

BW332 - Rechnernetze (mit Praktikum)

Veranstaltungsüberblick



Überblick & Ablauf der Veranstaltung „Rechnernetze“



Prof. Dr. Peer Küppers

E-Mail: peer.kueppers@hs-lu.de

Tel.: +49 (0)621 5203 - 416, Raum E1020

Veranstaltungszeiten:

Donnerstag, 14:15-15:45 Uhr, Raum E1106

Vorlesung & Praktikum (Übungen)

Bei Fragen:

- Nutzen Sie bitte das Forum in OLAT
- Schreiben Sie mir eine E-Mail
- Vereinbaren Sie einen Termin per E-Mail

- [!\[\]\(b93c3e1add16fe46100bba7a6da1e82f_img.jpg\) BW332 Rechnernetze](#)
- [!\[\]\(e03fe9a24fe5105c8f59e912ae7b8724_img.jpg\) Informationsseite](#)
- [!\[\]\(b4572a044582c68c9e6e6b6b9b95c325_img.jpg\) Online-VL](#)
- [!\[\]\(a4848a7f290c3fd14bf2276a5b09747a_img.jpg\) Mitteilungen](#)
- [!\[\]\(c1b926fcee63cdc7f15c603966ea3d76_img.jpg\) Material_191](#)
- [!\[\]\(8bca340a5031aa30dac36202fe939cc1_img.jpg\) Forum](#)

Willkommen im Kurs "BW332 Rechnernetze" im Sommersemester 2019

Prüfungsform

Klausur am PC

Links

- **Online-Whiteboard:**  https://realtimeboard.com/app/board/o9J_kyfQp8g=/

PDF-Kennwort

Das Kennwort für sämtliche PDF-Dateien lautet **hslu-191**

Semesterplanung (Änderungen vorbehalten)

Semesterplanung BW332 (Sommersemester 2019) : Ablauf					
BW332 Rechnernetze (mit Praktikum)					
KW	Datum	Tag	Typ	Nr.	Thema
12	21.03.2019	Do	VL	1	Veranstaltungsüberblick, Einführung in Computernetzwerke
13	28.03.2019	Do	VL	2	Einführung in Computernetzwerke (Teil 2)
14	04.04.2019	Do	Online	3	Einführung in Computernetzwerke (Teil 3) - Keine Präsenz - Online-Vorlesung
15	11.04.2019	Do	VL	4	Anwendungsschicht (Teil 1)
16	18.04.2019	Do	VL	5	Anwendunasschicht (Teil 2)

Kennwort: networks191

Link:

<https://olat.vcrp.de/url/RepositoryEntry/1865220845>

Ziele und Themen aus dem Modulhandbuch

Qualifikationsziele

„Die Studierenden sollen [...] Kenntnisse des Aufbaus von Rechnernetzen sowie Kenntnisse grundlegender Technologien für lokale Netze und Weitverkehrsnetze erworben haben. Zudem sollen die Studierenden Erfahrungen in der Administration von Netzwerkbetriebssystemen und beim Arbeiten mit [...] Rechnernetzen gesammelt haben.“

Themen

- Grundlagen der Kommunikation, informationstheoretische Grundlagen
- Referenzmodelle
- Router, Switch, Hub, Proxy und Gateway
- Physische Übertragung, Schnittstellen, Paketbildung
- Fehler- und Flusskontrolle
- Lokale Netze
- Weitverkehrsnetze
- Internet-Schicht
- Transportschicht
- Fehlererkennung und -korrektur
- Protokolle und Dienste
- Grundlagen zur Rechnernetz-Sicherheit

Workload BW332

90 h Gesamtworkload
24 h Kontaktzeit
66 h Selbststudium

Ziele der gesamten Veranstaltung



- **Wichtigkeit von Netzwerken verstehen**, insbesondere in Zeiten von Cloud Computing, Internet of Things, Industrie 4.0, Big Data, Mobile Applikationen etc.
 - Vernetzung ist die Grundlage für sämtliche moderne Anwendungen, bspw. Cloud-Dienste, Software-as-a-Service, Apps, Web-Services, ...
- Das notwendige **Technologieverständnis** aufbauen
 - Insbesondere Architekturen lokaler Netzwerke und des Internets
 - Dabei auch Fähigkeiten erwerben, um moderne Anwendungs- und Geschäftsanforderungen in Netzwerkanforderungen zu „übersetzen“
- Festigung sämtlicher Inhalte durch deren **praktische Anwendung**
 - Netzwerkanalyse-Tools
 - Aufbau von (simulierten) Netzwerkarchitekturen

Vorgehen



- Die Veranstaltung basiert auf einem „top-down“ Ansatz
(nach Kurose & Ross 2017)
 - Wir beginnen mit der obersten Netzwerkschicht (Anwendungsschicht) und arbeiten uns durch bis zur untersten, physischen Übertragungsschicht.
- Themen
- Grundlagen der Kommunikation, informationstheoretische Grundlagen
 - Referenzmodelle
 - Router, Switch, Hub, Proxy und Gateway
 - Physische Übertragung, Schnittstellen, Paketbildung
 - Fehler- und Flusskontrolle
 - Lokale Netze
 - Weitverkehrsnetze
 - Internet-Schicht
 - Transportschicht
 - Fehlererkennung und -korrektur
 - Protokolle und Dienste
 - Grundlagen zur Rechnernetz-Sicherheit
- ↑
- Dieser Aufbau legt den Fokus auf die in den letzten Jahren stark gewachsenen Netzwerkanwendungen: „high growth area in networking“ (Kurose & Ross 2017).

Ablauf



- Entsprechend ergibt sich folgender geplanter Ablauf (vgl. OLAT!)

Semesterplanung (Änderungen vorbehalten)

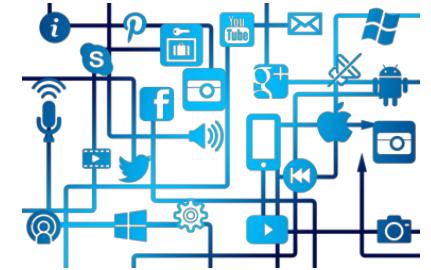
Semesterplanung BW332 (Sommersemester 2019) : Ablauf					
BW332 Rechnernetze (mit Praktikum)					
KW	Datum	Tag	Typ	Nr.	Thema
12	21.03.2019	Do	VL	1	Veranstaltungsüberblick, Einführung in Computernetzwerke
13	28.03.2019	Do	VL	2	Einführung in Computernetzwerke (Teil 2)
14	04.04.2019	Do	Online	3	Einführung in Computernetzwerke (Teil 3) - Keine Präsenz - Online-Vorlesung
15	11.04.2019	Do	VL	4	Anwendungsschicht (Teil 1)
16	18.04.2019	Do	VL	5	Anwendungsschicht (Teil 2)
17	25.04.2019	Do	VL	6	Transportschicht (Teil 1)
18	02.05.2019	Do	Online	7	Transportschicht (Teil 2) - Keine Präsenz - Online-Vorlesung
19	09.05.2019	Do	VL	8	Transportschicht (Teil 3)
20	16.05.2019	Do	VL	9	Netzwerkschicht (Teil 1)
21	23.05.2019	Do	VL	10	Netzwerkschicht (Teil 2)
22	30.05.2019	Do			Feiertag
23	06.06.2019	Do	VL	11	Netzwerkschicht (Teil 2)
24	13.06.2019	Do	Online	12	Klausurvorbereitung + PacketTracer Übungen
25	20.06.2019	Do			Feiertag
26	27.06.2019	Do			Start Prüfungen
27	04.07.2019	Do			
28	11.07.2019	Do			Ende Prüfungen

Literatur



- Kurose, James F.; Ross, Keith W. (2017): Computer networking. A top-down approach. Seventh edition, global edition. Boston, Columbus, Indianapolis, Amsterdam, Cape Town: Pearson Education.
 - Auf diesem Buch basiert die Veranstaltung. Es ist auch auf deutsch verfügbar.
 - Die 6. Auflage ist online verfügbar und ausreichend für diese Veranstaltung.
 - Verweise beziehen sich auf die 6. Auflage, so dass jeder sie nachschlagen kann!
 - Zusätzliches Material der Autoren:
https://wps.pearsoned.com/ecs_kurose_compnets_6/216/55463/14198700.cw/index.html
- Tanenbaum, Andrew S.; Wetherall, David (2014): Computer networks. 5. ed., Pearson new internat. ed. Harlow, Essex: Pearson Education (Pearson custom library). *Hinweis: auch auf deutsch verfügbar.*
- Zisler, Harald (2016): Computer-Netzwerke. Grundlagen, Funktionsweise, Anwendung. 4., aktualisierte und erweiterte Auflage. Bonn: Rheinwerk Verlag GmbH (Rheinwerk Computing).

Begleitmaterialien



- Vorlesungsunterlagen
 - In der Regel vor der Veranstaltung (mit „Lücken“) bereitgestellt.
 - Kleinere Übungen finden sich in den Vorlesungsfolien.
- Evtl. Übungsblätter
 - Diese sind teilweise vorzubereiten und werden in der Veranstaltung an den mit „UE“ gekennzeichneten Terminen besprochen.
- Links auf diverse Videos und Tools, die Zusammenhänge visualisieren
- Tools
 - Kommandozeilen-Programme (tracert, nslookup, ping, ...)
 - Teilweise Remote-Zugang zu Wireshark
 - Packet Tracer (Cisco) für die Simulation von Netzwerken und deren Konfiguration

**BW332 - Rechnernetze
(mit Praktikum)**

Vorlesungen 1-3
Einführung in Computernetzwerke



Ziele von VL 1-3



- Überblick über die **Terminologie** im Kontext von Rechnernetzen erhalten
- Ein erstes Grundverständnis für die **Zusammenhänge in Computernetzwerken** anhand des Internets erlangen
- Die **Herausforderungen** im Betrieb komplexer Netzwerke erkennen

Hinweis: Vorlesungen 1-3 enthalten sehr viel Material, das zunächst oberflächlich behandelt wird. Wir werden in den weiteren Veranstaltungen die Themen im Detail besprechen und Ihre Kenntnisse mit diversen Übungen festigen.

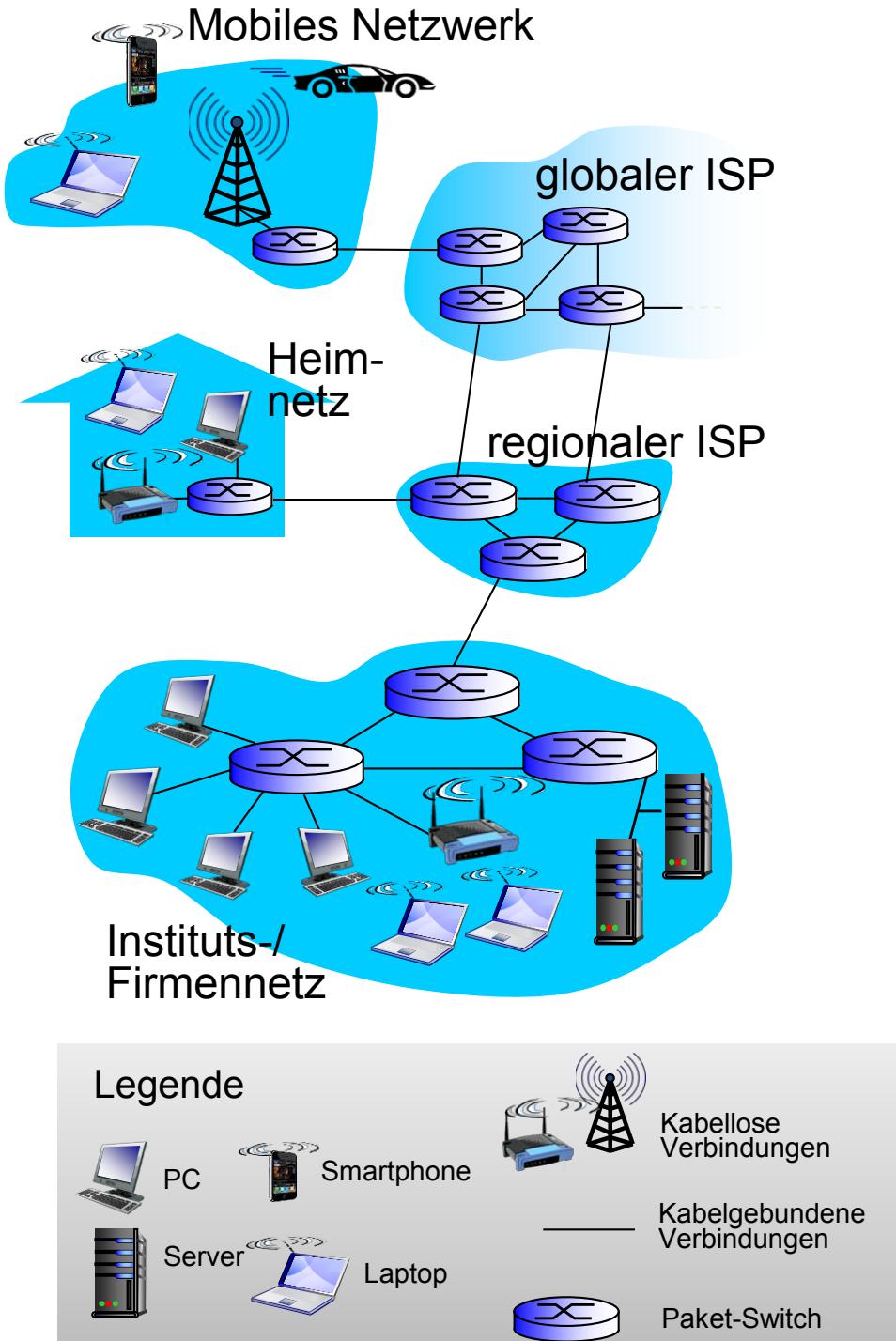
■ Internet und Protokolle

- Randbereich des Internets
- Inneres des Internets
- Netzwerkenngrößen
- Protokollsichten, Dienste und Netzwerksicherheit
- Geschichte des Internets
- Zusammenfassung und Ausblick

Internet – Grundlagen

- Millionen vernetzter Computer:
 - Hosts = Endsysteme
 - Auf denen Netzwerkanwendungen laufen
- Leitungen/Funkstrecken (“Links”)
 - Glasfaser, Kupfer, Funk, Satellit
 - Übertragungsrate: **Bandbreite**
- **Paket-Switches:** leiten Datenpakete (Einheiten von Daten) weiter
 - Router und Switches

ISP = Internet Service Provider



Beispielhafte Endsysteme und Anwendungen

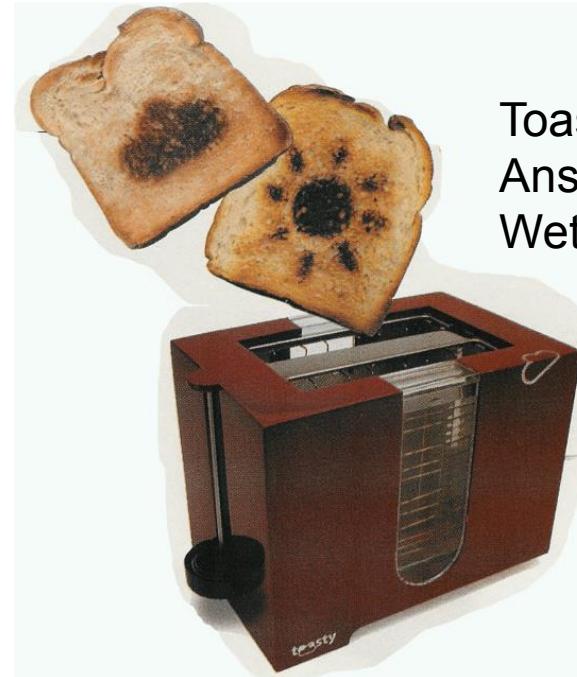
Heiamtomaticierung

SMART HOME



Bewässerung per App
mit Sensoranschluss und Verbindung
zu Wetterdiensten

... und viele mehr.



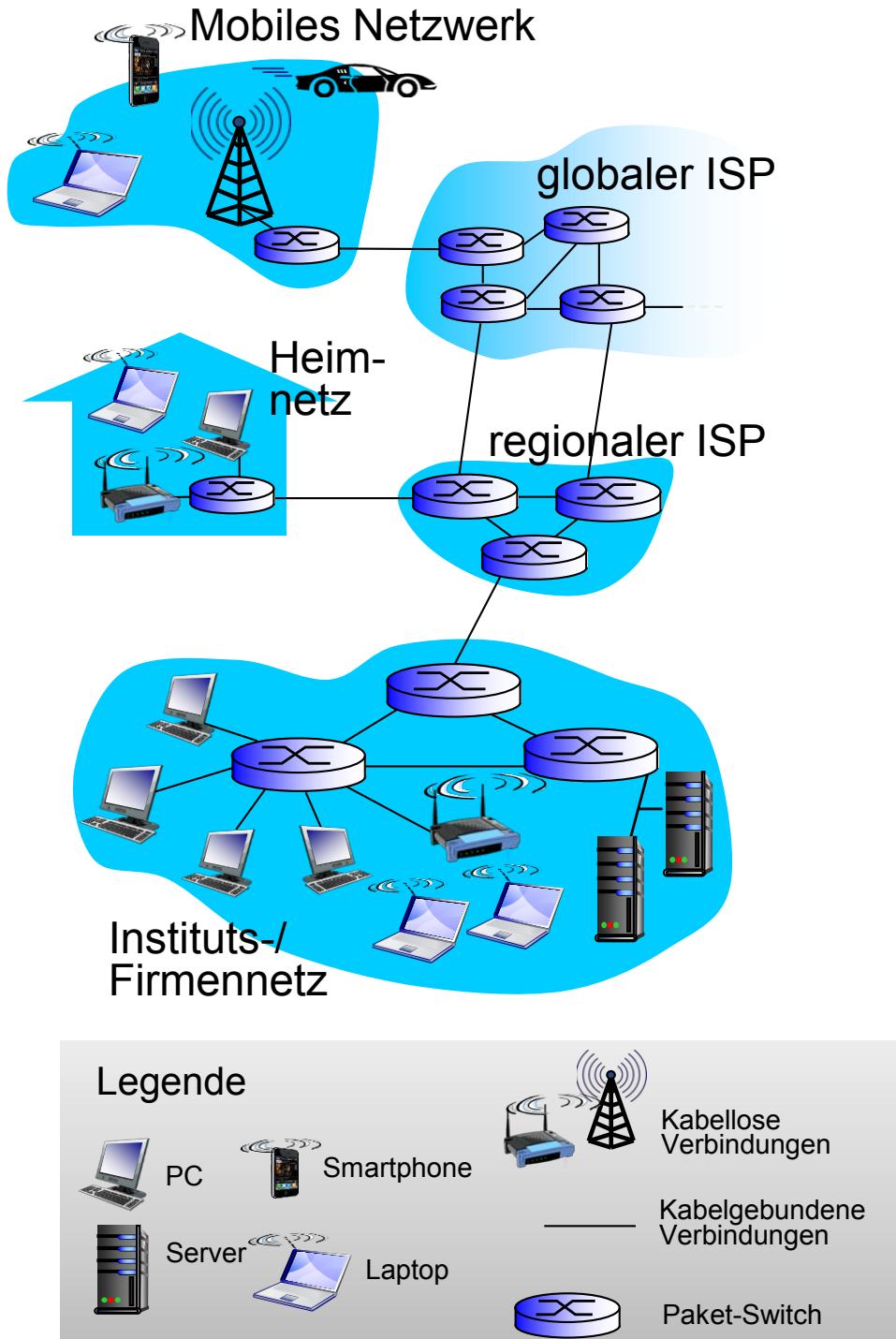
Toaster mit Web-
Anschluss +
Wetterprognose



Internet-Telefonie

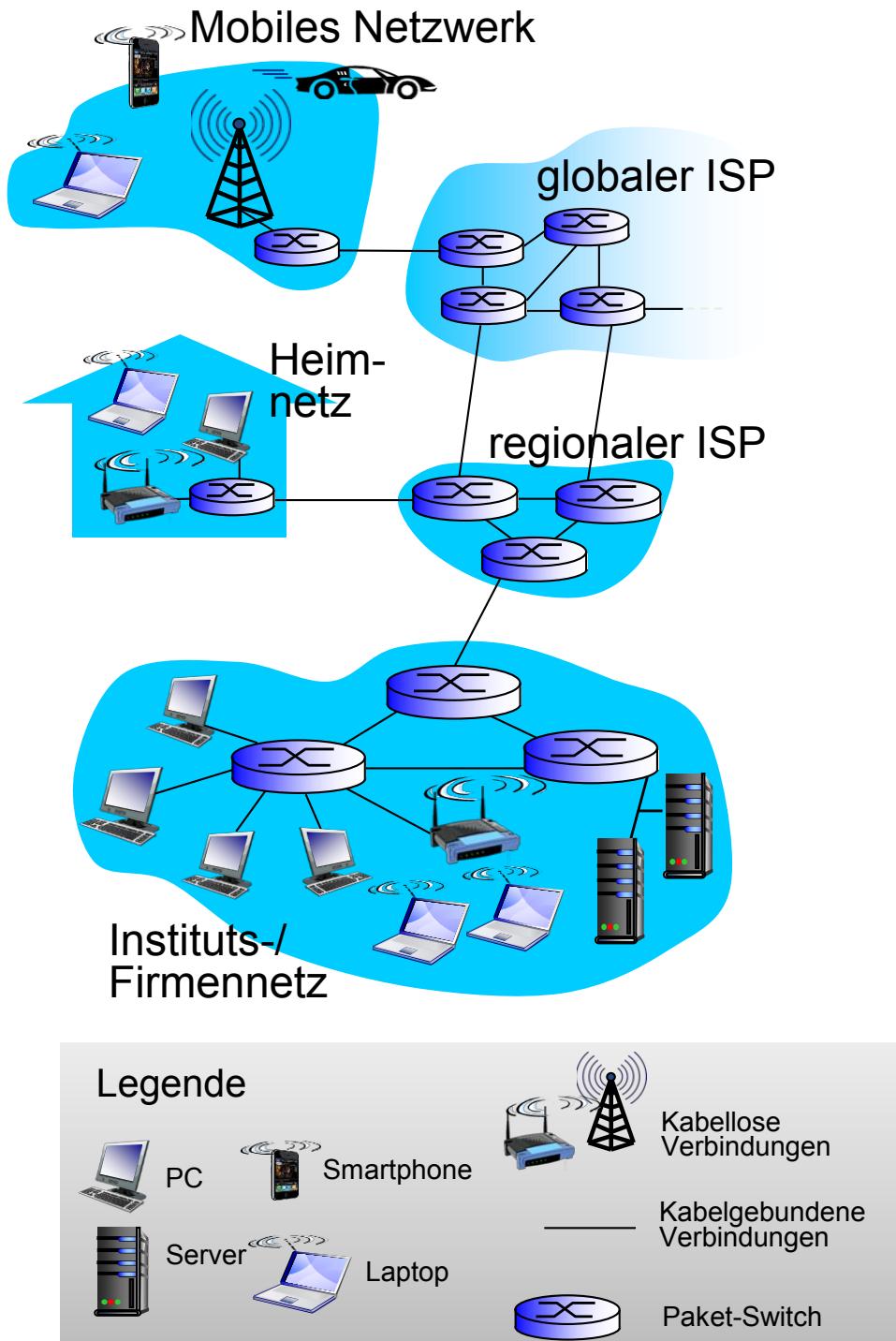
Internet – Grundlagen

- Internet: “Netzwerk von Netzwerken”
 - Hierarchisch
 - Öffentliches Internet und privates Intranet
 - Verbundene ISPs
- Protokolle kontrollieren das Senden und Empfangen von Nachrichten
 - z.B., TCP, IP, HTTP, Skype
- Internetstandards
 - RFC: Request For Comments
 - IETF: Internet Engineering Task Force



Internet – Dienste

- Das Internet ist eine Infrastruktur, die Anwendungen **Dienste** anbietet:
 - Web, VoIP, E-Mail, File Sharing, ...
- Dienste stellen **Schnittstellen** (Interfaces) zur Verfügung
 - Bspw. Programmierschnittstellen zum Senden und Empfangen von Daten („Verbindung zum Internet“)
- Beispiele
 - Zuverlässige Datenübertragung von einer Quelle zu einem Ziel
 - Unzuverlässige (“best effort”) Datenübertragung



Protokolle

„Menschliche“ Protokolle

- „Wie viel Uhr ist es?“
- „Ich habe eine Frage“
- Gegenseitiges Vorstellen

... es werden „standardisierte“ Nachrichten übertragen

... durch den Empfang dieser Nachrichten werden „standardisierte“ Aktionen ausgelöst

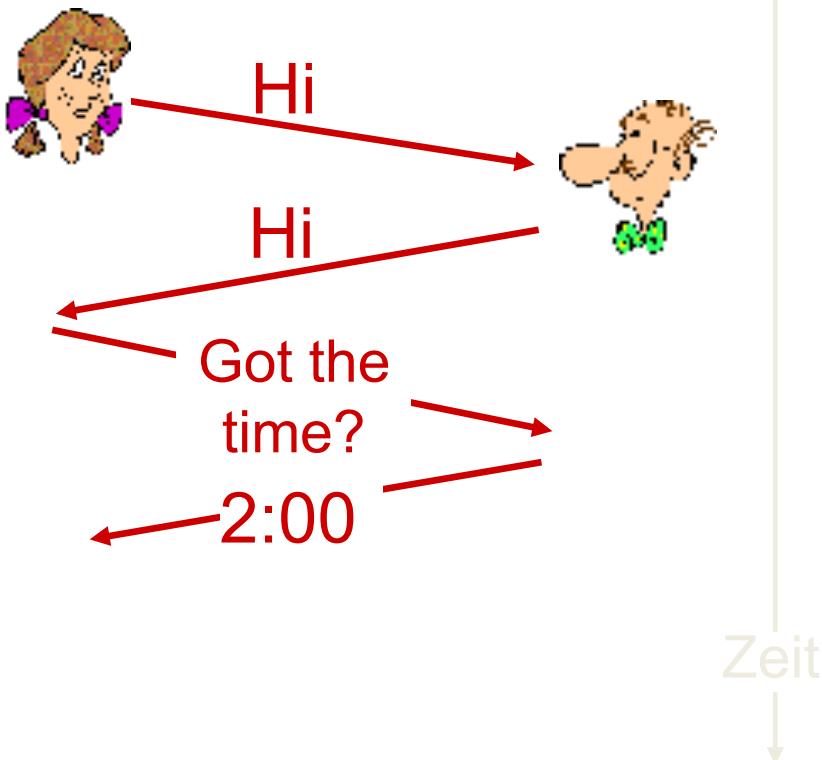
Netzwerkprotokolle

- Maschinen statt Menschen
- Sämtliche Kommunikation im Internet wird durch Protokolle geregelt

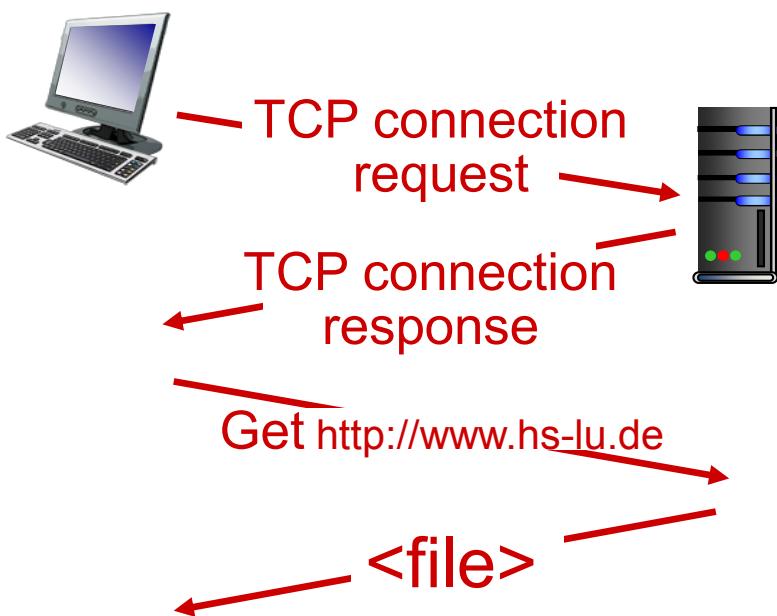
Protokolle definieren das **Format und die Reihenfolge**, in der Nachrichten von Systemen im Netzwerk gesendet und empfangen werden, **sowie die Aktionen**, welche durch diese Nachrichten ausgelöst werden.

Protokolle

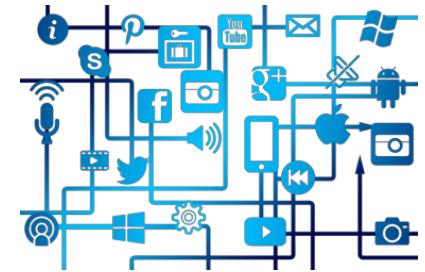
„Menschliches“ Protokoll



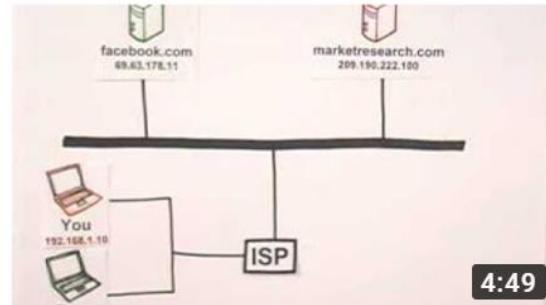
Computernetzwerk-Protokoll



Weiterführendes Material



- **Video „How the Internet Works in 5 Minutes“**
https://www.youtube.com/watch?v=7_LPdttKXPc

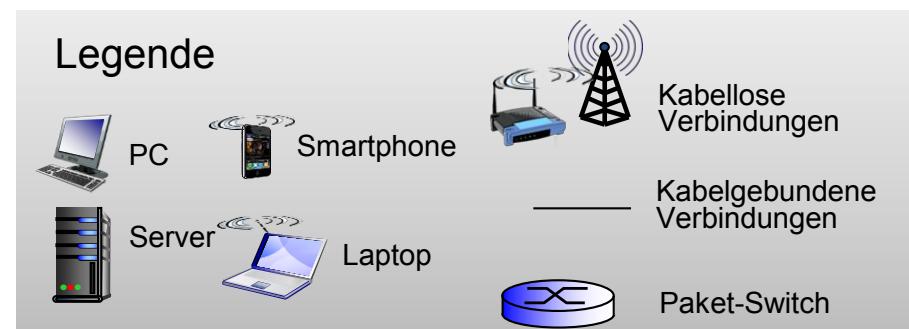
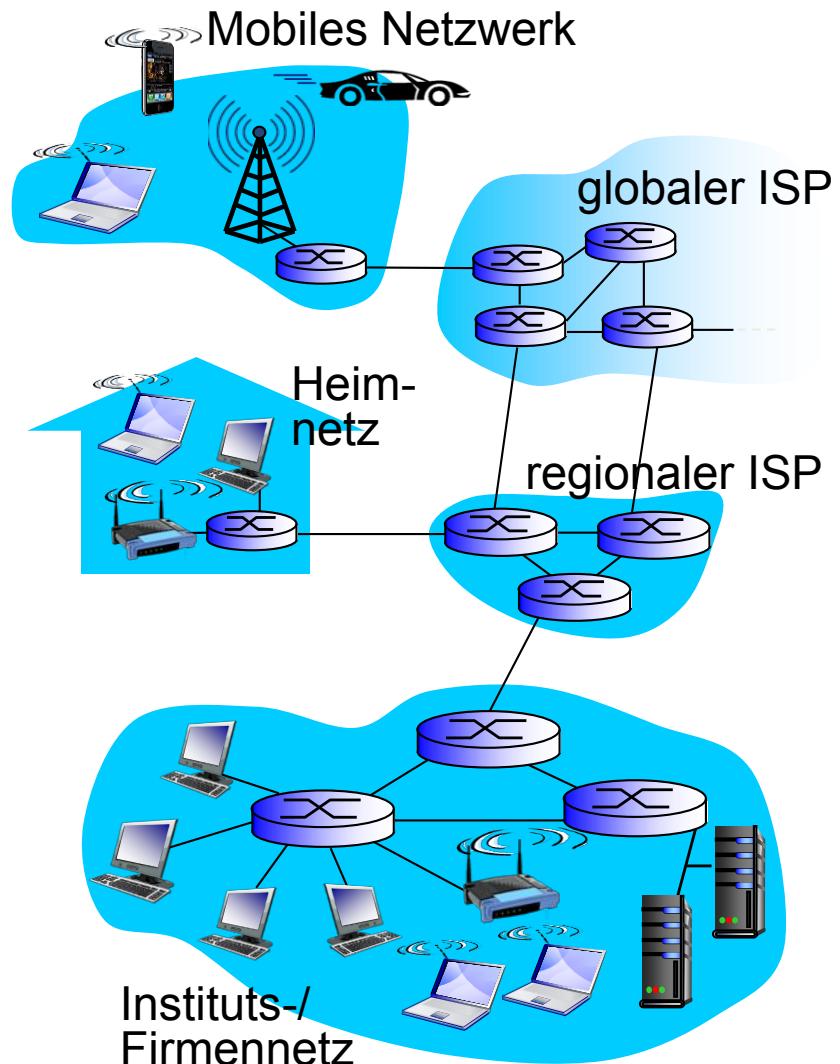


How the Internet Works in 5 Minutes
Aaron • 1,4 Mio. Aufrufe • vor 8 Jahren

- Internet und Protokolle**
- Randbereich des Internets**
- Inneres des Internets**
- Netzwerkkenngrößen**
- Protokollsichten, Dienste und Netzwerksicherheit**
- Geschichte des Internets**
- Zusammenfassung und Ausblick**

Netzwerkstruktur „Network edge“

- **Randbereich:** Anwendungen und Endsysteme
 - Hosts: Clients und Server
 - Server befinden sich oft in Data Centern
- **Zugangsnetzwerke, physikalische Medien:** Kabel und drahtlose Verbindungen
- Inneres des Netzwerkes (**network core**):
 - Verbundene Router
 - Netzwerke von Netzwerken



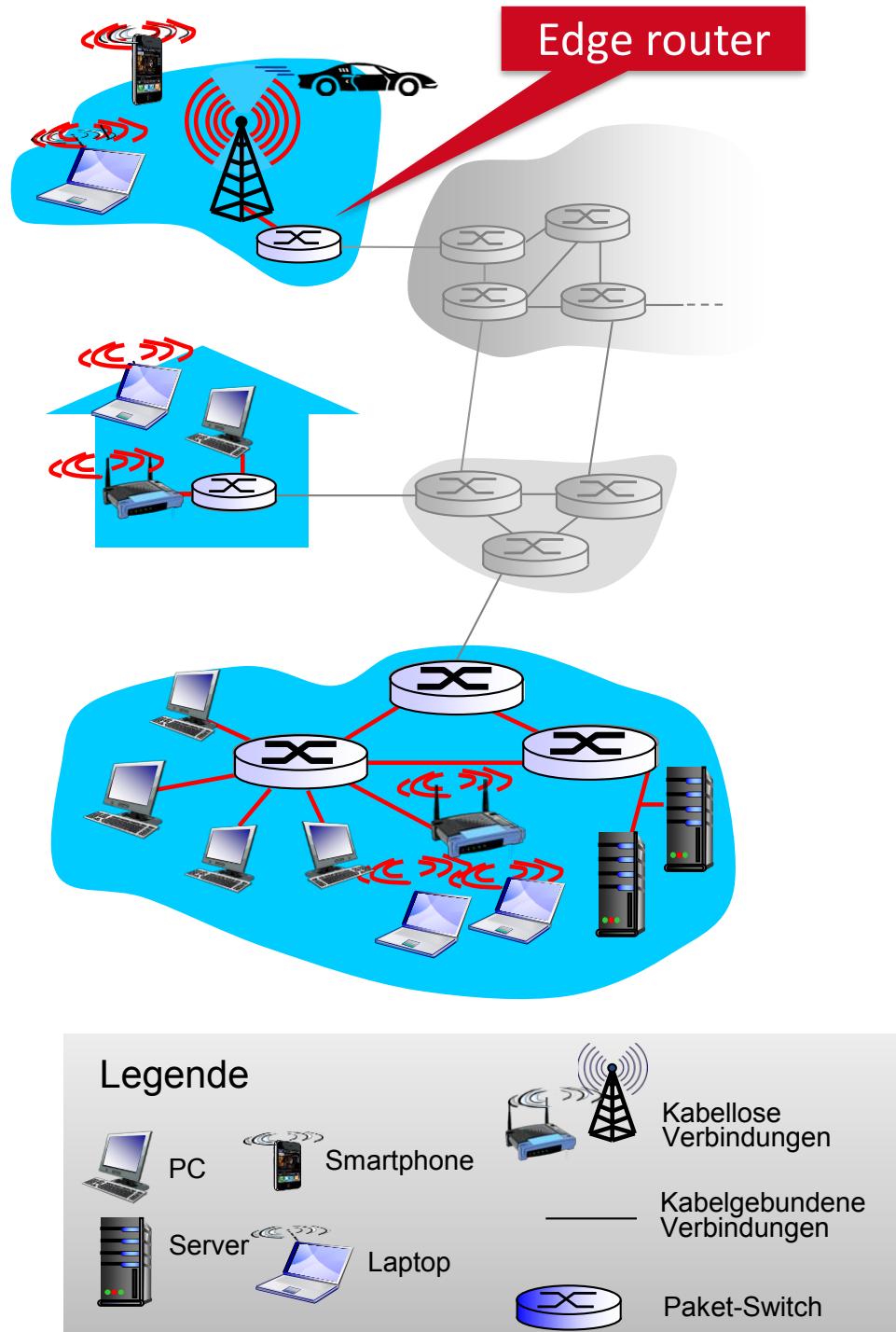
Randbereiche des Netzwerks

Wie können Endsysteme mit einem Rand-Router (edge router) verbunden werden?

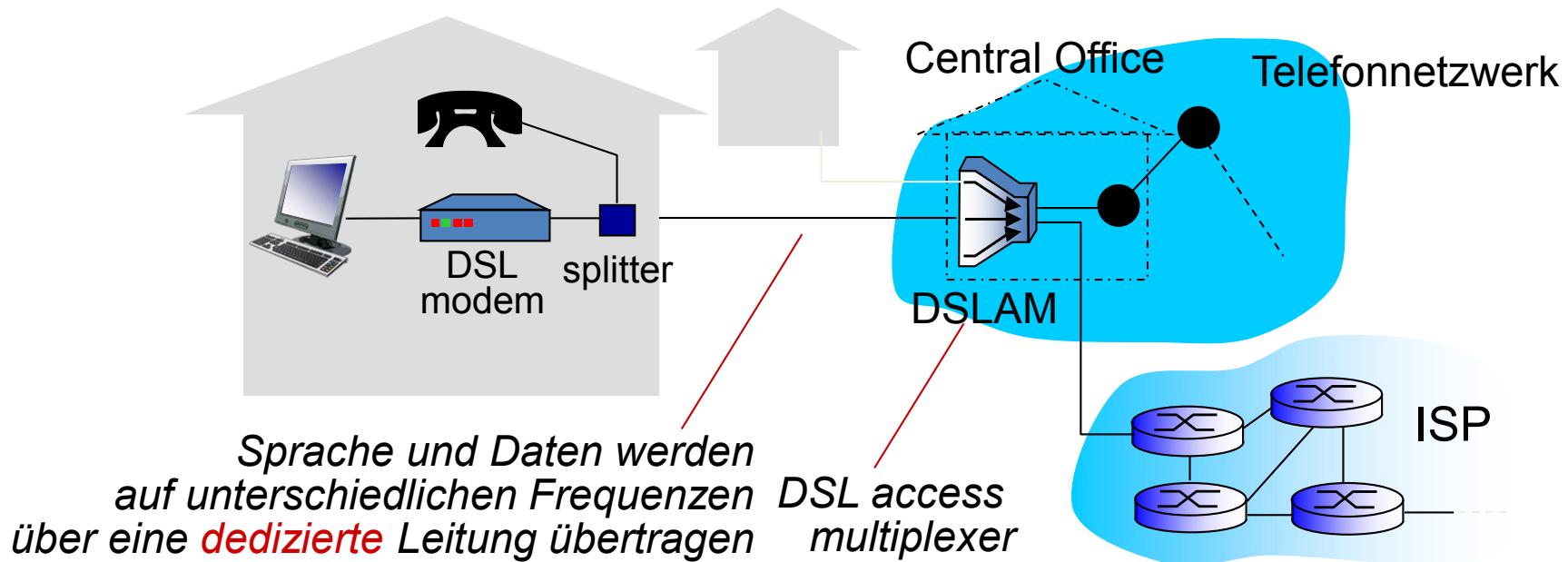
- Private Heimnetze
- Institutionen (Hochschule, Firmennetzwerke, ...)
- Mobile Anbindung

Wichtige Eigenschaften

- Bandbreite (Bits pro Sekunde) der Anbindung?
- Geteilte oder dedizierte (exklusive) Nutzung?

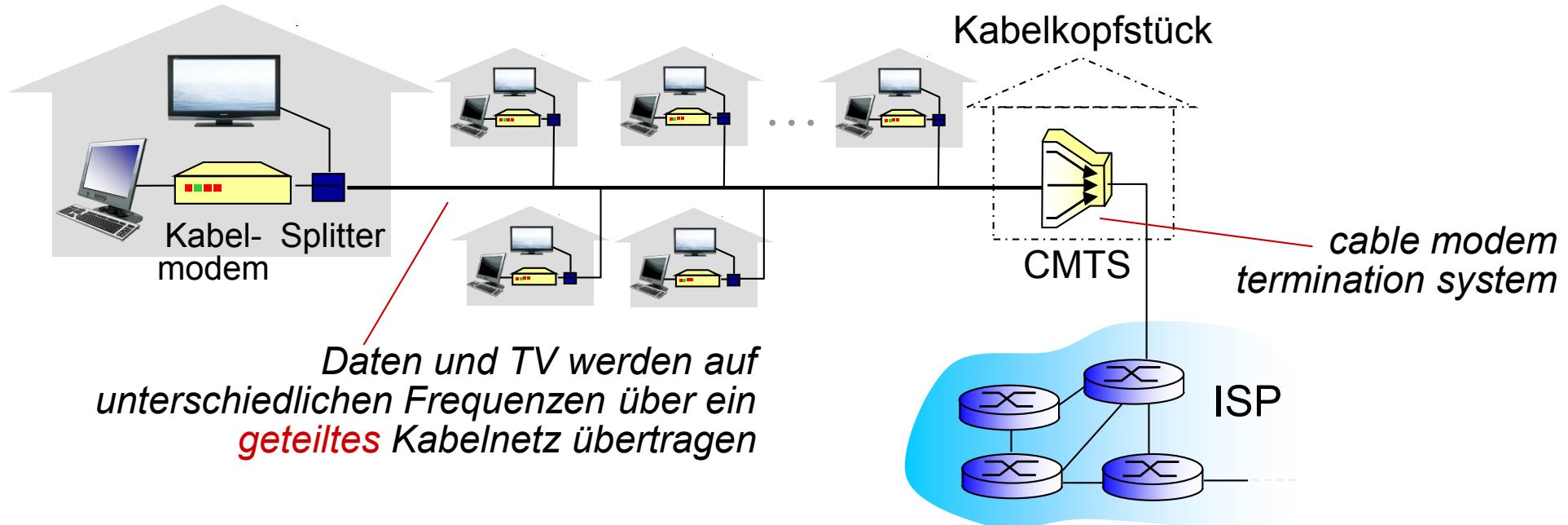


Zugangsnetz: digital subscriber line (DSL)



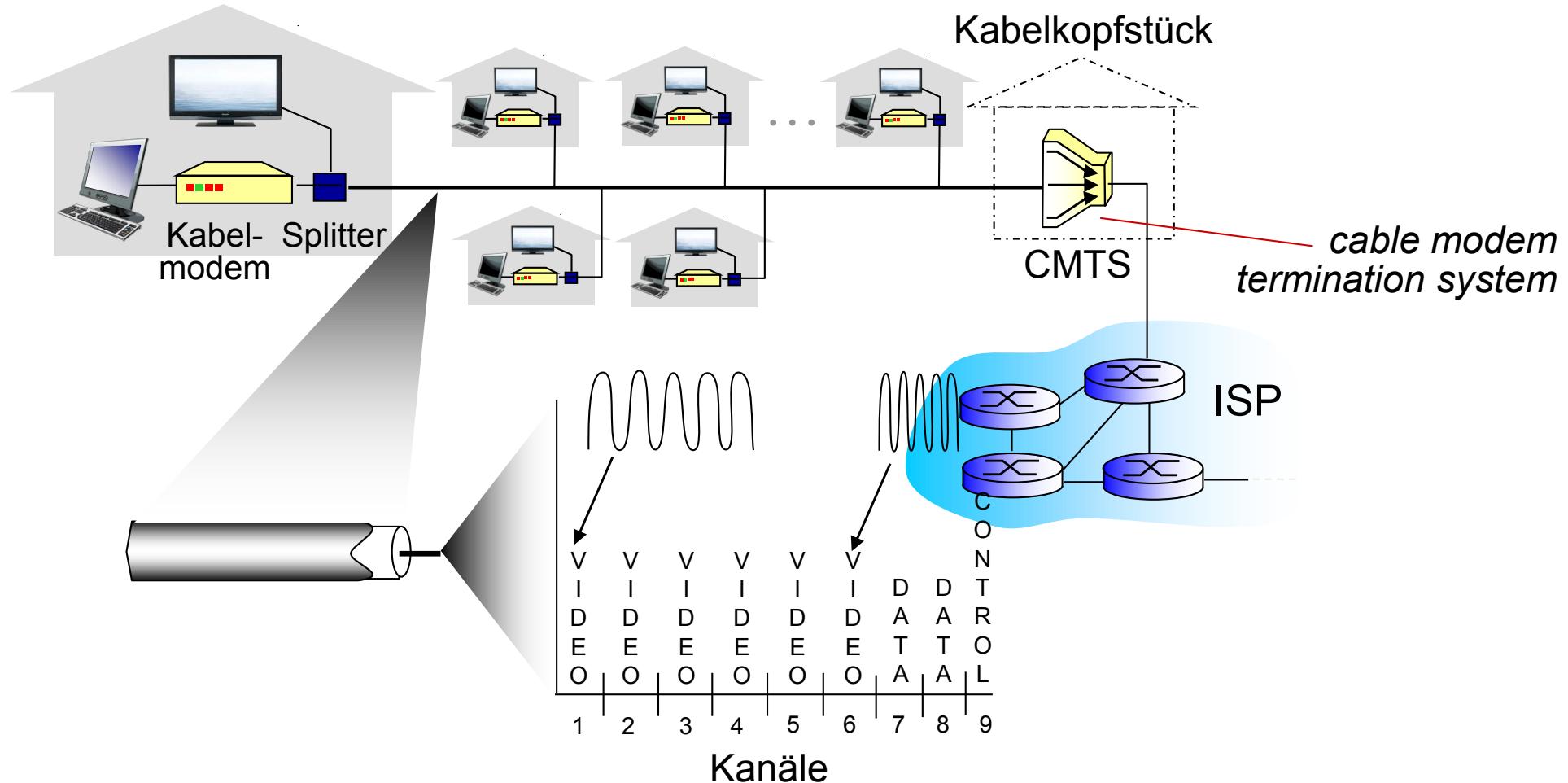
- Nutzung einer existierenden Telefonleitung (zum central office DSLAM)
 - Daten laufen über die DSL Leitung ins Internet (mittels ISP)
 - Sprache läuft über DSL in das Telefonnetzwerk
- Verschiedene Bandbreiten (heute bis zu 200Mbit/s mit VDSL2)

Zugangsnetz: Kabelnetzwerk



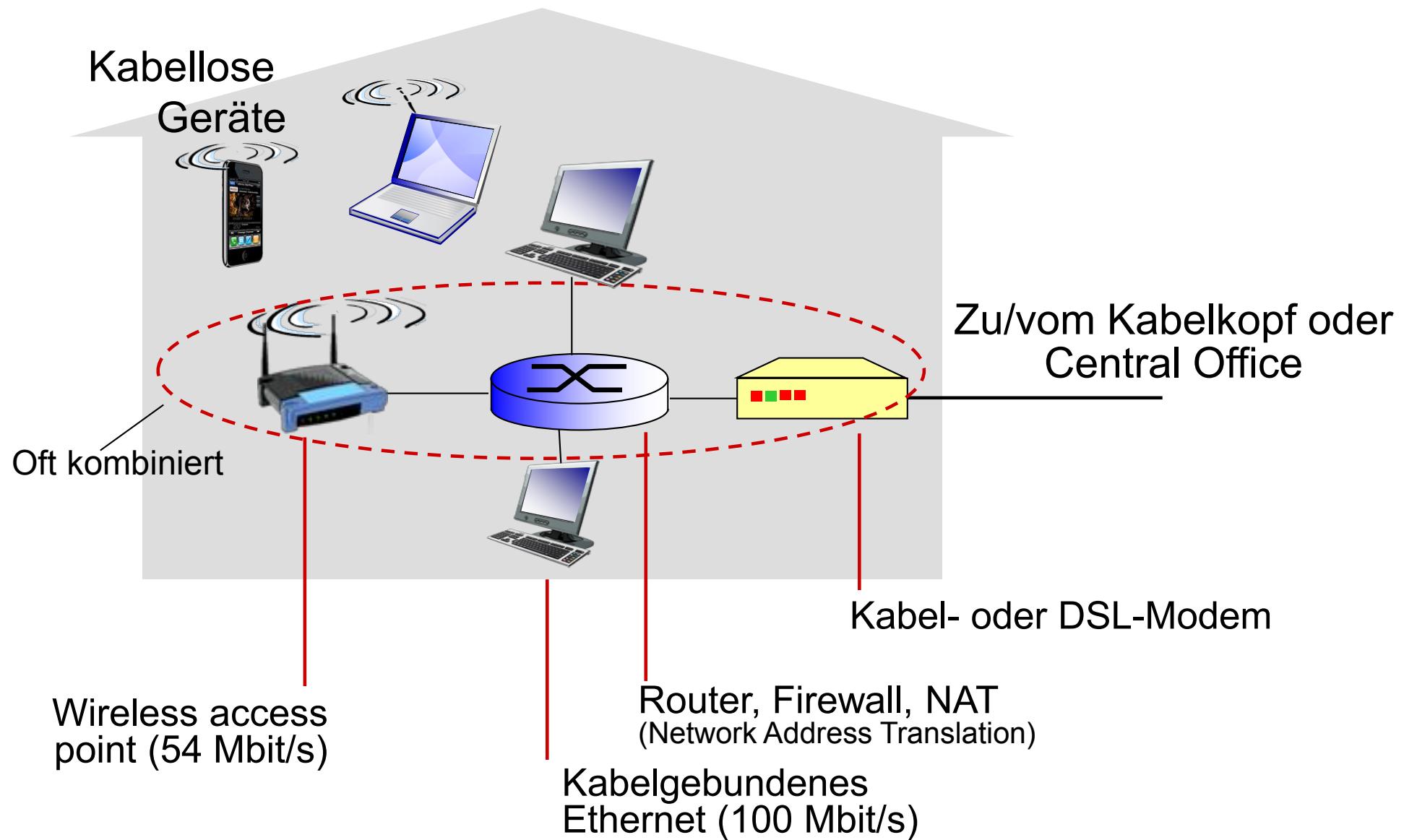
- HFC: hybrid fiber coax
- Koaxialkabel zum Anschluss der Haushalte; Glasfaserstrecken zum Anschluss an den ISP Router (und kabelnetzintern)
 - Mehrere Haushalte teilen sich das Zugangsnetzwerk zum Kabelkopfstück
 - Unterschied zu DSL!

Zugangsnetz: Kabelnetzwerk

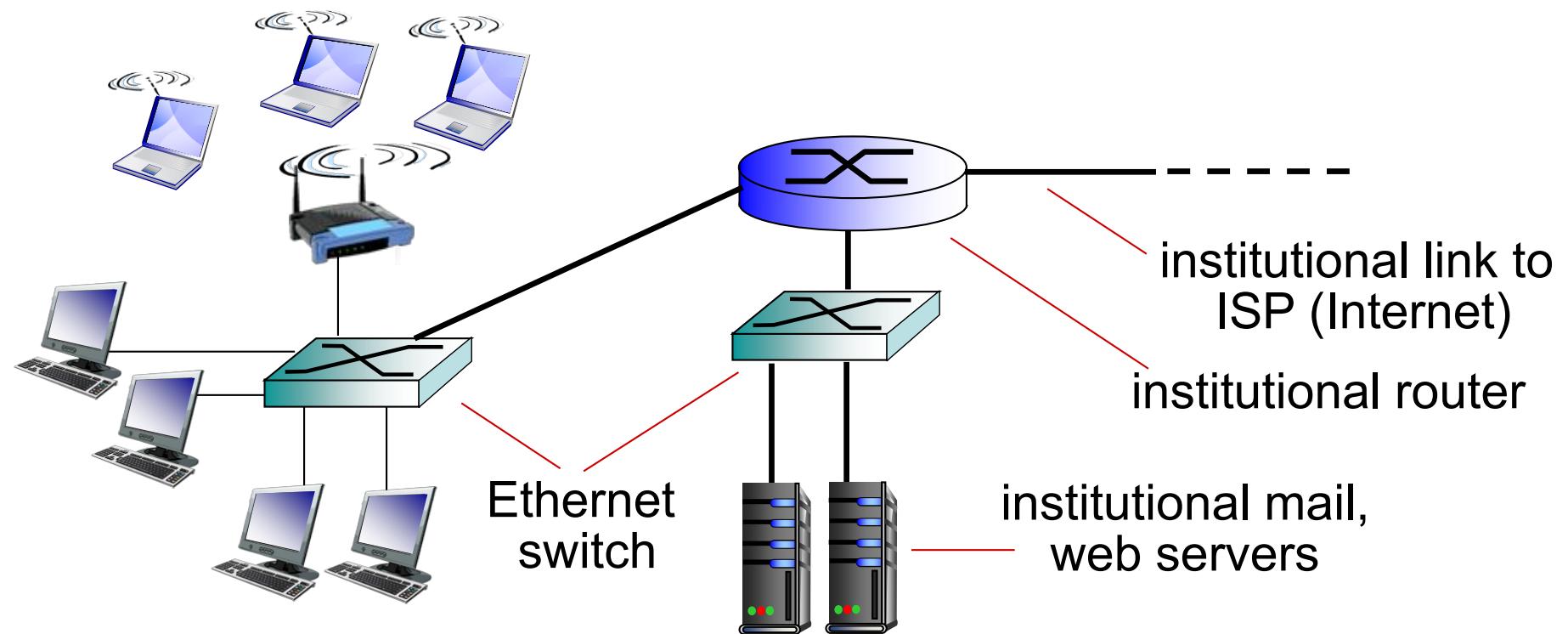


- Frequency Division Multiplexing: verschiedene Kanäle werden in unterschiedlichen Frequenzbändern übermittelt

Zugangsnetz: Heimnetzwerk



Zugangsnetz: Institutionelle Netze / Enterprise Access Networks, Ethernet

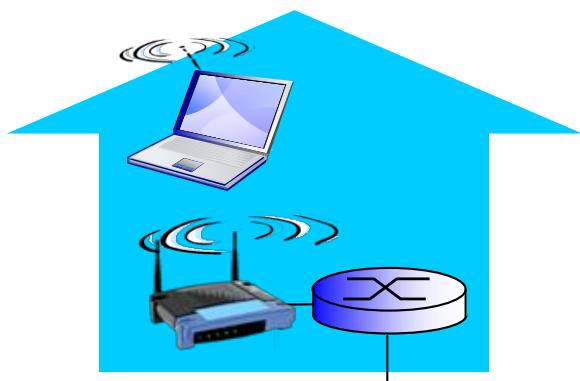


- Typischerweise in Unternehmen, Hochschulen etc. eingesetzt
- 10 Mbit/s, 100Mbit/s, 1Gbit/s, 10Gbit/s Übertragungsrate
- Üblicherweise werden Endsysteme an Ethernet Switches angeschlossen

Kabellose Netzwerke

Wireless LANs:

- Innerhalb von Gebäuden und im Nahbereich
- 802.11b/g/n (WiFi): 11, 54, 600 Mbit/s Übertragungsrate



Zum Internet

Wide-area wireless access

- Telekommunikations-Anbieter
- 3G, 4G: LTE

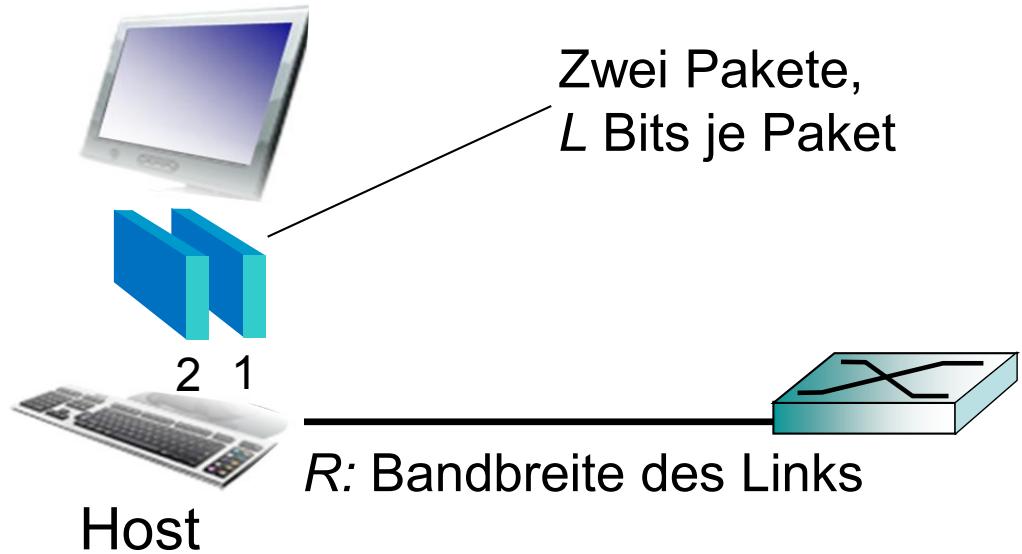


Zum Internet

Verbindung: Host und Link

Host in der Sendefunktion:

- Nimmt von Anwendungen Nachrichten entgegen
- Teilt diese in kleinere Stücke (Pakete) mit Länge L (Bit) auf
- Übermittelt die Pakete in das Zugangsnetzwerk (Link) mit der Übertragungsrate R (=Kapazität = Bandbreite)



$$\text{Verzögerung in der Paket-übertragung} = \frac{\text{Zeit, um } L\text{-Bit Paket über den Link zu übertragen}}{R \text{ (Bits/sec)}} = \frac{L \text{ (Bits)}}{R \text{ (Bits/sec)}}$$

Maßeinheiten für die Datenrate

- $1 \text{ kbit/s} = 1 \text{ kb/s} = 1 \text{ kbps} = 1.000 \text{ bit/s}$
- $1 \text{ Mbit/s} = 1 \text{ Mbps} = 1.000.000 \text{ bit/s} = 1.000 \text{ kbit/s}$
- $1 \text{ Gbit/s} = 1 \text{ Gbps} = 1.000.000.000 \text{ bit/s} = 1.000 \text{ Mbit/s}$

Achtung: Unterschied Bit und Byte (1 Byte = 8 Bit)

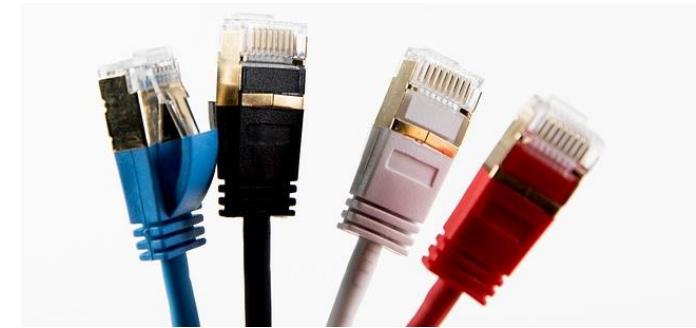
$1 \text{ kBps} = 8 \text{ kbps} = 8.000 \text{ bit/s} = 1.000 \text{ Byte/s}$

Byte werden üblicherweise mit „großem B“ abgekürzt.

Physikalische Medien

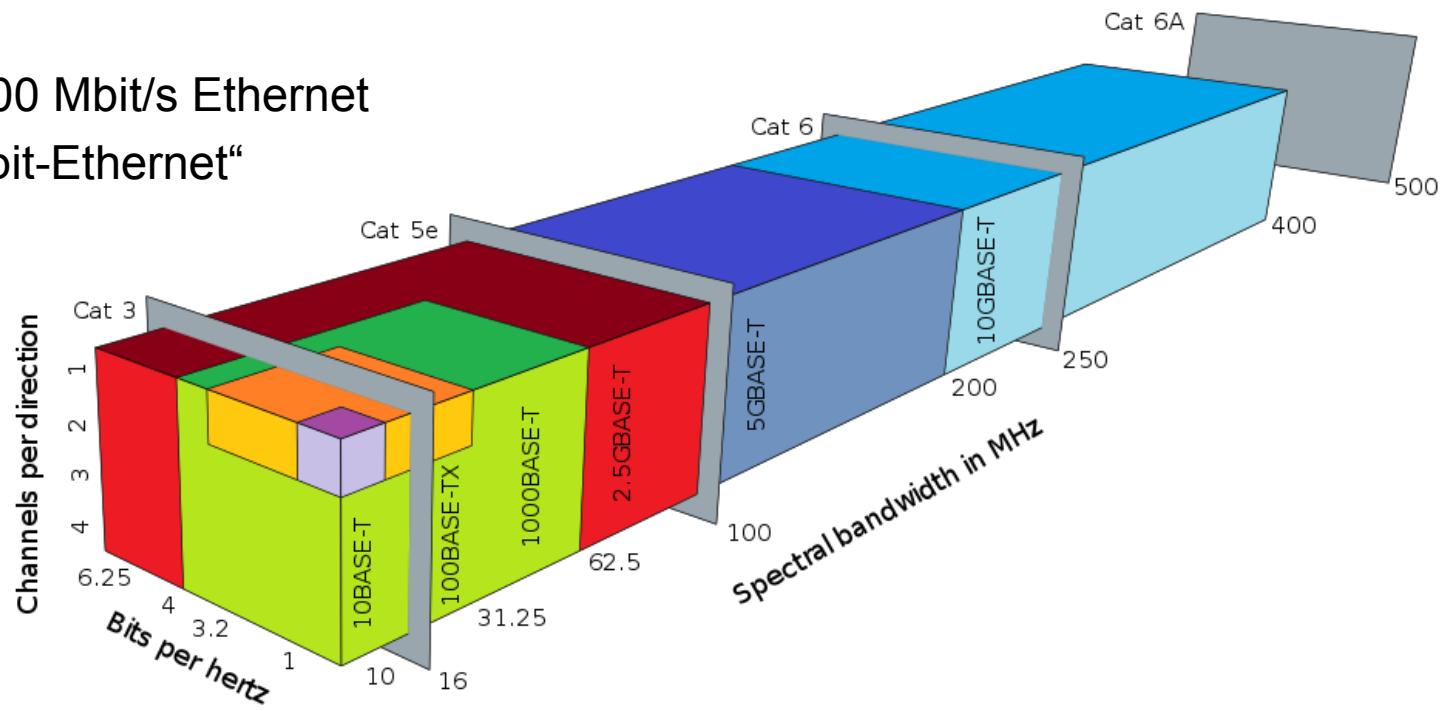
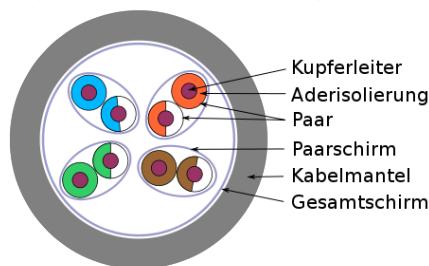
- Bit: wird von einem Sender zu einem Empfänger übertragen
- Leitung: das, was sich zwischen Sender und Empfänger befindet
- Gerichtete Medien (guided):
 - Signale breiten sich in festen Medien in eine Richtung aus: Kupfer-, Glasfaser-, Koaxialkabel
- Ungerichtete Medien (unguided):
 - Signale breiten sich frei aus: Funk, Mikrowellen

Physikalische Medien: Twisted Pair / Ethernet



- Vielfach verwendet:
Twisted Pair (TP)
- Zwei isolierte Kupferleiter
 - Kategorie 3: Telefonkabel, 10 Mbit/s
Ethernet
 - Kategorie 5: 100 Mbit/s Ethernet
 - Kat. 5e: „Gigabit-Ethernet“
 - ...

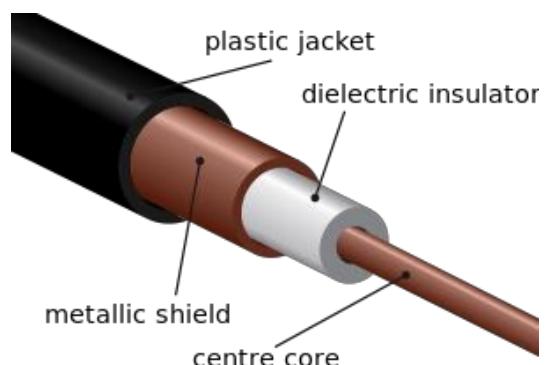
S/FTP (screened, foiled TP)



Physikalische Medien: Koaxial- und Glasfaserkabel

Koaxalkabel:

- Zwei konzentrisch angeordnete Kupferleiter
- Bidirektional
- Basisband:
 - Ein Kanal auf dem Kabel
 - „Altes“ Ethernet
- Breitband:
 - Mehrere Kanäle auf dem Kabel



Glasfaserkabel:

- Glasfaserkabel übertragen Lichtpulse, jeder Puls ist ein Bit
- Hohe Geschwindigkeit:
- Mehrere 10 Gbit/s – mehrere 100 Gbit/s
- Geringe Fehlerrate unempfindlich gegen elektromagnetische Störer

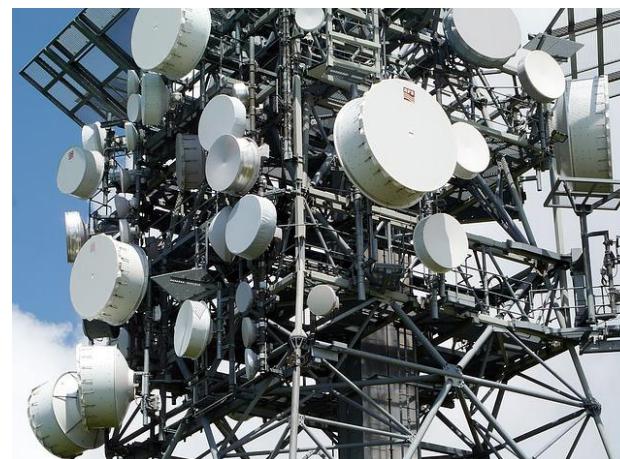


Physikalische Medien: Funk

- Signal wird von elektromagnetischen Wellen übertragen
- Kein „Draht“ (drahtlose Kommunikation)
- Bidirektional
- Signalausbreitung wird von der Umgebung beeinflusst:
 - Reflexion
 - Abschattung durch Hindernisse
 - Interferenz

Arten von Funk:

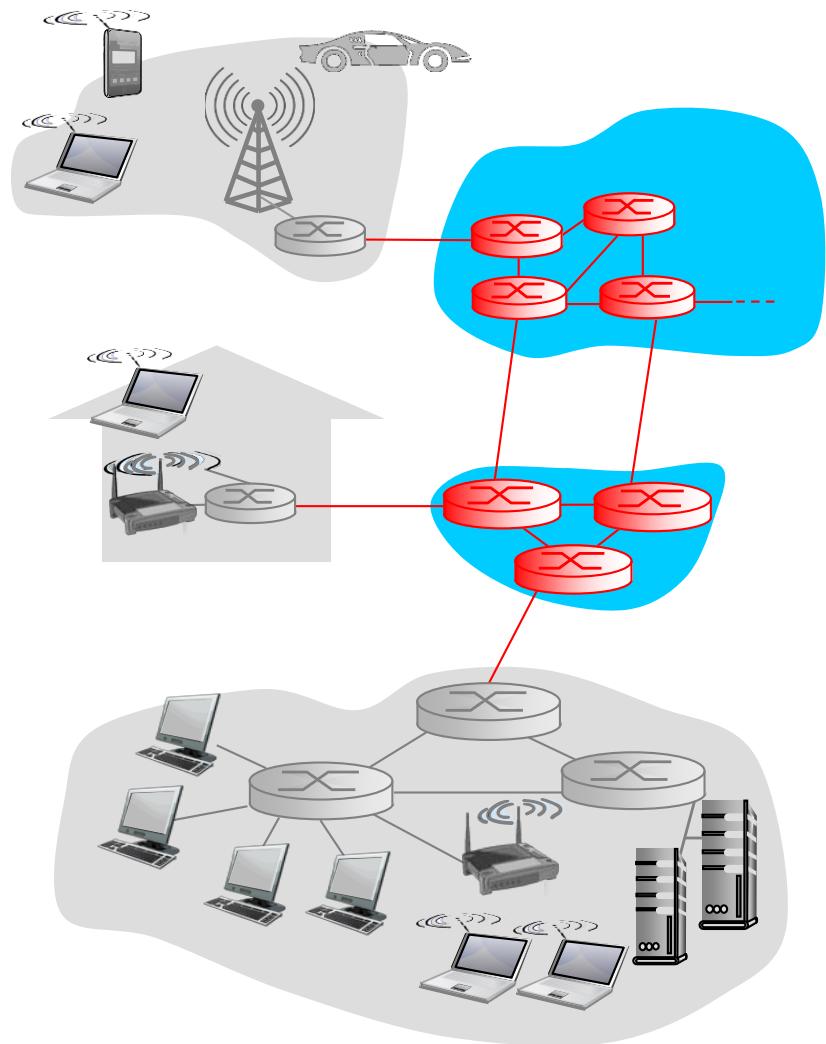
- WLAN (auch Wifi)
 - 11 Mbit/s, 54 Mbit/s, ...
- Weitverkehr (zellulär)
 - 3G/UMTS/LTE: bis zu 150 Mbit/s
- Mikrowelle
- Satellit



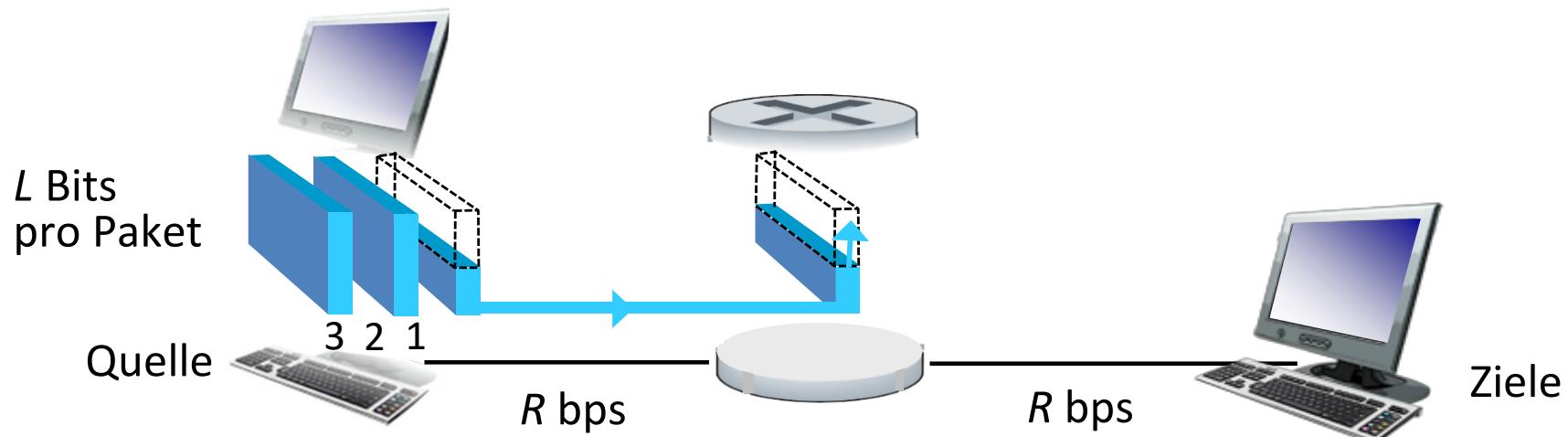
- Internet und Protokolle**
- Randbereich des Internets**
- Inneres des Internets**
- Netzwerkkenngrößen**
- Protokollsichten, Dienste und Netzwerksicherheit**
- Geschichte des Internets**
- Zusammenfassung und Ausblick**

Netzwerkstruktur „Network Core“

- Netz verbundener Router
- Zentraler Unterschied: Leitungs- vs. Paketvermittlung
 - Leitungsvermittlung: **dedizierte** Leitung (Telefonnetz)
 - Paketvermittlung: Daten werden in **diskreten** Einheiten (Paketen) durch das Netz geleitet (Internet)
 - Weiterleitung der Pakete von einem Router zum nächsten über diverse Verbindungen (Links) von der Quelle zum Ziel
 - Jedes Paket wird mit der vollen Verbindungskapazität von einem Router zum nächsten übertragen



Paketvermittlung: store-and-forward



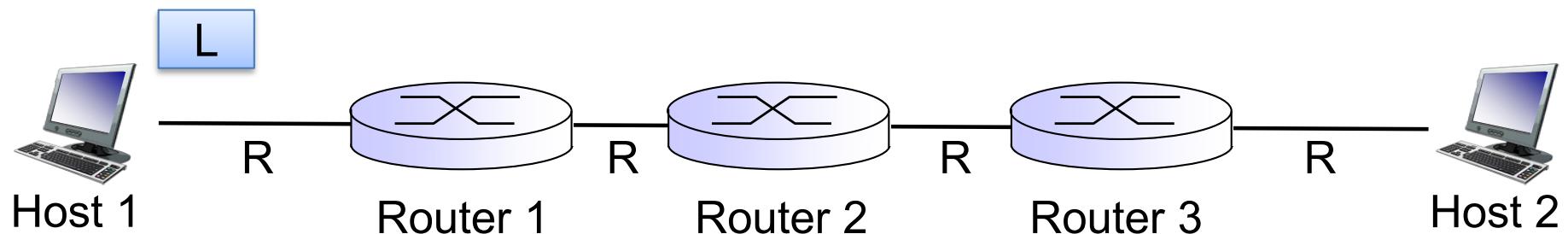
- L/R Sekunden dauert es, um ein L -Bit Paket über den Link (mit R bps) zu übertragen
 - „Store-and-forward“: das gesamte Paket muss beim Router ankommen, bevor es an den nächsten Link weitergegeben werden kann
- Verzögerung im Beispiel = $2 L/R$
 - Annahme: keine Ausbreitungsverzögerung

Verzögerung im Beispiel:

- $L = 7,5 \text{ Mbit}$
- $R = 1,5 \text{ Mbit/s}$
- Übertragungsverzögerung für einen „Hop“ = 5s
- Gesamtverzögerung der Übertragung = 10s

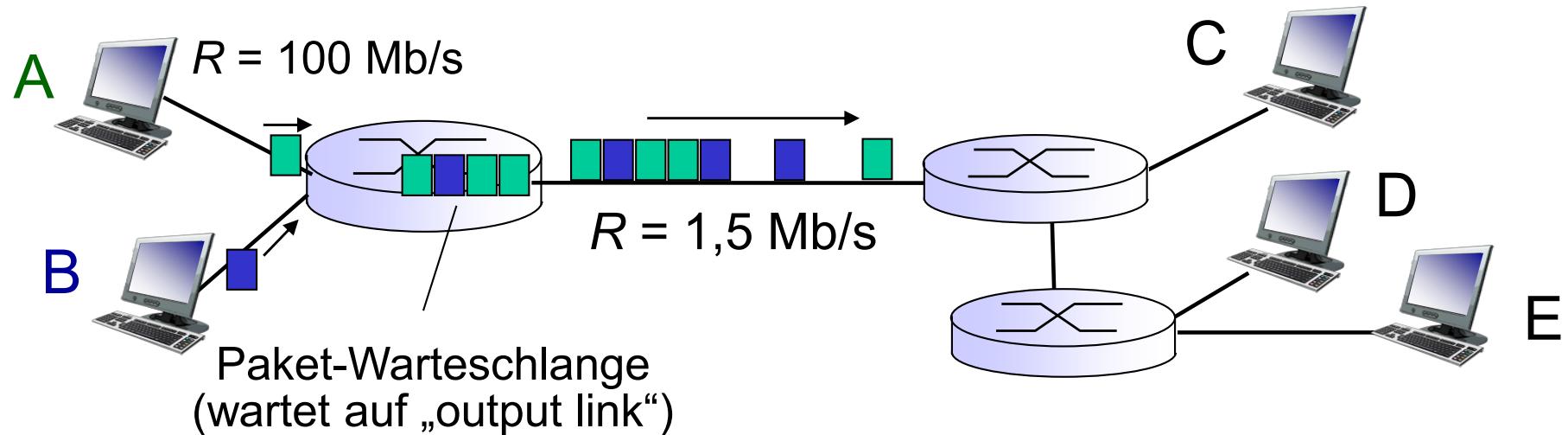
Übung

- Gegeben ist folgendes Netzwerk mit drei Routern:



- Die Übertragungsrate R zwischen allen Geräten ist 10Mbit/s
- Sie möchten eine Datei der Größe L = 5 Mbyte von Host 1 zu Host 2 übertragen.
- Wie groß ist die Gesamtverzögerung zur Übertragung des Pakets (unter Vernachlässigung der Ausbreitungsverzögerung, diese behandeln wir später)?

Paketvermittlung: Warteschlangen und Paketverlust

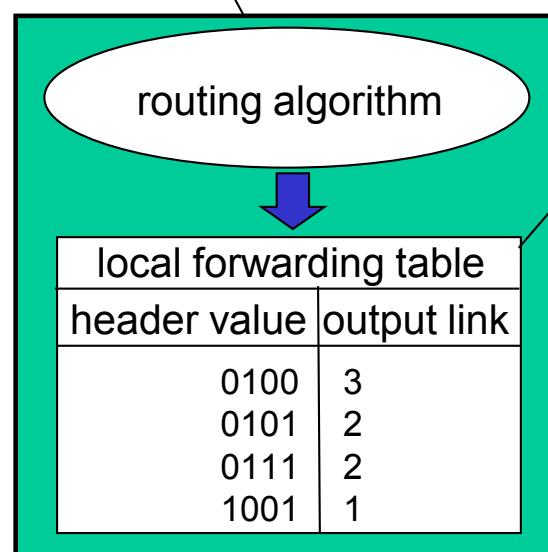


- Wenn die Ankunftsrate (in Bits) einer Verbindung dessen Übertragungsrate für einen gewissen Zeitraum übersteigt, dann werden
 - Pakete sich in die Router-Warteschlange einreihen und auf Übertragung warten und
 - Pakete unter Umständen verworfen (loss), wenn der Speicher (Puffer) voll ist.

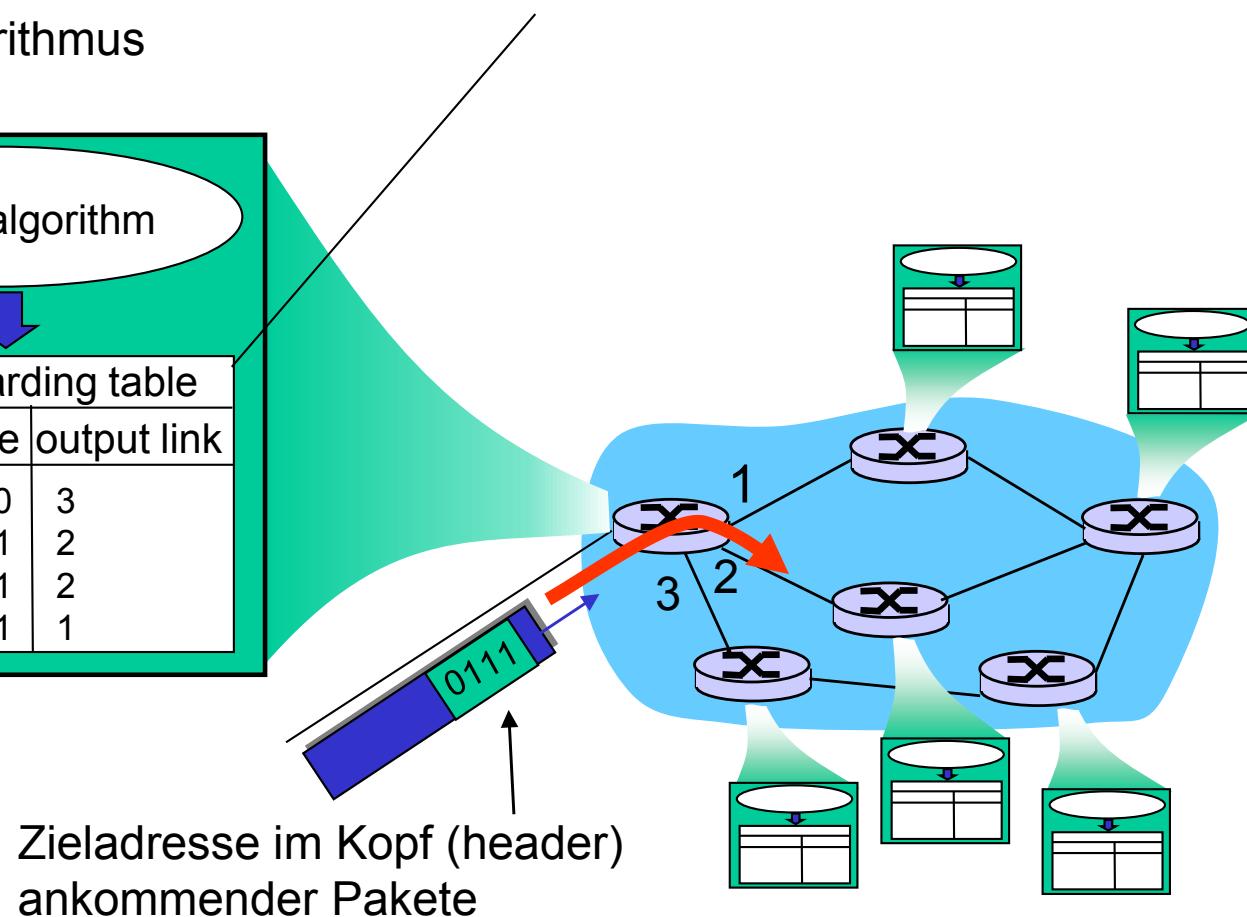
Hauptfunktionen des „Network Core“

Routing: passende Route von der Quelle zum Ziel für die Pakete ermitteln

- Routing-Algorithmus



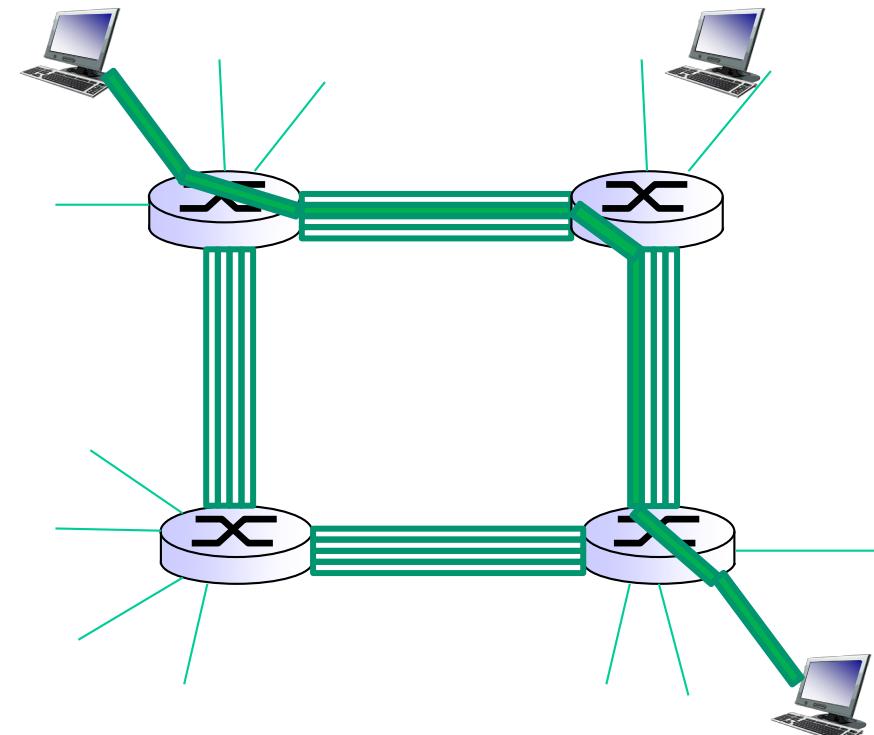
Forwarding: Pakete vom Router-Eingang zum passenden Ausgang bewegen



Alternative zur Paketvermittlung: Leitungsvermittlung

Ende-zu-Ende-Ressourcen werden für einen „Ruf“ reserviert

- Im Beispiel mit vier Einheiten („Leitungen“)
 - Der Anruf bekommt die 2. Einheit auf der oberen Verbindung und die 1. Einheit in der rechten Verbindung.
- Dedizierte Ressourcen: keine gemeinsame Nutzung
 - Garantierte Dienstgüte wie beim „Durchschalten“ einer physikalischen Verbindung
- Vor dem Austausch von Daten müssen die notwendigen Ressourcen reserviert werden
 - Einheiten werden Rufen zugewiesen
 - Einheiten bleiben ungenutzt, wenn sie von ihrem Ruf nicht verwendet werden (keine gemeinsame Nutzung von Ressourcen)
 - Traditionelle Telefonnetzwerke

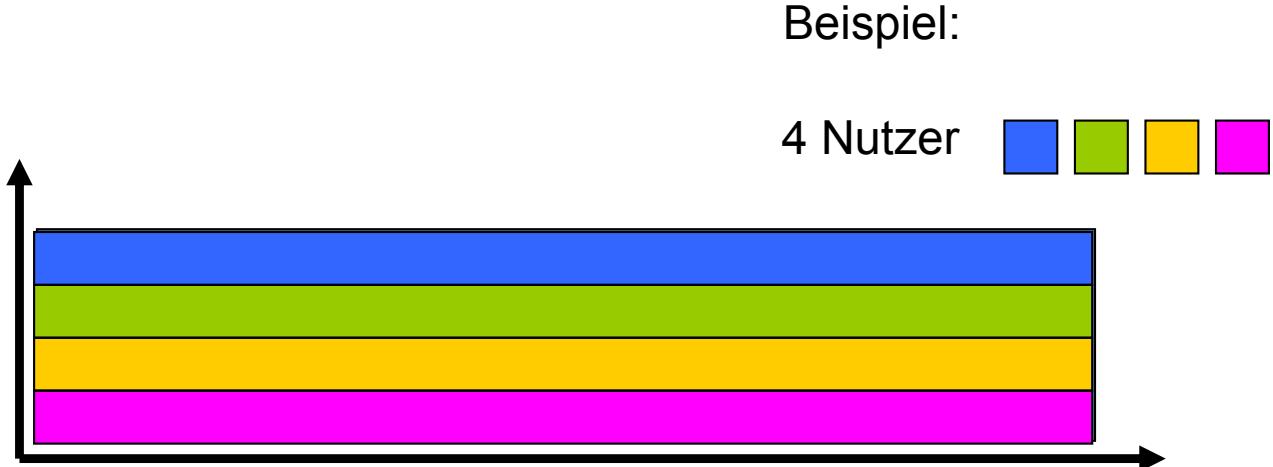


Alternative zur Paketvermittlung: Leitungsvermittlung

- Wie teilt man die Bandbreite einer Leitung in Einheiten auf?
 - Frequenzmultiplex (Frequency Division Multiplex, FDM)
 - Zeitmultiplex (Time Division Multiplex, TDM)

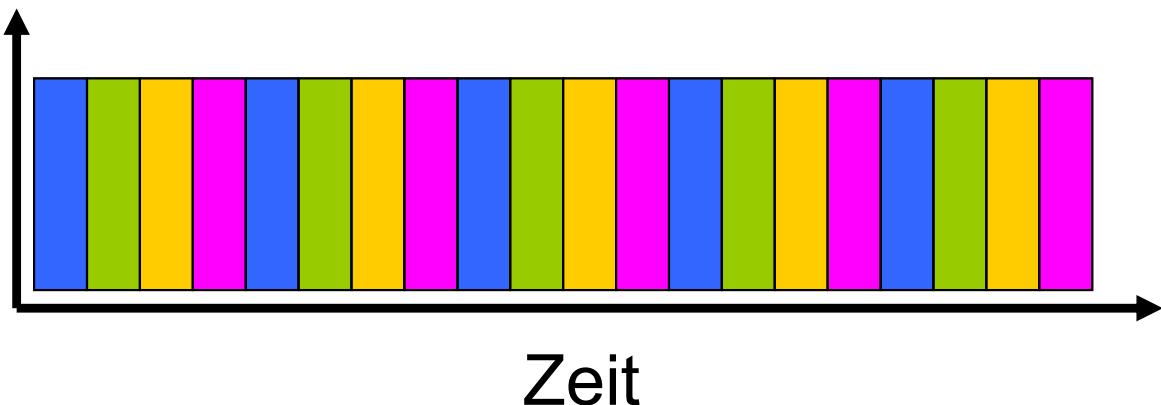
FDM

Frequenz

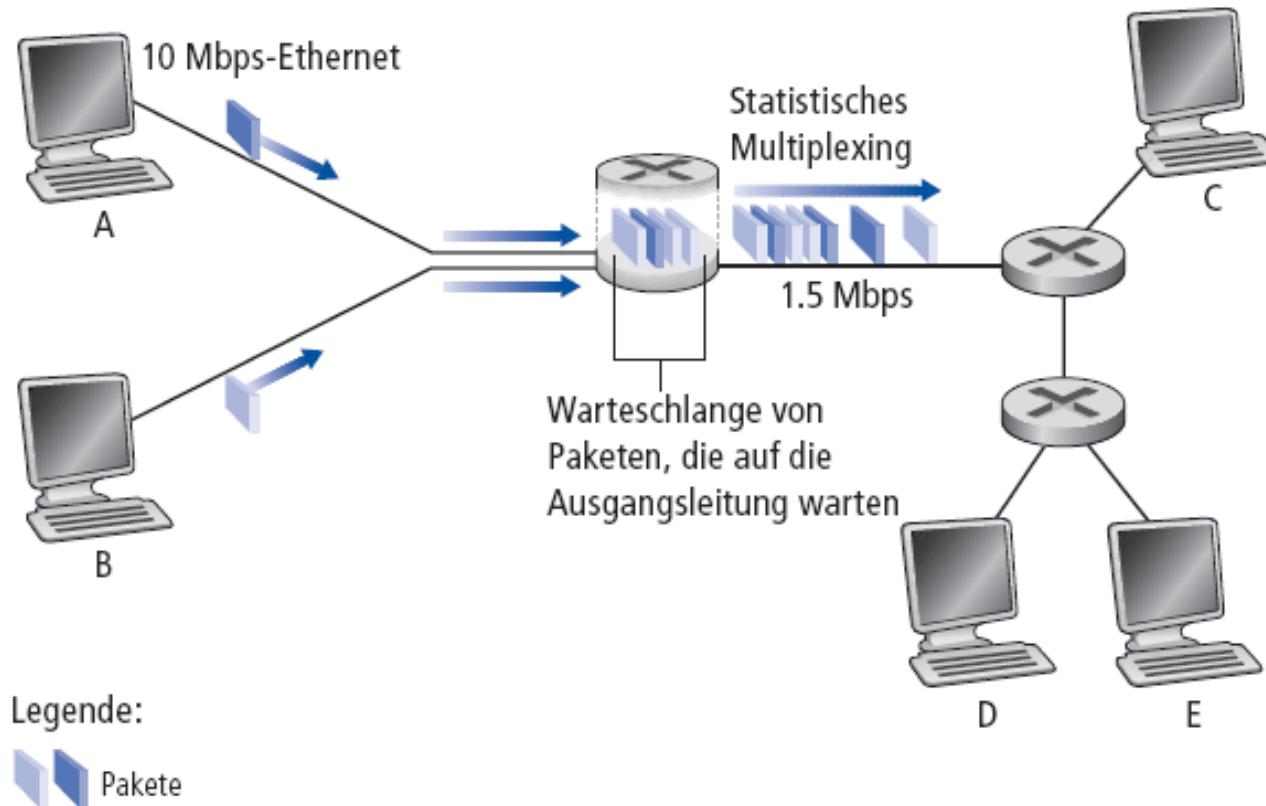


TDM

Frequenz



Paketvermittlung



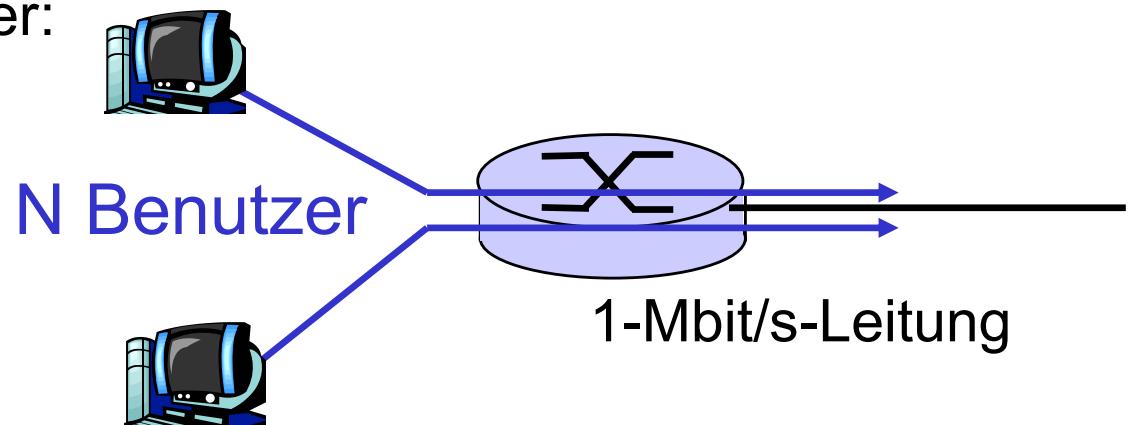
- Die Folge von Paketen auf der Leitung hat kein festes Muster, die Bandbreite wird nach Bedarf verteilt →

Paketvermittlung vs. Leitungsvermittlung

Paketvermittlung nutzt die Ressourcen häufig besser aus!

Beispiel:

- Leitung mit 1 Mbit/s
- Anforderungen der Benutzer:
 - 100 kbit/s, wenn aktiv
 - 10% der Zeit aktiv
- Leitungsvermittlung:
 - 10 Benutzer
- Paketvermittlung:
 - Mit 35 Benutzern ist die Wahrscheinlichkeit für mehr als 10 aktive Benutzer zur gleichen Zeit kleiner als 0,0004



Wie kommt man auf 0,0004?
Vgl. Kurose & Ross 2013, S. 31

Paketvermittlung vs. Leitungsvermittlung

Ist Paketvermittlung generell besser?

