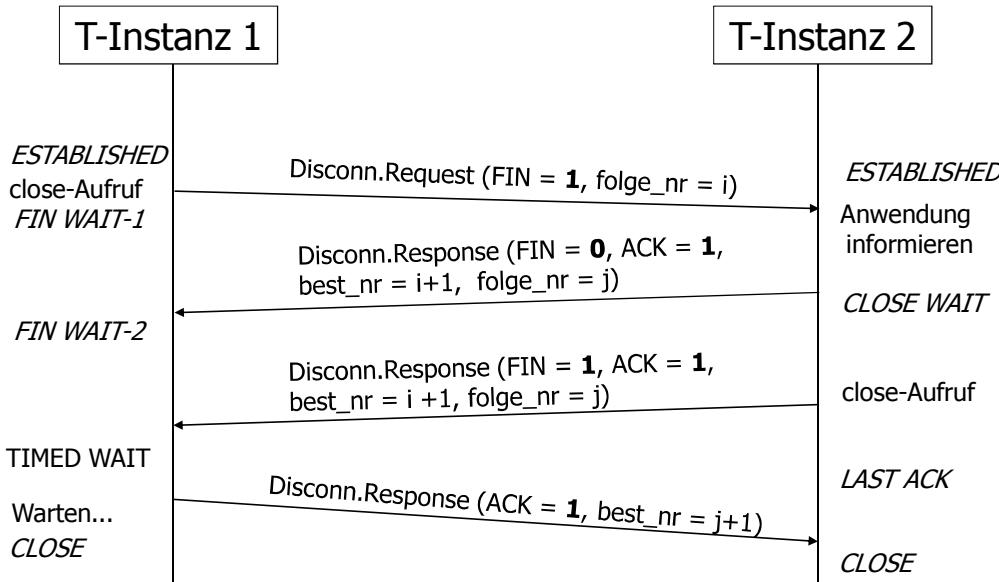
- Keine netzweit globale Uhr zur Bestimmung der Sequenznummer
- **Keine feste Vorgabe**, wie ISN ermittelt wird → auch implementierungsabhängig
- Generierung der ISNs anhand eines (fiktiven) Zeitgebers
 - Vorschlag im RFC 793, S. 28: Zyklus des Zeitgebers von 4,55 Stunden, Zeitgeber wird alle 4 ms erhöht
- Einsatz des 3-Way-Handshake zum Synchronisieren der Sequenznummern beim Verbindungsauftbau
- **Max. Segmentlebensdauer (MSL)** berücksichtigen: Wurde im RFC 792 auf 2 Minuten festgelegt
 - Diese Zeit muss gewartet werden, bevor nach einem Crash einer Verbindung eine neue ISN zugewiesen wird
 - Schwierig zu implementieren ohne persistente Speicherung der Verbindungsdaten!

Verbindungsabbau Protokoll (1)

- Verbindungsabbau-Protokoll:
 - Modifizierter Dreiwege-Handshake-Mechanismus
 - Jede der beiden Verbindungen der Vollduplex-Verbindung wird abgebaut, d.h. beide Seiten bauen ihre „Senderichtung“ ab
- Ablauf:
 - Aktiv abbauender Partner sendet zunächst ein Segment mit **FIN=1**
 - Passiver Partner antwortet zunächst mit einem **ACK** und informiert die Anwendung (Signalisierung implementierungsabhängig!)
 - Wenn die Anwendung **close** aufruft, sendet die Partnerinstanz ebenfalls ein Segment mit **FIN=1**
 - Aktiver Partner sendet abschließend ein Segment mit **ACK=1**

Verbindungsabbau Protokoll (2)

- Client baut die Verbindung ab (auch Server kann es)
- Alle Segmente mit Folgenummer < i bzw. j sind noch zu verarbeiten
- FIN-Segment zählt Sequenznummer **um 1 hoch** (zählt als 1 Datenbyte)



Zustände im TCP-Zustandsautomat:
ESTABLISHED, FIN WAIT-1, FIN WAIT-2, TIMED WAIT, CLOSE, CLOSE WAIT, LAST ACK

Verbindungsabbau

Signalisierung beim passiven Partner

- Laut Zustandsautomat gibt es **mehrere Varianten** (genau 4) des Verbindungsabbaus
 - Es geht auch mit drei Segmenten (FIN=1 + ACK=1)
- Die **Signalisierung** des Verbindungsabbaus an die Anwendung ist der Implementierung überlassen und im RFC nicht genau beschrieben
 - Es kann eine Weile dauern, bis die Anwendung auf das close-Ereignis reagiert
 - Anwendung kann evtl. sogar eine Benutzereingabe erfordern
 - Dies hängt vom Programm ab
- Auch **abnormale Beendigung** einer Verbindung ist möglich
 - Segment mit **RST-Bit=1** wird gesendet und der Empfänger bricht die Verbindung sofort ab

Überblick

1. Einordnung und Aufgaben des Protokolls
2. Der TCP-Header
3. Verbindungsauf- und abbau
- 4. Datenübertragung**

Tool zum Mitschneiden und Analysieren des Nachrichtenverkehrs

→ Ethereal/Wireshark-Sniffer

Sequenznummern und Quittierung

- Einsatz von Sequenznummern, die auf einzelnen Bytes, nicht auf TCP-Segmenten basieren (siehe Header->**Sequenznummer**)
- Die Sequenznummer enthält die Nummer des nächsten erwarteten Bytes
- Alle gesendeten Bytes werden vom Empfänger im Feld **Bestätigungsnummer kumulativ** quittiert
- Selektive, positive Quittierung auch möglich:
Vorschlag in einem eigenen RFC

Sequenznummern und Quittierung

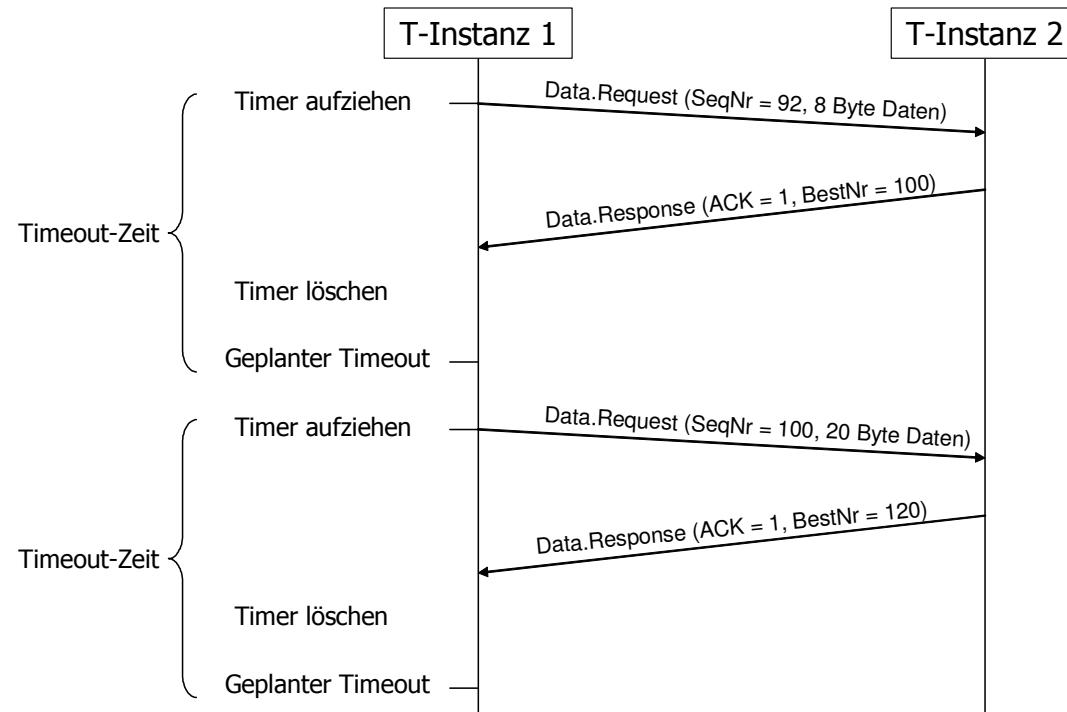
- Die Bestätigung muss von der empfangenden TCP-Instanz **nicht unbedingt sofort** gesendet werden, wenn noch Platz im Puffer ist
 - Hier besteht Implementierungsfreiheit für die Hersteller von TCP/IP-Stacks

Einschub:

- Sequenznummern-Anzahl bei TCP: 2^{32} (32-Bit-Feld)
- Vergabe mod 2^{32}
- Bei 64 Kbit/s kommt es frühestens nach 6,2 Tagen zu einer Wiederholung
- Bei 100 Mbit/s kommt es frühestens nach 340 s zu einer Wiederholung
- Bei 1 Gbit/s kommt es frühestens nach 34 s zu einer Wiederholung

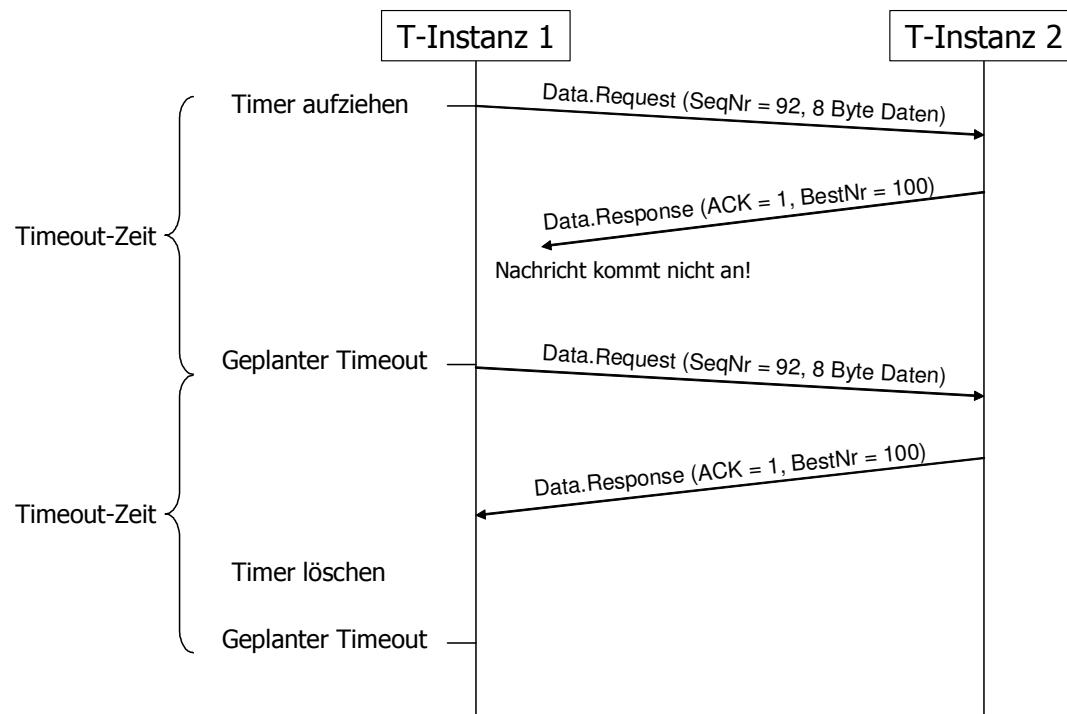
Szenario: Erfolgreiche Übertragung

- Sende-Instanz zieht Timer auf
- Timer wird nach ACK gelöscht



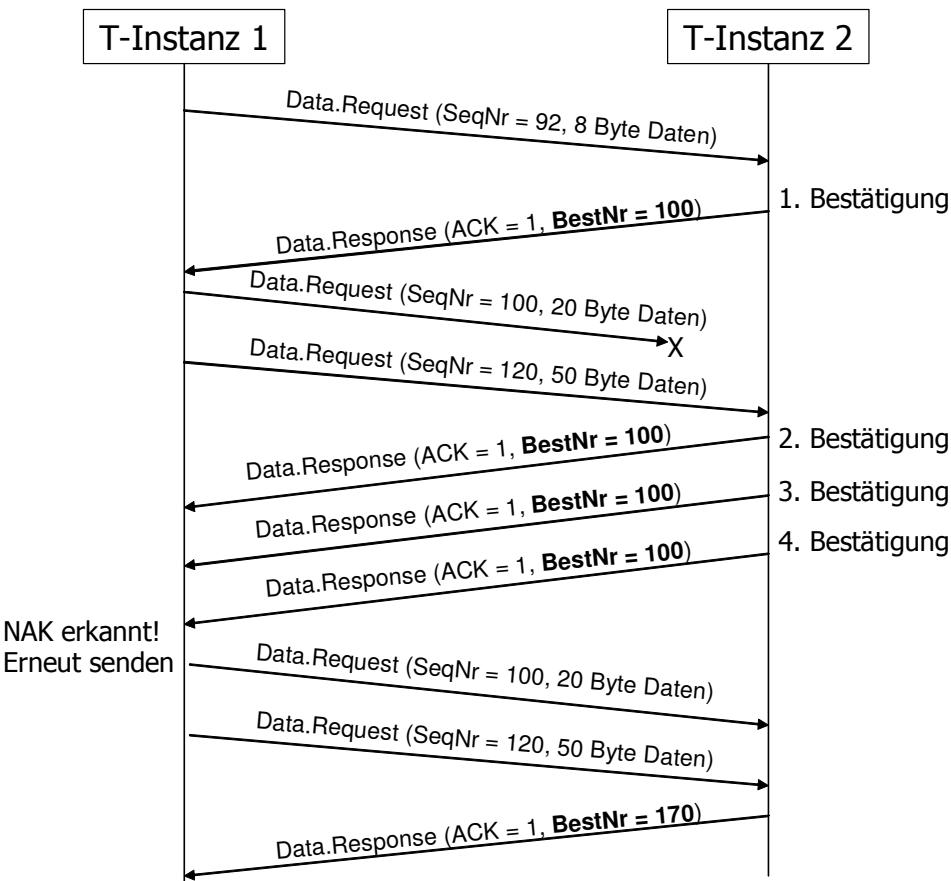
Szenario: Bestätigung geht verloren

- Sende-Instanz zieht Timer auf
- Timer läuft bei verlorengegangener Quittung ab
- TCP-Segment wird erneut gesendet



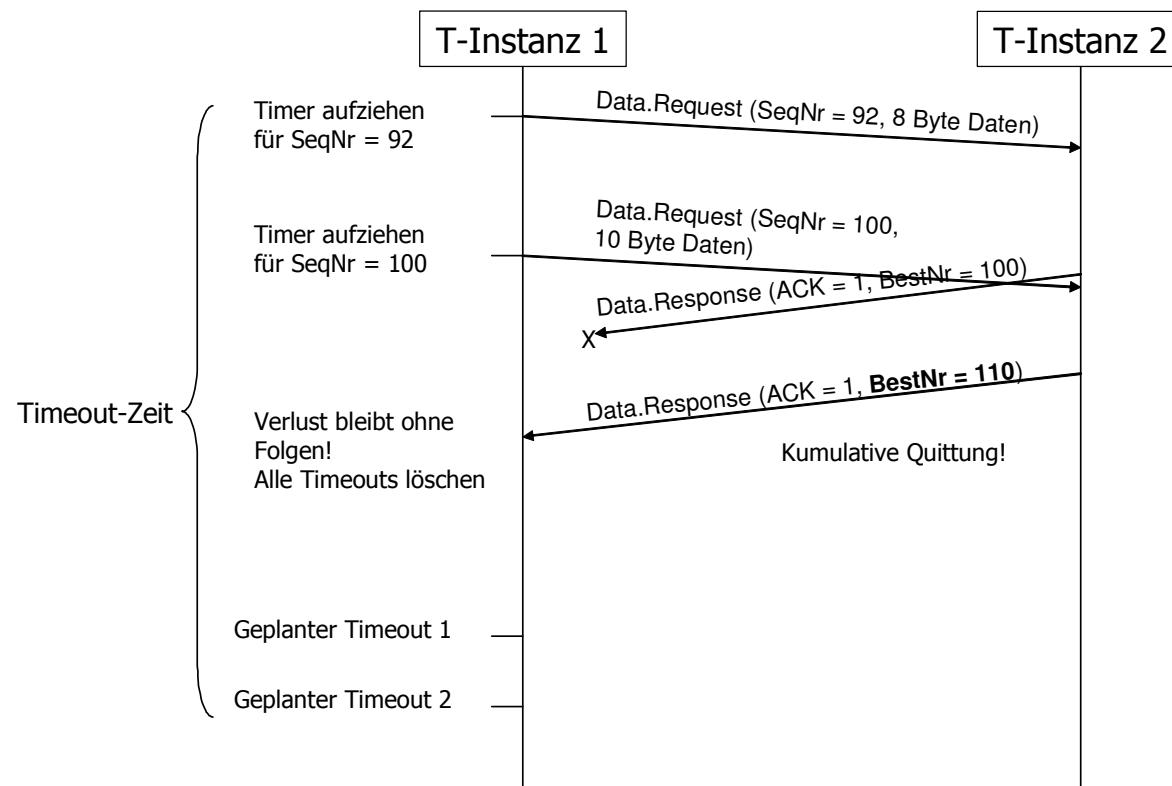
Szenario: Implizites NAK bewirkt Sendewiederholung

- Im Beispiel siehe 2. ACK wird **drei mal** vom Empfänger gesendet
- Sender erkennt Problem und sendet erneut → Problem wird vor dem Timerablauf erkannt
- Wird als **implizites NAK** bezeichnet!



Szenario: Kumulative Quittung verhindert erneutes Senden

- Verlust eines Segments bleibt ohne Folgen



Rückblick

1. Einordnung und Aufgaben des Protokolls
2. Der TCP-Header
3. Verbindungsauflauf- und abbau
4. Datenübertragung

Datenkommunikation

Transportschicht TCP (2) und Vergleich mit UDP

Wintersemester 2012/2013

Einordnung

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

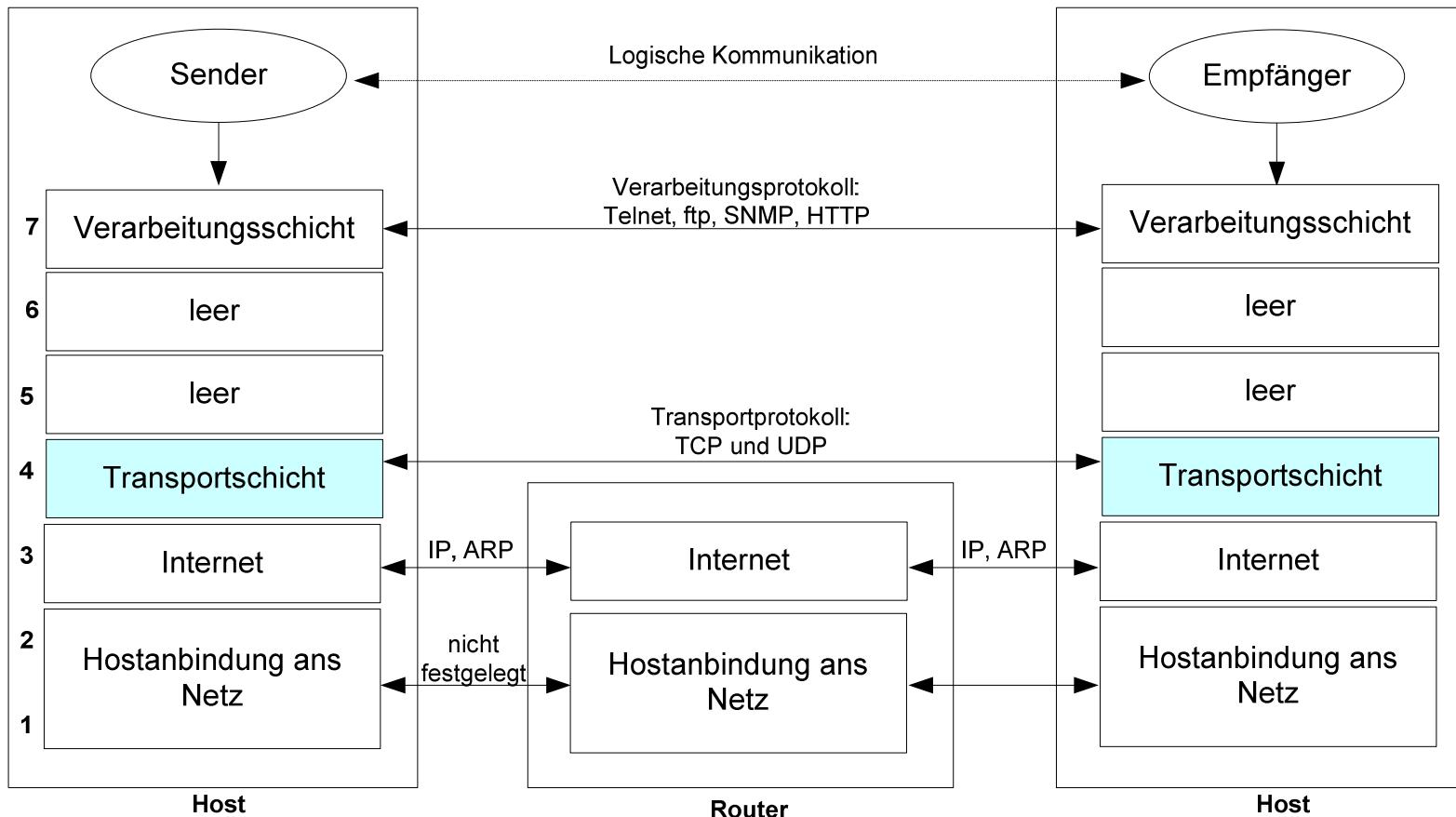
1. TCP (Transmission Control Protocol)

- **Sliding-Window-Mechanismus**
- Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
- Staukontrolle
- TCP-Timer
- TCP-Zustandsautomat

2. UDP (User Data Protocol)

- Einordnung und Aufgaben des Protokolls
- Der UDP-Header
- Datenübertragung

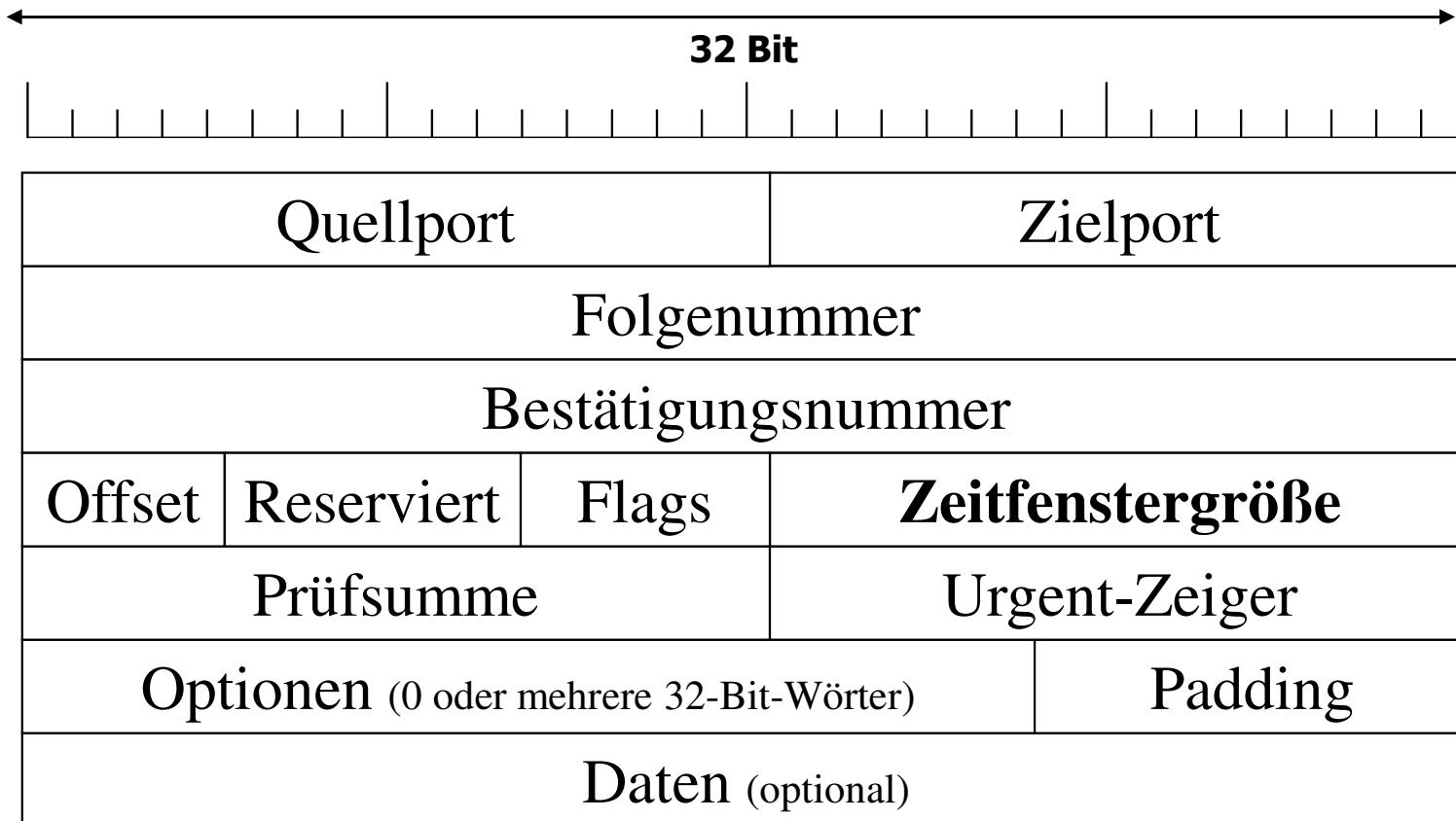
Wiederholung: TCP/IP-Referenzmodell



Sliding Window Mechanismus

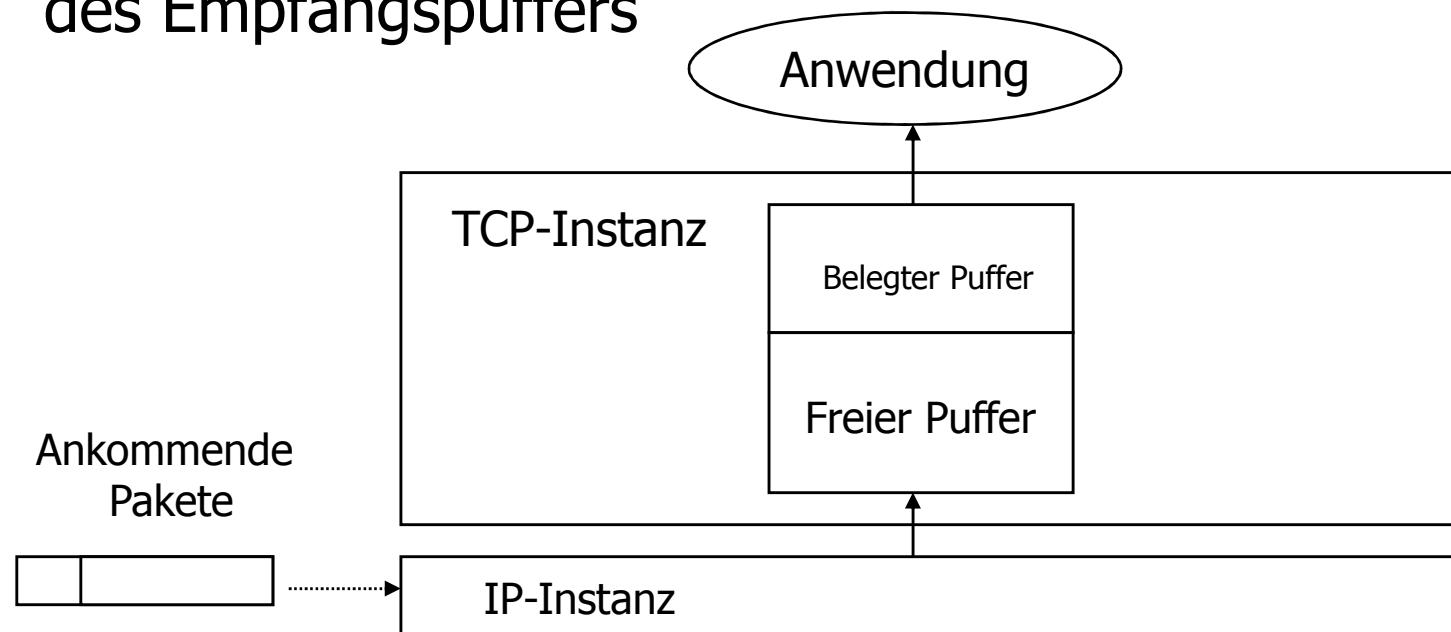
- Der Sliding Window Mechanismus erlaubt die Übertragung von mehreren TCP-Sequenzen, bevor ein **ACKnowlegde** eintrifft
- Bei TCP funktioniert Sliding Window auf Basis von Octets (Bytes)
- Die Octets (Bytes) eines Streams sind sequenziell nummeriert
- Flusskontrolle wird über das durch den Empfänger veranlasste Ausbremsen der Übertragung erreicht

TCP-Header (PCI, Protocol Control Information)

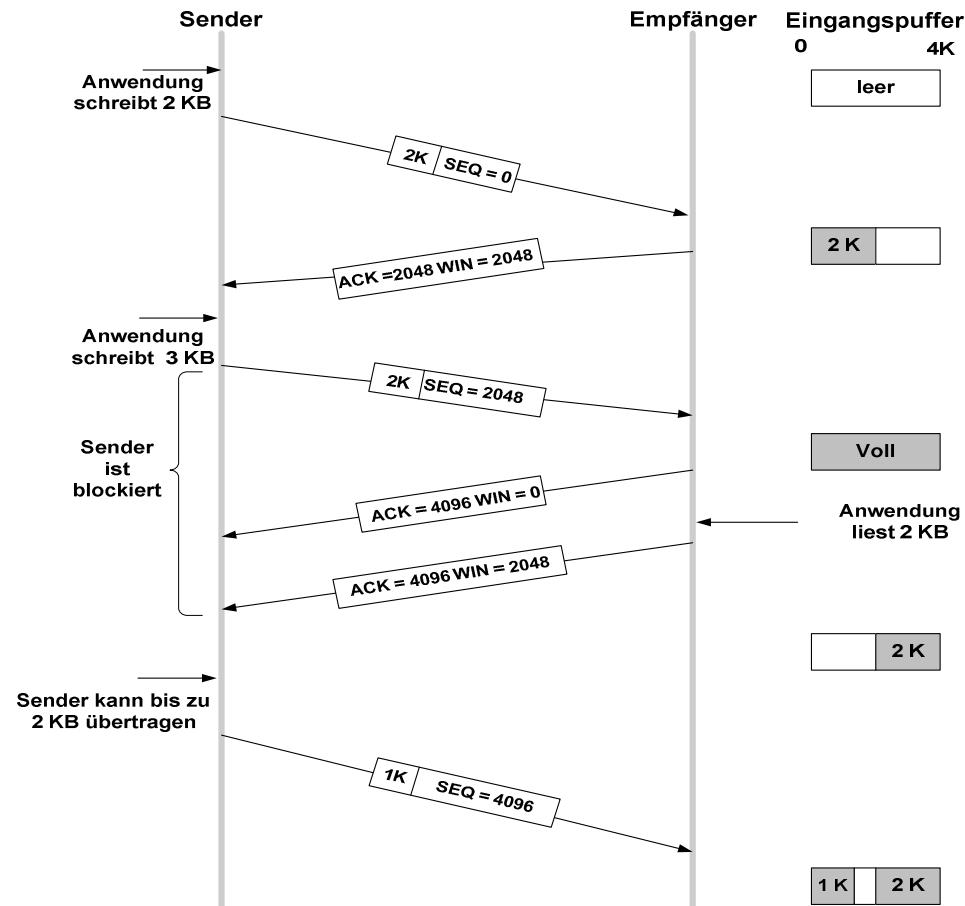


Sliding Window Mechanismus

- Die TCP-Instanzen reservieren beim Verbindungsaufbau Puffer für abgehende und ankommende Daten
- Empfänger informiert den Sender über den Füllstand des Empfangspuffers

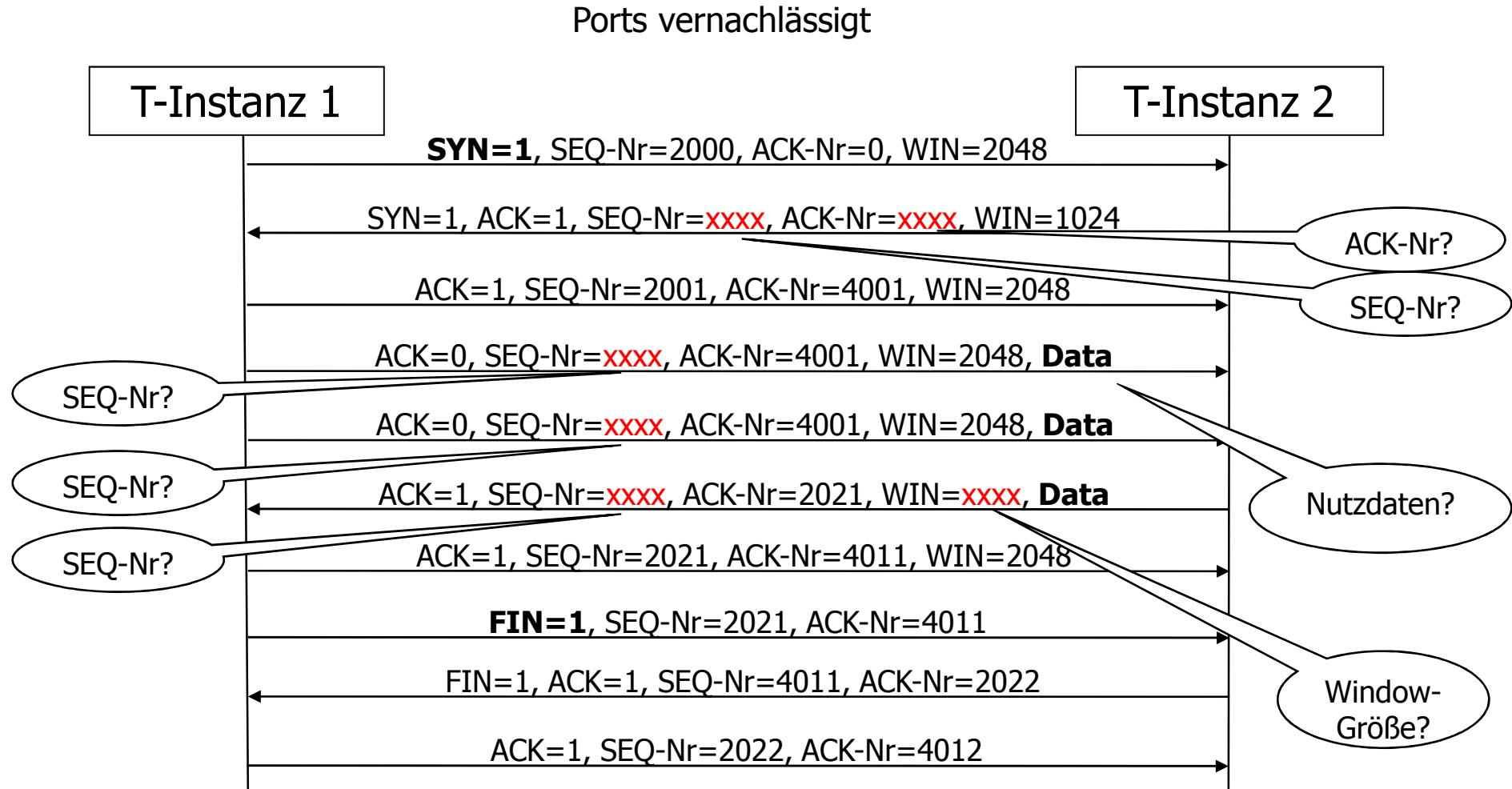


Fenstertechnik, Sliding-Window-Mechanismus

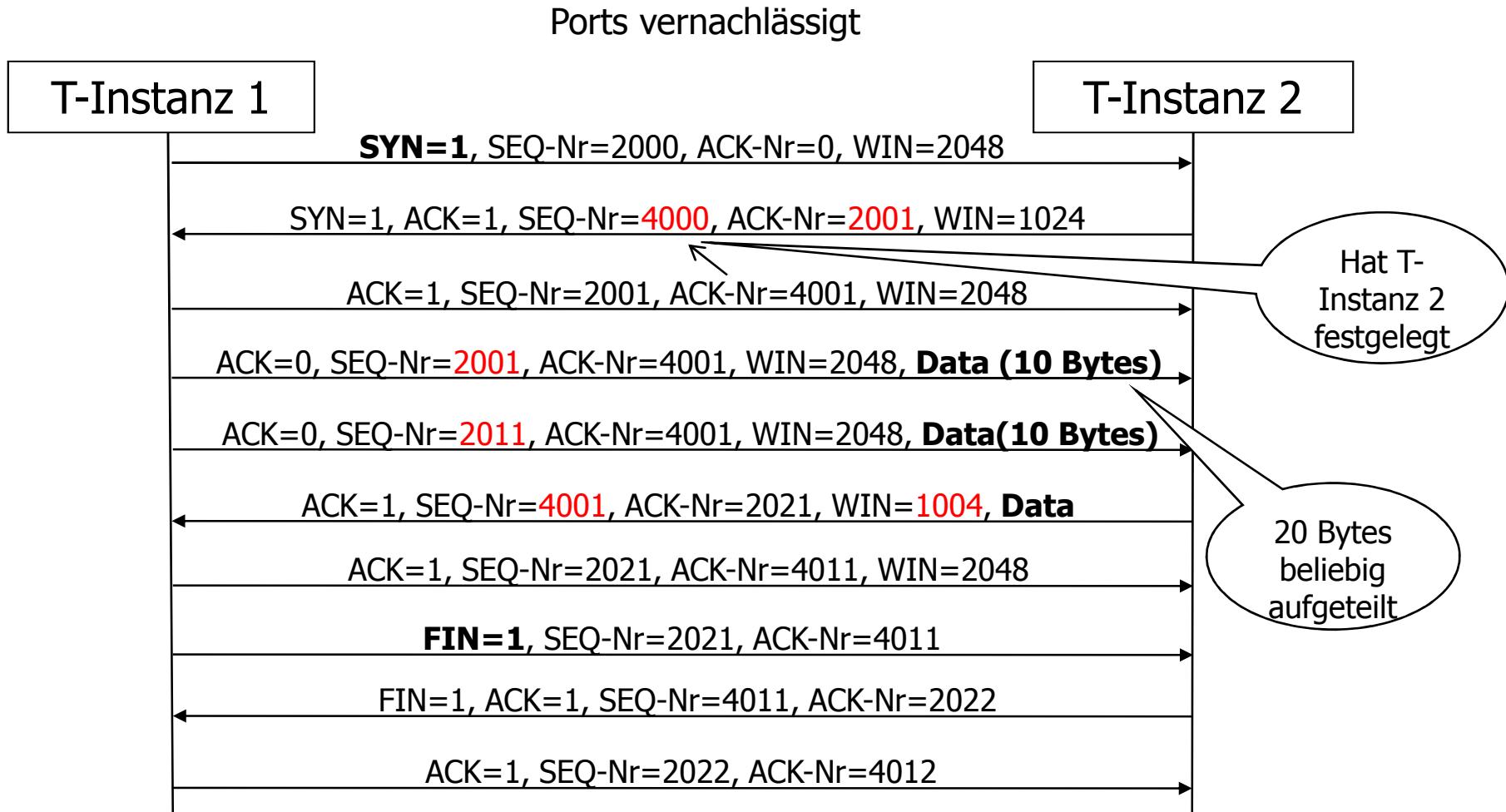


Vgl. Tanenbaum

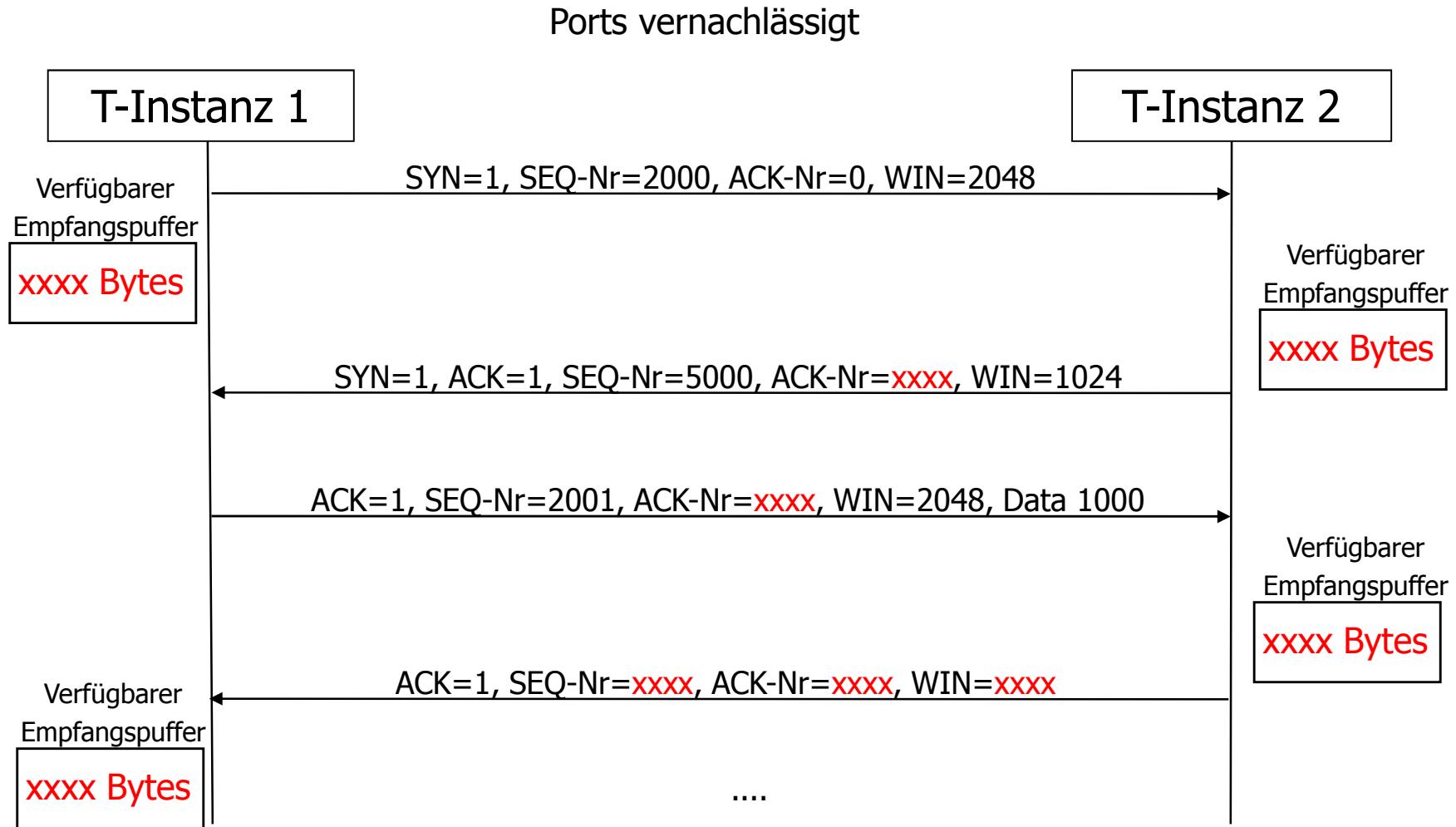
Nachrichtenfluss: Kleine Übung



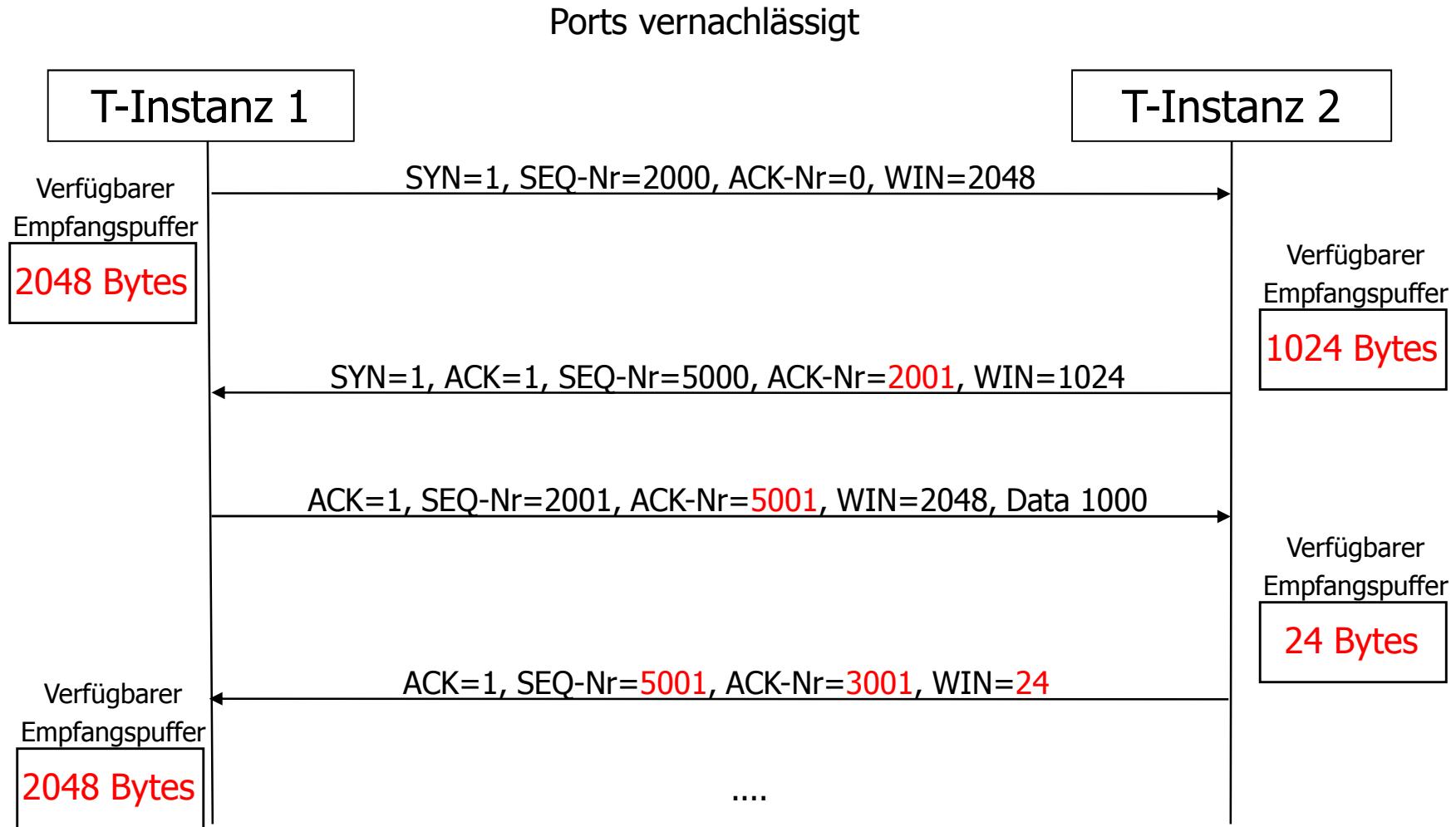
Nachrichtenfluss: Lösung



Nachrichtenfluss: Noch eine Übung



Nachrichtenfluss: Lösung



Überblick

1. TCP (Transmission Control Protocol)

- Sliding-Window-Mechanismus
- **Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom**
- Staukontrolle
- TCP-Timer
- TCP-Zustandsautomat

2. UDP (User Data Protocol)

- Einordnung und Aufgaben des Protokolls
- Der UDP-Header
- Datenübertragung

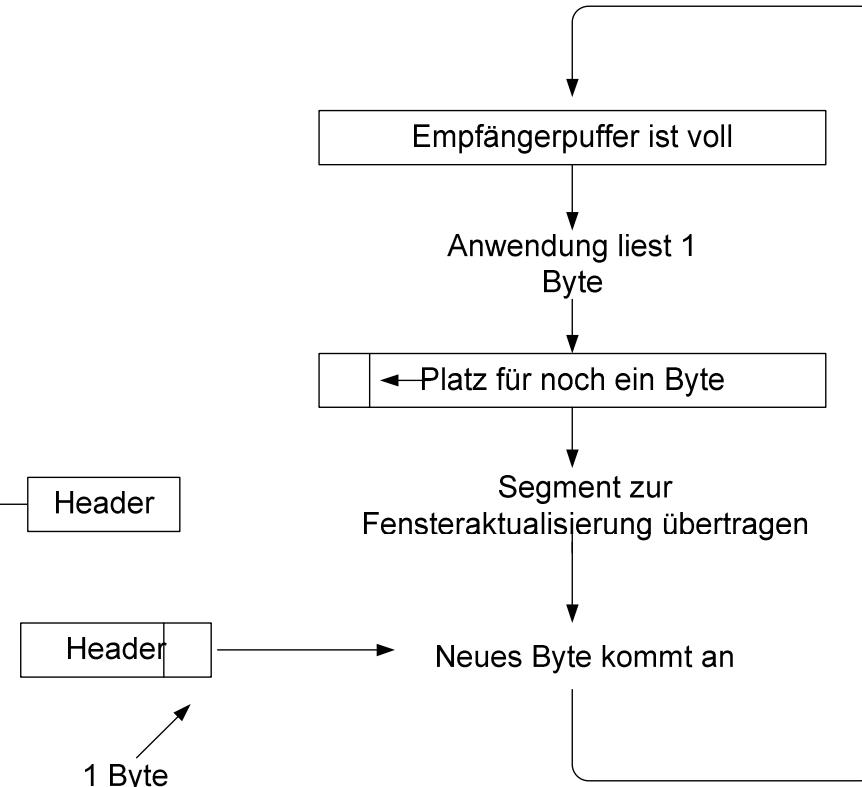
Algorithmus von Nagle

- Optimierung des Sendeverhaltens
 - Nagle versuchte aus Optimierungsgründen zu **verhindern**, dass immer **kleine Nachrichten** gesendet werden
 - Lösungsansatz: Zuerst wird nur 1 Byte gesendet, dann gesammelt und danach erst wieder ein größeres Segment gesendet
 - Kritik: Schlecht bei X-Windows oder bei telnet oder bei ssh. Warum?
 - Daher: Ausschaltbar über Socket-Option (NO_DELAY), in Java: `Socket.setTcpNoDelay(boolean)`
 - Spezifikation siehe RFC 1122

Silly-Window-Syndrom und Algorithmus von Clark

- Optimierung des Bestätigungsverhaltens
 - Problem: Silly Window Syndrom
- Clarks Lösung
 - Verhindert, dass Sende-Instanz ständig kleine Segmente sendet, da Empfängerprozess sehr langsam ausliest
 - Verzögerung der ACK-PDU oder Senden von ACK-PDU mit $WIN=0$ *)
- Nagle und Clark ergänzen sich in einer TCP-Implementierung

*) Hinweis: Verzögerung bis Empfänger eine halbe Segmentgröße gelesen hat oder der Sendepuffer halb leer ist



Vgl. Tanenbaum

Überblick

1. TCP (Transmission Control Protocol)

- Sliding-Window-Mechanismus
- Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
- **Staukontrolle**
- TCP-Timer
- TCP-Zustandsautomat

2. UDP (User Data Protocol)

- Einordnung und Aufgaben des Protokolls
- Der UDP-Header
- Datenübertragung

Staukontrolle bzw. Überlastkontrolle

Überblick

- **1986** gab es im Internet massive Stausituationen
- Seit **1989** ist Staukontrolle ein wichtiger Bestandteil von TCP
 - J. Nagle: Congestion Control in IP/TCP Internetworks, in RFC 896, 1984
- **Grundüberlegungen**
 - **IP reagiert nicht** auf Überlastsituationen
 - Verlust eines TCP-Segmentes wird von TCP als Auswirkung einer **Stausituation** im Netz interpretiert
 - Annahme: Netze sind prinzipiell stabil, **eine fehlende ACK-PDU** nach dem Senden einer Nachricht **wird als Stau** im Netz betrachtet

Staukontrolle bzw. Überlastkontrolle

Lösungsansatz (1)

- TCP **tastet** sich an die maximale Datenübertragungsrate einer Verbindung **heran**
- Neben dem Empfangsfenster wird ein neues Fenster eingeführt: Das Staukontrollfenster (bzw. Überlastfenster)
- Es baut auf dem Erkennen von Datenverlusten auf
- Die **Übertragungsrate** wird bei diesem Verfahren im Überlastfall vom Sender massiv **gedrosselt**, die Datenmengen werden kontrolliert
- Das verwendete Verfahren ist das reaktive **Slow-Start**-Verfahren (RFC 1122)
- TCP-Implementierungen **müssen** das Slow-Start-Verfahren unterstützen

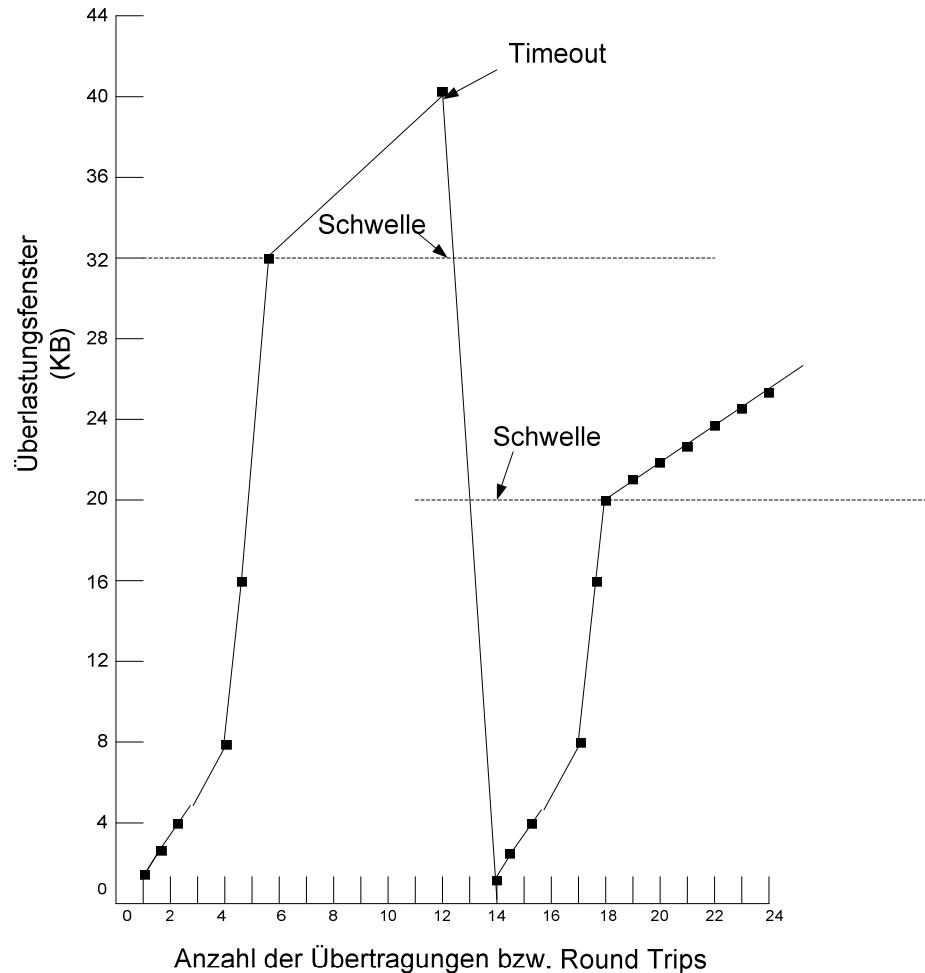
Staukontrolle bzw. Überlastkontrolle Lösungsansatz (2)

- **Quittungen** (ACK-PDUs) dienen als **Taktgeber** für den Sender
- Jede Verbindungsseite führt ein Überlast- und ein Empfangsfenster
- Es gilt:
Sendekredit für eine TCP-Verbindung =
$$\min \{\text{Überlastfenster}, \text{Empfangsfenster}\}$$
- Das Slow-Start-Verfahren kennt **zwei Phasen**:
 - Slow-Start-Phase
 - Probing-Phase

Staukontrolle bzw. Überlastkontrolle

Lösungsansatz (3)

- Initiale TCP-Segmentlänge im Beispiel 1 KB (MSS wird ausgetauscht beim Verbindungsauftbau)
- 1. Schwellwert im Beispiel 32 KB
- Timeout tritt ein bei Segmentlänge von 40 KB
- Schwellwert wird dann auf 20 KB gesetzt



Staukontrolle bzw. Überlastkontrolle

Lösungsansatz (4)

■ Slow-Start-Phase:

- Sender und Empfänger einigen sich auf eine erste sendbare TCP-Segmentlänge (MSS = z.B. 1460 Bytes)
- Sender sendet zunächst ein Segment dieser Länge
- Jeweils Verdoppelung des Überlastfensters bei erfolgreicher Übertragung aller Segmente bis zum Überlastfenster (exponentielle Steigerung)
 - Für jedes bestätigte TCP-Segment wird im nächsten Schritt das Überlastfenster verdoppelt
- Ein Schwellwert (Threshold) wird ermittelt
- Bei Erreichen des Schwellwerts geht es in die Probing-Phase über

→ Gar nicht so langsam!

Staukontrolle bzw. Überlastkontrolle

Lösungsansatz (5)

■ Die Probing-Phase:

- Bei jeder empfangenen Quittung wird die Größe des Überlastfensters „nur“ noch um eine Segmentlänge erhöht
- Berechnung des Überlastfenster bei Ankunft der ACK-PDUs für alle übertragenen Segmente innerhalb des Überlastfenster:
 - Neues Überlastungsfenster += 1 Segment
 - Weiterhin gilt:
 $\text{Sendekredit} = \min \{\text{Überlastfenster}, \text{Empfangsfenster}\}$
- Es tritt kein Problem auf
 - Überlastfenster steigt bis zum Empfangsfenster und bleibt dann konstant
 - Bei Änderung des Empfangsfensters wird Überlastfenster angepasst

Staukontrolle bzw. Überlastkontrolle

Lösungsansatz (6)

- **ACK wird nicht empfangen**
 - Man geht davon aus, dass ein weiterer Sender hinzugekommen ist
 - Mit diesem neuen Sender muss man die Pfadkapazität teilen
 - Der Schwellwert wird um die Hälfte der aktuellen Segmentanzahl reduziert
 - Die Segmentanzahl wird wieder auf das Minimum heruntergesetzt

- **Die Timerlänge ist entscheidend!**
 - Zu lang: Evtl. Leistungsverlust
 - Zu kurz: Erhöhte Last durch erneutes Senden
 - Dynamische Berechnung anhand der Umlaufzeit eines Segments (vgl. Zitterbart)

Weitere Mechanismus: Fast Recovery nach RFC 2581 (siehe auch implizites NAK)

- Ergänzendes Verfahren zur Staukontrolle: **Fast-Recovery-Algorithmus**
- Empfang von 4 Quittierungen (drei ACK-Duplikate) für eine TCP-PDU veranlasst sofortige Sendewiederholung
- Die Sendeleistung wird entsprechend angepasst
- Der Schwellwert wird auf die Hälfte des aktuellen Staukontrollfensters reduziert
- Da nur von einem Paketverlust und nicht von einem Stau im Netzwerk ausgegangen wird, wird das Staukontrollfenster auf einen Wert über dem Schwellwert eingestellt

Überblick

1. TCP (Transmission Control Protocol)

- Sliding-Window-Mechanismus
- Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
- Staukontrolle
- **TCP-Timer**
- TCP-Zustandsautomat

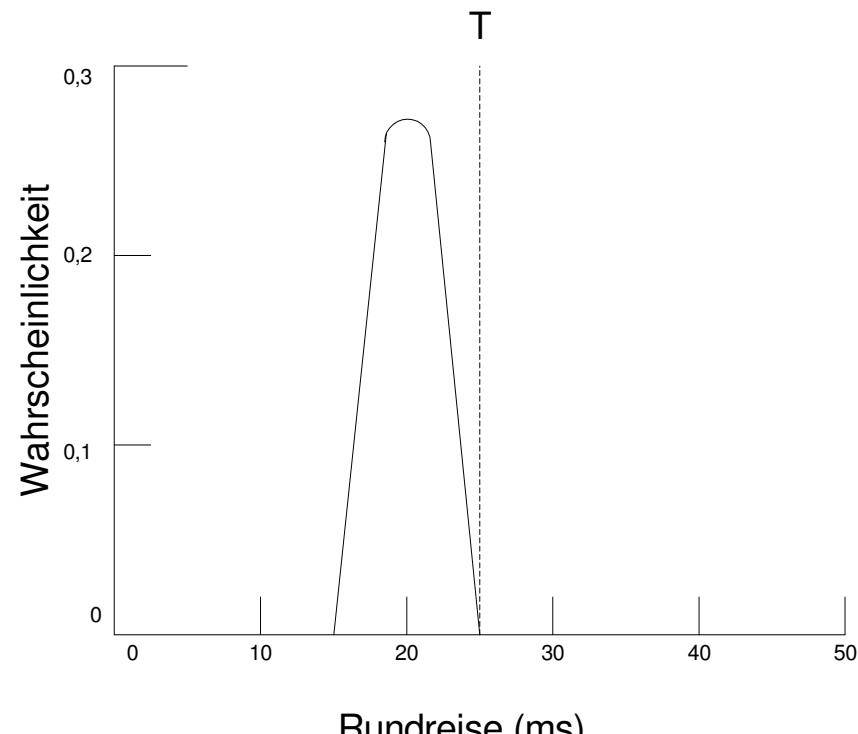
2. UDP (User Data Protocol)

- Einordnung und Aufgaben des Protokolls
- Der UDP-Header
- Datenübertragung

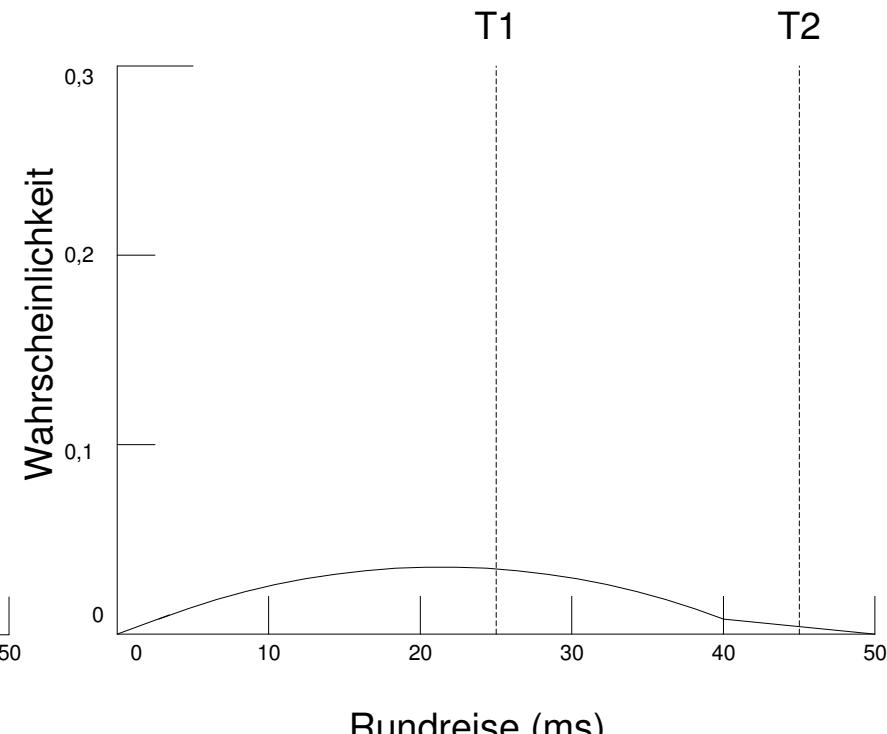
TCP-Timer

- TCP verwendet einige Timer:
 - **Retransmission Timer**
 - Zur Überwachung der TCP-Segmente nach Karn-Algorithmus
 - Wenn Timer abläuft, wird er verdoppelt und das Segment erneut gesendet (Wiederholung)
 - **Keepalive Timer**
 - Partner wird versucht zu erreichen
 - Gelingt es, bleibt Verbindung bestehen
 - **Timed Wait Timer**
 - Timer für Verbindungsabbau
 - Läuft über doppelte max. Paketlaufzeit (Default: 120 s)
 - Stellt sicher, dass alle gesendeten Pakete nach einem Disconnect-Request noch ankommen

TCP-Timer



(a)
Optimale Länge des Timers T



(b)
Schlechte Timer T_1 (zu kurz)
und T_2 (zu lang)

Überblick

1. TCP (Transmission Control Protocol)

- Sliding-Window-Mechanismus
- Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
- Staukontrolle
- TCP-Timer
- **TCP-Zustandsautomat**

2. UDP (User Data Protocol)

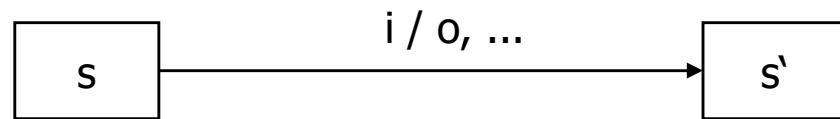
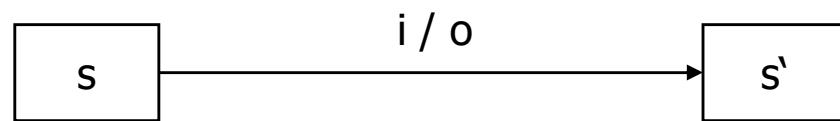
- Einordnung und Aufgaben des Protokolls
- Der UDP-Header
- Datenübertragung

Exkurs: Endliche Zustandsautomaten (1)

- Finite State Machine (FSM)
 - Deterministischer endlicher Automat mit Ausgabe (siehe Mealy-Automat)
 - Verwendet man gerne zur groben Beschreibung des Verhaltens von Protokollinstanzen
 - Ein endlicher Zustandsautomat lässt sich als Quintupel $\langle S, I, O, T, s_0 \rangle$ beschreiben:
 - S - endliche, nicht leere Menge von **Zuständen**
 - I - endliche, nicht leere Menge von **Eingaben**
 - O - endliche, nicht leere Menge von **Ausgaben**
 - $T \subseteq S \times (I \cup \{\tau\}) \times O \times S$ – eine **Zustandsüberführungsfunktion**
 - τ bezeichnet eine leere Eingabe
 - $s_0 \in S$ – **Initialzustand** des Automaten

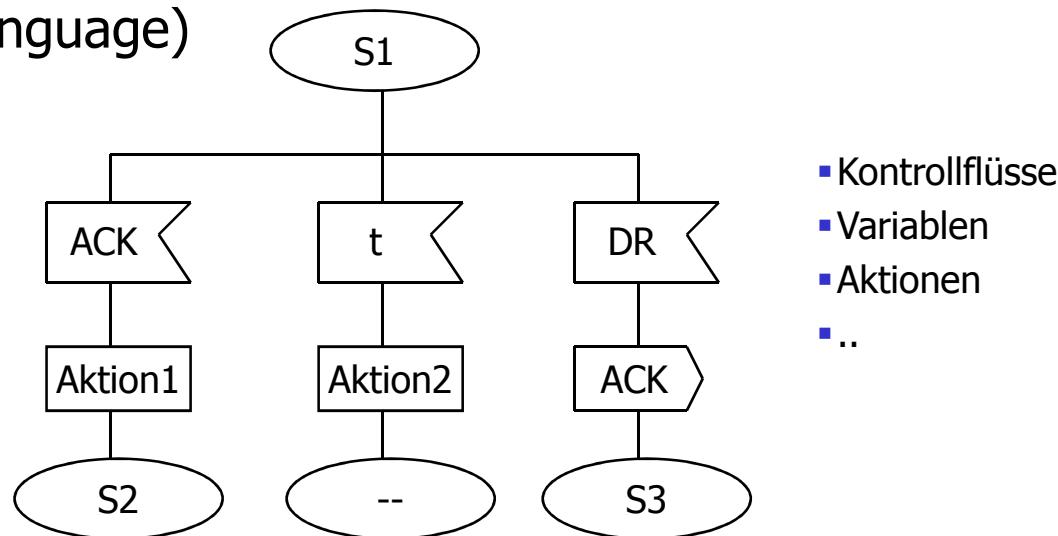
Exkurs: Endliche Zustandsautomaten (2)

- Eine Transition (Zustandsübergang) $t \in T$ ist definiert durch das Quadrupel $\langle s, i, o, s' \rangle$ wobei
 - $s \in S$ der aktuelle Zustand,
 - $i \in I$ eine Eingabe,
 - $o \in O$ eine zugehörige Ausgabe und
 - $s' \in S$ der Folgezustand ist
- Grafische Darstellung eines Zustandsübergangs

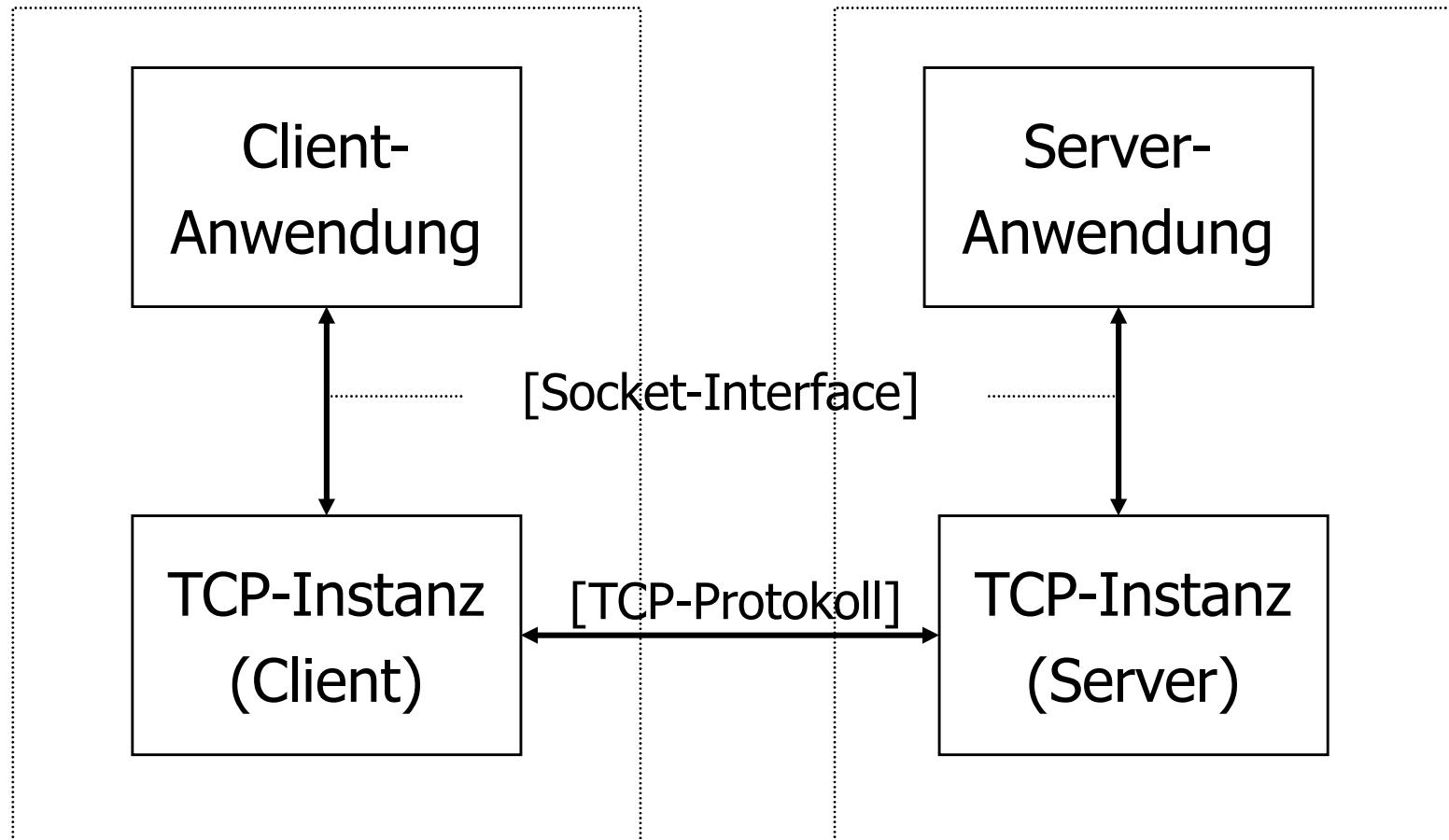


Exkurs: Endliche Zustandsautomaten (3)

- **Nachteil** von FSM: Keine weiteren Zustandsinformationen z.B. in Variable modellierbar
- Daher in der Praxis oft Nutzung **erweiterter endlicher Automaten** (EFSM) für die Detailspezifikation, um Zustandskontakte noch besser zu beschreiben → nutzt weitere Variable neben Zustandsvariable
- Modellierung z.B. in der Sprache **SDL** (Specification and Description Language)
- Beispiel:



Zustandsautomat



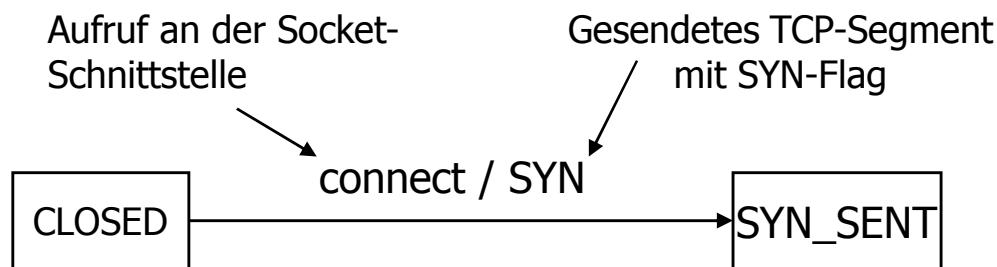
Je ein Zustandsautomat pro Transportverbindung wird
in jeder beteiligten TCP-Instanz verwaltet

Zustandsautomat

Zustand	Beschreibung
CLOSED	Keine Verbindung aktiv oder anstehend
LISTEN	Der Server wartet auf eine ankommende Verbindung
SYN_RCVD	Ankunft einer Verbindungsanfrage und Warten auf Bestätigung
SYN_SENT	Die Anwendung hat begonnen, eine Verbindung zu öffnen
ESTABLISHED	Zustand der normalen Datenübertragung
FIN_WAIT_1	Die Anwendung möchte die Übertragung beenden, Close-Aufruf wurde bereits abgesetzt
FIN_WAIT_2	Die andere Seite ist einverstanden, die Verbindung abzubauen, Bestätigung (ACK) gesendet
TIME_WAIT	Warten, bis keine Segmente mehr kommen
CLOSING	Beide Seiten haben versucht, gleichzeitig zu beenden
CLOSE_WAIT	Die Gegenseite hat den Abbau eingeleitet, warten auf Close-Aufruf der lokalen Anwendung
LAST_ACK	Warten, bis letzte Bestätigung (ACK) für Verbindungsabbau angekommen ist

TCP als FSM

- Ein TCP-Zustandsautomat lässt sich als Quintupel $\langle S, I, O, T, s_0 \rangle$ beschreiben:
 - $S = \{\text{CLOSED}, \text{LISTEN}, \text{SYN_RCVD}, \dots\}$
 - $I = \{\text{connect}, \text{send}, \text{close}, \text{SYN}, \text{ACK}, \text{FIN}, \dots\}$
 - $O = \{\text{SYN}, \text{ACK}, \text{FIN}, \dots\}$
 - $s_0 = \text{CLOSED} \in S$
- Hinweis: Wir beschreiben im Weiteren nur die Transitionen des Verbindungsauflaufs und abbaus
- Beispiel einer Transition:

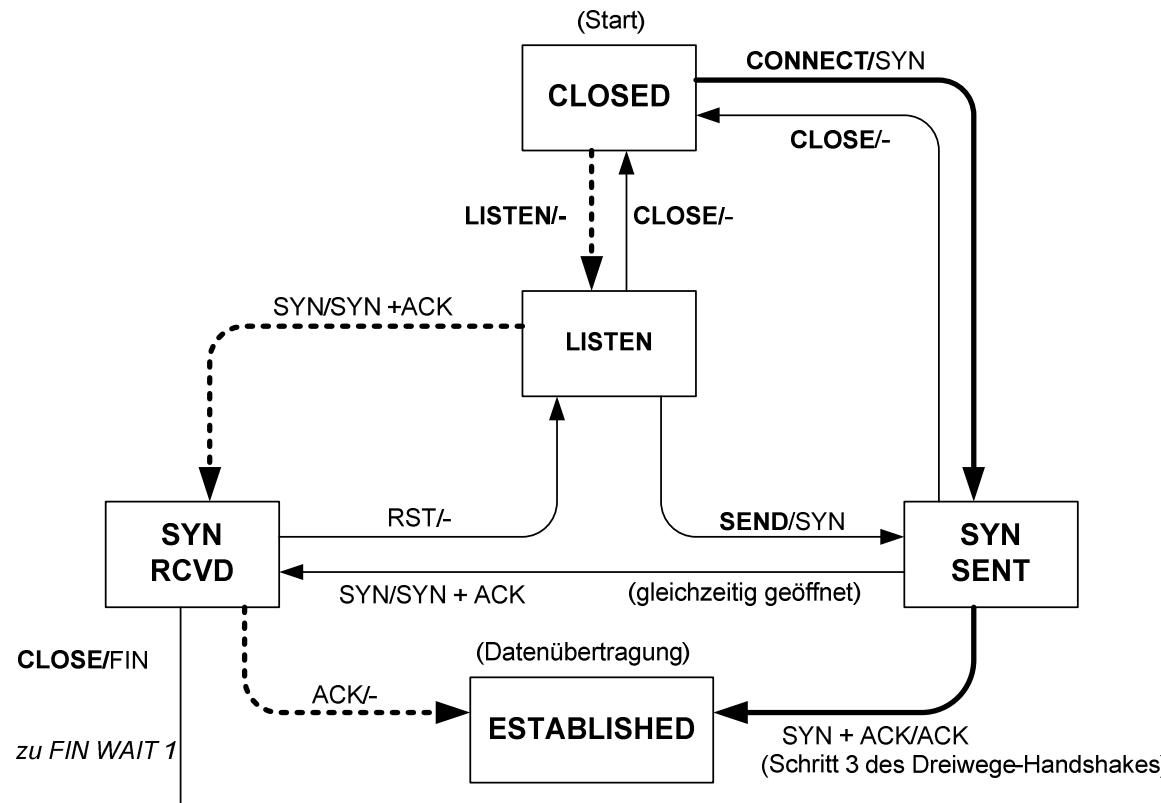


Finite-State-Machine-Modell (1)

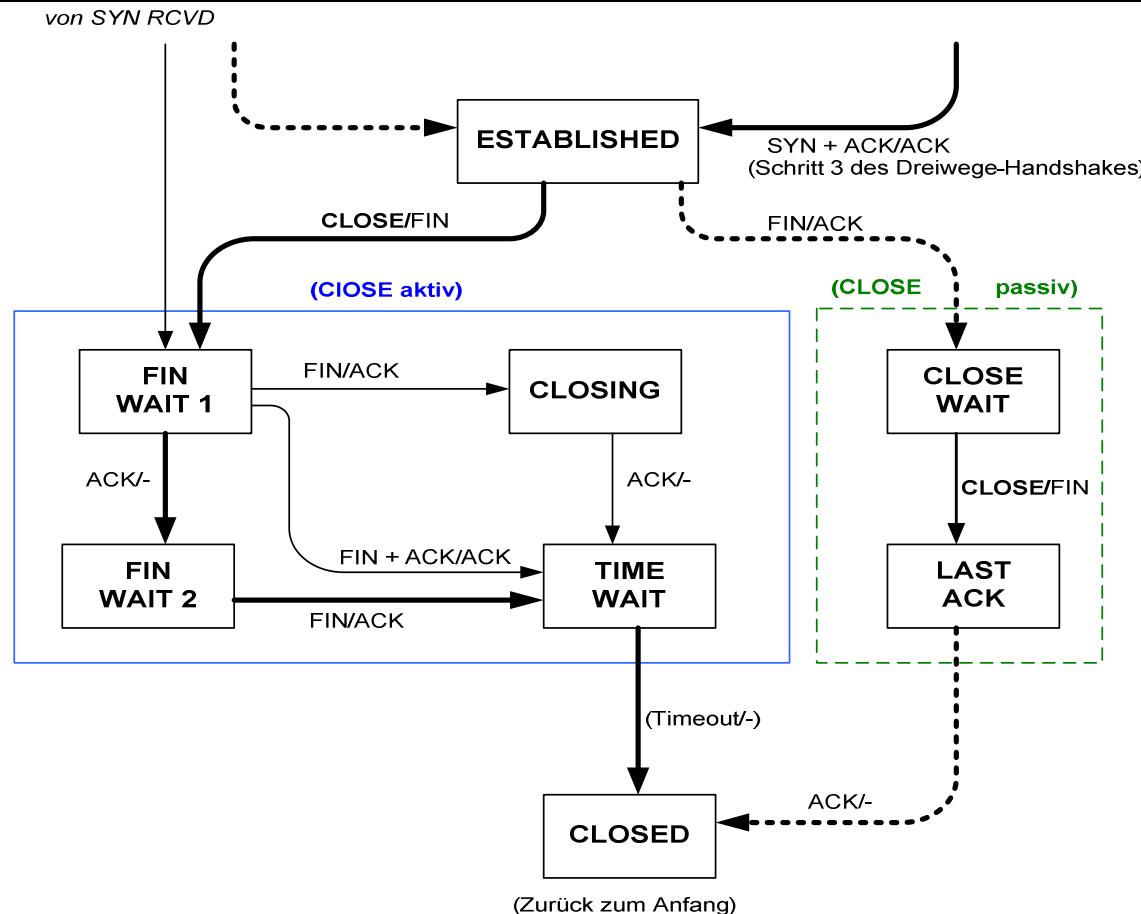
fette Linie: normaler Pfad des Clients

fette gestrichelte Linie: normaler Pfad des Servers

feine Linie: ungewöhnliche Ereignisse

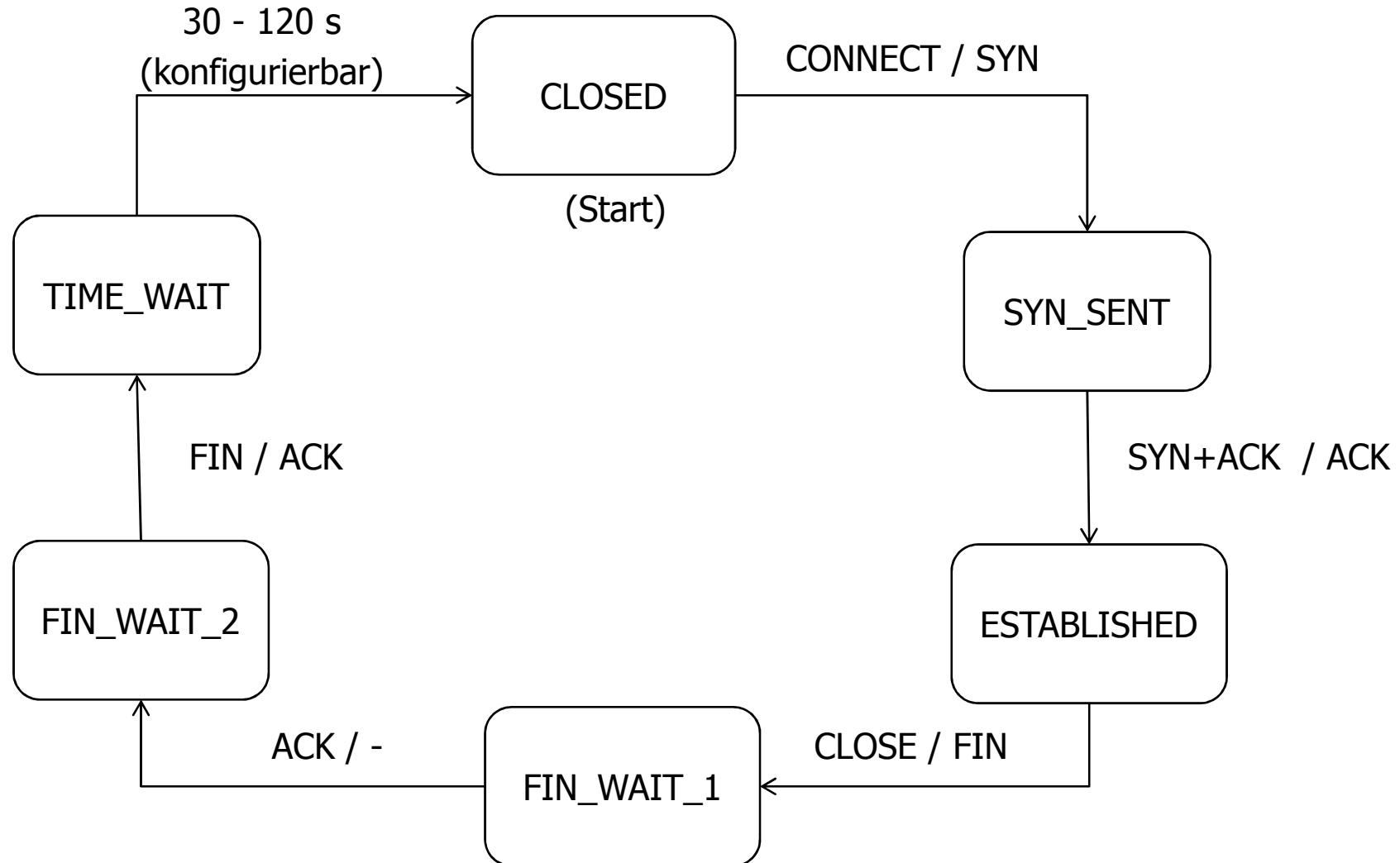


Finite-State-Machine-Modell (2)

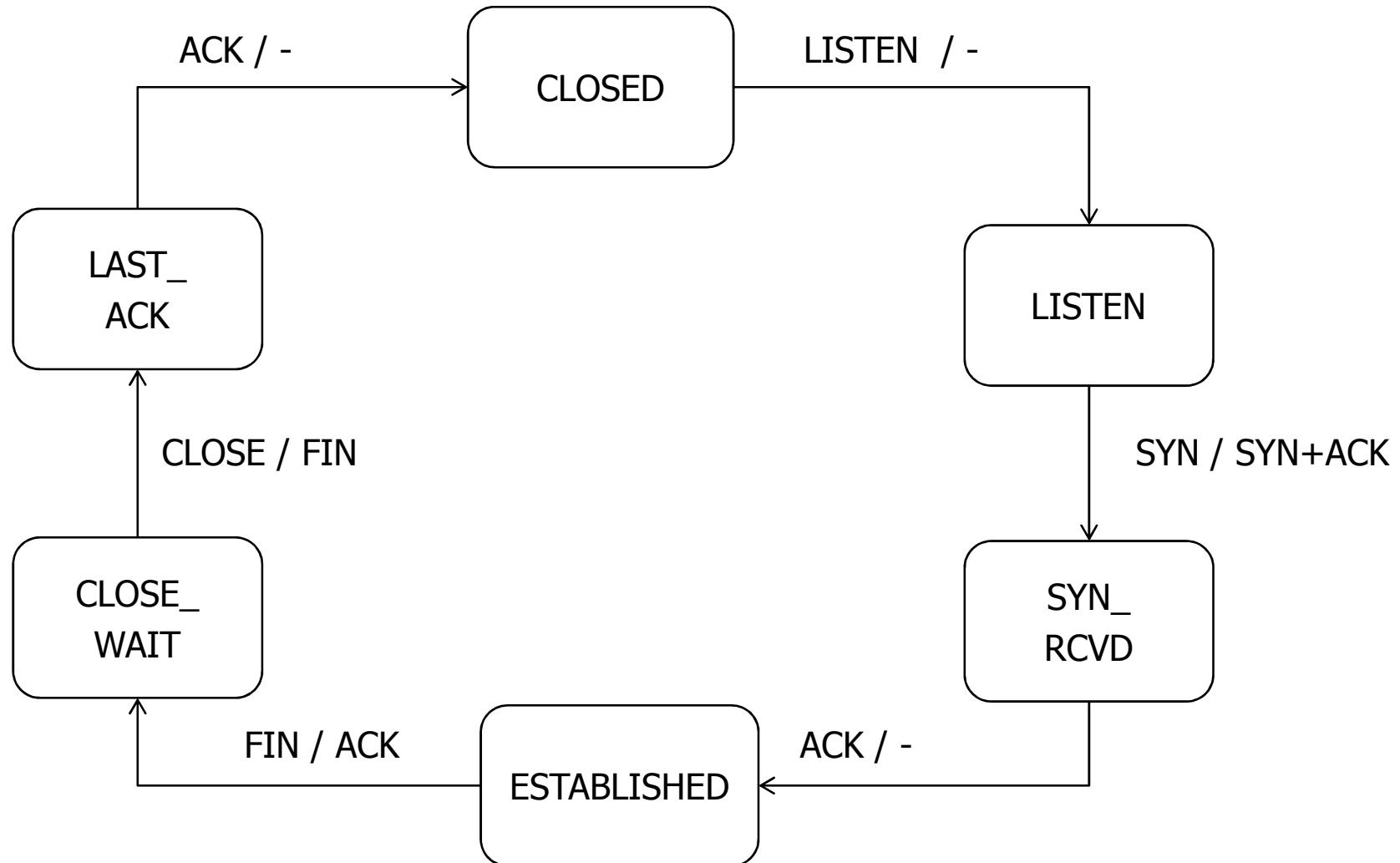


- Es gibt vier Möglichkeiten, die Verbindung abzubauen. Welche?
- 1) Close Aktiv 2) Close passiv 3) Beide gleichzeitig (**FIN WAIT1** → **CLOSING** → **TIME WAIT** → **CLOSED**) 4) Selten: Close auf passiver Seite bei **FIN** schon da

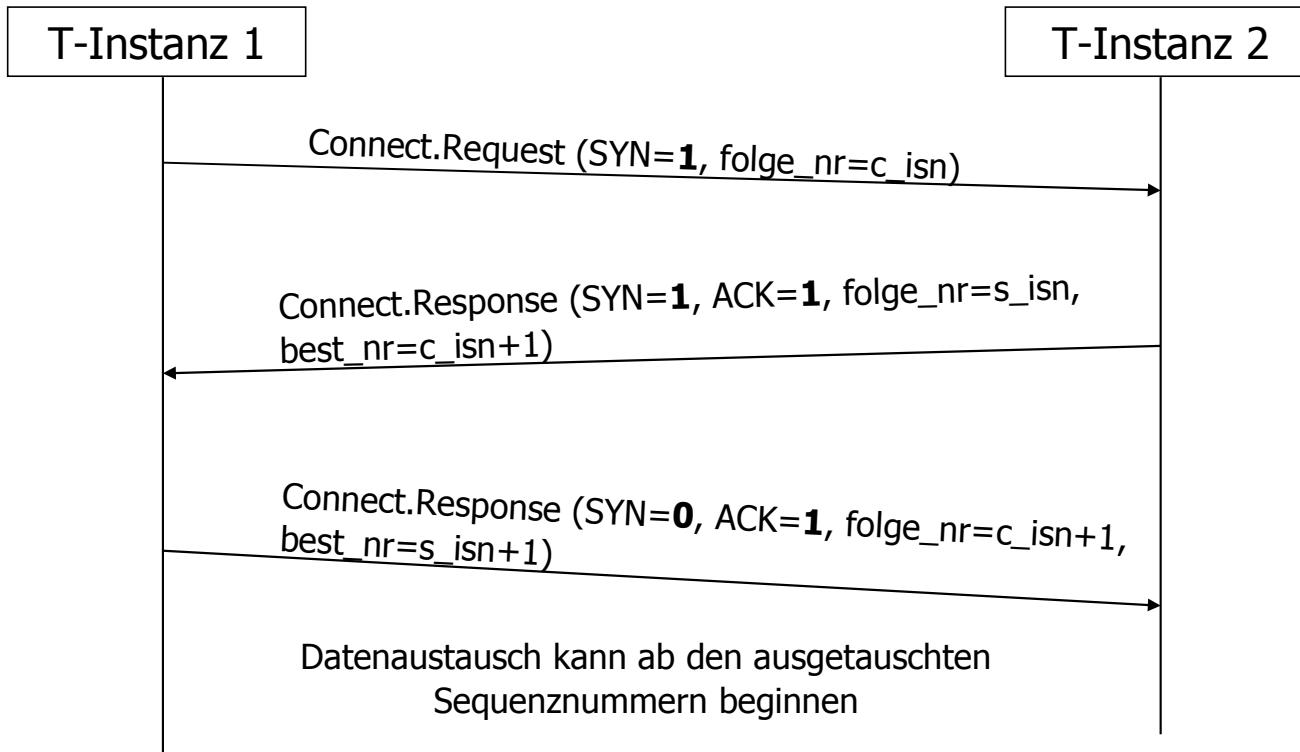
Finite-State-Machine-Modell, TCP-Client (vereinfacht)



Finite-State-Machine-Modell, TCP-Server (vereinfacht)



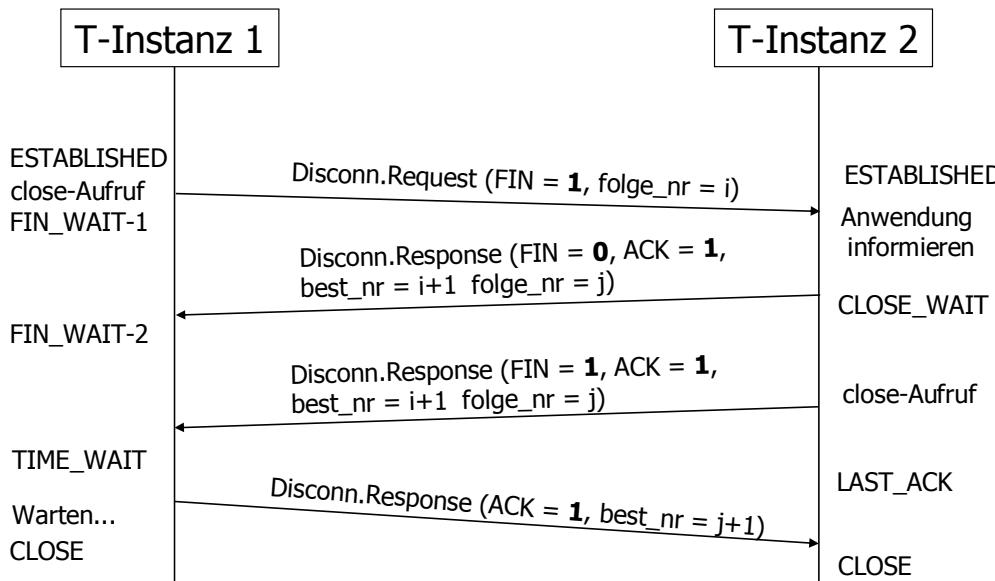
Verbindungsauftbau



c_isn = Initial Sequence Number des Clients (Instanz 1)
s_isn = Initial Sequence Number des Servers (Instanz 2)

Verbindungsabbau

- Client baut die Verbindung ab (auch Server kann es)
- Alle Segmente mit Folgenummer < i bzw. j sind noch zu verarbeiten



Zustände im TCP-Zustandsautomat:
ESTABLISHED, FIN_WAIT-1, FIN_WAIT-2, TIMED_WAIT, CLOSE, CLOSE_WAIT, LAST ACK

Übung

- Wozu dienen folgende Stati des TCP-Zustandsautomaten:
 - SYN_RECV
 - SYN_SENT
 - TIME_WAIT
 - CLOSE_WAIT
- Betrachten Sie mit dem Kommando **netstat** die Zustände diverser TCP-Verbindungen, die auf Ihrem Rechner laufen
 - Web-Browser
 - Chat-Anwendung
 - ...

