



Technische Hochschule
Ingolstadt
Fakultät Informatik

Kapitel 1: Wiederholung und Grundlagen

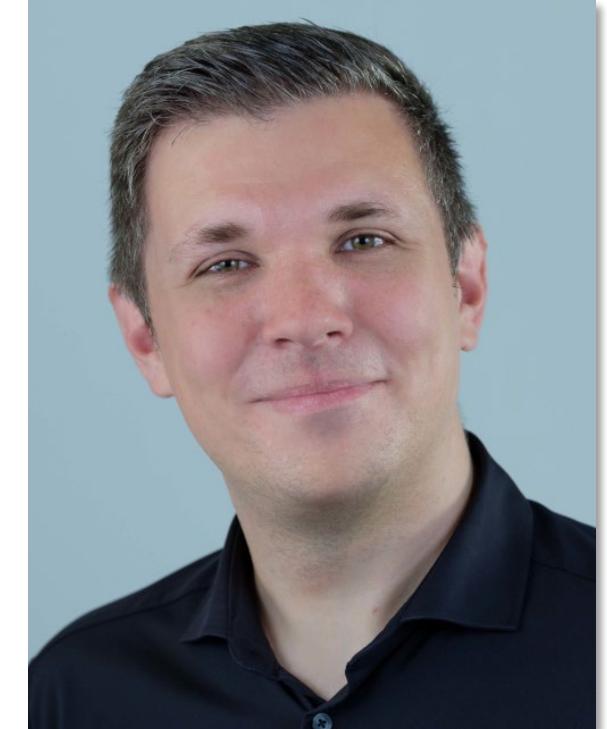
CASE_SMN WS 2023/2024

Vorlesung „Sicherheit moderner Netze“

03.10.2023

Prof. Dr. Michael Jarschel

- Büro: B109
- Sprechstunde: Donnerstags von 12:00-13:00 Uhr
- E-Mail: michael.jarschel@thi.de
- Vergangenheit:
 - Promotion an der Julius-Maximilians Universität Würzburg (Lehrstuhl für Kommunikationsnetze)
Fokus: Software Defined Networking
 - **2014-2016** Research Engineer Software-Defined Networks bei Nokia Technology & Innovation München
 - **2016-2021** (Senior) Research Engineer bei Nokia Bell Labs
Fokus: Telco Cloud, Edge Computing, Hardware Acceleration



Wichtig → Einschreiben bei Moodle



- Bitte schreiben Sie sich selbst in den Moodle-Kursraum ein:
- Kurs:
 - Sicherheit moderner Netze
 - IM_SMN_2726
 - Einschreibeschlüssel: **safe-and-secure**
- Hier finden Sie die Unterlagen zum Kurs, z.B. Manuskript / Vortragsfolien, ..
- Die Unterlagen erhalten Sie begleitend zur Vorlesung

Unser Ziel:

- **Kenntnisse über wesentliche Gefährdungen in modernen Kommunikationsnetzen und Ansätze wie die Netze dagegen gerüstet werden.**
- **Beurteilung der aktuellen Kommunikationsnetze (IP-Netze, Mobilfunk-Netze) nach Gesichtspunkten der Sicherheit und Zuverlässigkeit**
- **Struktur der aktuellen Netze und**
 - bekannte sicherheitsrelevanten Schwachstellen
 - Maßnahmen der Betreiber (Architektur, Netzdesign, Protokolle, zusätzliche Funktionen).

Was erscheint Ihnen wichtig?



- **Notieren Sie sich zum folgenden Video, was Ihnen wichtig/interessant erscheint!**
- **Beispielsweise:**
 - Sicherheitsgefahren
 - Angriffe
 - Überraschendes
 - Empfehlungen
 -



Keynote - Complexity: The Enemy of Security

*Mikko Hypponen, Computer Security Researcher,
F-Secure*

Was hat sich verändert?



- **Notieren Sie sich zum folgenden Video, die wichtigsten Unterschiede zum vorhergenden!**
- **Beispielsweise:**
 - Sicherheitsgefahren
 - Schwerpunkt von Angriffen
 - Sicherheitsfokusgebiete
 - Neue Entwicklungen
 -



Der Deutsche Begriff „Sicherheit“ umfasst 2 unterschiedliche Aspekte:

- **Security**
- => Maßnahmen zum Schutz vor betrügerischen Aktivitäten
 - Betrug
 - Fälschungen
 - Kriminalität
 - illegale Aktivitäten
 - Eindringversuche
- **Wie z.B.**
 - Firewalls,
 - Security Policy
 - Verschlüsselung
- **Safety**
- => Maßnahmen zum Schutz vor Unfällen, Ausfällen und Störungen
 - Ausfall von Netzkomponenten (Router, Laser,...)
 - Unterbrechung von Leitungen („Bagger“)
 - Störung des Betriebs (Stromausfall, Ausfall Klimaanlage,...)
- **Wie z.B.**
 - Redundante Systeme
 - Unterbrechungsfreie Stromversorgung,

Security ist notwendig, aber nicht hinreichend für Safety
Ein System ohne Safety ist nicht sicher

- **Wiederholung der Grundlagen der Datenkommunikation**
- **Analyse sicherheitsrelevanter Netzfunktionen**
(Sicherer Betrieb / Resilience, Schutz vor Angriffen)
- **Protokolle und Funktionen zur Erhöhung der Sicherheit im IP-Netz z.B. SSL / TLS, IPSec, Tunneling**
- **Architektur der aktuellen Kommunikationsnetze**
(Transport-Netz, IP-Netz, Access-Netze, Mobilfunk-Netze)
- **Netzarchitektur und Netzelemente zur Überwachung und Steuerung des Betriebs (Firewall, DMZ, DPI, ...)**
- **Sicherheit im Zugang: Autorisierung der Teilnehmer und Gestaltung des Zugangsnetzes (beim Festnetz / im Mobilfunknetz)**

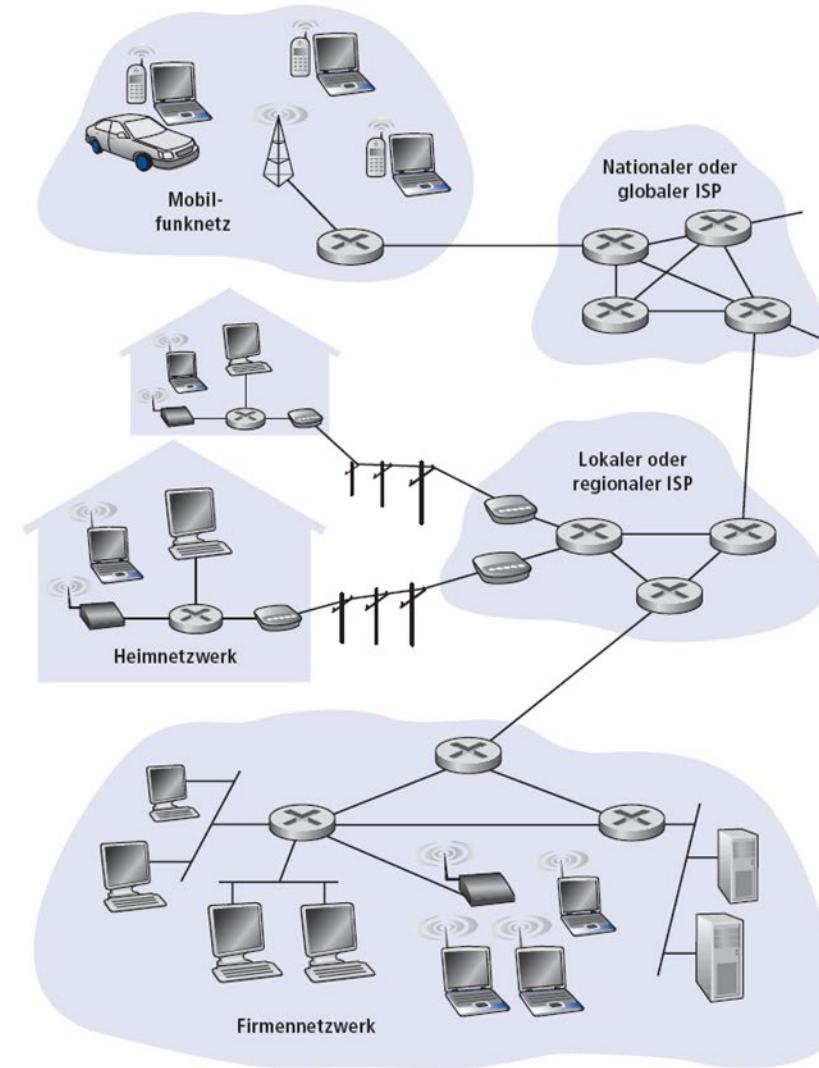


Grundlagen zum Internet

(eine kurze Wiederholung)

- **Millionen vernetzter Computer:**
Hosts = Endsysteme
 - Auf denen Netzwerkanwendungen und Applikationen laufen
 - Oft mit einer Schnittstelle zum Anwender (Menschen)
- **Leitungen/Funkstrecken (links)**
 - Glasfaser, Kupfer, Funk, Satellit
 - Transportsysteme
- **Router:**
 - IP-Router
 - Ethernet-Switche
- **Control-Elemente**
 - Server für den Zugang (RAS)
 - Server für Dienste (IMS)
 - Überwachung des Datenstroms (DPI, Firewall,...)

Unser Fokus hier

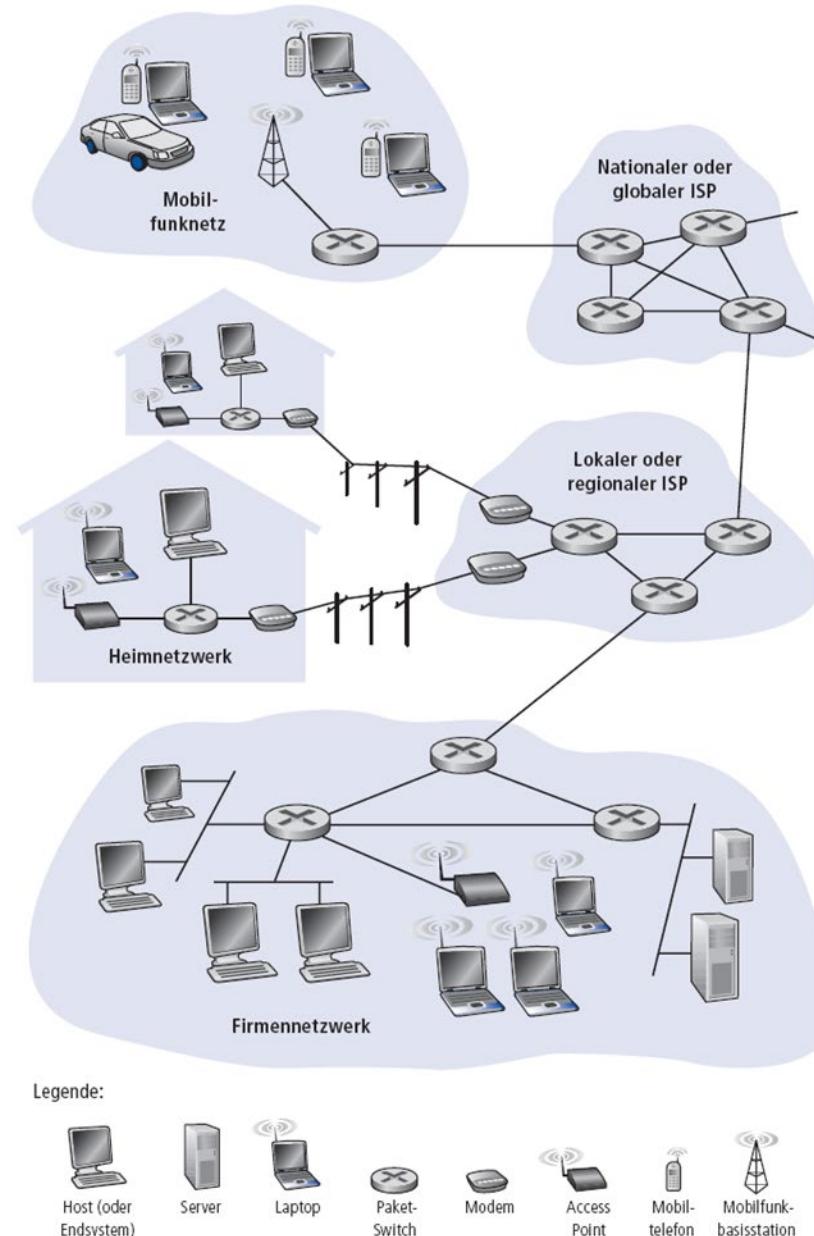


Legende:



[Quelle: Kurose, J., Ross, K.: Computernetzwerke : der Top-Down-Ansatz]