- Sehr gut für unregelmäßigen Verkehr (bursty traffic)
 - Gemeinsame Verwendung von Ressourcen
 - Einfacher, keine Reservierungen
-
- Wie kann man leitungsähnliches Verhalten bereitstellen?
 - Bandbreitengarantien werden gebraucht für Audio- und Videoanwendungen
 - Ungelöstes Problem

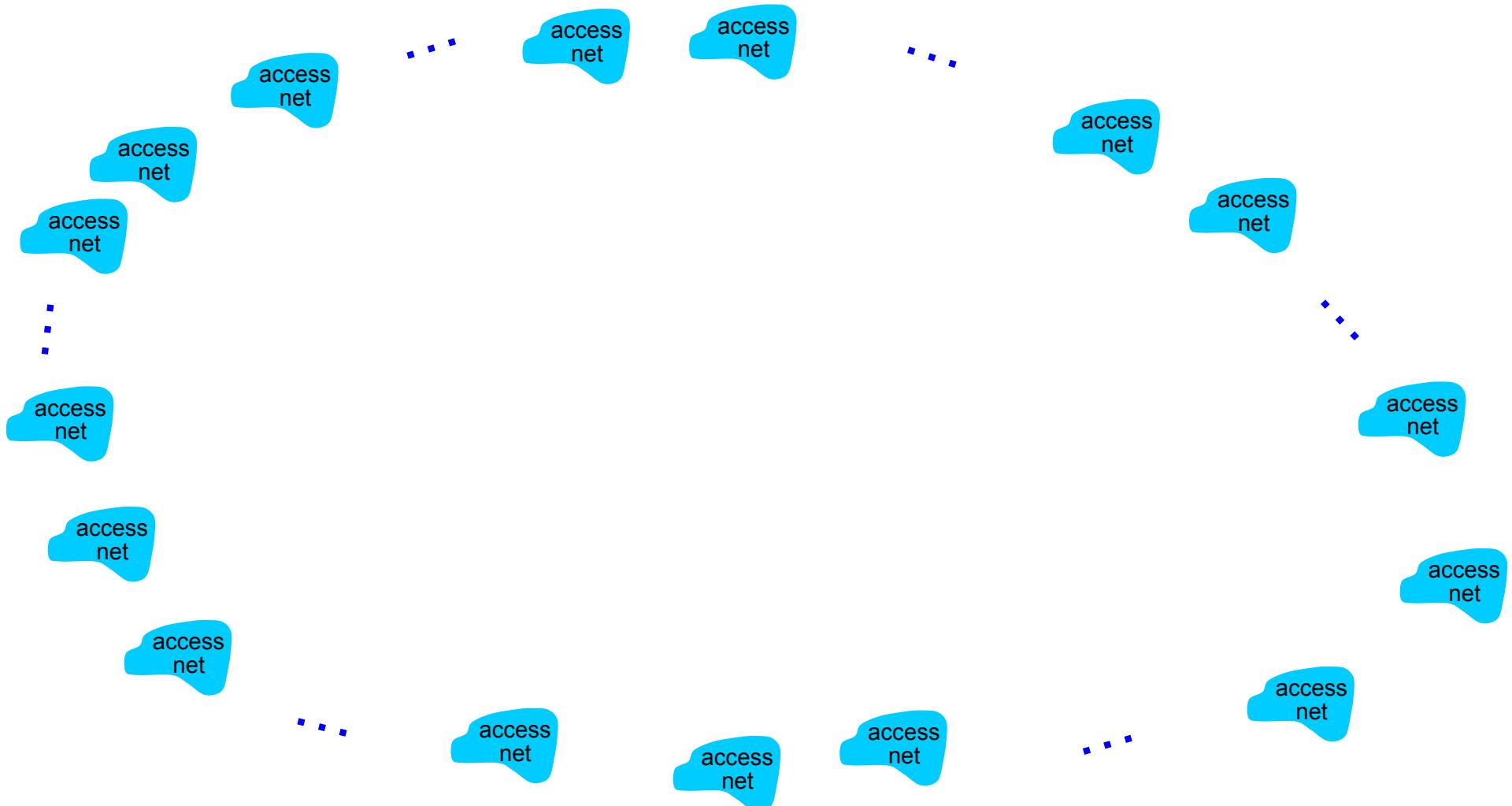
Internet Struktur: Netzwerk von Netzwerken

Wir haben schon gelernt:

- Das Internet basiert auf Paketvermittlung
- Endsysteme verbinden sich zum Internet mittels ISPs (Internet Service Providers)
 - Es gibt verschiedene Anbieter für Haushalte, Unternehmen, Universitäten etc.
- Die ISPs müssen ebenfalls untereinander verbunden sein
 - Damit zwei beliebige Endsysteme (Hosts) sich Pakete zusenden können
- Das resultierende Netzwerk ist sehr komplex und dessen Entwicklung wurde vielfach durch Politik und Wirtschaft beeinflusst.
- Wir schauen uns schrittweise an, wie die aktuelle Struktur des Internets aussieht.

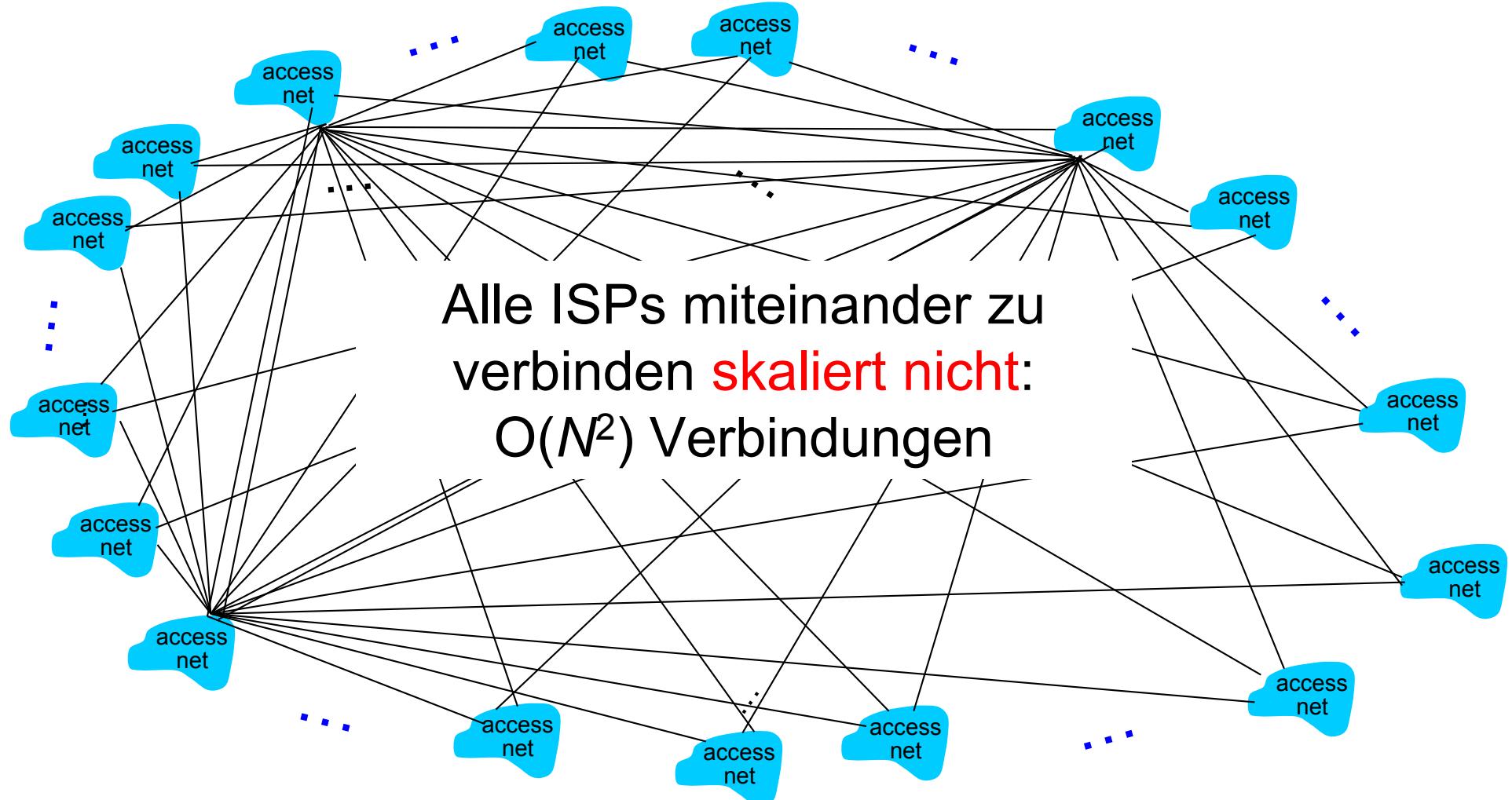
Internet Struktur: Netzwerk von Netzwerken

Wenn es Millionen von Zugangs-ISPs gibt, wie sollte man diese untereinander verbinden?



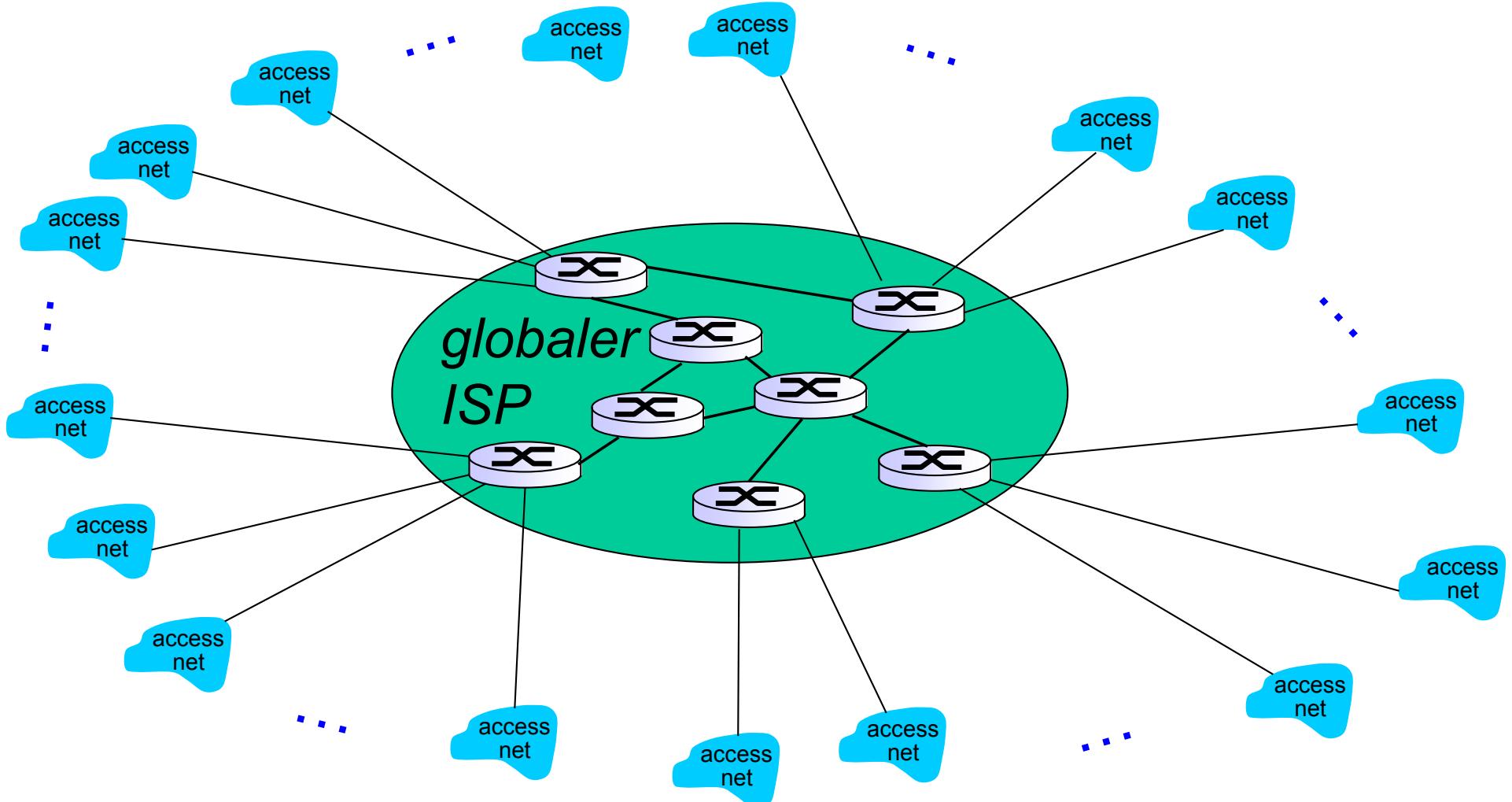
Internet Struktur: Netzwerk von Netzwerken

Option: jeder ISP wird mit jedem anderen verbunden



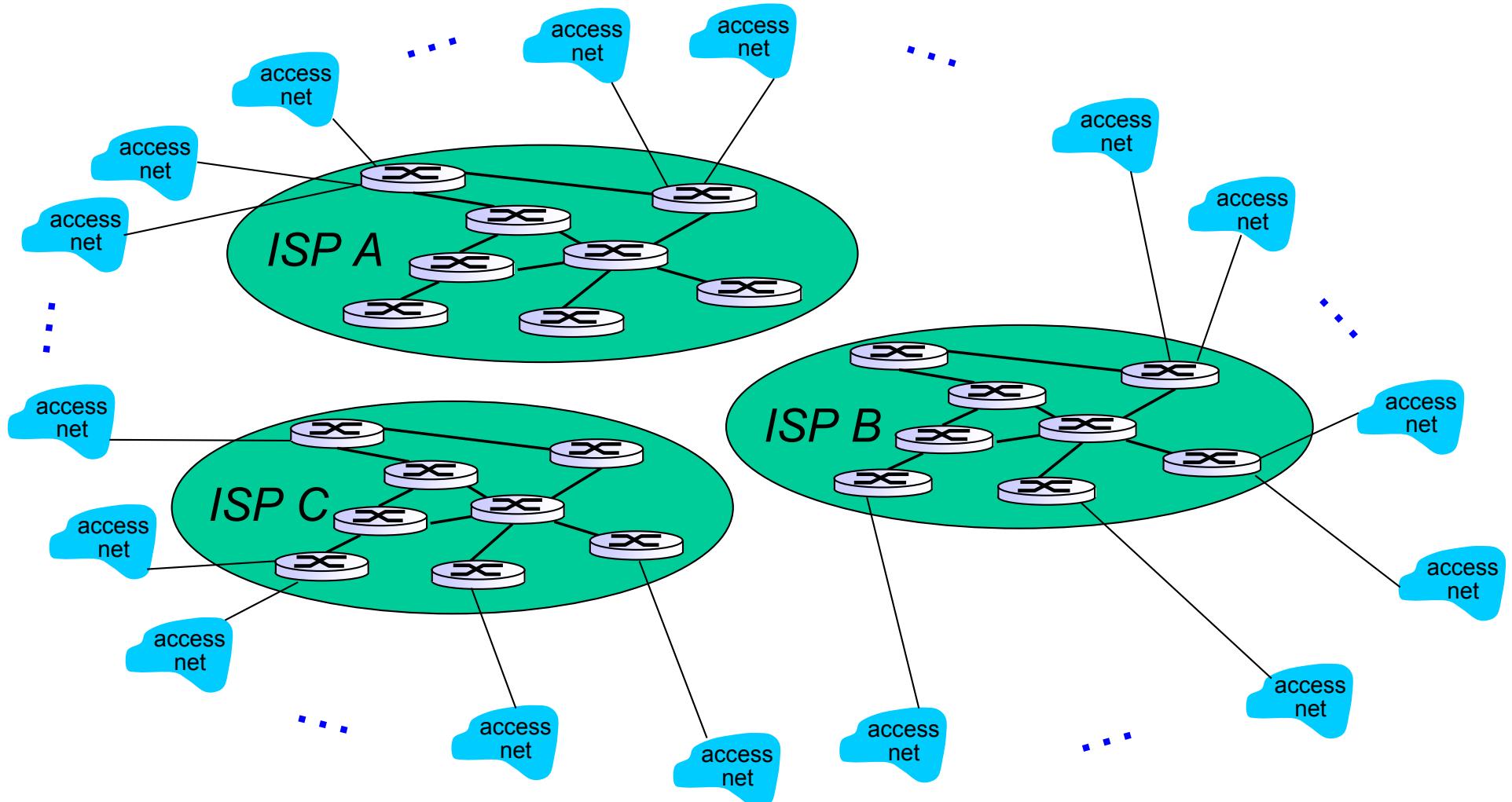
Internet Struktur: Netzwerk von Netzwerken

Option: jeder Zugangs-ISP wird mit einem globalen Transit-ISP verbunden; Zugangs-ISPs und globaler ISP sind wirtschaftlich/vertraglich einig.



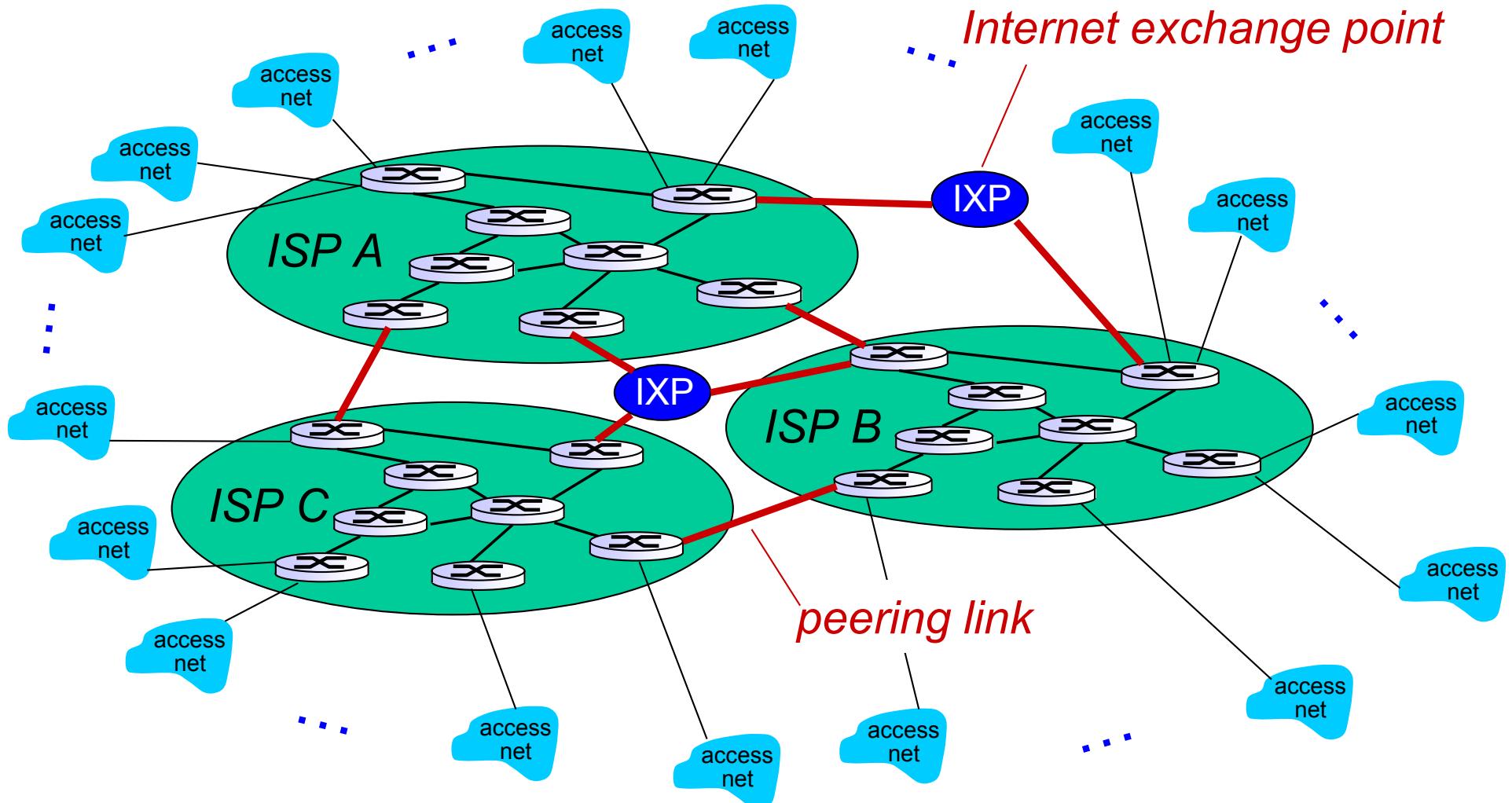
Internet Struktur: Netzwerk von Netzwerken

Aber: wenn das Angebot eines globalen ISPs gewinnbringend ist, wird es Wettbewerber geben...



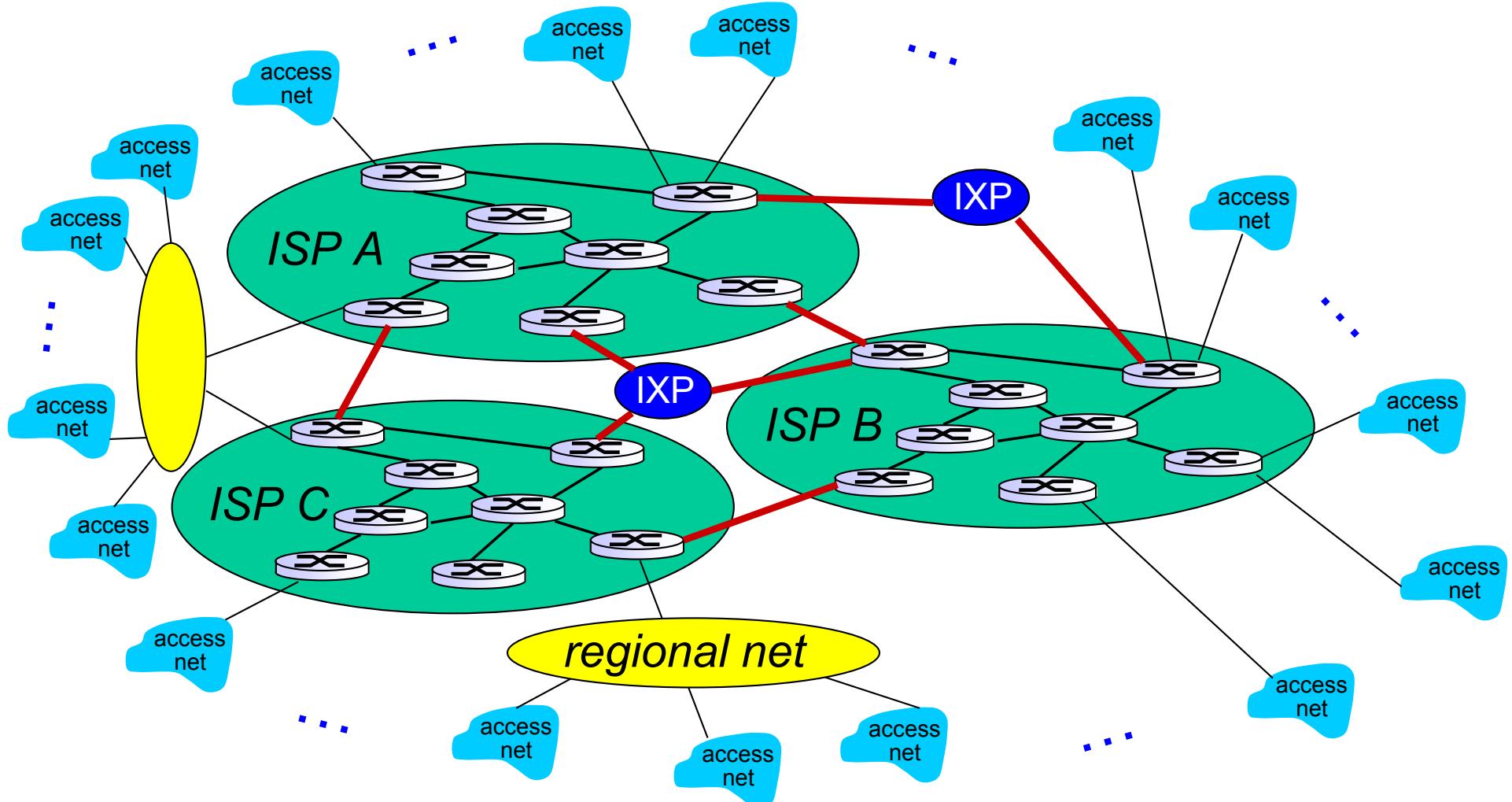
Internet Struktur: Netzwerk von Netzwerken

Aber: wenn das Angebot eines globalen ISPs gewinnbringend ist, wird es Wettbewerber geben... die verbunden werden müssen.



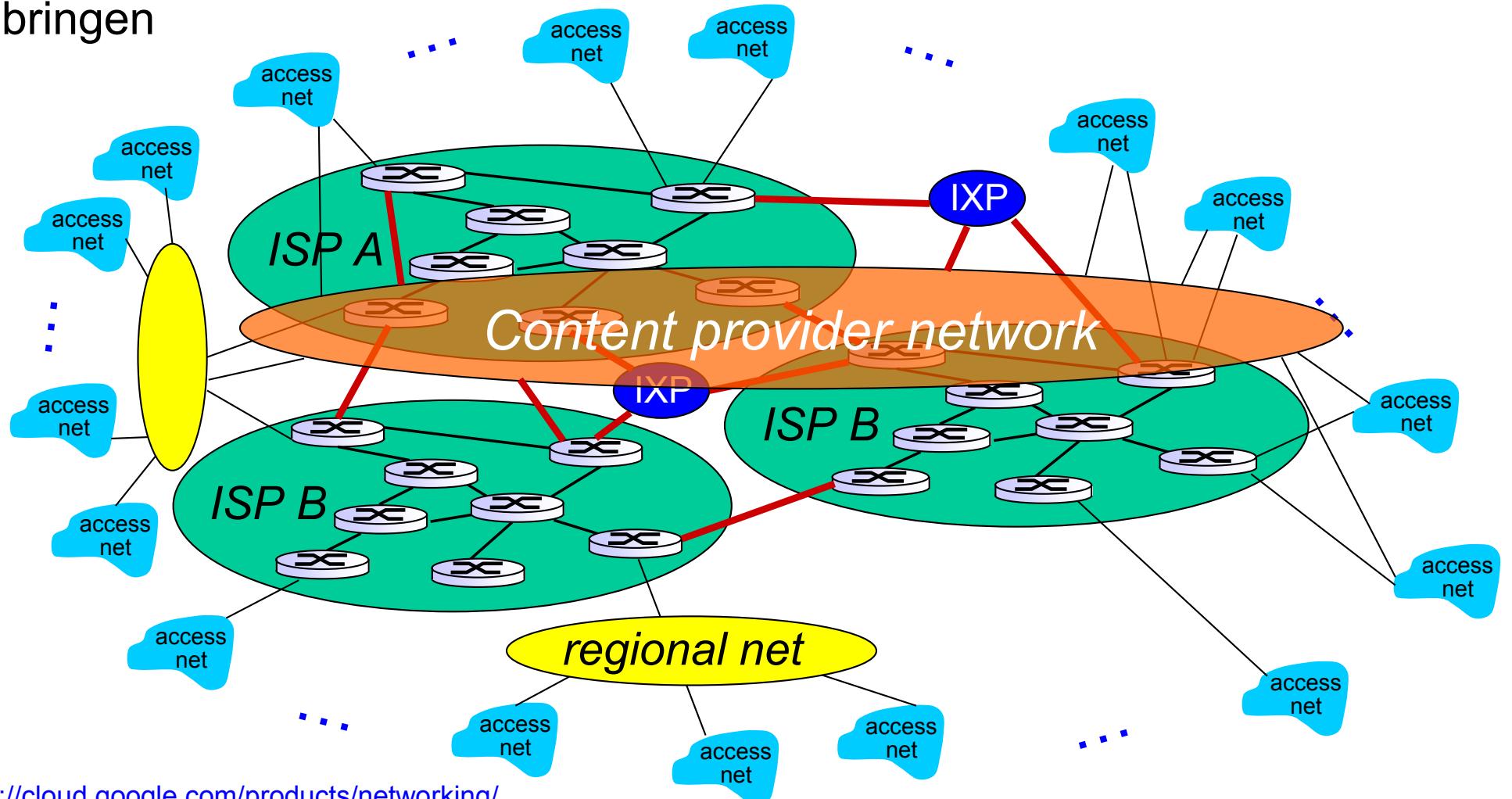
Internet Struktur: Netzwerk von Netzwerken

... und regionale Netzwerke können sich formen und dabei Zugangsnetzwerke bilden, die an die ISPs angeschlossen sind



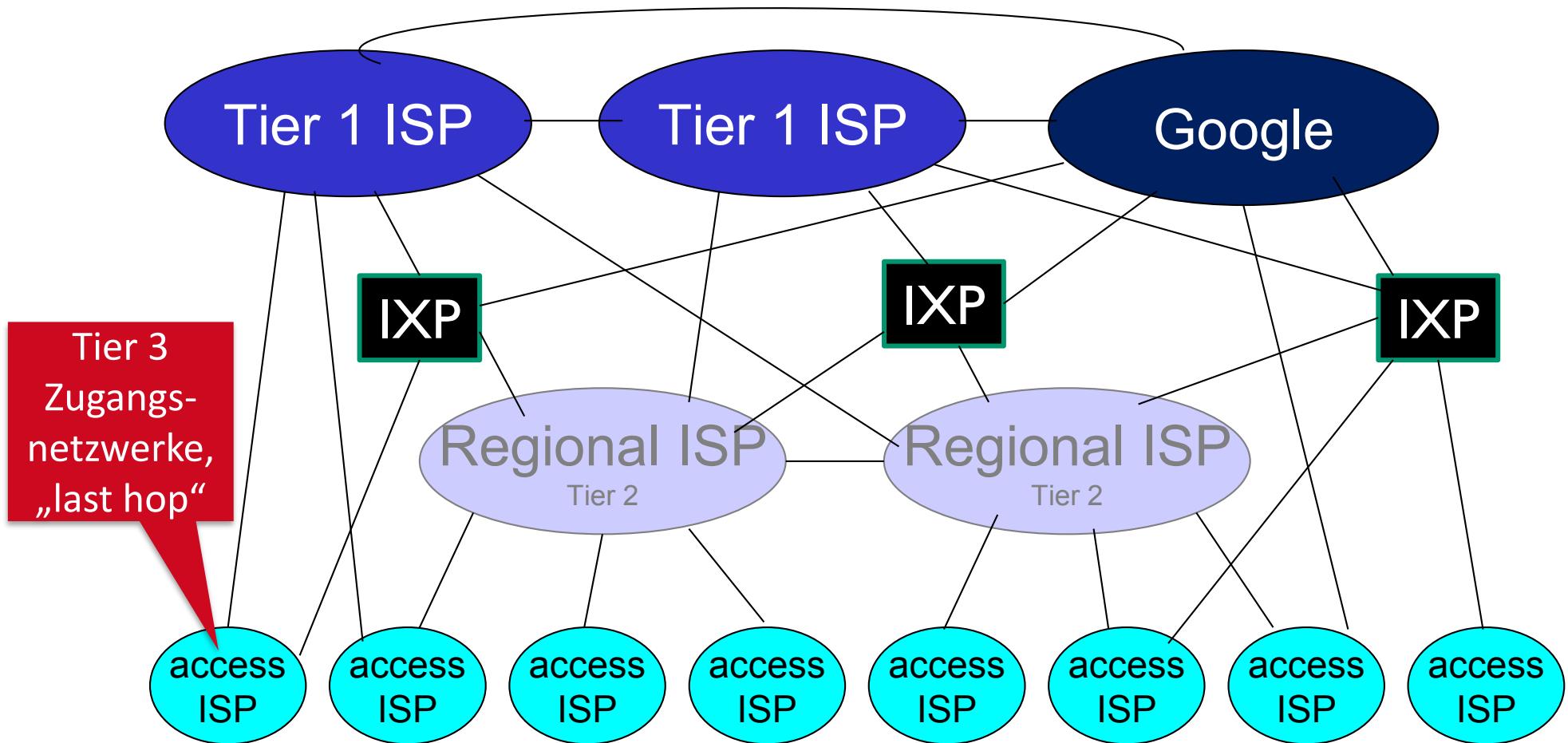
Internet Struktur: Netzwerk von Netzwerken

... und Anbieter von Inhalten (z.B. Google, Microsoft) können ihre eigenen Netzwerke betreiben, um ihre Inhalte nah an den Kunden zu bringen



<https://cloud.google.com/products/networking/>

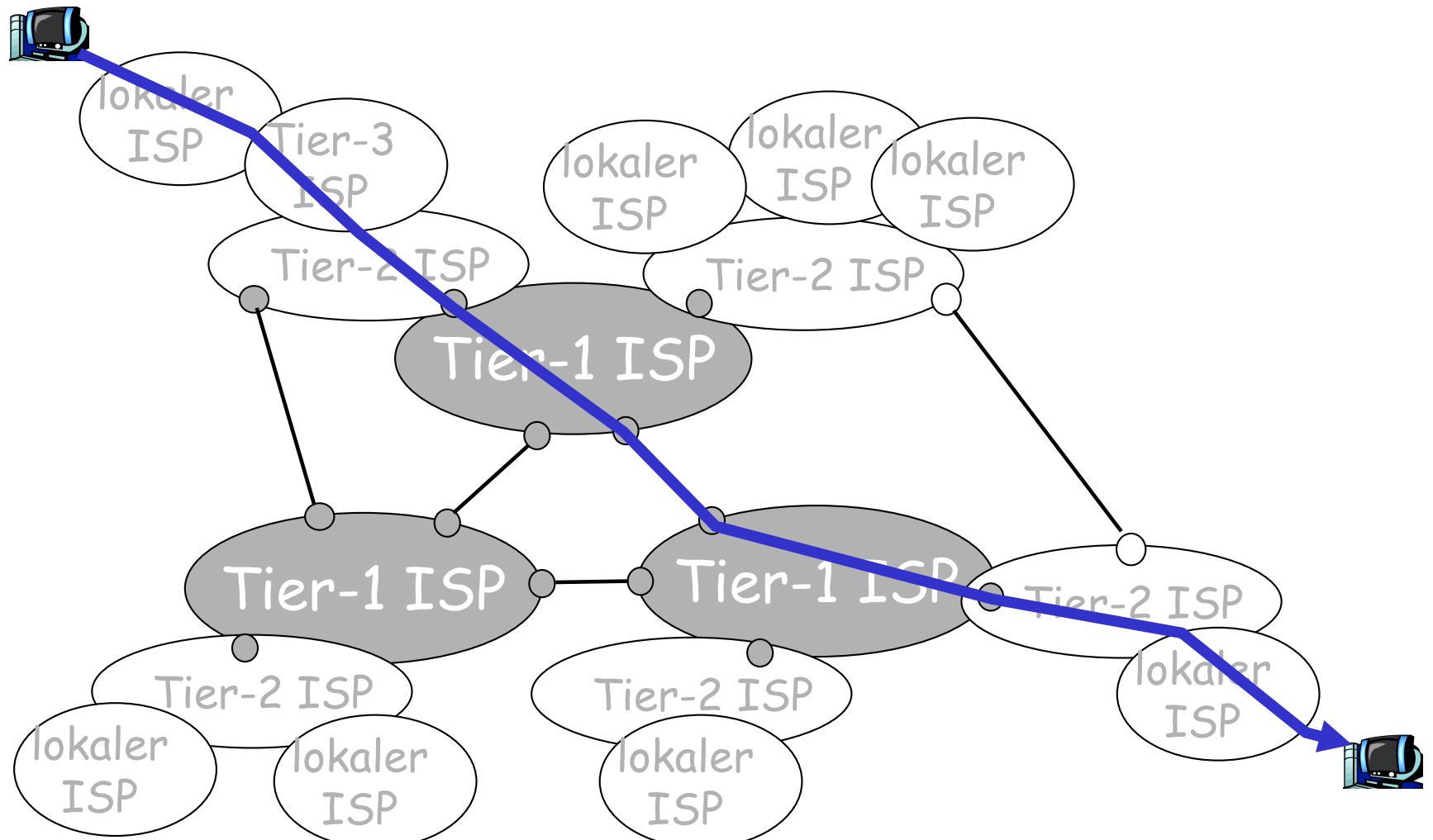
Internet Struktur: Netzwerk von Netzwerken



- Im Zentrum stehen wenige, gut verbundene, große Netzwerke
 - „Tier-1“ kommerzielle ISPs (z.B. Deutsche Telekom), national & international
 - Anbieternetzwerke (z.B. Google): private Netzwerke, umgehen oftmals ISPs
- „Tier-2“ ISPs: kleinere, oft nationale oder regionale ISPs

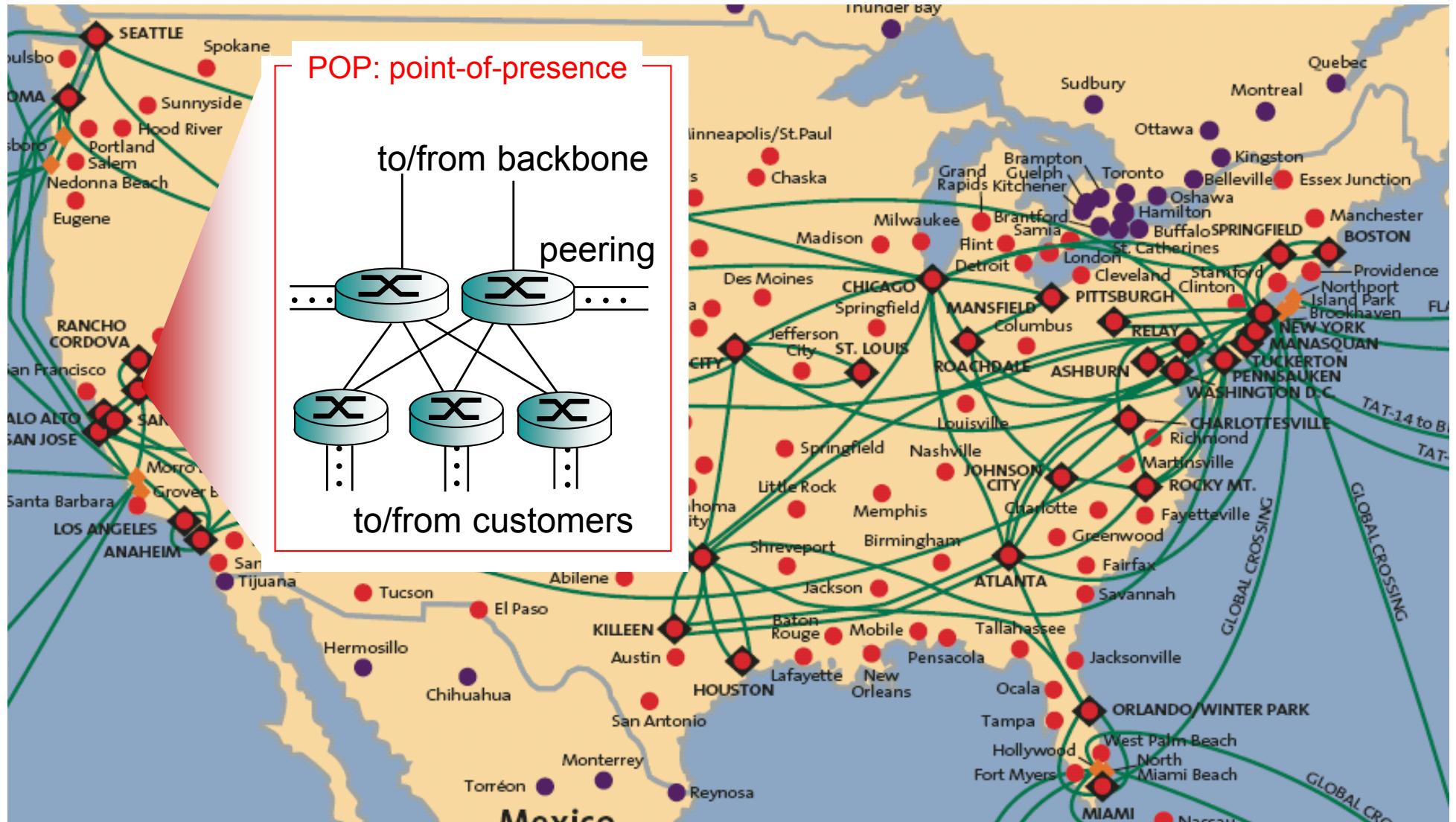
Internet Struktur: Netzwerk von Netzwerken

- Ein Paket durchquert viele Netzwerke



Internet Struktur: Netzwerk von Netzwerken

■ Tier-1 ISP: Beispiel Sprint

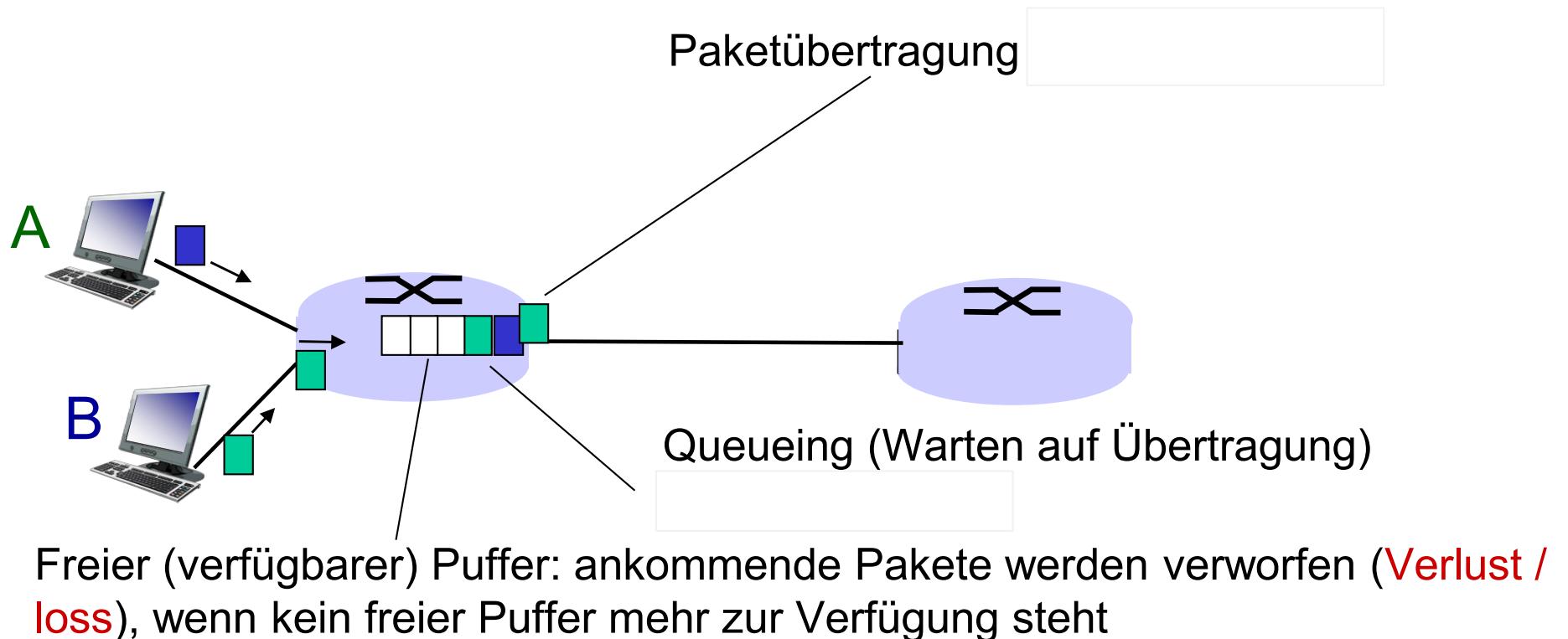


- Internet und Protokolle**
- Randbereich des Internets**
- Inneres des Internets**
- Netzwerkkenngrößen**
- Protokollsichten, Dienste und Netzwerksicherheit**
- Geschichte des Internets**
- Zusammenfassung und Ausblick**

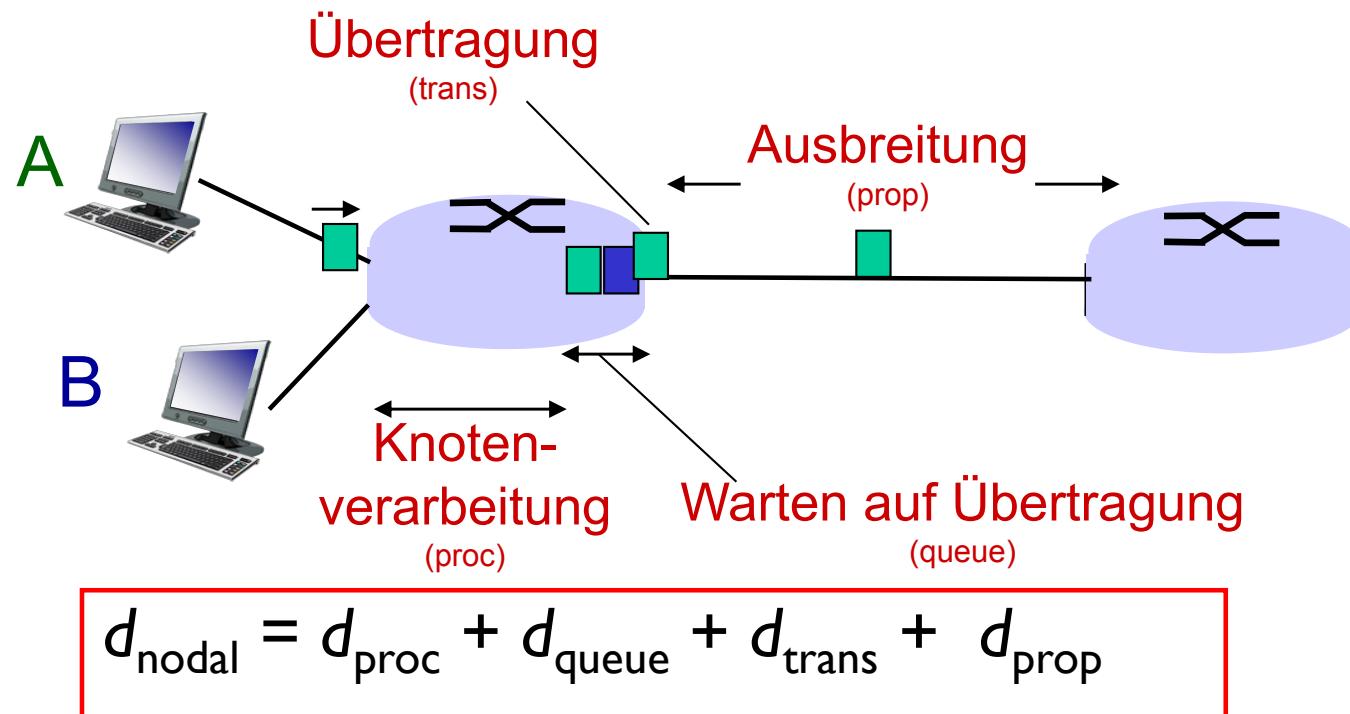
Paketverluste und Verzögerungen

Wie entstehen Paketverluste und Verzögerungen?

- Wenn die Ankunftsrate (temporär) die Kapazität der Ausgangsleitungen übersteigt
- warten Pakete in den Puffern von Routern auf den Versand.



Vier Quellen der Verzögerung



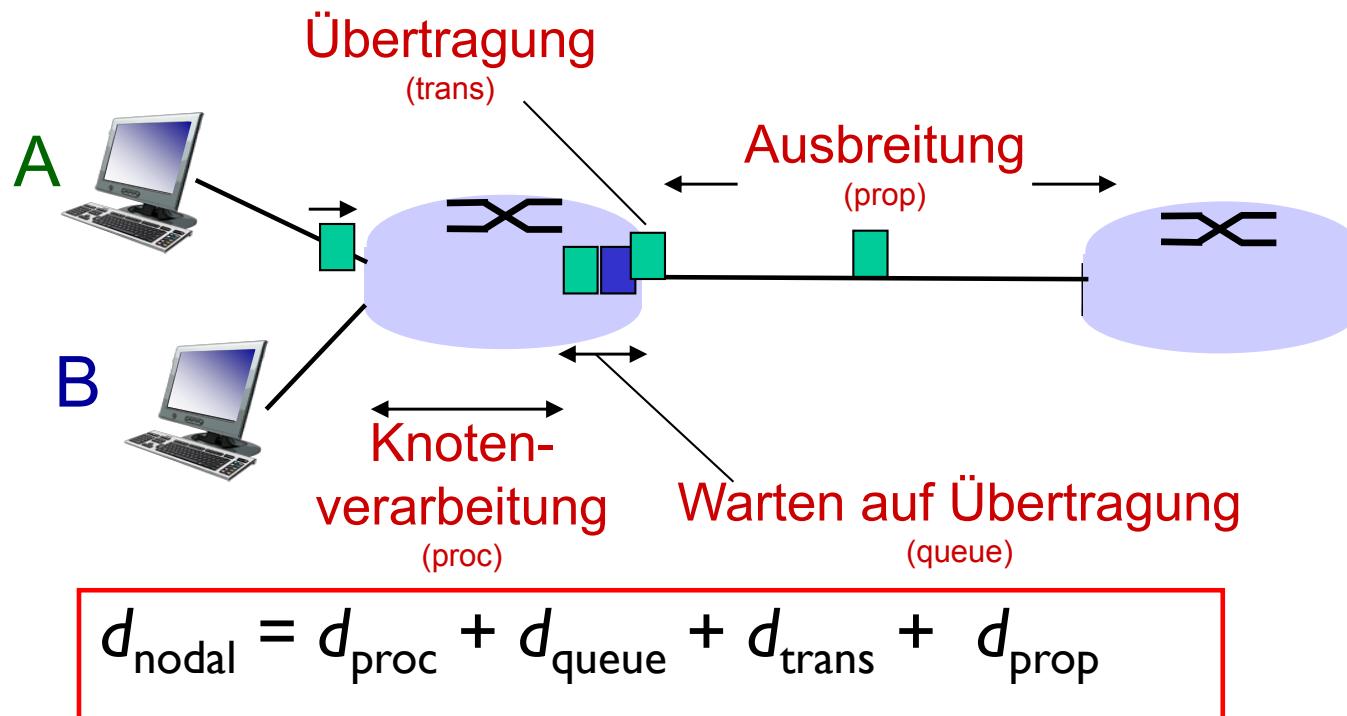
d_{proc} : processing on node

- Prüfung auf Bitfehler
- Wahl der ausgehenden Leitung
- typischerweise < ms

d_{queue} : queueing delay

- Wartezeit, bis das Paket auf die Ausgangsleitung gelegt werden kann
- Hängt von der Last auf der Ausgangsleitung ab

Vier Quellen der Verzögerung



d_{trans} : Übertragungsverzögerung

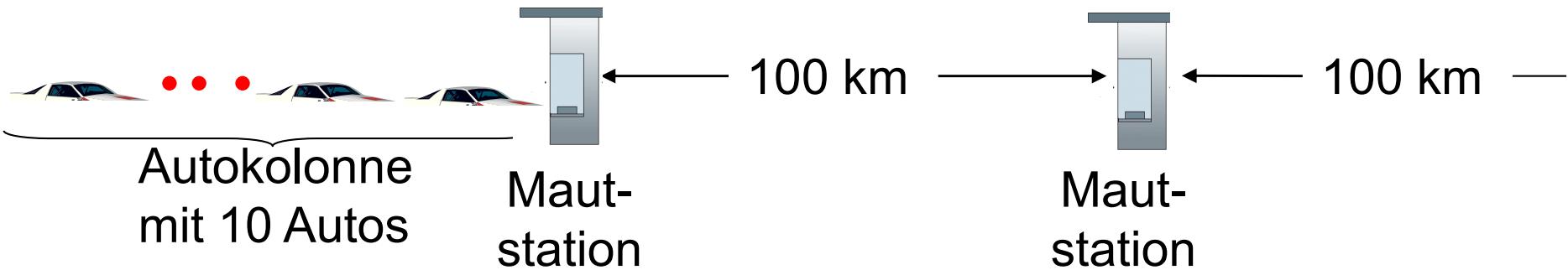
- R = Bandbreite einer Leitung (Bit/s)
- L = Paketgröße (Bit)
- $d_{\text{trans}} = L/R$

d_{trans} und d_{prop}
sind sehr unterschiedlich!

d_{prop} : Ausbreitungsverzögerung:

- d = Länge der Leitung
- s = Ausbreitungsgeschwindigkeit des Mediums ($\sim 2,8 \times 10^8$ m/s)
- $d_{\text{prop}} = d/s$

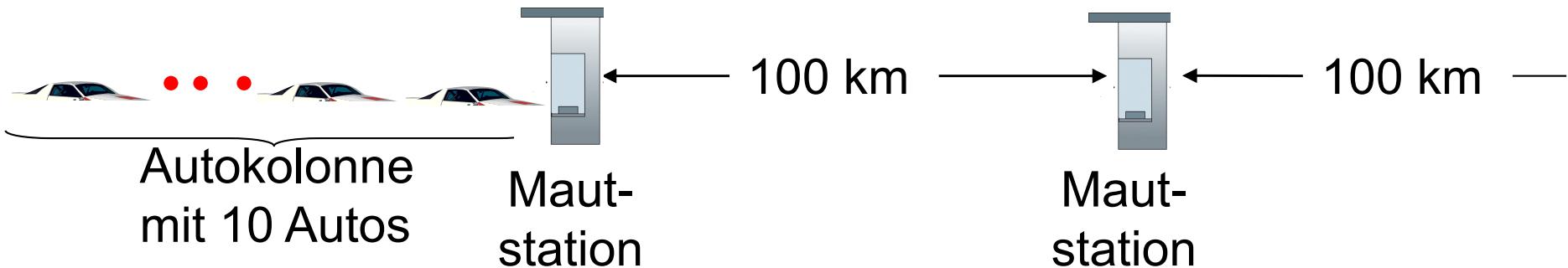
Transmission vs. Propagation: Analogie der Autokolonne



- Autos „breiten“ sich mit 100 km/h „aus“
- Mautstation benötigt 12 Sekunden zur Abfertigung (Übertragung) eines Autos
- Auto ~ Bit; Kolonne ~ Paket
- **Frage: Wie lange dauert es, bis das letzte Auto der Kolonne vor der zweiten Mautstation steht?**

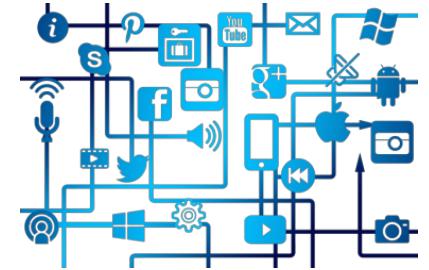
- Es dauert $10 \cdot 12 \text{ s} = 120 \text{ s}$, bis alle Autos die Mautstation passiert haben
- Das letzte Auto braucht dann noch $100 \text{ km} / (100 \text{ km/h}) = 1 \text{ h}$, bis es vor der zweiten Mautstation steht
-

Transmission vs. Propagation: Analogie der Autokolonne



- Autos „breiten“ sich nun mit 1000 km/h aus
 - Mautstation benötigt nun 60 s, um ein Auto abzufertigen
 - **Frage: Werden Autos der Kolonne vor der zweiten Mautstation ankommen, bevor alle anderen Autos die erste Station passiert haben?**
-
- Das erste Bit eines Paketes kann an einem Router ankommen, bevor das letzte Bit des Paketes den vorherigen Router verlassen hat!

Weiterführendes Material



- **Animation** zu „Transmission vs. Propagation Delay“
<http://www.ccs-labs.org/teaching/rn/animations/propagation/>
- Video zu „Transmission vs. Propagation Delay“
[http://mediaplayer.pearsoncmg.com/ph_cc_ecs_set.title.Propagation_Delay_and_Transmission_Delay_\(Chapter_01\)_/aw/streaming/ecs_kurose_compnetw_6/PropAndTransmit.m4v](http://mediaplayer.pearsoncmg.com/ph_cc_ecs_set.title.Propagation_Delay_and_Transmission_Delay_(Chapter_01)_/aw/streaming/ecs_kurose_compnetw_6/PropAndTransmit.m4v),
- https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/transmission-vs-propogation-delay/transmission-propagation-delay-ch1/index.html

Wartezeiten in Puffern

- R = Bandbreite (Bit/s)
- L = Paketgröße (Bit)
- a = durchschnittliche Paketankunftsrate

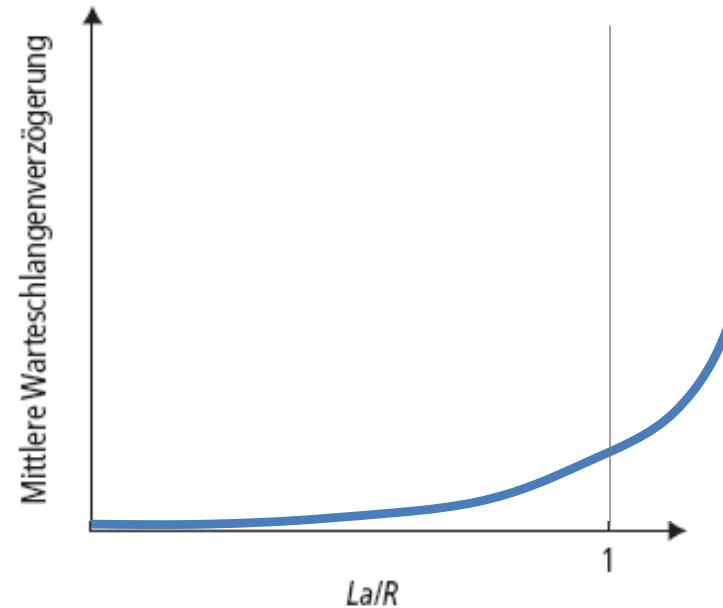
Verkehrswert = La/R

- $La/R \sim 0$: Wartezeit gering
- $La/R \sim 1$: Wartezeit steigt an
- $La/R > 1$: durchschnittliche Wartezeit ist unendlich (wenn über lange Zeit >1)!



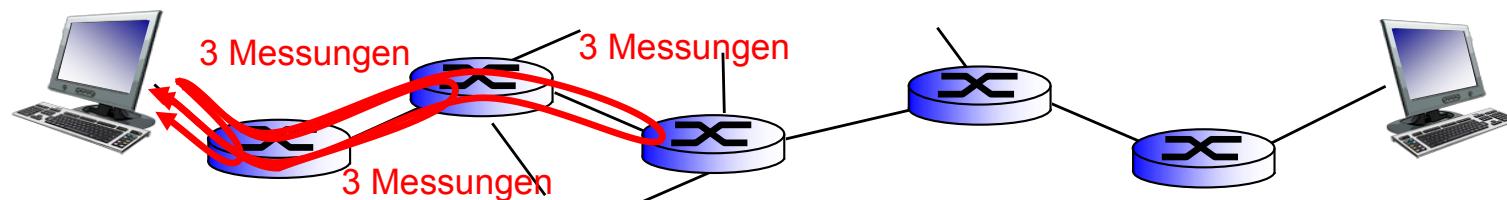
Animation (Queue)

<http://www.ccs-labs.org/teaching/rn/animations/queue/>



„Echte“ Verzögerungen im Internet

- Wie sehen echte Verzögerungen im Internet aus?
- traceroute: Misst die Verzögerung von einer Quelle zu allen Routern auf dem Weg zu einem Ziel.
- Für alle Router i:
 - Sender sendet je drei Pakete an die Router i, die auf dem Pfad zum Sender liegen
 - Router i schickt als Reaktion Pakete an den Sender
 - Sender misst die Zeit zwischen Senden des eigenen Paketes und Empfang des Paketes vom Router (round trip time, RTT)





Übung

- Öffnen Sie die Eingabeaufforderung in Windows (Windows-Taste drücken, cmd eingeben, Enter drücken)
- Tippen Sie ein: tracert www.hs-lu.de
 - Wie viele „hops“ werden benötigt, um von Ihrem Rechner bis zum Webserver der Hochschule zu gelangen?
 - Wie ist die durchschnittliche Dauer von Paketen für den Hin- und Rückweg bis zum Webserver?
- Tippen Sie ein: tracert gaia.cs.umass.de
 - Wie viele „hops“ werden diesmal benötigt?
 - Wie ist die durchschnittliche Dauer von Paketen für den Hin- und Rückweg innerhalb Europas?
 - Wie ist die durchschnittliche Dauer von Paketen für den Hin- und Rückweg in den USA?
 - Ab welchem Router steigt die RTT deutlich?

„Echte“ Verzögerungen im Internet

Aus Heimnetz

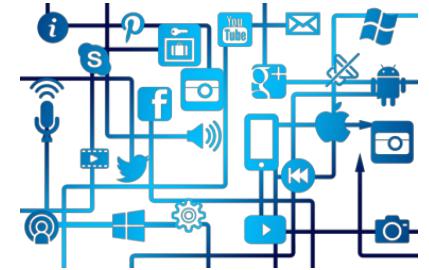
traceroute gaia.cs.umass.edu (Windows: tracert)

```
Routenverfolgung zu gaia.cs.umass.edu [128.119.245.12]
über maximal 30 Hops:
1  <1 ms    <1 ms    <1 ms    fritz.box [192.168.178.1]
2  9 ms     8 ms     9 ms     HSI-KBW-091-089-194-001.hsi2.kabel-badenwuerttemberg.de [91.89.194.1]
3  9 ms     8 ms     9 ms     172.30.21.25
4  122 ms   108 ms   114 ms   de-fra04a-rc1-ae48-0.aorta.net [84.116.191.33]
5  *         *         *       Zeitüberschreitung der Anforderung.
6  100 ms   93 ms    92 ms    us-nyc01b-rd2-ae9-0.aorta.net [84.116.140.170]
7  91 ms    95 ms    91 ms    us-nyc01b-ri2-ae3-0.aorta.net [84.116.137.194]
8  96 ms    92 ms    92 ms    lag-5.ear3.NewYork1.Level3.net [4.68.72.9]
9  117 ms   116 ms   117 ms   UNIVERSITY.ear3.NewYork1.Level3.net [4.71.230.234]
10 117 ms   118 ms   124 ms   core2-rt-xe-0-0-0.gw.umass.edu [192.80.83.105]
11 117 ms   118 ms   117 ms   lgrc-rt-106-8-po20.gw.umass.edu [128.119.0.109]
12 117 ms   117 ms   116 ms   128.119.3.32
13 107 ms   110 ms   108 ms   nscls1bbs1.cs.umass.edu [128.119.240.253]
14 119 ms   116 ms   116 ms   gaia.cs.umass.edu [128.119.245.12]
```

Aus HS-LU Netz

```
Routenverfolgung zu gaia.cs.umass.edu [128.119.245.12]
über maximal 30 Hops:
1  20 ms    17 ms    13 ms    VPN-2 [172.28.16.1]
2  *         *         *       Zeitüberschreitung der Anforderung.
3  15 ms    16 ms    15 ms    143.93.200.1
4  13 ms    14 ms    15 ms    elea.fh-ludwigshafen.de [143.93.192.1]
5  18 ms    18 ms    30 ms    s-fh-lu-2.rlp-net.net [217.198.243.97]
6  16 ms    16 ms    15 ms    g-decix-1.rlp-net.net [217.198.247.65]
7  15 ms    16 ms    17 ms    g-telecity-1.rlp-net.net [217.198.240.6]
8  15 ms    16 ms    17 ms    te0-1-1-1.rcr21.b015749-1.fra03.atlas.cogentco.com [149.14.156.137]
9  30 ms    20 ms    19 ms    be2502.ccr42.fra03.atlas.cogentco.com [154.54.39.181]
10 22 ms    22 ms    26 ms    be2814.ccr42.ams03.atlas.cogentco.com [130.117.0.141]
11 48 ms    33 ms    34 ms    be12488.ccr42.lon13.atlas.cogentco.com [130.117.51.41]
12 92 ms    93 ms    93 ms    be2983.ccr32.bos01.atlas.cogentco.com [154.54.1.178]
13 95 ms    95 ms    93 ms    te0-0-2-0.rcr12.orh01.atlas.cogentco.com [154.54.6.26]
14 95 ms    95 ms    95 ms    38.104.218.14
15 100 ms   99 ms    99 ms    69.16.1.0
16 99 ms    100 ms   100 ms   core2-rt-xe-0-1-0.gw.umass.edu [192.80.83.113]
17 100 ms   99 ms    99 ms    lgrc-rt-106-8-po20.gw.umass.edu [128.119.0.109]
18 109 ms   99 ms    100 ms   128.119.3.32
19 106 ms   101 ms   100 ms   nscls1bbs1.cs.umass.edu [128.119.240.253]
20 100 ms   101 ms   99 ms    gaia.cs.umass.edu [128.119.245.12]
```

Weiterführendes Material



- **Web-basiertes traceroute**

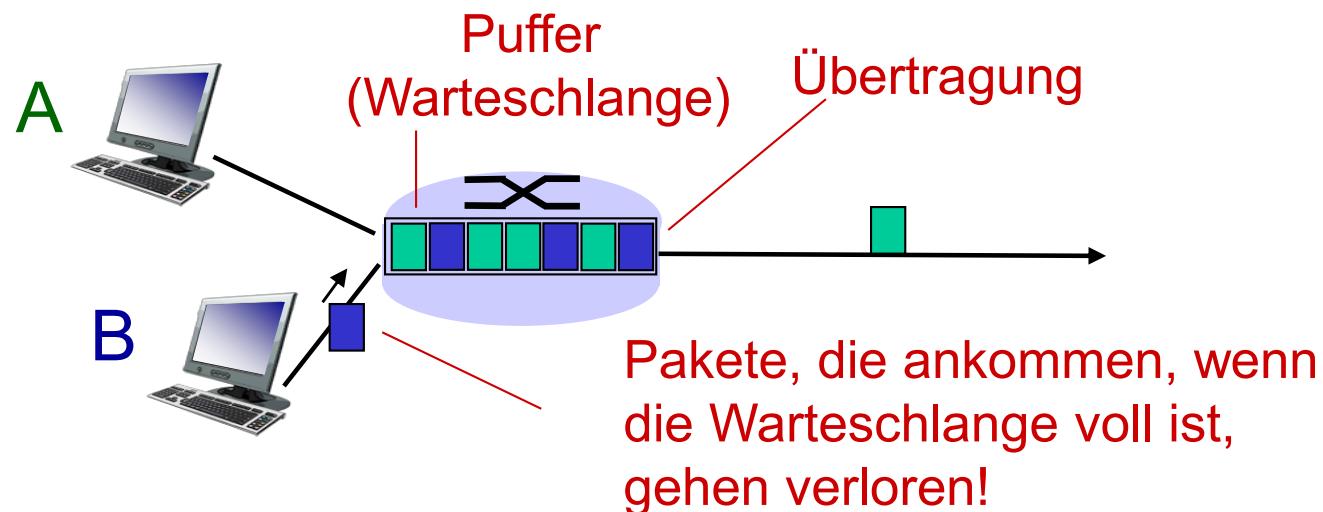
<http://www.dnstools.ch/visual-traceroute.html>

- Video zu traceroute von Kurose & Ross

http://mediaplayer.pearsoncmg.com/_ph_cc_ecs_set.title.Traceroute_aw/streaming/ecs_kurose_comnetw_6/traceroute_videonote.m4v

Paketverlust

- Warteschlange vor einer Ausgangsleitung hat endlich viele Pufferplätze
- Wenn alle Pufferplätze belegt sind, werden neu ankommende Pakete verworfen: **Paketverlust**
- Verlorene Pakete können vom vorangegangenen Knoten oder vom Sender erneut übertragen werden – oder auch gar nicht!

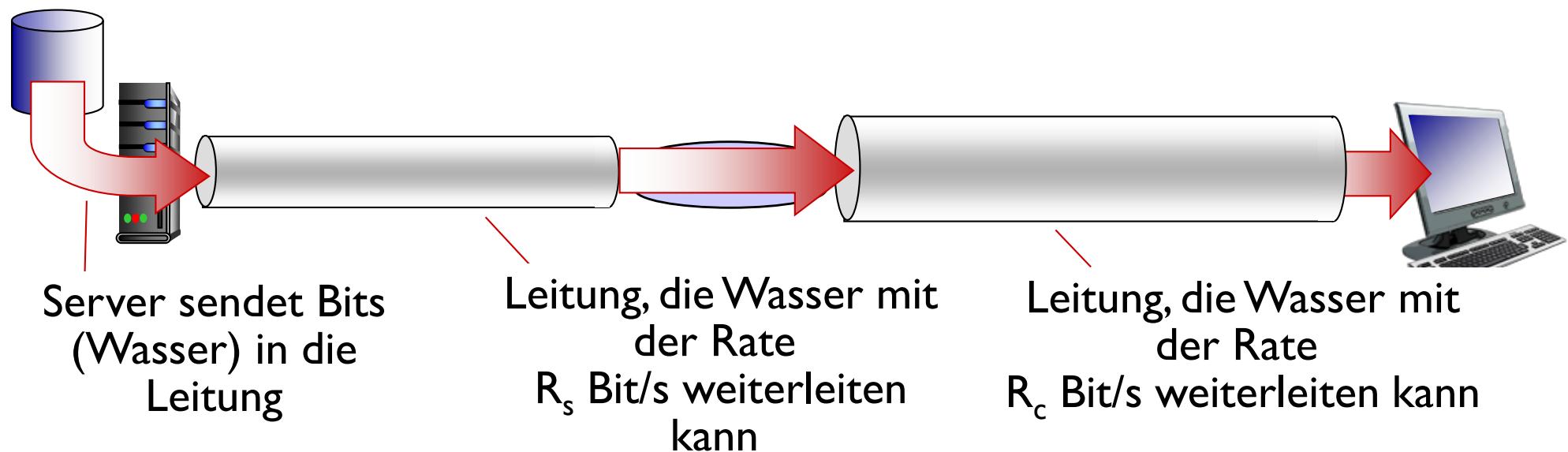


Animation (Loss): <http://www.ccs-labs.org/teaching/rn/animations/queue/>

Durchsatz

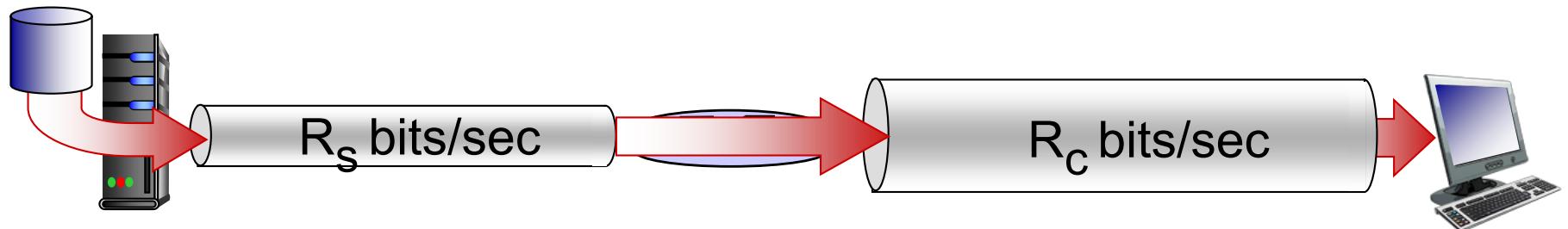
- **Durchsatz:** Rate (Bit/Zeiteinheit), mit der Daten zwischen Sender und Empfänger ausgetauscht werden

-
-

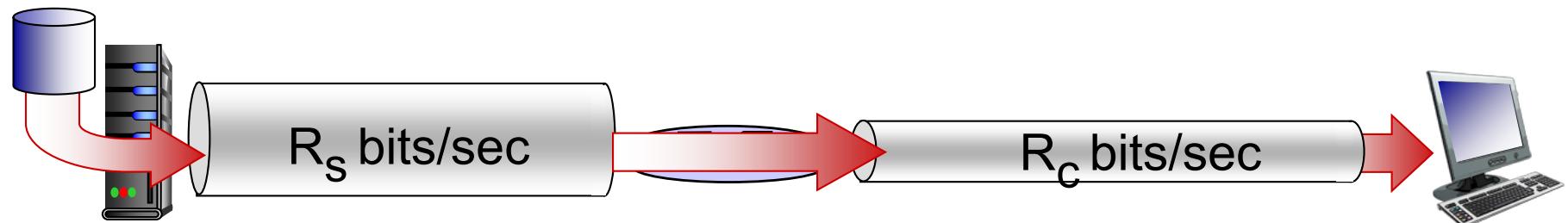


Durchsatz

- $R_s < R_c$ Was ist der durchschnittliche Ende-zu-Ende-Durchsatz?

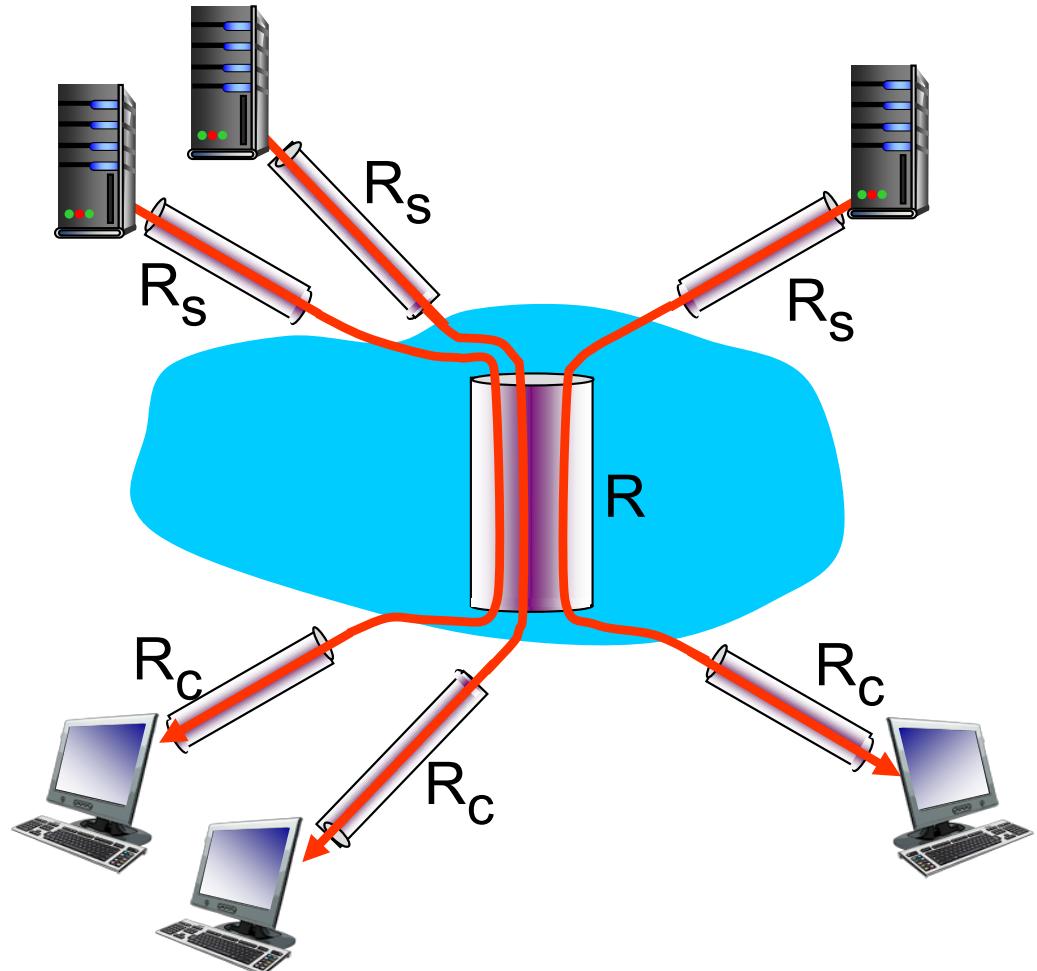


- $R_s > R_c$ Was ist der durchschnittliche Ende-zu-Ende-Durchsatz?



Durchsatz im Internet

- Durchsatz für Ende-zu-Ende-Verbindungen:
 $\min(R_c, R_s, R/10)$
Im Beispielszenario
- In der Realität: Häufig sind R_c oder R_s die Engpässe



Szenario: 10 Verbindungen teilen sich den Engpass des Backbone-Netzwerkes (R Bit/s)

- Internet und Protokolle**
- Randbereich des Internets**
- Inneres des Internets**
- Netzwerkkenngrößen**
- Protokollsichten, Dienste und Netzwerksicherheit**
- Geschichte des Internets**
- Zusammenfassung und Ausblick**

Protokollsichten

Netzwerke sind komplex!



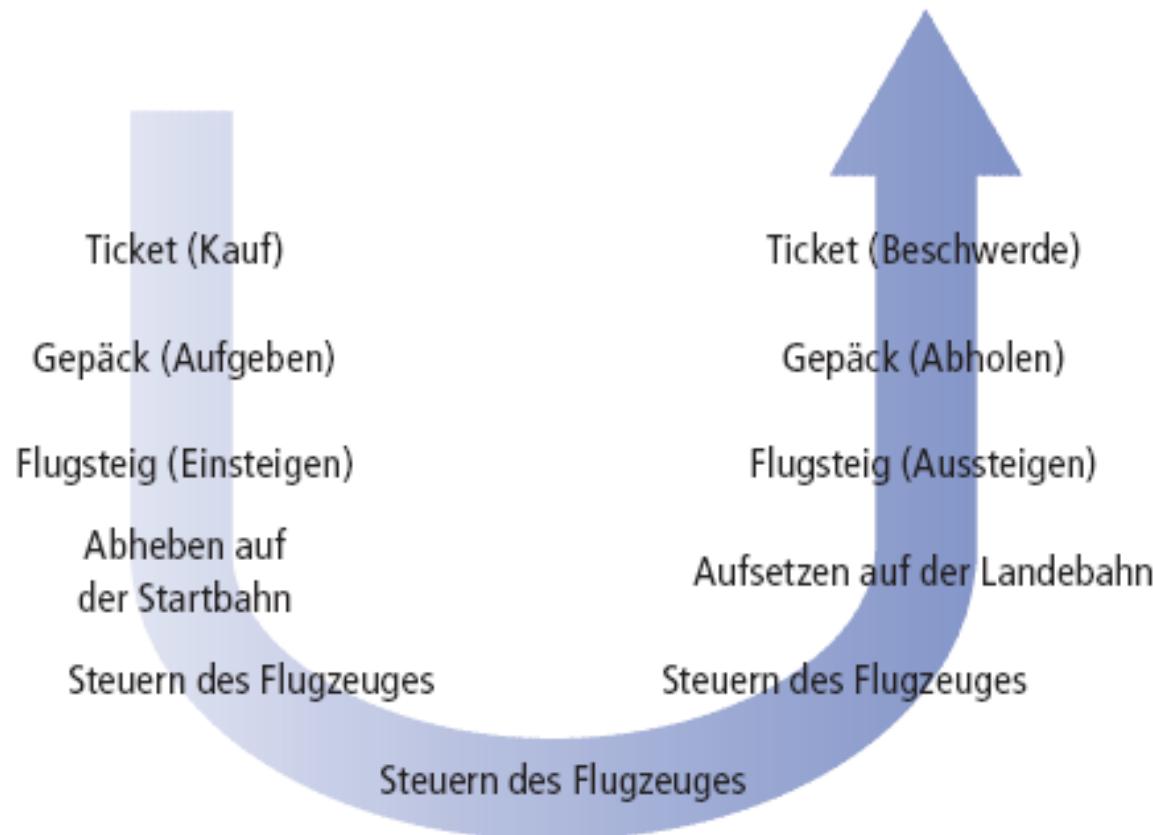
- Viele Elemente:
 - Hosts
 - Router
 - Leitungen auf Basis verschiedener Medien
 - Anwendungen
 - Protokolle
 - Hardware, Software

Daher stellt sich folgende Frage:

Kann die Funktionalität von Netzwerken vernünftig strukturiert werden?

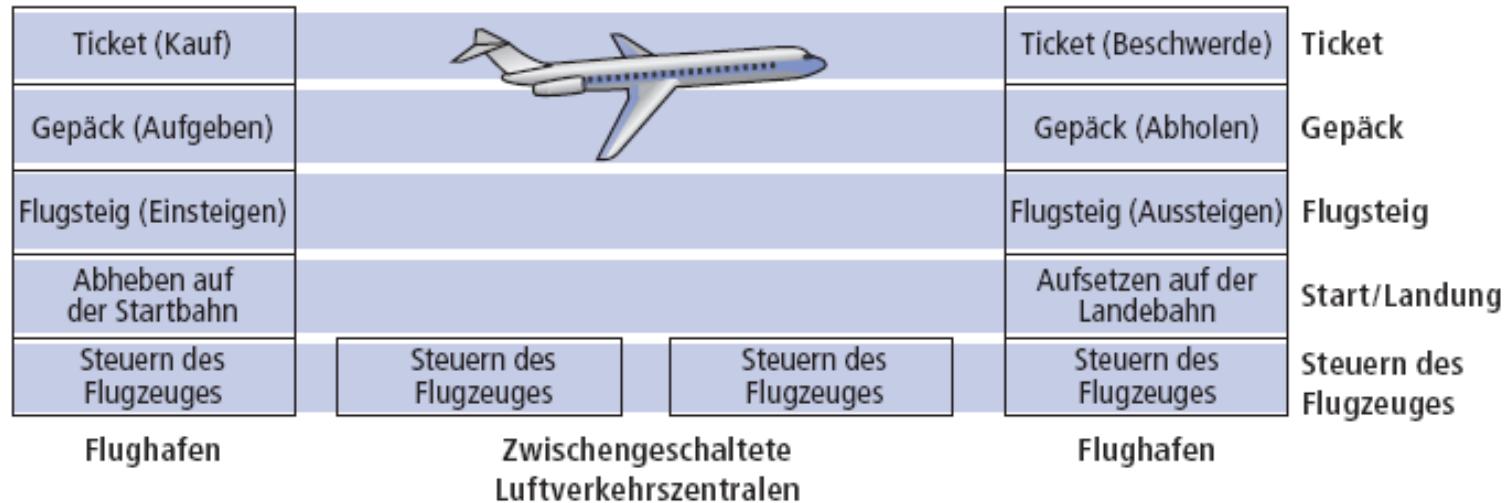
Oder: Können wir wenigstens unsere Betrachtung von Netzwerken strukturieren?

Protokollsichten – Luftfahrtanalogie



Beispiel Luftfahrt: eine Folge einzelner Schritte

Protokollsichten – Luftfahrtanalogie



Schichten: Jede Schicht implementiert einen Dienst

- mit Hilfe von „schichtinternen“ Aktionen
- unter Verwendung von Diensten der Schicht, die unter ihr liegt

Protokollsichten – Notwendigkeit

Warum Schichten?

Umgang mit komplexen Systemen:

- **Strukturierung** ermöglicht die Identifikation und das Verständnis des Zusammenspiels einzelner Bestandteile des Systems
 - **Referenzmodell** für die Diskussion des Systems
- **Modularisierung** vereinfacht Wartung und Arbeiten mit dem System
 - Änderungen an der Implementierung einer Schicht sind transparent für den Rest des Systems
 - *Beispiel: Eine Veränderung der Einsteigeprozедur am Gate beeinflusst nicht den Rest des Systems*

Protokollstapel des Internets

- **Anwendungsschicht:** Unterstützung von Netzwerkanwendungen
 -
- **Transportschicht:** Datentransfer zwischen Prozessen
 -
- **Netzwerkschicht (auch Vermittlungsschicht):** Weiterleiten der Daten von einem Sender zu einem Empfänger
 -
- **Sicherungsschicht:** Datentransfer zwischen benachbarten Netzwerksystemen
 -
- **Bitübertragungsschicht:** Bits auf der Leitung



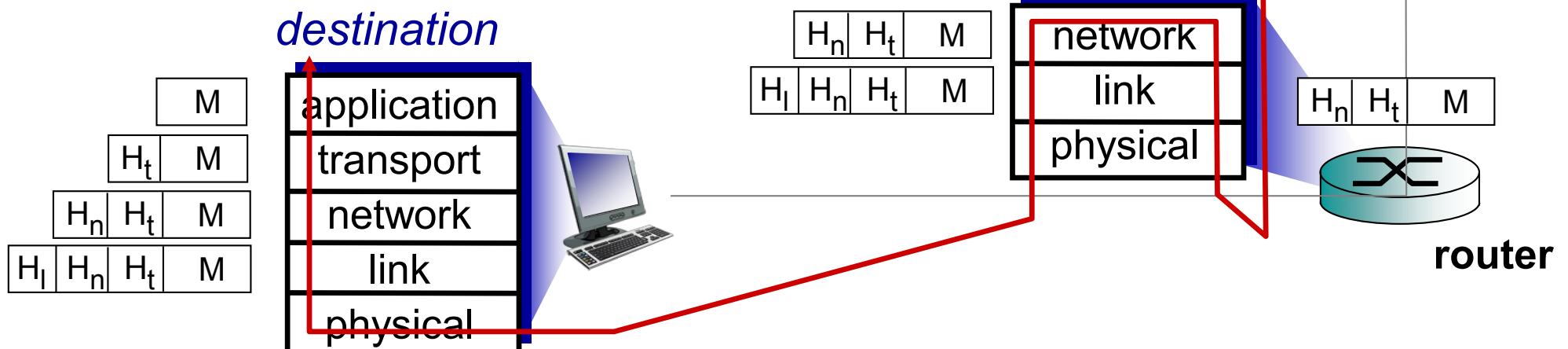
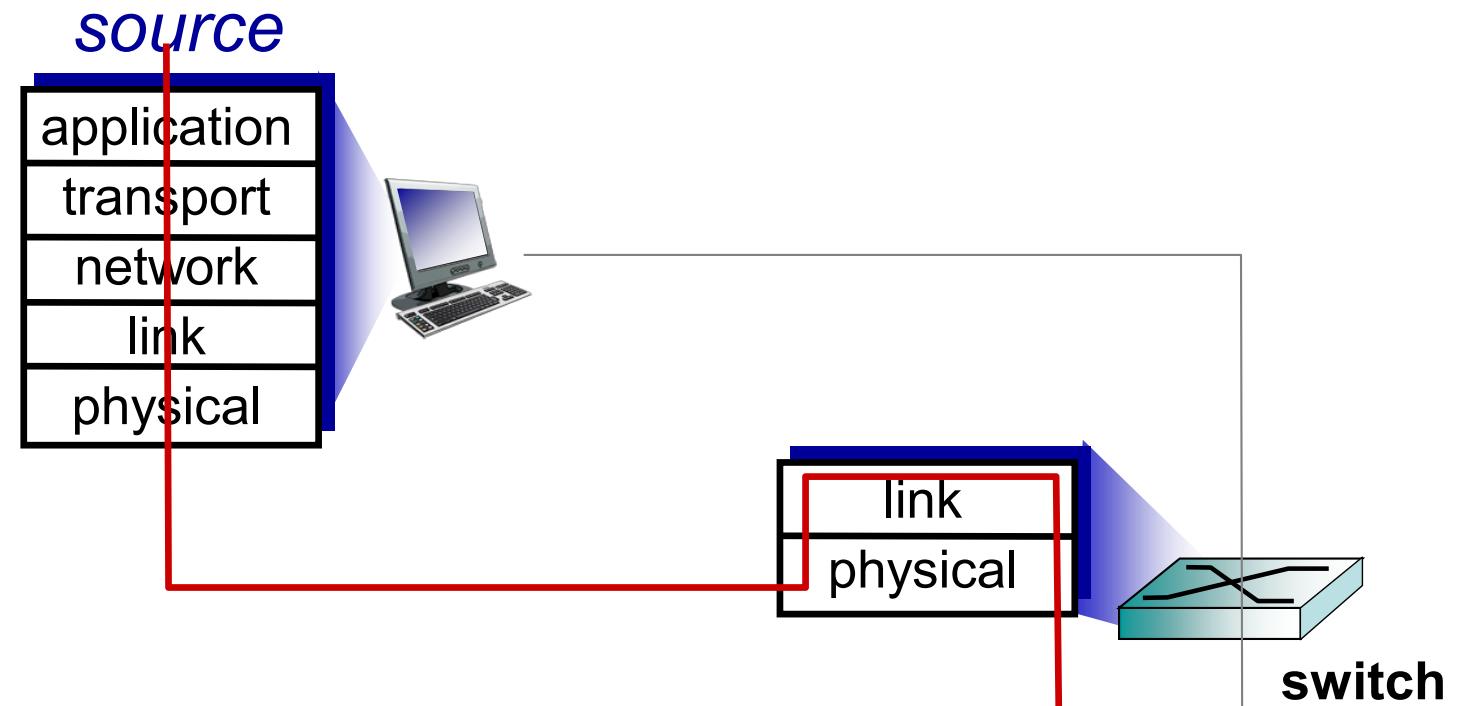
ISO/OSI-Referenzmodell

- **Darstellungsschicht:** Ermöglicht es Anwendungen, die Bedeutung von Daten zu interpretieren, z.B. Verschlüsselung, Kompression, Vermeidung systemspezifischer Datendarstellung
- **Kommunikationssteuerungsschicht:** Synchronisation, Setzen von Wiederherstellungspunkten
- Der Protokollstapel des Internets bietet diese Funktionalitäten nicht!
 - Wenn benötigt, müssen sie von der Anwendung implementiert werden

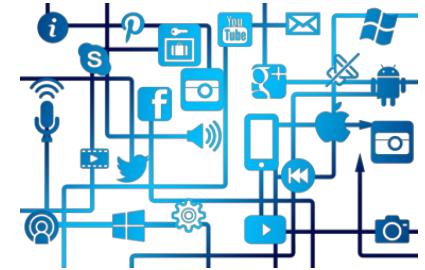


Kapselung

message	M
segment	H _t M
datagram	H _n H _t M
frame	H _l H _n H _t M



Weiterführendes Material



- Video zur Kapselung im Internet-Protokollstapel

<https://www.youtube.com/watch?v=nomyRJehhnM>

- Achtung: wir haben noch nicht über Adressierung und Ports gesprochen.
- Das Video sollte aber dennoch verständlich sein, um das Prinzip der Kapselung zu verstehen.

ISO/OSI und TCP/IP im Vergleich

Zum Nachlesen



OSI-Schicht	TCP/IP-Schicht	Beispiel
Anwendungen (7)	Anwendungen	HTTP, UDS, FTP, SMTP, POP, Telnet, OPC UA
Darstellung (6)		
Sitzung (5)		
Transport (4)	Transport	TCP, UDP, SCTP
Vermittlung (3)	Internet	IP (IPv4, IPv6), ICMP (über IP)
Sicherung (2)	Netzzugang	Ethernet, Token Bus, Token Ring, FDDI, IPoAC
Bitübertragung (1)		

Das TCP/IP-Modell definiert und referenziert eine umfangreiche Sammlung von Protokollen, die die Kommunikation zwischen Computern/Endsystemen regeln. Zur Definition eines Protokolls verwendet TCP/IP spezielle Dokumente, die als RFCs (Request for Comments) bezeichnet werden.

Im TCP/IP-Modell werden keine Standardisierungs- und Spezifikationsaufwände wiederholt, die bereits von anderen Standardisierungsorganisationen oder Anbieterkonsortien erledigt wurden. Die Standards oder Protokolle solcher Gruppen werden referenziert, z.B. auf den durch das IEEE (Institute of Electrical and Electronic Engineers) definierte Ethernet-Standard. Deswegen spezifiziert TCP/IP Ethernet nicht in RFCs.

Kapselung / Schichtenkonzept

Zum Nachlesen



Konzept

- Interaktion
gleichrangiger
Schichten auf
verschiedenen
Computern

Beschreibung

- Die beiden Computer verwenden Protokolle.
 - Die Protokolle verwenden Header um Informationen auszutauschen.
 - Der sendende Computer erstellt Header.
 - Der empfangende Computer verarbeitet die Header-Daten.
-
- Interaktion
benachbarter
Schichten auf
demselben Computer
 - Untergeordnete Schichten vermitteln Dienste an übergeordnete Schichten.
 - Die Soft- oder Hardware, die die übergeordnete Schicht implementiert, fordert bei der ihr untergeordneten Schicht die Ausführung der erforderlichen Funktionalität an.

Netzwerksicherheit



Netzwerksicherheit befasst sich mit

- Angriffen auf Computernetzwerke
- Schutzmechanismen gegen diese Angriffe
- Dem Design von Architekturen, die „immun“ gegen Angriffe sind

Das Internet wurde nicht mit dem Ziel der Sicherheit entworfen

- Vision: “Eine Gruppe von Benutzern, die sich gegenseitig vertrauen, sind über ein transparentes Netzwerk miteinander verbunden“
- Die Entwickler von Internetprotokollen versuchen, Sicherheit nachträglich einzubauen
- Inzwischen: Sicherheit wird in allen Protokollsichten untersucht!

Typische Attacken – Infizieren von Rechnern

Spyware

- Infektion durch Laden einer Webseite, die Spyware enthält
- Aufzeichnen und Weitermelden von Tastenanschlägen, besuchten Websites etc.