Einschub für Interessierte: Zustandsautomaten implementieren

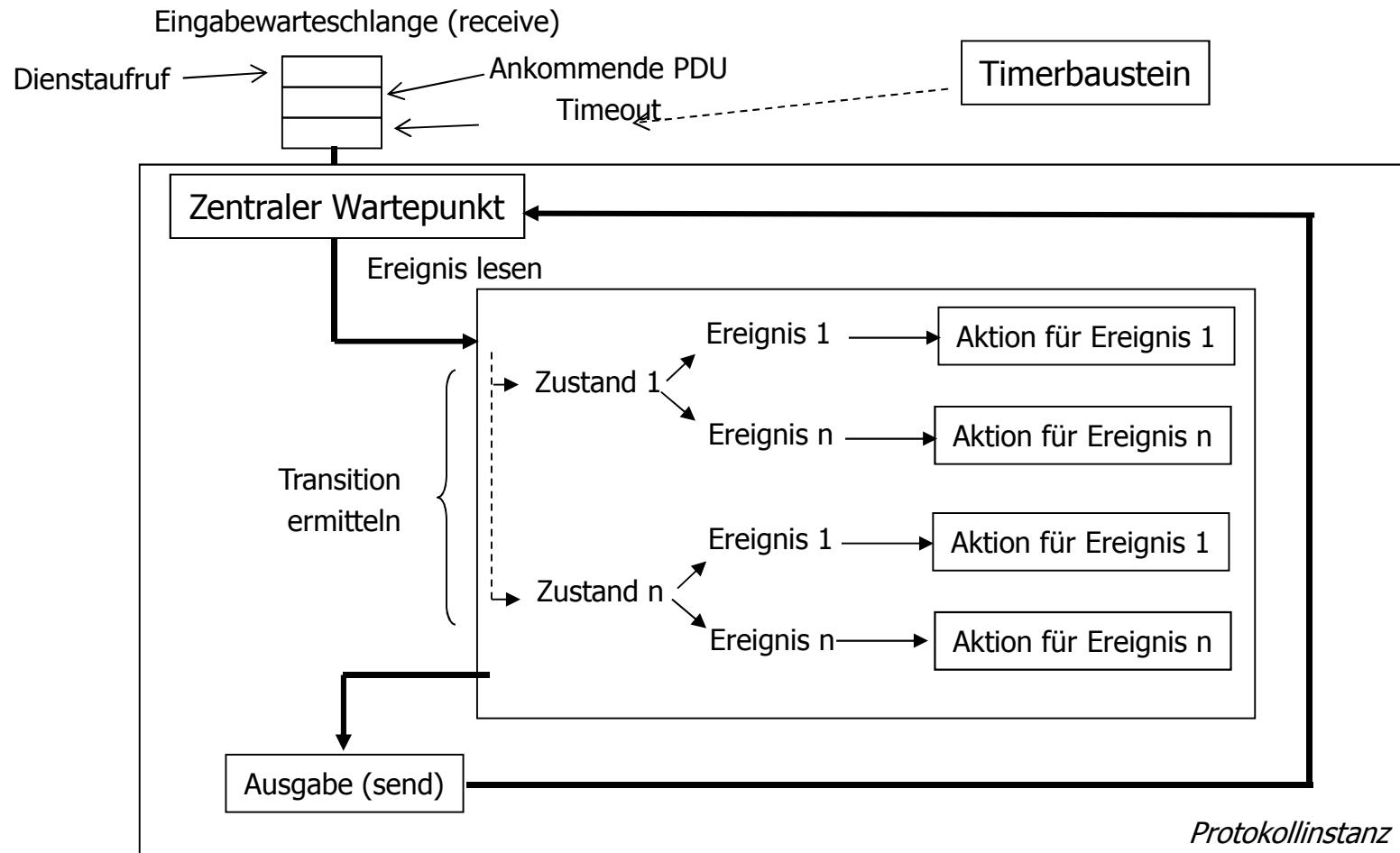
- Wie implementiert man einen Zustandsautomaten?
 - Welches Prozess-/Threadmodell verwendet man?
 - Ein bekanntes Modell: Server-Modell
 - Auftretende Ereignisse (Nachrichten, Timeouts, Dienstaufrufe) müssen verarbeitet werden und führen zu Zustandsänderungen und Aktionen
- Mehrere Varianten
 - Bei jedem Ereignis über Switch-Case-Konstruktionen prüfen, welche Aktion ausgeführt werden soll und ob das Ereignis zulässig ist
 - Nutzung des State Patterns zur Vermeidung von ewigen Switch-Case-Konstruktionen
- State Pattern (Objektorientierter Ansatz)
 - Jeder Zustand wird als eigene Objektklasse implementiert, das von einer Basisobjektklasse erbt

Einschub für Interessierte: Zustandsautomaten implementieren - Server-Modell (1)

■ Das Server-Modell

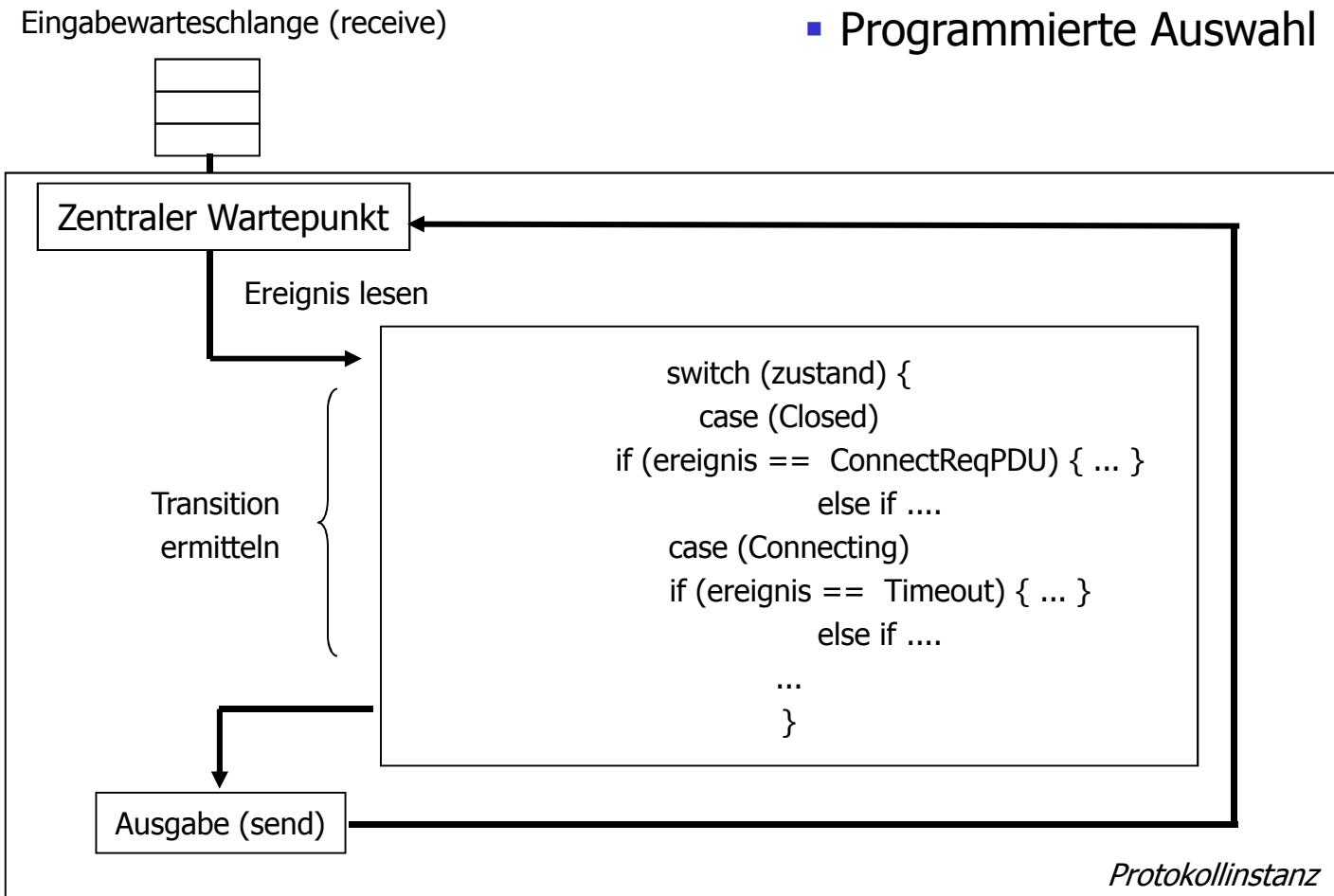
- Protokollinstanz wird als zyklisch arbeitender sequentieller Prozess/Thread umgesetzt
- Eingabewarteschlange für auftretende Ereignisse wird an einem Ereigniswartepunkt abgearbeitet
- Ereignisse werden zyklisch in einer „Endlosschleife“ abgearbeitet:
- Grober Algorithmus:
 - • Ereignis aus der Warteschlange lesen
 - Analyse des eingehenden Ereignisses (Nachricht)
 - Aktion auf Basis des aktuellen Zustandes auswählen
 - Ggf. Fehleranalyse und Fehlerreaktion
 - Aktion ausführen, Transition durchführen (in Zustand gehen) und Ausgabe erzeugen
 - Und dann wieder von vorne: Nächstes Ereignis aus der Warteschlange lesen

Einschub für Interessierte: Zustandsautomaten implementieren - Server-Modell (2)



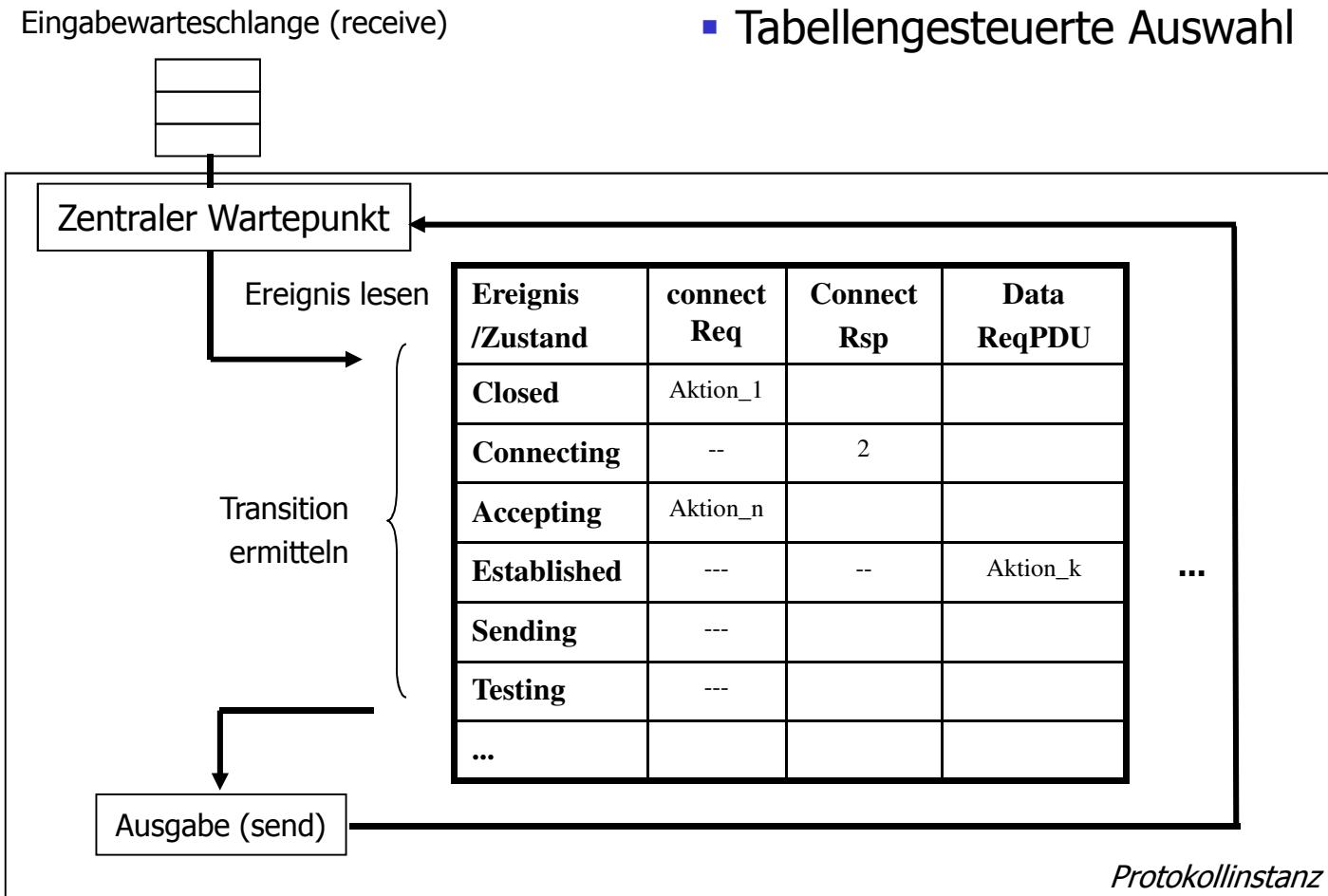
König, H.: Protocol Engineering, Teubner, 2003

Einschub für Interessierte: Zustandsautomaten implementieren - Server-Modell (3)



König, H.: Protocol Engineering, Teubner, 2003

Einschub für Interessierte: Zustandsautomaten implementieren - Server-Modell (4)



König, H.: Protocol Engineering, Teubner, 2003

Einschub für Interessierte: Zustandsautomaten implementieren - Server-Modell (5)

■ Kritik am Server-Modell

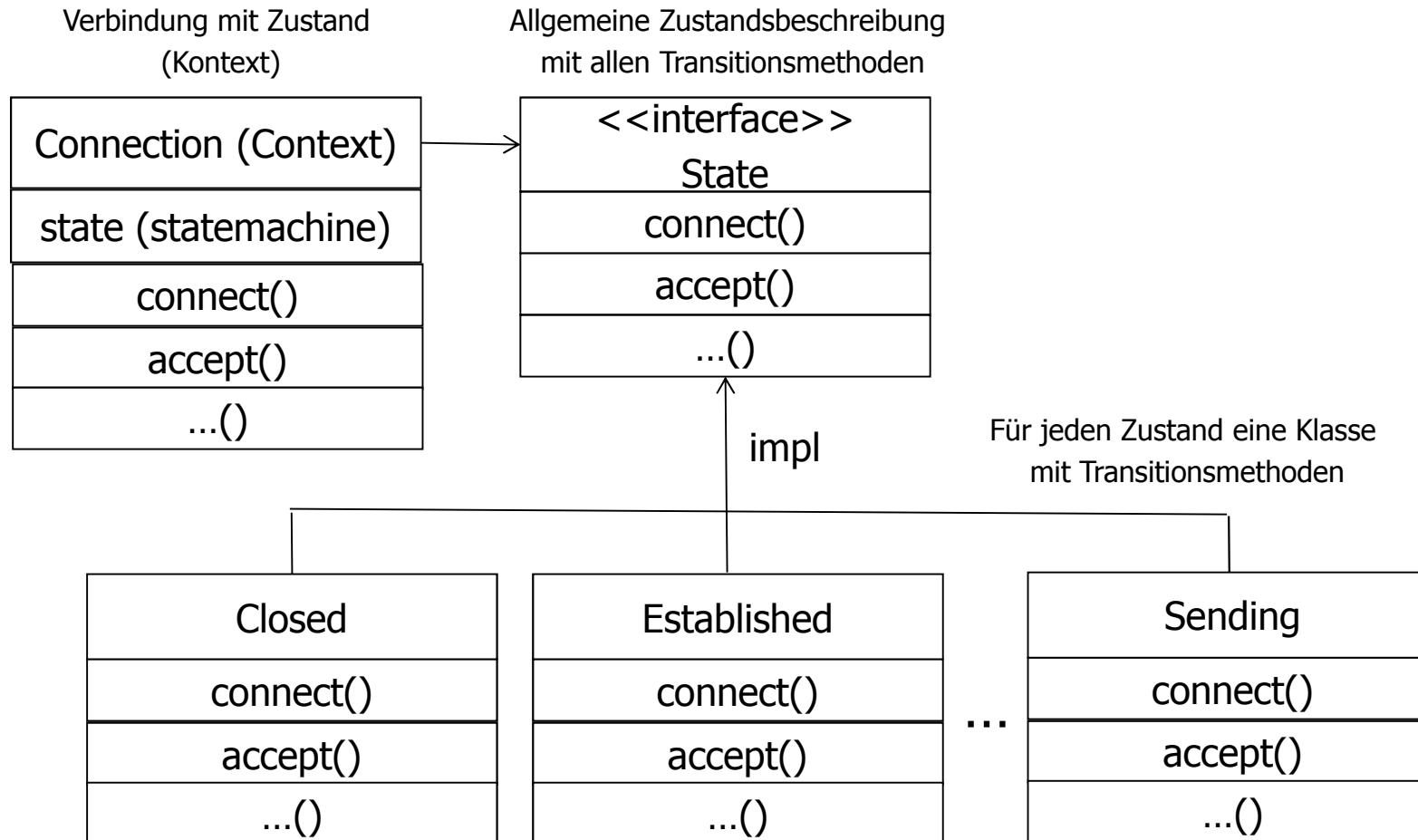
- Positiv

- Einfacher Entwurf, da Zustandsautomat direkt abgebildet wird
- Einfach zu implementieren

- Negativ

- Bei vielen Zuständen unübersichtlich, da viele Switch-Case und If-Konstruktionen zur Ermittlung der nächsten Transition notwendig sind
- Evtl. nicht so leistungsfähig, da alles in einem Prozess/Thread erledigt wird
 - → Verbesserung durch Ausführen einer Aktion in einem eigenen Thread

Einschub für Interessierte: Zustandsautomaten implementieren - State Pattern



Überblick

1. TCP (Transmission Control Protocol)

- Sliding-Window-Mechanismus
- Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
- Staukontrolle
- TCP-Timer
- TCP-Zustandsautomat

2. UDP (User Data Protocol)

- Einordnung und Aufgaben des Protokolls
- Der UDP-Header
- Datenübertragung

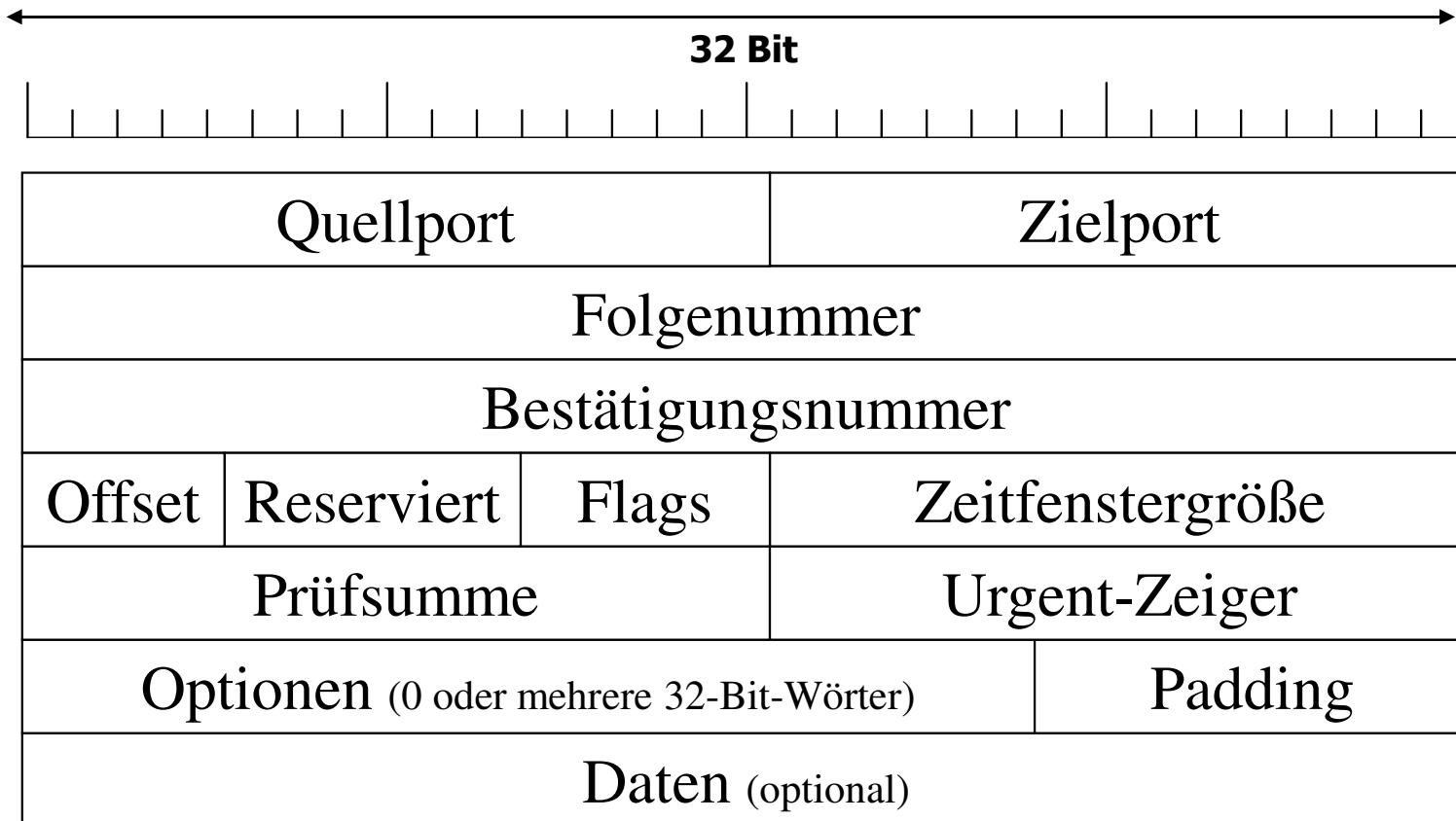
Einordnung und Aufgaben

- **Unzuverlässiges**, verbindungsloses Transportprotokoll
 - **Keine Empfangsbestätigung** für Pakete
 - UDP-Nachrichten **können** ohne Kontrolle **verloren gehen**
 - Eingehende Pakete werden **nicht in einer Reihenfolge sortiert**
 - Maßnahmen zur Erhöhung der Zuverlässigkeit müssen im Anwendungsprotokoll ergriffen werden, z.B.
 - ACK und Warten mit Timeout
 - Wiederholtes Senden bei fehlendem ACK
- **Vorteile** von UDP gegenüber TCP
 - Bessere Leistung möglich, aber nur, wenn TCP nicht nachgebaut werden muss
 - Multicast- und Broadcast wird unterstützt

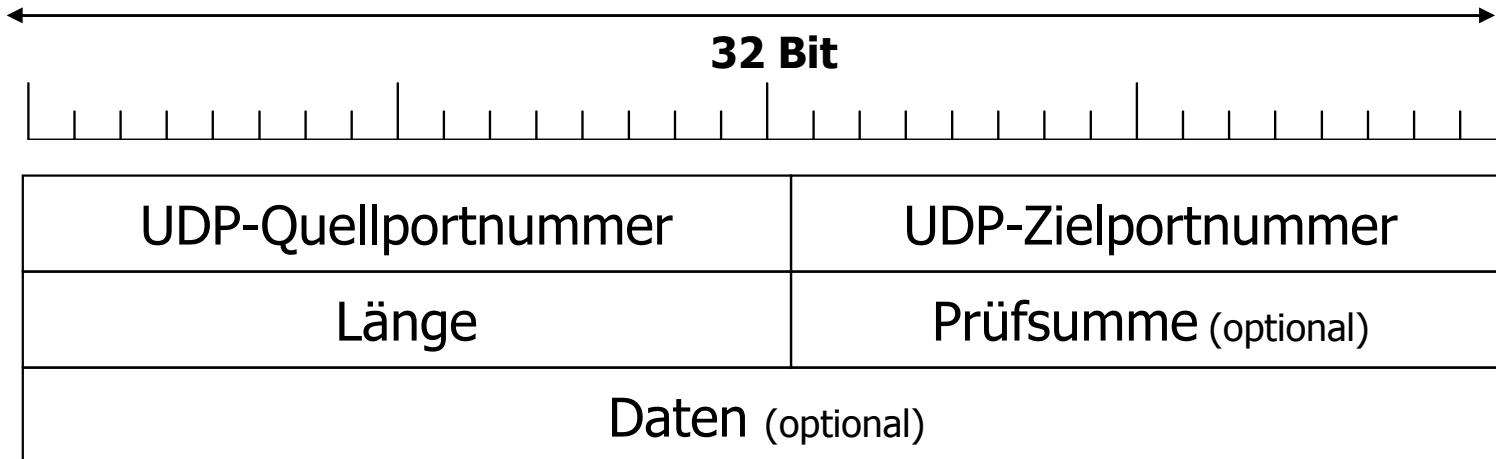
Einordnung und Aufgaben

- Bei UDP ist **keine explizite Verbindungsaufbau-Phase** erforderlich und entsprechend auch **kein Verbindungsabbau**
- Userprozess erzeugt ein UDP-Socket und kann Nachrichten senden und empfangen
- Nachrichten werden bei UDP als **Datagramme** bezeichnet
- In den Datagrammen wird die T-SAP-Adresse des Senders und des Empfängers gesendet (UDP-Ports)

TCP-Header (PCI, Protocol Control Information)



UDP-Header (PCI)

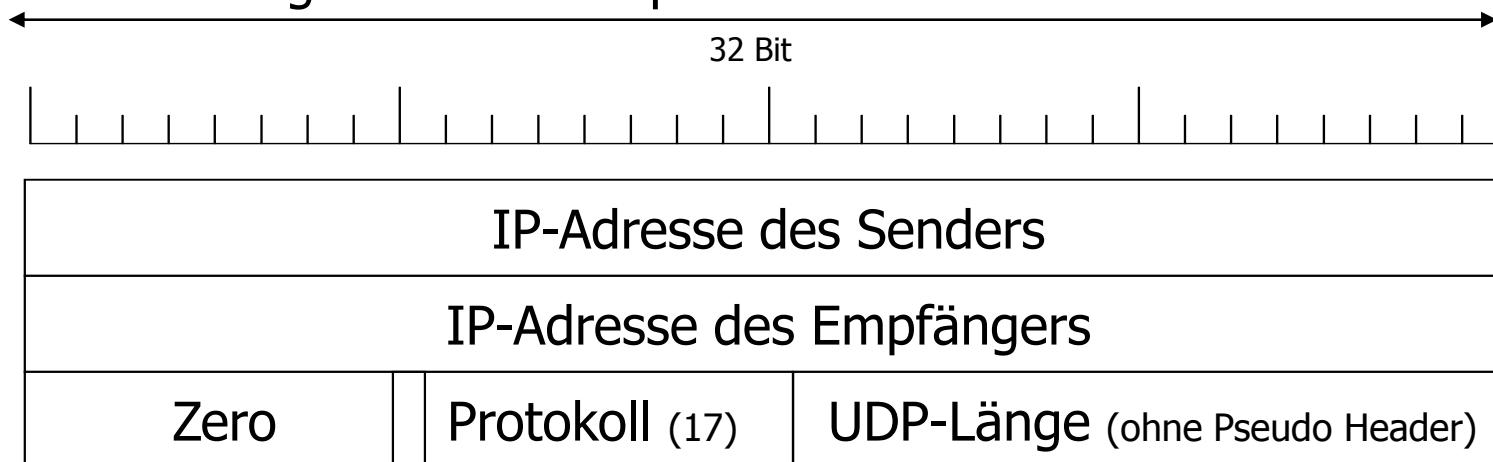


UDP-Header

- **UDP-Ziel-Portnummer:** Nummer des empfangenden Ports
- **UDP-Quell-Portnummer:** Nummer des sendenden Ports
- **Länge:** Größe des UDP-Segments inkl. Header in Byte
- **Prüfsumme (optional):** Prüft das Gesamtsegment (Daten + Header) einschließlich eines Pseudoheaders
- **Daten:** Nettodaten der Nachricht

Pseudoheader (1)

- Die Prüfsumme ist die einzige Möglichkeit, die intakte Übertragung beim Empfänger zu verifizieren
- Vor dem Berechnen der Prüfsumme wird ein Pseudoheader ergänzt
- Das Segment wird auf eine durch 16 Bit teilbare Größe aufgefüllt (gerade Anzahl an Bytes)
- Berechnung auch wie bei TCP über Addition von 16-Bit-Worten und Bildung des Einerkomplements der Summe



Pseudoheader (2)

- Der Empfänger muss bei Empfang einer UDP-Nachricht folgendes unternehmen:
 - die IP-Adressen aus dem ankommenden IP-Paket lesen
 - Der Pseudoheader muss zusammengebaut werden
 - Die Prüfsumme muss ebenfalls berechnet werden
 - Die mit gesendete Prüfsumme mit der berechneten vergleichen
- Wenn die beiden Prüfsummen identisch sind, dann muss das Datagramm seinen Zielrechner und auch den richtigen UDP-Port erreicht haben
 - Ziel des Pseudoheaders ist es somit, beim Empfänger herauszufinden, ob das Paket den richtigen Empfänger gefunden hat

Einige well-known UDP-Portnummern

UDP- Portnummer	Protokoll, Service
53	DNS – Domain Name Service
520	RIP – Routing Information Protocol
161	SNMP – Simple Network Management Protocol
69	TFTP – Trivial File Transfer Protocol

Begrenzte Nachrichtenlänge bei UDP

- Die Länge eines UDP-Segments ist minimal 8 Byte und maximal $2^{16} - 1 = 65.535$ Byte
- Nettodatenlänge = $2^{16} - 1 - 8 = 65.527$ Byte
- Darüber hinaus ist es sinnvoll, die UDP-Segmente nicht länger als in der möglichen IP-Paketlänge zu versenden, da sonst fragmentiert werden muss

Rückblick

1. TCP (Transmission Control Protocol)

- Sliding-Window-Mechanismus
- Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
- Staukontrolle
- TCP-Timer
- TCP-Zustandsautomat

2. UDP (User Data Protocol)

- Einordnung und Aufgaben des Protokolls
- Der UDP-Header
- Datenübertragung

Datenkommunikation

Grundlagen der Vermittlungsschicht

Wintersemester 2012/2013

Einordnung

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

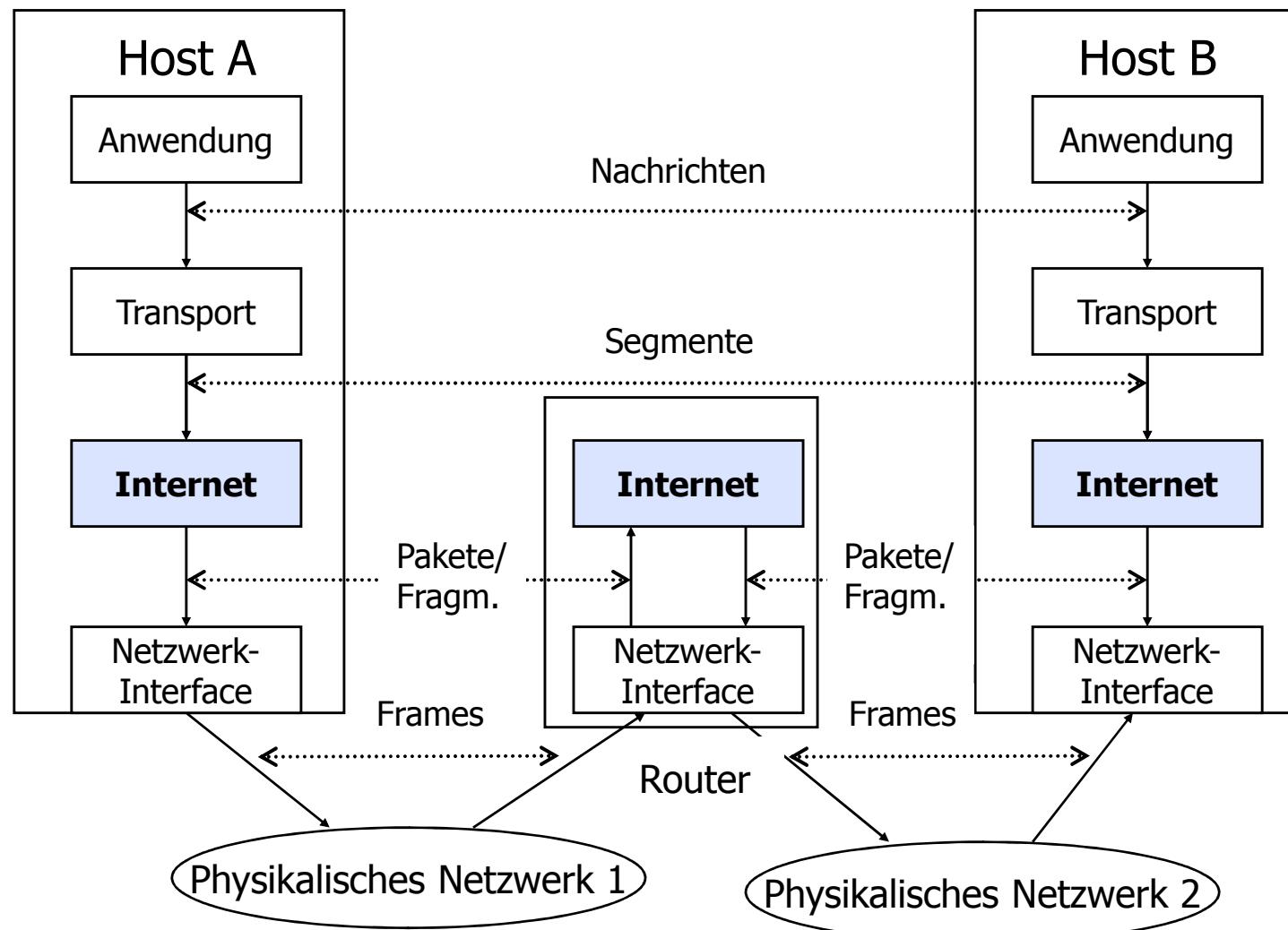
Überblick

1. Einordnung

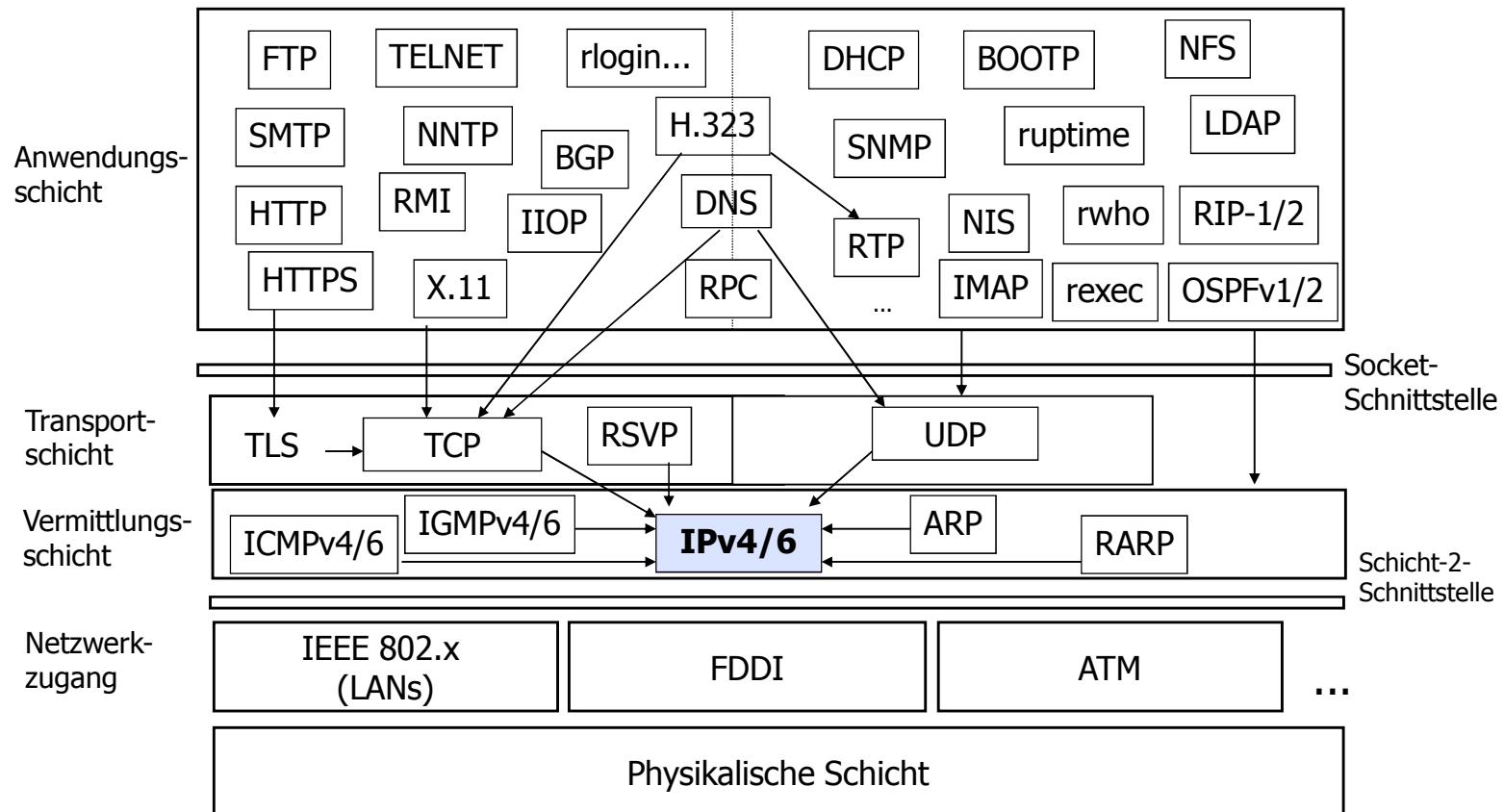
2. Überblick über die Vermittlungsschicht

- Aufgaben und Dienste
- Vermittlungsverfahren
- Wegewahl, Routing
- Diverse Protokollmechanismen

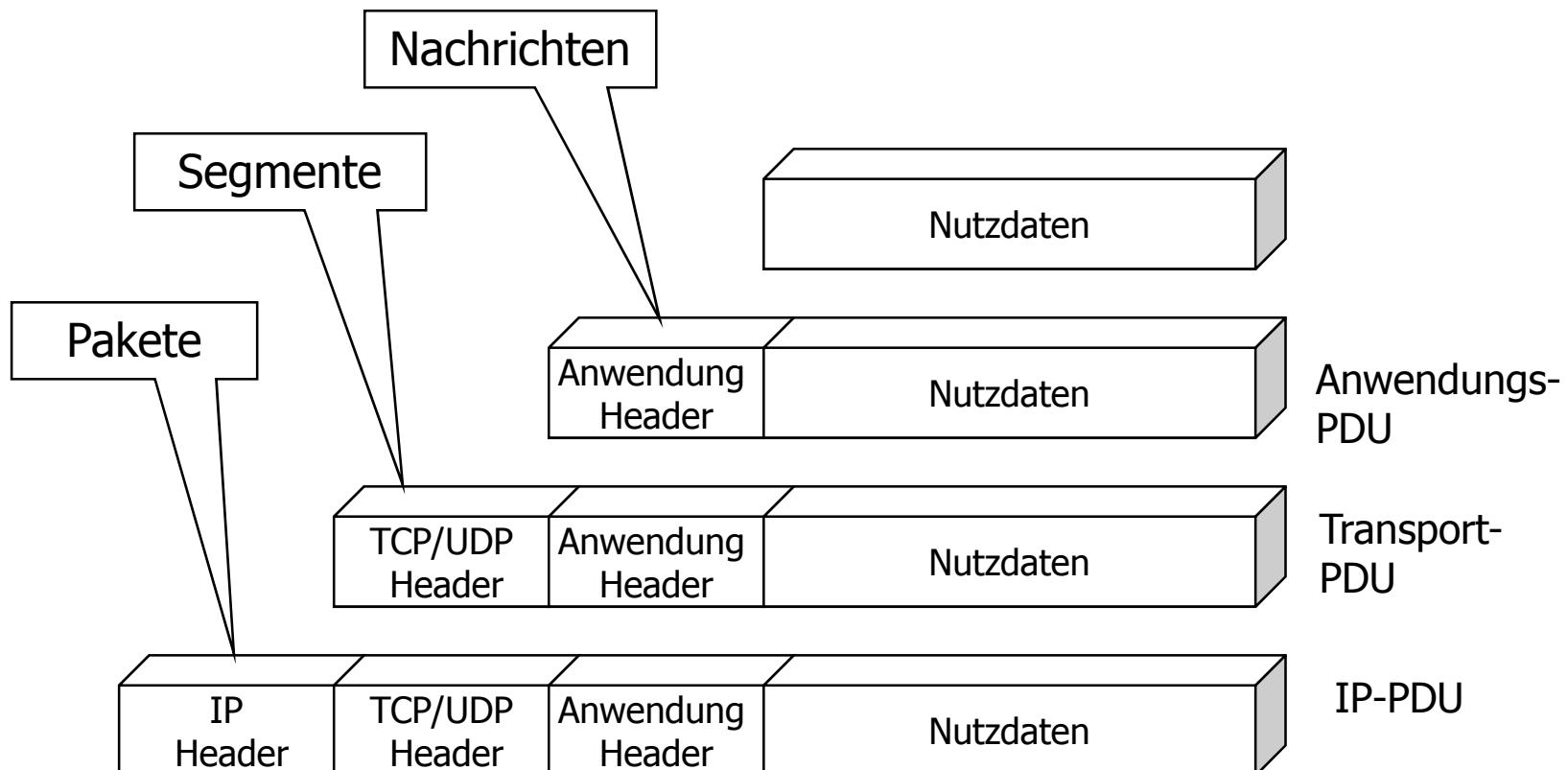
Einordnung: TCP-Referenzmodell



Einordnung TCP/IP-Protokollfamilie



Einordnung TCP/IP-Referenzmodell, Protokollkapselung

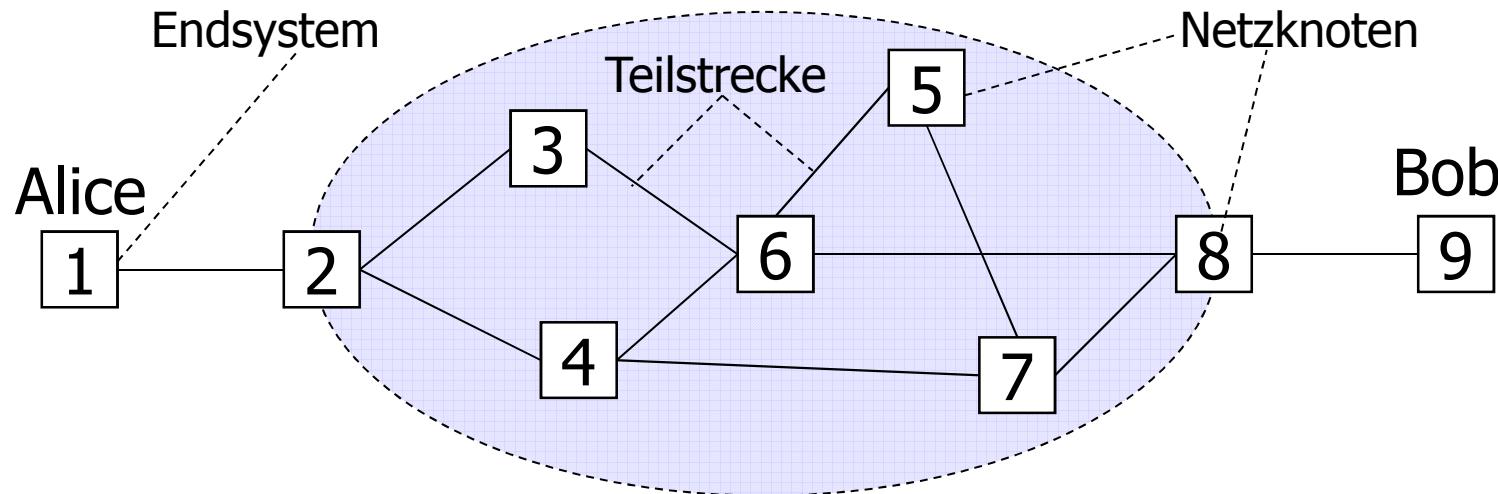


PDU = Protocol Data Unit

Überblick

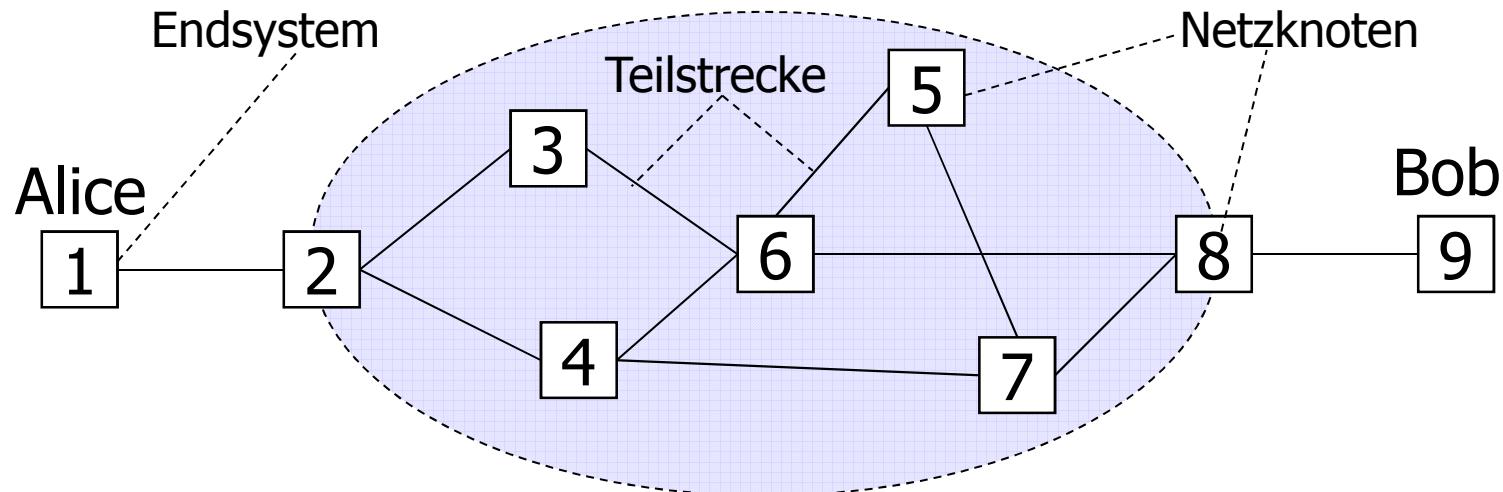
1. Einordnung
2. Überblick über die Vermittlungsschicht
 - **Aufgaben und Dienste**
 - Vermittlungsverfahren
 - Wegewahl, Routing
 - Diverse Protokollmechanismen

Überblick über die Vermittlungsschicht: Aufbau eines Vermittlungsnetzes



- **Netzknoten** sind über **Teilstrecken** miteinander verbunden
- **Endsysteme** (Alice, Bob) sind mit Netzknoten verbunden
- Netzknoten verwaltet zwei oder mehr Verbindungen
- Endsystem hat meist nur eine Verbindung

Überblick über die Vermittlungsschicht: Aufgaben (1)



- Nachricht soll von ‚Alice‘ zu ‚Bob‘ übertragen werden
- Voraussetzung ist eine eindeutige **Adressierung**
- Aufgabenstellung ist vergleichbar mit der Zustellung einer Postkarte

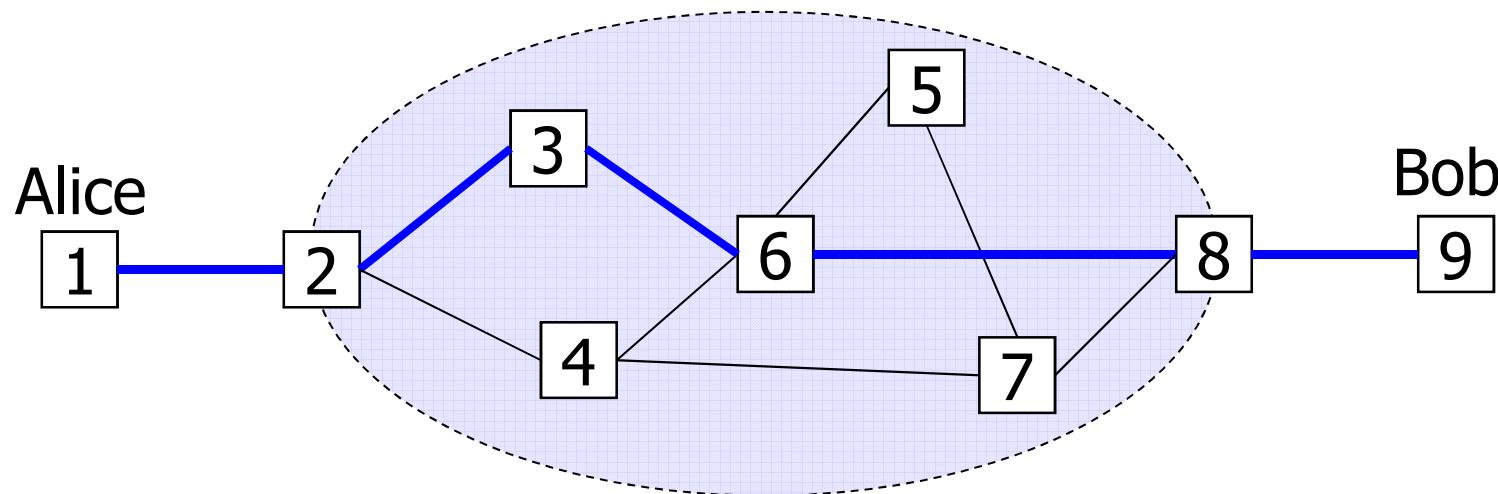
Überblick über die Vermittlungsschicht: Aufgaben (2)

- Die Endsysteme kommunizieren über einen oder mehrere **Netzknotenrechner** (kurz: Netzknoten, Knoten, Router)
- Die Übertragungswege werden von Knoten zu Knoten bereitgestellt (**Teilstrecken**)
- Die **Fehlersicherung** findet auf den Teilstrecken (Schicht zwei) statt
- Die Schnittstelle zur Vermittlungsschicht ist auch meist die **Netzbetreiberschnittstelle**

Überblick über die Vermittlungsschicht: Aufgaben (3)

- Zu den Aufgaben gehören:
 - Wegewahl (auch Routing genannt)
 - Multiplexen und Demultiplexen
 - Staukontrolle (Congestion Control)
 - Fragmentierung/Defragmentierung
- Oft diskutiert:
 - Ist ein verbindungsloser oder verbindungsorientierter Dienst an der Schnittstelle zur Transportschicht besser? (vgl. Tanenbaum)
 - Einschub: Das Internet, verbindungslos oder verbindungsorientiert?

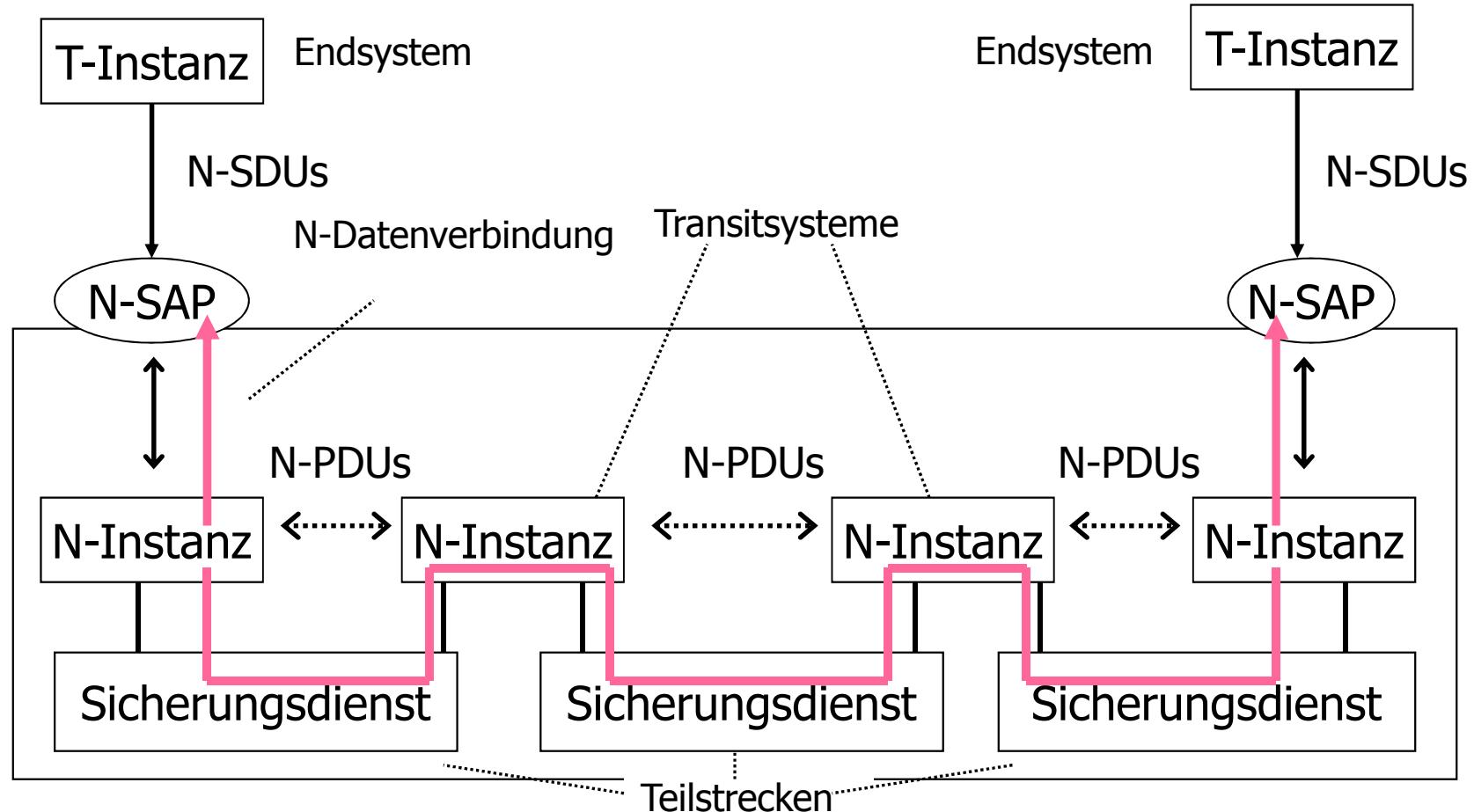
Überblick über die Vermittlungsschicht: Vermittlung, Switching



- Den **Gesamtvorgang** der **Verbindungsherstellung**, des **Haltens** und des **Abbauens** einer Verbindung bezeichnet man als **Vermittlung** (engl. Switching)
- Beispiel: Verbindung von ‚Alice‘ (1) zu ‚Bob‘ (9) über die Netzknoten 2, 3, 6, 8

Überblick über die Vermittlungsschicht: Dienste der Vermittlungsschicht

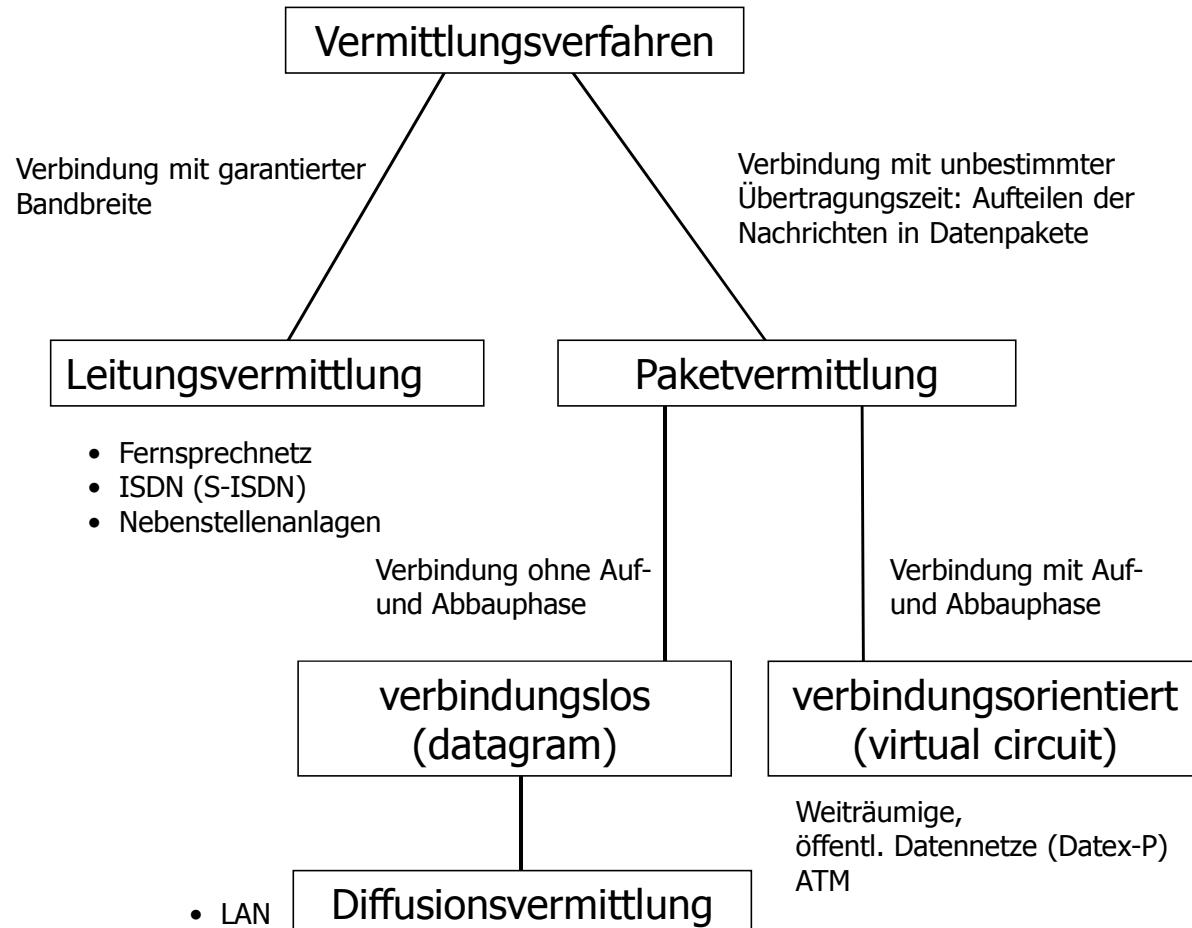
Vgl.: Gerdzen, P., Kommunikationssysteme 1



Überblick

1. Einordnung
2. Überblick über die Vermittlungsschicht
 - Aufgaben und Dienste
 - **Vermittlungsverfahren**
 - Wegewahl, Routing
 - Diverse Protokollmechanismen

Überblick über die Vermittlungsschicht: Vermittlungsverfahren



Vgl. auch : Gerdsen, P., Kommunikationssysteme 1

Überblick über die Vermittlungsschicht: Leitungsvermittlung

- Klassisches Switching-Verfahren, auch **circuit switching** oder **Durchschaltevermittlung** genannt
 - Über die gesamte Verbindung wird **ein physikalischer Verbindungsweg** durch das Netzwerk geschaltet
 - Es wird eine **feste Bandbreite garantiert** und zwar unabhängig von dem, was tatsächlich übertragen wird
 - Evtl. wird Bandbreite unnötig reserviert
 - Blockierungen (Ablehnung eines Verbindungswunsches), wenn kein Verbindungsweg mehr frei ist
 - Beispielnetze, die Leitungsvermittlung nutzen:
 - Analoges Fernsprechnetz
 - Digitales ISDN
-

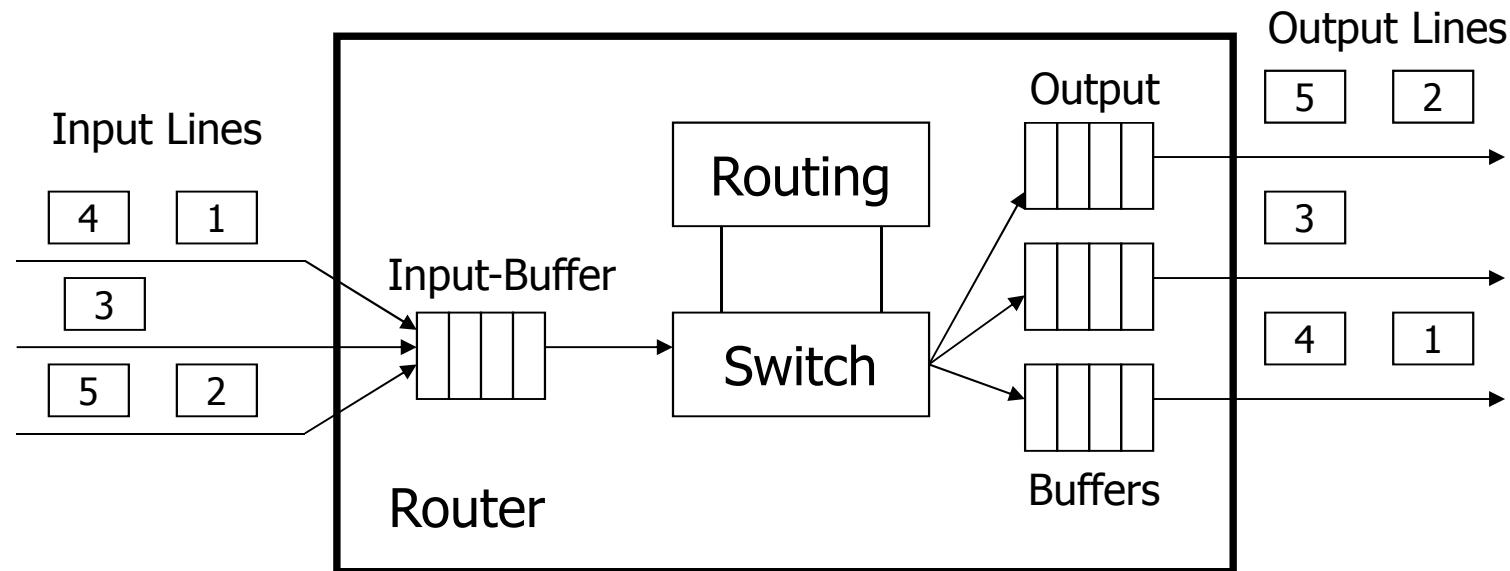
Überblick über die Vermittlungsschicht: Paketvermittlung

- Merkmale:
 - Komplette Nachricht mit Zieladresse wird ins Netz gesendet
 - Netz überträgt die Nachricht evtl. über mehrere Knoten mit Zwischenspeicherung
 - Nachricht wird ggf. in einzelne Pakete (N-PDUs) zerlegt und versendet
- Ist für die Datenübertragung effizienter
- Keine garantierte Bandbreite, dafür aber keine Blockierungen
- Beispiel:
 - Internet Protocol
 - Breitband-ISDN auf Basis von ATM (sehr kurze Pakete, Zellen genannt)

Überblick über die Vermittlungsschicht: Paketvermittlung

- Prinzip der Paketvermittlung in einem Knotenrechner

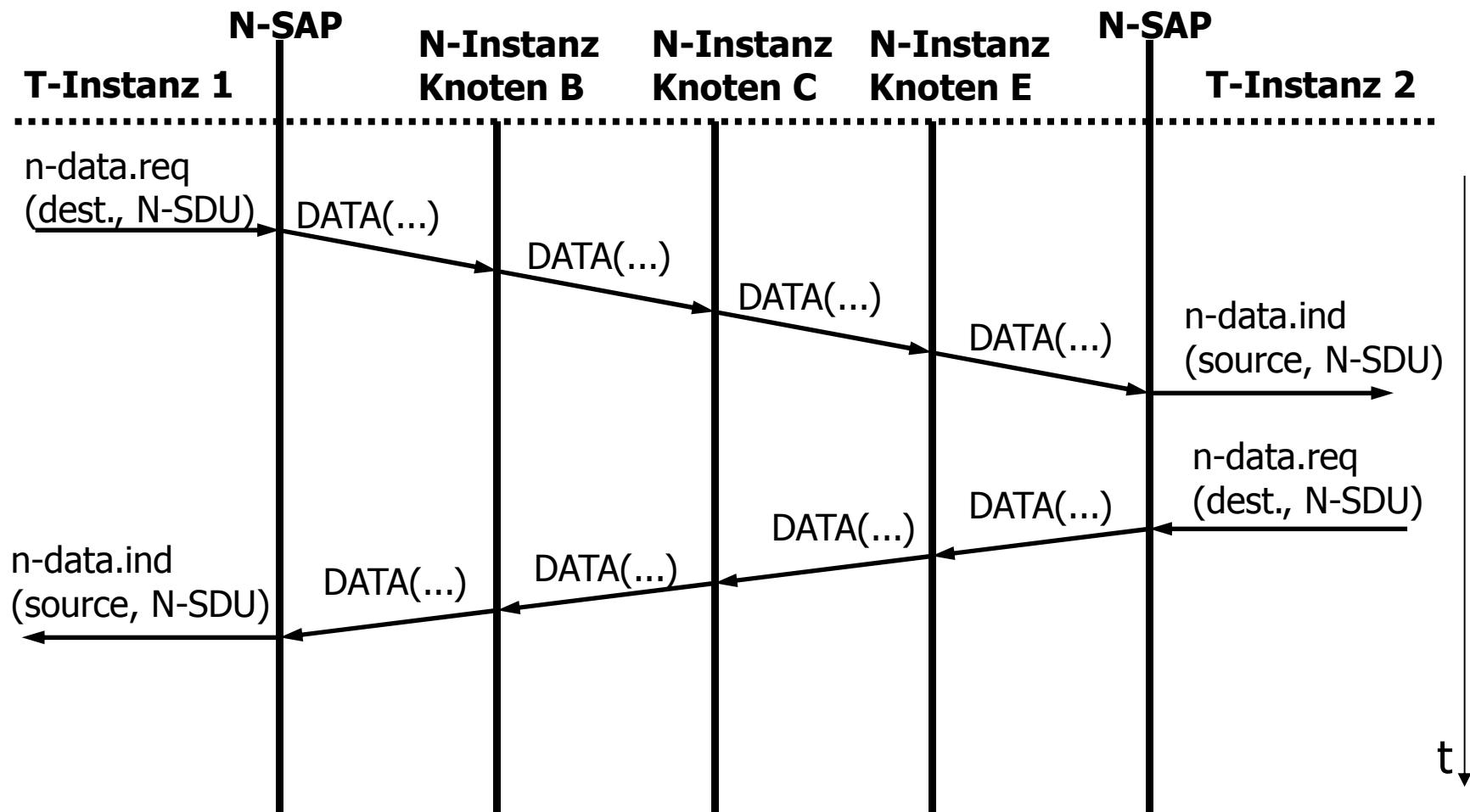
Vgl.: Gerdzen, P., Kommunikationssysteme 1



Überblick über die Vermittlungsschicht: Nutzung von Datagrammen

- Datagramme (N-PDUs) werden bei einer einfachen Paketvermittlung ohne vorhergehenden Verbindungsaufbau verwendet
- Jedes Datagramm enthält die Quell- und die Zieladresse
- Die Knoten **ermitteln** für jedes Datagramm einen **optimalen Weg**
- Wird auch als **verbindungslose** Vermittlung bezeichnet
- Nur ein einfacher data-Dienst zum Senden von Datagrammen erforderlich

Überblick über die Vermittlungsschicht: Datagramm-Vermittlung



Überblick über die Vermittlungsschicht: Datagramm-Vermittlung

- Einfache Form der Datagramm-Vermittlung ist die **Diffusionsvermittlung**
- Hier sendet jeder Knoten die empfangenen Pakete an alle Nachbarknoten weiter
 - mit Ausnahme des sendenden Knotens!
- Sinnvoll bei Netzen mit geringer Knotenzahl
- Klassische Vermittlungsform **in LANs**
 - Siehe z.B. Ethernet-LAN

Überblick über die Vermittlungsschicht: Nutzung von Virtual Circuits

- Virtual Circuits werden auch „**scheinbare Verbindungen**“ genannt
- Reduzierung des aufwändigen Routens bei jedem Paket durch verbindungsorientiertes Verfahren
- Verbindung bleibt für die Dauer der Datenübertragung erhalten
- Kein physikalisches Durchschalten der Verbindung, sondern Nutzung von Routing-Informationen in den Knoten

Überblick über die Vermittlungsschicht: Nutzung von Virtual Circuits

- Drei Phasen der Datenübertragung mit entsprechenden Diensten:
 - Verbindungsaufbau (connect-Dienst)
 - Datenübertragung (data-Dienst)
 - Verbindungsabbau (disconnect-Dienst)
- Die Verbindung zwischen zwei Endsystemen wird schrittweise über Teilstrecken aufgebaut
 - Knoten müssen in der Verbindungsaufbauphase Informationen über das Mapping von eingehenden Paketen zu Ausgangsteilstrecken speichern
 - Statusverwaltung
 - Verbindungstabellen in den Knoten erforderlich

Überblick über die Vermittlungsschicht: Übung zur Wiederholung

- Was versteht man unter einem Teilstreckennetz, auch Store-and-Forward-Netz genannt?
- Was ist im Gegensatz dazu ein Diffusions- oder Broadcast-Netz?

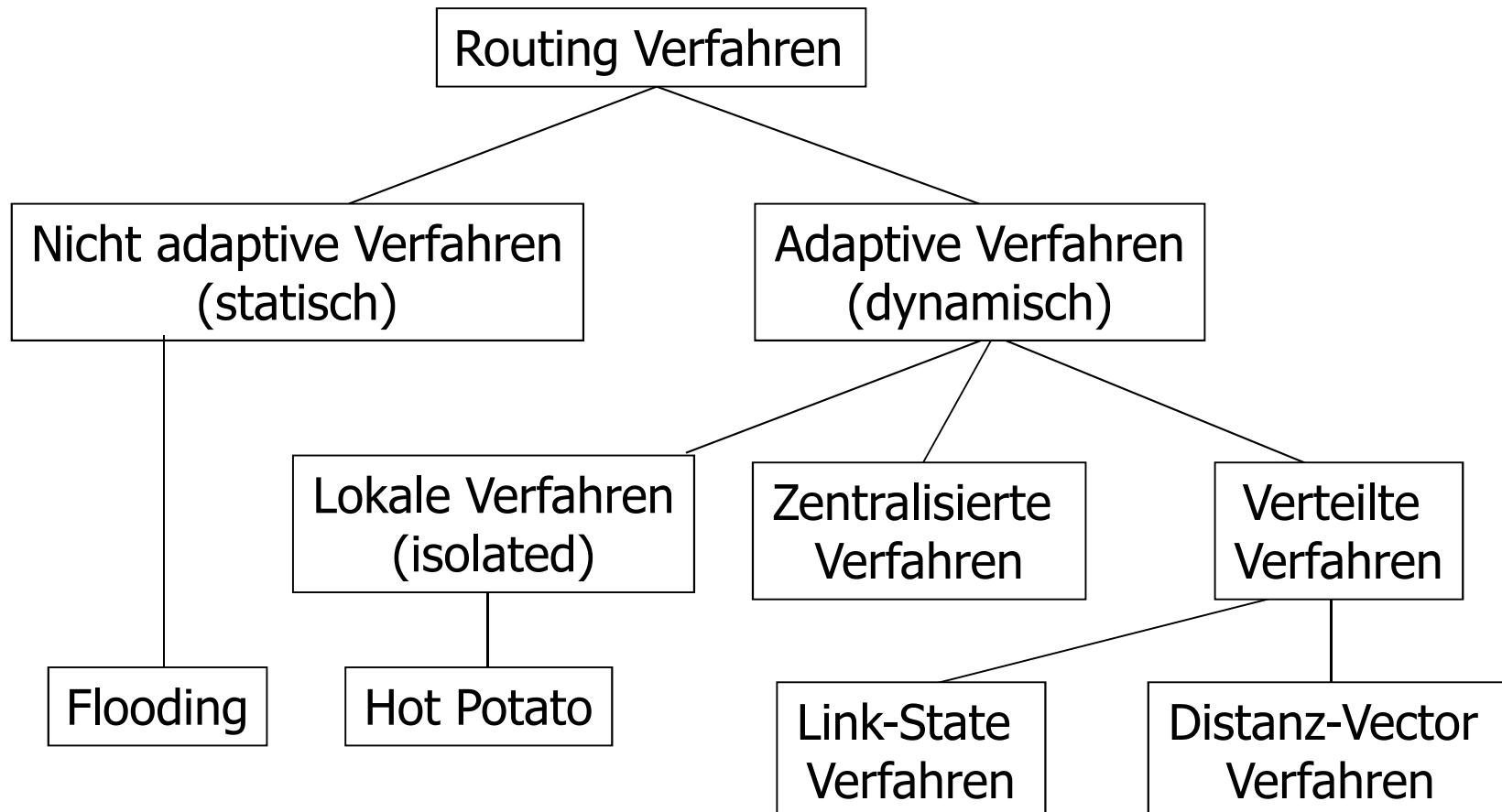
Überblick

1. Einordnung
2. Überblick über die Vermittlungsschicht
 - Aufgaben und Dienste
 - Vermittlungsverfahren
 - **Wegewahl, Routing**
 - Diverse Protokollmechanismen

Überblick über die Vermittlungsschicht: Routing

- Die Wegewahl (Routing) ist eine der wesentlichen Aufgaben der Schicht-3-Instanzen
- Ziel ist es den ‚optimalen‘ Weg zwischen den Endsystemen zu wählen
- Notwendig bei alternativen Wegen zwischen den Endsystemen
- Verschiedene Routing-Kriterien und -Algorithmen sind möglich:
 - Suche der geringsten Entfernung
 - Möglichst geringe Anzahl von Hops (Anzahl der zu durchlaufenden Knoten)
 - Geringste Netzlast
 - ...

Überblick über die Vermittlungsschicht: Routing – Klassifikation der Verfahren



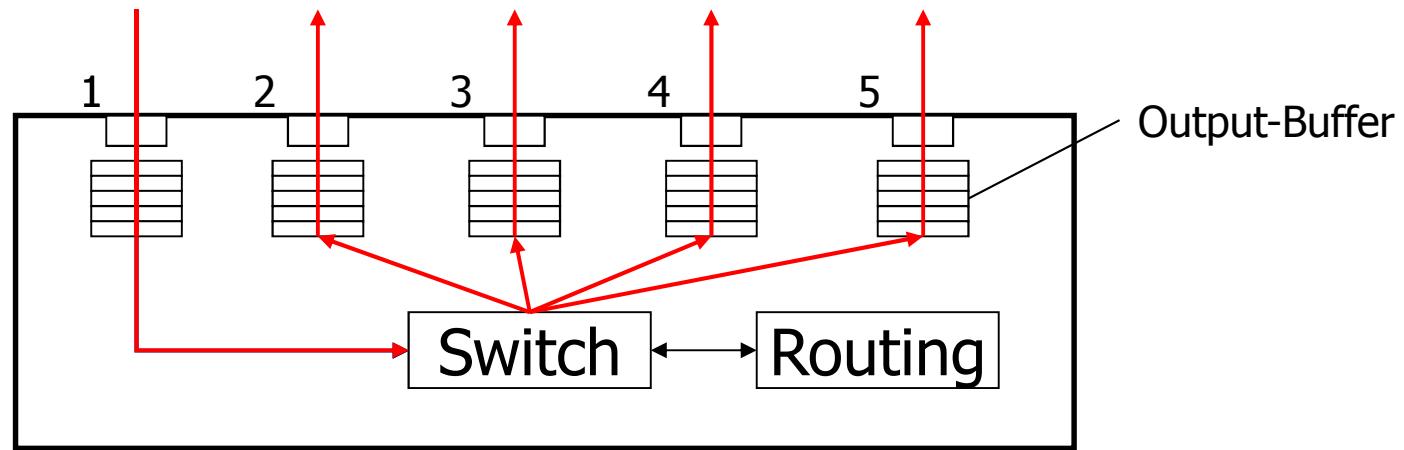
Überblick über die Vermittlungsschicht: Routing - Verfahren

- **Statische** Algorithmen:
 - Keine Messungen, vorher ermittelte Metriken
 - Statische Routing-Tabellen, die bei der Knotenkonfiguration eingerichtet werden (vor Beginn des Betriebs)
- **Dynamische** (adaptive) Algorithmen:
 - Verkehrsmessungen
 - Routing-Tabellen werden dynamisch angepasst (Metriken)
 - Optimierungskriterien können sich dynamisch verändern und werden im Algorithmus berücksichtigt
 - Möglichkeiten:
 - **Isoliertes Routing**: Knoten trifft Entscheidungen alleine
 - **Zentrales** Routing über einen zentralen Knoten (Routing-Kontroll-Zentrum), Zentrale ermittelt und überträgt alle Routing-Tabellen
 - **Dezentrales** Routing mit Routing-Funktionalität in jedem Knoten

Überblick über die Vermittlungsschicht: Routing - Einige Algorithmen

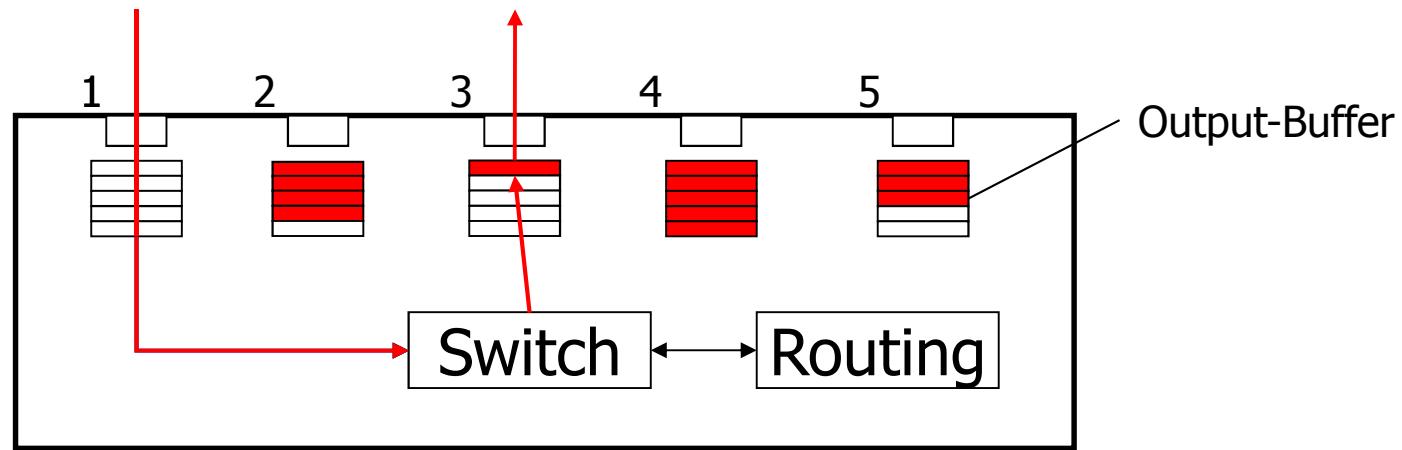
- **Statische** Algorithmen:
 - Shortest-Path-Routing
 - Flooding
- **Dynamische** (adaptive) Algorithmen (heute üblich in modernen Netzen):
 - Distance-Vector-Routing
 - ursprünglicher Algorithmus im ARPANET, RIP
 - Link-State-Routing
 - löste Distance-Vector-Routing Ende der 70er im ARPANET ab
 - Hierarchisches Routing

Überblick über die Vermittlungsschicht: Routing-Beispiel: Flooding (statisch)



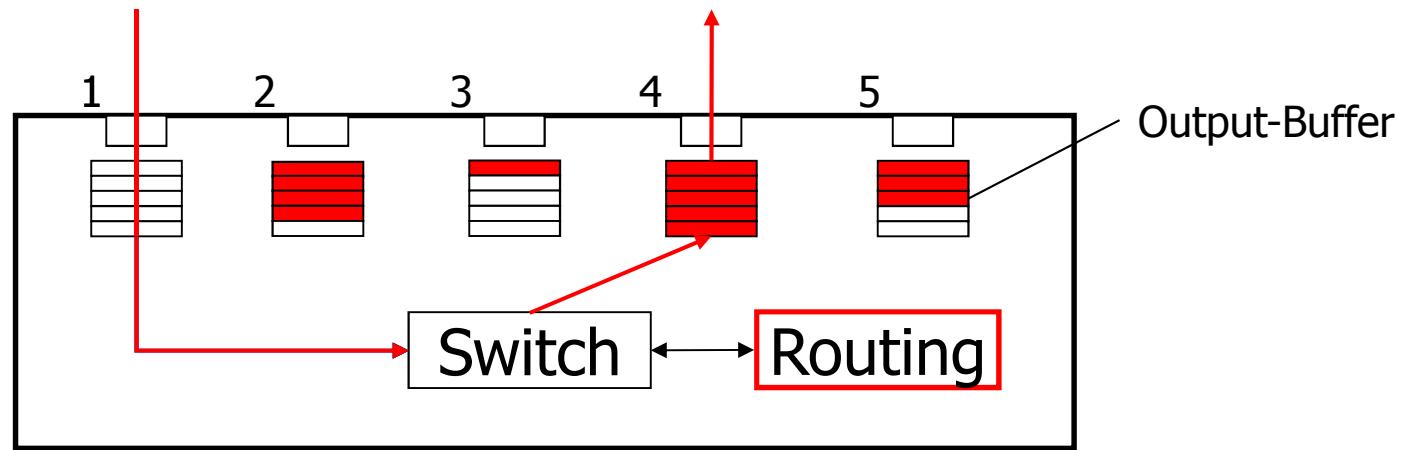
- Eingehende Pakete werden über alle Teilstrecken weiter versendet
- Pakete werden nicht über die eingehende Leitung und nur einmal weiter versendet
- Statischer und sehr einfacher Routing-Algorithmus
- Viele doppelte Pakete und somit ineffizient

Überblick über die Vermittlungsschicht: Routing-Beispiel: Hot Potato (dynamisch / lokal)



- Eingehende Pakete werden so schnell wie möglich zum nächsten Netzknoten gesendet
- Es wird der Ausgang mit dem am geringsten belegten Output-Buffer gewählt
- Dynamischer und sehr einfacher Routing-Algorithmus
- Wird in seiner reinen Form nicht verwendet

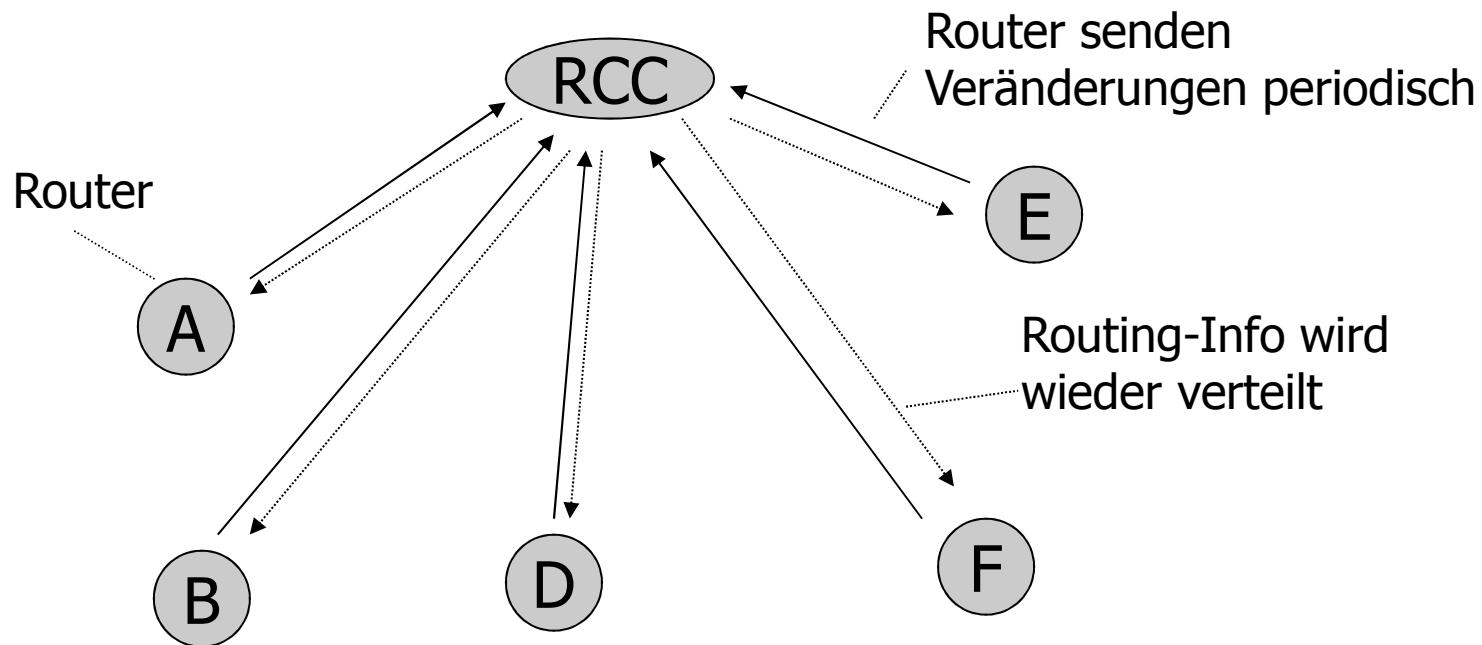
Überblick über die Vermittlungsschicht: Routing-Beispiel: Link-State (dynamisch / verteilt)



- Jeder Router verwaltet eine Kopie der Netzwerktopologie und berechnet selbst die optimale Route für ein Paket
- Verschiedene Optimierungskriterien sind möglich
- Dynamischer und verteilter Routing-Algorithmus
- Router muss komplexe Aufgaben erledigen

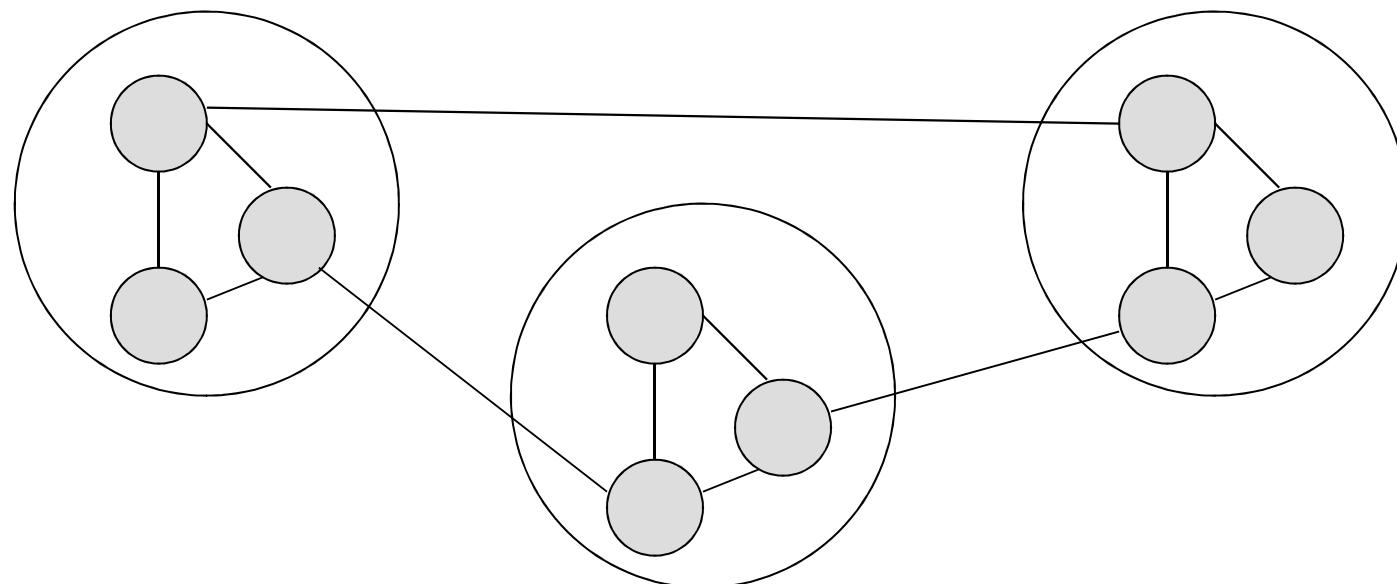
Überblick über die Vermittlungsschicht: Routing – Zentralisiertes Routing

- Es gibt ein Routing Control Center (RCC)
- Verfahren ist nicht fehlertolerant (Engpass) aber konsistent, jedoch Gefahr der veralteten Informationen
- Internet zentral oder dezentral?

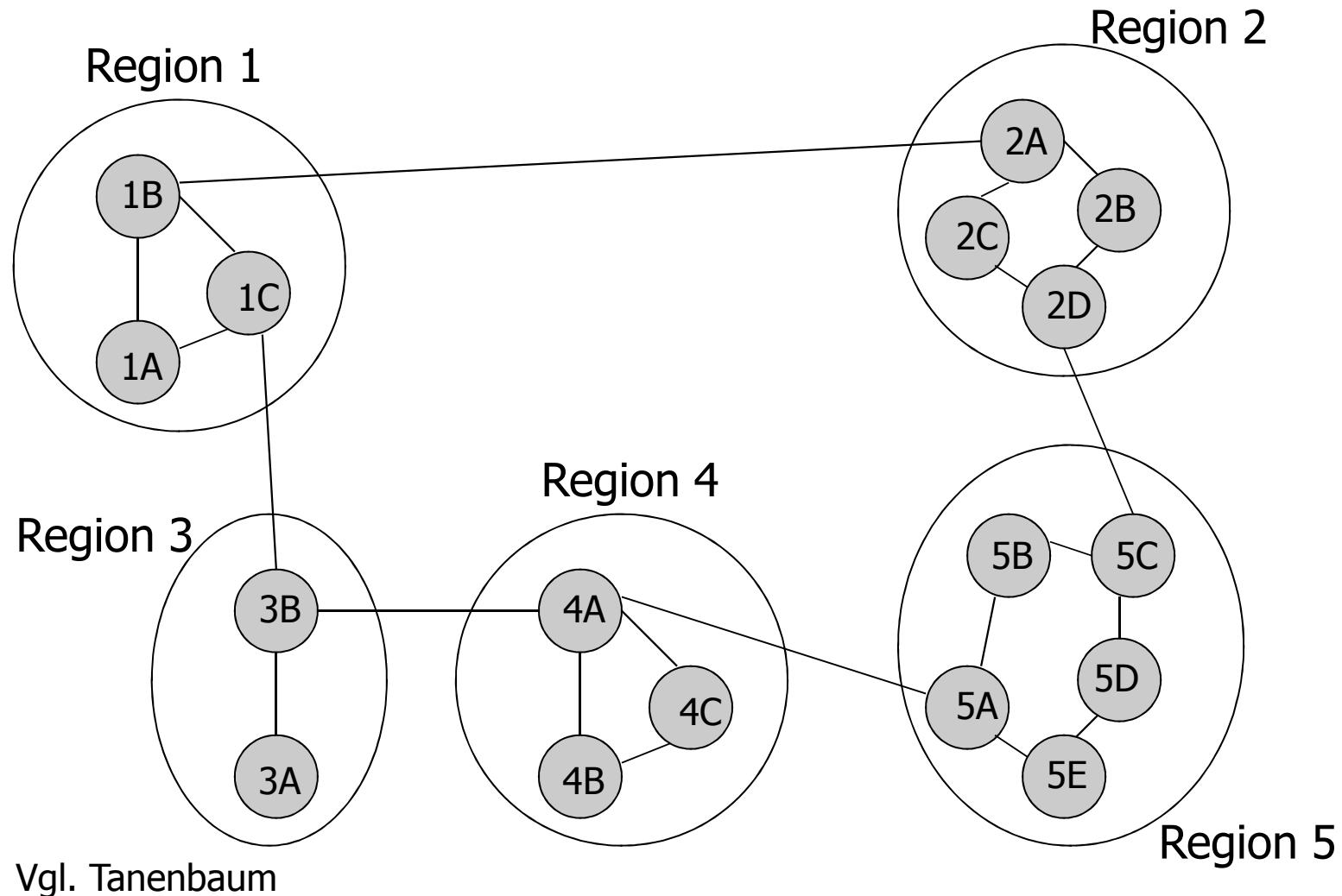


Überblick über die Vermittlungsschicht: Routing – Hierarchisches Routing

- Große Netze erfordern (zu) große Routing-Tabellen mit langen Suchzeiten
- Verringerung der Routing-Tabellen: Netze hierarchisch organisieren
 - Z.B mit folgenden Hierarchiestufen: Regionen – Cluster – ...



Überblick über die Vermittlungsschicht: Routing – Hierarchisches Routing, Beispiel



Überblick über die Vermittlungsschicht: Routing – Hierarchisches Routing, Beispiel

Routing-Tabelle für 1A (vorher)

Ziel	Leitung	Teilstr.
1A	--	--
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

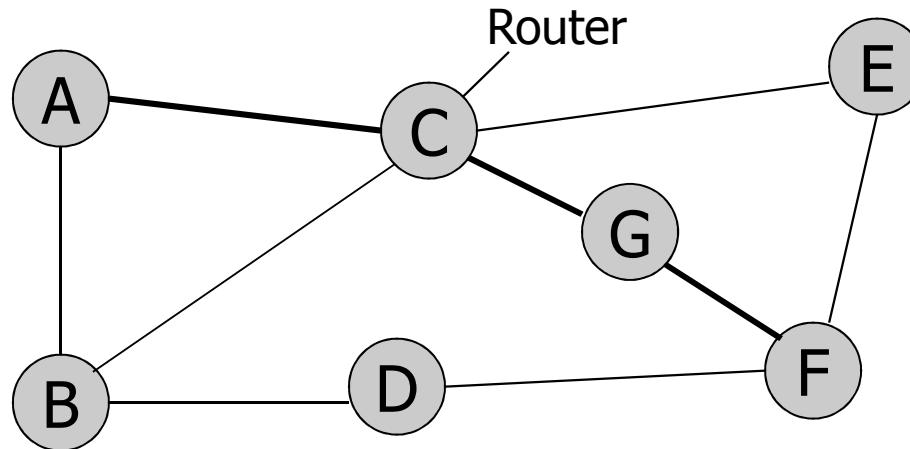
Routing-Tabelle für 1A (nachher)

Ziel	Leitung	Teilstr.
1A	--	--
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

- Reduktion von 17 auf 7 Einträge!
- Nachteil: Evtl. ansteigende Pfadlängen

Überblick über die Vermittlungsschicht: Routing - Optimalitätsprinzip

- Das Optimierungsprinzip (Optimalitätsprinzip nach Richard Bellmann) besagt:
 - Wenn Router C auf dem optimalen Pfad zwischen A und F liegt, dann fällt der Pfad von C nach F ebenso auf diese Route



- Die Annahme, es existiert eine bessere Route zwischen C und F, führt zu einem Widerspruch

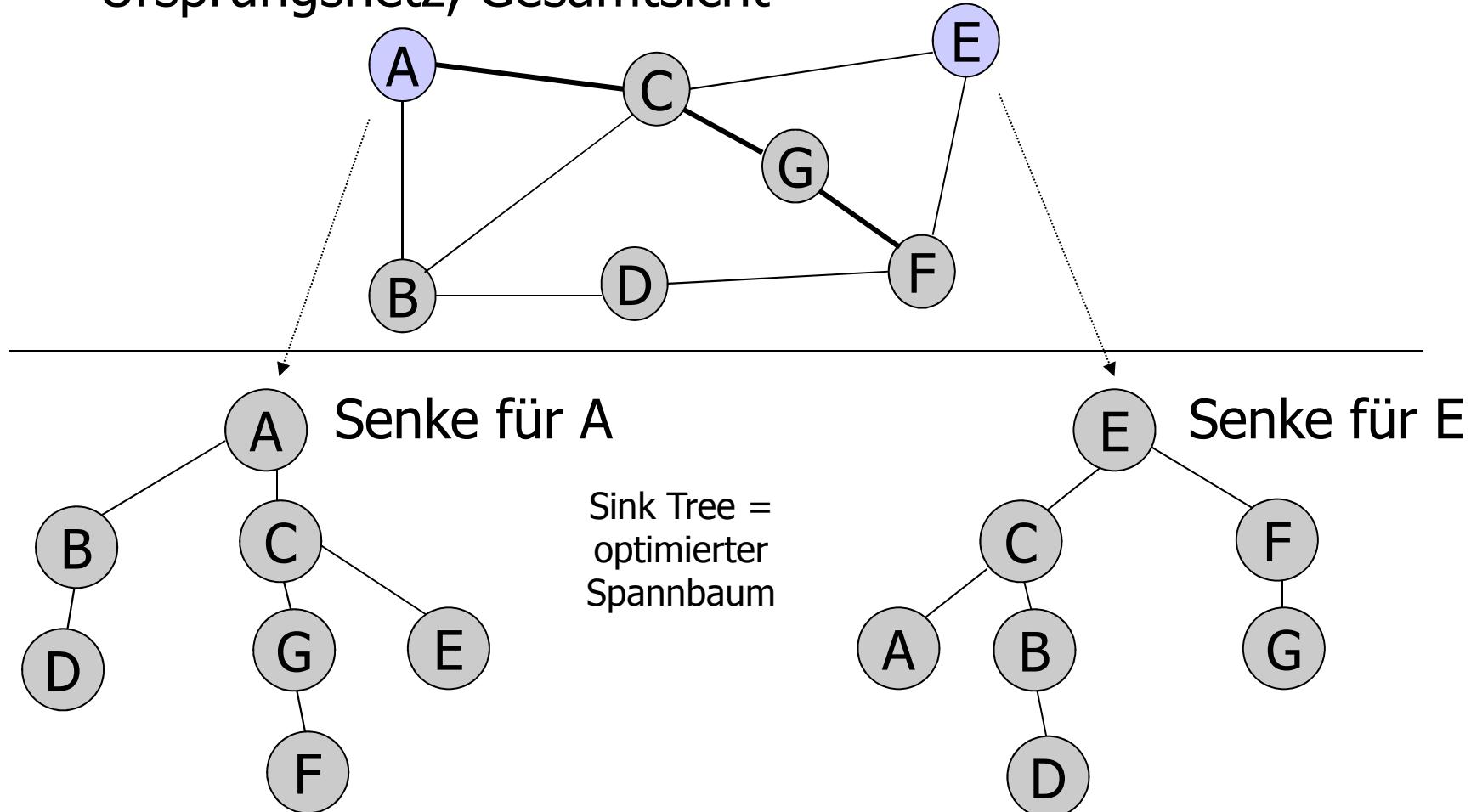
Überblick über die Vermittlungsschicht: Routing - Optimalitätsprinzip

- Optimalitätsprinzip angewendet auf das Routing:
 - Die optimalen Routen von allen Quellen zu einem bestimmten Ziel bilden einen **Baum**, dessen Wurzel das Ziel ist
 - Dieser Baum enthält keine Schleifen und heißt **Sink Tree** oder **Senke** (optimierter Spanning Tree)
- Ziel von Routing-Algorithmen
 - Senken für alle Router ermitteln
 - Senken für das Routing nutzen

Def. Spanning Tree: Teilgraph eines ungerichtete Graphen, der alle Knoten des Graphen enthält. Ein minimaler „Spannbaum“ eines kantengewichteten Graphen hat die kleinste Summe aller Kantengewichte. Es gibt also keinen Spannbaum für den Graphen, der ein kleineres Kantengewicht hat.