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

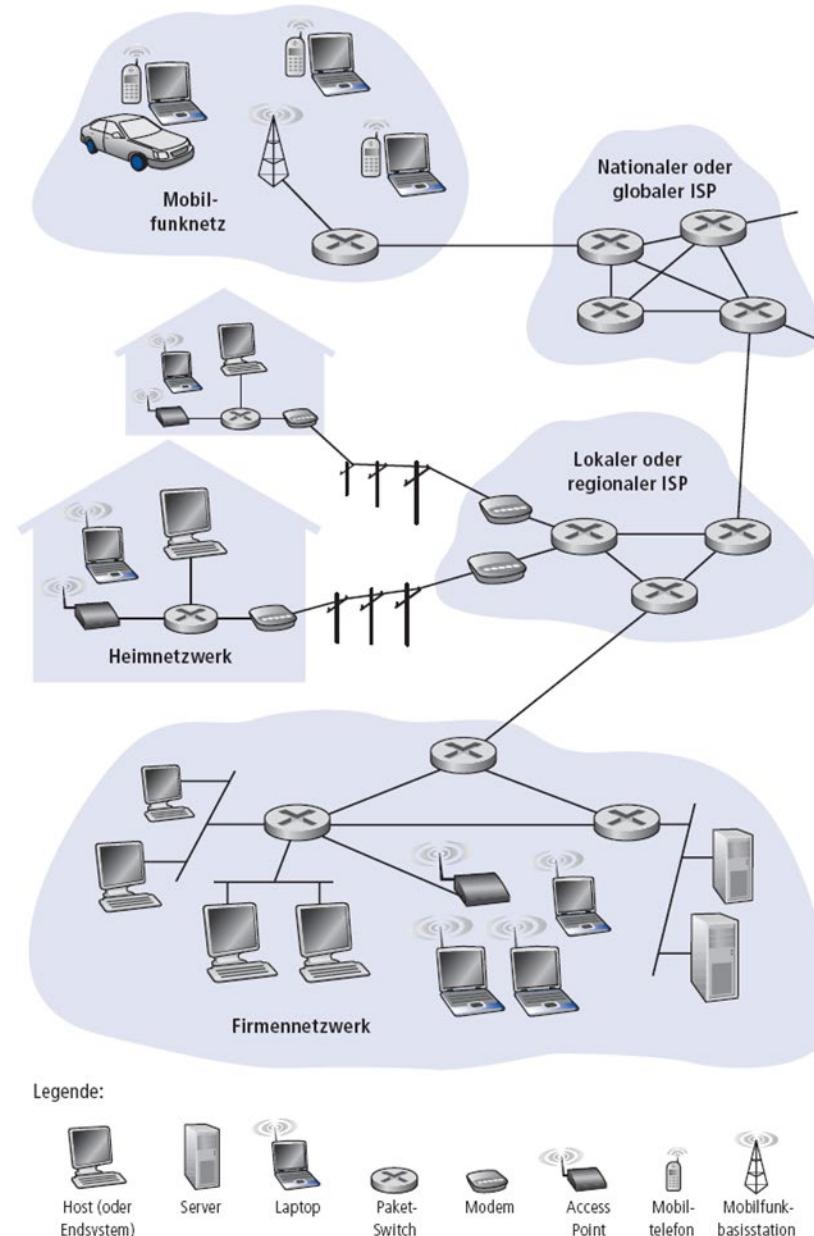


[Quelle: Kurose, J., Ross, K.: Computernetzwerke : der Top-Down-Ansatz]

Die Struktur eines Netzwerkes:



- **Randbereich:**
 - Anwendungen und Endsysteme
- **Zugangsnetzwerke, physikalische Medien: („Access“)**
 - Kabel (Kupfer und Glasfasern) und
 - drahtlose Verbindungen
- **Inneres des Netzwerkes („Core“):**
 - Router
 - Verbindungen (Leitungen)
 - Netzwerke von Netzwerken



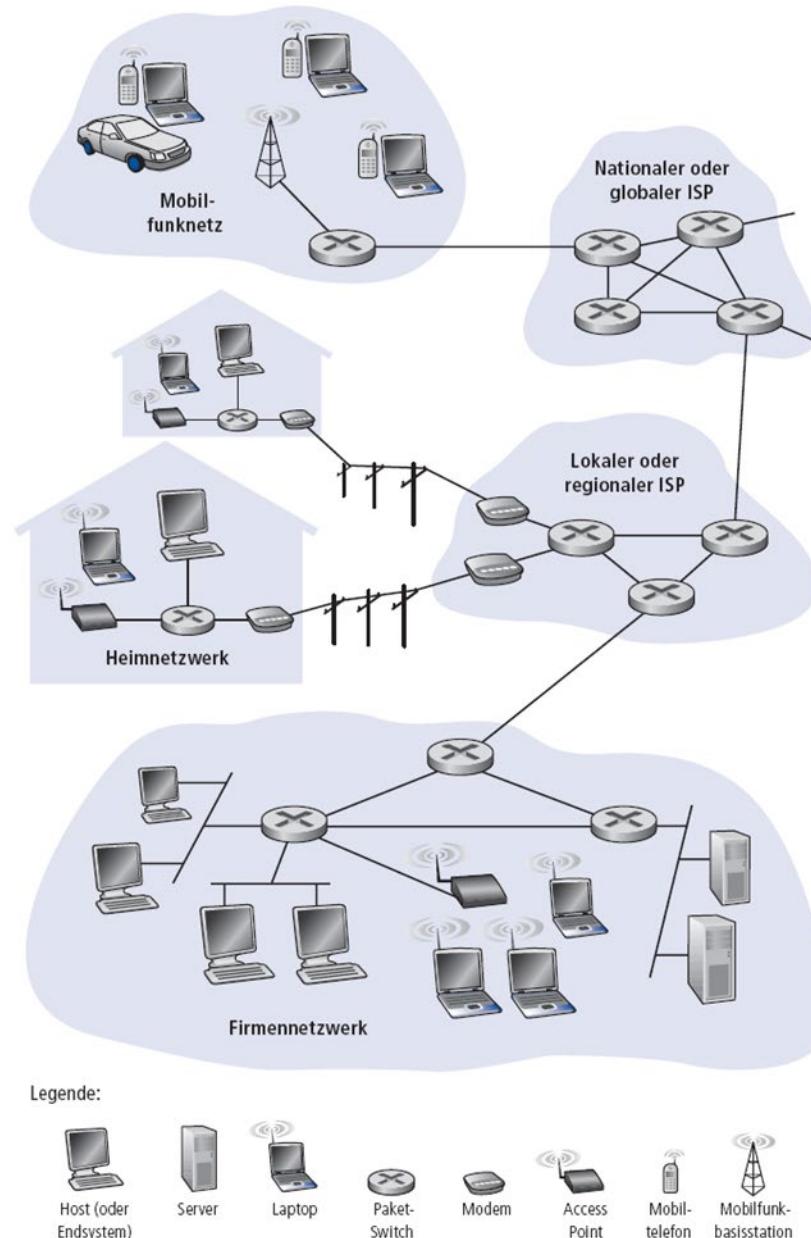
[Quelle: Kurose, J., Ross, K.: Computernetzwerke : der Top-Down-Ansatz]



Grundlagen zum Internet

→ *Zugang zum Netz*

- **Frage: Wie werden Endsysteme an das Netzwerk angebunden?**
 - Anbindung von Privathaushalten
 - Anbindung von Institutionen
 - Mobile Anbindung
- **Wichtige Eigenschaften:**
 - Bandbreite (Bits pro Sekunde) der Anbindung
 - Geteilte oder exklusive Nutzung der Anbindung („shared Medium“?)
- **Wie sieht das Netz im Innern aus?**
 - Welche Systeme?
 - Struktur der Netzes?

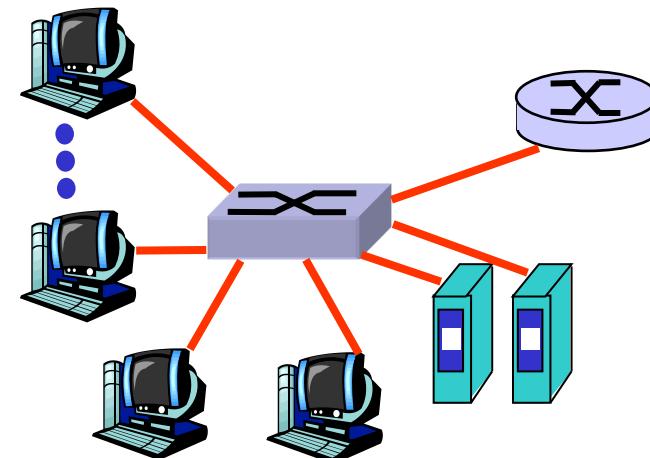


[Quelle: Kurose, J., Ross, K.: Computernetzwerke : der Top-Down-Ansatz]

- Firmen werden meist mit Glasfasern direkt angebunden
 - Datenraten $n * 10$ Mbit/s bis $n * 1$ Gbit/s, in Extremfällen auch mehr

- Innerhalb von Firmen:

- Lokale Netze
(LAN = Local Area Networks, Ethernet-Netze)
 - Datenraten: 100 Mbit/s, 1000 Mbit/s (PC-Anschluss)
evtl. 10 Gbit/s (Server)
 - Struktur: Direkt-Anschluss an PC,
Ethernet-Switche, Router
 - Zusätzlich WLAN für mobile Clients
(Smartphone, Notebooks)



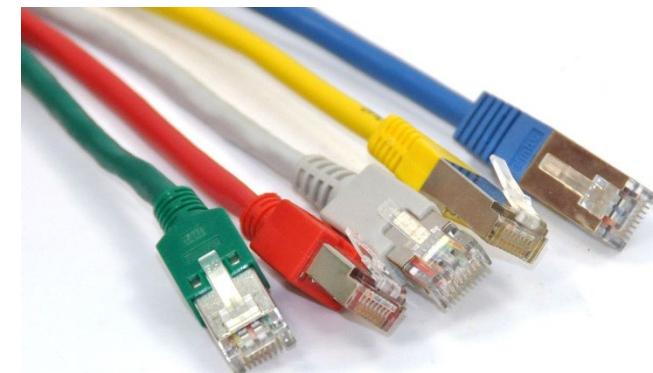
[Quelle: Kurose, J., Ross, K.: Computernetzwerke : der Top-Down-Ansatz]

- **Bit:** wird von einem Sender zu einem Empfänger übertragen
- **Leitung:** das, was sich zwischen Sender und Empfänger befindet
- **Gerichtete Medien:**
 - Signale breiten sich in festen Medien in eine Richtung aus:
 - Kupfer-, Glasfaser-, Koaxialkabel
 - Richtfunk
 - Free Space Optics
- **Ungerichtete Medien:**
 - Signale breiten sich frei aus:
 - Funk, Mikrowellen

■ **Twisted Pair (TP)**

Zwei isolierte Kupferleiter, verdrillt, meist nicht abgeschirmt (UTP, Unshielded TP)

- Kategorie 3: Telefonkabel, 10 Mbit/s Ethernet
 - Kategorie 5:
100 Mbit/s Ethernet
 - Kategorie 6 / 7:
1000 Mbit/s Ethernet
- Zusätzliche Abschirmungen