Viren

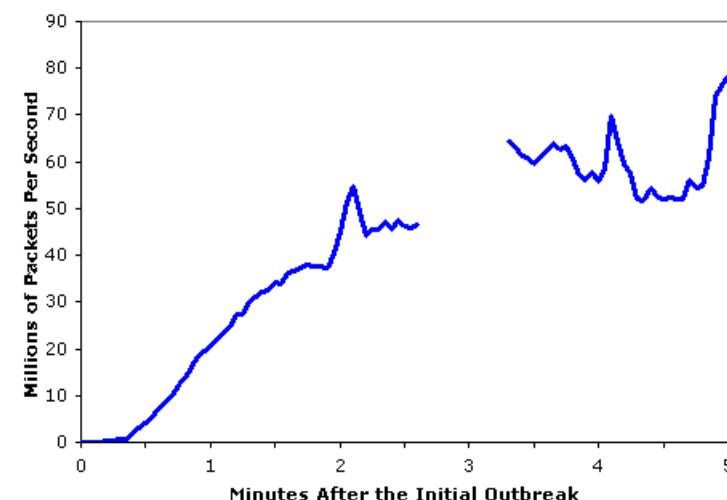
- Infektionen über empfangene Objekte (z.B. per E-Mail), erfolgen aktiv
- Selbst replizierende Viren verbreiten sich über weitere Endsysteme und Benutzer

→ Nutzung infizierter Systeme in einem „botnet“

Würmer

- Infektion durch Objekte, die ohne Benutzereingriff empfangen wurden
- Selbst replizierende Würmer verbreiten sich über weitere Endsysteme und Benutzer

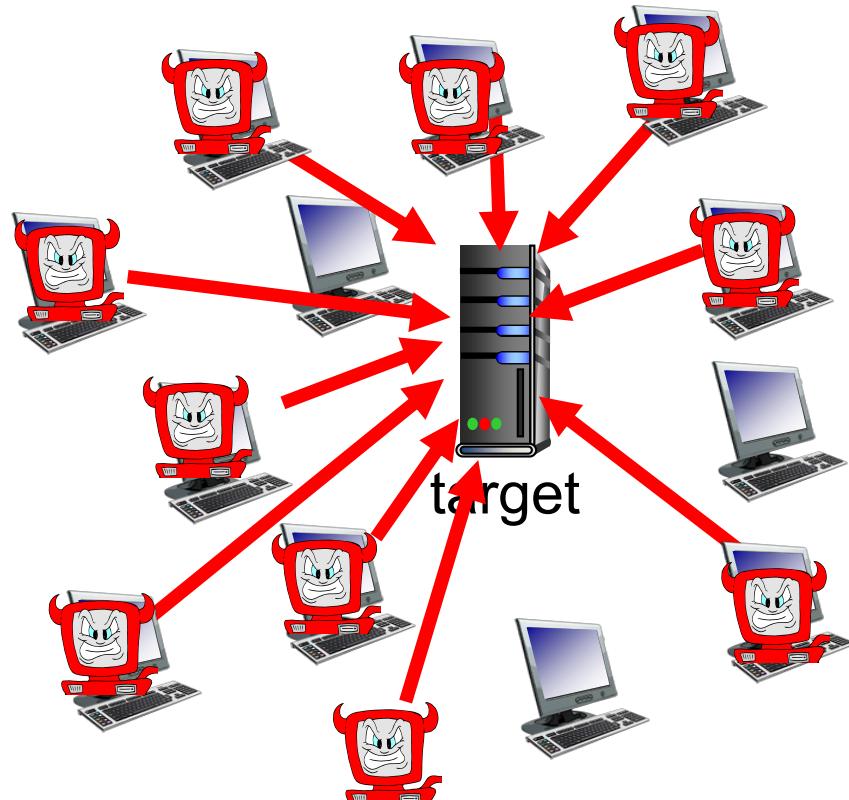
Aggregate Scans/Second in the first 5 minutes based on Incoming Connections To the WAIL Tarpit



Sapphire-Wurm: scans/s in den ersten 5 Minuten des Ausbruchs (Daten von CAIDA, UWisc)

Denial of Service-Angriffe (DoS)

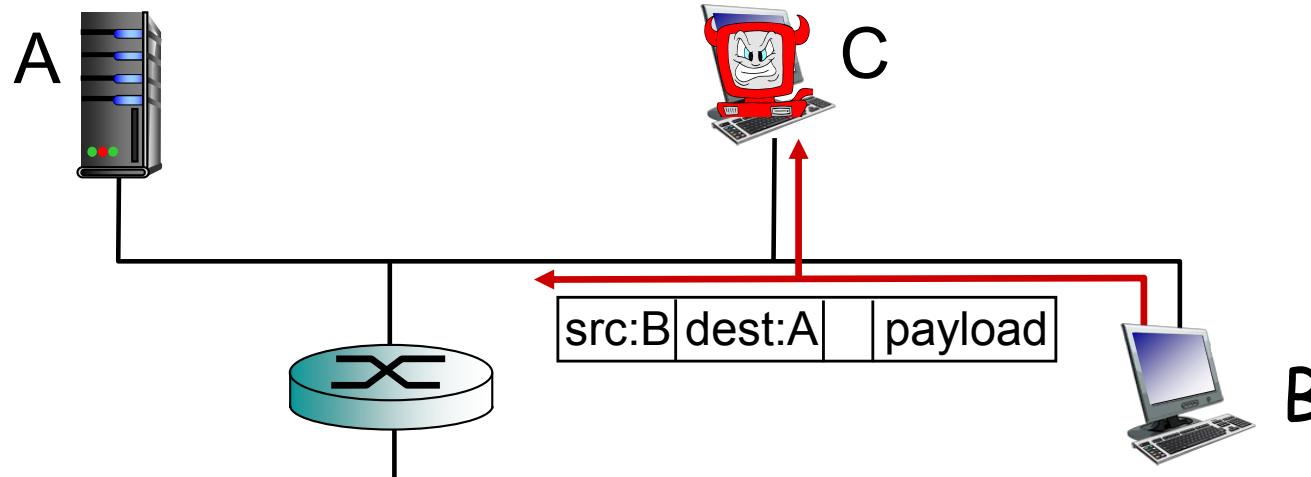
- Angreifer verhindern den Zugriff von Benutzern auf Ressourcen (Server, Bandbreite), indem diese durch den Angreifer belegt werden
1. Ziel auswählen
 2. Kompromittiere andere Systeme (s.o.)
 3. Sende viele Pakete von den kompromittierten Systemen an das Ziel



Typische Attacken – Mithören, Verändern und Löschen von Paketen

Mithören von Paketen

- Broadcast-Medien (Ethernet, WLAN)
- Netzwerkkarten im „Promiscuous Mode“ zeichnen alle Pakete auf, die sie hören können

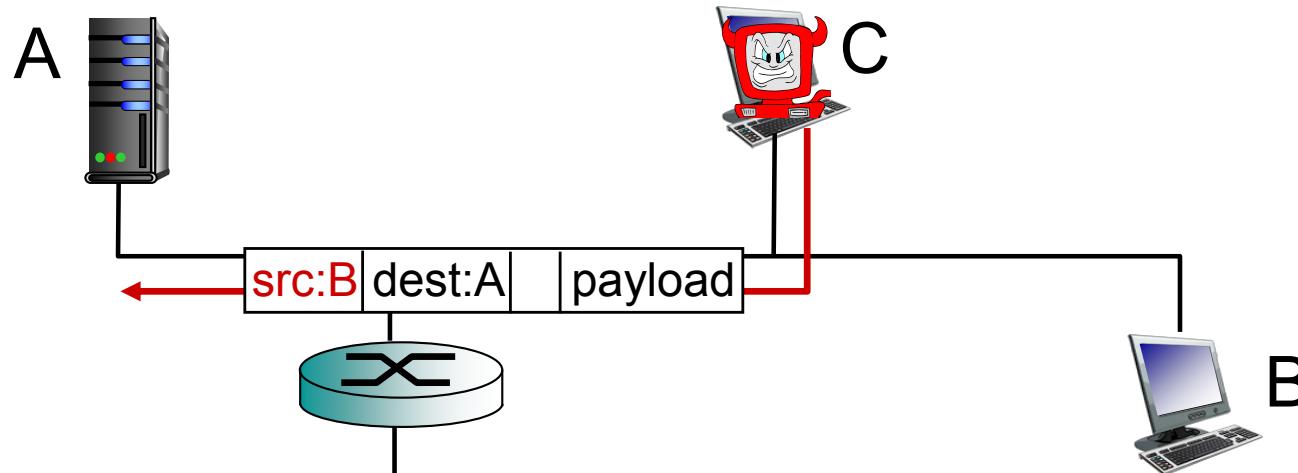


- Wireshark ist z.B. ein „Paket-Sniffer“

Typische Attacken – Sich als jemand anders ausgeben

IP Spoofing

- Sende Pakete mit falscher Absenderadresse



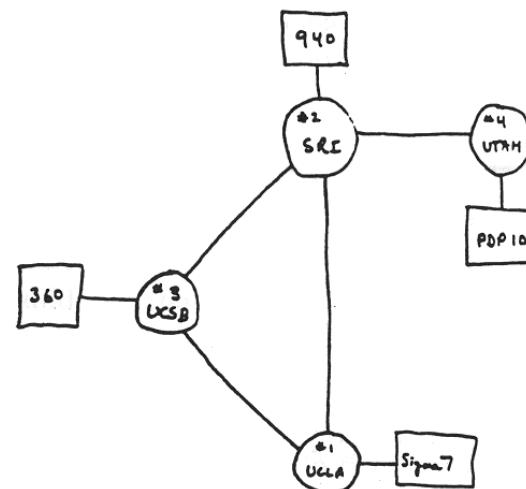
- Internet und Protokolle**
- Randbereich des Internets**
- Inneres des Internets**
- Netzwerkkenngrößen**
- Protokollsichten, Dienste und Netzwerksicherheit**
- Geschichte des Internets**
- Zusammenfassung und Ausblick**

Geschichte des Internets

1961-1972: Paketvermittlung

- 1961: Kleinrock – Warteschlangentheorie zeigt die Effizienz der Paketvermittlung
- 1964: Baran – Paketvermittlung in Militäernetzen
- 1967: ARPAnet von der Advanced Research Projects Agency geplant
- 1969: erster ARPAnet-Knoten in Betrieb

- 1972:
 - ARPAnet, öffentliche Vorführung
 - NCP (Network Control Protocol), erstes Protokoll zwischen Hosts
 - Erstes E-Mail-Programm
 - ARPAnet hat 15 Knoten



THE ARPANET

Geschichte des Internets

1972–1980: Netzwerk von Netzwerken

- 1970: ALOHAnet
Satellitennetzwerk auf Hawaii
- 1974: Cerf und Kahn – Architektur für die Verbindung von Netzwerken
- 1976: Ethernet: Xerox PARC
- Späte 1970er: proprietäre Architekturen: DECnet, SNA, XNA
- Späte 1970er: Pakete fester Größe (später ATM)
- 1979: ARPAnet hat 200 Knoten

Cerf und Kahn: Prinzipien für die Verbindung von Netzen

- Minimalismus, Autonomie – keine internen Änderungen an den einzelnen Netzwerken
- Best-Effort-Dienst – keine Garantien
- Kein (Verbindungs-) Zustand in den Routern
- Dezentrale Kontrolle

Definition der aktuellen Internetarchitektur

Geschichte des Internets

1980–1990: Neue Protokolle, Ausbreitung des Netzwerkes

- 1983: Einführung von TCP/IP
- 1982: Definition des SMTP-E-Mail-Protokolls
- 1983: Definition von DNS zur Übersetzung von Namen auf IP-Adressen
- 1985: Definition von ftp
- 1988: Überlastkontrolle in TCP
- Neue nationale Netzwerke: Csnet, BITnet, NSFnet, Minitel
- 100.000 Endsysteme sind an einen Verbund von Netzwerken angeschlossen

Geschichte des Internets

1990–200X: Kommerzialisierung, WWW, neue Anwendungen

- Anfang 1990er Jahre:
ARPAnet wird eingestellt
- 1991: NSF hebt die Einschränkungen bezüglich der kommerziellen Nutzung des NSFnet auf
- Anfang 1990er Jahre: Web
 - Hypertext [Bush 1945, Nelson 1960er]
 - HTML, HTTP: Berners-Lee
 - 1994: Mosaic, später Netscape
 - Späte 1990er Jahre:
Kommerzialisierung des Web

Späte 1990er–200X:

- Mehr „Killeranwendungen“: Instant Messaging, P2P-Filesharing
- Netzwerksicherheit wird immer wichtiger
- Geschätzte 50 Millionen Endsysteme, 100 Millionen Anwender
- Backbone-Leitungen mit Gbit/s

Geschichte des Internets

2005 – heute

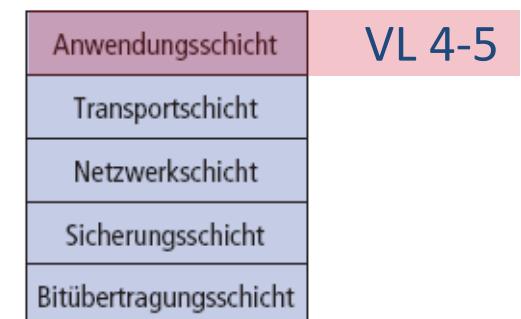
- >750 Millionen Hosts
 - Smartphones und Tablets
- Umfangreiche Umsetzung von Breitband-Zugängen
- Steigende Präsenz von kabellosem high-speed Zugang
- Erfolg von Sozialen Netzwerken
 - Facebook: 1,5 Mrd. Nutzer
- Dienstanbieter (Google, Microsoft) betreiben eigene Netze
 - Bypass Internet, direkter Zugriff auf die angebotenen Dienste
- Cloud-Angebote wachsen stetig

- Internet und Protokolle**
- Randbereich des Internets**
- Inneres des Internets**
- Netzwerkkenngrößen**
- Protokollsichten, Dienste und Netzwerksicherheit**
- Geschichte des Internets**
- Zusammenfassung und Ausblick**

Zusammenfassung (VL 1-3) und Ausblick



- In den letzten beiden Vorlesungen wurde eine Menge Stoff vermittelt:
 - Überblick über das Internet und Computernetzwerke
 - Randbereich und Inneres des Internets, Zugangsnetzwerke
 - Paketvermittlung vs. Leitungsvermittlung, Internet-Struktur
 - Kennzahlen (Paketverlust, Verzögerung, Durchsatz) und Sicherheitsaspekte
 - Protokolle, Dienste, Schichtenmodelle
 - Historischer Abriss
- Sie sollten nun ein Grundverständnis für Netzwerke, das Internet und damit verbundene Herausforderungen haben.
- In den folgenden beiden Veranstaltungen werden wir uns dem top-down Ansatz folgend auf die oberste Schicht konzentrieren: **Anwendungsschicht**



Vielen Dank für Ihre Aufmerksamkeit!



**BW332 - Rechnernetze
(mit Praktikum)**

**Vorlesungen 4-5
Anwendungsschicht**



Ziele



- Die Aufgaben und Protokolle der Anwendungsschicht verstehen und selbst anwenden können.
- Allgemeine Anforderungen von Anwendungen an das Netzwerk wiedergeben können und unterschiedliche Anwendungs-Architekturen (Client-Server und Peer-to-Peer) kennenlernen und deren Anwendungsfälle abgrenzen können.
- Das Protokoll HTTP als Grundlage des WWW im Detail durchdringen sowie weitere Protokolle zum Dateitransfer (FTP) und E-Mail (SMTP, POP3, IMAP) kennenlernen und beispielhaft anwenden.
- Das Domain Name System verstehen.

■ Grundlagen von Netzwerk-Applikationen

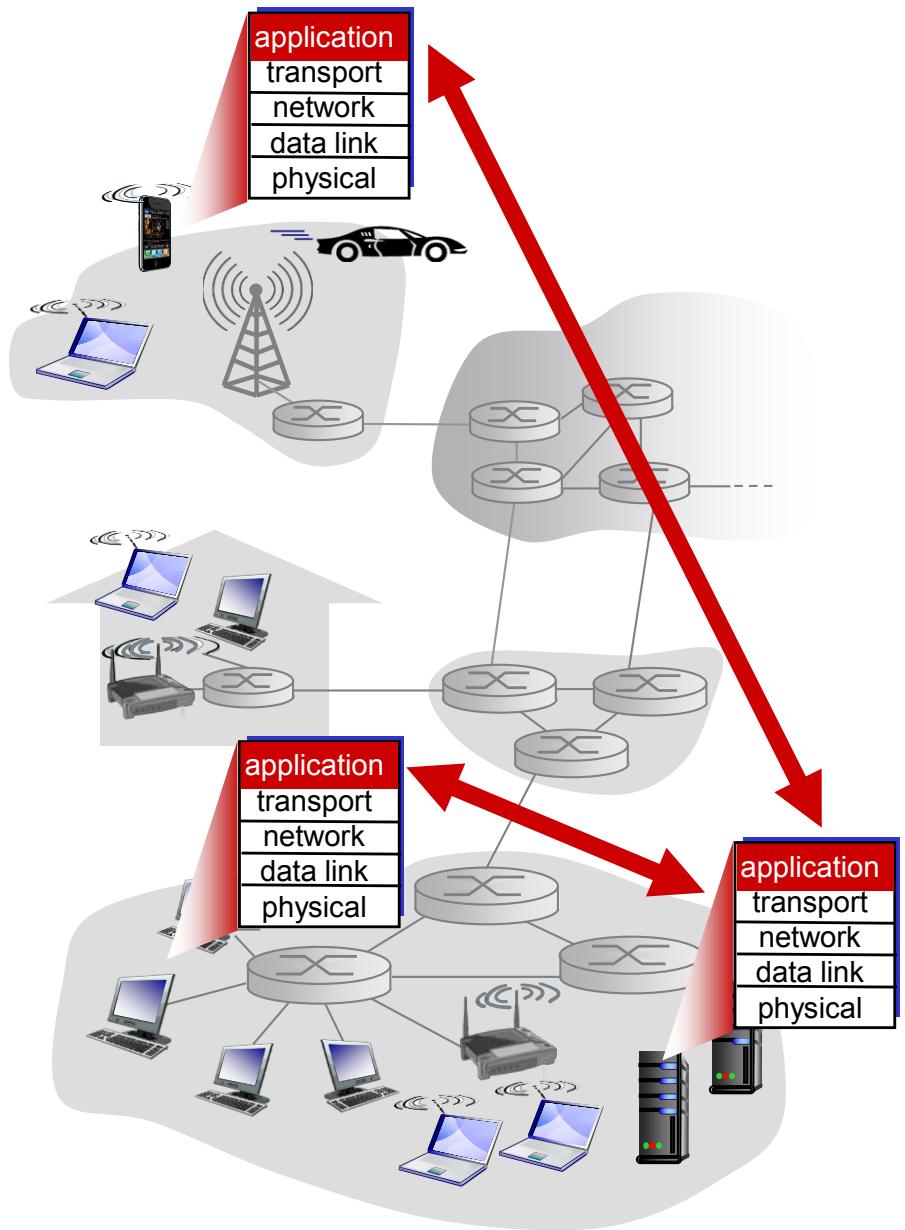
- Web und HTTP
- FTP und E-Mail
- Domain Name System
- Peer-to-Peer Anwendungen
- Socket-Programmierung
- Zusammenfassung und Ausblick

Einige Netzwerkanwendungen

- E-Mail
- Web
- Instant Messaging
- Terminalfernzugriff
- P2P-Filesharing
- Netzwerkspiele
- Streaming von Videoclips (youtube, netflix, ...)
- Voice over IP (VoIP)
- Videokonferenzen
- Websuche
- Soziale Netzwerke
- uvm.

Entwickeln einer Netzwerkanwendung

- Entwicklung von Programmen, die
 - auf mehreren (verschiedenen) **Endsystemen** laufen und
 - über das **Netzwerk kommunizieren**.
 - Beispiel: Die Software eines Webservers kommuniziert mit dem Browser (Software).
- Es ist dazu nicht nötig, Software für das Innere des Netzwerks zu programmieren
 - Im Inneren des Netzwerkes werden keine **Nutzer-Anwendungen** ausgeführt.
 - Nutzer-Anwendungen greifen „nur“ auf die angebotenen Netzwerk-Dienste zu.
 - Dies erlaubt eine schnelle Entwicklung und Verbreitung von Applikationen.



Verschiedene Architekturen

- Client-Server
- Peer-to-Peer (P2P)
- Kombination von Client-Server und P2P

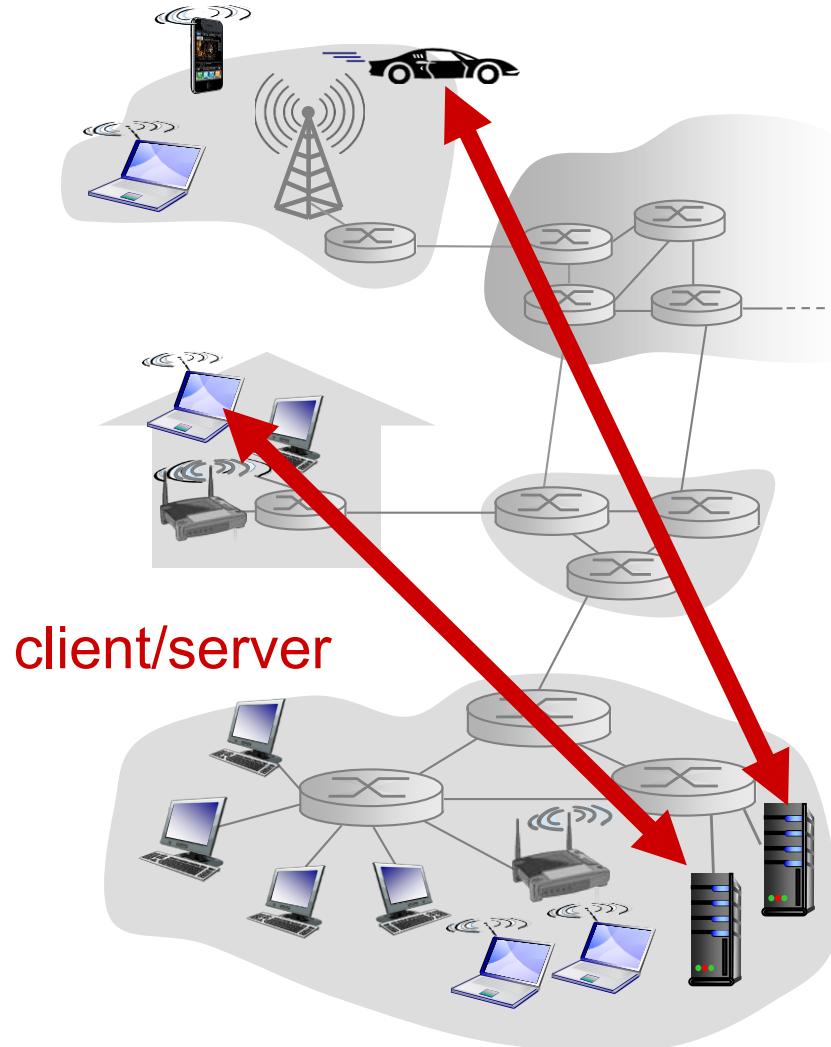
Client-Server-Architektur

- Server:

-
-
-

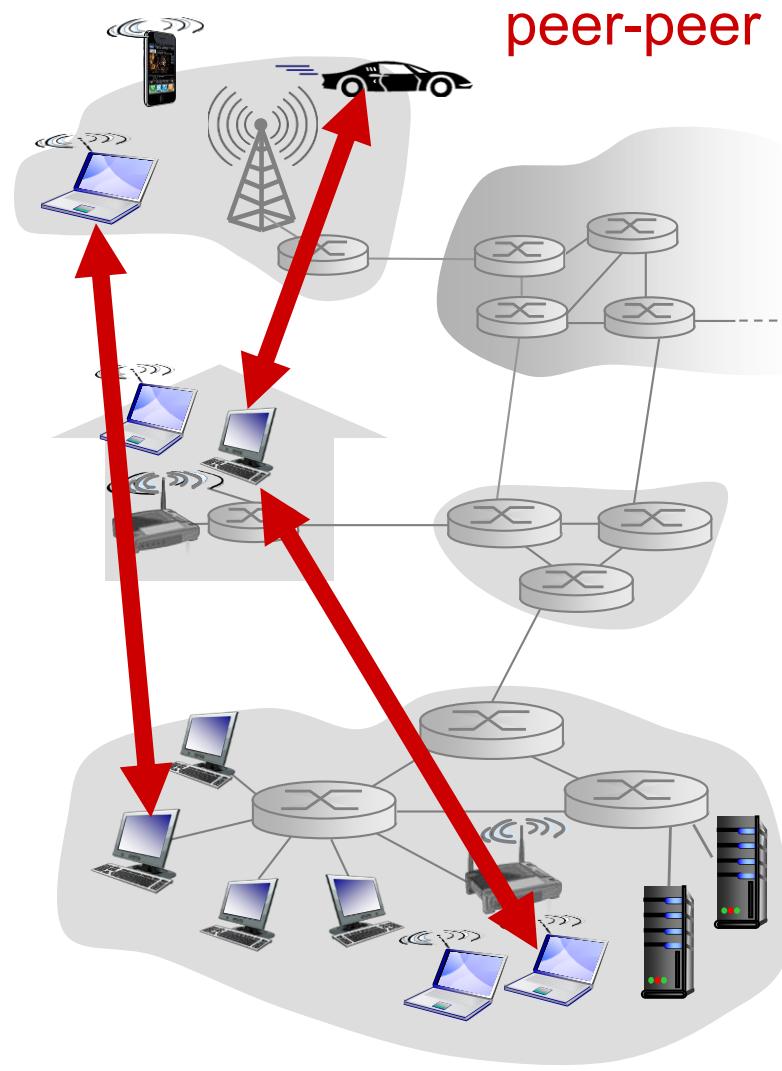
- Clients:

- Kommunizieren mit Servern
- Können nur sporadisch angeschlossen sein
- Können dynamische IP-Adressen haben
- Kommunizieren nicht direkt miteinander



Reine P2P-Architektur

-
- Beliebige Endsysteme kommunizieren direkt miteinander
- Peers sind nur sporadisch angeschlossen und wechseln ihre IP-Adresse
 - Selbst-Skalierung: neue Peers bringen neue Kapazität für den verteilten Dienst (aber auch Nachfrage nach dem Dienst)
- Gut skalierbar, aber schwer zu warten und zu kontrollieren!
 - Komplexes Management der Applikations-Teilnehmer nötig



Kombination von Client-Server und P2P

- Skype
 - P2P-Anwendung für Voice-over-IP
 - Zentraler Server: Adresse des Kommunikationspartners finden
 - Verbindung zwischen Clients: direkt (nicht über einen Server)
- Instant Messaging
 - Chat zwischen zwei Benutzern: P2P
 - Zentralisierte Dienste: Erkennen von Anwesenheit, Zustand, Aufenthaltsort eines Anwenders
 - Benutzer registriert seine IP-Adresse beim Server, sobald er sich mit dem Netz verbindet
 - Benutzer fragt beim Server nach Informationen über seine Freunde und Bekannten

Kommunizierende Prozesse

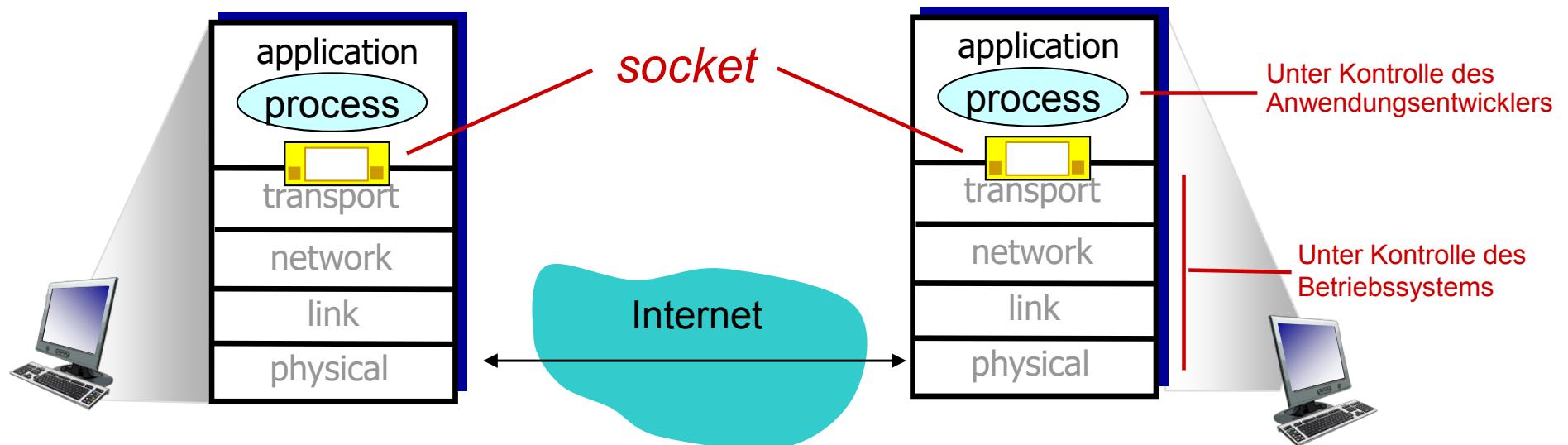
- Prozess: Programm, das auf einem Host läuft
 - Innerhalb eines Hosts können zwei Prozesse mit Inter-Prozess-Kommunikation Daten austauschen (durch das Betriebssystem)
 - Prozesse auf verschiedenen Hosts kommunizieren, indem sie **Nachrichten über ein Netzwerk** austauschen

Clients, Server

- Client-Prozess: Prozess, der die Kommunikation **beginnt**
- Server-Prozess: Prozess, der darauf **wartet**, kontaktiert zu werden
- Anmerkung: Anwendungen mit einer P2P-Architektur haben gleichzeitig Client- und Server-Prozesse

Sockets

- Prozesse senden / empfangen Nachrichten über einen Socket



- Ein Socket lässt sich mit einer Tür vergleichen
 - Der sendende Prozess schiebt die Nachrichten durch die Tür
 - Der sendende Prozess verlässt sich auf die Transportinfrastruktur auf der anderen Seite der Tür, um die Nachricht zum Socket des empfangenden Prozesses zu bringen
- API: (1) Wahl des Transportprotokolls; (2) Einstellen einiger Parameter (mehr dazu später)

Adressierung von Prozessen

- Um eine Nachricht empfangen zu können, muss ein Prozess identifiziert werden können.
- Ein Host besitzt eine eindeutige, 32 Bit lange IP-Adresse.
- Genügt die IP-Adresse, um einen Prozess auf diesem Host zu identifizieren?
 -
- Prozesse werden durch eine IP-Adresse UND eine Portnummer identifiziert
- Beispiel-Portnummern:
 - HTTP-Server: 80
 - E-Mail-Server: 25
- Um an den Webserver gaia.cs.umass.edu eine HTTP-Nachricht zu schicken:
 - IP-Adresse: 128.119.245.12
 - Portnummer: 80
- Mehr dazu in Kürze...

Anwendungsprotokolle bestimmen ...

- **Arten von Nachrichten**
 - z.B. Request, Response
- **Syntax** der Nachrichten
 - Welche Felder sind vorhanden und wie werden diese voneinander getrennt?
- **Semantik** der Nachrichten
 - Bedeutung der Informationen in den Feldern
- **Regeln** für das Senden von und Antworten auf Nachrichten

Öffentlich verfügbare Protokolle:

- Definiert in RFCs
- Ermöglichen Interoperabilität
- z.B.

Proprietäre Protokolle:

- z.B.

Wahl des Transportdienstes

Datenverlust

- Einige Anwendungen können Datenverlust tolerieren (z.B. Audioübertragungen)
- Andere Anwendungen benötigen einen absolut zuverlässigen Datentransfer (z.B. Dateitransfer)

Zeitanforderungen

- Einige Anwendungen (z.B. Internettelefonie oder Netzwerkspiele) tolerieren nur eine sehr geringe Verzögerung

Bandbreite

- Einige Anwendungen (z.B. Multimedia-Streaming) brauchen eine Mindestbandbreite, um zu funktionieren
- Andere Anwendungen verwenden einfach die verfügbare Bandbreite (bandbreitenelastische Anwendungen)

Sicherheit

- Verschlüsselung, Datenintegrität, ...

Beispiele für Anforderungen von Anwendungen

Anwendung	Datenverlust	Bandbreite	Echtzeit
Dateitransfer	Kein Verlust	Elastisch	Nein
E-Mail	Kein Verlust	Elastisch	Nein
Web	Kein Verlust	Elastisch (wenige Kbps)	Nein
Internettelefonie/ Bildkonferenz	Toleriert Verluste	Audio: wenige Kbps bis 1 Mbps Video: 10 Kbps bis 5 Mbps	Ja: einige Hundert ms
Gespeichertes Audio/Video	Toleriert Verluste	Wie oben	Ja: wenige Sekunden
Interaktive Spiele	Toleriert Verluste	Wenige Kbps bis 10 Kbps	Ja: einige Hundert ms
Instant Messaging	Kein Verlust	Elastisch	Ja und nein

Dienste der Transportprotokolle

TCP-Dienste:

- **Verbindungsorientierung:** Herstellen einer Verbindung zwischen Client und Server
- **Zuverlässiger Transport** zwischen sendendem und empfangendem Prozess
- **Flusskontrolle:** Sender überlastet Empfänger nicht
- **Überlastkontrolle:** Bremsen des Senders, wenn das Netzwerk überlastet ist
-

UDP-Dienste:

- **Unzuverlässiger Transport** von Daten zwischen Sender und Empfänger
-
- Wozu soll das gut sein? Warum gibt es UDP?

Beispiel Internet-Applikationen

Anwendung	Anwendungsschichtprotokoll	Zugrunde liegendes Transportprotokoll
E-Mail-Dienst	SMTP [RFC 2821]	TCP
Remote-Terminalzugang	Telnet [RFC 854]	TCP
World Wide Web	HTTP [RFC 2616]	TCP
Dateitransfer	FTP [RFC 959]	TCP
Multimedia-Streaming	HTTP (z.B. YouTube), RTP	TCP oder UDP
Internettelefonie	SIP, RTP oder proprietär (z.B. Skype)	Normalerweise UDP

Securing TCP

TCP & UDP

- Keine Verschlüsselung
- Klartext-Passwörter, die über Sockets gesendet werden, durchlaufen das Internet auch im Klartext

SSL

- Stellt verschlüsselte TCP-Verbindungen zur Verfügung
- Stellt Datenintegrität sicher
- Authentifizierung der Endsysteme

SSL befindet sich auf der Anwendungsschicht

- Anwendungen nutzen SSL Bibliotheken, die mit TCP „sprechen“ (darauf zurückgreifen)

SSL Socket API

- Klartext-Passwörter, die über SSL-Sockets gesendet werden, durchlaufen das Internet verschlüsselt.

Agenda

Grundlagen von Netzwerk-Applikationen

Web und HTTP

FTP und E-Mail

Domain Name System

Peer-to-Peer Anwendungen

Socket-Programmierung

Zusammenfassung und Ausblick

Web und HTTP

Definitionen

- Eine **Webseite** besteht aus **Objekten**.
- Objekte können sein: HTML-Dateien, JPEG-Bilder, Java-Applets, Audiodateien, ...
- Eine Webseite hat eine **Basis-HTML-Datei**, die mehrere referenzierte Objekte beinhalten kann.
- Jedes Objekt kann durch eine **URL** (Uniform Resource Locator) adressiert werden.
- Beispiel für eine URL:

www.someschool.edu/someDept/pic.gif

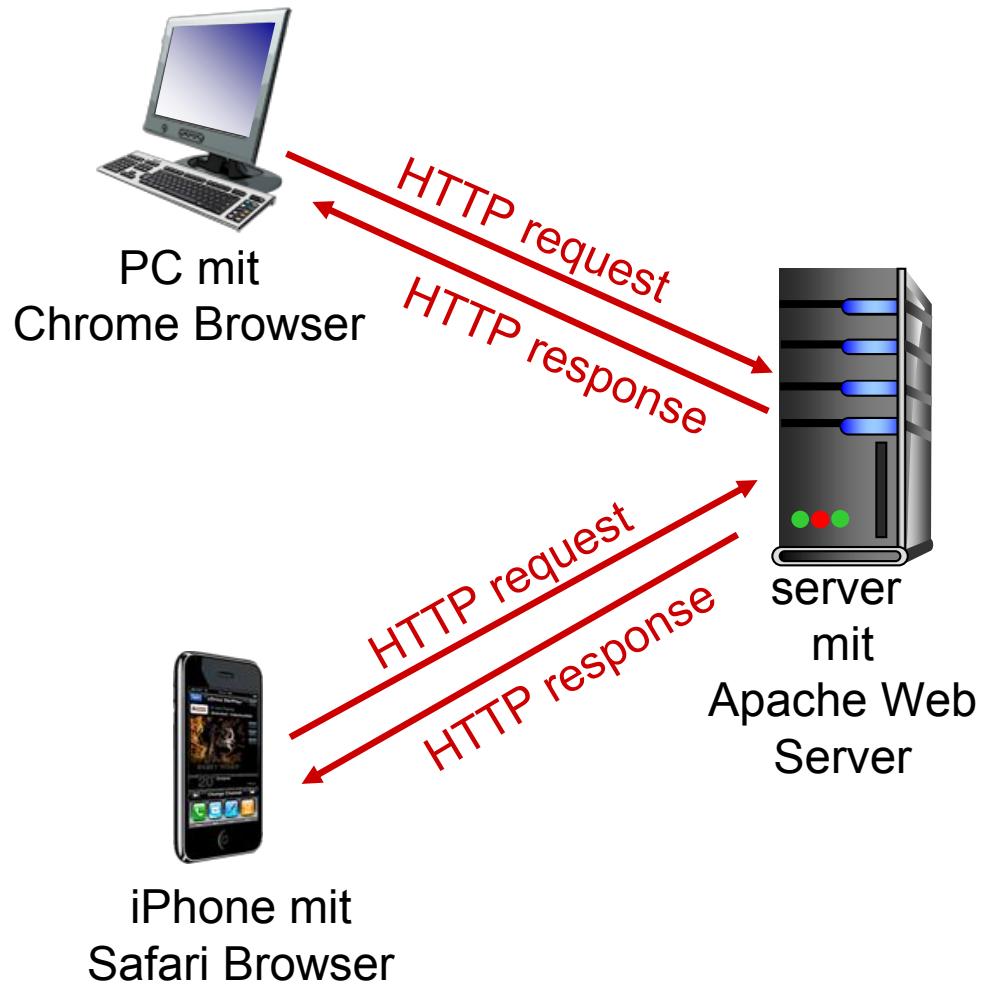
Hostname

Pfad

HTTP: Überblick

HTTP: HyperText Transfer Protocol

- Das Anwendungsprotokoll des Web
- Client/Server-Modell
 - Client: Browser, der Objekte (mittels HTTP) anfragt und erhält sowie diese anzeigt
 - Server: Webserver verschiickt Objekte (mittels HTTP) auf Anfrage

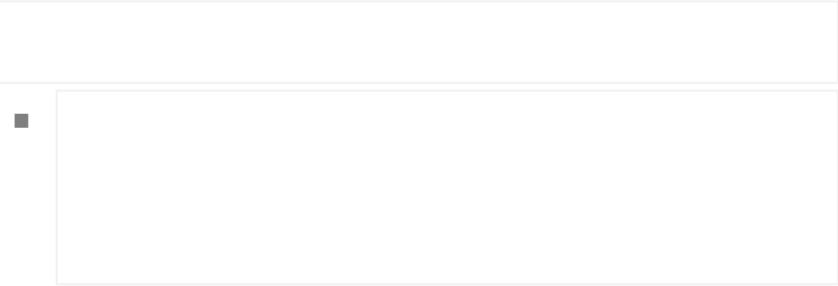


HTTP: Überblick (Fortsetzung)

- HTTP verwendet TCP

- Client baut mit der Socket-API eine TCP-Verbindung zum Server auf.
- Server wartet auf Port 80.
- Server nimmt die TCP-Verbindung des Clients an.
- HTTP-Nachrichten (Protokollnachrichten der Anwendungsschicht) werden zwischen Browser (HTTP-Client) und Webserver (HTTP-Server) ausgetauscht.
- Die TCP-Verbindung wird geschlossen.

-



nebenbei

Protokolle, die einen Zustand verwalten, sind komplex!

- Der Zustand muss gespeichert und verwaltet werden.
- Wenn Server oder Client abstürzen, dann muss der Zustand wieder synchronisiert werden.

HTTP-Verbindungen

Nichtpersistentes HTTP

- Maximal ein Objekt wird über eine TCP-Verbindung übertragen.
 - Danach wird die Verbindung geschlossen.
- Sollen mehrere Objekte heruntergeladen werden, dann werden mehrere Verbindungen benötigt.

Persistentes HTTP

- Mehrere Objekte können über eine TCP-Verbindung übertragen werden.

Nichtpersistentes HTTP

- Es soll folgende URL geladen werden:

www.someSchool.edu/someDepartment/home.index

(beinhaltet Text und Referenzen
auf 10 JPEG-Bilder)

1a. HTTP-Client initiiert TCP-
Verbindung zum HTTP-Server
(Prozess) auf
www.someSchool.edu, Port 80

1b. HTTP-Server auf Host
www.someSchool.edu wartet
auf TCP-Verbindungen an Port
80, akzeptiert Verbindung,
benachrichtigt Client

2. HTTP-Client schickt einen
HTTP-Request (beinhaltet die
URL
someDepartment/home.index)
über den TCP-Socket

3. HTTP-Server empfängt den
HTTP-Request, konstruiert eine
HTTP-Response-Nachricht,
welche das angefragte Objekt
beinhaltet, und sendet diese
über den Socket an den Client

Zeit
↓

Nichtpersistentes HTTP (Forts.)

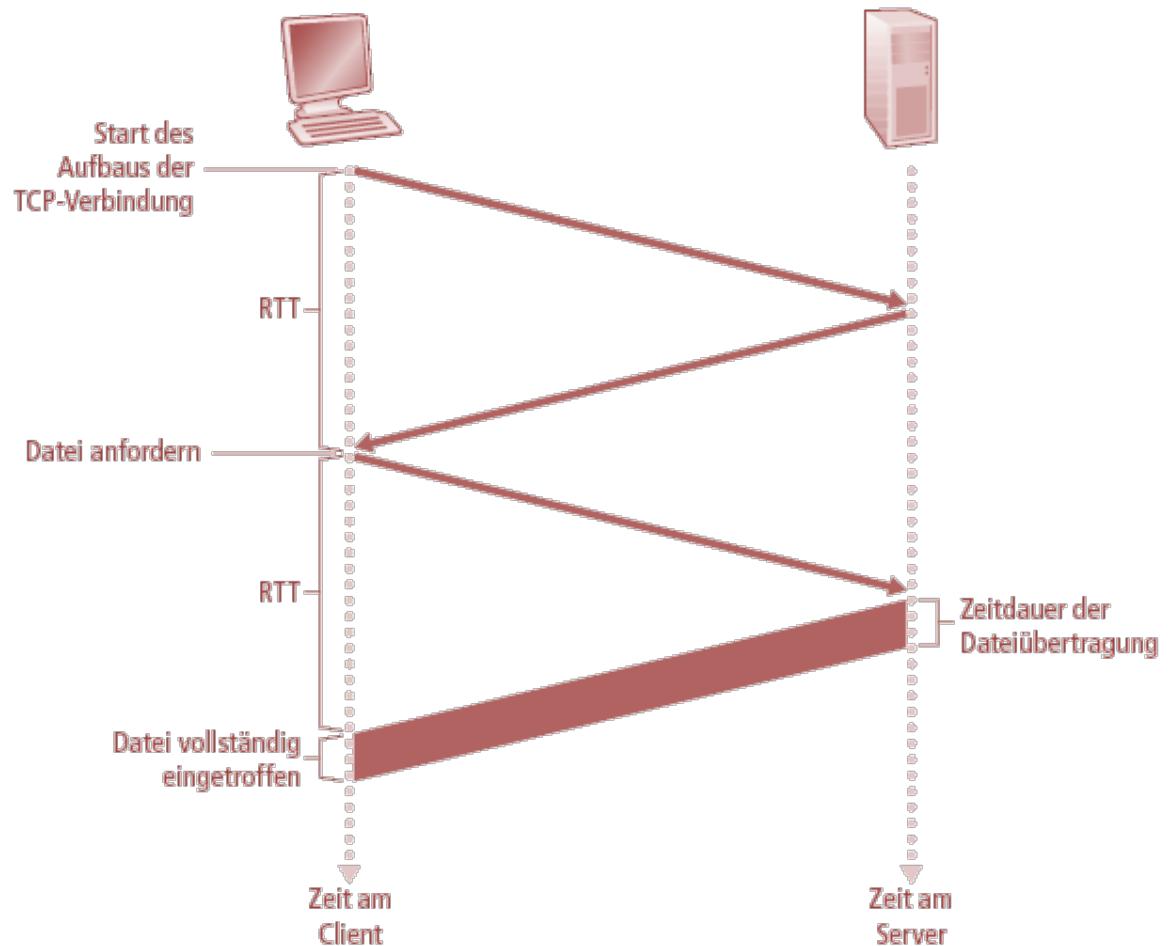
Zeit
↓



4. HTTP-Server schließt die TCP-Verbindung
5. HTTP-Client empfängt die Nachricht und stellt fest, dass zehn JPEG-Objekte referenziert werden.
6. Schritte 1 bis 5 werden für jedes der zehn JPEG-Objekte wiederholt, dann kann die Seite vollständig angezeigt werden.

Nichtpersistentes HTTP: Verzögerung

- Definition von RTT (Round Trip Time):
 -
- Verzögerung:
 - Eine RTT für den TCP-Verbindungsaufbau
 - Eine RTT für den HTTP-Request, bis das erste Byte der HTTP-Response beim Client ist
 - Zeit für das Übertragen der Daten auf der Leitung
 - Zusammen = 2 RTT + Übertragungsverzögerung



Persistentes HTTP

Probleme mit nichtpersistenterem HTTP:

- 2 RTTs pro Objekt
- Aufwand im Betriebssystem für jede TCP-Verbindung
- Browser öffnen oft mehrere parallele TCP-Verbindungen, um die referenzierten Objekte zu laden

Persistentes HTTP

- Server lässt die Verbindung nach dem Senden der Antwort offen.
- Nachfolgende HTTP-Nachrichten können über dieselbe Verbindung übertragen werden.

Persistent **ohne** Pipelining:

- Client schickt neuen Request erst, nachdem die Antwort auf den vorangegangenen Request empfangen wurde.
- Eine RTT für jedes referenzierte Objekt

Persistent mit Pipelining:

- Client schickt Requests, sobald er die Referenz zu einem Objekt findet
- Idealerweise wird nur wenig mehr als eine RTT für das Laden aller referenzierten Objekte benötigt

HTTP-Request-Nachricht

- Zwei Arten von HTTP-Nachrichten: Request, Response
- HTTP-Request-Nachricht:
 - ASCII (vom Menschen leicht zu lesen)

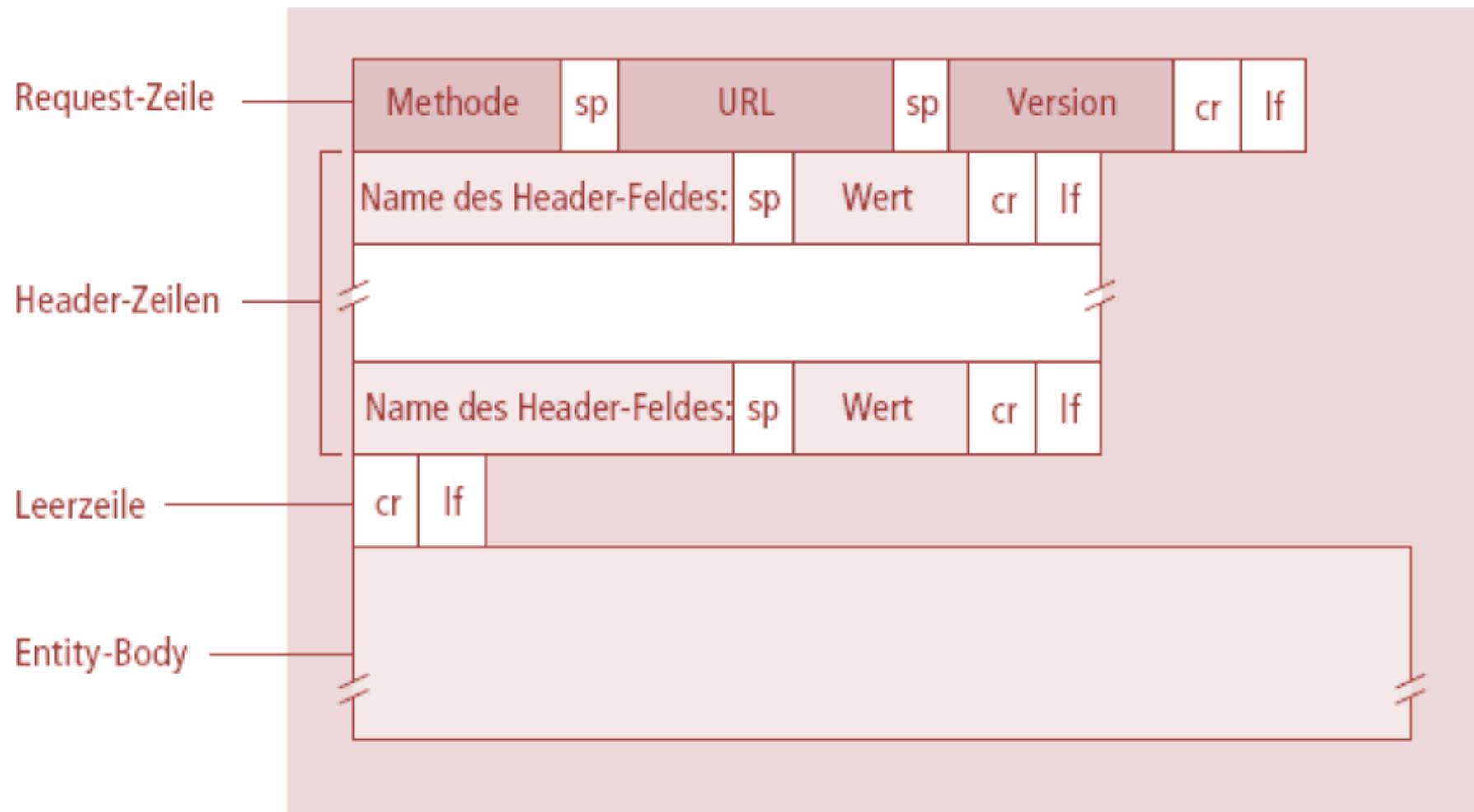
Request-Zeile
(GET, POST,
HEAD commands)

Header-Zeilen

Zusätzlicher Carriage Return +
Zeilenvorschub zeigt das Ende der Nachricht an

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:de
```

HTTP-Request-Nachricht: Format



Hochladen von Informationen

Post-Anweisung:

- Webseiten beinhalten häufig Formulare, in denen Eingaben erfolgen sollen
- Eingaben werden zum Server im Datenteil (entity body) der Post-Anweisung übertragen

Get-Anweisung:

- Eingabe wird als Bestandteil der URL übertragen:

www.somesite.com/animalsearch?monkeys&banana

<http://javascript-coder.com/files/htm-form-tutorial/html-form-tutorial-example-1.html>

Typen von Anweisungen

HTTP/1.0

- GET
- POST
- HEAD
 - Bittet den Server, nur die Kopfzeilen der Antwort (und nicht das Objekt) zu übertragen

HTTP/1.1

- GET, POST, HEAD
- PUT
 - Lädt die im Datenteil enthaltene Datei an die durch eine URL bezeichnete Position hoch
- DELETE
 - Löscht die durch eine URL angegebene Datei auf dem Server

HTTP-Response-Nachricht

Statuszeile
(Statuscode,
Statusnachricht)

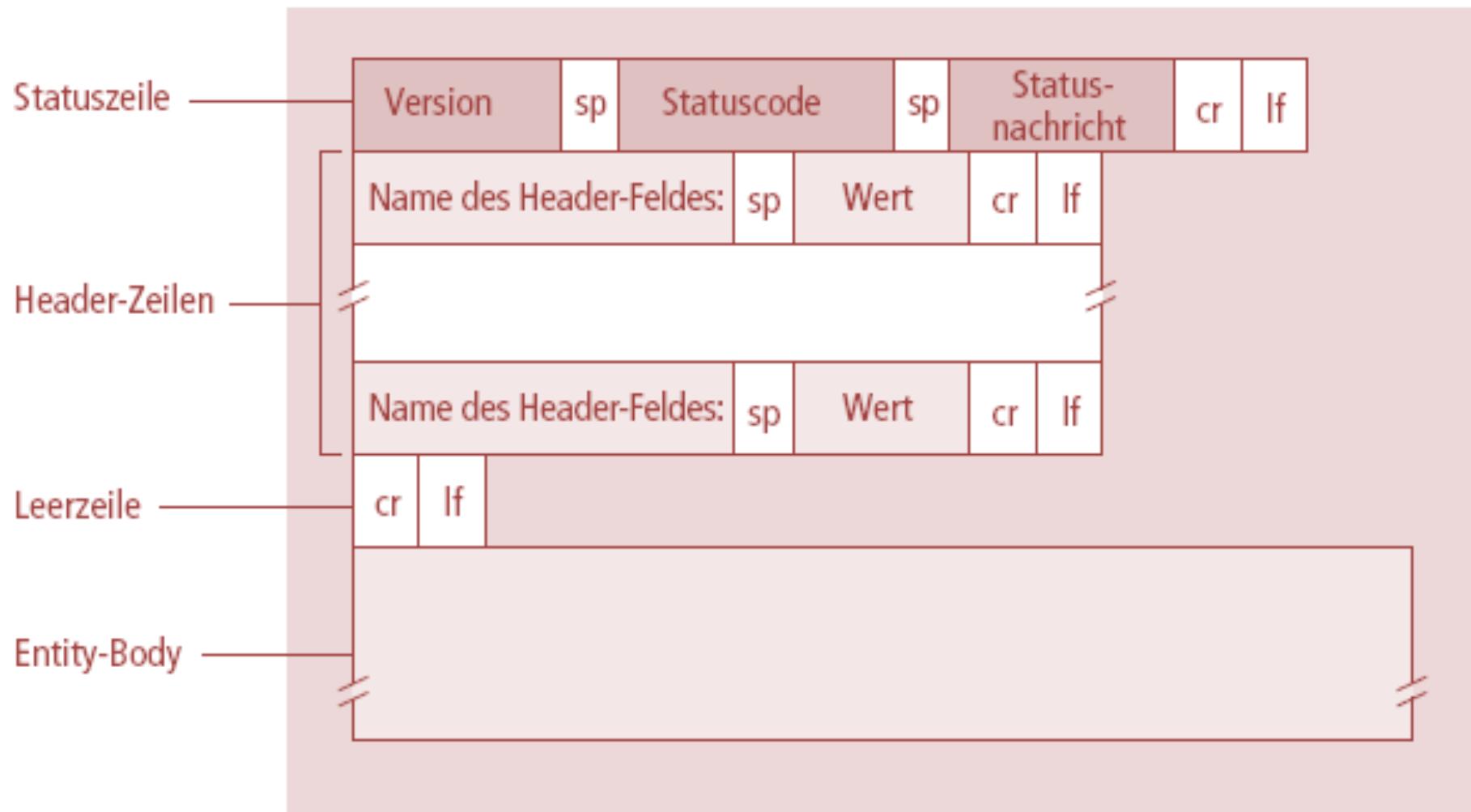
Header-Zeilen

Entity Body: Daten, z.B. die
angefragte HTML-Datei

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 .....
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

HTTP-Response-Nachricht: Format



Statuscodes für HTTP-Response

Der Status steht in der ersten Zeile der Response-Nachricht (von Server an Client)

Beispiele für Statuscodes:

200 OK

- Request war erfolgreich, gewünschtes Objekt ist in der Antwort enthalten

301 Moved Permanently

- Gewünschtes Objekt wurde verschoben, neue URL ist in der Antwort enthalten

400 Bad Request

- Request-Nachricht wurde vom Server nicht verstanden

404 Not Found

- Gewünschtes Objekt wurde nicht gefunden

505 HTTP Version Not Supported



HTTP im Einsatz (Wireshark)

New Tab - Chromium@ubuntu-512mb-fra1-01

The Wireshark Network Analyzer

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression... +

Welcome to Wireshark

Open

/home/wireshark-traces/tcp-ethereal-trace-1 (177 KB)
/home/student1/wireshark-traces/tcp-ethereal-trace-1 (not found)
/home/student1/wireshark-traces/dns-ethereal-trace-1 (not found)
/home/student1/wireshark-traces/dhcp-ethereal-trace-1 (not found)

Capture

...using this filter: X

eth0
any
Loopback: lo
nflog
nfqueue
usbmon1
 Cisco remote capture: cisco
 Random packet generator: randpkt
 EFLU remote capture: eflu

Learn

[User's Guide](#) · [Wiki](#) · [Questions and Answers](#) · [Mailing Lists](#)

You are running Wireshark 2.2.6 (Git Rev Unknown from unknown).

No interfaces selected No Packets Profile: Default

Hinweis: Sie können Wireshark auf Ihrem privaten PC installieren. Im PC-Pool ist dies leider nicht möglich.
Den Screencast finden Sie im Materialordner.



HTTP im Einsatz (Wireshark)

Capturing from eth0 [Wireshark 1.12.1 (Git Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: http and ip.addr==192.252.144.35 Expression... Clear Apply Save

Source	Destination	Protocol	Length	Info
5817240 10.132.0.2	192.252.144.35	TCP	74	41928-80 [SYN] Seq=0 Win=28400 Len=0 MSS=1420 SACK_PERM=1
6736220 192.252.144.35	10.132.0.2	TCP	74	80-41928 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460
6736560 10.132.0.2	192.252.144.35	TCP	66	41928-80 [ACK] Seq=1 Ack=1 Win=28416 Len=0 TSval=633103 T
4381450 10.132.0.2	192.252.144.35	HTTP	497	GET /html/demo/hello.html HTTP/1.1
5289750 192.252.144.35	10.132.0.2	TCP	66	80-41928 [ACK] Seq=1 Ack=432 Win=30080 Len=0 TSval=259015
5305230 192.252.144.35	10.132.0.2	HTTP	1000	HTTP/1.1 200 OK (text/html)

Frame 39954: 497 bytes on wire (3976 bits), 497 bytes captured (3976 bits) on interface 0

Ethernet II, Src: 42:01:0a:84:00:02 (42:01:0a:84:00:02), Dst: 42:01:0a:84:00:01 (42:01:0a:84:00:01)

Internet Protocol Version 4, Src: 10.132.0.2 (10.132.0.2), Dst: 192.252.144.35 (192.252.144.35)

Transmission Control Protocol, Src Port: 41928 (41928), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 431

HyperText Transfer Protocol

 > GET /html/demo/hello.html HTTP/1.1\r\n

 Host: www.december.com\r\n

 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0\r\n

 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n

 Accept-Language: en-US,en;q=0.5\r\n

 Accept-Encoding: gzip, deflate\r\n

 Cookie: __utvc=6%7C15; __utuv=5accf604338be752005; __qca=P0-366999022-1523381765404\r\n

 Connection: keep-alive\r\n

0000 42 01 0a 84 00 01 42 01 0a 84 00 02 08 00 45 00 B.....B.E.

0010 01 e3 77 38 40 00 40 06 66 37 0a 84 00 02 c0 fc ..w@.@. f7.....

0020 90 23 a3 c8 00 50 8a ea df cf 4a fe dc 74 80 18 #...P.. .J.t..

0030 00 de 5d 7b 00 00 01 01 08 0a 00 09 ac bc 9a 62 .}{.....b

0040 97 66 47 45 54 20 2f 68 74 6d 6c 2f 64 65 6d 6f .fGET /h tml/demo

eth0: <live capture in progress> F... Packets: 87739 - Displayed: 10 (0.0%) Profile: Default

Follow TCP Stream (tcp.stream eq 112)

Stream Content

```
GET /html/demo/hello.html HTTP/1.1
Host: www.december.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: __utvc=6%7C15; __utuv=5accf604338be752005; __qca=P0-366999022-1523381765404
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Tue, 10 Apr 2018 17:43:56 GMT
Server: Apache
Last-Modified: Thu, 29 Jun 2006 17:05:33 GMT
ETag: "299-4175ff09d1140"
Accept-Ranges: bytes
Content-Length: 665
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

Entire conversation (1365 bytes)

Find Save As Print ASCII EBCDIC Hex Dump C Arrays Raw

Help Filter Out This Stream Close

Hello World Demonstrati x +

← → ⌂ ⌂ www.december.com/html/demo/hello.html ... ⌂ ⌂ ⌂

Hello, World!

This is a minimal "hello world" HTML document. It demonstrates the basic structure of an HTML file and anchors.

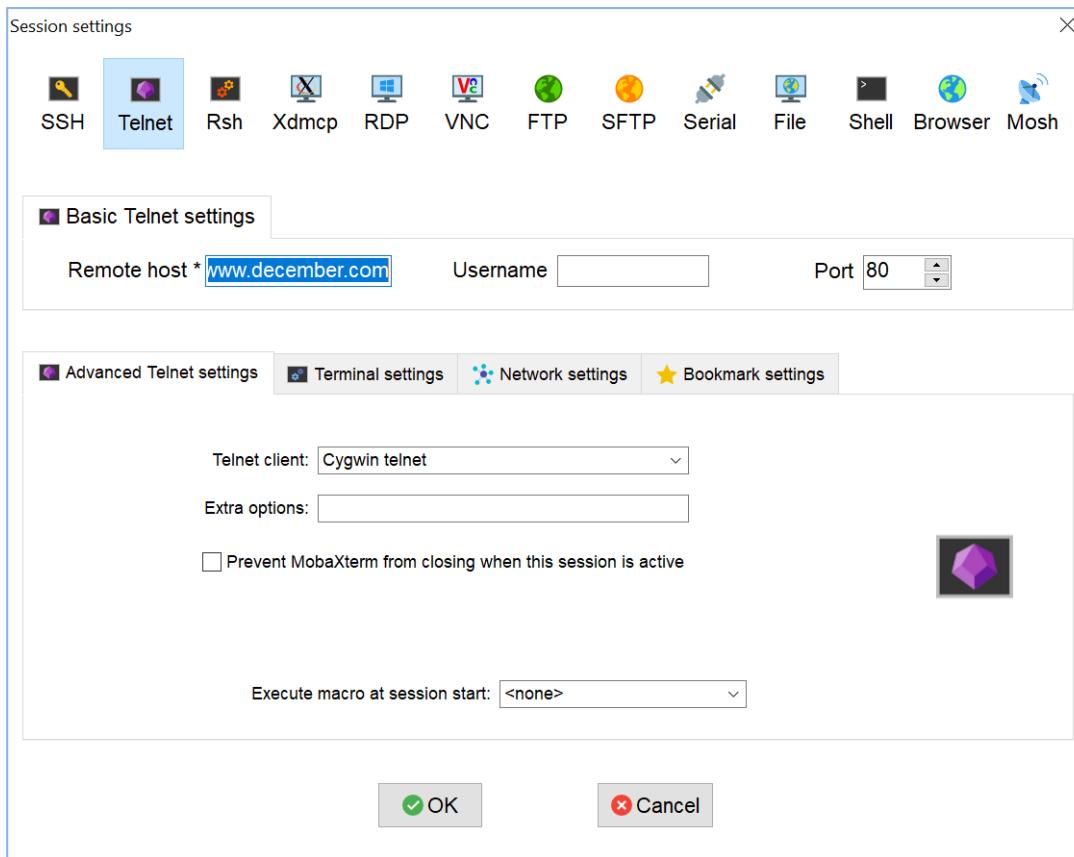
For more information, see the HTML Station at: <http://www.december.com/html/>

© John December (john@december.com) / 2001-04-06

Hinweis: Sie können Wireshark auf Ihrem privaten PC installieren. Im PC-Pool ist dies leider nicht möglich.
Den Screencast finden Sie im Materialordner.

Übung

1. Öffnen einer Telnet-Verbindung zu einem Webserver:
Wir nutzen dafür MobaXTerm. Legen Sie folgende Session an:



Übung

2. Öffnen einer Telnet-Verbindung zu einem Webserver:

telnet www.december.com 80

(dies macht MobaXTerm für Sie,
wenn Sie auf connect klicken!)

Öffnet eine TCP-Verbindung zu Port 80
(Standard-HTTP-Server-Port) auf december.com.
Alles, was jetzt eingegeben wird, wird an
Port 80 auf december.com geschickt

2. Eingeben eines HTTP-GET-Requests:

**GET /html/demo/hello.html HTTP/1.1
Host: www.december.com**

Durch diese Eingabe (am Ende zweimal
Return tippen!) wird ein minimaler
(aber vollständiger) GET-Request
an den HTTP-Server geschickt

Wenn Sie sich vertippen kann es u.U. nicht funktionieren. Pasen Sie
gut auf die korrekte Schreibweise auf.

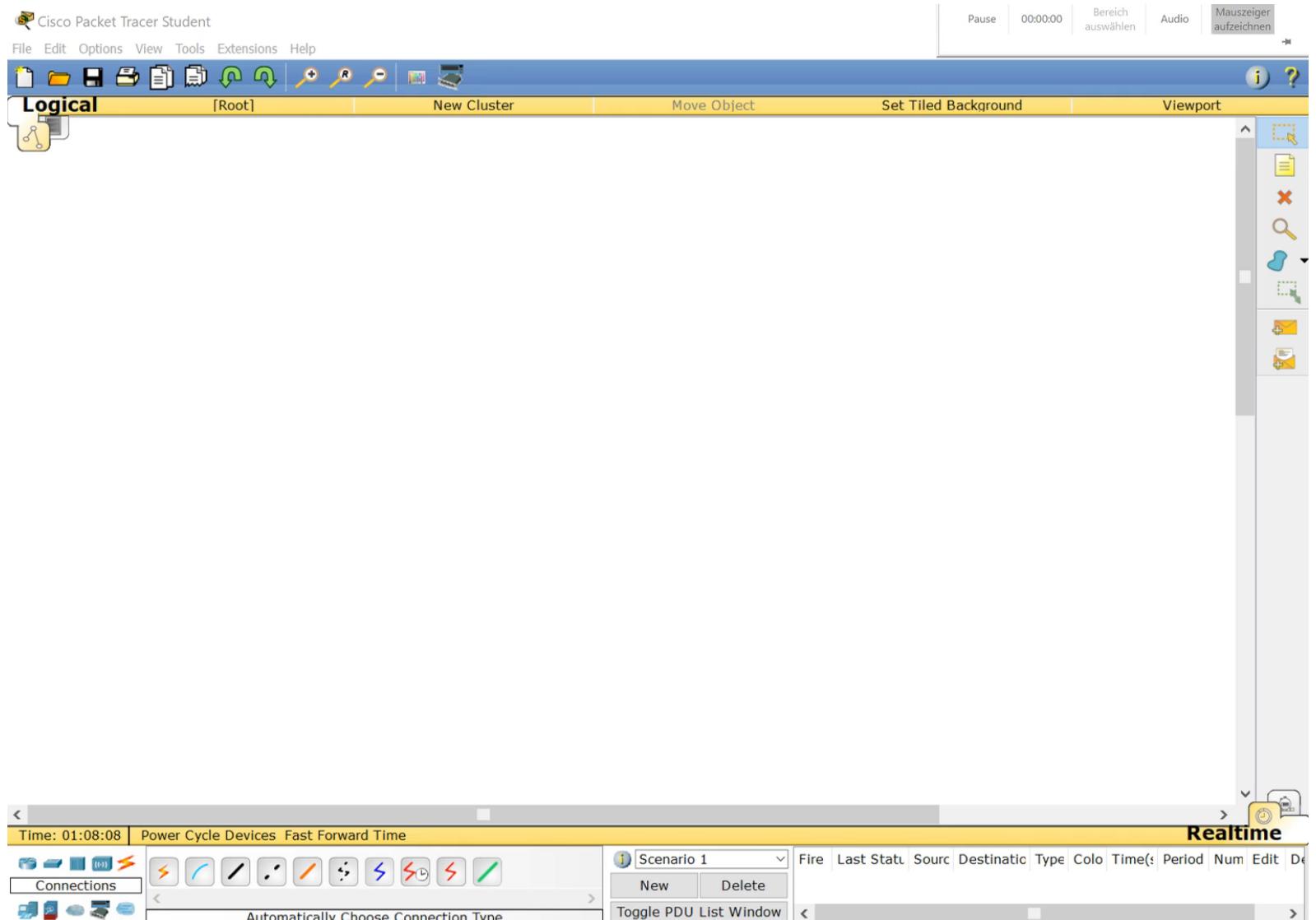
3. Antwort analysieren

Übung

HTTP im Einsatz (Cisco Packet Tracer)

Bitte setzen Sie im Packet-Tracer folgende Umgebung auf und nutzen Sie die Simulation, um das HTTP-Protokoll nachzuvollziehen.

Hinweis: Der Cisco Packet Tracer steht im PC-Pool zur Verfügung. Den Screencast finden Sie im Materialordner



Zustand halten: Cookies

Viele Websites verwenden Cookies

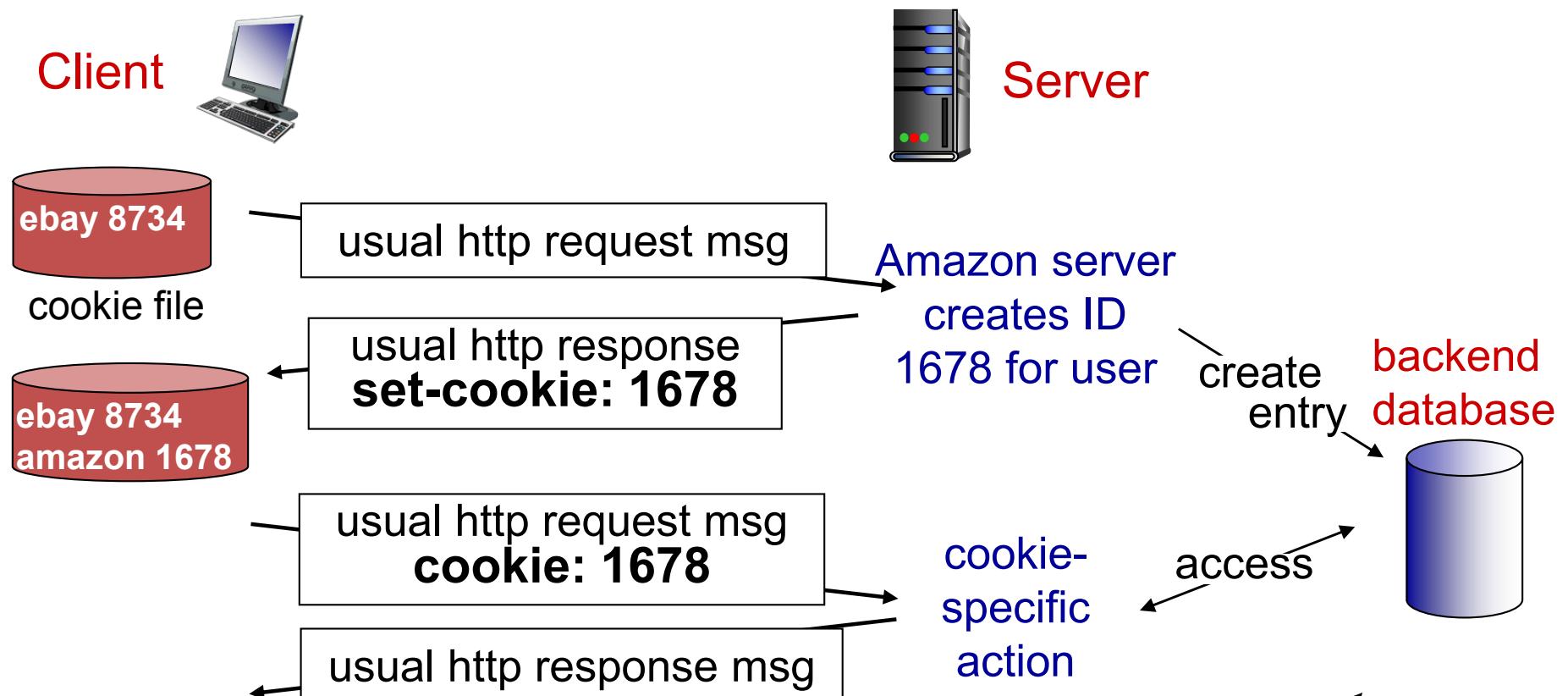
Vier Bestandteile:

- 1) Cookie-Kopfzeile in der HTTP-Response-Nachricht
- 2) Cookie-Kopfzeile in der HTTP-Request-Nachricht
- 3) Cookie-Datei, die auf dem Rechner des Anwenders angelegt und vom Browser verwaltet wird
- 4) Backend-Datenbank auf dem Webserver

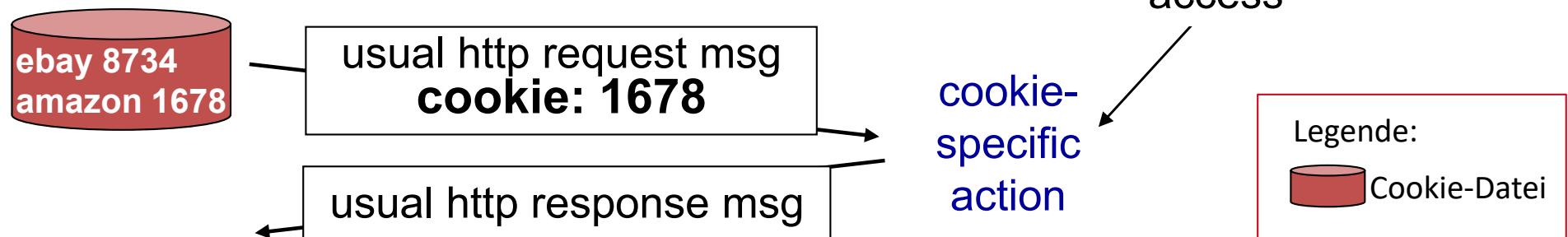
Beispiel:

- Susanne greift auf das Internet immer von ihrem PC aus zu
- Sie besucht eine bestimmte E-Commerce-Seite zum ersten Mal
- Wenn der initiale HTTP-Request beim Server ankommt, generiert er:
 - Eine eindeutige ID
 - Einen Eintrag in die Backend-Datenbank für diese ID

Zustand halten: Cookies



Eine Woche später:



Legende:
Cookie-Datei

Cookies

Einsatz von Cookies:

-
-
-
-

Cookies und Privatsphäre:

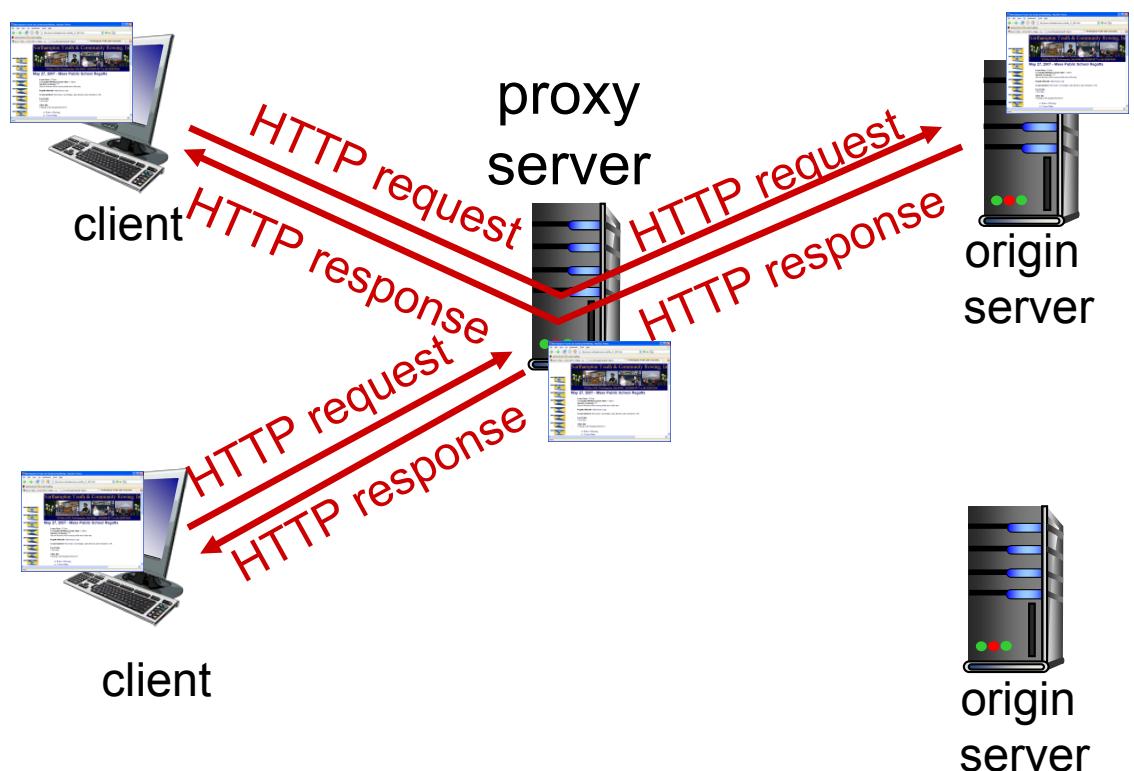
- Cookies ermöglichen es Websites, viel über den Anwender zu lernen:
 - Formulareingaben (Name, E-Mail-Adresse)
 - Besuchte Seiten

- Cookies: HTTP-Nachrichten beinhalten den Zustand

Web-Caches (Proxyserver)

Ziel: Client-Anfragen ohne den ursprünglichen Webserver beantworten

- Benutzer konfiguriert Browser: Webzugriff über einen Cache
- Browser sendet alle HTTP-Requests an den Cache
 - Objekt im Cache: Cache gibt Objekt zurück
 - Sonst: Cache fragt das Objekt vom ursprünglichen Server an und gibt es dann an den Client zurück



Mehr zum Thema Web-Caching

- Cache arbeitet als Client UND als Server
 - Server: aus Sicht des ursprünglich anfragenden Hosts (Clients)
 - Client: für den Ziel-Server, der angefragt wird
- Üblicherweise ist der Cache beim ISP installiert (Universität, Firma, ISP für Privathaushalte)

Warum Web-Caching?

-
-
- Bei Existenz vieler Caches: „arme“ Inhaltsanbieter können ihre Inhalte gut verbreiten (Ähnliches kann durch P2P-Filesharing erreicht werden)

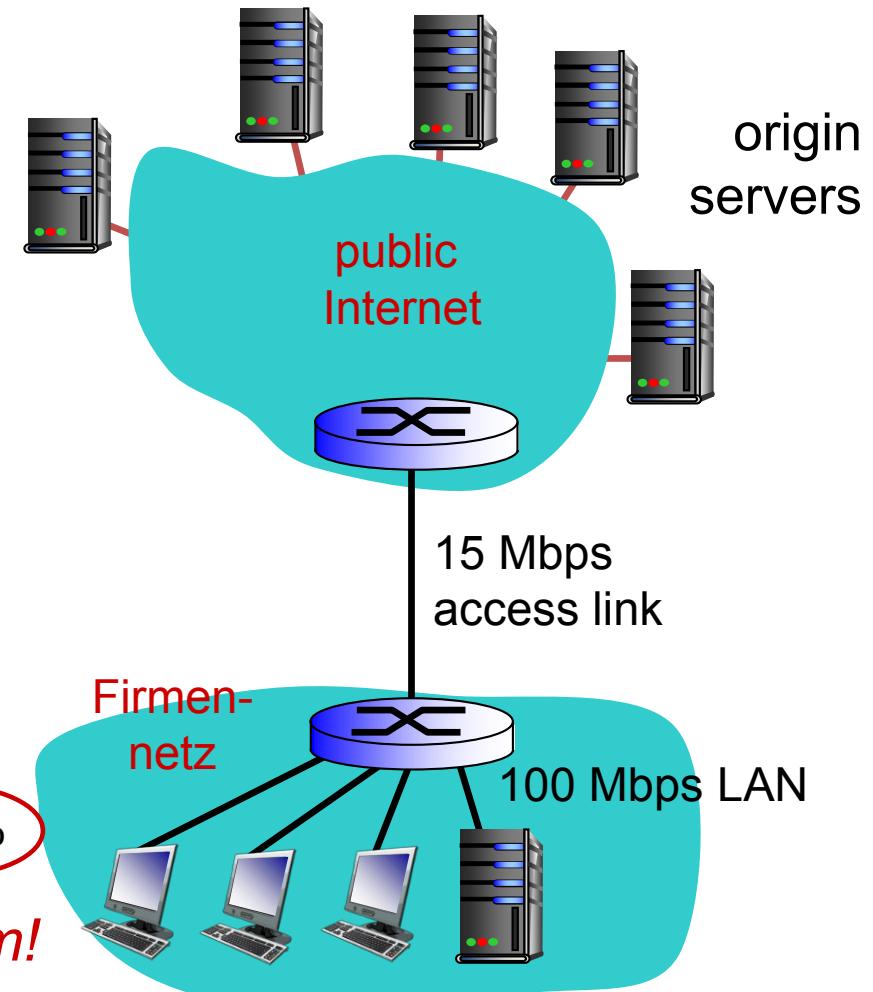
Beispiel für Web-Caching

■ Annahmen

- Durchschnittliche Größe eines Objektes = 1 MBit
- Durch. Rate von Anfragen aller Webbrowser der Firma = 15/s
- RTT: zum Server und zurück = 2 sec
- Zugangs-Link Datenrate: 15 Mbps

■ Resultat

- Auslastung des LAN = 15%
$$= (15\text{req./s}) \cdot (1\text{MBit/requ.}) / (100\text{Mbit/s})$$
- Auslastung der Zugangsleitung = 100%
$$= (15\text{req./s}) \cdot (1\text{MBit/requ.}) / (15\text{Mbit/s})$$
- Verzögerung = Internet + Zugangsleitung + LAN
Problem!
- = 2s + einige Minuten + μ Sekunden



Beispiel für Web-Caching: stärkerer Zugangs-Link

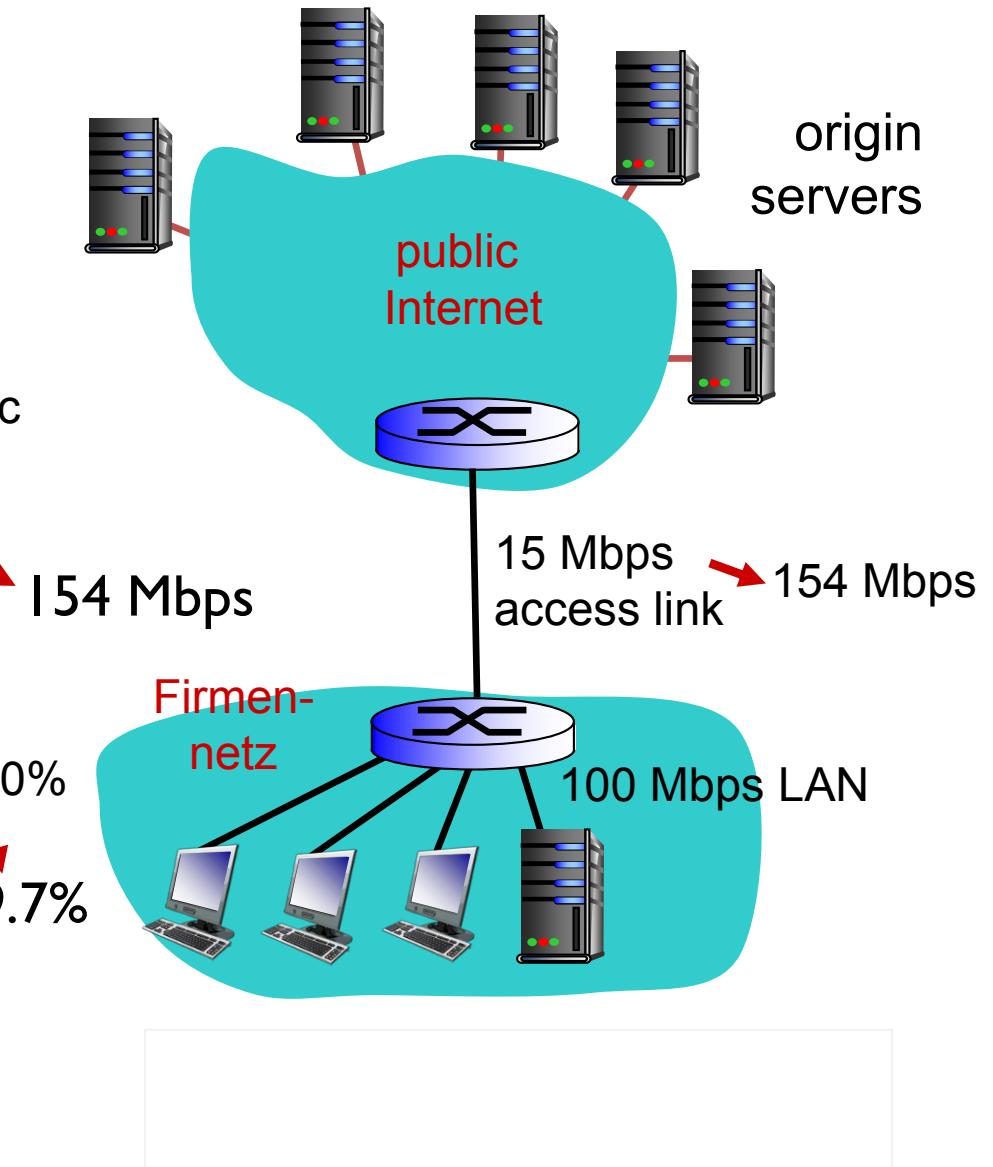
■ Annahmen

- Durchschnittliche Größe eines Objektes = 1 MBit
- Durch. Rate von Anfragen aller Webbrowser der Firma = 15/s
- RTT: zum Server und zurück = 2 sec
- Zugangs-Link Datenrate: 15 Mbps

■ Resultat

- Auslastung des LAN = 15%
- Auslastung der Zugangsleitung = 100%
- Verzögerung = Internet + Zugangsleitung + LAN
- = 2s + Minuten + mSek

mSek



Beispiel für Web-Caching: lokaler Cache

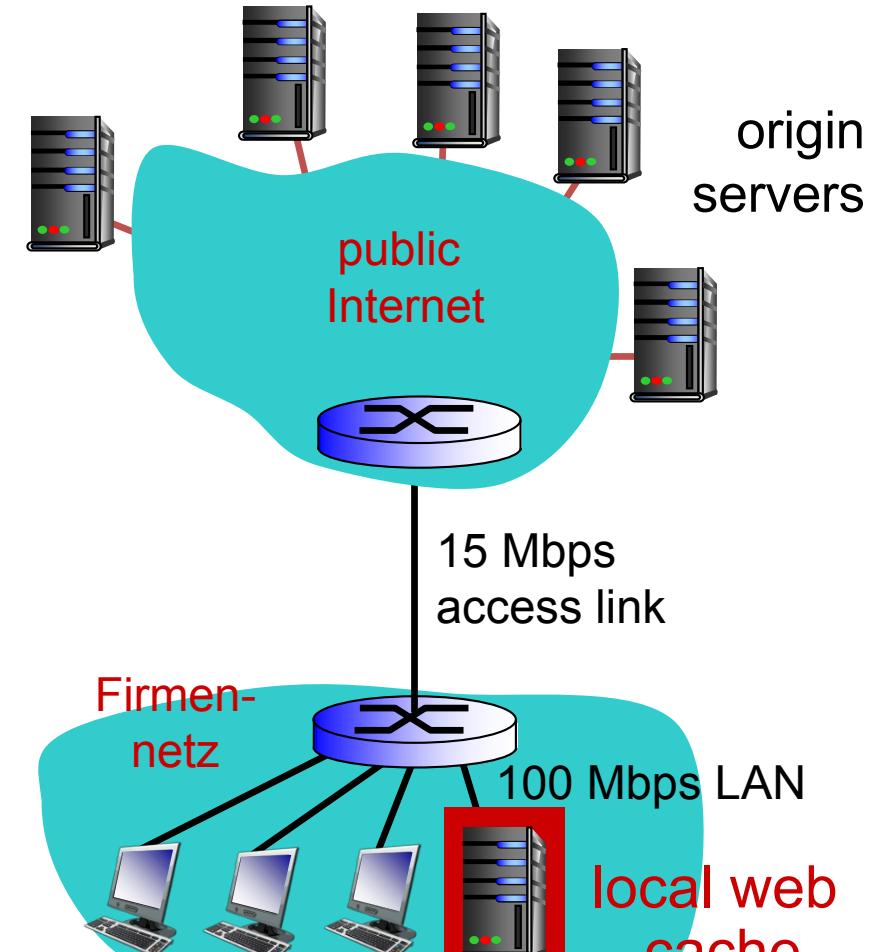
■ Annahmen

- Durchschnittliche Größe eines Objektes = 1 MBit
- Durch. Rate von Anfragen aller Webbrowser der Firma = 15/s
- RTT: zum Server und zurück = 2 sec
- Zugangs-Link Datenrate: 15 Mbps

■ Resultat

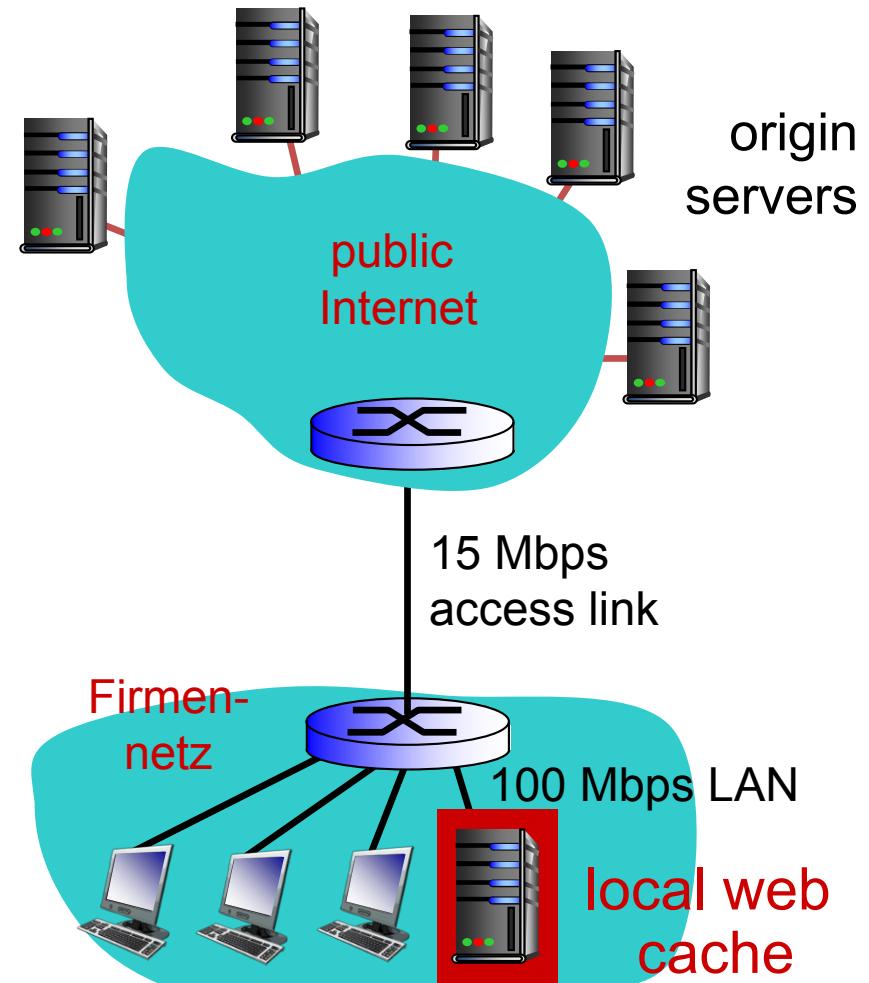
- Auslastung des LAN = 15%
- Auslastung der Zugangsleitung = ?
- Verzögerung = ?

Wie berechnen wir die Auslastung und Verzögerung?



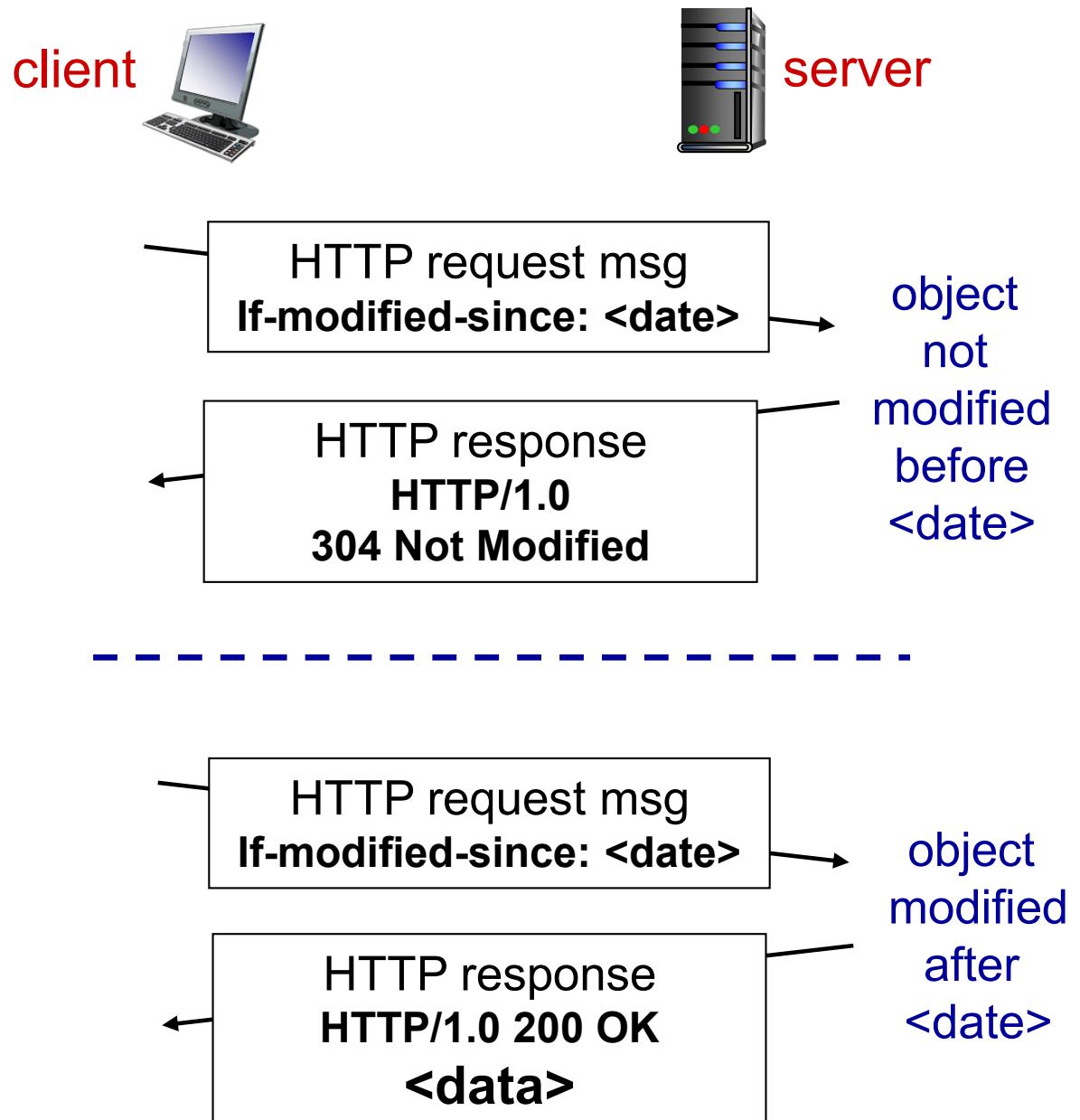
Beispiel für Web-Caching

- Annahme: Cache-Trefferrate = 0,4
- Resultat
 - 40% der Requests werden nahezu sofort (durch Cache) beantwortet
 - 60% der Requests werden durch normale Webserver beantwortet
- Auslastung der Zugangsleitung auf 60% reduziert
- Verzögerungen werden vernachlässigbar klein (z.B. 10 mSek)
 - Verzögerung = Internet + Zugangsleitung + LAN = $0,6 * (2,01\text{s}) + 0,4 * \text{Millisekunden} < 1,4\text{s}$
 - Weniger Verzögerung (und billiger) als 154Mbps-Zugangslink!

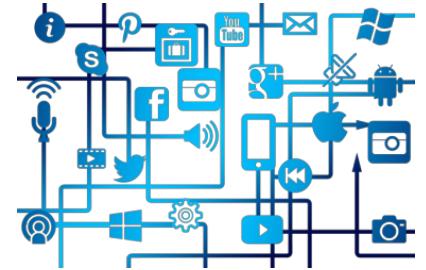


Bedingtes GET

- **Ziel:** Objekt nicht senden, wenn der Cache eine aktuelle Version besitzt
- Cache: Angeben des Änderungsdatums der gespeicherten Version (kann der HTTP-Response entnommen werden) durch folgende Zeile im HTTP-Request:
If-modified-since:
 <date>
- Server: Response enthält kein Objekt, wenn die Version im Cache aktuell ist:
HTTP/1.0 304 Not Modified



Begleitmaterialien



- https://www.w3schools.com/tags/ref_httpmethods.asp

w3schools.com THE WORLD

HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP JQUERY ANGULAR W3.CSS XML MORE ▾ REFERENCES

HTML Reference

- HTML by Alphabet
- HTML by Category
- HTML Attributes
- HTML Global Attributes
- HTML Events
- HTML Colors
- HTML Canvas
- HTML Audio/Video
- HTML Character Sets
- HTML Doctypes
- HTML URL Encode
- HTML Language Codes
- HTML Country Codes
- HTTP Messages
- HTTP Methods**
- PX to EM Converter
- Keyboard Shortcuts

HTML Tags

- <!-->
- <!DOCTYPE>
- <a>
- <abbr>
- <acronym>
- <address>
- <applet>
- <area>
- <article>
- <aside>
- <audio>
-
- <base>

HTTP Methods: GET vs. POST

◀ Previous Next ▶

The two most used HTTP methods are: GET and POST.

What is HTTP?