Überblick über die Vermittlungsschicht: Routing - Optimierungsprinzip

- Ursprungsnetz, Gesamtsicht



Überblick über die Vermittlungsschicht: Beispiel - Shortest-Path-Routing

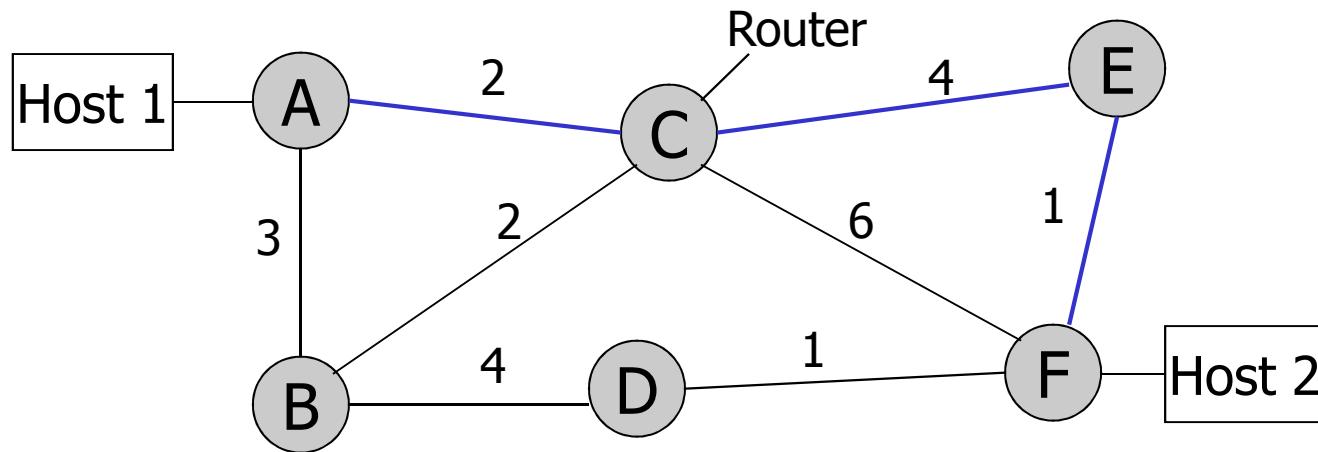
- Graph des Teilnetzes wird erstellt (statisch oder dynamisch)
 - Knoten entspricht Router
 - Kante entspricht einer Leitung zwischen zwei Routern
- Kante wird beschriftet („Pfadlänge“), die Metrik hierfür kann berechnet werden aus
 - Entfernung
 - Bandbreite
 - Durchschnittsverkehr
 - Durchschnittliche Warteschlangenlänge in den Routern
 - Verzögerung
 - ...
- Berechnung des kürzesten Pfads z.B. über Dijkstras oder Bellmanns Algorithmus (siehe Tanenbaum)

Überblick über die Vermittlungsschicht: Beispiel - Shortest-Path-Routing

- Dijkstras Algorithmus (1959)
- Problemstellung aus der Graphentheorie
 - Finde für einen **Startknoten s** und einen **Endknoten e** eines **gewichteten Graphen G** mit der **Knotenmenge V**, der **Kantenmenge E** und der **Kostenfunktion k** einen Weg zwischen s und e mit minimalen Kosten
 - Die Kostenfunktion k bezieht sich auf eine Kante zwischen zwei Knoten

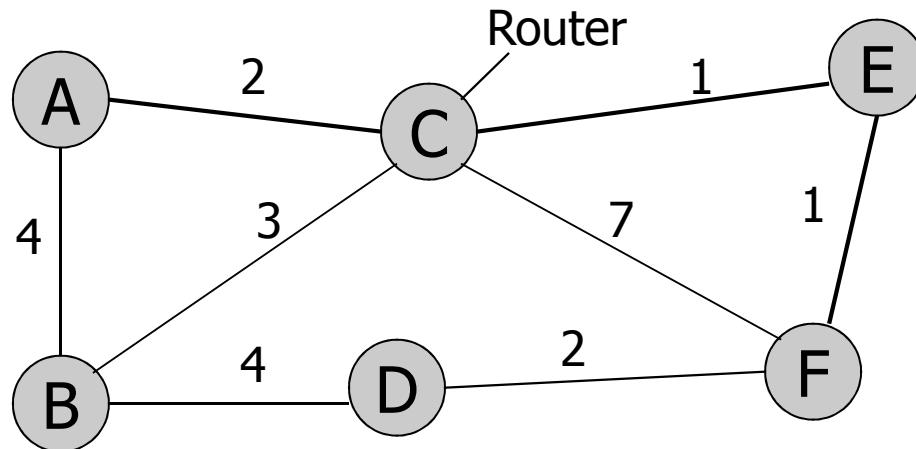
Überblick über die Vermittlungsschicht: Beispiel – Shortest-Path-Routing

- Beispiel: Der kürzeste Pfad zwischen Host1 und Host2 geht von A nach F über A C E F
- Summierte Pfadlänge = 7



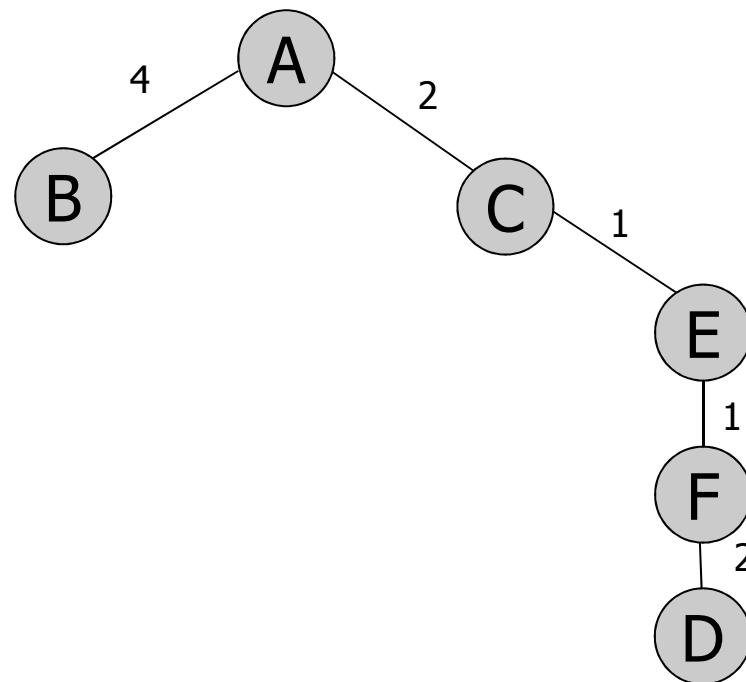
Überblick über die Vermittlungsschicht: Routing - Übung

- Ermitteln Sie die Summe der Kantengewichte zu den Sink Trees (optimale Spannbäume) des vorliegenden kantengewichteten Graphen für die Knoten A und E und zeichnen Sie die Graphen.
- Beschriften Sie die Kanten mit den Kantengewichten!



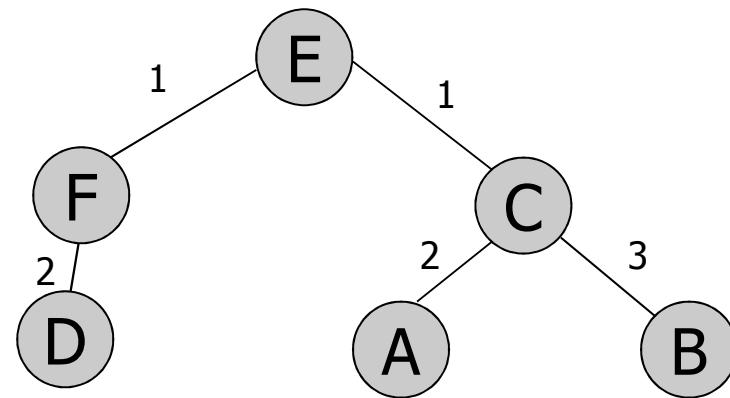
Überblick über die Vermittlungsschicht: Routing – Lösung zur Übung (1)

- Sink Tree (= optimaler Spannbaum) für A
- Summe der Kantengewichte = 10



Überblick über die Vermittlungsschicht: Routing – Lösung zur Übung (2)

- Sink Tree (= optimaler Spannbaum) für E
- Summe der Kantengewichte = 9



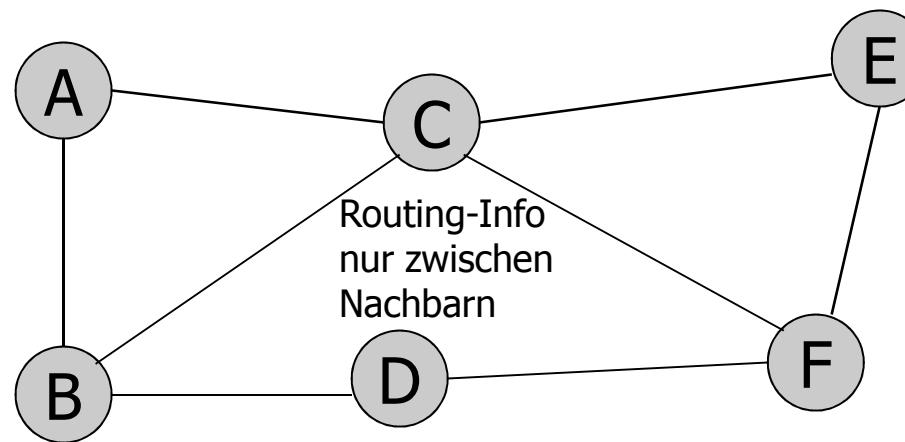
Überblick über die Vermittlungsschicht: Beispiel – Distance-Vector-Routing

- Andere Bezeichnung: Bellman-Ford-Routing (Bellman, 1957 und Ford, 1962)
- Jeder Router führt eine **dynamisch aktualisierte Routing-Tabelle** mit allen Zielen
 - Einträge enthalten bevorzugte Ausgangsleitung zu einem Ziel
- **Metrik** kann z.B. sein:
 - Verzögerung in ms
 - Anzahl der Teilstrecken (**Hops**) zum Ziel

Ziel	Distanz	Nächster Knoten	Hops

Überblick über die Vermittlungsschicht: Beispiel – Distance-Vector-Routing

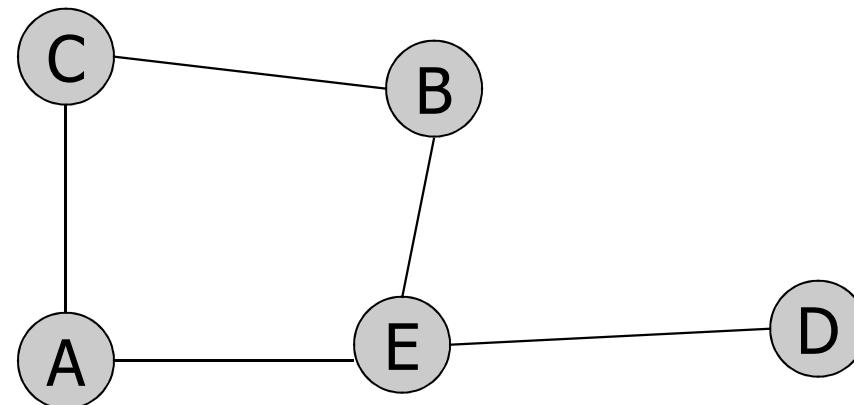
- Verteilt – iterativ – asynchron (unabhängig voneinander)
- **Benachbarte** Router tauschen Routing-Information aus



- **Schleifen** möglich „**Count-to-Infinity-Problem**“
 - Bei Ausfall eines Links evtl. keine Terminierung mehr sichergestellt
- **Gute Nachrichten** verbreiten sich schnell
- **Schlechte Konvergenz**
 - Schlechte Nachrichten verbreiten sich sehr langsam in Netz

Überblick über die Vermittlungsschicht: Distance-Vector-Routing - Kommunikation

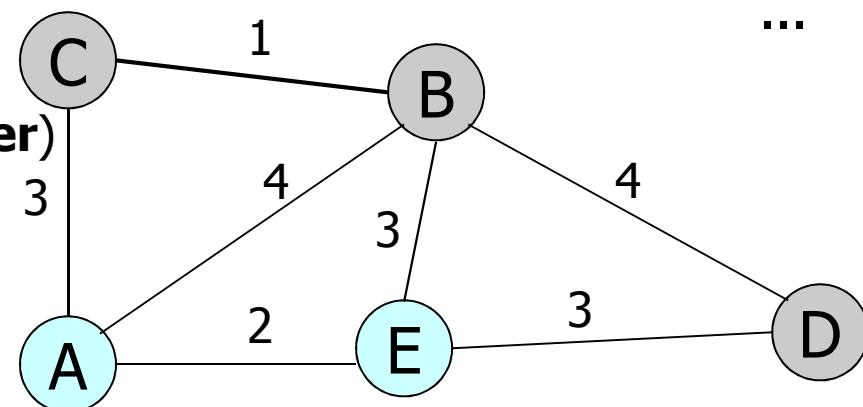
- Routing-Informationen werden nur zwischen Nachbarn ausgetauscht
- Es wird nur die Sicht der Nachbarn und nicht die gesamte Topologie kommuniziert
- Beispiel:
 - C kommuniziert nur mit A und B
 - E Kommuniziert nur mit A, B und D
 - D kommuniziert nur mit E,...
 - B teilt z. B. C mit, dass er E über einen und D über zwei Hops erreichen kann



Überblick über die Vermittlungsschicht: Beispiel – Distance-Vector-Routing

Knoten A				Knoten E			
Ziel	Distanz	Nächster Knoten	Hops	Ziel	Distanz	Nächster Knoten	Hops
B	4	B	1	A	2	A	1
C	3	C	1	B	3	B	1
E	2	E	1	C	4	B	2
D	5	E	2	D	3	D	1

Nach einiger Zeit (**Konvergenzdauer**)
verfügen alle Router über optimale
Routing-Tabellen



Überblick über die Vermittlungsschicht: Link-State-Routing

- Jeder Router verwaltet eine **Kopie der Netzwerktopologie** (Link-State-Datenbank)
- **Zielsetzung:** Jeder Knoten muss alle Kosteninformationen kennen
- Jeder Router verteilt die lokale Information per Flooding an alle anderen Router im Netz
- Die Berechnung der Routen erfolgt **dezentral**
- Jeder Knoten errechnet den **absolut kürzesten Pfad**

Überblick über die Vermittlungsschicht: Link-State-Routing

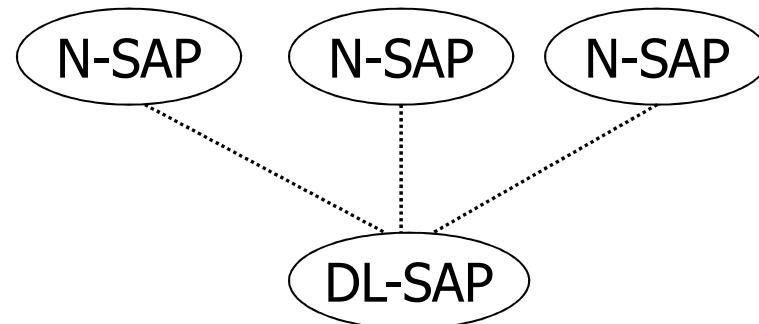
- **Berechnung der kürzesten Pfade** z.B. über Shortest-Path-Algorithmus (z.B. Dijkstras Algorithmus)
- **Keine Schleifen** möglich, da jeder Knoten die gleiche Information über die Topologie besitzt
- Schnelle Reaktion auf Topologieänderungen möglich

Überblick

1. Einordnung
2. Überblick über die Vermittlungsschicht
 - Aufgaben und Dienste
 - Vermittlungsverfahren
 - Wegewahl, Routing
 - **Diverse Protokollmechanismen**

Überblick über die Vermittlungsschicht: Multiplexen

- Gemeinsame Verwendung einer Teilstrecke, also einer Schicht-2-Verbindung, für mehrere Schicht-3-Verbindungen
- Erster Schicht-3-Verbindungsaufbau baut auch die Teilstreckenverbindung auf
- Weitere Schicht-3-Verbindungen können dann bestehende Schicht-2-Verbindung nutzen

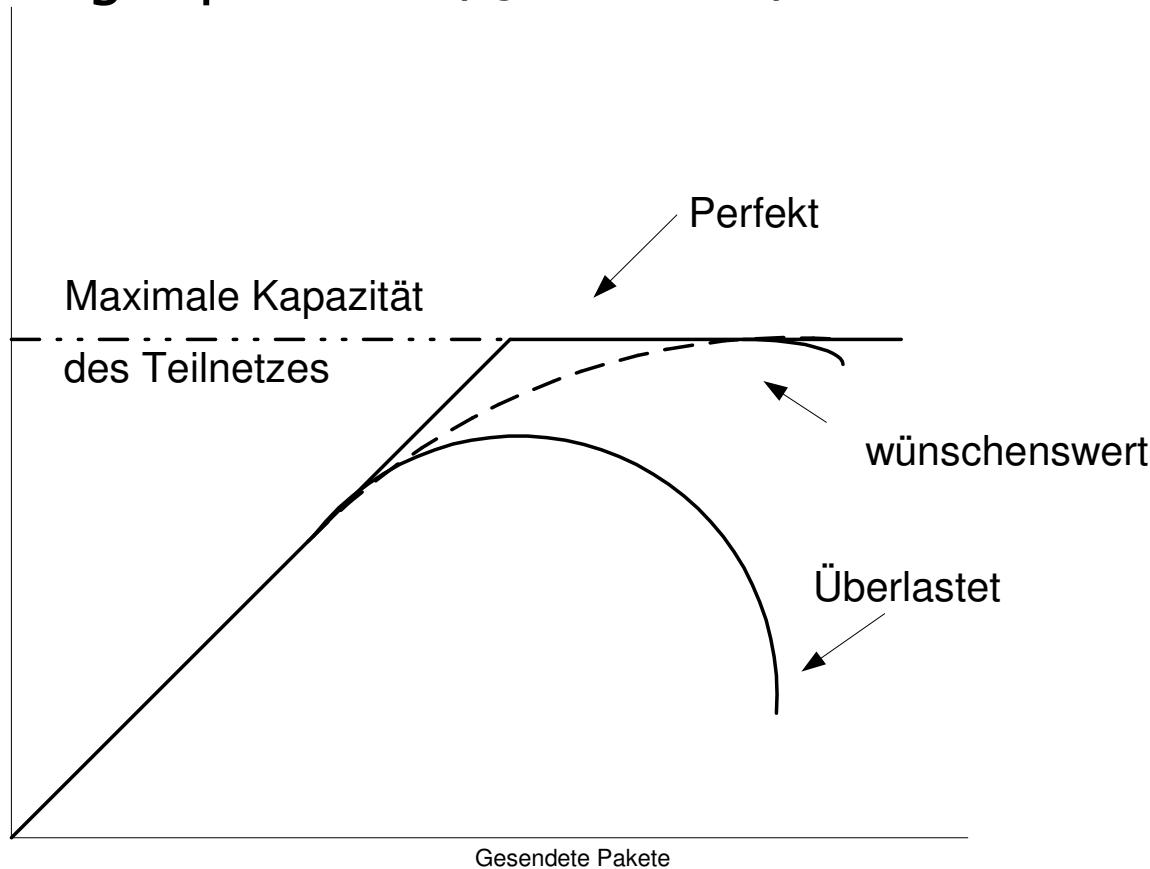


Überblick über die Vermittlungsschicht: Überlastung und Staukontrolle

- Zu viele Pakete (mehr als die Übertragungskapazität) in einem Netz führen zum **Abfall der Leistung** → Überlastung (Congestion), Verstopfung
- Ursachen:
 - Viele Pakete zu einer Zeit → Lange Warteschlangen
 - Langsame Prozessoren in den Netzknoten
 - Zu wenig Speicher in den Netzknoten
- Eine Überlastung kann zu einem **Teufelskreis** führen
 - Pakete gehen verloren
 - Pakete werden evtl. in Netzknoten verworfen
 - Sendungswiederholungen erhöhen die Last

Überblick über die Vermittlungsschicht: Überlastung und Staukontrolle

- „Bei übermäßiger Verkehrsbelastung des Netzes sinkt die Leistung rapide ab“ (Vgl. Tanenbaum)



Überblick über die Vermittlungsschicht: Überlastung und Staukontrolle

- Durch Staukontrolle sollen Verstopfungen bzw. Überlastungen im Netz vermieden werden
- Möglichkeiten der Staukontrolle: (vgl. Kerner, S.165ff)
 - Lokale Steuerung (gehört zur Schicht 2), da sie sich auf Einzelleitungen bezieht
 - Ende-zu-Ende-Steuerung zwischen Endsystemen (Schicht 4)
 - Globale Steuerung über das gesamte Netz (Schicht 3)
- Die Schichten 2-4 können Maßnahmen ergreifen
- Im Gegensatz zur Flusssteuerung ist die Staukontrolle ein Mechanismus mit **netzglobalen Auswirkungen**

Überblick über die Vermittlungsschicht: Staukontrolle, Maßnahmen

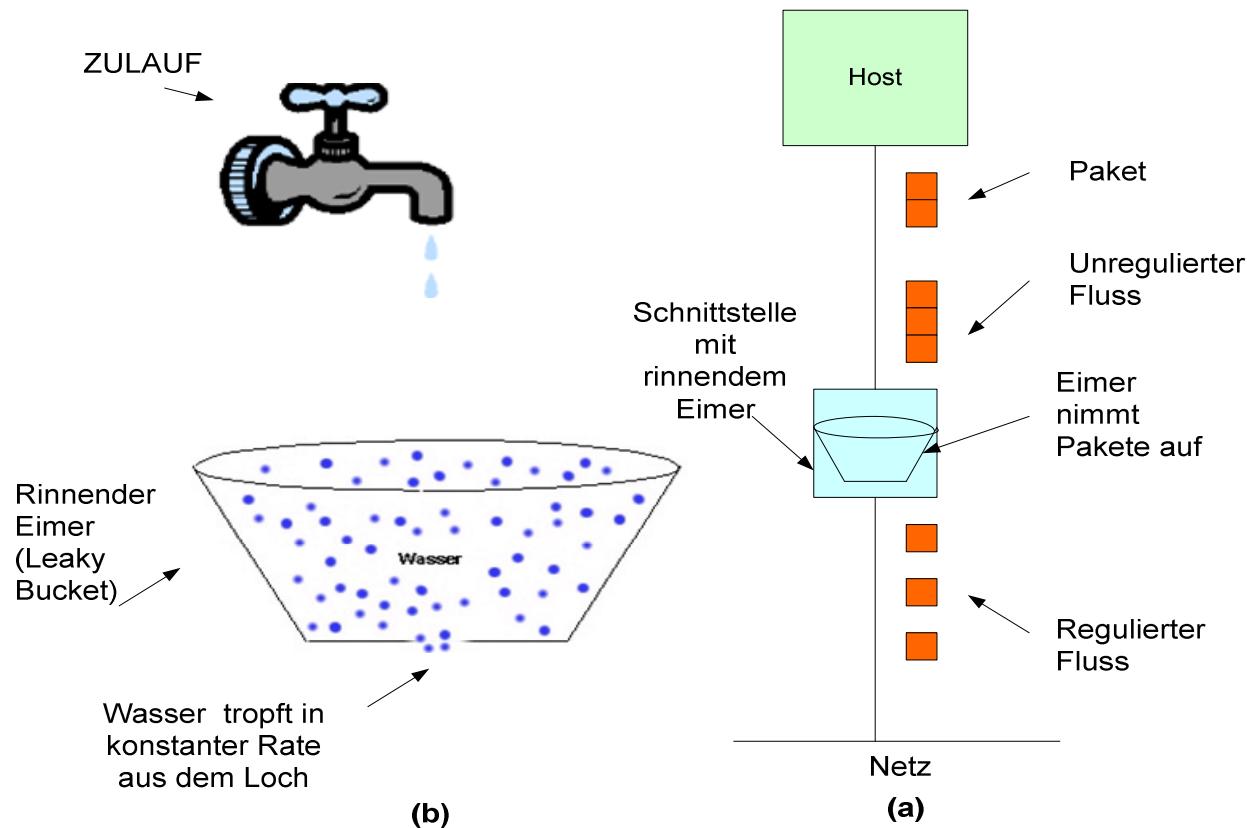
Schicht	Maßnahmen (vgl. Tanenbaum)
Transportschicht	<ul style="list-style-type: none">• ...• ...• ...
Vermittlungsschicht	<ul style="list-style-type: none">• Virtuelle Verbindungen statt Datagrammen in Teilnetzen• Warteschlangen verwalten• Routing-Algorithmus• Verwaltung der Lebensdauer von Paketen
Sicherungsschicht	<ul style="list-style-type: none">• Erneute Übertragung• Zwischenspeichern von Paketen• Bestätigungen• Flusssteuerung

Überblick über die Vermittlungsschicht: Staukontrolle, Traffic Shaping

- Eine wesentliche Ursache für Überlastungen sind „**Verkehrsspitzen**“ (vgl. Tanenbaum)
- **Traffic Shaping** ist eine Maßnahme zur Regulierung der durchschnittlichen Datenübertragungsrate von Endsystemen
- Überwachung der Endsysteme (**Traffic Policing**) durch den Netzbetreiber
- Besser realisierbar für virtuelle Verbindungen (virtual circuits) als für Datagramm-orientierte Netze
- Nutzung des **Leaky-Bucket**-Algorithmus

Überblick über die Vermittlungsschicht: Staukontrolle, Leaky-Bucket-Algorithmus

(Vgl. Tanenbaum)



Überblick über die Vermittlungsschicht: Staukontrolle, Leaky-Bucket-Algorithmus

- Endsysteme (Hosts) verfügen über Netzwerkschnittstellen (Kernel, Netzwerkkarte) mit einer **internen Warteschlange** → rinnender Eimer
- Wenn die Warteschlange voll ist, wird ein neues Paket **schn im Endsystem verworfen**
- Sender, Empfänger und Netzwerk müssen sich einig sein
 - **Flussspezifikation** für virtuelle Verbindungen bei Verbindungsaufbau
 - Man einigt sich über die max. Paketgröße, die max. Übertragungsrate,...
- Weitere Algorithmen: siehe Tanenbaum

Rückblick

1. Einordnung
2. Überblick über die Vermittlungsschicht
 - Aufgaben und Dienste
 - Vermittlungsverfahren
 - Wegewahl, Routing
 - Diverse Protokollmechanismen

Datenkommunikation

Internet und IP-Protokoll

Wintersemester 2012/2013

Überblick

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

1. Überblick

- Internet-Vermittlungsschicht
- Autonome Systeme (AS)
- Organisation
- IPv4: Überblick und Aufgaben

2. IPv4-Adressierung

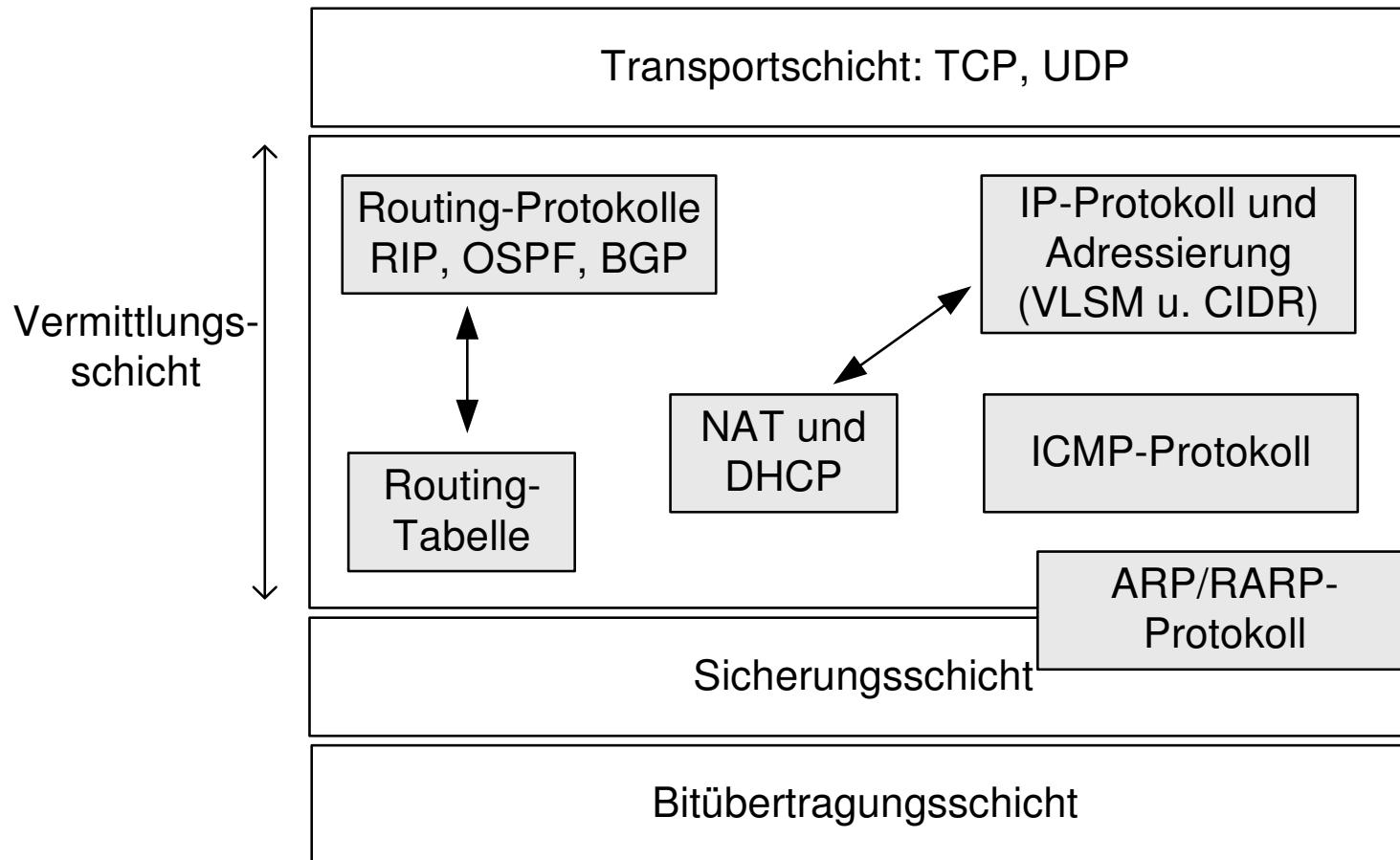
- IPv4: Adressierung und Adressenknappheit
- IPv4-Subnetting
- VLSM und CIDR

3. IPv4-PDU

- Aufbau und Felder

4. Fragmentierung

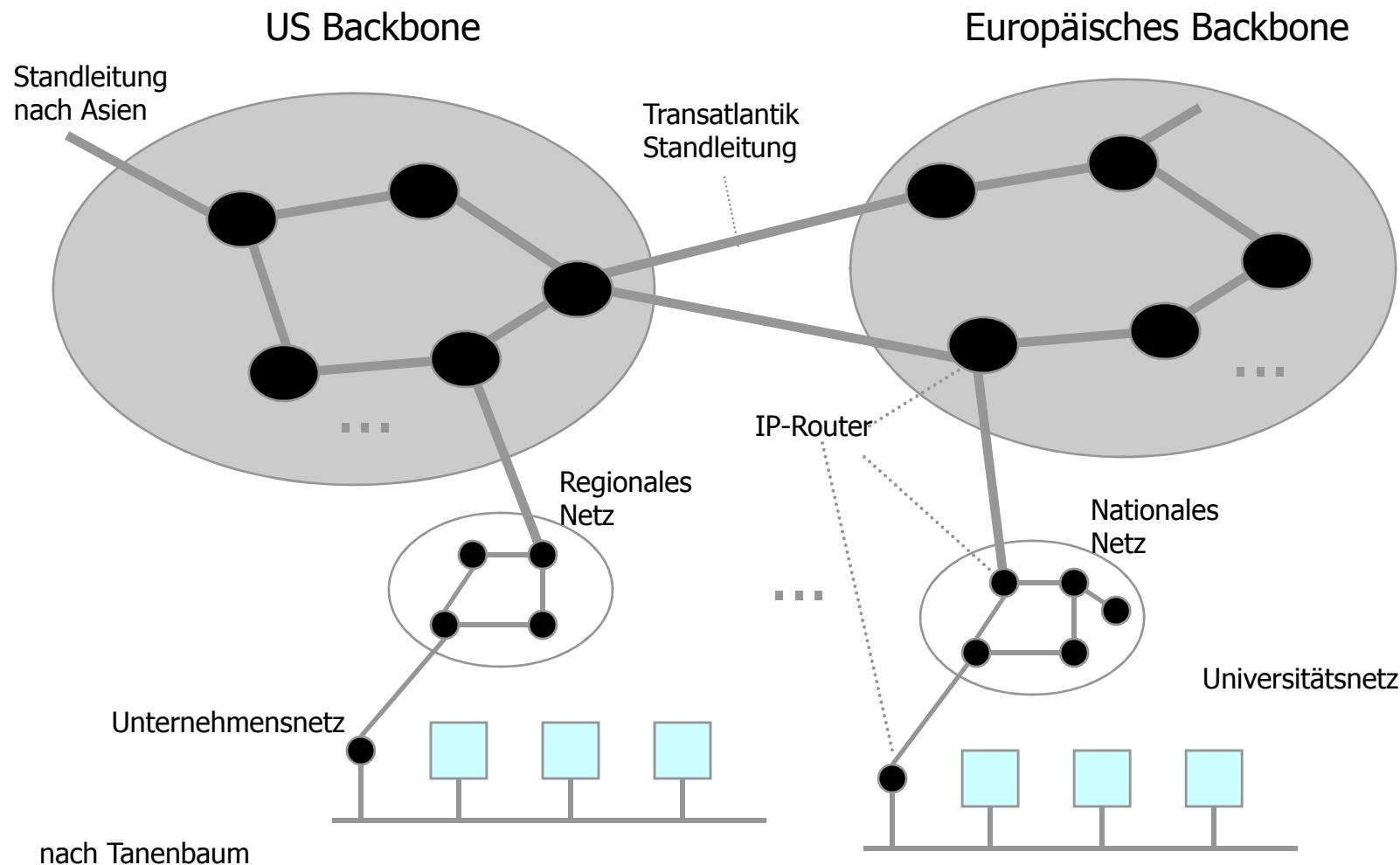
Überblick: Die Internet-Vermittlungsschicht



Überblick über das Internet

- Das Internet ist eine **hierarchische Organisation** des Netzwerks
- **Große Backbones** sind über Leitungen mit hoher Bandbreite und schnellen Routern verbunden
- An den Backbones hängen **regionale Netze**
- An den regionalen Netzen hängen die **Netze von Unternehmen, Universitäten, Internet Service Providern (ISP),...**
- Der Zugriff über das Internet von Deutschland aus auf einen Server in den USA wird über mehrere IP-Router geroutet

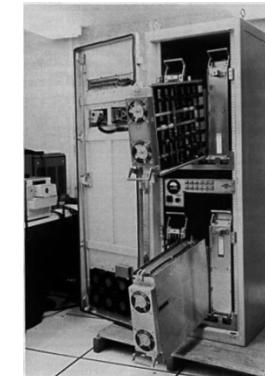
Überblick über das Internet, Backbone



Das Internet heute, AS

- Autonome Systeme haben sich unterschiedlich entwickelt → z.B. verschiedene Routing-Strategien
- Es gibt derzeit mehr als 110.000 autonome Systeme weltweit
- Jedes AS hat eine **eindeutige Nummer**
 - 11, Harward-University
 - 1248, Nokia
 - 2022, Siemens
 - 3680, Novell
 - 4183, Compuserve
 - 6142, Sun
 - 12816, MWN

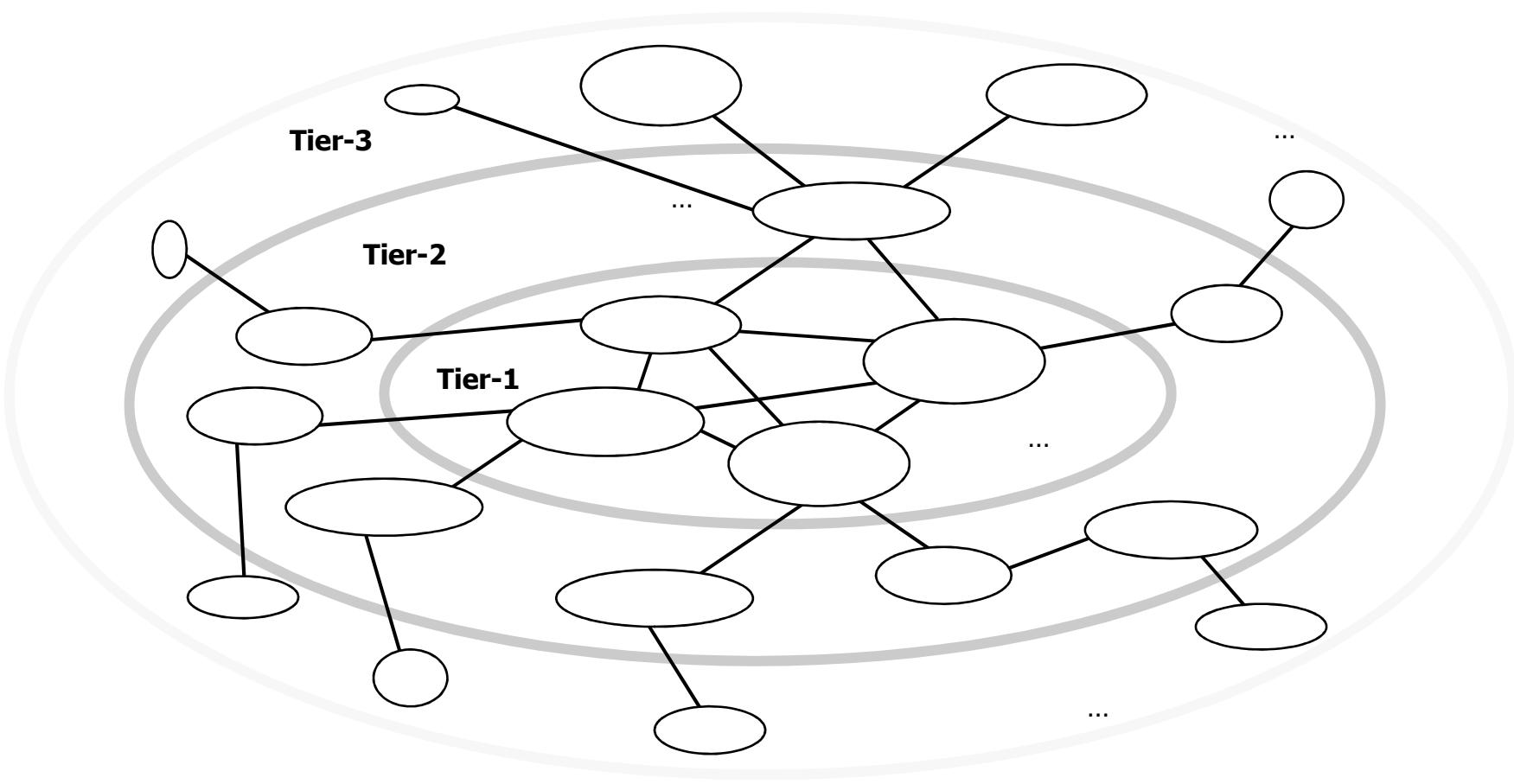
ARPANET-IMP



Das Internet heute, Begriffe

- Autonome Systeme = AS
- Internet-Provider = Provider
- Tier-1-AS, Tier-2-AS, Tier-3-AS (andere Bezeichnung: Tier-x-Netzwerke)
 - Edge-Networks = Tier-1-AS
- Internet-Knoten (oft gemeinsam von ISPs finanziert)
 - = IX (Internet Exchange) = IXP (Internet Exchange Point)
 - = NAP (Network Access Point) → USA
- Public und private Peering-Points
- Peering-Agreement = Peering-Vereinbarung:
 - Übertragung ohne Gebühr bei gleichberechtigten Peers
- Transit-Vereinbarung:
 - Gebühr nach vereinbarter Bandbreite

Das Internet heute, Verbindungen zwischen AS

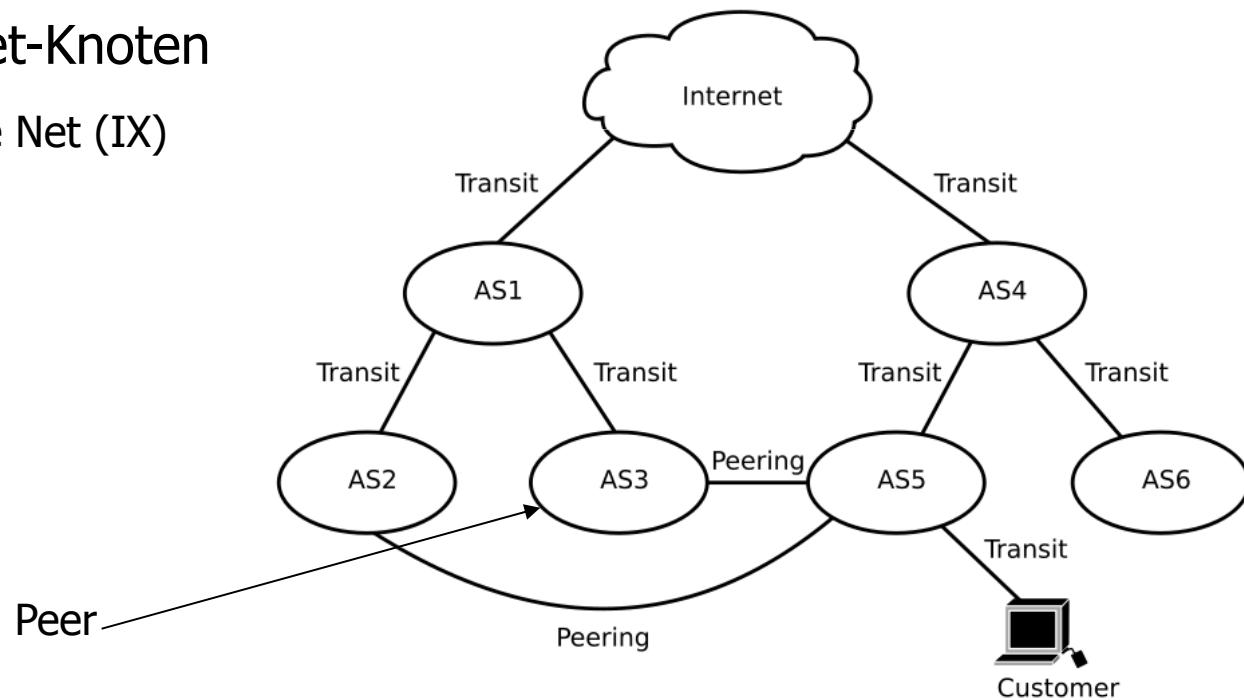


Das Internet heute: AS-Kategorien

- **Tier-3:** Kleine, lokale Provider, Endkundengeschäft
 - M-net in Bayern (Hauptgesellschafter: Münchner Stadtwerke)
 - Hansenet (Hamburg, Tochter der Telecom Italia)
 - Versatel (Berlin)
- **Tier-2:** Betreiber großer, überregionaler Netze
 - Deutsche Telekom
 - France Telecom und France Telecom
 - Tiscali (Telekom-Unternehmen, Italien)
- **Tier-1:** Betreiber von globalen Internet-Backbones
 - AT&T (US-amerikanischer Telekomanbieter)
 - AOL (US-amerikanischer Online-Dienst)
 - NTT (Nippon Telegraph and Telephone Corporation)
 - Verizon Communications (US-amerikanischer Telekomanbieter)

Das Internet heute, Kunden – Provider - Peers

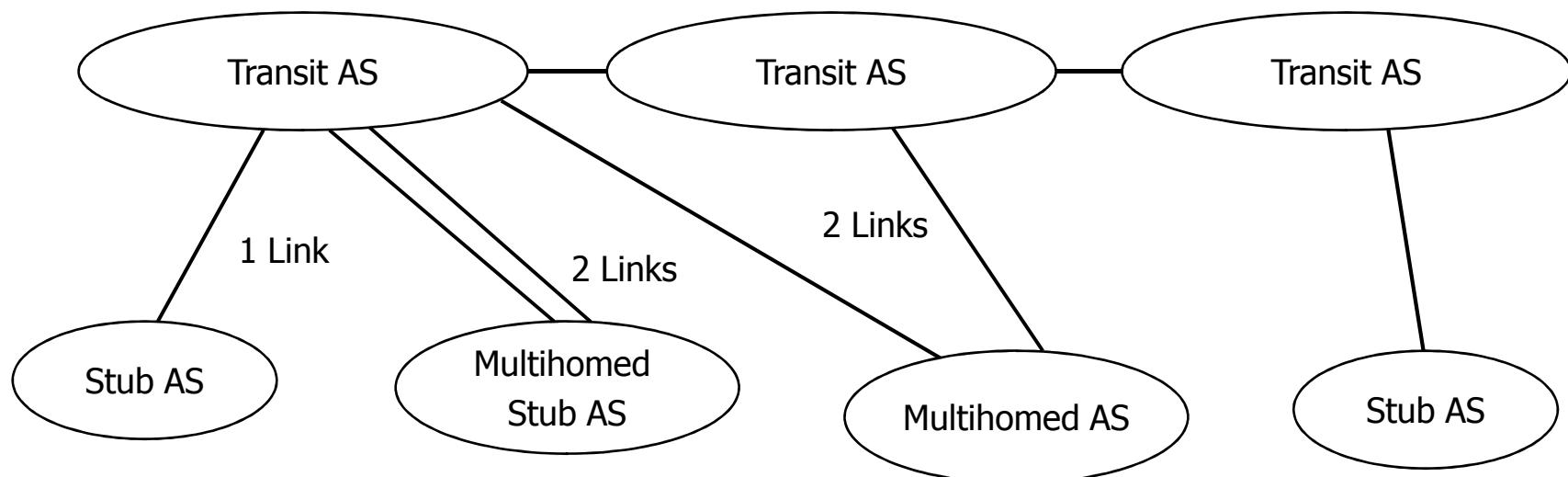
- Rollen: Kunden - Provider - Peers
- Peering-Abkommen zwischen AS (Tier-1, Tier-2, Tier-3)
- Tier-1-Provider „peeren“ kostenlos miteinander, die anderen je nach Vereinbarung
- Netz aus Internet-Knoten
→Internet Exchange Net (IX)



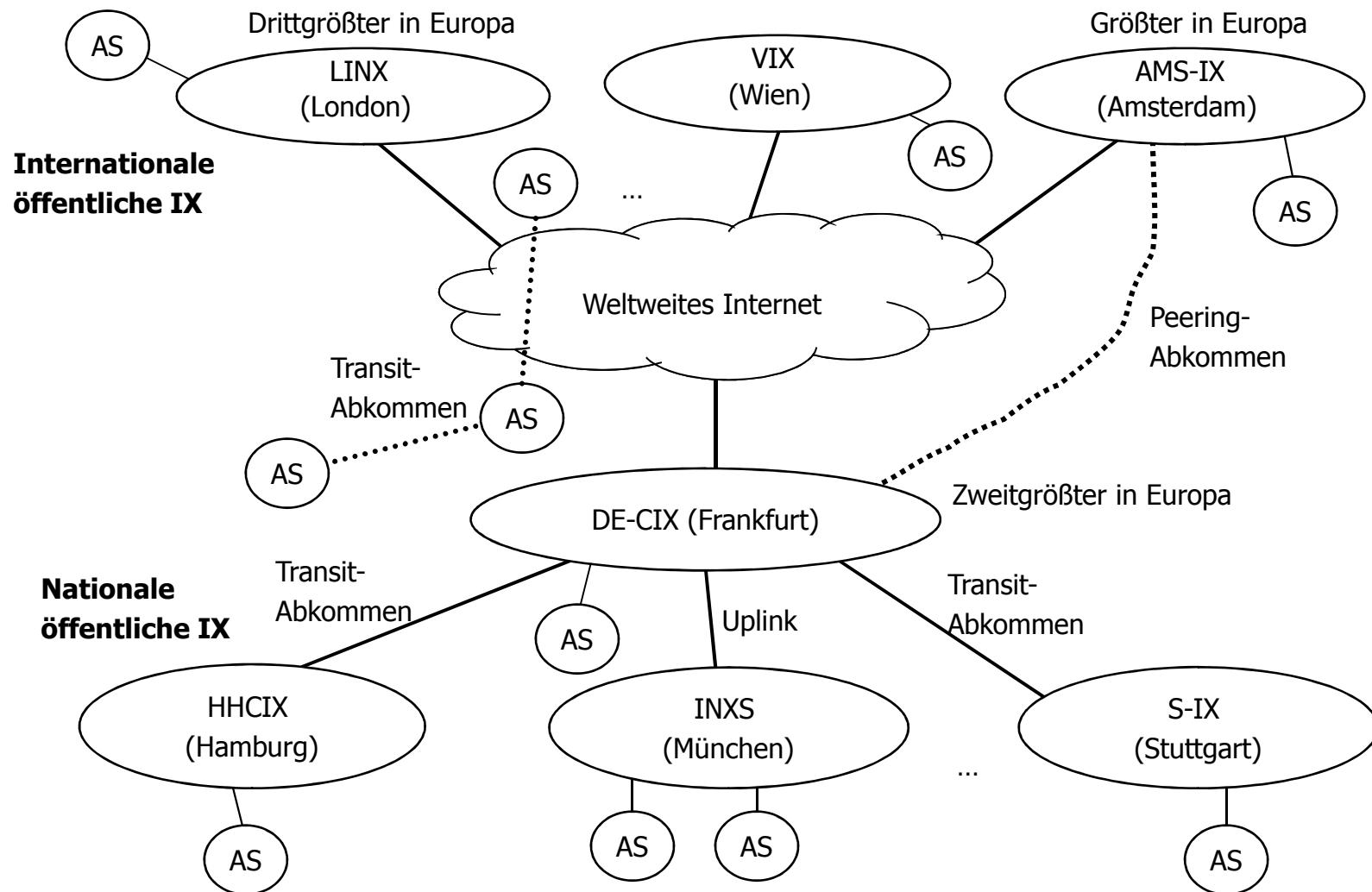
Quelle: Wikipedia

Das Internet heute, Links

- Begriffe:
 - Stub AS (sollte es nicht geben)
 - Multihomed Stub AS
 - Multihomed AS

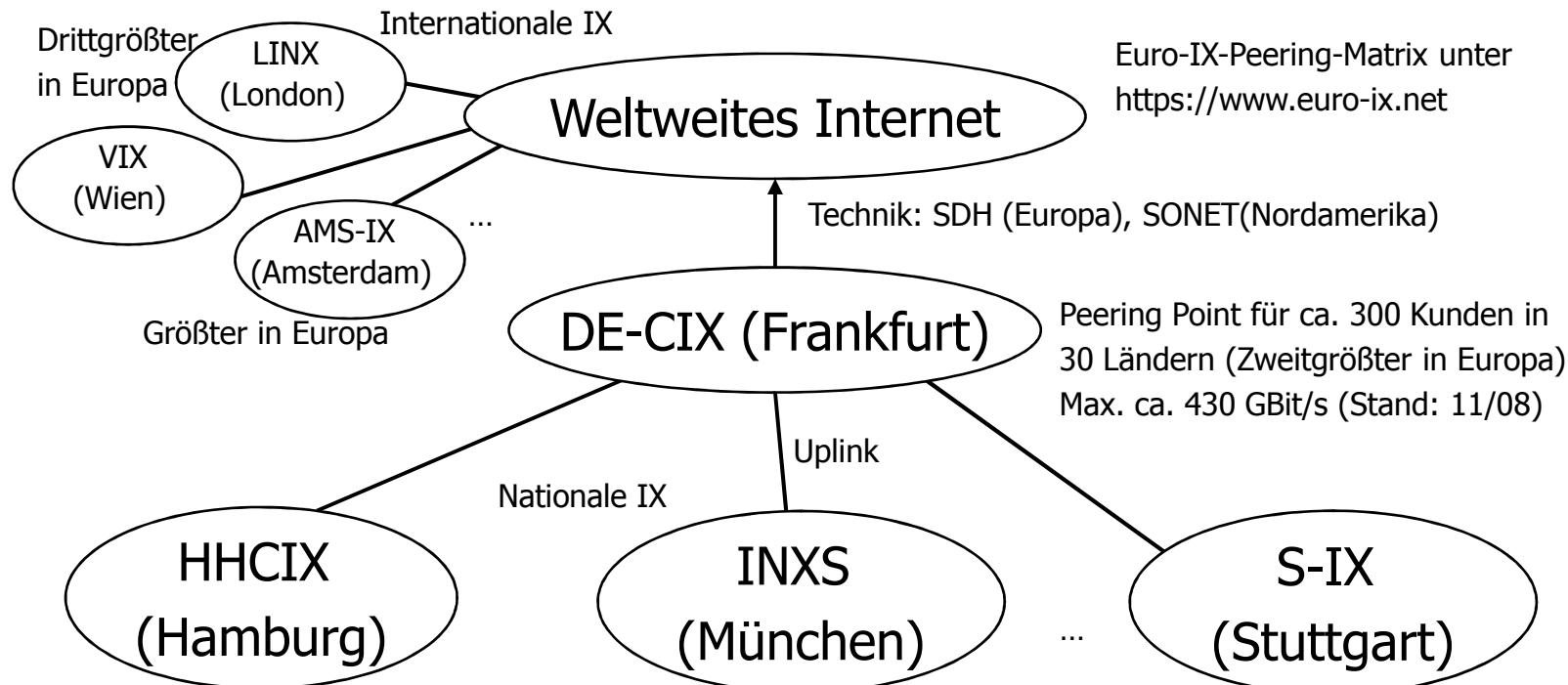


Das Internet heute, Typische Verbindungen



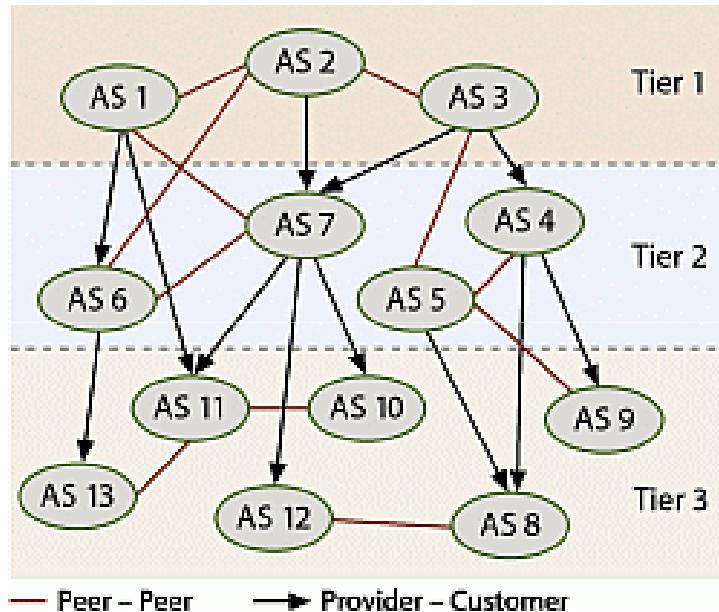
Das Internet heute, physikalische Sicht

- DE-CIX Management GmbH: Betreibt DE-CIX international Internet Exchange („IX“) in Frankfurt
- Einer der größten IX weltweit (<http://www.de-cix.de>)
- Top-80 in Europa: <http://www.alrond.com/de>

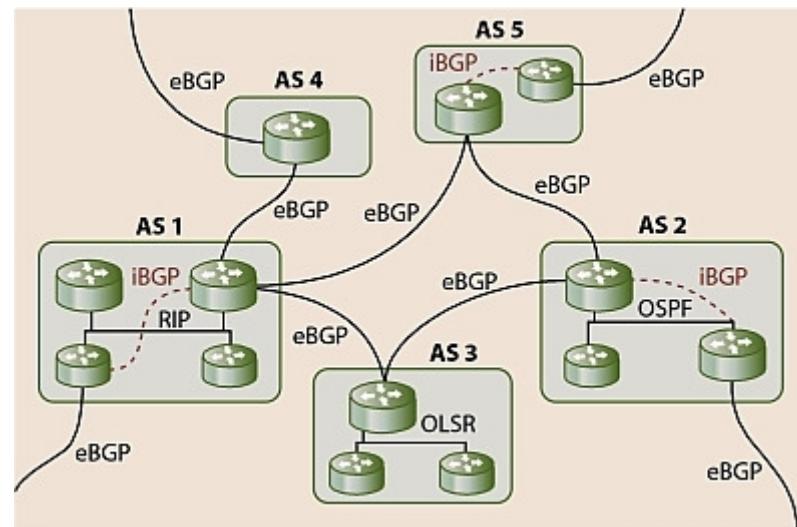


Das Internet heute, physikalische Sicht

Tier-x-Hierarchie



AS-Routing-Sicht



Quelle: <http://www.heise.de/netze/artikel/print/115741>

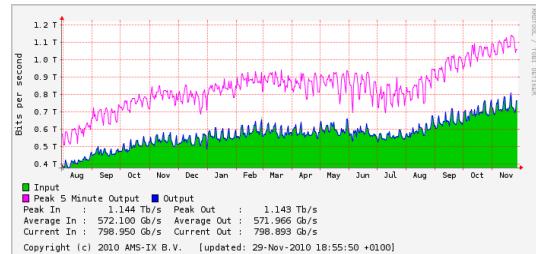
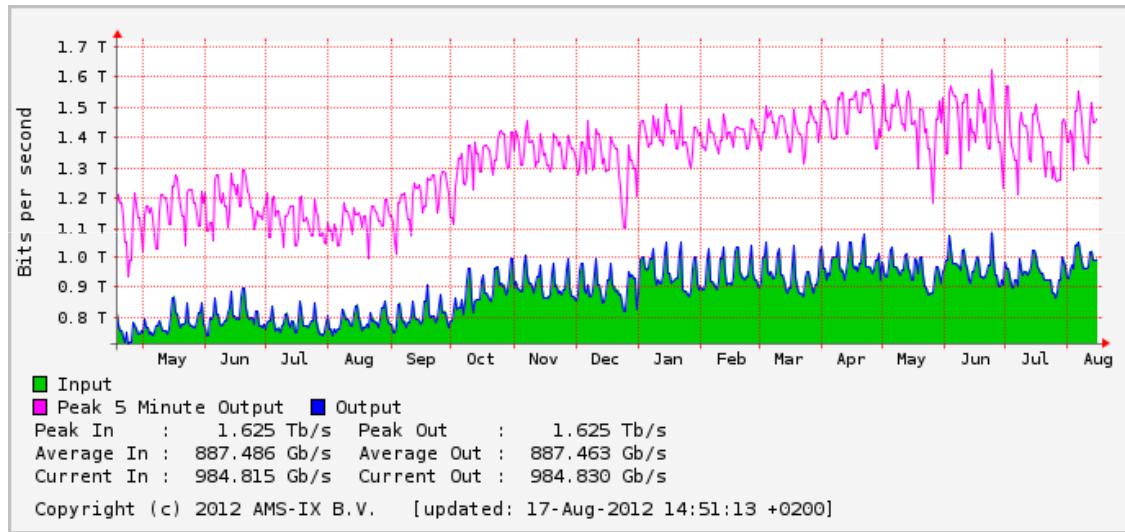
Das Internet heute, öffentliche Peering-Punkte (1)

- Top-80 in Europa: <http://www.alrond.com/de> (2008)

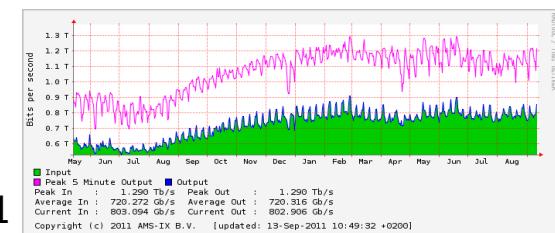
Pos.	IX	Max Gbit/s	Avg Gbit/s	Internet Exchange Name	Country
1	AMS-IX	453,04	293,17	Amsterdam Internet Exchange	 NL
2	DE-CIX	432	236,4	German Internet Exchange	 DE
3	LINX	293,05	215,69	London Internet Exchange	 GB
4	EQUINIX	233,27	181,46	EQUINIX (6 US-Points)	 US
5	JPNAP	221,99	175,87	Japan Network Access Point	 JP
6	NETNOD	126,16	78	Netnod Internet Exchange i Sverige	 SE
7	ESPANIX	103,5	83,7	España Internet Exchange	 ES
8	JPIX	101,01	64,42	Japan Internet Exchange	 JP

Das Internet heute, öffentliche Peering-Punkte (2)

- AMS-IX im August 2012
- Quelle: <http://www.ams-ix.net/>



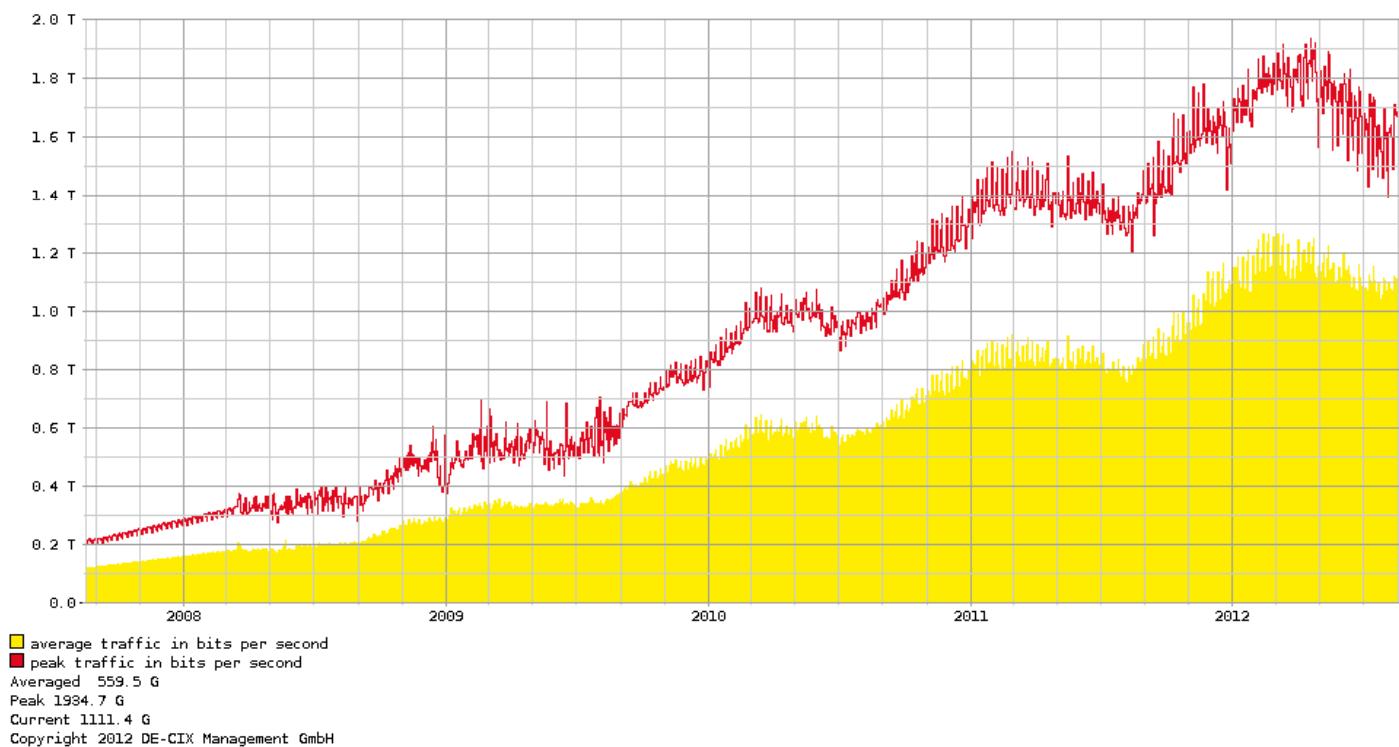
Vergleich: November 2010



Vergleich: September 2011

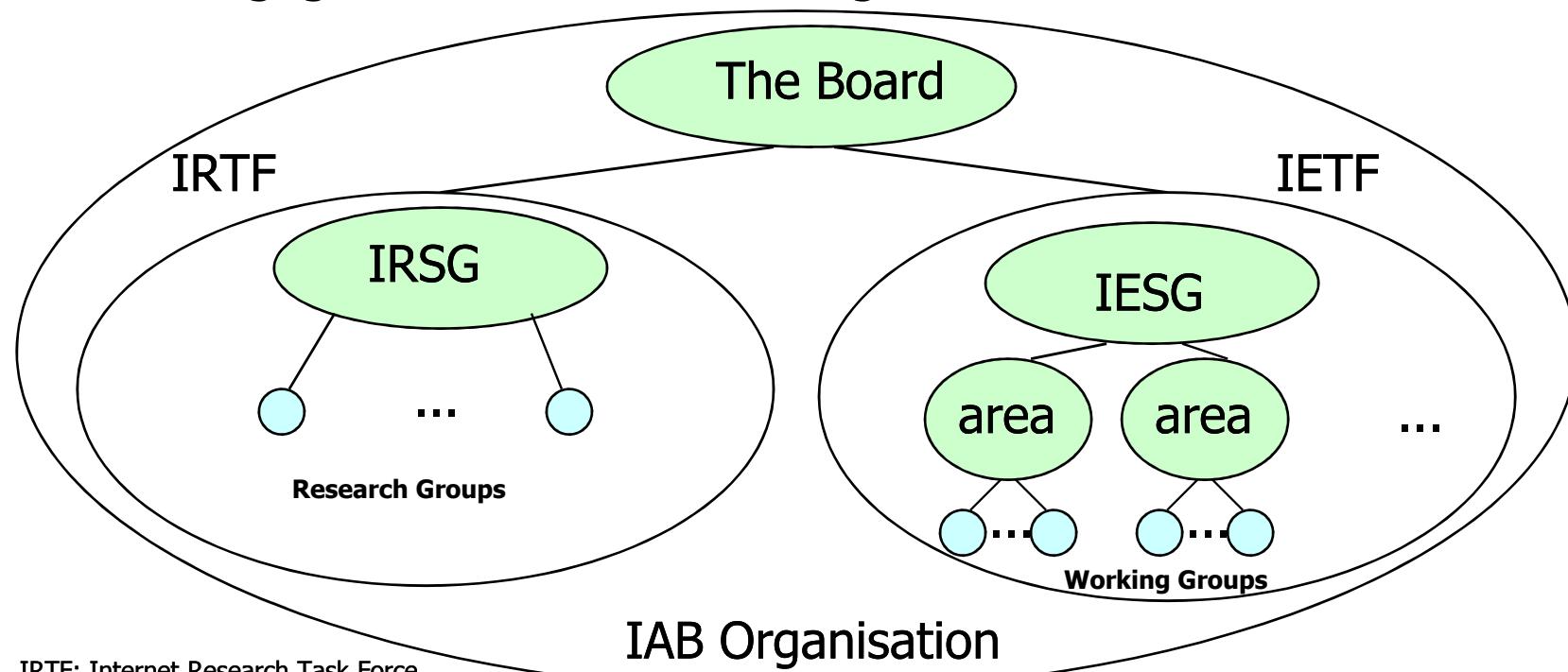
Das Internet heute, öffentliche Peering-Punkte (3)

- DE-CIX im August 2012, mittlerweile der Größte
- Quelle: <http://www.de-cix.net/>, siehe auch: Statistiken zu Internet-Knoten in Wikipedia/Internet-Knoten



Das Internet heute: IAB

- Zuständig für die Weiterentwicklung des Internet ist das IAB (Internet Activity (heute: Architecture) Board), das bereits **1983** gegründet und **1989** umorganisiert wurde.



IRTF: Internet Research Task Force
IETF: Internet Engineering Task Force
IRSG: Internet Research Steering Group
IESG: Internet Engineering Steering Group

Das Internet heute, IAB

- Das **IAB** (Board) bestimmt die Richtlinien der Politik
- Die **IETF** kümmert sich in verschiedenen Bereichen (areas) um kurz- und mittelfristige Probleme
- Die **IESG** koordiniert die IETF Working Groups
- Die **IRTF** ist ein Forschungsbereich, der die TCP/IP-Forschungsthemen koordiniert
- Die **IRSG** koordiniert die Forschungsaktivitäten der einzelnen Gruppen
- Die Working Groups setzen sich aus freiwilligen Mitarbeitern zusammen

Das Internet heute, NIC und DENIC

- Das NIC (Network Information Center, gesprochen NICK), ist zuständig für
 - die Dokumentation und
 - die Verwaltung der umfangreichen Information über
 - Protokolle,
 - Standards,
 - Services, usw.
- Das NIC verwaltet das Internet, z.B. auch die Domänennamen (www.nic.net)
- In Deutschland ist die DENIC (www.denic.de) als nationale Vertretung eingerichtet (Frankfurt)

Das Internet heute, Dokumentation, RFCs

- Die Dokumentation und die Standards werden als technische Reports gesammelt und heißen RFCs (Request for Comments)
- RFCs durchlaufen während ihrer Lebenszeit verschiedene Stati (Proposed Standard → Draft Standard → Internet Standard)
- Manche RFCs werden auch nie zum Internet Standard
- RFCs sind frei verfügbar (z.B. unter www.rfc-editor.org)

Internet Protocol: Hauptaufgaben

- **Paketvermitteltes** (datagramm-orientiertes) und verbindungsloses Protokoll der Vermittlungsschicht
- IP dient der **Beförderung von Datagrammen** von einer Quelle zu einem Ziel evtl. über verschiedene Zwischenknoten (IP-Router)
- **Routing-Unterstützung** ist eine zentrale Aufgabe von IP (Routingprotokoll basiert auf diversen Protokollen)
- Datagramme werden während des Transports zerlegt und am Ziel wieder zusammengeführt, bevor sie der Schicht 4 übergeben werden = **Fragmentierung**

Internet Protocol: Hauptaufgaben

- IP stellt einen **ungesicherten verbindungslosen** Dienst zur Verfügung
 - Es existiert **keine Garantie der Paketauslieferung**
 - Die Übertragung erfolgt nach dem **Best-Effort-Prinzip**
 - „Auslieferung nach bestem Bemühen“
 - Jedes Paket des Datenstroms wird **isoliert** behandelt
 - Das IP-Paket wird in einem Rahmen der zugrundeliegenden Schicht 2 transportiert, für den Längenrestriktionen bestehen:
 - bei Ethernet ist eine Länge von 1500 Bytes üblich
 - MTU = Maximum Transfer Unit

Überblick

1. Überblick

- Internet-Vermittlungsschicht
- Autonome Systeme (AS)
- Organisation
- IPv4: Überblick und Aufgaben

2. IPv4-Adressierung

- IPv4: Adressierung und Adressenknappheit
- IPv4-Subnetting
- VLSM und CIDR

3. IPv4-PDU

- Aufbau und Felder

4. Fragmentierung

Adressierung im Internet, IP-Adressen

- IP-Adressen sind **32 Bit** lange Adressen
 - **Tupel** aus (Netzwerknummer, Hostnummer)
- IP-Adresse von Quelle und Ziel werden **in allen** IP-Paketen mit übertragen
- Es gibt verschiedene Adressformate
 - man unterscheidet die Klassen A, B, C, D und E
- Schreibweise für IP-Adressen in gepunkteten Dezimalzahlen, jeweils ein Byte als Dezimalzahl zwischen 0 und 255
 - Beispiel:
 - Hexformat: 0xC0290614
 - Dezimalzahl: 192.41.6.20

Adressierung im Internet, IP-Adressformate

4 Byte, 32 Bit			
Klasse A	0	Netz	Host
Klasse B	10	Netz	Host
Klasse C	110	Netz	Host
Klasse D	1110	Multicast-Adresse	
Klasse E	11110	Reservierte Adresse	

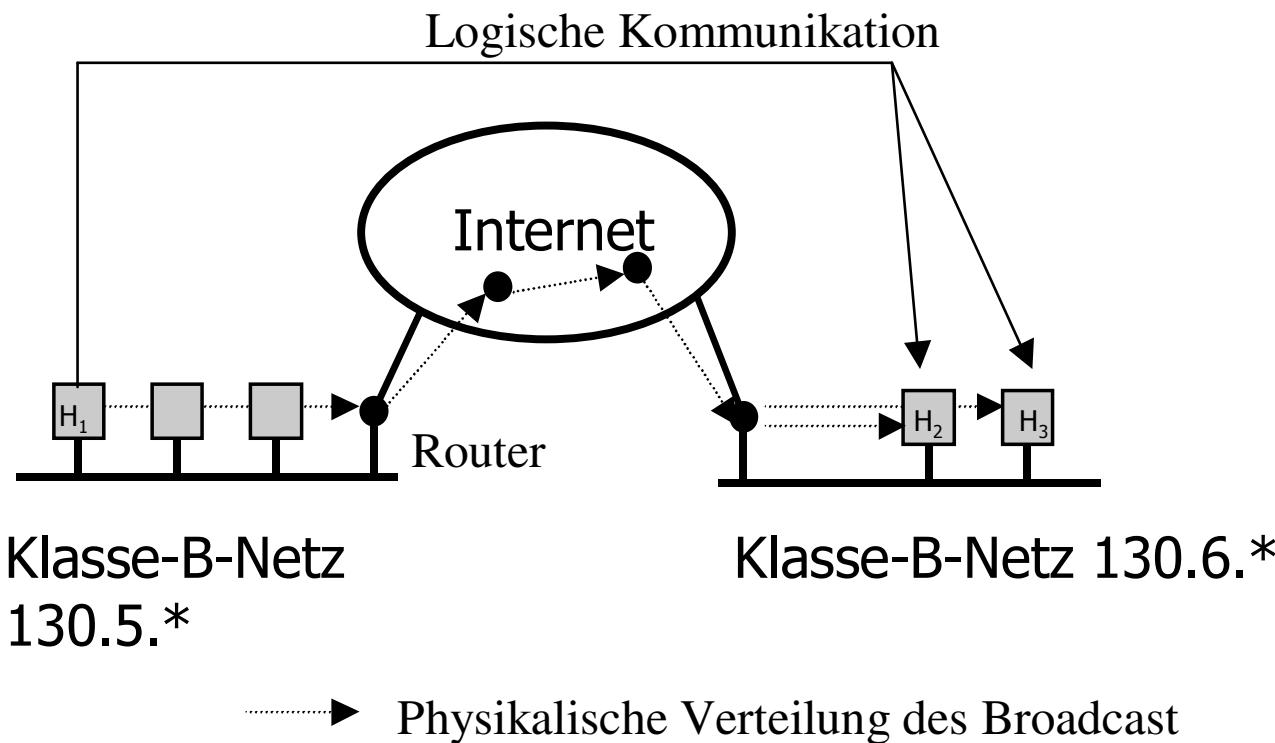
Alle Adressen der Form 127.xx.yy.zz sind Loopback-Adressen!

Adressierung im Internet, IP-Adressen

- Die niedrigste IP-Adresse ist **0.0.0.0**
- Die höchste IP-Adresse ist **255.255.255.255**
- Die Adresse **0.0.0.0** hat die besondere Bedeutung „ein bestimmtes Netz“ oder „ein bestimmter Host“
- Die Adresse **255.255.255.255** (-1) wird als Broadcast-Adresse verwendet
 - Limited Broadcast → im lokalen Netz
- Weiterer Broadcast-Typ:
 - Directed Broadcast → Broadcast an anderes Netzwerk

Direkter Broadcast

Ziel-IP-Adresse:
130.6.255.255



Adressierung im Internet

- Netznummern werden vom NIC bzw. in Deutschland von der DENIC zugewiesen!
- **Adressenknappheit** wird erwartet:
 - Neue Version **IPv6** soll hier Abhilfe schaffen

Klasse	Anzahl Netze	Max. Anzahl Hosts je Netz	Anteil am IP-Adressraum	Adressen insgesamt
A (/8)	126 (2^7)	16.777.214 ($2^{24} - 2$)	50 %	2.147.483.638 (2^{31})
B (/16)	16.384 (2^{14})	65.534 ($2^{16} - 2$)	25 %	1.073.741.824 (2^{30})
C (/24)	2.097.152 (2^{21})	254 ($2^8 - 2$)	12,5 %	536.870.912 (2^{29})

Adressierung im Internet, Subnetting

- Die Hostadresse kann zu besseren organisatorischen Gliederung noch mal zur **Subnetz**-Bildung in zwei Teile zerlegt werden:

- Teilnetznummer
- Hostnummer

Beispiel mit
Klasse B

10	Netz	Subnetz	Host
----	------	---------	------

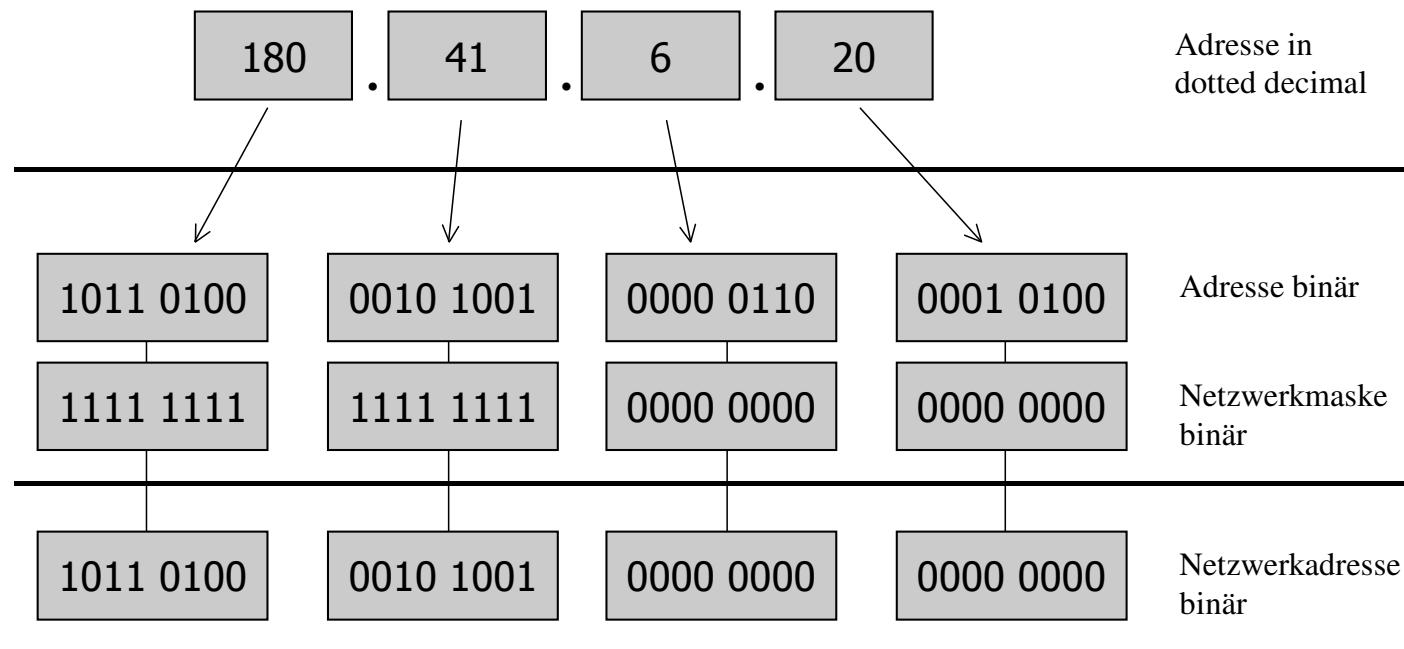
- Außerhalb des Teilnetzes ist die Aufgliederung nicht sichtbar
- IP-Router in einem Teilnetz berücksichtigen die Subnetzadresse
- Netzwerkmaske** wird als Bitmaske verwendet, um die Bits der Subnetzwerksnummer zu identifizieren

Adressierung im Internet, Subnetting

- Der lokale Administrator besitzt alle Freiheiten zur Bildung von Subnetzen, ohne die Komplexität auf den Internet-Router zu übertragen
- Bei einer Klasse B-Adresse wäre folgende Struktur denkbar:
 - 3. Byte gibt die Organisationseinheit im Unternehmen an (Subnetz-Adresse)
 - 4. Byte erlaubt die Nummerierung der Geräte: (Stationsadresse)
 - Netzkomponenten (Switch, Hub, etc.): 1 - 9
 - Arbeitsplätze: 10 - 249
 - Server: 250 - 254

Adressierung, Beispiel

- Hier handelt es sich um eine Klasse B-Adresse, warum?

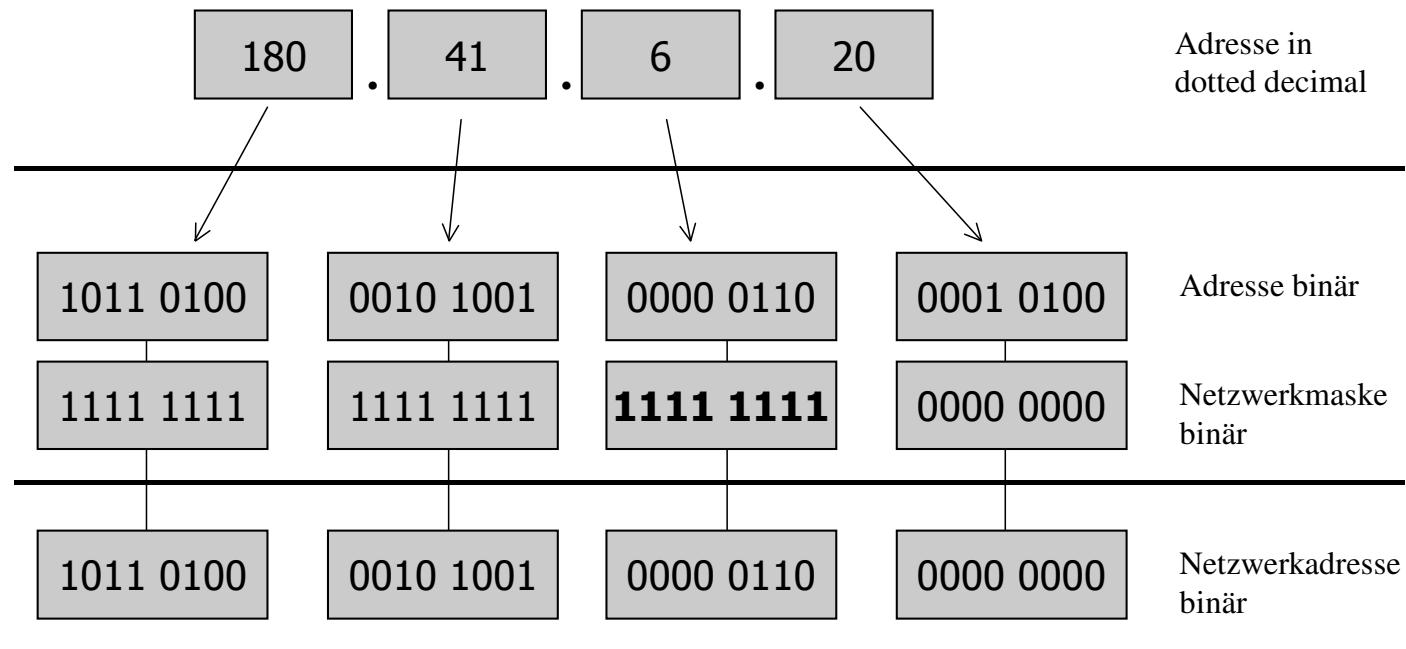


Logische Und-Verknüpfung der IP-Adresse mit der Netzmaske

Die Netzwerkadresse ist demnach: 180.41.0.0

Netzwerkmaske in dotted decimal: 255.255.0.0

Adressierung, Beispiel mit Subnetzadressierung



Subnetz mit zwei Byte für Netzwerknummer und ein Byte für Subnetzwerknummer
Die Netzmaske ist hier: 255.255.255.0

CIDR und VLSM

- Problem:
 - Vergeudung von vielen IP-Adressen durch die Aufteilung des Adressraums in Klassen (siehe Klasse-B-Adressen)
 - Aber die Routing-Tabellen der Router sind schon sehr groß und sollten nicht weiter wachsen
- VLSM und CIDR helfen, die Adressproblematik etwas abzumildern
- **VLSM** = Variable Length Subnet Mask
- **CIDR** = Classless InterDomain Routing
- CIDR ist VLSM im öffentlichen Internet!

CIDR und VLSM

- Konzept von VLSM/CIDR:
 - Restliche Klasse-C-Netze (ca. 2 Millionen) werden in Blöcken variabler Länge vergeben → Vergabe durch ISP
 - Beispiel: Braucht ein Standort 2000 Adressen erhält er acht aufeinanderfolgende Klasse-C-Netze zugewiesen, kann also auf eine B-Adresse verzichten
- Weitere Verbesserung für das Routing durch Zuordnung der Klasse-C-Adressenbereiche zu Zonen, z.B.
 - Europa: 194.0.0.0 bis 195.255.255.255
 - Nordamerika: 198.0.0.0 bis 199.255.255.255
 - Europäischer Router erkennt anhand der Zieladresse, ob ein Paket in Europa bleibt oder z.B. zu einem amerikanischen Router weitergeleitet werden soll

CIDR und VLSM: Netzwerkpräfix-Notation

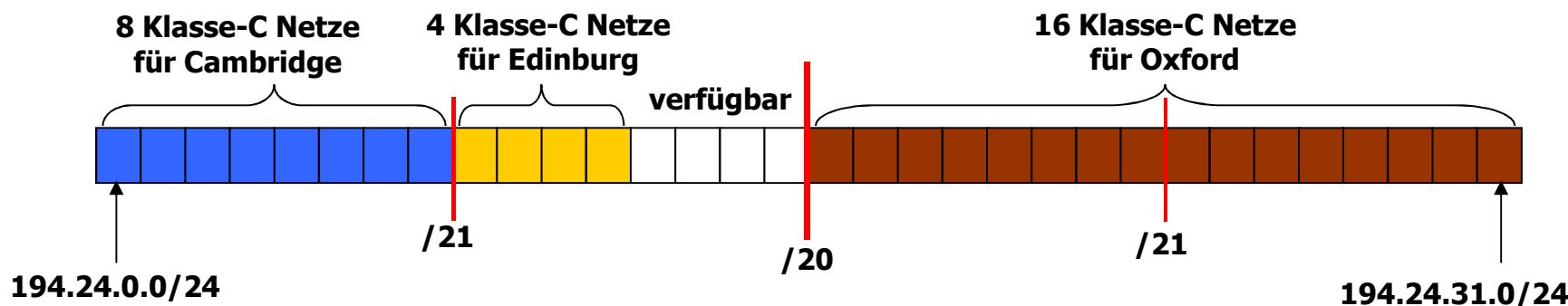
- Netzwerkpräfix-Notation (NP-Notation) ermöglicht die Angabe der Netz-Id-Bits in der IP-Adresse
- Notationsbeispiel: 194.24.19.25/20
 - IP-Adresse binär:
 - **11000010.00011000.00010011.00011001** -> Klasse C
 - Die Präfixlänge (hier 20) gibt die Anzahl der fortlaufenden Einsen in der Netzwerkmaske an:
 - Netzwerkmaske der IP-Adresse:
 - **11111111. 11111111. 11110000. 00000000** = 255.255.240.000
 - Klasse A = /8
 - Klasse B = /16
 - Klasse C = /24

CIDR und VLSM: Beispiel (vgl. Tanenbaum)

- Cambridge University benötigt 2000 (fast 2^{11}) öffentliche IP-Adressen
 - Klasse-C-Netz mit max. 256 (2^8) Adressen reicht nicht aus
 - Alternative ist ein Klasse-B Netz mit 65536 (2^{16}) Adressen
 - > 63488 öffentliche IP-Adressen werden nicht benötigt und somit verschwendet!
(über 95% der bereitgestellten Adressen!!!)
 - Besser: Nutzung mehrerer zusammenhängender Klasse-C Netze
 - Für den Hostanteil der Adresse werden zusätzlich 3 Bit benötigt (2^{11} Adressen)
 - Cambridge University wird folgender Adressbereich zugewiesen:
194.24.0.0/21 -> 194.24.0.0 bis 194.24.7.255 (Netzwerkmaske
255.255.248.0)
oder
11000010.00011000.00000000.00000000 bis
11000010.00011000.00000111.11111111 mit
11111111.11111111.11111000.00000000 als Netzwerkmaske

CIDR und VLSM: Beispiel (vgl. Tanenbaum)

- Nach dem gleichen Verfahren werden auch den Universitäten Oxford und Edinburgh mehrere Klasse-C Netze zugewiesen
 - Oxford benötigt 4000 (fast 2^{12}) öffentliche IP-Adressen
 - Edinburgh benötigt 1000 (fast 2^{10}) öffentliche IP-Adressen
- Folgende Adressbereiche werden zugewiesen:
 - Cambridge: 194.24.0.0/21 194.24.0.0 bis 194.24.7.255
 - Edinburgh: 194.24.8.0/22 194.24.8.0 bis 194.24.11.255
 - Verfügbar: 194.24.12.0/22 194.24.12.0 bis 194.24.12.255
 - Oxford: 194.24.16.0/20 194.24.16.0 bis 194.24.31.255



CIDR und VLSM: Beispiel (vgl. Tanenbaum)

- Ein Standard IP-Router kann die mittels VLSM zusammengefassten Klasse-C-Netze nicht erkennen
- Das Routing muss um CIDR erweitert werden

Routingtabelle ohne CIDR

194.24.0.0	-> Cambridge
194.24.1.0	-> Cambridge
...	
194.24.7.255	-> Cambridge
194.24.8.0	-> Edinburgh
...	
194.24.11.255	-> Edinburgh
194.24.16.0	-> Oxford
...	
194.24.31.255	-> Oxford

Routingtabelle mit CIDR

194.24.0.0/21	-> Cambridge
194.24.8.0/22	-> Edinburgh
194.24.16.0/20	-> Oxford

Nur 3 statt 28 Einträge!!!

CIDR und VLSM:

Beispiel LRZ - Hochschule München - Fakultät 07 (1)

- Das Leibnitz-Rechenzentrum (LRZ) organisiert den privaten IP-Adressbereich 10.0.0.0/8 und ordnet der Hochschule München den Adressbereich 10.28.0.0/16 zu

10.20.0.0 -> Lothstraße 34

10.21.0.0 -> Lothstraße 21

10.22.0.0 -> Karlstraße 6

10.23.0.0 -> Infanteriestraße 13/14

...

10.26.0.0 -> Pasing

...

10.28.0.0 -> Hesstraße (Informatik)

...

- Kein Zugriff von außen möglich, nur über VPN

CIDR und VLSM:

Beispiel LRZ - Hochschule München - Fakultät 07 (2)

- Die Fakultät untergliedert den Adressbereich derzeit weiter in vier /18-Subnetze
- Beispiel: Netz 0
 - 10.28.0.0/18 mit 18 Bit Netzwerkanteil und $2^{14} - 2$ Hostadressen
 - Binär: 0000 1010 0001 1100 **00**xx xxxx xxxx xxxx
- Beispiel: Netz 1
 - 10.28.64.0/18 mit 18 Bit Netzwerkanteil und $2^{14} - 2$ Hostadressen
 - Binär: 0000 1010 0001 1100 **01**xx xxxx xxxx xxxx
- Alle Netze
 - 10.28.0.0/18 → Netz 0: Wird im Münchener Hochschulnetz geroutet
 - 10.28.64.0/18 → Netz 1: Wird innerhalb der Hochschule München geroutet
 - 10.28.128/18 → Netz 2: wie Netz 1, aber DHCP-Adressvergabe
 - 10.28.192.0/18 → Netz 3: kein Routing

CIDR und VLSM, Übung

- Welche IP-Adressen repräsentiert die CIDR-Adresse 180.41.214.192/28 (Netzwerkpräfix-Notation)?

- Lösung:
 - Insgesamt 4 Bit für Hosts
 - Adressbereich von 180.41.214.192 bis 180.41.214.207
 - Also von 1011 0100 . 0010 1001 . 1101 0110 . 1100 **0000**
bis 1011 0100 . 0010 1001 . 1101 0110 . 1100 **1111**
 - Das sind 16 IP-Adressen, 14 davon sind nutzbar

CIDR und VLSM, Übung

- Ein Unternehmen besitzt die Klasse-C-Adresse 193.1.1.0 und hat 8 Abteilungen
- Jede Abteilung hat 25 Rechner
- Jede Abteilung soll einen eigenen Adressbereich erhalten
- Teilen Sie das Klasse-C-Netzwerk in 8 Subnetze auf und verwenden Sie dabei CIDR-Adressen!
 - Hinweis: Für 8 Subnetze braucht man 3 Bit

CIDR und VLSM, Übung

Ergänzen Sie die CIDR-Formate (Werte x1 – x8 und y1 – y8)!

Netz	Adresse in Bitdarstellung	CIDR-Format	Rechner
Basisnetz	11000001.00000001.00000001.00000000	193.1.1.0/24	254
Subnetz 0	11000001.00000001.00000001. 00000000	193.1.1.x1/y1	30
Subnetz 1	11000001.00000001.00000001. 00100000	193.1.1.x2/y2	30
Subnetz 2	11000001.00000001.00000001. 01000000	193.1.1.x3/y3	30
Subnetz 3	11000001.00000001.00000001. 01100000	193.1.1.x4/y4	30
Subnetz 4	11000001.00000001.00000001. 10000000	193.1.1.x5/y5	30
Subnetz 5	11000001.00000001.00000001. 10100000	193.1.1.x6/y6	30
Subnetz 6	11000001.00000001.00000001. 11000000	193.1.1.x7/y7	30
Subnetz 7	11000001.00000001.00000001. 11100000	193.1.1.x8/y8	30

CIDR und VLSM, Übung

Ergänzen Sie die CIDR-Formate (Werte x1 – x8 und y1 – y8)!

Netz	Adresse in Bitdarstellung	CIDR-Format	Rechner
Basisnetz	11000001.00000001.00000001.00000000	193.1.1.0/24	254
Subnetz 0	11000001.00000001.00000001. 00000000	193.1.1.0/27	30
Subnetz 1	11000001.00000001.00000001. 00100000	193.1.1.32/27	30
Subnetz 2	11000001.00000001.00000001. 01000000	193.1.1.64/27	30
Subnetz 3	11000001.00000001.00000001. 01100000	193.1.1.96/27	30
Subnetz 4	11000001.00000001.00000001. 10000000	193.1.1.128/27	30
Subnetz 5	11000001.00000001.00000001. 10100000	193.1.1.160/27	30
Subnetz 6	11000001.00000001.00000001. 11000000	193.1.1.192/27	30
Subnetz 7	11000001.00000001.00000001. 11100000	193.1.1.224/27	30

Überblick

1. Überblick

- Internet-Vermittlungsschicht
- Autonome Systeme (AS)
- Organisation
- IPv4: Überblick und Aufgaben

2. IPv4-Adressierung

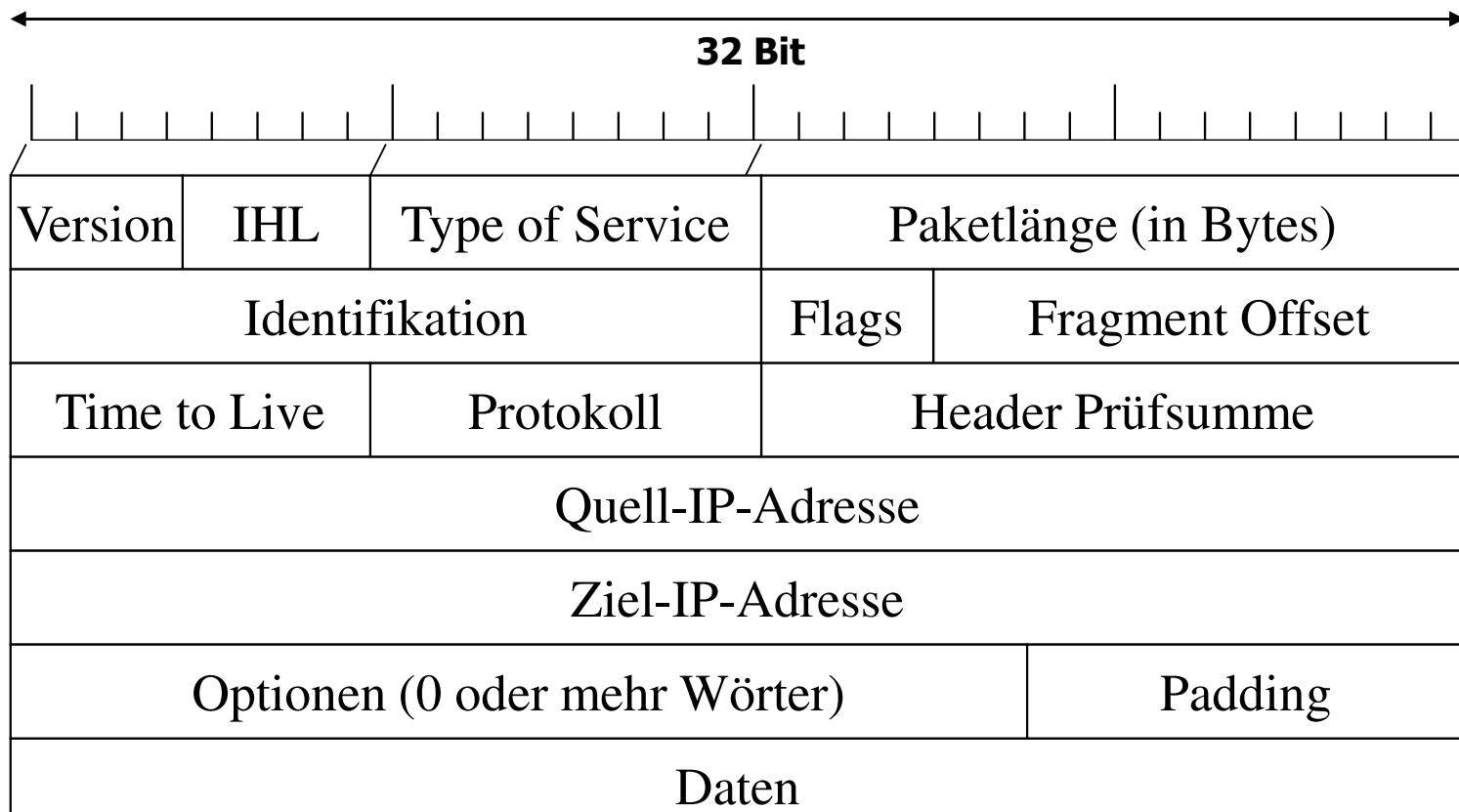
- IPv4: Adressierung und Adressenknappheit
- IPv4-Subnetting
- VLSM und CIDR

3. **IPv4-PDU**

- Aufbau und Felder

4. Fragmentierung

Protokollheader



Headerlänge: 20 – 60 Byte

Protokollheader

Version	IHL	Type of Service	Paketlänge (in Bytes)
---------	-----	-----------------	-----------------------

- **Version:** Spezifiziert die genutzte IP-Version; z.Zt. Wechsel von IPv4 auf IP Next Generation (IPv6)
- **IHL:** Gibt die Länge des Paket-Headers an, gemessen in 32-Bit-Worten; ist aufgrund der variablen Länge des Optionsfeldes nötig (mind. 5 → keine Option, max. 15 Worte → 60 Byte)
- **Type of Service:** Dieses 8-Bit-Feld ist wiederum aufgeteilt in:
 - Priorität (3 Bit): 000=Standard, 001=Priorität, ..., 101=kritisch, ...
 - ToS-Spezifikation (3 Bit): Flags zur Angabe von Servicetypen (hoher Durchsatz, niedrige Verzögerung, ...)
 - 2 unbenutzte Bit, **IPv4 nutzt Type of Service nicht**
- **Paketlänge:** Gesamtlänge des Datenpaketes inkl. Header; gemessen in 8-Bit-Worten
 - max. 65.535 Byte

Protokollheader

Identifikation	Flags	Fragment Offset
----------------	-------	-----------------

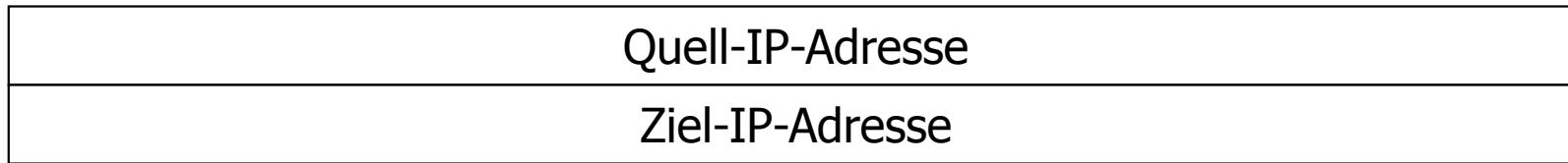
- **Identifikation:** Alle Fragmente eines Datagramms erhalten hier den gleichen Wert
- **Flags:** (3 Bit) Dient der Kontrolle der Fragmentierung
 - Geben an, ob das Feld geteilt werden muss und weitere Pakete folgen oder ob das aktuelle Paket das letzte ist
 - Drei Flags, erstes unbenutzt: 0|DF|MF
 - DF=1 → Fragmentierung ist nicht erlaubt
 - More Fragments=0 → letztes Fragment; MF=1 weitere folgen
- **Fragment Offset:** Dient der korrekten Herstellung der Ursprungssequenz, da Pakete das Ziel in unterschiedlicher Reihenfolge erreichen können, gemessen in 8-Byte-Worten
 - Dient der Ermittlung der relativen Lage des Fragments im Datagram
 - 13 Bit

Protokollheader

Time to Live	Protokoll	Header Prüfsumme
--------------	-----------	------------------

- **Time to live (TTL):** Gibt an, wie lange ein Datagramm im Internet verbleiben darf
 - Es dient dazu, zu alte Pakete vom Netz zu nehmen, bei 0 wird Paket verworfen und eine ICMP-Nachricht zum Quellhost gesendet
 - War gedacht als Zeit in Sekunden (max. 255 s)
 - Wird aber als Hop-Count genutzt, jeder Router subtrahiert 1
- **Protokoll:** Definiert das darüberliegende Protokoll, an das IP die Daten des Pakets weiterreicht (6=TCP, 89=OSPF,...)
- **Header-Prüfsumme:** Header-Absicherung
 - 16-Bit-Wörter aufsummieren und Einerkomplement bilden
 - Prüft also **nur** den Header, Daten in höheren Protokollen prüfen!
 - Wird für jede Teilstrecke neu berechnet werden, warum?