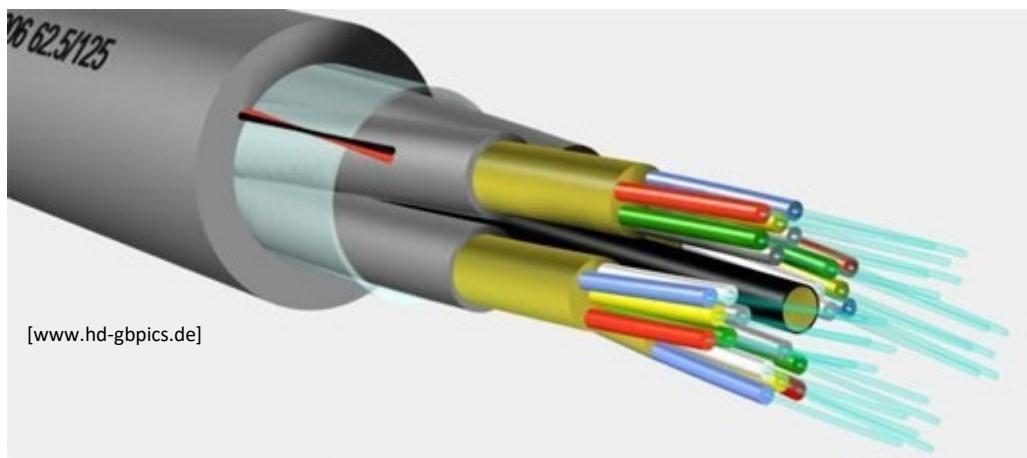
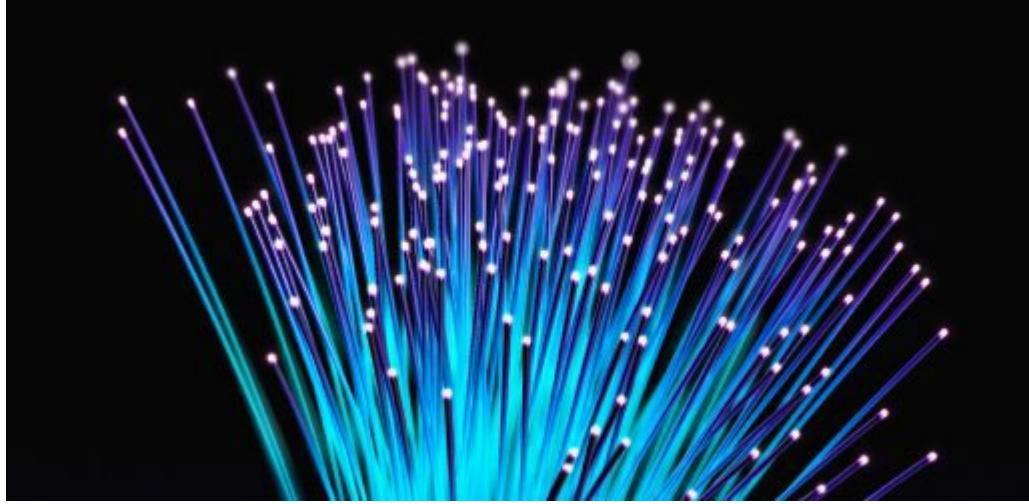
Koaxialkabel:

- Zwei konzentrisch angeordnete Kupferleiter (= Wellenleiter)
- Bidirektional
- Basisband:
 - Ein Kanal auf dem Kabel
 - „Altes“ Ethernet (Bus)
- Breitband:
 - Mehrere Kanä
 - HFC



Glasfaserkabel:

- Glasfaserkabel übertragen Lichtpulse, jeder Puls ist ein Bit
- Hohe Geschwindigkeit:
 - Mehrere 10 Gbit/s bis mehrere 100 Gbit/s
 - Parallel Übertragung mehrerer Signale auf unterschiedlichen Wellenlängen („Farben“, → Wellenlängen-Multiplex, WDM)
- Geringe Fehlerrate; unempfindlich gegen elektromagnetische Störer
- Extrem hohe Reichweiten ohne Verstärker (bis über 1000 km möglich)
- Weites Spektrum an Systemen im Markt
 - Billige System für kurze Stecken (Plastik, LED)
 - Teure Systeme für große Datenraten / große Entfernung (Monomode-Glasfaser, Laser)



[www.hd-gbpics.de]



[www.partsdata.de]

Glasfaserkabel:
In der Werbung
Lokale Verbindung (LAN)
Weitverkehrskabel

- **Signal wird von elektromagnetischen Wellen übertragen**
- **„Drahtlose Kommunikation“**
- **deswegen auch Bidirektional**
- **Signalausbreitung wird von der Umgebung beeinflusst:**
 - Reflexion
 - Abschattung durch Hindernisse
 - Interferenz mit anderen Sendern
 - Schnelle Änderungen der Ausbreitung der Signale durch Störungen, Überlagerungen, Interferenzen, ...

Arten von Funk:

- **WLAN (Wifi, 802.11)**
 - Zahlreiche Varianten im 2,4 und 5 GHz Band
 - Nutzdaten: meist << 100 Mbit/s
 - Leistung stark begrenzt (100 mW)
- **Mobilfunknetze**
 - 3G (UMTS/HSPA): einige Mbit/s
 - 4G (LTE): einige 10*Mbit/s möglich
 - 5G: 100 Mbit/s mögliche
- **Mikrowellen (Richtfunk, im Netz)**
 - Kanäle mit $n \cdot 10$ Mbit/s bis zu einigen Gbit/s
 - Sichtverbindung erforderlich
- **Satellit (für Sonderfälle)**
 - Kanäle mit bis zu 45 Mbit/s (oder mehrere kleine Kanäle)
 - Geostationär mit 270 ms Ende-zu-Ende-Verzögerung
 - Satelliten in geringer Höhe (LEO) → Starlink

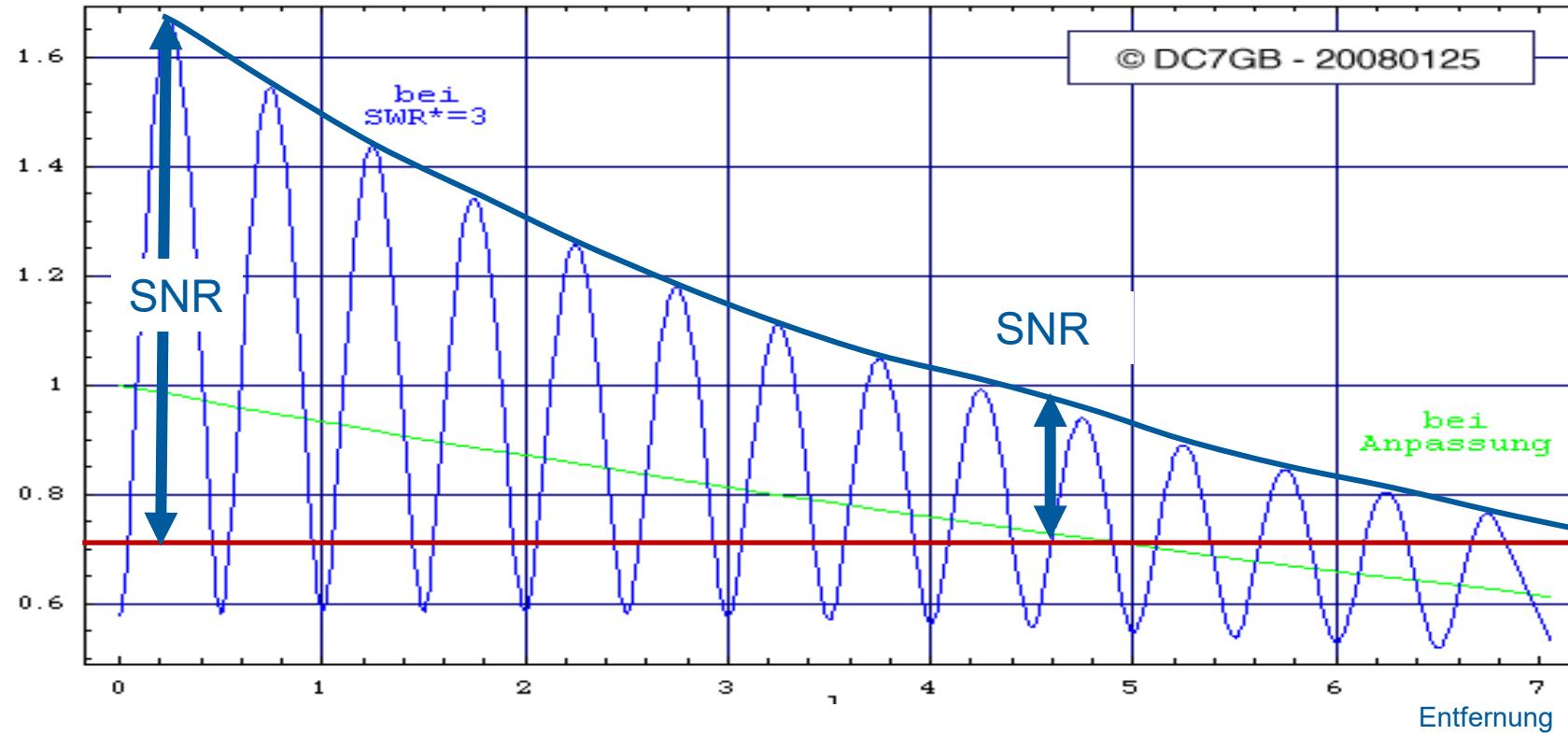
- **Öffentlicher Mobilfunk bietet heute leistungsfähige Datenübertragung an:**
 - Erweiterungen des 3G-Netzes (UMTS): HSPA bis 42 Mbit/s
 - LTE (4G)-Netz mit (theoretisch) 150 Mbit/s je Zelle
 - Weiterentwicklung zu 5G (LTE-Advanced) 10 Gbit/s je Zelle
 - Begrenzung:
 - Verfügbares Spektrum → max. mögliche Datenrate in einer Zelle
 - Absolute Frequenz (800, 1800, 2600 MHz-Band)
 - Reichweite, physikalische Ausbreitung
 - Anzahl der gleichzeitig aktiven Nutzer
 - Neue Frequenzen (700 und 800 MHz-Band) sollen freigegeben werden für die Versorgung des ländlichen Raums
(Digitale Dividende, Frequenzen werden frei durch die Digitalisierung der TV-Kanäle)
- **WLAN (WiFi, IEEE 802.11):**
 - Unlizenziertes Band, daher beliebige Nutzung und Störungen möglich
 - Primär zur Kommunikation innerhalb einer Wohnung / Haus
 - Öffentliche WiFi-Hotspots (Innenstadt / Zentrum / Züge / ...) zur Entlastung der Mobilfunknetze
 - Ziel: für mobile, nomadische Nutzer

Kapazität eines Übertragungskanals



- Auf einer Leitung nimmt Signalstärke mit der Entfernung ab (Dämpfung)
- Die Dämpfung ist abhängig von:
 - Entfernung
 - Frequenz
- Jedes Signal wird mit den **Störungen** der Umgebung überlagert („Noise“)
diese Störungen sind i.A. gleichmäßig verteilt und auf der gesamten Leitung gleich.
- Um ein Nutz-Signal wieder erkennen zu können, muss es stärker als die Störungen sein:
- → Messgröße: **Abstand Signal - Störung = Signal/Noise Ratio, SNR**
- Konsequenzen:
Wichtig für eine Übertragung sind:
 - Frequenzen des Signals
 - Amplitude (Signalstärke), nimmt über Entfernung ab (frequenzabhängig)
 - Erforderlicher Abstand des Signals zu den Störungen (SNR)

Kapazität eines Übertragungskanals (Prinzip-Bild)



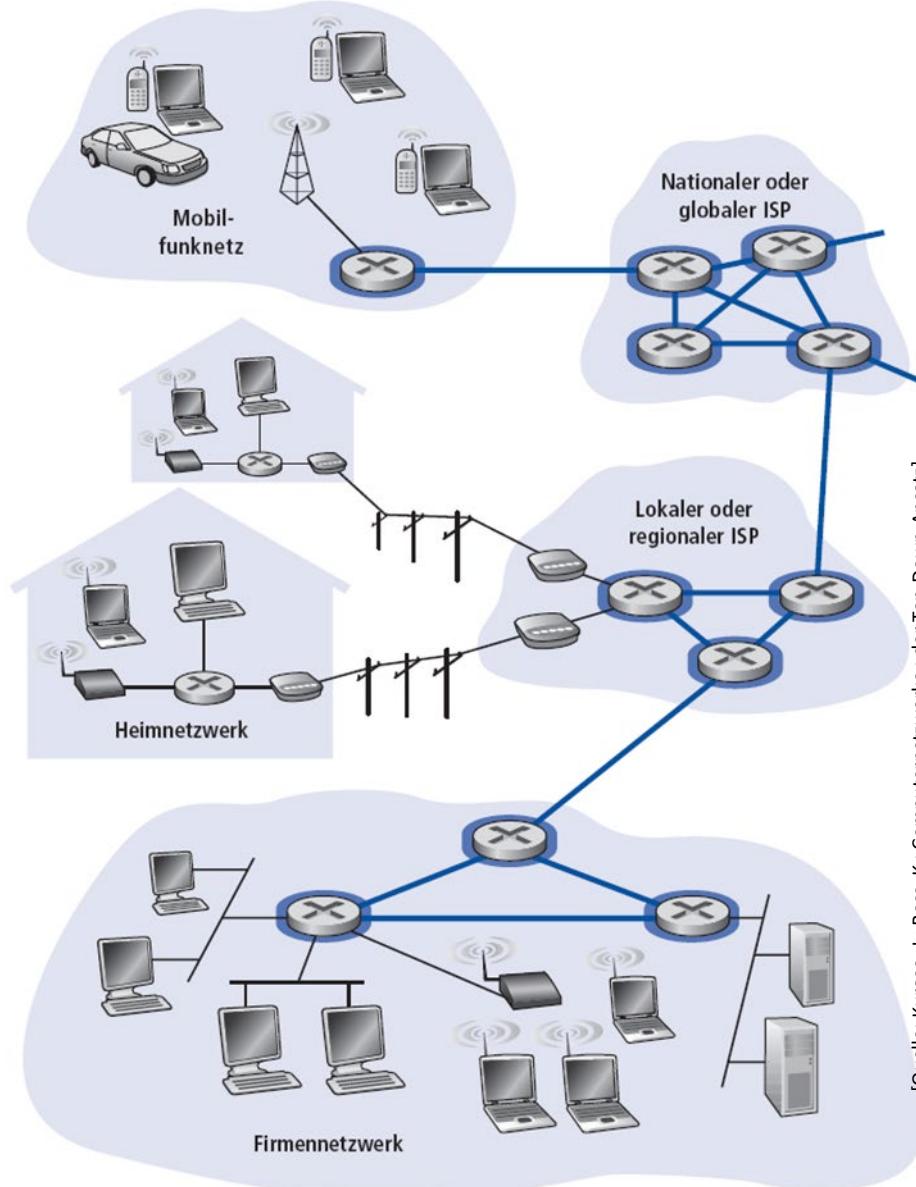
- Nutzsignal (blau) nimmt über der Entfernung ab
- Störsignal (Noise, rot) ist immer ungefähr gleich stark
- Zum guten Empfang muss der Nutzsignal das Störsignal überragen => SNR muss groß genug sein