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers. HTTP works as a request-response protocol between a client and server. A web browser may be the client, and an application on a computer that hosts a web site may be the server. Example: A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

Two HTTP Request Methods: GET and POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

The GET Method

Note that the query string (name/value pairs) is sent in the URL of a GET request:

```
/test/demo_form.php?name1=value1&name2=value2
```

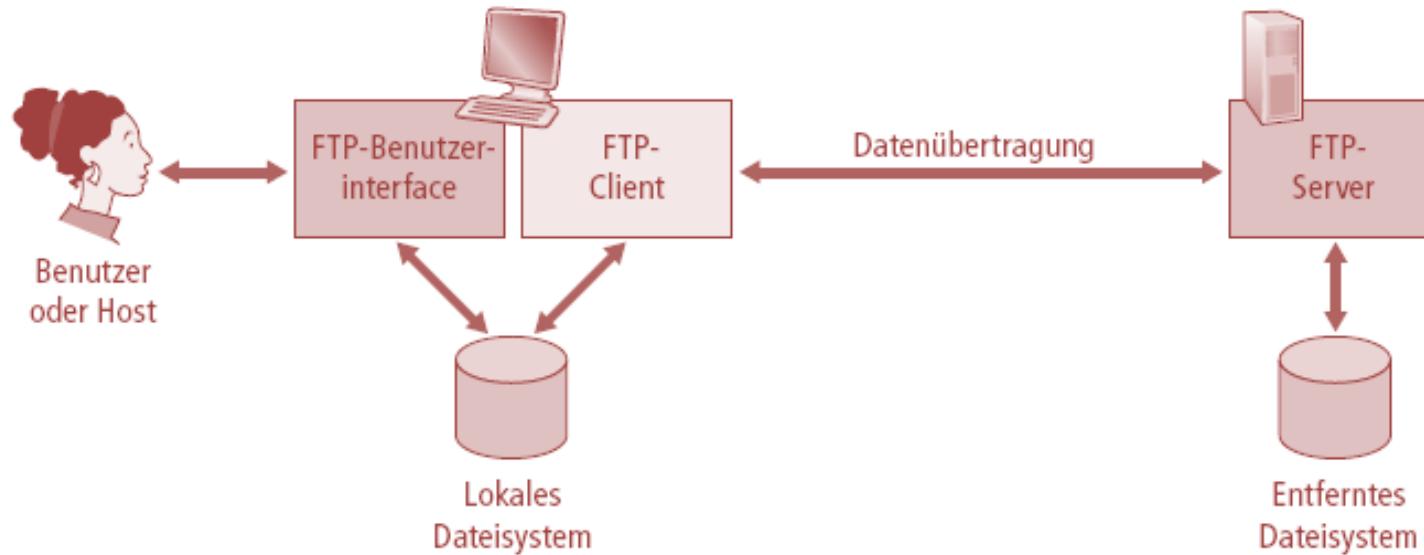
Some other notes on GET requests:

- <http://formsmarts.com/html-form-example> kann z.B. mit Wireshark genutzt werden, um eine POST-Nachricht zu verfolgen.

Agenda

- Grundlagen von Netzwerk-Applikationen**
- Web und HTTP**
- FTP und E-Mail**
- Domain Name System**
- Peer-to-Peer Anwendungen**
- Socket-Programmierung**
- Zusammenfassung und Ausblick**

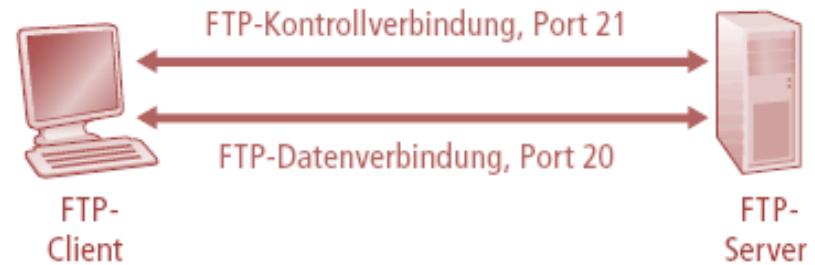
FTP: Das File Transfer Protocol



- Übertragen einer Datei von/zu einem entfernten Rechner
- Client/Server-Modell
 - *Client*: Seite, die den Transfer initiiert (vom oder zum entfernten Rechner)
 - *Server*: entfernter Rechner
- FTP: RFC 959
- FTP-Server: TCP Port 21

FTP: Verschiedene Kanäle für Kontroll- und Datenverbindungen

- FTP-Client kontaktiert den FTP-Server auf Port 21, wobei er TCP als Transportprotokoll nutzt.
- Client autorisiert sich über die Kontrollverbindung.
- Client betrachtet das entfernte Verzeichnis, indem er Kommandos über die Kontrollverbindung schickt.
- Empfängt der Server ein Kommando für eine Dateiübertragung, öffnet der Server eine zweite TCP-Verbindung zum Client (Datenverbindung)
- Nach der Übertragung einer Datei schließt der Server die Datenverbindung



- Wird eine weitere Datei angefordert, öffnet der Server eine zweite TCP-Datenverbindung
- Der FTP-Server hält „Statusinformationen“ vor: aktuelles Verzeichnis, frühere Authentifizierung

FTP-Kommandos, Antworten

Kommandos:

- Als ASCII-Text über die Kontrollverbindung
- **USER *username***
- **PASS *password***
- **LIST** gibt eine Liste der Dateien im aktuellen Verzeichnis zurück
- **RETR *filename*** lädt eine entfernte Datei auf den lokalen Rechner
- **STOR *filename*** überträgt eine lokale Datei auf den entfernten Rechner

Antworten:

- Statuscode und Erläuterung (wie bei HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

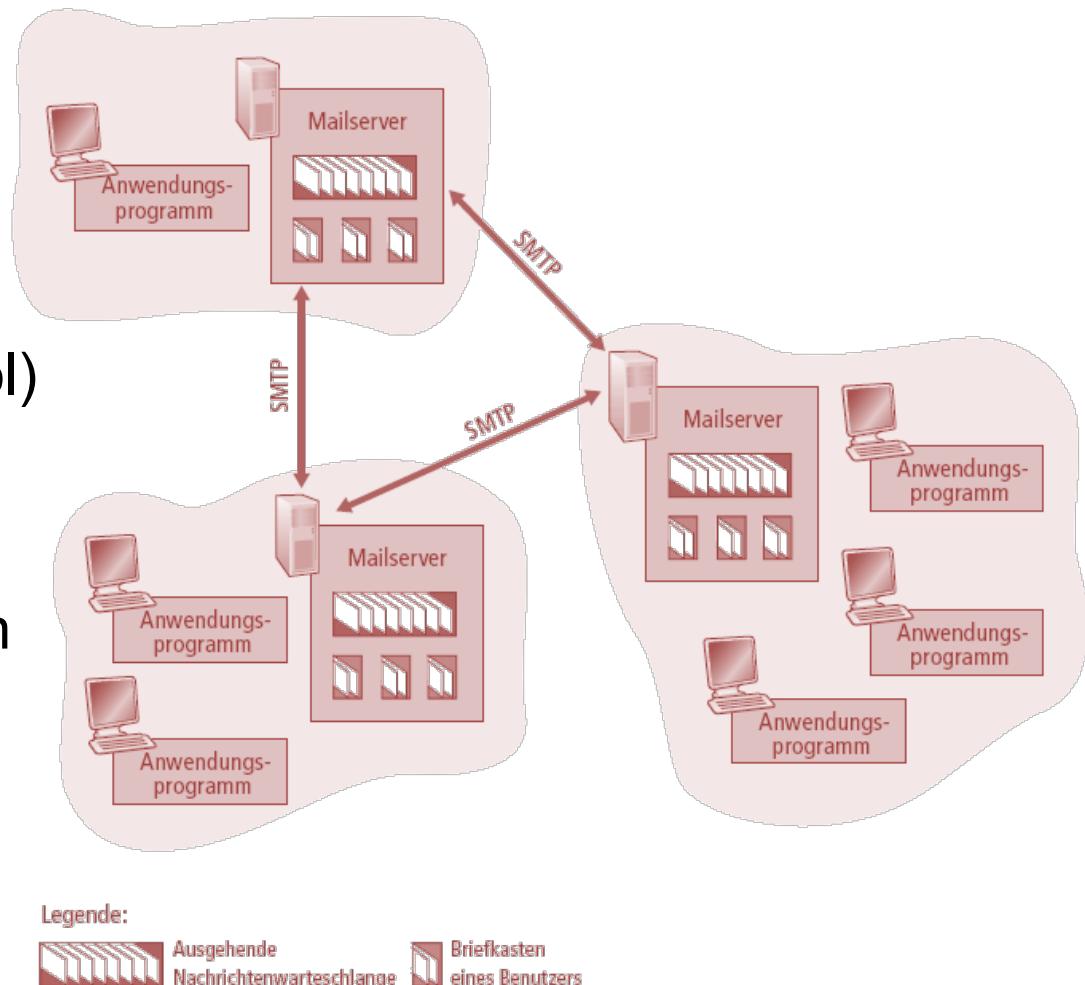
Electronic Mail

Drei Hauptbestandteile:

- Anwendungsprogramm
- Mailserver
- Übertragungprotokoll: SMTP
(Simple Mail Transfer Protocol)

Anwendungsprogramm:

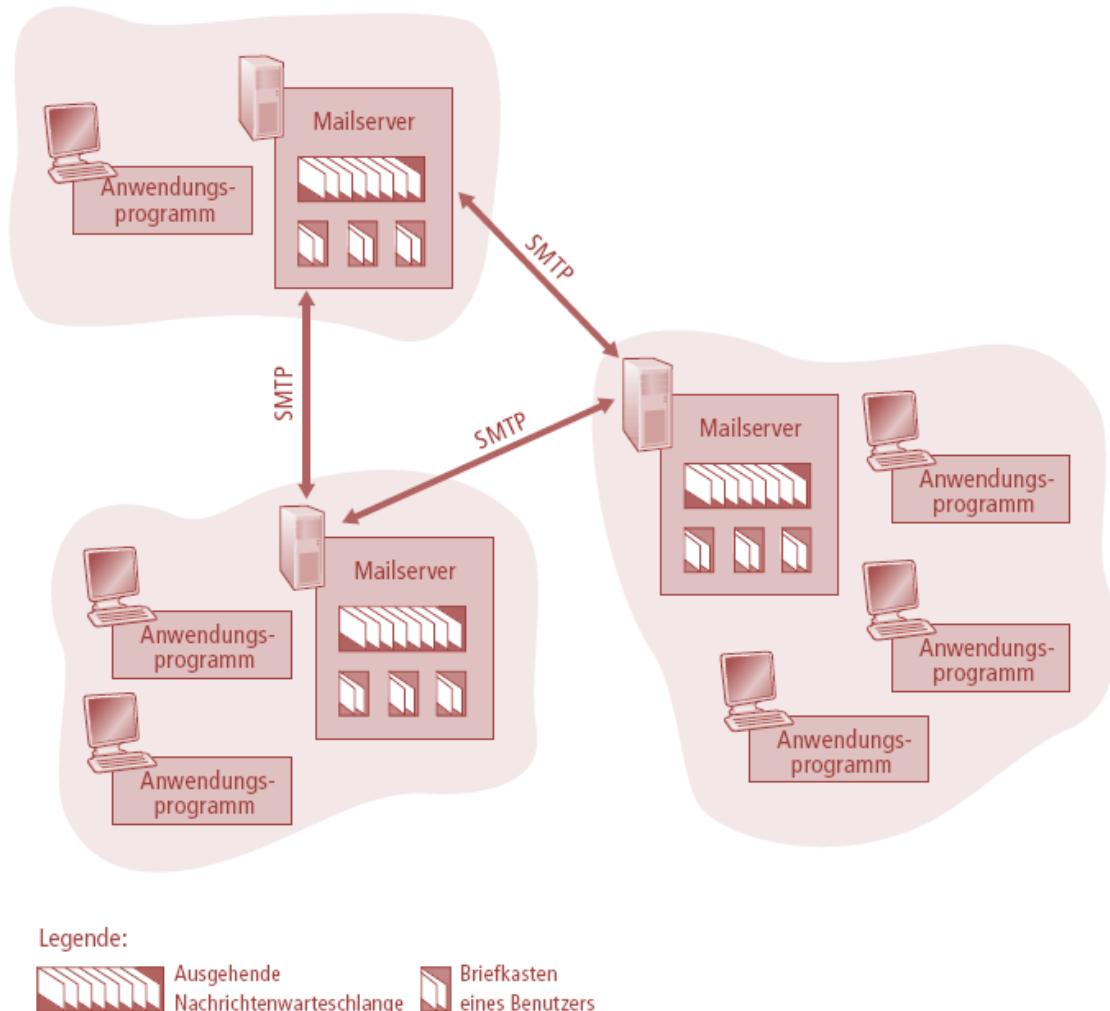
- Auch “Mail Reader”
- Erstellen, Editieren, Lesen von E-Mail-Nachrichten
- z.B. Outlook, Thunderbird, iPhone mail client
- Eingehende und ausgehende Nachrichten werden auf dem Server gespeichert



Electronic Mail: Mailserver

Mailserver

- Die **Mailbox** enthält die eingehenden Nachrichten eines Benutzers
- Die **Warteschlange** für ausgehende Nachrichten enthält die noch zu sendenden E-Mail-Nachrichten
- **SMTP** wird verwendet, um Nachrichten zwischen Mailservern auszutauschen
 - Client: sender Mailserver
 - Server: empfangender Mailserver

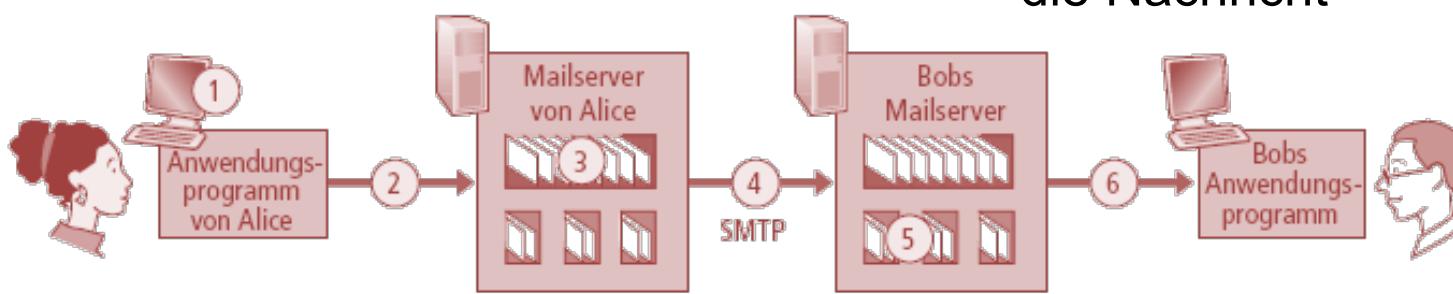


Electronic Mail: SMTP [RFC 2821]

- Ursprüngliche Version aus dem Jahr 1982
- TCP wird zum zuverlässigen Transport von E-Mail-Nachrichten vom Client zum Server (Port 25) verwendet
- Direkter Transport der Nachrichten: vom sendenden Server zum empfangenden Server
- Drei Phasen des Mail-Versands: analog zu einer Unterhaltung
 - Handshaking (Begrüßung)
 - Transfer of Messages (Austausch von Informationen)
 - Closure (Verabschiedung)
- Interaktion basiert auf dem Austausch von Befehlen (Commands) und Antworten (Responses) – genau wie HTTP und FTP
 - **Command:** ASCII-Text
 - **Response:** Statuscode und Bezeichnung
- Nachrichten müssen in 7-Bit-ASCII kodiert sein

Beispiel: Alice schickt Bob eine Nachricht

- 1) Alice verwendet Anwendungsprogramm, um Nachricht an bob@hs-lu.de zu erstellen
- 2) Alices Anwendung versendet die Nachricht an ihren Mail-Server; Nachricht wird in der Warteschlange gespeichert
- 3) Alices Mailserver öffnet als Client eine TCP-Verbindung zu Bobs Mailserver
- 4) SMTP-Client versendet die Nachricht von Alice über die TCP-Verbindung
- 5) Bobs Mailserver empfängt die Nachricht von Alices Mailserver und speichert diese in Bobs Mailbox
- 6) Bob verwendet (irgendwann) sein Anwendungsprogramm und liest die Nachricht



Legende:



Nachrichtenwarteschlange



Briefkasten eines Benutzers

Beispiel für eine SMTP-Sitzung

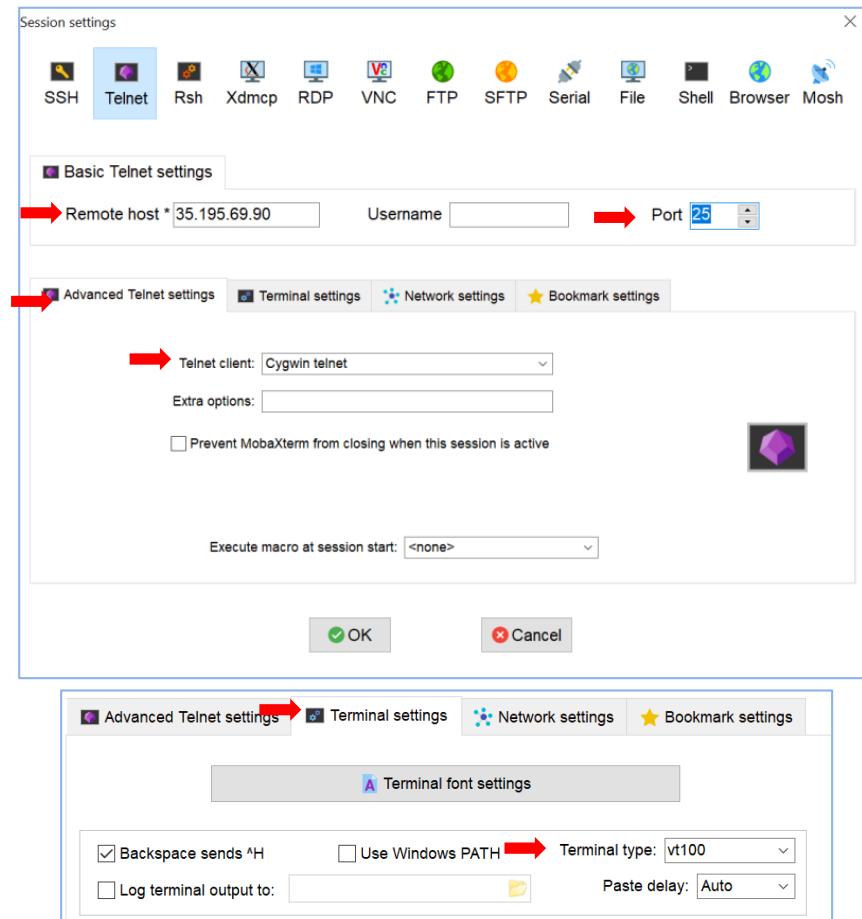
```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Übung

SMTP im Einsatz (telnet)

Bitte nutzen Sie telnet wie folgt, um eine Beispiel-Email zu senden:

- Legen Sie eine telnet-Session in MobaXTerm an
 - Ziel-Host: 35.195.17.212 (kann sich geändert haben!)
 - Port: 25
 - Advanced Settings: cygwin telnet
 - Terminal Settings: vt100
- Wenn die Verbindung geöffnet ist, geben Sie die Kommandos auf der nächsten Seite jeweils ein und bestätigen mit Enter pro Zeile



Übung

SMTP im Einsatz (telnet)

Senden Sie nun eine E-Mail nach dem SMTP-Standard

HELO hs-lu.de
MAIL FROM: abc@hs-lu.de
RCPT TO: xyz@hs-lu.de
DATA
Dann der E-Mail Header:

- Subject: abc
- From: <abc@hs-lu.de>
- To: <xyz@hs-lu.de>
- Eine Leerzeile
- Dann Ihre Nachricht
- Am Ende ein . (Punkt) und Enter

QUIT

```
Trying 35.195.69.90...
Connected to 35.195.69.90.
Escape character is '^]'.
220 bw332.c.bw332-198707.internal Python SMTP proxy version 0.2
HELO hs-lu.de
250 bw332.c.bw332-198707.internal
MAIL FROM: abc@hs-lu.de
250 Ok
RCPT T0: xyz@hs-lu.de
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: this is an email subject
From: <abc@hs-lu.de>
To: <xyz@hs-lu.de>

This is the content of the email

Bye bye

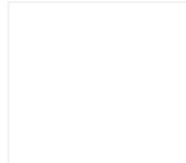
.

250 Ok
QUIT
```

SMTP: Zusammenfassung

- SMTP verwendet eine dauerhafte Verbindung für den Versand von E-Mails
- SMTP verwendet sowohl für Header als auch für Daten 7-Bit-ASCII
- Ein SMTP-Server verwendet CRLF.CRLF, um das Ende einer Nachricht zu signalisieren

Vergleich mit HTTP:

- HTTP: 
- SMTP:
- Beide interagieren mittels ASCII-Befehl/Antwort-Paaren sowie Statuscodes
- HTTP: Jedes Objekt ist in einer eigenen Antwortnachricht gekapselt
- SMTP: Mehrere Objekte können in einer Nachricht (multipart msg) versendet werden

Format einer E-Mail-Nachricht

SMTP: Protokoll für den Austausch von E-Mail-Nachrichten

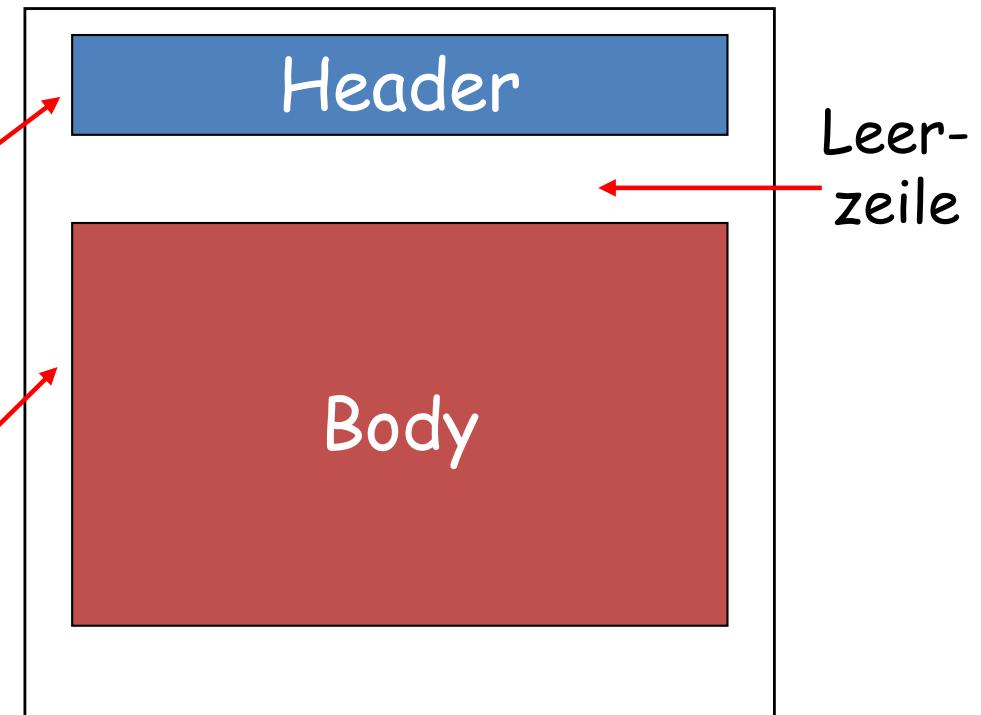
RFC 822: Standard für Textnachrichten:

- Header-Zeilen, z.B.

- To:
 - From:
 - Subject:

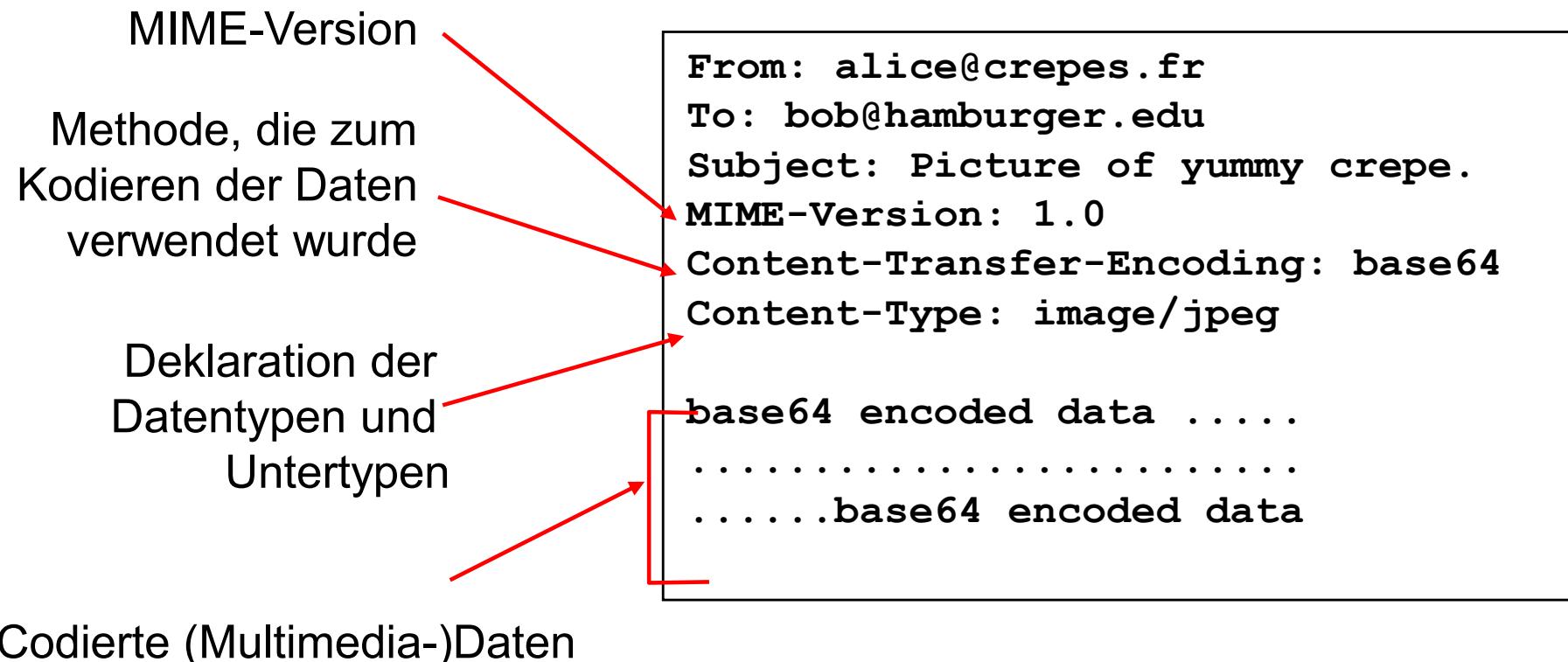
Keine SMTP-Befehle!

- Body
 - Die eigentliche Nachricht in ASCII



Nachrichtenformat: Multimedia-Erweiterung

- MIME: Multimedia Mail Extension, RFC 2045, 2056
- Zusätzliche Zeilen im Header deklarieren den MIME-Typ des Inhaltes



Datentypen in MIME

Text

Beispiele für Subtypen:

- **plain**
- **html**

Bilder

Beispiele für Subtypen:

- **jpeg**
- **gif**
- **png**

Audio

Beispiele für Subtypen:

- **basic** (8-bit mu-law encoded),
- **32kadpcm** (32 kbps coding)

Video

Beispiele für Subtypen:

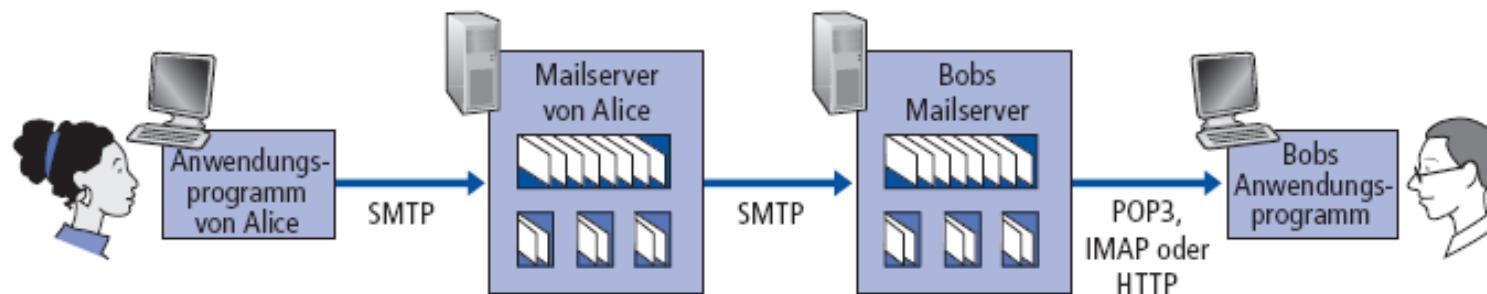
- **mpeg**
- **quicktime**

Anwendungen

- Daten müssen von der Anwendung vor der Wiedergabe interpretiert werden
- Beispiele für Subtypen:
msword, **octet-stream**

Mail-Zugriffsprotokolle

- SMTP: Zustellung/Speicherung auf dem Mailserver des Empfängers
- Zugriffsprotokoll: Protokolle zum Zugriff auf E-Mails
- Abruf vom Server
 - POP: Post Office Protocol [RFC 1939]
 - Autorisierung (Anwendung <--> Server) und Zugriff/Download
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - Größere Funktionalität (deutlich komplexer)
 - Manipulation der auf dem Server gespeicherten Nachrichten
 - HTTP: Hotmail, Yahoo!Mail etc.



POP3-Protokoll

Autorisierungsphase:

- Befehle des Clients:
 - **user**: Benutzername
 - **pass**: Passwort
- Antworten des Servers:
 - **+OK**
 - **-ERR**

Transaktionsphase:

- **list**: Nachrichten auflisten
- **retr**: Nachrichten herunterladen
- **dele**: Löschen von Nachrichten
- **Quit**: Ende

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 und IMAP

Mehr zu POP3

- Vorheriges Beispiel nutzte den “*Download-and-Delete*”-Modus, d.h., andere E-Mail-Clients haben danach keine Möglichkeit mehr, die Mails zu lesen
- Der “*Download-and-Keep*”-Modus ermöglicht den reinen Lesezugriff auf Nachrichten, d.h., verschiedene Clients haben Zugriff
- POP3 ist zustandslos zwischen einzelnen Sitzungen

IMAP

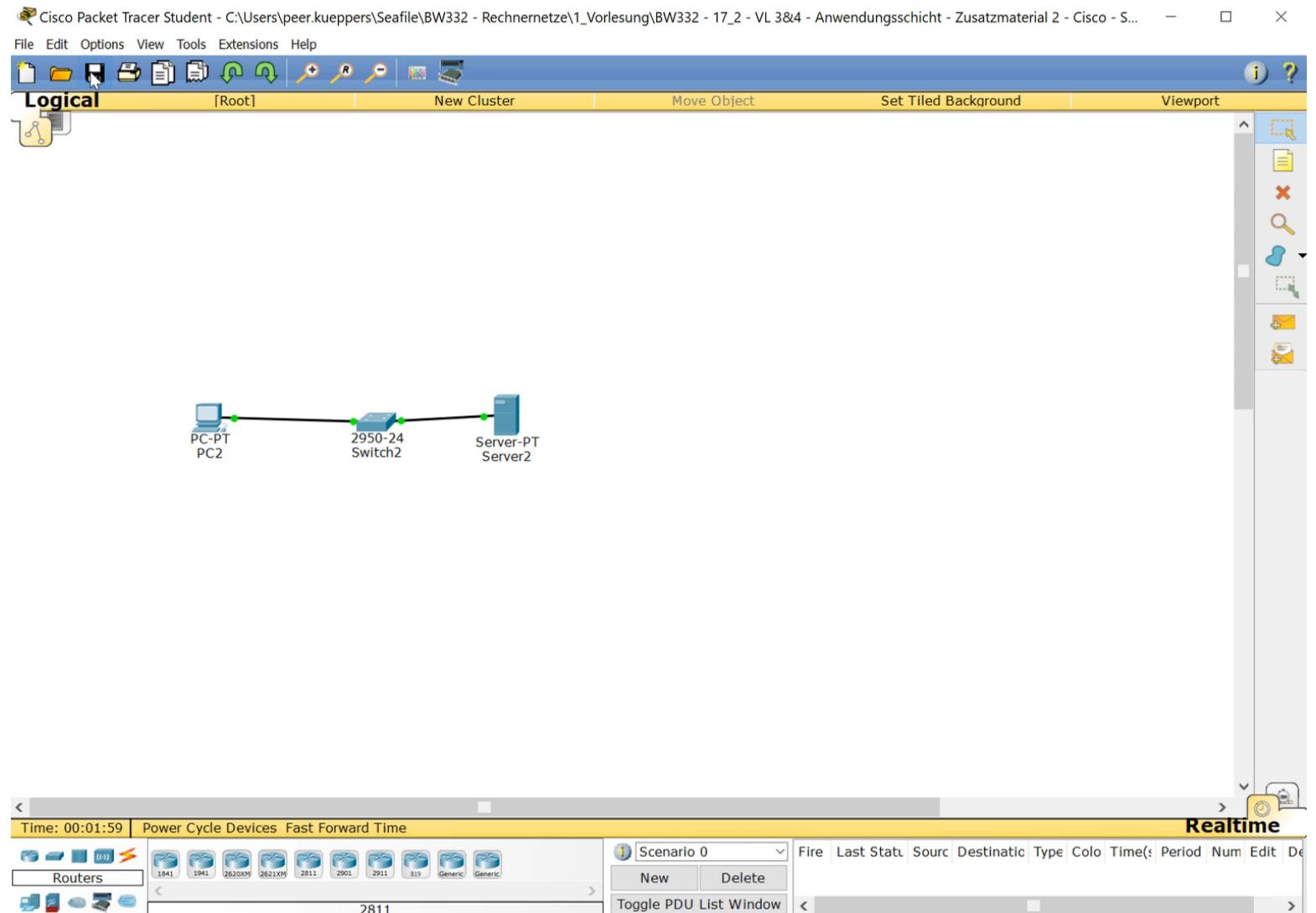
- Alle Nachrichten bleiben an einem Ort: auf dem Server
- Nachrichten können auf dem Server in Ordner verwaltet werden
- IMAP bewahrt den Zustand zwischen einzelnen Sitzungen:
 - Namen von Ordner und Zuordnung von Nachrichtennummer und Ordnername bleiben erhalten

Übung

SMTP im Einsatz (Cisco Packet Tracer)

Bitte setzen Sie im Packet-Tracer folgende Umgebung auf und nutzen Sie die Simulation, um das SMTP-Protokoll nachzuvollziehen.

Hinweis: Der Cisco Packet Tracer steht im PC-Pool zur Verfügung. Den Screencast finden Sie im Materialordner



Agenda

- Grundlagen von Netzwerk-Applikationen**
- Web und HTTP**
- FTP und E-Mail**
- Domain Name System**
- Peer-to-Peer Anwendungen**
- Socket-Programmierung**
- Zusammenfassung und Ausblick**

DNS: Domain Name System

Menschen: verschiedene Identifikationsmechanismen

- Name, Ausweisnummer

Internet-Hosts, Router:

- IP-Adresse (32 Bit) – für die Adressierung in Paketen
- “Name”, z.B., www.yahoo.com – von Menschen verwendet

Wie findet die Abbildung zwischen IP-Adressen und Namen statt?

Domain Name System:

- Verteilte Datenbank, implementiert eine Hierarchie von Nameservern
- Protokoll der Anwendungsschicht, wird von Hosts verwendet, um Namen aufzulösen (Abbildung zwischen Adresse und Name)
 - zentrale Internetfunktion, implementiert als Protokoll der Anwendungsschicht
 - Grund: Komplexität liegt dadurch nur am Rand des Netzwerkes!

DNS: Domain Name System

DNS-Dienste

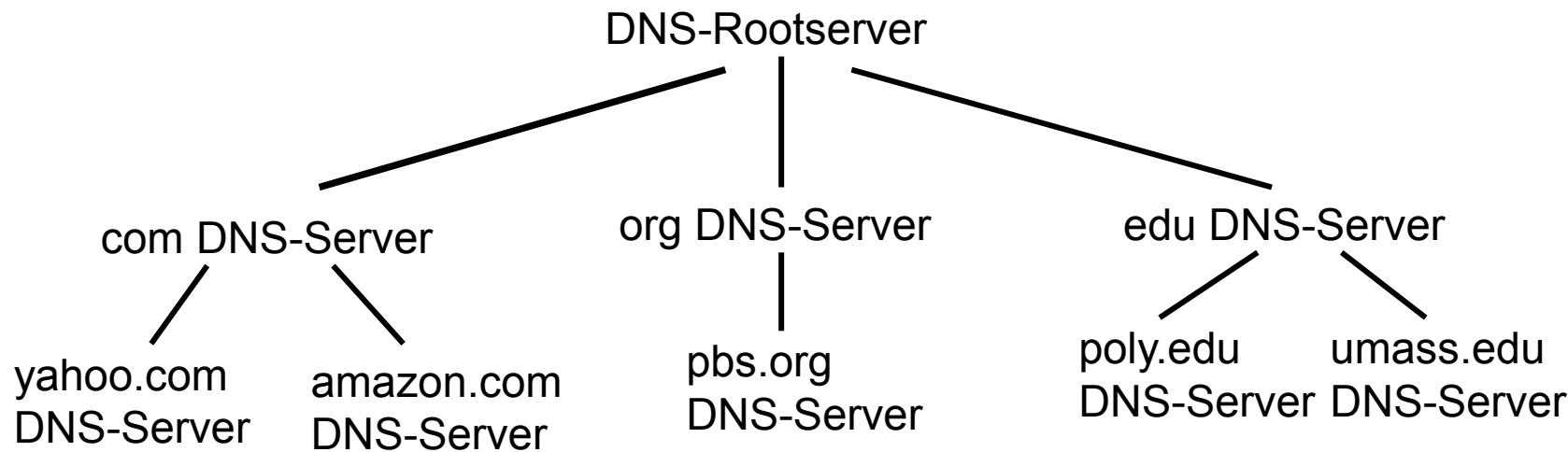
- Übersetzung von Hostnamen in IP-Adressen
- Aliasnamen für Hosts
 - Kanonische Namen und Aliasnamen
- Aliasnamen für Mailserver
- Lastausgleich
 - Replizierte Webserver: mehrere IP-Adressen von einem kanonischen Namen

Warum ist DNS nicht zentralisiert?

- Robustheit gegenüber Fehlern und Angriffen
- Datenverkehrsaufkommen
- Große „Distanz“ zur zentralisierten Datenbank
- Wartung



Verteilte, hierarchische Datenbank

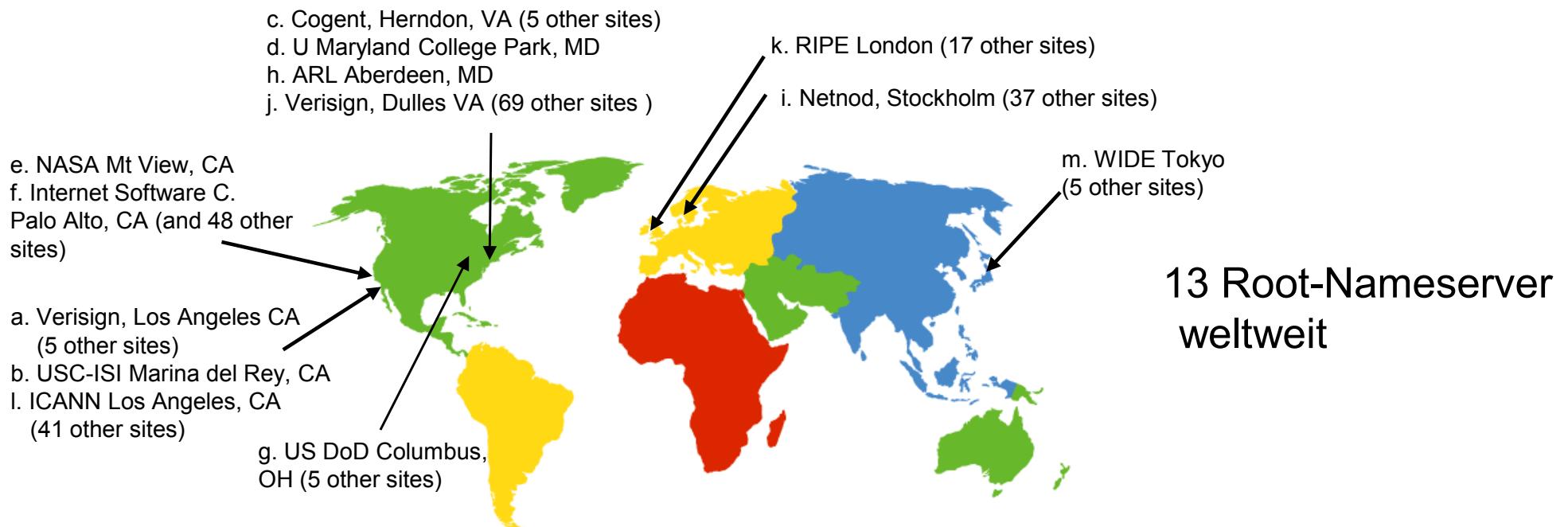


Client sucht die IP-Adresse von www.amazon.com – erste Annäherung:

- Client fragt seinen lokalen DNS-Server
- Dieser fragt einen DNS-Rootserver, um den DNS-Server für com zu finden
- Danach fragt er den com-DNS-Server, um den amazon.com-DNS-Server zu finden
- Dann wird der amazon.com-DNS-Server gefragt, um die IP-Adresse zu www.amazon.com zu erhalten

DNS: Root-Nameserver

- Wird vom lokalen Nameserver kontaktiert, wenn dieser einen Namen nicht auflösen kann (kein Mapping vorliegt)
- Root-Nameserver:
 - Kennt die Adressen der Nameserver der Top-Level-Domains (com, net, org, de, uk, ...)
 - Gibt diese Informationen an die lokalen Nameserver weiter



TLD- und Autoritative Server

- **Top-Level-Domain (TLD)-Server:**
 - Verantwortlich für com, org, net, edu etc. sowie für alle Länder-Domains, z.B. de, uk, fr, ca, jp
 - Network Solutions ist verantwortlich für den com-TLD-Server
 - Educause hat die Verantwortung für den edu-TLD-Server
- **Autoritativer DNS-Server:**
 - DNS-Server einer Organisation, der eine autorisierte Abbildung der Namen dieser Organisation auf IP-Adressen anbietet.
 - Verwaltet von der entsprechenden Organisation oder einem Service Provider

Lokale Nameserver

- Gehört nicht zur Hierarchie der DNS-Server
- Jeder ISP (ISP für Privatkunden, Firmen, Universität) besitzt einen lokalen Nameserver
 - Werden auch “Default-Nameserver” genannt
- Wenn ein Host eine DNS-Anfrage startet, dann schickt er diese an seinen lokalen Nameserver
 - Dieser kümmert sich um die Anfrage so lange, bis eine endgültige Antwort vorliegt
 - Dazu kontaktiert er bei Bedarf Root-Nameserver, TLD-Nameserver und autoritative Nameserver
 - Dann schickt er die Antwort an den Host zurück

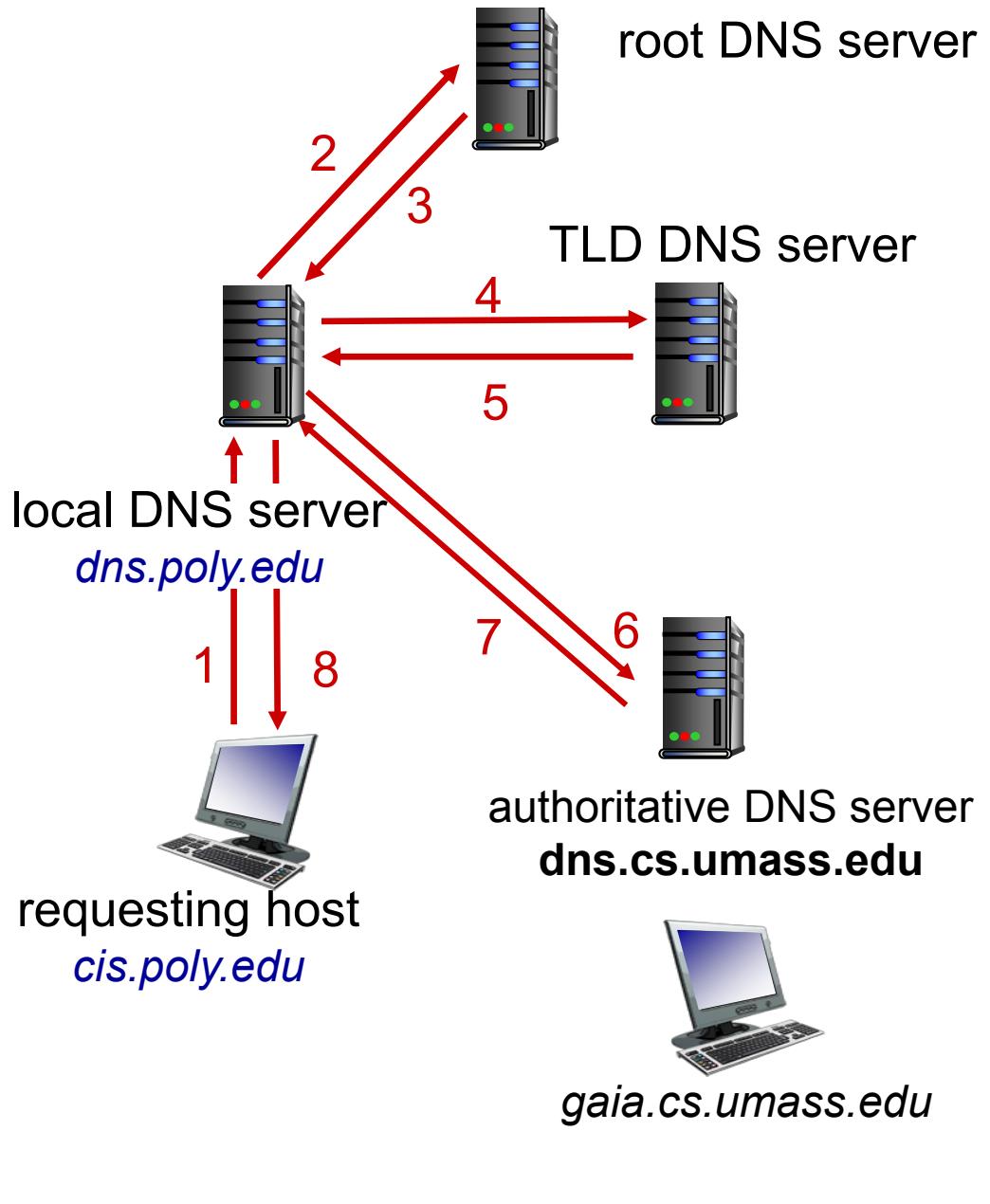
Beispiel für die Namensauflösung mit DNS

- Host cis.poly.edu fragt nach der IP-Adresse von gaia.cs.umass.edu

Variante A: iteratives Vorgehen

- Angesprochene Server in der Hierarchie antworten mit einem Verweis auf andere Server
- “Ich kenne den Namen nicht, frag’ diesen Server ...”

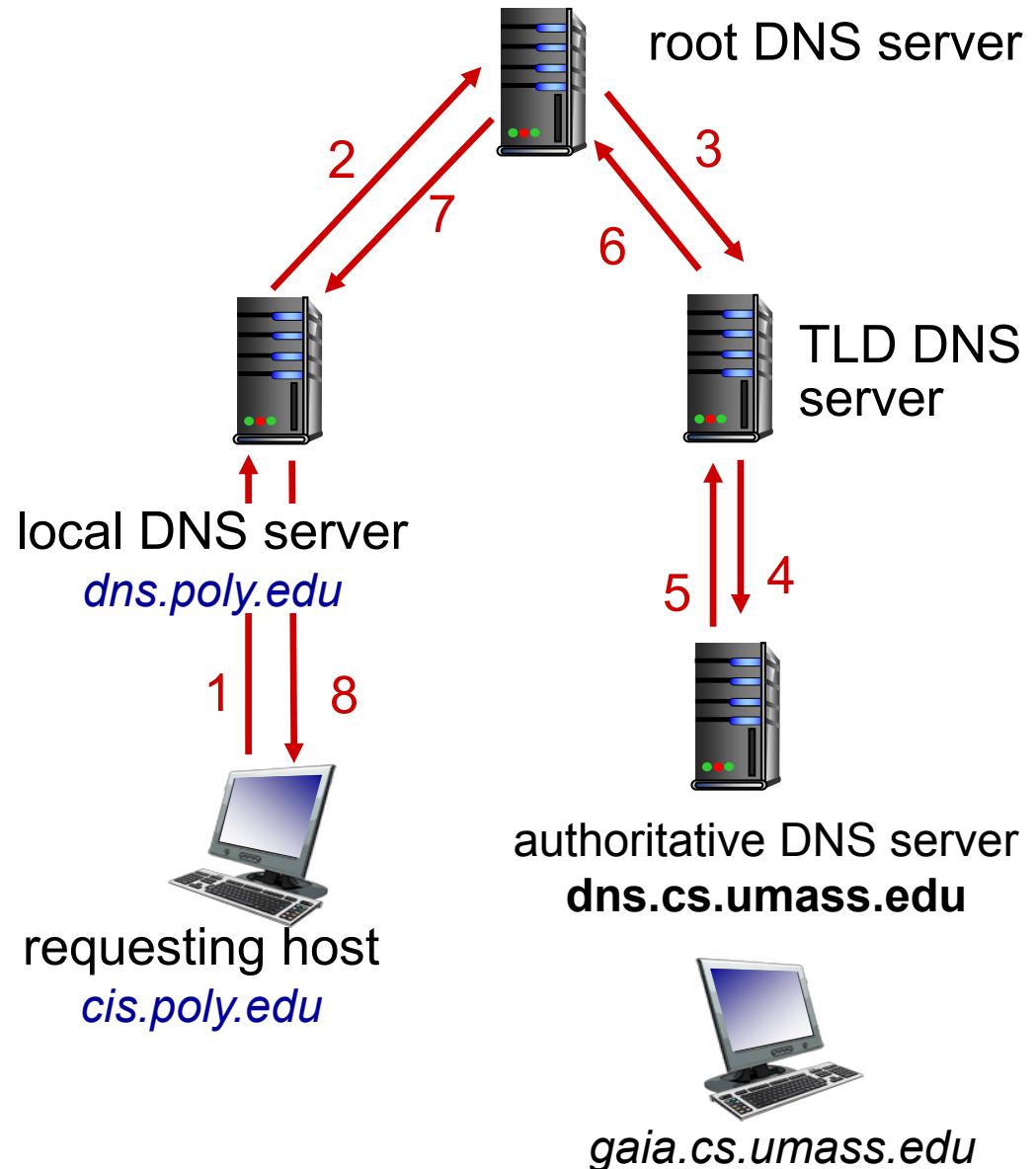
→ `dig +trace +norecurse`



Beispiel für die Namensauflösung mit DNS

Variante B: Rekursives Vorgehen

- Die Aufgabe zur Namensauflösung wird an den gefragten Nameserver delegiert.
- Zusätzliche Belastung!
- Root-Nameserver erlauben dies häufig nicht
- Andere Nameserver dagegen schon!
- Vorteil: Caching der Antworten



DNS: Caching

- Sobald ein Nameserver eine Abbildung zur Namensauflösung kennenlernt, merkt er sich diesen in einem Cache
 - Die Einträge im Cache werden nach einer vorgegebenen Zeit wieder gelöscht
 - Die Adressen der TLD-Server werden üblicherweise von den lokalen Nameservern gecached
 - Root-Nameserver werden eher selten angesprochen
- Mechanismen zur Pflege von Cache-Einträgen und zur Benachrichtigung bei Änderungen werden von der IETF entwickelt
 - RFC 2136

DNS Resource Records

DNS: Verteilte Datenbank für Resource Records (RR)

RR-Format: (name, wert, typ, ttl)

- Typ=A
 - name ist der Hostname
 - value ist die IP-Adresse
 - (relay1.bar.foo.com, 145.37.93.126, A)
- Typ=NS
 - name ist eine Domain (z.B. foo.com)
 - value ist der Hostname des autoritativen Nameservers für diese Domain
 - (foo.com, dns.foo.com, NS)
- Typ=CNAME
 - name ist ein Alias für einen kanonischen (echten) Namen:
 - value ist der kanonische Name
 - (foo.com, relay1.bar.foo.com, CNAME)
- Typ=MX
 - value ist der Name des Mailservers für die Domain name
 - (foo.com, mail.bar.foo.com, MX)

TTL=time to live

DNS-Protokoll - Nachrichtenformat

- DNS-Protokoll: Query- und Reply-Nachrichten, beide mit demselben Nachrichtenformat

Header-Felder

- **Identifikation:** 16-Bit-ID wird für die Query-Nachricht vergeben, die Reply-Nachricht verwendet dieselbe ID

- **Flags:**

- query/reply
- recursion desired
- recursion available
- reply is authoritative

Identifikation	Flags	
Anzahl Questions	Anzahl Answers-RRs	12 Byte
Anzahl Authority-RRs	Anzahl Additional RRs	
Question (variable Anzahl von Fragen)		Name-, Typfelder einer Anfrage
Answer (variable Anzahl von Resource Records)		RRs als Antwort auf eine Anfrage
Authority (variable Anzahl von Resource Records)		Daten über autoritative Server
Additional (variable Anzahl von Resource Records)		Zusätzliche, möglicherweise hilfreiche Informationen

→ Wireshark + ping

Neue Einträge in DNS einfügen

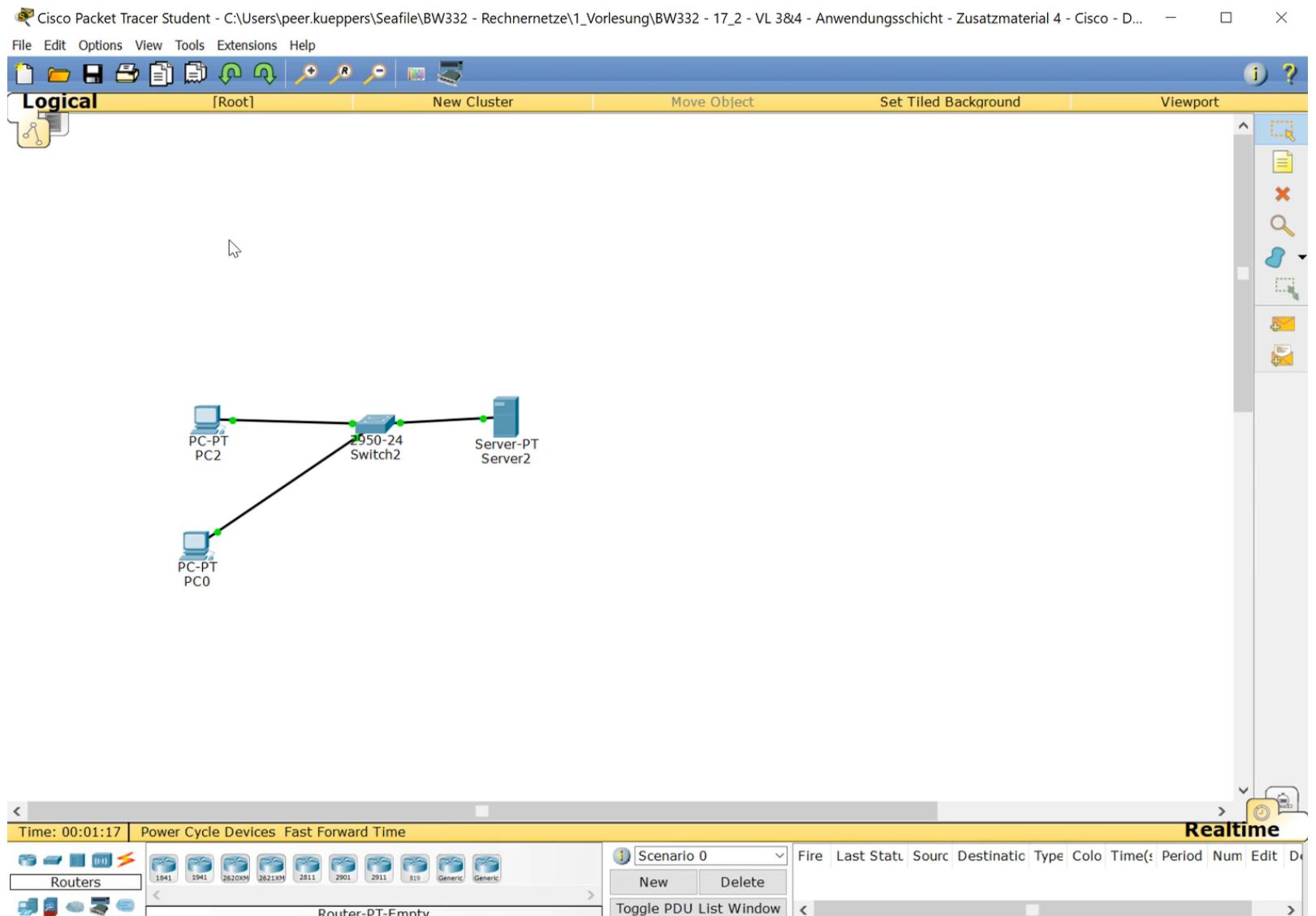
- Beispiel: neues Startup “Network Utopia”
- Registrieren des Namens networkuoptopia.com bei einem DNS-Registrar (z.B. Network Solutions)
 - Bereitstellen der Namen und der IP-Adressen der autoritativen Server (Primary und Secondary)
 - Registrar trägt zwei RRs beim TLD-Server für com ein:
 - (networkuoptopia.com, dns1.networkuoptopia.com, NS)
 - (dns1.networkuoptopia.com, 212.212.212.1, A)

Übung

DNS im Einsatz (Cisco Packet Tracer)

Bitte setzen Sie im Packet-Tracer folgende Umgebung auf und nutzen Sie die Simulation, um das DNS-Protokoll nachzuvollziehen.

Hinweis: Der Cisco Packet Tracer steht im PC-Pool zur Verfügung. Den Screencast finden Sie im Materialordner



Agenda

Grundlagen von Netzwerk-Applikationen

Web und HTTP

FTP und E-Mail

Domain Name System

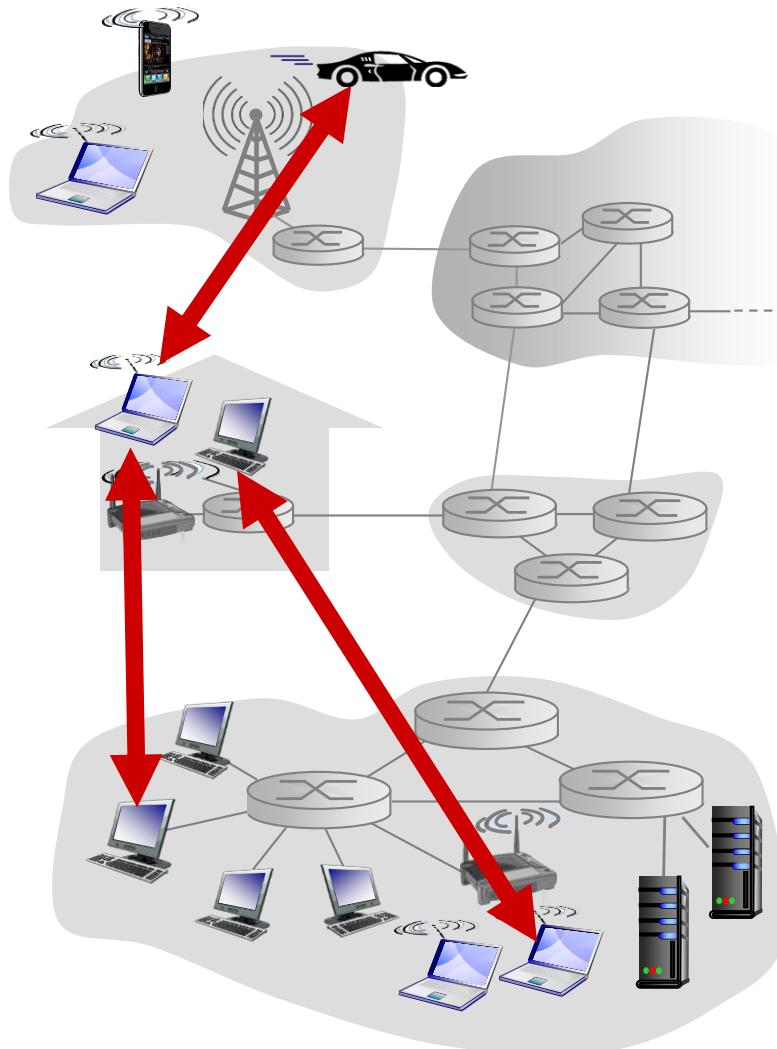
Peer-to-Peer Anwendungen

Socket-Programmierung

Zusammenfassung und Ausblick

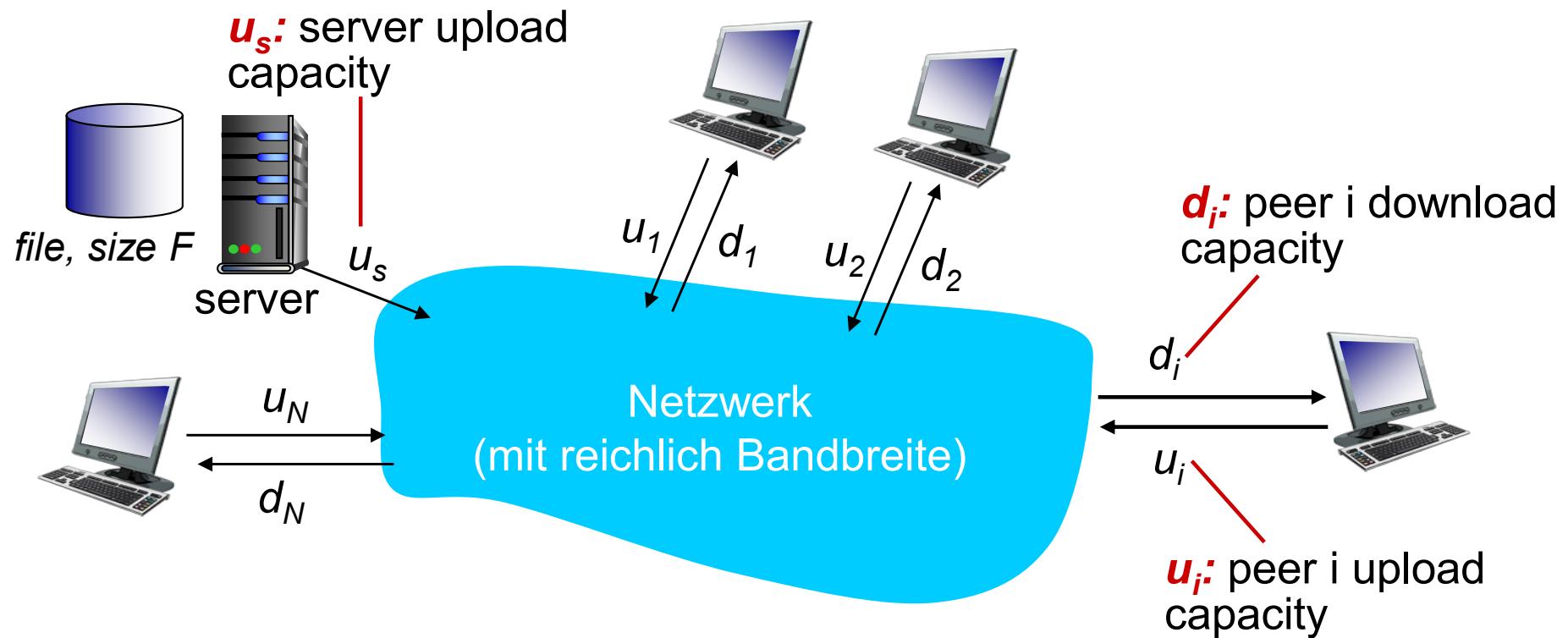
Peer-to-Peer (P2P) Architektur

- Kein „always-on“ server
- Beliebige Endsysteme kommunizieren direkt miteinander
- Peers sind nur vorübergehend miteinander verbunden und wechseln ihre IP-Adressen
- Beispiele:
 - P2P Filesharing (BitTorrent)
 - VoIP (Skype)



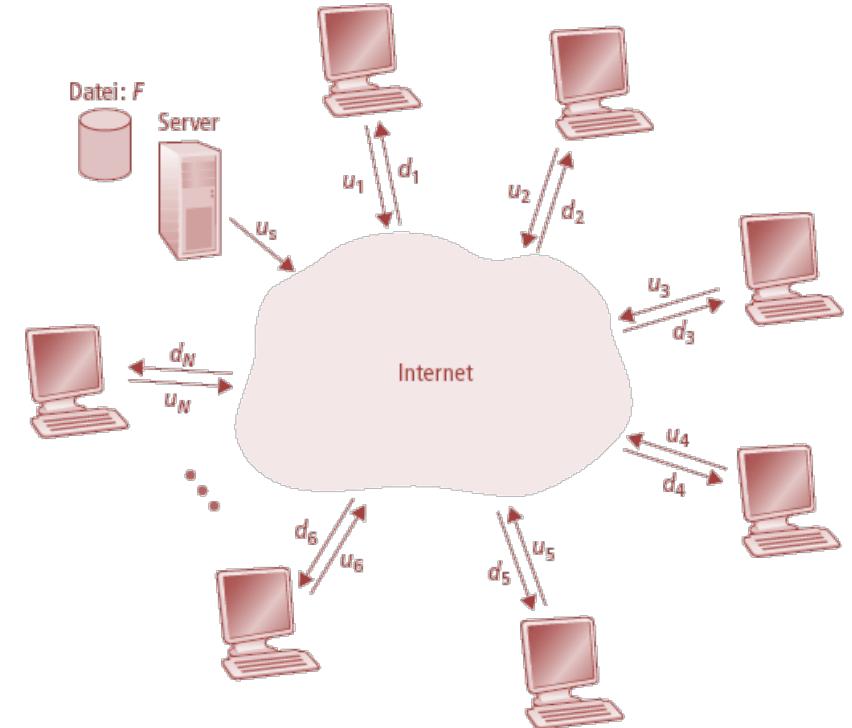
Dateiverteilung: Client-Server

- Wie lange dauert es, bis die Datei mit Größe F von einem Server auf N Peers verteilt wurde?
 - Peer upload/download Kapazität ist der limitierende Faktor



Dateiverteilung: Client-Server

- Server Übertragung: sendet sequentiell N Kopien der Datei (upload):
 - Dauer für eine Kopie: F/u_s
 - Dauer für N Kopien: NF/u_s
- Client: jeder Client muss eine Kopie herunterladen
- Client i benötigt F/d_i Sekunden für den Download



Zeit, um die Datei an N Computer mittels Client/Server zu übertragen

$$D_{c-s} \geq \max\{NF/u_s, F/\min(d_i)\}$$

Wächst für große N linear mit N

Dateiverteilung: P2P

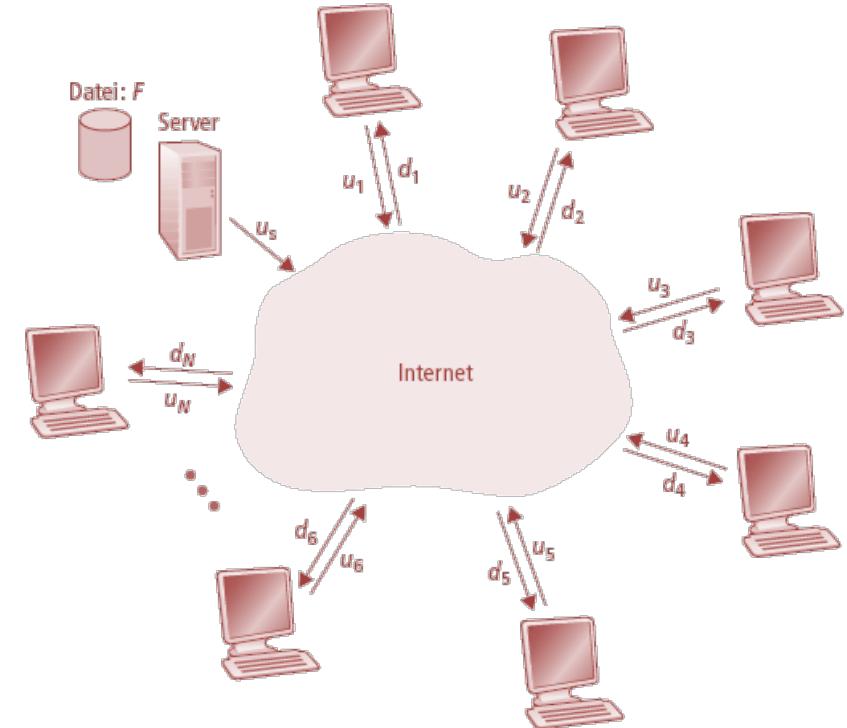
- Server muss eine Kopie senden: F/u_s
- Client i braucht F/d_i Sekunden für den Download
- NF Bits müssen insgesamt heruntergeladen werden
- Höchstmögliche Datenrate ins Netz:

$$u_s + \sum_{i=1,N} u_i$$

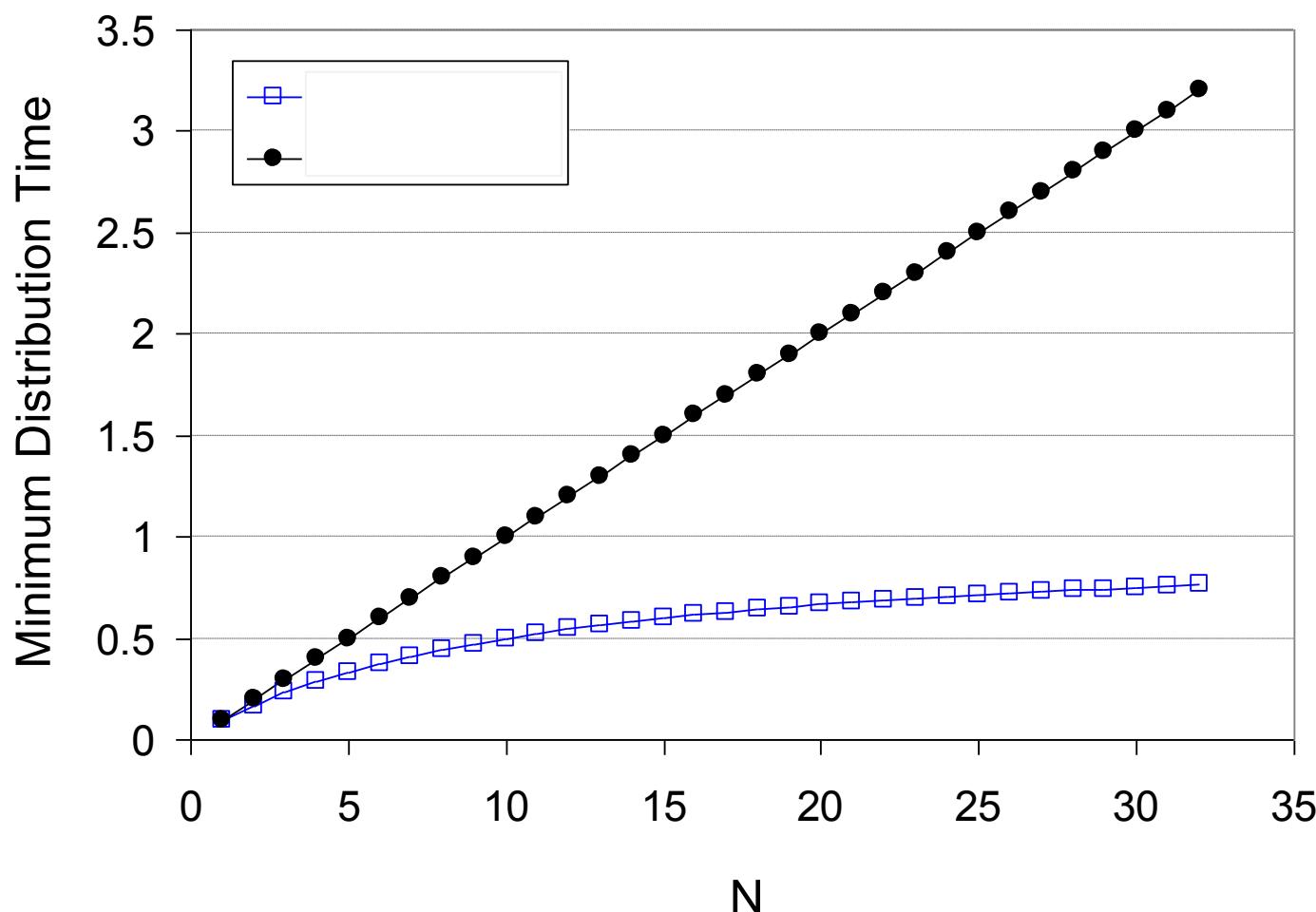
$$d_{P2P} = \max \{ F/u_s, F/\min(d_i), NF/(u_s + \sum_{i=1,N} u_i) \}$$

Wächst linear mit N ...

... aber genauso dieser Ausdruck, weil jeder Peer Kapazität einbringt



Vergleich: Client-Server und P2P

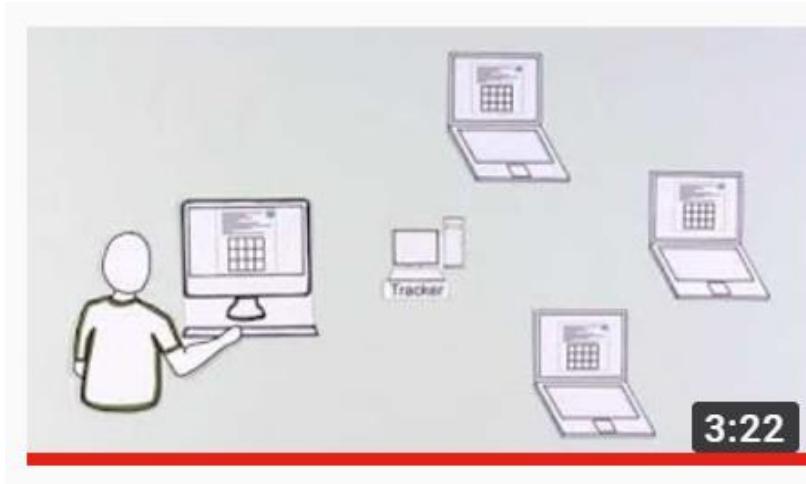


Beispiel: BitTorrent

- Bei vielen P2P-Netzwerken besteht das Problem, dass viele Teilnehmer Dateien herunterladen, aber nur wenige Daten bereitstellen
- BitTorrent ist ein P2P-Ansatz zum Verbreiten von Dateien, bei dem auf Fairness Wert gelegt wird.
- Es ist weit verbreitet.
 - Beispiel: Verteilen von Patches (World of Warcraft)
- Nicht ganz unproblematisch, insbesondere wenn Firewalls die benötigten Ports sperren



- **Video** zur grundlegenden Funktionsweise von BitTorrent
<https://www.youtube.com/watch?v=6PWUCFmOQwQ>



BitTorrent, how it works?

Cornel • 85.000 Aufrufe • vor 6 Jahren

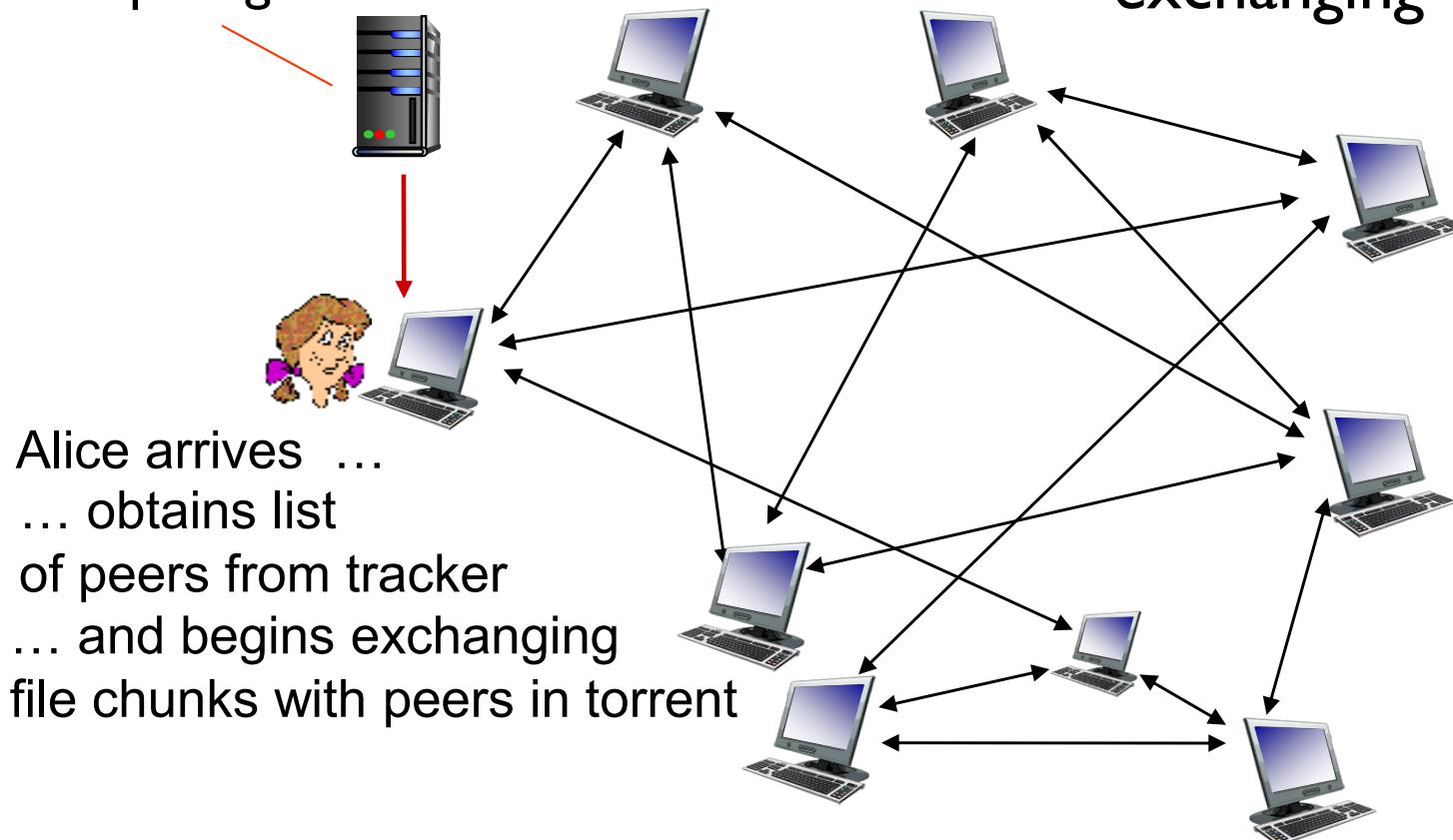
uTorrent a (very) tiny **BitTorrent** client.

BitTorrent Grundlagen

- Dateien werden in 256Kb chunks aufgeteilt
- Peers senden und empfangen diese Chunks

tracker: tracks peers
participating in torrent

torrent: group of peers
exchanging chunks of a file



BitTorrent Grundlagen

■ Bestandteile:

- Eine statische „Metainfo“-Datei mit der Endung .torrent
 - Beinhaltet statische Informationen zum Download
- Ein gewöhnlicher Webserver
 - Auf diesem ist die .torrent-Datei abgelegt (und verlinkt)
- BitTorrent-Clients der Endanwender
 - Interpretieren die .torrent-Datei
 - Laden Teile des Downloads von anderen Clients herunter
 - Stellen Teile des Downloads anderen Clients zur Verfügung
 - Sorgen für Fairness
- Ein Tracker
 - Vermittelt die Clients untereinander
- Ein „Seed“/„Original-Downloader“
 - Wenigstens eine Urquelle für den Download muss bereitgestellt werden

BitTorrent Client

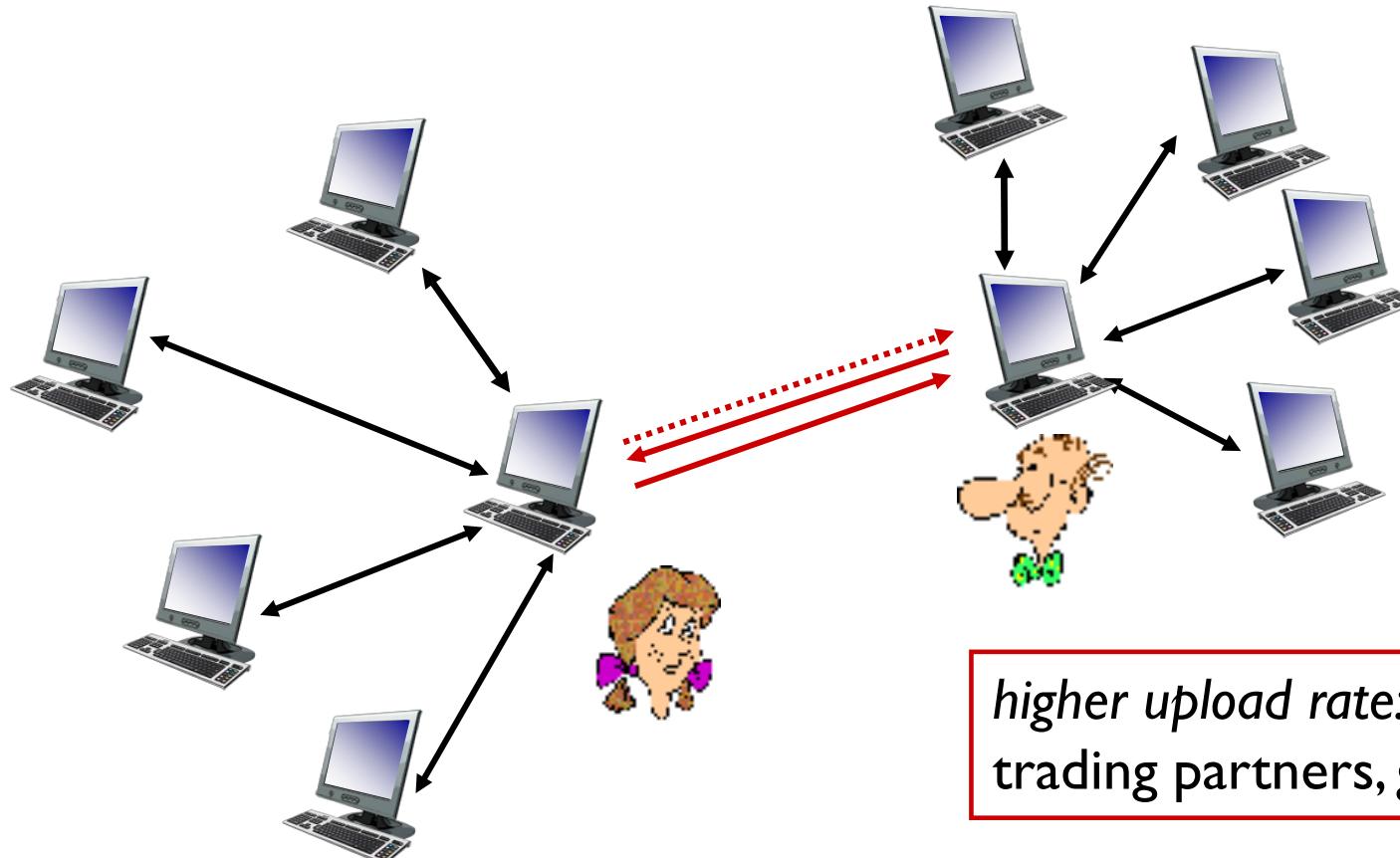
- Lädt zu Beginn die .torrent-Datei vom Webserver
- Kontaktiert regelmäßig den Tracker
- Up-/Download
 - Baut zu allen Peers, die ihm bekannt sind, eine TCP-Verbindung auf
 - Die Verbindungen sind bidirektional
 - Zuerst wird angekündigt, welche Teile des Downloads man besitzt
 - Erhält man im Laufe der Zeit neue Teile, dann kündigt man dies ebenfalls an
 - Auf jeder Seite der Verbindung hat jede Verbindung zwei Zustandsinformationen:
 - 1. Interest: ist auf 1 gesetzt, wenn man vom Kommunikationspartner Teile herunterladen möchte (da er Teile hat, die man selbst nicht besitzt)
 - 2. Choked: Dies ist der Fairness-Mechanismus von BitTorrent – man bedient nur Peers, die nicht gechoked sind

Fairness

- Wie wird das Choked-Bit gesetzt?
 - Bei den vier Verbindungen, von denen man den höchsten Download hat, wird das Choked-Bit gelöscht
 - Strategie: „Wie Du mir, so ich Dir“ (engl. Tit-for-Tat)
 - Dies wird alle 10 Sekunden neu bestimmt
 - Zusätzlich wird reihum bei jeder Verbindung für 30 Sekunden das Choke-Bit gelöscht
 - Damit gibt man neuen Kommunikationspartnern eine Chance
 - „Optimistic Unchoking“
- Dies ist eine sehr ähnliche Strategie, wie sie auch zur Lösung des Gefangenendilemmas in der Spieltheorie eingesetzt wird

BitTorrent: tit-for-tat

- (1) Alice “optimistically unchoke” Bob
- (2) Alice becomes one of Bob’s top-four providers; Bob reciprocates
- (3) Bob becomes one of Alice’s top-four providers



Agenda

Grundlagen von Netzwerk-Applikationen

Web und HTTP

FTP und E-Mail

Domain Name System

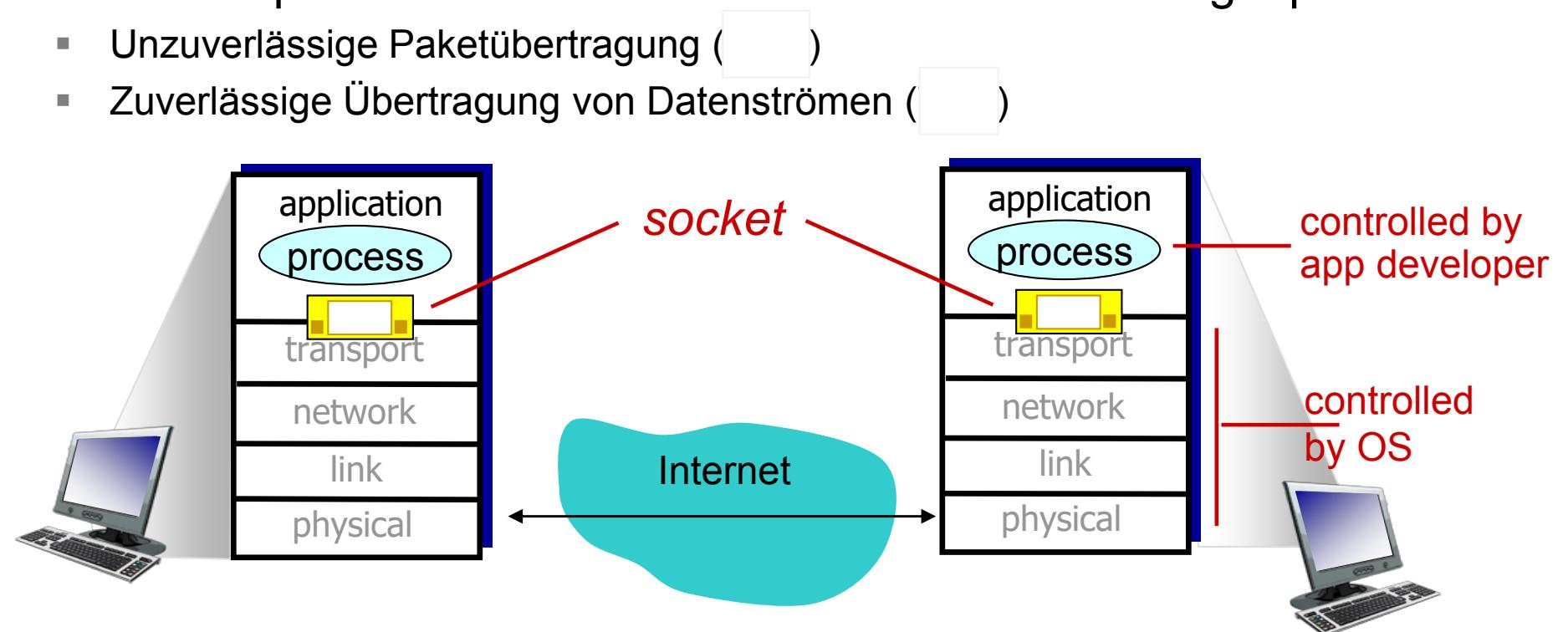
Peer-to-Peer Anwendungen

Socket-Programmierung

Zusammenfassung und Ausblick

Socket-Programmierung

- Ziel: zeigen, wie man eine Client/Server-Anwendung programmiert, die über Sockets kommuniziert
- Sockets werden von Anwendungen erzeugt, verwendet und geschlossen und arbeiten nach dem Client/Server-Paradigma.
- Zwei Transportdienste werden über die Socket-API angesprochen:
 - Unzuverlässige Paketübertragung ()
 - Zuverlässige Übertragung von Datenströmen ()



Socket-Programmierung

Anwendungsbeispiel

- Der Client liest eine Zeile von Zeichen (Daten) über die Tastatur ein und sendet diese zu einem Server.
- Der Server empfängt die Daten und konvertiert diese in Großbuchstaben.
- Der Server sendet die modifizierten Daten zurück zum Client.
- Der Client empfängt die modifizierten Daten und zeigt sie auf dem Bildschirm an.

Socket-Programmierung mit UDP

UDP: keine Verbindung zwischen Client und Server

- Keine „Handshakes“ vor dem Senden der Daten
- Der Sender fügt an jedes Paket die IP-Zieladresse und Zielport an.
- Der Empfänger extrahiert die IP-Adresse und Port vom empfangenen Paket

UDP: übertragene Daten können verloren gehen oder in falscher Reihenfolge eintreffen.

Aus Sicht der Applikation:

- UDP bietet einen nicht-zuverlässigen (unreliable) Transfer von Byte-Gruppen (Datagrammen) zwischen Client und Server

Socket-Programmierung mit UDP

server (running on serverIP)

create socket, port= x:

```
serverSocket =  
socket(AF_INET,SOCK_DGRAM)
```

read datagram from
serverSocket

write reply to
serverSocket
specifying
client address,
port number

client

create socket:

```
clientSocket =  
socket(AF_INET,SOCK_DGRAM)
```

Create datagram with server IP and
port=x; send datagram via
clientSocket

read datagram from
clientSocket
close
clientSocket

Socket-Programmierung mit UDP: Beispiel-Client

Python UDPCClient

```
Python's socket  
Library einbinden → from socket import *  
  
UDP socket erzeugen → serverName = '46.101.190.70'  
Tastatur-Input lesen → serverPort = 12000  
Server-Namen und Port  
anhängen und senden → clientSocket = socket(AF_INET, SOCK_DGRAM)  
Antwort vom Server → message = input('Input lowercase sentence:')  
empfangen → clientSocket.sendto(message.encode(),(serverName,  
serverPort))  
Antwort ausgeben und  
Socket schließen → modifiedMessage, serverAddress =  
clientSocket.recvfrom(2048)  
→ print(modifiedMessage)  
→ clientSocket.close()
```

Socket-Programmierung mit UDP: Beispiel-Server

Python UDPServer

```
from socket import *
serverPort = 12000
UDP socket erzeugen → serverSocket = socket(AF_INET, SOCK_DGRAM)
Socket an den lokalen → serverSocket.bind(("", serverPort))
Port 12000 binden → print("The server is ready to receive")
Schleife (unendlich) → while 1:
Vom UDP socket lesen und → message, clientAddress = serverSocket.recvfrom(2048)
insbesondere auch die Client → modifiedMessage = message.upper()
Adresse (IP und Port) holen → serverSocket.sendto(modifiedMessage, clientAddress)
Den modifizierten →
String senden →
```

Socket-Programmierung mit TCP

Client kontaktiert Server

- Server-Prozess muss laufen
- Server muss einen Socket (eine Tür) angelegt haben, der Client-Anfragen entgegennimmt

Vorgehen im Client:

- Anlegen eines Client-TCP-Sockets
- Angeben von IP-Adresse und Portnummer des Server-prozesses
- Durch das Anlegen eines Client-TCP-Sockets wird eine TCP-Verbindung zum Server-prozess hergestellt

- Wenn der Serverprozess von einem Client kontaktiert wird, dann erzeugt er einen neuen Socket, um mit diesem Client zu kommunizieren
 - So kann der Server mit mehreren Clients kommunizieren
 - Portnummern der Clients werden verwendet, um die Verbindungen zu unterscheiden

Anwendungsperspektive

TCP stellt einen zuverlässigen, reihenfolgeerhaltenden Transfer von Bytes zwischen Client und Server zur Verfügung

Socket-Programmierung mit TCP

server (running on hostid)

create socket,
port=**x**, for incoming
request:
serverSocket = socket()

wait for incoming
connection request
connectionSocket = serverSocket.accept()

read request from
connectionSocket

write reply to
connectionSocket

close
connectionSocket

client

create socket,
connect to **hostid**, port=**x**
clientSocket = socket()

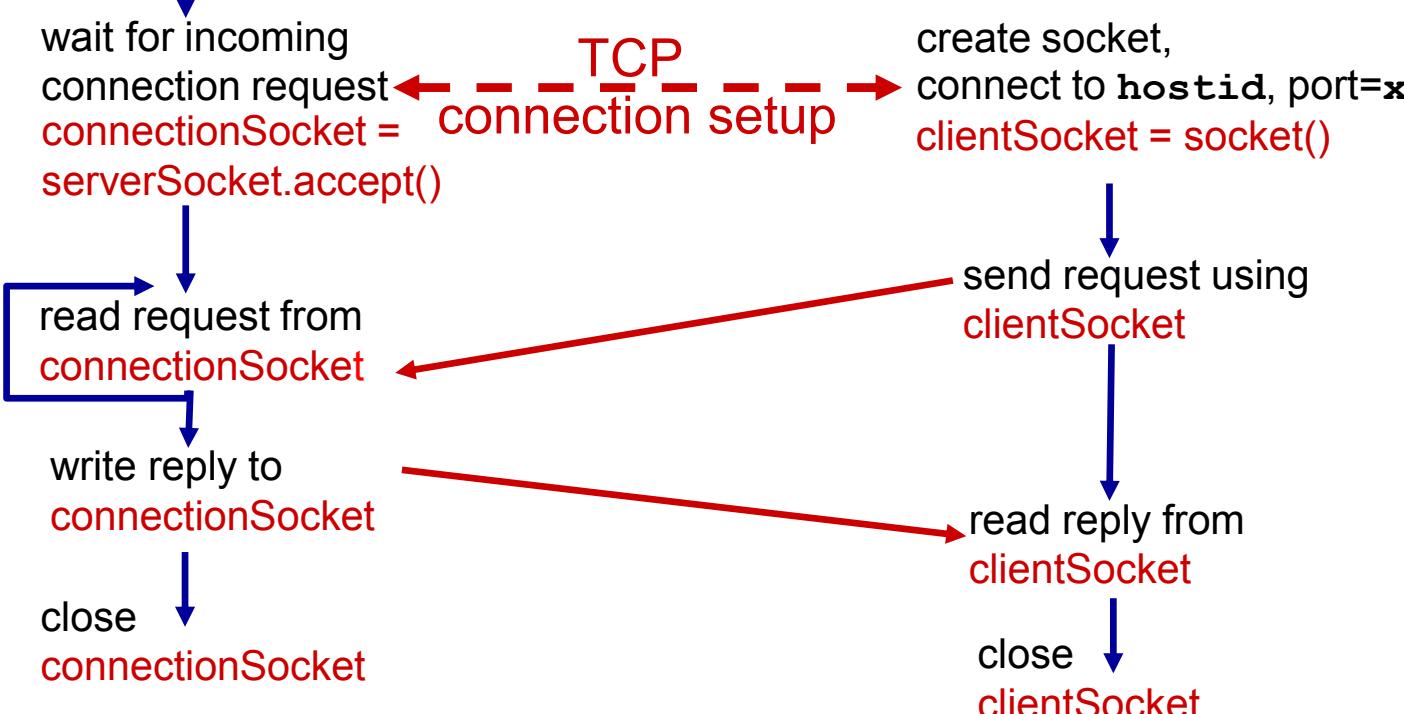
send request using
clientSocket

read reply from
clientSocket

close
clientSocket

TCP

connection setup



Socket-Programmierung mit TCP: Beispiel-Client

Python TCPClient