Protokollheader



- **Quell-IP-Adresse und Ziel-IP-Adresse:**
 - Jeweils 32 Bits
 - Identifikation der einzelnen Endsysteme (Hosts)
 - Hier stehen die IP-Adressen von Sender- und Empfängerhost

Protokollheader

Optionen	Padding
Daten	

- **Optionen:** Zusätzliche, optionale Angaben:
 - *Loose Source Routing* → Möglichkeit, den Weg eines Paketes durch das Internet partiell vorgeben; RFC 791
 - *Strict Source Routing* → die Pakete müssen die Pfadvorgabe einhalten (max. 9 Knoten in der Route), RFC 791
 - *Record Routing* → Jeder durchlaufene Router trägt seine Adresse ein, ...
 - Wird selten verwendet!!
- **Padding:** Wenn eine Option genutzt wird, ist das Datagramm bis zur nächsten 32-Bit-Grenze mit Nullen aufzufüllen
- **Daten:** Die Nutzdaten der höheren Schicht

Überblick

1. Überblick

- Internet-Vermittlungsschicht
- Autonome Systeme (AS)
- Organisation
- IPv4: Überblick und Aufgaben

2. IPv4-Adressierung

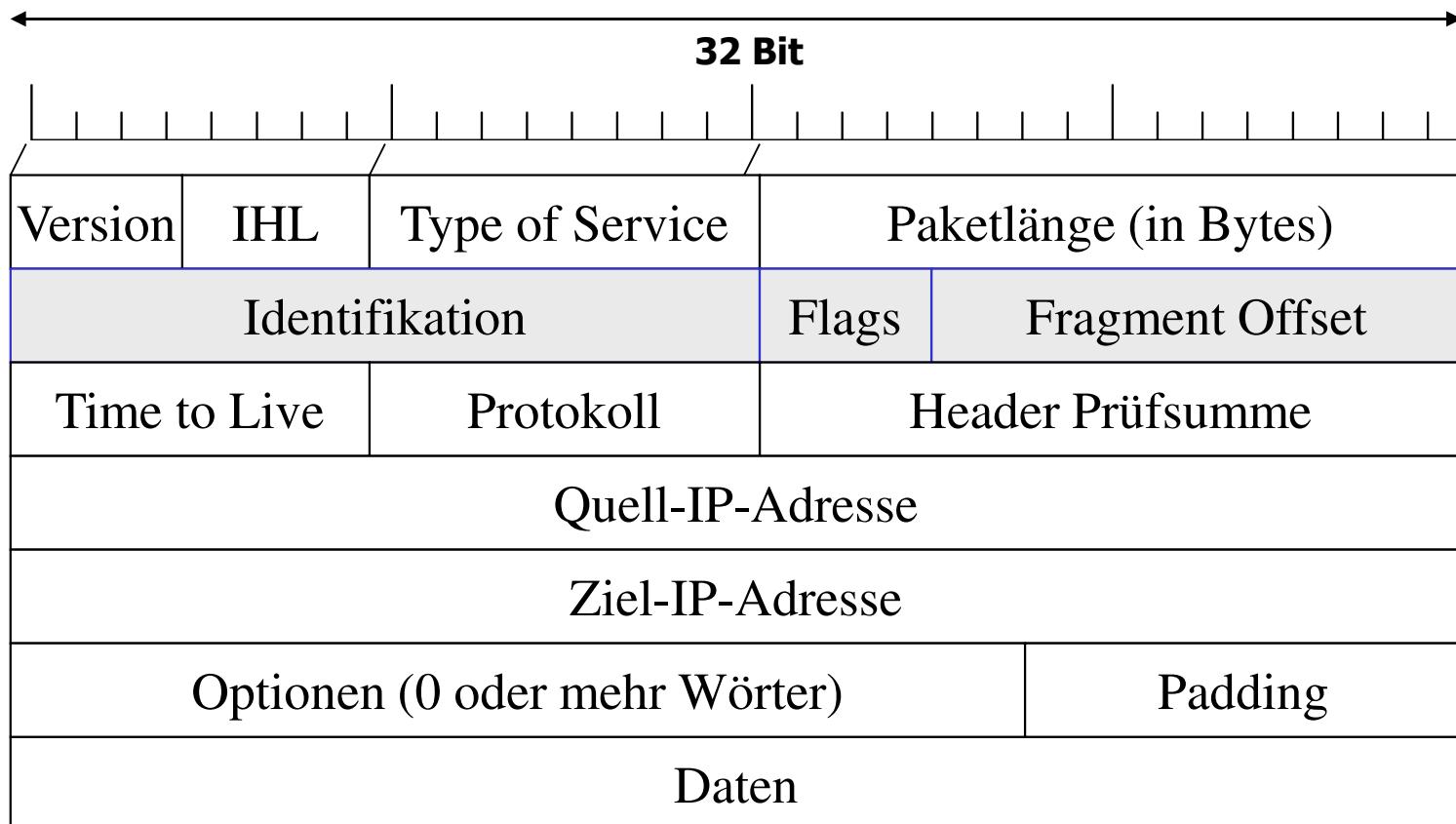
- IPv4: Adressierung und Adressenknappheit
- IPv4-Subnetting
- VLSM und CIDR

3. IPv4-PDU

- Aufbau und Felder

4. Fragmentierung

Protokollheader



Fragmentierung, Ablauf

- Wenn ein IP-Paket von einem Netzknoten zum anderen weitergeleitet wird, muss es
 - evtl. verschiedene physikalische Netze durchqueren
 - in unterschiedlich zulässige Paketgrößen aufgeteilt werden
- Daher besteht die Notwendigkeit, IP-Datagramme zu zerlegen und am Ziel wieder zusammenzusetzen
 - Fragmentierung und Defragmentierung
 - Alle Router müssen Fragmente der Größe **576 Byte** oder kleiner akzeptieren
- Sobald eine Fragmentierung einsetzt, laufen in einem Knoten mehrere Schritte ab

Fragmentierung, Ablauf

- Das **DF**-Flag wird überprüft, um festzustellen, ob eine Fragmentierung erlaubt ist. Ist das Bit auf „1“ gesetzt und es wäre eine Fragmentierung notwendig, wird das Paket verworfen
- Ansonsten wird entspr. der zulässigen Paketgröße das Datenfeld des Ur-Paketes in mehrere Teil zerlegt
- Alle neu entstandenen Pakete weisen - mit Ausnahme des letzten Paketes - eine Länge mit einem **Vielfachen von 8 Byte** auf
- Alle Datenteile werden in neu erzeugte IP-Pakete eingebettet. Die Header dieser Pakete sind Kopien des Ursprungskopfes mit einigen Modifikationen

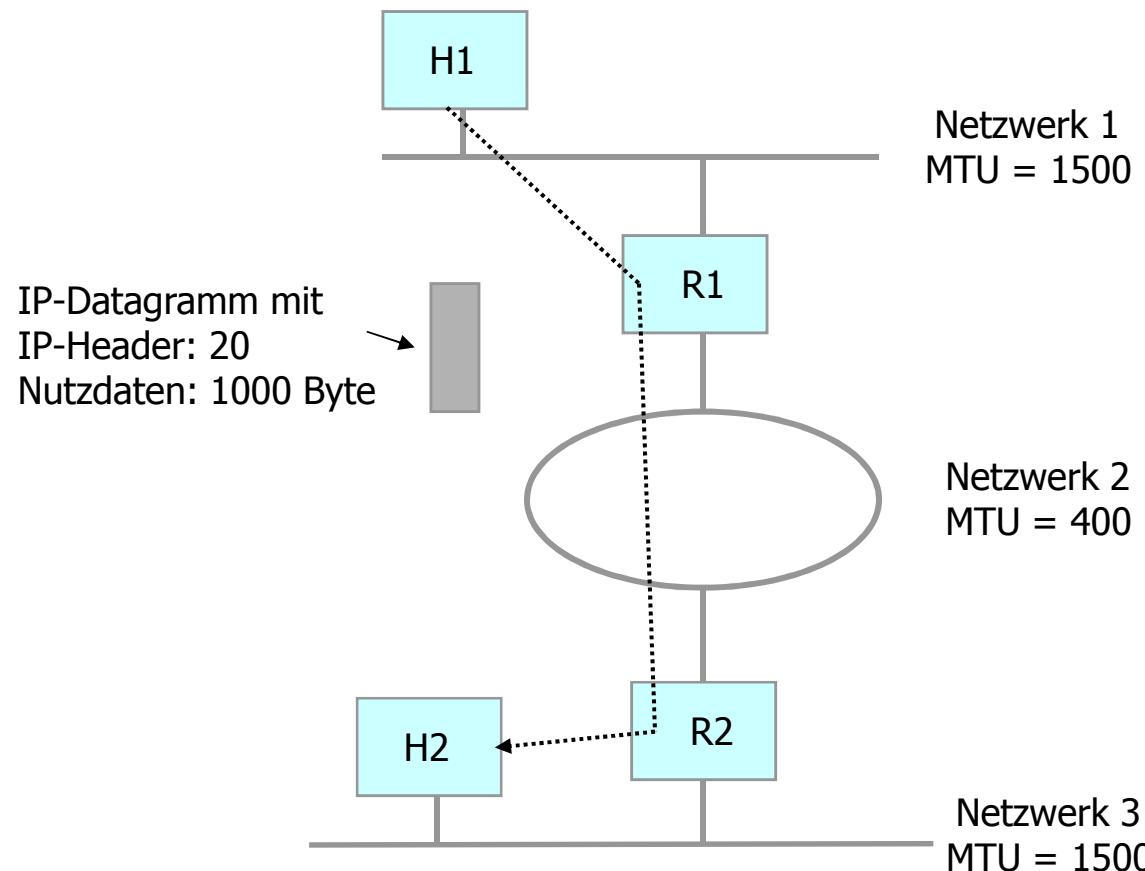
Fragmentierung, Ablauf

- Header-Modifikation bei der Fragmentierung:
 - Das **MF**-Flag wird in allen Fragmenten mit Ausnahme des letzten auf „1“ gesetzt
 - Das **Fragment-Offset**-Feld enthält Angaben darüber, wo das Datenfeld in Relation zum Beginn des nicht fragmentierten Ur-Paketes platziert ist
 - Enthält das Ur-Paket Optionen, wird abhängig vom Type-Byte entschieden, ob die Option in jedes Paketfragment aufgenommen wird (z.B. Protokollierung der Route)
 - Die **Headerlänge** (IHL) und die Paketlänge sind jeweils **neu** zu bestimmen
 - Die **Headerprüfsumme** wird neu berechnet

Fragmentierung, Ablauf

- Ablauf der Defragmentierung:
 - Die Zielstation setzt die Fragmente eines Datagramms wieder zusammen
 - Die **Zusammengehörigkeit** entnimmt sie dem **Identifikationsfeld**
 - Die ankommenden Fragmente werden zunächst gepuffert
 - Bei Eintreffen des ersten Fragments wird ein **Timer** gestartet
 - Ist der **Timer** abgelaufen bevor alle Fragmente eingetroffen sind, wird alles **verworfen**
 - Im anderen Fall wird das Datagramm am N-SAP **zur Transportschicht hochgereicht**

Fragmentierung, Übung



MTU = Maximum Transfer Unit

Fragmentierung, Übung

Fragment 1

Rest des Headers		
Identifikation=120	Flags:MF = _____	FO = _____
Datenbyte 0 ... _____		

Fragment 2

Rest des Headers		
Identifikation=_____	Flags:MF = _____	FO = _____
Datenbyte _____ ... _____		

Fragment 3

Rest des Headers		
Identifikation=_____	Flags:MF = _____	FO = _____
Datenbyte _____ ... _____		

Fragmentierung, Übung

Fragment 1

Rest des Headers		
Identifikation=120	Flags: MF = 1	FO = 0
Datenbyte 0 ... 375		

Fragmentierung, Übung

Fragment 1

Rest des Headers		
Identifikation=120	Flags:MF = 1	FO = 0
Datenbyte 0 ... 375		

Fragment 2

$$376 / 8 = 47$$

Rest des Headers		
Identifikation=120	Flags:MF = 1	FO = 47
Datenbyte 376 ... 751		

Fragmentierung, Übung

Fragment 1

Rest des Headers		
Identifikation=120	Flags:MF = 1	FO = 0
Datenbyte 0 ... 375		

Fragment 2

$$376 / 8 = 47$$

Rest des Headers		
Identifikation=120	Flags:MF = 1	FO = 47
Datenbyte 376 ... 751		

Fragment 3

$$752 / 8 = 94$$

Rest des Headers		
Identifikation=120	Flags:MF = 0	FO = 94
Datenbyte 752 ... 999		

Rückblick

1. Überblick

- Internet-Vermittlungsschicht
- Autonome Systeme (AS)
- Organisation
- IPv4: Überblick und Aufgaben

2. IPv4-Adressierung

- IPv4: Adressierung und Adressenknappheit
- IPv4-Subnetting
- VLSM und CIDR

3. IPv4-PDU

- Aufbau und Felder

4. Fragmentierung

Datenkommunikation

Vermittlungsschicht Routing im Internet

Wintersemester 2012/2013

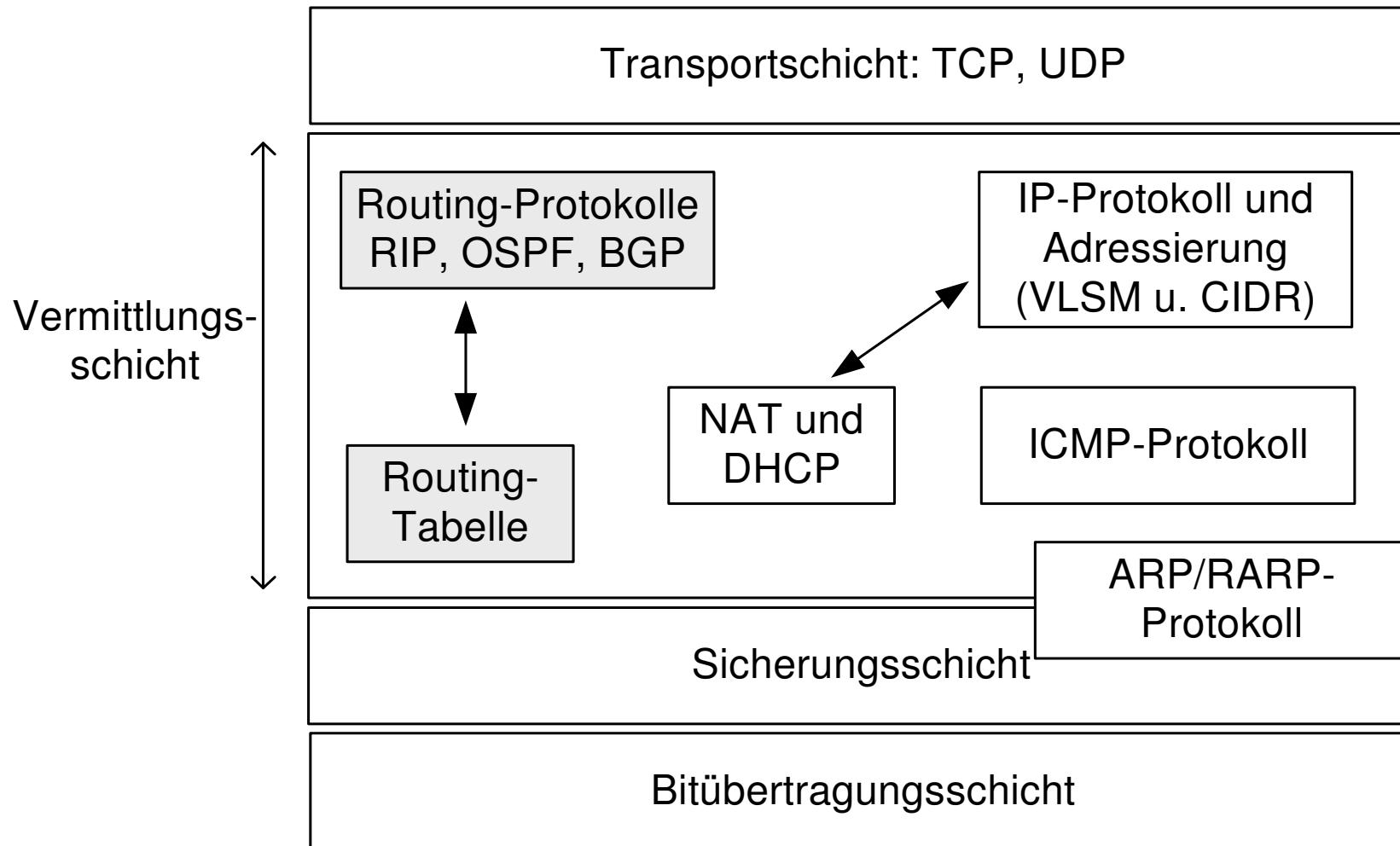
Überblick

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

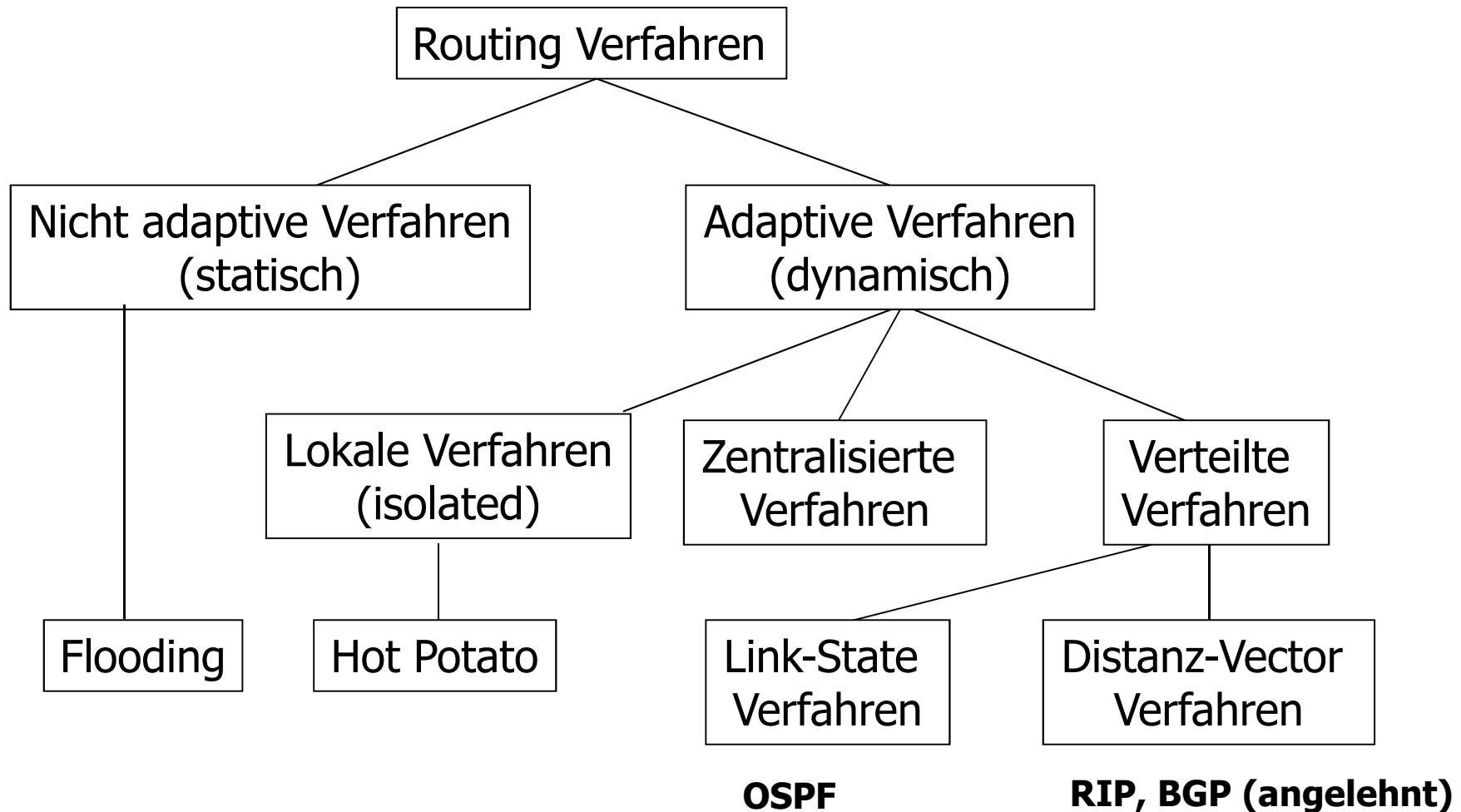
Überblick

- 1. Überblick, Routing-Tabellen**
2. IGP und EGP: Überblick
3. RIP-1 und RIP-2
4. OSPF und OSPFv2
5. BGP

Überblick: Die Internet-Vermittlungsschicht



Überblick: Routing – Einordnung der Verfahren



Internet Protocol: Routing-Tabellen

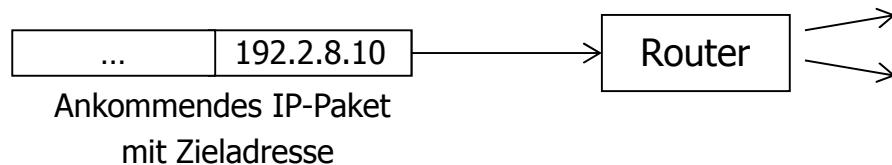
- Jeder IP-Router verwaltet eine Routing-Tabelle

Netzwerkziel	Netzwerkmaske	Nächster Router	Ausgangsport	Metrik
...

- Ausgangsport = die dem Interface zugeordnete IP-Adresse
- Metrik = Anzahl der Hops zum Ziel
- Hinweis:
 - Die Netzwerkmaske ist erst seit der Einführung von CIDR notwendig, vorher hat man aus den ersten drei Bits der Zieladresse die Netzwerkklassse ermittelt

Internet Protocol: Routenbestimmung: Regelwerk

- IP-Paket kommt am Router an. Was passiert?
 - Zieladresse des Pakets wird mit Einträgen in den Routing-Tabellen verglichen
 - Bitweise Und-Verknüpfung zwischen Zieladresse aus IP-Paket und Netzwerkmaske aus Routeneintrag (für alle Einträge)
 - Vergleich des Ergebnisses mit Netzwerkziel aus Routeneintrag
 - Übereinstimmung → potenzielle Route gefunden!
 - Die Route mit der größten Übereinstimmung (Bits von links nach rechts) wird ausgewählt
 - Bei gleichwertigen Einträgen: Beste Metrik entscheidet!
 - Keine Übereinstimmung → Standardroute



Internet Protocol: Routing-Tabellen: Beispiel IPv4-Routing-Tabelle

> netstat -r (oder route print unter Windows)

Aktive Routen:

Netzwerkziel	Netzwerkmaske	Gateway	Schnittstelle	Metrik
0.0.0.0	0.0.0.0	10.28.1.253	10.28.16.21	20
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
10.28.16.21	255.255.255.255	127.0.0.1	127.0.0.1	20
224.0.0.0	240.0.0.0	10.28.16.21	10.28.16.21	20
255.255.255.255	255.255.255.255	10.28.16.21	10.28.16.21	1

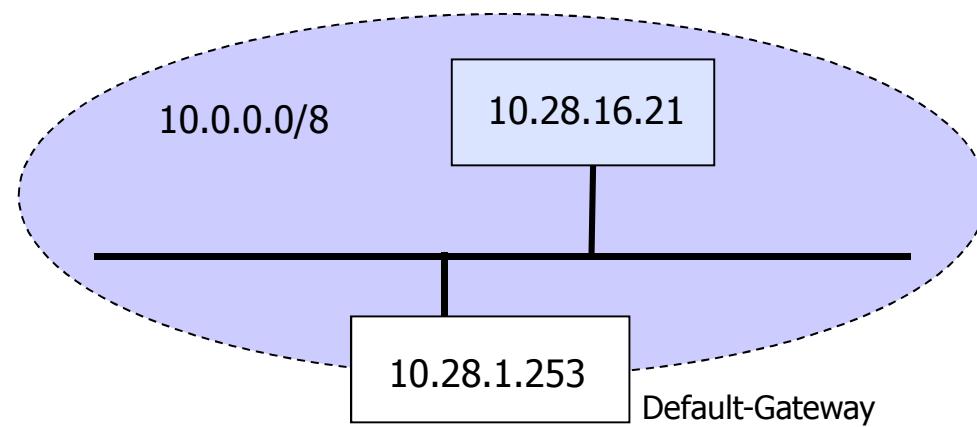
Standardgateway: 10.28.1.253

Annahmen:

- Eigene IP-Adresse: 10.28.16.21
- Nur eine Ethernet-Karte im Rechner
Standard-Gateway: 10.28.1.253

Anmerkungen:

- „Gateway“ entspricht „Nächster Router“
- „Schnittstelle“ entspricht „Ausgangsport“



Internet Protocol: Routing-Tabellen: Beispiel - Interpretation

> **netstat -r**

Aktive Routen:

Netzwerkziel	Netzwerkmaske	Gateway	Schnittstelle	Metrik
0.0.0.0	0.0.0.0	10.28.1.253	10.28.16.21	20

- Dies ist die **Standardroute**: Immer Netzwerkziel 0.0.0.0 und Netzwerkmaske 0.0.0.0 (/0)
- Jede IPv4-Zieladresse, für die eine bitweise logische UND-Operation mit 0.0.0.0 ausgeführt wird, führt zu dem Ergebnis 0.0.0.0
- Die Standardroute führt daher zu einer Übereinstimmung mit jeder IPv4-Zieladresse
- Wenn die Standardroute die längste übereinstimmende Route ist, lautet die Adresse des nächsten Knotens 10.28.1.253 (Standard-Gateway) und die Schnittstelle für den nächsten Knoten ist der Netzwerkadapter mit der IPv4-Adresse 10.28.16.21 (einiger LAN-Adapter)

Internet Protocol: Routing-Tabellen: Beispiel - Interpretation

> **netstat -r**

Aktive Routen:

Netzwerkziel	Netzwerkmaske	Gateway	Schnittstelle	Metrik
0.0.0.0	0.0.0.0	10.28.1.253	10.28.16.21	20
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1

- **Loopback-Route:** Netzwerkziel 127.0.0.0 und der Netzwerkmaske 255.0.0.0 (/8)
- Für alle Pakete, die an Adressen in der Form 127.x.y.z gesendet werden, wird die Adresse des nächsten Knotens auf 127.0.0.1 (die Loopback-Adresse) gesetzt
- Die Schnittstelle für den nächsten Knoten ist die Schnittstelle mit der Adresse 127.0.0.1 (die IP-Loopback-Schnittstelle)
- Das Paket wird nicht in das Netzwerk gesendet

Internet Protocol: Routing-Tabellen: Beispiel - Interpretation

> netstat -r

Aktive Routen:

Netzwerkziel	Netzwerkmaske	Gateway	Schnittstelle	Metrik
0.0.0.0	0.0.0.0	10.28.1.253	10.28.16.21	20
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
10.28.16.21	255.255.255.255	127.0.0.1	127.0.0.1	20

- **Hostroute:** Netzwerkziel 10.28.16.21 und Netzwerkmaske 255.255.255.255 (/32) für die IPv4-Adresse des Hosts
- Für alle (von einer lokalen Anwendung) an die Adresse 10.28.16.21 (eigene IP-Adresse) gesendeten IPv4-Pakete wird die Adresse des nächsten Knotens auf 127.0.0.1 gesetzt
- Die Schnittstelle für den nächsten Knoten ist also die Loopback-Schnittstelle
- Das Paket wird nicht in das Netzwerk gesendet

Internet Protocol: Routing-Tabellen: Beispiel - Interpretation

> netstat -r

Aktive Routen:

Netzwerkziel	Netzwerkmaske	Gateway	Schnittstelle	Metrik
...				
224.0.0.0	240.0.0.0	10.28.16.21	10.28.16.21	20
255.255.255.255	255.255.255.255	10.28.16.21	10.28.16.21	1

- Der Eintrag mit dem Netzwerkziel 224.0.0.0 und der Netzwerkmaske 240.0.0.0 (/4) ist eine Route für **Multicast-Verkehr**, der von diesem Host gesendet wird
- Für alle Multicast-Pakete wird die Adresse des nächsten Knotens auf die Zieladresse gesetzt und für die Schnittstelle des nächsten Knotens wird der LAN-Adapter festgelegt
- Der Eintrag mit dem Netzwerkziel 255.255.255.255 und der Netzwerkmaske 255.255.255.255 (/32) ist eine Hostroute, die der **limited Broadcast-Adresse** entspricht
- Für alle an 255.255.255.255 gesendeten IPv4-Pakete wird die Adresse des nächsten Knotens auf 255.255.255.255 gesetzt und die Schnittstelle des nächsten Knotens ist der LAN-Adapter

Internet Protocol: Routing-Tabellen: Beispiel 2 – iSYS-Netz

route print

Schnittstellenliste

0x1MS TCP Loopback interface
0x2 ...00 15 f2 16 ee 5a VIA-kompatibler Fast Ethernet-Adapter - Paketplaner-Miniport

Aktive Routen:

Netzwerkziel	Netzwerkmaske	Gateway	Schnittstelle	Anzahl
0.0.0.0	0.0.0.0	192.168.2.1	192.168.2.116	20
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
192.168.2.0	255.255.255.0	192.168.2.116	192.168.2.116	20
192.168.2.116	255.255.255.255	127.0.0.1	127.0.0.1	20
192.168.2.255	255.255.255.255	192.168.2.116	192.168.2.116	20
224.0.0.0	240.0.0.0	192.168.2.116	192.168.2.116	20
255.255.255.255	255.255.255.255	192.168.2.116	192.168.2.116	1

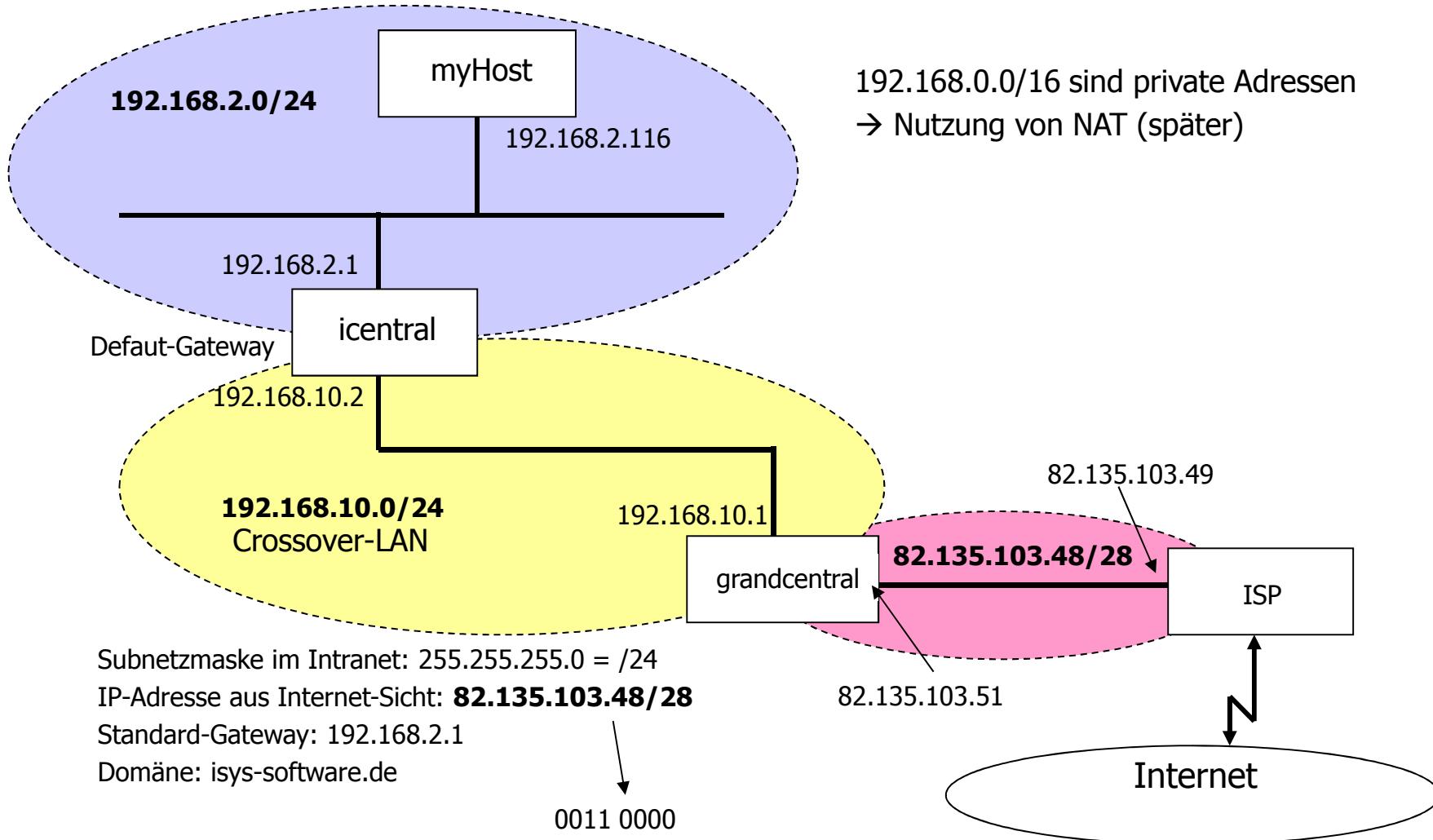
Standardgateway: **192.168.2.1**

Ständige Routen:

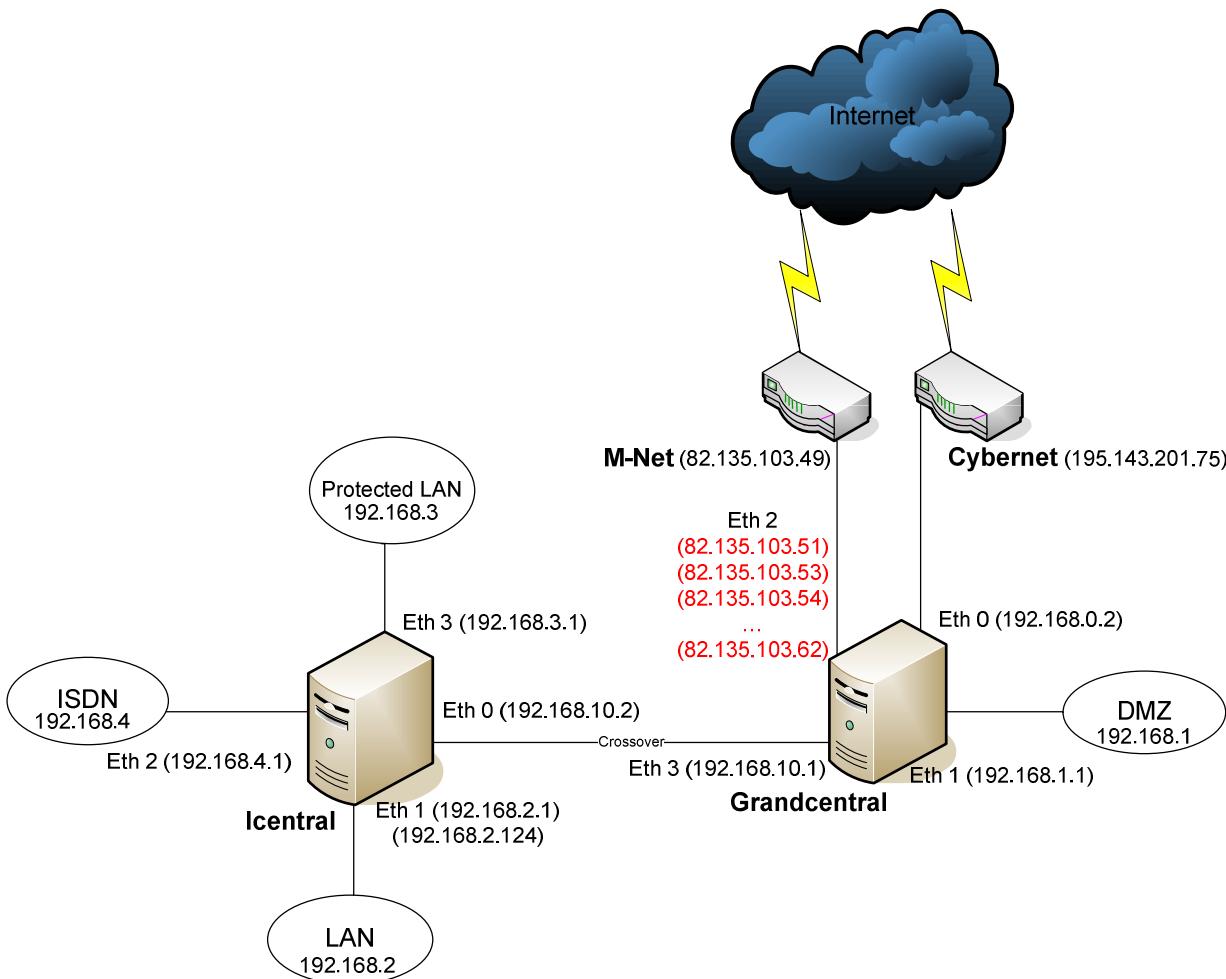
Keine

- Eigener Host, an dem Kommando abgesetzt wurde: 192.168.2.116
- Subnetzmaseke: 255.255.255.0
- Eine Ethernet-LAN-Karte

Internet Protocol: Routing-Tabellen: Beispiel 2 – iSYS-Netz



Internet Protocol: Routing-Tabellen: Beispiel 2 – iSYS-Netz



Internet Protocol: Routing-Tabellen: Beispiel 2 - Routen tracen

Kommandos tracert oder pathping unter Windows

C:\Dokumente und Einstellungen\mandl> **tracert -w 30 www.fh-muenchen.de**

Routenverfolgung zu rz.fh-muenchen.de [129.187.244.212] über maximal 30

Abschnitte:

```
1  <1 ms  <1 ms  <1 ms  icentral.isys-software.de [192.168.2.1]
2  <1 ms  <1 ms  <1 ms  grandcentral.isys-software.de [192.168.10.1]
3  59 ms  60 ms  60 ms  sub87-230-127-254.he-dsl.de [87.230.127.254]
4  64 ms  60 ms  *       ge-4-0-3-100.jc-blue.cgn.hosteurope.de [80.237.129.73]
5  65 ms  61 ms  60 ms  koln-s1-rou-1071.DE.eurorings.net [134.222.99.85]
6  63 ms  63 ms  63 ms  koln-s1-rou-1072.DE.eurorings.net [134.222.227.14]
7  63 ms  72 ms  *       ffm-s1-rou-1021.DE.eurorings.net [134.222.227.17]
8  63 ms  72 ms  63 ms  ffm-s2-rou-1071.DE.eurorings.net [134.222.227.146]
9  64 ms  81 ms  62 ms  ir-frankfurt2.g-win.dfn.de [80.81.192.222]
10  *      71 ms  72 ms  xr-gar1-te2-2.x-win.dfn.de [188.1.145.54]
11  71 ms  78 ms  72 ms  kr-lrz.x-win.dfn.de [188.1.37.90]
12  78 ms  70 ms  70 ms  129.187.1.226
13  *      92 ms  71 ms  rz.fh-muenchen.de [129.187.244.212]
```

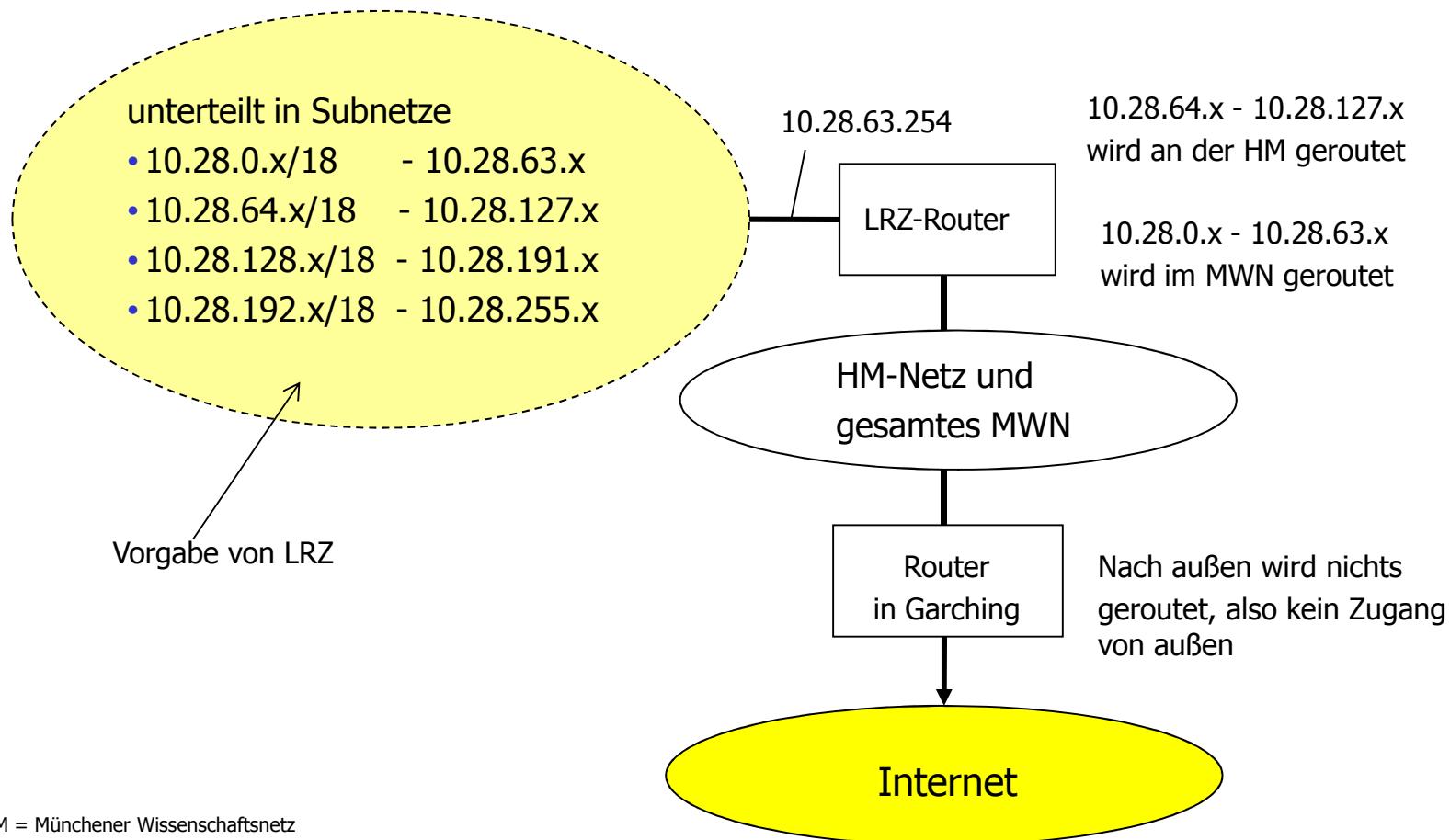
Ablaufverfolgung beendet.

Internet Protocol: Routing-Tabellen: Beispiel 2 – iSYS-Netz

- Globale IP-Adresse von iSYS in Präfixnotation: 82.135.103.48/28 (nicht mehr ganz aktuell, nur ein Beispiel)
 - 82.135.103.48 = 0011 0000 (nicht nutzbar)
 - 82.135.103.49 = 0011 0001
 - 82.135.103.50 = 0011 0010
 - 82.135.103.51 = 0011 0011
 - 82.135.103.52 = 0011 0100 (www.isys-software.de, Webserver, siehe nslookup)
 - 82.135.103.53 = 0011 0101 (imap.isys-software.de, E-Mail-Server, siehe nslookup)
 - 82.135.103.54 = 0011 0110
 - 82.135.103.55 = 0011 0111
 - 82.135.103.56 = 0011 1000
 - 82.135.103.57 = 0011 1001
 - 82.135.103.58 = 0011 1010
 - 82.135.103.59 = 0011 1011
 - 82.135.103.60 = 0011 1100
 - 82.135.103.61 = 0011 1101
 - 82.135.103.62 = 0011 1110
 - 82.135.103.63 = 0011 1111 (nicht nutzbar)
- 14 verfügbare Adressen!!

Internet Protocol: Routing-Tabellen: Beispiel 3 – HM - Fakultät-07-Netz

Fakultätsnetz mit IP-Adressraum **10.28.0.0/16**



MWN = Münchener Wissenschaftsnetz
LRZ = Leibnitz Rechenzentrum

Internet Protocol: Routenbestimmung: Regelwerk – Übung (1)

- Bei IP-Router R1 kommt von R5 aus dem Netzwerk 128.10.0.0/16 ein IP-Paket mit der Zieladresse 193.1.1.200 an
- Welche Route wählt R1 mit folgender Routing-Tabelle?

Routing-Tabelle R1:

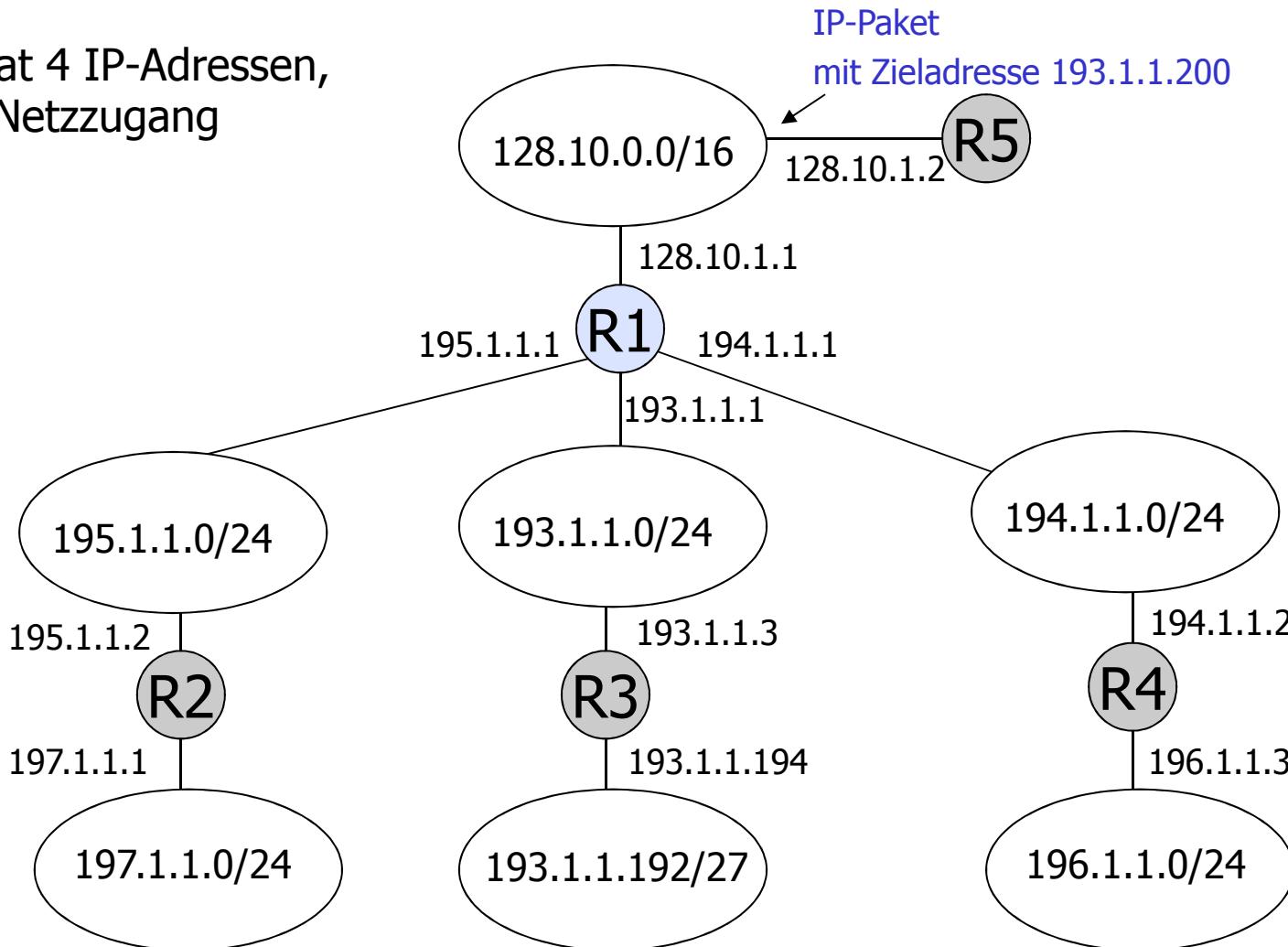
Netzwerkziel	Netzwerkmaske	Gateway	Schnittstelle	Anzahl
0.0.0.0	0.0.0.0	193.1.1.3	193.1.1.1	1
128.10.0.0	255.255.0.0	128.10.1.2	128.10.1.1	1
193.1.1.192	255.255.255.192	193.1.1.3	193.1.1.1	1
196.1.1.0	255.255.255.0	194.1.1.2	194.1.1.1	2
197.1.1.0	255.255.255.0	195.1.1.2	195.1.1.1	2

Standardgateway: 193.1.1.3

- Hinweis: Route mit größter Übereinstimmung finden!

Internet Protocol: Routenbestimmung: Regelwerk – Übung (2)

R1 hat 4 IP-Adressen,
1 je Netzzugang



Internet Protocol: Routenbestimmung: Regelwerk – Übung (2)

■ Lösung:

- Zieladresse $193.1.1.200 = 1100\ 0001\ .\ 0000\ 0001\ .\ 0000\ 0001\ .\ 1100\ 1000$
- Stimmt am besten überein mit Routingtabellen-Eintrag

193.1.1.192	255.255.255.192	193.1.1.3	193.1.1.1	
				1
- Nachweis:

$1100\ 0001\ .\ 0000\ 0001\ .\ 0000\ 0001\ .\ 1100\ 1000$ (Zieladresse 193.1.1.200)

$1111\ 1111\ .\ 1111\ 1111\ .\ 1111\ 1111\ .\ 1100\ 0000$ (Netzwerkmaske 255.255.255.192)

$1100\ 0001\ .\ 0000\ 0001\ .\ 0000\ 0001\ .\ 1100\ 0000$ (Ergebnis der Und-Operation)

$1100\ 0001\ .\ 0000\ 0001\ .\ 0000\ 0001\ .\ 1100\ 0000$ (Netzwerkziel 193.1.1.192 in Routing-Tabelle)

Übereinstimmung mit Zieladresse 193.1.1.192 aus der Routing-Tabelle an
26 Stellen des Netzwerkanteils → beste Route!

→ Ausgewählte Schnittstelle: 193.1.1.3

Überblick

1. Überblick, Routing-Tabellen
- 2. IGP und EGP: Überblick**
3. RIPv1 und RIPv2
4. OSPF und OSPFv2
5. BGP

Internet Protocol: Routing in autonomen Systemen

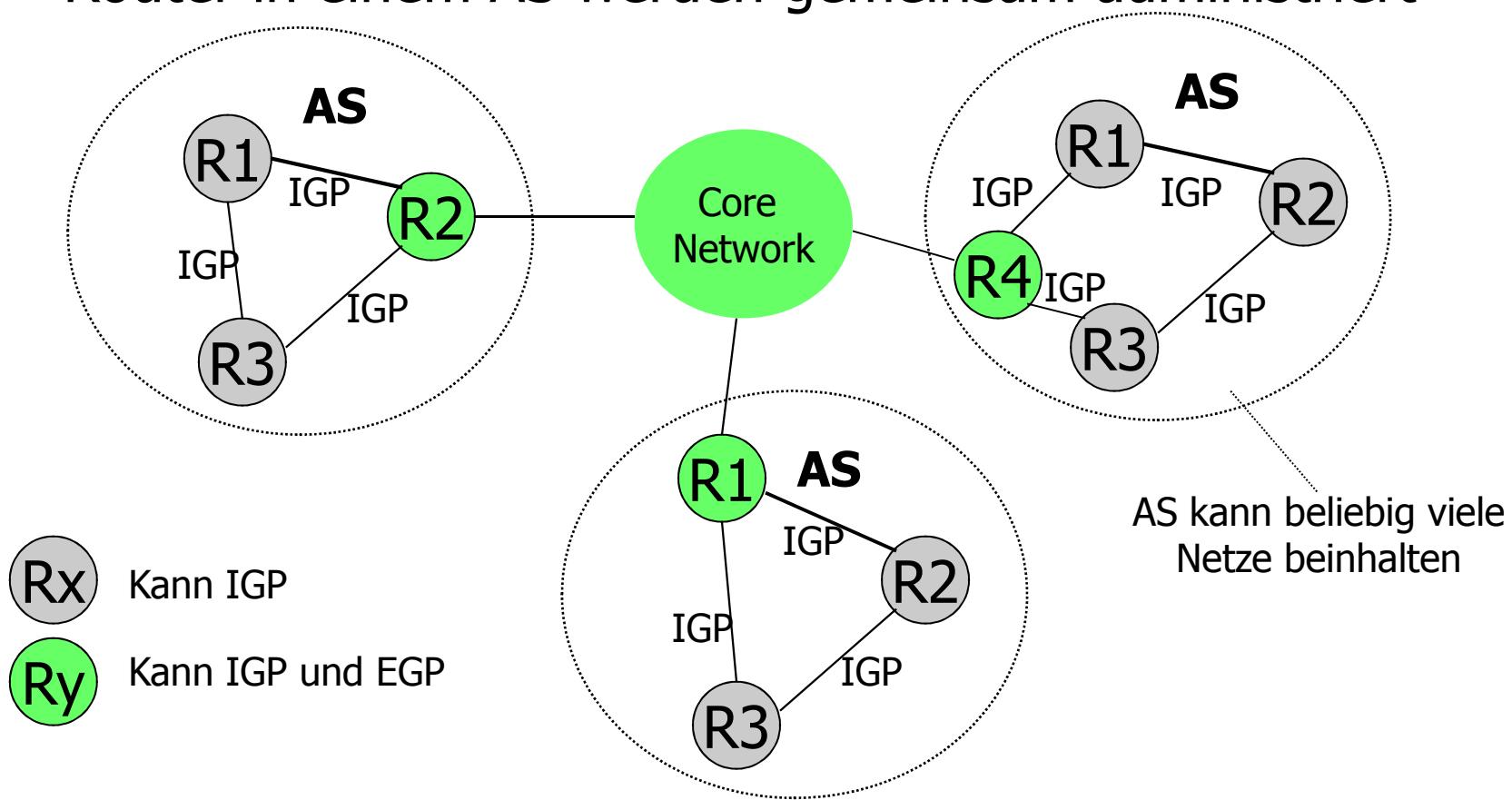
- Jedes autonome System kann intern eigene Routing-Algorithmen verwenden
- Routing-Protokolle für autonome Systeme werden als Interior Gateway Protokolle (**IGP**) bezeichnet
- Älteres Verfahren für kleinere Netze: **RIP**
 - Distance-Vector-Protokoll, aber: Count-to-Infinity-Problem
- Nachfolger von RIP seit 1990 ist **OSPF** (Open Shortest Path First), RFC 1247
 - Wird von der Internet-Gemeinde empfohlen
 - Netz wird als gerichteter Graph abstrahiert
 - Kanten zwischen den Knoten werden mit Kosten gewichtet (Bandbreite, Entfernung, Verzögerung,...)
 - Entscheidung über das Routing anhand der Kosten

Internet Protocol: Routing zwischen autonomen Systemen

- Zwischen AS werden andere Routing-Protokolle benötigt (**Exterior Gateway Protocol, EGP**)
 - Andere Ziele werden verfolgt, Beispiele:
 - Nicht alle Pakete sollen befördert werden
 - Für Transitverkehr muss bezahlt werden
 - Wichtige Informationen nicht durch unsichere autonome Systeme senden
 - ...
 - Routing-Regeln sind erforderlich, die vom Routing-Protokoll unterstützt werden müssen
 - Im Internet wird das Border Gateway Protokoll (**BGP**) empfohlen
 - Pfadvektorprotokoll, verwandt zu Distance-Vector-Protokollen
 - Im Gegensatz zum Distance-Vector-Protokoll werden ganze Pfade ausgetauscht und damit können Routing-Schleifen vermieden werden

Internet Protocol: Routing in autonomen Systemen

- IGP innerhalb eines AS muss gleich sein
- Router in einem AS werden gemeinsam administriert



Überblick

1. Überblick, Routing-Tabellen
2. IGP und EGP: Überblick
- 3. RIPv1 und RIPv2**
4. OSPF und OSPFv2
5. BGP

Internet Protocol: Routing in autonomen Systemen, RIP

- RIP (Routing Information Protocol) wurde ursprünglich von XEROX entwickelt
- Wird in kleinen AS immer noch stark verwendet
- Einfach und leicht zu implementierendes **Distance-Vector-Protocol**
- Als Metrik wird „**Hop-Count**“ verwendet
- RIPv1 ist **klassenorientiert** und ermittelt das Zielnetzwerk anhand der ersten Bits der Ziel-IP-Adresse
 - 0 = Klasse A, 10 = Klasse B, 110 = Klasse C, ...
- Implementierung unter Unix durch **routed**-Prozess
- Routing-Tabelle anschauen: **netstat -r**

Internet Protocol: Routing in autonomen Systemen, RIP

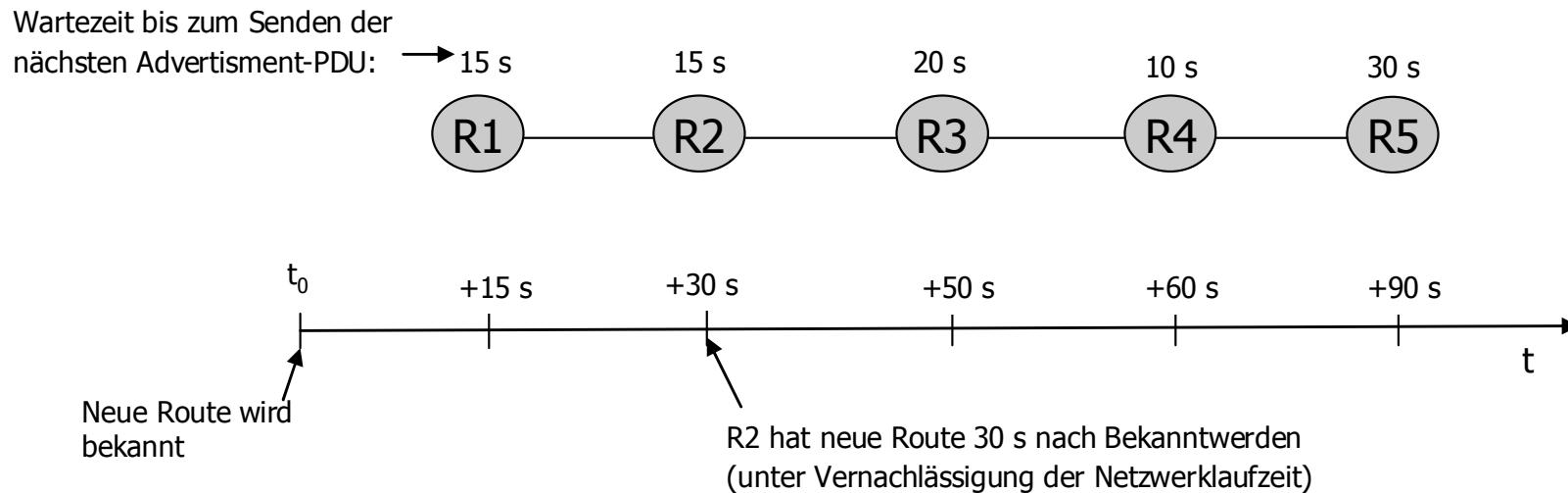
- RIP versendet die Routing-Einträge alle 30 s in sog. Advertisement-PDUs
 - RIPv1 über MAC-Broadcast
 - RIPv2 über Multicast auf Subnetzebene
- Weitere Timer für das Deaktivieren (180 s) und Entfernen (240 s) von Routing-Einträgen definiert
- Nicht geeignet für WAN-Routing, eher im LAN wegen Broadcast/Multicast
- Max. 25 Routeneinträge pro RIP-Nachricht und es werden immer alle Routen übertragen
 - Ggf. mehrere RIP-PDUs senden

Internet Protocol: Routing in autonomen Systemen, RIP

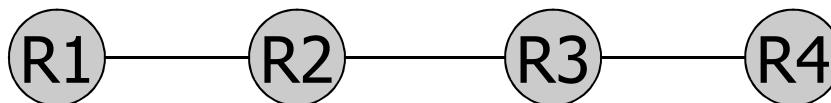
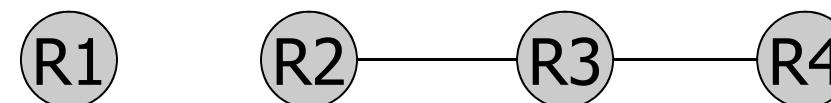
- **Konvergenzzeit:** Zeit, die erforderlich ist, bis alle Router die aktuelle Struktur eines Netzes kennen gelernt haben → Konsistentes Netzwerk
 - Ist bei RIP relativ lange, da RIP-Router Änderungen erst verarbeiten und dann an die Nachbarn propagieren
- **Split Horizon** ist eine Methode, um die Konvergenzzeit kürzer zu halten und um Routing-Schleifen zu vermeiden
- Anm: Max. 15 Hops wurden gewählt, um die Konvergenzzeit zu beschränken

Internet Protocol: Routing in autonomen Systemen, RIP

- Konvergenzzeit und das Problem der langsamen Konvergenz
- Verbreitung von Routing-Tabellen-Einträgen in mehreren Takten
 $\approx 30\text{ s}$ bestimmt die Konvergenzzeit
- Bis zum nächsten Senden einer Advertisement-PDU dauert es im Mittel 15 s , zufallsabhängig

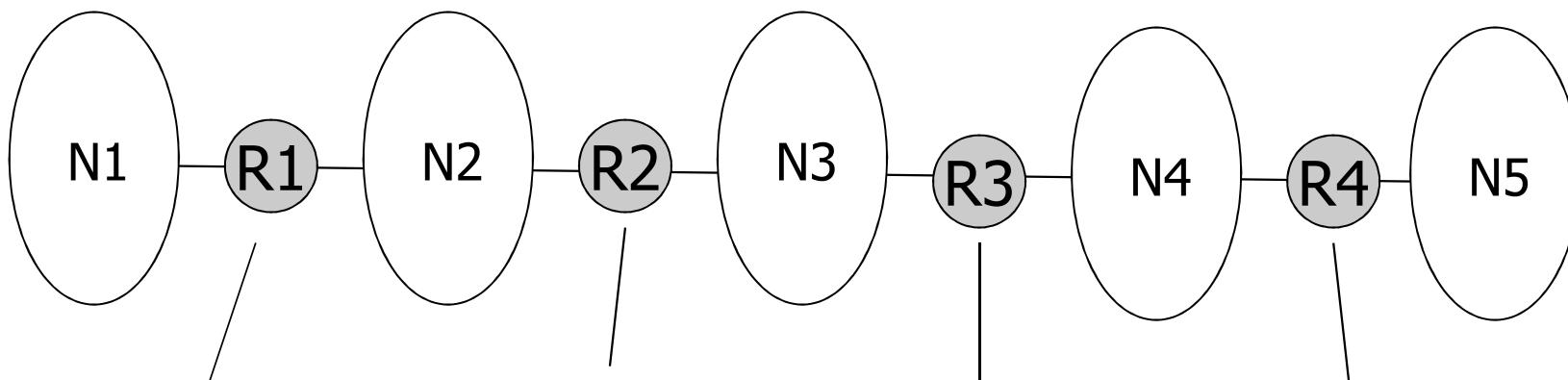


Internet Protocol: Routing in autonomen Systemen, RIP-Beispiel (1)

- **Routing-Schleifen** („Count-to-Infinity-Problem“) möglich
 - Lösung 1: **Split-Horizon-Technik** → Routing-Tabellen enthalten **zusätzlich** die Info, woher die Routing-Info kommt
 - Lösung 2: **Poison-Reverse** (vergifteter Rückweg) → Alle Routen werden propagiert, aber zum Ursprungsnetz hin als „nicht erreichbar“ („vergiftet“)
- Beispiel:
 - a) Alle Verbindungen R1-R2, R2-R3 und R3-R4 intakt
 - b) Verbindung R1-R2 fällt aus
- Was passiert mit und ohne Split-Horizon?

Internet Protocol: Routing in autonomen Systemen, RIP-Beispiel (2)

- Routing-Tabellen im eingeschwungenen Zustand (nach einer angemessenen Konvergenzzeit)



Router R1		
Ziel-netz	Anzahl Hops	Über Router
N1	1	direkt
N2	1	direkt
N3	2	R2
N4	3	R2
N5	4	R2

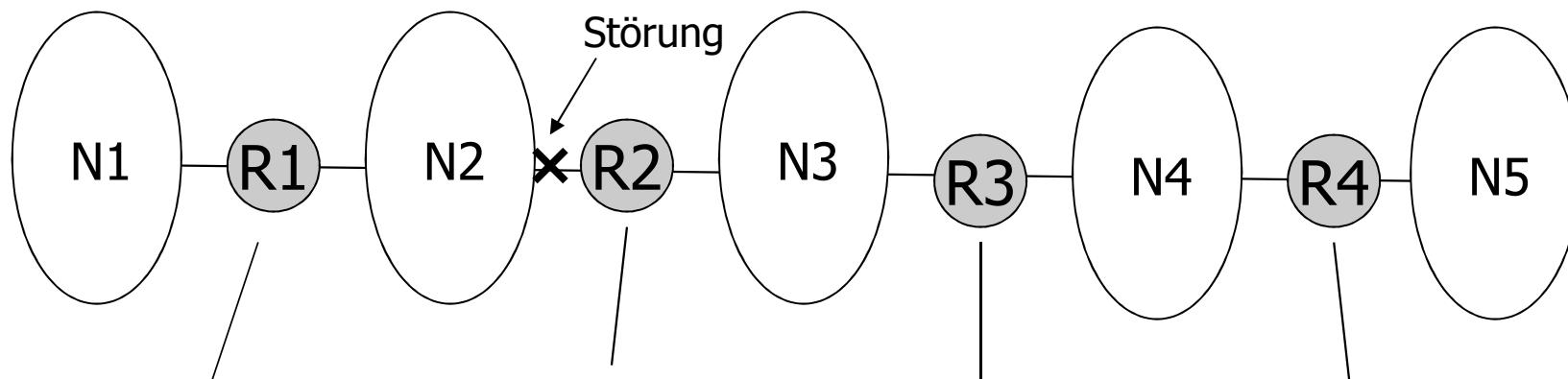
Router R2		
Ziel-netz	Anzahl Hops	Über Router
N1	2	R1
N2	1	direkt
N3	1	direkt
N4	2	R3
N5	3	R3

Router R3		
Ziel-netz	Anzahl Hops	Über Router
N1	3	R2
N2	2	R2
N3	1	direkt
N4	1	direkt
N5	2	R4

Router R4		
Ziel-netz	Anzahl Hops	Über Router
N1	4	R3
N2	3	R3
N3	2	R3
N4	1	direkt
N5	1	direkt

Internet Protocol: Routing in autonomen Systemen, RIP-Beispiel (3)

- Störung im Netzwerkzugang von R2 zu N2: R2 korrigiert sofort



Router R1			Router R2			Router R3			Router R4		
Ziel-netz	Anzahl Hops	Über Router									
N1	1	direkt	N1	16	---	N1	3	R2	N1	4	R3
N2	1	direkt	N2	16	---	N2	2	R2	N2	3	R3
N3	2	R2	N3	1	direkt	N3	1	direkt	N3	2	R3
N4	3	R2	N4	2	R3	N4	1	direkt	N4	1	direkt
N5	4	R2	N5	3	R3	N5	2	R4	N5	1	direkt

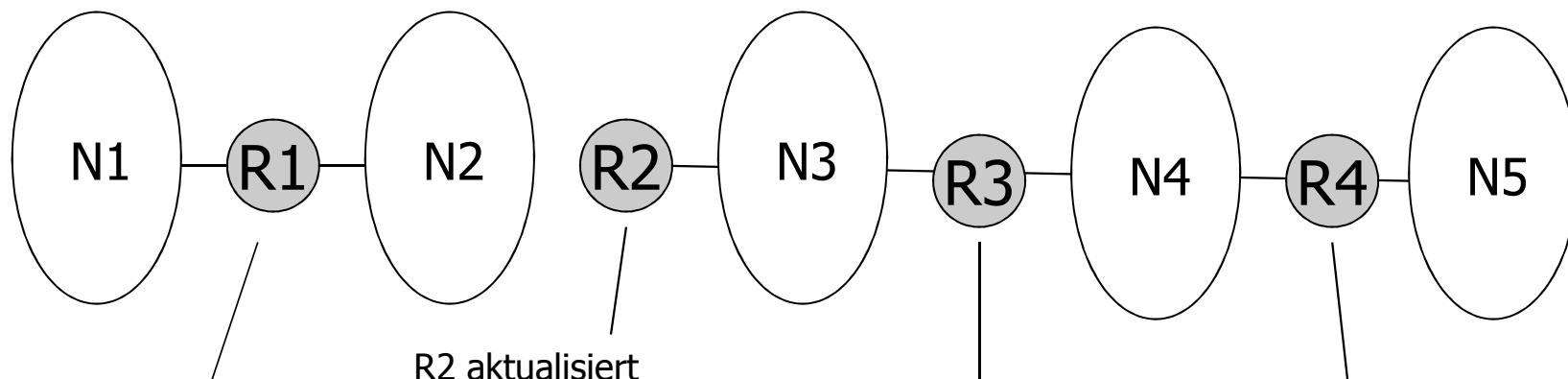
Internet Protocol: Routing in autonomen Systemen, RIP-Beispiel (4)

■ Ohne Split Horizon:

- R3 hat noch die Routing-Info, **dass N1 über drei Hops** erreichbar ist
- R3 propagiert diese Info an R2, also an den Router, über den R1 erreicht wurde
- R2 glaubt dies und sendet Pakete zu R1 nun über R3
- Ping-Pong-Effekt, Routing-Schleife bis Hop-Count = 16, dann erst wird R1 als nicht erreichbar markiert
- Im Folgenden aus Sicht von R2 und R3 skizziert!

Internet Protocol: Routing in autonomen Systemen, RIP-Beispiel (5)

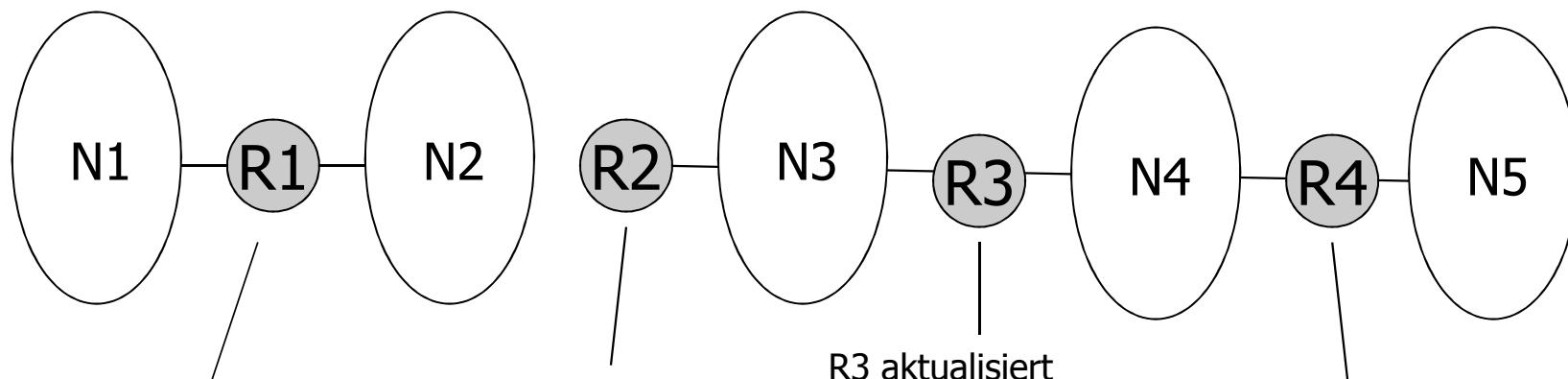
- Nun sendet R3 eine RIPv1-Advertisement-PDU an seinen Nachbarn R2 → $\{(N1, 3), (N2, 2), (N3, 1), (N4, 1), (N5, 2)\}$



Router R1			Router R2			Router R3			Router R4		
Ziel-netz	Anzahl Hops	Über Router									
N1	1	direkt	N1	4	R3	N1	3	R2	N1	4	R3
N2	1	direkt	N2	3	R3	N2	2	R2	N2	3	R3
N3	2	R2	N3	1	direkt	N3	1	direkt	N3	2	R3
N4	3	R2	N4	2	R3	N4	1	direkt	N4	1	direkt
N5	4	R2	N5	3	R3	N5	2	R4	N5	1	direkt

Internet Protocol: Routing in autonomen Systemen, RIP-Beispiel (6)

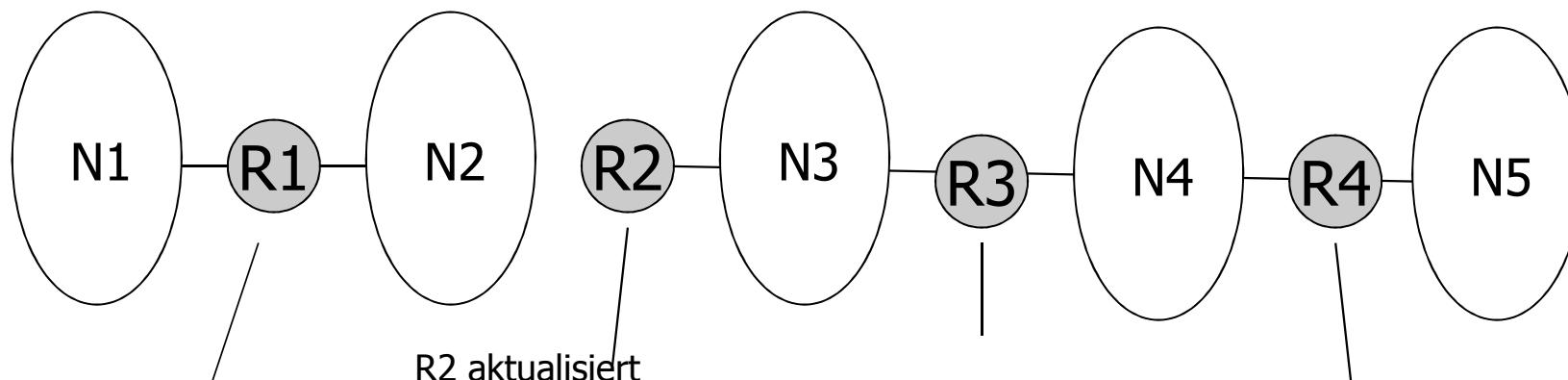
- Als nächstes sendet R2 eine RIPv1-Advertisement-PDU an seinen Nachbarn R3 → $\{(N1, 4), (N2, 3), (N3, 1), (N4, 2), (N5, 3)\}$



Router R1			Router R2			Router R3			Router R4		
Ziel-netz	Anzahl Hops	Über Router									
N1	1	direkt	N1	4	R3	N1	5	R2	N1	4	R3
N2	1	direkt	N2	3	R3	N2	4	R2	N2	3	R3
N3	2	R2	N3	1	direkt	N3	1	direkt	N3	2	R3
N4	3	R2	N4	2	R3	N4	1	direkt	N4	1	direkt
N5	4	R2	N5	3	R3	N5	2	R4	N5	1	direkt

Internet Protocol: Routing in autonomen Systemen, RIP-Beispiel (7)

- Nun sendet R3 wieder eine RIPv1-Advertisement-PDU an seinen Nachbarn R2 → $\{(N1, 5), (N2, 4), (N3, 1), (N4, 1), (N5, 2)\}$



Router R1			Router R2			Router R3			Router R4		
Ziel-netz	Anzahl Hops	Über Router									
N1	1	direkt	N1	6	R3	N1	5	R2	N1	4	R3
N2	1	direkt	N2	5	R3	N2	4	R2	N2	3	R3
N3	2	R2	N3	1	direkt	N3	1	direkt	N3	2	R3
N4	3	R2	N4	2	R3	N4	1	direkt	N4	1	direkt
N5	4	R2	N5	3	R3	N5	2	R4	N5	1	direkt

USW.

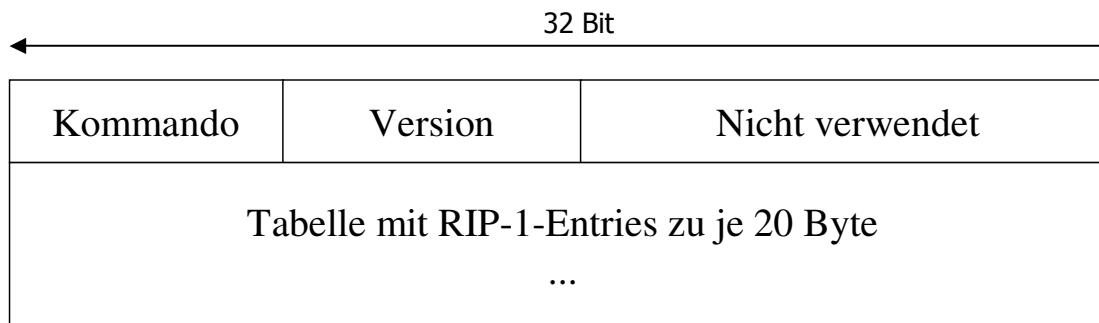
Internet Protocol: Routing in autonomen Systemen, RIP-Beispiel (8)

- **Mit Split Horizon:**

- R3 weiß, woher die Routing-Info für **N1** kommt (von R2)
- Route mit höheren Kosten wird nicht zurückpropagiert
- Keine Routing-Schleife (in diesem Fall)

Internet Protocol: Routing in autonomen Systemen, RIPv1-PDU

- Metrik = 16 → Netzwerkziel nicht erreichbar
- AFI = Adressierungsart, bei IP-Adressen immer 0x02

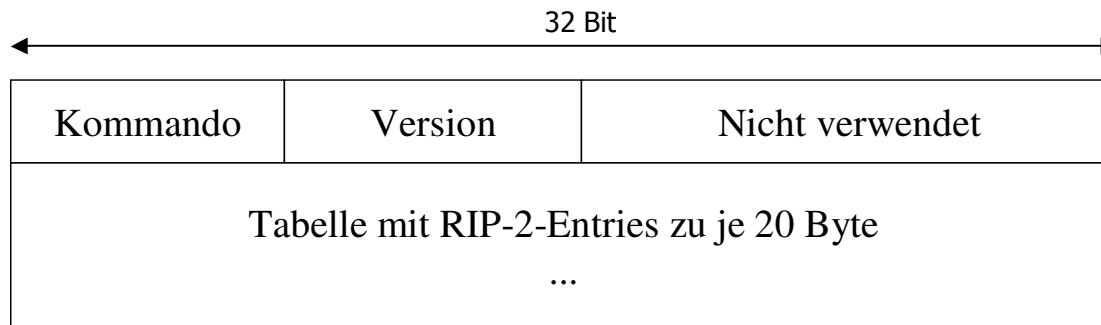


RIP-1-Entry:

Address-Family-Identifier	Nicht verwendet
IPv4-Adresse	
Nicht verwendet	
Nicht verwendet	
Metrik	

Internet Protocol: Routing in autonomen Systemen, RIPv2-PDU

- Next-Hop: Direkte Angabe eines Zielhosts möglich



RIP-2-Entry:

Address-Family-Identifier	Route-Tag
IPv4-Adresse	
Subnet-Mask	
Next-Hop	
Metrik	

Internet Protocol: Routing in autonomen Systemen, RIP

■ Sonstiges zu RIP

- RIPv1 **unterstützt** CIDR/VLSM **nicht**
 - Subnetzmaske wird in RIPv1 **nicht** übermittelt
 - Klasse wird aus den ersten Bits der Ziel-IP-Adresse ermittelt
- RIPv2 **unterstützt** CIDR/VLSM
- RIPv2 kann **Split-Horizon** und Split-Horizon mit Poisson-Reverse
- RIPv2 kann selbst ausgelöste Router-Aktualisierungen (**Triggered Updates**) bei Ankunft einer Advertisement-PDU
 - Höhere, aber immer noch nicht perfekte Konvergenz, mehr Netzwerklast
- RIPv1 kommuniziert über **Broadcast**, RIPv2 über **Multicast**

Überblick

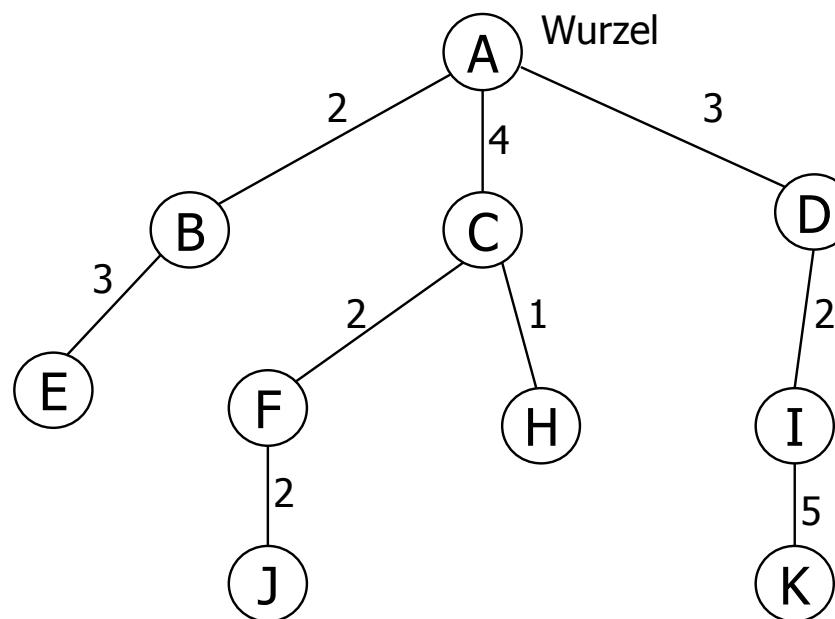
1. Überblick, Routing-Tabellen
2. IGP und EGP: Überblick
3. RIPv1 und RIPv2
- 4. OSPF und OSPFv2**
5. BGP

Internet Protocol: Routing in autonomen Systemen, OSPF

- OSPF ist **für große Unternehmensnetze** gedacht, für kleine wird noch RIP bzw. statische Routing-Tabellen verwendet
- **Offener Standard** (Open SPF), RFC 1247 u. 2328
- OSPF ist ein **Link-State-Protocol**
 - „Link State“ ist der Zustand einer Verbindung zweier Router
→ zustandsorientiert statt entfernungsorientiert (RIP)
- Kommunikation mit unmittelbaren, **designierten Nachbarn** zum Austausch der Routing-Information
- Jeder Router führt **eigene Datenbasis** (Link-State-Datenbank) mit allen Routing-Einträgen des Netzes

Internet Protocol: Routing in autonomen Systemen, OSPF, SPF-Baum

- Jeder IP-Router erzeugt aus seiner Sicht einen Spanning Tree (SPF-Baum) für das ganze Netzwerk
- Wurzel ist der Router selbst
- Verzweigung = günstigste Route



Internet Protocol: Routing in autonomen Systemen, OSPF, Sonst. Features

- **Load Balancing** bei Pfaden mit gleichen Kosten
 - gleichmäßige Verteilung, besser als bei RIP
- Nutzung spezieller **Multicast-Adressen** zur Kommunikation
- Unterstützung der Router-**Authentifizierung** zur Vermeidung von Angriffen (mehr Sicherheit)

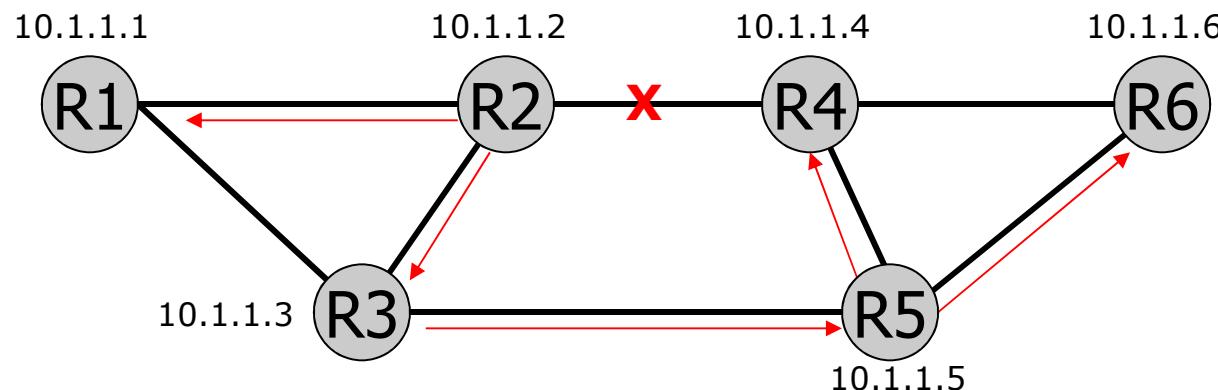
- Metriken (Kosten) in RFCs nicht festgelegt
 - Cisco IOS (Internet Operating System) verwendet z.B. als Metrik die Gesamtbandbreite aller Ausgangsschnittstellen vom aktuellen Router bis zum Zielnetzwerk

Internet Protocol: Routing in autonomen Systemen, OSPF, Funktionalität

- Alle Router suchen beim Start ihre Nachbarn mit **Hello-PDUs**, aber nicht alle angrenzenden Router werden auch zu Nachbarn (sog. **adjacents**)
- **Zyklischer Abgleich** der Link-State-Datenbanken mit den Nachbarn
- **Lebendüberwachung** periodisch unter den Nachbarn
- **Link State Updates** (Konsistente Datenhaltung in allen Routern) periodisch **und** bei Topologieänderungen
- **Refreshing** spätestens alle 30 Minuten

Internet Protocol: Routing in autonomen Systemen, OSPF-Konvergenz

- Verteilung einer Veränderung im Netz geht schnell und Information wird vor der Verarbeitung weiterkommuniziert
 - Hohe Konvergenz
- Beispiel für Routen-Austausch:
 - Verbindung zwischen 10.1.1.2 und 10.1.1.4 fällt aus
 - Link-State Updates werden über das ganze Netz verteilt
 - Nachdem DB synchronisiert ist, gibt es nur noch eine kürzeste Route

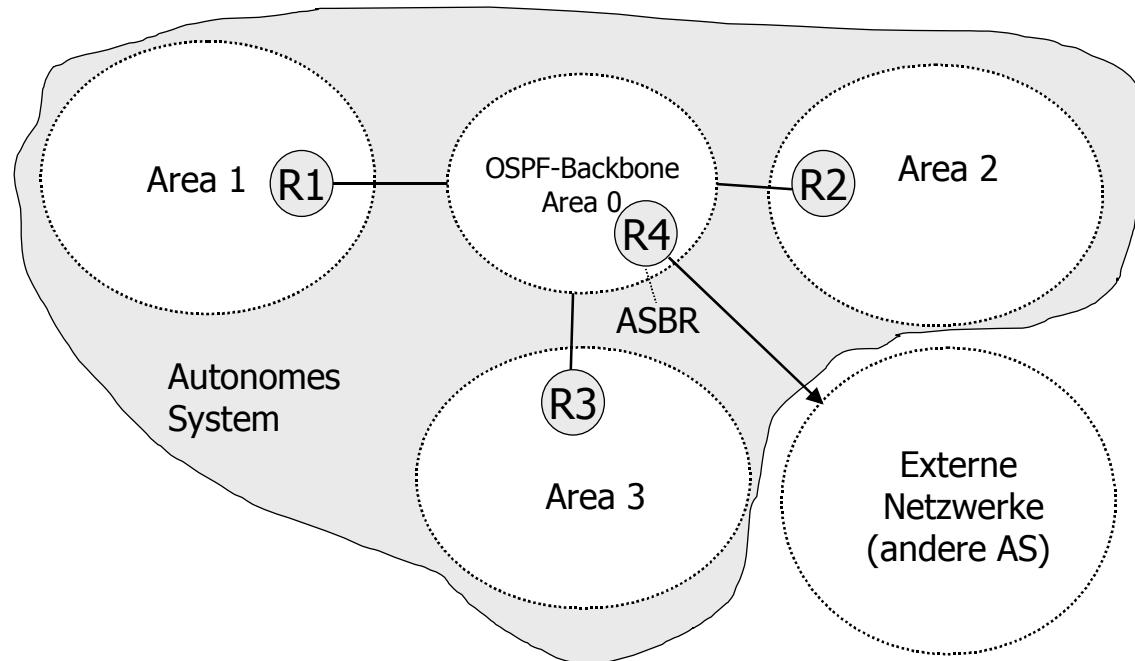


Vgl. Dirk Jakob, Folienvortrag

Internet Protocol: Routing in autonomen Systemen, OSPF-Routertypen

- Bei OSPF gibt es vier Router-Klassen
 - Interne Router der Area
 - Router an Bereichsgrenzen (Area-Grenzen)
 - Verbinden zwei oder mehrere Areas
 - Backbone-Router
 - Befinden sich im Backbone
 - AS-Grenz-Router (ASBR)
 - Vermitteln zwischen autonomen Systemen
- Bei Einsatz in Broadcast-orientierten LANs:
Verwendung von sog. designierten Routern
 - Alle Router bauen eine Nachbarschaft (Adjacencies) zu diesem Router auf → Reduzierung der Kommunikation

Internet Protocol: Routing in autonomen Systemen, OSPF-Backbone



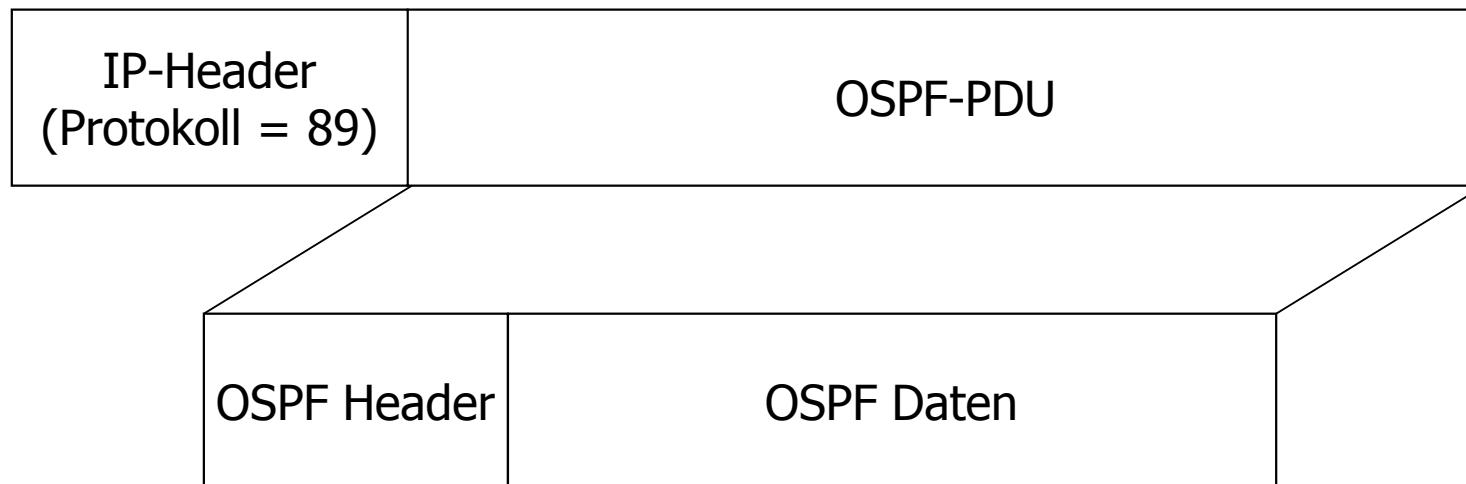
- R1, R2, R3 = Router an Bereichsgrenzen
- R4 = AS-Grenz-Router (ASBR), vermittelt zwischen autonomen Systemen

Internet Protocol: Routing in autonomen Systemen, OSPF-Nachbarschaften

- Nachbarschaften werden bei der Initialisierung aufgebaut
 - Senden von Hello-PDUs alle 10 s
 - Hello-PDUs enthalten bekannte Nachbar-Router
- Wenn Nachbarschaft aufgebaut ist, wird alle 40 s eine Hello-PDU als Heartbeat gesendet
 - Bleibt diese aus, wird Nachbar für ausgefallen erklärt
- Die Verteilung von Änderungen der Routing-Tabellen erfolgt immer zu allen Nachbarn

Internet Protocol: Routing in autonomen Systemen, OSPF-PDUs

- Es gibt fünf OSPF-PDU-Typen:
 - **Hello**: Feststellung der Nachbarn, Aufbau von Nachbarschaften
 - **Database Description**: Bekanntgabe der neuesten Daten
 - **Link State Request**: Informationen vom Partner anfordern
 - **Link State Update**: Informationen an Nachbarn verteilen
 - **Link State Acknowledgement**: Bestätigung eines Updates



Internet Protocol: Routing in autonomen Systemen, OSPF-PDUs

- PDUs werden direkt über IP gesendet, (siehe IP-Header, Protokoll = 89)
- Direkte Übertragung an Nachbarn oder über spezielle Multicast-Adressen
- OSPF-PDUs werden nur zwischen Nachbarn ausgetauscht
- Kein Weiterrouten außerhalb des eigenen Netzes (IP-Header, TTL=1)

Vgl. Comer, Volume I, S. 255 ff für weitere Informationen zu den PDU-Inhalten

Übungen (1)

- Welche Routing-Informationen verwaltet RIP und welche OSPF?
 - RIP verwaltet als Distanz-Vektor-Protokoll die Entfernung (Anzahl Hops) und Richtung zum Ziel (Vektor)
 - OSPF verwaltet sog. Links in einer sog. Link-State-Datenbasis. In dieser wird die gesamte Topologie des Netzes verwaltet
 - Bei OSPF sind also alle Routen im Netz genau bekannt, bei RIP nicht
- Warum konvergiert OSPF im Vergleich zu RIPv2 besser?
 - Beide senden Veränderungen sofort (Triggered Update)
 - RIPv2 verarbeitet die Routing-Updates zuerst und dann werden sie an die Nachbarn weitergereicht, bei OSPF ist dies genau umgekehrt

Übungen (2)

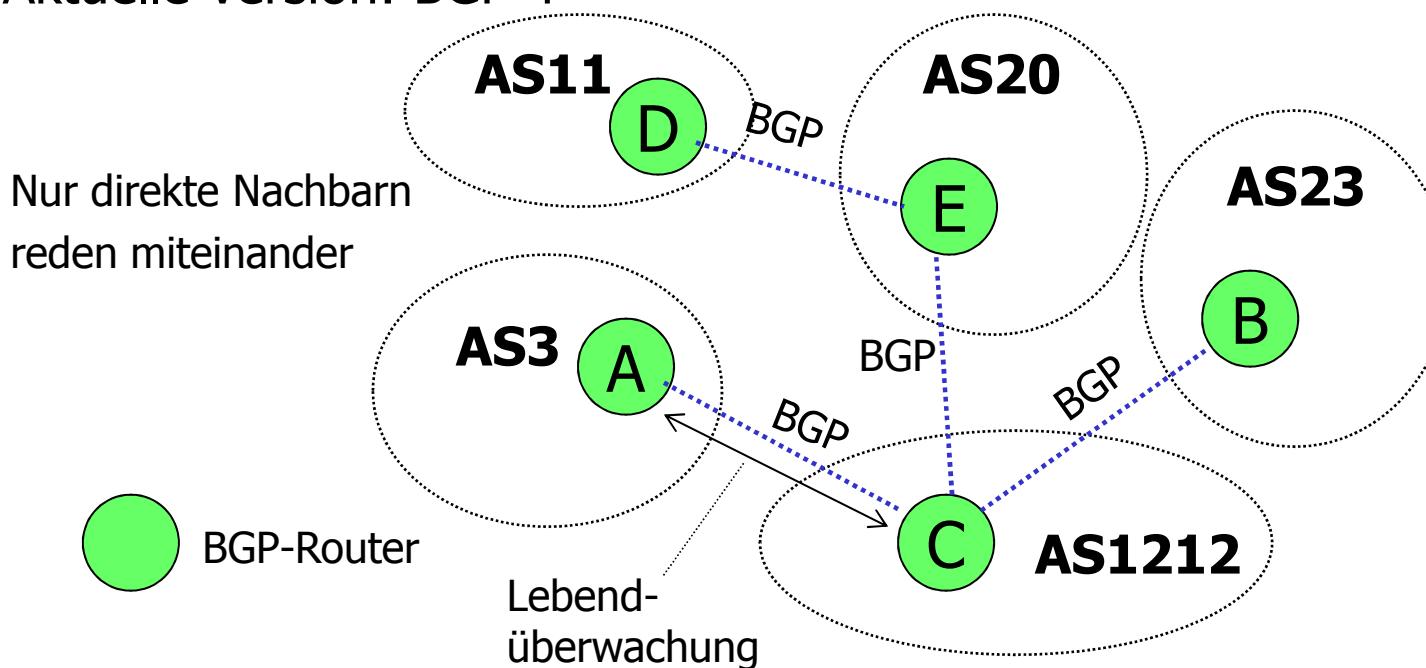
- Wie werden bei RIP und bei OSPF die Kosten (Metrik) berechnet?
 - Bei RIP: Anzahl Hops ist die verwendete Metrik
 - Bei OSPF: Nicht festgelegt, Cisco nutzt z.B. die Bandbreite des Gesamtpfades als Metrik, je größer, umso besser. Dies geht, da die Gesamttopologie bekannt ist.