Grundlagen der Kommunikation

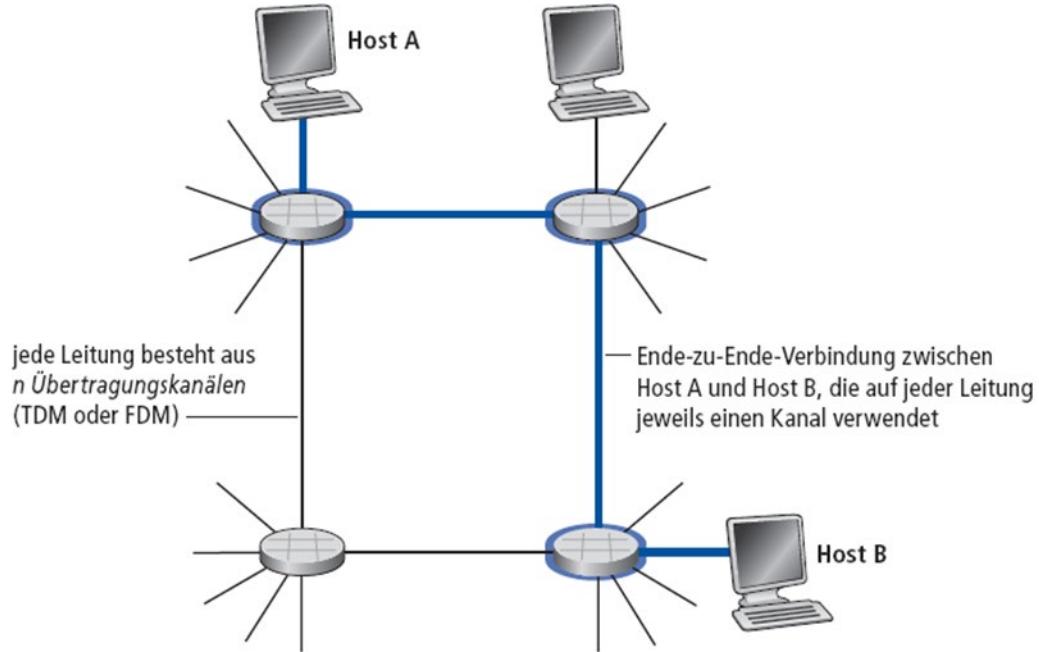
→ *Vermittlungsprinzipien*

- Mehrere, miteinander verbundene Netze
- Viele, untereinander verbundene Router
- Die zentrale Frage:
Wie werden Daten durch das Netzwerk geleitet?
 - **Leitungsvermittlung:** eine dedizierte Leitung wird für jeden Ruf geschaltet: Telefonnetz
 - **Paketvermittlung:** Daten werden in diskreten Einheiten durch das Netzwerke geleitet: Internet



[Quelle: Kurose, J., Ross, K.: Computer networks : the Top-Down Approach]

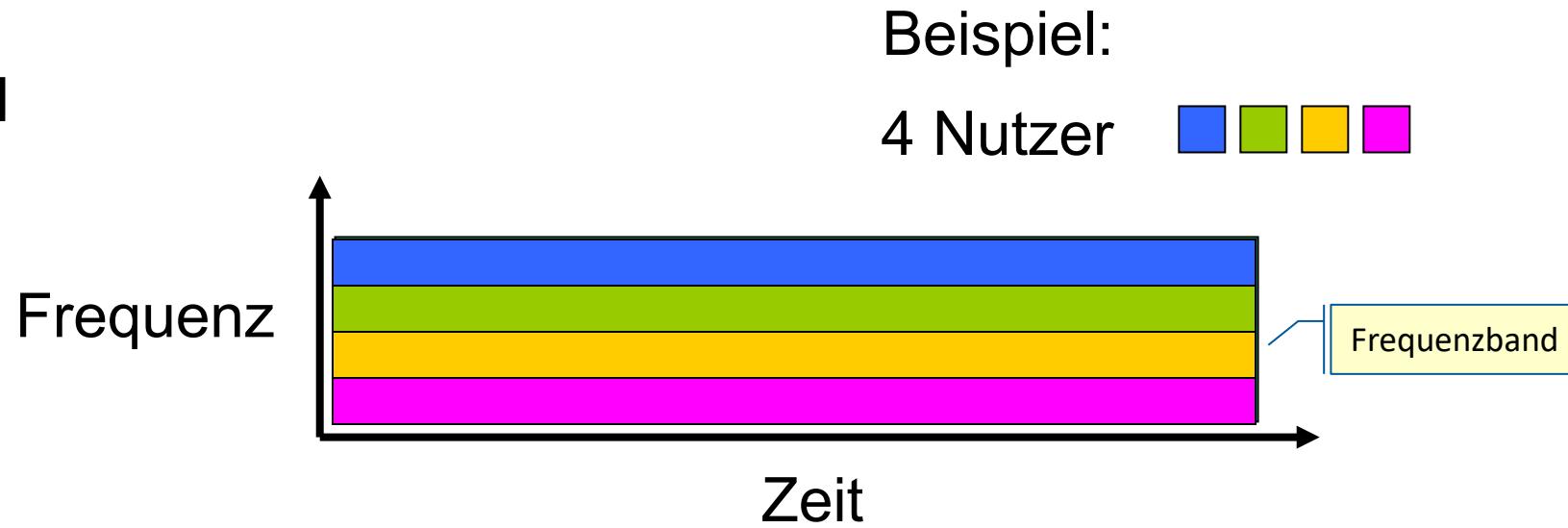
- Ende-zu-Ende-Ressourcen werden für einen Ruf **reserviert**
- Bandbreite auf Leitungen, Kapazität in Routern
- **Dedizierte Ressourcen:** keine gemeinsame Nutzung
- Garantierte Dienstgüte wie beim "Durchschalten" einer physikalischen Verbindung
- Vor dem Austausch von Daten müssen die notwendigen Ressourcen reserviert werden;
Signalisierung erforderlich



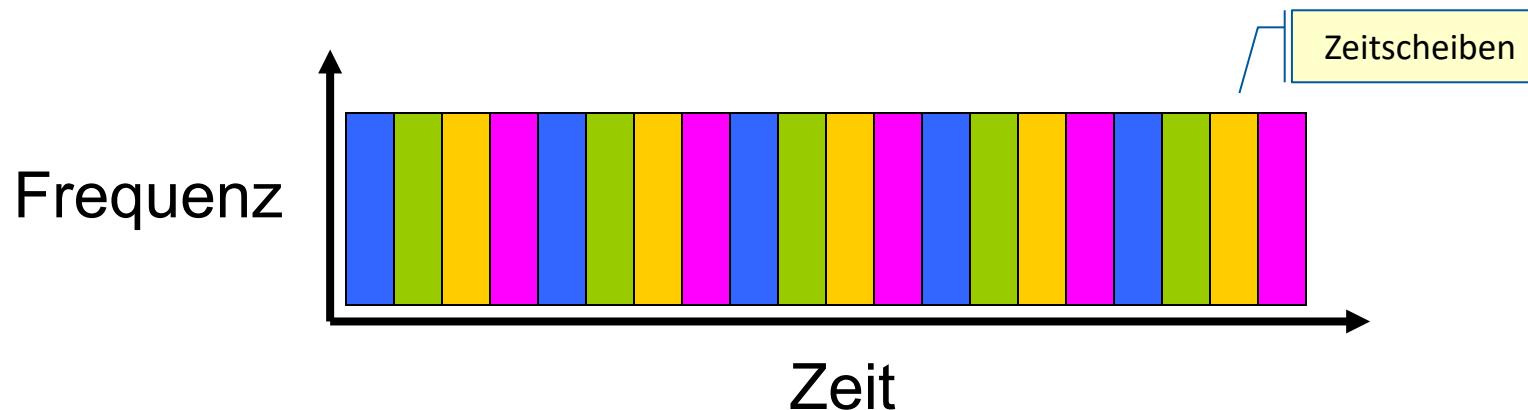
- Netzwerkressourcen (z.B. Bandbreite) werden in Einheiten aufgeteilt
- Einheiten werden Rufen zugewiesen
- Einheiten bleiben ungenutzt, wenn sie von ihrem Ruf nicht verwendet werden (keine gemeinsame Nutzung von Ressourcen)

- Wie teilt man die Bandbreite einer Leitung in Einheiten auf?
 - Frequenzmultiplex (Frequency Division Multiplex, FDM)
 - Zeitmultiplex (Time Division Multiplex, TDM)
 - Wellenlängenmultiplex (Wavelength Division Multiplex, WDM) auf Glasfasern
 - Code-Multiplex (Code Division Multiplex, CDMA)

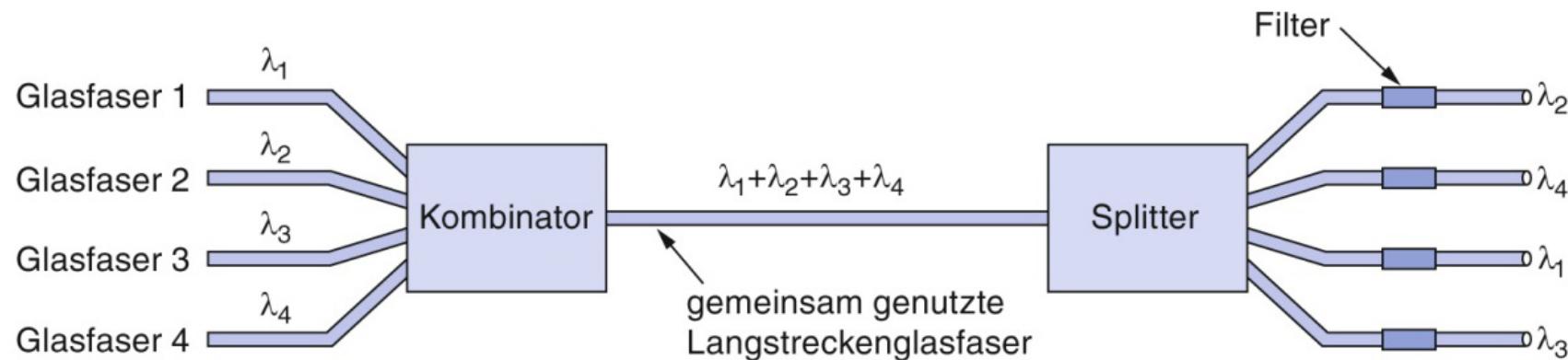
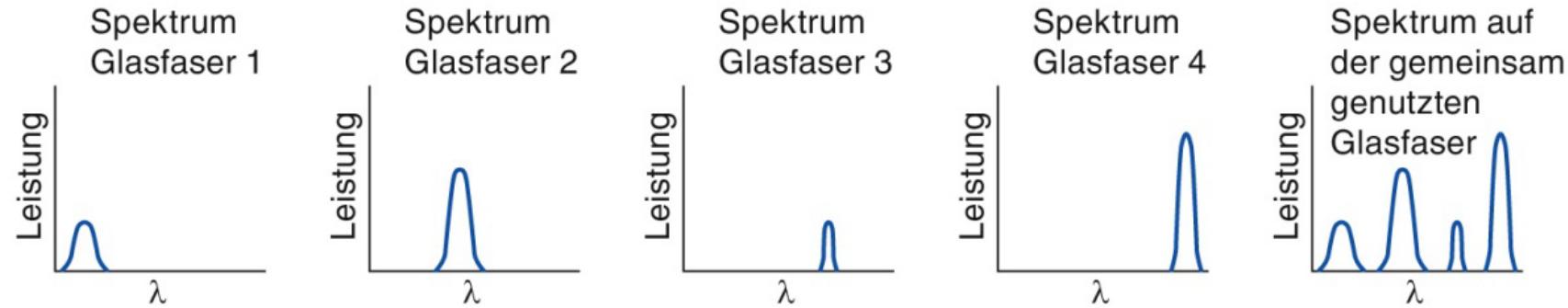
FDM