```
Python's socket  
Library einbinden → from socket import *  
  
TCP socket erzeugen → serverName = '46.101.190.70'  
                      serverPort = 12000  
  
Tastatur-Input lesen → clientSocket = socket(AF_INET, SOCK_STREAM)  
                      clientSocket.connect((serverName,serverPort))  
  
Server-Namen und Port  
müssen nicht angehängt  
werden → sentence = input("Input lowercase sentence:")  
  
Antwort vom Server  
empfangen → clientSocket.send(sentence.encode())  
                      modifiedSentence = clientSocket.recv(1024)  
                      print ('From Server:')  
  
Antwort ausgeben und  
Socket schließen → print (modifiedSentence)  
                      clientSocket.close()
```

Socket-Programmierung mit TCP: Beispiel-Server

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(("",serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP welcoming
Socket erzeugen → from socket import *

server startet auf eingehende
TCP-Verbindungen zu
"hören" → serverPort = 12000

Schleife (unendlich) → serverSocket = socket(AF_INET,SOCK_STREAM)
→ serverSocket.bind(("",serverPort))
→ serverSocket.listen(1)

Server wartet durch accept()
auf eingehende Anfragen, ein
neuer Socket wird dann
geöffnet → print("The server is ready to receive")
→ while 1:

Die Daten werden vom Socket
gelesen (aber nicht die Adresse
und Port wie bei UDP!) und
gesendet → connectionSocket, addr = serverSocket.accept()
→ sentence = connectionSocket.recv(1024)
→ capitalizedSentence = sentence.upper()
→ connectionSocket.send(capitalizedSentence)
→ connectionSocket.close()

Verbindung schließen
(aber nicht den "Welcoming
Socket") → connectionSocket.close()

Die Python-Dateien werden im Materialordner bereitgestellt.

Agenda

Grundlagen von Netzwerk-Applikationen

Web und HTTP

FTP und E-Mail

Domain Name System

Peer-to-Peer Anwendungen

Socket-Programmierung

Zusammenfassung und Ausblick

Zusammenfassung (VL 4-5) und Ausblick



- In den letzten Vorlesungen haben wir uns mit der Anwendungsschicht befasst:
 - Die Konzeption und Implementierung von Protokollen der Anwendungsschicht wurde behandelt und an beispielhaften Protokollen im Detail analysiert.
 - Protokolle sind die Grundlage für sämtliche Kommunikation in Netzwerken!
 - Sie haben verschiedene Anwendungs-Architekturen in Bezug auf Netzwerke kennengelernt.
 - Sie sollten nun das Domain Name System verstehen und einen ersten Einblick in die Socket-Programmierung erhalten haben.
- In den folgenden beiden Veranstaltungen werden wir uns dem top-down Ansatz folgend auf die nächste Schicht konzentrieren, die für die Anwendungsschicht die Transportdienste bereitstellt: **Transportschicht**



Fragen



Vielen Dank für Ihre Aufmerksamkeit!

**BW332 - Rechnernetze (mit
Praktikum)**

**Vorlesungen 6-8
Transportschicht**

Ziele von VL 6-8

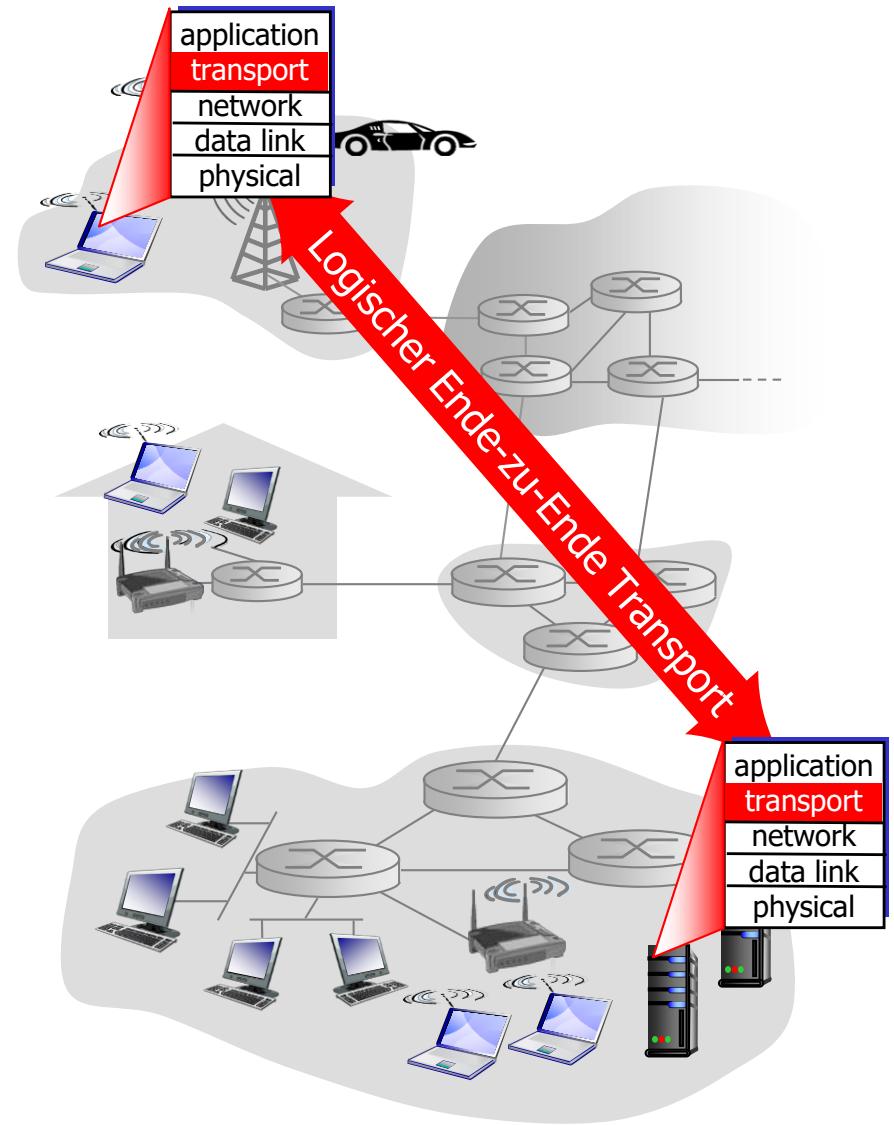


- Die Grundlagen und Hintergründe von Diensten auf der Transportschicht verstehen
 - Multiplexing / Demultiplexing
 - Verlässlicher Datentransfer
 - Flusskontrolle
 - Congestion Control
- Internetprotokolle der Transportschicht im Detail verstehen und hinsichtlich Ihrer Eignung in verschiedenen Anwendungsszenarien bewerten können.
- Das verbindungslose Protokoll UDP sowie das verbindungsorientierte Protokoll TCP im Detail nachvollziehen können.

- Dienste der Transportschicht
- Multiplexing und Demultiplexing
- Verbindungslose Übertragung: UDP
- Zuverlässige Datenübertragung
- Verbindungsorientierte Übertragung: TCP
- Überlastkontrolle: Grundlagen und Umsetzung in TCP
- Zusammenfassung und Ausblick

Transportdienste und Protokolle

- Stellen *logische Kommunikation* zwischen Anwendungsprozessen auf verschiedenen Hosts zur Verfügung
- Transportprotokolle laufen auf Endsystemen
 - Sender: teilt Anwendungsnachrichten in Segmente auf, gibt diese an die Netzwerkschicht weiter
 - Empfänger: fügt Segmente wieder zu Anwendungsnachrichten zusammen, gibt diese an die Anwendungsschicht weiter
- Es existieren verschiedene Transportschichtprotokolle
 - Internet: TCP und UDP



Netzwerk- vs. Transportschicht

- *Netzwerkschicht:*
logische Kommunikation
zwischen Hosts
- *Transportschicht:*
logische Kommunikation
zwischen Prozessen
 - verwendet und erweitert
die Dienste der
Netzwerkschicht

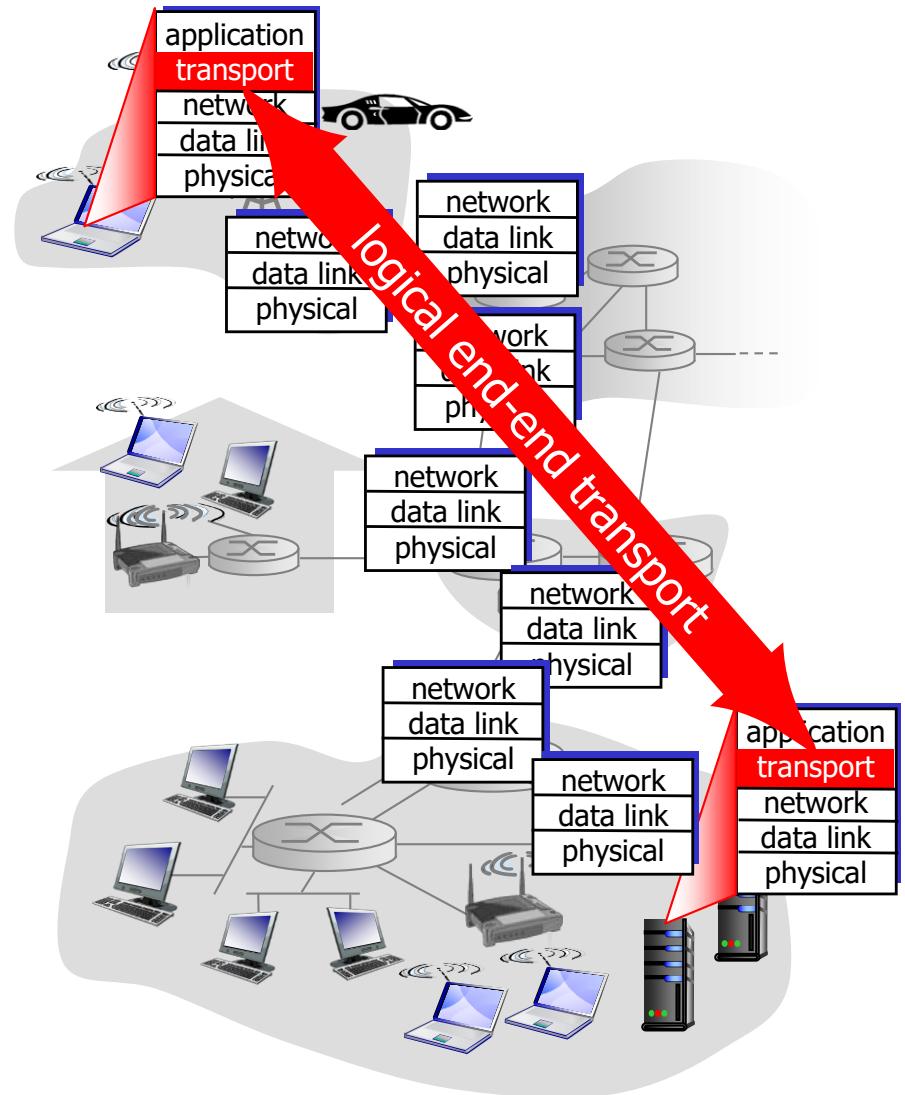
Analogie:

*12 kids in Ann's house sending
letters to 12 kids in Bill's house:*

- hosts = houses
- processes = kids
- app messages = letters in envelopes
- transport protocol = Ann and Bill who multiplex from / demultiplex to kids
- network-layer protocol = postal service

Transportprotokolle im Internet

- Zuverlässige, reihenfolgeerhaltende Auslieferung (TCP)
 - Überlastkontrolle
 - Flusskontrolle
 - Verbindungsmanagement
- Unzuverlässige Datenübertragung ohne Reihenfolgeerhaltung: UDP
 - Minimale Erweiterung der “Best-Effort”-Funktionalität von IP (*Netzwerkschicht, kommt später*)
- Dienste, die nicht zur Verfügung stehen:
 - Garantien bezüglich Bandbreite oder Verzögerung



- Dienste der Transportschicht
- Multiplexing und Demultiplexing
- Verbindungslose Übertragung: UDP
- Zuverlässige Datenübertragung
- Verbindungsorientierte Übertragung: TCP
- Überlastkontrolle: Grundlagen und Umsetzung in TCP
- Zusammenfassung und Ausblick

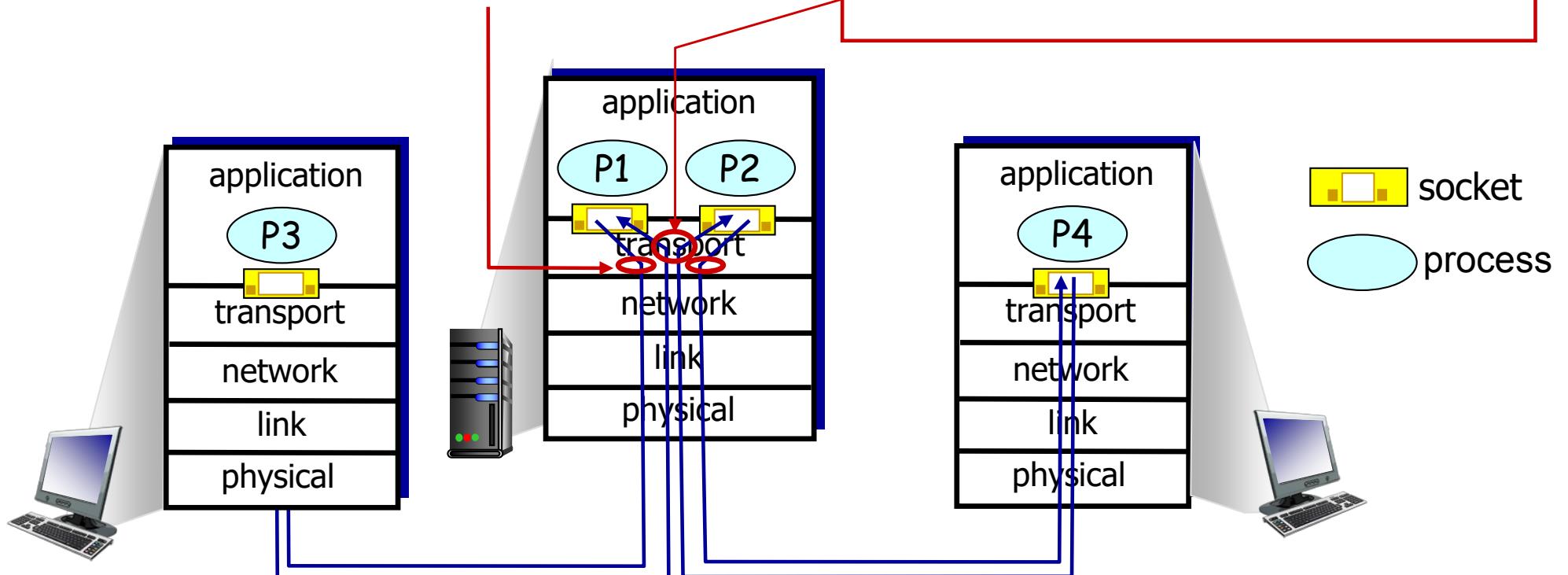
Multiplexing / Demultiplexing

Multiplexing beim Sender:

Daten von mehreren Sockets einsammeln, Daten mit einem Header versehen (der später für das Demultiplexing verwendet wird)

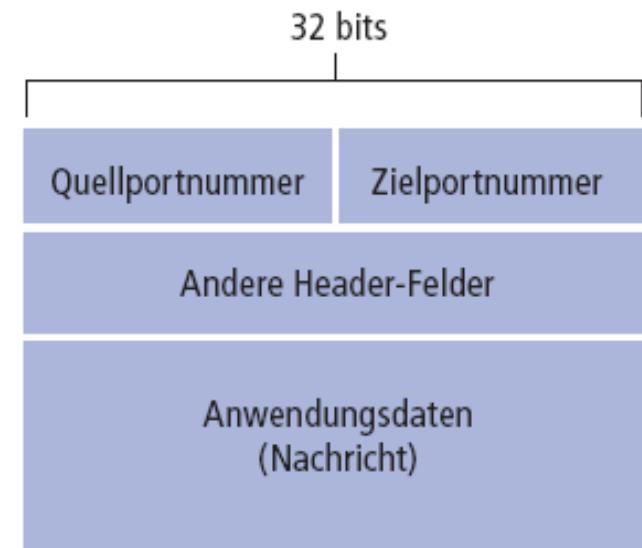
Demultiplexing beim Empfänger

Empfangene Segmente am richtigen Socket abliefern



Funktion von Multiplexing / Demultiplexing

- Host empfängt IP-Datagramme
 - Jedes Datagramm hat eine Absender-IP-Adresse und eine Empfänger-IP-Adresse
 - Jedes Datagramm beinhaltet ein Transportschichtsegment
 - Jedes Segment hat eine Absender- und eine Empfänger-Portnummer
- Hosts nutzen IP-Adressen und Portnummern, um Segmente an den richtigen Socket weiterzuleiten



TCP-/UDP-Nachrichtenformat

Verbindungsloses (De-)Multiplexing: Beispiel

- Erinnern wir uns an das Beispiel aus der Anwendungsschicht
 - (1) Server: Wir erzeugen einen Socket (mit lokaler Portnummer 12000) und warten auf den Empfang von Zeichenketten.
 - (2) Client:
 - Wir erzeugen einen Socket (ohne Angabe einer Portnummer, in dem Fall wird diese automatisch festgelegt)
 - Wir senden über den Client-Socket an den Server, unter Angabe der Ziel-IP Adresse und des Ziel-Ports (12000) und warten danach auf eine Antwort
 - (3) Server: empfängt die Zeichenkette und sendet diese in Großbuchstaben an die Ziel-IP Adresse des Clients und den Port des Client-Programms (ausgelesen aus dem empfangenen Paket)
 - (4) Client: Nach dem Empfang wird die Zeichenkette auf der Konsole ausgegeben.

Erinnerung: Socket-Programmierung mit UDP (Server)

VL5

Python UDPServer

```
from socket import *
serverPort = 12000
UDP socket erzeugen → serverSocket = socket(AF_INET, SOCK_DGRAM)
Socket an den lokalen → serverSocket.bind(("", serverPort))
Port 12000 binden → print("The server is ready to receive")
Schleife (unendlich) → while 1:
Vom UDP socket lesen und → message, clientAddress = serverSocket.recvfrom(2048)
insbesondere auch die Client → modifiedMessage = message.upper()
Adresse (IP und Port) holen → serverSocket.sendto(modifiedMessage, clientAddress)
Den modifizierten →
String senden →
```

Erinnerung: Socket-Programmierung mit UDP (Client)

VL5

Python UDPCClient

```
Python's socket  
Library einbinden → from socket import *  
  
UDP socket erzeugen → serverName = '46.101.190.70'  
Tastatur-Input lesen → serverPort = 12000  
Server-Namen und Port  
anhängen und senden → clientSocket = socket(AF_INET, SOCK_DGRAM)  
Antwort vom Server → message = input('Input lowercase sentence:')  
empfangen → clientSocket.sendto(message.encode(),(serverName,  
serverPort))  
Antwort ausgeben und  
Socket schließen → modifiedMessage, serverAddress =  
clientSocket.recvfrom(2048)  
→ print(modifiedMessage)  
→ clientSocket.close()
```

Verbindungsloses (De-)Multiplexing: Beispiel

- Erinnern wir uns an das Beispiel aus der Anwendungsschicht

- (1) Server: Wir erzeugen einen Socket (mit lokaler Portnummer 12000) und warten auf den Empfang von Zeichenketten.
- (2) Client:
 - Wir erzeugen einen Socket (ohne Angabe einer Portnummer, in dem Fall wird diese automatisch festgelegt)
 - Wir senden über den Client-Socket an den Server, unter Angabe der Ziel-IP Adresse und des Ziel-Ports (12000) und warten danach auf eine Antwort
- (3) Server: empfängt die Zeichenkette und sendet diese in Großbuchstaben an die Ziel-IP Adresse des Clients und den Port des Client-Programms (ausgelesen aus dem empfangenen Paket)
- (4) Client: Nach dem Empfang wird die Zeichenkette auf der Konsole ausgegeben.

Multiplexing:
Evtl. mehrere Client-Anwendungen
senden gleichzeitig

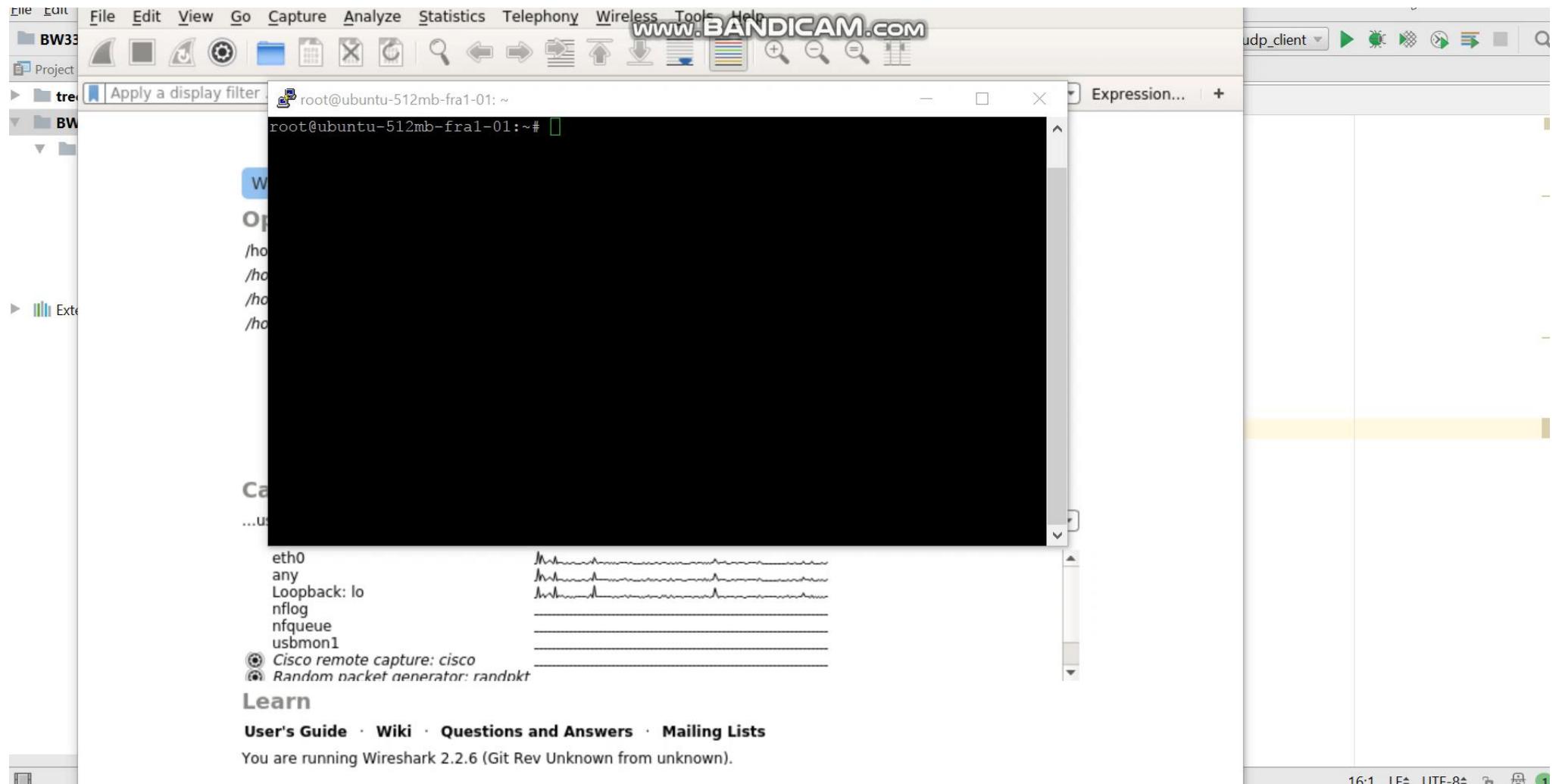
Demultiplexing:
Anhand des Zielports
im Paket erreichen
die Daten unser
Server-Programm

Multiplexing

Demultiplexing



Multiplexing / Demultiplexing mit UDP



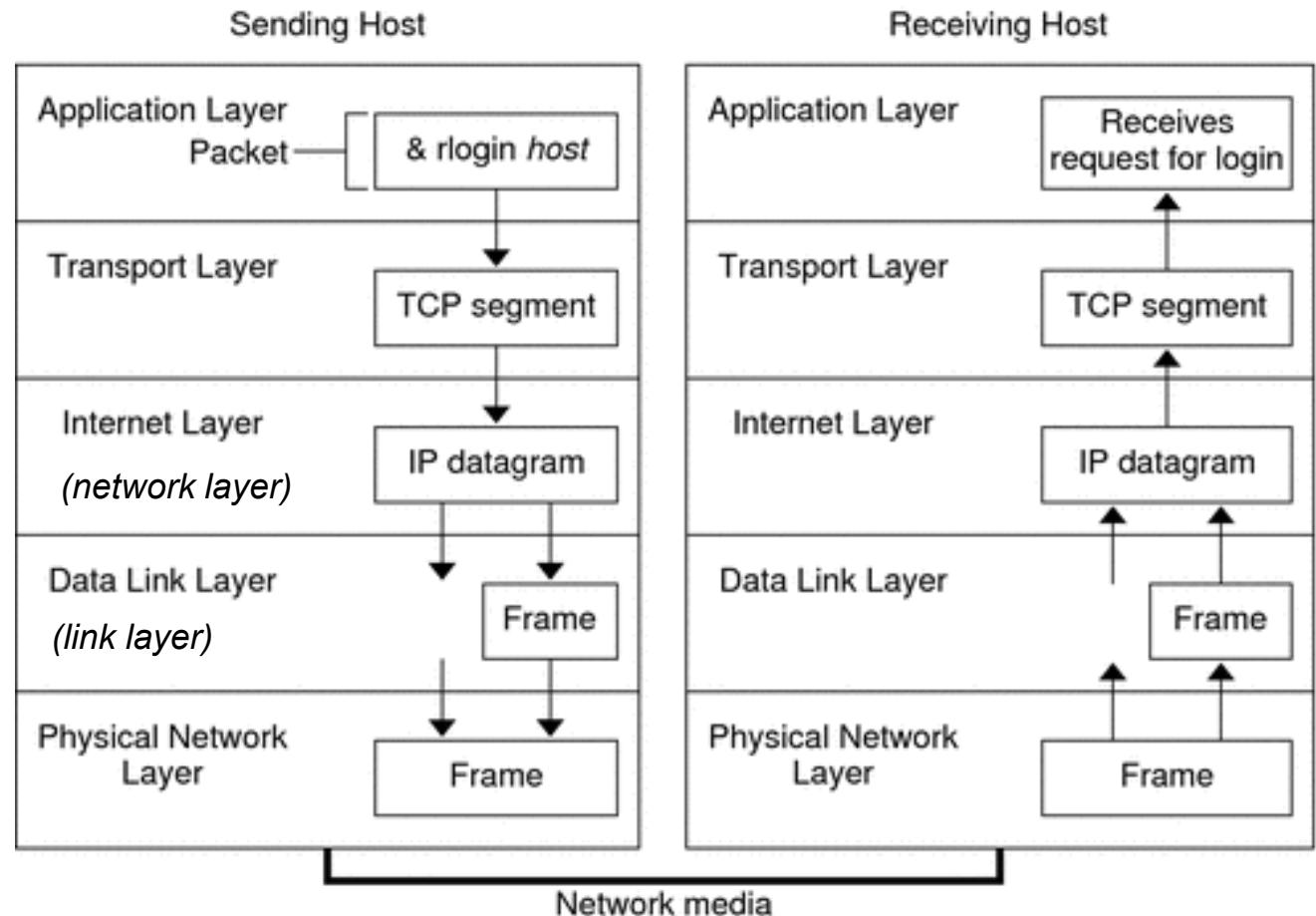
Hinweis: Sie können Wireshark auf Ihrem privaten PC installieren. Im PC-Pool ist dies leider nicht möglich.
Den Screencast finden Sie im Materialordner.

Verbindungsloses (De-)Multiplexing

- Empfängt ein Host ein UDP-Datagramm, dann
 - wird **nur die Ziel-Portnummer** geprüft und
 - das UDP-Datagramm an den Socket gesendet, der diesem Port zugeordnet ist.
- Folgerung: IP-Datagramme mit derselben Ziel-Portnummer aber unterschiedlichen Quell-IP Adressen und/oder Quell-Portnummer werden an denselben Socket auf dem Ziel-Host weitergeleitet.

Verbindungsloses (De-)Multiplexing

- Hinweise zu den Begrifflichkeiten
 - Bei UDP spricht man eigentlich nicht von Segmenten
 - Die Begrifflichkeiten sind nicht strikt definiert; oft spricht man einfach nur von „Paket“.

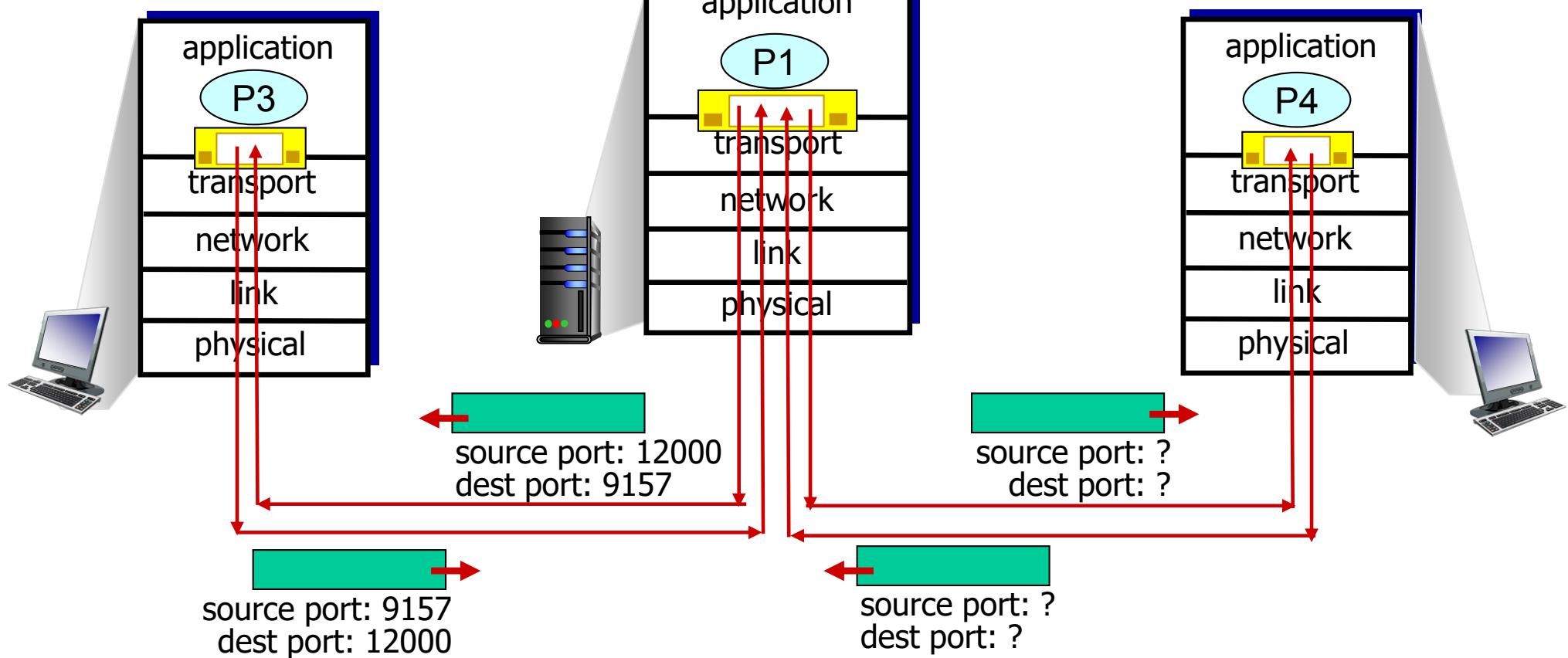


Verbindungsloses (De-)Multiplexing: Beispiel