Überblick

1. Überblick, Routing-Tabellen
2. IGP und EGP: Überblick
3. RIP-1 und RIP-2
4. OSPF und OSPFv2
- 5. BGP**

Internet Protocol: Routing über autonome Systeme hinweg, BGP

- BGP (= Border Gateway Protocol) ist ein EGP (RFC 1654), derzeit im Internet nur BGP im Einsatz
- BGP ermöglicht das Routing zwischen verschiedenen autonomen Systemen (AS)
- Aktuelle Version: BGP 4

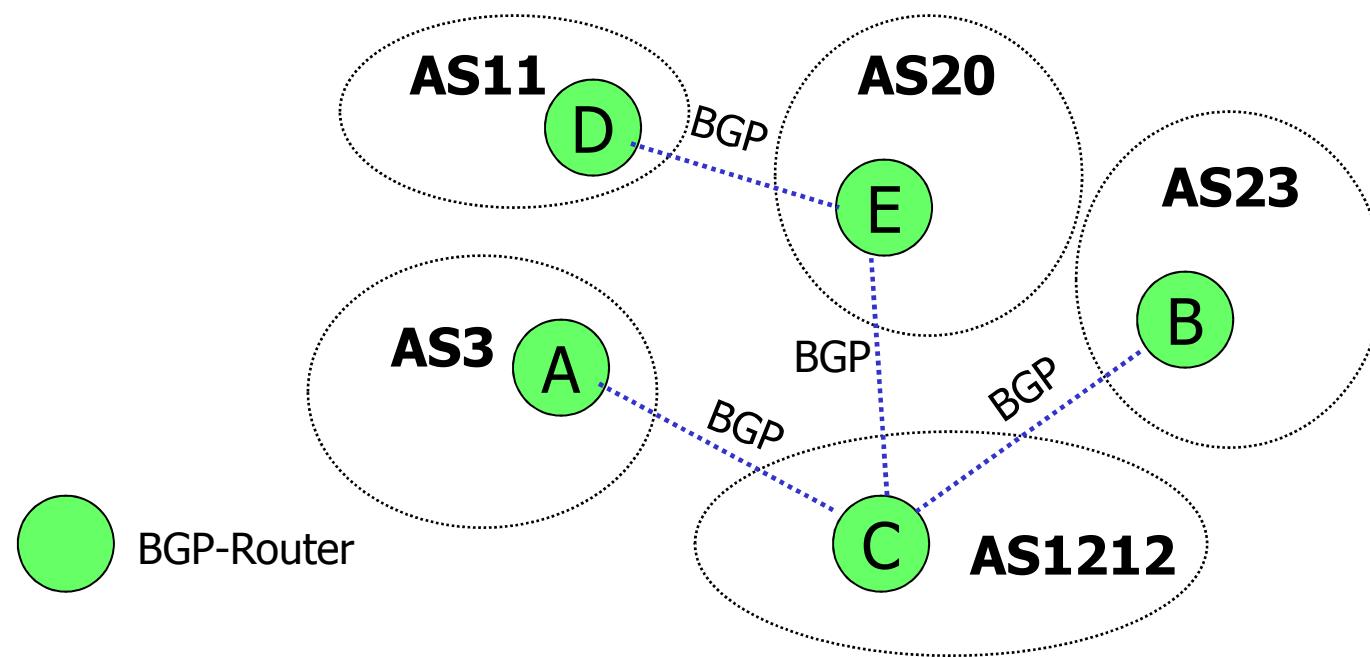


Internet Protocol: Routing über autonome Systeme hinweg, BGP

- BGP ist ein **Pfadvektorprotokoll** (Path Vector Protocol), ähnelt dem Distance-Vector-Protokoll
- BGP-Router kennen die besten Route zu anderen AS als **vollständigen Pfad (auf AS-Ebene)**
- Jeder BGP-Router führt eine **Datenbank** mit Routen zu allen erreichbaren autonomen Systemen
- Routing-Tabellengröße:
 - ~ 200.000 Einträge und mehr nicht selten

Internet Protocol: Routing über autonome Systeme hinweg, BGP

- Routing-Tabelle von D enthält folgende Routen:
 - AS11 – AS20
 - AS11 – AS20 – AS1212
 - AS11 – AS20 – AS1212 – AS23
 - AS11 – AS20 – AS1212 – AS3



Internet Protocol: Routing über autonome Systeme hinweg, BGP

- Ein BGP-Router informiert **periodisch** alle **Nachbar-**
BGP-Router **genau** über die zu nutzenden Routen
 - UPDATE-PDUs werden versendet, sog. Advertisements
- Zyklen in Routen (Routing-Schleifen) werden bei
Übernahme der Information geprüft
 - Eigene AS-Nummer darf nicht in Route sein
 - Count-to-Infinity-Problem tritt nicht auf
- BGP-Router überwachen sich gegenseitig (Heartbeat-
Protokoll → KEEPALIVE-PDU)

Internet Protocol: Routing über autonome Systeme hinweg, BGP

- BGP-Router verwendet zur Auswahl der besten Routen eine **Routing-Policy**
- Routing-Policy (Regeln), Beispiele:
 - Kein Verkehr über einen bestimmten Knoten
 - Sicherheitsaspekte
 - Kostenaspekte
 - ...
- Eine Route, die die Regeln verletzt, wird auf „unendlich“ gesetzt
- BGP nutzt TCP (Port 179) als Transportprotokoll für seine Nachrichten (verbindungsorientiert!)

Rückblick

1. Überblick, Routing-Tabellen
 2. IGP und EGP: Überblick
 3. RIPv1 und RIPv2
 4. OSPF und OSPFv2
 5. BGP
-
- Interessant wäre auch
 - Routing für mobile Systeme, Ad-hoc Routing,...
 - Routing für Multicast
 - OSPF for IPv6 (OSPFv3)

Datenkommunikation

Internet-Steuerprotokolle und IPv6

Wintersemester 2012/2013

Überblick

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

1. Steuerprotokolle

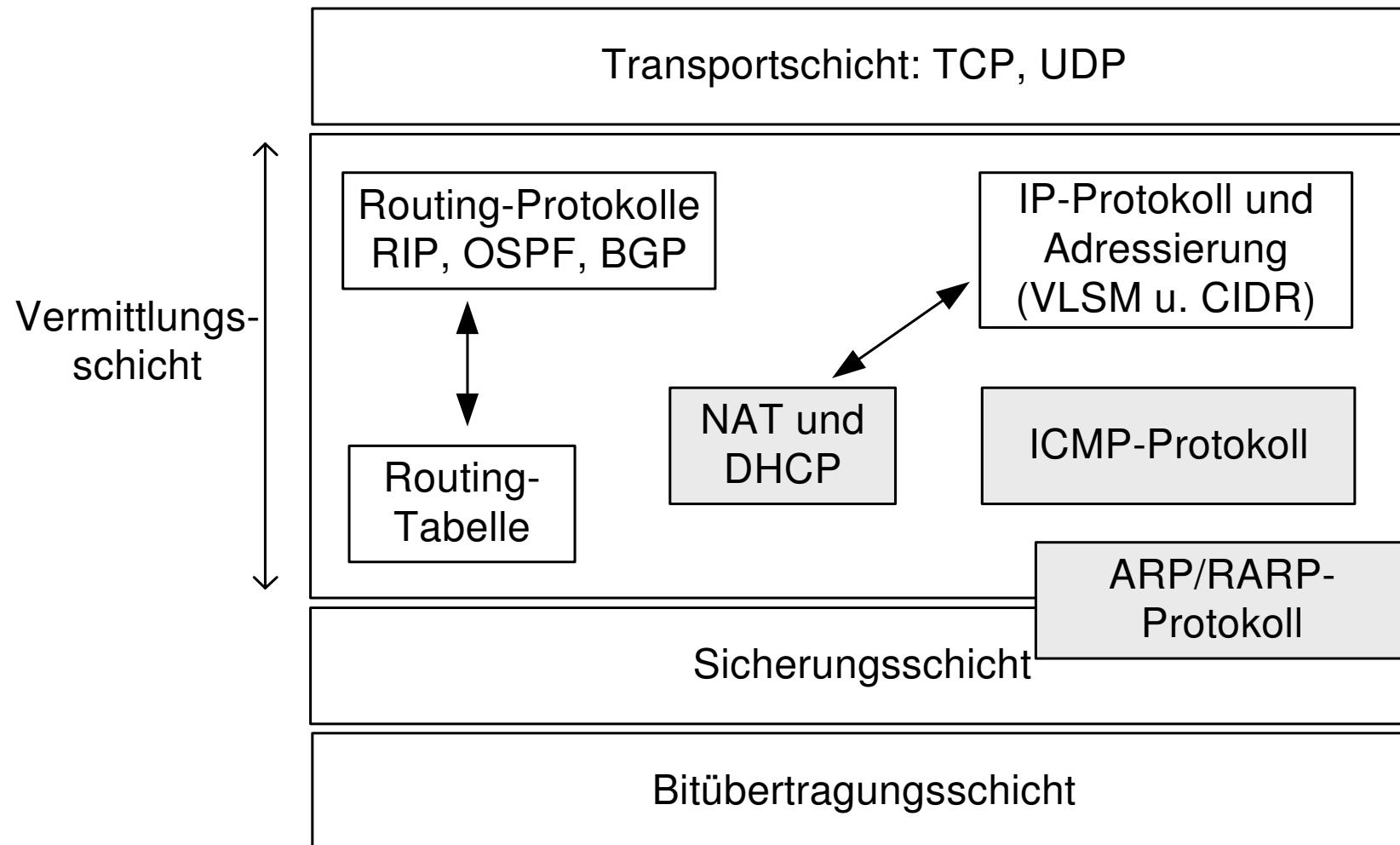
- **ICMP**
- ARP und RARP
- NAT
- DHCP

2. IPv6

- Grundlagen und Adressierung
- IPv6-PDU
- Automatismen, Neighbor Discovery

Einordnung

Die Internet-Vermittlungsschicht

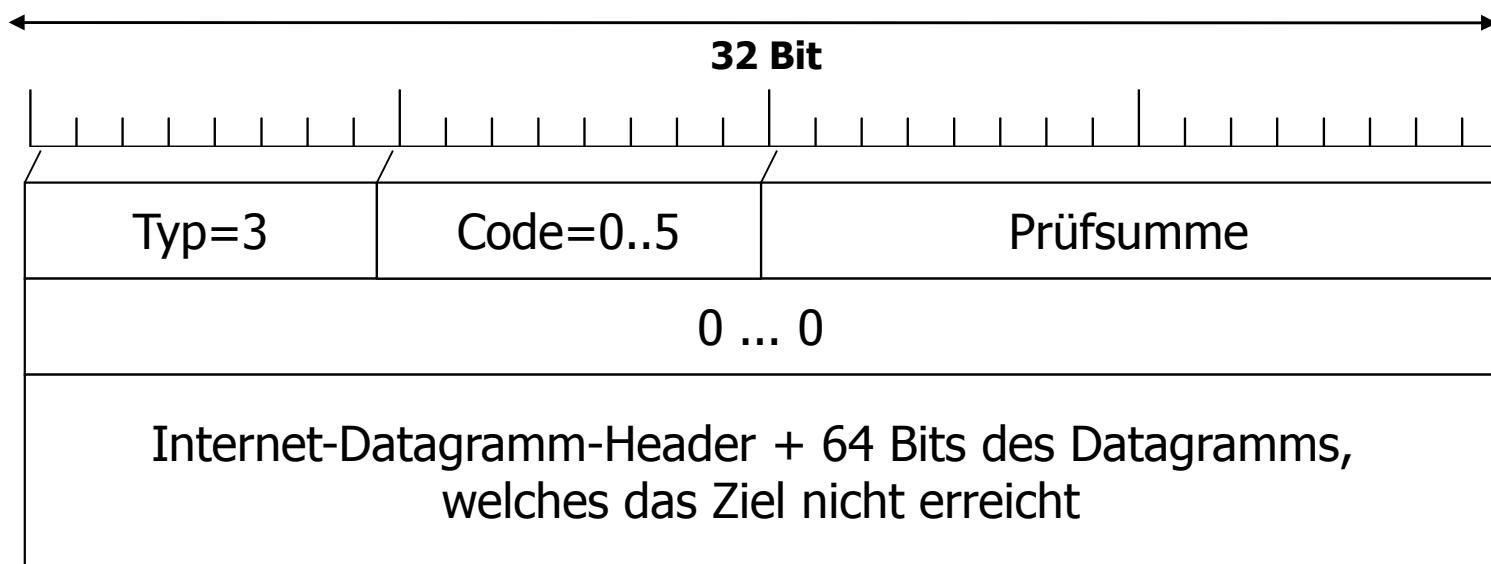


ICMP: Einführung

- ICMP (Internet Control Message Protocol, RFC 792)
 - Dient der Übertragung von unerwarteten Ereignissen und für Testzwecke
 - Beispiel 1: Ein Netzwerk ist nicht erreichbar: Ein IP-Router sendet in diesem Fall die ICMP-PDU „Network Unreachable“
 - Beispiel 2: Das ping-Kommando verwendet z.B. ICMP-PDUs (Echo Request, Echo Reply)
 - Beispiel 3: Das Kommando traceroute (tracert) nutzt ICMP (Typ=11, Time-to-live exceeded)
- ICMP-Nachrichten werden in IP-Datagrammen versendet

ICMP: PDU

- ICMP-Beispiel: **Destination unreachable** → Ein Router kann ein Datagramm nicht ausliefern



- Router sendet ICMP-Nachricht an den Absender
- Code: 0 = Netzwerk nicht erreichbar, 1 = Rechner nicht erreichbar,...

Überblick

1. Steuerprotokolle

- ICMP
- **ARP und RARP**
- NAT
- DHCP

2. IPv6

- Grundlagen und Adressierung
- IPv6-PDU
- Automatismen, Neighbor Discovery

ARP

- ARP (Address Resolution Protocol), RFC 826
 - ARP dient dem **dynamischen Mapping** von IP-Adressen auf Schicht-2-Adressen (MAC-Adressen)
 - Jeder Host kennt seine eigene Schicht-2-Adresse, nicht aber die Adressen der anderen Hosts
 - Jeder Host führt einen **ARP-Cache** und merkt sich darin Schicht-2-Adressen, die über ARP im **IP-Broadcasting** erfragt werden können → **Periodisches Löschen vermeidet Inkonsistenz!**
 - Ist der Zielhost nicht gespeichert, wird ein **ARP-Broadcast** mit der IP-Zieladresse als Parameter versendet
 - Der Zielrechner antwortet mit einem **ARP-Reply** (MAC-Adresse)
 - IP-Router übernehmen Rolle des **ARP-Proxy**

ARP-Cache

C:\>**arp -a**

Schnittstelle: 192.168.2.116 --- 0x2

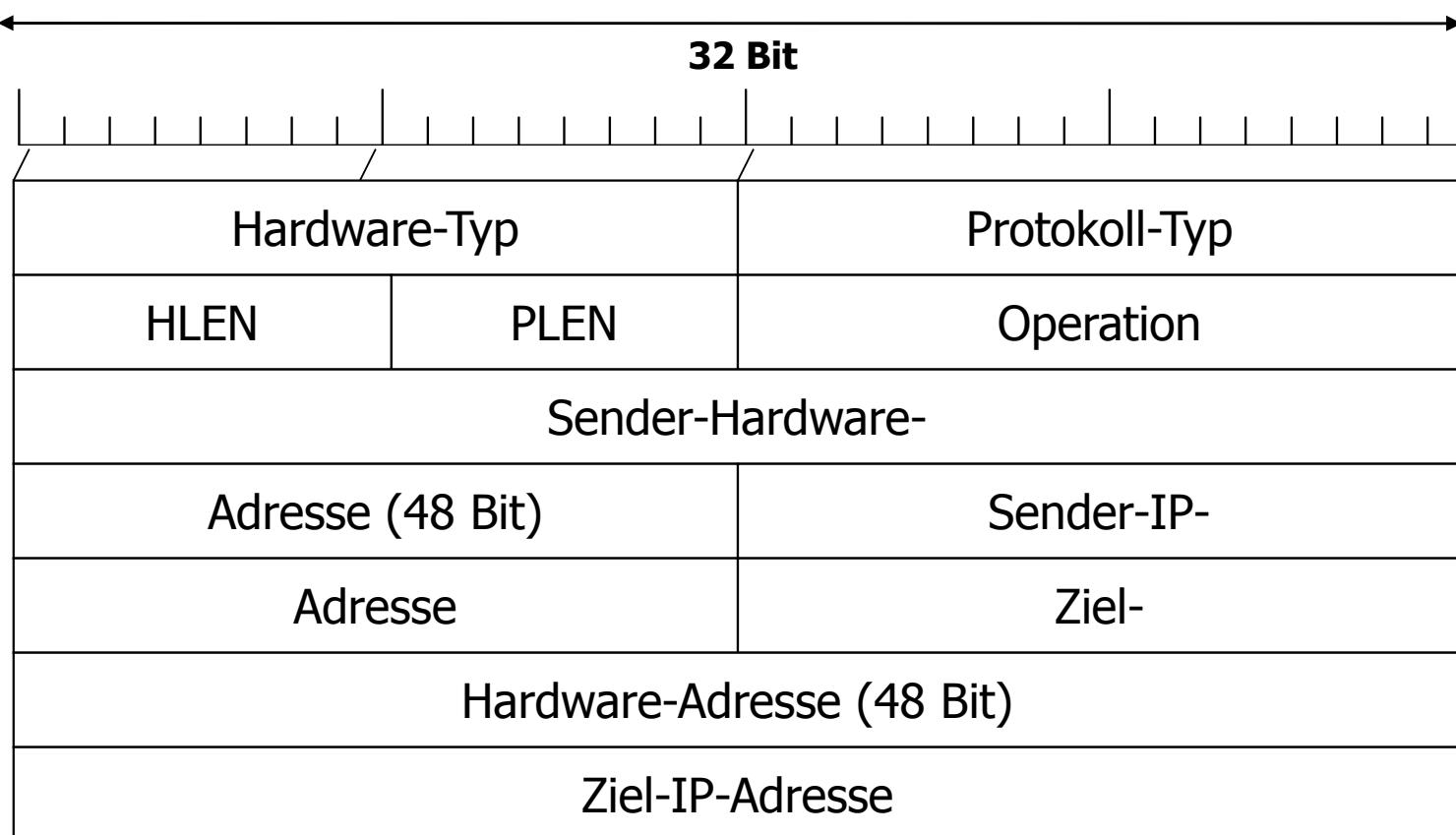
Internetadresse	Physikal. Adresse	Typ
192.168.2.1	00-50-fc-cb-7e-da	dynamisch
192.168.2.14	00-e0-4c-10-17-32	dynamisch
192.168.2.250	08-00-37-31-de-ae	dynamisch

RARP

- RARP = Reverse ARP
 - RARP wird verwendet, wenn die eigene IP-Adresse eines Hosts nicht bekannt ist, aber benötigt wird
 - RARP sendet RARP-Request mit der eigenen MAC-Adresse als Broadcast
 - Anwendungsfall:
 - Plattenlose Desktop-Arbeitsplätze, die beim Booten Ihre IP-Adresse ermitteln wollen

ARP-PDU

- Nachrichtenformat eines ARP/RARP-Pakets

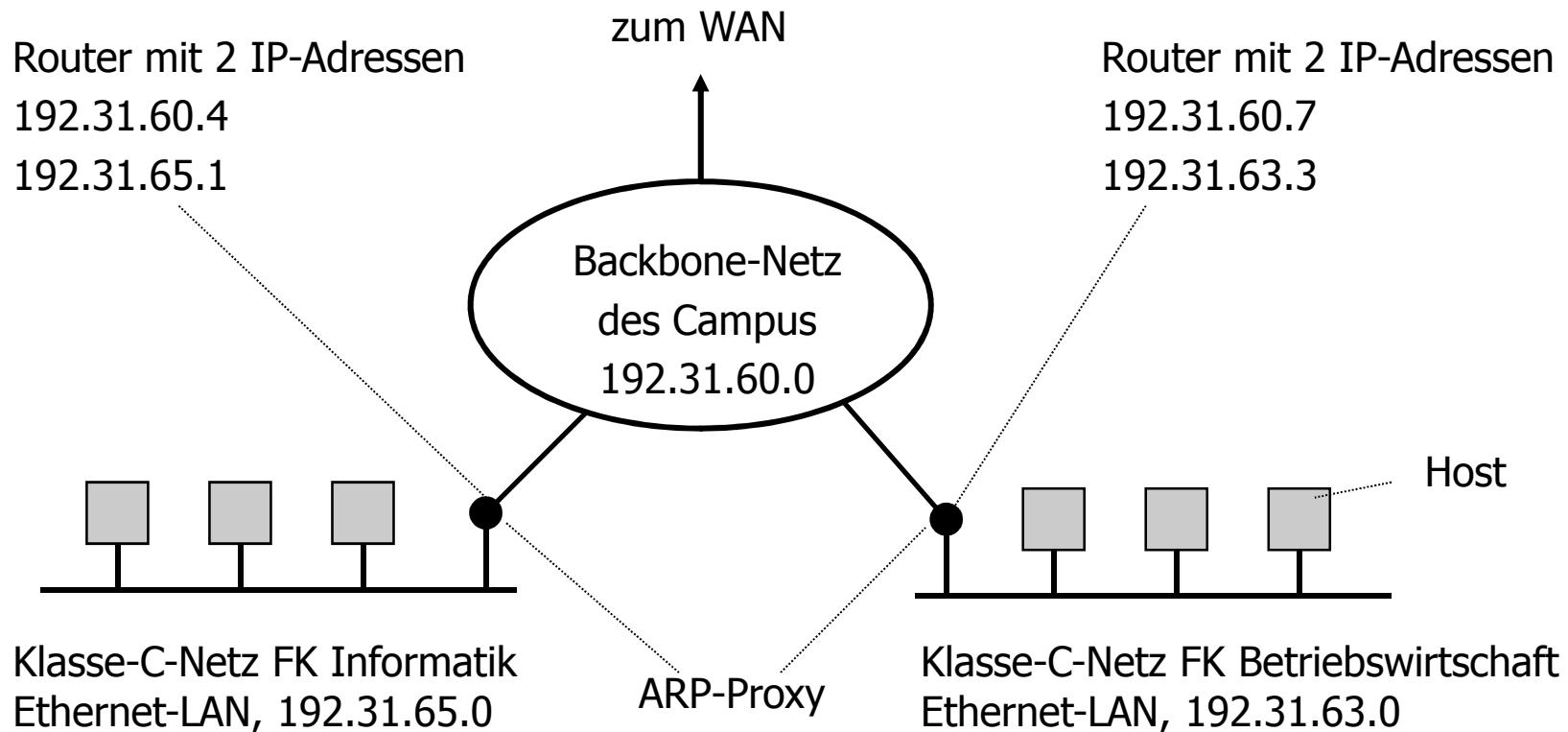


ARP-PDU

- **Hardware-Typ:**
 - Hardware-Typ, 1 = Ethernet
- **Protokoll-Typ:**
 - Typ des High-Level-Protokolls, X`0800` = IP
- **HLEN:**
 - Hardware-Adressenlänge
- **PLEN:**
 - IP-Adressenlänge
- **Operation:**
 - 1 = ARP-Request
 - 2 = ARP-Response
 - 3 = RARP-Request
 - 4 = RARP-Response
- ...

ARP: Beispiel

- Typisches Netzbeispiel: Campusnetz (nach Tanenbaum)



Überblick

1. Steuerprotokolle

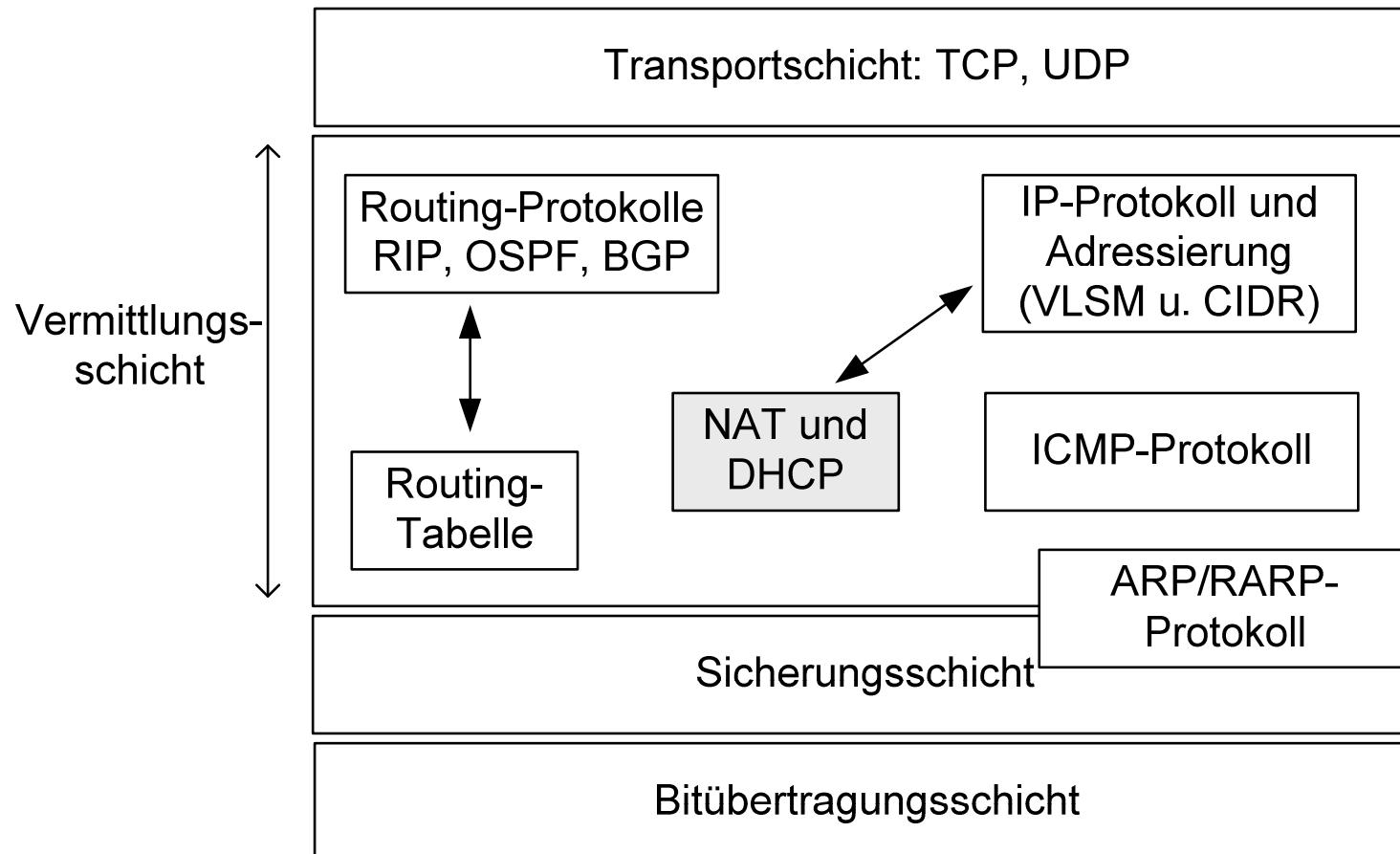
- ICMP
- ARP und RARP
- **NAT**
- DHCP

2. IPv6

- Grundlagen und Adressierung
- IPv6-PDU
- Automatismen, Neighbor Discovery

Einordnung

Die Internet-Vermittlungsschicht



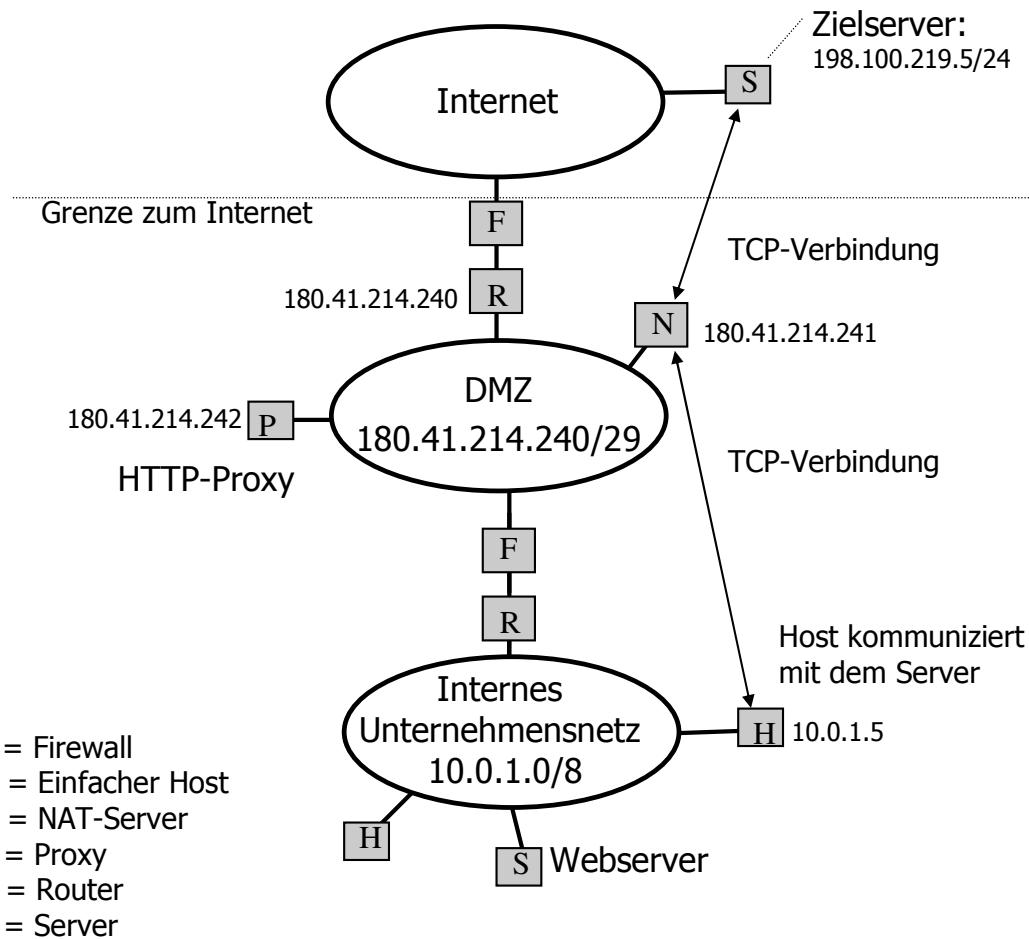
Network Address Translation (NAT): Einführung

- Bei NAT handelt sich vorwiegend um eine Möglichkeit, die **Sicherheit** im Unternehmensnetz zu erhöhen
 - NAT dient auch dazu, Netzwerkadressen einzusparen
 - Für ein Netz (Unternehmensnetz) benötigt man **nur noch eine bzw. wenige offizielle IP-Adresse**
 - Intern kann dann eine beliebige, nach außen nicht sichtbare, Netzwerknummer verwendet werden
- Private IP-Adressen!

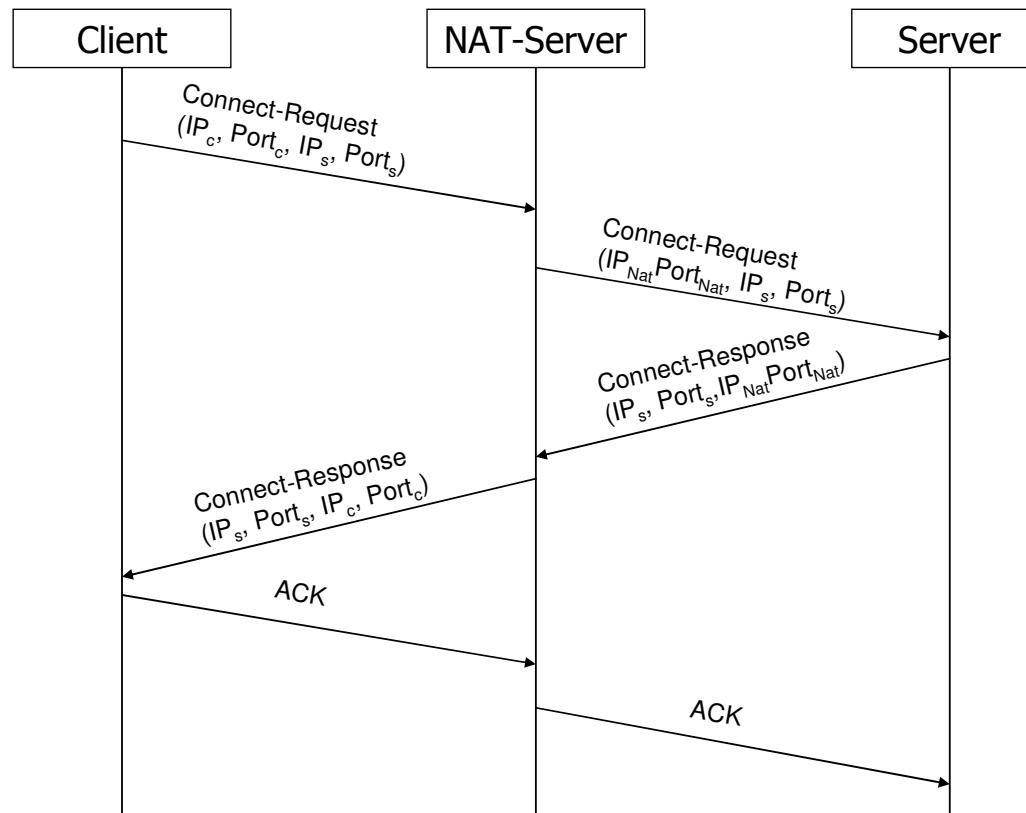
NAT, NAPT

- IP-Router bzw. NAT-Server „**mappen**“ bei NAT ankommende Pakete auf interne Hostadressen und umgekehrt
- IP-Router arbeiten nach außen als **Stellvertreter** (Proxies) für alle internen Hosts
- Verschiedene Varianten von NAT verfügbar, auch NAPT = Network Address **Port** Translation

NAT: Beispiel



NAT: Nachrichtenfluss



- Diskussion: End-to-End-Beziehung

Überblick

1. Steuerprotokolle

- ICMP
- ARP und RARP
- NAT
- **DHCP**

2. IPv6

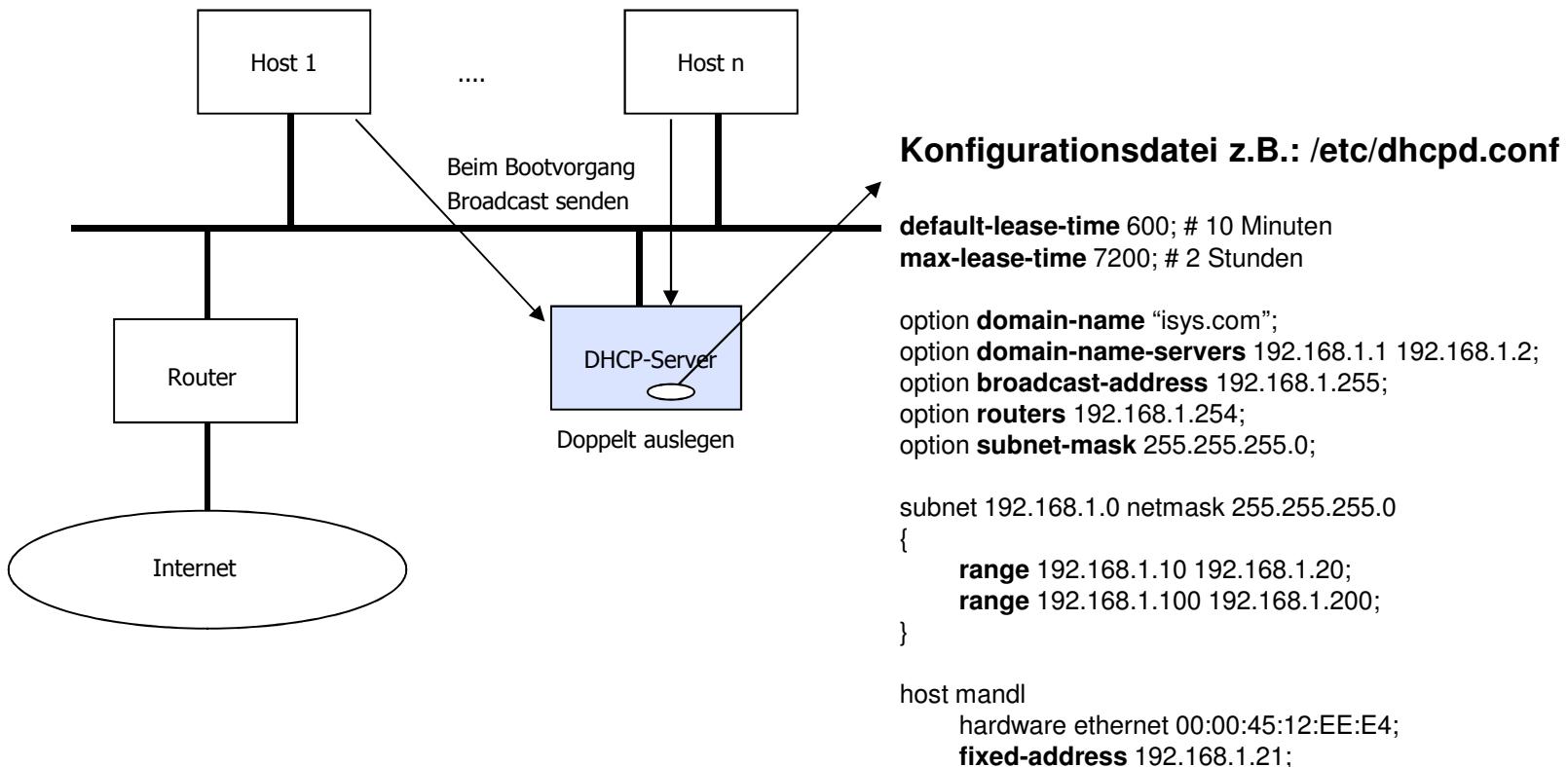
- Grundlagen und Adressierung
- IPv6-PDU
- Automatismen, Neighbor Discovery

DHCP: Einführung

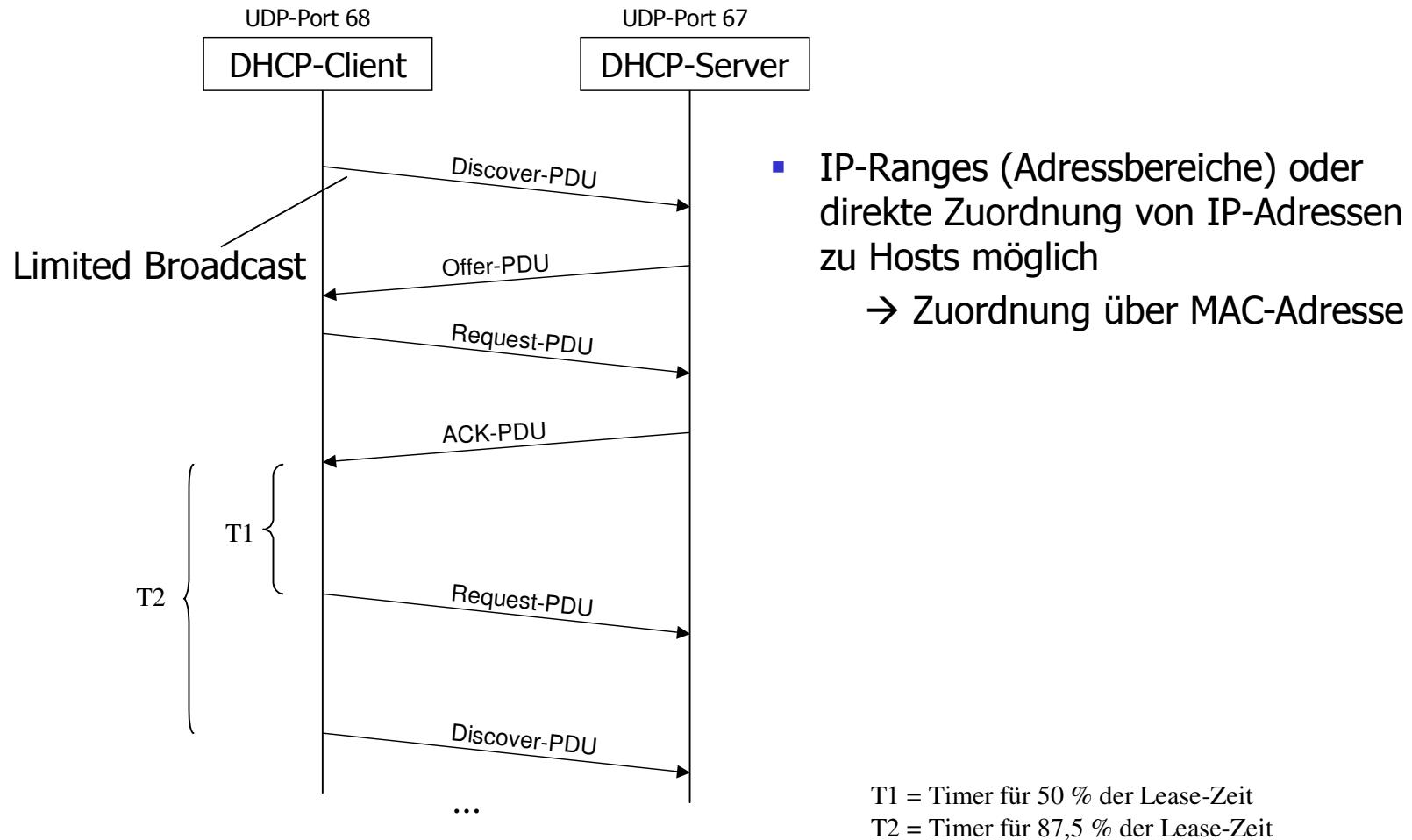
- Manuelle Netzwerkkonfiguration ist schon bei kleinen Netzen ein Problem
- Dynamic Host Configuration Protocol schafft Abhilfe
 - Hosts müssen Adressen nicht mehr kennen, sie werden dynamisch beim Booten besorgt
- Dynamische Vergabe von IP-Adressen und weiteren Netzwerk-Parametern über einen DHCP-Server:
 - Subnetzmaske
 - DNS-Server-Adresse
 - IP-Router-Adresse
 - ...

DHCP: Beispielnetz

- Meist beziehen nur Arbeitsplatzrechner Ihre IP-Adresse vom DHCP-Server



DHCP: Kommunikation beim Bootvorgang



DHCP: Leases

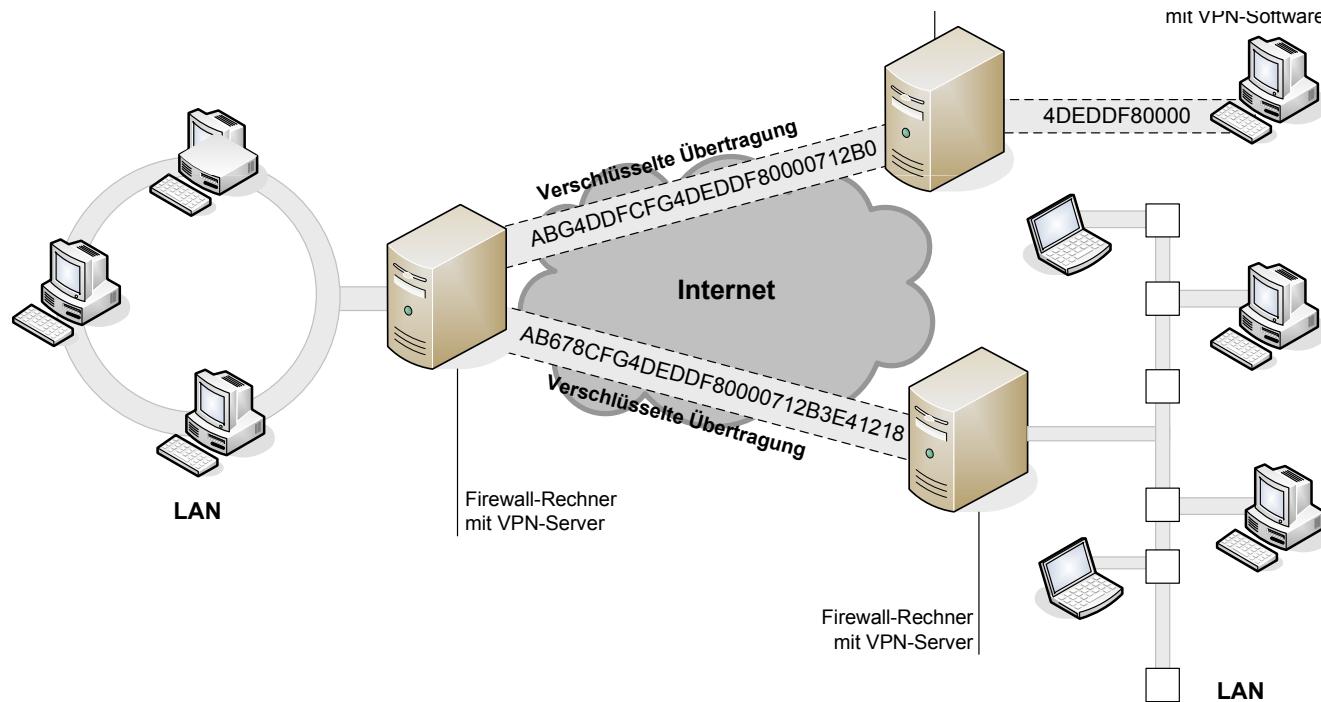
- Lease-Zeit = Nutzungszeit für IP-Adresse
 - Parameter (Timer) werden beim dyn. Konfigurieren an den Client gesendet
 - Parameter T1 gibt standardmäßig 50 % der Lease-Zeit an
 - Client sendet erneut einen DHCP-Request
 - Parameter T2 87,5 % der Lease-Zeit
 - Wenn kein ACK vom Server kommt, dann erneutes DHCP-Discovery durch Client

Ergänzung: Wichtige Administrations-Kommandos

- Diagnose- und Konfigurationskommandos im TCP/IP-Umfeld, die man öfter mal braucht:
 - ping
 - hostname
 - netstat
 - nslookup (kommt später bei DNS)
 - arp
 - traceroute (Windows: tracert)
 - ifconfig
 - route
 - ipconfig (Windows)
 - nbtstat (Windows)

Ergänzung: Virtual Private Networks (VPN)

- Sicherheitsprotokolle:
 - IPSec und IKE (Key Exchange Protocol)
 - IPSec-Tunnel



Überblick

1. Steuerprotokolle

- ICMP
- ARP und RARP
- NAT
- DHCP

2. IPv6

- **Grundlagen und Adressierung**
- IPv6-PDU
- Automatismen, Neighbor Discovery

Grundlegendes

- CIDR reicht nicht für alle Zeit, daher wurde eine neue Version von IP konzipiert (seit 1990)
- Zukunftsszenarien:
 - **Jeder Fernseher ist möglicherweise bald ein Internet-Knoten** (Video-on-Demand)
 - **Millionen von drahtlosen** Systemen im Internet
- Hauptziel von IPv6 (IPng) ist es, die **Adressproblematik** umfassend und langfristig zu lösen
- Koexistenz mit IPv4 erforderlich und angestrebt

Weitere Ziele

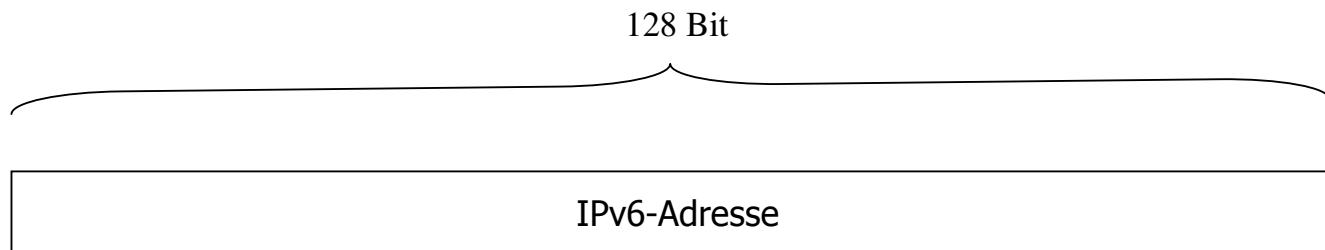
- **Vereinfachung** des Protokolls zur schnelleren Bearbeitung von Paketen in Routern
- Umfang der **Routing-Tabellen reduzieren**
- **Anwendungstypen** wie Multimedia-Anwendungen (Echtzeitanwendungen) unterstützen
 - Unterstützung von **Flussmarken**
- Höhere **Sicherheit** (Datenschutz, Authentifikation)
- **Multicasting** besser unterstützen
- **Mobile IP-Adressen**: Möglichkeit schaffen, dass Hosts ihr Heimatnetz verlassen können
- Möglichkeiten der **Weiterentwicklung** schaffen

IPv6-Adressen

- **16-Byte-Adressen** (128 Bits) mit neuer Notation
- Analogie zur Anzahl der vorhandenen Adressen (2^{128}):
 - Wäre die ganze Welt mit Computern bedeckt, könnte man mit IPv6 **$7*10^{23}$** IP-Adressen pro m² ermöglichen
- Verschiedene Klassen von Adressen
 - **Unicast**-Adressen
 - Der traditionelle Adresstyp
 - Adressieren einen Netzanschluss eines Hosts oder Routers
 - **Anycast**-Adressen
 - Adressierung einer Gruppe von Interfaces
 - Aber nur ein Mitglied der Gruppe bekommt das Paket
 - Auswahl übernimmt der zuständige Router
 - Nutzung innerhalb von Teilnetzen, kein Routing außerhalb
 - **Multicast**-Adressen
 - Adressierung einer Gruppe von Interfaces
 - **Keine** Broadcast-Adresse mehr in IPv6!!
- Aufteilung des IPv6-Adressraums in **RFC 4291** geregelt

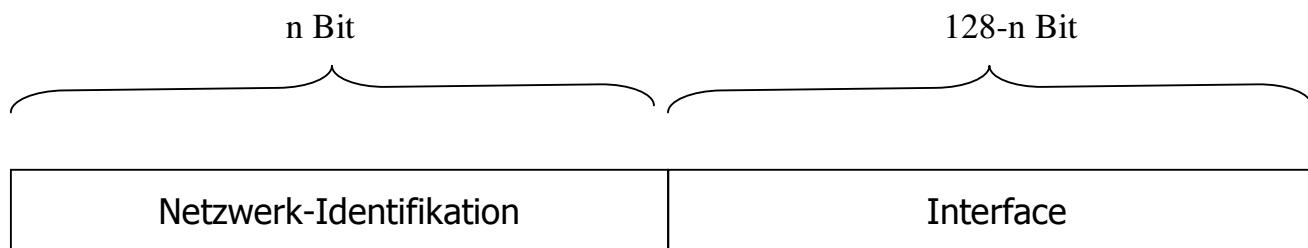
IPv6-Adressaufbau: Struktur

- Unstrukturierte IPv6-Adresse



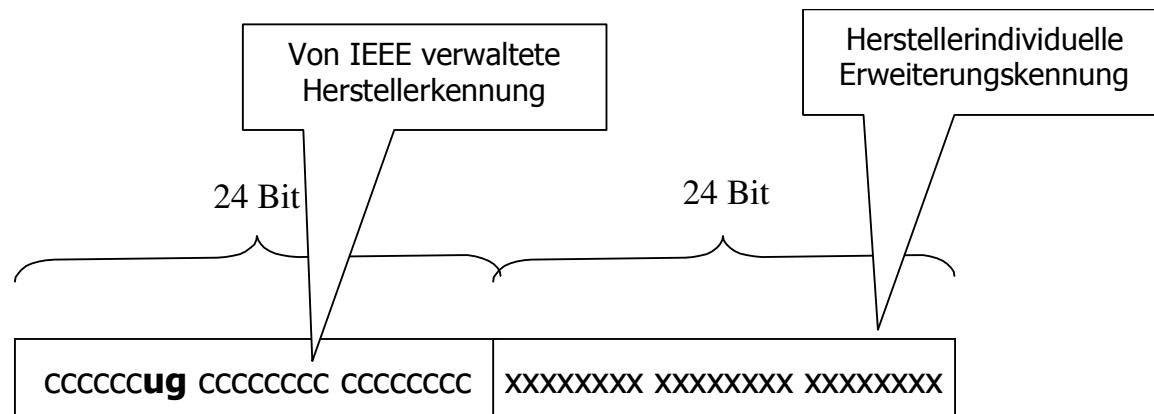
- Strukturierte IPv6-Adresse

- Netzwerkanteil über CIDR-Notation (x/y) kennzeichnen, ISP erhält /32-Adressen von NIC oder DENIC usw. und gibt meist /64-Adressen weiter



Einschub: IEEE 802.3-Adresse (Schicht 2)

- 48-Bit-MAC-Adresse, als IEEE 802.3-Adresse bezeichnet
 - 24-Bit-Firmenkennung
 - 24-Bit-Erweiterungskennung (Platinenkennung)
 - Wird bei der Herstellung zugewiesen und ist global eindeutig
 - Dies ist die bekannte MAC-Adresse (Media Access Control)



c = Company-Bit

x = Bit vom Interface-Hersteller vergeben

u = universal oder local

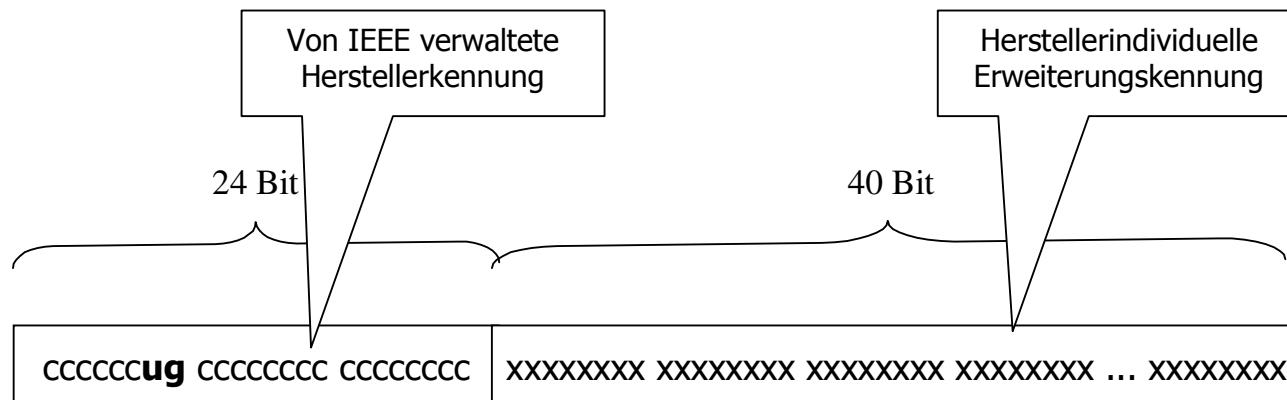
g = group oder individual

Einschub: IEEE 802.3-Adresse (Schicht 2)

- U/L (Universal/Local)
 - Das U/L-Bit ist das siebte Bit des ersten Byte und wird zur Bestimmung, ob es sich um eine universell oder eine lokal verwaltete Adresse handelt, verwendet.
 - U/L-Bit = 0 → Adresse von IEEE verwaltet
 - U/L-Bit = 1 → Netzwerkadministrator verwaltet Adresse lokal
- I/G (Individual/Group)
 - Das I/G-Bit ist das Bit mit niedrigster Priorität des ersten Byte
 - Dient zur Festlegung, ob es sich um eine individuelle Adresse (Unicast) oder eine Gruppenadresse (Multicast) ist
 - I/G-Bit = 0 → Unicastadresse
 - I/G-Bit = 1 → Multicastadresse
- Bei einer 802.x-Standard-Netzwerkadapteradresse gilt:
 - U/L-Bit = I/G-Bit = 0 → universell verwaltete MAC-Unicastadresse

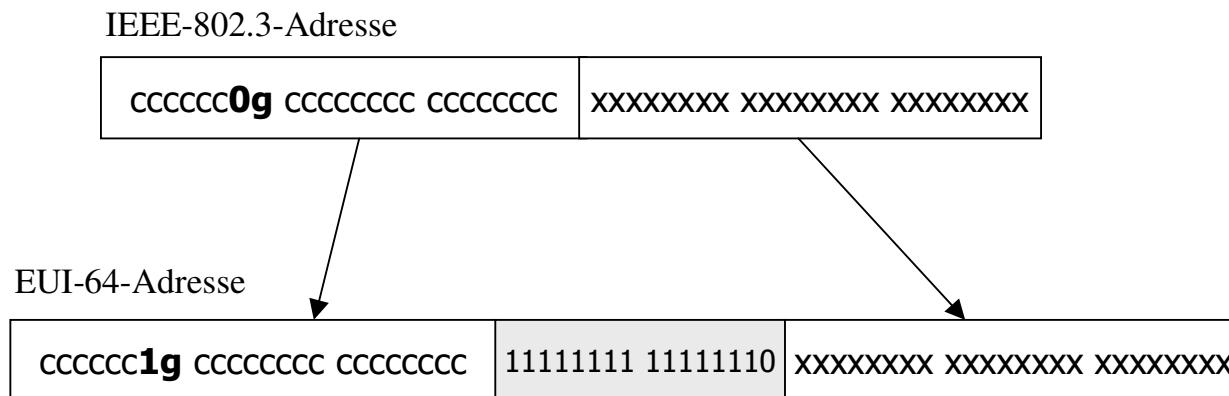
Einschub: EUI-64-Adresse (Schicht 2)

- IEEE EUI-64-Adresse (Extended Unique Identifier) ist ein neuer Standard in der Adressierung von Netzwerkschnittstellen. Zwei Adressteile:
 - Firmenkennung ist 24 Bit lang
 - Erweiterungskennung ist 40 Bit → größerer Adressbereich für Hersteller von Netzwerkadaptersn
- Die IEEE EUI-64-Adresse verwendet die U/L- und I/G-Bits auf dieselbe Art wie die IEEE 802.3-Adresse



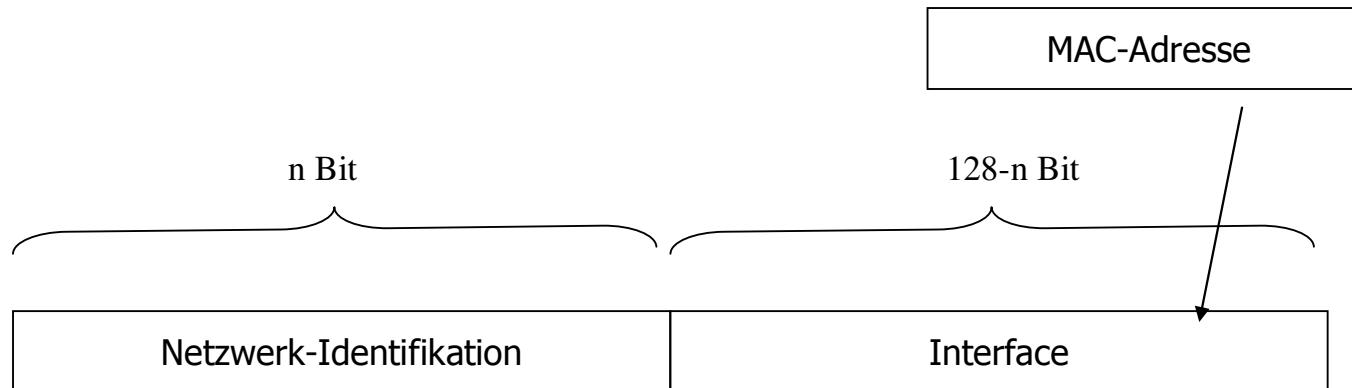
Einschub: Mapping IEEE 802.3-Adresse → EUI-64-Adresse (Schicht 2)

- IEEE EUI-64-Adresse ist länger, also nicht direkt abbildung
- 16 Bit 0b1111111111111110 = 0xFFFF werden zwischen der Firmenkennung und der Erweiterungskennung in die IEEE 802.3-Adresse eingefügt
- u-Bit wurde für IPv6 wegen einfacherer Schreibweise einer linken lokalen Adresse invertiert :
 - Beispiel: FE80::124 einfacher als FE80::200:0:0:124



IPv6-Adressaufbau: Netzwerk-Id

- MAC-Adresse wird als Interface-Identifikation übernommen
 - Sicherheitsproblem, konnte man leicht manipulieren
 - Umfangreiche Diskussionen dazu
 - Daher RFC 4941 (privacy Extensions), um zufällige Interface Identifier zu erzeugen
- Abbildung IEEE-803.3-Adresse auf IEEE EUI-64-Adresse

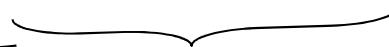


IPv6-Adressaufbau und Regeln

- Adressen-Notation mit 8 Gruppen zu je vier Hex-Zahlen abgetrennt durch Doppelpunkte, CIDR-Notation (x/y) auch zulässig

Beispiel:

8000:0000:0000:0000:**0123**:5555:89AB:CDEF



- Führende Nullen können in jeder Gruppe weggelassen werden und Gruppen mit lauter Nullen können durch einen Doppelpunkt ersetzt werden, aber „::“ nur an einer Stelle möglich:

8000::123:5555:89AB:CDEF



- IPv4-Adressen können mit speziellen Unicast-Adressen (Präfix ::FFFF/96) abgebildet werden (Mapping-Adressen)

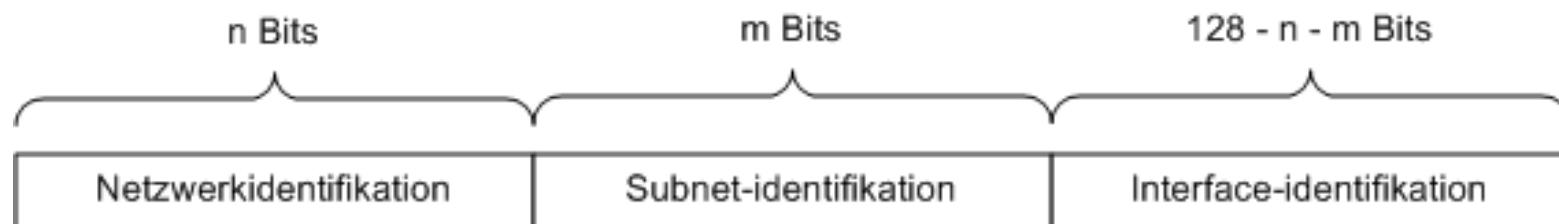
Beispiel: 192.168.0.1 → ::FFFF:C0A8:1

Besondere IPv6-Adressen, Sonderformen

- ::0 entspricht 0.0.0.0 in IPv4 (undefinierte Adresse)
 - Synonym: 0:0:0:0:0:0:0 oder ::/128
- ::1 entspricht der Loopback-Adresse 127.0.0.1 in IPv4
 - Synonym: 0:0:0:0:0:0:1 oder ::1/128
- FF00::/8 weist auf eine Multicast-Adresse hin
- FE80::/10 weist auf eine Link-Lokal-Adresse hin

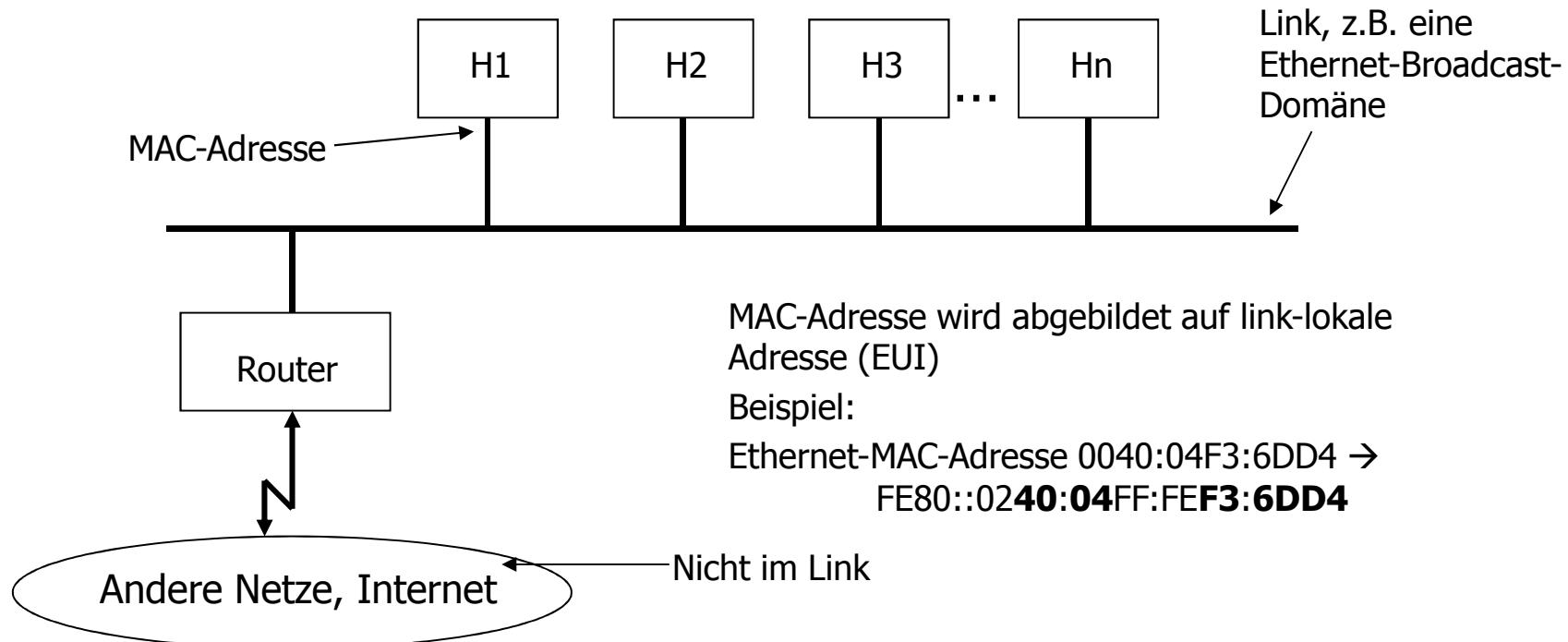
Globale Unicast-Adressen: Aufbau, Struktur

- Dienen dazu, einen Host (Knoten) im Internet **global** eindeutig zu identifizieren → **öffentliche** Adressen (wie Klasse A, B, C aus IPv4)!
- Hierarchiebildung möglich → Provider-/Netzbetreiberzuordnung
- Adresspräfix binär: 001 -> Hex: 20 .. 3F
- Eine Unicast-Adresse hat z.B. folgenden Aufbau:



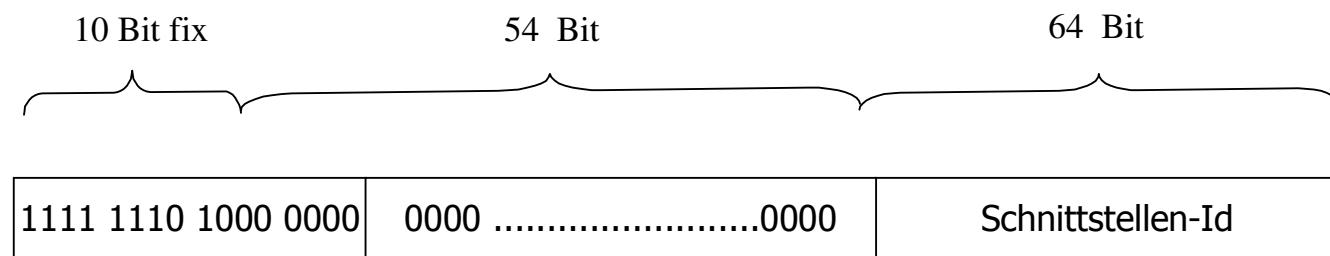
Verbindungslokale Adressbereiche (link-lokal)

- Link-Lokal bezieht sich auf das lokale Netz
- Ausbreitung nur in einem Teilnetz (Subnetz)
- Präfix der Adressen: 1111 1110 10 → FE80::/10 – FEBF::/10



Besondere IPv6-Adressen, Link-lokale Adresse

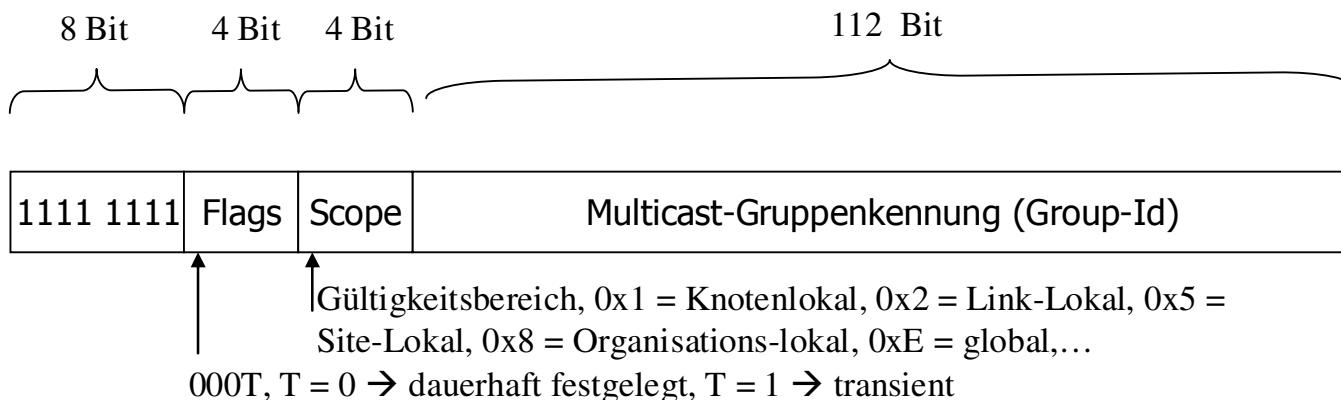
- **Aufbau einer link-lokalen Adresse**



Besondere IPv6-Adressen, Sonderformen

■ Multicast-Adressen

- Multicast-Adressen beginnen mit 0xFF
- Knotenlokale Adresse **FF01**:... → knotenlokal, Nachricht verlässt Knoten nicht
- Für das gleiche Link-Segment **FF02**:... → Nachricht verlässt Knoten, bleibt aber im Subnetz
- **FF0E**: Entspricht der IPv4-Broadcast-Adresse (limited)
- ...



Überblick

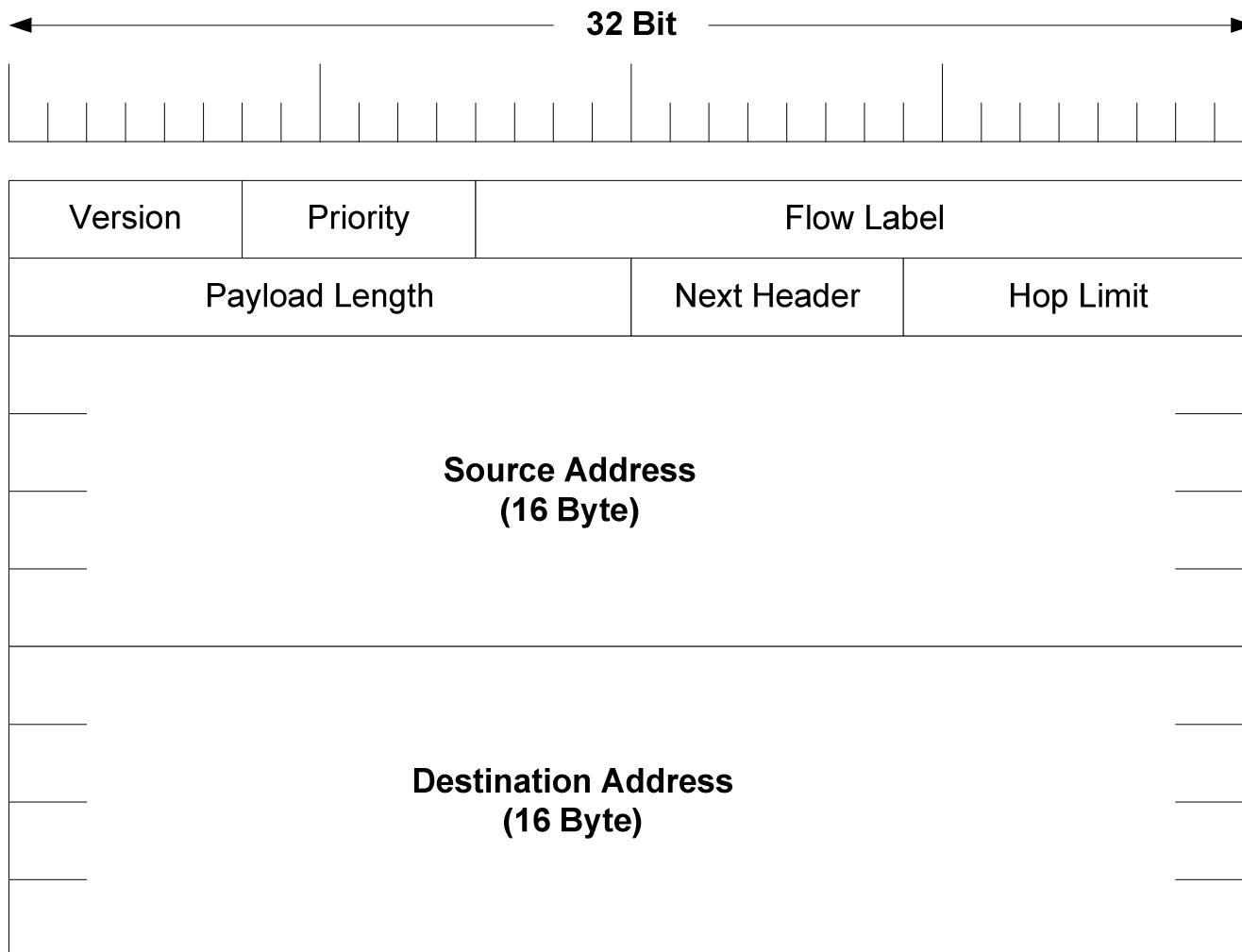
1. Steuerprotokolle

- ICMP
- ARP und RARP
- NAT
- DHCP

2. IPv6

- Grundlagen und Adressierung
- **IPv6-PDU**
- Automatismen, Neighbor Discovery

IPv6-Header

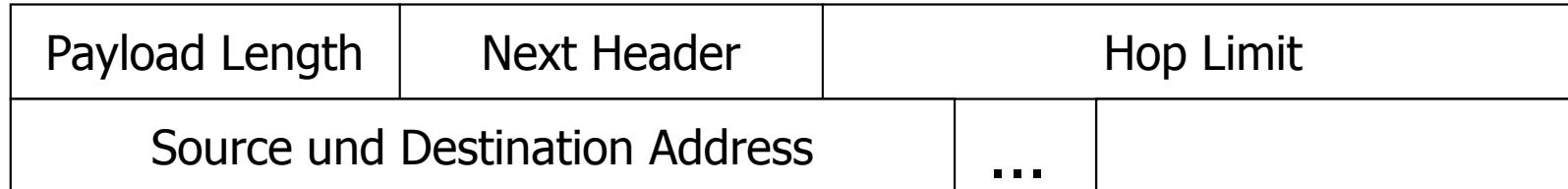


IPv6-Header

Version	Priorität	Flow Label (Flussmarke)
---------	-----------	-------------------------

- **Version** Versionsnummer des Internet-Protokolls (6)
- **Priorität:** auch **Traffic Class / DS** = Differentiated Service Field (neu) mit neuen Werten (RFC 2474, 2478)
 - Information für Router, interessant bei Überlastsituationen
 - stoßartiger Verkehr (ftp, NFS) → hoher Durchsatz
 - interaktiver Verkehr (telnet) → geringe Verzögerung
 - Verkehrsarten ohne Staukontrolle (z.B. für Videoanwendungen)
- **Flussmarke:** Identifikation des Flusses, falls ungleich 0
 - Zweck: Zusammengehörige Datenflüsse (Video/Audio) auf Netzebene speziell behandeln
 - Quelladresse+Zieladresse+Flussmarke kennzeichnen einen Fluss
 - Flussmarken werden im Quellknoten in die IPv6-PDU eingetragen

IPv6-Header



- **Payload Length:** Nutzdatenlänge **ohne** die 40 Bytes des IPv6-Headers, es gibt aber auch Jumbo-Pakete
- **Next Header:** Verweis auf ersten Erweiterungs-Header
 - Letzter Header verweist auf Protokolltyp der nächst höheren Schicht (siehe IPv4-Feld **Protokoll**)
- **Hop Limit:** Verbleibende Lebenszeit des Pakets in Hops
 - Jeder Router zählt Hop Limit um 1 herunter
 - Entspricht dem TTL-Feld in IPv4
 - Name entspricht jetzt der eigentlichen Nutzung im Internet
- **Source und Destination Adresse:** IPv6-Adressen der Quelle und des Ziels

IPv6-Header, Erweiterungs-Header

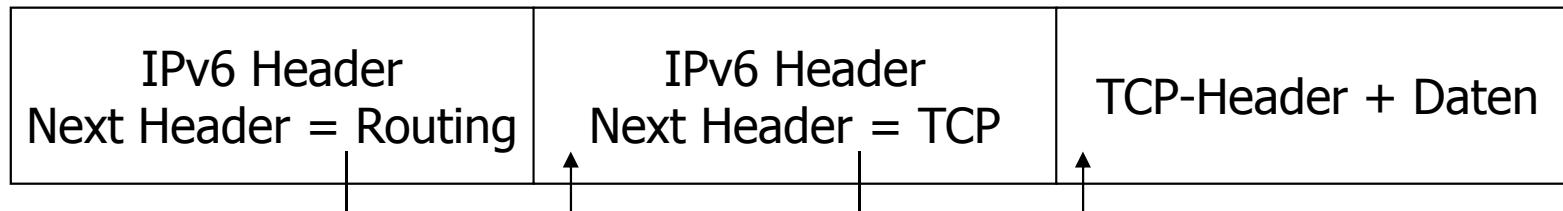
- Kodierung im Next-Header-Feld
 - EGP = 0x08
 - Routing = 0x2B
 - Fragment = 0x44
 - TCP = 0x06 ...

Erweiterungs-Header	Beschreibung
Optionen für Teilstrecken (Hop-by-Hop)	Verschiedene Informationen für Router
Routing	Definition einer vollen oder teilweisen Route
Fragmentierung	Verwaltung von Datagrammfragmenten
Authentifikation	Echtheitsüberprüfung des Senders
Verschlüsselte Sicherheitsdaten	Informationen über den verschlüsselten Inhalt
Optionen für Ziele	Zusätzliche Informationen für das Ziel

IPv6-Header, Erweiterungs-Header

- Header und Erweiterungs-Header sind miteinander **verkettet**, jeder Typ **max. einmal**
- Die Erweiterungen werden **nicht** in den Routern bearbeitet, nur in den Endsystemen
- Eine **Ausnahme**: Routing-Erweiterungs-Header
- Reihenfolge der Header ist festgelegt

- Beispiel eines IPv6-Headers mit einem Erweiterungs-Header und einer anschließenden TCP-PDU



IPv6-Header, Erweiterungs-Header

- Der **Routing-Header** dient der Quelle zur Festlegung des Weges bis zum Ziel

Next Header	Header Ext. Länge	Routing Typ	Verbl. Segmente
1-24 Adressen			

...

- Next Header:** siehe vorne
- Header Ext. Länge:** Länge des Routing-Headers
- Routing Typ:** Gibt den Typ der Routing-Headers an
- Verbleibende Segmente:** Anzahl der folgenden Adressen, die besucht werden müssen

IPv6-Header, Erweiterungs-Header

- Der **Fragmentierungs-Header** wird verwendet, um größere Dateneinheiten zu senden, als zugelassen
 - PDU-Länge > MTU des Pfades (MTU = Maximum Transmission Unit)
 - Minimum auch in IPv6 576 Bytes
- Fragmentierung erfolgt bei IPv6 nur im Quellknoten, Router fragmentieren nicht → geringe Routerbelastung

Next Header	reserviert	Fragment Offset	00M
Identifikation			

- Fragment Offset:** Position der Nutzdaten relativ zum Beginn der PDU (Ursprungs-Dateneinheit) → 13 Bit (wie IPv4)
- Identifikation:** Id der PDU (wie IPv4)
- M:** More-Flag, M=1 → weitere Fragmente folgen (wie IPv4)

IPv6, Sicherheitsaspekte

- Im Gegensatz zu IPv4 sind in IPv6 schon Sicherheitsmechanismen im Protokoll spezifiziert (siehe IPv4+**IPsec**)
 - Authentifizierung
 - Verschlüsselung
- **MD5-Algorithmus** (Message Digest) kann zur Authentifizierung der Partner verwendet werden
- Verschlüsselung des Nutzdatenteils wird mit einer Variante des **DES- oder AES-Verschlüsselungsalgorithmus** unterstützt
 - DES = Data Encryption Standard
 - AES = Advanced Encryption Standard
 - Symmetrisches Verschlüsselungsverfahren

IPv6, Flussmarken

- Ziel: Aufbau von **Pseudoverbindungen** zwischen Quelle und Ziel mit QS-Merkmalen wie Verzögerung und Bandbreite
 - Ressourcenreservierung
 - Datenströme für Echtzeitanwendungen
- Flexibilität von Datagramm-Netzen kombiniert mit virtuellen Verbindungen
- Ein „Fluss“ wird durch Quell- und Zieladresse sowie einer Flussnummer identifiziert
- Router führen eine Sonderbehandlung durch
- Noch in der Experimentierphase!

Überblick

1. Steuerprotokolle

- ICMP
- ARP und RARP
- NAT
- DHCP

2. IPv6

- Grundlagen und Adressierung
- IPv6-PDU
- **Automatismen, Neighbor Discovery**

Autokonfiguration: Einige wichtige Features

- **Selbstkonfiguration:** Host konfiguriert seine eigene Adresse dynamisch (kein ARP mehr notwendig):
 - Die **dynamische Adress-Auflösung** für Layer-2-Adressen wie es heute im **ARP-Protokoll** abgewickelt wird
- **Router Discovery:** Das Auffinden von Routern im gleichen Link (Subnetz)
- **Parameter Discovery:** Die dynamische Zuordnung von Konfigurationsparametern wie der maximalen MTU und dem Hop-Limit an IPv6-Endsysteme
- Die automatische **IP-Adress-Konfiguration** für Interfaces zur Laufzeit
- Die Suche nach dem optimalen MTU zwischen Sender und Empfänger (**Path MTU Discovery**)

Einige wichtige Features

Beispiel: Router-Discovery

- Wenn ein Endsystem seinen nächsten Router sucht, sendet es eine ***Router-Solicitation-Nachricht*** über Multicast an die Adresse **FF02::2**
- Router antworten mit einer ***Router-Advertisement-Nachricht***
- Damit unterstützt das ND-Protokoll das Auffinden des verantwortlichen Routers zur Laufzeit → DHCP kann auch wegfallen
- Mehrere Router können aktiv sein
- Das ND-Protokoll nutzt zur Abwicklung seiner Aufgaben einige ICMPv6-Nachrichten

Einige wichtige Features

Beispiel: Parameter-Discovery

- Netzwerkparameter werden vom Host zum Startzeitpunkt auch über ***Router-Solicitation-Nachricht*** besorgt (DHCP-Aufgaben)
- Nachricht geht an Multicast-Adresse **FF02::2**
- Ein Router antwortet mit einer ***Router-Advertisement-Nachricht*** an die Link-Adresse des Endsystems
- Folgende Parameter kann eine *Router-Advertisement*-Nachricht u.a. übertragen:
 - *Max-Hop-Limit*: Dies ist der Wert „Hop-Limit“ der in die IPv6-PDUs eingetragen wird
 - *Retransmission-Timer*: Zeit in Millisekunden, die seit dem Absenden der *Solicitation*-Nachricht ablaufen darf, bevor wiederholt wird
 - ...
 - Über ein *Optionsfeld* wird z.B. vom Router auch die *MTU-Size* übermittelt

Rückblick und Weiterführendes

1. Steuerprotokolle

- ICMP
- ARP und RARP
- NAT
- DHCP

2. IPv6

- Grundlagen und Adressierung
- IPv6-PDU
- Automatismen, Neighbor Discovery

Was ist noch interessant:

- RSVP, IGMP, ...
- Mobiles Routing
- Sicherheit: IPsec-Protokolle und VPNs,...
- Migration IPv4 → IPv6, ...

Datenkommunikation

Anwendungsschicht

Fallstudien DNS, HTTP, E-Mail,...

Wintersemester 2012/2013

Überblick

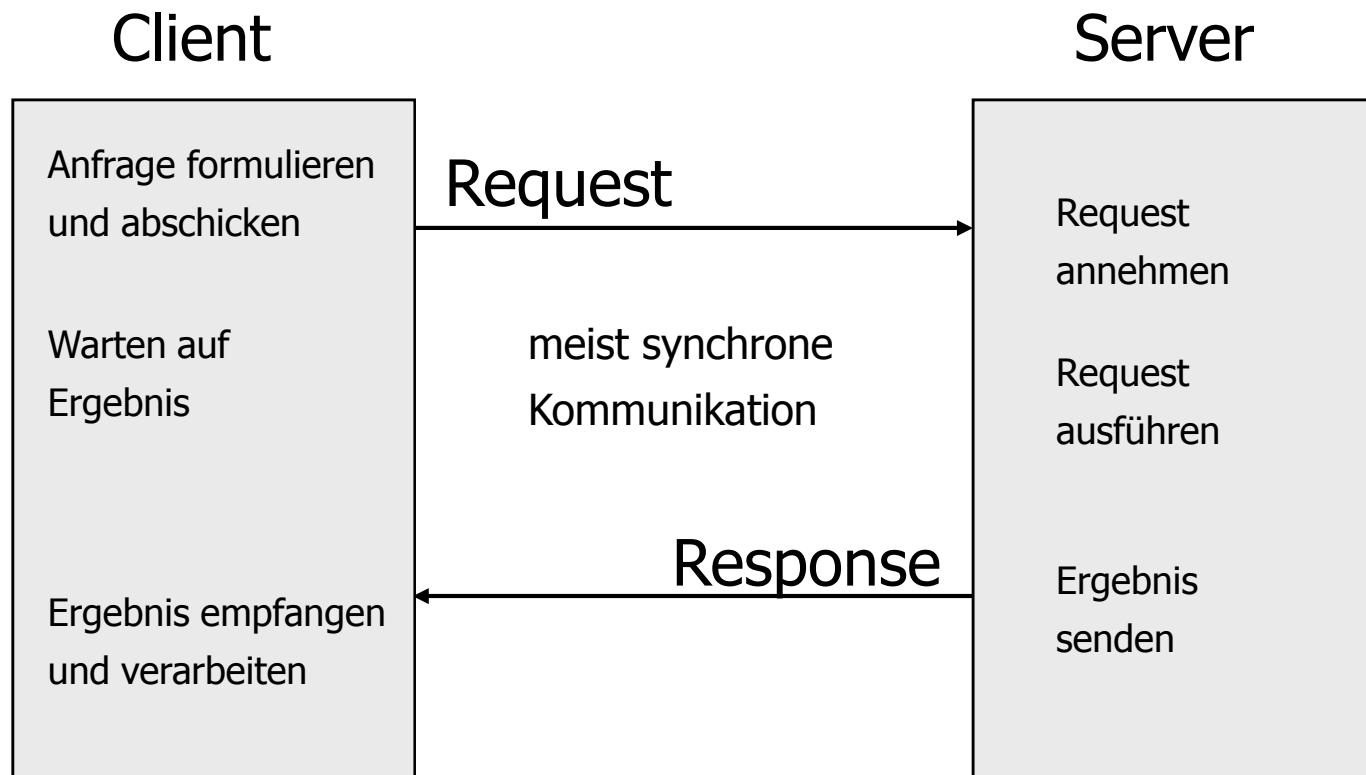
1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

- 1. Client-/Server versus Peer-to-Peer (P2P)**
2. Domain Name System
3. HTTP
4. E-Mail
5. Multimedia-Protokolle

Client-/Server-Kommunikation

- Klare Rollenaufteilung

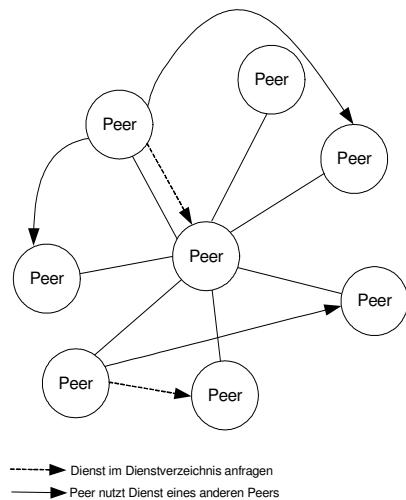


Peer-to-Peer-Architekturen (1)

- Gleichberechtigte Peers sind in der Rolle von Server und Client (meist im Internet)
- Beispiele für P2P-Systeme/-Protokolle:
 - **BitTorrent**: Filesharing-System im Internet
 - **Skype**: VoIP-System
 - **Bitcoin**: Digitales Cash-System
 - **Napster**: Austausch von Musik-Dateien (rechtl. Probleme)
 - **Gnutella**: Filesharing-Netzwerk im Internet (nicht mehr relevant)
 - **KaZaa**: Internet-Tauschbörse für Musikdateien, Videos, Textdokumente und Bilder

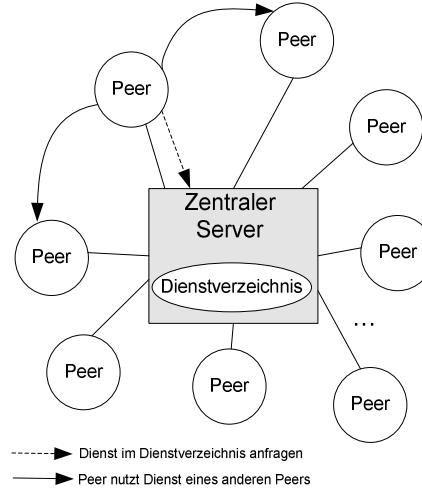
Peer-to-Peer-Architekturen (2)

- Verschiedene Varianten: pur, hybrid, Superpeer



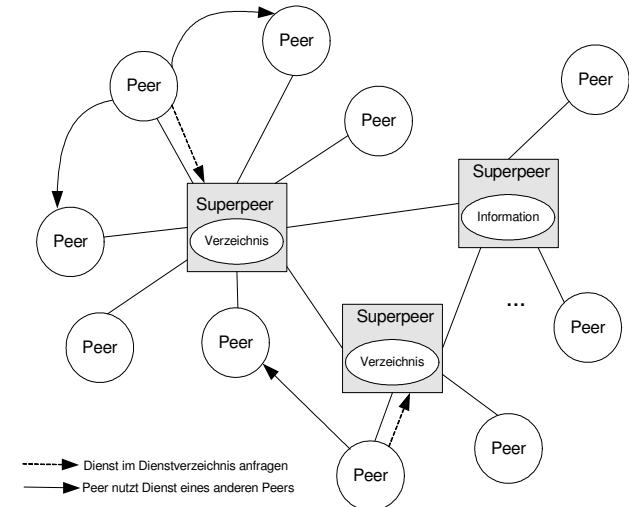
Pur

Beispiel: Gnutella



Hybrid

Beispiele: Napster
BitTorrent



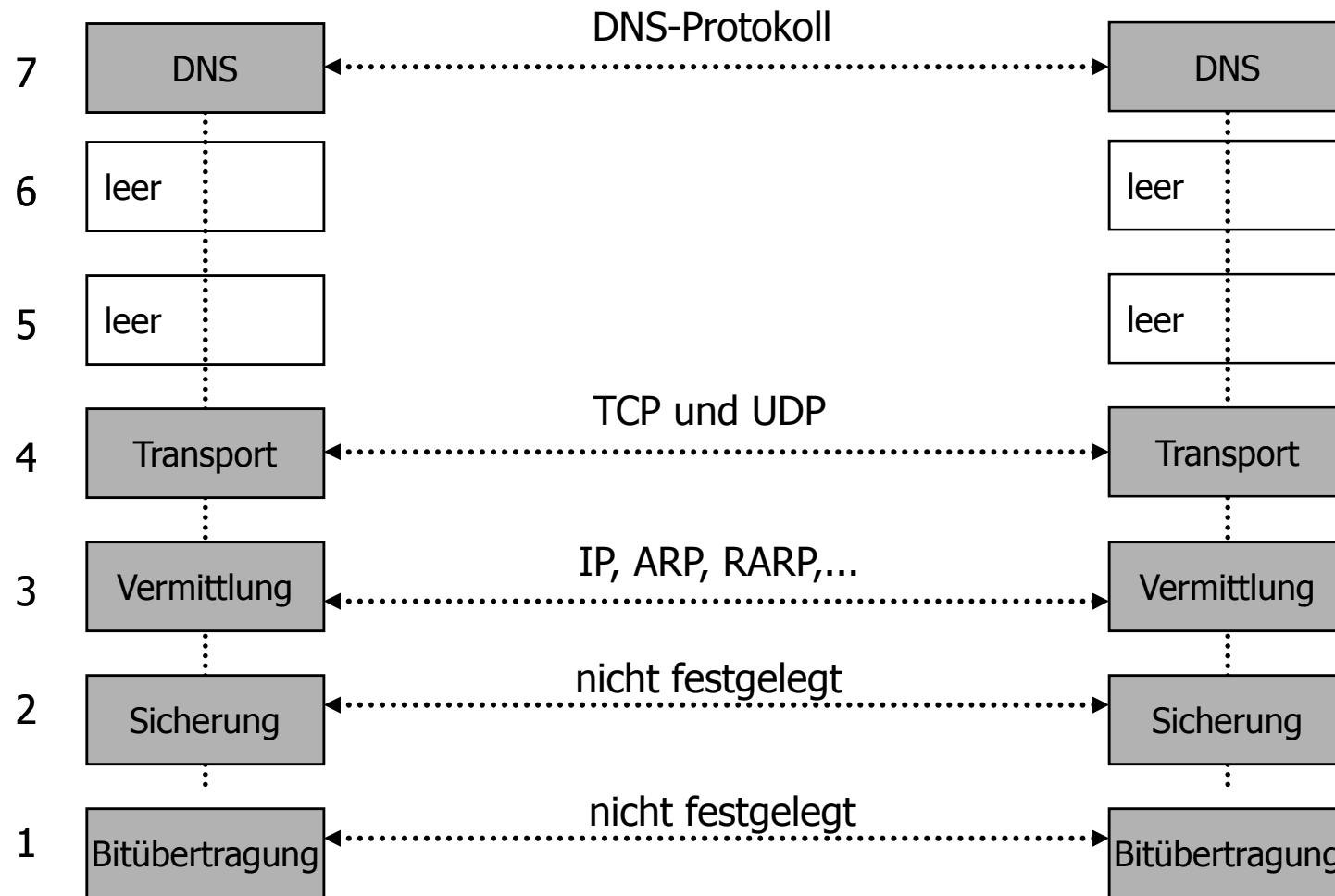
Superpeer:

Beispiele: Skype, KaZaa

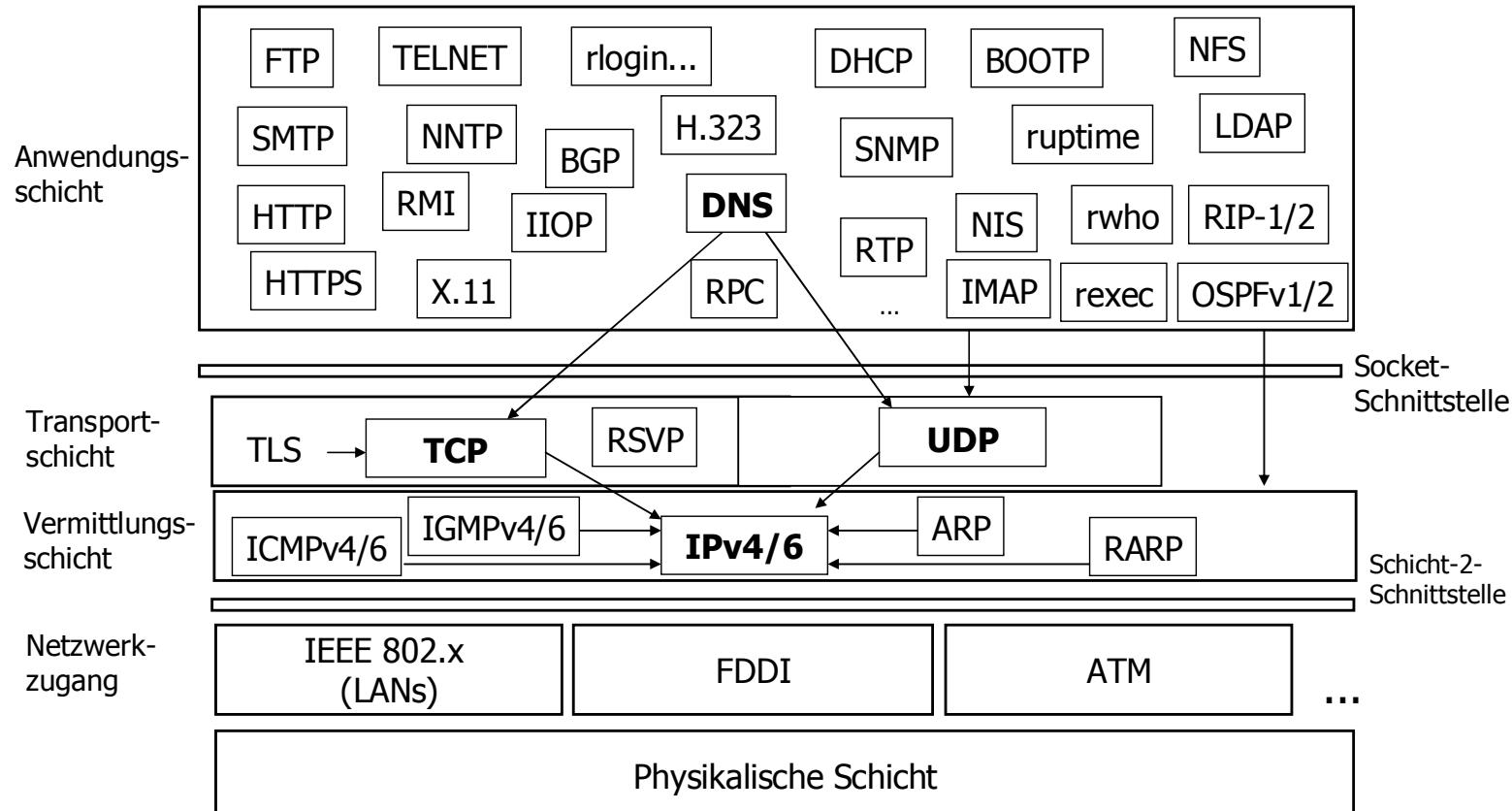
Überblick

1. Client-/Server versus Peer-to-Peer (P2P)
- 2. Domain Name System**
3. HTTP
4. E-Mail
5. Multimedia-Protokolle

Überblick Einordnung in die TCP/IP-Protokollfamilie



Übersicht Einordnung in die TCP/IP-Protokollfamilie



DNS-Port: 53 TCP/UDP

Domain Name System (DNS): Hintergrund

- Im ARPANET mit einigen hundert Hosts war die Verwaltung der Adressen noch einfach
 - Es gab eine Datei **hosts.txt** auf einem Verwaltungsrechner, die alle IP-Adressen enthielt (flacher Namensraum)
 - Die Datei wurde nachts auf die anderen Hosts kopiert
- Als das Netz immer größer wurde, führte man DNS ein
- DNS dient prinzipiell der **Abbildung von Hostnamen auf eMail- und IP-Adressen**
- DNS ist ein Internet **Directory Service**

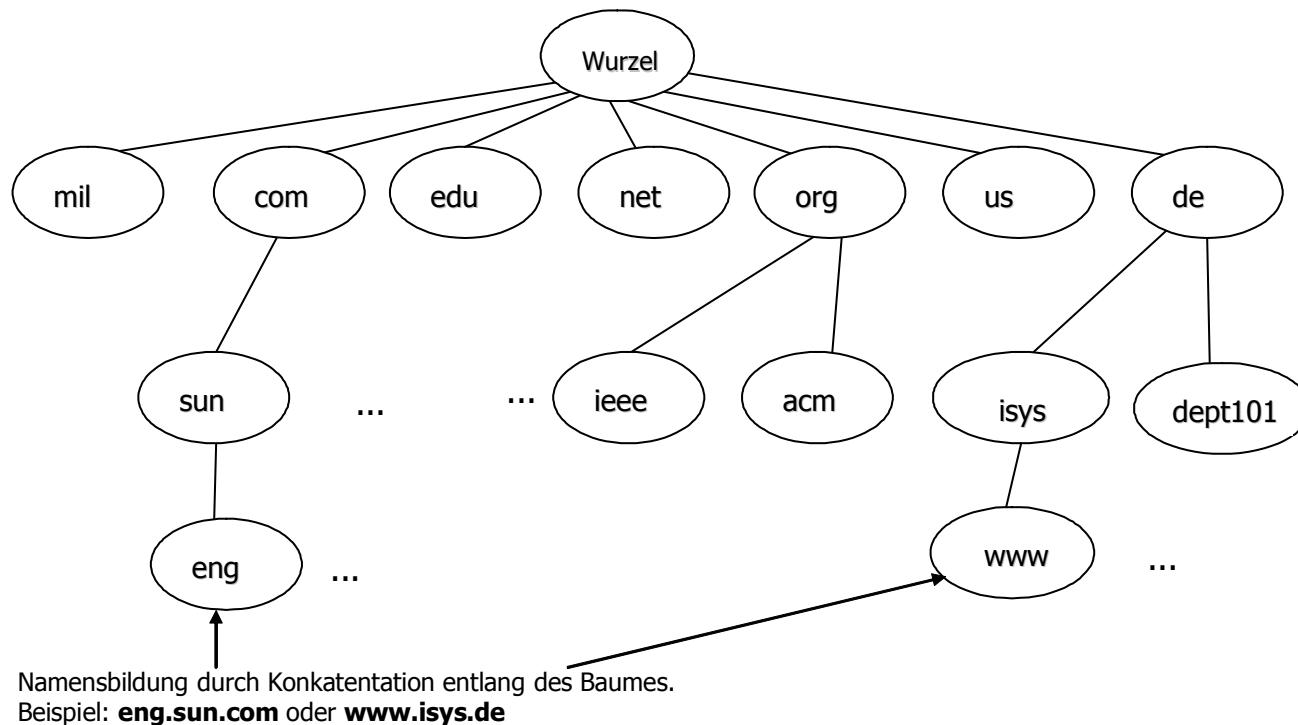
Achtung: DNS nicht mit Routing verwechseln!