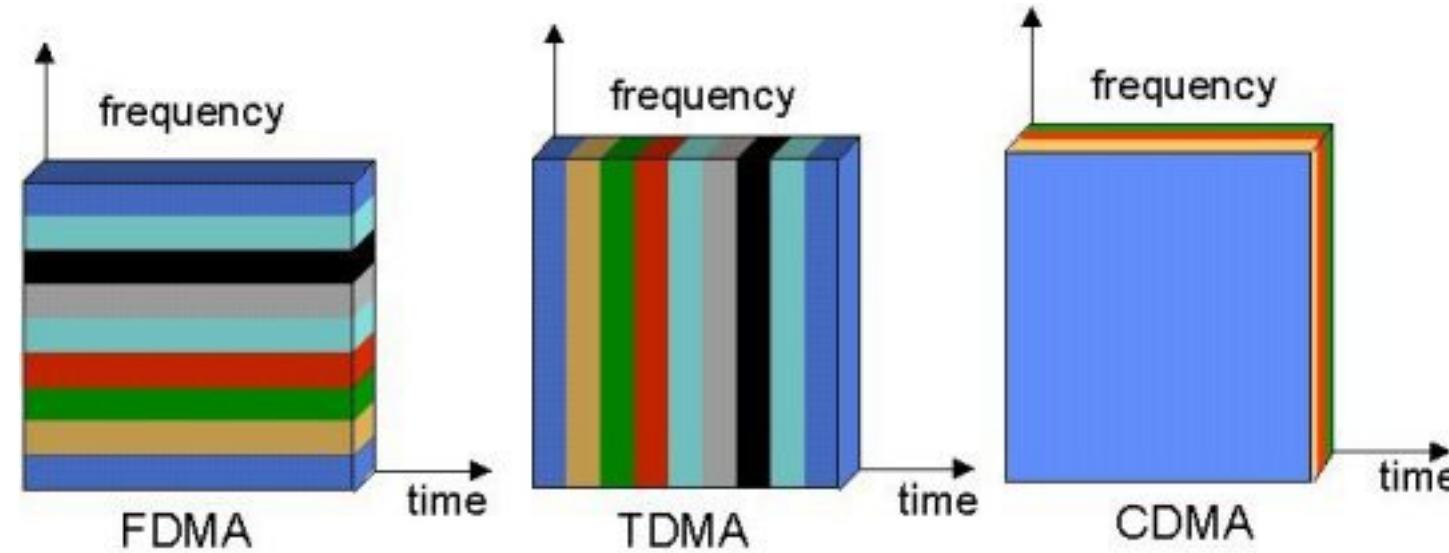
TDM



Prinzip des Wellenlängenmultiplexing (WDM)

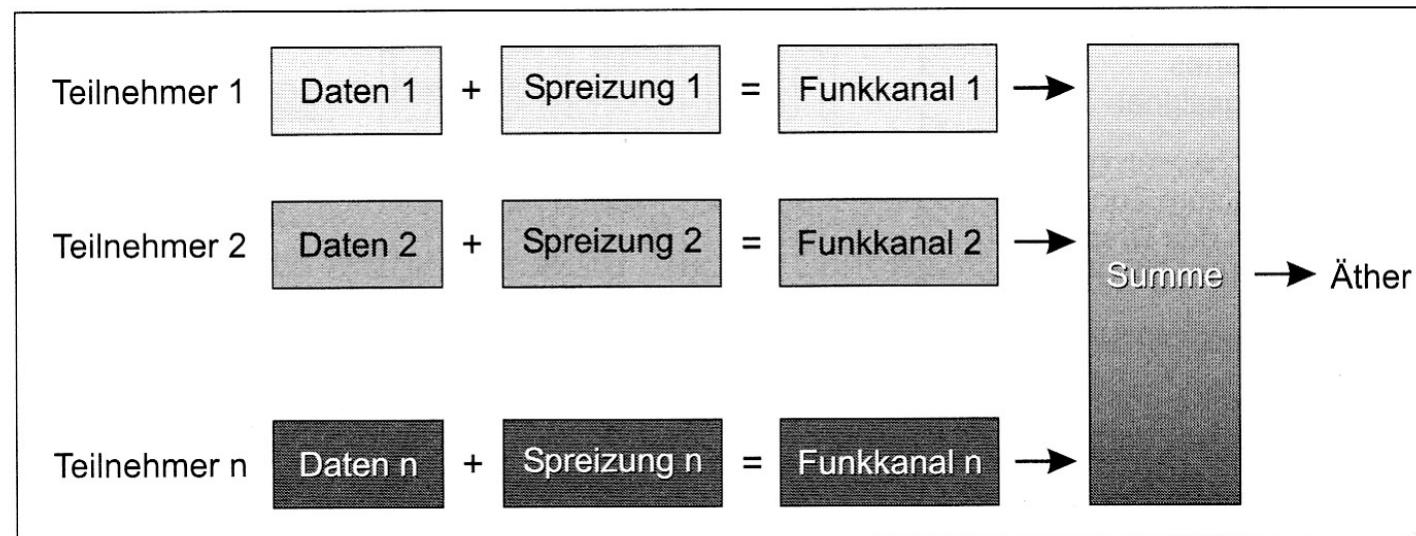


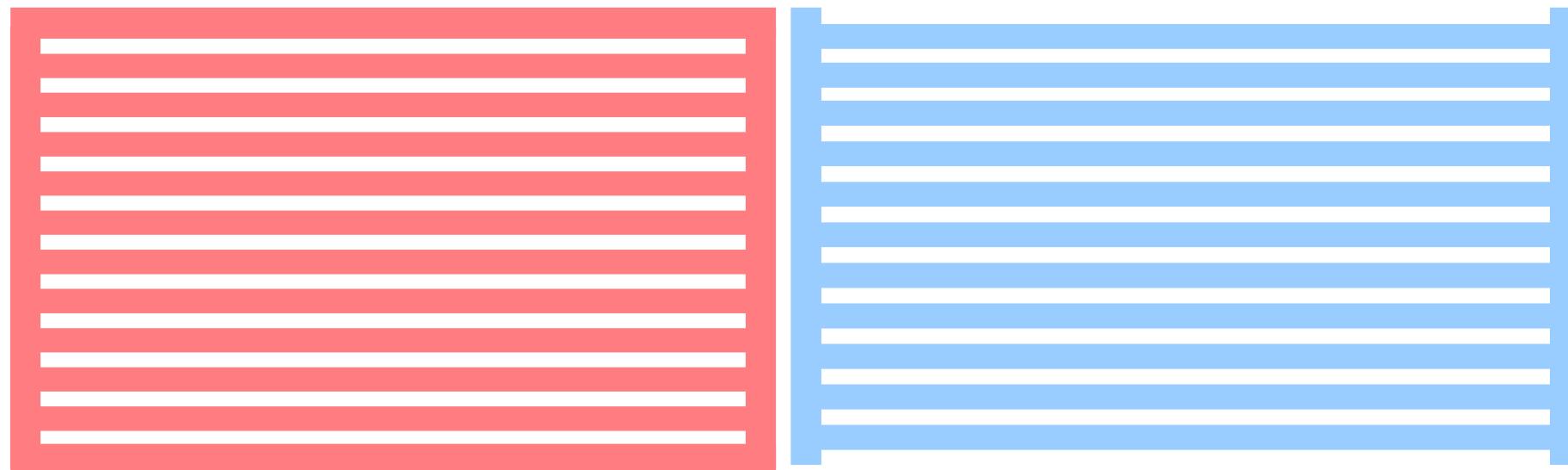
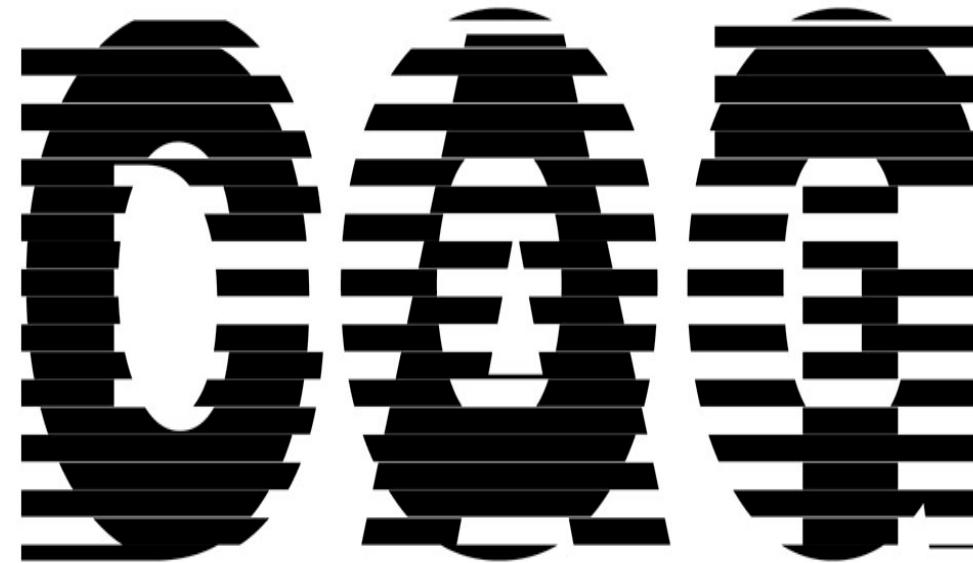
Beispiel für passives WDM (pWDM),
Weitverkehr WDM-Systeme senden mit gleicher Leistung auf allen λ