```
DatagramSocket  
mySocket2 = new  
DatagramSocket  
(9157);
```

```
DatagramSocket  
serverSocket = new  
DatagramSocket  
(12000);
```

```
DatagramSocket  
mySocket1 = new  
DatagramSocket  
(5775);
```

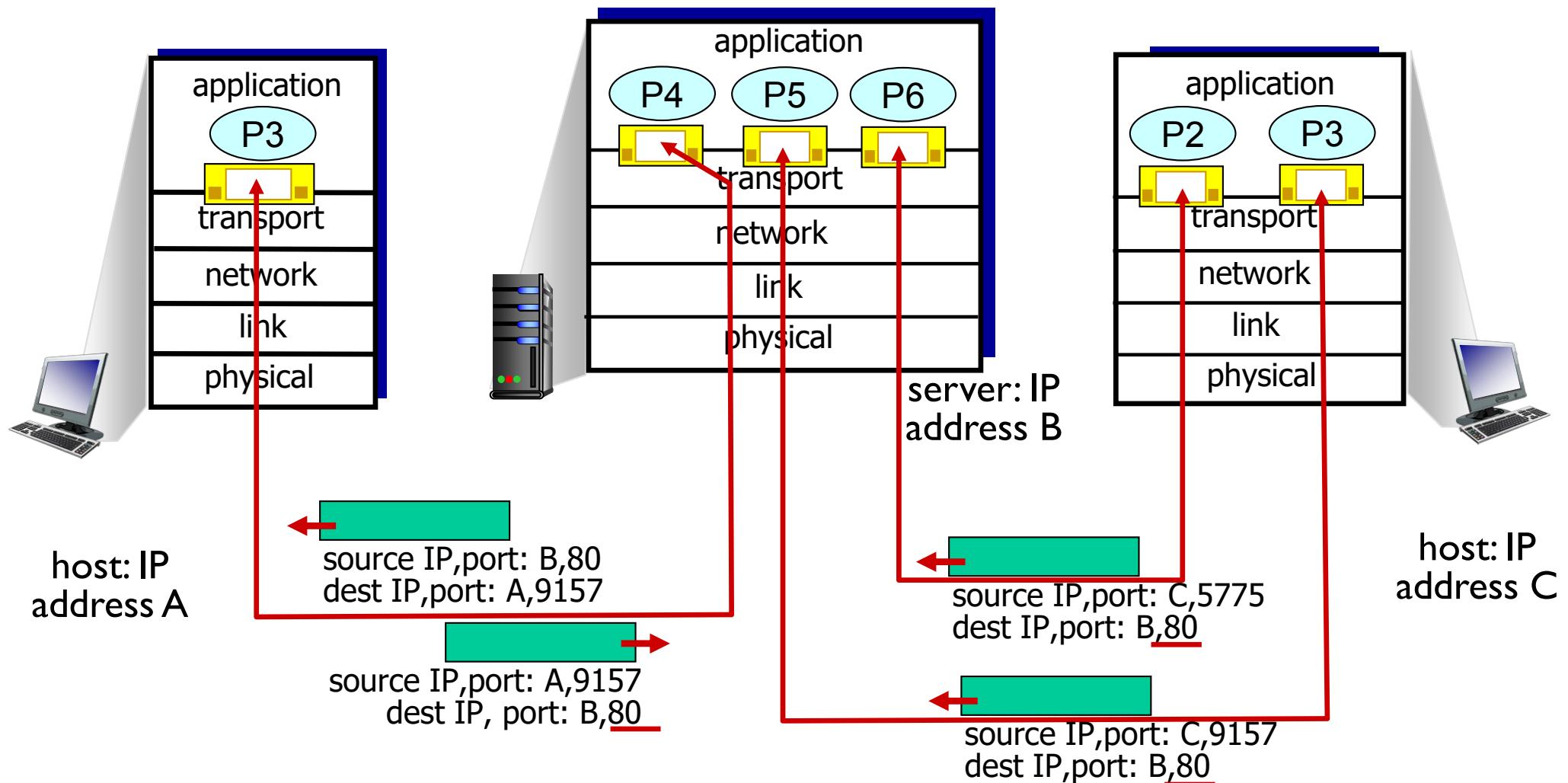


Verbindungsorientiertes (De-)Multiplexing

- TCP-Socket wird durch ein 4-Tupel identifiziert:
 - Absender-IP-Adresse
 - Absender-Portnummer
 - Empfänger-IP-Adresse
 - Empfänger-Portnummer
- Empfänger nutzt **alle vier** Werte, um den richtigen TCP-Socket zu identifizieren
- Ein Server kann viele TCP-Sockets gleichzeitig offen haben:
 - Jeder Socket wird durch sein eigenes 4-Tupel identifiziert
- Webserver haben verschiedene Sockets für jeden einzelnen Client
 - Bei nichtpersistentem HTTP wird jede Anfrage über einen eigenen Socket beantwortet (dieser wird nach jeder Anfrage wieder geschlossen).

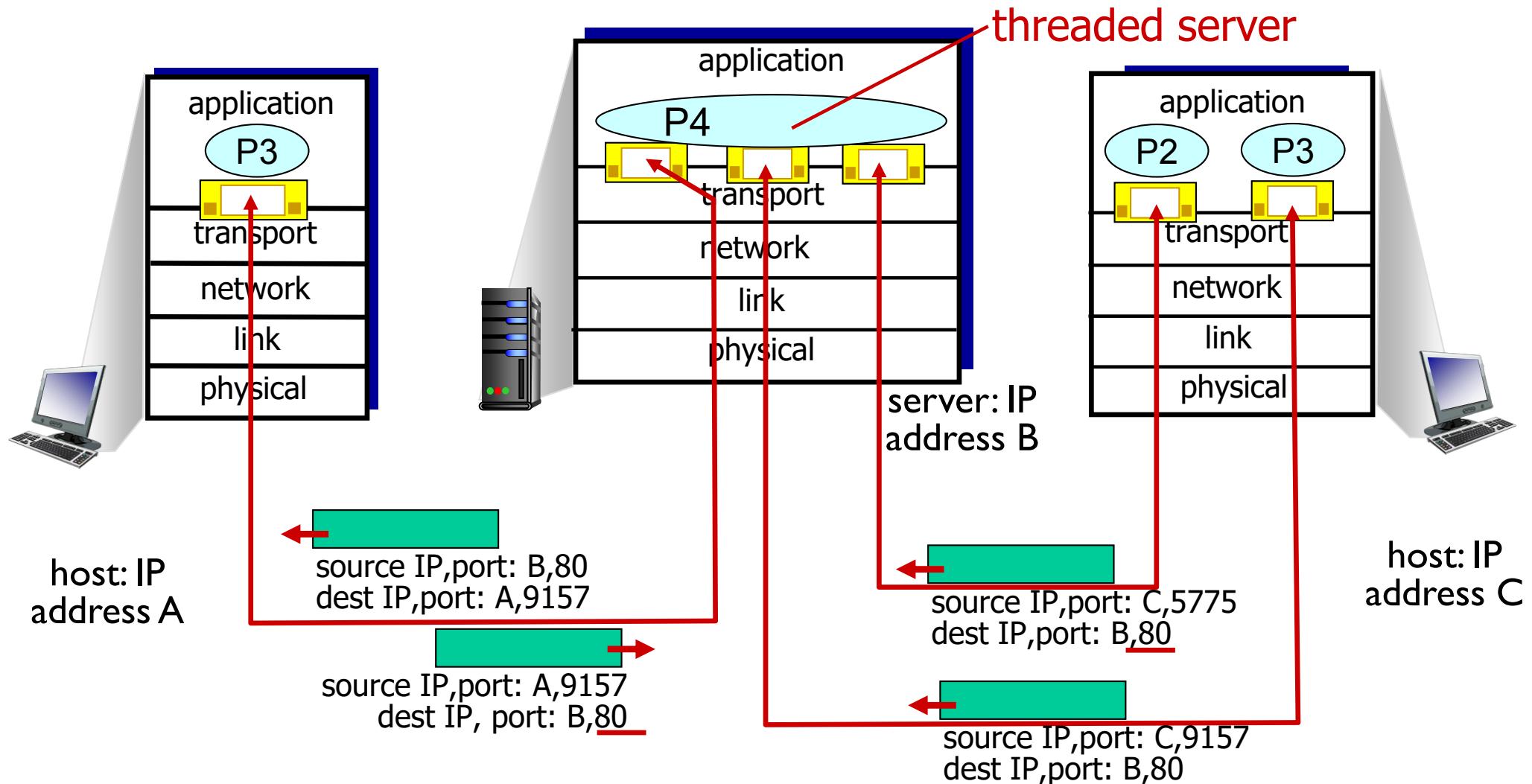
Verbindungsorientiertes (De-)Multiplexing

- Drei Segmente (alle mit Ziel IP-Adresse B und Ziel-Port 80)
 - Diese werden auf unterschiedliche Sockets „demultiplexed“:



Verbindungsorientiertes (De-)Multiplexing

- Meist läuft nur eine Instanz, bspw. eines Webservers, die dann in mehreren nebenläufigen Threads mehrere Sockets verwendet:



Verbindungsorientiertes (De-)Multiplexing: Beispiel

tcp.port == 12000

No.	Time	Source	Destination	Protocol	Length	Info
12387	71.839283868	109.84.1.105	46.101.190.70	TCP	74	4014 → 12000 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 S...
12388	71.839352914	46.101.190.70	109.84.1.105	TCP	74	12000 → 4014 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0...
12389	71.841040712	109.84.1.105	46.101.190.70	TCP	66	4014 → 12000 [ACK] Seq=1 Ack=1 Win=14848 Len=0 TSva...
12390	71.841110206	109.84.1.105	46.101.190.70	TCP	95	4014 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=14848 Len=2...
12391	71.841127437	46.101.190.70	109.84.1.105	TCP	66	12000 → 4014 [ACK] Seq=1 Ack=30 Win=28992 Len=0 TSv...
12392	71.842429589	46.101.190.70	109.84.1.105	TCP	95	12000 → 4014 [PSH, ACK] Seq=1 Ack=30 Win=28992 Len=...
12393	71.842526361	46.101.190.70	109.84.1.105	TCP	66	12000 → 4014 [FIN, ACK] Seq=30 Ack=30 Win=28992 Len=...
12394	71.843598723	109.84.1.105	46.101.190.70	TCP	66	4014 → 12000 [ACK] Seq=30 Ack=30 Win=14848 Len=0 TS...
12407	71.883789247	109.84.1.105	46.101.190.70	TCP	66	4014 → 12000 [ACK] Seq=30 Ack=31 Win=14848 Len=0 TS...
12408	71.900117929	109.84.1.105	46.101.190.70	TCP	66	4014 → 12000 [FIN, ACK] Seq=30 Ack=31 Win=14848 Len=...
12409	71.900172471	46.101.190.70	109.84.1.105	TCP	66	12000 → 4014 [ACK] Seq=31 Ack=31 Win=28992 Len=0 TS...

Frame 12390: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface 0

Ethernet II, Src: JuniperN 34:67:f0 (40:a6:77:34:67:f0), Dst: 86:78:6a:53:2c:57 (86:78:6a:53:2c:57)

Internet Protocol Version 4, Src: 109.84.1.105, Dst: 46.101.190.70

Transmission Control Protocol, Src Port: 4014, Dst Port: 12000, Seq: 1, Ack: 1, Len: 29

Source Port: 4014
Destination Port: 12000
[Stream index: 4]
[TCP Segment Len: 29]
Sequence number: 1 (relative sequence number)
[Next sequence number: 30 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Header Length: 32 bytes
Flags: 0x018 (PSH, ACK)
Window size value: 29
[Calculated window size: 14848]
[Window size scaling factor: 512]
Checksum: 0x46c4 [unverified]
[Checksum Status: Unverified]

0000	86 78 6a 53 2c 57 40 a6 77 34 67 f0 08 00 45 00	.xjS,W@. w4g...E.
0010	00 51 b5 1a 40 00 35 06 35 24 6d 54 01 69 2e 65	.Q..@5. 5\$mT.i.e
0020	be 46 0f ae 2e e0 b7 5c ec 9d 17 dc 20 66 80 18	.F.....\ f..
0030	00 1d 46 c4 00 00 01 01 08 0a 90 78 59 86 22 6e	..F..... ...xY."n
0040	67 43 54 68 69 73 20 69 73 20 74 68 65 20 54 43	gCThis i s the TC
0050	50 20 74 65 73 74 20 6d 65 73 73 61 67 65 2e	P test m essage.

Verbindungsorientiertes (De-)Multiplexing: Beispiel

tcp.port == 12000

No.	Time	Source	Destination	Protocol	Length	Info
12387	71.839283868	109.84.1.105	46.101.190.70	TCP	74	4014 → 12000 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 S...
12388	71.839352914	46.101.190.70	109.84.1.105	TCP	74	12000 → 4014 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0...
12389	71.841040712	109.84.1.105	46.101.190.70	TCP	66	4014 → 12000 [ACK] Seq=1 Ack=1 Win=14848 Len=0 TSva...
12390	71.841110206	109.84.1.105	46.101.190.70	TCP	95	4014 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=14848 Len=2...
12391	71.841127437	46.101.190.70	109.84.1.105	TCP	66	12000 → 4014 [ACK] Seq=1 Ack=30 Win=28992 Len=0 TSv...
12392	71.842429589	46.101.190.70	109.84.1.105	TCP	95	12000 → 4014 [PSH, ACK] Seq=1 Ack=30 Win=28992 Len=...
12393	71.842526361	46.101.190.70	109.84.1.105	TCP	66	12000 → 4014 [FIN, ACK] Seq=30 Ack=30 Win=28992 Len=...
12394	71.843598723	109.84.1.105	46.101.190.70	TCP	66	4014 → 12000 [ACK] Seq=30 Ack=30 Win=14848 Len=0 TS...
12407	71.883789247	109.84.1.105	46.101.190.70	TCP	66	4014 → 12000 [ACK] Seq=30 Ack=31 Win=14848 Len=0 TS...
12408	71.900117929	109.84.1.105	46.101.190.70	TCP	66	4014 → 12000 [FIN, ACK] Seq=30 Ack=31 Win=14848 Len=...
12409	71.900172471	46.101.190.70	109.84.1.105	TCP	66	12000 → 4014 [ACK] Seq=31 Ack=31 Win=28992 Len=0 TS...

Frame 12392: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface 0

Ethernet II, Src: 86:78:6a:53:2c:57 (86:78:6a:53:2c:57), Dst: IETF-VRRP-VRID_64 (00:00:5e:00:01:64)

Internet Protocol Version 4, Src: 46.101.190.70, Dst: 109.84.1.105

Transmission Control Protocol, Src Port: 12000, Dst Port: 4014, Seq: 1, Ack: 30, Len: 29

Source Port: 12000
Destination Port: 4014
[Stream index: 4]
[TCP Segment Len: 29]
Sequence number: 1 (relative sequence number)
[Next sequence number: 30 (relative sequence number)]
Acknowledgment number: 30 (relative ack number)
Header Length: 32 bytes
Flags: 0x018 (PSH, ACK)
Window size value: 453
[Calculated window size: 28992]
[Window size scaling factor: 64]
Checksum: 0x5bac [unverified]
[Checksum Status: Unverified]

0000	00 00 5e 00 01 64 86 78 6a 53 2c 57 08 00 45 00	..^..d.x jS,W..E.
0010	00 51 47 23 40 00 40 06 98 1b 2e 65 be 46 6d 54	.QG#@. @. ...e.FmT
0020	01 69 2e e0 0f ae 17 dc 20 66 b7 5c ec ba 80 18	.i..... f.\....
0030	01 c5 5b ac 00 00 01 01 08 0a 22 6e 67 44 90 78	...[..... ..."ngD.X
0040	59 86 54 48 49 53 20 49 53 20 54 48 45 20 54 43	Y.THIS I S THE TO
0050	50 20 54 45 53 54 20 4d 45 53 53 41 47 45 2e	P TEST M ESSAGE.

- Dienste der Transportschicht
- Multiplexing und Demultiplexing
- Verbindungslose Übertragung: UDP
- Zuverlässige Datenübertragung
- Verbindungsorientierte Übertragung: TCP
- Überlastkontrolle: Grundlagen und Umsetzung in TCP
- Zusammenfassung und Ausblick

UDP: User Datagram Protocol – Grundlagen

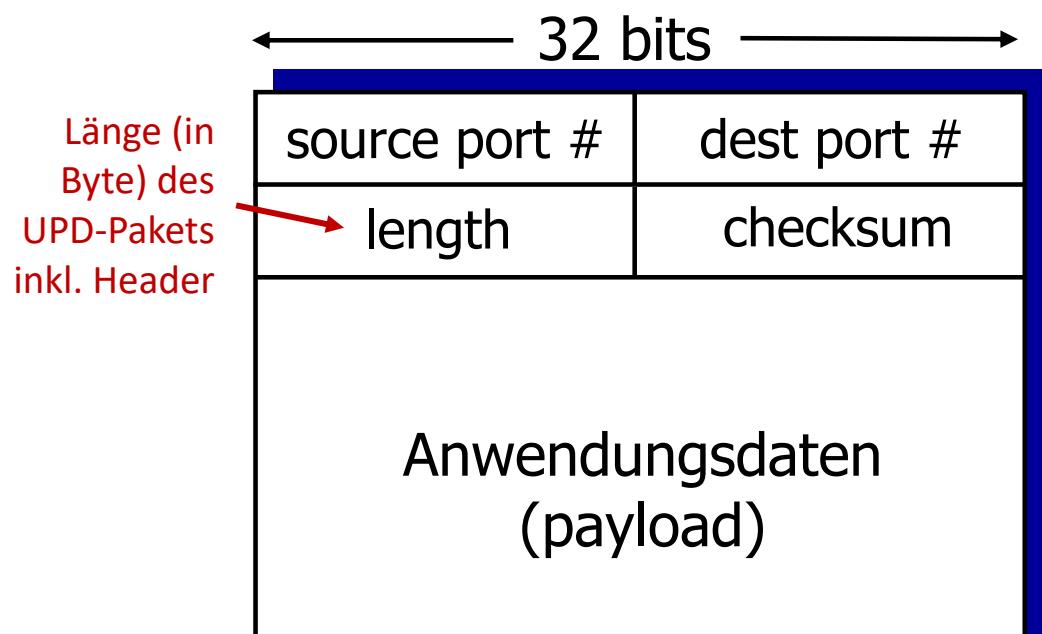
- Minimales Internet-Transportprotokoll
- “Best-Effort”-Dienst, UDP-Pakete können
 - **verloren** gehen und
 - in der **falschen Reihenfolge** an die Anwendung ausgeliefert werden.
- **Verbindungslos**
 - Kein Handshake zum Verbindungsaufbau
 - Jedes UDP-Paket wird unabhängig von allen anderen behandelt.

Warum gibt es UDP?

- Kein Verbindungsaufbau (der zu Verzögerungen führen kann)
- Einfach und effizient: kein Verbindungszustand im Sender oder Empfänger
- Kleiner Header
- Keine Überlastkontrolle: UDP kann so schnell senden, wie von der Anwendung gewünscht

UDP: User Datagram Protocol – Grundlagen

- Häufig für Anwendungen im Bereich Multimedia-Streaming eingesetzt
 - verlusttolerant
 - Mindestrate
- Andere Einsatzgebiete
 - DNS
 - SNMP
- Zuverlässiger Datentransfer über UDP: „Zuverlässigkeit“ muss auf Anwendungsschicht implementiert werden.
 - Anwendungsspezifische Fehlerkorrektur!
- UDP Paketformat (Datagramm)
 - Source port # = Quellportnummer
 - Dest port # = Zielportnummer
 - Checksum = Prüfsumme



UDP: Fehlererkennung mit Prüfsummen

Ziel: Fehler im übertragenen Datagramm erkennen (z.B. verfälschte Bits)

Sender

- Betrachte Paket als Folge von 16-Bit-Integer-Werten
- Prüfsumme: Addition (1er-Komplement-Summe) dieser Werte
 - Achtung: evtl. Überträge, s.u.!
- Sender legt das invertierte Resultat im UDP-Prüfsummenfeld ab

Empfänger

- Berechne die Prüfsumme des empfangenen Pakets inkl. des Prüfsummenfeldes
- Sind im Resultat alle Bits 1?
 - NEIN – Fehler erkannt
 - JA – kein Fehler erkannt
Vielleicht liegt trotzdem ein Fehler vor? Mehr dazu später
...

UDP: Fehlererkennung mit Prüfsummen

- **Beispiel (Sender):** addiert zwei 16-Bit-Integer Werte (Paketdaten)

Paket	{	1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
Übertrag		1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 ↑ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
Summe		1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
Prüfsumme		0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1

Übertrag auf
Ergebnis an der
niedrigsten
Stelle addieren

- Hinweis: Ein eventuell auftretender Übertrag aus der höchsten Stelle wird zum Resultat an der niedrigsten Stelle addiert

UDP: Fehlererkennung mit Prüfsummen

- **Beispiel (Empfänger):** addiert alle drei 16-Bit-Integer Werte
(Paketdaten + Prüfsumme)

Paket	{	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
Prüfsumme		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	
<hr/>																		
		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0	←
		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	
<hr/>																		

Summe 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

- Ist die Summe 1111111111111111 dann ist das Paket korrekt übertragen worden.
- Falls nicht, dann wird die Anwendung darüber informiert, dass es einen Fehler gab.
 - Die Anwendung muss dann entscheiden, wie damit umgegangen wird.
 - Man nennt dies Fehlererkennung auf Ende-zu-Ende Basis, d.h. es wird sich nicht auf eventuelle Prüfmechanismen auf den unteren Protokollschichten verlassen.



Übung

- Berechnen Sie die Prüfsumme eines Datenpakets, das die folgenden drei 16-Bit Werte enthält:

0110011001100000

0101010101010101

1000111100001100

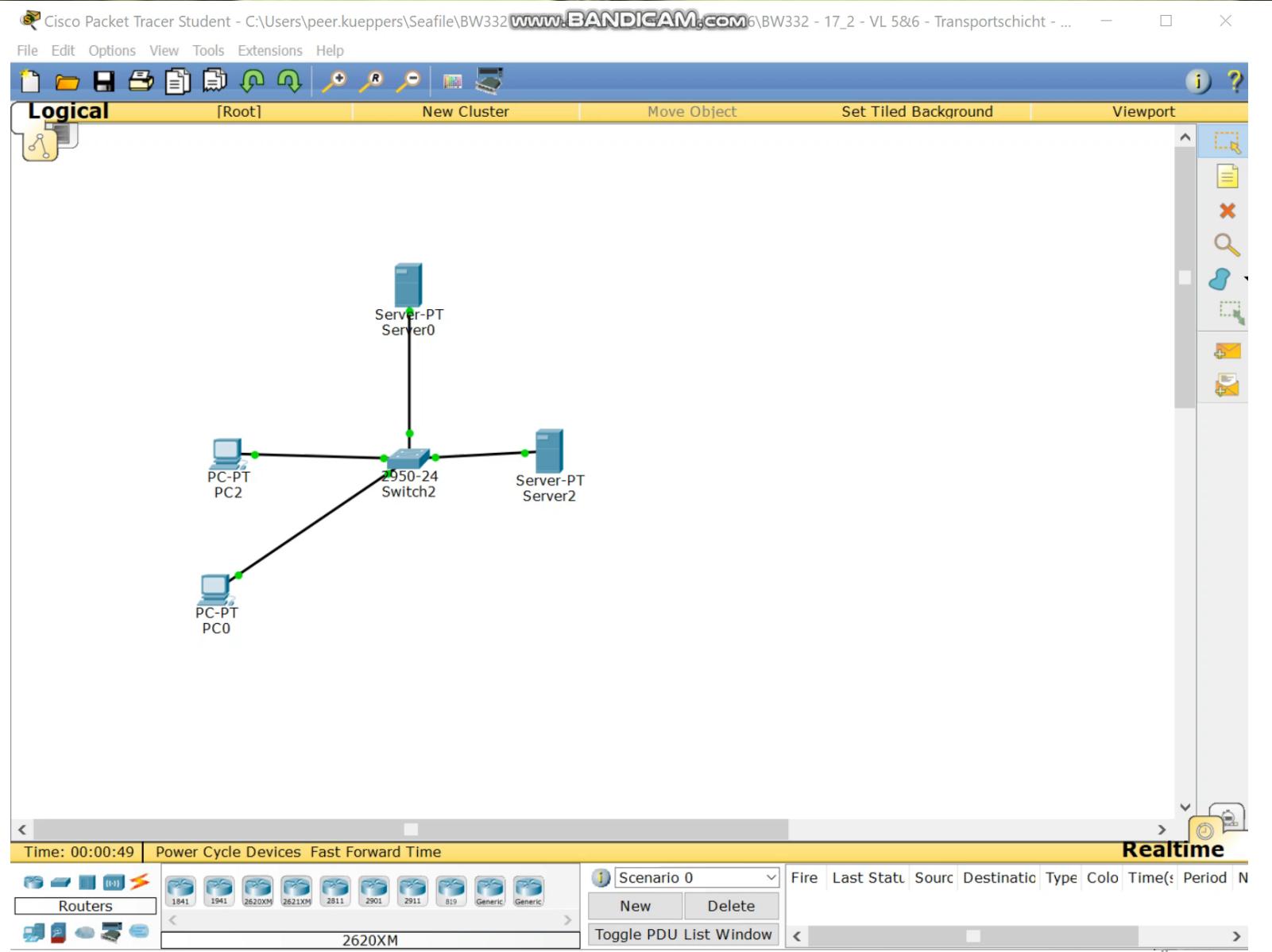


Übung

- Das DNS-Protokoll basiert auf UDP (s.o.).
 - Wir haben dieses also bereits verwendet.
 - Wir können unser Cisco Packet Tracer-Projekt aus der letzten Veranstaltung aus einer weiteren Perspektive betrachten: nicht mehr nur die Anwendungsschicht, sondern auch die Transportschicht mit DNS / UDP!
- Rufen Sie Ihr Cisco Packet Tracer-Projekt auf.
 - Benennen Sie für eine bessere Unterscheidung die Server in „DNS“ und „Webserver“ um.
 - Wechseln Sie in den Simulationsmodus und rufen Sie vom Client aus die Seite www.hs-lu.de auf.
 - Betrachten Sie die UDP-Pakete, auch im Hinblick auf die verwendeten Ports.

Übung

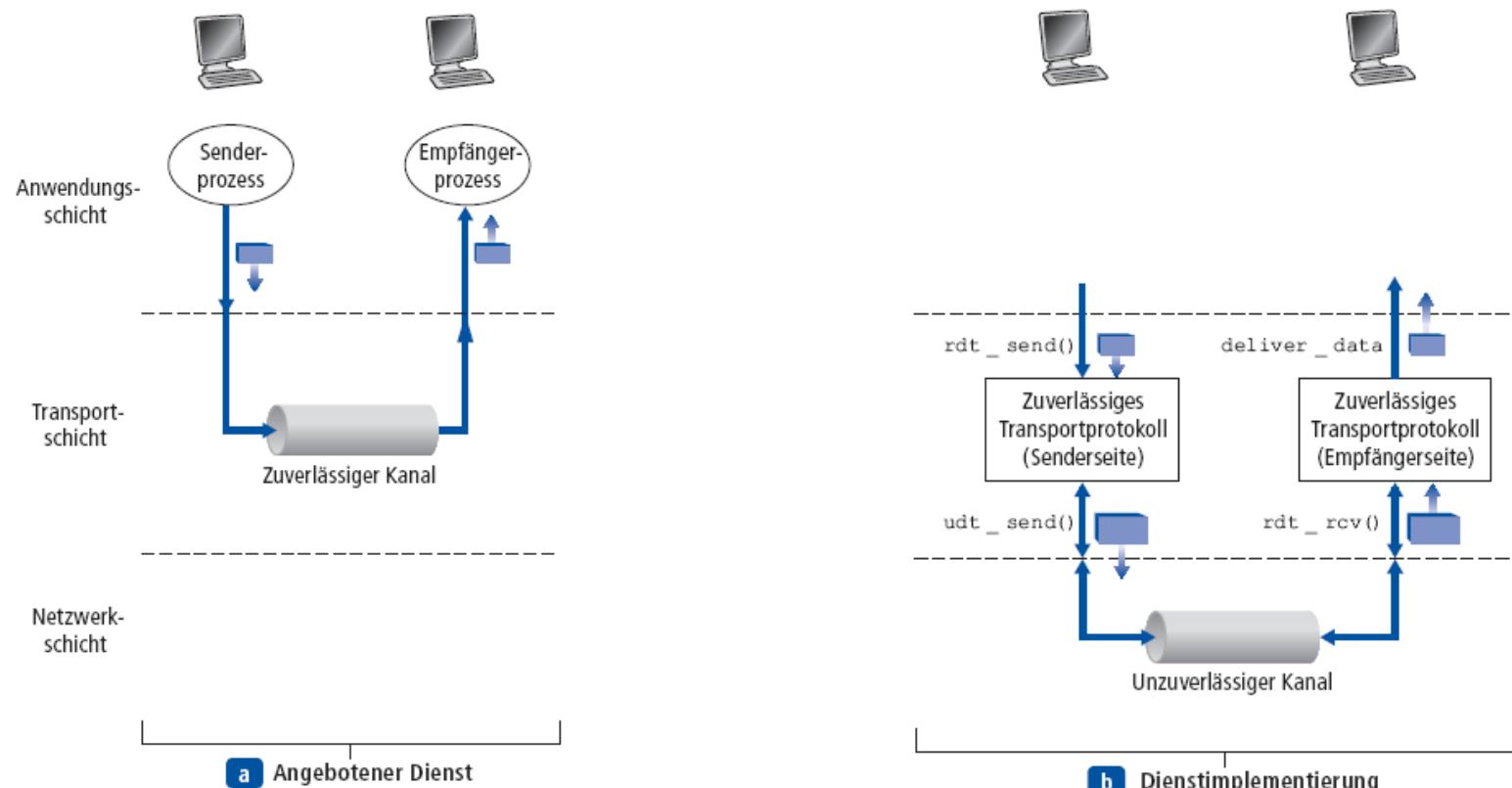
■ Beispiel:



- Dienste der Transportschicht
- Multiplexing und Demultiplexing
- Verbindungslose Übertragung: UDP
- Zuverlässige Datenübertragung
- Verbindungsorientierte Übertragung: TCP
- Überlastkontrolle: Grundlagen und Umsetzung in TCP
- Zusammenfassung und Ausblick

Grundlagen der zuverlässigen Datenübertragung

- Wichtig für Anwendungs-, Transport- und Sicherungsschicht
 - Gehört zu den Top 10 der wichtigsten Netzwerkprobleme!

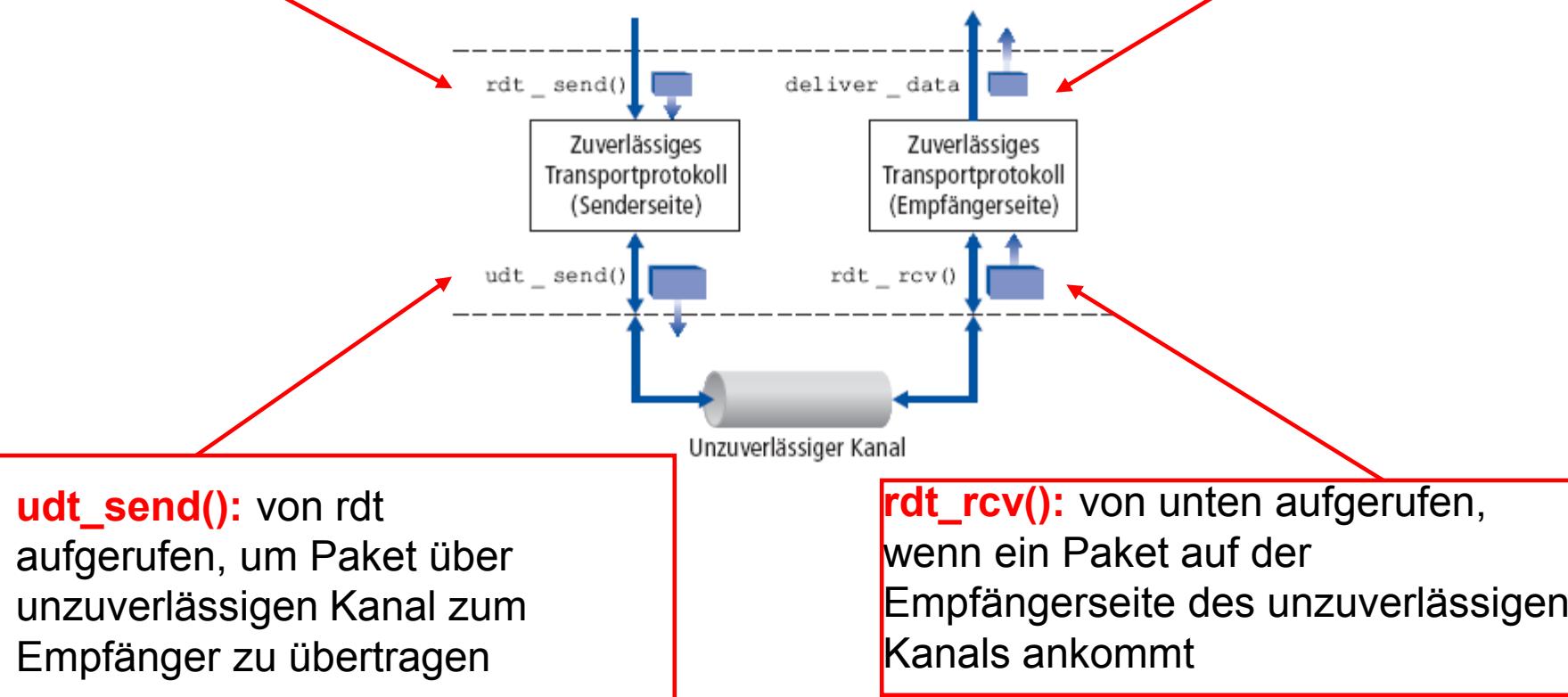


- Die Eigenschaften des unzuverlässigen Kanals bestimmen die Komplexität des Protokolls zur zuverlässigen Datenübertragung (reliable data transfer, rdt).

Erste Schritte der zuverlässigen Datenübertragung

rdt_send(): von oben (z.B. der Anwendung) aufgerufen, übergebene Daten sollen beim Empfänger nach oben ausgeliefert werden

deliver_data(): von rdt aufgerufen, um Daten nach oben weiterzugeben



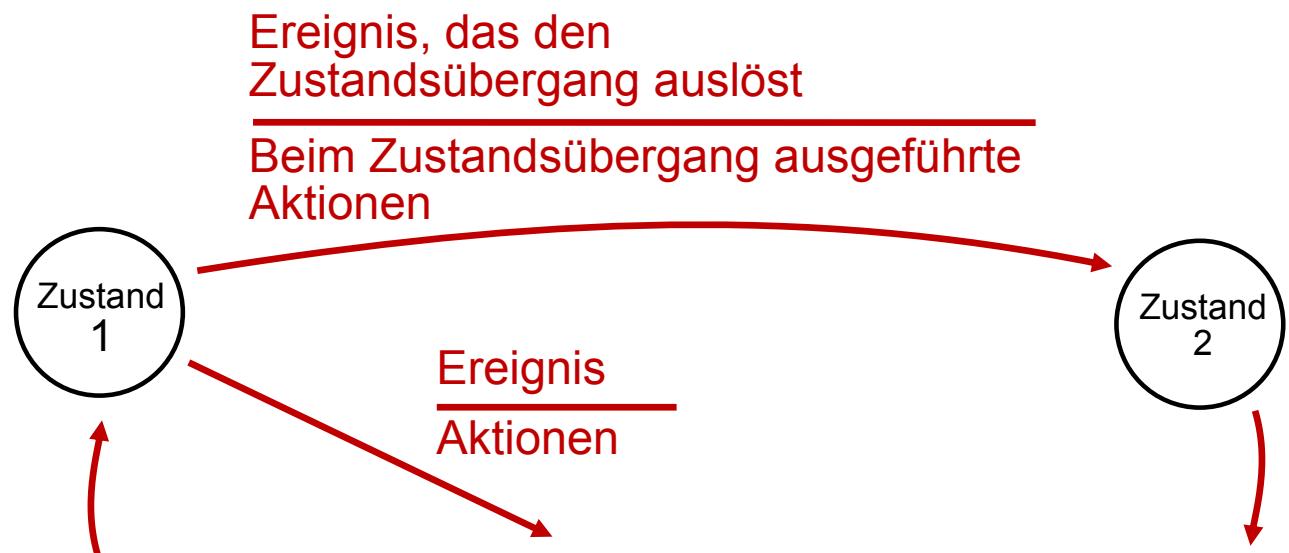
udt_send(): von rdt aufgerufen, um Paket über unzuverlässigen Kanal zum Empfänger zu übertragen

rdt_recv(): von unten aufgerufen, wenn ein Paket auf der Empfängerseite des unzuverlässigen Kanals ankommt

Vorgehen zur Erklärung der zuverlässigen Datenübertragung

- Wir werden inkrementell ein Protokoll zur zuverlässigen Übertragung von Daten über unzuverlässige Kanäle entwickeln.
- Betrachtet wird nur unidirektionaler Datenverkehr
 - Aber Kontrollinformationen können in beide Richtungen fließen!
- Verwendung endlicher Automaten (finite state machines, FSM), um Sender und Empfänger zu spezifizieren:

Zustand: Durch den Zustand und ein Ereignis ist der Nachfolgezustand bestimmt



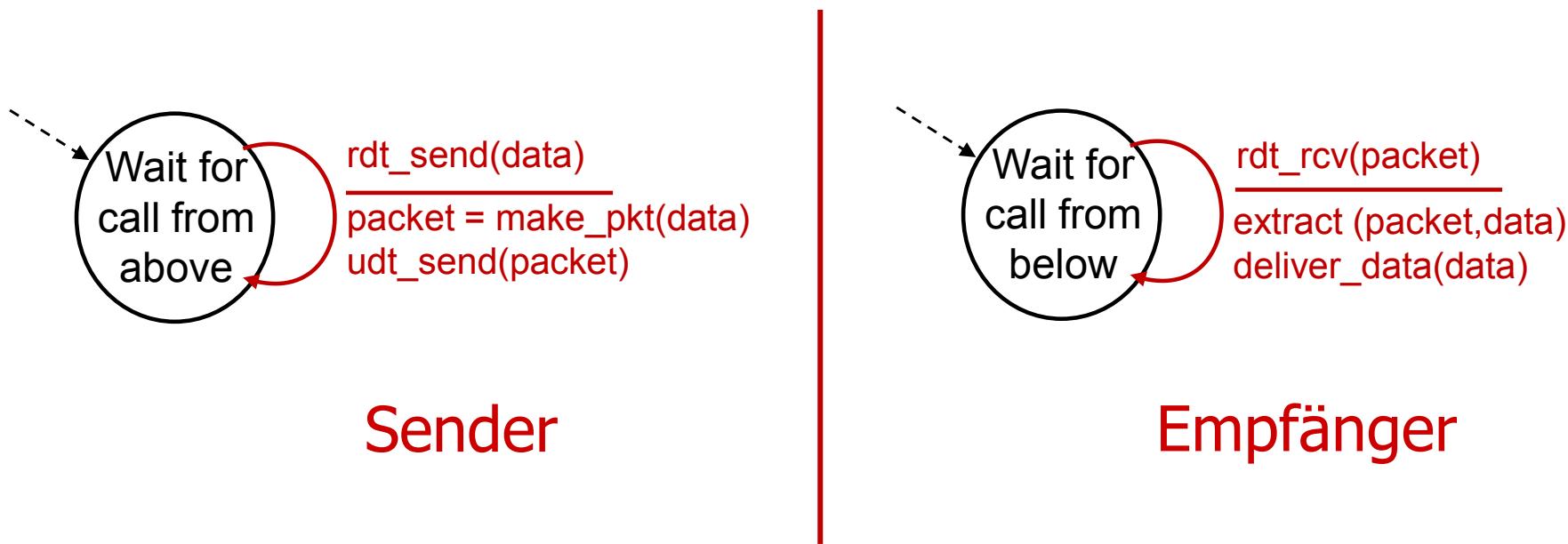
Vorgehen zur Erklärung der zuverlässigen Datenübertragung

- rdt1.0: zuverlässige Übertragung über einen **zuverlässigen Kanal**
- rdt2.0: zuverlässige Übertragung über einen **Kanal mit Bitfehlern**
 - rdt2.1: Behandlung von verfälschten ACKs/NAKs
 - rdt2.2: Protokoll ohne NAKs
- rdt3.0: zuverlässige Übertragung über einen **Kanal mit Bitfehlern und Paketverlusten**

- Protokolle mit Pipelining
 - Go-Back-N (GBN)
 - Selektive Wiederholung (Selective Repeat)

rdt1.0: zuverlässiger Kanal

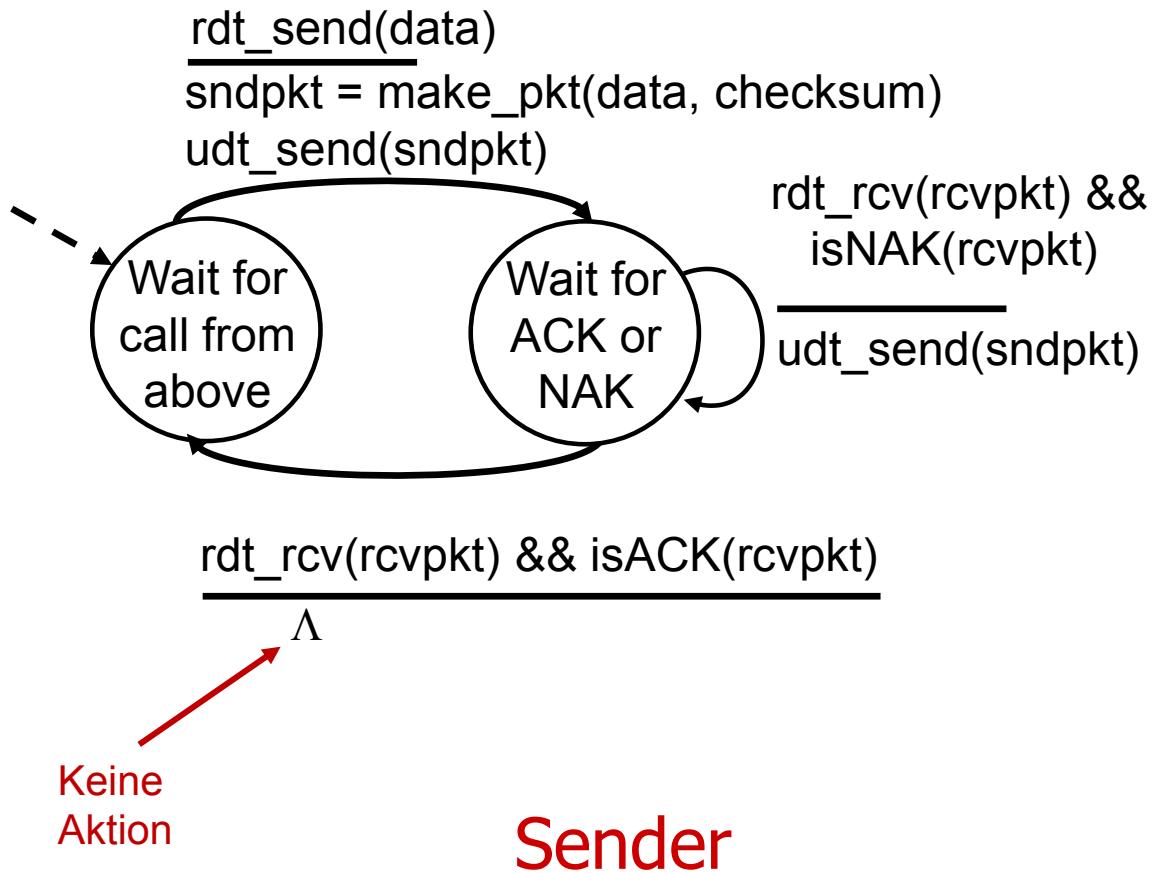
- Der Übertragungskanal ist **absolut zuverlässig**:
 - Keine Verfälschung von Bits
 - Kein Verlust ganzer Pakete
- Je ein endlicher Automat für Sender und Empfänger:
 - Sender übergibt Daten an den zuverlässigen Kanal
 - Empfänger erhält die Daten vom zuverlässigen Kanal



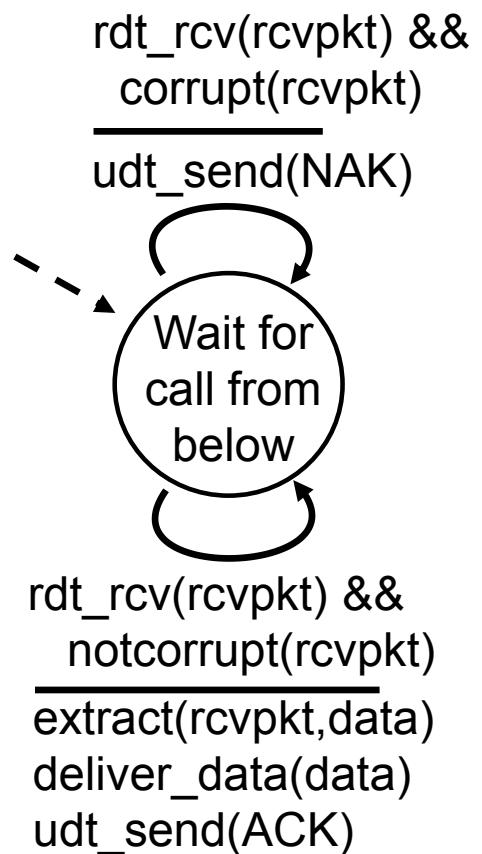
rdt2.0: Kanal mit Bitfehlern

- Der Kanal kann Bits verfälschen.
 - Wir gehen davon aus, dass dies durch eine Prüfsumme erkannt wird.
- Wie kann man nun eine zuverlässige Datenübertragung erreichen?
 - **Acknowledgements (ACKs)**: Empfänger sagt dem Sender explizit, dass das Paket erfolgreich empfangen wurde.
 - **Negative Acknowledgements (NAKs)**: Empfänger sagt dem Sender explizit, dass das Paket fehlerbehaftet war.
 - Sender wiederholt Übertragung für diese Pakete.
- Neue Mechanismen in rdt2.0:
 - Fehlererkennung
 - Rückmeldung durch den Empfänger:
 - Kontrollnachrichten vom Empfänger an den Sender

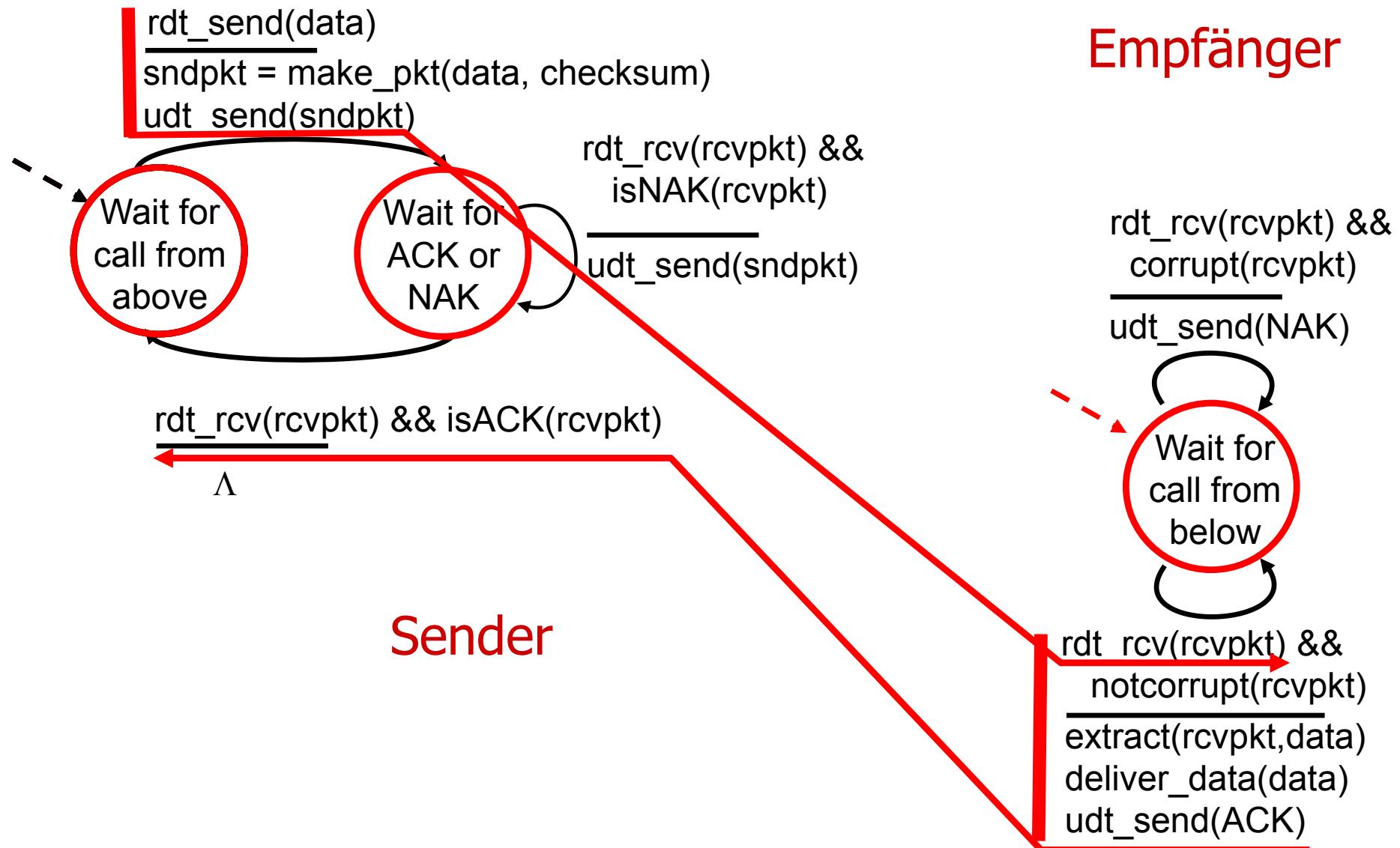
rdt2.0: FSM-Spezifikation



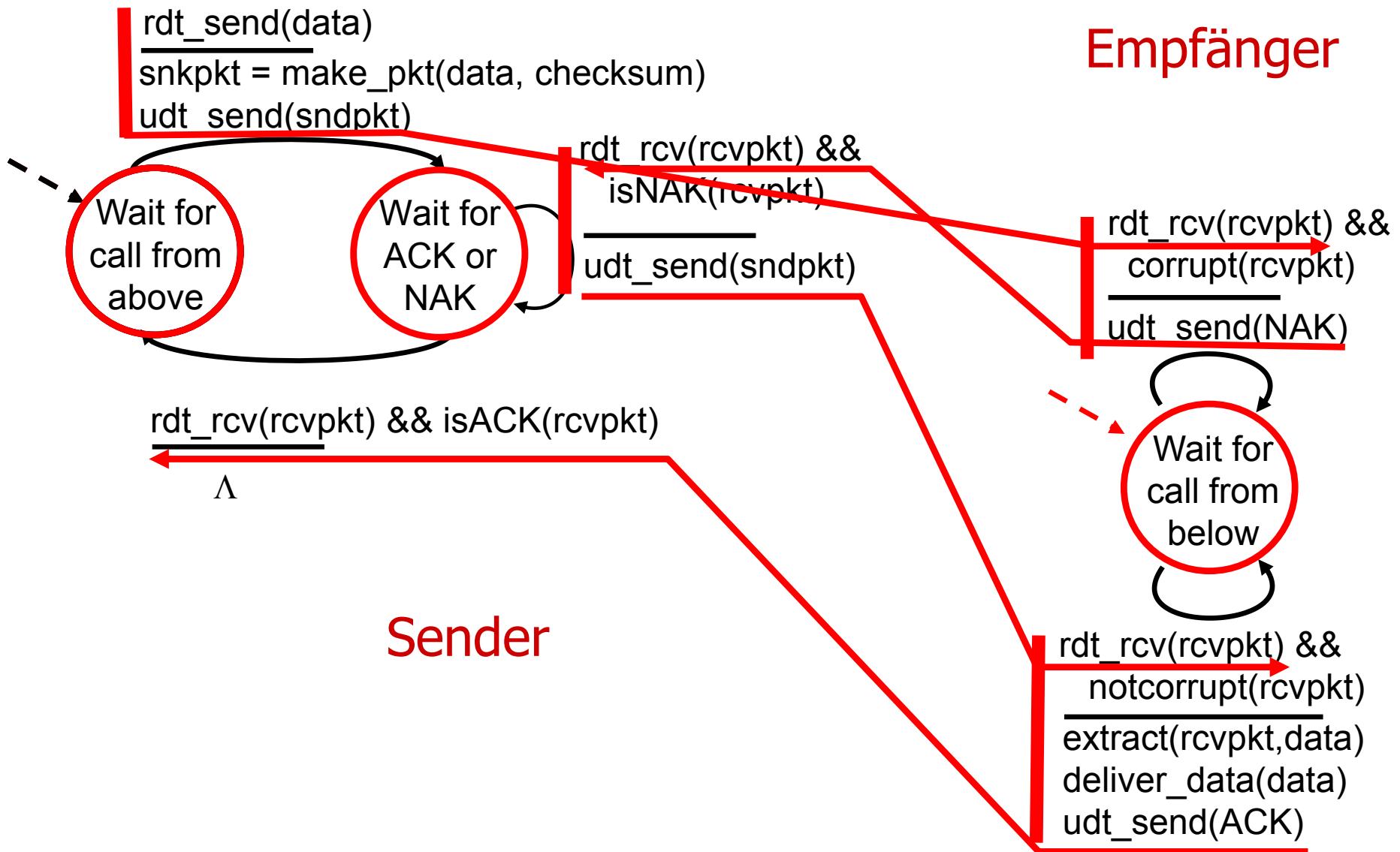
Empfänger



rdt2.0: Ablauf ohne Fehler



rdt2.0: Ablauf mit Fehler



rdt2.0: Problem – rdt2.0 funktioniert nicht

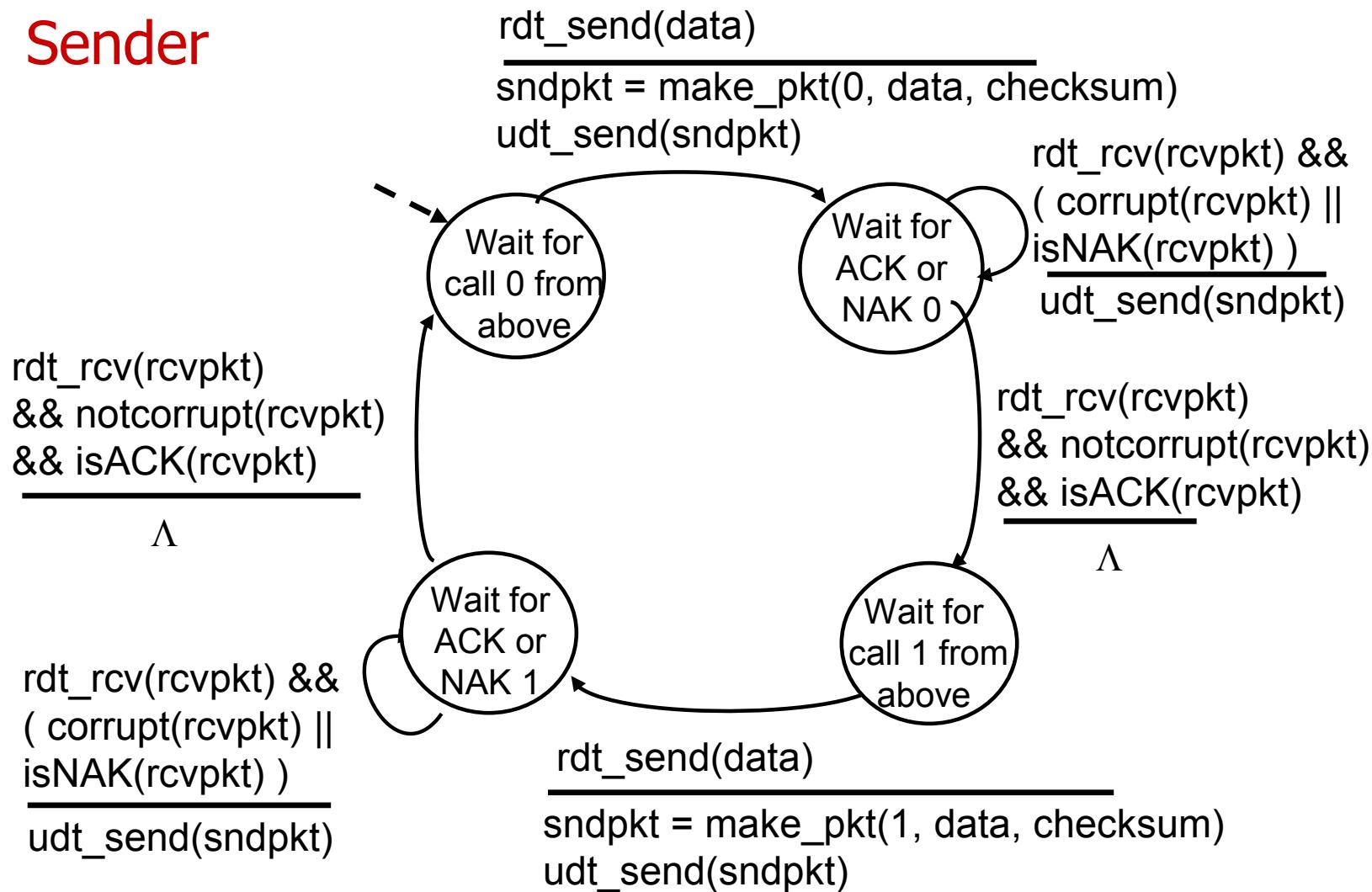
- Was passiert, wenn ein ACK/NAK verfälscht wird?
 - Sender weiß nicht, was beim Empfänger passiert ist
- Wie kann man dies lösen?
 - Sender sendet ACK/NAK für das ACK/NAK des Empfängers? Was passiert, wenn dieses verfälscht wird?
 - Übertragungswiederholung?
 - Kann zur erneuten Übertragung eines bereits korrekt empfangenen Pakets führen
- Behandlung von Duplikaten:
 - Sender wiederholt die Übertragung, wenn ACK/NAK verfälscht wurde
 - Dabei wird eine **eindeutige Sequenznummer** im Paket hinzugefügt
 - Empfänger verwirft Pakete mit der gleichen Sequenznummer

stop and wait

Sender sendet ein Paket und wartet dann auf Antwort vom Empfänger

rdt2.1: Handling von verfälschten ACK/NAKs

Sender



Achtung: wir nehmen in rdt2.1 zwar Bitfehler an, aber ganze Pakete gehen nicht verloren!

rdt2.1: Handling von verfälschten ACK/NAKs

Empfänger

rdt_rcv(rcvpkt) && (corrupt(rcvpkt))

sndpkt = make_pkt(NAK, chks)
 udt_send(sndpkt)

rdt_rcv(rcvpkt) &&
 not corrupt(rcvpkt) &&
 has_seq1(rcvpkt)

sndpkt = make_pkt(ACK, chks)
 udt_send(sndpkt)

rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
 && has_seq0(rcvpkt)

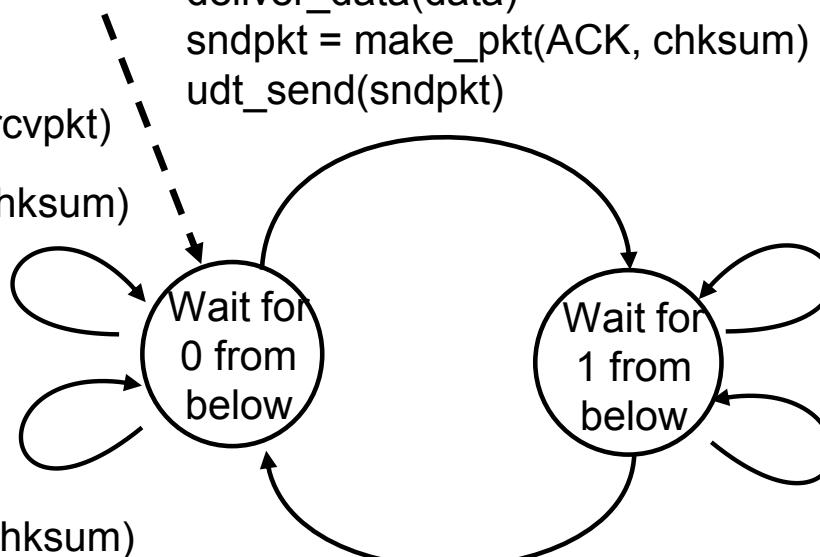
extract(rcvpkt,data)
 deliver_data(data)
 sndpkt = make_pkt(ACK, chks)
 udt_send(sndpkt)

rdt_rcv(rcvpkt) && corrupt(rcvpkt)

sndpkt = make_pkt(NAK, chks)
 udt_send(sndpkt)

rdt_rcv(rcvpkt) &&
 not corrupt(rcvpkt) &&
 has_seq0(rcvpkt)

sndpkt = make_pkt(ACK, chks)
 udt_send(sndpkt)



rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
 && has_seq1(rcvpkt)

extract(rcvpkt,data)
 deliver_data(data)
 sndpkt = make_pkt(ACK, chks)
 udt_send(sndpkt)

Achtung: wir nehmen in rdt2.1 zwar Bitfehler an, aber ganze Pakete gehen nicht verloren!

rdt2.1: Diskussion

Sender

- Sequenznummern hinzugefügt
- Zwei Sequenznummern reichen (0,1).
- Verfälschte ACKs und NAKs werden korrekt behandelt
- Doppelte Anzahl von Zuständen im Vergleich zu rdt2.0:
 - Zustände müssen sich „merken“, ob das aktuelle Paket die Sequenznummer 0 oder 1 hat

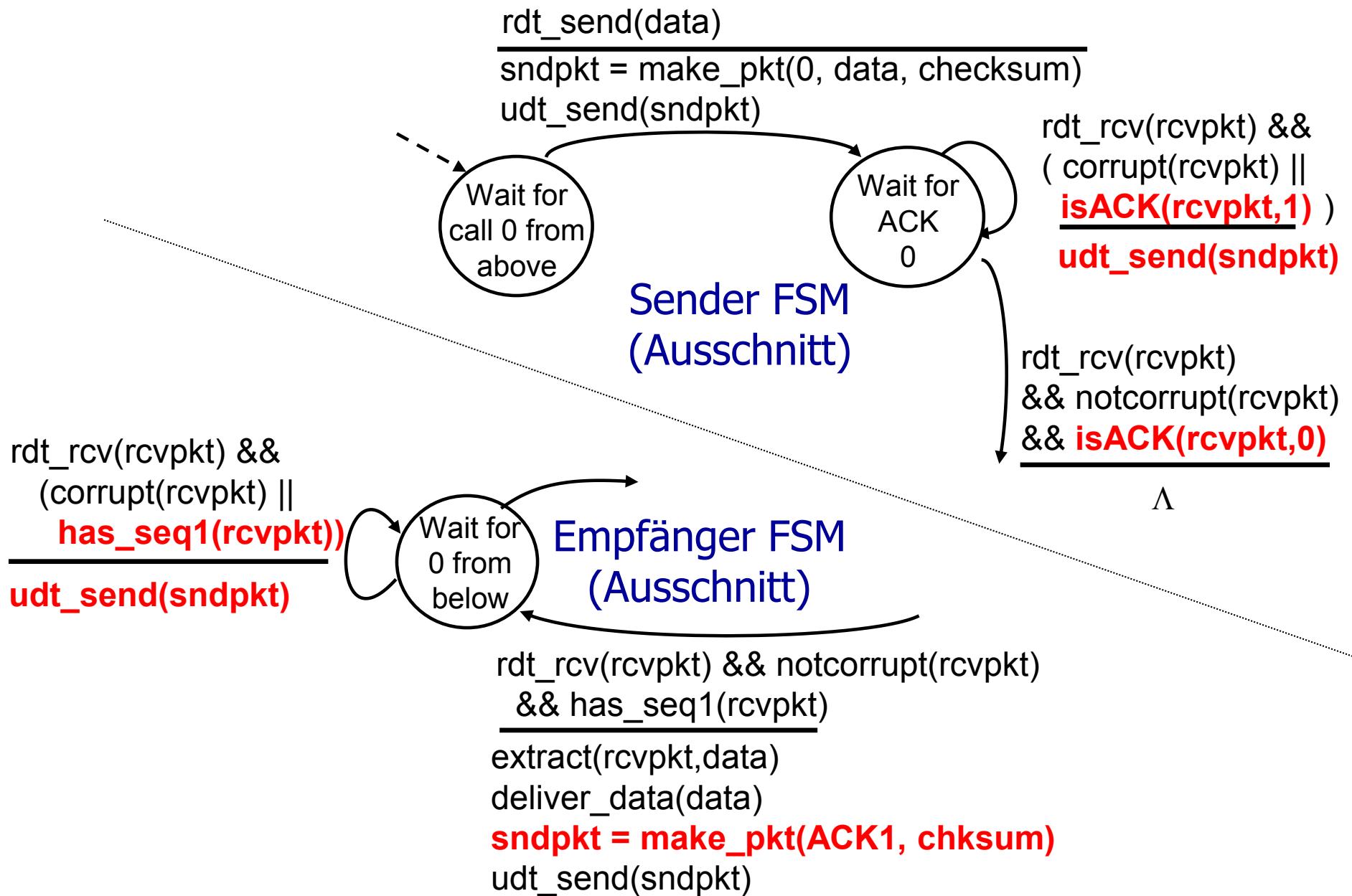
Empfänger

- Muss überprüfen, ob empfangene Pakete Duplikate sind
 - Zustand bestimmt, ob die nächste erwartete Sequenznummer 0 oder 1 ist
- Wichtig: Der Empfänger weiß NICHT, ob der Sender das letzte ACK/NAK unverfälscht empfangen hat!
 - Lässt sich erst am nächsten empfangenen Datenpaket erkennen

rdt2.2: Verzicht auf NAKs

- Gleiche Funktionalität wie rdt2.1
- Aber: keine NAKs!
- Anstelle von NAKs: Empfänger schickt ein ACK für das letzte korrekt empfangene Paket
 - Empfänger muss die Sequenznummer des bestätigten Paketes im ACK mitschicken
- „Alte“ Sequenznummer im ACK (also ein Duplikat) wird vom Sender als NAK interpretiert: das aktuelle Paket muss dann erneut gesendet werden.

rdt2.2: Verzicht auf NAKs



rdt3.0: Kanäle mit Bitfehlern und kompletten Paketverlusten

Neue Annahme:

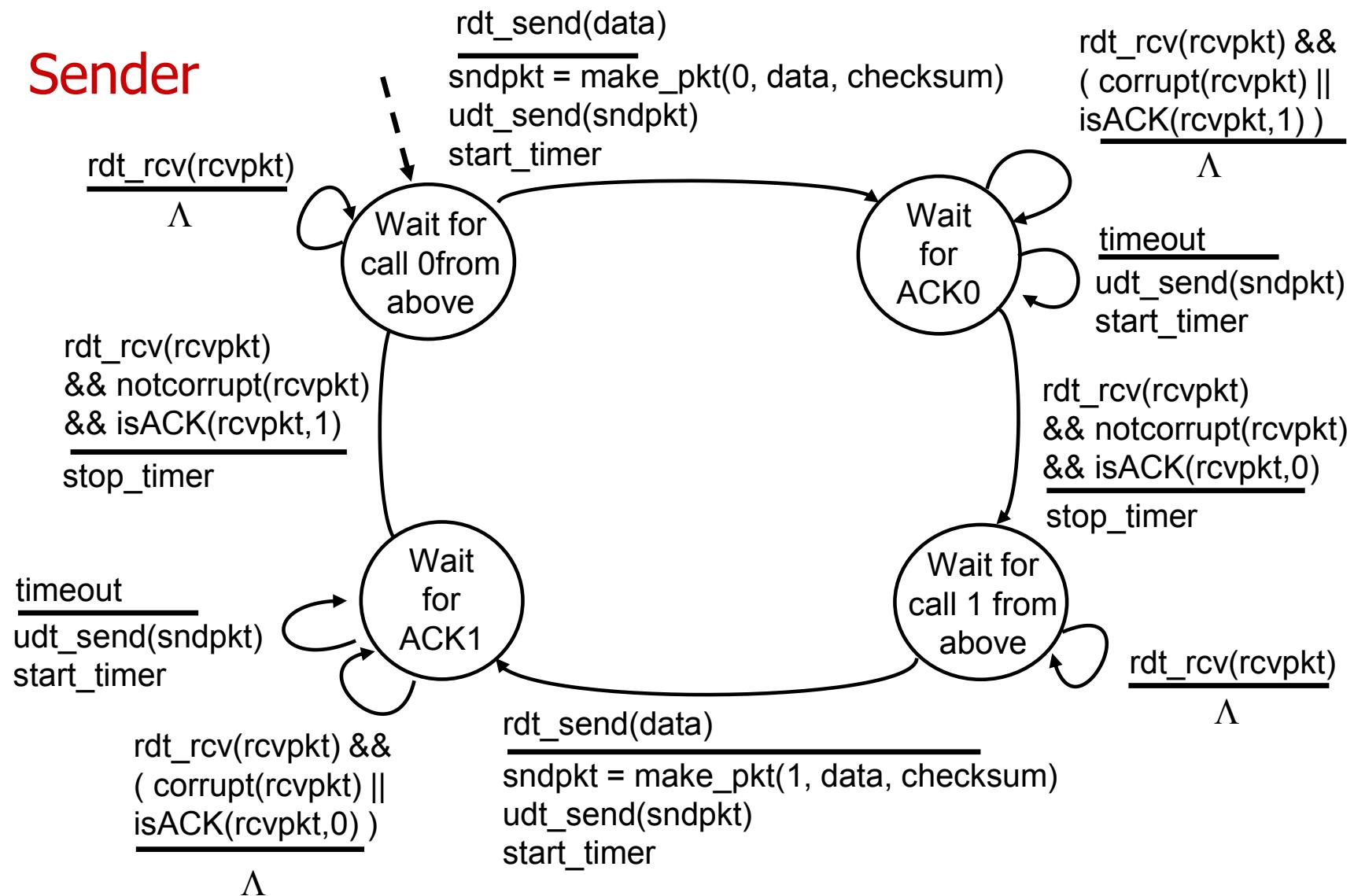
- Der Kanal kann nun auch ganze Pakete verlieren.
 - Fehlererkennung, Sequenznummern, ACKs und Übertragungs-wiederholungen helfen weiter, reichen aber nicht aus.
- Problem: Wie wird der Verlust von Paketen behandelt?

Ansatz: Sender wartet eine „vernünftige Zeitspanne“ auf ein ACK

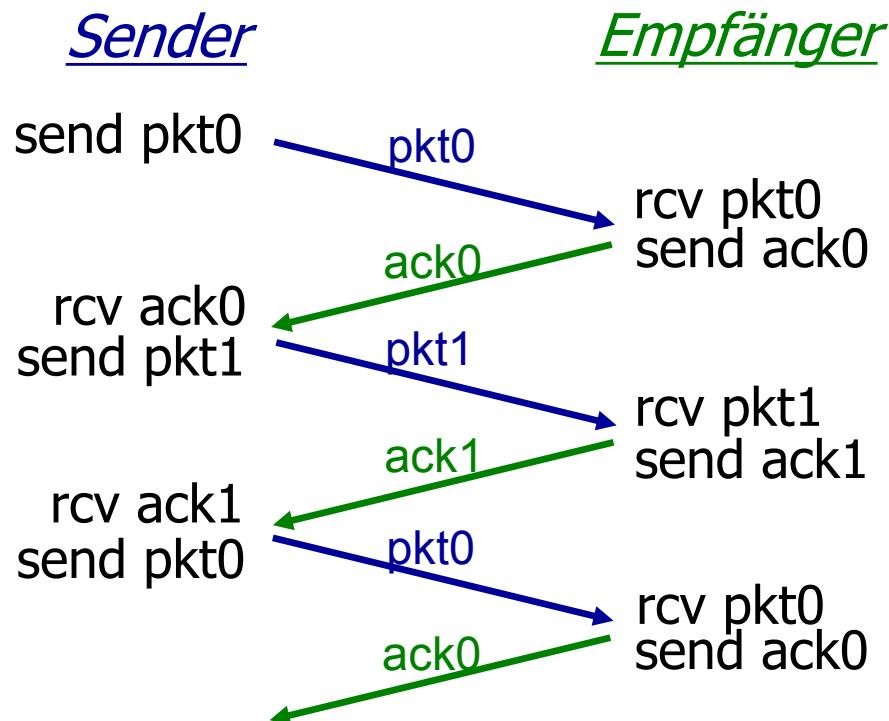
- Wenn dann kein ACK angekommen ist, wird die Übertragung wiederholt
- Wenn das Paket (oder das ACK) nur verzögert wurde und nicht verloren gegangen ist:
 - → Paket ist ein Duplikat
 - Dies wird durch seine Sequenznummer erkannt und vom Empfänger verworfen
- Erfordert Timer zum Stoppen der Zeit (countdown).

rdt3.0: Kanäle mit Bitfehlern und kompletten Paketverlusten

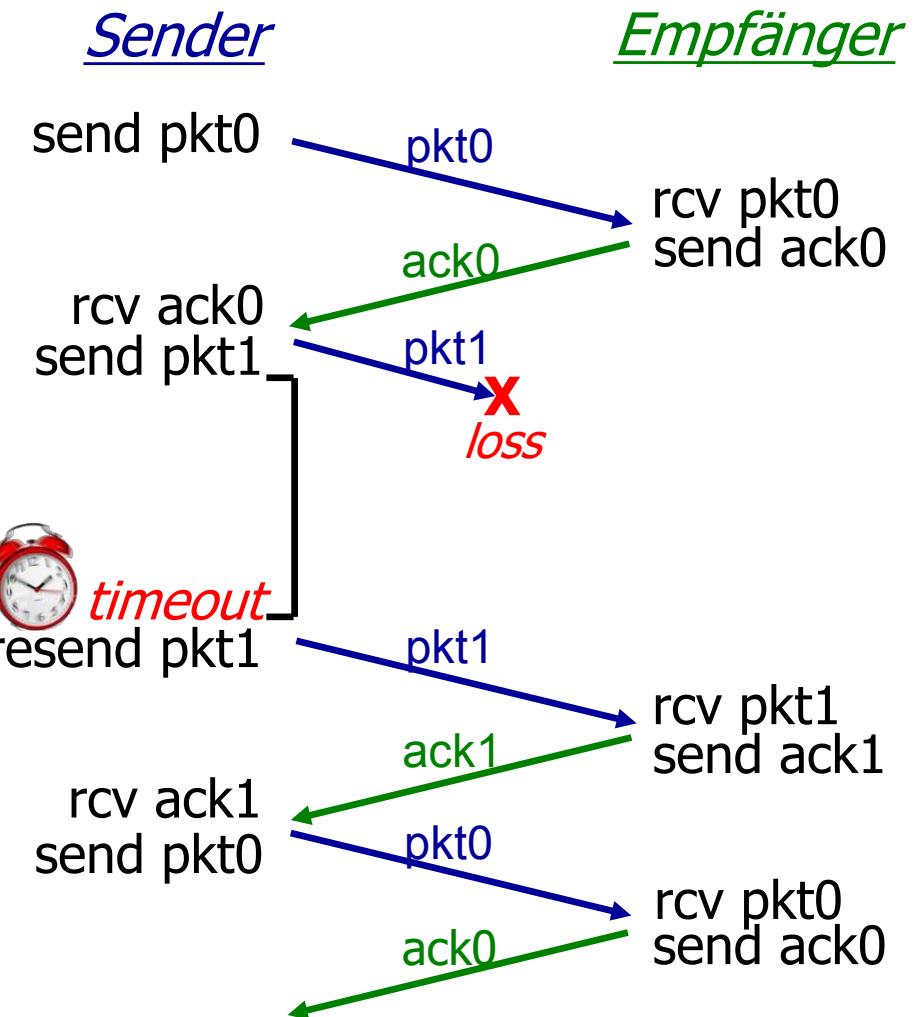
Sender



rdt3.0: Ablauf

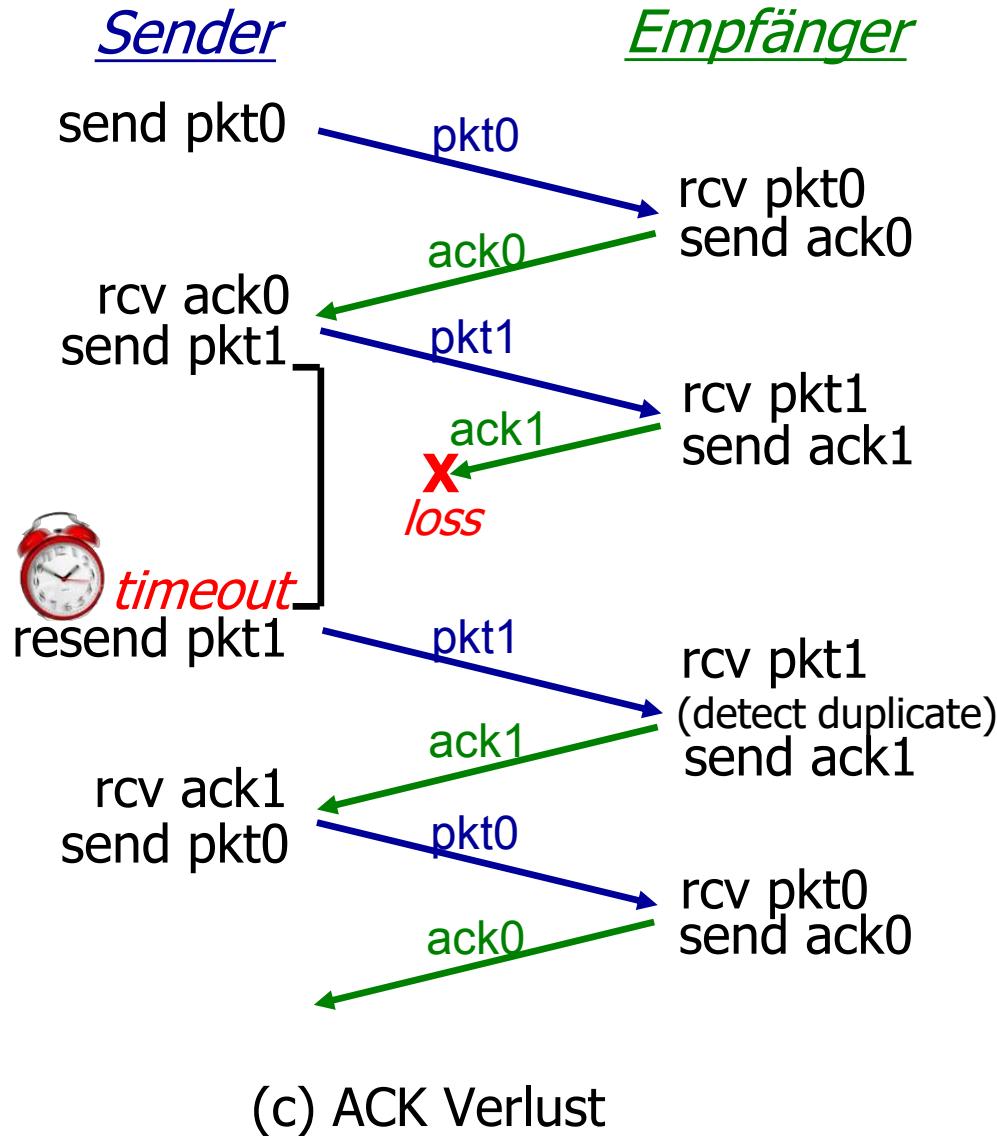


(a) Kein Paketverlust

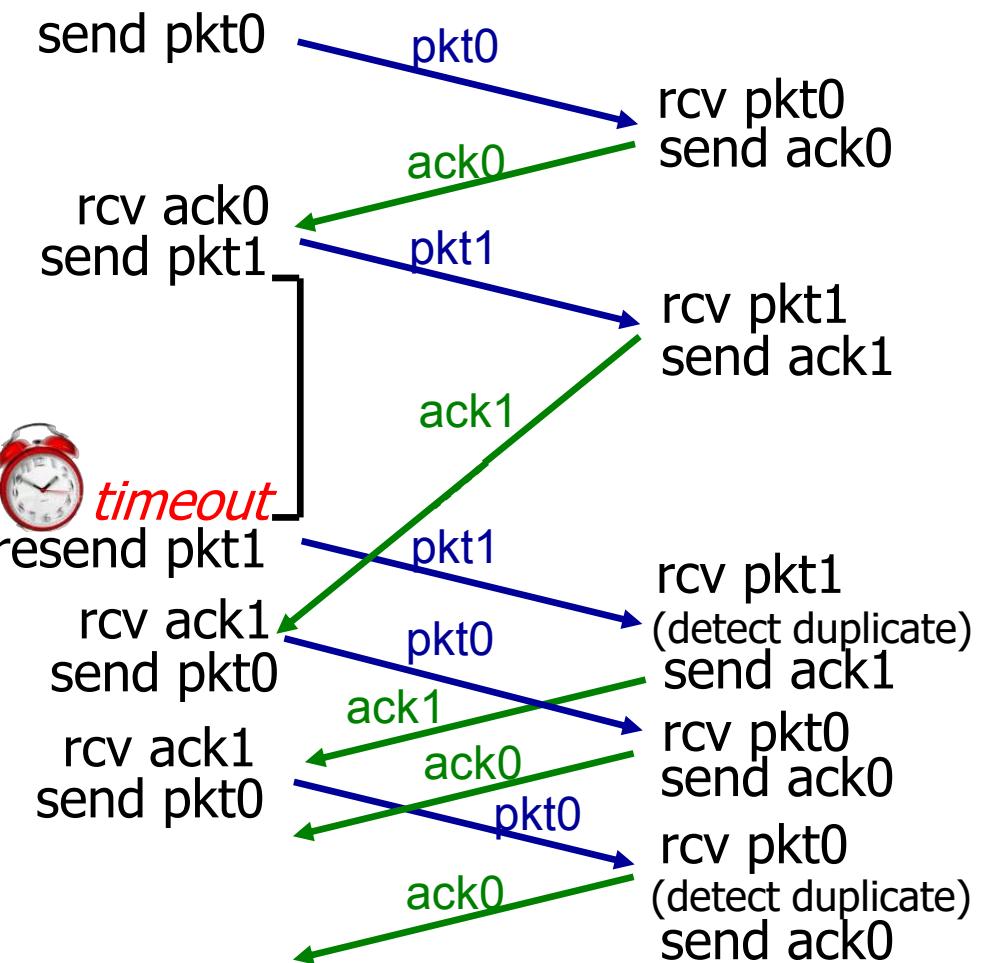


(b) Mit Paketverlust

rdt3.0: Ablauf



Sender *Empfänger*



(d) Vorzeitiger timeout/ verzögertes ACK

rdt3.0: Diskussion und Performance

- rdt3.0 funktioniert, aber die Performance ist schlecht!
- Beispiel: 1 GBit/s Link, 15 ms Ausbreitungsverzögerung, 8000 Bit Paketgröße:

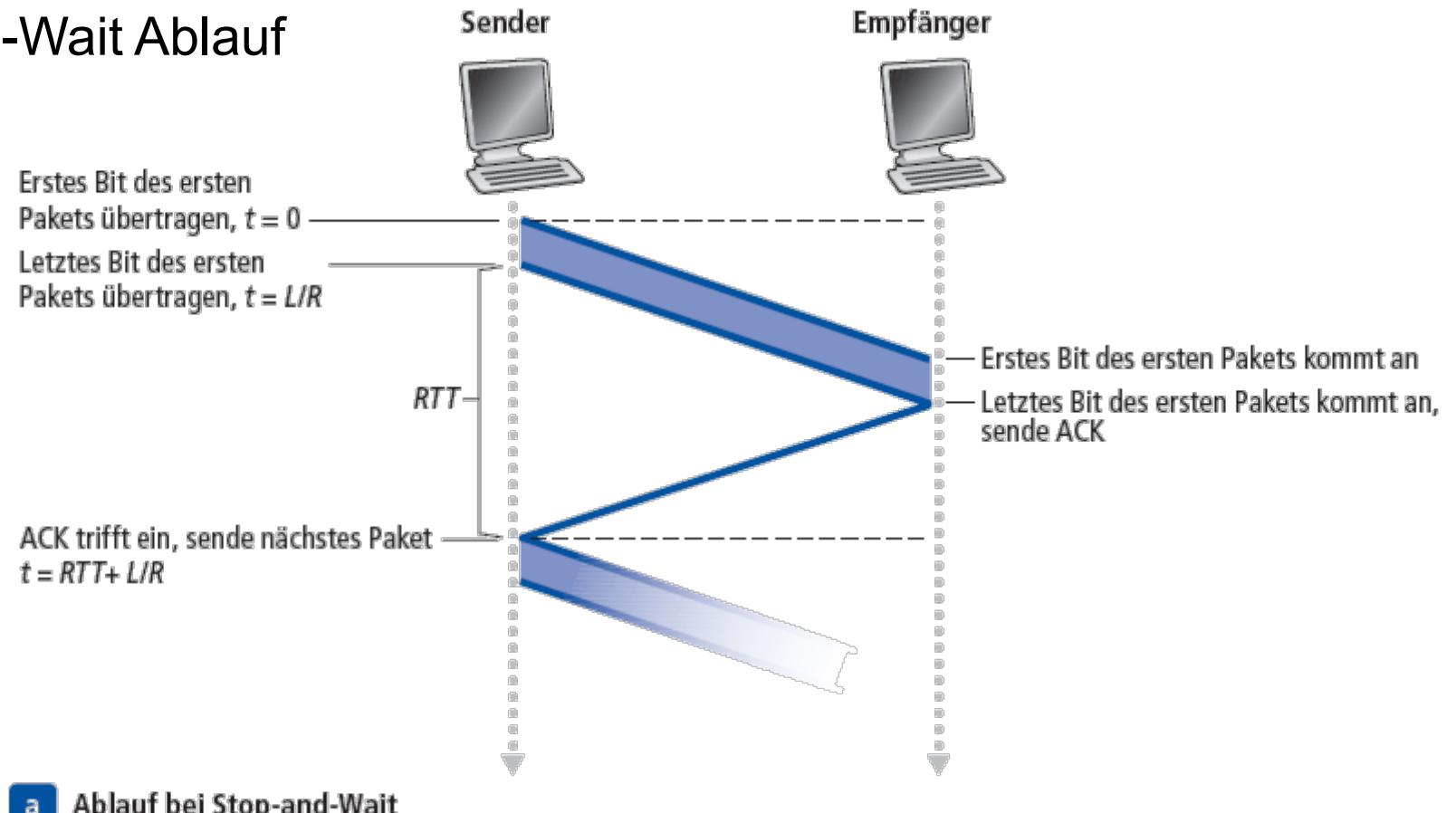
$$T_{\text{transmit}} = \frac{L}{R} = \frac{8000 \text{ bit}}{10^9 \text{ Bit/s}} = 8 \cdot 10^{-3} \text{ ms}$$

$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{0.008}{30.008} = 0.00027$$

- U_{sender} : Utilization (engl. für Auslastung) – Anteil der Zeit, in der der Sender tatsächlich sendet
- Einmal 8000 Bit alle ~30 ms -> 33KBit/s Durchsatz über einen Link mit 1 GBit/s
- Das Protokoll beschränkt die Ausnutzung physikalischer Ressourcen!

rdt3.0: Diskussion und Performance

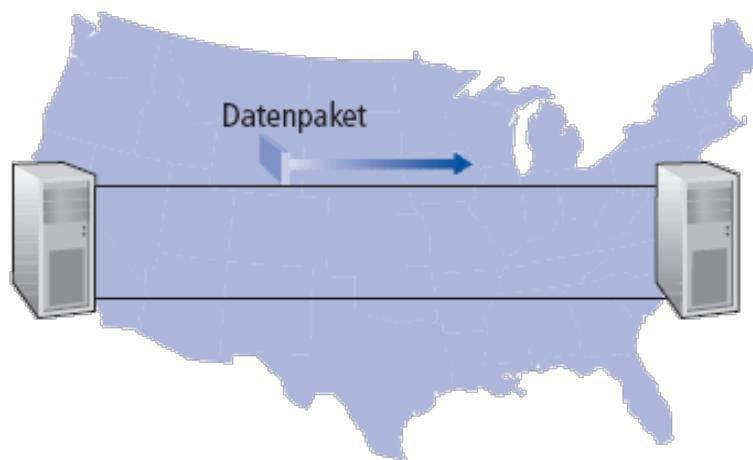
- Stop-and-Wait Ablauf



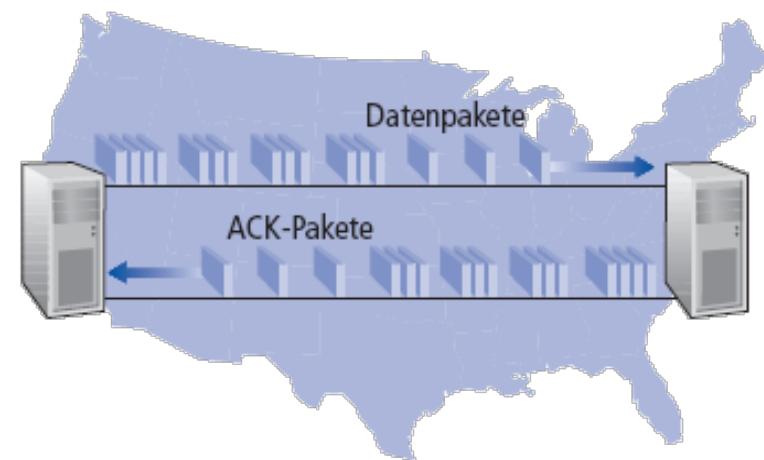
$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Protokolle mit Pipelining

- Pipelining: Sender lässt nicht nur eines, sondern mehrere unbestätigte Pakete zu
 - Die Anzahl der Sequenznummern muss erhöht werden.
 - Pakete müssen beim Sender und/oder Empfänger gepuffert werden



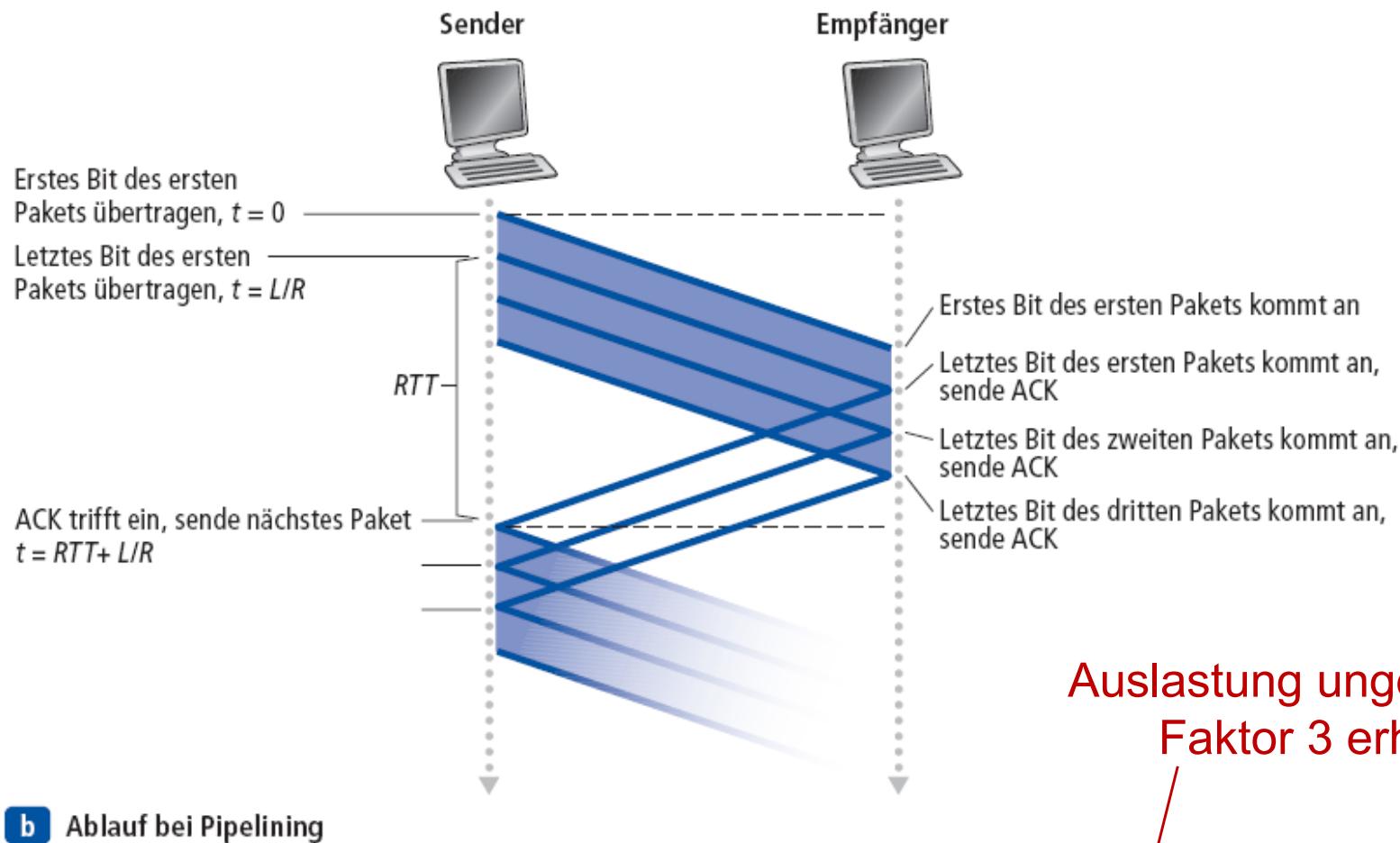
a Ablauf bei Stop-and-Wait



b Ablauf bei Pipelining

- Zwei prinzipielle Arten von Protokollen mit Pipelining:
Go-Back-N und Selective Repeat

Pipelining erhöht die Auslastung



$$U_{\text{sender}} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Protokolle mit Pipelining: Überblick

Go-back-N

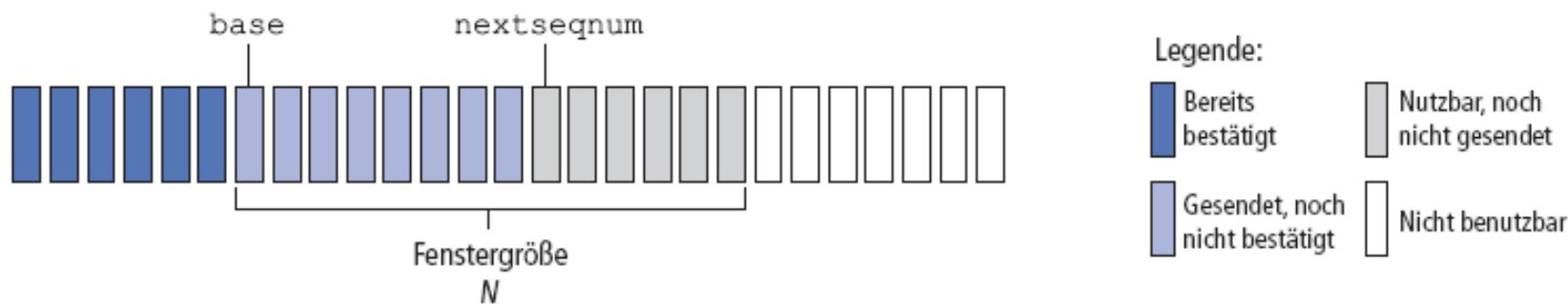
- Der Sender kann bis zu N nicht-bestätigte Pakete in der Pipeline haben.
- Der Empfänger sendet kumulative ACKs
 - Also wird nicht zwingend jedes Paket bestätigt.
- Der Sender hat einen Timer für das älteste, nicht-bestätigte Paket.
 - Wenn dieser Timer abgelaufen ist, werden alle nicht-bestätigten Pakete erneut übertragen.

Selective Repeat

- Der Sender kann bis zu N nicht-bestätigte Pakete in der Pipeline haben.
- Der Empfänger sendet für jedes Paket ein individuelles ACK.
- Der Sender hat einen Timer für jedes nicht bestätigte Paket.
 - Wenn dieser Timer abgelaufen ist, wird nur das nicht-bestätigte Paket erneut übertragen.

Protokolle mit Pipelining: Go-Back-N (GBN)

- Sender: k-Bit-Sequenznummer im Paketkopf
- Ein „Fenster“ (Window) von bis zu N aufeinanderfolgenden unbestätigten Paketen wird zugelassen



- ACK(n): Bestätigt alle Pakete bis zu (und einschließlich) dem Paket mit Sequenznummer n
 - Doppelte ACKs sind möglich
- Nur ein Timer!
- Timeout(n): alle Pakete mit der Sequenznummer n und höher neu übertragen

Protokolle mit Pipelining: GBN-Ablauf

Sender Fenster ($N=4$)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

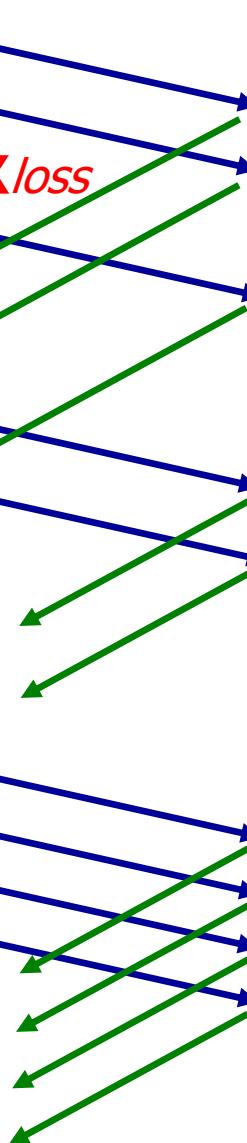
Sender

send pkt0
send pkt1
send pkt2
send pkt3
(wait)



pkt 2 timeout

send pkt2
send pkt3
send pkt4
send pkt5



Empfänger

receive pkt0, send ack0
receive pkt1, send ack1

receive pkt3, discard,
(re)send ack1

receive pkt4, discard,
(re)send ack1

receive pkt5, discard,
(re)send ack1

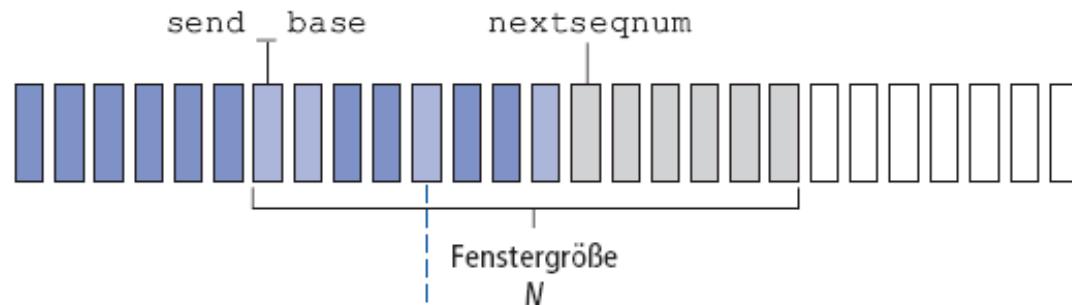
rcv pkt2, deliver, send ack2
rcv pkt3, deliver, send ack3
rcv pkt4, deliver, send ack4
rcv pkt5, deliver, send ack5

Kein Puffer beim Empfänger (merkt sich nur die erwartete Sequenznummer, *nextseqnum*). Out-of-order Pakete werden verworfen.

Protokolle mit Pipelining: Selective Repeat (SR)

- Der Empfänger bestätigt jedes empfangene Paket einzeln.
- Der Empfänger puffert korrekt empfangene Pakete, die außer der Reihe empfangen wurden, in einem Empfängerfenster.
 - Ausliefern an die nächste Schicht, wenn dies in der richtigen Reihenfolge möglich ist
- Der Sender wiederholt eine Übertragung nur für individuelle Pakete.
 - Jeweils ein Timer für jedes unbestätigte Paket
- Fenster des Senders
 - N aufeinanderfolgende Sequenznummern
 - Beschränkt wieder die unbestätigten, ausstehenden Pakete

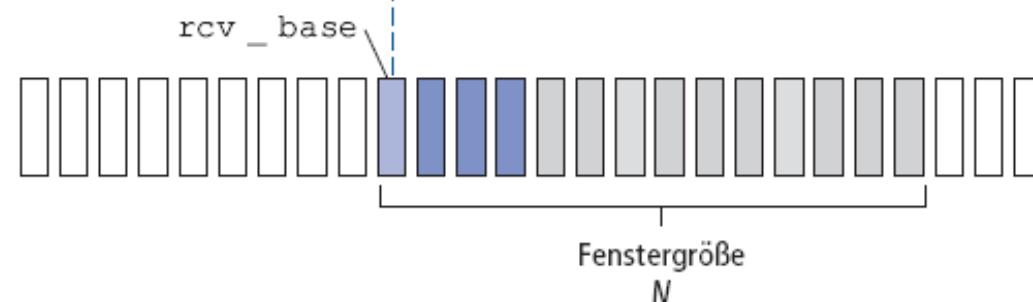
Protokolle mit Pipelining: SR – Sender und Empfängerfenster



a Sequenznummern aus Sicht des Senders

Legende:

- | | |
|--------------------------------|------------------------------|
| Bereits bestätigt | Nutzbar, noch nicht gesendet |
| Gesendet, noch nicht bestätigt | Nicht benutzbar |



b Sequenznummern aus Sicht des Empfängers

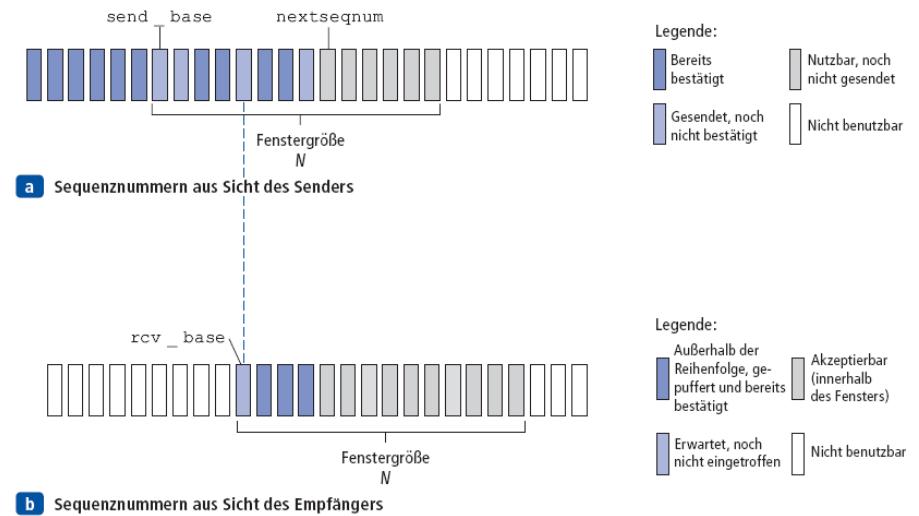
Legende:

- | | |
|--|---------------------------------------|
| Außerhalb der Reihenfolge, gepuffert und bereits bestätigt | Akzeptierbar (innerhalb des Fensters) |
| Erwartet, noch nicht eingetroffen | Nicht benutzbar |

Protokolle mit Pipelining: SR – Sender und Empfänger

Sender

- Daten von oben (Applikation):
 - Wenn nächste Sequenznummer im Fenster liegt: Paket senden, Timer(n) starten!
- Timeout(n):
 - Paket n erneut übertragen, Timer(n) starten
- ACK(n) aus $[sendbase, sendbase+N]$
 - Paket n als empfangen markieren
 - Wenn n die kleinste unbestätigte Sequenznummer ist, Fenster zur neuen kleinsten unbestätigten Sequenznummer verschieben



Empfänger

- Paket aus $[rcvbase, rcvbase+N-1]$
 - Sende ACK(n)
 - Außer der Reihe: Puffern
 - In der Reihe: Ausliefern (auch alle gepufferten Pakete ausliefern, die jetzt in der Reihe sind), Fenster zum nächsten erwarteten Paket verschieben
- Paket aus $[rcvbase-N, rcvbase-1]$
 - Sende ACK(n) (erneut!)
- Sonst: Ignoriere das Paket

Protokolle mit Pipelining: SR-Ablauf

Sender Fenster (N=4)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

(Fenster voll, warten!)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

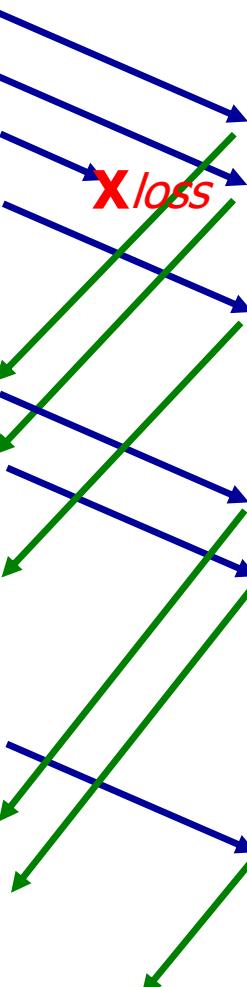
rcv ack0, send pkt4
rcv ack1, send pkt5

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

pkt 2 timeout
send pkt2
record ack4 arrived
record ack5 arrived

Sender

send pkt0
send pkt1
send pkt2
send pkt3



Empfänger

receive pkt0, send ack0
receive pkt1, send ack1
receive pkt3, buffer, send ack3
receive pkt4, buffer, send ack4
receive pkt5, buffer, send ack5
rcv pkt2; deliver pkt2, pkt3, pkt4, pkt5; send ack2

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

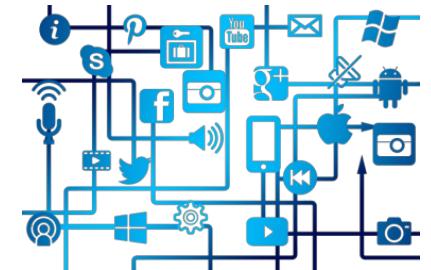
0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

Was passiert, wenn ack2 ankommt?

Begleitmaterialien



- **Selective Repeat / Go-Back-N Simulation**

http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

- Alternative Simulationen

- Go-Back-N

https://media.pearsoncmg.com/aw/ecs_kurose_comnetwork_7/cw/content/interactiveanimations/go-back-n-protocol/index.html

Video: <https://www.youtube.com/watch?v=9BuaeEjleQI>

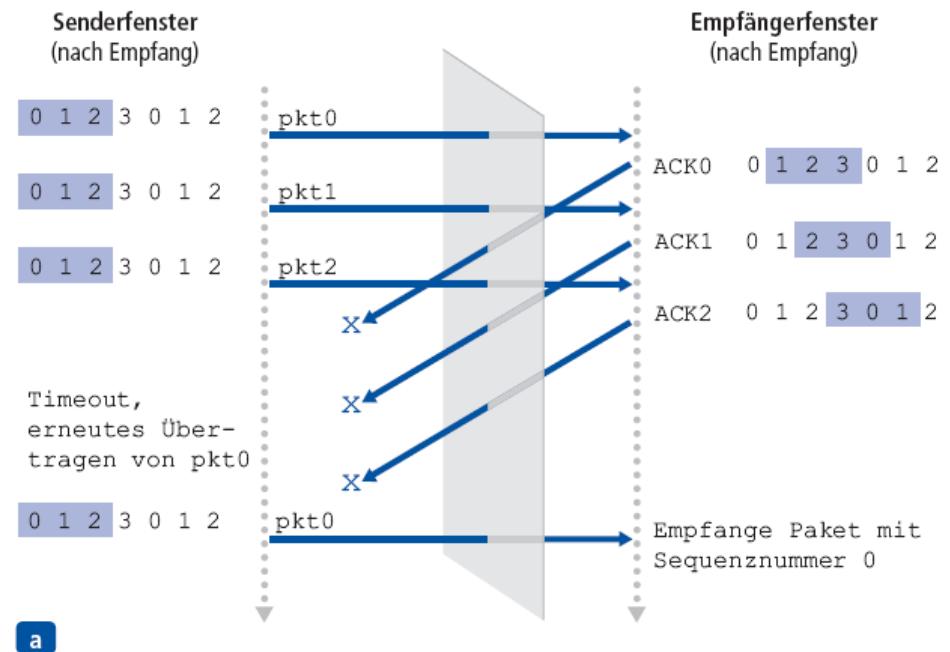
- Selective Repeat

https://media.pearsoncmg.com/aw/ecs_kurose_comnetwork_7/cw/content/interactiveanimations/selective-repeat-protocol/index.html

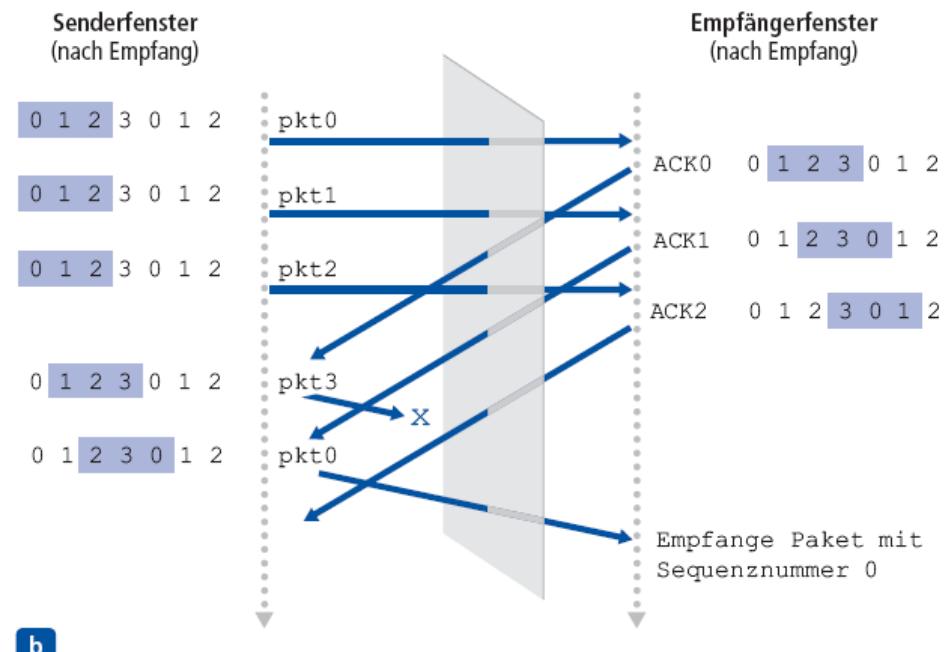
Video: <https://www.youtube.com/watch?v=Cs8tR8A9jm8>

Problem bei Selective Repeat

- Beispiel:
 - Sequenznummern: 0-3
 - Fenstergröße=3
- Empfänger kann beide Fälle nicht unterscheiden!
- Gibt verdoppeltes altes Paket als neue Daten nach oben!
- Relation zwischen Fenstergröße und der Anzahl an Sequenznummern:
 - Fenstergröße $\leq \frac{1}{2} \#$ Sequenznummern
 - Ohne Beweis, vgl. Kurose & Ross, Kapitel 3.4



a

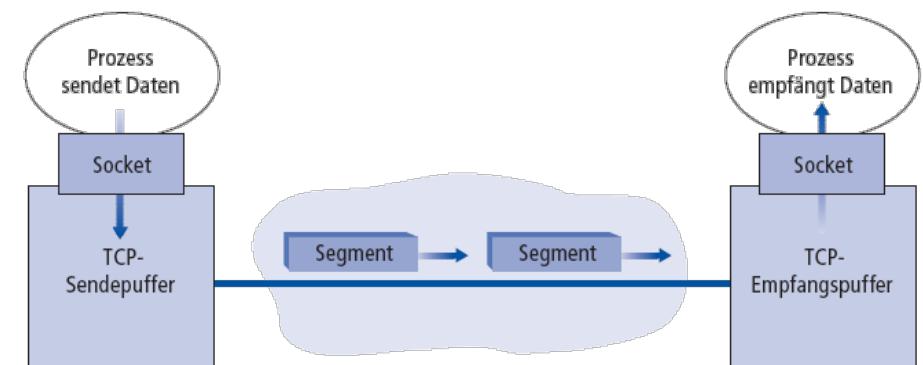


b

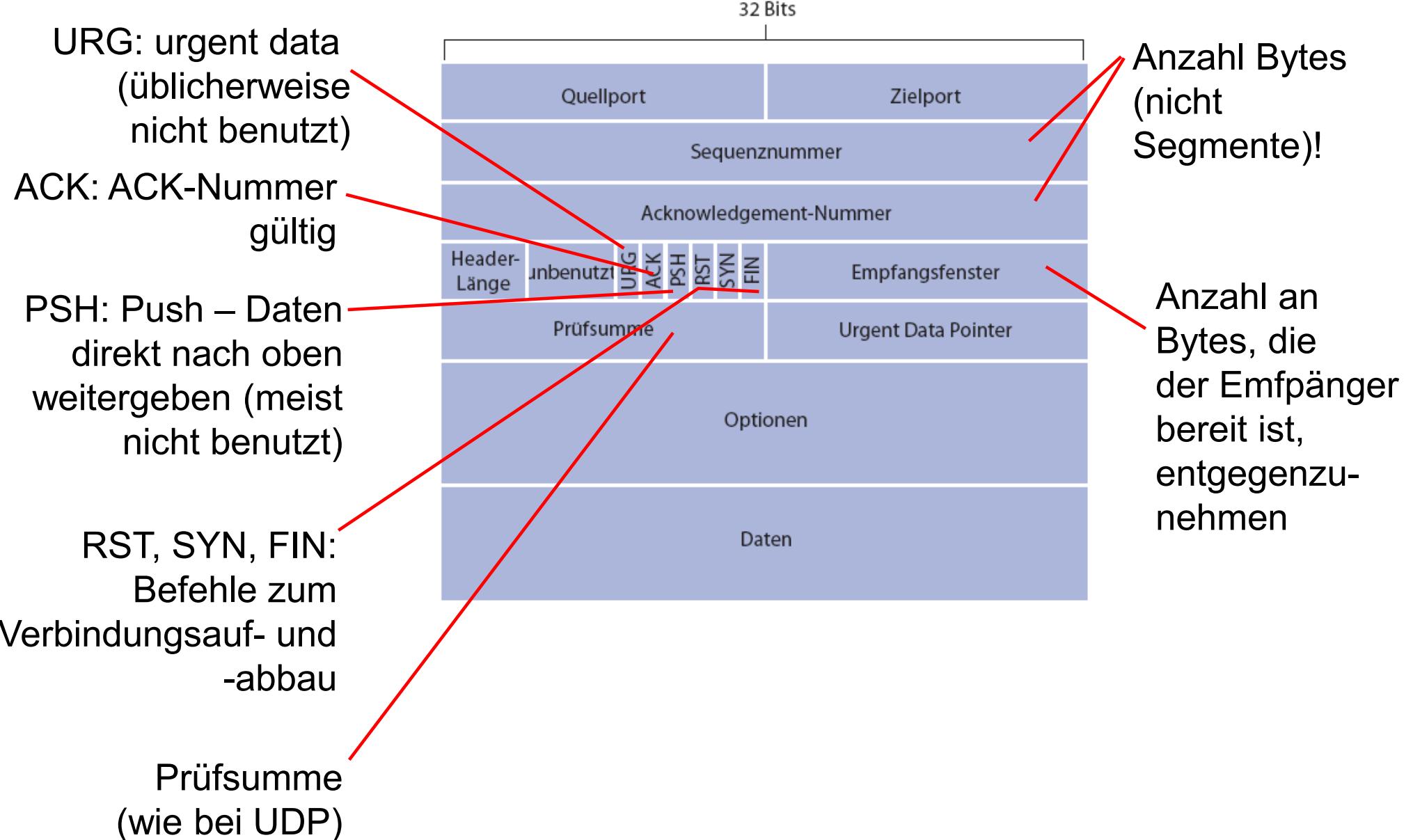
- Dienste der Transportschicht
- Multiplexing und Demultiplexing
- Verbindungslose Übertragung: UDP
- Zuverlässige Datenübertragung
- Verbindungsorientierte Übertragung: TCP
- Überlastkontrolle: Grundlagen und Umsetzung in TCP
- Zusammenfassung und Ausblick

Transmission Control Protocol (TCP) – Überblick

- Punkt-zu-Punkt
 - Ein Sender, ein Empfänger
- Zuverlässiger, reihenfolgeerhaltender Byte-Strom
 - Keine “Nachrichtengrenzen”
- Pipelining
 - TCP-Überlast- und -Flusskontrolle verändern die Größe des Fensters
- Sender- & Empfängerfenster
 - Vgl. zuverlässige Datenübertragung
- Vollduplex
 - Daten fließen in beide Richtungen
 - MSS: Maximum Segment Size
- Verbindungsorientiert
 - „Handshaking“ (Austausch von Kontrollnachrichten) initialisiert den Zustand im Sender und Empfänger, bevor Daten ausgetauscht werden
- Flusskontrolle
 - Sender überfordert Empfänger nicht

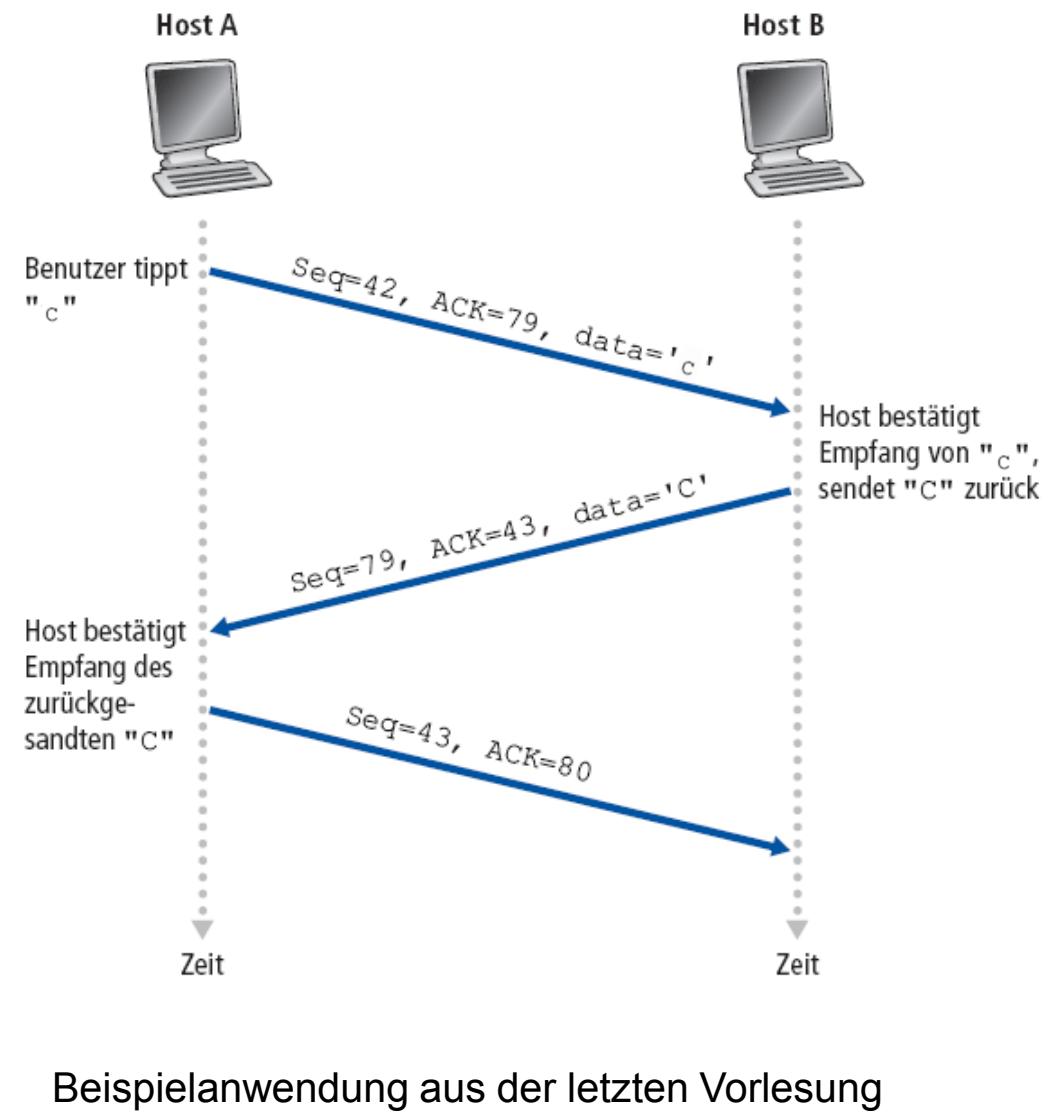


TCP: Segmentaufbau



TCP: Sequenznummern und ACKs

- Sequenznummern:
 - Nummer des ersten Byte im Datenteil
- ACKs
 - Sequenznummer des nächsten Byte, das von der Gegenseite erwartet wird
 - Kumulative ACKs
- Wie werden Segmente behandelt, die außer der Reihe ankommen?
 - Wird von der TCP-Spezifikation nicht vorgeschrieben!
 - Bestimmt durch die Implementierung



TCP: Round Trip Time und -Timeout

Frage: Wie bestimmt TCP den Wert für den Timeout?

- Größer als die Rundlaufzeit (Round Trip Time, RTT)
 - Aber RTT ist nicht konstant
- Zu kurz: unnötige Timeouts
 - Unnötige Übertragungswiederholungen
- Zu lang: langsame Reaktion auf den Verlust von Segmenten

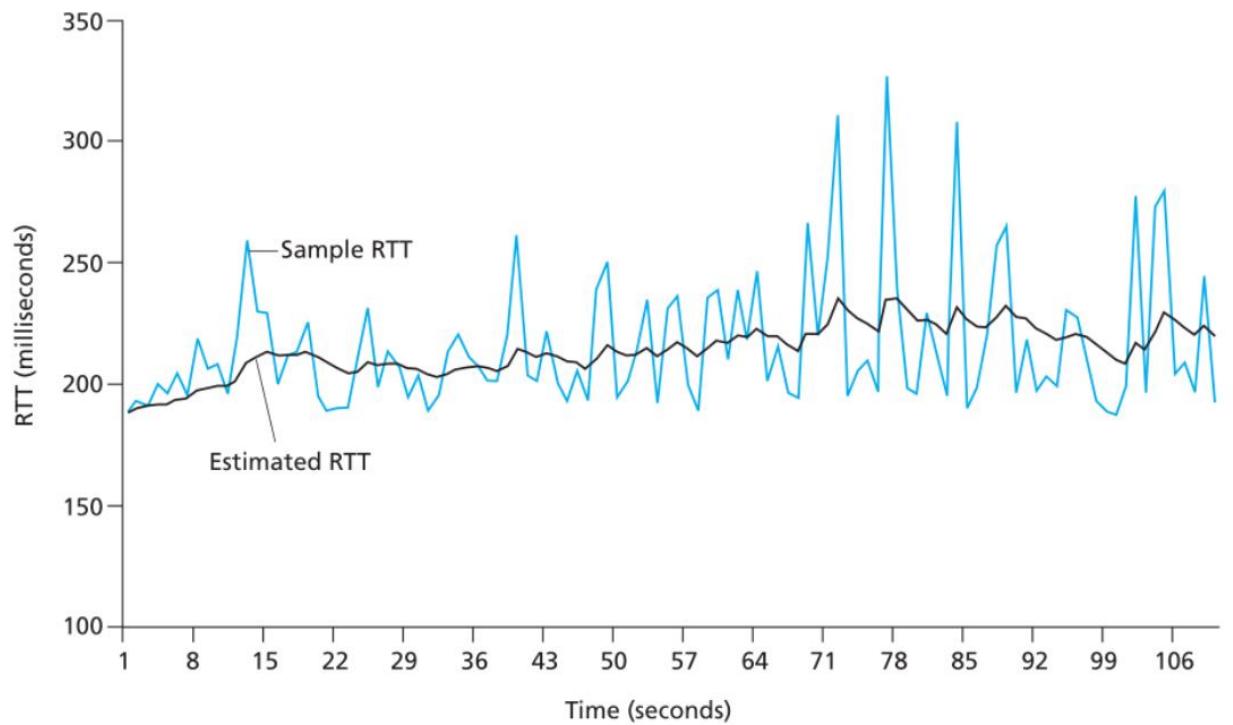
Frage: Wie kann man die RTT schätzen?

- **SampleRTT**: gemessene Zeit vom Absenden eines Segments bis zum Empfang des dazugehörigen ACKs
- Segmente mit Übertragungswiederholungen werden ignoriert
- SampleRTT ist nicht konstant, wir brauchen einen „glatteren“ Wert
- Durchschnitt über mehrere Messungen

TCP: Round Trip Time und -Timeout

■ Exponentielle Glättung („Exponential weighted moving average“)

- Einfluss vergangener Messungen verringert sich exponentiell schnell
- Üblicher Wert:
 $\alpha = 0,125$



- $\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$

TCP: Round Trip Time und -Timeout

- Bestimmen des Timeout
 - EstimatedRTT plus “Sicherheitsabstand”
 - Größere Schwankungen von EstimatedRTT → größerer Sicherheitsabstand
- Bestimme, wie sehr SampleRTT von EstimatedRTT abweicht:
 - $\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$
(üblicherweise: $\beta = 0.25$)
- Timeout
 - $\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$



TCP: Zuverlässiger Datentransfer

- TCP stellt einen zuverlässigen Datentransfer über den unzuverlässigen Datentransfer von IP zur Verfügung.
 - Pipelining von Segmenten
 - Kumulative ACKs
 - nur ein Timer für Übertragungswiederholungen
- Übertragungswiederholungen werden ausgelöst durch:
 - Timeout
 - Doppelte ACKs
- Zu Beginn betrachten wir einen vereinfachten TCP-Sender
 - Ignorieren von doppelten ACKs
 - Ignorieren von Fluss- und Überlastkontrolle

TCP: Zuverlässiger Datentransfer – Ereignisse beim Sender

Daten von Anwendung erhalten:

- Erzeuge Segment mit geeigneter Sequenznummer
 - Nummer des ersten Byte im Datenteil
- Timer starten, wenn er noch nicht läuft
 - Timer für das älteste unbestätigte Segment
- Laufzeit des Timers: TimeOutInterval
 - Bestimmt nach oben dargestelltem Verfahren

Timeout:

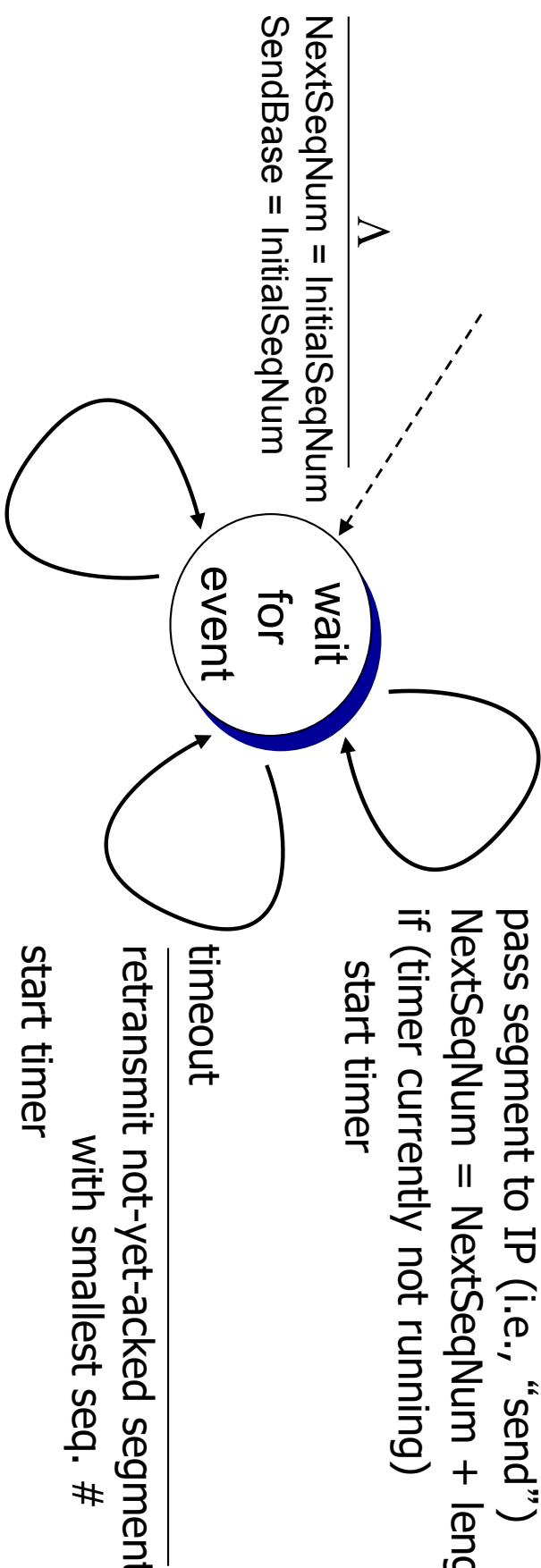
- Erneute Übertragung des Segments, für das der Timeout aufgetreten ist
- Starte Timer neu

ACK empfangen:

- Wenn damit bisher unbestätigte Daten bestätigt werden:
 - Aktualisiere die Informationen über bestätigte Segmente
 - Starte Timer neu, wenn noch unbestätigte Segmente vorhanden sind

TCP: Zuverlässiger Datentransfer – Ereignisse beim Sender

data received from application above
 create segment, seq. #: NextSeqNum
 pass segment to IP (i.e., “send”)
 Λ
 NextSeqNum = NextSeqNum + length(data)
 if (timer currently not running)



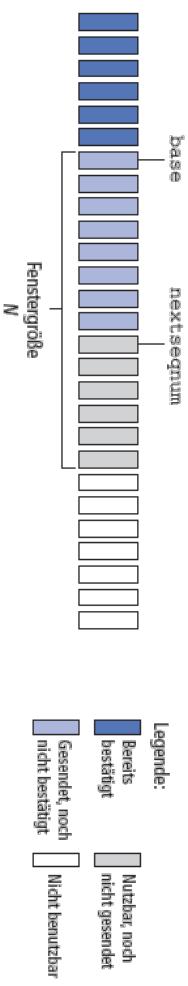
ACK received, with ACK field value y