Domänen und Subdomänen

- DNS ist in den RFCs 1034 und 1035 definiert
- DNS ist ein hierarchischer Namensverzeichnis für IP-Adressen (Adressbuch des Internets)
- DNS verwaltet eine **Datenbank**, die sich über zahlreiche Internet-Hosts erstreckt
- Konzeptionell ist das Internet in **mehrere hundert Domänen** aufgeteilt
- Die Domänen sind wiederum in **Teildomänen (Subdomains)** untergliedert, usw.

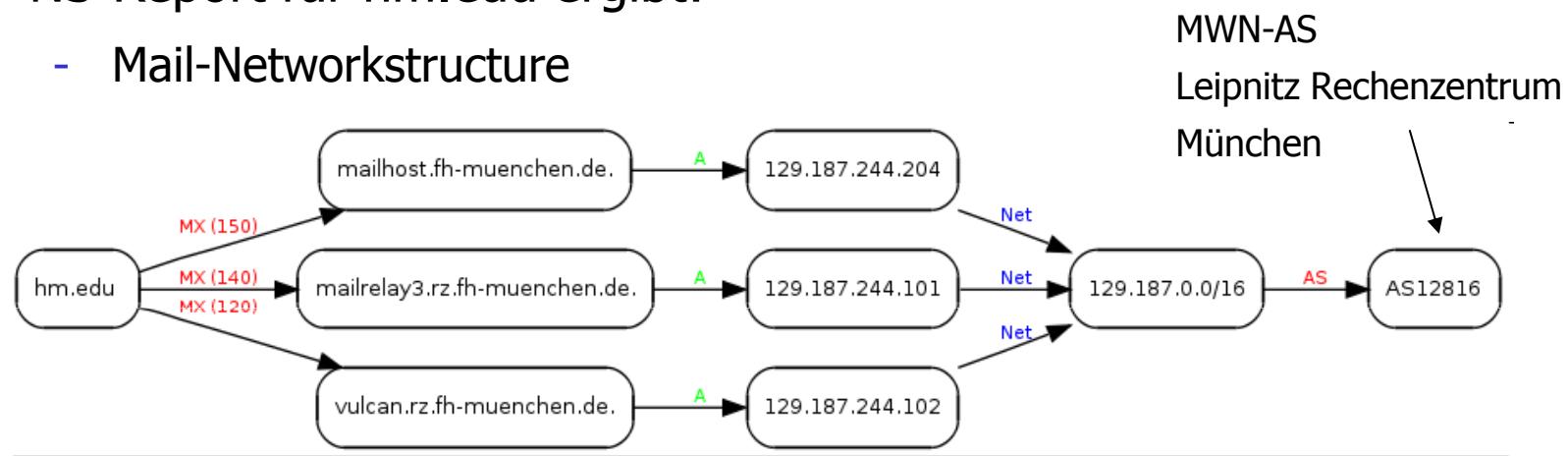
DNS-Namensraum

- Weltweit verteilter Namensraum
- **Baum**, an dessen Blättern dann die Hosts hängen
(rein organisatorisch, nicht physikalisch)

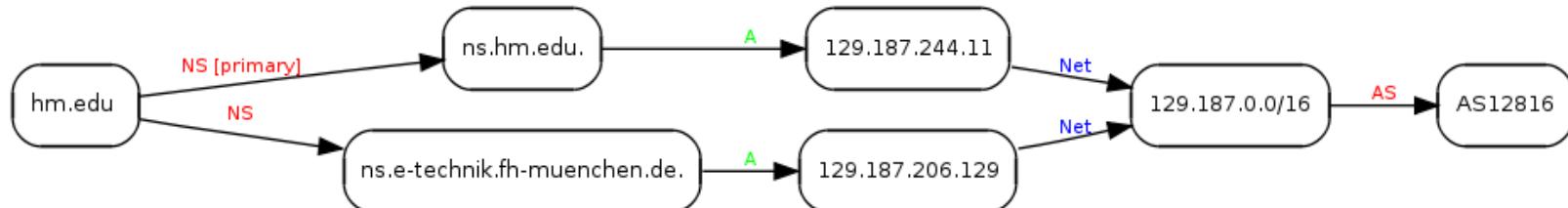


Einschub: Suche nach Domain-Information (1)

- Beispieldienst über <http://serversniff.net> (oder .de)
- NS-Report für hm.edu ergibt:
 - Mail-Networkstructure



- Nameserver-Networkstructure



Einschub: NS-Report für hm.edu (Auszug)

NS-Record(s) for domain hm.edu:

ns.fh-muenchen.de. 129.187.244.11

ns.e-technik.fh-muenchen.de. 129.187.206.129

MX-Servers for hm.edu:

150 mailhost.fh-muenchen.de. 129.187.244.204

140 mailrelay3.rz.fh-muenchen.de. 129.187.244.101

120 vulcan.rz.fh-muenchen.de. 129.187.244.102

SOA-Record for hm.edu:

ns.hm.edu. hostmaster.hm.edu. 2009121462 10800 1800 3600000 86400

Serial: 2009121462

Refresh: 10800

Retry: 1800

Expire: 3600000 (1000 hours or 42 days)

TTL: 86400

Recursive-Queries:

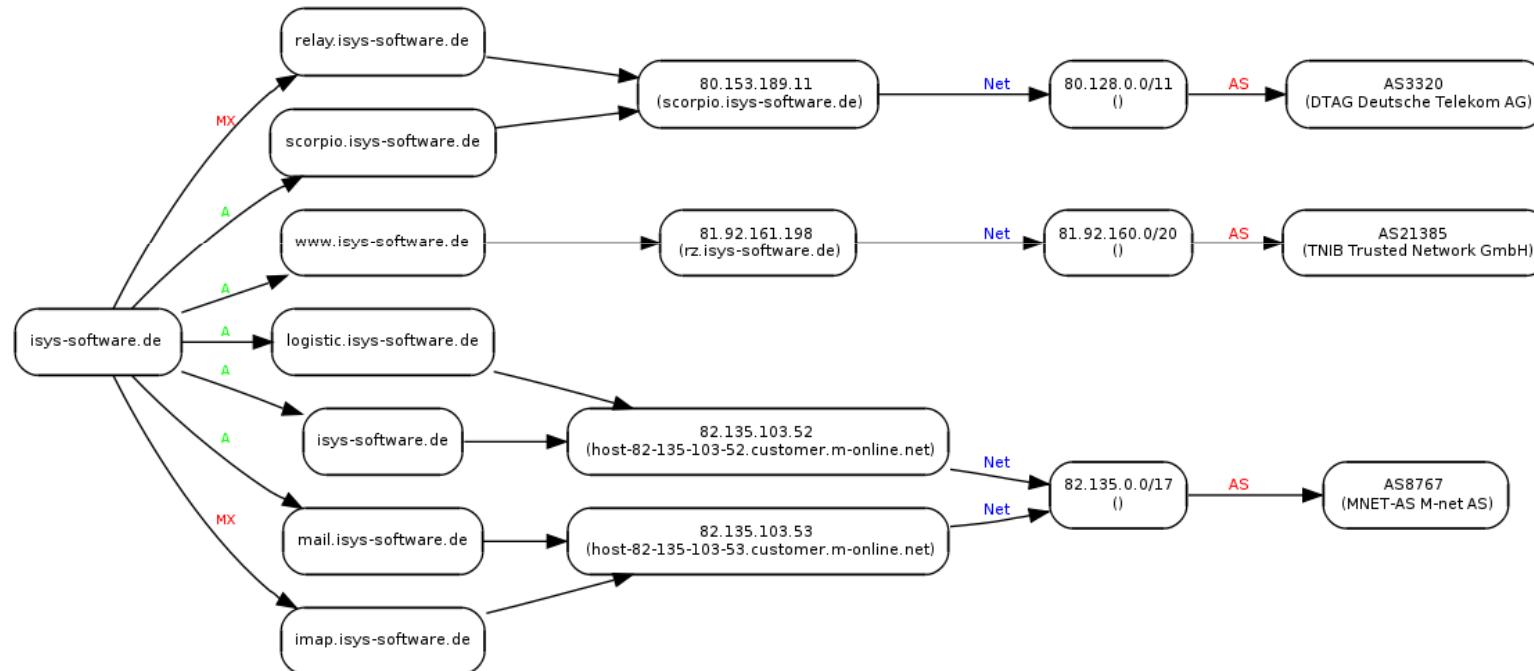
ns.e-technik.fh-muenchen.de. YES - recursive queries allowed!

ns.hm.edu. YES - recursive queries allowed!

...

Einschub: Suche nach Domain-Information (2)

- Beispieldienst über <http://serversniff.de>
- Domainreport für [isys-software.de](http://www.isys-software.de) ergibt:



Weitere Info zur Domäne über
<http://who.is/whois-de/ip-address/isys-software.de/>

Nicht gesponserte Domains

- **ICANN** (Internet Corporation for Assigned Names and Numbers) pflegt diese „nicht-gesponserten“ Domains
- Man unterscheidet:
 - Geographische oder länderspezifische (**Country-Code, ccTLDs**) Domains wie de, at, us, uk, gb, usw.
 - Für jedes Land ist nach ISO 3166 ein Code mit zwei Buchstaben vorgesehen.
 - Es gibt derzeit über 200 ccTLDs.
 - Für die Europäische Union wurde .eu als Gemeinschaft ebenfalls dieser Art von Domains zugeordnet, obwohl .eu als eine Ausnahme behandelt wird (Liste der Ausnahmen zu ISO 3166).
 - Allgemeine Domains für Organisationen (**generic** oder **gTLDs**)
 - **Infrastruktur-Domains** als Sonderfall (spezielle Domain .arpa)
- Es gibt darüber hinaus „gesponserte“ Domains

Generic TLDs

Domain	Beschreibung
com	Kommerzielle Organisationen (sun.com, ibm.com, ...)
edu	Bildungseinrichtungen (hm.edu)
gov	Amerikanische Regierungsstellen (nfs.gov)
mil	Militärische Einrichtung in den USA (navy.mil)
net	Netzwerkorganisationen (nsf.net)
org	Nichtkommerzielle Organisationen
int	Internationale Organisationen (nato.int)
bis	Business, für Unternehmen
info	Informationsanbieter
arpa	TLD des ursprünglichen Arpanets, die heute als sog. <i>Address and Routing Parameter Area</i> verwendet wird (auch als "Infrastruktur-Domain" bezeichnet).
pro	Professions, Berufsgruppen der USA, Deutschland und des Vereinigten Königreichs

Gesponserte Domains

- Werden von unabhängigen Organisationen in Eigenregie kontrolliert und finanziert
- Eigene Richtlinien für die Vergabe von Domainnamen
- Zu den „gesponserten“ TLDs gehören:
 - **.aero:** Aeronautics, für in der Luftfahrt tätige Organisationen, weltweiter Einsatz
 - **.coop:** Steht für *cooperatives* (Genossenschaften), weltweiter Einsatz
 - **.info:** Informationsanbieter, weltweiter Einsatz
 - **.int:** Internationale Regierungsorganisationen (Beispiel www.nato.int oder www.eu.int)
 - **.mobi:** Darstellung von Webseiten speziell für mobile Endgeräte, weltweiter Einsatz ...

DNS-Namensraum und DNS-Datenbank

- DNS ist eine baumförmige **weltweite Vernetzung** von Name-Servern
- DNS bildet eine weltweit verteilte Datenbank (**DNS-Datenbank**)
- Jeder Knoten im DNS-Baum stellt eine Domäne dar und kann mit einem Verzeichnis in einem Dateisystem verglichen werden
- Jeder Knoten hat einen Namen

Root-Name-Server

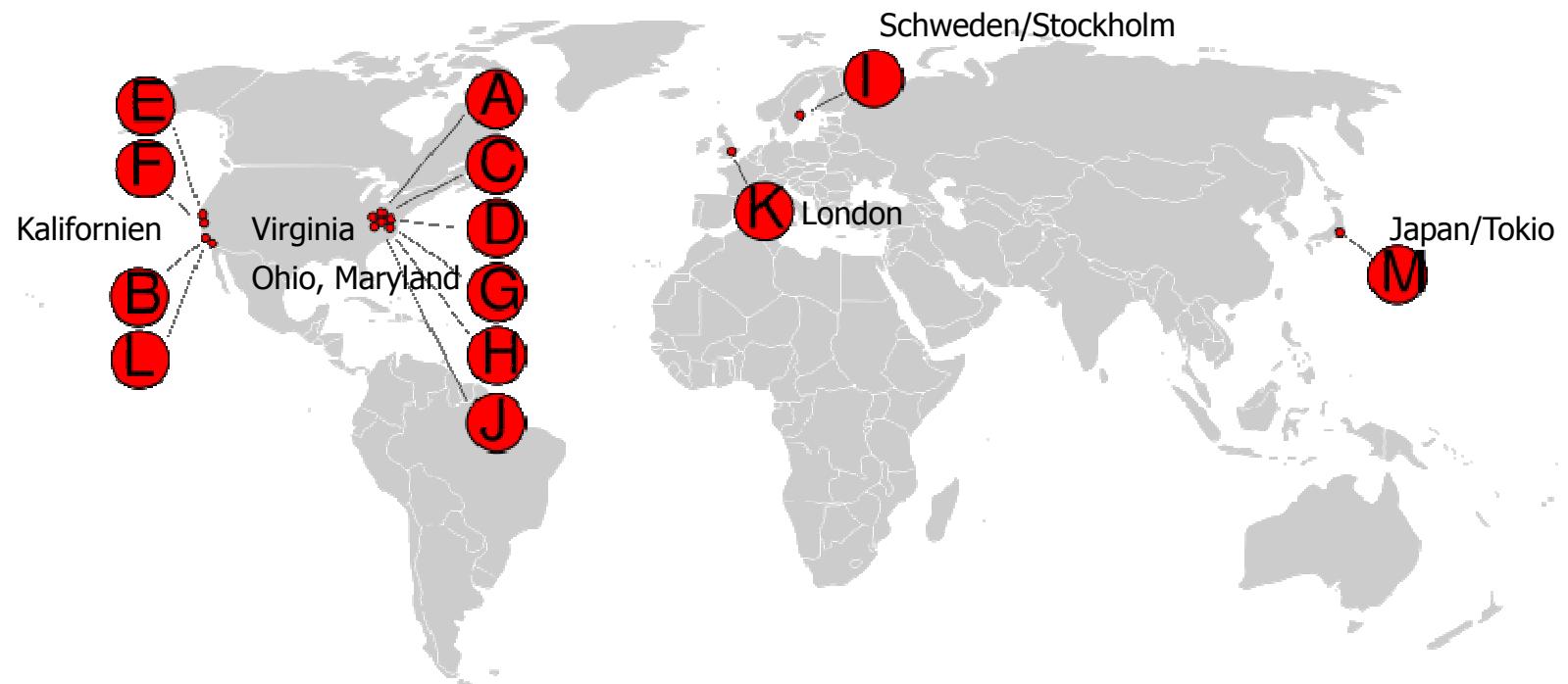
- Es gibt derzeit **weltweit 13 sog. Root-Name-Server (A...M)**
 - 10 in Nordamerika, 1 in Stockholm, 1 in London, 1 in Tokio
- Root-Name-Server geben Informationen über die weitere Suche
- Ein Root-Name-Server
 - verfügt über eine Referenz-Datenbank aller von der ICANN freigegebenen Top-Level-Domains (TLD) und
 - die wichtigsten Referenzen auf die sog. Top-Level-Domain-Server
- Ein Root-Name-Server kennt somit immer einen DNS-Server, der eine Anfrage beantworten kann

Root-Name-Server

- DNS erlaubt DNS-PDUs (UDP) bis zu einer Größe von 512 Byte
 - Daher max. 13 DNS-Records in einer PDU
 - EDNS (Extended DNS) als DNS-Erweiterung ermöglicht die zehnfache Größe (RFC 2671)
- A und J haben **zentrale** Bedeutung. Sie verteilen die Datenbasis für alle anderen Root-Name-Server zweimal täglich
- Root-Name-Server bestehen aus z.T. **vielen Einzelservern**:
 - Root-Name-Server F besteht aus 33 Serverrechnern (2009)
 - Root-Name-Server K besteht aus aus 16 Serverrechnern (2009)
- Diese Root-Name-Server sind auf der ganzen Welt verteilt und einige nutzen heute einen **Anycast-Mechanismus** (eine IP-Adresse) zur Datenverteilung

Root-Name-Server (früher)

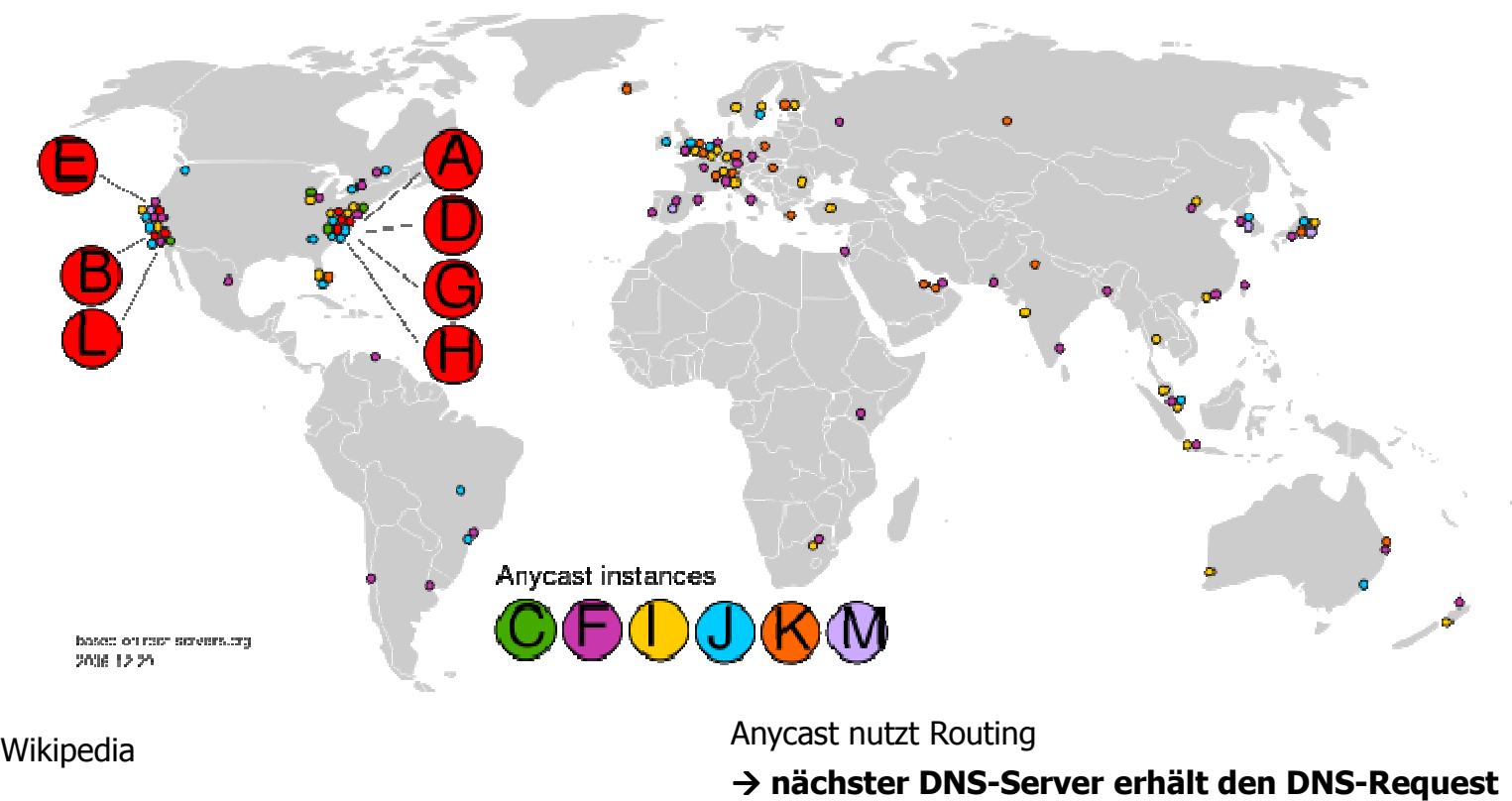
- Single Point of Failure: Synchronisation erfolgte über Root-Name-Server A (VeriSign = Amerikanisches Unternehmen und Zertifizierungsstelle für digitale Zertifikate)
- Viele DoS-Angriffe



Quelle: Wikipedia

Root-Name-Server (heute, mit Anycast)

- Mehr Ausfallsicherheit, in 2009 mehr als 120 Root-Name-Server verfügbar
- Keine Synchronisation mit A → Kein Single Point of Failure



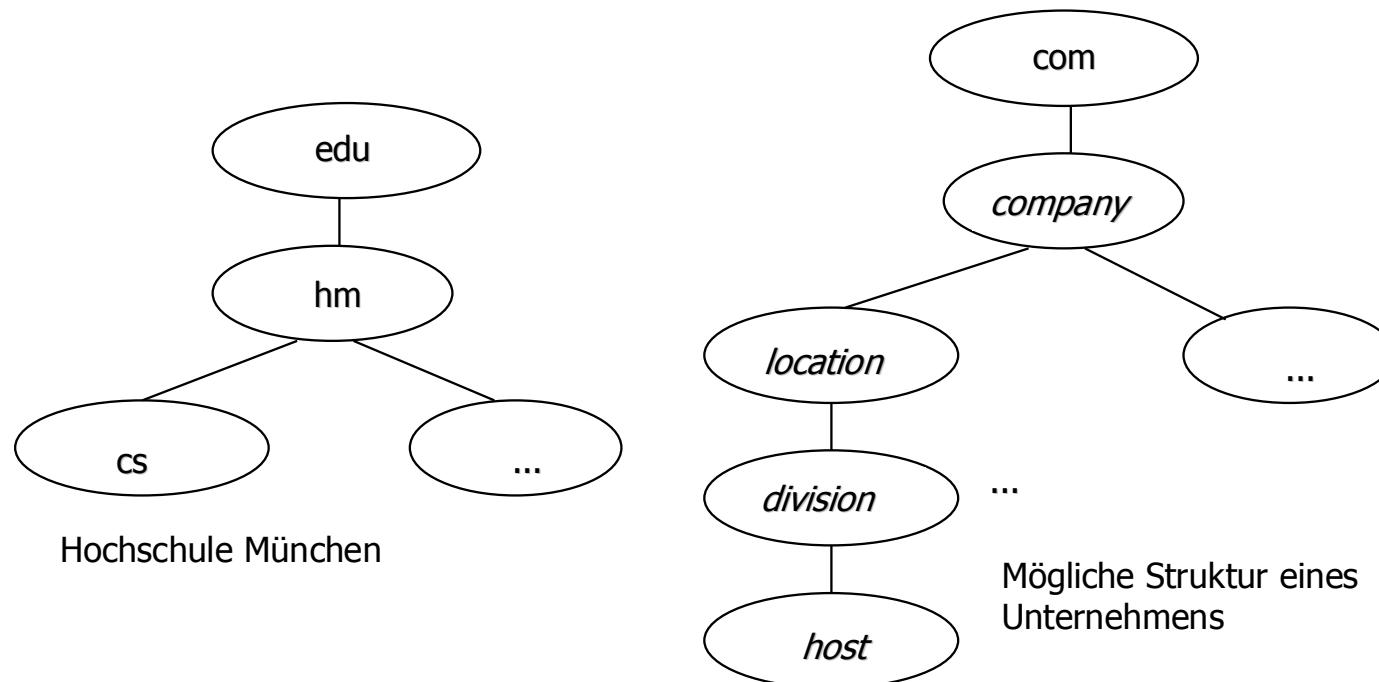
Root-Name-Server

Root-Server	Alter Name	Betreiber	Ort
A	ns.internic.net	VeriSign	Dulles, Virginia, USA
B	ns1.isi.edu	ISI	Marina Del Rey, Kalifornien, USA
C	c.psi.net	Cogent Communications	Nutzt Anycast
D	terp.umd.edu	University of Maryland	College Park, Maryland, USA
E	ns.nasa.gov	NASA	Mountain View, Kalifornien, USA
F	ns.isc.org	ISC	Nutzt Anycast
G	ns.nic.ddn.mil	U.S. DoD NIC	Columbus, Ohio, USA
H	aos.arl.army.mil	U.S. Army Research Lab	Aberdeen Proving Ground, Maryland, USA
...			

Quelle: Wikipedia

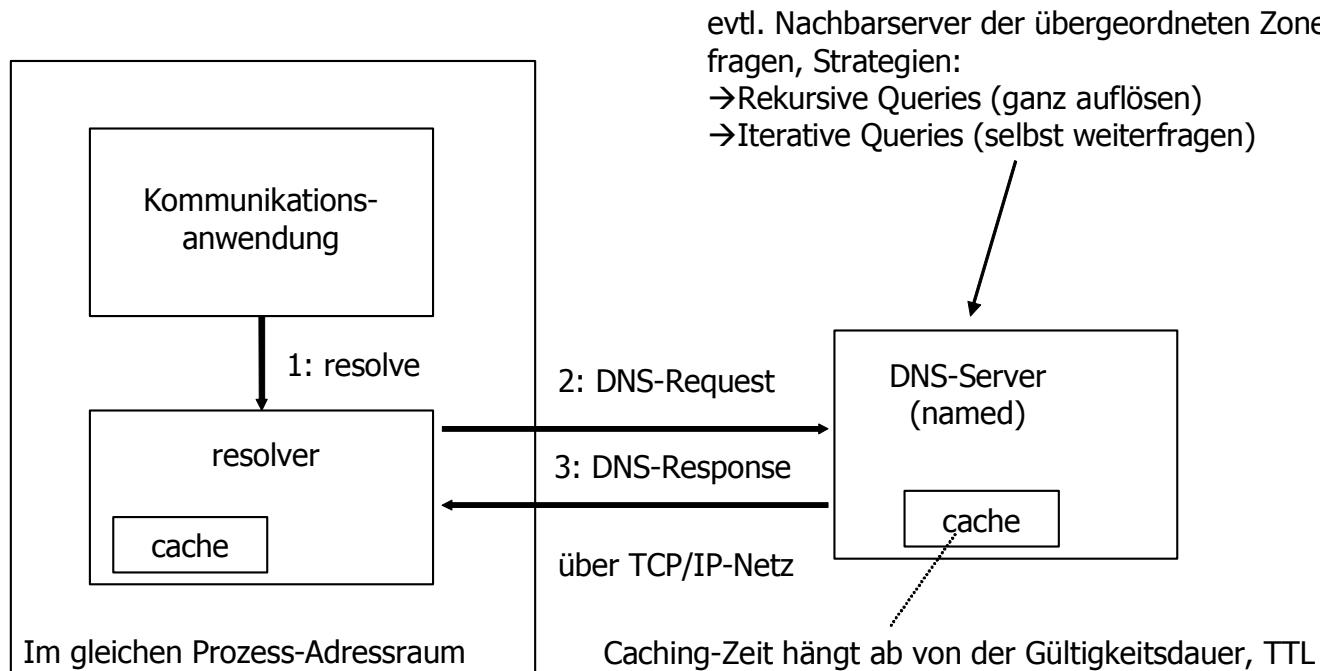
DNS-Namensraum

- **Beispiele** für Domänen mit untergeordneten Subdomänen



Adressauflösung

- Eine Anwendung, die eine Adresse benötigt, wendet sich lokal an einen **Resolver** (Library), der eine Anfrage an einen DNS-Server (unter Unix z.B. **named**) richtet.



Name Server und Zonen

- DNS-Name-Server verwaltet jeweils **Zonen** des DNS-Baums, wobei eine Zone an einem Baumknoten beginnt und die darunter liegenden Zweige beinhaltet
- Ein Name-Server (bzw. die entspr. Organisation) kann die Verantwortung für Subzonen an einen weiteren Name Server **delegieren**
- Die Name-Server kennen jeweils ihre Nachbarn in der darunter- oder darüberliegenden Zone
- Informationen des DNS werden in sog. **Resource Records** verwaltet

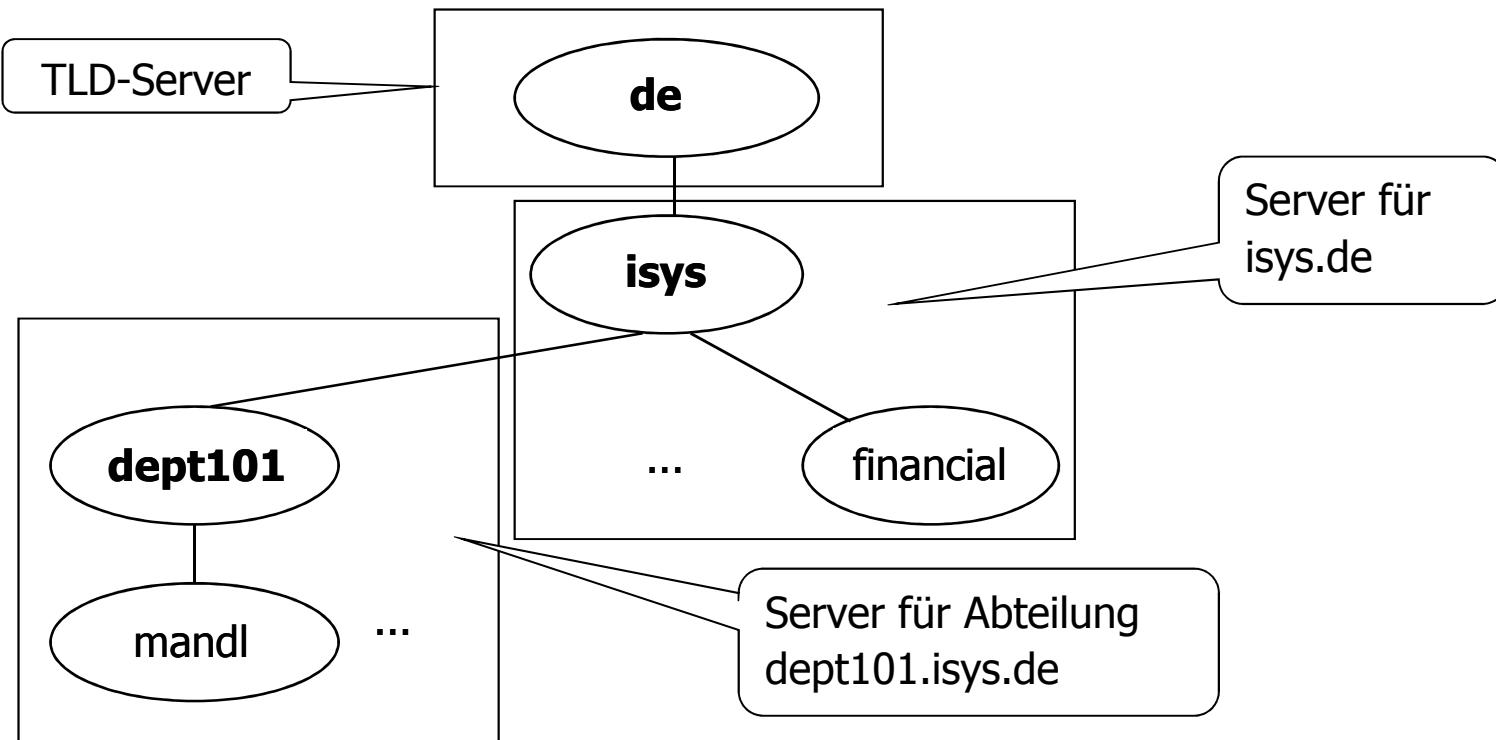
Autoritative Name Server und andere...

- **Autoritativer** Name Server:
 - Verantwortlich für eine Zone
 - Mind. einer muss in Zone sein (Primary Nameserver)
 - Kennt alle Adressen der Zone genau
 - Siehe Konfigurationsdatei (**SOA-Resource-Record, = „Start of Authority“**, Angaben zur Verwaltung einer Zone)
- **Nicht-autoritativer** Name Server:
 - Bezieht Informationen von anderen Servern
 - Verfalldatum über TTL-Parameter geregelt
- **Zonentransfer** zwischen Primary und Secondary Server

Name Server und Zonen

■ Beispiel für die Delegation:

- Root-Name-Server kennt isys-Server
- isys-Server kennt dept101-Server



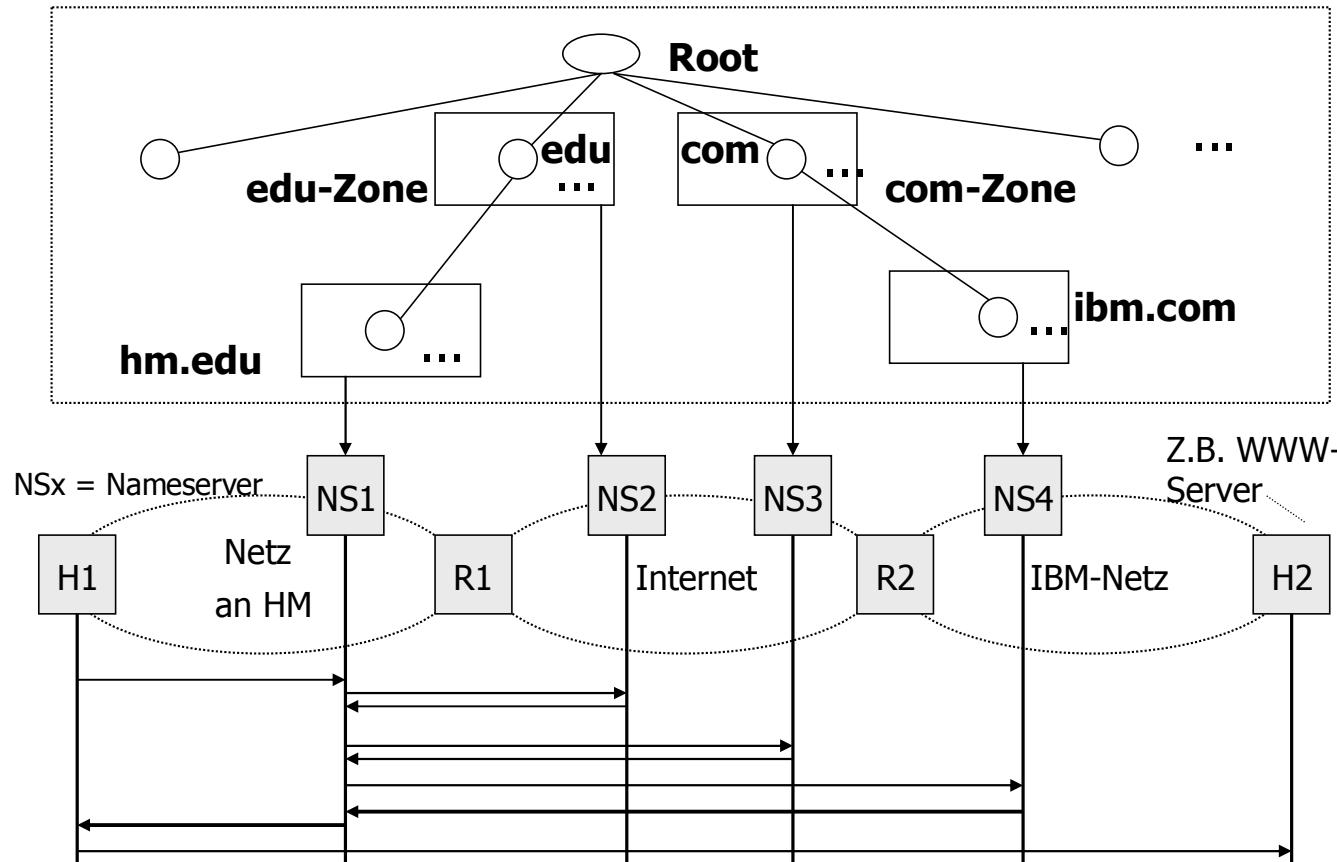
Name Server

- In jeder Zone muss es mindestens zwei Name Server geben (**primary und secondary**), die auch miteinander kommunizieren
- Für normale Anfragen wird ein **UDP-basiertes** Protokoll verwendet, größere Abfragen über TCP
- Primary und Secondary kommunizieren über **TCP**
- Name Server werden von Internet Service Providern (ISP) und Netzbetreibern betrieben
- Die Domain **.de** verwaltet das Deutsche Network Information Center (**DENIC**)
 - betrieben in Frankfurt (früher TU Dortmund, dann TU Karlsruhe)

Domainnamen

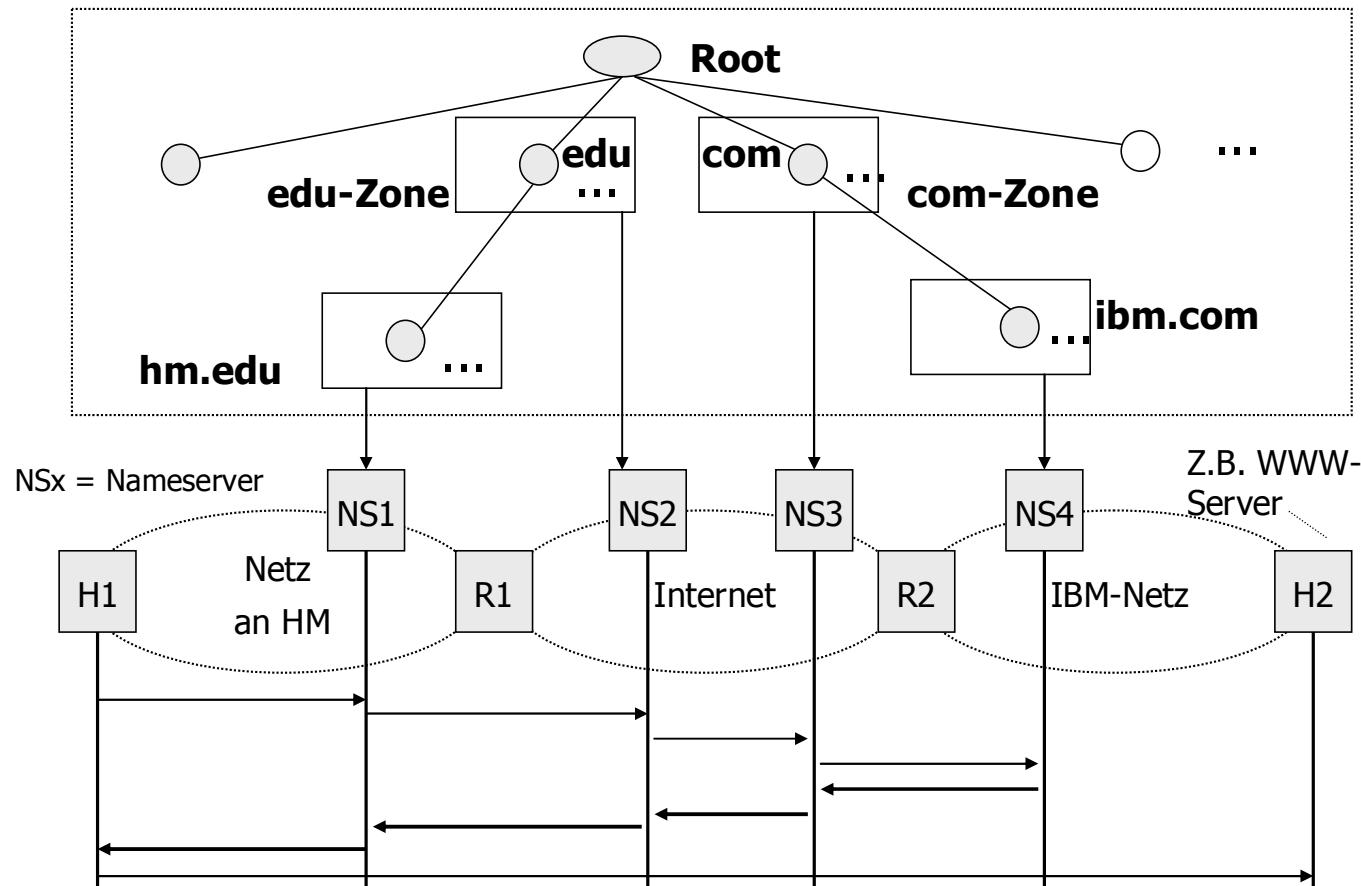
- DNS-Namen bestehen aus zwei Teilen, einem Hostnamen und einem Domainnamen, die zusammen den „**Fully Qualified Domain Name**“ (FQDN) bilden
 - Max. 255 Zeichen, keine Sonderzeichen, Umlaute oder Leerzeichen
 - Max. 63 Zeichen pro Teildomain oder Hostname
 - Beispieladresse: www.isys-software.de (isys-software.de ist der Domainname)
- Zu jeder Domain gehört eine Körperschaft, die diese verwaltet und für die Namensvergabe zuständig ist, oberste Körperschaft ist das NIC

Beispiel für eine iterative Auflösung einer IP-Adresse



- Nameserver verfügen über eigene Resolver, die üblicherweise iterativ arbeiten

Beispiel für eine rekursive Auflösung einer IP-Adresse



- Resolver von Clients arbeiten rekursiv

DNS Records

- Aufbau eines Resource Records als Quintupel folgender Form:

(Domainname, Time-to-live, Type, Class, Value)

Domainname: Name der Domäne

Time-to-live: Stabilität (Gültigkeitsdauer) des Werts als Integerzahl (je höher desto stabiler), für das Caching wichtig (kurz: TTL)

Type: Typ des Records (A = IP-Adresse, NS = Name Server Record, MX = Mail-Record, PT = Reverse Record, SOA = Start of Authority)

Class: immer IN (Internet)

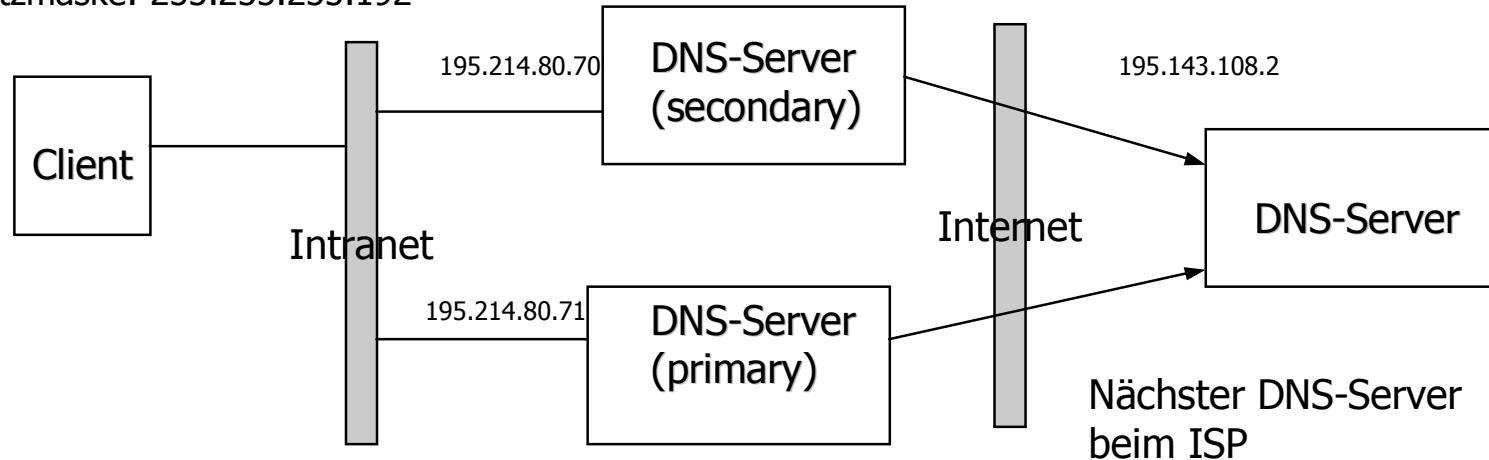
Value: Wert des Records je nach Typ: z.B. bei Typ = A → IP-Adresse

Unternehmensnetz anbinden

- Beispiel eines Unternehmensnetzes
- DNS-Server-Eintrag
 - Es gibt zwei DNS-Server mit den Adressen 195.214.80.70 und 195.214.80.71
 - Nächster DNS-Server beim ISP: 195.143.108.2

Unternehmens-IP-Adresse: 195.214.80.64

Netzmaske: 255.255.255.192

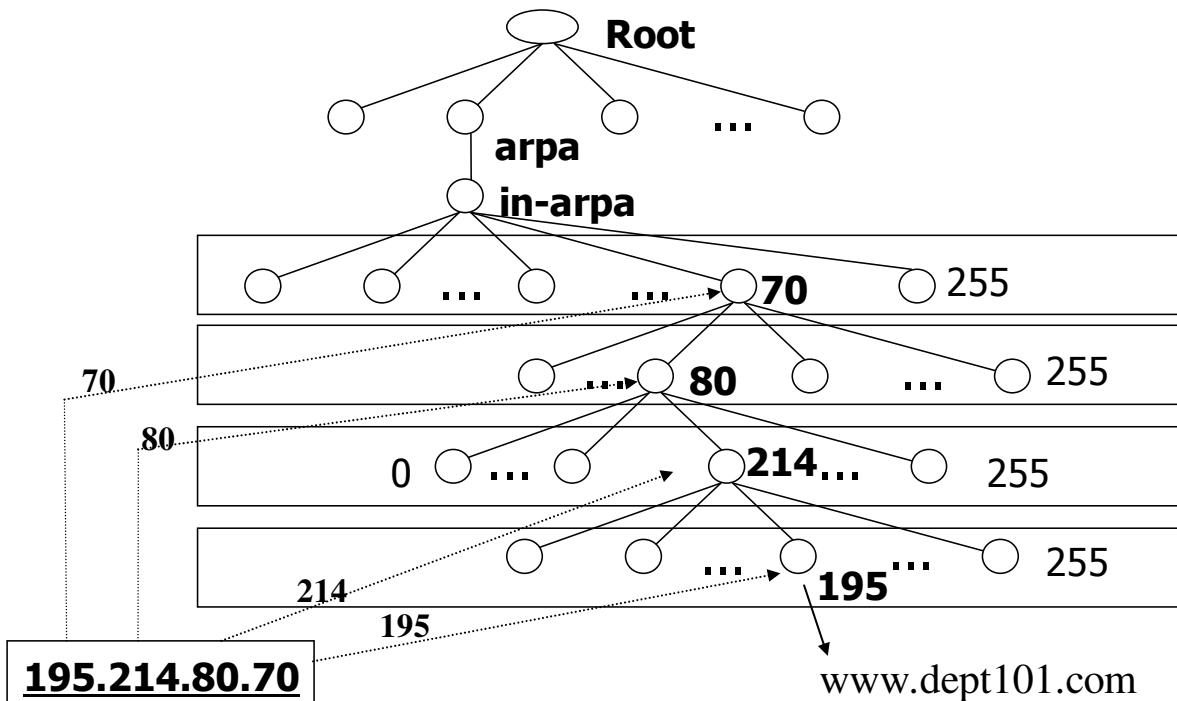


Auflösung von IP-Adressen auf Hostnamen

- Über DNS lassen sich auch IP-Adressen auf Hostnamen auflösen (**inverse Abfrage**)
- Benötigt man z.B. bei der Fehlerbehebung in TCP/IP-Netzen und für Logdateien
- Diese Zuordnungen werden in der besonderen Domäne **in-addr.arpa** durch InterNIC verwaltet
- Die Knoten der Domain sind nach Zahlen in der für IP-Adressen üblichen Repräsentation benannt
 - in-addr.arpa hat 256 Subdomains
 - Die Subdomains haben jeweils wieder 256 subdomains
 - In der untersten (vierten) Stufe werden die Resource Records mit vollem Hostnamen eingetragen

Auflösung von IP-Adressen auf Hostnamen: Reverse DNS

- Aufwändig: Durchsuchen des gesamten Domänen-Baums nach einer IP-Adresse
 - Daher eigene Domäne (Infrastruktur-Domäne): **in-addr.arpa-Domäne**
 - Unterhalb dieser Domäne existieren nur drei Subdomänen-Ebenen bei IPv4
 - Bei IPv6 → **ipv6.arpa** → entsprechend mehr Ebenen



Reverse DNS: RR-Record-Beispiel

- Eintrag für Reverse DNS:

70.80.214.195.in-addr.arpa. 1285 IN **PTR** server.example.com

- Korrespondierender RR-Record:

server.example.com 1800 IN **A** 195.214.80.70

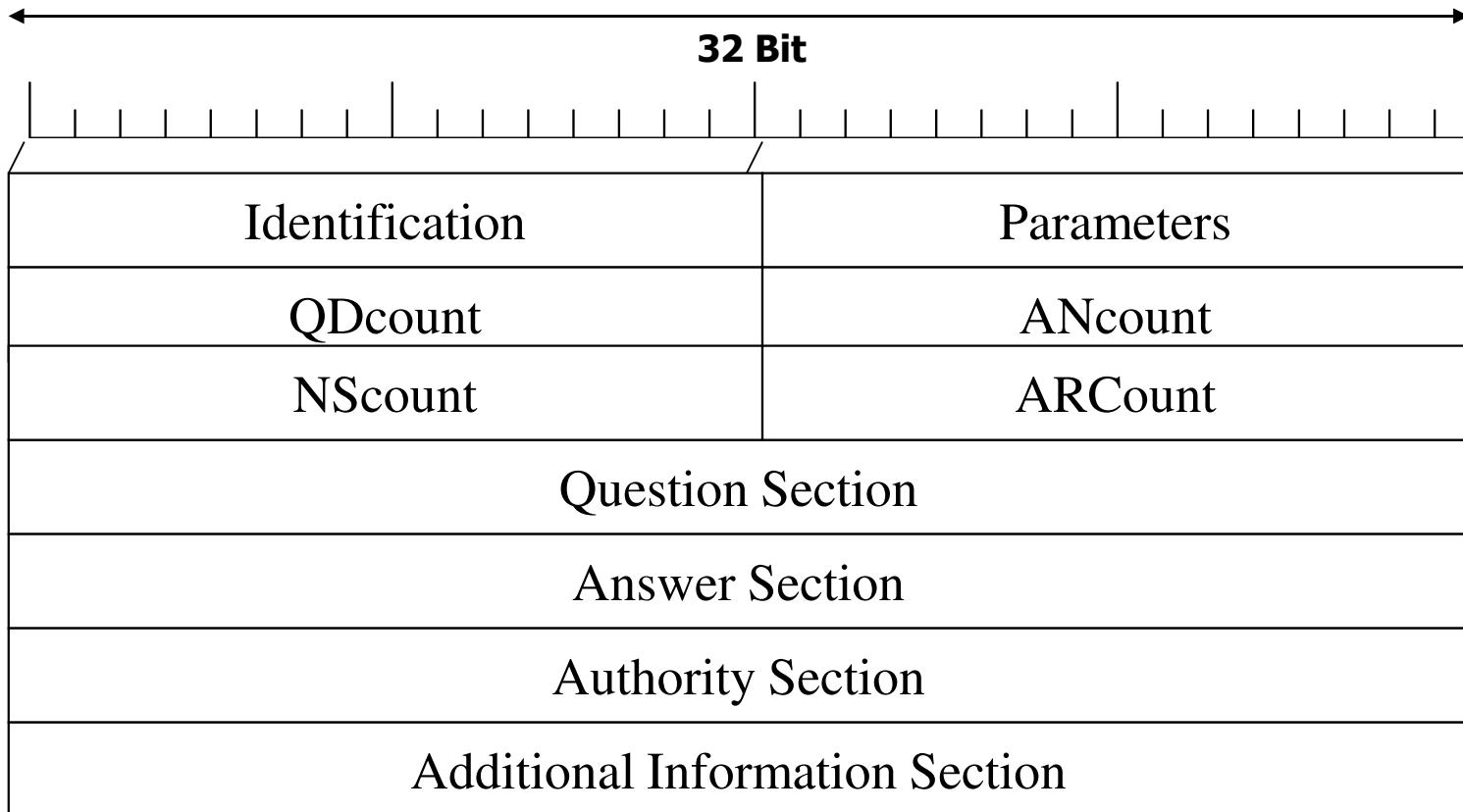
TTL

Reverse-
Eintrag

Hinweise:

- *reverse* → umgekehrte Reihenfolge in der in-addr.arpa-Domäne bzw. in der ip6.arpa-Domäne
- *inverse* → umgekehrte Informationsanfrage, also ein Name und nicht eine IP-Adresse wird gesucht

DNS-PDU



DNS-PDUs

- DNS-Nachrichten setzen sich aus Sektionen (Sections) zusammen:
 - Der **Header** besteht aus den ersten sechs Feldern (Header Section)
 - **Question Section:** Enhält Felder zur Spezifikation der Anfrage
 - **Answer Section:** Enthält die Antwort eines Name-Servers in Form von Resource Records (RRs)
 - **Authority Section:** Enthält RRs eines autorisierten Servers
 - **Additional Information Section:** Enthält zusätzliche Informationen zur Anfrage oder zur Antwort

DNS-Header

- **Identification (2 Bytes)**
 - Id der Anwendung, die die Abfrage abgesetzt hat
- **Parameters**
 - 0 := Anfrage, 1 := Response
 - Zusätzliche Information: Hinweis, ob es eine normale oder eine inverse Abfrage handelt
 - Inverse Abfrage: Für IP-Adresse einen Hostnamen suchen
- **QDcount**
 - Anzahl der Einträge in der Question Section
- **ANCOUNT**
 - Anzahl an Resource Records (RR) in der Authority Section
- **ARCOUNT**
 - Anzahl an RRs in der Additional Information Section

Exkurs:

DNS-Paketlänge unter Windows 2003

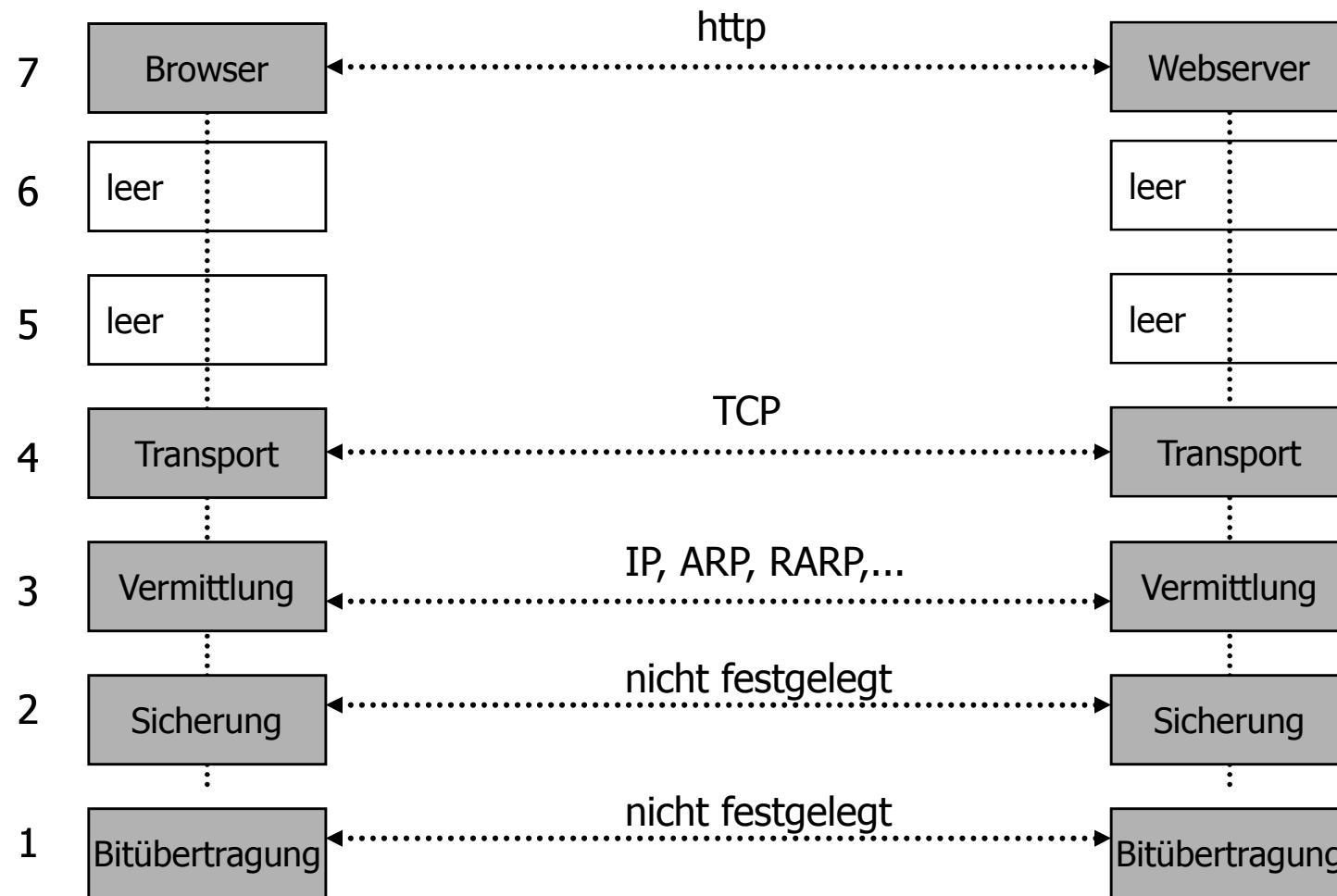
- Ändern der UDP-Paketlänge für DNS unter Windows 2003:
 - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DNS\Parameters → **MaximumUdpPacketSize**
 - Standardwert: 1280 Byte
 - Werte zwischen 512 and 16384 sind möglich

Überblick

1. Client-/Server versus Peer-to-Peer (P2P)
2. Domain Name System
- 3. HTTP**
4. E-Mail
5. Multimedia-Protokolle

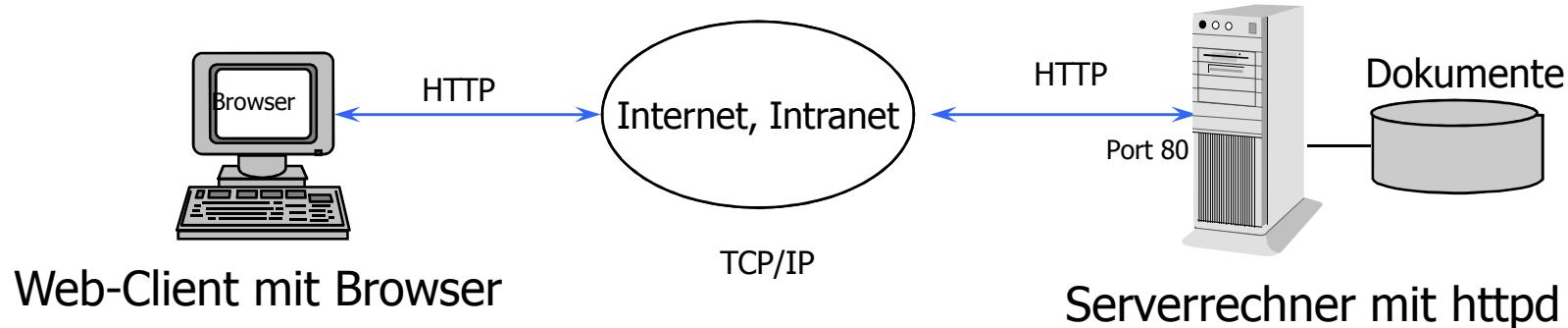
Überblick

Einordnung in die TCP/IP-Protokollfamilie



Web-Basiskomponenten

- HTML (Hypertext Markup Language) dient als Auszeichnungssprache (Markup Language)
- Dokumente (auch HTML-Seiten genannt) können über Links beliebig verknüpft werden
- HTTP (Hypertext Transfer Protocol) auf Basis von TCP/IP als textbasiertes Kommunikationsprotokoll
- Web-Browser auf der Clientseite als grafische Benutzeroberfläche
- Web-Server (httpd) auf der Serverseite mit Dokumenten (HTML)



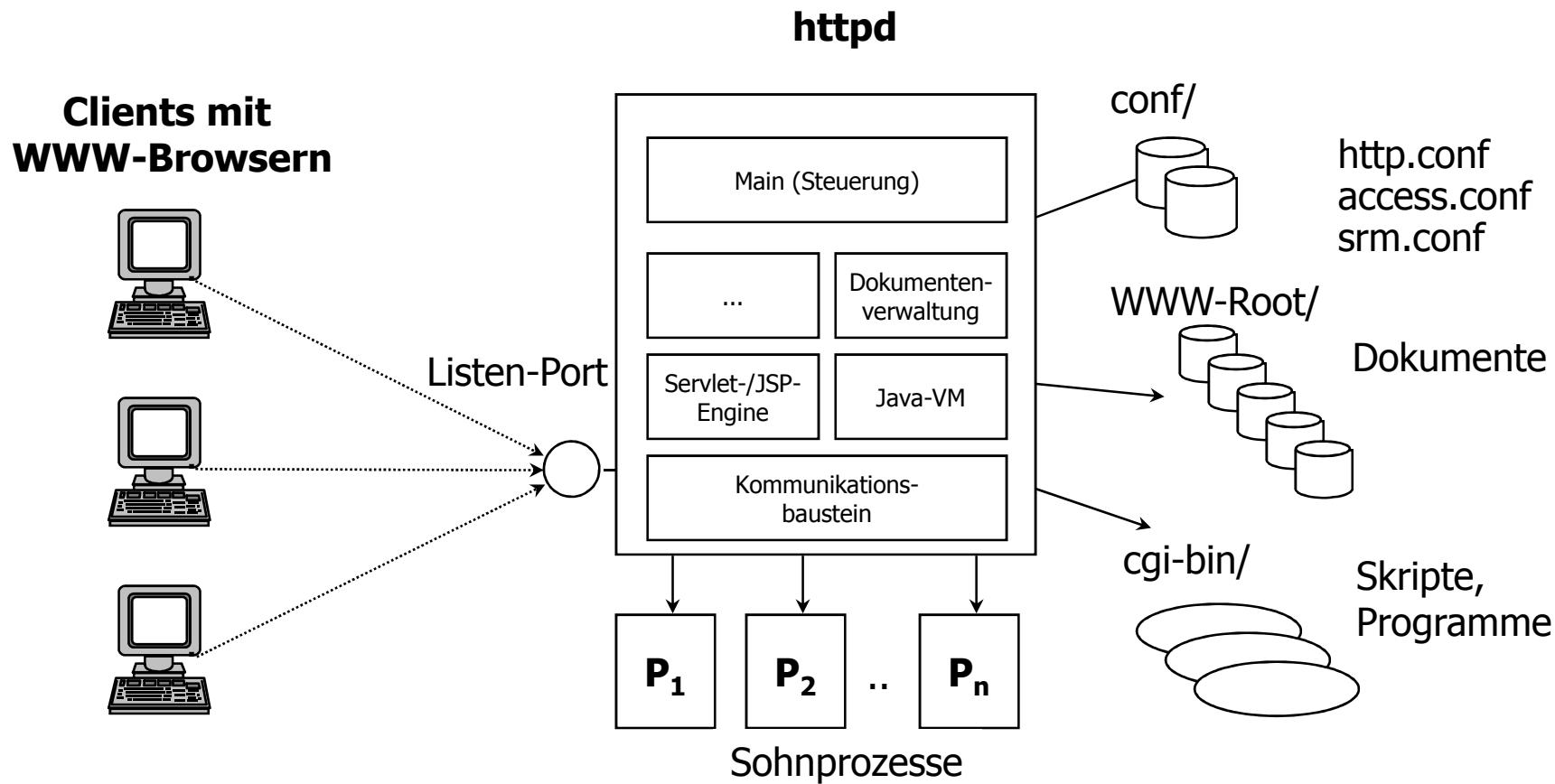
Der Web-Server: Arbeitsweise

- Web-Server wartet auf ankommende Verbindungsaufbauwünsche standardmäßig auf TCP-**Port 80** mit Socket-Funktion `listen()`
- Verbindungsaufbauwunsch wird entgegengenommen
- HTTP-Request wird empfangen und in einem eigenen Thread oder Prozess bearbeitet
- Das gewünschte Dokument wird lokalisiert (wir nehmen hier die Adressierung einer statischen Webseite an)
- Hierzu wird im konfigurierten Root-Verzeichnis des Web-Servers gesucht

Der Web-Browser: Grundlegendes

- Web-Browser stellen die Clients dar und bieten dem Anwender eine komfortable Benutzeroberfläche
- Die gängigsten Web-Browser sind:
 - Microsoft Internet Explorer
 - Mozilla Firefox
 - Apple Safari
 - ...
- Web-Browser sind komplexe Softwareprogramme mit vielen Möglichkeiten der Einstellung
- Sie können erweitert werden: Plug-ins
- Sie arbeiten nicht immer gleich → schafft Probleme bei der Entwicklung von Web-Anwendungen

Zusammenspiel der Komponenten



Ein einfaches HTML-Beispiel

```
<HTML>
<HEAD>
    <TITLE> Beispiel einer Web-Seite </TITLE>
</HEAD>
<BODY>
    <H1> Test-Webseite </H1>
    <H2> <I> Interessante Links: </I></H2>
    <UL compact>
        <LI><A HREF="http://www.isys-software.de"> iSYS Web-Site</a> </li>
        <LI><A HREF="http://www.fh-muenchen.de"> Web-Site der FH-M&uuml;nchen</a>
            </li>
    </UL>
</BODY>
</HTML>
```

Kompakt darzustellende Liste

Listeneintrag

Hyperlink

- Textbasierte Auszeichnungssprache (Markup Language)
 - HTML Version 5
-

HTTP, Einführung

- HTTP (= HyperText Transfer Protocol) bildet als Kommunikationsprotokoll zwischen Web-Client- und Server das **Rückgrat** des Webs
- Man muss HTTP verstehen, wenn man gute Web-Anwendungen entwickeln will
- RFC 2616 (vorher 2068) der Network Working Group regelt den Protokollstandard
- HTTP ist ein **verbbindungsorientiertes** Protokoll
- Nutzung von **MIME**-Bezeichnern für content-type (Multipurpose Internet Mail Extensions), z.B. text/html, image/gif

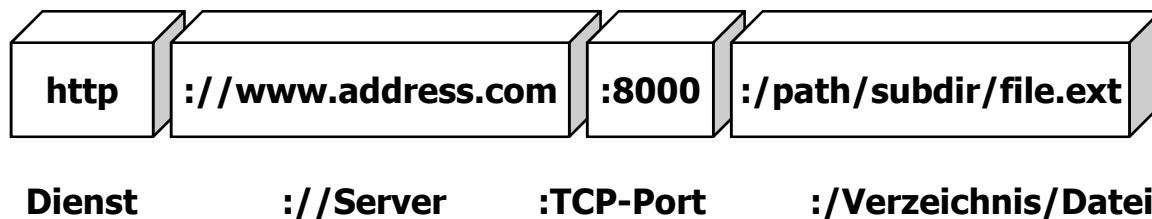
Das Protokoll, Einführung

- Für jede Seite und jede Grafik, die vom Server gelesen wird, wird in HTTP V1.0 standardmäßig eine TCP-Verbindung zwischen WWW-Browser und WWW-Server aufgebaut werden
 - Nicht mehr so in HTTP V1.1 (1999)
- HTTP ist vom Protokolltyp her ein **zustandsloses Request/Response-Protokoll**
 - Weder Sender noch Empfänger merken sich irgendwelche Stati zur Kommunikation
 - Ein Request ist mit einem Response vollständig abgearbeitet
- **Protokolloperationen** sind z.B. GET, HEAD, PUT, POST,...

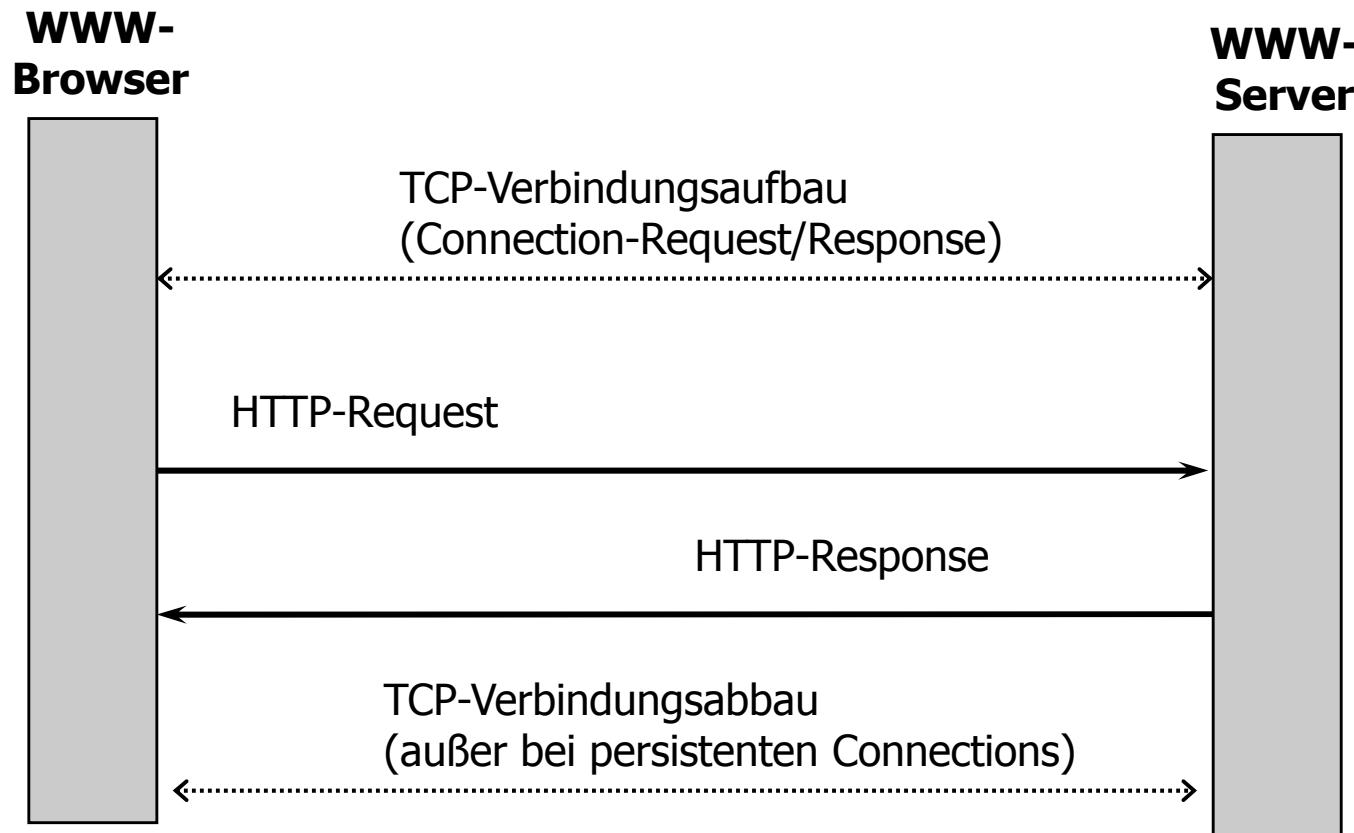
Adressierung im WWW

- Die Adressierung über **URIs** bzw. **URLs**
- Uniform Resource Locator (URL) identifiziert das Dokument
- Uniform Resource Identifier (URI): Allgemeinerer Begriff als Überbegriff für alle Adressierungsmuster, die im WWW unterstützt werden

URL-Aufbau:

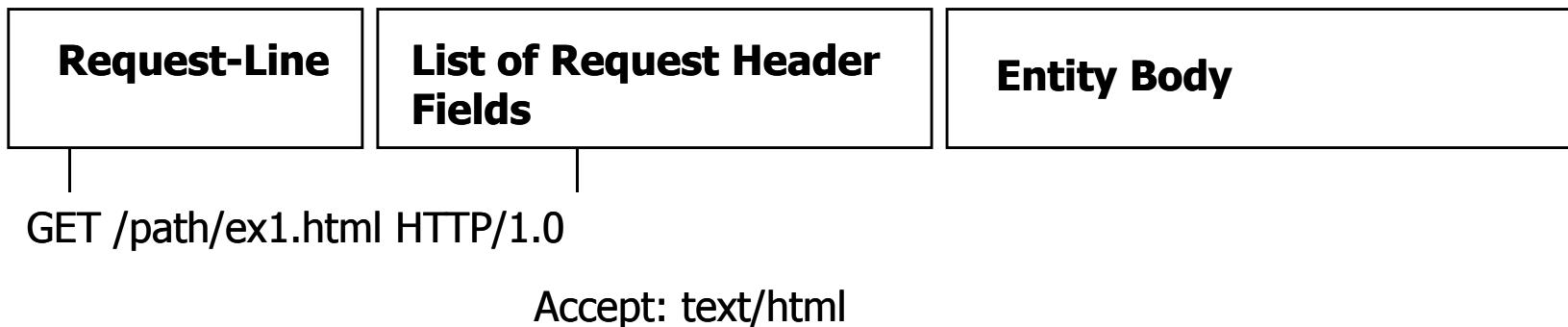


Ablauf der Kommunikation

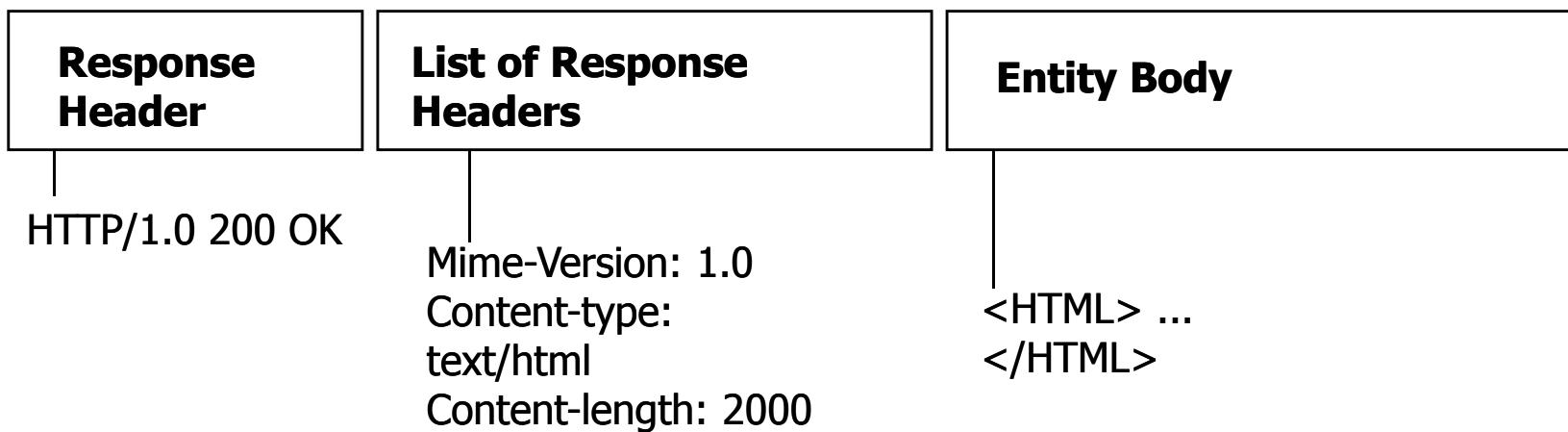


Aufbau der HTTP-PDUs

HTTP-Request



HTTP-Response



Ablauf der Kommunikation

Vom Client initiierte Aktionen:

1. Browser ermittelt Hostnamen des Servers aus URI und besorgt sich über DNS die IP-Adresse des Servers
2. Client baut aktiv eine TCP-Transportverbindung zum Socket des Servers mit Portnr. 80 auf.
3. httpd im Server nimmt die Verbindung an.
4. Client sendet HTTP-Befehl, z.B. **GET /index/html HTTP/1.1**
5. Client sendet weitere optionale HTTP-Header (eigene Konfiguration und akzeptierte Dokument-Formate,..), z.B. **Accept: image/gif**
6. Client sendet eine Leerzeile (Ende des Requests anzeigen)
7. Client sendet evtl. noch zusätzliche Daten als Parameter (Input-Felder aus Forms)

Ablauf der Kommunikation

Vom Server initiierte **Aktionen**:

1. Nach der Bearbeitung des Requests sendet der Server eine Statuszeile (Response Header) mit drei Feldern (Version, Statuscode, lesbarer Text), z.B. **HTTP/1.1 200 OK**
2. Anschließend sendet der Server HTTP-Headers wie z.B.
Content-type: text/html
Content-length: 2482
3. Danach wird eine Leerzeile gesendet und das angeforderte Dokument
4. Falls nicht vom Client ein Header „**Connection: Keep alive**“ gesendet wurde, wird die TCP-Verbindung wieder abgebaut
→ in HTTP 1.1 wird „**Keep alive**“ standardmäßig verwendet. Dadurch können Applets, Graphiken, Rahmen,... über dieselbe Verbindung übertragen werden

Übung

Da HTTP ein ASCII-basiertes Protokoll ist, kann man den Ablauf der Kommunikation auch mit einer Telnet-Verbindung verfolgen. Probieren Sie dies an einem beliebigen WWW-Server aus.

- telnet <url des Servers> 80
- Was passiert?
- Fordern Sie eine WWW-Seite mit GET-Operation an
- Was passiert?

Protokolloperation GET und HEAD

- Wichtig sind vor allem die Operationen GET und POST
- Bei GET ist der Body-Bereich immer leer und zur Übertragung von Input-Parametern für serverseitige Programme wird die URL ergänzt (nach einem Fragezeichen kommen die Parameter)

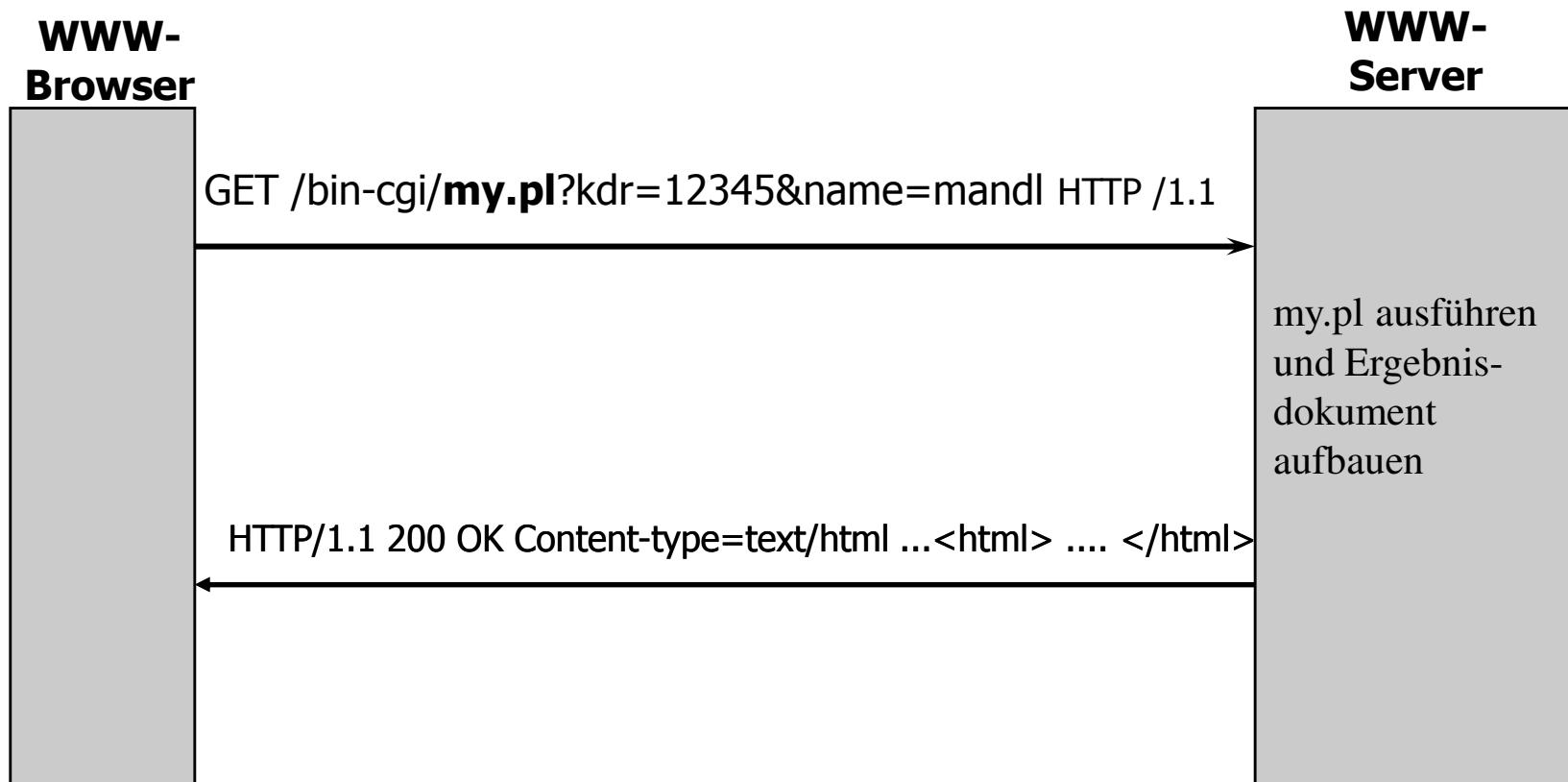
Beispiel: `GET /bin-cgi/my.pl?kdr=12345&name=mandl HTTP 1/1`

- Im Beispiel sendet der Server nach Abarbeitung des CGI-Perlscrips `my.pl` den Output des Scripts als Response zurück.
- Die HEAD-Operation funktioniert wie GET nur dass hier vom Server kein Ergebnisdokument gesendet wird

Beispiel: `HEAD /index/html HTTP/1.1`

→ Sinnvoller Einsatz z.B. dann, wenn Client nur das Datum der letzten Änderung oder die Dokumentgröße wissen will

Beispiel einer GET-Operation



Protokolloperation POST

- Über die POST-Methode können in einem Client-Request Daten an den Server zur Weiterverarbeitung über ein Programm gesendet werden
- Die Daten werden im Body gesendet
- Der Server gibt die Daten an das mit der angegebenen URI adressierte Programm weiter. Meist werden die Daten URL-codiert übergeben
- Am Response ändert sich nichts

Beispiel: **POST /cgi-bin/my2.pl HTTP/1.1**

Content-type: application/x-www-form-urlencoded

Content-length: 20

monat=oktober&tag=24

Header

- Es gibt verschiedene Header-Kategorien, die eine Menge von Möglichkeiten bieten. Header-Kategorien sind:
 - Allgemeine Header für Client und Server
 - Request Header für Client
 - Response Header für Server
 - Entity Header für Client und Server
- Aufbau von Headern (Groß-/Kleinschreibung spielt keine Rolle):
`<header-name>: <header-value>`
z.B. [Content-Type: text/html](#) oder [content-type: text/html](#)
- Header kann sich über mehrere Zeilen erstrecken, Folgezeilen müssen mit Blank oder Tab beginnen

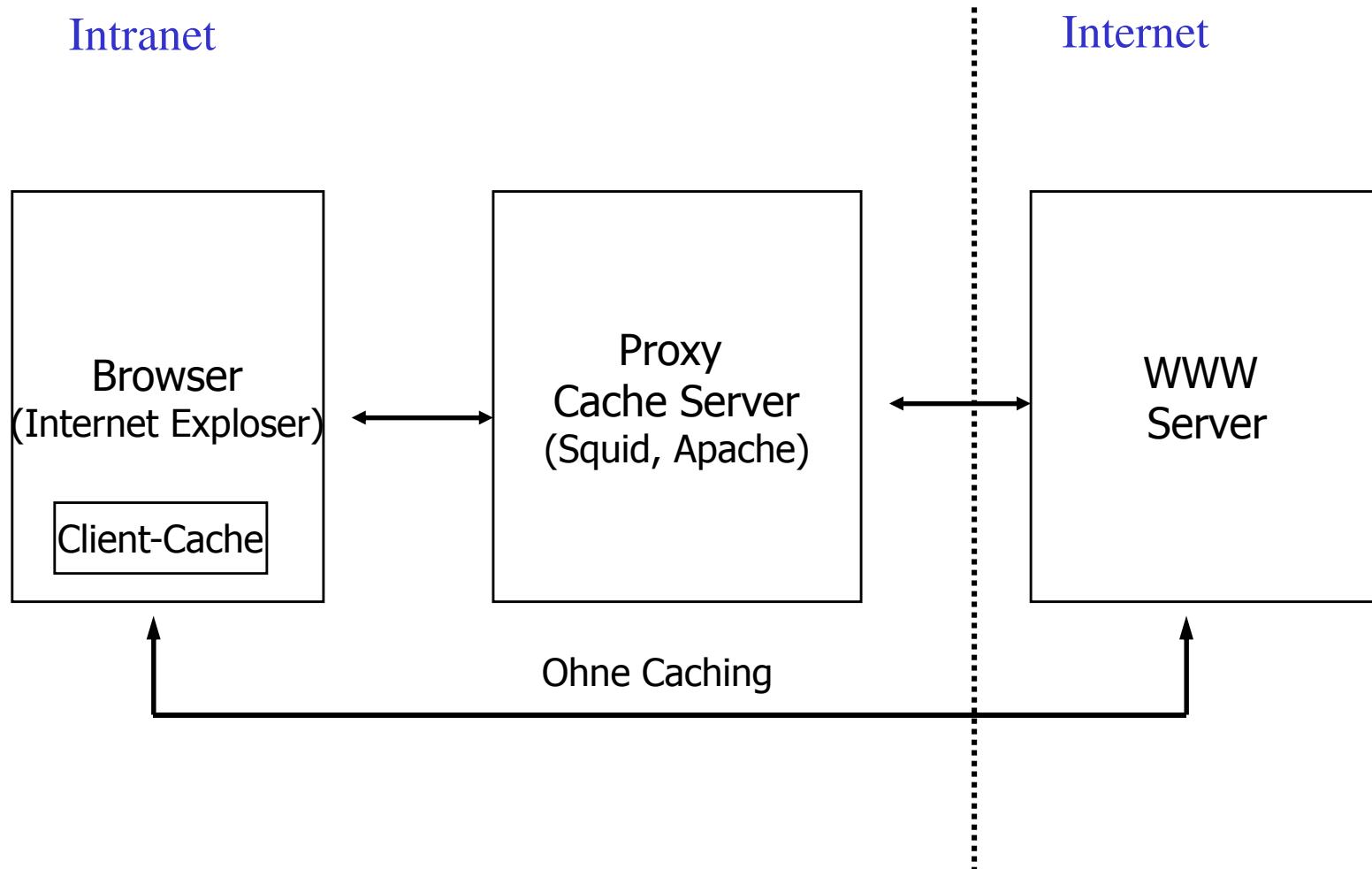
Header: Ausgewählte Beispiele

- Allgemeine Header
 - Cache-Control: Direktiven → Definiert in einer Liste Caching-Direktiven (Liste mit Komma abgetrennt zulässig)
 - z.B. im Request-Header: Cache-Control: no-cache → keinen Cache verwenden
 - z.B. im Response-Header: Cache-Control: no-cache → nicht in den Cache legen
- Request-Header für Client
 - Accept: <typ>/<untotyp> (Liste mit Komma abgetrennt zulässig)
 - z.B. Accept: text/* → Alle Textuntotypen sind erlaubt (html, plain, enriched,...)
 - Cookie: <name>=<wert> (Liste mit Strichpunkten abgetrennt ist zulässig). Speichert das Name-Value-Paar für die URL
 - z.B. Cookie: mps=1234123

Proxy Caching

- Z.B. Squid oder Apache Cache
- Arbeitsweise ist im HTTP-RFC definiert
- Gründe für Caching:
 - Performance-Optimierung
 - Sicherheitsaspekte bei Firewalleinsatz
 - Begrenzte Anzahl an vorhandenen IP-Adressen
- In Mehrbenutzerumgebungen kann ein „gecachtes“ Dokument mehreren Web-Clients zur Verfügung gestellt werden: Erhöhung der Hitrate
- Regeln für das Caching sind konfigurierbar

Proxy Caching

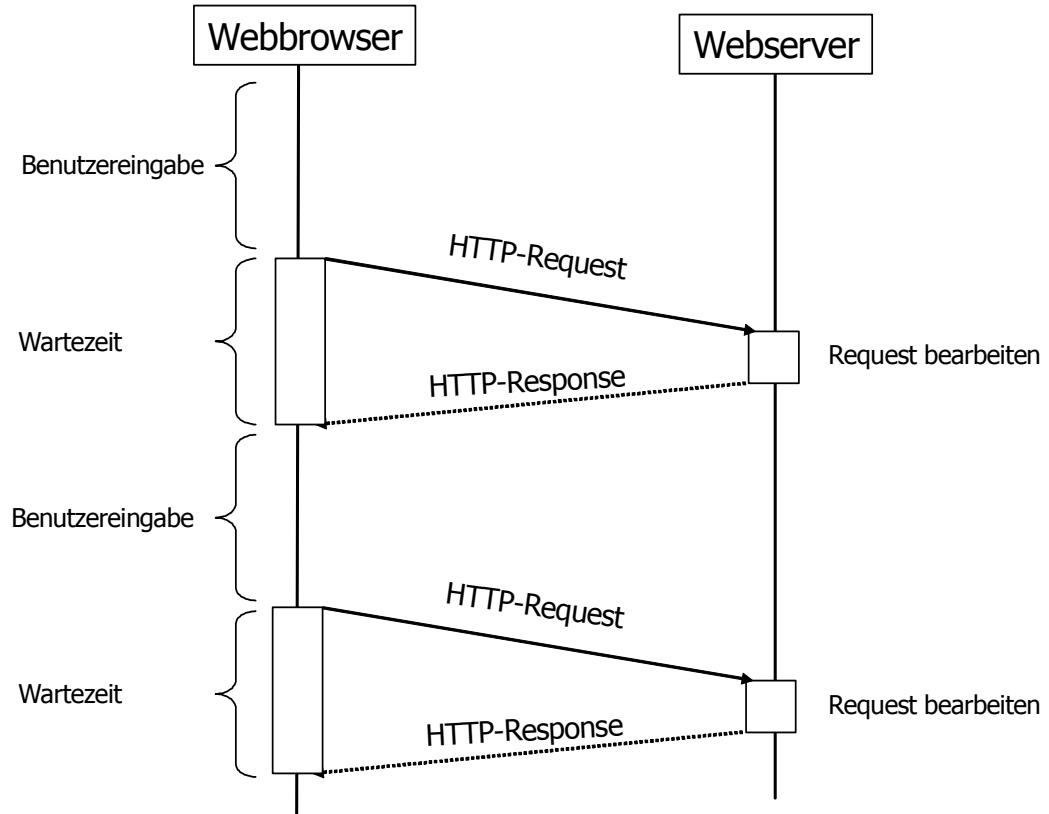


AJAX - Einführung

- Asynchronous JavaScript and XML
- Der Begriff geht aus einem Artikel von Jesse James Garrett hervor „Ajax: A New Approach to Web Applications“
- Es beschreibt keine neue Technologie, sondern einen neuen Ansatz, der auf bereits vorhandenen Technologien aufbaut
- Die traditionelle Webkommunikation soll durch asynchrone Mechanismen benutzerfreundlicher gestaltet werden

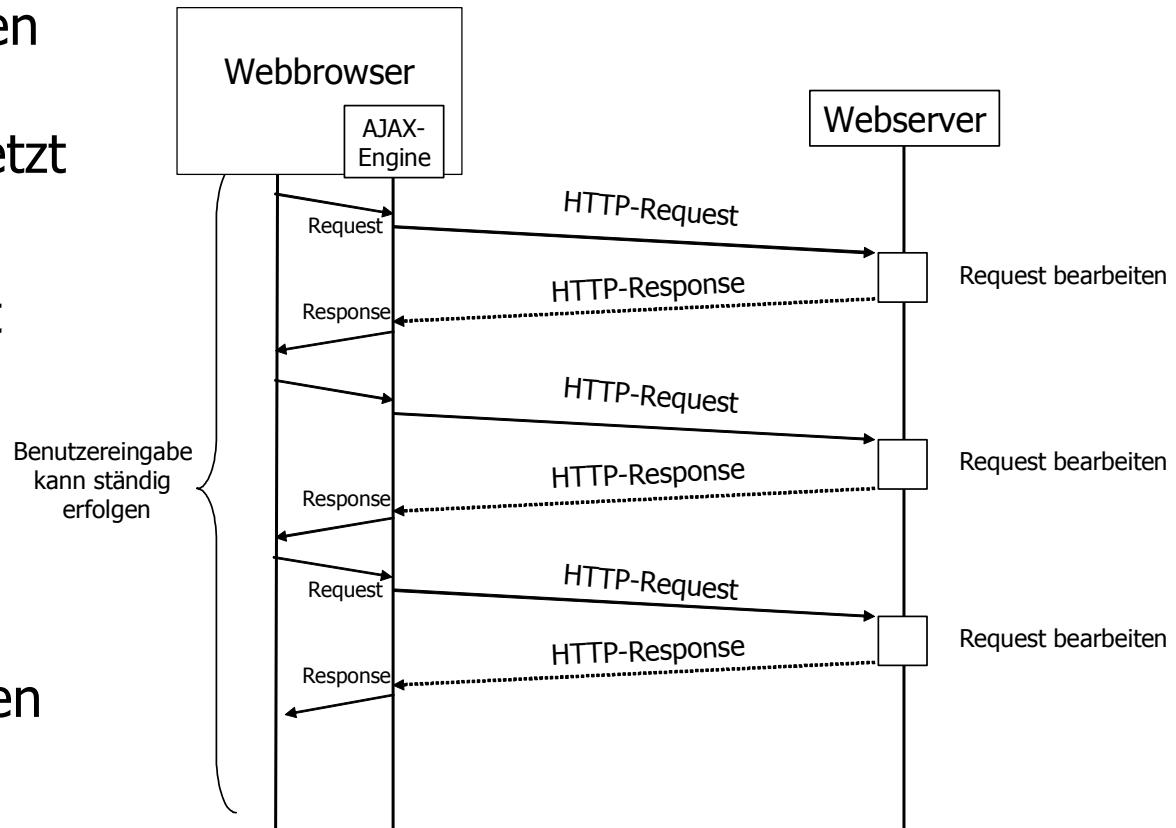
Herkömmliche Webkommunikation

- Formularbasiert
- Es wird immer die komplette Seite übertragen
- Nach jeder Anfrage muss der Benutzer warten und kann nicht interagieren