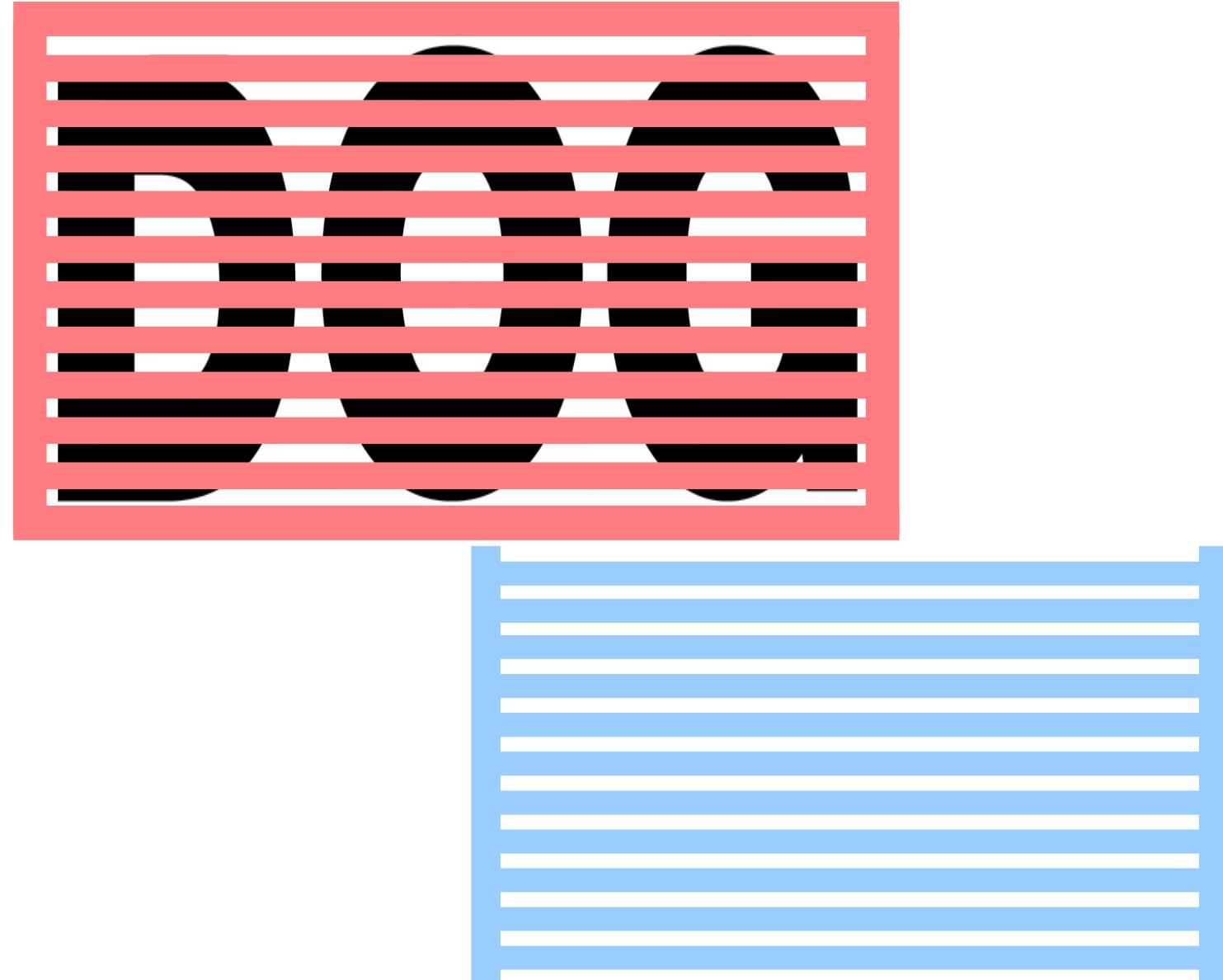


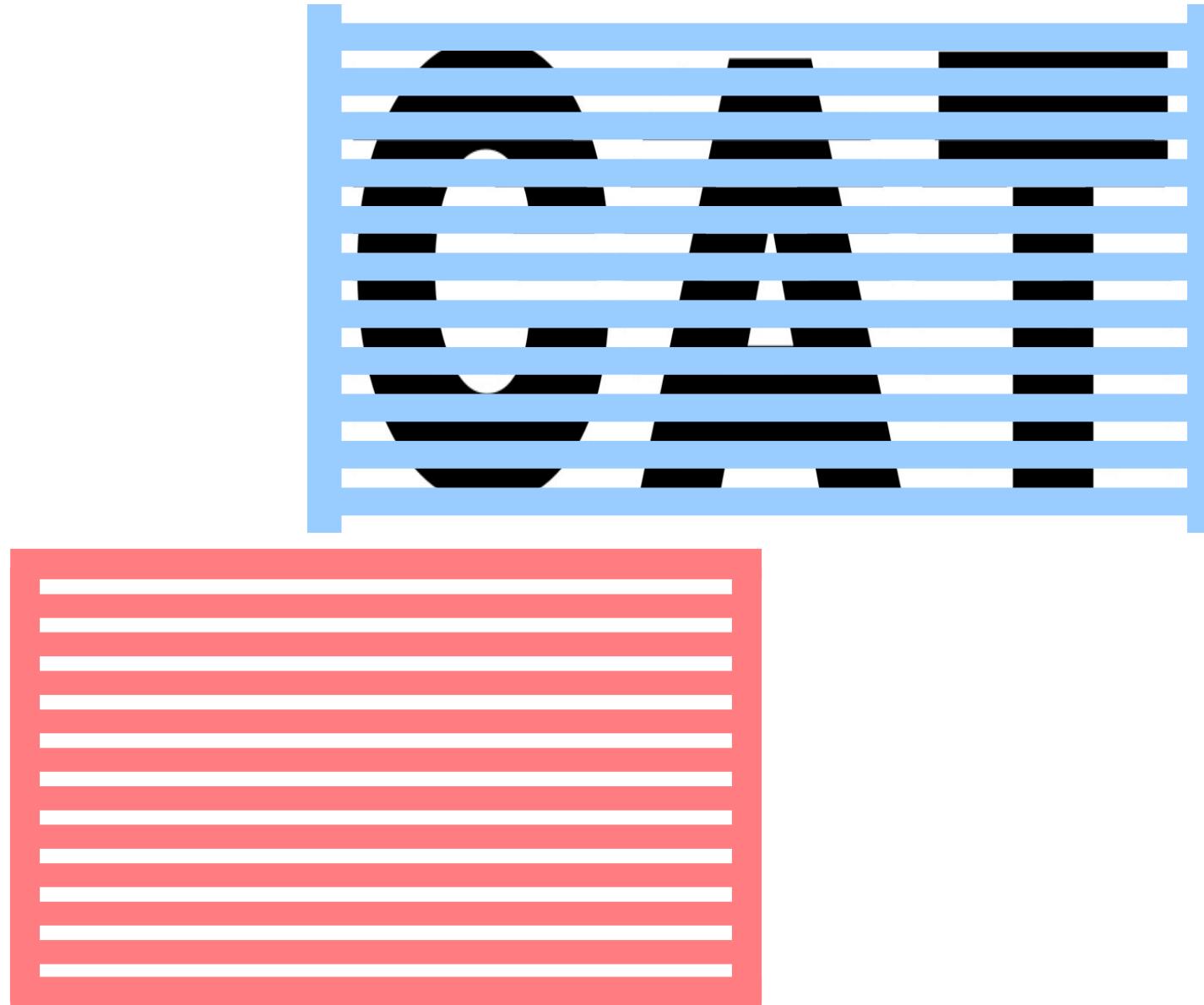
- **CDMA (Code Division Multiple Access):**
 - Trennung der Kanäle zu den einzelnen Teilnehmern durch unterschiedliche Codierung der Nutzinformation
 - Alle Stationen verwenden die gleiche Trägerfrequenz (5 MHz-Block)
 - Keine räumliche Trennung durch unterschiedliche Frequenzen

- Alle Stationen benutzen die gleiche, sehr breite Trägerfrequenz (5 MHz-Kanal)
- Die Daten werden mit unterschiedlichen Codes gespreizt und dann gemeinsam übertragen
- Die Codes der unterschiedlichen Teilnehmer sind orthogonal, d.h. die ursprünglichen Signale können mit diesem Code wieder extrahiert werden

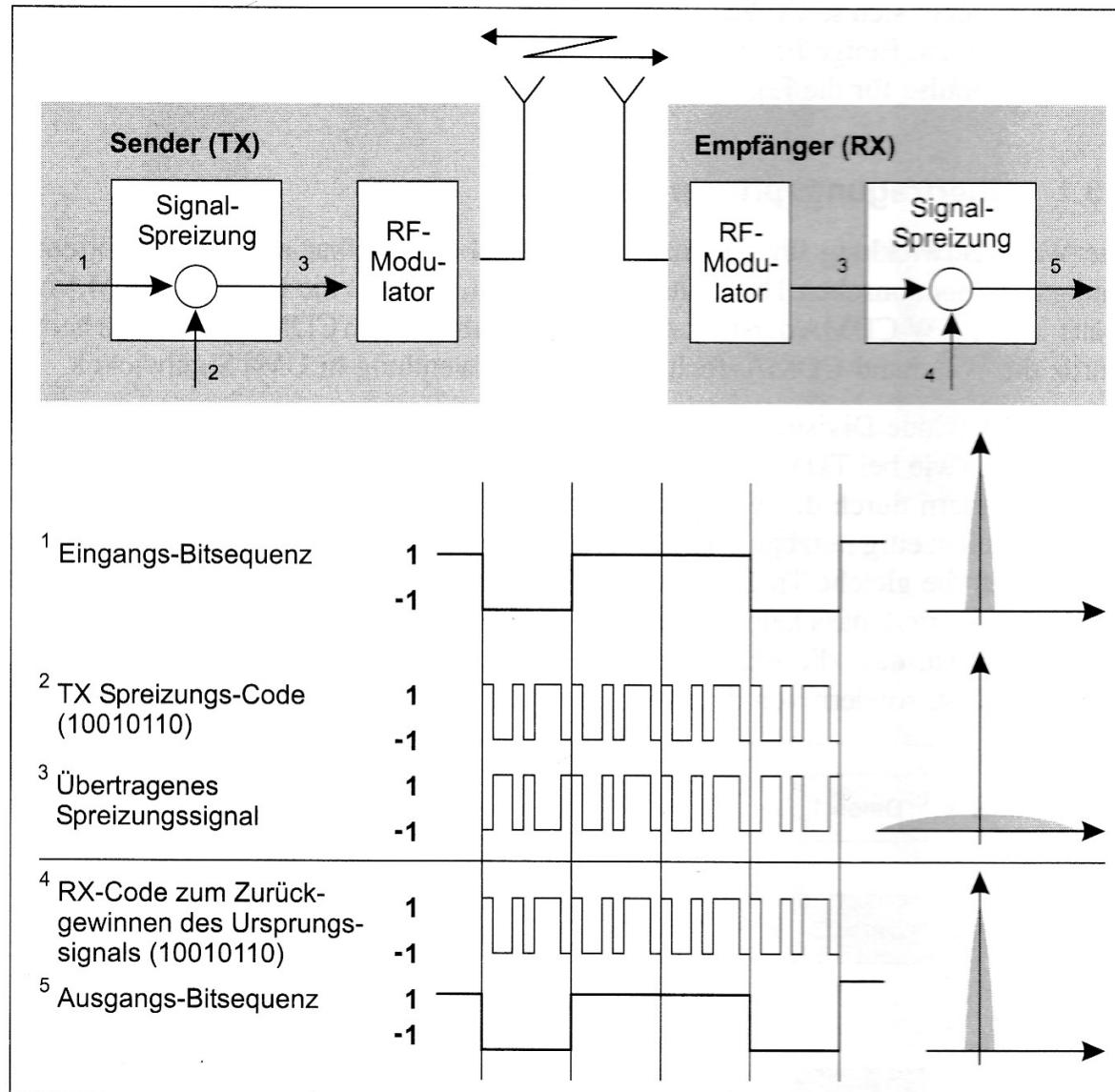






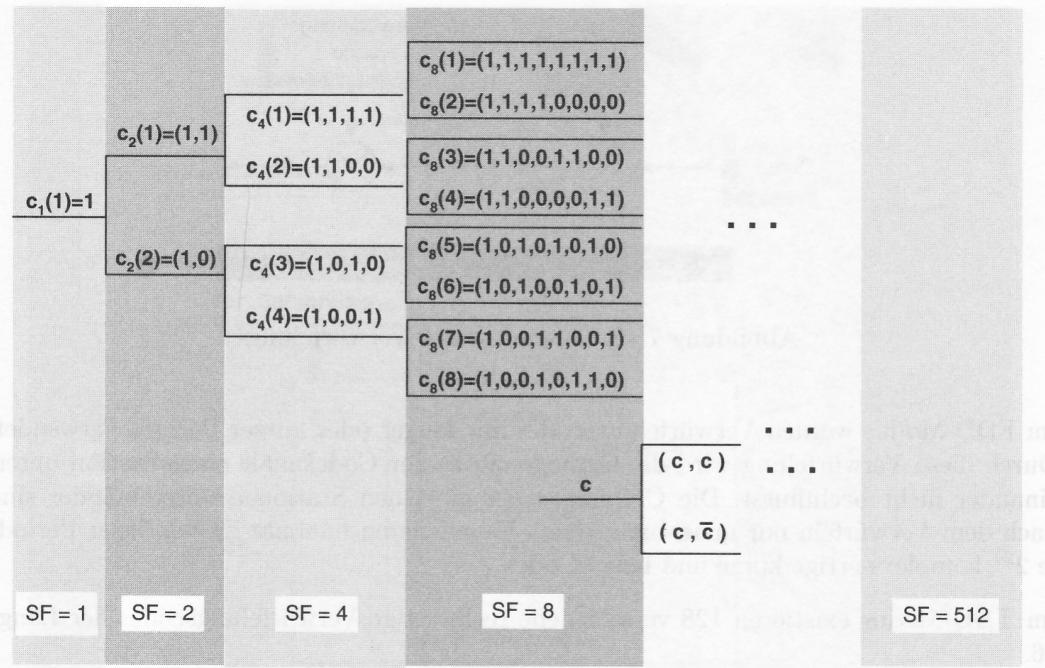


Spreizen des Signals bei CDMA



- Eingangssignal (Bits) wird mit Spreizcode (Chips) multipliziert
→ Folge von Chips
- Spreizfaktor: Chips/Bit: ~ 4...512
- Spreizfaktor (Länge des Codes) ist variable
→ Unterschiedliche Datenraten
- Spektrale Leistungsdichte des Signals wird über das ganze Band verteilt
→ Unempfindlicher gegen Störungen
- Weitere Maßnahmen zur Reduktion von Störungen (Verwürfelung, ...)
- Rel. Abhörsicher

Spreizcodes: OVSF (Orthogonal Variable Spreading Factor)

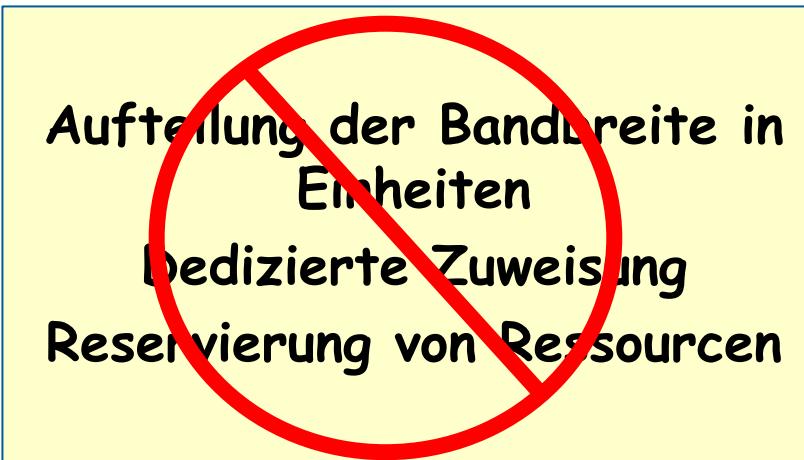


- Länge des Codes – max. 512 Chips - gibt die Bitrate des Nutzers an
Damit sind unterschiedliche Datenraten möglich
(SF: Spreizfaktor)
- Chiprate: 3,84 Mcps (chip per second) im 5 Mhz Kanal
- Bsp.: SF=8 => 384 kbit/s; SF=32 => 64 kbit/s; SF=128 => 12,2 kbit/s (Sprache)