```

if (y > SendBase) {
    SendBase = y
}
    
```

if (there are currently not-yet-acked segments)
 start timer

else stop timer



Hinweis: *SendBase-1 ist das letzte erfolgreich bestätigte Byte*

Erinnerung an GBN:

TCP: Beispiele für Übertragungswiederholungen

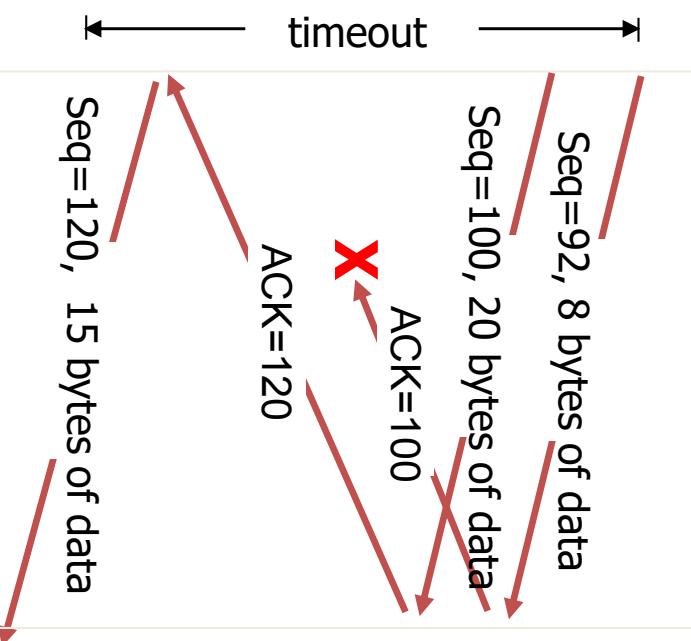


Paketverlust (ACK)

Wichtig: Segment 100 wird nicht erneut gesendet!

Verfrühter Timeout

TCP: Beispiele für Übertragungswiederholungen



Seq=120, 15 bytes of data

timeout

Seq=100, 20 bytes of data
ACK=100

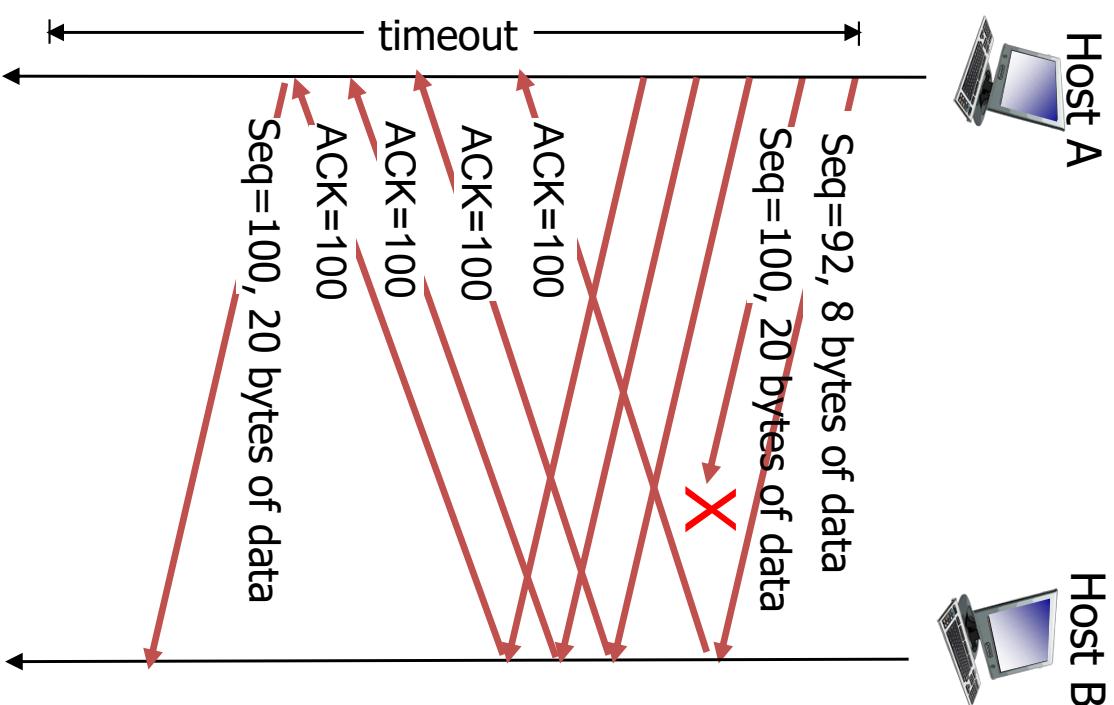
Kumulatives ACK

Kumulatives ACK (120) vermeidet, dass Segment 92 erneut übertragen wird!

TCP: Fast Retransmit

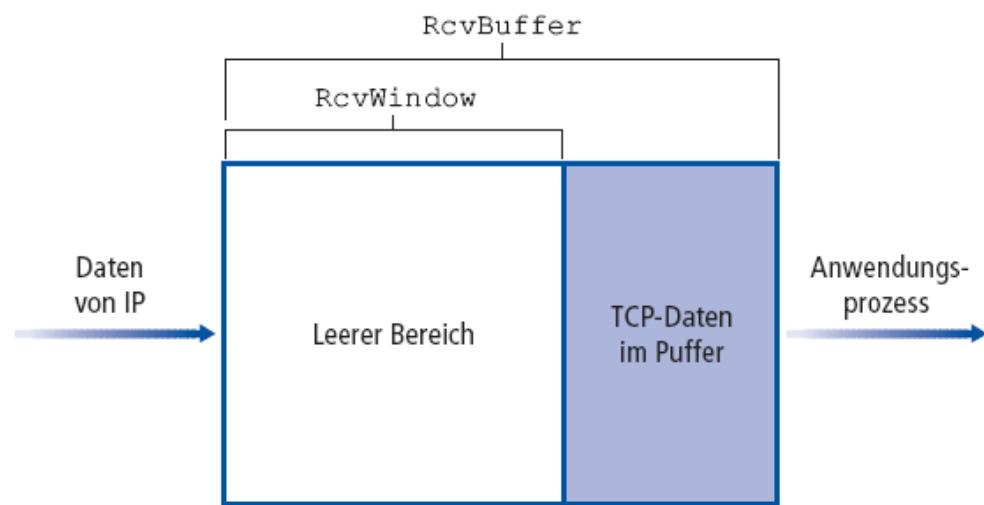
- Zeit für Timeout ist unter Umständen sehr lang (s.o.)
 - Große Verzögerung vor einer Neuübertragung
- Deshalb nutzt TCP eine Möglichkeit zum schnellen Erkennen von Paketverlusten durch doppelte ACKs
 - Sender schickt häufig viele Segmente direkt hintereinander
 - Wenn ein Segment verloren geht, führt dies zu vielen doppelten ACKs
 - Wenn der Sender 3 Duplikate eines ACK erhält, dann nimmt er an, dass das Segment verloren gegangen ist

- Fast Retransmit (schnelle Sendewiederholung): Segment erneut schicken, bevor der Timer ausläuft



TCP: Flusskontrolle

- Die Empfängerseite von TCP hat einen Empfängerpuffer



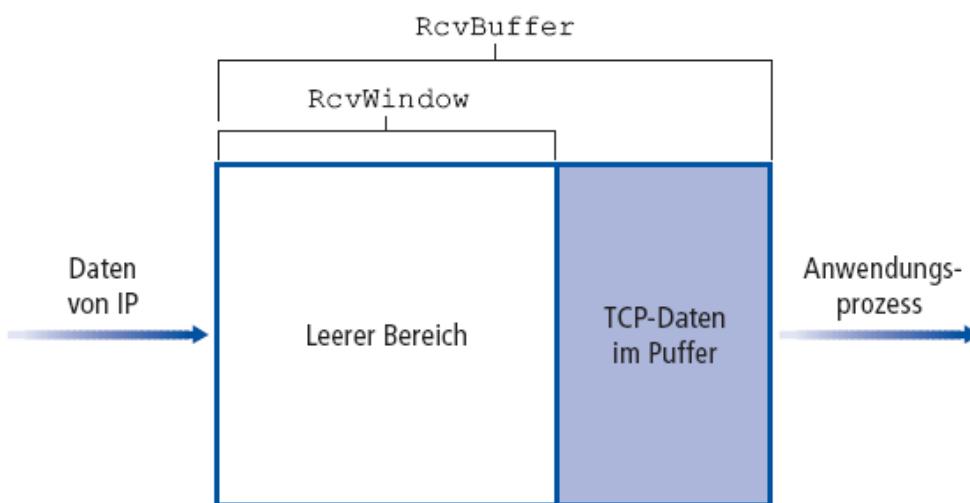
- Flusskontrolle: Dienst zum Angleichen von Geschwindigkeiten

- Die Senderate wird an die Verarbeitungsrate der Anwendung auf Empfängerseite angepasst.
- Der Sender schickt also nicht mehr Daten, als der Empfänger in seinem Puffer speichern kann.

- Die Anwendung kommt unter Umständen nicht mit dem Lesen hinterher

TCP: Flusskontrolle – Funktionsweise

- Annahme: Empfänger verwirft Segmente, die außer der Reihe ankommen
- Platz im Puffer = **RcvWindow**
- Empfänger meldet den aktuellen Platz durch **RcvWindow** im TCP-Header dem Sender an
 - Standardwert RcvBuffer: 4096 Byte
 - Oft vom Betriebssystem angepasst

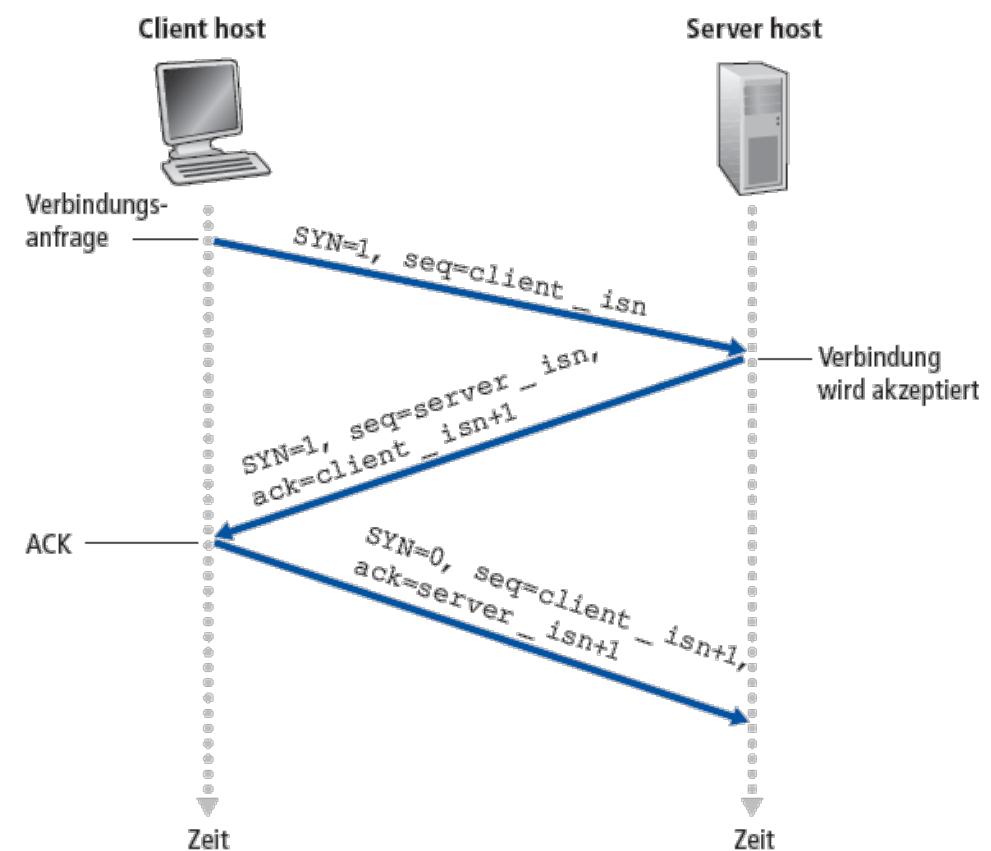


- Sender begrenzt seine unbestätigt (ohne ACK) gesendeten Daten auf **RcvWindow**
 - Dies garantiert, dass der Puffer des Empfängers nicht überläuft

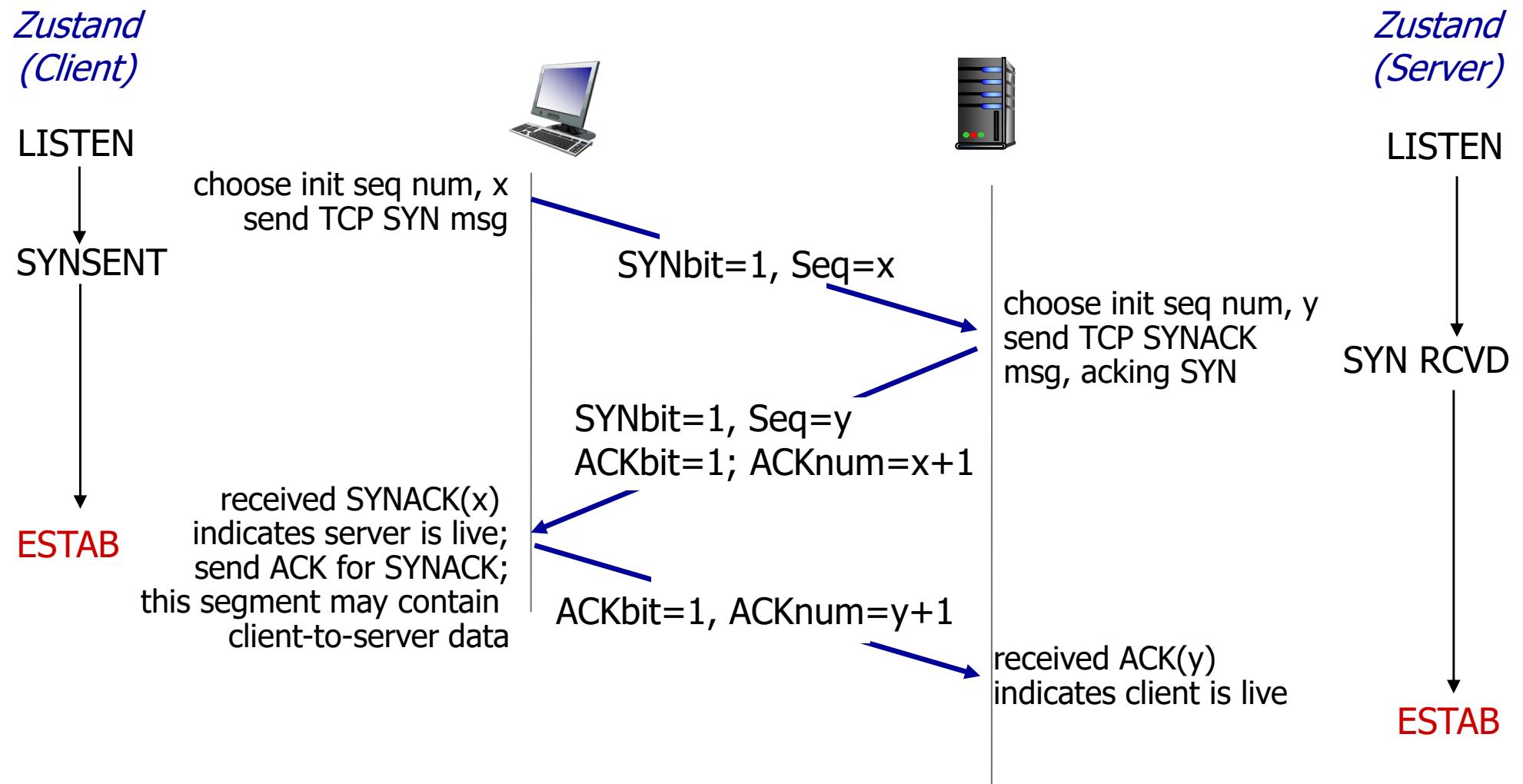
<http://www.ccs-labs.org/teaching/rn/animations/flow/>

TCP: Verbindungsmanagement – Verbindungsauftbau

- Erinnerung: TCP-Sender und -Empfänger bauen Verbindung auf, bevor sie Daten austauschen
 - Initialisieren der TCP-Variablen: Sequenznummern, Informationen für Flusskontrolle (z.B. RcvWindow)
- Drei-Wege-Handshake:
 - Schritt 1: Client sendet TCP-SYN-Segment an den Server
 - Initiale Sequenznummer (Client->Server)
 - keine Daten
 - Schritt 2: Server empfängt SYN und antwortet mit SYNACK
 - Server legt Puffer an
 - Initiale Sequenznummer (Server->Client)
 - Schritt 3: Client empfängt SYNACK und antwortet mit einem ACK – dieses Segment darf bereits Daten beinhalten



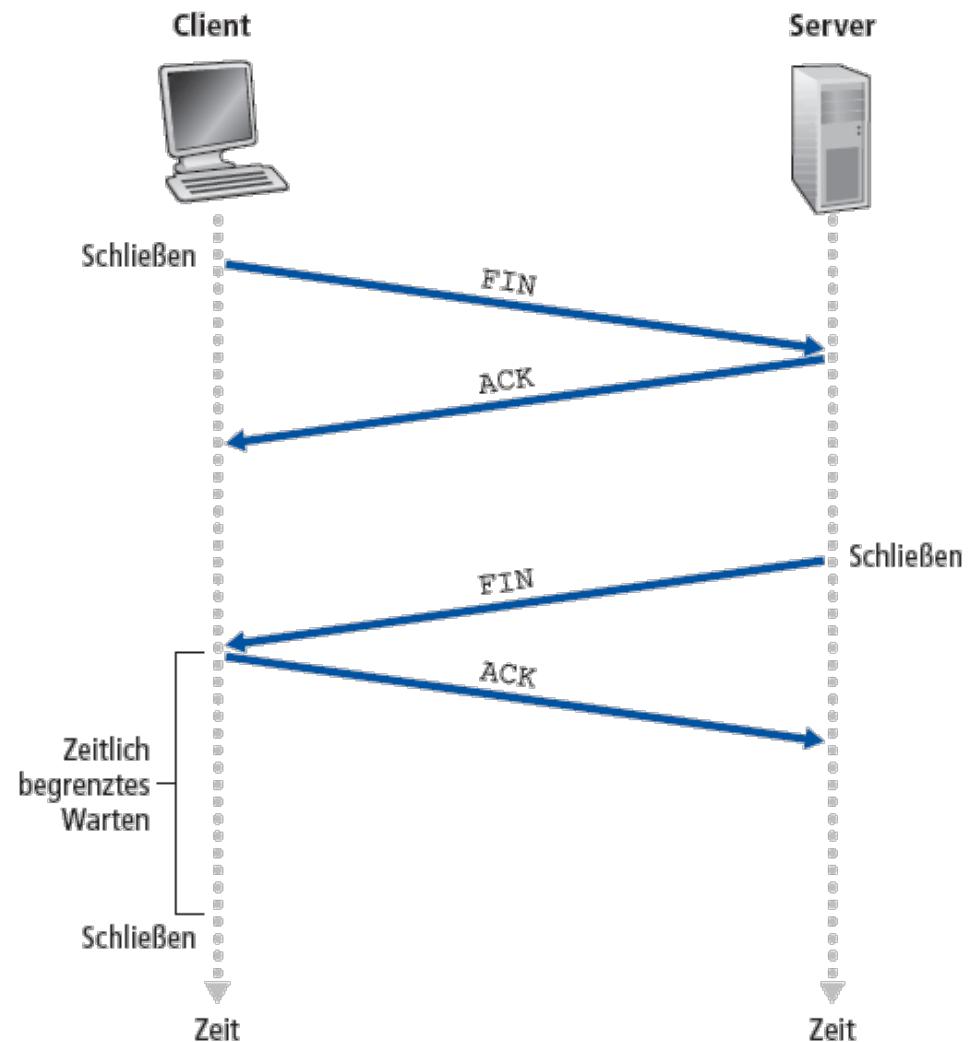
TCP: Verbindungsauftbau mit 3-Wege-Handshake



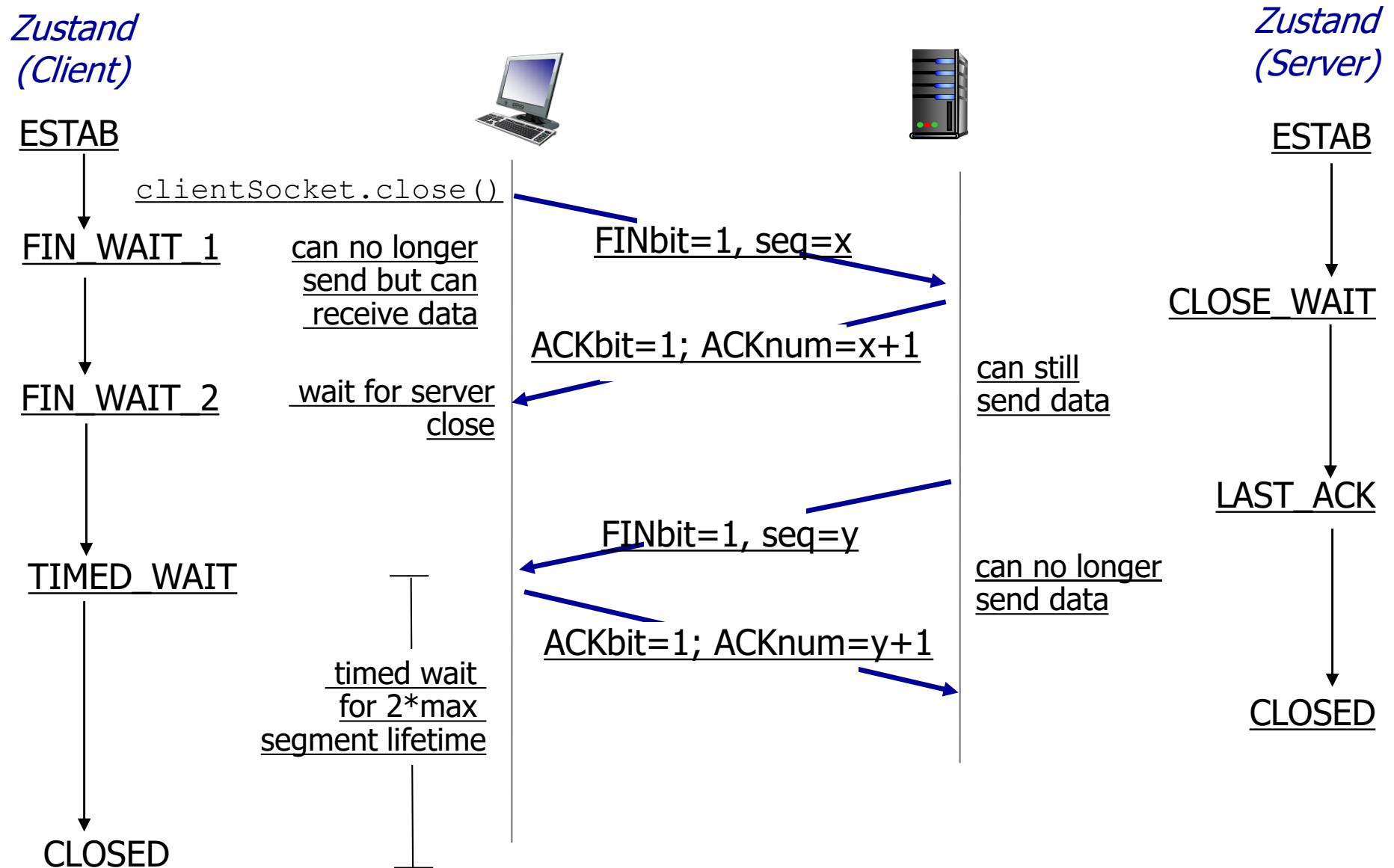
TCP: Verbindungsmanagement – Verbindungsabbau

Client schließt socket

- Schritt 1: Client sendet ein TCP-FIN-Segment an den Server
- Schritt 2: Server empfängt FIN, antwortet mit ACK; dann sendet er ein FIN (kann im gleichen Segment erfolgen)
- Schritt 3: Client empfängt FIN und antwortet mit ACK
- Schritt 4: Server, empfängt ACK und schließt Verbindung



TCP: Verbindungsmanagement – Verbindungsabbau



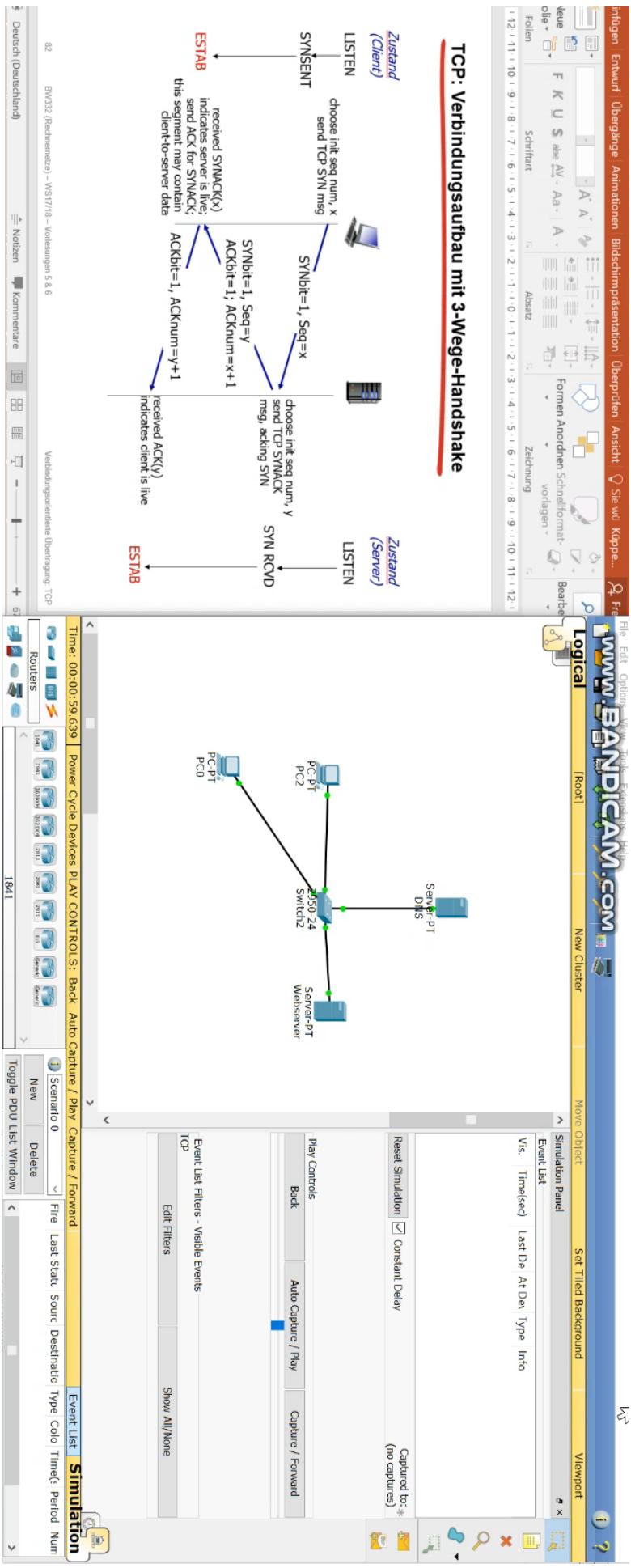


Übung

- Die Protokolle HTTP und SMTP basieren auf TCP (s.o.).
 - Wir haben dieses also bereits verwendet.
 - Wir können unser Cisco Packet Tracer-Projekt aus Übung 5.2 somit aus einer weiteren Perspektive betrachten: die Transportschicht mit HTTP & SMTP unter Verwendung von TCP!
- Rufen Sie Ihr Cisco Packet Tracer-Projekt aus Übung 5.2 auf.
 - Wechseln Sie in den Simulationsmodus und rufen Sie vom Client aus die Seite www.hs-lu.de auf.
 - Betrachten Sie die TCP-Pakete, auch im Hinblick auf die verwendeten Ports.

Übung

Beispiel:



TCP Verbindungsauflösung und -abbau

tcp.port == 12000						
No.	Time	Source	Destination	Protocol	Length	Info
12387	71.839283868	109.84.1.105	46.101.190.70	TCP	74	4014 - 12000 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 S...
12388	71.839352914	46.101.190.70	109.84.1.105	TCP	74	12000 - 4014 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0...
12389	71.841040712	109.84.1.105	46.101.190.70	TCP	66	4014 - 12000 [ACK] Seq=1 Ack=1 Win=14848 Len=0 TSva...
12390	71.841110206	109.84.1.105	46.101.190.70	TCP	95	4014 - 12000 [PSH, ACK] Seq=1 Ack=1 Win=14848 Len=2...
12391	71.841127437	46.101.190.70	109.84.1.105	TCP	66	12000 - 4014 [ACK] Seq=1 Ack=30 Win=28992 Len=0 TSv...
12392	71.842429589	46.101.190.70	109.84.1.105	TCP	95	12000 - 4014 [PSH, ACK] Seq=1 Ack=30 Win=28992 Len=...
12393	71.842526361	46.101.190.70	109.84.1.105	TCP	66	12000 - 4014 [FIN, ACK] Seq=30 Ack=30 Win=28992 Len=...
12394	71.843598723	109.84.1.105	46.101.190.70	TCP	66	4014 - 12000 [ACK] Seq=30 Ack=30 Win=14848 Len=0 TS...
12407	71.883789247	109.84.1.105	46.101.190.70	TCP	66	4014 - 12000 [ACK] Seq=30 Ack=31 Win=14848 Len=0 TS...
12408	71.900117929	109.84.1.105	46.101.190.70	TCP	66	4014 - 12000 [FIN, ACK] Seq=30 Ack=31 Win=14848 Len=0 TS...
12409	71.900172471	46.101.190.70	109.84.1.105	TCP	66	12000 - 4014 [ACK] Seq=31 Ack=31 Win=28992 Len=0 TS...

Frame 12390: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface 0

► Ethernet II, Src: JuniperN34:67:f0 (40:a6:77:34:67:f0), Dst: 86:78:6a:53:2c:57 (86:78:6a:53:2c:57)

► Internet Protocol Version 4, Src: 109.84.1.105, Dst: 46.101.190.70

▼ Transmission Control Protocol, Src Port: 4014, Dst Port: 12000, Seq: 1, Ack: 1, Len: 29

Source Port: 4014

Destination Port: 12000

[Stream index: 4]

[TCP Segment Len: 29]

Sequence number: 1 (relative sequence number)

[Next sequence number: 30 (relative sequence number)]

Acknowledgment number: 1 (relative ack number)

Header Length: 32 bytes

Flags: 0x018 (PSH, ACK)

Window size value: 29

[Calculated window size: 14848]

[Window size scaling factor: 512]

Checksum: 0x46c4 [unverified]

[Checksum Status: Unverified]

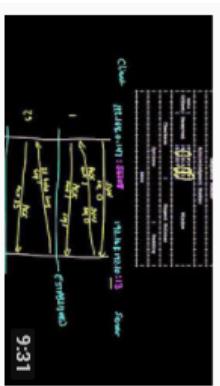
0000 86 78 6a 53 2c 57 40 a6 77 34 67 f0 08 00 45 00 .xj\$,W@. w4g...E.
0010 00 51 b5 1a 40 00 35 06 35 24 6d 54 01 69 2e 65 .Q..@.5. 5\$mT.i.e
0020 be 46 0f ae 2e e0 b7 5c ec 9d 17 dc 20 66 80 18 .F.....\ f..
0030 00 1d 46 c4 00 00 01 01 08 0a 90 78 59 86 22 6e .F.....: ..XY ."n
0040 67 43 54 68 73 20 69 73 20 74 68 65 20 54 43 gCThis i s the TC
0050 50 20 65 73 74 20 6d 65 73 61 67 65 2e P test m essage.

Den TCP Verbindungsauflösung und -abbau haben wir auch bereits beim (De-)Multiplexing sehen können.

Weiterführendes Material

- Video, das den Verbindungsauflauf- und -abbau im Zusammenhang mit dem Paketformat erklärt

<https://www.youtube.com/watch?v=F27PLin3TV0>



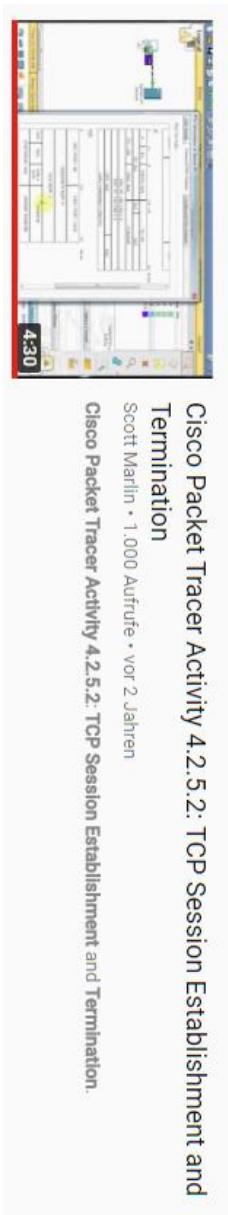
TCP connection walkthrough | Networking tutorial (13 of 13)

Ben Eater • 113.000 Aufrufe • vor 2 Jahren

Walk through TCP connection and termination packet by packet. Support me on Patreon:
<https://www.patreon.com/beneater> This ...

- Video, das anhand von Cisco Packet Tracer den Verbindungsauflauf- und -abbau bei TCP erklärt

<https://www.youtube.com/watch?v=NIA2gZ8uHtg>



Cisco Packet Tracer Activity 4.2.5.2: TCP Session Establishment and

Termination

Scott Marlin • 1.000 Aufrufe • vor 2 Jahren

Cisco Packet Tracer Activity 4.2.5.2: TCP Session Establishment and Termination.

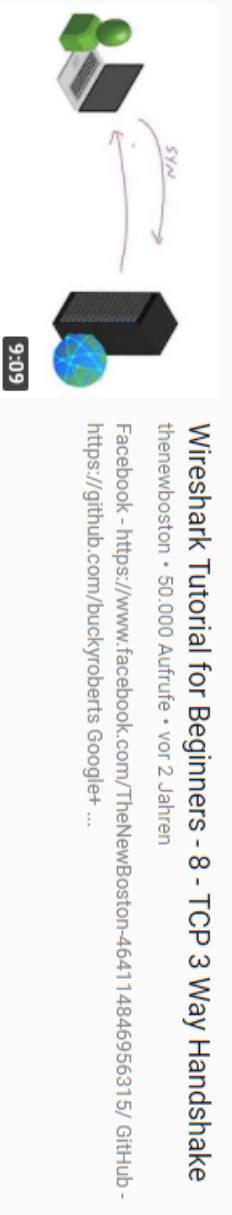
- Video, das den TCP Verbindungsauflauf- und -abbau anhand von Wireshark erklärt

https://www.youtube.com/watch?v=jL_UW0SK13nA

Wireshark Tutorial for Beginners - 8 - TCP 3 Way Handshake

thenewboston • 50.000 Aufrufe • vor 2 Jahren

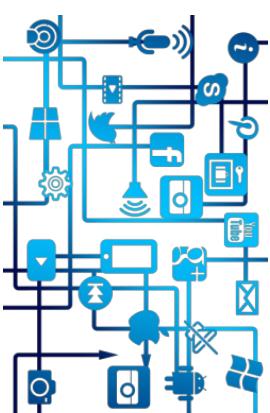
Facebook: <https://www.facebook.com/TheNewBoston-464114846956315/> GitHub: <https://github.com/buckyroborts> Google+: ...



9:09



9:30

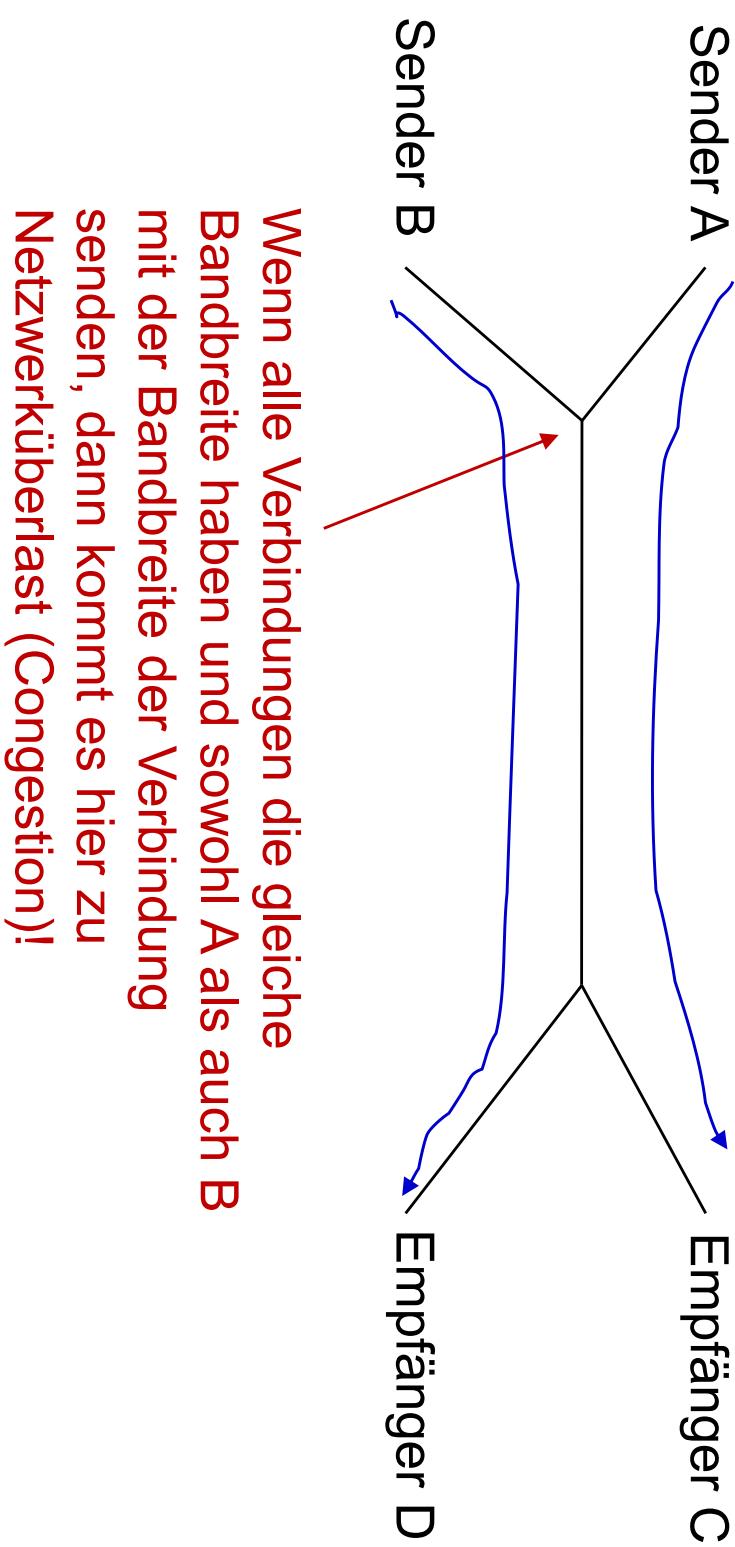


Agenda – VL 6-8

- Dienste der Transportschicht
- Multiplexing und Demultiplexing
- Verbindungslose Übertragung: UDP
- Zuverlässige Datenübertragung
- Verbindungsorientierte Übertragung: TCP
- Überlastkontrolle: Grundlagen und Umsetzung in TCP
- Zusammenfassung und Ausblick

Überlastsituationen

- Problem: Wenn alle Sender im Netz immer so viele Pakete losschicken, wie bei den Empfängern in den Puffer passen (s.o.), dann kann es zu Überlast (congestion) im Netz kommen, da nicht auf die momentane Situation im Netz Rücksicht genommen wird.



Wenn alle Verbindungen die gleiche Bandbreite haben und sowohl A als auch B mit der Bandbreite der Verbindung senden, dann kommt es hier zu Netzwerküberlast (Congestion)!

Überlastsituationen

- Überlast
 - Informell: „Zu viele Systeme senden zu viele Daten, das Netzwerk kann nicht mehr alles transportieren“
 - Wichtig: Unterschied zu der vorher betrachteten Flusskontrolle!
- Erkennungsmerkmale für Überlastsituationen
 - Paketverluste (Pufferüberlauf in den Routern)
 - Lange Verzögerungen (Warten in den Routern)
- Eines der zentralen Probleme in Computeretzwerken!

Überlastsituationen

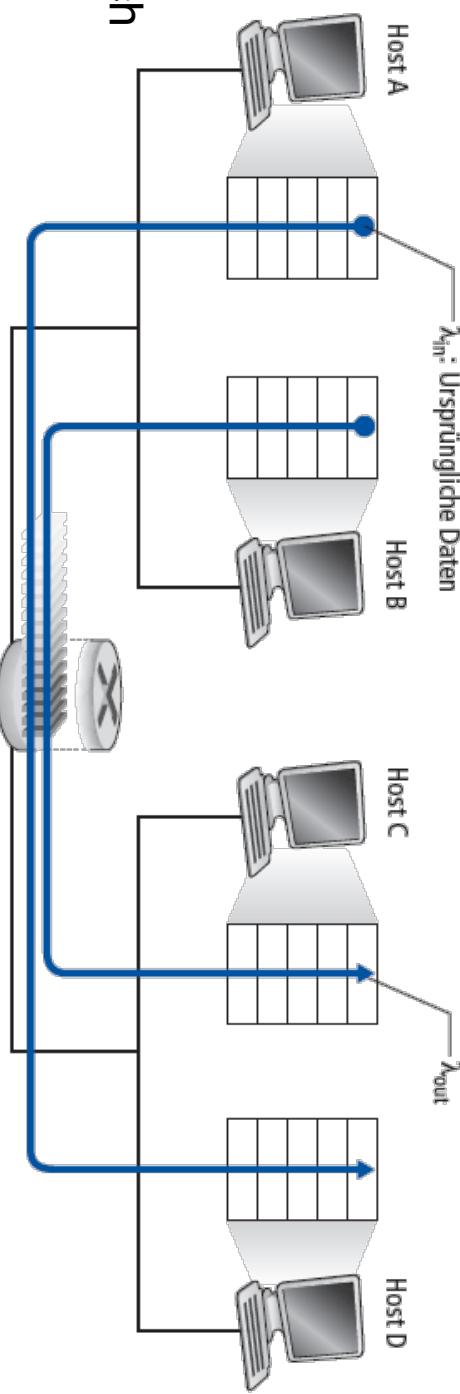
■ Beispiel 1:

- zwei Sender, zwei Empfänger, ein Router mit unendlichem Puffer, keine Übertragungs-wiederholungen

- Geteilte Bandbreite R, also $R/2$ je Host

Es hilft Host A nicht, mehr Daten an das Netzwerk zu übergeben (begrenzte Bandbreite $R/2$)

Je mehr Daten übergeben werden, desto größer wird die Verzögerung (Warteschlange im Puffer wächst stetig).



λ_{out}

$R/2$

Verzögerung

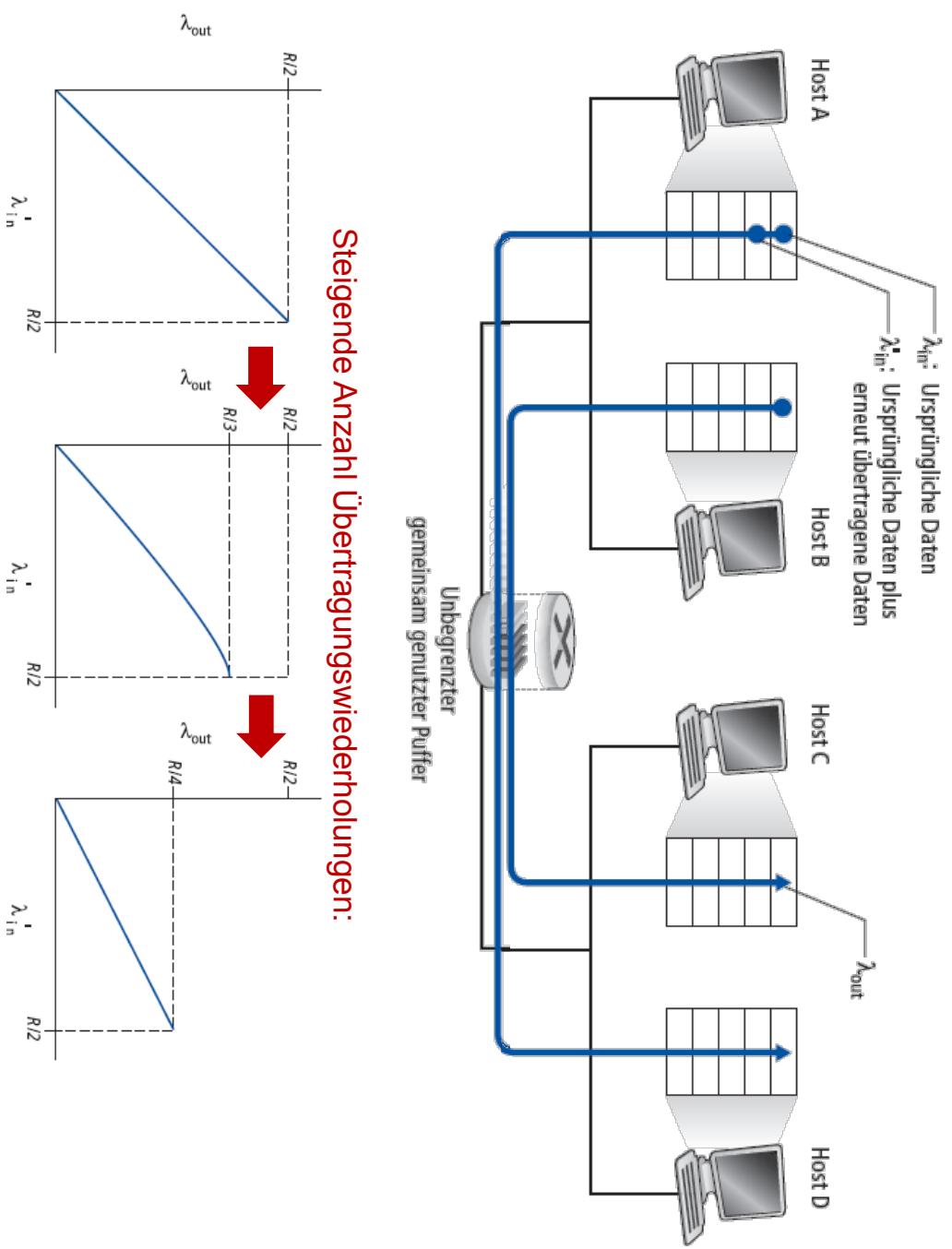
Überlastsituationen

■ Beispiel 2:

- zwei Sender, zwei Empfänger, ein Router mit begrenztem Puffer, führt zur Übertragungs-wiederholungen

Geteilte

Bandbreite R,
also $R/2$ je Host



Steigende Anzahl Übertragungswiederholungen:

■ Übertragungswiederholungen für verlorene sowie verzögerte Pakete machen λ'_{in} größer bei gleichem λ'_{out} (Ankunft beim Empfänger).

- Insgesamt mehr „Arbeit“ im Netzwerk durch Übertragungswiederholungen, d.h. ohne dass sich λ'_{in} und λ'_{out} verändern (dies nennt man auch denselben „Goodput“).
- Überflüssige Übertragungswiederholungen: Leitung befördert mehrere Kopien desselben Pakets

Ansätze zur Kontrolle von Überlast

Ende-zu-Ende

- Keine explizite Unterstützung durch das Netzwerk
- Überlast wird von den Endsystemen durch Paketverlust und erhöhte Verzögerung festgestellt.
- Dies ist das Vorgehen im Internet (TCP).

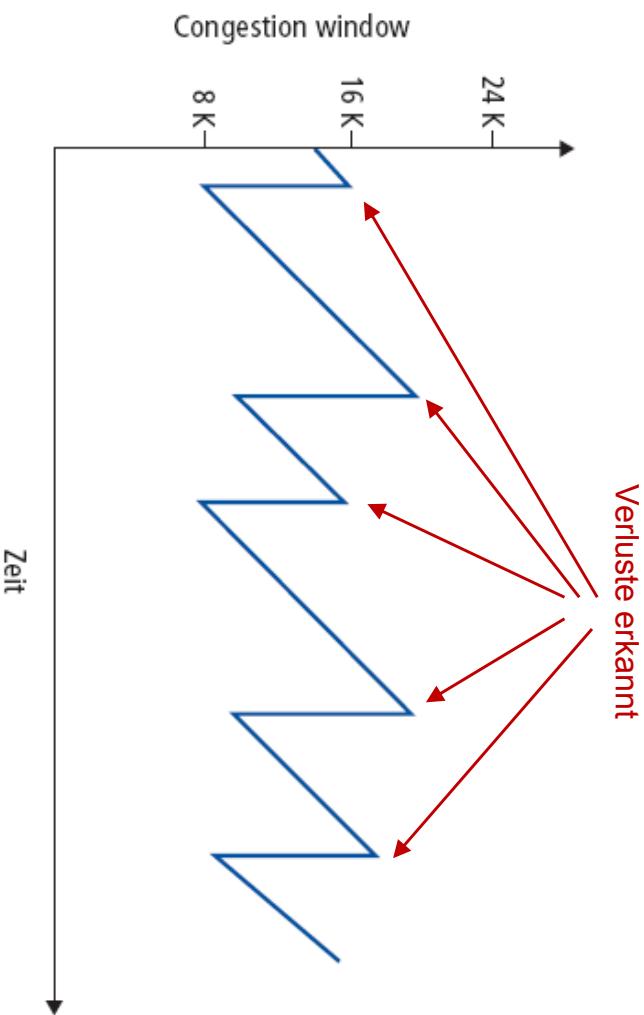
Netzwerkunterstützt

- Router geben den Endsystemen Hinweise:
 - Durch ein einzelnes Bit, welches im Paketheader von den Routern gesetzt werden kann (SNA, DECbit, TCP/IP ECN, ATM)
 - Durch Vorgabe einer expliziten Senderate

TCP-Überlastkontrolle

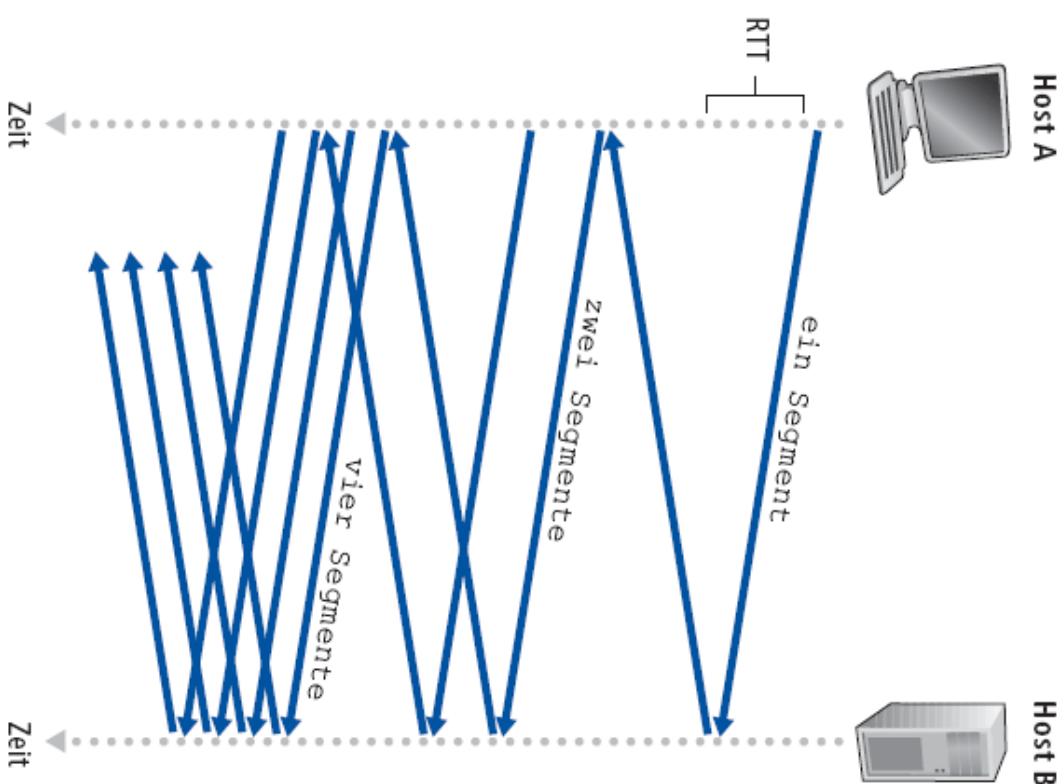
Additive Increase / Multiplicative Decrease

- Ansatz: Erhöhe die Übertragungsrate (Fenstergröße), um nach überschüssiger Bandbreite zu suchen, bis ein Verlust eintritt
 - Additive Increase: Erhöhe das „congestion window“ (das Fenster beim Sender, s.o.) um 1, bis ein Verlust erkannt wird.
 - Multiplicative Decrease: Halbiere das „congestion window“, wenn ein Verlust erkannt wird.



TCP-Überlastkontrolle: Slow Start

- Bei Verbindungsbeginn:
Erhöhe die Rate exponentiell schnell bis zum ersten Verlustereignis
- Verdoppeln von CongWin



■ Ergebnis: Initiale Rate ist gering, wächst aber exponentiell schnell

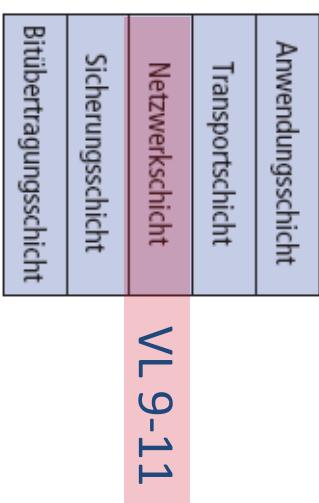
Agenda – VL 6-8

- **Zusammenfassung und Ausblick**
 - Überlastkontrolle: Grundlagen und Umsetzung in TCP
- **Dienste der Transportschicht**
 - Multiplexing und Demultiplexing
 - Verbindungslose Übertragung: UDP
 - Zuverlässige Datenübertragung
 - Verbindungsorientierte Übertragung: TCP

Zusammenfassung (VL 6-8) und Ausblick



- In den letzten beiden Vorlesungen haben wir uns mit der **Transportschicht** befasst:
 - Die Konzeption und Implementierung von Protokollen der Anwendungsschicht wurde behandelt und an beispielhaften Protokollen im Detail analysiert.
 - Protokolle sind die Grundlage für sämtliche Kommunikation in Netzwerken!
 - Sie haben verschiedene Anwendungs-Architekturen in Bezug auf Netzwerke kennengelernt.
 - Sie sollten nun das Domain Name System verstehen und einen ersten Einblick in die Socket-Programmierung erhalten haben.
- In den folgenden beiden Veranstaltungen werden wir uns dem top-down Ansatz folgend auf die nächste Schicht konzentrieren, die für die Transportschicht die Übertragungsdienste bereitstellt: **Netzwerkschicht**





Vielen Dank für Ihre Aufmerksamkeit!

**BW332 - Rechnernetze
(mit Praktikum)**

**Vorlesungen 9-11
Netzwerkschicht**

Ziele von VL 9-11

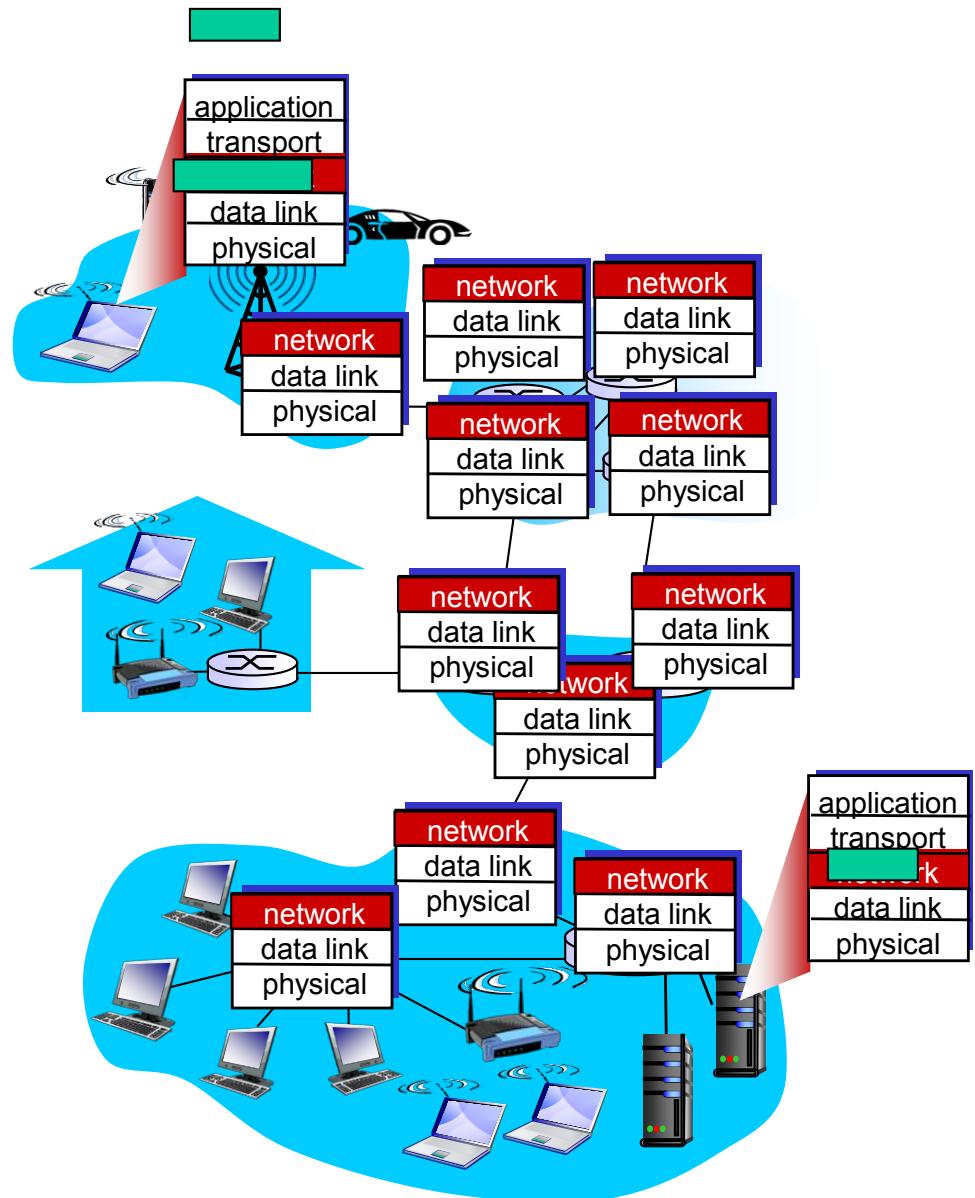


- Die Grundlagen von IP und Datagrammnetzwerken verstehen und von verbindungsorientierten Netzwerkschichten abgrenzen können.
- Die Hauptaufgaben eines Routers verstehen und den prinzipiellen Aufbau wiedergeben können.
- Das IP-Protokoll verstehen und in den Kontext des Internet-Protokollstapels setzen können.
 - Dabei Aspekte wie Fragmentierung und Adressierung wiedergeben und die Herausforderungen in IPv4 diskutieren können (inkl. NAT).
 - Die dynamische IP-Adressierung nachvollziehen und anwenden können.
- Einen Überblick über das Routing im Internet erhalten und anhand des Cisco Packet Tracers zwei lokale Netzwerke miteinander verbinden.

- Einleitung und Datagramme vs. virtuelle Leitungen**
- Aufbau eines Routers
- IP – Internet Protocol
- Routing im Internet
- Zusammenfassung und Ausblick

Netzwerkschicht

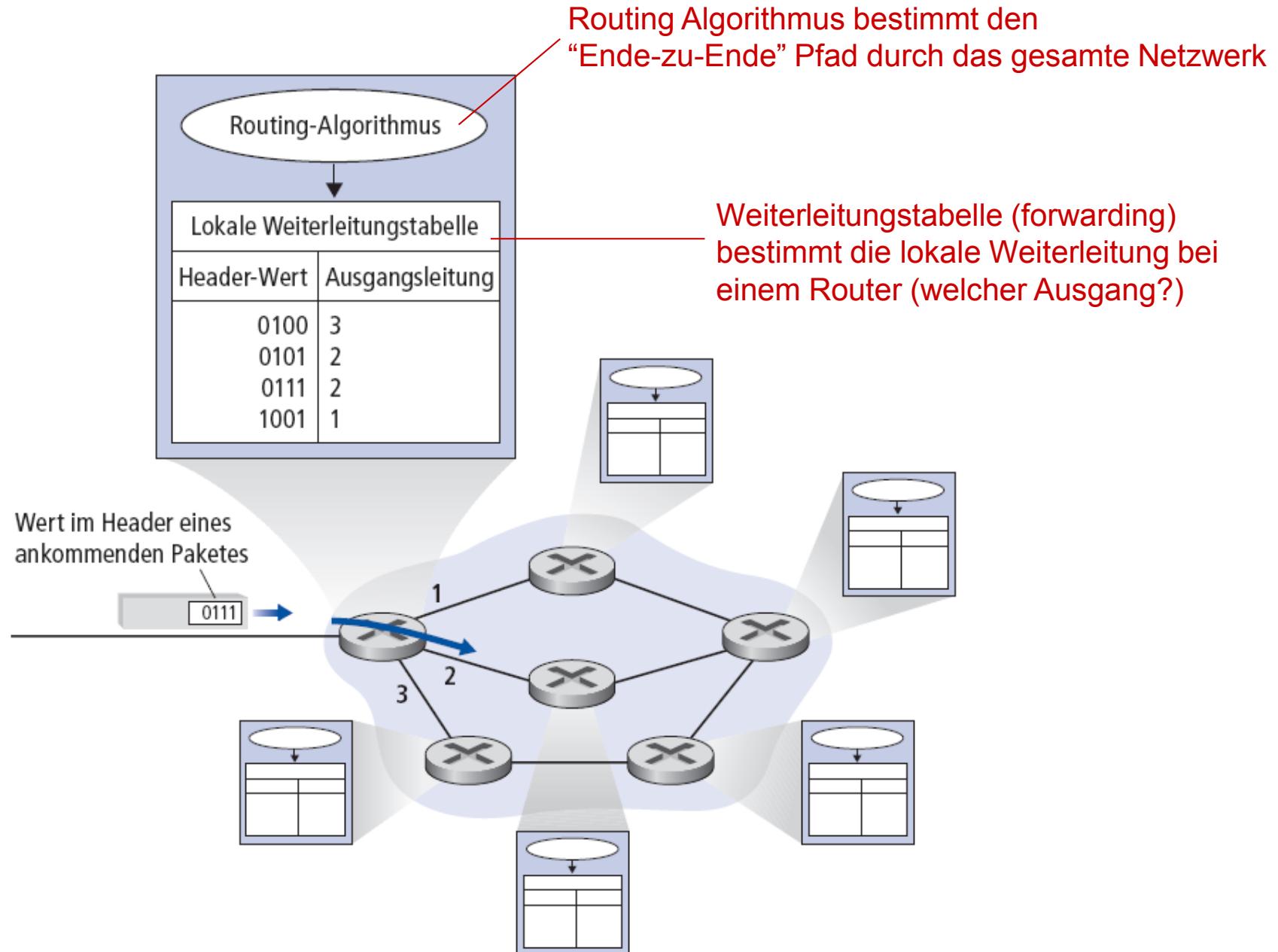
- Auch als Vermittlungsschicht oder „Network Layer“ bezeichnet
- Daten von der nächsthöheren Schicht (Transportschicht) des Senders entgegennehmen
 - In Datagramme verpacken
 - Durch das Netzwerk leiten
- Auspacken des Vermittlungspakets beim Empfänger
- Ausliefern der Daten an die nächsthöhere Schicht (Transportschicht) des Empfängers
- Die Netzwerkschicht existiert in jedem Host und Router!



Funktionen der Netzwerkschicht

- Weiterleiten von Paketen (Forwarding):
 - Router nimmt Paket auf einer Eingangsleitung entgegen
 - Router bestimmt die Ausgangsleitung anhand lokaler Informationen (z.B. Routing-Tabelle)
 - Router legt das Paket auf die Ausgangsleitung
- Wegewahl (Routing):
 - Router kommunizieren miteinander, um geeignete Wege durch das Netzwerk zu bestimmen
 - Als Ergebnis erhalten sie Informationen, wie welches System im Netzwerk zu erreichen ist (z.B. wird eine Routingtabelle mit Einträgen gefüllt)
- Metapher:
 - Routing = Planen einer Strecke für eine Autofahrt
 - Weiterleitung = Verhalten an einer Autobahnkreuzung

Routing und Forwarding



Verbindungsau- und -abbau

- Es gibt Protokolle auf der Netzwerkschicht, die mittels virtueller Verbindungen arbeiten. Bei diesen Protokollen ist dann die dritte wichtige Funktion der Verbindungsau- und -abbau
 - Beispielprotokolle: ATM (Asynchronous Transfer Mode), Frame Relay, X.25
- Bevor Daten übertragen werden können, wird eine virtuelle Verbindung vom Sender zum Empfänger aufgebaut
 - Nach der Übertragung wird diese Verbindung wieder abgebaut
- Abgrenzung der Schichten bei derartigen Protokollen:
 - **Netzwerkschicht:** virtuelle Verbindung zwischen zwei Hosts (dazwischenliegende Router können involviert sein)
 - **Transportschicht:** zwischen zwei Prozessen
- Das Internet setzt auf der Netzwerkschicht keine virtuellen Verbindungen ein!

Dienstmodelle der Netzwerkschicht

- Genau wie bei der Transportschicht kann man sich auch bei der Netzwerkschicht Gedanken über das „Dienstmodell“ machen: was „leistet“ die Netzwerkschicht für die Transportschicht?

Beispiele für einzelne Datagramme:

- Garantierte Zustellung
- Garantierte Zustellung in weniger als 40 ms

Beispiele für einen Fluss von Datagrammen:

- Reihenfolgeerhaltende Auslieferung
- Garantierte minimale Datenrate
- Beschränkung der Schwankungen in der Zeit, die für den Transport von Data-grammen benötigt wird

Dienstmodelle der Netzwerkschicht

Netzwerk-architektur	Dienst-modell	Band-breiten-garantien	Garantie der Ver-lustfreiheit	Reihen-folge	Zeit-garantien	Hinweis auf Überlast
Internet	Best Effort	keine	nein	beliebige möglich	nicht unterstützt	keiner
ATM	CBR	garantiert eine konstante Rate	ja	in korrekter Reihenfolge	unterstützt	Überlast tritt nicht auf
ATM	ABR	garantiertes Minimum	nein	in korrekter Reihenfolge	nicht unterstützt	Überlasthinweise werden verwendet

Verbindungsorientierte vs. Verbindungslose Netzwerkschicht

- Ein Datagrammnetzwerk verwendet eine verbindungslose Netzwerkschicht (Internet!)
- Ein Netzwerk mit virtuellen Leitungen verwendet eine verbindungsorientierte Netzwerkschicht (z.B. ATM)
 - Analog zur Transportschicht. Aber:
 - Dienst bei der Netzwerkschicht: **Host-zu-Host** (nicht Prozess-zu-Prozess)
 - Es gibt im Gegensatz zur Transportschicht keine Wahlmöglichkeit: Ein Netzwerk bietet das eine oder das andere an
 - Transportschicht: Wahl zwischen TCP und UDP
 - Implementierung: im Inneren des Netzwerkes (im Gegensatz zur Transportschicht)
 - Achtung! In höheren Schichten kann – wie wir bereits wissen – noch ein verbindungsorientierter Dienst (z.B. TCP) über eine verbindungslose Netzwerkschicht (z.B. IP) realisiert werden!

Verbindungsorientierte Netzwerkschicht

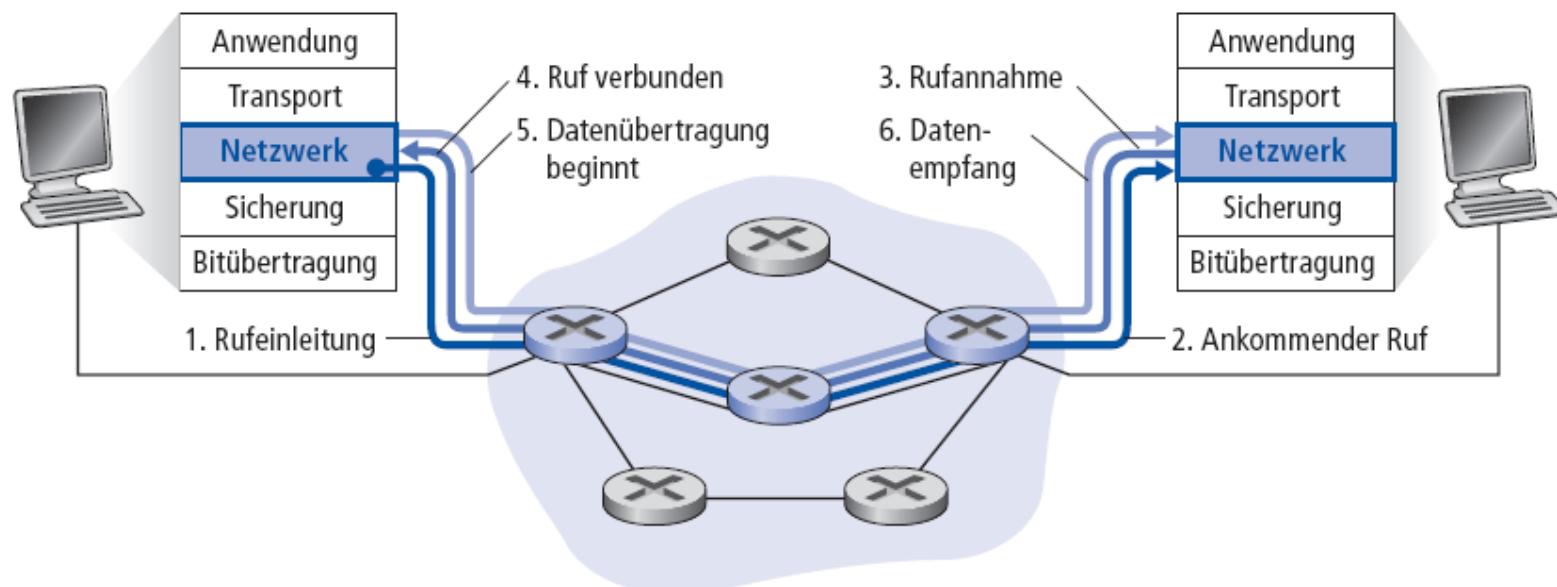
Virtuelle Leitungen: Der Pfad zwischen Sender und Empfänger verhält sich wie eine Telefonleitung.

- Aufbau einer Verbindung, bevor Daten transportiert werden können.
Danach: Abbau der Verbindung
- Jedes Paket beinhaltet einen **VC-Identifier** (VC = Virtual Channel) und keine Zieladresse (wie bei Datagrammen)
- *Jeder* Router auf dem Pfad vom Sender zum Empfänger verwaltet einen **Zustand** für diese Verbindung.
- Ressourcen des Routers können einer virtuellen Leitung zugeordnet sein.
 - zugeordnete Ressourcen = vorhersagbare Dienstgüte = fest definierbarer Quality-of-Service (QoS)

Verbindungsorientierte Netzwerkschicht

Signalisierungsprotokolle

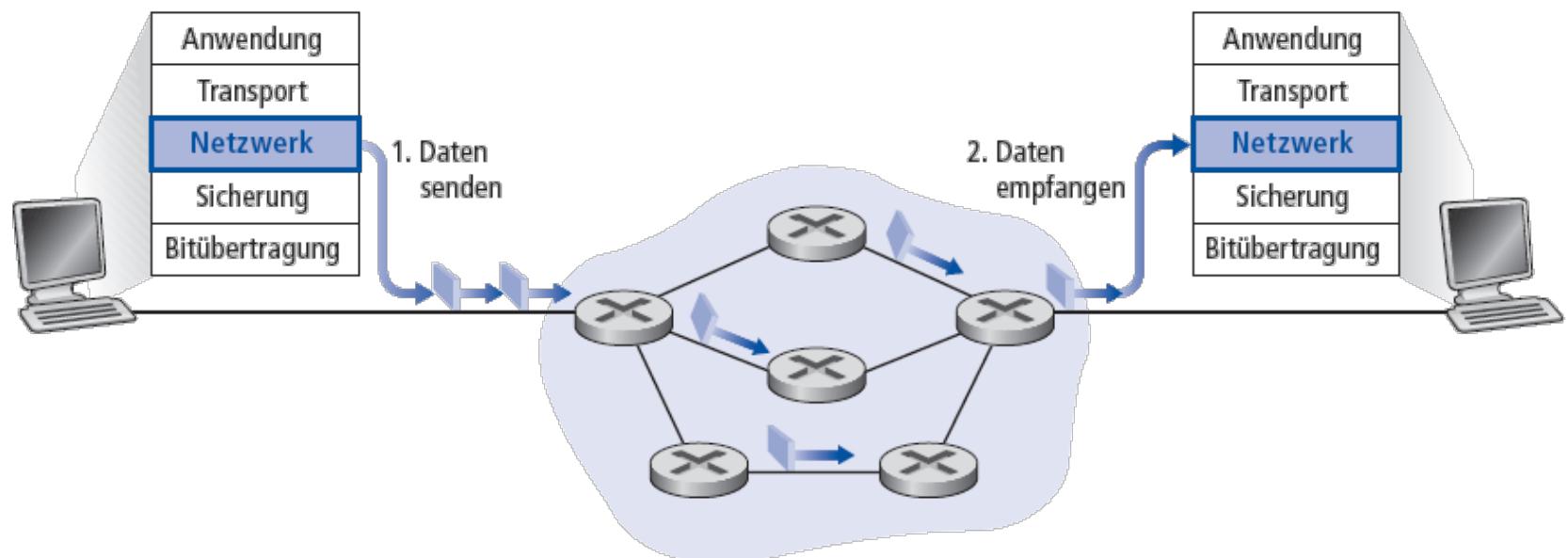
- Verwendet zum Aufbau, Aufrechterhalten und Abbau von virtuellen Leitungen
- Verwendet in ATM, Frame Relay und X.25
- Nicht im Internet!



Verbindungslose Netzwerkschicht

Datagrammnetzwerke

- Kein Verbindungsaufbau auf der Netzwerkschicht
- Router halten keinen Zustand für Ende-zu-Ende-Verbindungen
 - Auf Netzwerkebene gibt es das Konzept einer „Verbindung“ nicht
- Pakete werden unter Verwendung einer Zieladresse weitergeleitet
 - Pakete für dasselbe Sender-Empfänger-Paar können unterschiedliche Pfade nehmen



Verbindungslose Netzwerkschicht

Weiterleitungstabelle (Bsp.)

Zieladressbereiche statt einzelner Adressen

Zieladressbereich	Schnittstelle
11001000 00010111 00010000 00000000	
bis	0
11001000 00010111 00010111 11111111	
11001000 00010111 00011000 00000000	
bis	1
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000	
bis	2
11001000 00010111 00011111 11111111	
sonst	3

Verbindungsorientierte vs. Verbindungslose Netzwerkschicht

Internet

- Datenaustausch zwischen Computern
 - Keine Echtzeitanforderungen
- Mächtige Endsysteme
 - Können sich anpassen und Fehler beheben
 - Konsequenz: einfaches Netzwerk, **Komplexität in den Endsystemen**
- Vielzahl verschiedener Links
 - Unterschiedliche Charakteristika
 - Einheitlicher Dienst schwierig zu realisieren

ATM

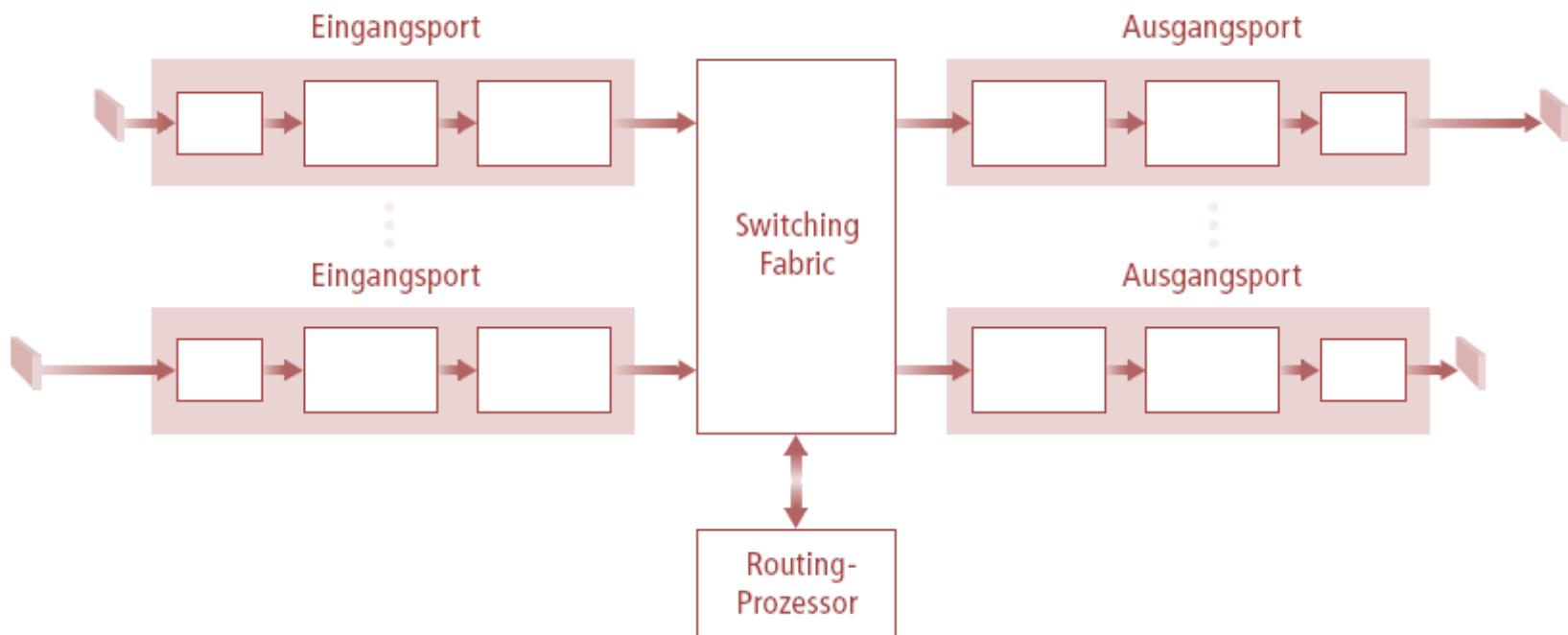
- Stammt von der klassischen Telefontechnologie ab
- Menschliche Kommunikation
 - Hohe Anforderungen an Echtzeit und Zuverlässigkeit
 - Dienstgarantien sind notwendig
- Einfache Endsysteme
 - Telefone
 - Konsequenz: einfache Endsysteme, **Komplexität im Netzwerk**

- Einleitung und Datagramme vs. virtuelle Leitungen**
- Aufbau eines Routers**
- IP – Internet Protocol**
- Routing im Internet**
- Zusammenfassung und Ausblick**

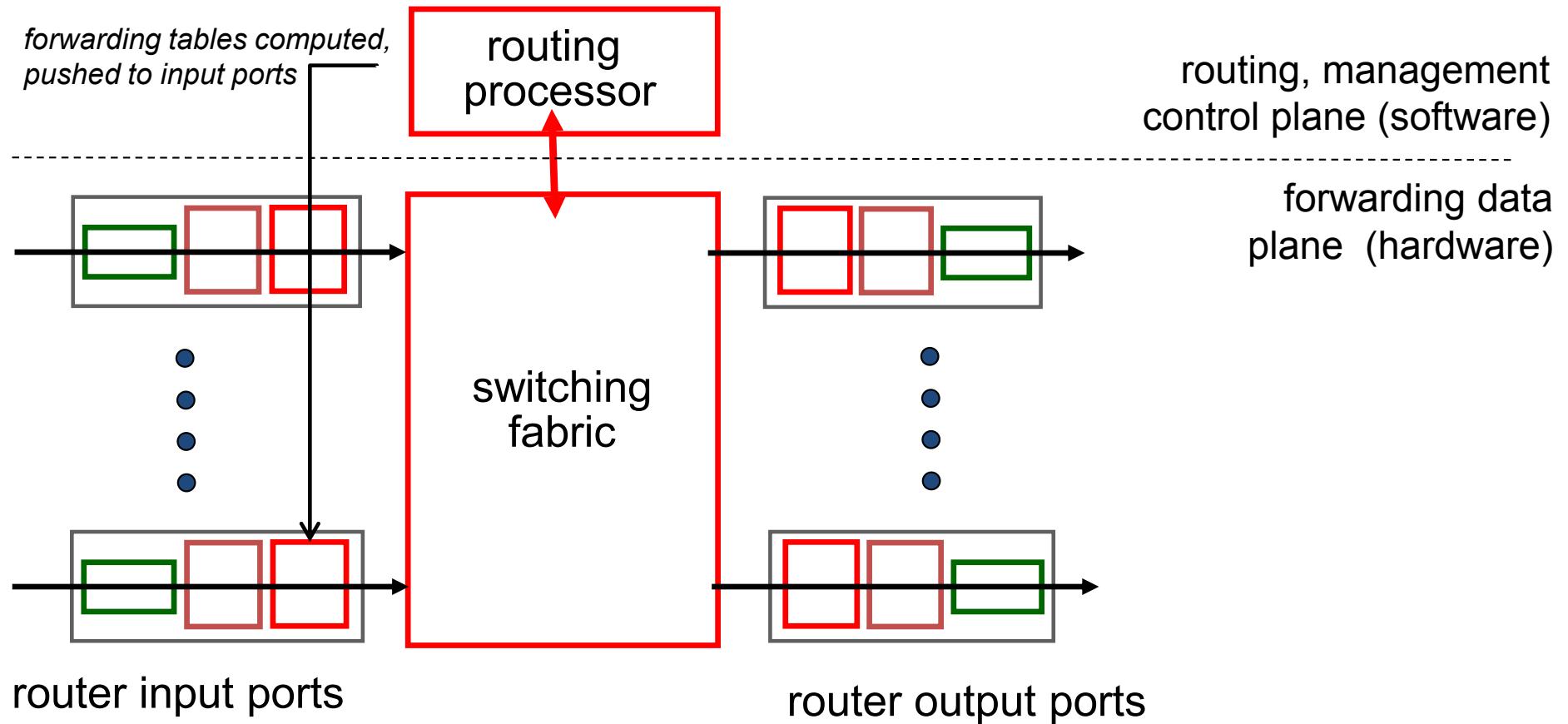
Übersicht über Routerarchitekturen

Router haben zwei wichtige Aufgaben

- Ausführen von Routing-Algorithmen und -Protokollen
 - RIP, OSPF, BGP
- Weiterleiten von Datagrammen von einem eingehenden zu einem ausgehenden Link

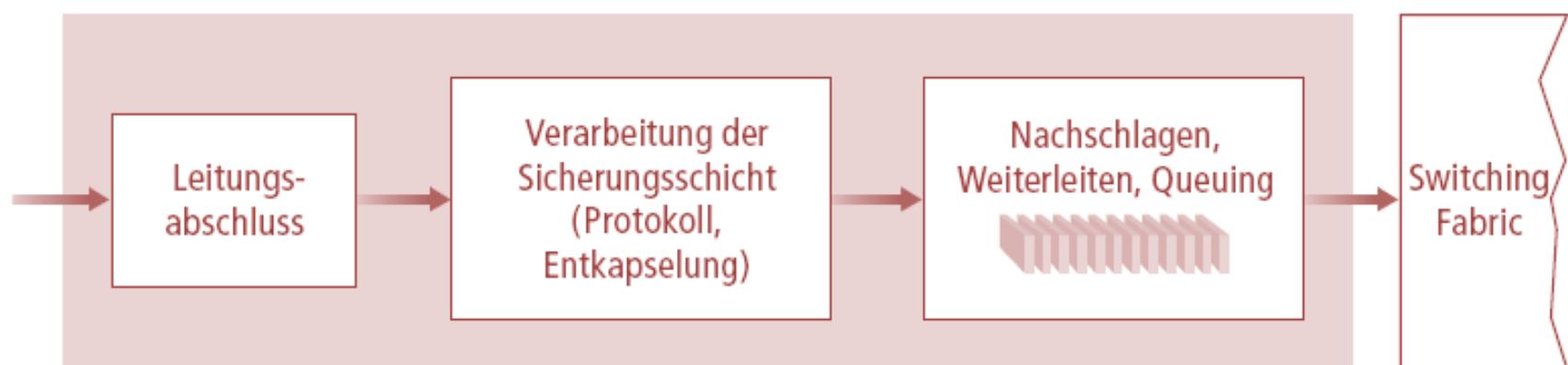


Übersicht über Routerarchitekturen

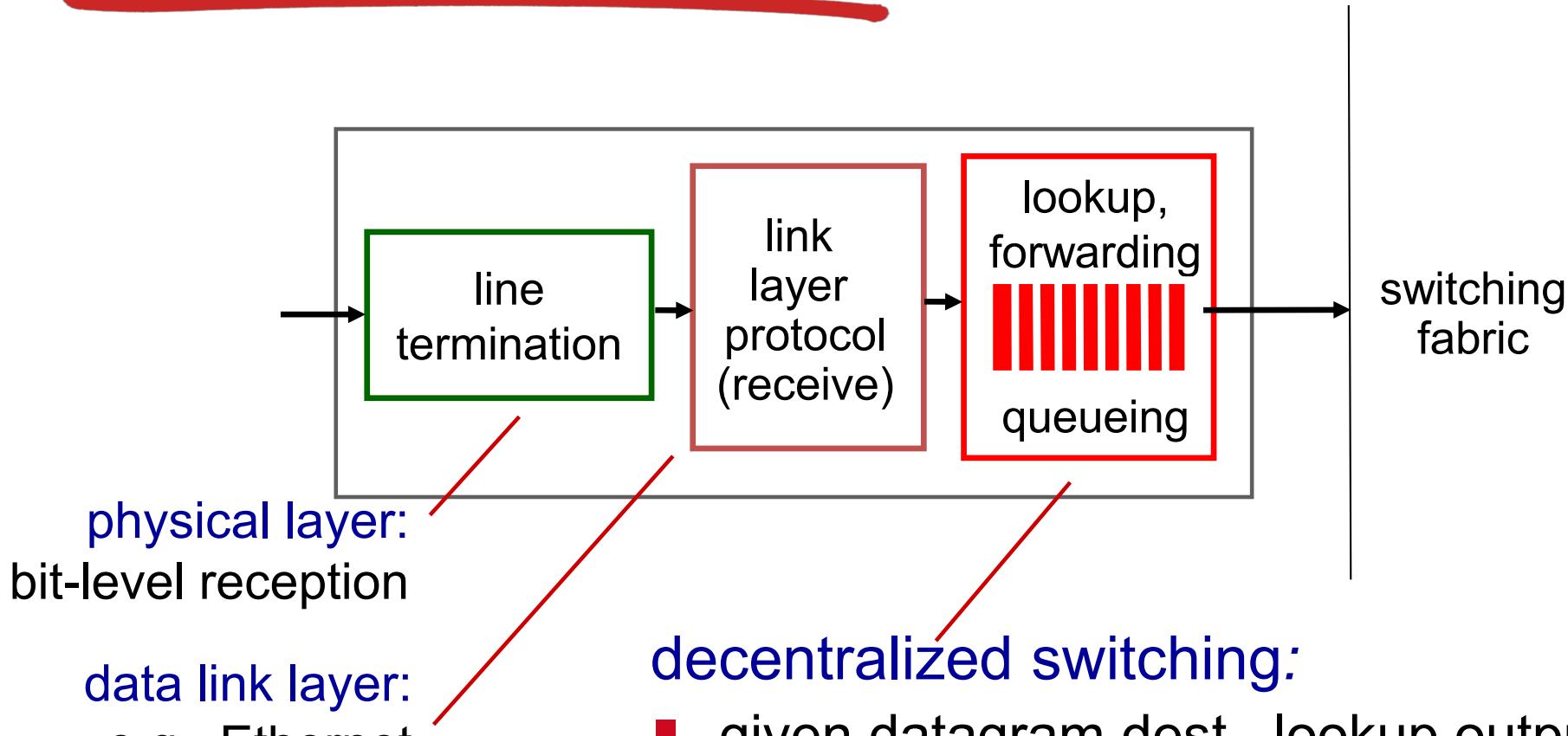


Verarbeitung am Eingangsport

- Implementierung der unter der Netzwerkschicht liegenden Schichten
 - Sicherungsschicht: z.B. Ethernet
 - Leitungsabschluss: physikalische Schicht, Bits empfangen
- Nachschlagen, Weiterleiten, Queuing:
 - Suche nach einem geeigneten Ausgangsport
 - Dezentral, Kopie der Routing-Tabelle (oder Teile davon) notwendig
 - Ziel: Behandlung der Pakete mit „line speed“, also mit der Geschwindigkeit der Eingangsleitung des Ports
 - Puffern von Paketen, wenn die Switching Fabric belegt ist



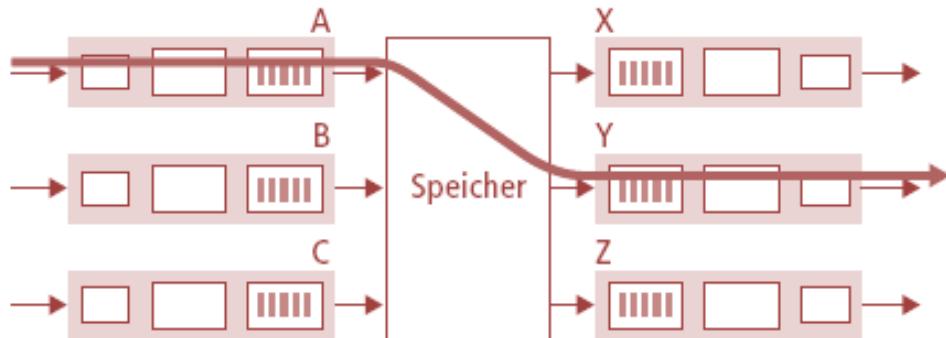
Verarbeitung am Eingangsport



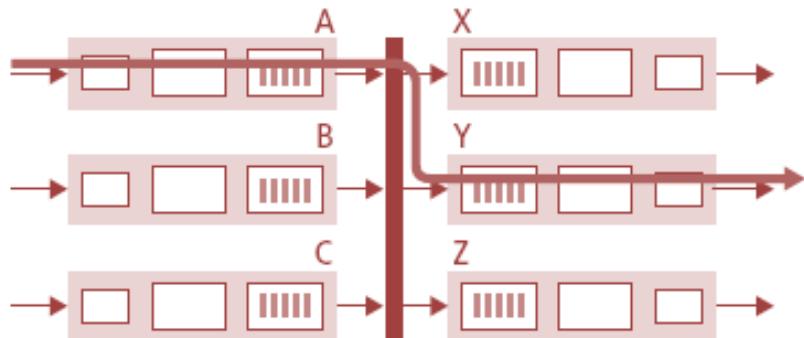
- given datagram dest., lookup output port using forwarding table in input port memory ("*match plus action*")
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Switching Fabric: 3 Arten

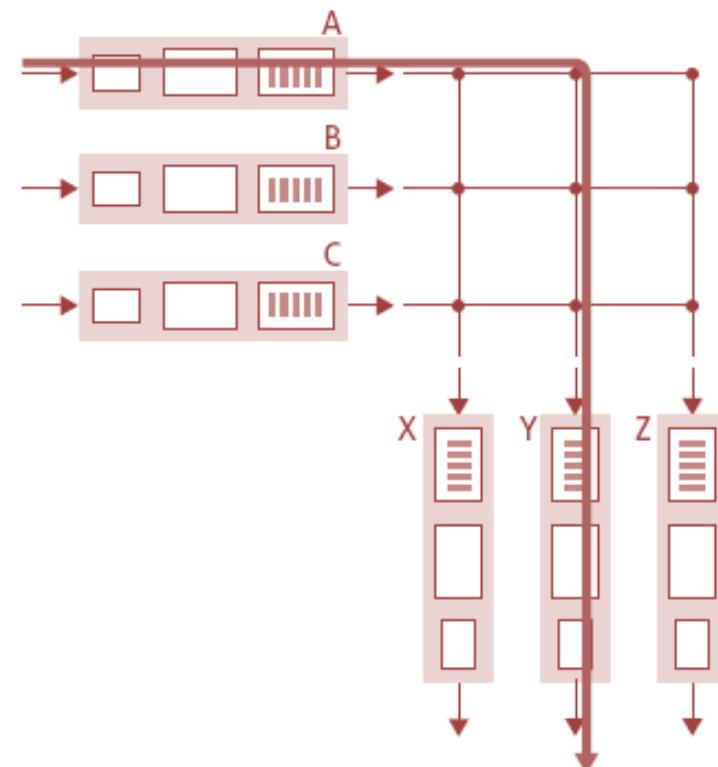
Speicher



Bus



Crossbar



Legende:

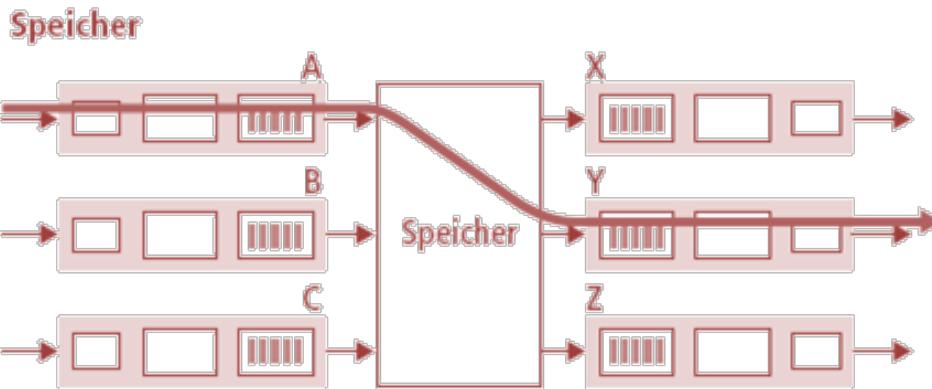


Eingangsport



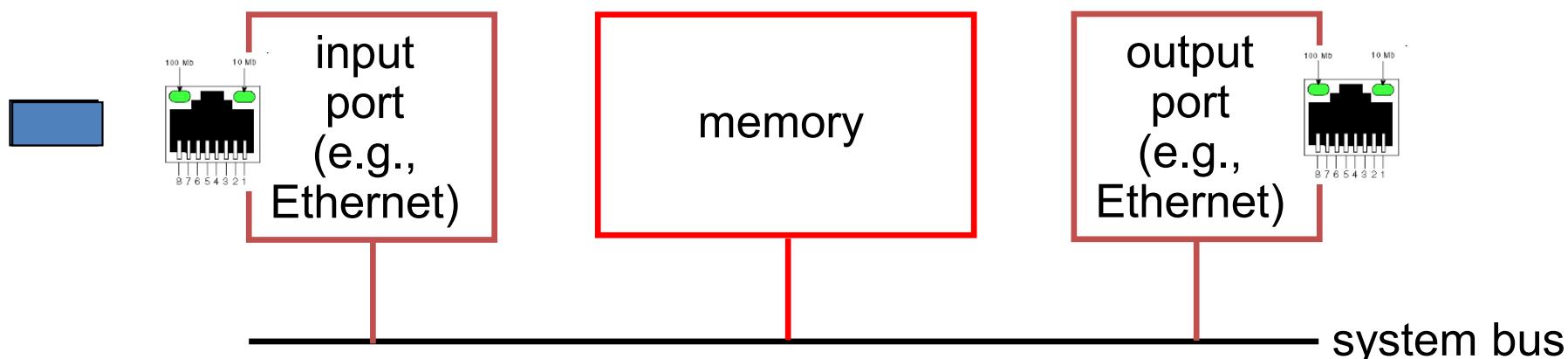
Ausgangsport

Switching Fabric: Speicher

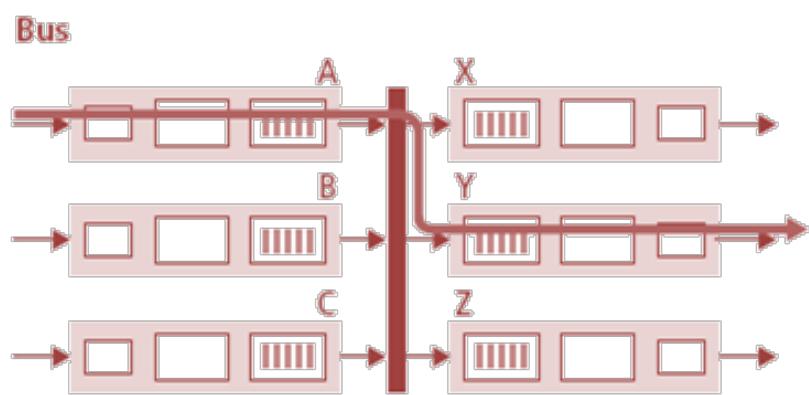


■ Erste Routergeneration

- Paket von Eingangsport in Hauptspeicher kopieren
- Paket vom Hauptspeicher in Ausgangsport kopieren
- Geschwindigkeit durch Speicherbus beschränkt!
 - Zwei Speicherzugriffe: einer zum Schreiben, einer zum Lesen

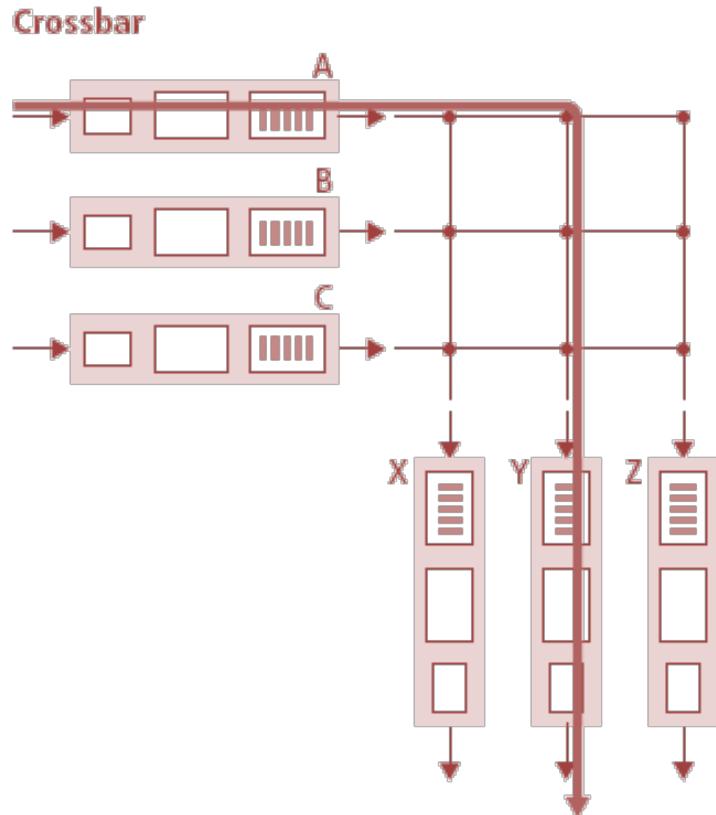


Switching Fabric: Bus



- Alle Ports teilen sich einen gemeinsamen Bus
- Bus Contention: Die gesamte Kommunikation erfolgt über den Bus, dieser beschränkt die Bandbreite des Routers
 - Aber: nur eine Busoperation (nicht zwei!)
- Beispiel: 32-Gbps-Bus, Cisco 5600, ausreichend für Zugangsrouter und Router für Firmennetze (nicht geeignet im Backbone)

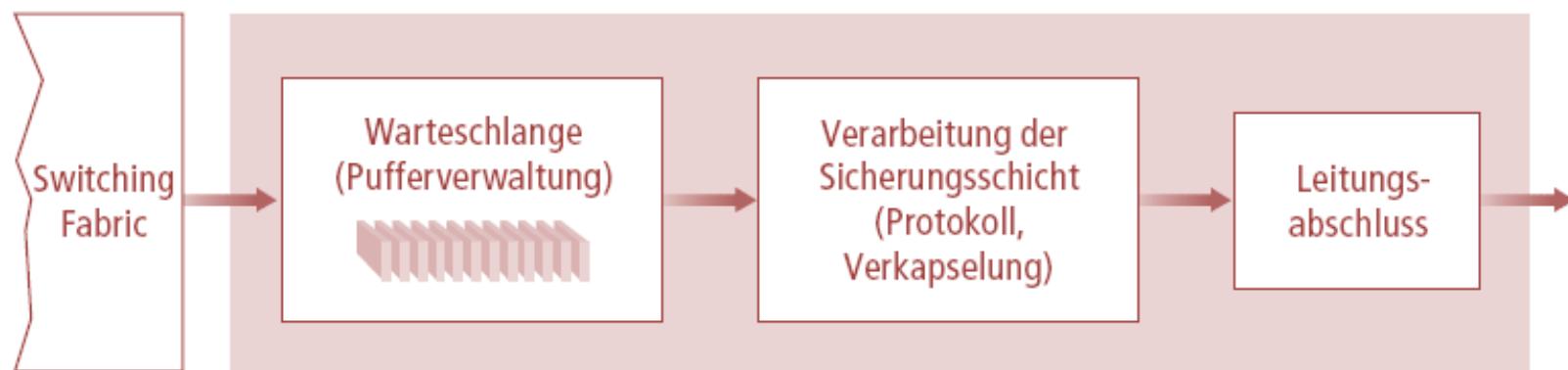
Switching Fabric: Spezialnetz



- Ports sind über ein Netzwerk miteinander verbunden
 - Beispielsweise alle Eingangsports über einen Crossbar mit allen Ausgangsports
 - Technologie ursprünglich für das Verbinden mehrerer Prozessoren in einem Parallelrechner entwickelt
- Beispiel: Cisco 12000, Switching von 60 Gbps durch das interne Netz

Verarbeitung am Ausgangsport

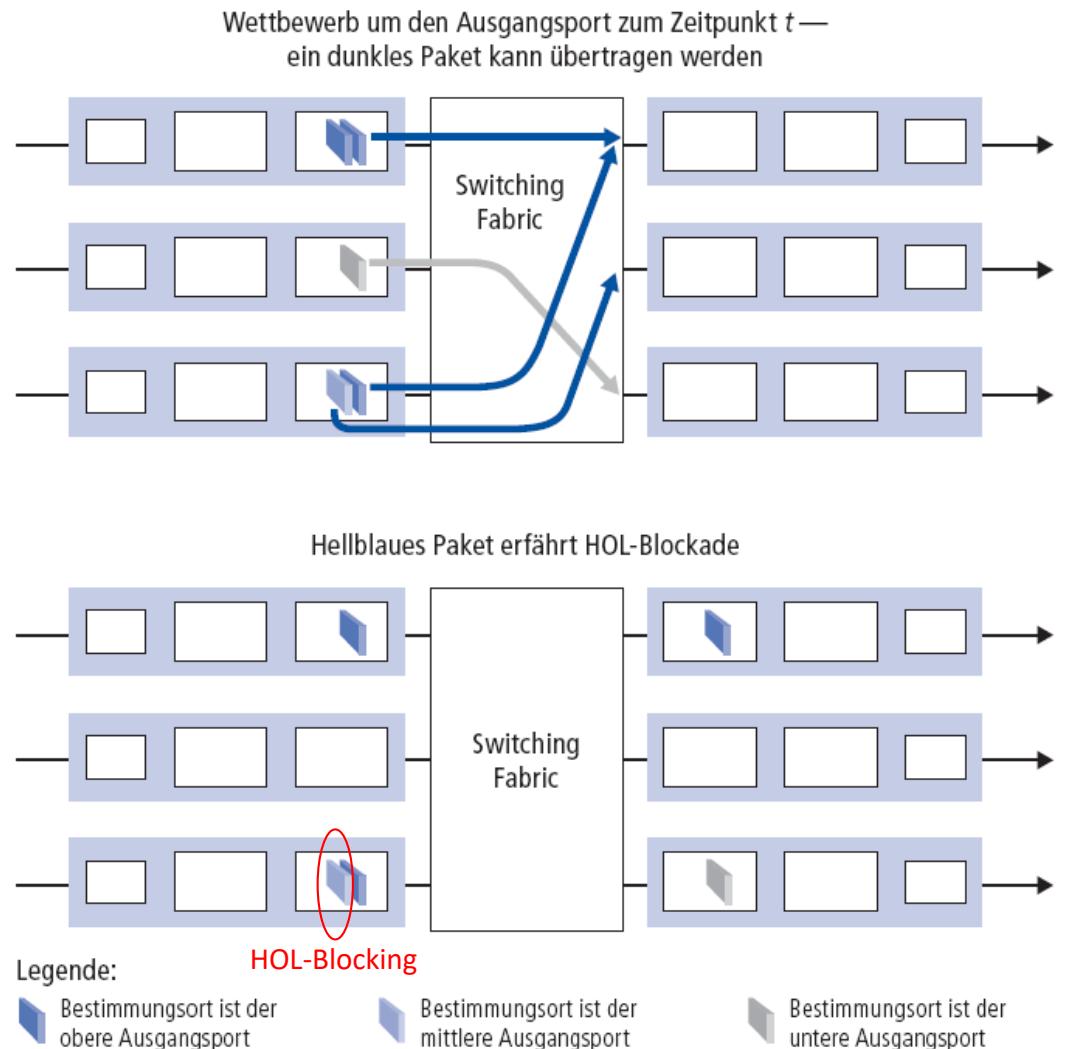
- Prinzipiell: analog zum Eingangsport, aber einfacher, da die Entscheidung über die Weiterleitung schon getroffen wurde



- Puffern von Paketen, wenn sie schneller aus der Switching Fabric kommen, als sie auf die Leitung gelegt werden können
- Auswirkungen:
 - Gepufferte Pakete werden verzögert
 - Wenn der Puffer überläuft, müssen Pakete verworfen werden

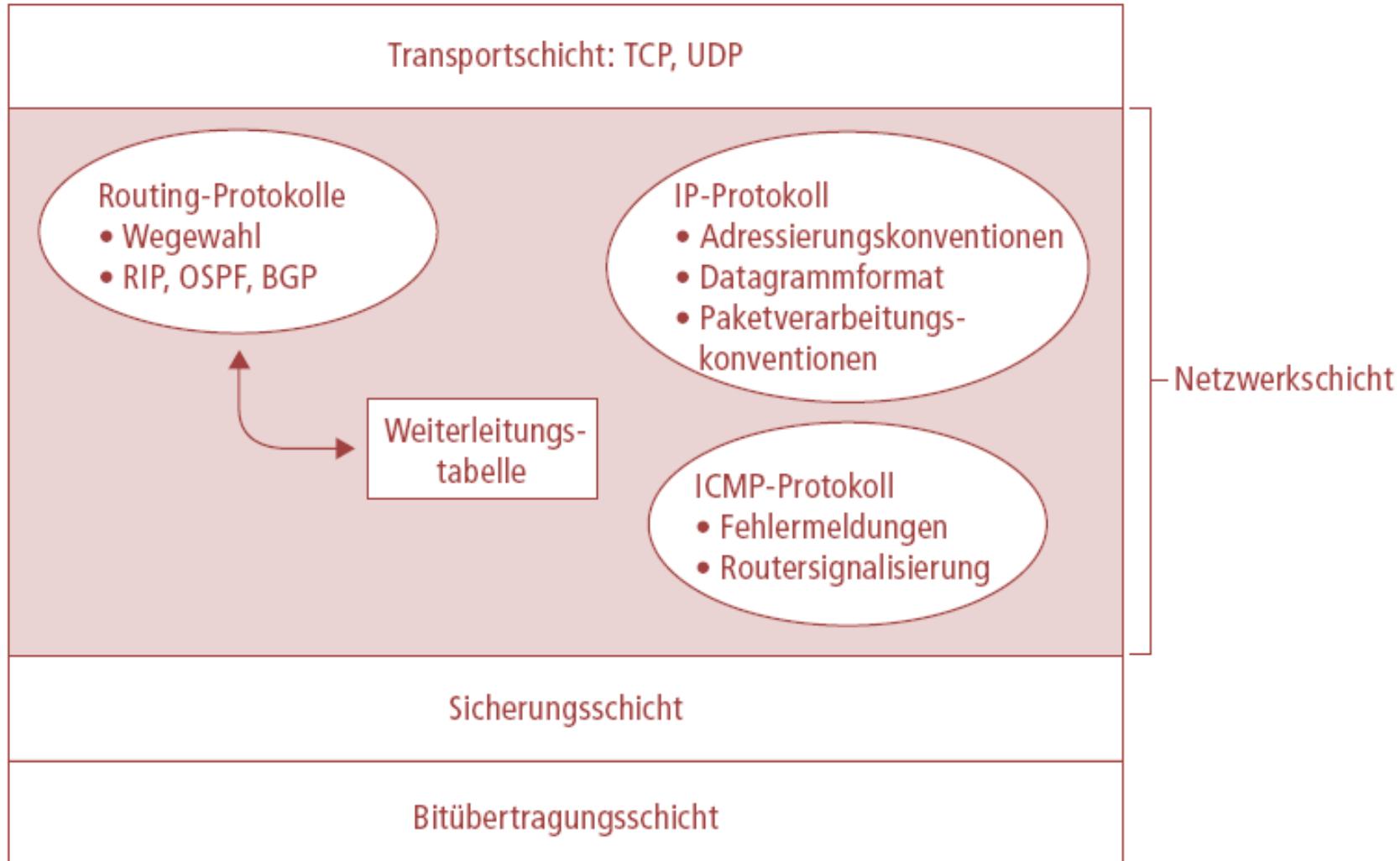
Puffern am Eingangsport

- Wenn die Switching Fabric ein Paket nicht direkt weiterleiten kann, muss dieses im Eingangsport gepuffert werden
- Dort kann es ein Paket blockieren, das eigentlich bereits durch die Switching Fabric geleitet werden könnte.
 - Head-of-Line (HOL) Blocking

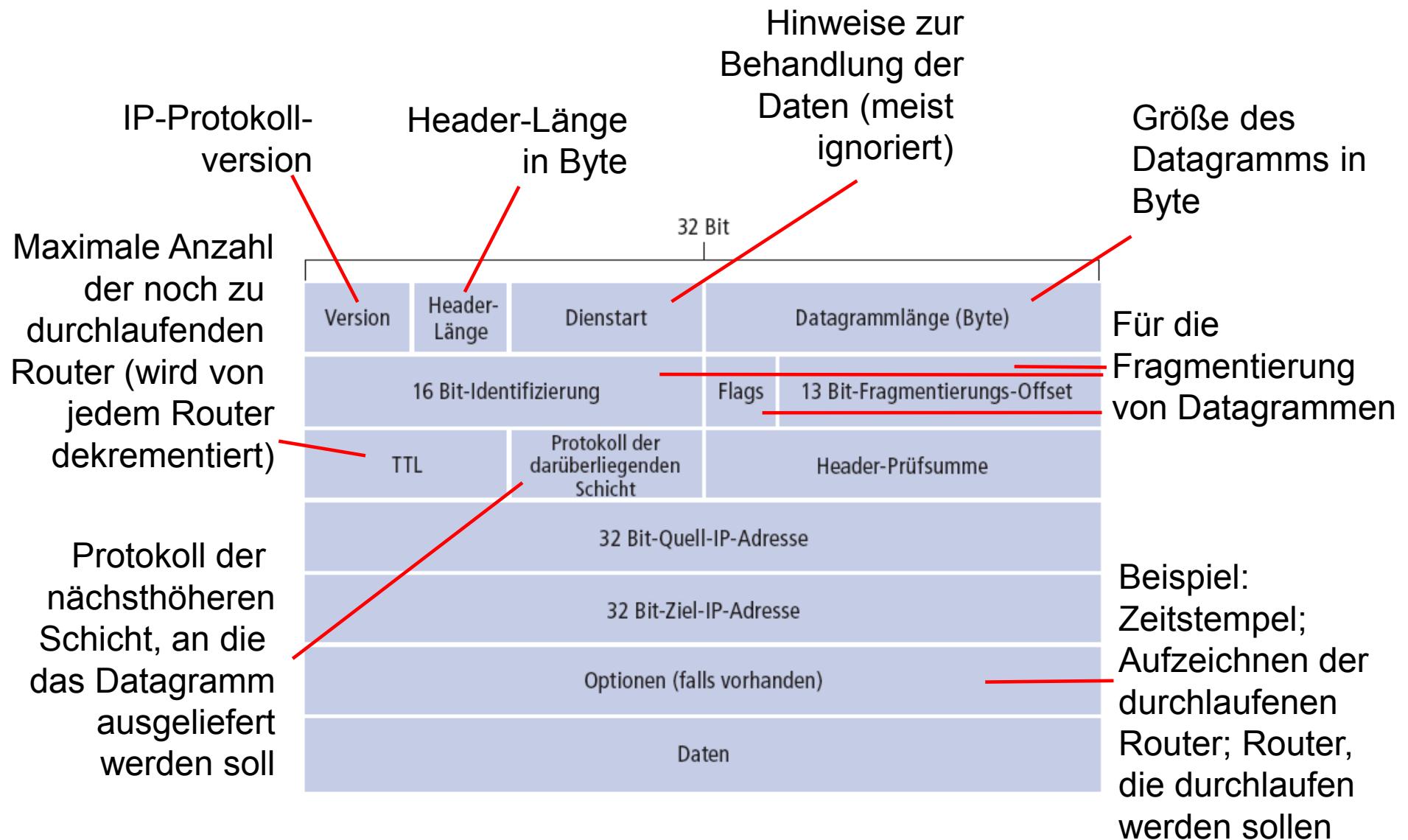


- Einleitung und Datagramme vs. virtuelle Leitungen**
- Aufbau eines Routers**
- IP – Internet Protocol**
- Routing im Internet**
- Zusammenfassung und Ausblick**