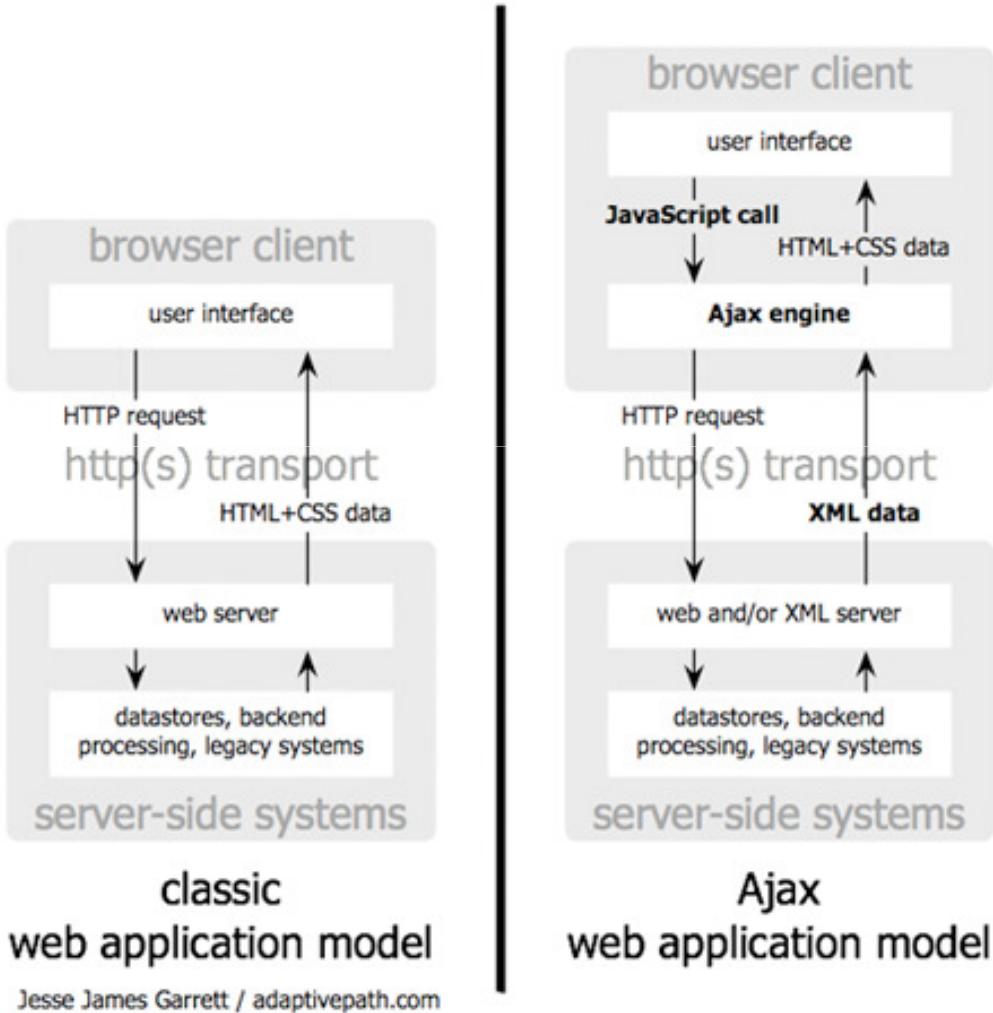


AJAX – asynchrone Webkommunikation

- Die Anfragen werden asynchron im Hintergrund abgesetzt
- Es werden nur Nutzdaten versandt
- Die Antworten werden in die Seite eingearbeitet
- Der Benutzer kann jederzeit interagieren



AJAX – Kommunikationsablauf



AJAX – XMLHttpRequest

- Zentraler Baustein von AJAX
- Ermöglicht die asynchrone Kommunikation
 - Absenden von http-Requests
 - Überwachen der Kommunikation und der eingehenden Antwort über Event-Handler
- Integration im IE über ActiveX, bei Mozilla nativ im JavaScript Dialekt
- Der Zugriff erfolgt JavaScript

AJAX – XMLHttpRequest – Methoden und Attribute

Name	Beschreibung
readyState	Der Status des Request-Objekts
onreadystatechange	Dieser Event-Handler wird aufgerufen, wenn sich der Status des Request-Objekts (readyState) ändert.
responseText	Die Nutzdaten der Antwort als Text
responseXML	Die Daten als DOM-Objekt (Typ document). Ist nur gefüllt, wenn der Server die Daten als XML sendet.
status	der HTTP-Status-Code der Anfrage [1]
open(method, url, async, user, pwd)	Bereitet das Request-Objekt auf eine Übertragung an die Adresse URL mit der HTTP-Protokolloperation(GET/POST) vor. Zusätzlich wird angegeben ob die Anfrage asynchron oder synchron gesendet wird(true/false). Optional werden hier User/Passwort zur Authentifizierung über HTTP angegeben.
abort()	Bricht den Request ab.
send(requestBody)	Sendet den über open() initialisierten Request mit dem übergebenen responseBody(nur bei POST) ab.
setRequestHeader(name,wert)	Setzt ein Request-Header-Attribut

AJAX – XMLHttpRequest – readyState

Wert	Bezeichnung	Beschreibung
0	UNSENT	Status nach der Instanzierung des Objekts
1	OPENED	Das Objekt ist bereit die Anfrage zu senden.
2	HEADERS_RECEIVED	Die Anfrage wurde gesendet.
3	LOADING	Header und Status sind verfügbar. Der Message-Body wird empfangen.
4	DONE	Die Anfrage ist beendet. Alle Daten liegen nun vor und können abgerufen werden

AJAX – XMLHttpRequest – Ein einfaches Beispiel



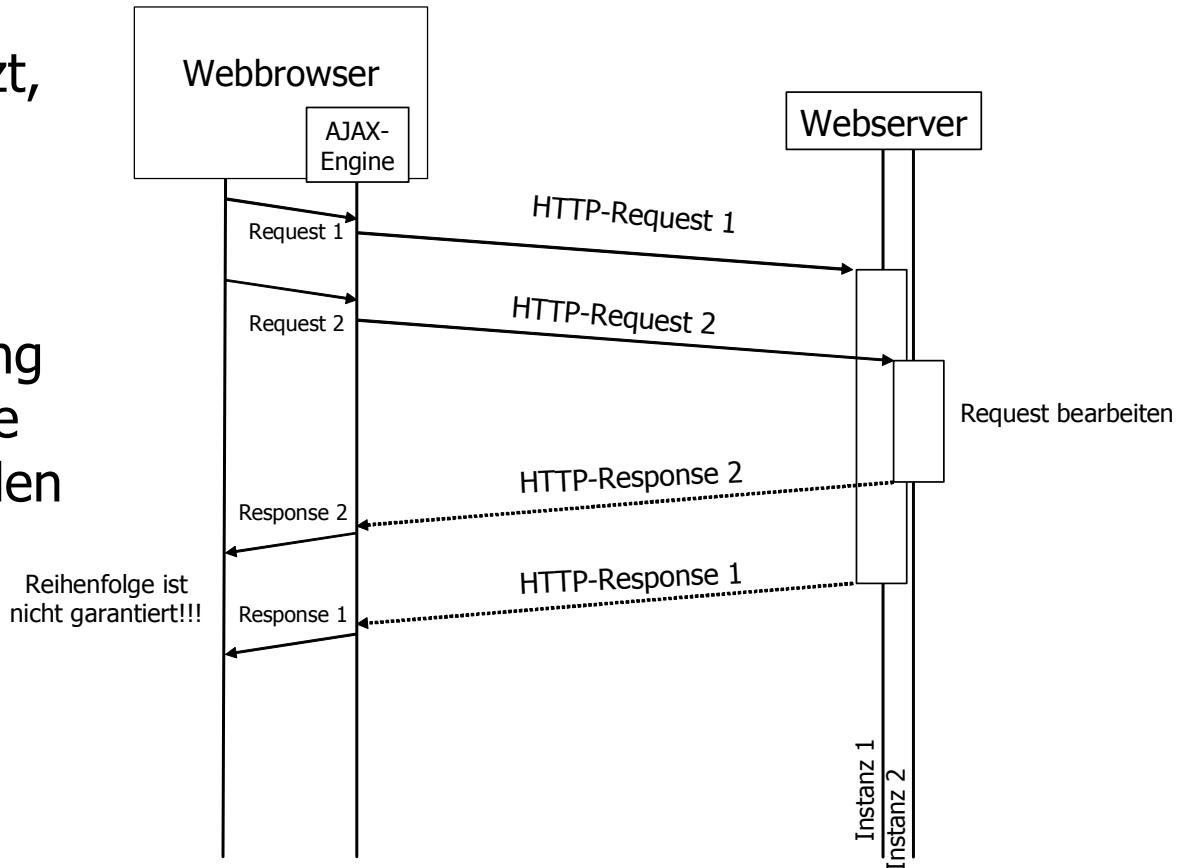
- Beim Drücken des Buttons wird JavaScript-Funktion „loadZitat()“ aufgerufen und die Anfrage an den Server gesendet
- Die Antwort wird nach ca. 3 Sekunden empfangen und auf der Seite angezeigt
- In der Zwischenzeit kann der Benutzer weiter auf der Seite arbeiten, ist als nicht blockiert

AJAX – XMLHttpRequest – Eventhandler

```
function loadZitat() {
    var request = new XMLHttpRequest();
    request.onreadystatechange= function() {
        if(request.readyState==1) {
            document.getElementById("zitat").innerHTML=
                "Warte auf Antwort...";
        }
        if(request.readyState==4) {
            if(request.status==200) {
                document.getElementById("zitat").innerHTML=
                    "Zitat des Tages: \n" +request.responseText;
            }
        }
    }
    request.open("GET", "zitat.php", true);
    request.send(null);
}
```

AJAX – Konkurrierende Requests

- Request 1 wird vor Request 2 abgesetzt, die Antwort kommt jedoch später an
- Die Synchronisierung muss auf Clientseite implementiert werden



AJAX – Vor-/Nachteile

Vorteile	Nachteile
Es werden weniger Daten, nur Nutzdaten, ausgetauscht.	Viele kleine Anfragen, zusätzliche Netzlast
Daten werden im Hintergrund ausgetauscht, Anwendung ist nicht blockiert	Kompliziertes State-Management, Keine Reihenfolgegarantie.
Einzelne Validierungen von Formularfeldern schon während der Eingabe (Benutzerfreundlichkeit)	Applikationslogik muss im Browser implementiert werden (JavaScript)
Benutzerfreundliche UI-Elemente	Browser-Inkompatibilitäten

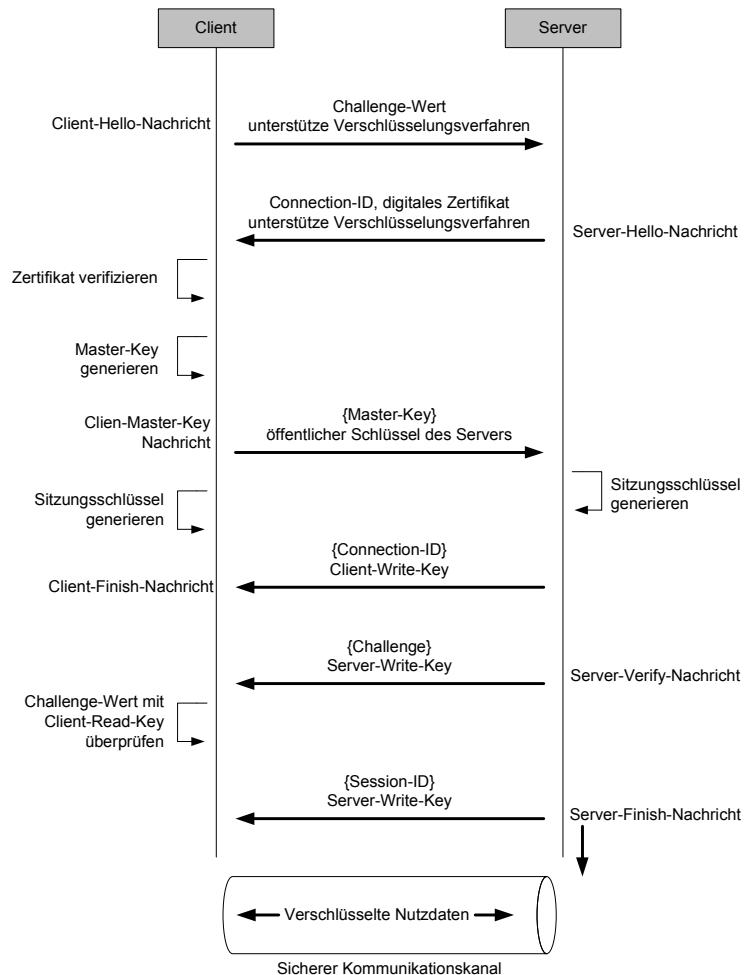
AJAX – Frameworks

- Zahlreiche Frameworks erleichtert die Entwicklung von benutzerfreundlichen Anwendungen
 - Clientseitig: vielfältige innovative und benutzerfreundliche UI-Elemente werden bereitgestellt, z.B. DOJO, Scriptaculous,...
 - Serverseitig: meist in Webframeworks integriert, z.B. ASP.NET AJAX, JSF, Google Web Toolkit
- Bei den meisten Frameworks wird angestrebt, komplett auf JavaScript-Programmierung zu verzichten und Elemente mit browserübergreifender Kompatibilität bereitzustellen

Einschub: TLS, SSL

- ***Transport Layer Security*** (TLS) bzw. dessen Vorgängerversion ***Secure Sockets Layer*** (SSL) sind Verschlüsselungsprotokolle
 - TLS ist die **standardisierte Weiterentwicklung** der SSL-Version 3.0
 - Heute wird SSL/TLS von allen gängigen Web-Browsern und Web-Servern unterstützt
 - SSL wurde ursprünglich von der Firma Netscape entwickelt und **1996** an die IETF (Internet Engineering Task Force) übergeben
 - **1999** wurde TLS in einer ersten Version im **RFC 2246** verabschiedet wurde
-

Einschub: TLS, SSL: Protokollverlauf



Überblick

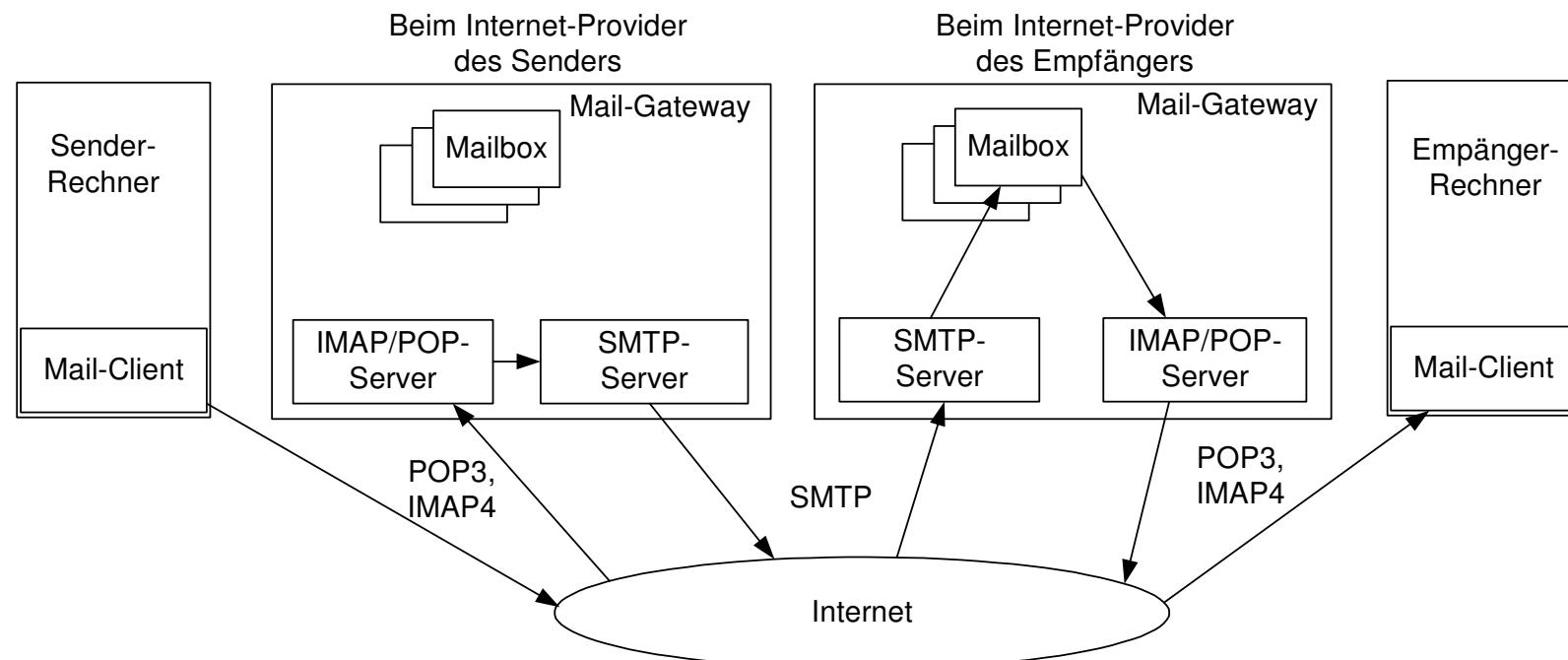
1. Client-/Server versus Peer-to-Peer (P2P)
2. Domain Name System
3. HTTP
- 4. E-Mail**
5. Multimedia-Protokolle

Grundlegendes

- Austausch von elektronischen Nachrichten
- Elektronische Mailbox mit einer eindeutigen E-Mail-Adresse (z.B. mandl@cs.hm.edu) in einem **E-Mail-Gateway**
- Die Mail-Gateways kommunizieren untereinander über das Protokoll **SMTP** (Simple Mail Transfer Protocol, well-known TCP-Port = 25)
- **Mailzugangsprotokolle** (Zugang zur Mailbox)
 - **POP3** (Post Office Protocol, Version 3, well-known TCP-Port = 110) → Privatleute
 - **IMAP4** (Internet Message Access Protocol, well-known TCP-Port = 143) → Unternehmen, IMAP kann mehr

Einschub: Architekturüberblick

- Jeder Mail-Client (*Microsoft Outlook, Mozilla Thunderbird,...*) baut eine TCP-Verbindung zu seinem SMTP-Server (*sendmail, qmail*) bei seinem Internet-Provider auf
- SMTP-Server kommunizieren untereinander



Überblick

1. Client-/Server versus Peer-to-Peer (P2P)
2. Domain Name System
3. HTTP
4. E-Mail
- 5. Multimedia-Protokolle**

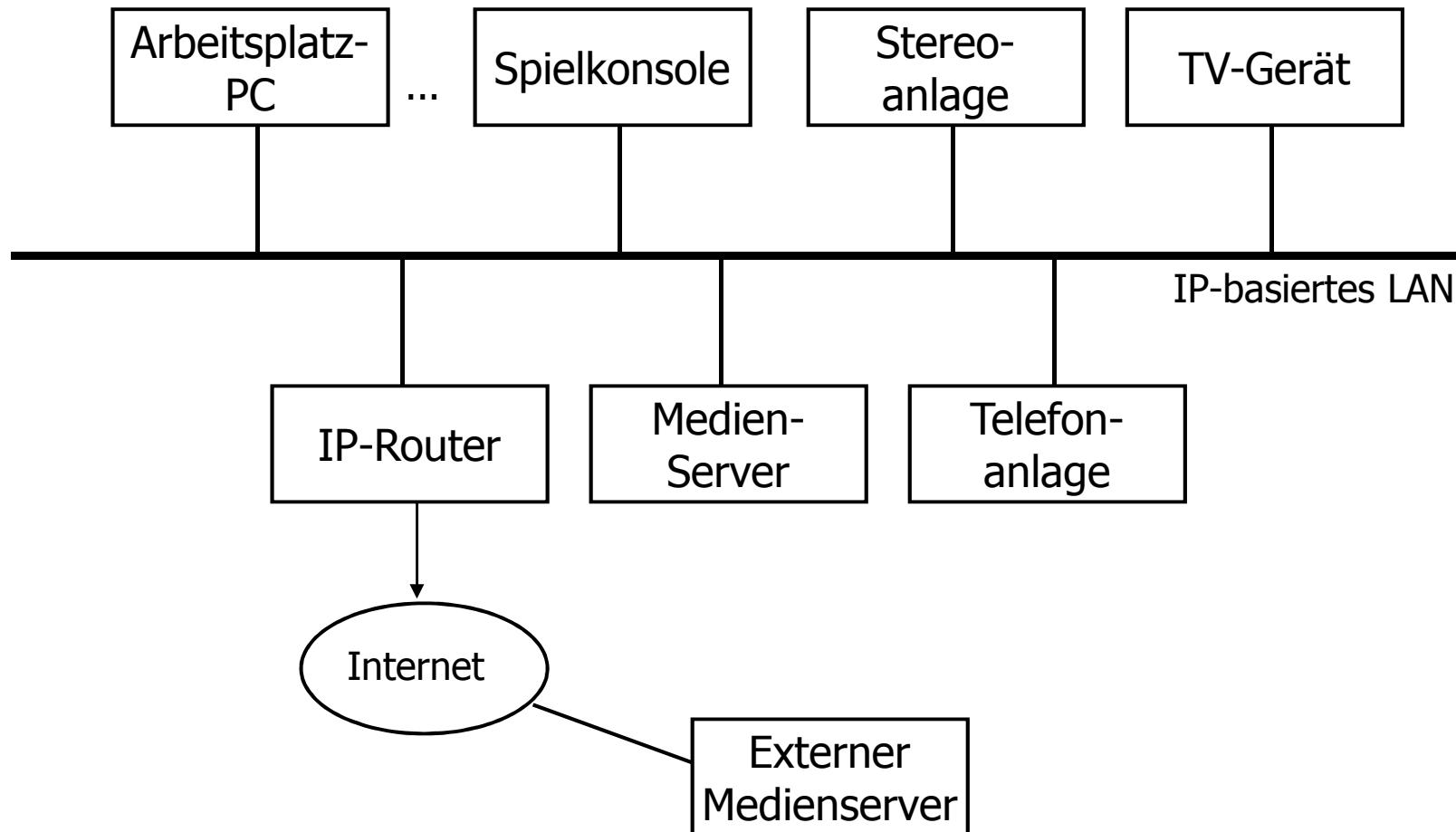
Grundlegendes

- **Multimedia:** Digitale Informationen, die sich aus verschiedenen Medientypen zusammensetzen
 - Textdaten, Audio- und Videodaten, Bilder, Fotos usw.
- Multimedia-Verarbeitung hat immense Anforderungen an Netzwerkprotokolle
- **Multimedia-Anwendungen**
 - Video-Übertragungen, Video-on-Demand, Web-TV
 - IP-Telefonie
 - Internet-Radio
 - Telefonkonferenzen, Videokonferenzen
 - Interaktive Internet-Spiele

→ Realzeitanforderungen!!!

Typisches Multimedia-Netzwerk im Intranet

- Beispielszenario im Home-Office



Spezielle Anforderungen

- Hohe Übertragungsraten
- Hohe Verzögerungssensibilität
- Echtzeitanforderungen

→ Komprimierungstechniken erforderlich:

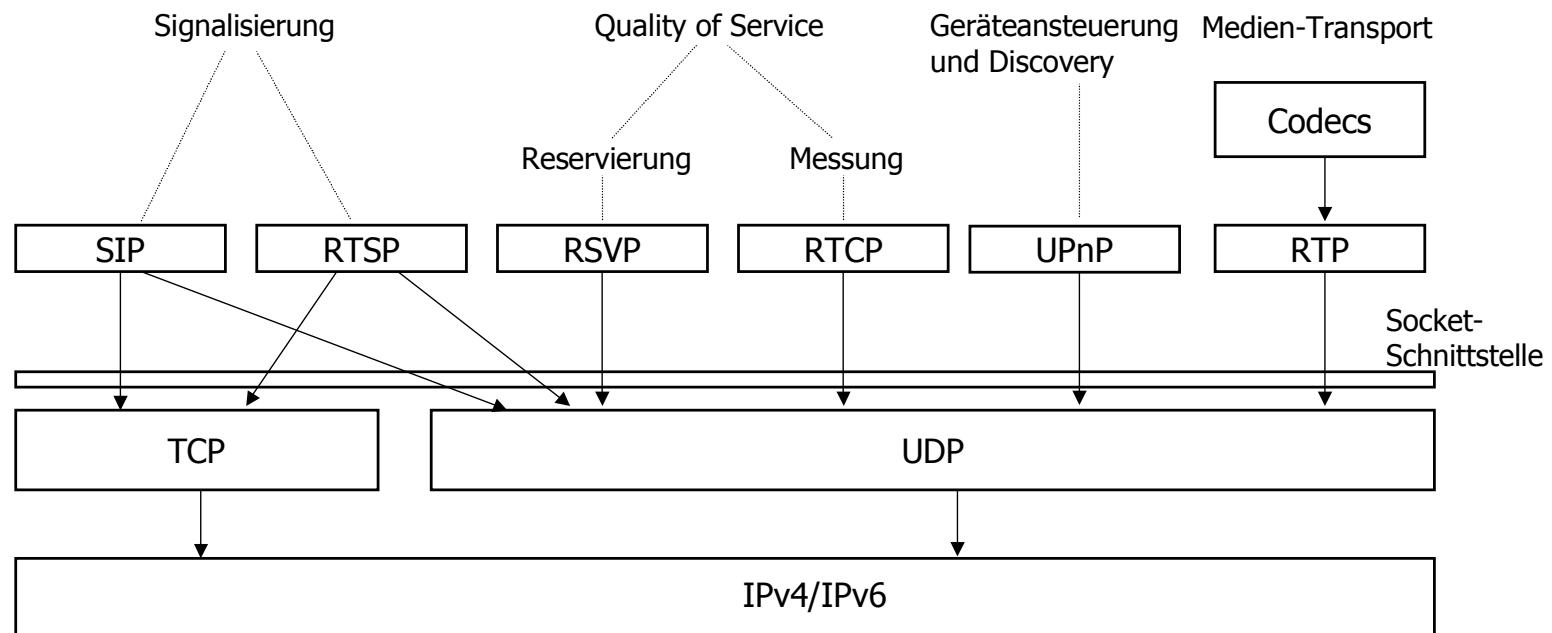
- JPEG
- MP3
- MPEG,...

→ Hohe Anforderungen an Quality of Service

→ Das Internetprotokoll IPv4 kann hier mit seinem Best-Effort-Ansatz nicht so viel bieten

Realtime-Multimedia-Protokolle

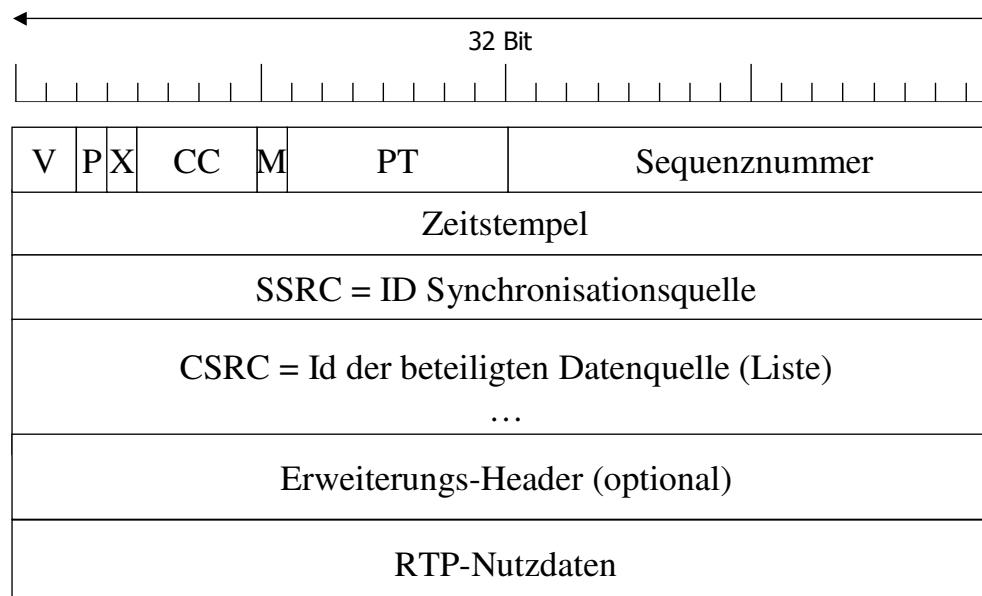
■ Überblick



Realtime-Multimedia-Protokolle

Beispiel: RTP

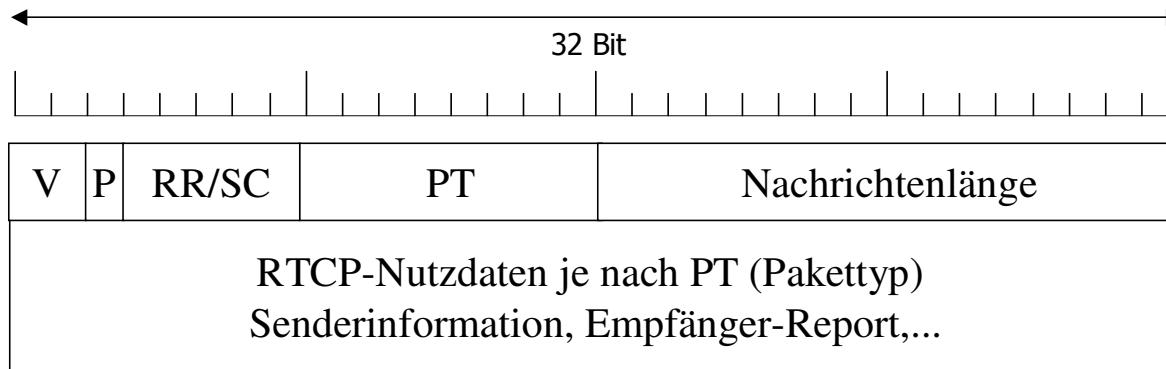
- RTP = Realtime Transport Protocol
- Aufgabe: „Transportprotokoll“ zur Beförderung von Medienströmen
- RFC 3550



Realtime-Multimedia-Protokolle

Beispiel RTCP

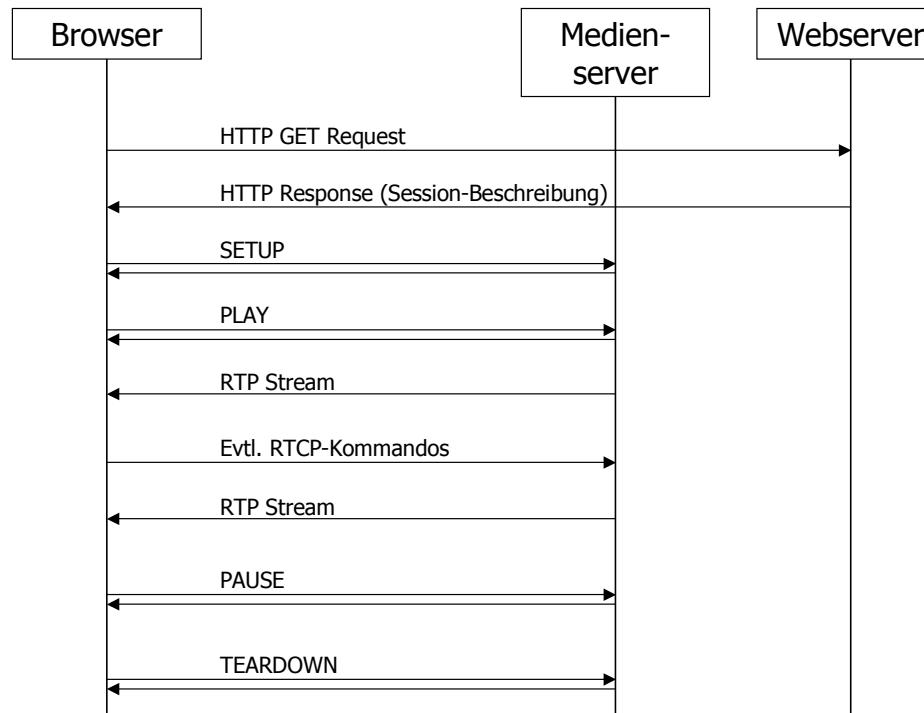
- RTCP = Real Time Transport Control Protocol
- Aufgabe: Kontrollprotokoll für Medienströme
- Auch im RFC 3550 genormt



Realtime-Multimedia-Protokolle

Beispiel RTSP

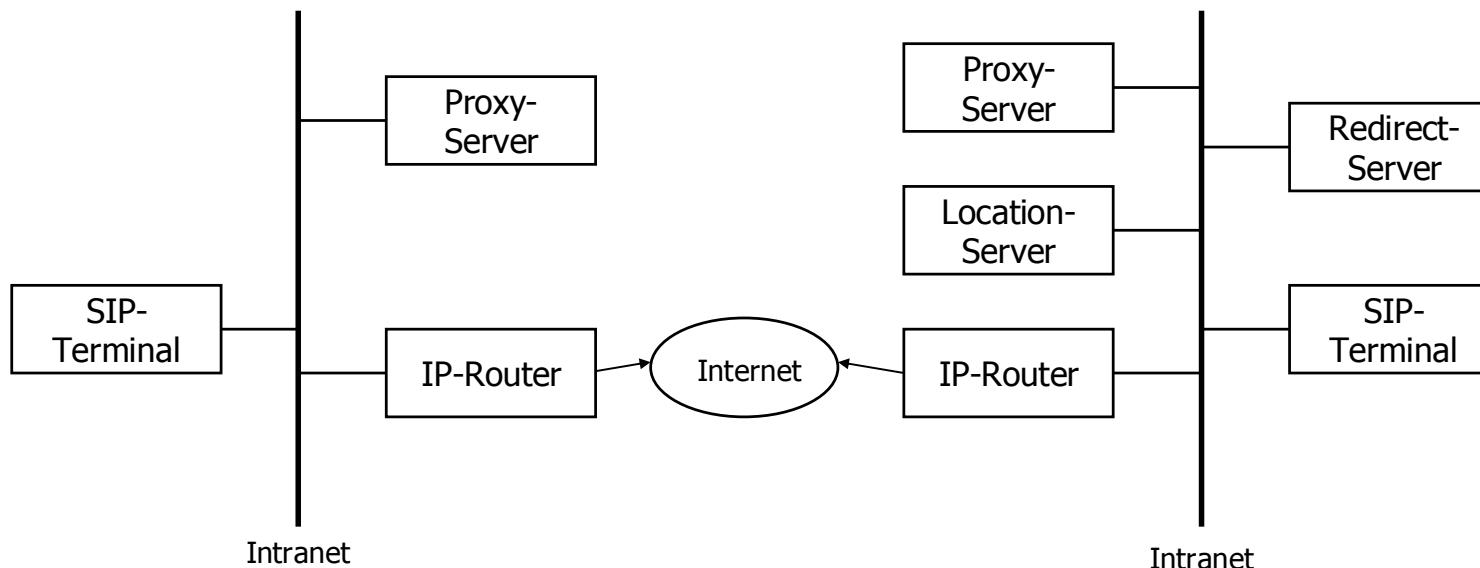
- RTSP = Real-Time Streaming Protocol
- Aufgabe: Steuerung von Datenströmen (z.B. Medienströmen)
- RFC 2326



Realtime-Multimedia-Protokolle

Beispiel SIP

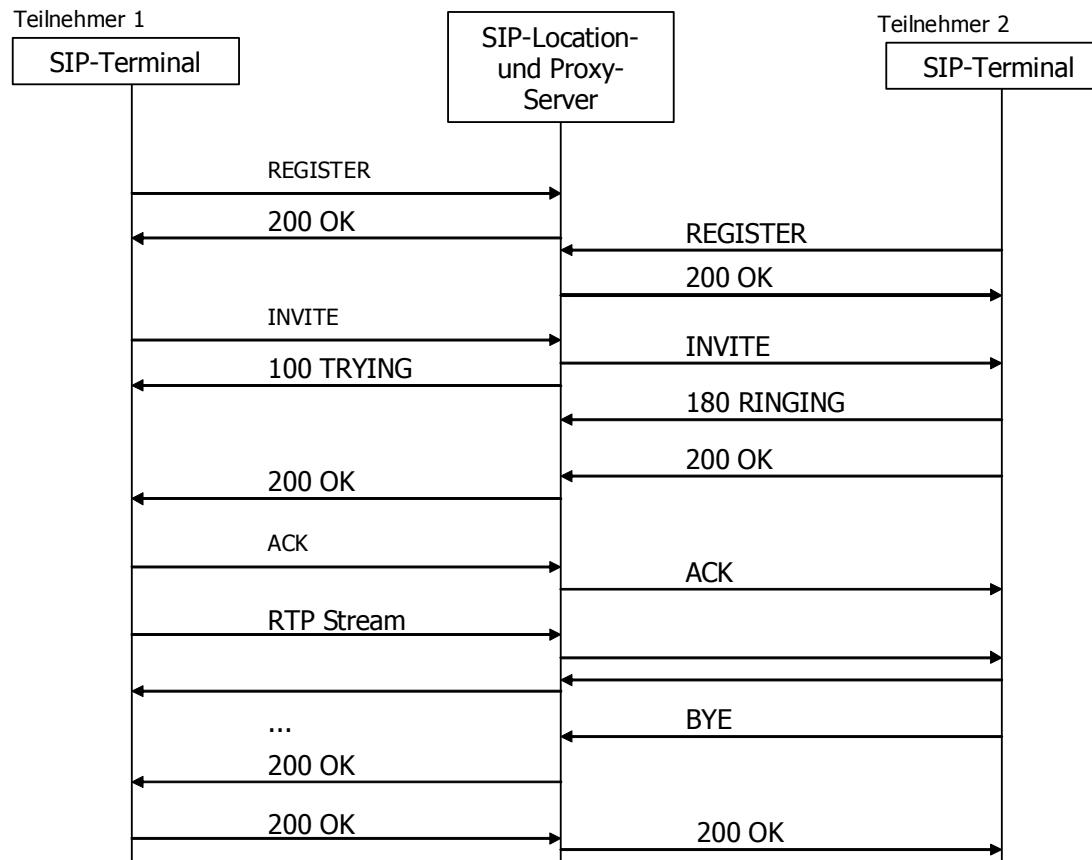
- Session Initiation Protocol
- Aufgabe: Aufbau und Unterhaltung von Sitzungen
- Typische Konfiguration



Realtime-Multimedia-Protokolle

SIP

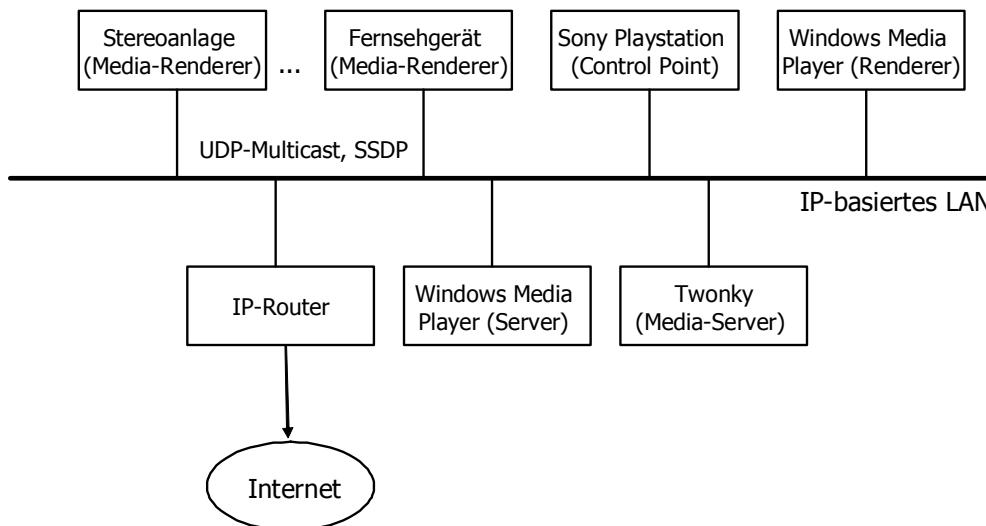
- Typischer Ablauf einer SIP-Kommunikation



Realtime-Multimedia-Protokolle

Beispiel UPnP AV

- UPnP AV = Universal Plug & Play Audio Video
 - Von *Digital Living Network Alliance (DLNA)* unterstützt
- Beispiel einer Home-Umgebung:
 - Media-Server, Media-Renderer, Control-Point



Rückblick

1. Client-/Server versus Peer-to-Peer (P2P)
2. Domain Name System
3. HTTP
4. E-Mail
5. Multimedia-Protokolle

Datenkommunikation

Mobile IP und TCP

Wintersemester 2012/2013

Überblick

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

1. Einführung

2. Mobile IP

- Mobilität und IP-Adressvergabe
- Dreiecksrouting
- Reverse Tunneling
- Handoff / Roaming
- Optimierungen
- Mobilitätsunterstützung bei IPv6

3. Mobile TCP

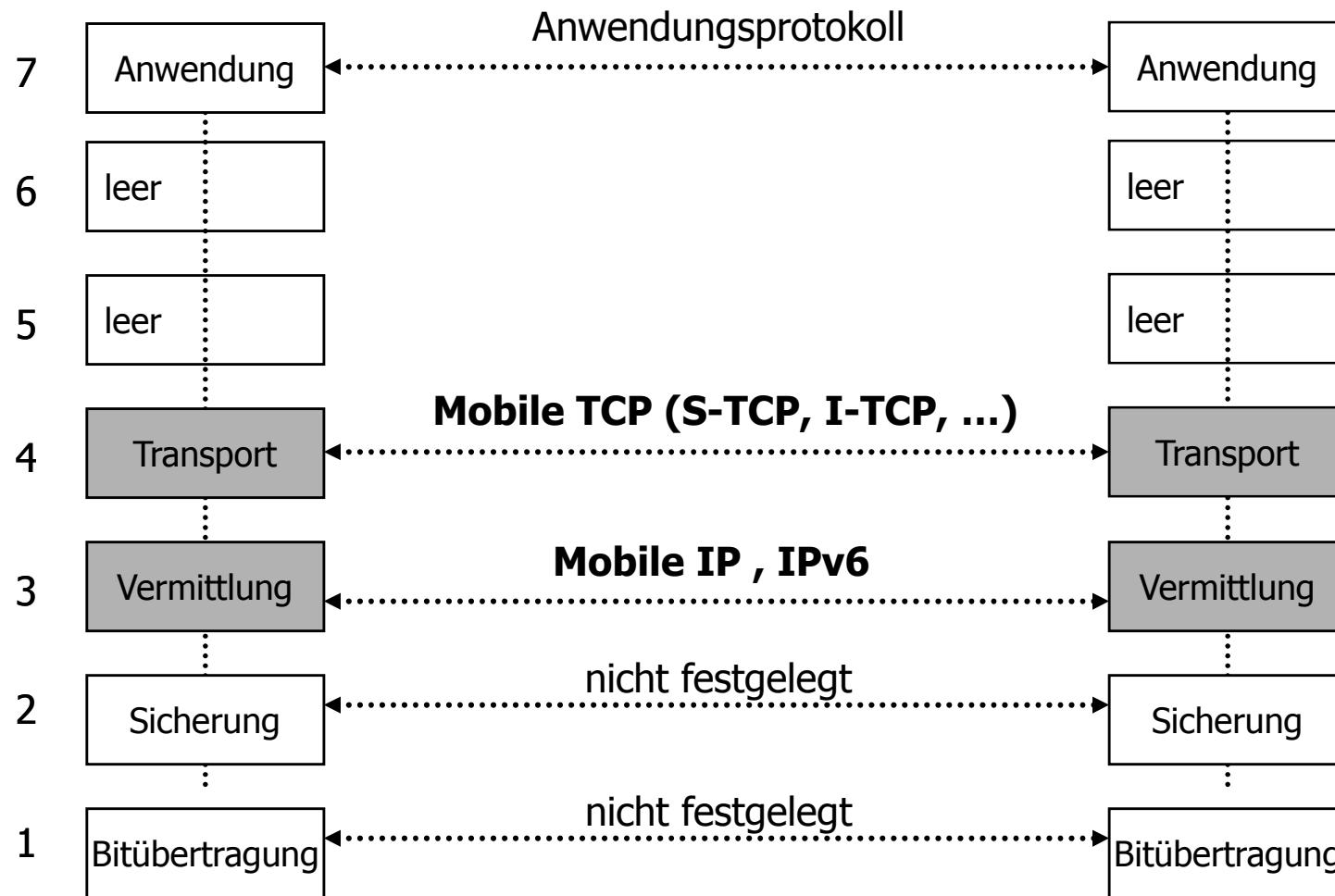
- Wiederholung und Überblick
- PEP – Performance Enhancing Proxy
- Indirektes TCP (I-TCP)
- Snooping TCP (S-TCP)

Überblick – Beispiele mobiler Kommunikation

- Funk (Rundfunk, Richtfunk, Amateurfunk)
 - Mobiltelefone (z.B. GSM, GPRS, HSCSD, UMTS)
 - Wireless LAN
 - Infrarot
 - Bluetooth
-
- Im Gegensatz zur drahtgebundenen Kommunikation ermöglicht die mobile Kommunikation eine nahezu freie Bewegung während der Nutzung des Kommunikationsdienstes
 - Was bedeutet mobil?

Überblick

Einordnung in die TCP/IP-Protokollfamilie

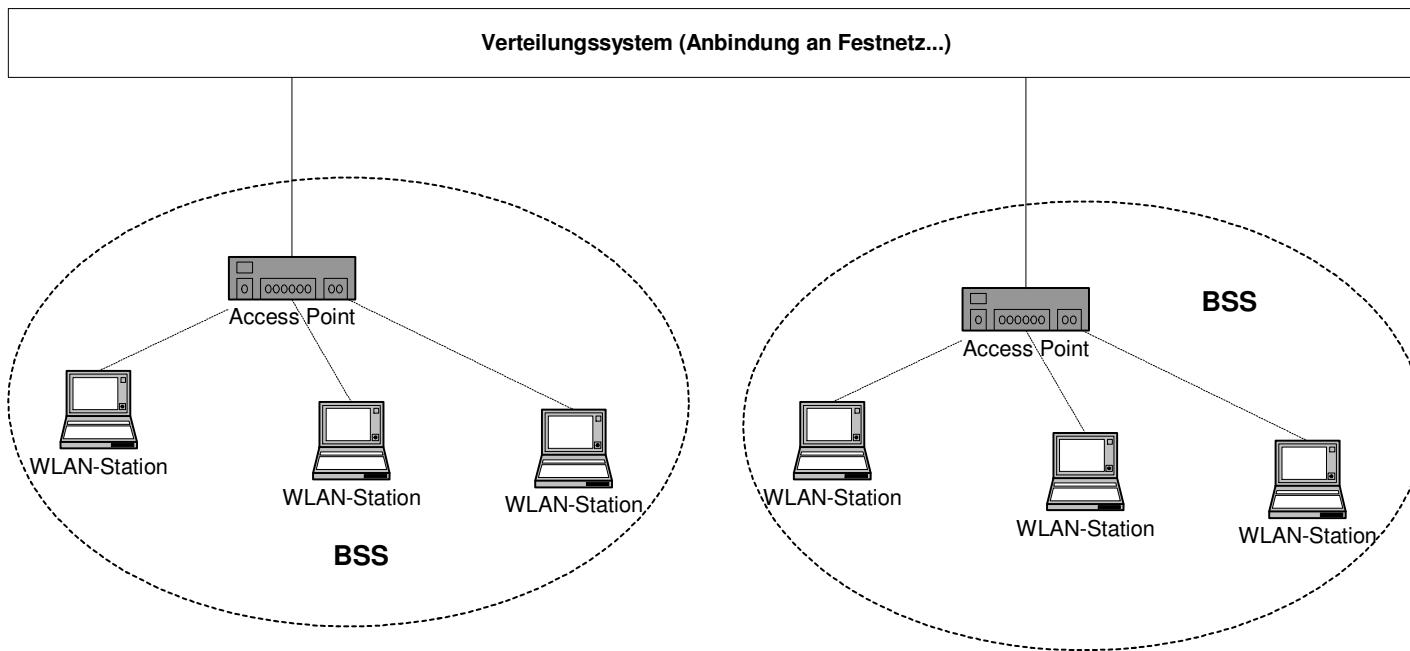


Einschub: Wireless LAN (1)

- IEEE 802.11a-n (bis 600 Mbit/s) und weitere
- Nutzung in Hotels, Flughäfen, Restaurants, (Autos)
- Basisstationen (Access Points) erforderlich
- Infrastruktur-Modus und Ad-hoc-Modus
- Zugriff auf das Medium (Luft)
 - **CSMA/CA:** „CA“ steht für Collision Avoidance, also (möglichst) Vermeiden von Kollisionen
 - Prinzip: „Listen before talk“
 - Medium wird hierzu abgehört und es wird erst übertragen, wenn es frei ist
 - Von einer Station wird vor dem Senden einer Nachricht zunächst eine kurze Steuernachricht abgeschickt, die vom Empfänger bestätigt wird
 - Erst nach dem Empfang der Bestätigung beginnt der Sender die eigentlichen Daten zu senden
- Problem: „Hidden Stations“ (versteckte Endgeräte)

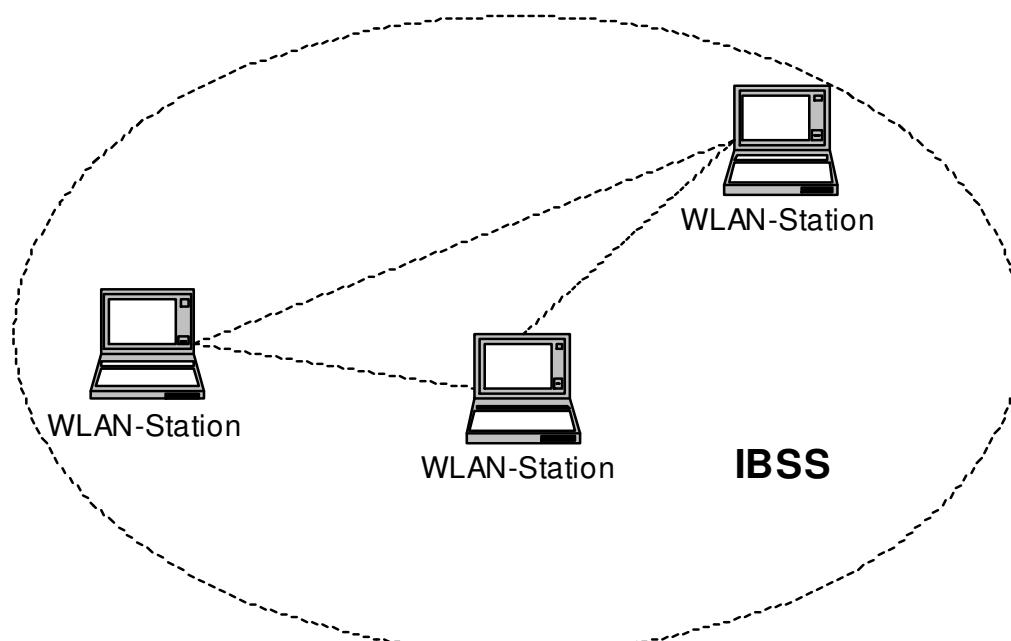
Einschub: Wireless LAN: Infrastrukturmodus (2)

- Alle Stationen und der für sie zuständige Access-Point werden zusammen als Basic Service Set (BSS) bezeichnet



Einschub: Wireless LAN: Ad-hoc-Modus (3)

- IBSS = Independent Basic Service Set



Überblick

1. Mobile IP

- **Mobilität und IP-Adressvergabe**
- Dreiecksrouting
- Reverse Tunneling
- Handoff / Roaming
- Optimierungen
- Mobilitätsunterstützung bei IPv6

2. Mobile TCP

Wiederholung – Vermittlungsschicht insb. IP (1)

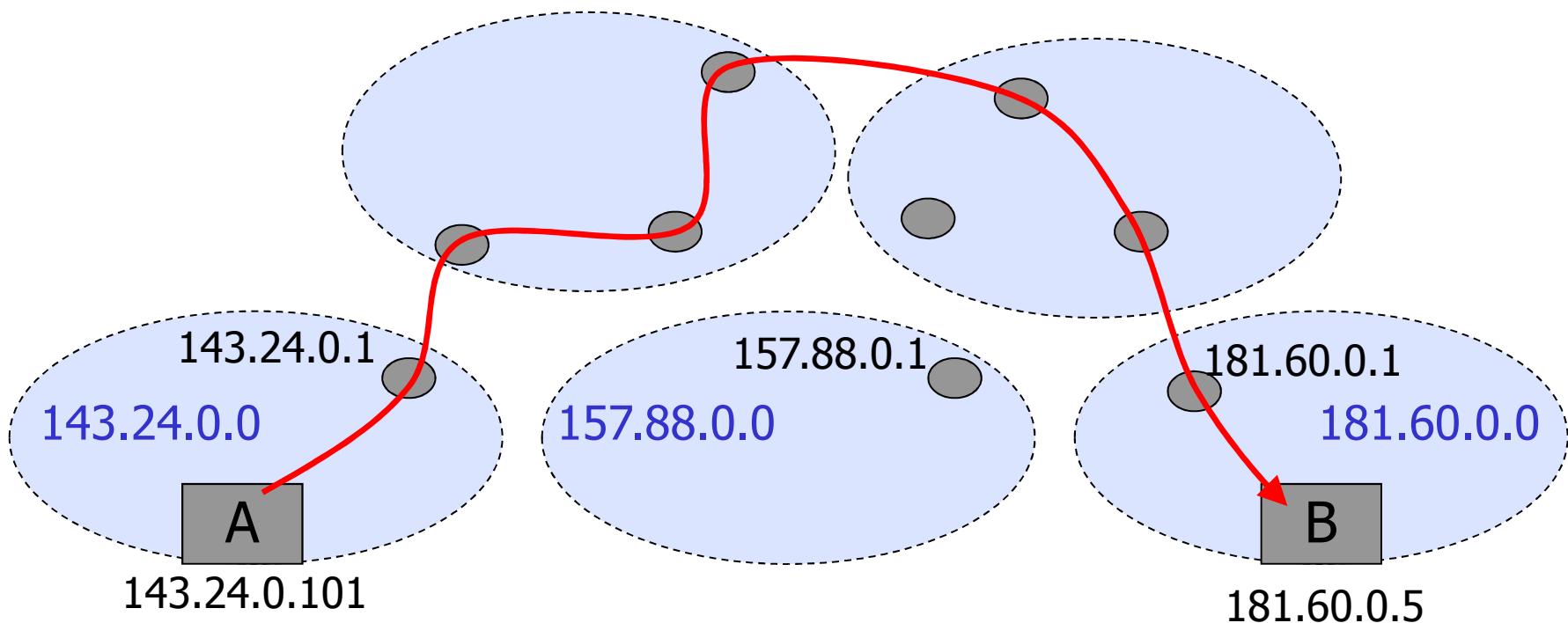
- Das Internet ist eine **hierarchische Organisation** bestehend aus Teilnetzen (Autonome Systeme)
- Eine grundlegende Aufgabe der Vermittlungsschicht ist die Wegewahl (**Routing**)
- IP ist ein **paketvermittelndes** und **verbindungsloses** Protokoll der Vermittlungsschicht
- Übertragungswege werden von **Knoten zu Knoten** bereitgestellt (Teilstrecken)
- Voraussetzung ist eine eindeutige **Adressierung**

Wiederholung – Vermittlungsschicht, IP (2)

- IP (RFC 791) wurde 1981 spezifiziert und wurde für die **drahtgebundene** und **stationäre** Kommunikation konzipiert
- IP setzt zur Optimierung der Wegewahl **hierarchisches Routing** ein
- Aufteilung des Internets in Teilnetze
- IP-Adressen bestehen aus einem **Netz- und Host-Anteil** um die Teilnetze identifizieren zu können
- IP-Adresse ist nur innerhalb des entsprechenden Teilnetzes gültig

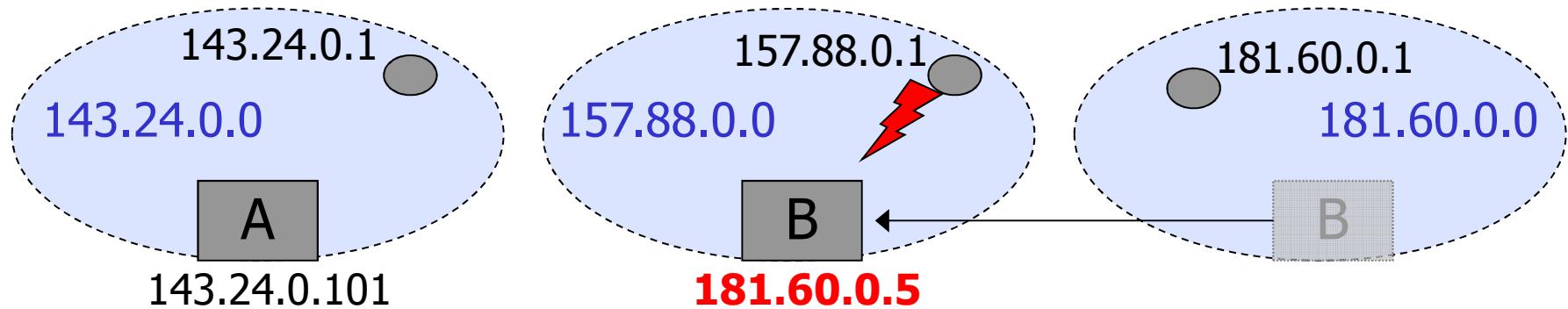
Mobilität und IP-Adressvergabe (1)

- Kommunikation von Knoten A zu Knoten B
- Beide befinden sich in eigenen Teilnetzen
- Hierarchisches Routing



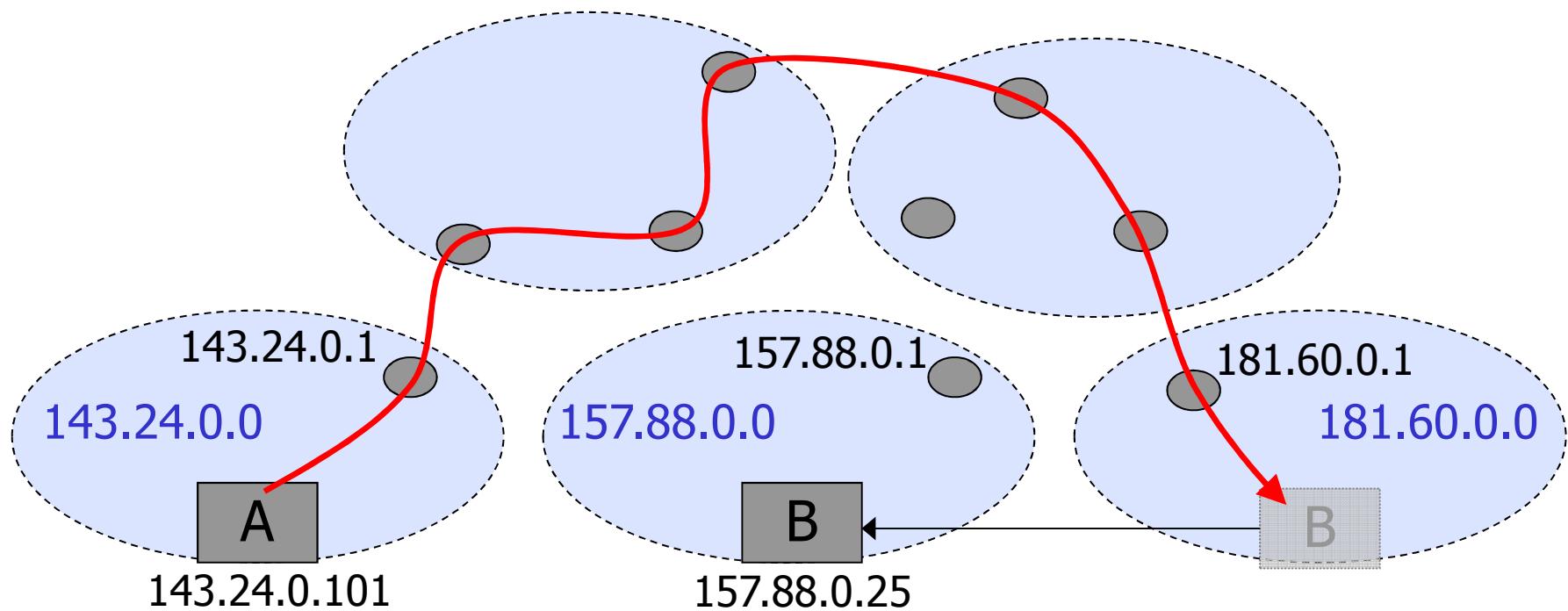
Mobilität und IP-Adressvergabe (2)

- Knoten B bewegt sich in ein anderes Teilnetz
- Knoten B behält seine IP-Adresse (feste IP-Adresse)
- **Adresse ist topologisch nicht korrekt!**
- Knoten B kann keine Pakete empfangen



Mobilität und IP-Adressvergabe (3)

- Knoten B bewegt sich in ein anderes Teilnetz
- Knoten B bekommt z.B. über DHCP **neue Adresse**
- Knoten A sendet aber an die alte Adresse



Mobilität und IP-Adressvergabe (4)

- Die Mobilität kann einen **Wechsel des Teilnetzes** verursachen
 - Topologisch korrekte IP-Adresse muss konfiguriert werden
 - Kommunikationspartner sendet aber an die bekannte IP-Adresse
 - Neue IP-Adresse könnte über DNS verteilt werden Aktualisierung der DNS-Einträge jedoch zu aufwändig
-
- **Mobilität führt bei IP erst durch den Wechsel eines Teilnetzes zu Problemen**

Überblick

1. Einführung

2. Mobile IP

- Mobilität und IP-Adressvergabe
- **Dreiecksrouting**
- Reverse Tunneling
- Handoff / Roaming
- Optimierungen
- Mobilitätsunterstützung bei IPv6

3. Mobile TCP

Mobilität und IP-Adressvergabe - Mobile IP

- Zur Version 4 des Internet Protokolls (IPv4) wurde eine Erweiterung zur Unterstützung der Mobilität beschrieben
- Die Erweiterung wird **Mobile IP** genannt und ist im wesentlichen im RFC 3344 spezifiziert.
- Mobile IP ist kompatibel zum normalen IPv4

- Mobile IP ermöglicht einen Teilnetzwechsel unter Beibehaltung einer festen IP-Adresse

Dreiecksrouting – Terminologie (RFC 3344) (1)

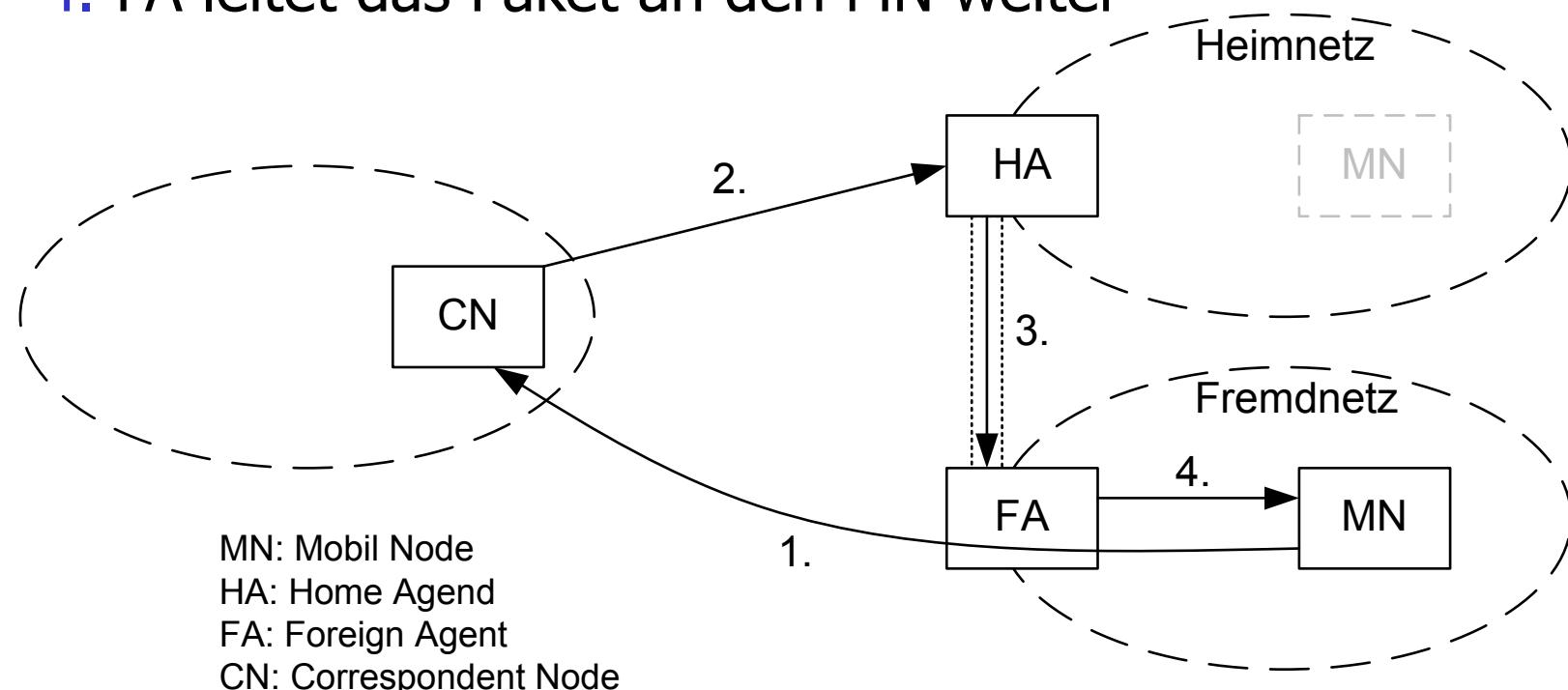
- **Mobile Node (MN)**: Ein Host (oder Router), der seinen Anschluss an ein Teilnetz wechselt
- **Home Network**: Ein Teilnetz, welches dem *Mobile Node* als ursprüngliches Netzwerk zugeordnet ist
- **Home Address**: Eine IP-Adresse, die einem Mobile Node, unabhängig von möglichen Teilnetzwechseln, fest zugeordnet ist. Die Netzwerknummer der *Home Address* muss der Identifikation des Heimnetzes (*Home Network*) entsprechen
- **Home Agent (HA)**: Ein spezieller Router im Heimnetz (*Home Network*), der die Mobilitätsunterstützung ermöglicht

Dreiecksrouting – Terminologie (RFC 3344) (2)

- **Foreign Network (FN):** Ein Teilnetz, ungleich dem Heimnetz (Home Network), in das der Mobile Node während der Kommunikation wechselt
- **Care-of-Address (CoA):** Eine topologisch korrekte IP-Adresse, die dem Mobile Node im Fremdnetz (Foreign Network) zugewiesen wird
- **Foreign Agent (FA):** Ein spezieller Router in einem Fremdnetz (Foreign Network), der die Mobilitätsunterstützung ermöglicht
- **Correspondent Node (CN):** Ein beliebiger Host, mit dem Mobile Node über IP kommuniziert

Dreiecksrouting – Funktionsweise (1)

1. *Mobile Node* sendet Paket an *Correspondent Node*
2. CN antwortet an die Adresse aus dem Heimnetz
3. *Home Agent* leitet die Pakete an das Fremdnetz weiter
4. FA leitet das Paket an den MN weiter

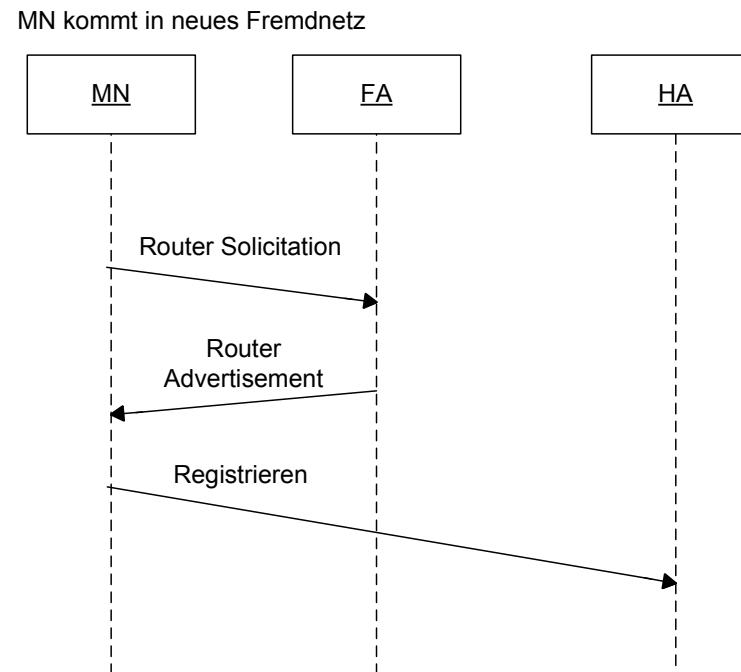


Dreiecksrouting – Funktionsweise , Nachrichtenfluss (2)

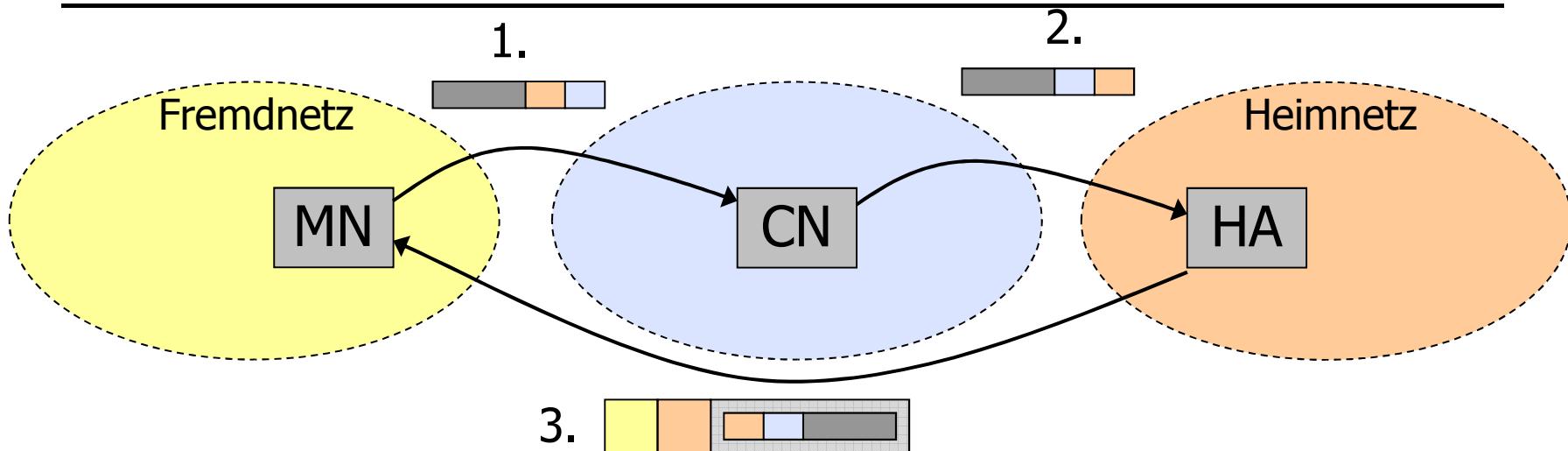
- MN benötigt eine CoA, wenn er in ein Fremdnetz kommt
- Dazu sendet MN eine **Router-Solicitation-Nachricht** (auf Basis von ICMP)
- FA antwortet mit einer **Router Advertisement**-Nachricht
 - (Erweitert um sog. Mobility Agent Advertisement Extension)
 - → CoA wird mitgeteilt
 - FA verwaltet „Visitor-Liste“
- MN muss dann **CoA an HA melden** (registrieren)
 - Information heißt auch „Mobility-Bindings“

Dreiecksrouting – Funktionsweise, Nachrichtenfluss (3)

- Grober Kommunikationsablauf bei Eintritt eines MN in ein Fremdnetz
- FA sendet zudem „Mobility Agent Advertisements“, um CoAs zu vergeben

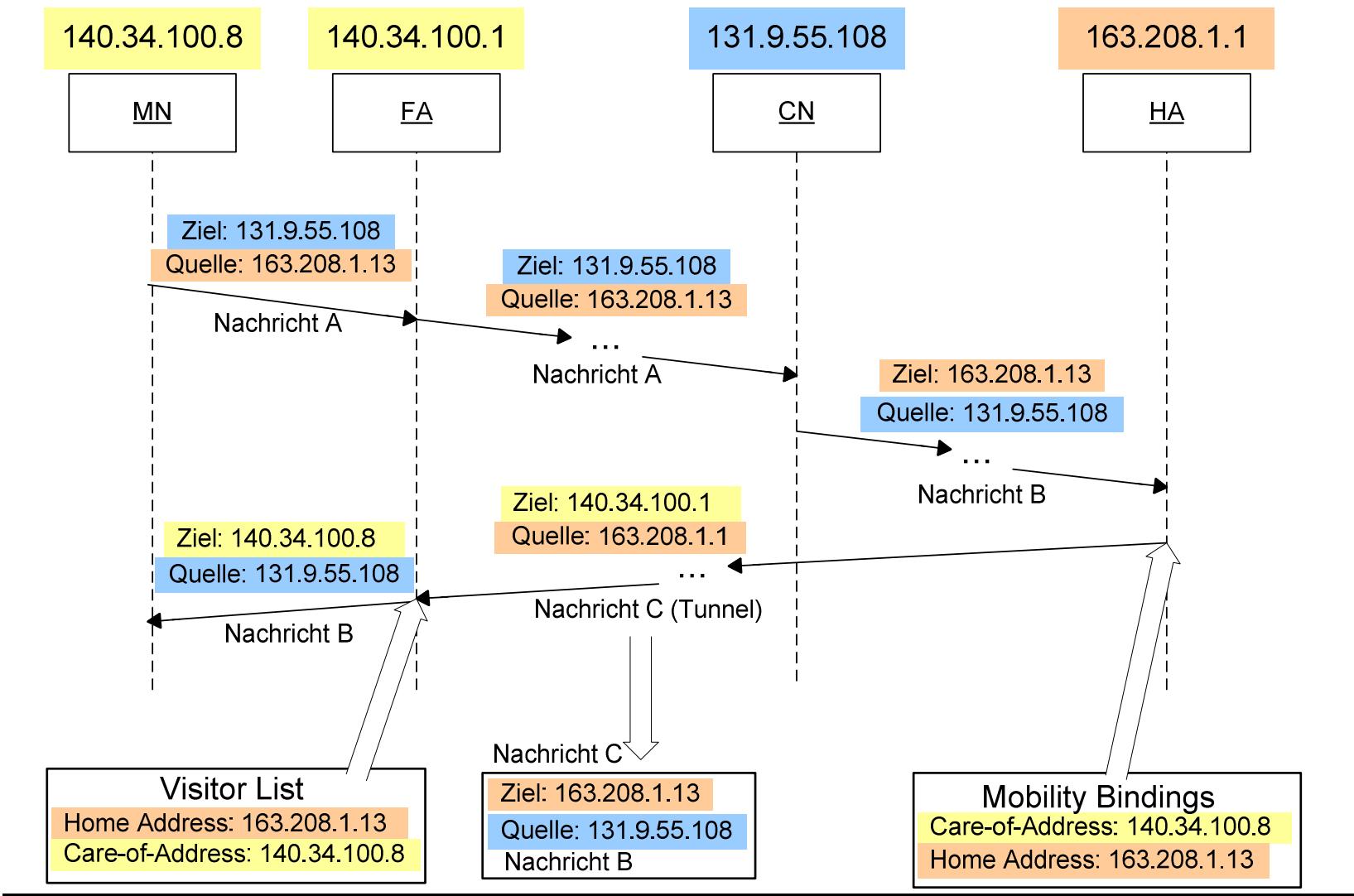


Dreiecksrouting – Funktionsweise, Sendephase (4)



- 1.** MN sendet direkt an CN:
 - Quelladresse: IP-Adresse aus dem Heimnetz des MN
 - Zieladresse: IP-Adresse des CN
- 2.** CN antwortet an MN (über HA)
 - Quelladresse: IP-Adresse des CN
 - Zieladresse: Quell-IP-Adresse von MN aus dem Heimnetz
- 3.** HA übermittelt Antwort an MN über **IP-Tunnel** an das Fremdnetz:
 - HA und FA sind die Endpunkte des Tunnels

Dreiecksrouting – Funktionsweise, Sendephase (5)

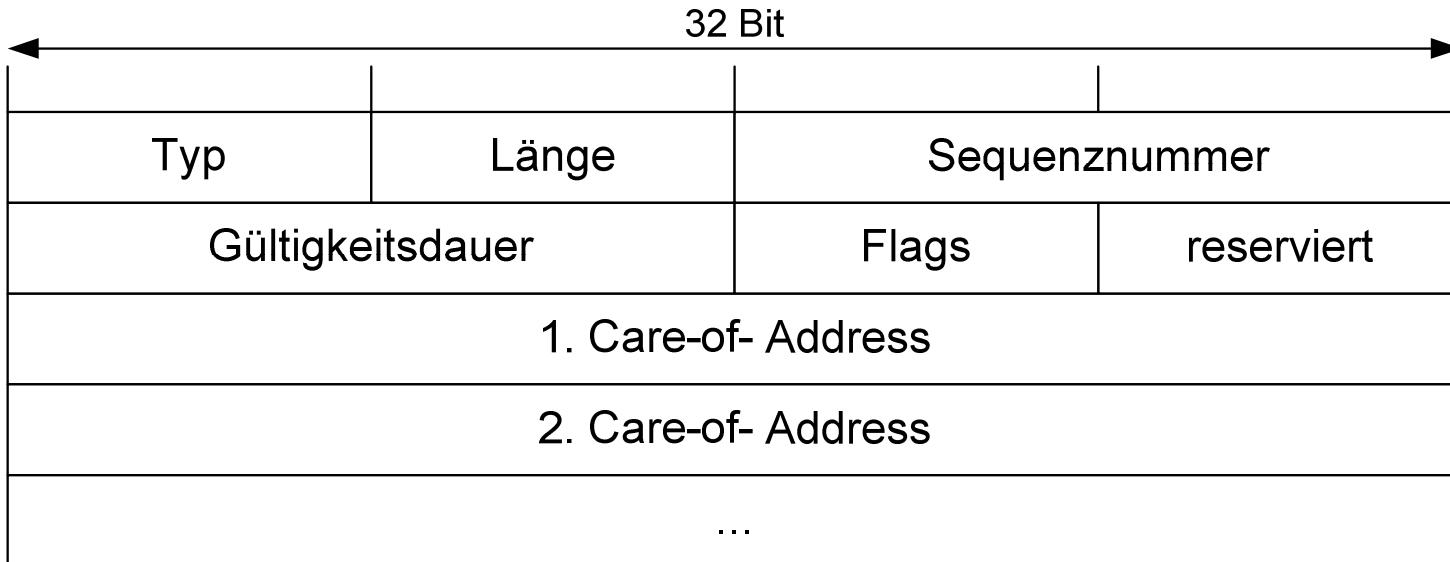


Dreiecksrouting – Funktionsweise, Sendephase (6)

- Mobilitätsunterstützung für IPv4 -> RFC 3344
- Automatische Anpassungen der IP-Adresse an das aktuelle Teilnetz (*Agent Erkennung*)
- Aber Erreichbarkeit über eine feste IP-Adresse
- Router im Heimnetz (*HA*) leitet die Pakete an das aktuelle Teilnetz weiter
- Dem *HA* muss die aktuelle IP-Adresse mitgeteilt werden (Registrierung der Care-of-Adresse)
- Kommunikationspartner kennen nur die feste Adresse aus dem Heimnetz

Dreiecksrouting – Agent-Erkennung

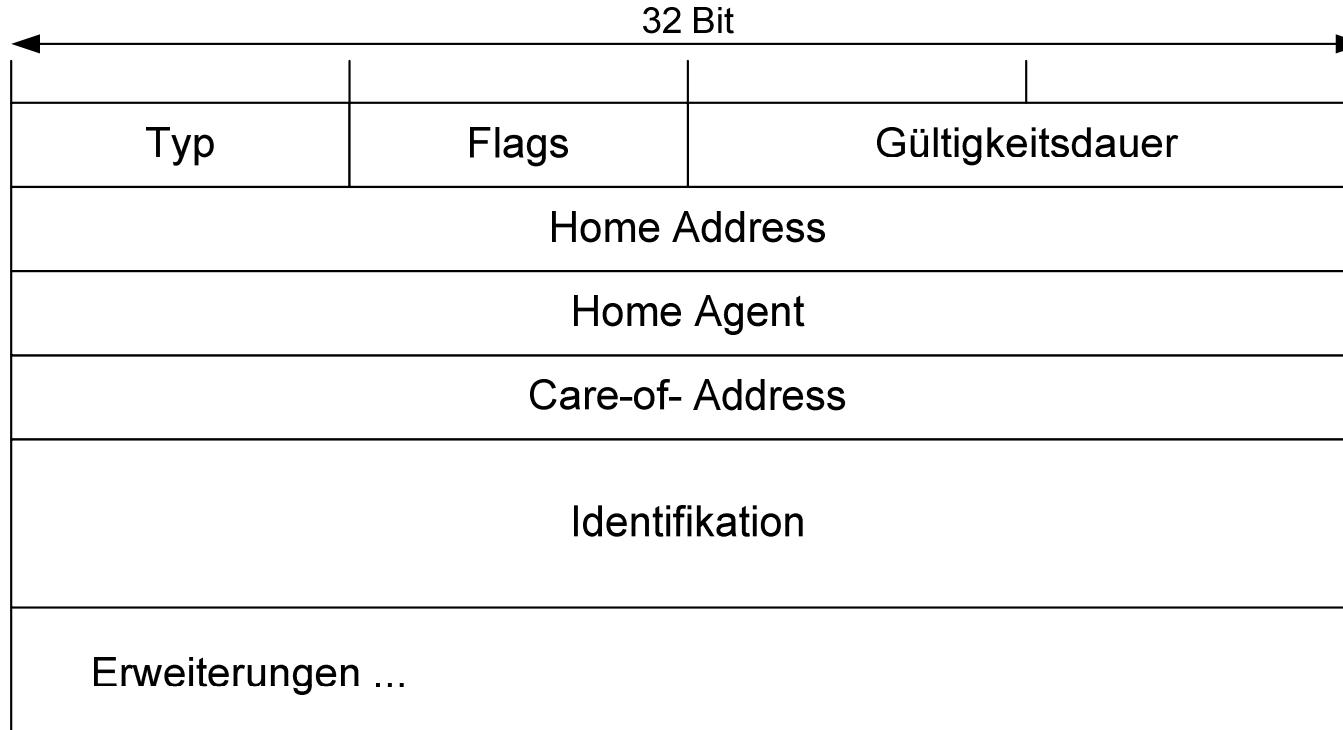
- Mobility Agent Advertisement Extension nach RFC 3344



- Auf Basis von ICMP
- MN erkennt am Netz-Anteil der Absenderadresse das aktuelle Teilnetz und somit auch einen Teilnetzwechsel

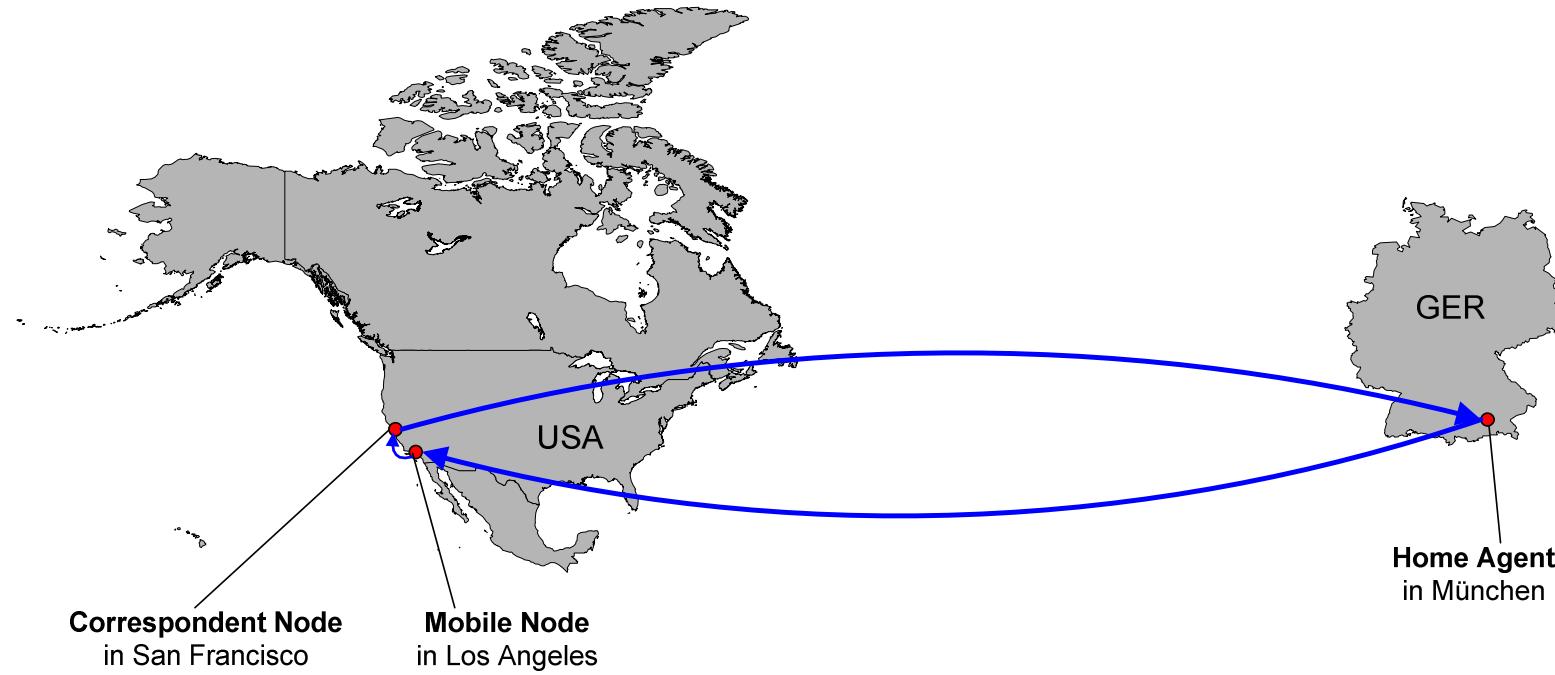
Dreiecksrouting – Registrierung der Care-of-Adresse

- Registration Request nach RFC 3344



- MN meldet dem HA seine Care-of-Adresse
- HA speichert diese „Mobility-Bindings“

Dreiecksrouting – Probleme – ineffizienter Pfad



- Pakete vom CN zum MN müssen über den HA weitergeleitet werden
- Im Vergleich zur direkten Route kann dies ein deutlich ineffizienterer Pfad sein

Dreiecksrouting – weitere Probleme

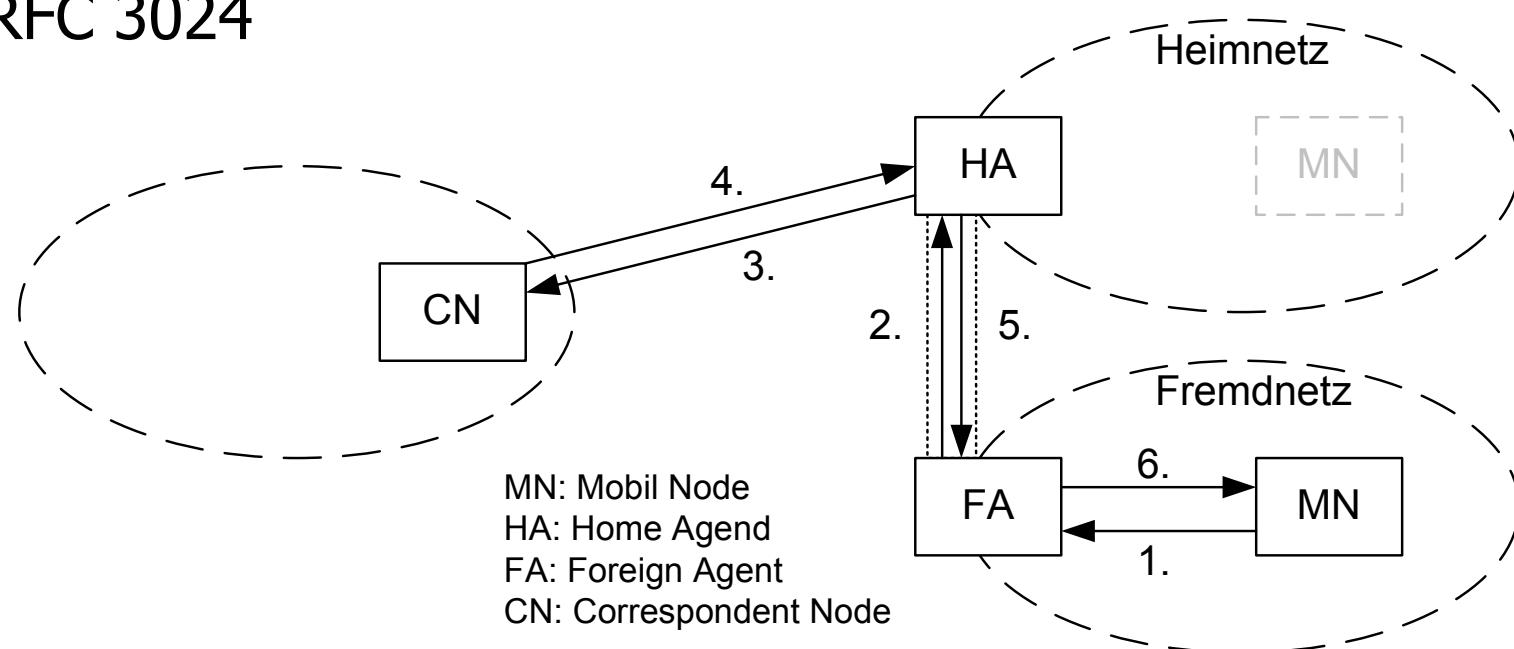
- Alle Pakete vom *Correspondent Node* zum *Mobile Node* werden über den *Home Agent* geroutet
 - Neben der ineffizienten Route wird der *Home Agent* somit zum **kritischen Punkt** (*Single-Point-of-Failure*)
 - Fällt der *Home Agent* aus ist keine Kommunikation zum *Mobile Node* möglich
-
- Der MN verwendet auch in Fremdnetzen die IP-Adresse aus dem Heimnetz als Quelladresse
 - Eine Firewall im Fremdnetz kann dies als Fälschung (IP-Spoofing) erkennen und das Paket verwerfen

Überblick

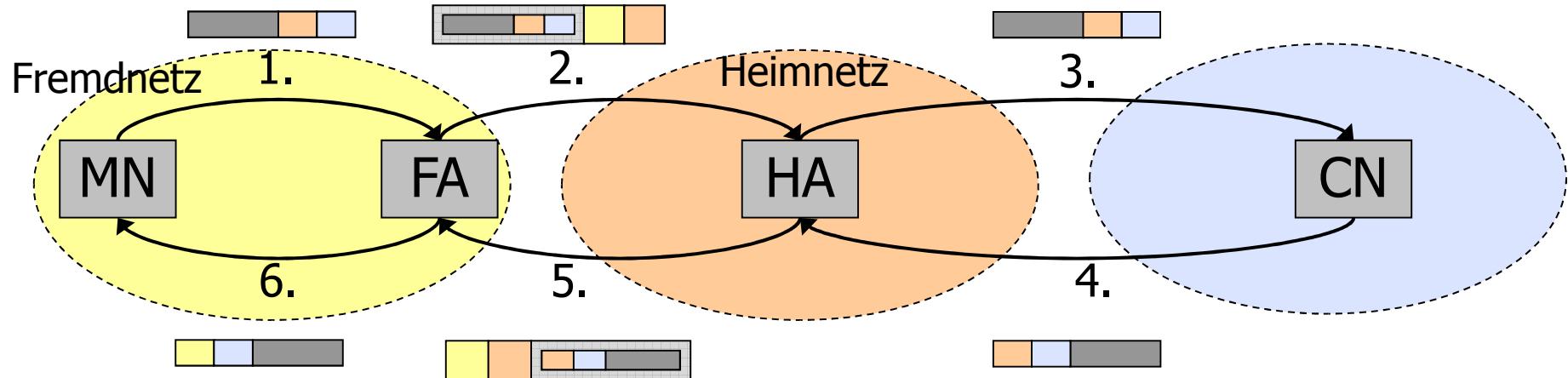
1. Einführung
2. **Mobile IP**
 - Mobilität und IP-Adressvergabe
 - Dreiecksrouting
 - **Reverse Tunneling**
 - Handoff / Roaming
 - Optimierungen
 - Mobilitätsunterstützung bei IPv6
3. Mobile TCP

Reverse Tunneling – Funktionsweise (1)

- Kommunikation zwischen MN und CN wird in beiden Richtungen über den HA geroutet
- Bidirekionaler IP-Tunnel zwischen HA und FA
- Keine IP-Spoofing-Erkennung
- RFC 3024



Reverse Tunneling – Funktionsweise (2)



Nachricht 1 und 3:

- Quelladresse: IP-Adresse aus dem Heimnetz
- Zieladresse: IP-Adresse der CN

Nachricht 2 und 5:

- Bidirektonaler IP-Tunnel zwischen FA und HA

Nachricht 4 und 6:

- Quelladresse: IP-Adresse der CN
- Zieladresse: IP-Adresse aus dem Heimnetz oder CoA

Reverse Tunneling – Probleme

- Kommunikation in beide Richtungen wird über den HA geroutet
- Die Probleme einer ineffizienten Route und einer möglichen Überlastung des HA (SpoF) werden somit noch verschärft
- Es besteht die Gefahr eines „Reverse-Tunnel Hijackings“ (RFC 3024) und somit eine Gefährdung des Heimnetzes. Ausreichende Authentifizierung ist notwendig

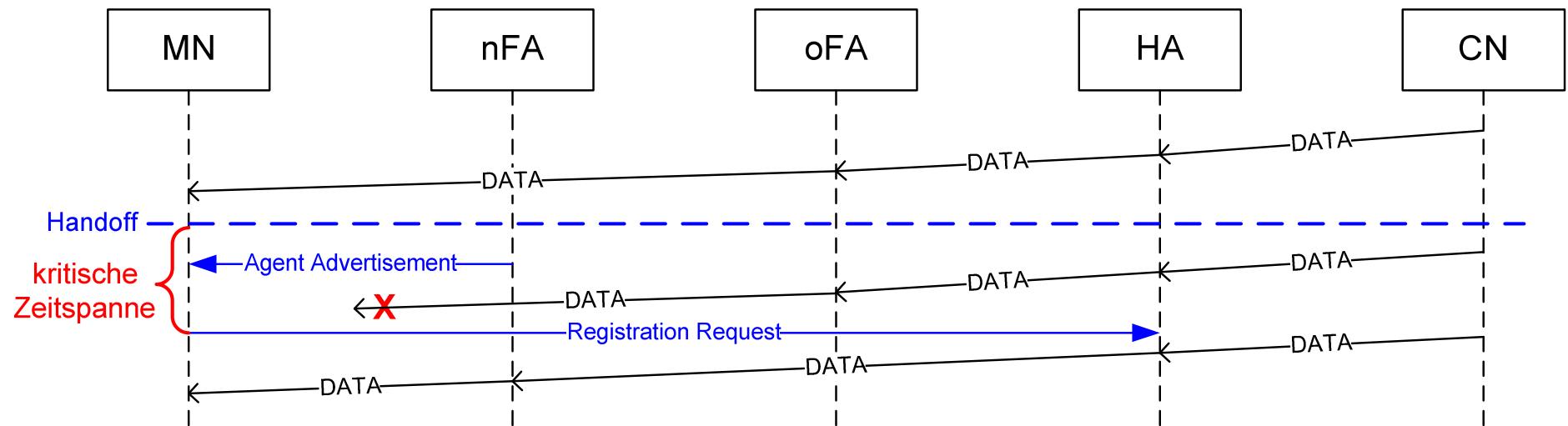
Überblick

1. Einführung
2. **Mobile IP**
 - Mobilität und IP-Adressvergabe
 - Dreiecksrouting
 - Reverse Tunneling
 - **Handoff / Roaming**
 - Optimierungen
 - Mobilitätsunterstützung bei IPv6
3. Mobile TCP