[Quelle: Duque-Antón, 2002]

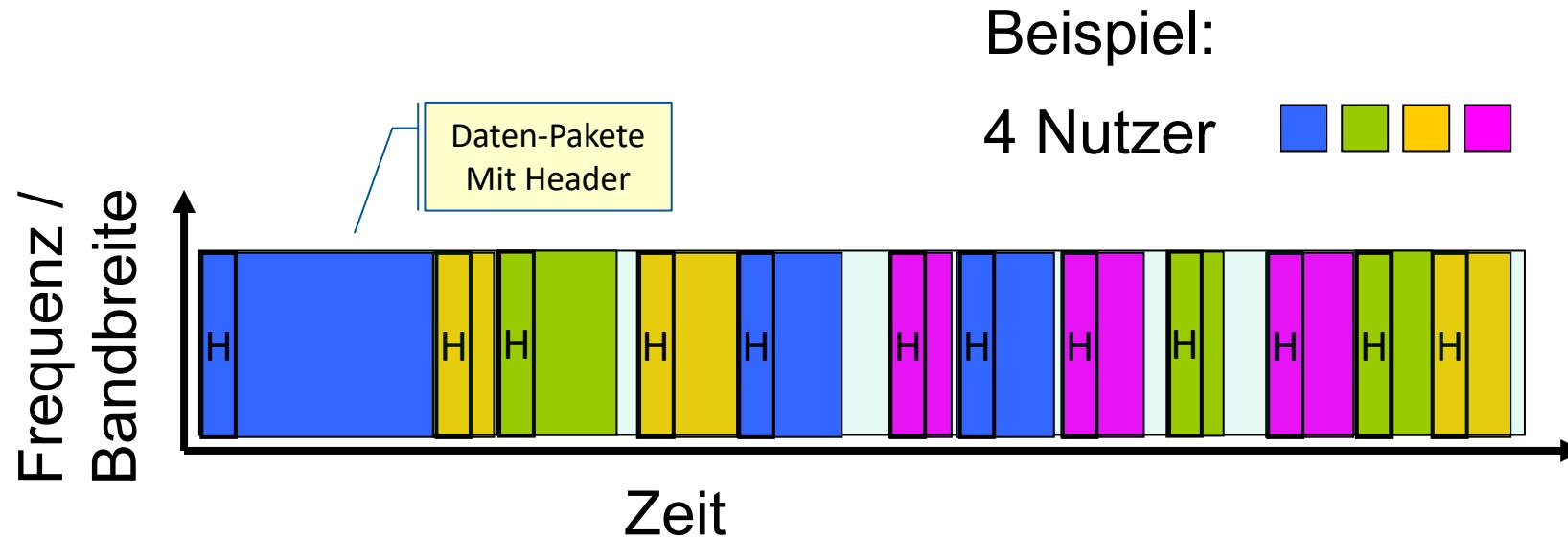
- „Urform“ der Leitungsvermittlung: Sprache / Telefonie
 - Telefon-Festnetz
 - Mobilfunk: Telefonie im 2G (GSM) und 3G (UMTS) Netz
- Leitungsvermittlung:
 - Dienstunabhängig in den Transportnetzen
 - Schalten:
 - Automatisch (Signalisierung)
 - Gesteuert (Netz-Management)
 - zur
 - Anpassung an Verkehr
 - Umgehen von Störungen / Fehlern

- Jeder Ende-zu-Ende-Datenstrom wird in **Pakete** aufgeteilt
- Die Pakete aller Nutzer **teilen** sich die Netzwerkressourcen („Leitungen“)
- Jedes Paket nutzt kurz die **volle Bandbreite** der Leitung
- Ressourcen werden **nach Bedarf** verwendet

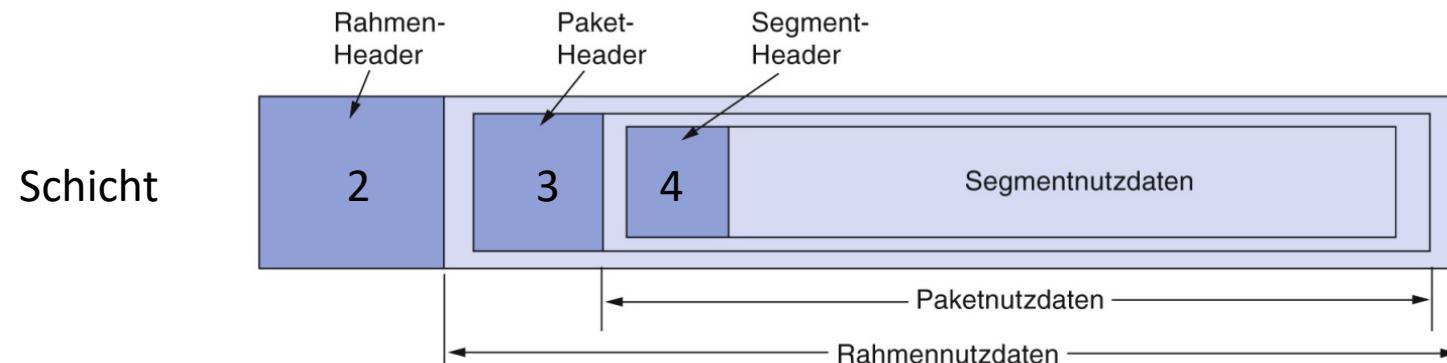


Wettbewerb um Ressourcen:

- **Die Nachfrage nach Ressourcen kann das Angebot übersteigen**
- **Überlast:** Pakete werden zwischengespeichert und warten darauf, eine Leitung benutzen zu können
- **Store and Forward:** Pakete durchqueren eine Leitung nach der anderen
 - Knoten (z.B. Router) empfangen ein komplettes Paket, bevor sie es weiterleiten
 - Dadurch entstehen zufällige Wartezeiten für das einzelne Paket.
(→ Stochastik; Wahrscheinlichkeits-rechnung und Statistik)



Daten-Paket: besteht aus dem Header (Steuerinfo, für jede Protokoll-Schicht) und der „Nutzlast“ (engl. Payload), den Daten, die zwischen den Anwendungen ausgetauscht wird. Jede Protokoll-Schicht verarbeitet und bearbeitet „ihren“ Header

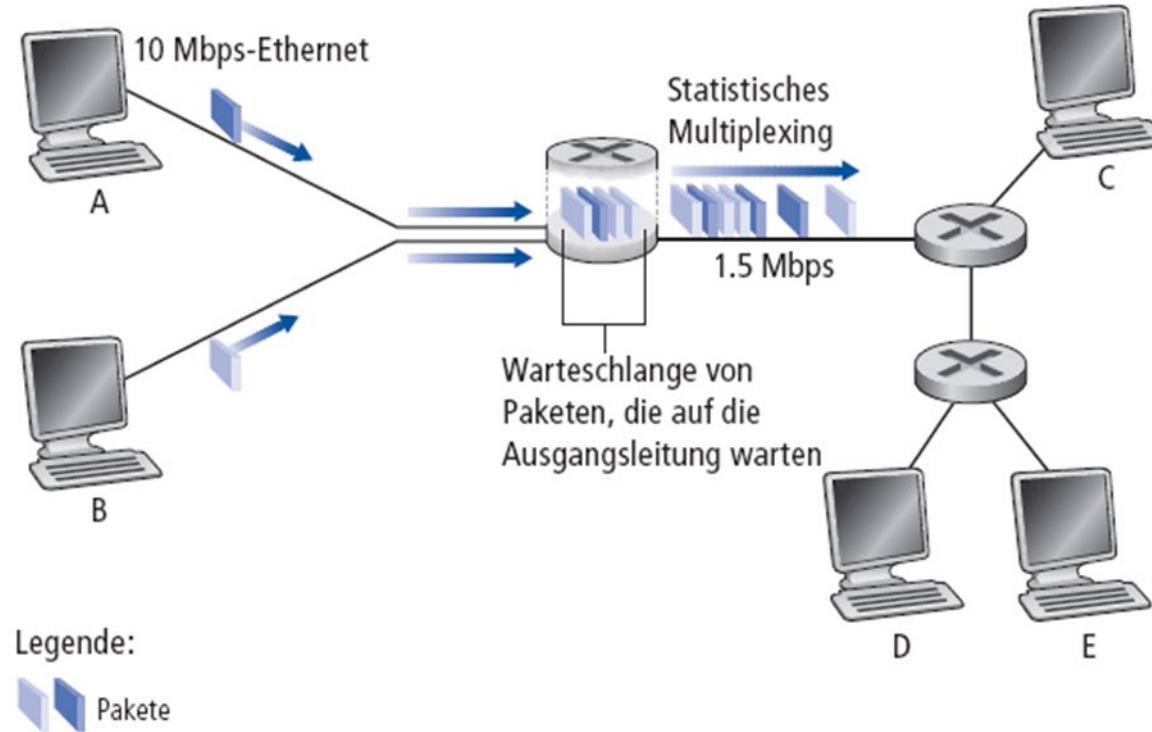


- **1 kbit/s = 1 kb/s = 1 kbps = 1.000 bit/s**
 - 1 kByte/s = 8 kbps = 8000 bit/s
- **1 Mbit/s = 1 Mbps = 1.000.000 bit/s**
- **1 Gbit/s = 1 Gbps = 1.000.000.000 bit/s**
- **Hinweis:**
 - Bei Übertragungsraten wird mit 10-er Potenzen gerechnet:
 - $k = 1.000$ (also 10^3)
 - $M = 1.000.000$ (also 10^6)
 - $G = 1.000.000.000$ (also 10^9)
 - Bei Datenvolumen (Speicherplatz) wird in 2-er Potenzen gerechnet:
 - $1 \text{ KByte} = 2^{10} = 1.024 \text{ Byte}$
 - $1 \text{ MByte} = 2^{20} = 1.048.576 \text{ Byte}$
 - $1 \text{ GByte} = 2^{30} = 1.073.741.824 \text{ Byte}$

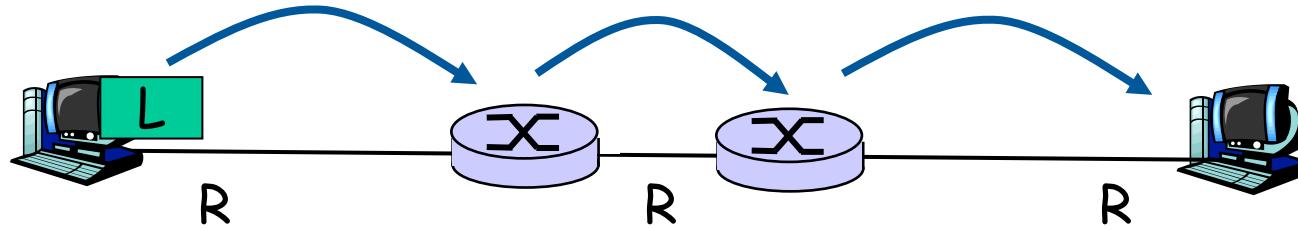


*Internet = Paketvermittlung:
→Der Zufall regiert!
(Stochastik)*

Die **Stochastik** (von altgriechisch *stochastikē technē*, also ‚Kunst des Vermutens‘, ‚Ratekunst‘) ist ein Teilgebiet der Mathematik und fasst als Oberbegriff die Gebiete **Wahrscheinlichkeitstheorie** und **Statistik** zusammen.