Netzwerkschicht: Übersicht der Funktionalitäten



Netzwerkschicht: Datagrammformat



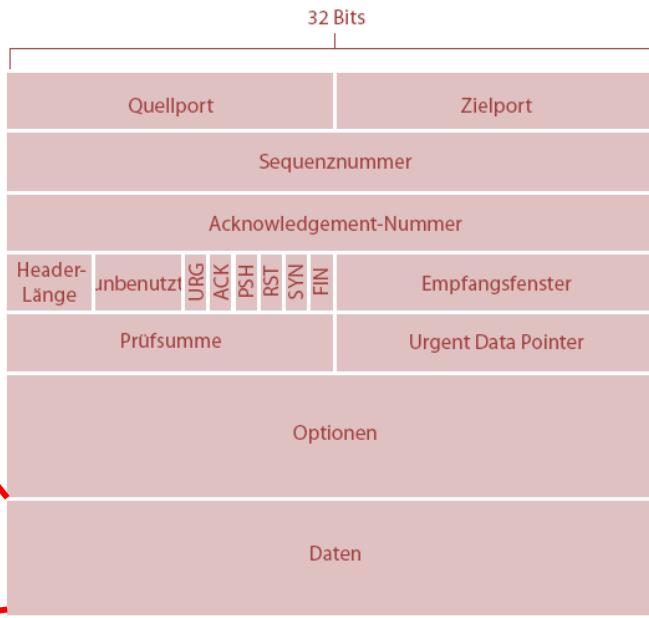
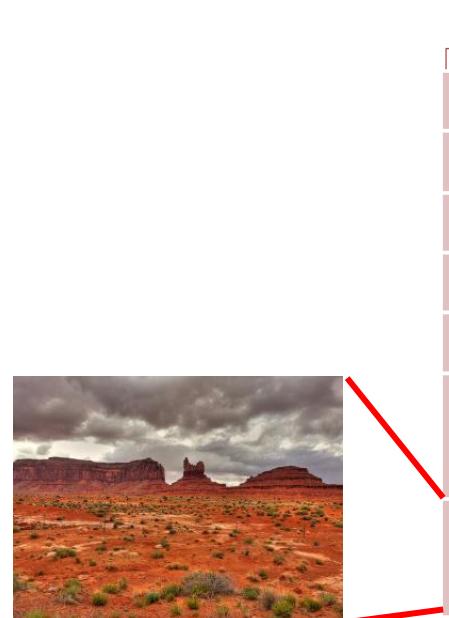
Netzwerkschicht: Overhead

- Wie groß ist der Overhead bei der Verwendung von TCP/IP?
 - Mindestens 20 Bytes für TCP
 - Mindestens 20 Bytes für IP

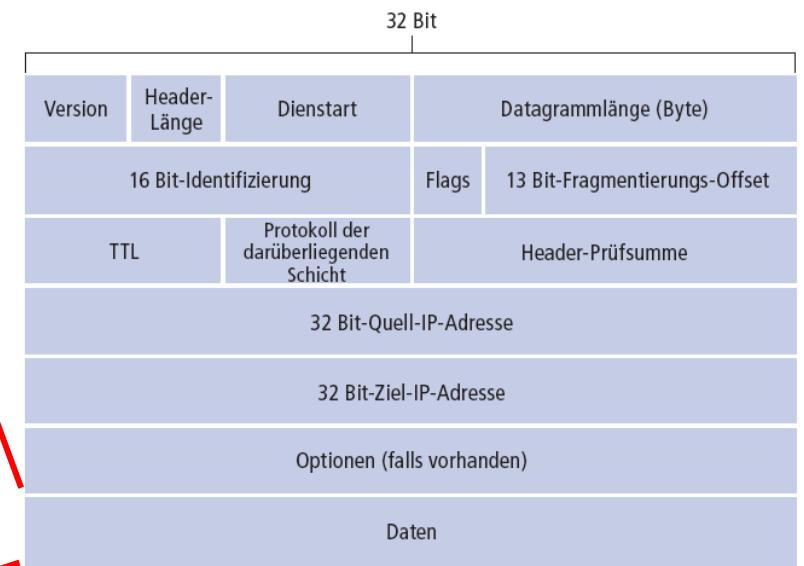
= mindestens 40 Bytes + eventuellen Overhead aus der Anwendungsschicht (bspw. SSL-Verschlüsselung, HTTP-Steuerbefehle etc.)

Applikation

TCP



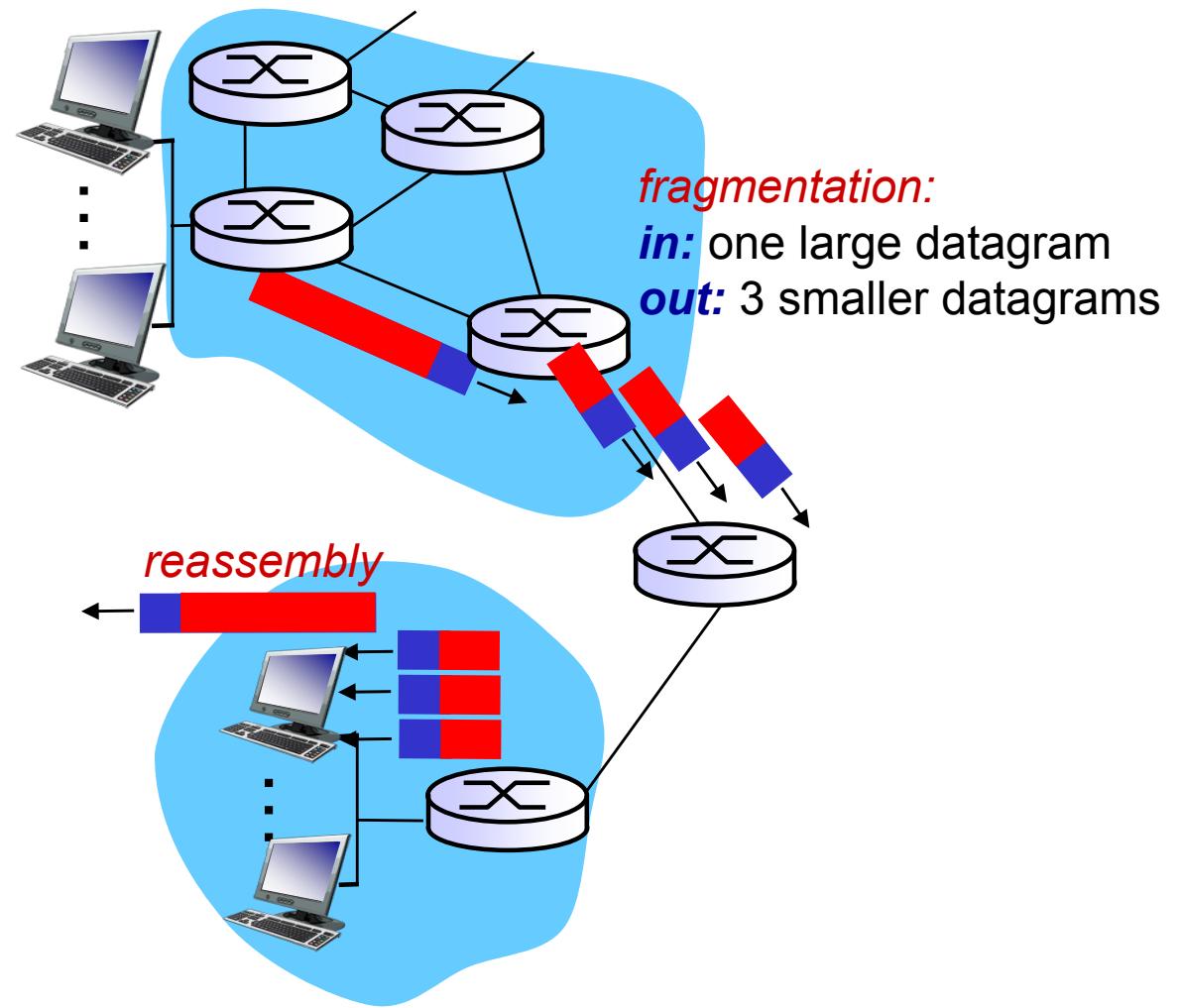
IP



Fragmentierung im Internet Protocol

Links haben Maximalgröße für Pakete

- Maximum Transmission Unit (MTU)
- Verschiedene Links haben unterschiedliche MTUs
 - Unterschiedliche Konfiguration / verwendete Technologien auf unteren Schichten
- IP-Datagramme müssen daher unter Umständen aufgeteilt werden
 - Aufteilung (Fragmentierung) erfolgt in den Routern
 - Zusammensetzen (Reassembly) beim Empfänger
 - IP-Header enthält die notwendigen Informationen hierzu



Fragmentierung im Internet Protocol

■ Beispiel

- 4000 Byte Datagramm
- MTU des nächsten Links = 1500 Byte
- IP-Header = 20 Byte

1480 Byte in Datenfeld

offset =
 $1480/8$

Wichtig: der Offset wird in 8 Byte-Blöcken angegeben

	length =4000	ID =x	fragflag =0	offset =0	...
--	-----------------	----------	----------------	--------------	-----

Ein großes Datagramm wird zu mehreren kleinen Datagrammen, die erst beim Empfänger wieder zusammengesetzt werden

	length =1500	ID =x	fragflag =1	offset =0	...
--	-----------------	----------	----------------	--------------	-----

	length =1500	ID =x	fragflag =1	offset =185	...
--	-----------------	----------	----------------	----------------	-----

	length =1040	ID =x	fragflag =0	offset =370	...
--	-----------------	----------	----------------	----------------	-----

Fragmentierung im Internet Protocol

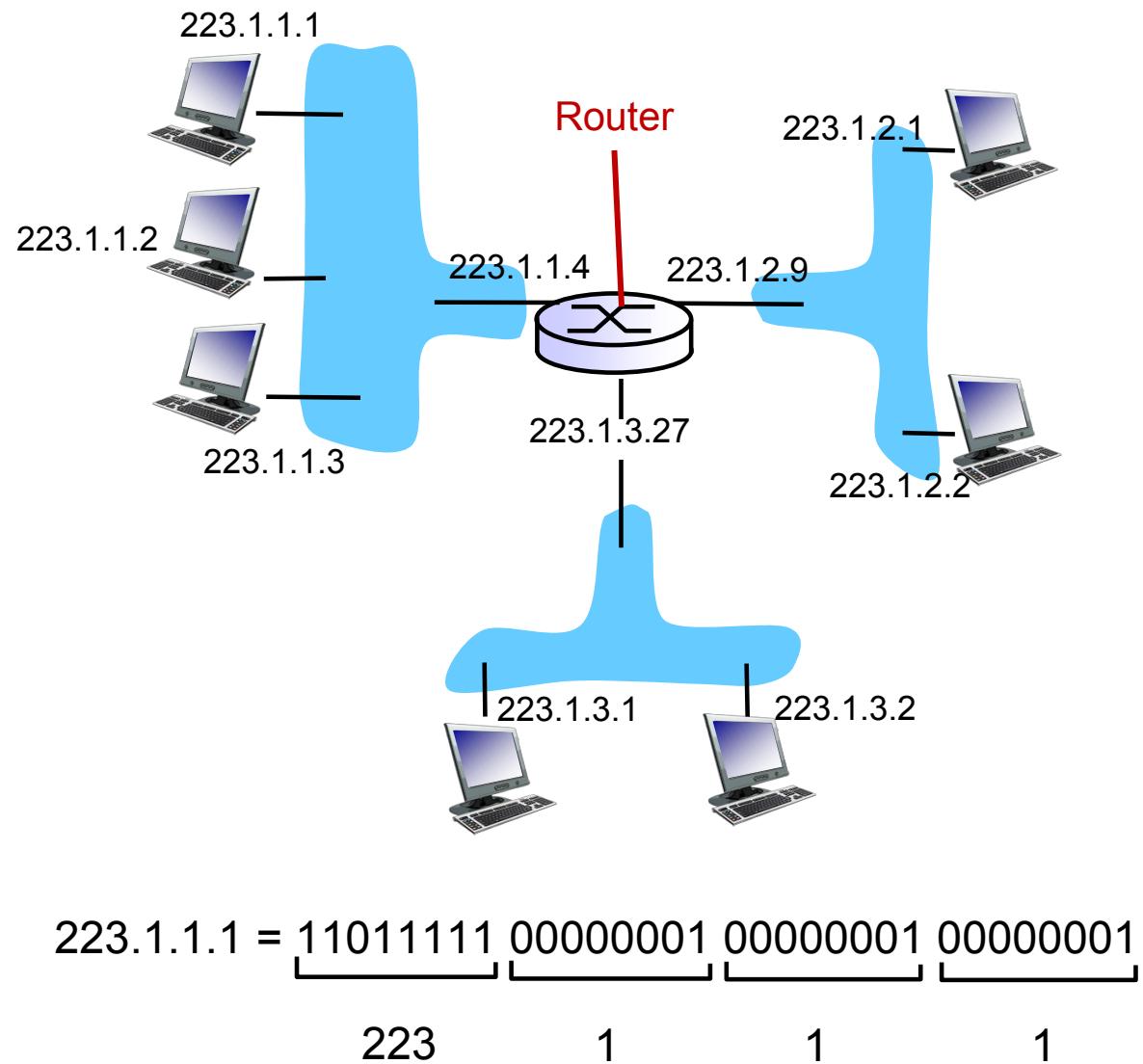
Fragment	Bytes	ID	Offset	Flag
1. Fragment	1.480 Byte im Datenfeld des IP-Datagramms	Identifizierung = 777	Offset = 0 (d.h., die Daten sollten beginnend bei Byte 0 eingefügt werden)	Flag = 1 (d.h., da kommt noch mehr)
2. Fragment	1.480 Datenbytes	Identifizierung = 777	Offset = 185 (d.h., die Daten sollten bei Byte 1.480 beginnend eingefügt werden; beachten Sie, dass $185 \cdot 8 = 1.480$)	Flag = 1 (d.h., da kommt noch mehr)
3. Fragment	1.020 Datenbytes $(= 3.980 - 1.480 - 1.480)$	Identifizierung = 777	Offset = 370 (d.h., die Daten sollten beginnend bei Byte 2.960 eingefügt werden; beachten Sie, dass $370 \cdot 8 = 2.960$)	Flag = 0 (d.h., es ist das letzte Fragment)

Fragmentierung im Internet Protocol

- Endsystem/Anwendung muss sich keine Gedanken über die Größe von MTUs verschiedener Links auf dem Weg vom Sender zum Empfänger machen
 - Entspricht dem Prinzip einer geschichteten Architektur
- Aber:
 - Aufwand in den Routern
 - Wenn ein Fragment verloren geht, ist das ganze Datagramm verloren
- Daher wird die Fragmentierung als problematisch eingeschätzt.
- Lösung: Bestimmen der kleinsten MTU des Weges (Path MTU)
 - Setze DF (Don't-Fragment-Bit) im Header des IP-Paketes
 - Wenn fragmentiert werden soll, wird das Paket verworfen und der Sender per ICMP benachrichtigt.
 - Sender wählt dann kleinere MTU
 - Wiederholen, bis akzeptable MTU gefunden wurde

Internet Protocol: Adressierung (IPv4)

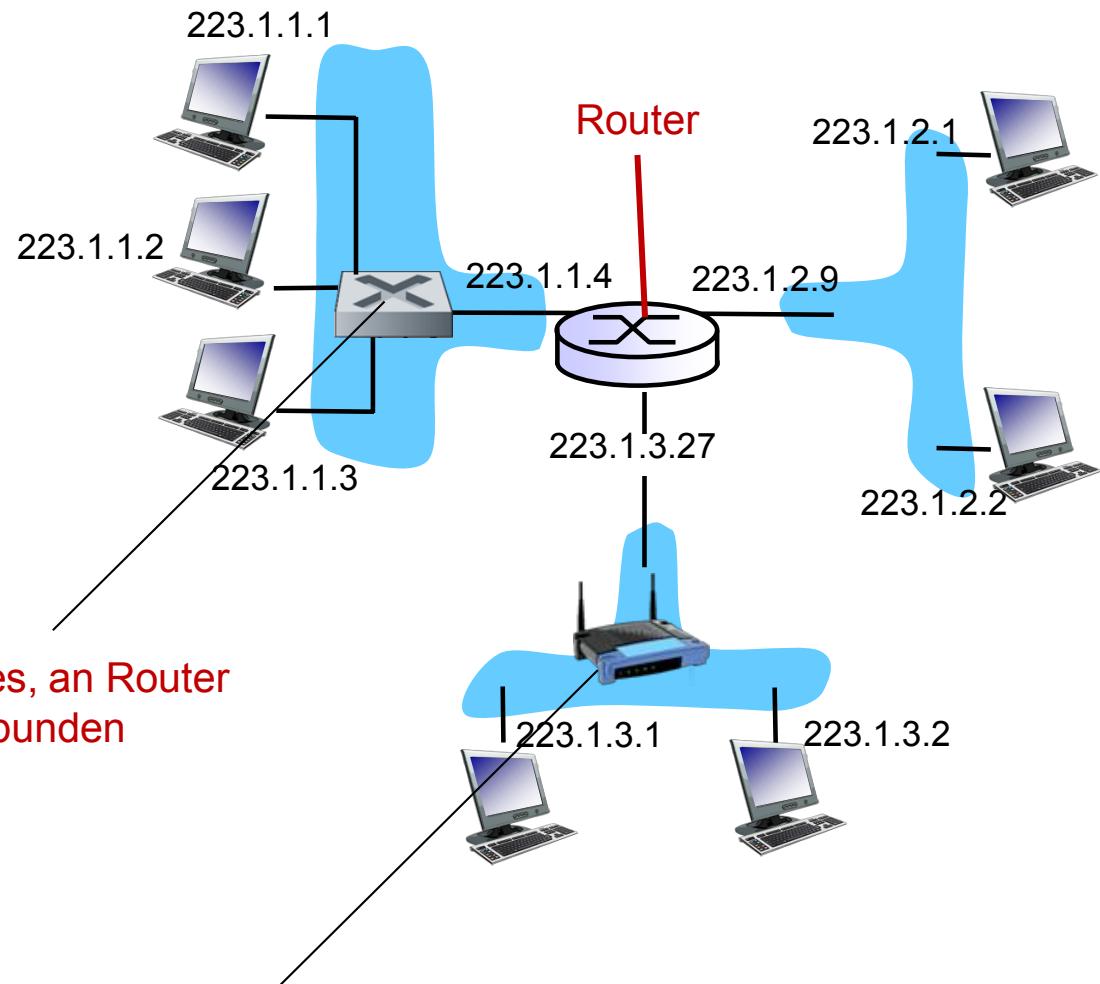
- IP-Adresse: 32-Bit-Kennung für das Interface (Schnittstelle) eines Endsystems oder eines Routers
- Interface: Verbindung zwischen dem System und dem Link
 - Wird normalerweise durch eine Netzwerkkarte bereitgestellt
 - Router haben typischerweise mehrere Interfaces
 - Endsysteme können ebenfalls mehrere Interfaces haben
 - Jedes Interface besitzt eine IP-Adresse



Internet Protocol: Adressierung (IPv4)

- Wie sind die Interfaces miteinander verbunden?

- Dies haben wir in Vorlesungen 1-3 diskutiert.
- Für IP nicht relevant (Schichtenarchitektur!)
- Hier ein paar Beispiele:

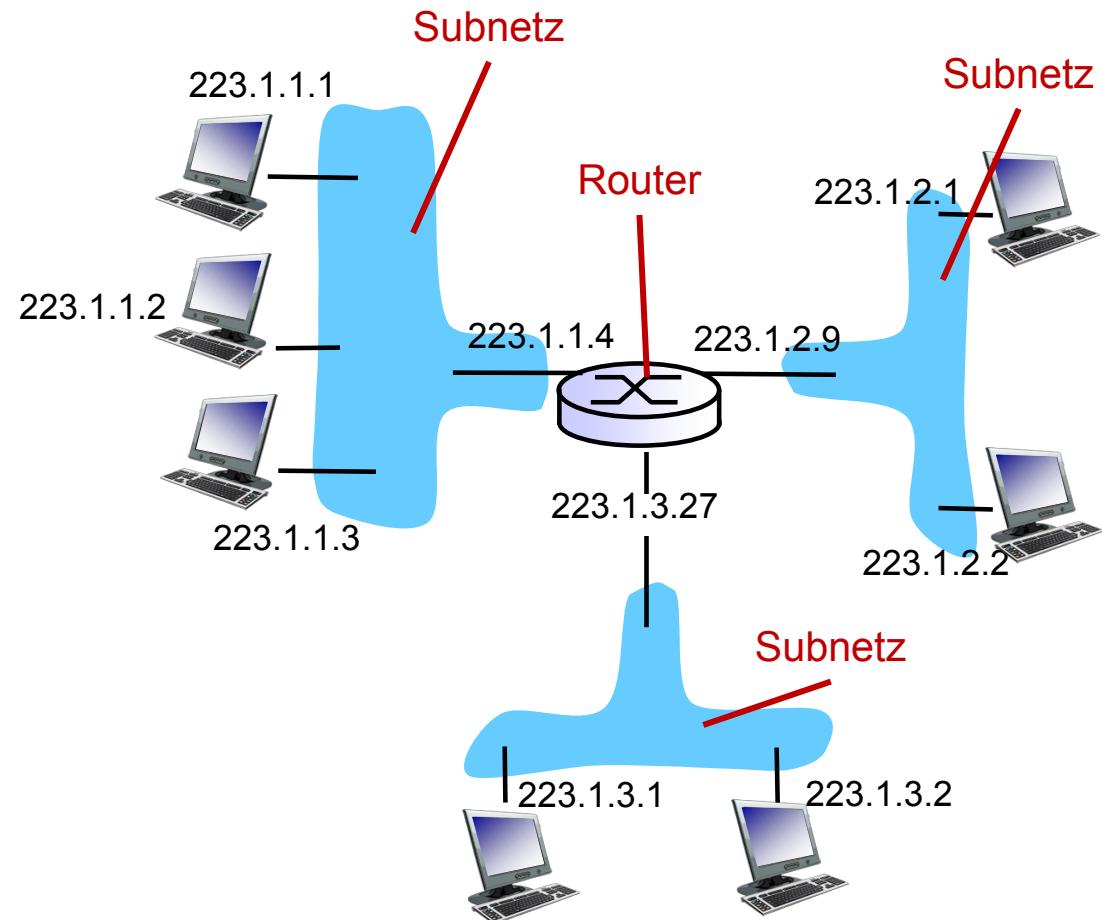


Kabelgebundene Ethernet Interfaces, an Router mittels eines Ethernet Switch angebunden

Kabellose WiFi Interfaces, mittels WiFi-Basis mit dem Router oder einem weiteren Switch verbunden.

Internet Protocol: Adressierung (IPv4)

- IP-Adressen haben zwei Bestandteile:
 - **netid**: die oberen Bits der Adresse, identifiziert ein Netzwerk
 - **hostid**: die unteren Bits der Adresse, identifiziert ein Interface eines Systems
- Was ist ein (Sub-)Netzwerk?
 - Alle Interfaces mit derselben netid formen ein Netzwerk
 - Alle Interfaces eines Netzwerkes können sich direkt (**ohne einen Router zu durchqueren**) erreichen

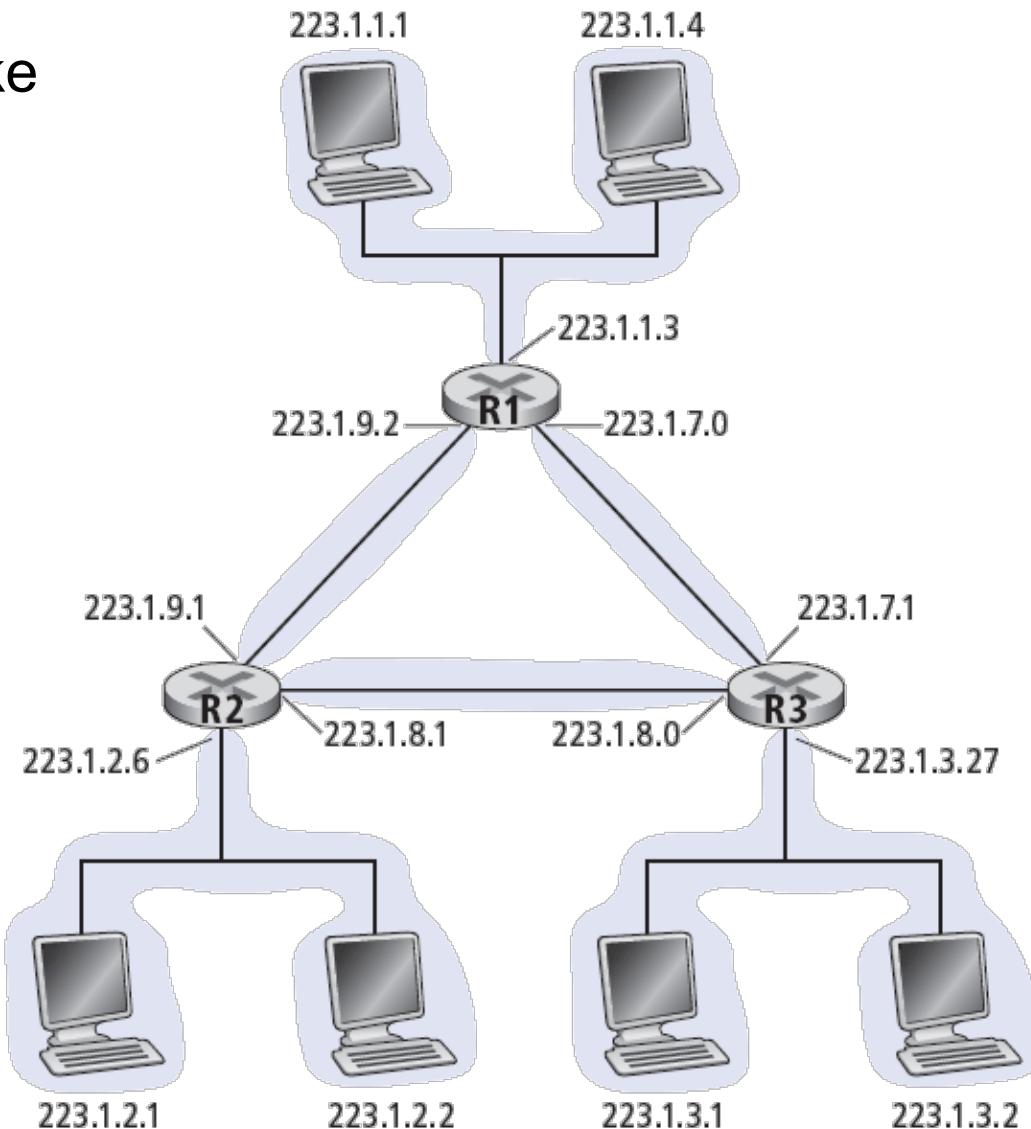


Drei IP-Netzwerke, die mit einem Router verbunden sind. Die netid steht hier in den oberen 24 Bit.

Subnetzmaske: /24 oder 255.255.255.0

Internet Protocol: Adressierung (IPv4)

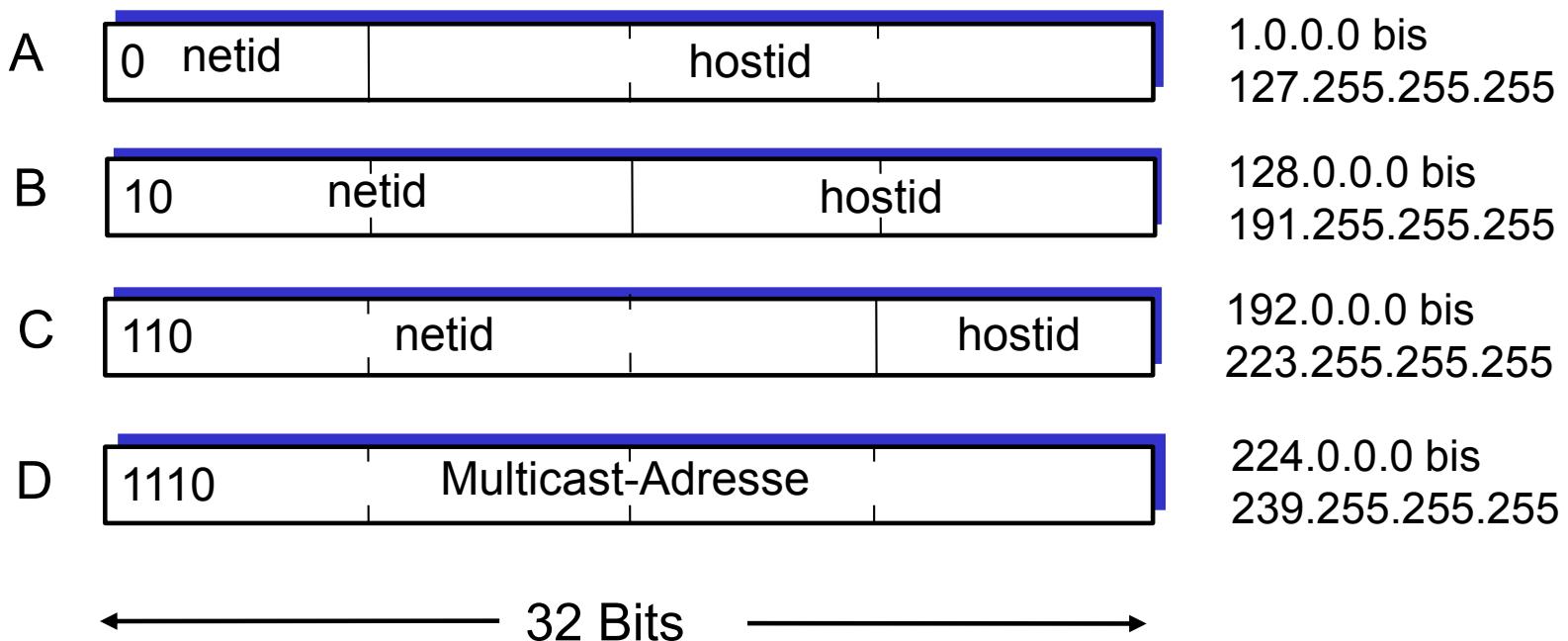
- Wie viele Subnetzwerke sehen Sie?



IP-Adressierung mit Adressklassen („früher“)

- Früher wurden IP-Adressen in Adressklassen aufgeteilt
- Die Klasse bestimmte das Verhältnis der Längen netid/hostid
- Dies nennt man „classfull“ addressing oder auch klassenbasierte Adressierung

Klasse



Klassenlose IP-Adressierung („heute“)

- Klasse-A- und -B-Adressen haben Platz für mehr Endsysteme, als man in einem Netzwerk sinnvoll unterbringen kann
 - Verschwendungen von IP-Adressen: Ein Unternehmen mit 2.500 Rechnern, die eine IP-Adresse bekommen sollen, verschwendet bei einem Klasse B-Netzwerk ca. 62.500 Adressen.
 - Klasse C-Netzwerke sind aber oftmals zu klein für ein Unternehmen.
 - Das Unternehmen mit 2.500 Rechnern könnte 10 Klasse C-Netzwerke beantragen und selbst mit „internen“ Routern und dem Internet verbinden.
 - Problem: alle Routing-Tabellen im Internet müssen diese 10 Netzwerke auch aufnehmen, um Pakete weiterleiten zu können.
 - Dadurch wurden die Routing-Tabellen sehr schnell sehr groß (ein Eintrag pro netid ist notwendig) und mussten häufig angepasst werden.
- 1993 wurde von der Internet Engineering Task Force (IETF) die klassenlose Adressierung eingeführt, um diese Probleme zu lösen.

Klassenlose IP-Adressierung („heute“)

■ Classless InterDomain Routing (CIDR)

- Der Subnetz-Anteil einer IP-Adresse kann eine beliebige Länge aufweisen.
- Adress-Format: $a.b.c.d/x$, mit x als die Anzahl von Bits, die das Subnetz in der Adresse einnimmt.



- Dies ist eine alternative (und kürzere) Schreibweise zu der Subnetzmaske, die Sie bereits im Cisco Packet Tracer bei der statischen Adressierung von Hosts verwendet haben.

11111111 11111111 11111110 00000000

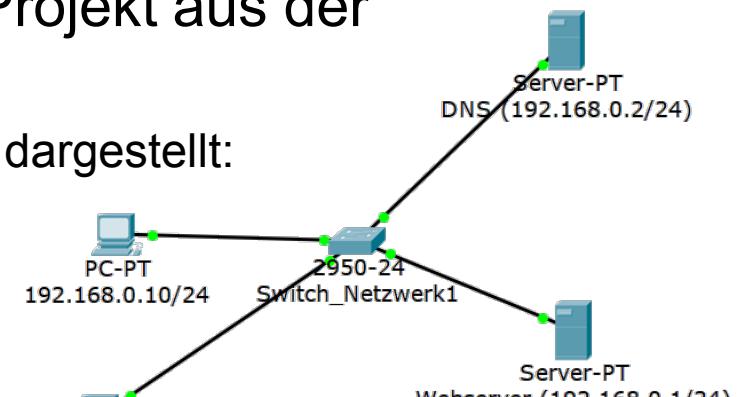
Welche ist die erste IP-
Adresse in diesem Netzwerk?
Welche die letzte?

Subnetzmaske, die Sie eingeben würden: 255.255.254.0

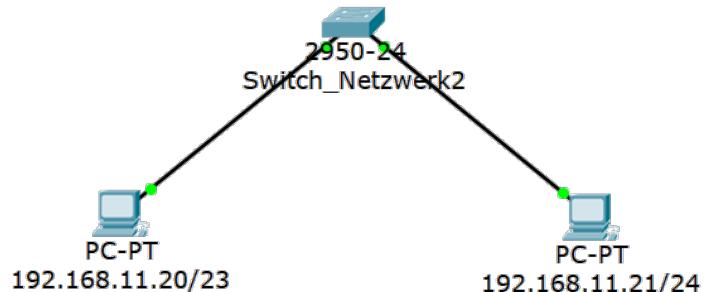
Übung

- Rufen Sie das letzte Cisco Packet Tracer-Projekt aus der Transportschicht auf.

- Benennen Sie die Hosts und den Switch wie hier dargestellt:



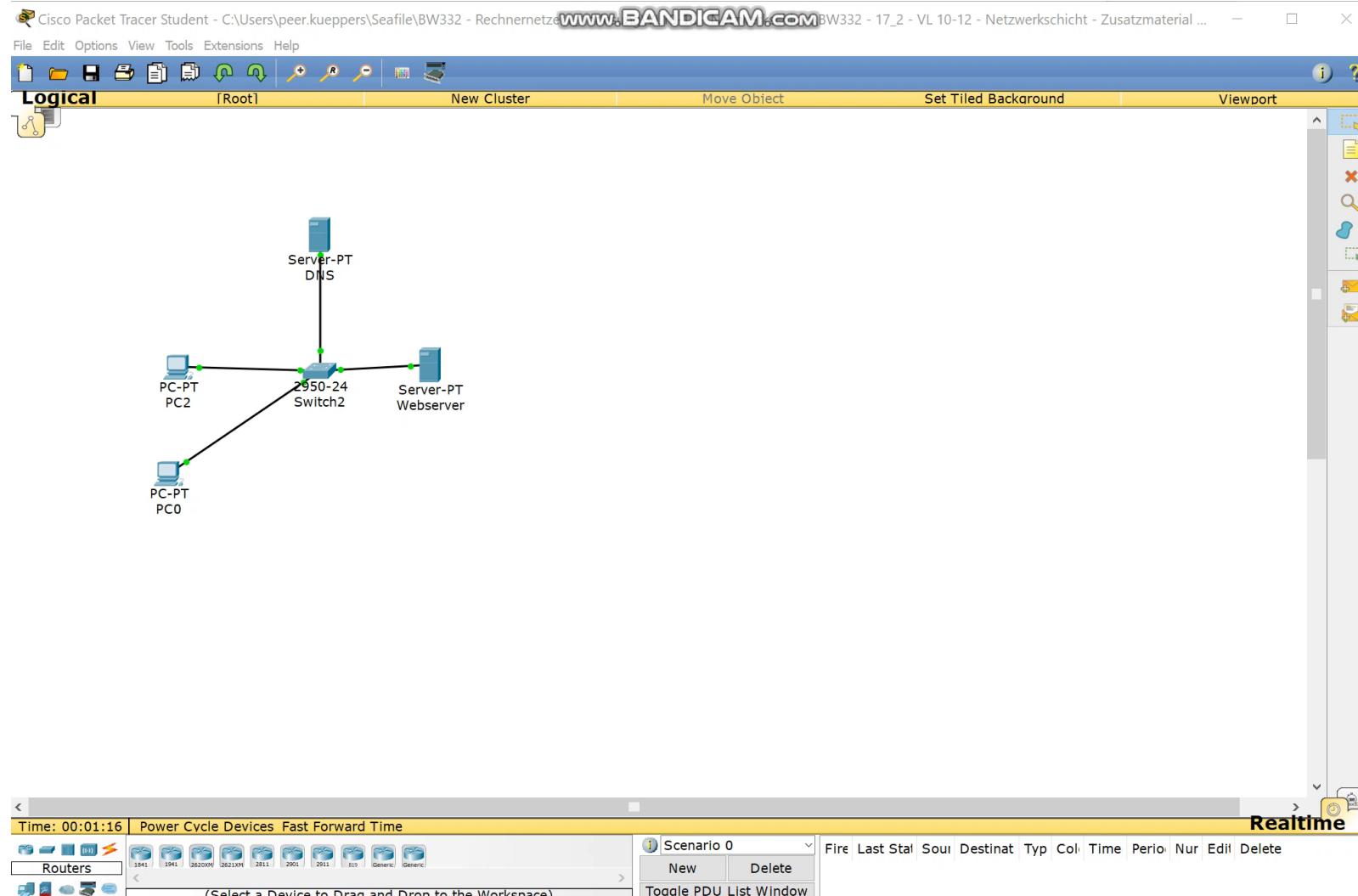
- Legen Sie ein zweites Netzwerk mit 2 Hosts und den folgenden IP-Adressen an.



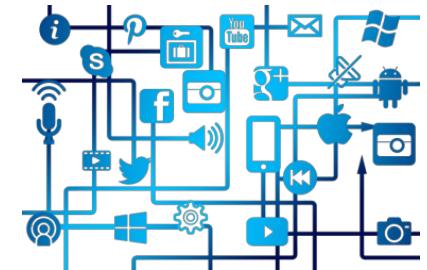
- Achten Sie auf die Subnetz-Unterschiede
 - Wie müssen die Subnetzmasken der beiden PCs lauten?
 - Versuchen Sie mit einem ping von 192.168.11.20/23 den anderen PC zu erreichen.
 - Was müssen Sie an 192.168.11.21/24 ändern, damit die PCs via IP kommunizieren können?

Übung

■ Lösungsvorschlag:



Begleitmaterialien



- Video zu CIDR

<https://www.youtube.com/watch?v=Q1U9wVXRuHA&t=2s>

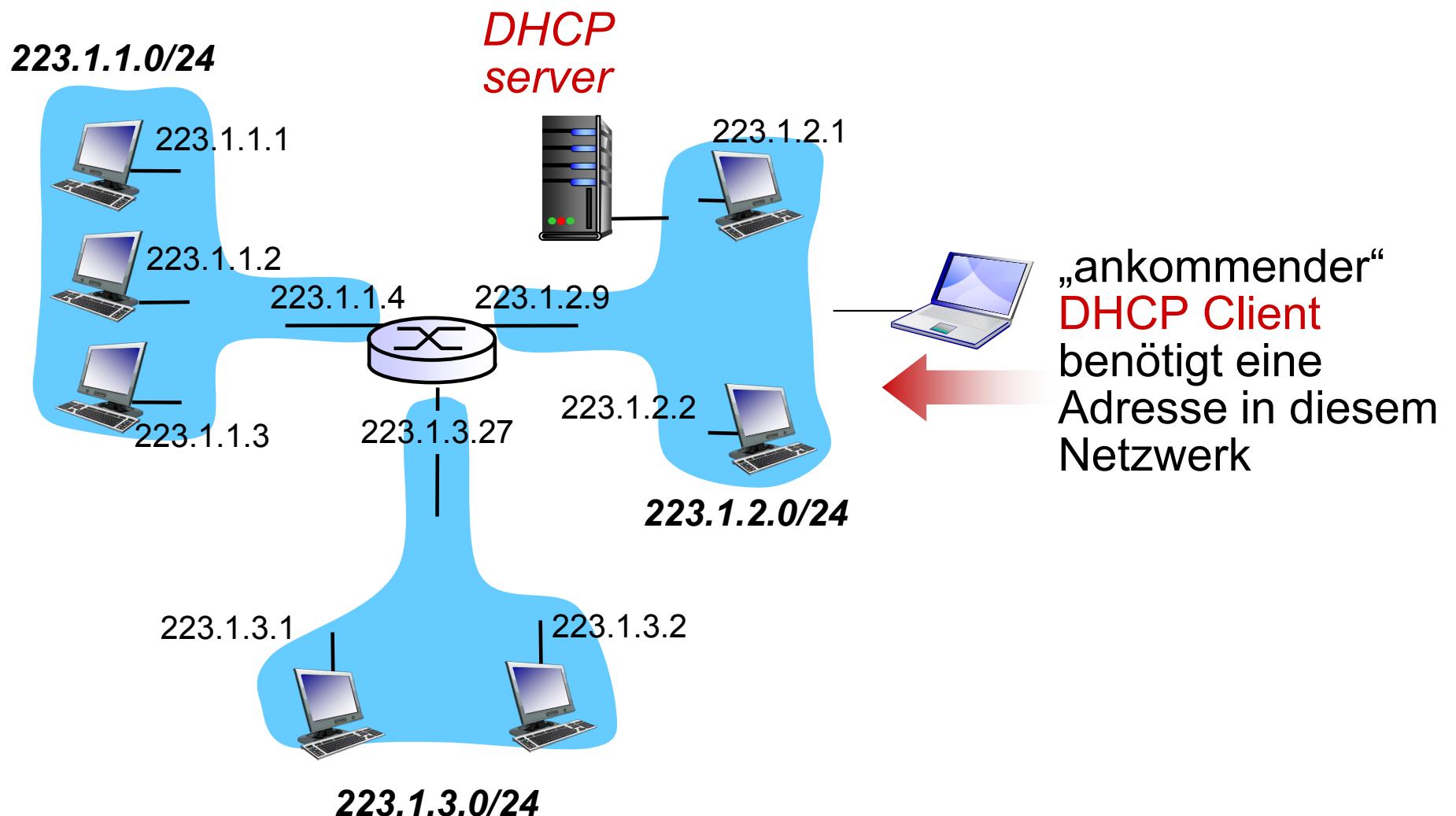
Adressvergabe für Hosts

- Problem: Wie bekommen Endsysteme ihre IP-Adresse (und andere Parameter der Netzwerkschicht, z.B. die Subnetzmaske)?
- Möglichkeit 1: durch manuelle Konfiguration
 - IP-Adresse
 - Subnetzmaske
 - Evtl. weitere Parameter
 - → sehr hoher Administrationsaufwand und fehleranfällig
- Möglichkeit 2: DHCP - Dynamic Host Configuration Protocol
 - Dynamisches Beziehen der Adresse und weiterer Parameter von einem Server
 - „Plug-and-Play“

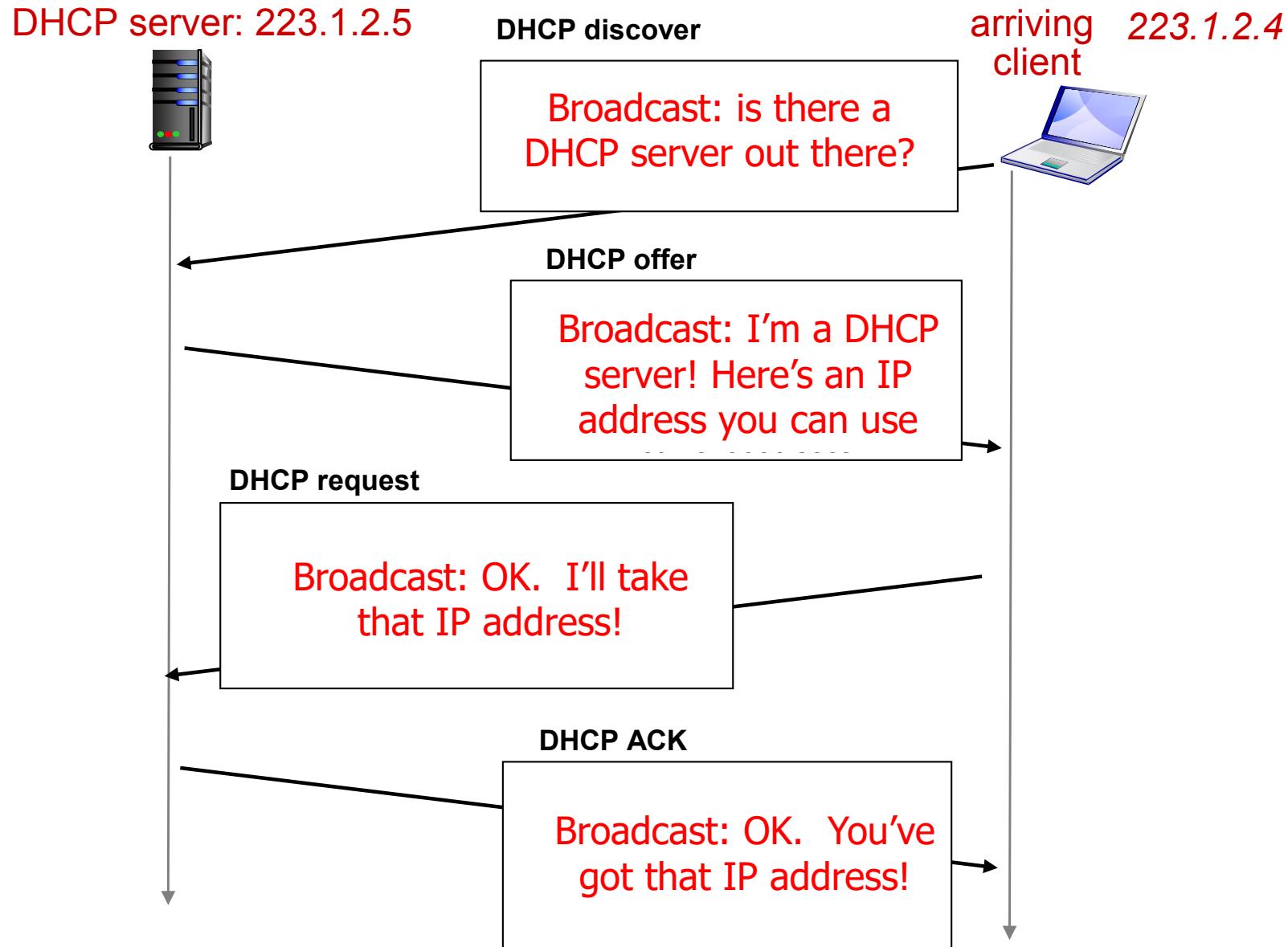
DHCP-Protokoll

- Ziele von DHCP
 - Automatische Vergabe von Adressen und Parametern
 - Keine Konfiguration der Endsysteme notwendig
 - Unterstützung von mobilen Benutzern
- Prinzipieller Ablauf des Protokolls
 - Host schickt eine DHCP-Discover-Nachricht per IP-Broadcast (optional)
 - Broadcast bedeutet, dass an alle erreichbaren Hosts im lokalen Netzwerk eine Nachricht gesendet wird
 - DHCP-Server antwortet mit einer DHCP-Offer-Nachricht (optional)
 - Host beantragt eine IP-Adresse: DHCP-Request-Nachricht
 - DHCP-Server vergibt Adresse: DHCP-Ack-Nachricht

DHCP-Protokoll



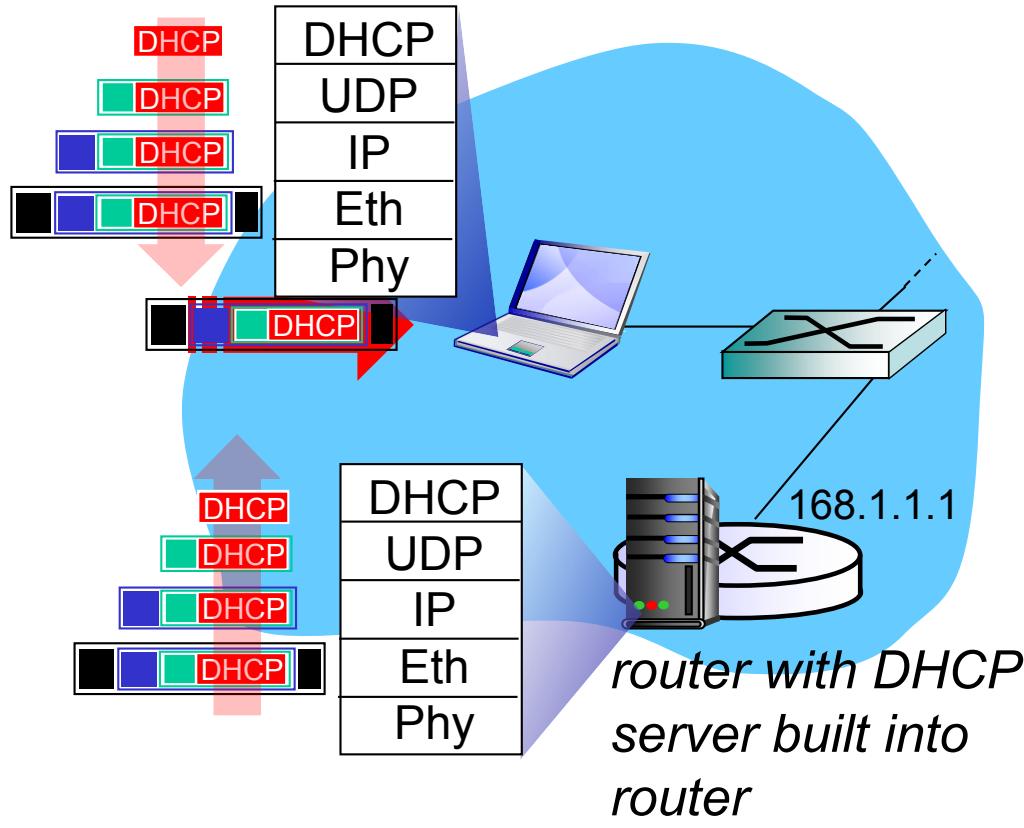
DHCP-Protokoll



DHCP-Protokoll

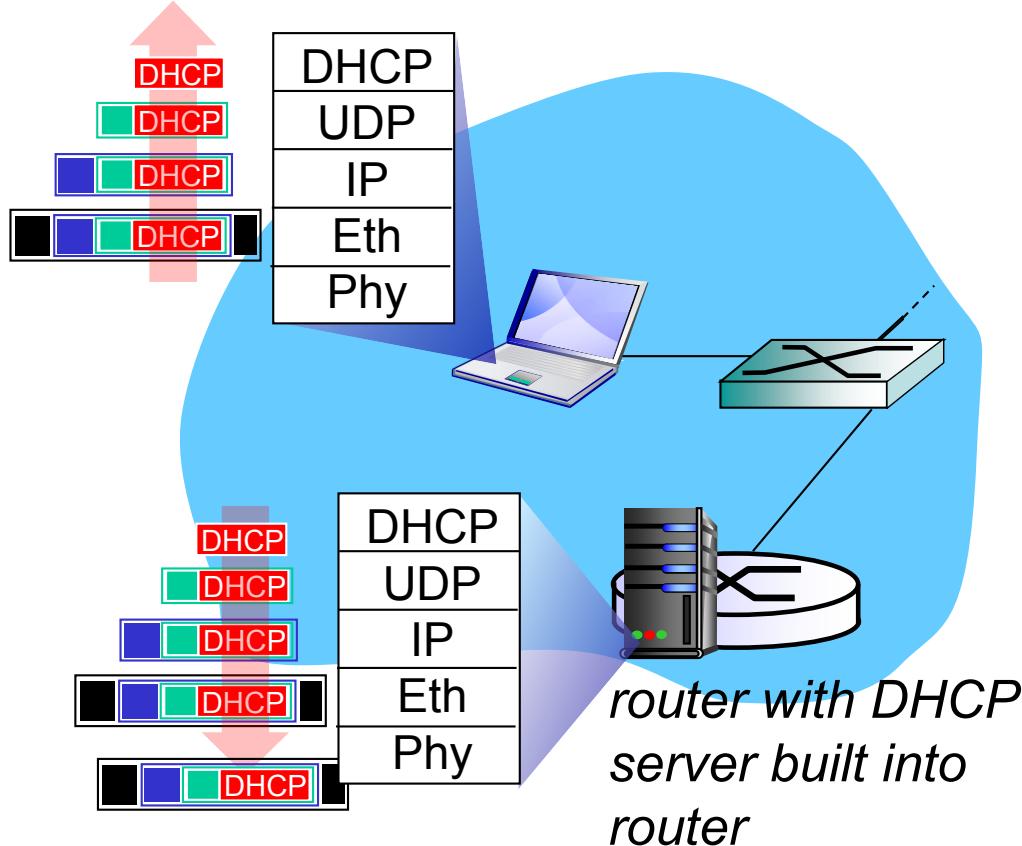
- DHCP kann mehr als nur die zugewiesene IP-Adresse liefern:
 - Adresse des ersten Routers (first-hop router) für den Host (*damit er ins Internet kommt*)
 - Name and IP-Adresse eines DNS-Servers (*damit domains aufgelöst werden können*)
 - Die Netzwerkmaske, damit der Host Subnetz- und Host-Anteile bei IP-Adressen trennen kann.

DHCP-Beispiel



- Der Laptop benötigt eine IP-Adresse und die Adressen des first-hop Routers und DNS-Servers: nutze DHCP
- DHCP Request gekapselt in UDP, dies gekapselt in IP, und dies wiederum gekapselt in 802.1 Ethernet (untere Schicht)
- Ethernet Frame wird mittels Broadcast (Zieladresse: FFFFFFFFFFFF) an alle Systeme im LAN geschickt und dabei vom Router mit dem DHCP server empfangen
- Ethernet demultiplexed zu IP, das demultiplexed zu UDP und das demultiplexed zu DHCP-Nachricht

DHCP-Beispiel



- DHCP-Server erstellt ein DHCP ACK mit der Client IP-Adresse, IP-Adresse des first-hop Routers, sowie Name & IP Adresse des DNS-Servers
- Kapselung der DHCP-Nachricht und demultiplexing hoch bis zur DHCP-Schicht des Laptops (DHCP-Client)
- Der Client kennt nun seine IP-Adresse und alle weiteren notwendigen Parameter.
- Er kommt insbesondere ins Internet.

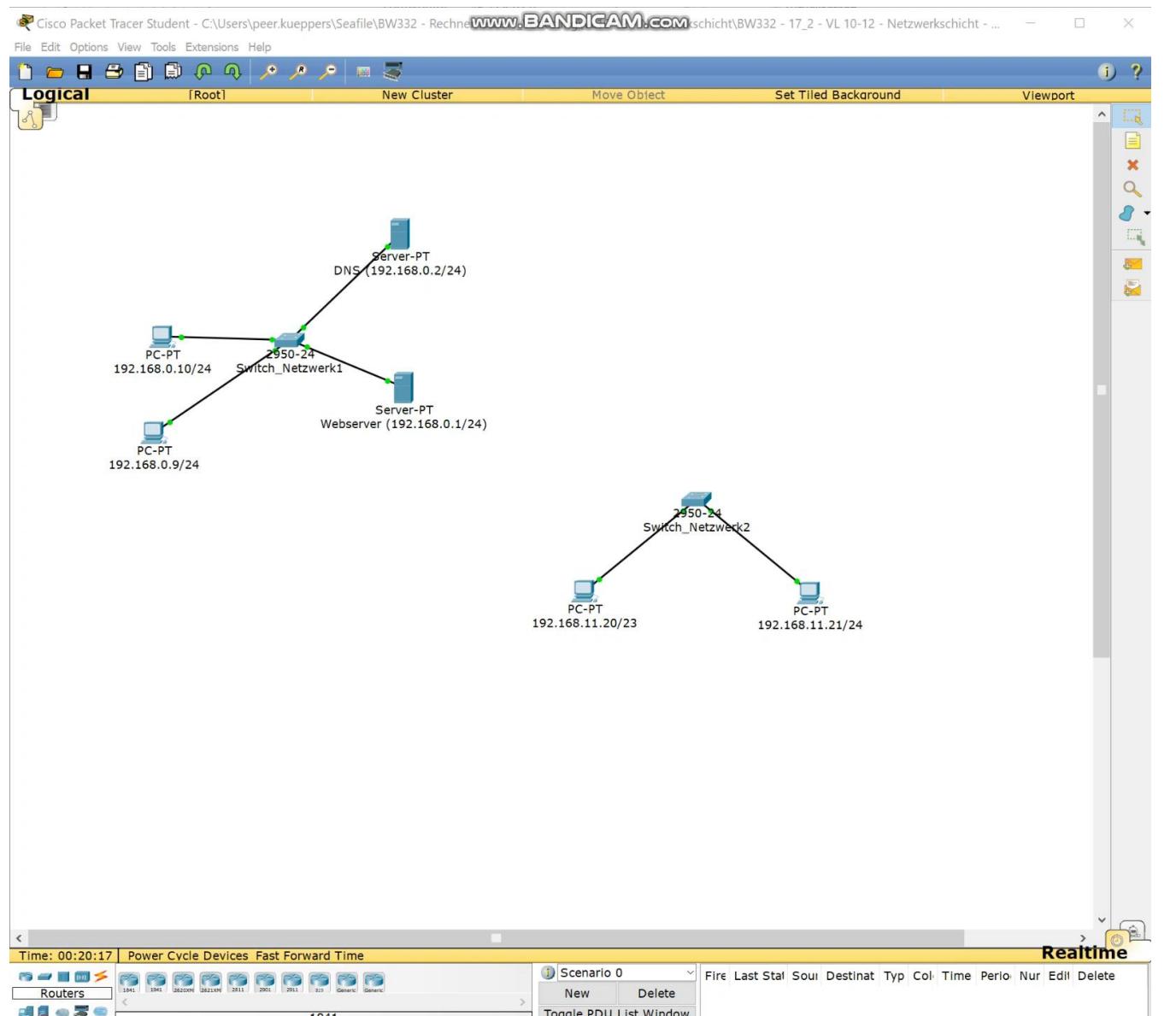


Übung

- Rufen Sie das Cisco Packet Tracer-Projekt aus Übung 6.1 auf.
 - Wir möchten nun in beiden Netzwerken DHCP verwenden.
 - In Netzwerk 1 soll der DNS-Server auch DHCP-Server werden.
 - Dieser behält also seine statische IP-Adresse.
 - Achten Sie auf die Subnetzmaske und nutzen Sie den maximal möglichen Umfang der IP-Range. Die maximale Anzahl Nutzer können Sie auf dem Standard lassen.
 - Den first hop-Router (Default Gateway) müssen Sie noch nicht einstellen.
 - Den „Pool Name“ können Sie auf dem Standard belassen. Markieren Sie in der Liste unten den existierenden „serverPool“ und drücken Sie dann auf „Save“.
 - Alle anderen Hosts sollen per DHCP ihre IP-Adresse zugewiesen bekommen.
 - Führen Sie beim Umstellen von „static“ auf „DHCP“ bei einem der Hosts eine Simulation durch, um die Adressvergabe zu verfolgen. Protokoll: UDP, Achten Sie Broadcast-Nachrichten!
 - In Netzwerk 2 wird ein neuer Server für DHCP benötigt.
 - Bitte fügen Sie diesen hinzu und konfigurieren Sie ihn (Hinweise siehe oben; achten Sie auf die Subnetzmaske).
 - Die beiden Hosts sollen als DHCP-Clients sich ihre IP-Adressen nun automatisch holen.
 - Führen Sie auch hier eine beispielhafte Simulation durch.

Übung

■ Lösungsvorschlag:

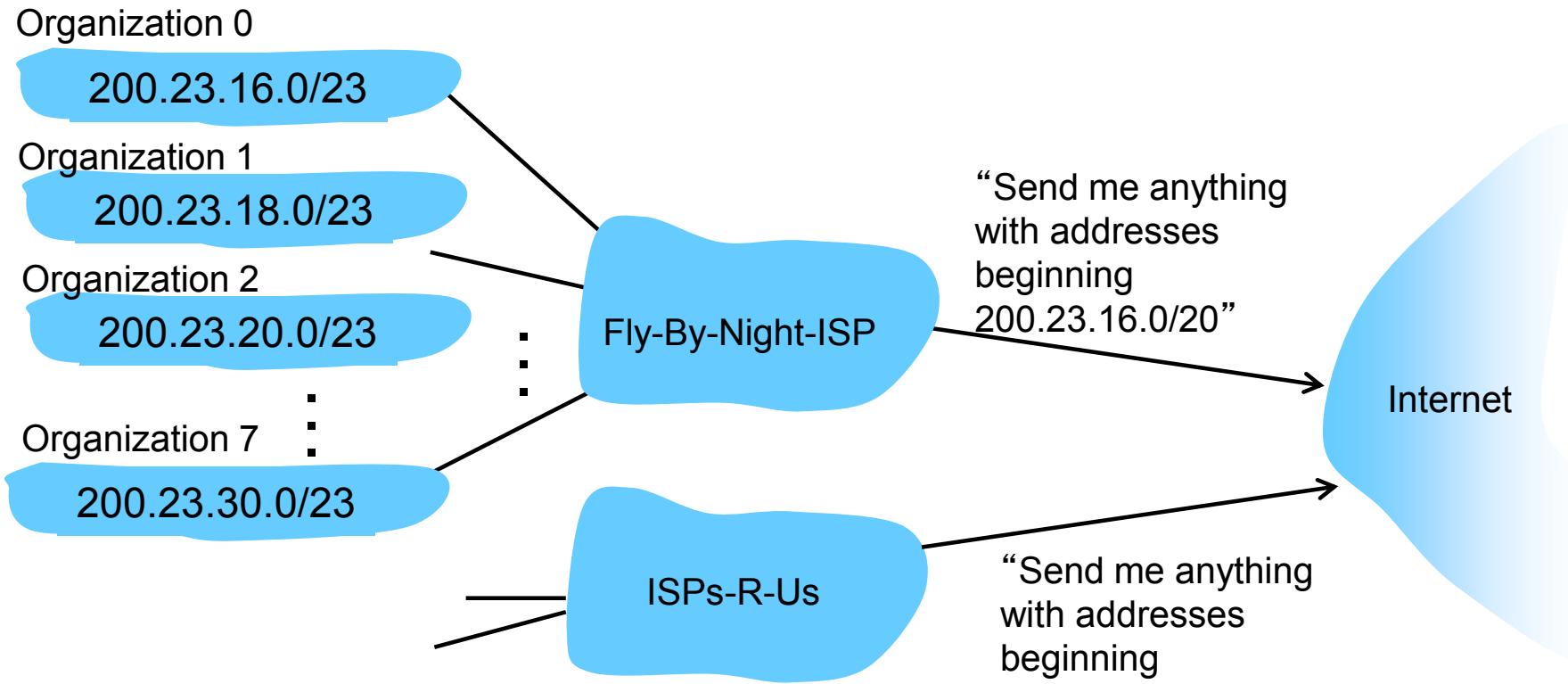


Adressvergabe

- Bei der klassenlosen Adressierung werden zusammenhängende Adressbereiche von der Internet Assigned Numbers Authority (IANA) an die Regional Internet Registries (RIR) vergeben
 - APNIC (Asia Pacific Network Information Centre) - Asien/Pazifik
 - ARIN (American Registry for Internet Numbers) - Nordamerika und Afrika südlich der Sahara
 - LACNIC (Regional Latin-American and Caribbean IP Address Registry) – Lateinamerika und einige karibische Inseln
 - RIPE NCC (Réseaux IP Européens) - Europa, Mittlerer Osten, Zentralasien und afrikanische Länder nördlich des Äquators
- Diese vergeben zusammenhängende Adressbereiche an ISPs
- ISPs vergeben zusammenhängende Adressbereiche an ihre Kunden
- Dadurch: Aggregation in Routing-Tabellen möglich
- Classless Interdomain Routing (CIDR) verwendet dies

Effizientes Routing durch Aggregation

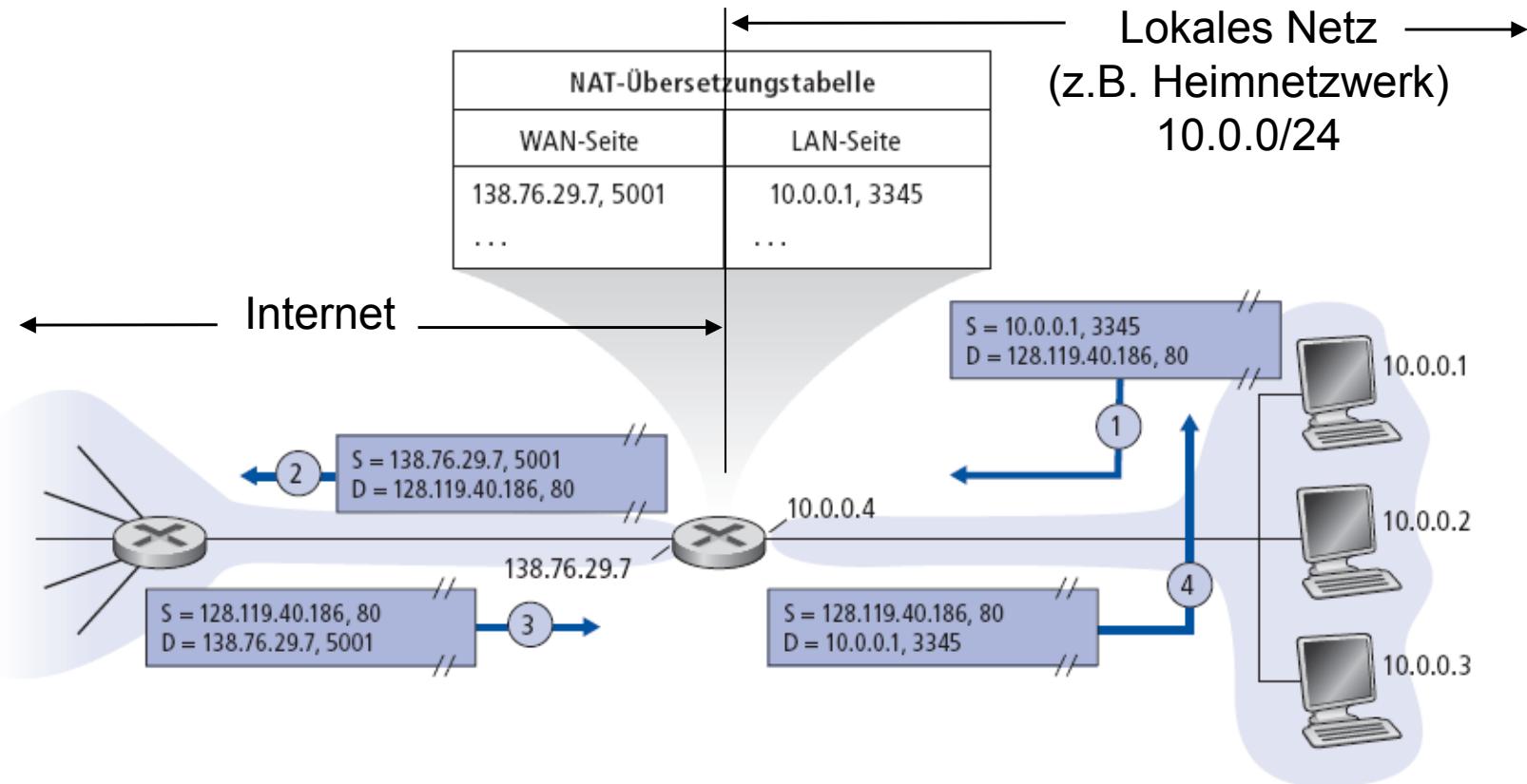
ISP's block	<u>11001000 00010111 00010000 00000000</u>	200.23.16.0/20
Organization 0	<u>11001000 00010111 00010000 00000000</u>	200.23.16.0/23
Organization 1	<u>11001000 00010111 00010010 00000000</u>	200.23.18.0/23
Organization 2	<u>11001000 00010111 00010100 00000000</u>	200.23.20.0/23
...
Organization 7	<u>11001000 00010111 00011110 00000000</u>	200.23.30.0/23



Network Address Translation (NAT)

- Motivation:
 - Häufig hat man nur eine IP-Adresse, aber mehrere Endsysteme (eigentlich jedes WLAN in Heimnetzwerken)
 - Diese ist meist nur temporär (per DHCP) zugewiesen
 - Man möchte bei einem Provider-Wechsel nicht die IP-Adressen der Endsysteme verändern
 - IP-Adressen im eigenen Netzwerk sollen aus Sicherheitsgründen nicht vom Internet aus sichtbar sein
 - Interne IP-Adressen sollen veränderbar sein, ohne dass der Rest des Internets darüber informiert werden muss
- Idee:
 - Vergebe lokale (weltweit nicht eindeutige) Adressen an die Systeme im eigenen Netzwerk
 - Router zur Anbindung an das Internet übersetzt diese Adressen in eine gültige, weltweit eindeutige IP-Adresse
 - Dazu wird die Adressierung auf der Transportschicht gebraucht („missbraucht“): Ports

Network Address Translation (NAT)



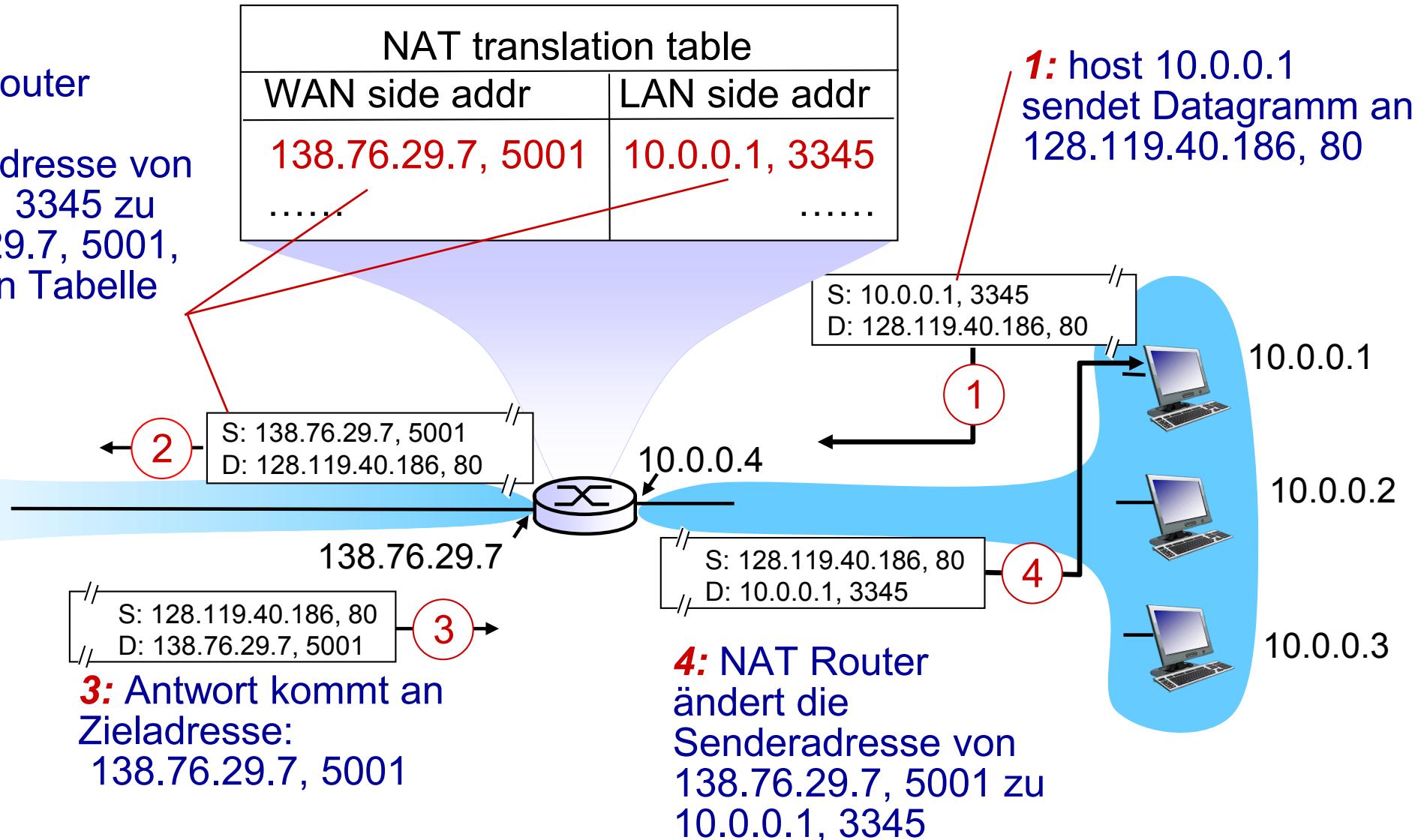
Alle Datagramme, die das lokale Netz **verlassen**, haben die **gleiche** NAT-IP-Adresse als Absender: 138.76.29.7;
Unterscheidung der einzelnen Hosts im LAN nur durch Portnummern

Network Address Translation (NAT)

- Implementierung: Ein NAT-Router muss Folgendes tun:
 - Ausgehende Datagramme: ersetze (Sender-IP-Adresse, Portnummer) im Absenderfeld für jedes ins Internet geleitete Datagramm durch (NAT-IP-Adresse, neue Portnummer)
 - Kommunikationspartner wird die Antworten an (NAT-IP-Adresse, neue Portnummer) schicken
 - Speichere in einer NAT-Tabelle die Abbildung zwischen (Sender-IP-Adresse, Portnummer) und (NAT-IP-Adresse, neue Portnummer)
 - Ankommende Datagramme: ersetze (NAT-IP-Adresse, neue Portnummer) im Empfängerfeld durch (Sender-IP-Adresse, Portnummer) aus der NAT-Tabelle

Network Address Translation (NAT)

2: NAT router
ändert
Senderadresse von
10.0.0.1, 3345 zu
138.76.29.7, 5001,
Eintrag in Tabelle

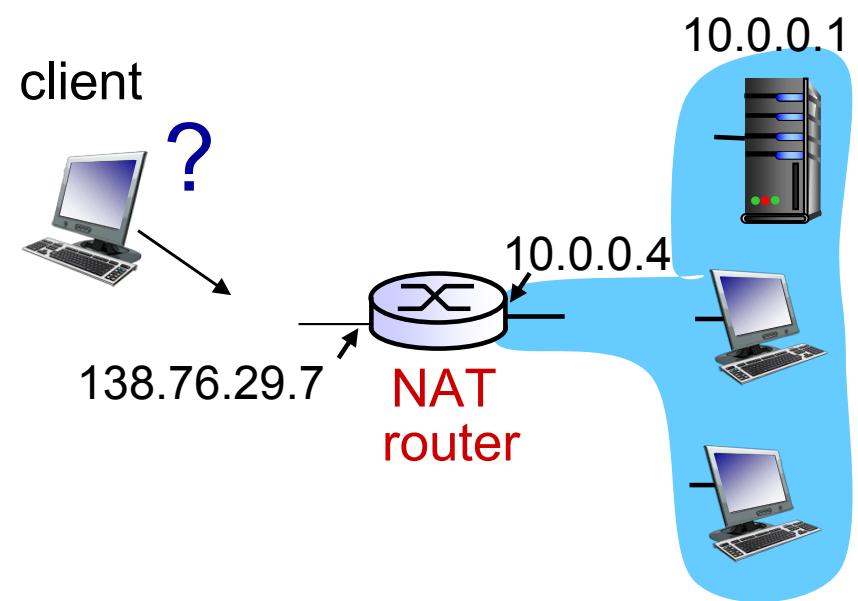


Network Address Translation (NAT)

- Transportschicht: 16-Bit „Port Number“-Feld ermöglicht
 - mehr als 60.000 gleichzeitige Verbindungen mit einer IP-Adresse
- NAT ist nicht unumstritten
 - Router sollten eigentlich nur Informationen bis Schicht 3 verwenden
 - Verletzung des sogenannten Ende-zu-Ende-Prinzips (end-to-end principle):
 - Transparente Kommunikation von Endsystem zu Endsystem, im Inneren des Netzes wird nicht an den Daten „herumgepfuscht“
 - Bei NAT: Der Anwendungsentwickler muss die Präsenz von NAT-Routern berücksichtigen.
- NAT dient hauptsächlich der Bekämpfung der Adressknappheit im Internet.
 - Dies sollte besser über IPv6 erfolgen.

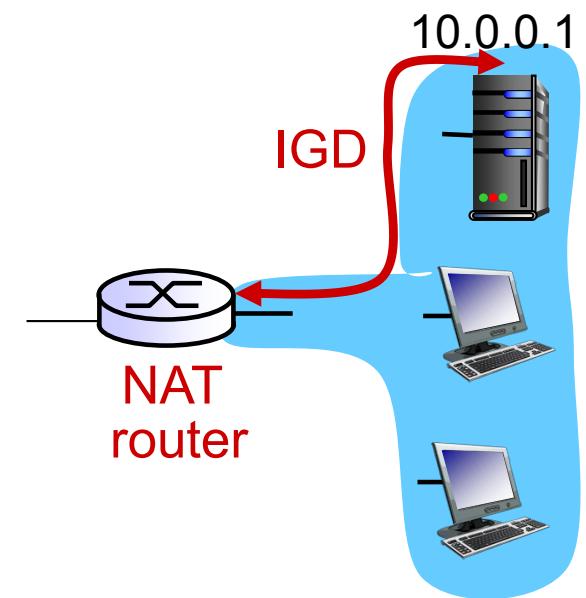
Durchqueren von NAT

- NAT traversal
- Der Client möchte den Server mit der Adresse 10.0.0.1 kontaktieren
 - Die Adresse 10.0.0.1 ist eine lokale Adresse und kann nicht als Adresse im globalen Internet verwendet werden
 - Die einzige nach außen sichtbare Adresse ist:
138.76.29.7
- Lösung 1: Statische Konfiguration von NAT, so dass eingehende Anfragen geeignet weitergeleitet werden
 - Beispiel: (123.76.29.7, Port 2500) wird immer an 10.0.0.1, Port 25000 weitergeleitet



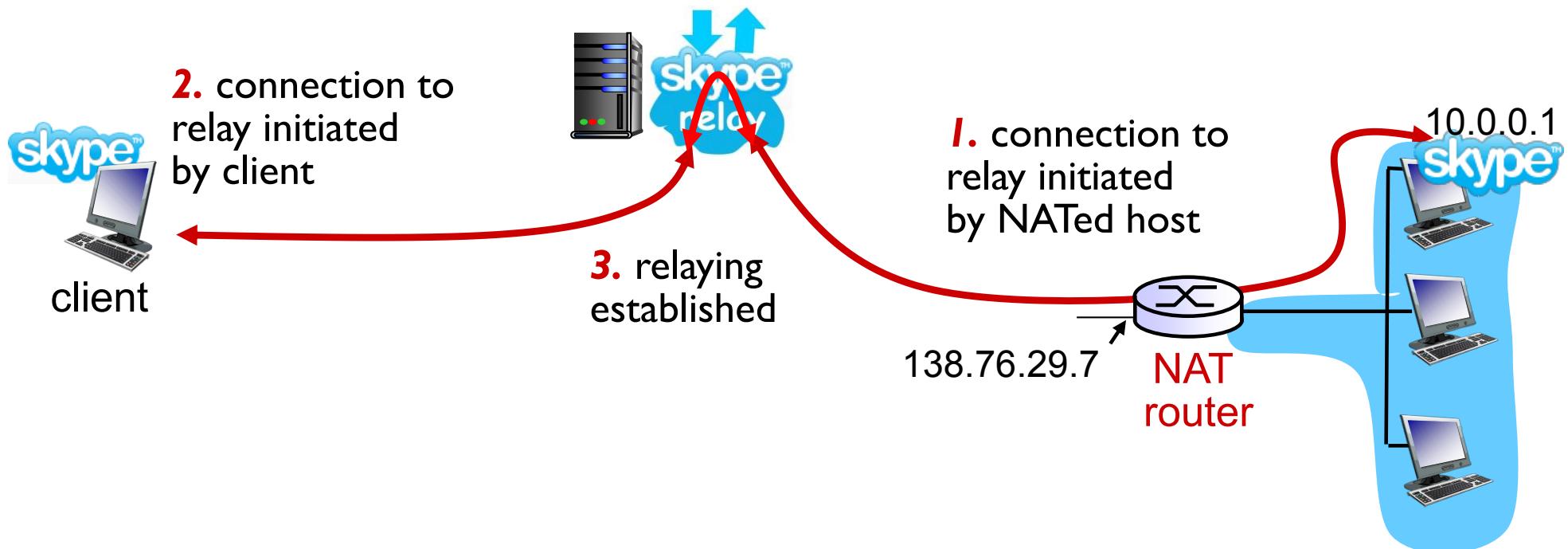
Durchqueren von NAT

- Lösung 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Dies ermöglicht dem Host hinter dem NAT Folgendes:
 - Herausfinden der öffentlichen IP-Adresse des NAT-Routers (138.76.29.7)
 - Kennenlernen existierender Abbildungen in der NAT-Tabelle
 - Einträge in die NAT-Tabelle einfügen oder aus ihr löschen
 - Das heißt automatische Konfiguration von statischen NAT-Einträgen



Durchqueren von NAT

- Lösung 3: Relaying (z.B. von Skype verwendet)
 - Server hinter einem NAT-Router baut eine Verbindung zu einem Relay auf (welches nicht hinter einem NAT-Router liegt)
 - Client baut eine Verbindung zum Relay auf
 - Relay leitet die Pakete vom Client zum Server und umgekehrt weiter

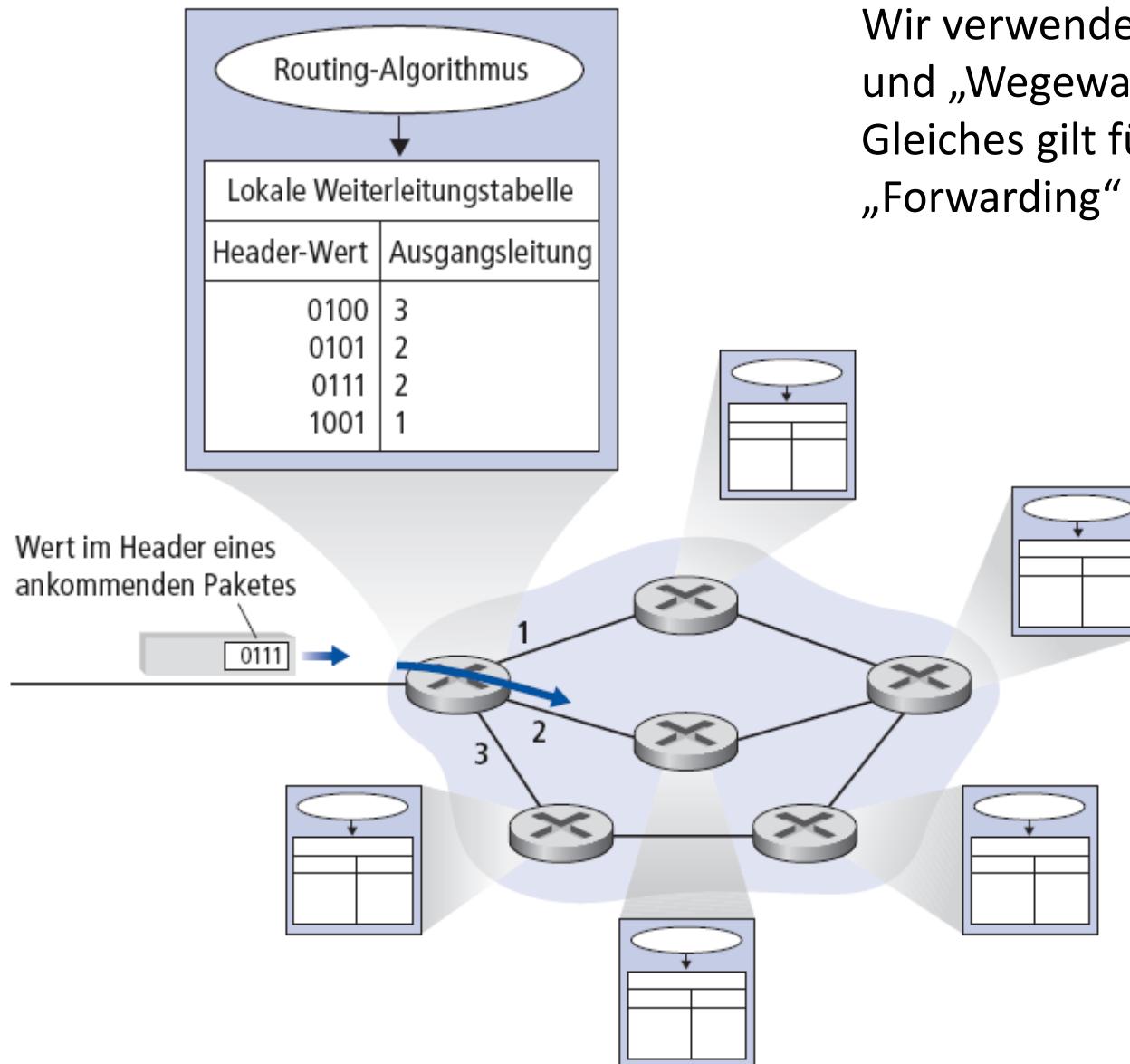


IPv6: Motivation

- Ursprüngliche Motivation: 32-Bit-Adressen werden in naher Zukunft komplett zugeteilt sein
 - NAT hat dies ein wenig verzögert, aber das grundlegende Problem nicht gelöst
 - Beispiel: Was passiert, wenn jedes Handy eine feste IP-Adresse bekommen soll?
 -
 - Weitere Motivation:
 - Vereinfachtes Header-Format für eine schnellere Verarbeitung in den Routern
 - Header soll Dienstgütemechanismen (Quality of Service, QoS) unterstützen
 - IPv6-Datagrammformat:
 - Header fester Länge (40 Byte)
 - keine Fragmentierung in den Routern erlaubt
- Adressen 128 Bit (statt 32 Bit bei IPv4)
-
- The diagram illustrates the structure of an IPv6 datagram header. At the top, it says "32 Bit". Below is a table with the following fields and their widths:
- | | | | | |
|---------------------------|----------------|------------|-----------------|-----------|
| Version | Verkehrsklasse | Flow-Label | | |
| Nutzdatenlänge | | | Nächster Header | Hop-Limit |
| Quelladresse
(128 Bit) | | | | |
| Zieladresse
(128 Bit) | | | | |
| Daten | | | | |
- Two red arrows point from the text "Adressen 128 Bit (statt 32 Bit bei IPv4)" to the "Quelladresse (128 Bit)" and "Zieladresse (128 Bit)" fields in the diagram.

- Einleitung und Datagramme vs. virtuelle Leitungen**
- Aufbau eines Routers**
- IP – Internet Protocol**
- Routing im Internet**
- Zusammenfassung und Ausblick**

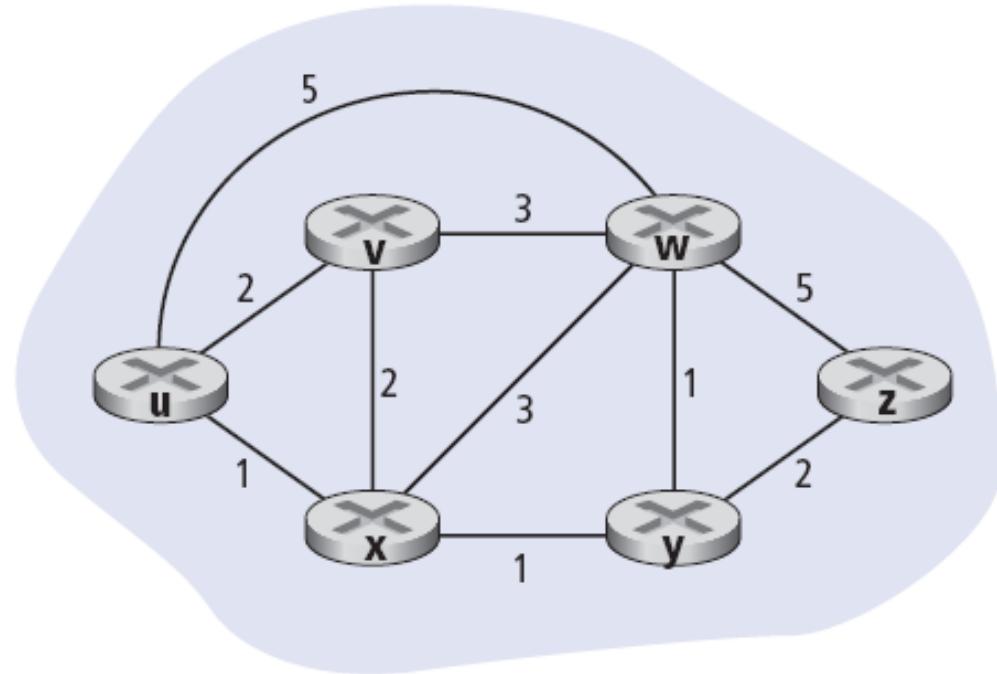
Routing und Forwarding



Wir verwenden „Routing“ und „Wegewahl“ als Synonyme.
Gleiches gilt für
„Forwarding“ und „Weiterleitung“.

Ein Netzwerk als Graph

- Graph: $G = (N, E)$
- $N = \text{Menge von Routern} = \{ u, v, w, x, y, z \}$
- $E = \text{Menge von Links} = \{ (u,v), (u,x), (u,w), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

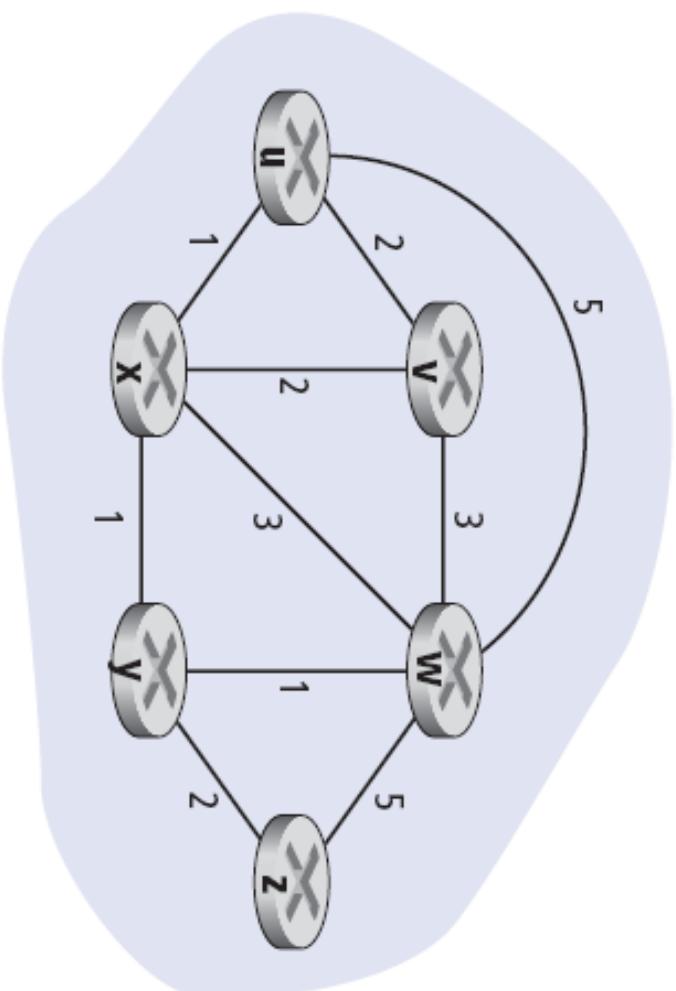


Ein Netzwerk als Graph: Kosten

- $c(x,x')$ = Kosten von Link (x,x')
- z.B. $c(w,z) = 5$
- Kosten können:
 - immer 1 sein
 - invers proportional zur Datenrate sein
 - invers proportional zum Verkehrsaufkommen sein
 - ...

Kosten eines Pfades $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

Frage: Was ist der günstigste Weg von u nach z?



Routing-Algorithmus: Algorithmus, der den günstigsten Weg findet

Klassifikation von Routing-Algorithmen

Globale oder dezentrale Informationen?

- **Globale Informationen:**

- Alle Router kennen die vollständige Topologie des Graphen (alle Knoten, alle Kanten, alle Kosten)
- Link-State-Routing

- **Dezentrale Informationen:**

- Router kennt die Kanten und Kosten zu seinen direkten Nachbarn
- Router tauschen iterativ Informationen mit ihren Nachbarn aus
- Distance-Vector- oder Distanzvektor-Routing

Statisch oder dynamisch?

- **Statisch:**

- Routen ändern sich langsam/selten
- Einmaliges Ausführen des Algorithmus OK
- Manuelle Konfiguration (manchmal) OK

- **Dynamisch:**

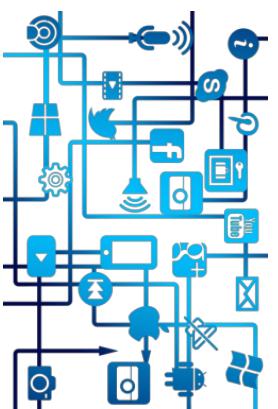
- Routen ändern sich schnell/ständig
- Periodisches Ausführen des Algorithmus notwendig
 - Als Reaktion auf Änderungen von Links

- Manuelle Konfiguration nicht möglich

↓
Im Internet werden eine Vielzahl von Routing-Algorithmen eingesetzt (RIP, OSPF, BGP). Diese behandeln wir nicht. Stattdessen schauen wir uns das manuelle Routing noch einmal im Detail im Cisco Packet Tracer an.

Begleitmaterialien

- Video zum Routing Information Protocol (RIP)
<https://www.youtube.com/watch?v=0efXawUgNZg>
- Video zum Border Gateway Protocol (BGP)
<https://www.youtube.com/watch?v=z8INzy9E628>



Übung

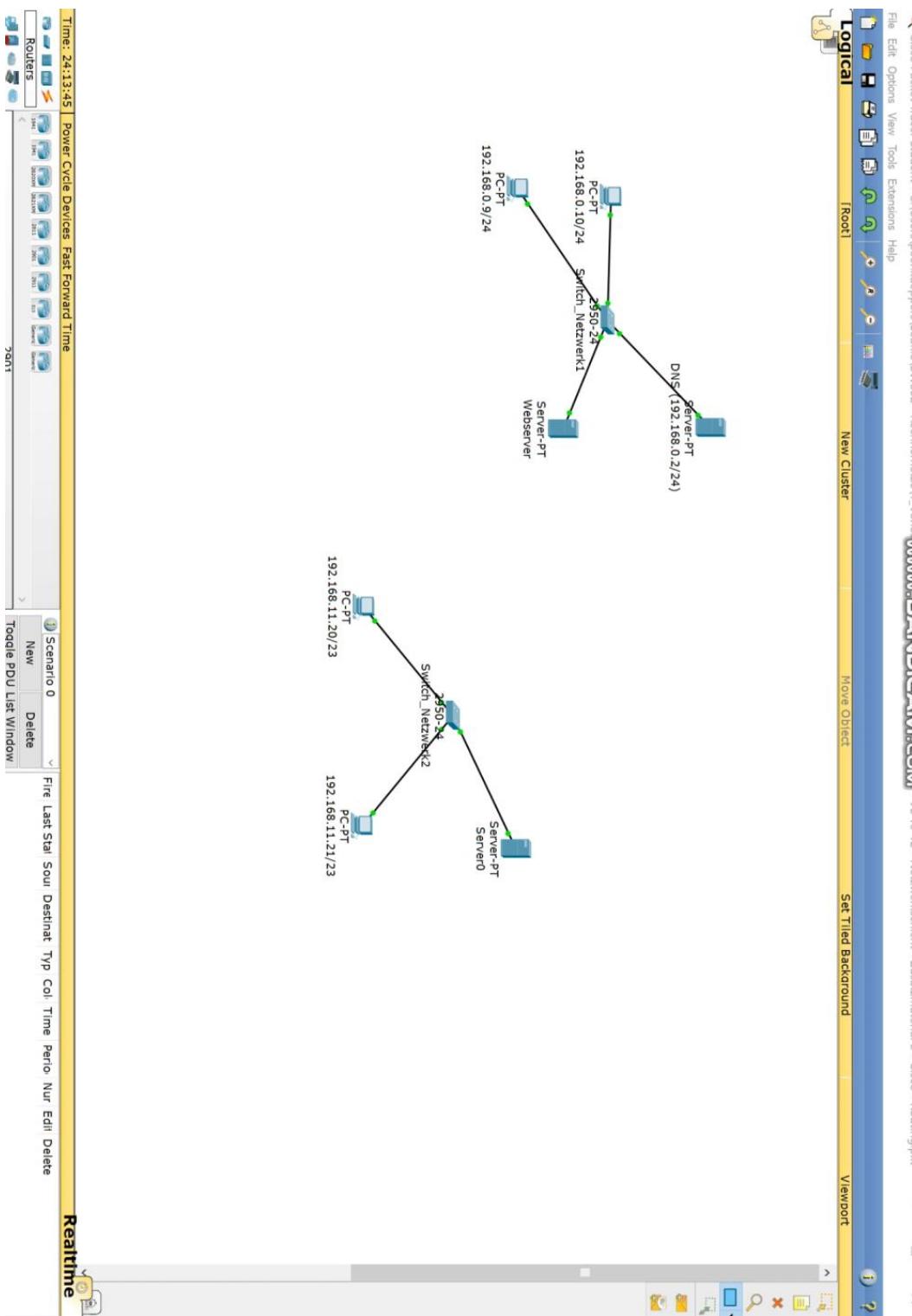
- Rufen Sie das Cisco Packet Tracer-Projekt aus Übung 6.2 auf.
- Wir möchten nun beide Netzwerke mittels Routern verbinden.
- Diese zwei Router fügen wir ein.
 - Beide müssen mit dem jeweiligen Switch verbunden werden und die Netzwerk-Interfaces mit der festen IP-Adresse (192.168.0.1 bzw. 192.168.10.1) und dem richtigen Subnetz aktiviert werden.
 - Die Router müssen über das jeweils zweite Interface miteinander verbunden werden. Dort vergeben wir die IP-Adressen 10.0.0.1 (linker Router) und 10.0.0.2 (rechter Router).
- Nun müssen wir die DHCP-Konfigurationen ändern.
 - Da Router (bzw. das Standard-Gateway) üblicherweise die erste Adresse in einem Subnetz tragen (so wie oben angelegt), ändern wir die verfügbaren Ranges (im linken Netzwerk soll ab 192.168.0.3 die Adressvergabe starten (da hier noch der Nameserver vorhanden ist), im rechten ab 192.168.10.2). Im Moment kann die Router-Adresse noch vergeben sein.
 - Zusätzlich tragen wir in beiden DHCP-Servern die entsprechende Router-IP-Adresse als Default Gateway ein. Im linken DHCP-Server darüber hinaus noch den DNS-Server.
 - Danach bei allen Clients (bis auf den DNS-Server) von static auf dynamic IP-Configuration.
 - Beim DNS-Server von Netzwerk 1 müssen wir noch das Default Gateway in der IP-Konfiguration auf 192.168.0.1 setzen.

Übung

- Zuletzt müssen wir noch die Routing-Tabellen auf den Routern pflegen. Wir werden nur statische Routen festlegen.
 - Dazu muss im linken Router (von Netzwerk 1) eingetragen werden, dass next hop = 10.0.0.2 für Zieladressen 192.168.10.0 ist (die Null am Ende besagt alle Adressen in dem Subnetz).
 - Im rechten Router (von Netzwerk 2) wird eingetragen, dass für Addressbereich 192.168.0.0 der next hop = 10.0.0.1 ist (also der linke Router).

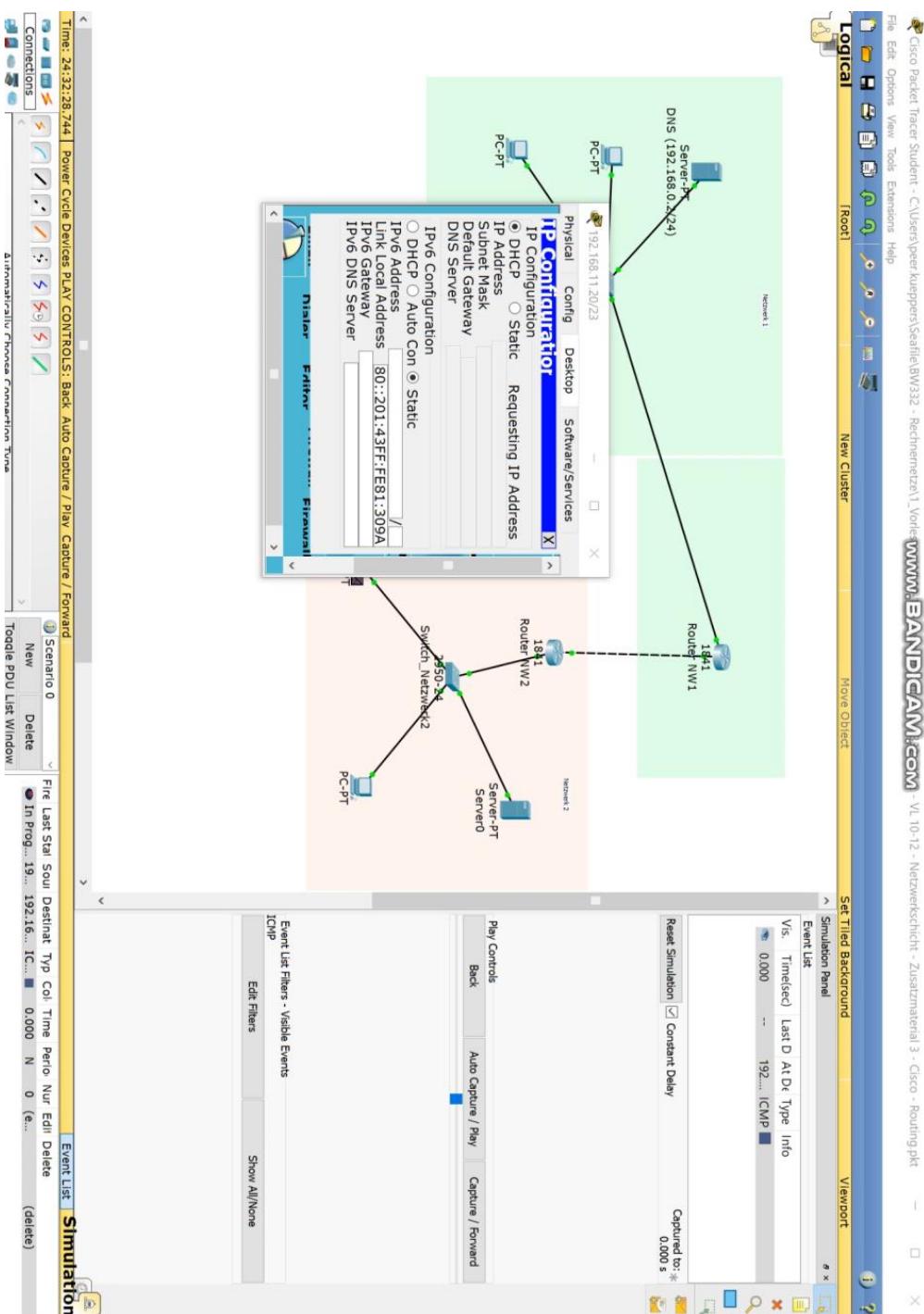
Übung

Lösungsvorschlag:



Übung

Lösungsvorschlag:



Agenda – VL 9-11

- Einleitung und Datagramme vs. virtuelle Leitungen
- Aufbau eines Routers
- IP – Internet Protocol
- Routing im Internet
- Zusammenfassung und Ausblick

Zusammenfassung (VL 9-11) und Ausblick

- In den letzten Vorlesungen haben wir die Netzwerkschicht behandelt:
 - Hier haben wir den grundsätzlichen Unterschied zwischen verbindungslosen (Datagramm-) und verbindungsorientierten Netzwerkschicht-Protokollen betrachtet.
 - Da das Internet eine Datagramm-Netzwerkschicht implementiert (und das bedeutendste Netzwerk ist), haben wir uns darauf fokussiert und die Geräte, die für die Realisierung der Netzwerkschicht im Internet verantwortlich sind, im Detail betrachtet: Router.
 - Die Netzwerkschicht im Internet wird durch IP implementiert. Wir haben daher dieses Protokoll diskutiert, insbesondere in Bezug auf die Adressierung.
 - Dabei haben wir DHCP als Protokoll für die Vergabe von IP-Adressen behandelt, NAT (Network Address Translation) als wichtigen Teil im Umfeld von IPv4 kennengelernt und einen kurzen Ausblick auf IPv6 vorgenommen.
 - Letztlich haben wir uns Routing-Protokolle angesehen und anhand des Cisco Packet-Tracers das Routing zwischen Netzwerken ausprobiert.
- Die weiteren Schichten unterhalb der Netzwerkschicht behandeln wir nicht mehr im Detail. Diese wurden bereits in den Vorlesungen 1-3 besprochen.



A large pile of white question marks of various sizes and orientations, filling most of the frame. One question mark stands out from the crowd, being colored red.

Vielen Dank für Ihre Aufmerksamkeit!