Handoff / Roaming - Überblick

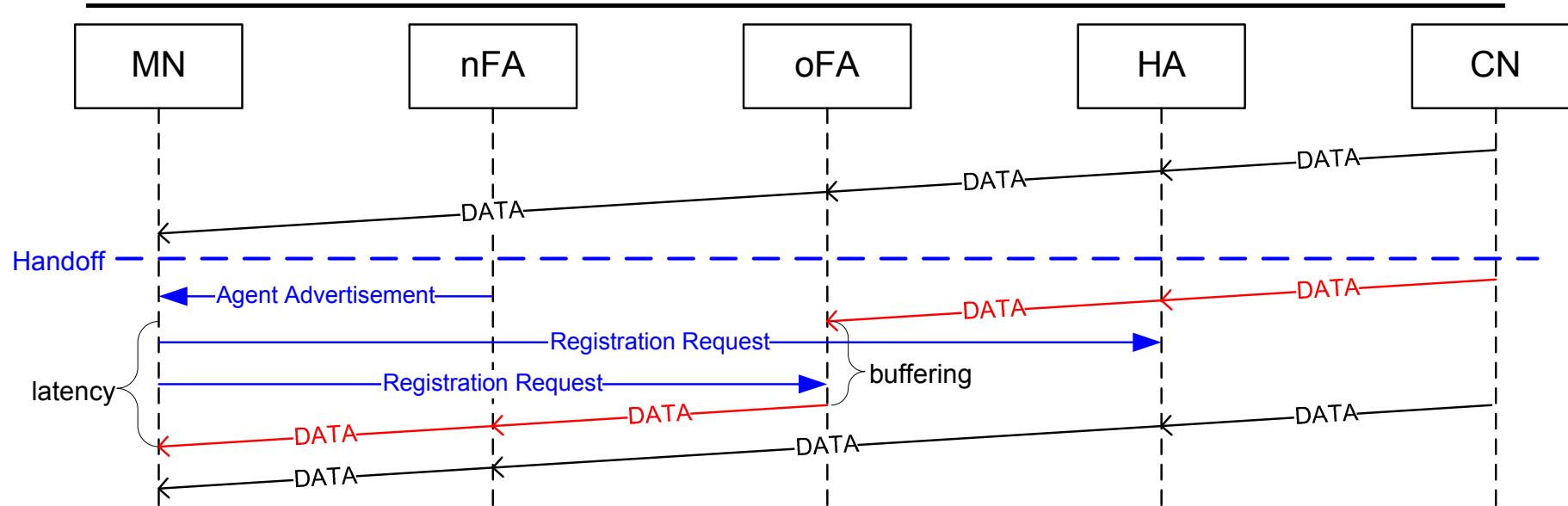
- Wechsel eines Teilnetzes während einer aktiven Kommunikation wird als Handoff / Roaming bezeichnet
- **Kritische Zeitspanne** zwischen dem Abmelden beim alten Teilnetz und dem Anmelden beim neuen Teilnetz
- Während der kritischen Zeitspanne ist der MN nicht erreichbar – es droht **Datenverlust**
- Höhere Schichten (z.B. Transportschicht) müssen den Datenverlust erkennen und beheben – dies führt zu einer **schlechteren Übertragungsleistung**

Handoff / Roaming – kritische Zeitspanne



- Teilnetzwechsel ohne speziellen Handoff-Verfahren
 - Nach dem Handoff muss der MN eine neue CoA zugewiesen bekommen und beim HA registrieren
 - In der Zwischenzeit leitet der HA Pakete für den MN noch an den alten FA weiter -> Datenverlust

Handoff / Roaming – Post-Registration-Handoff



- Nachsenden der Pakete vom alten FA zum neuen FA
- Neue CoA muss auch beim alten FA registriert werden
- Alter FA muss bis zur Registrierung eintreffende Pakete zwischenspeichern
- Beim Handoff kann es zu einer Verzögerung kommen

Überblick

1. Einführung

2. Mobile IP

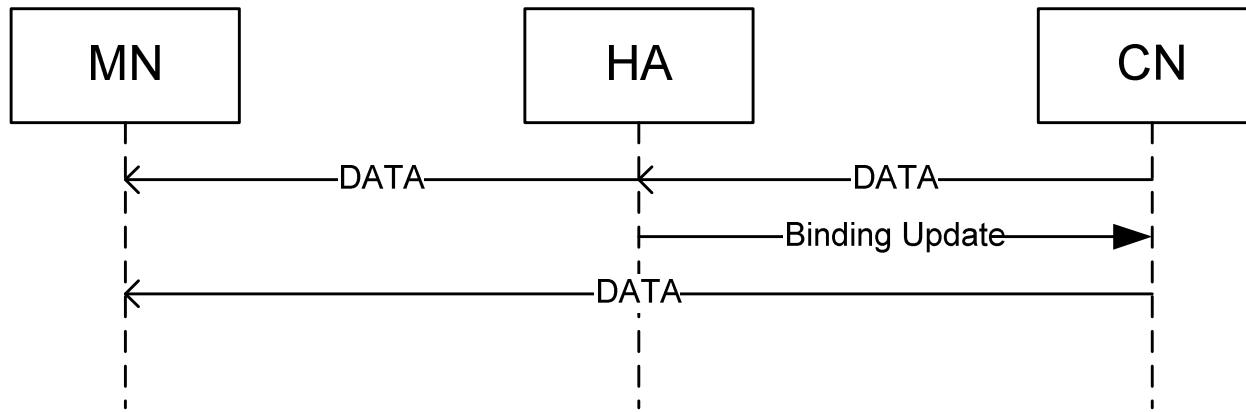
- Mobilität und IP-Adressvergabe
- Dreiecksrouting
- Reverse Tunneling
- Handoff / Roaming
- **Optimierungen**
- Mobilitätsunterstützung bei IPv6

3. Mobile TCP

Optimierung - Encapsulation in der Mobile Node

- Der MN kann Aufgaben des FA übernehmen
 - Der MN bezieht dazu (z.B. DHCP) eine gültige IP-Adresse für das Fremdnetz
-
- MN übernimmt die Registrierung der neuen IP-Adresse (*co-located Care-of Address*) beim HA
 - MN wird auch zum Endpunkt des IP-Tunnels (Encapsulation in der MN)
 - Es muss keinem FA Vertrauen entgegen gebracht werden
 - Handoff-Verfahren benötigen jedoch einen FA

Optimierung – HA Redirects



- CN antwortet zunächst an die Adresse aus dem Heimnetz
- HA leitet die Nachricht weiter und teilt der CN die aktuelle CoA der MN mit
- CN kann weitere Pakete direkt an die CoA senden

Optimierung – HA Redirects - Bewertung

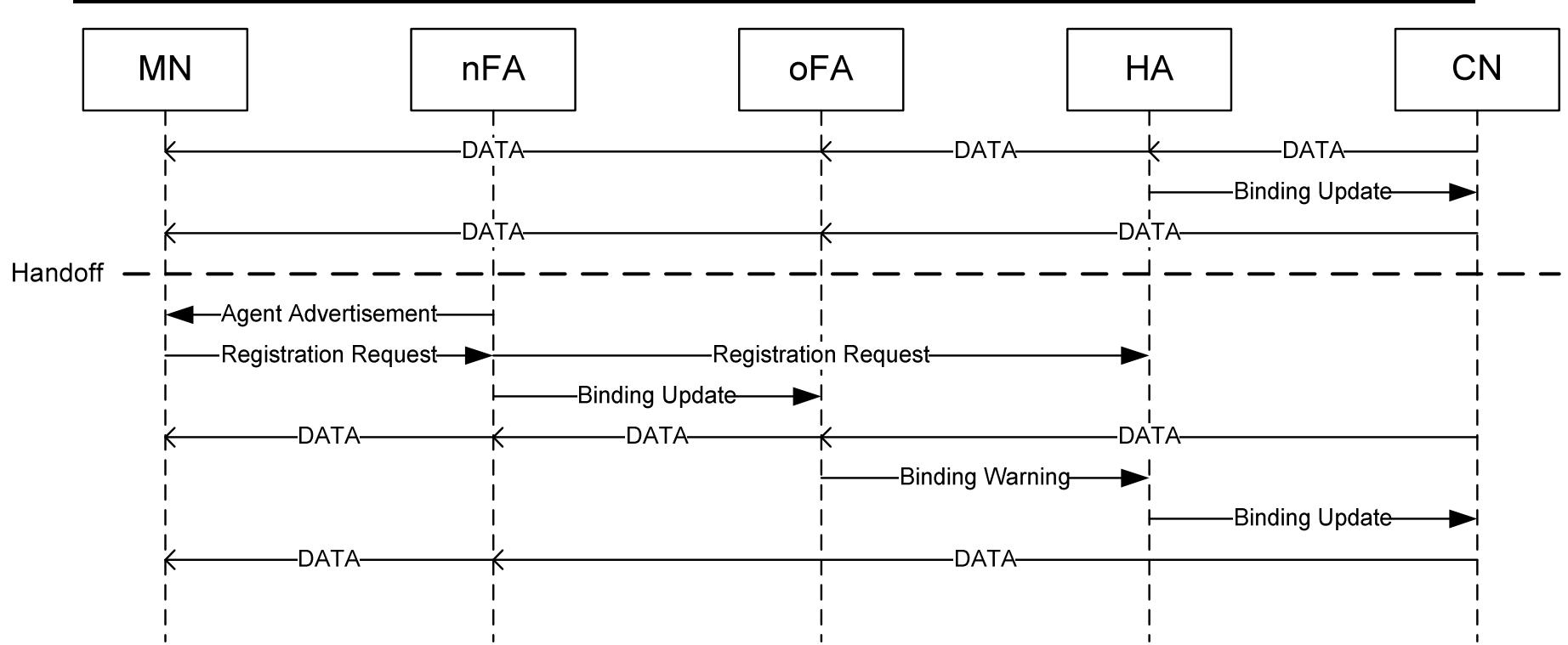
Vorteile

- Direktes und somit effizienteres Routing
- Entlastung des *Home Agents*

Nachteile

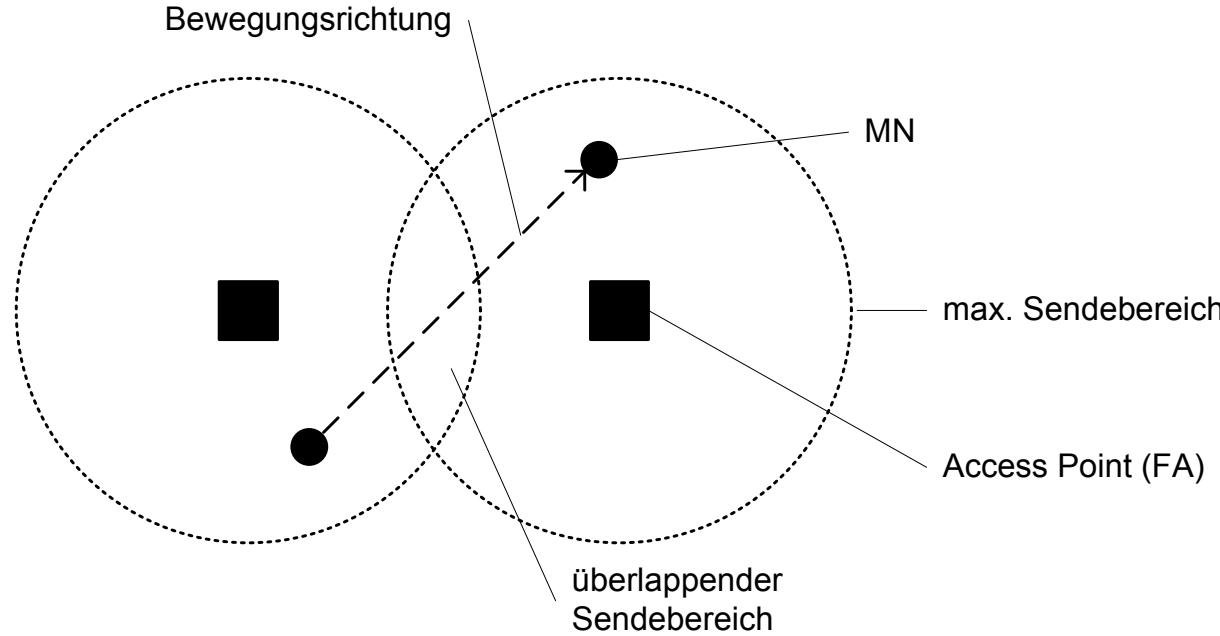
- CN muss Binding-Updates verwalten
- CN muss IP-Tunneling unterstützen
- Mit den Binding-Updates wird dem CN die aktuelle IP-Adresse des MN mitgeteilt. Es können Bewegungsdaten zu einer MN aufgezeichnet werden

Optimierung – HA Redirects - Handoff



- „Alter“ FA sende *Binding Warning* an HA
- HA sendet die neue CoA als *Bindung Update* an CN

Optimierung – Handoffverfahren



- Beim Handoff ist oft eine Verbindung zu zwei FAs möglich
- Überlappender Sendebereich ermöglicht eine Optimierung
- RFC 4881 schlägt vor, die klare Trennung zur Schicht 2 aufzuheben und so Latenzzeiten zu verringern

Optimierung – Handoffverfahren

- Ereignisse der Schicht 2 (z.B. Empfang einer neuen BSSID) werden an die Schicht 3 (IP) weitergegeben
- Ereignisse der Schicht 2 können von allen Beteiligten (MN, oFA, nFA) erkannt werden
- Nach Registrierung des Schicht 2 Ereignisses kann bevorstehender Handoff vorbereitet werden

z.B. Post-Registration-Handoff

- IP-Tunnel zum Weiterleiten der Nachrichten (von oFA zu nFA) wird vorbereitet

Überblick

1. Einführung

2. Mobile IP

- Mobilität und IP-Adressvergabe
- Dreiecksrouting
- Reverse Tunneling
- Handoff / Roaming
- Optimierungen
- **Mobilitätsunterstützung bei IPv6**

3. Mobile TCP

Mobility Support in IPv6 - Überblick

- Bei der Entwicklung von IPv6 wurde auch die Mobilitätsunterstützung berücksichtigt
- Die als Erweiterung zu IPv4 entworfenen Verfahren und deren Optimierung wurden in den IPv6 Standard integriert
- Die neuen Möglichkeiten von IPv6 wurden dabei berücksichtigt
- Siehe dazu: RFC 3775 (Mobility Support in IPv6)

Mobility Support in IPv6 – Vergleich zu IPv4

- IPv6 Router unterstützen Mobilität – keine speziellen FAs nötig
- Routing-Optimierung ist fester Bestandteil von IPv6
- Nutzung spezielle IPv6-Erweiterungsheader (z.B. Vermeidung von IP-Spoofing)
- Authentifizierung ist Bestandteil von IPv6 und erhöht somit die Sicherheit

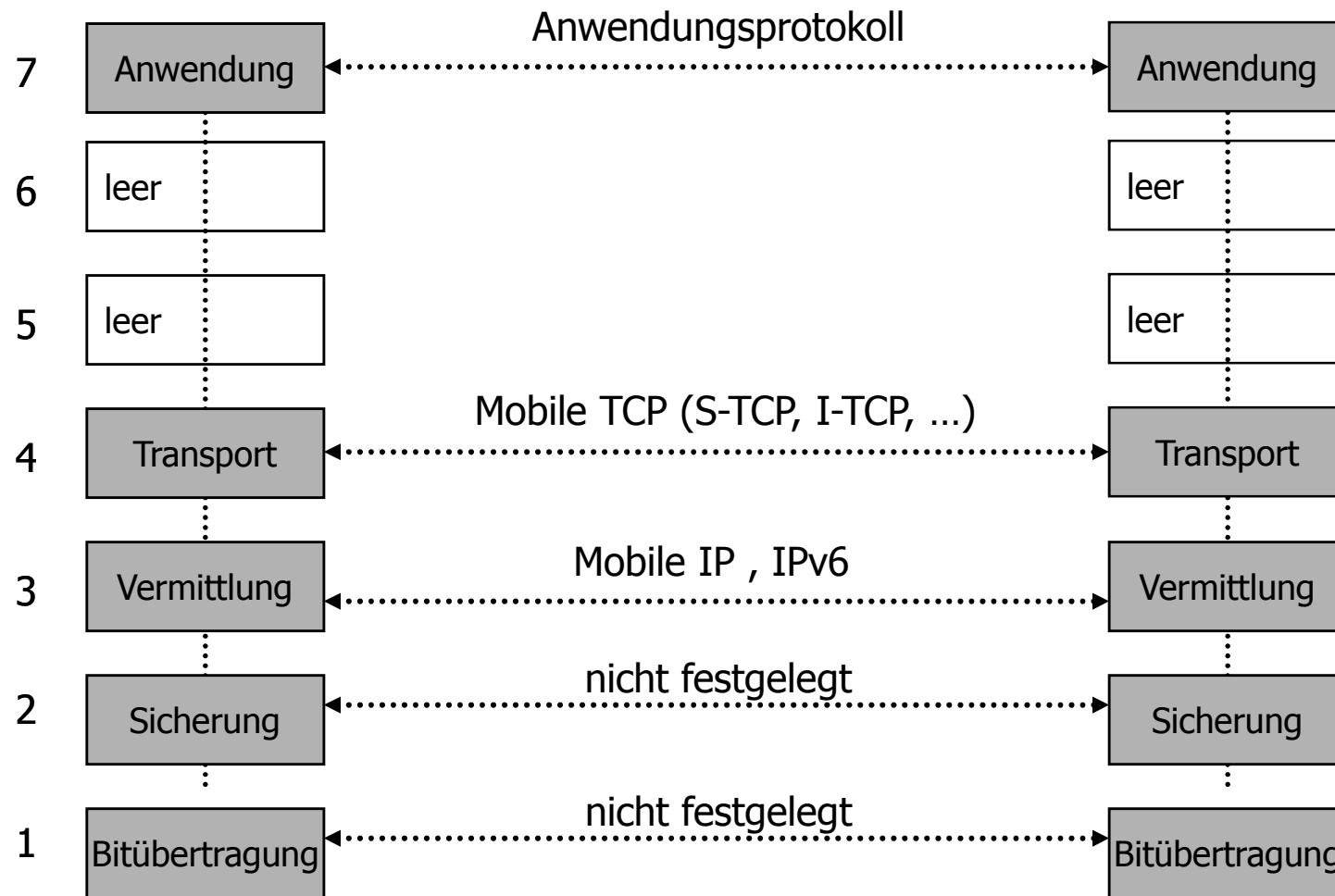
Überblick

1. Einführung
2. Mobile IP
- 3. Mobile TCP**

- **Wiederholung und Überblick**
- PEP – Performance Enhancing Proxy
- Indirektes TCP (I-TCP)
- Snooping TCP (S-TCP)

Überblick

Einordnung in die TCP/IP-Protokollfamilie

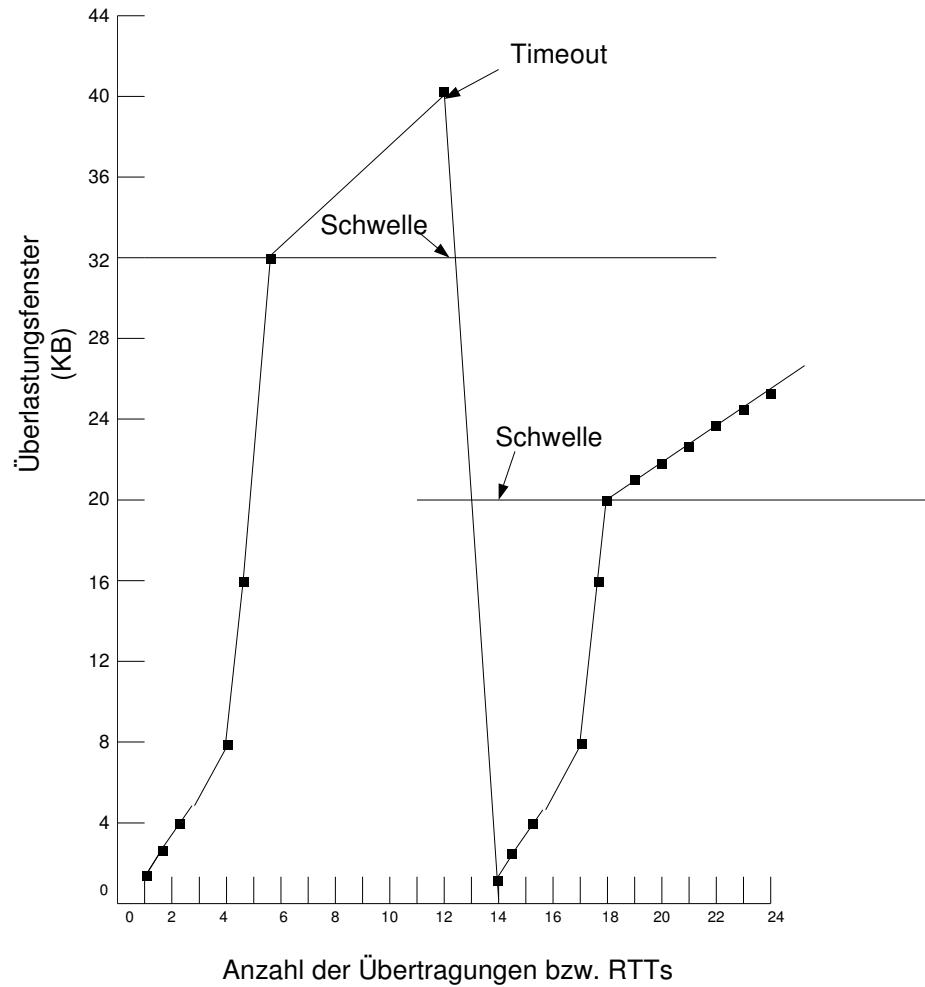


Wiederholung – Transportschicht insb. TCP

- Transportschicht (z.B. TCP) nutzt die Dienste der Vermittlungsschicht (z.B. IP)
- Aufgabe der Transportschicht ist eine **zuverlässige Datenübertragung** sicherzustellen:
 - Quittierung
 - Übertragungswiederholung
 - Fluss- und Staukontrolle
- Fehler aus den **unteren Schichten** müssen von der Transportschicht behandelt werden
- TCP ist wie IP auf eine **stationäre und drahtgebundene** Kommunikation ausgelegt

Wiederholung – TCP Staukontrolle (1)

- Erste TCP-Segmentlänge 1 KB
- 1. Schwellwert bei 32 KB
- Timeout bei Segmentlänge von 40 KB
- Schwellwert wird dann auf 20 KB gesetzt



Wiederholung – TCP Staukontrolle (2)

- TCP bemerkt einen Paketverlust oder eine deutliche Verzögerung durch das Ausbleiben der Quittierungen
- **Drahtgebundene** Datenübertragung **sehr stabil**
- TCP geht bei einem Paketverlust daher zunächst von einer **Stausituation** aus
- TCP reagiert mit einer drastischen **Reduzierung der Übertragungsleistung** um den Stau aufzulösen

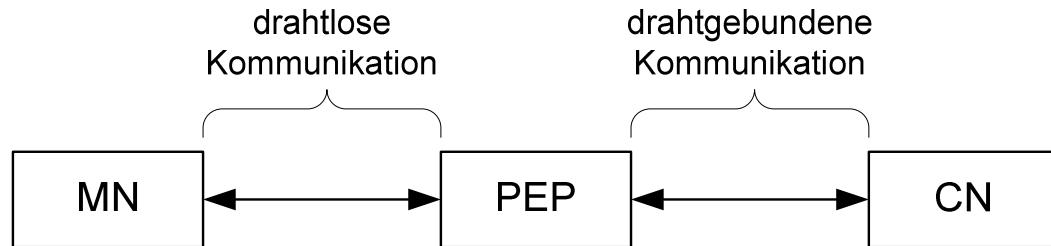
Überblick – Probleme bei Mobilität

- **Drahtlose** Datenübertragung ist **fehleranfälliger**
 - Die Übertragungsleistung kann stark schwanken und bis zum Paketverlust führen
 - Teilnetzwechsel (Handoff) kann zu deutlichen Verzögerungen bei der Übertragung führen
 - Reduzierung der Übertragungsleistung ist bei Paketverlust nicht hilfreich (nur bei Stausituationen angemessen)
 - Bei mobiler Kommunikation ist ein schnelles Nachsenden der fehlenden Pakete wichtig
 - **TCP-Standardverfahren** zur Stau- und Flusskontrolle müssen für die Mobilität **angepasst** werden
-

Überblick

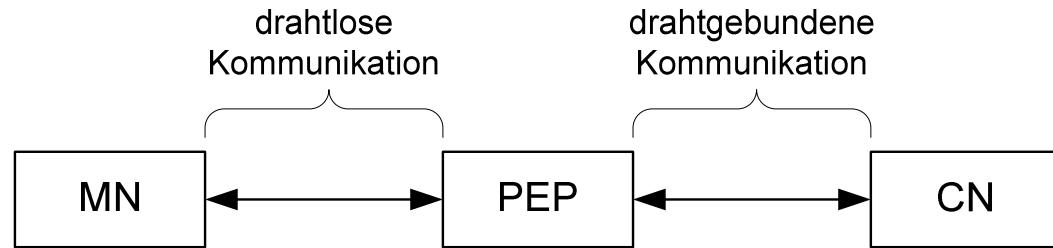
1. Einführung
2. Mobile IP
- 3. Mobile TCP**
 - Wiederholung und Überblick
 - **PEP – Performance Enhancing Proxy**
 - Indirektes TCP (I-TCP)
 - Snooping TCP (S-TCP)

PEP – Performance Enhancing Proxy



- Die im Internet verwendeten Übertragungstechnologien (Schicht 1 und 2) unterscheiden sich zum Teil wesentlich
- Mit einem PEP (siehe RFC 3135) können Netze an der Stelle des Medienwechsels aufgeteilt werden
- Es entstehen medieneinheitliche Teilstrecken
- Auf den Teilstrecken können optimierte Protokolle für die spezielle Technologie eingesetzt werden
- PEP stellt die Kompatibilität sicher

PEP – Optimiertes TCP



- Beispielsweise kann ein Access-Point als PEP arbeiten
- Zwischen MN und Access-Point wird drahtlose Kommunikation eingesetzt
 - Dabei Einsatz einer optimierten Stau- und Flusskontrolle
- Ab dem Access-Point bis zum CN drahtgebundenen Kommunikation
 - Dabei Einsatz der normalen TCP-Stau- und -Flusskontrolle

Überblick

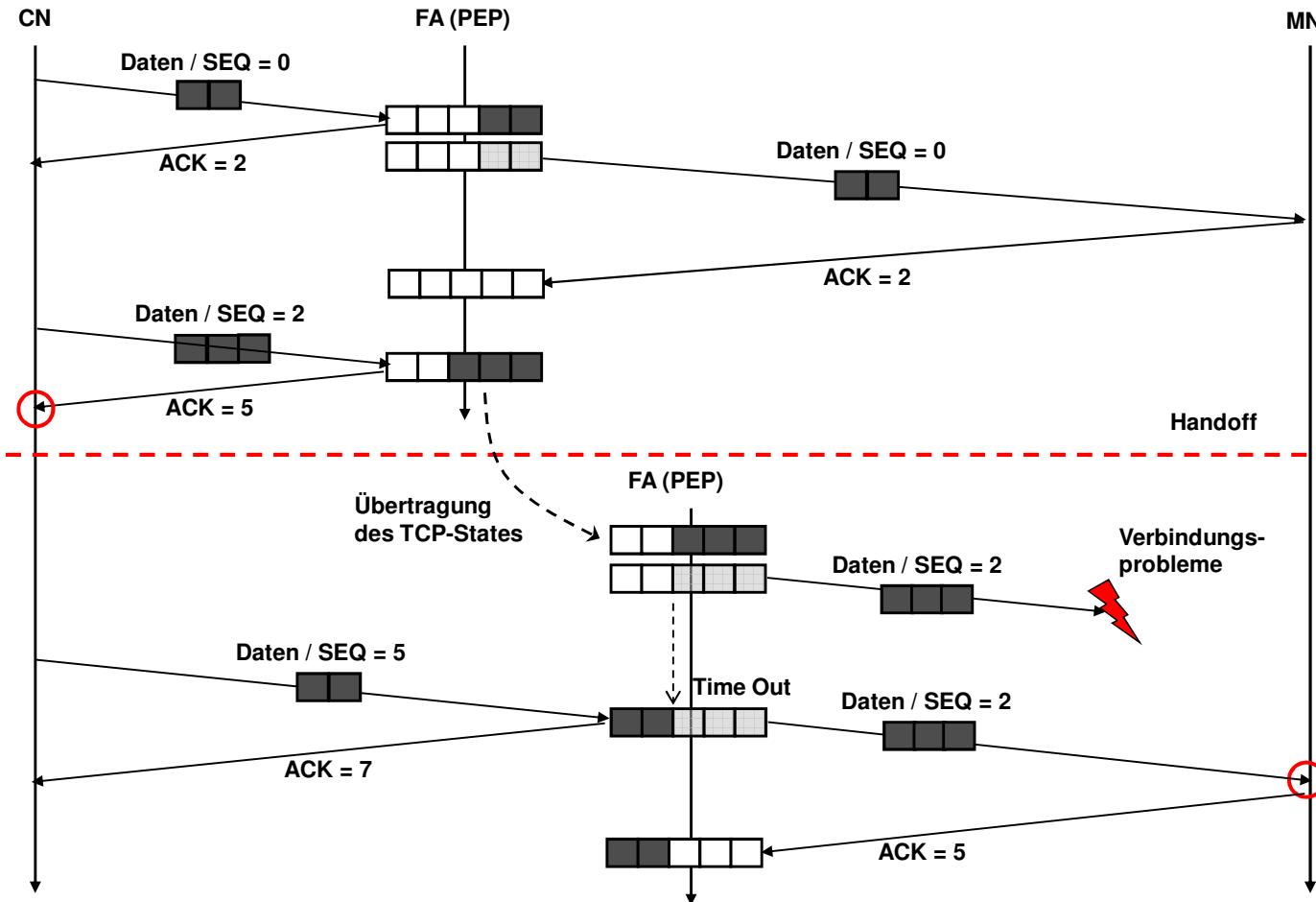
1. Einführung
2. Mobile IP
- 3. Mobile TCP**

- Wiederholung und Überblick
- PEP – Performance Enhancing Proxy
- **Indirektes TCP (I-TCP)**
- Snooping TCP (S-TCP)

I-TCP - Überblick

- **Umsetzung der PEP-Technologie** zur Optimierung von TCP für mobile Kommunikation
- Die TCP-Verbindung zwischen den Kommunikationspartner wird am PEP **aufgetrennt**
- Da somit keine direkte TCP-Verbindung zwischen den Kommunikationspartnern besteht wird, dieser Vorschlag Indirektes-TCP genannt
- PEP verknüpft die beiden TCP-Verbindungen und stellt so die Kompatibilität sicher
→ **keine echte End-zu-End-Semantik** bei I-TCP!

I-TCP – Beispiel (CN sendet an MN)



I-TCP – Funktionsweise

- PEP empfängt Segmente, speichert diese in einem **Zwischenspeicher** und versendet Quittierungen
- Die zwischengespeicherten Segmente werden dann an das eigentliche Ziel **weitergeleitet**
- **PEP erwartet** für die weitergeleiteten Segmente **Quittierungen**
- Treffen die Quittierungen nicht ein, wird das **Senden wiederholt**
- Der eigentliche Kommunikationspartner merkt die Übertragungsfehler und das wiederholte Senden nicht, da er schon eine Quittierung erhalten hat

I-TCP – Optimierung (Beispiel: CN sendet an MN)

- Zwischen dem PEP und dem MN optimierte Sendewiederholung
- Ausbleibende Quittierungen werden schnell erkannt, da meist keine weiteren Knoten zwischen PEP und MN
- Übertragungsleistung wird bei erkannten Fehlern nicht reduziert – Segmente werden schnell nachgesendet
- Der Sender (CN) bemerkt Übertragungsprobleme zwischen PEP und CN nicht
- Zwischen CN und PEP kann das Standard-TCP verwendet werden

I-TCP – Probleme (Beispiel: CN sendet an MN)

- Bei Übertragungsproblemen zwischen PEP und MN wird der CN nicht benachrichtigt
 - Evtl. erhält der Sender (CN) eine Bestätigung bevor der Empfänger (MN) die Daten tatsächlich erhalten hat
 - Somit **keine echte End-zu-End-Verbindung** zwischen CN und MN!
-
- Auch bei TCP führt ein Handoff zu Problemen
 - Die noch zwischengespeicherten Nachrichten müssen bei I-TCP zum neuen PEP übertragen werden

Überblick

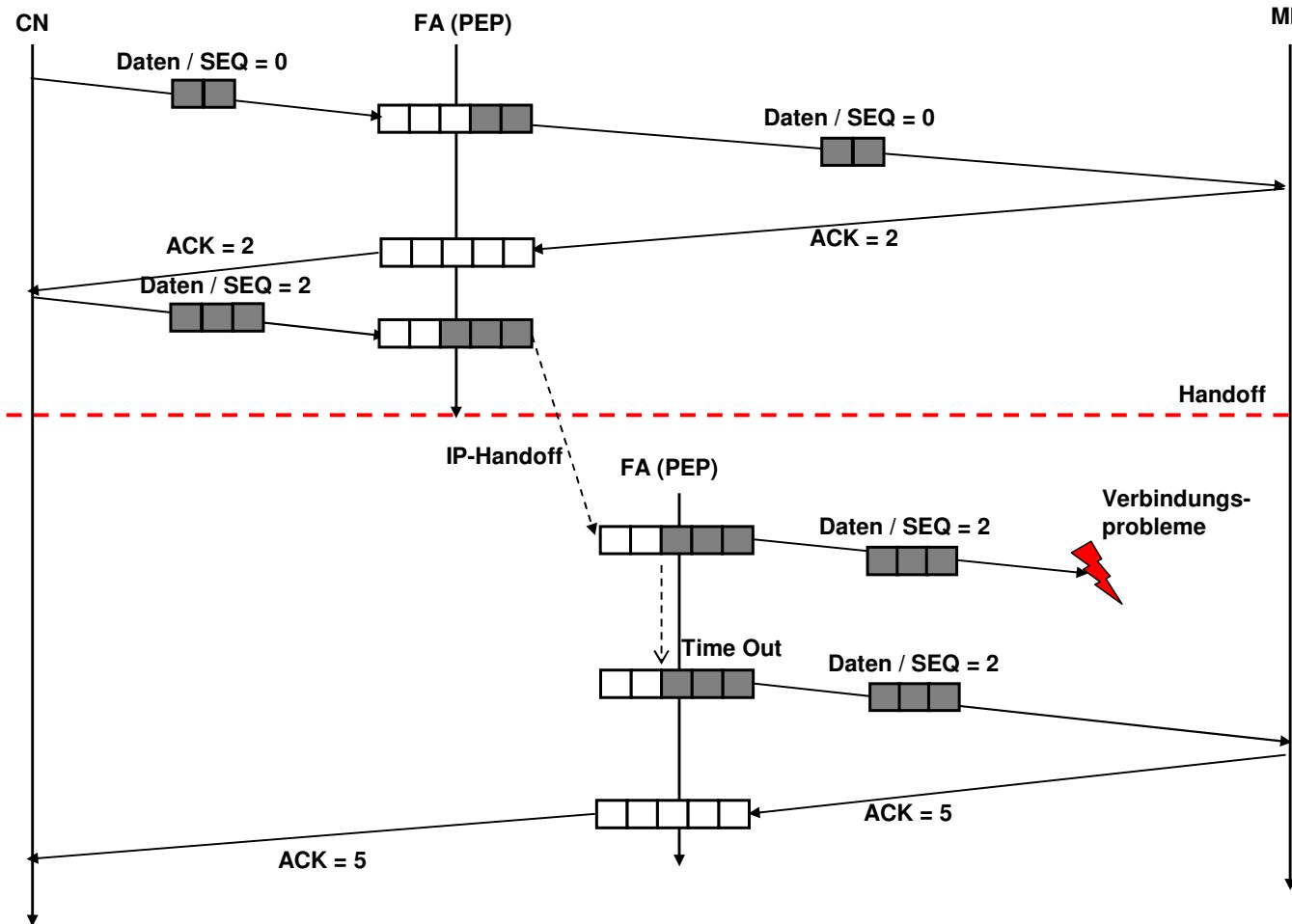
1. Einführung
2. Mobile IP
- 3. Mobile TCP**

- Wiederholung und Überblick
- PEP – Performance Enhancing Proxy
- Indirektes TCP (I-TCP)
- **Snooping TCP (S-TCP)**

S-TCP - Überblick

- Weitere Umsetzung der **PEP-Technologie** zur Optimierung von TCP für mobile Kommunikation
- PEP **ohne Verlust der End-zu-End-Verbindung**
- Pakete zwischen den Kommunikationspartner werden vom PEP „abgehört“
- PEP sendet bei S-TCP selbst keine Quittierung
- Doppelte Quittierung (um fehlende Segmente anzuzeigen) werden jedoch unterdrückt
- Verfahren unterscheidet sich je nach Senderichtung

S-TCP – Beispiel (CN sendet an MN)



S-TCP – Funktionsweise (Beispiel: CN sendet an MN)

- PEP liest Segmente mit und speichert diese vorübergehend
 - Auch Quittierungen werden nur mitgelesen und gespeichert
 - Treten Verbindungsprobleme zwischen PEP und MN auf werden Nachrichten nachgesendet
 - Doppelte Quittierungen werden vom PEP unterdrückt
-
- Sendet der MN an den CN ist das Verfahren ähnlich
 - Bei fehlenden Segmenten sendet der PEP doppelte Quittierungen um die Fehler schnell zu beheben

Rückblick

1. Einführung

2. Mobile IP

- Mobilität und IP-Adressvergabe
- Dreiecksrouting
- Reverse Tunneling
- Handoff / Roaming
- Optimierungen
- Mobilitätsunterstützung bei IPv6

3. Mobile TCP

- Wiederholung und Überblick
- PEP – Performance Enhancing Proxy
- Indirektes TCP (I-TCP)
- Snooping TCP (S-TCP)

Anmerkungen:

- Mobile IP und TCP tun sich in der Praxis noch schwer
→ Komplizierte Verfahren
- Meist nur reine DHCP-Nutzung vorzufinden
- Neuer Standard für Mobile IP nun verabschiedet

Datenkommunikation

Studienarbeit

Wintersemester 2012/2013

Überblick

1. Einführung in die Aufgabenstellung

- Überblick und Lernziele

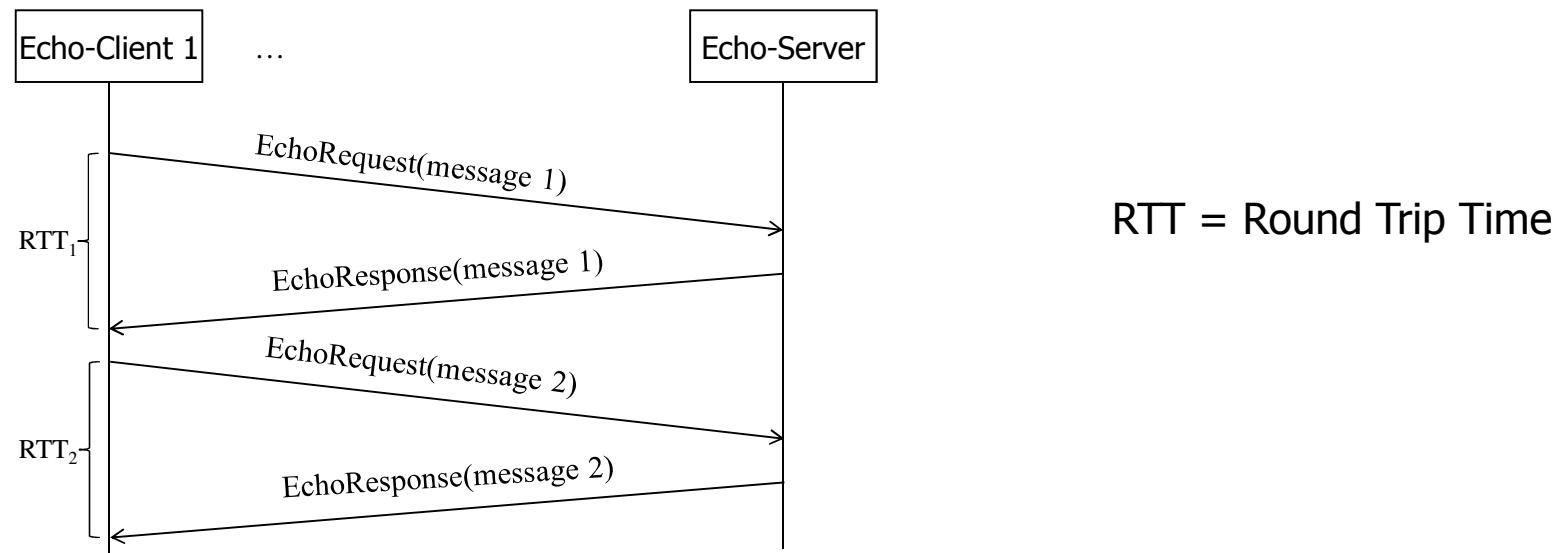
2. Teilaufgaben 1 - 7

Überblick über die Aufgabenstellung

- **Konzeption und Implementierung** einer Kommunikationssoftware
- Szenario in diesem Semester ist eine einfache **Echo-Anwendung** (Client-/Server)
- Nutzung *mehrerer Transportprotokolle* (TCP, UDP, LWTRT, RMI), Transportzugriff über **Sockets**
- Programmieren in einer höheren Programmiersprache → wir verwenden **Java**
- **Leistungsvergleich** für verschiedene Lösungen

Echo-Anwendung

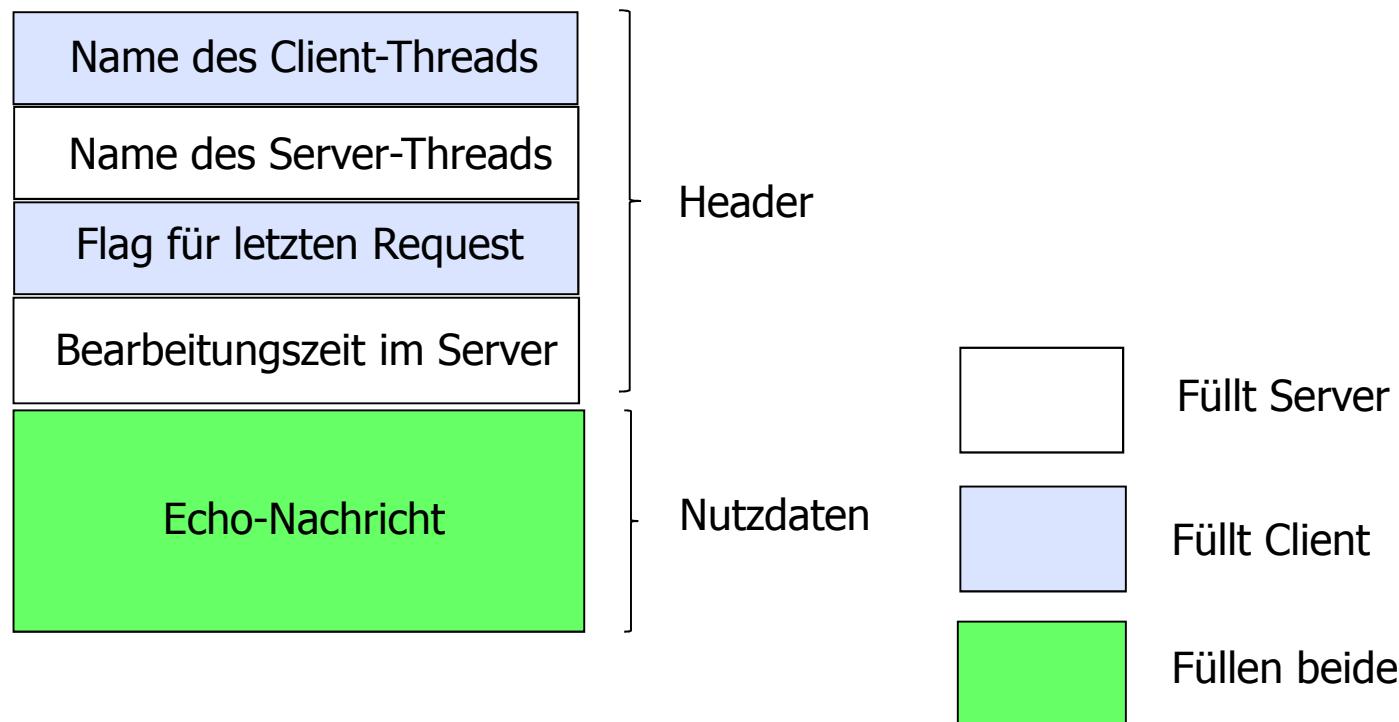
- Client sendet Anfrage (Echo-Request)
- Server registriert Client, falls noch nicht erfolgt, in einer „Client-Liste“
- Server antwortet mit Response (Echo-Response)
- Nachrichtenformat ist definiert → Echo-PDU



Echo-Protokoll (1)

Nachrichtenaufbau

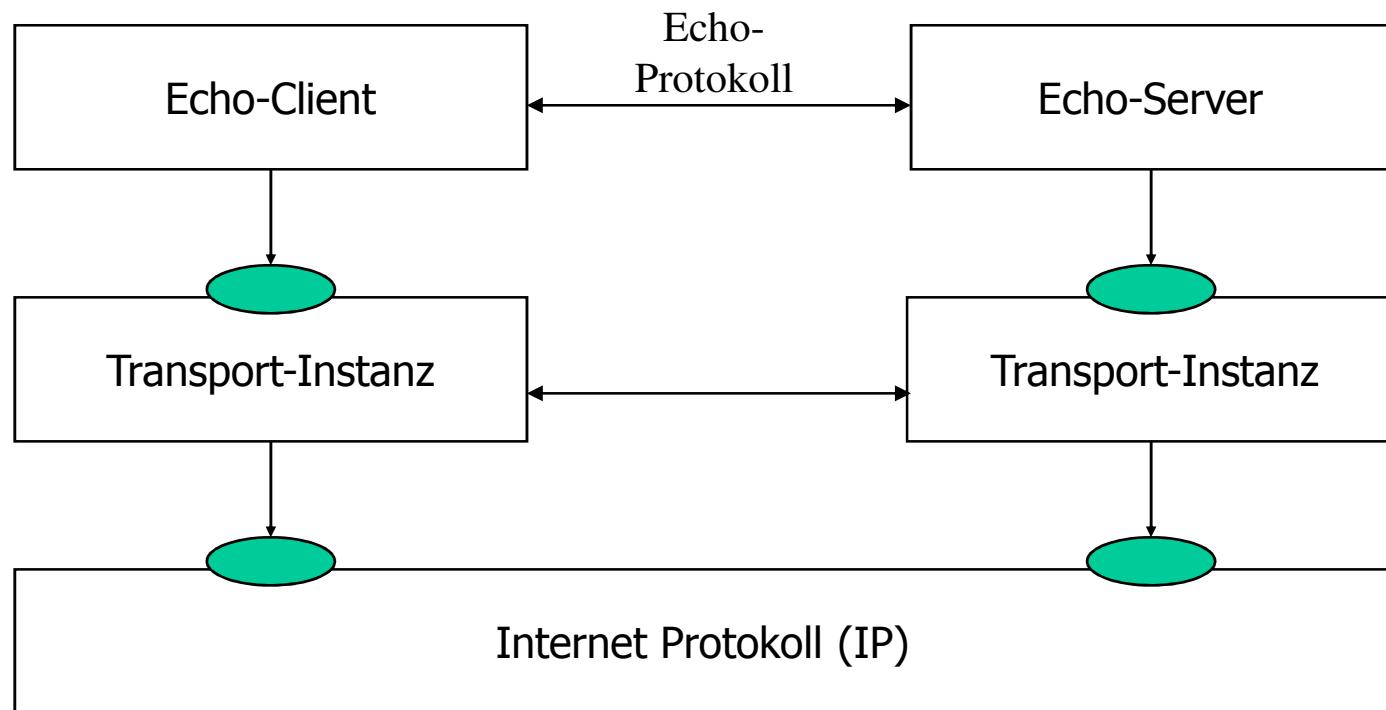
- Einfaches Nachrichtenformat in Klasse EchoPDU definiert:



Echo-Protokoll (2): Regelwerk

- Name des Client-Threads wird durch den Client belegt
 - Auf Eindeutigkeit achten, jeder Client-Thread erhält eigenen Namen → Thread.setName()
- Name des Server-Threads wird durch den Server belegt
 - Jeder Server-Thread erhält einen eigenen Namen
- Flag für letzten Request: Angabe des Client-Threads, ob es seine letzte Echo-PDU ist (Letzter Request = true)
 - Wird vom Server verwendet, damit der Client wieder aus der Client-Liste gelöscht werden kann
- Bearbeitungszeit im Server:
 - Zeit, die der Server für die Bearbeitung des Request benötigt (Zeitmessung in Nanosekunden)
- Echo-Nachricht: Eigentliche Nutzdaten

Grobe Schichtenarchitektur



Lernziele

- **Erfahrungen** in der Programmierung von Kommunikationsanwendungen (hier: Client-/Server-Anwendungen) **sammeln**
- **Schichtenorientiertes Denken** schulen
- Problemstellungen der **Protokollimplementierung** praktisch erfahren
 - Komplexität anderer Protokolle soll verständlicher werden
 - Methodisches Vorgehen beim Vergleich von Lösungsansätzen insbesondere in der Leistungsanalyse

Überblick

1. Einführung in die Aufgabenstellung

- Überblick und Lernziele

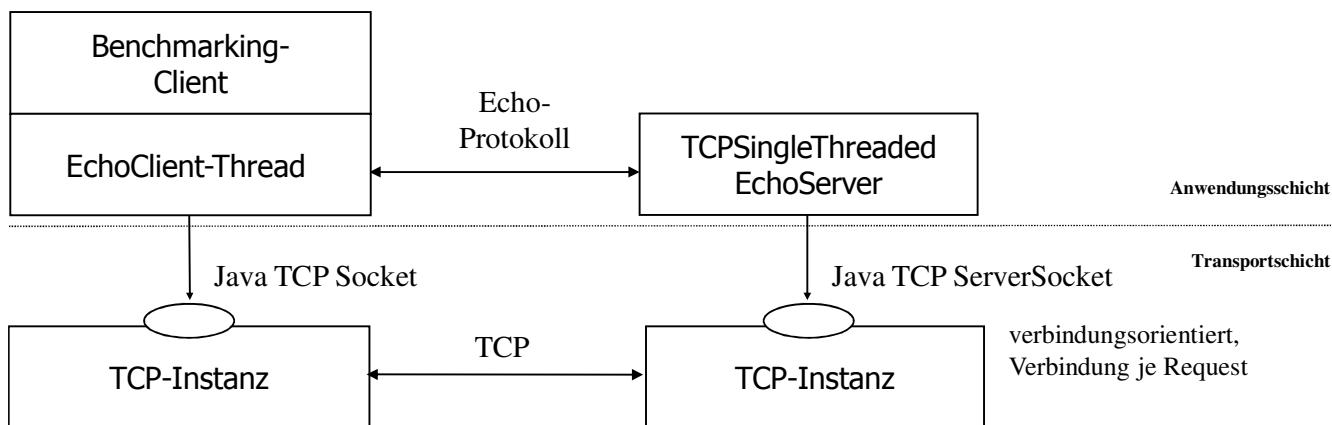
2. Teilaufgaben 1 - 7

Single-threaded versus Multi-threaded Server

- Serverseitig gibt es große Unterschiede in der Implementierung der **nebenläufigen** Verarbeitung
- **Single-threaded:** Im Wesentlichen übernimmt ein Thread die Bearbeitung aller ankommenden Requests von Clients
- **Multi-threaded:** Jeder Client oder jeder Request wird in einem eigenen Thread bearbeitet
- Weitere Unterscheidung: Verbindungsorientiert oder verbindungslos

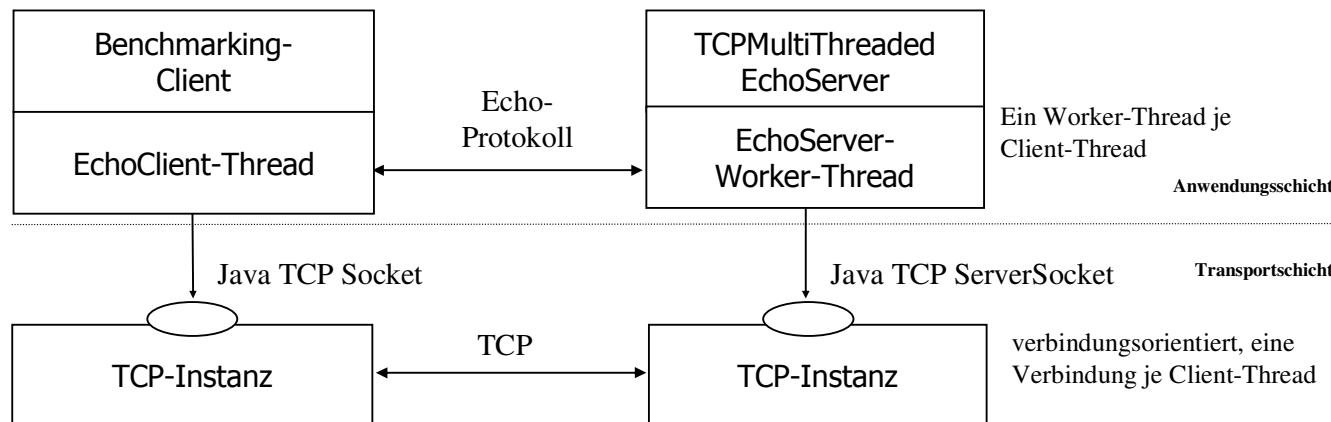
Teilaufgabe 1: Single-threaded TCP-Server

- Aufbau einer TCP-Verbindung zwischen Client und Server **für jeden Echo-Request** mit anschließendem (sofortigem) Abbau der Transportverbindung
- **Benchmarking-Client** zur Parametrisierung wird hier gleich mit entwickelt
 - **Datensammlung** für Leistungsvergleich einbauen (Basisklassen vorhanden)



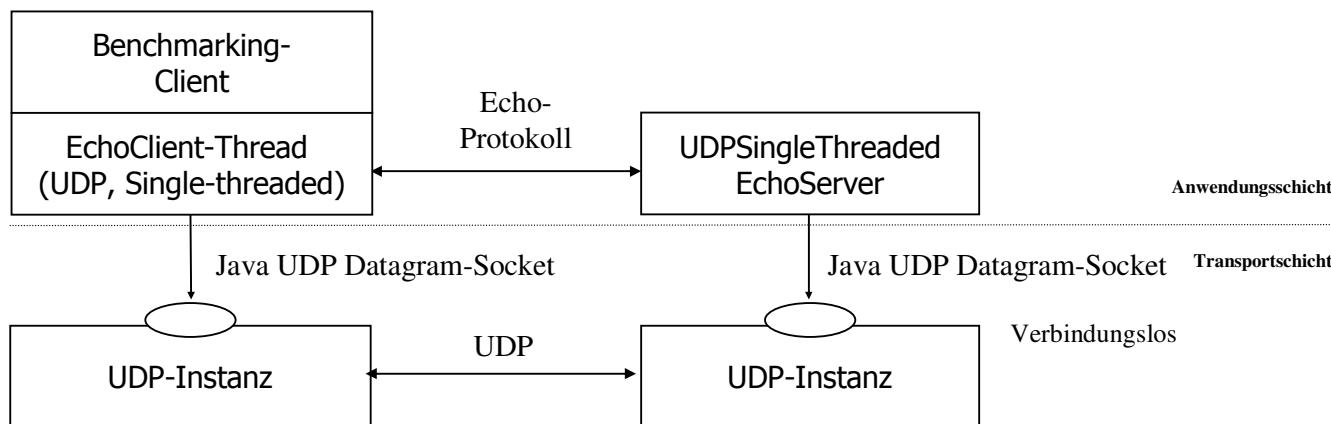
Teilaufgabe 2: Multi-threaded TCP-Server

- Serverseitig eigener „**Worker-Thread**“ für jeden Client-Thread
- Transportverbindung zwischen Client und Server wird am Anfang aufgebaut
- Transportverbindung bleibt bis zum Beenden eines Client-Threads bestehen



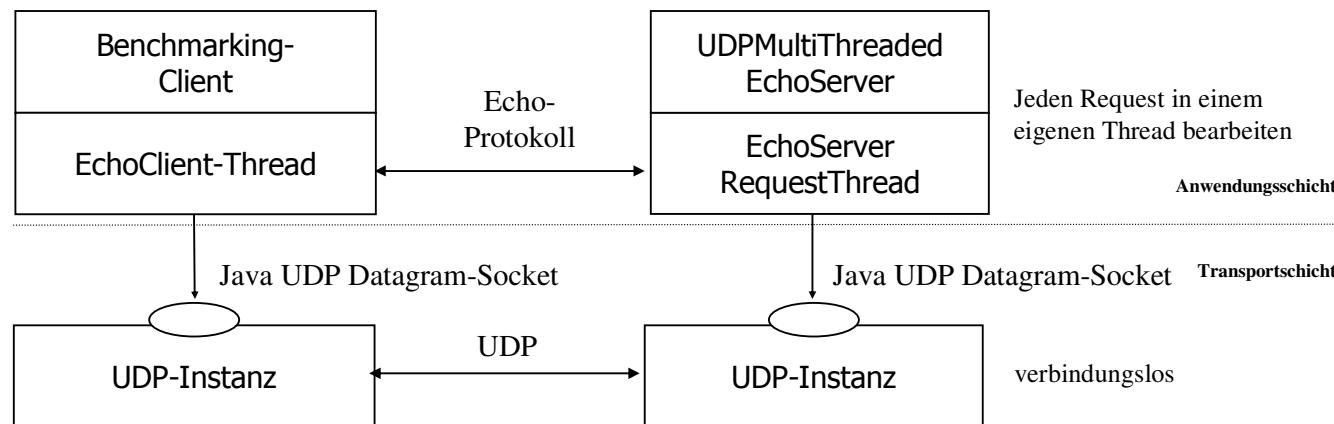
Teilaufgabe 3: Single-threaded UDP-Server

- **Ein Thread im Server**, der alle Echo-Requests bearbeitet
- **Verbindungslose** Kommunikation auf Basis von UDP-Datagram-Sockets



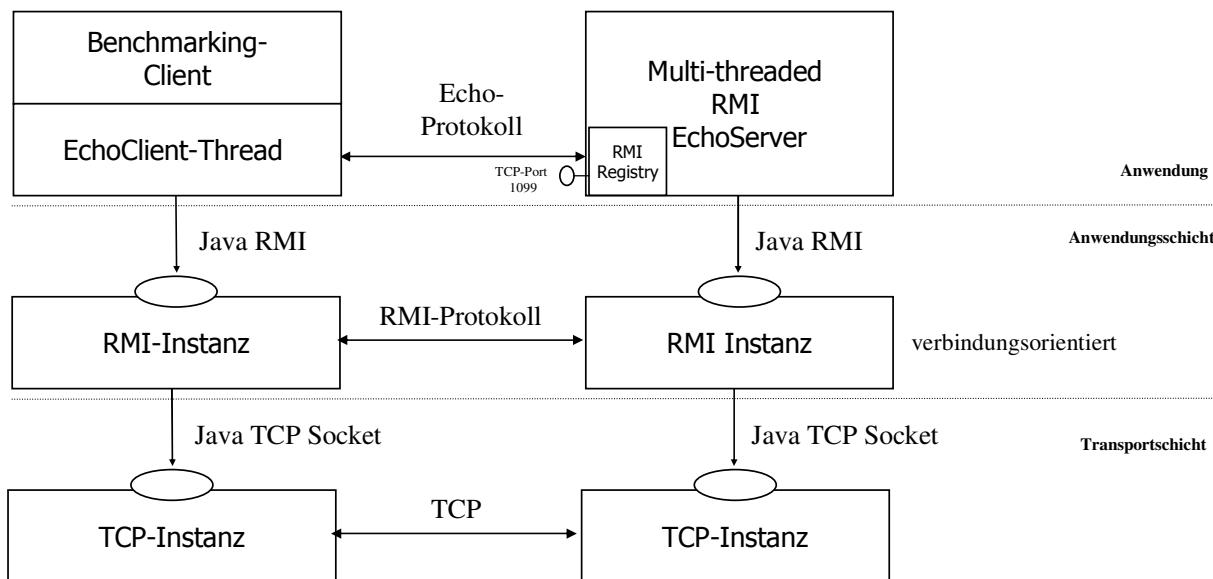
Teilaufgabe 4: Multi-threaded UDP-Server

- **Jeder Echo-Request** wird im Server in einem eigenen „**Worker-Thread**“ bearbeitet
- **Verbindungslose** Kommunikation auf Basis von UDP-Datagram-Sockets



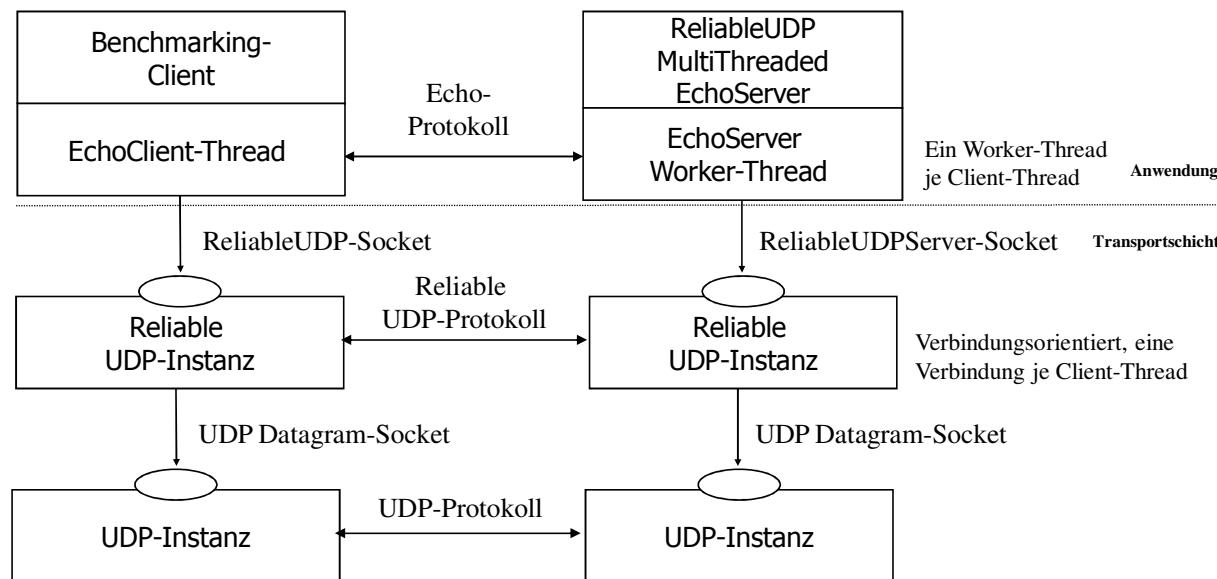
Teilaufgabe 5: Multi-threaded RMI-Server

- Java RMI = Remote Method Invocation ist ein **höherwertiger** Client-/Server-Mechanismus und setzt auf TCP auf



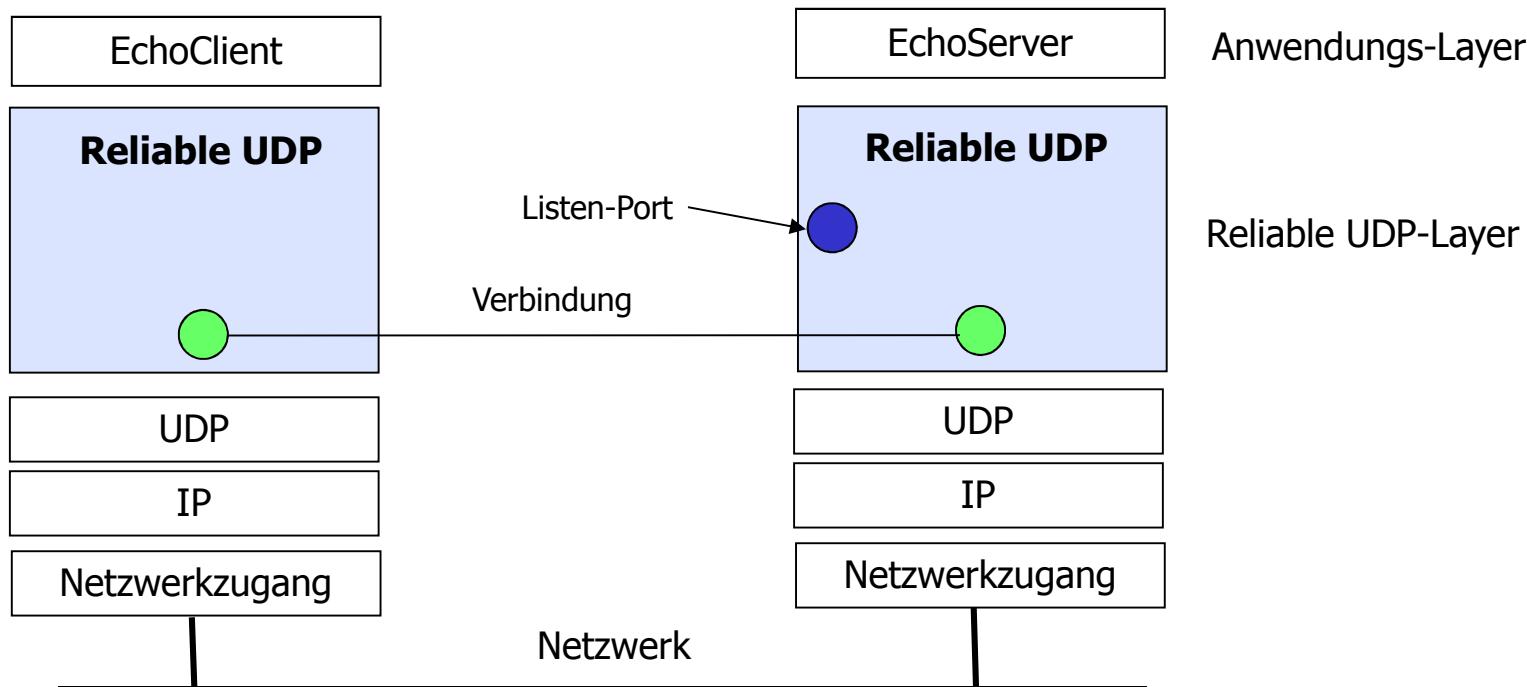
Teilaufgabe 6: Reliable multi-threaded UDP-Server

- **Eigenentwicklung** eines gesicherten Transportprotokolls auf Basis von UDP
 - Einfaches Stop-and-Wait-Protokoll mit positiv-selektiver Quittierung
 - Timerüberwachung für alle Nachrichten



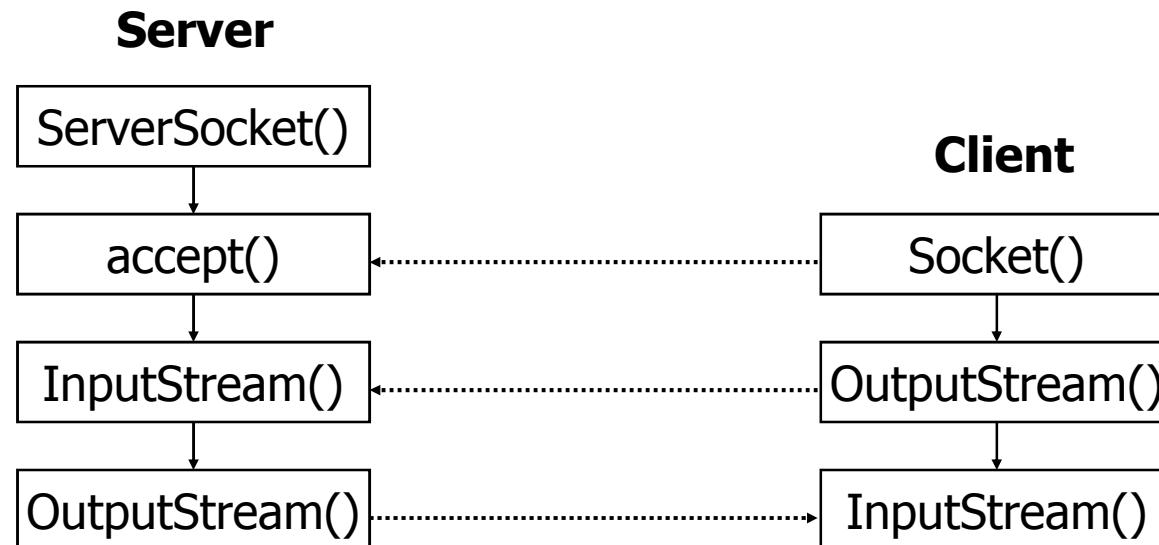
Teilaufgabe 6: ReliableUdpSocket Stack

- Verwendeter Stack: Über UDP nochmals eine Schicht, die für eine zuverlässige Übertragung sorgt

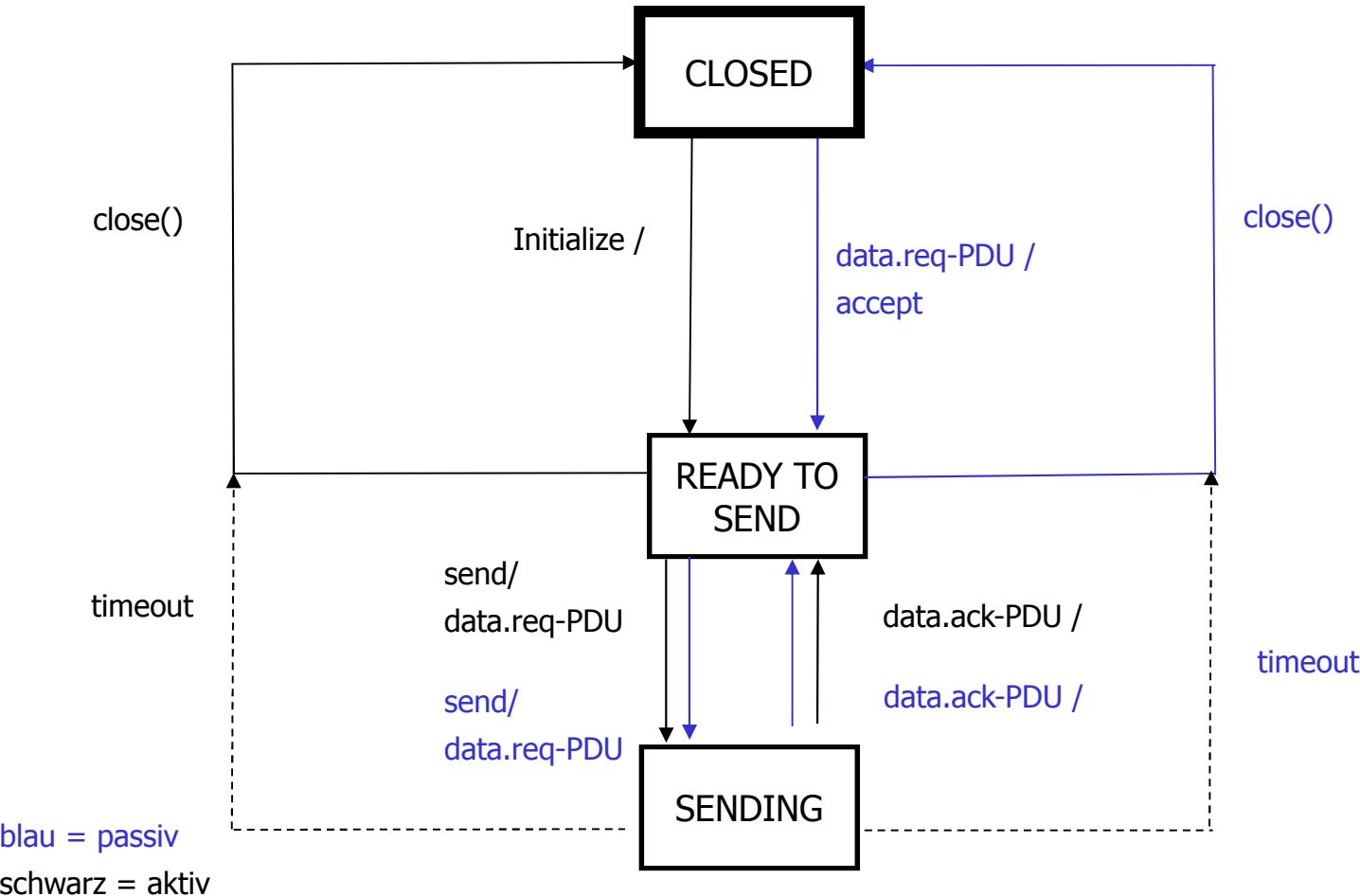


Teilaufgabe 6: ReliableUdpSocket Streams zur Kommunikation

- Im Gegensatz zur UDP-Implementierung (Teilaufgaben 3 und 4) werden in der zuverlässigeren Variante Streams verwendet.

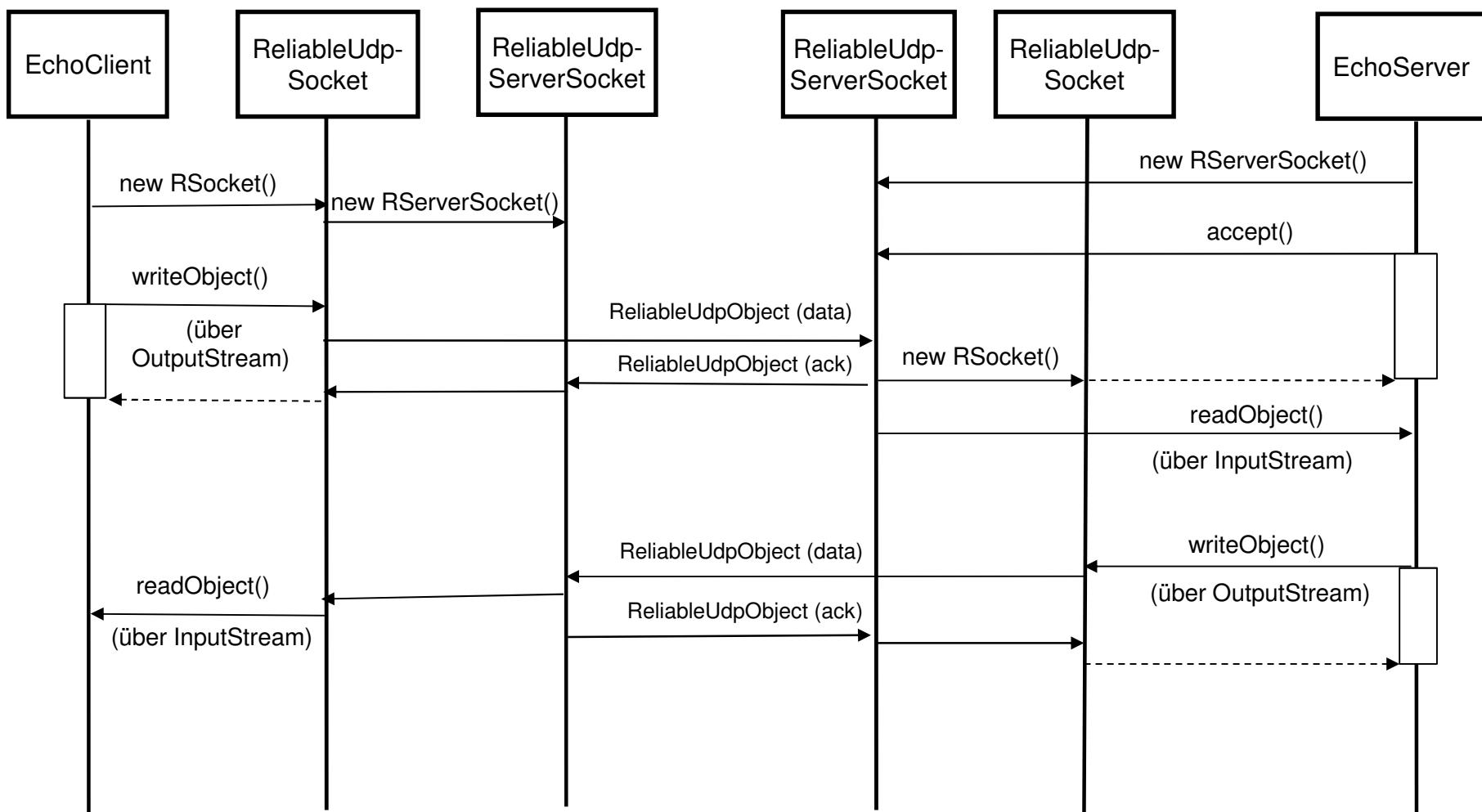


Teilaufgabe 6: ReliableUdpSocket Zustandsautomat



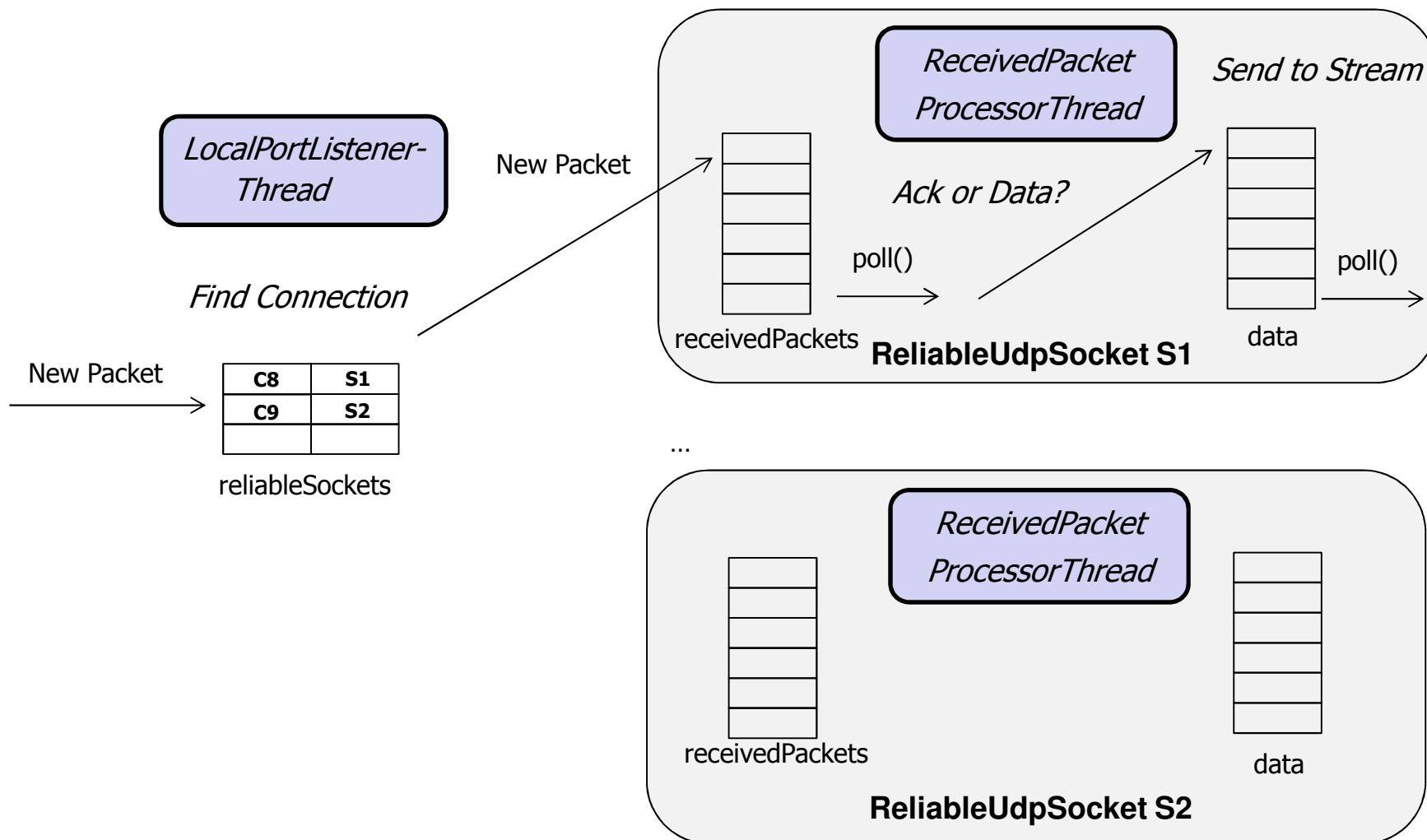
Teilaufgabe 6: ReliableUdpSocket

Verbindungsauftbau- und Datenaustauschphase



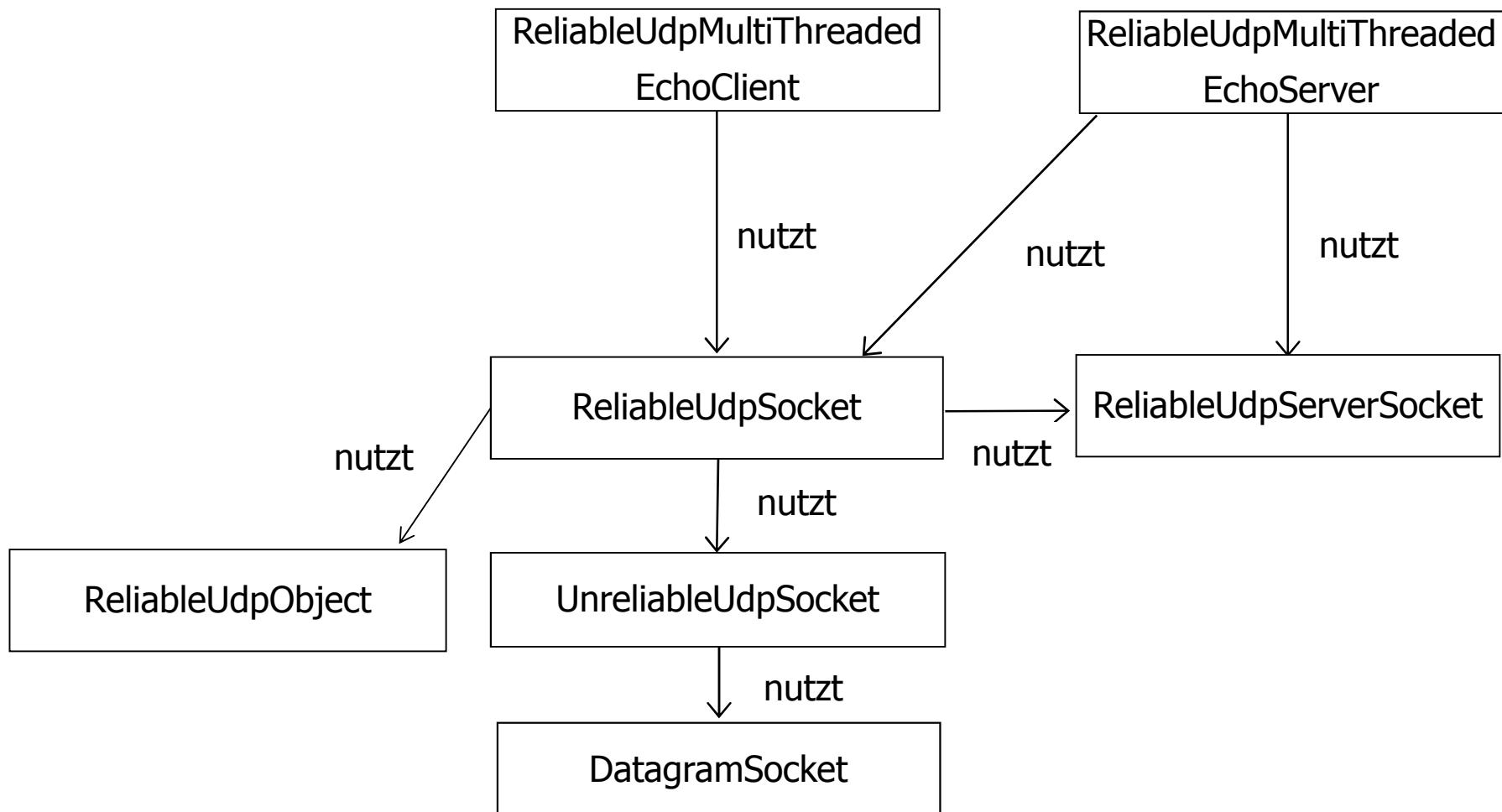
Teilaufgabe 6: ReliableUdpSocket

Verwenden von BlockingQueues zum Datenaustausch

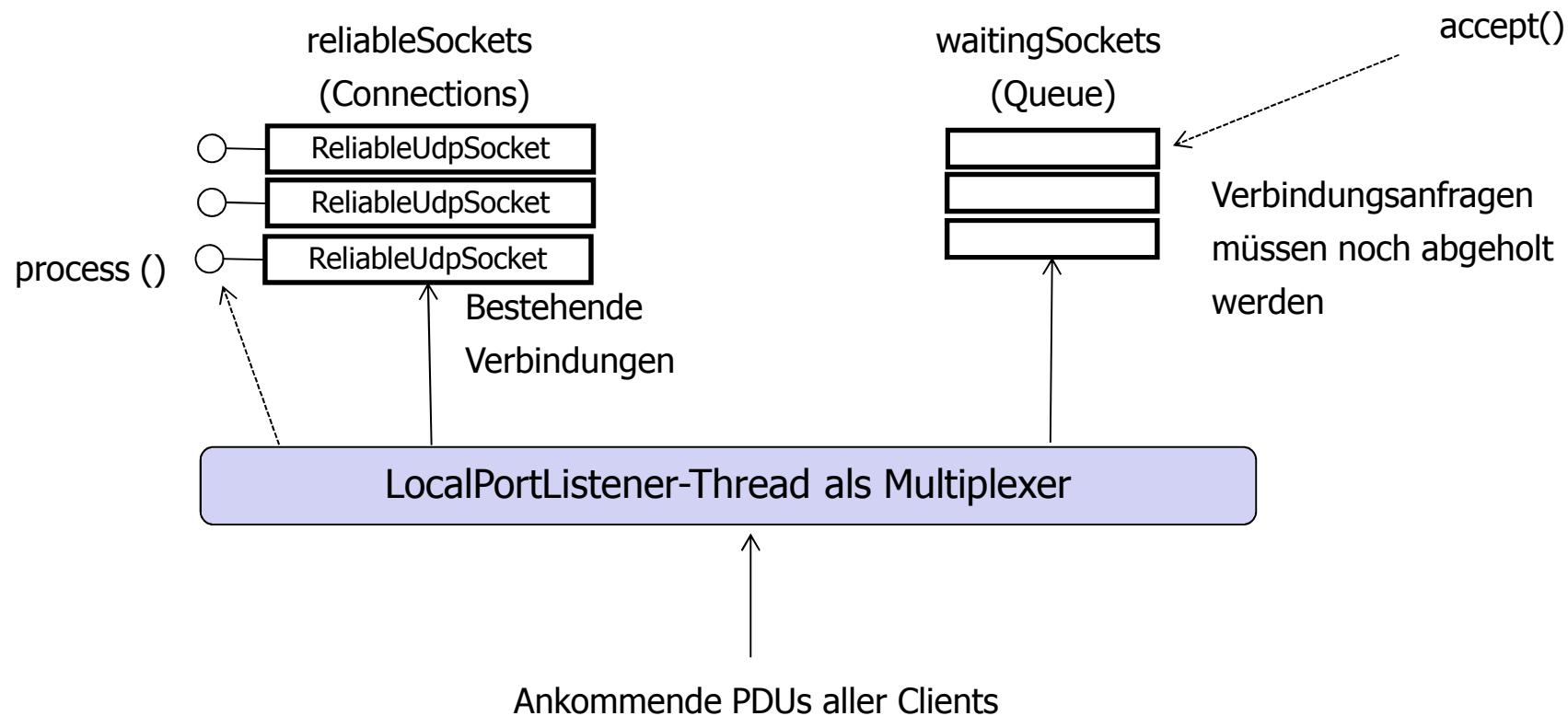


Teilaufgabe 6: ReliableUdpSocket

Wichtige Objektklassen im Überblick

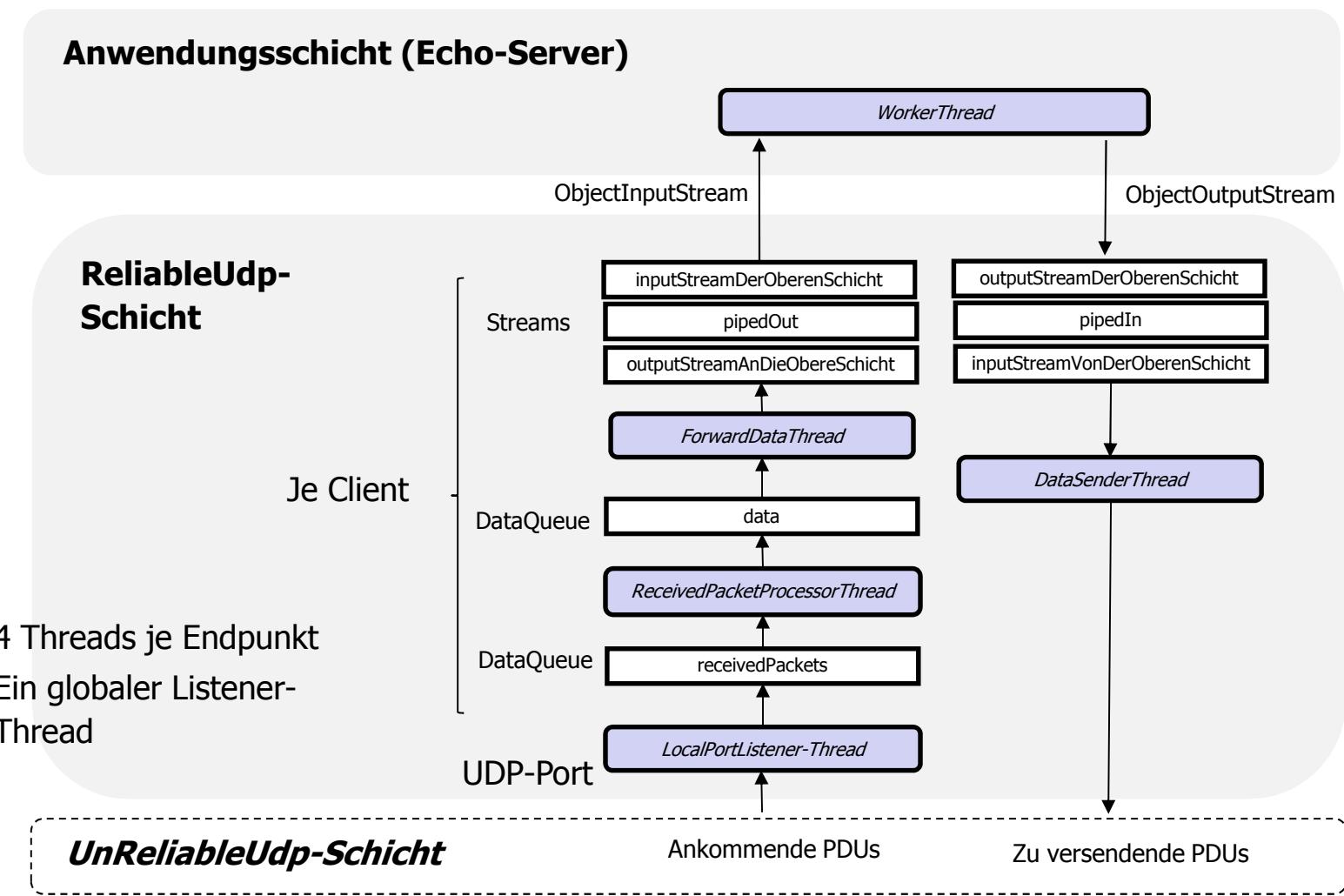


Teilaufgabe 6: ReliableUdpSocket Multiplexing im Server



Teilaufgabe 6: ReliableUdpSocket

Nachrichtenverarbeitung im Server



Generell

Logging und Debugging

- Logging / Debugging in getrennte Logdateien (Client und Server)
- Logsatz schreiben (Typen: INFO, DEBUG, ERROR) bei
 - Ankommenden PDUs nach dem Empfang (Ereignis als DEBUG, Inhalt als INFO)
 - Abgehende PDUs, vor dem Senden (Ereignis als DEBUG, Inhalt als INFO)
 - Methodenaufrufen (DEBUG)
 - Fehlersituationen (ERROR)

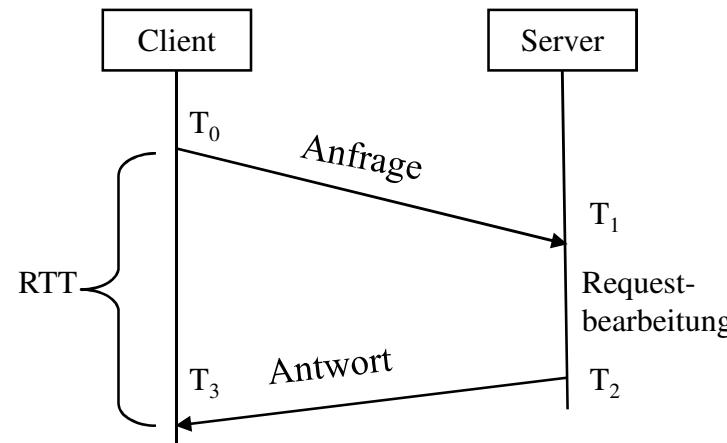
Teilaufgabe 7: Leistungsbewertung und Vergleich

- Leistungsbewertung für alle Multi-Threaded-Lösungen
- Verwendete Metrik: **RTT**
 - Serverbearbeitungszeit soll auch ermittelt werden
- Testdurchführung mit allen multi-threaded Implementierungen **mehrmais** (mind. 5 mal) **wiederholen**
- RTT-Berechnung: Mittelwert, Minimum, Maximum (über vorgegebene Klasse *SharedClientStatistics*)
- Leistungsauswertung

Teilaufgabe 7: Metriken: RTT

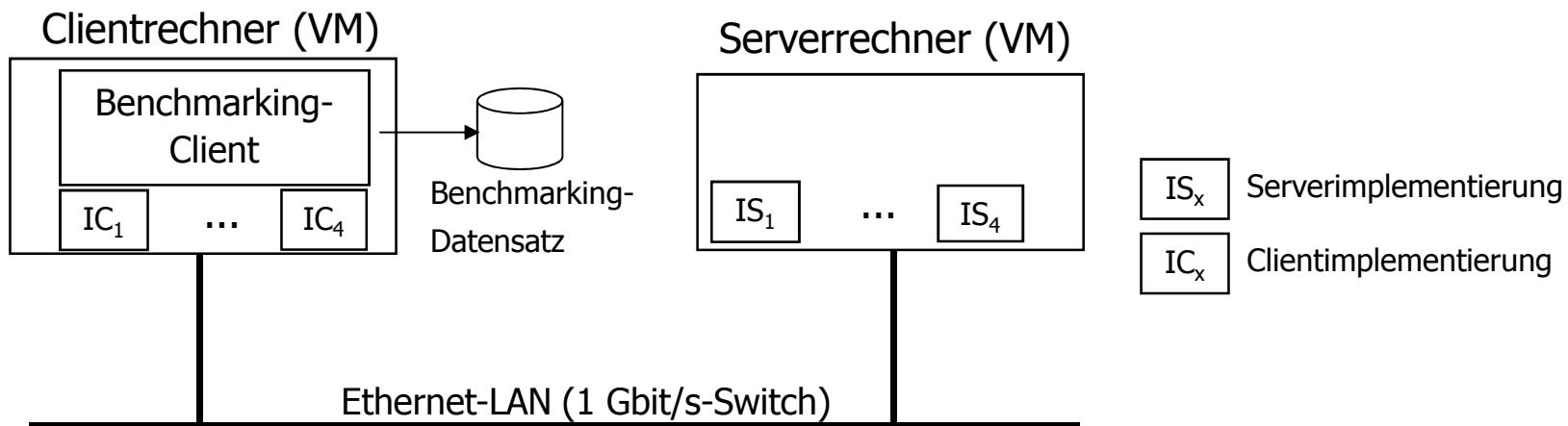
■ Definition:

- Unter der Round-Trip-Time versteht man die Reaktions- oder Bearbeitungszeit eines Anwendungssystems für eine Anfrage
- RTT bezeichnet die Zeitspanne, die erforderlich ist, um eine Nachricht bzw. einen Request von einem Sender zu einem Empfänger zu senden und die Antwort (den Response) des Empfängers wieder im Sender zu empfangen.
- Messung in Millisekunden
- $RTT = T_3 - T_0$



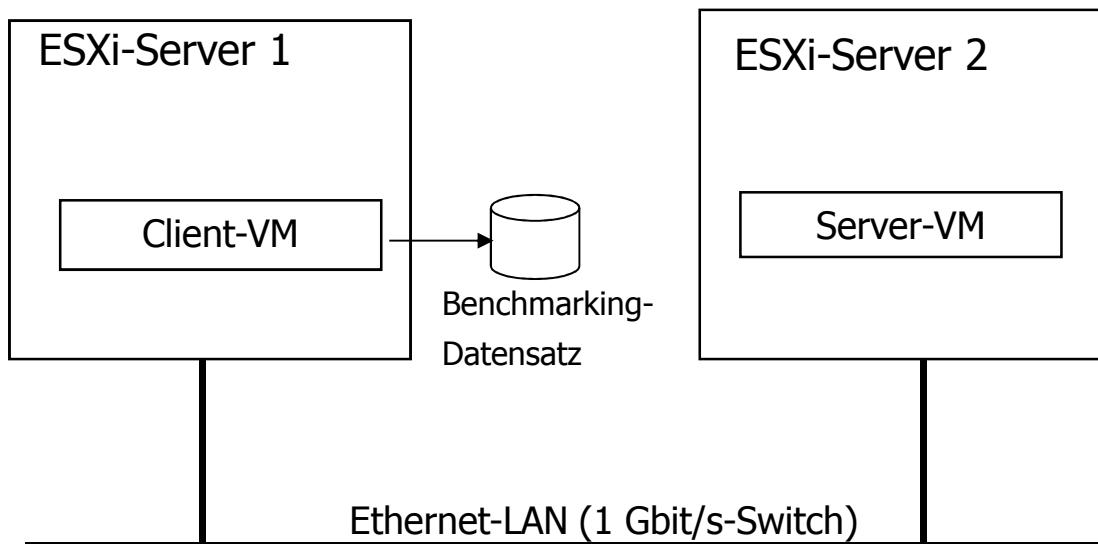
Teilaufgabe 7: Aufbau der Messumgebung (1)

- 1 virtueller Clientrechner zur Simulation der Clients (Client-Threads) mit Benchmarking-Client
- 1 virtueller Serverrechner
- Testumgebung genau beschreiben: Rechnerausstattung, Netzwerk,... → Nachvollziehbarkeit des Benchmarks!!



Teilaufgabe 7: Aufbau der Messumgebung (2)

- VMware-Umgebung wird für Messungen verwendet
- Jede Gruppe bekommt 2 virtuelle Maschinen mit Windows XP als Betriebssystem
- VMs liegen auf unterschiedlichen ESXi-Servern



Teilaufgabe 7: Sammlung der Messdaten

- RTT-Daten in Client-Threads bei jedem Request sammeln
- Abspeichern der Messergebnisse in einer Datei
 - vordefinierte Java-Klasse Nutzen
- Mehrfaches Wiederholen jedes Benchmarks (Achtung: zeitintensiv!)
- Daten anschließend auswerten

Teilaufgabe 7: Messungen: Auswertung

- Grafische Darstellungen erstellen (über Excel)
 - Veränderung der Anzahl Client-Threads bei konstanter Nachrichtenlänge, konstanter Denkzeit und 100 Echo-Nachrichten je Echo-Client-Thread
 - Alle Multi-threaded Varianten in eine Grafik