- Die Folge von Paketen auf der Leitung hat kein festes Muster, die Bandbreite wird nach Bedarf verteilt
- ➔ **statistisches Multiplexing**
- **Zum Vergleich bei TDM:** Jede Verbindung erhält immer den gleichen Zeitrahmen in einem sich wiederholenden Muster



- Es dauert L/R Sekunden, um ein Paket von L Bits auf einer Leitung mit der Rate R Bit/s zu übertragen
- Store and Forward: Das gesamte Paket muss bei einem Router angekommen sein, bevor es auf der nächsten Leitung übertragen werden kann
- Verzögerung = $3 L/R$
(wenn die Ausbreitungsverzögerung (Laufzeit) vernachlässigt wird)

L = Paketgröße (bit)

R = Bandbreite der Leitung (Bit/s)
auch: Übertragungsrate

Beispiel:

- $L = 7,5 \text{ Mbit}$
- $R = 1,5 \text{ Mbit/s}$
- Übertragungszeit = 15 s

- **Paketvermittlung nutzt die Ressourcen häufig besser aus!**

- **Beispiel: Leitung mit 1 Mbit/s**

- **Jeder Benutzer:**

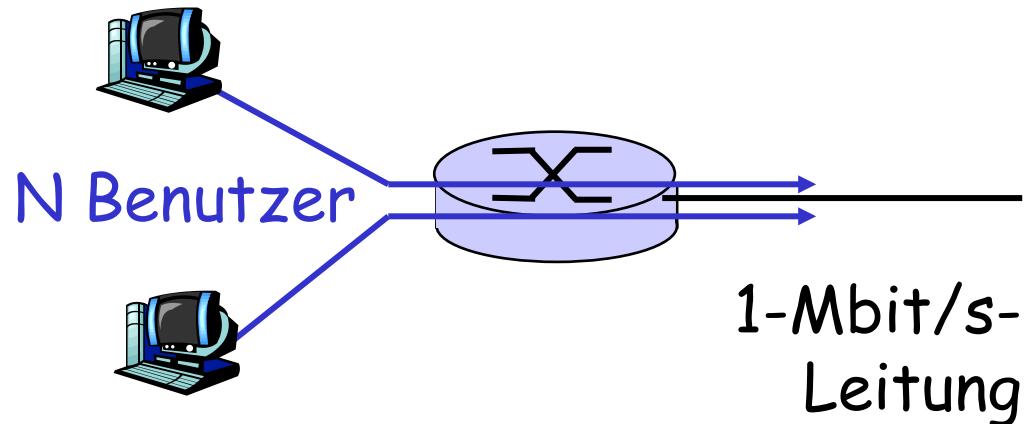
- 100 kbit/s, wenn aktiv
- 10% der Zeit aktiv

- **Leitungsvermittlung:**

- 10 Benutzer möglich
- 11. Benutzer? => Besetzt!

- **Paketvermittlung:**

- Mit 35 Benutzern
ist die Wahrscheinlichkeit
für mehr als 10 aktive Benutzer zur
gleichen Zeit kleiner als 0,0004

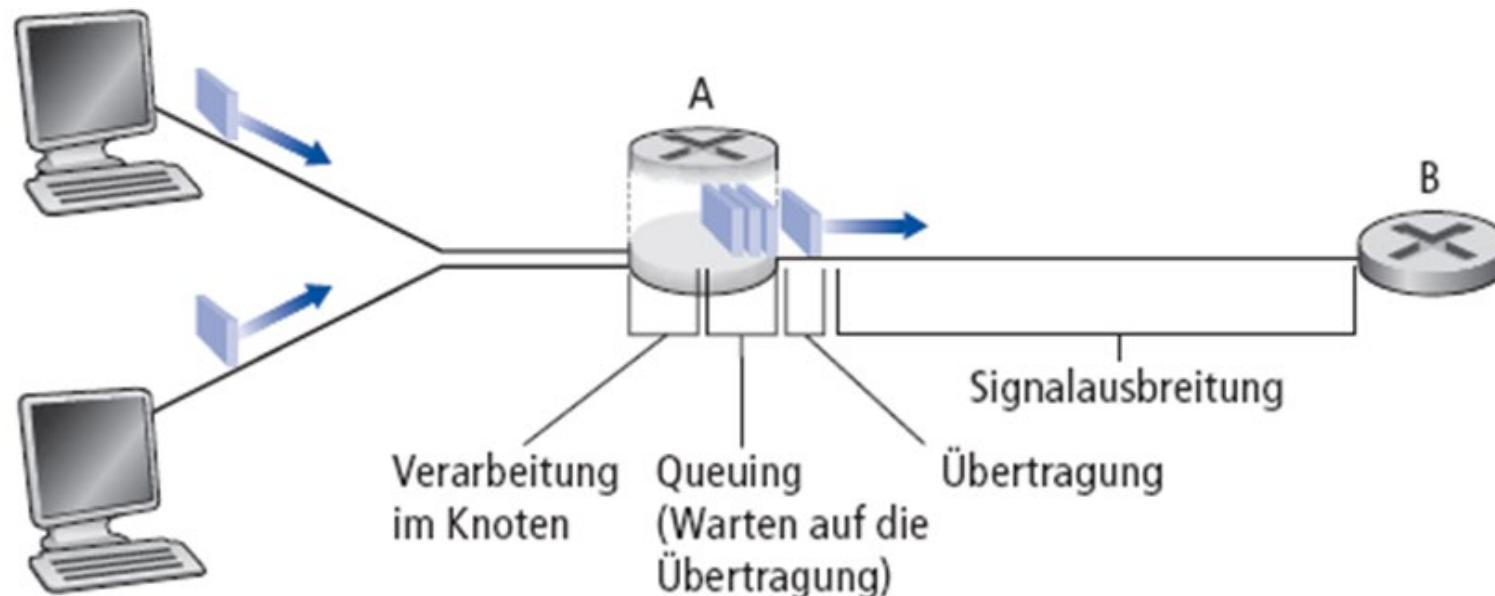


Ist Paketvermittlung generell besser?

- Sehr gut für unregelmäßigen Verkehr (**bursty traffic**)
 - Gemeinsame Verwendung von Ressourcen
 - Einfacher, keine Reservierungen
- **Problem Überlast:** Verzögerung und Verlust von Paketen
 - Protokolle für zuverlässigen Datentransfer und Überlastkontrolle werden benötigt
- Frage: Wie kann man leitungsähnliches Verhalten bereitstellen?
 - Bandbreitengarantien werden gebraucht für Audio- und Videoanwendungen
 - Logische (Virtuelle) Verbindungen mit reservierten Ressourcen (Bandbreite) → Steuerung
 - Oder: „schmeiß Bandbreite auf das Problem“ → Überdimensionierung

Grundlagen Paketvermittlung:
→ *Verzögerungen*
und Verluste

- Pakete warten in den Puffern von Routern
- ... wenn die **momentane** Ankunftsrate die Kapazität der Ausgangsleitungen übersteigt

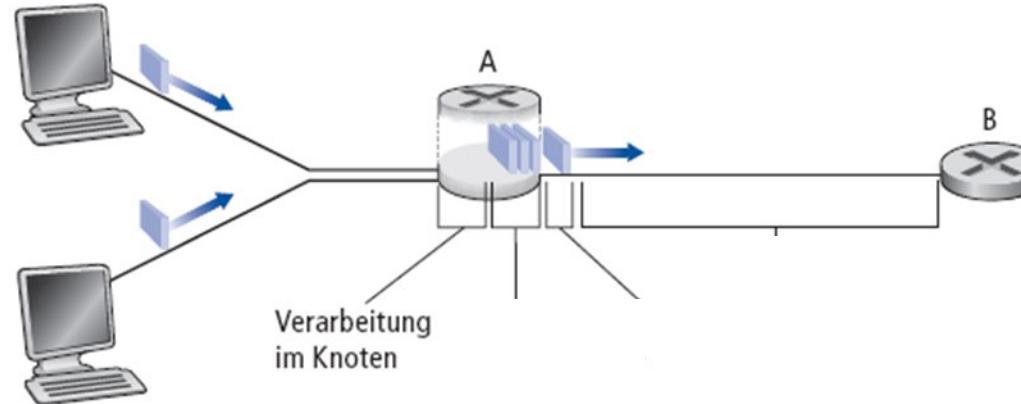


1. Verarbeitung im Knoten:

- Auf Bitfehler prüfen
- Auswertung der Adresse
- Wahl der ausgehenden Leitung

2. Warten auf die Übertragung

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



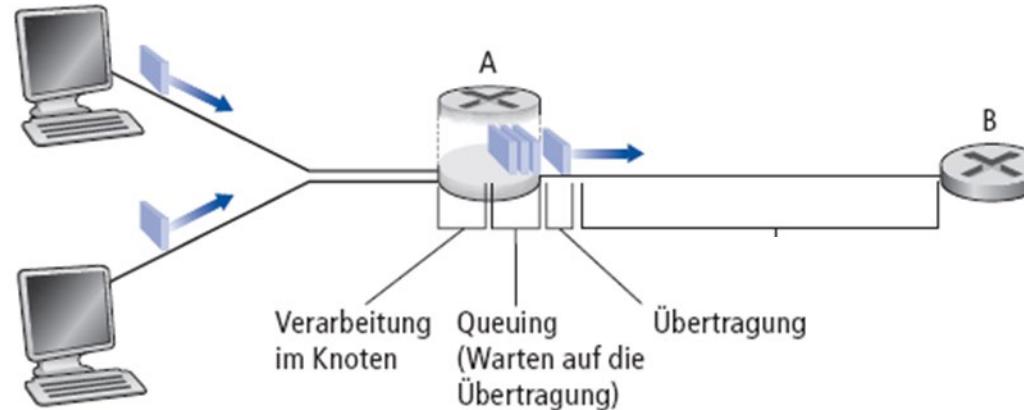
3. Übertragungsverzögerung:

- R = Bandbreite einer Leitung (Bit/s)
- L = Paketgröße (Bit)
- **Übertragungsverzögerung = L/R**

(auch Übertragungszeit)

4. Ausbreitungsverzögerung

- d = Länge der Leitung
- s = Ausbreitungsgeschwindigkeit des Mediums ($\sim 2 \dots 3 \times 10^8$ m/s)
~ „Lichtgeschwindigkeit“ auf Medium
- **Ausbreitungsverzögerung = d/s**



3. Übertragungsverzögerung:

- R = Bandbreite einer Leitung (Bit/s)
- L = Paketgröße (Bit)
- **Übertragungsverzögerung = L/R**

(auch Übertragungszeit)

4. Ausbreitungsverzögerung

- d = Länge der Leitung
- s = Ausbreitungsgeschwindigkeit des Mediums ($\sim 2 \dots 3 \times 10^8$ m/s)
~ „Lichtgeschwindigkeit“ auf Medium
- **Ausbreitungsverzögerung = d/s**

(auch Laufzeit)

Beispiel:

Wie groß ist die Ausbreitungsverzögerung (Laufzeit hin und zurück) für ein Signal quer durch Deutschland?

Antwort:

$d = 1.000 \text{ km} * 2$ (Hin und zurück)

$d/s = 2.000.000 \text{ m} / 2 \cdot 10^8 \text{ m/s} = 10^{-2} \text{ s} = 10 \text{ ms}$

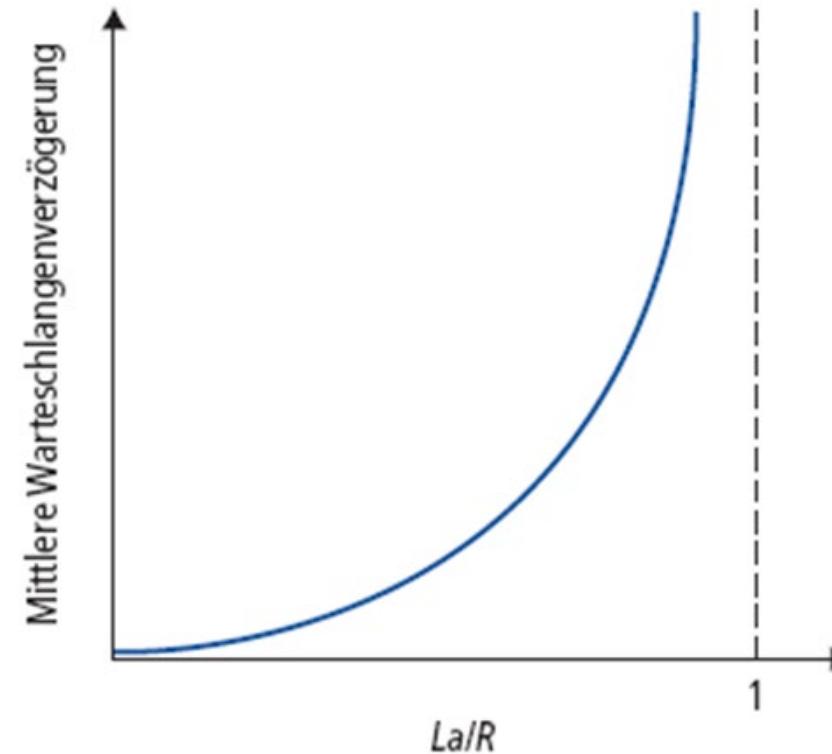
$$d_{\text{gesamt}} = d_{\text{Verarbeitung}} + d_{\text{Warten}} + d_{\text{Übertragung}} + d_{\text{Ausbreitung}}$$

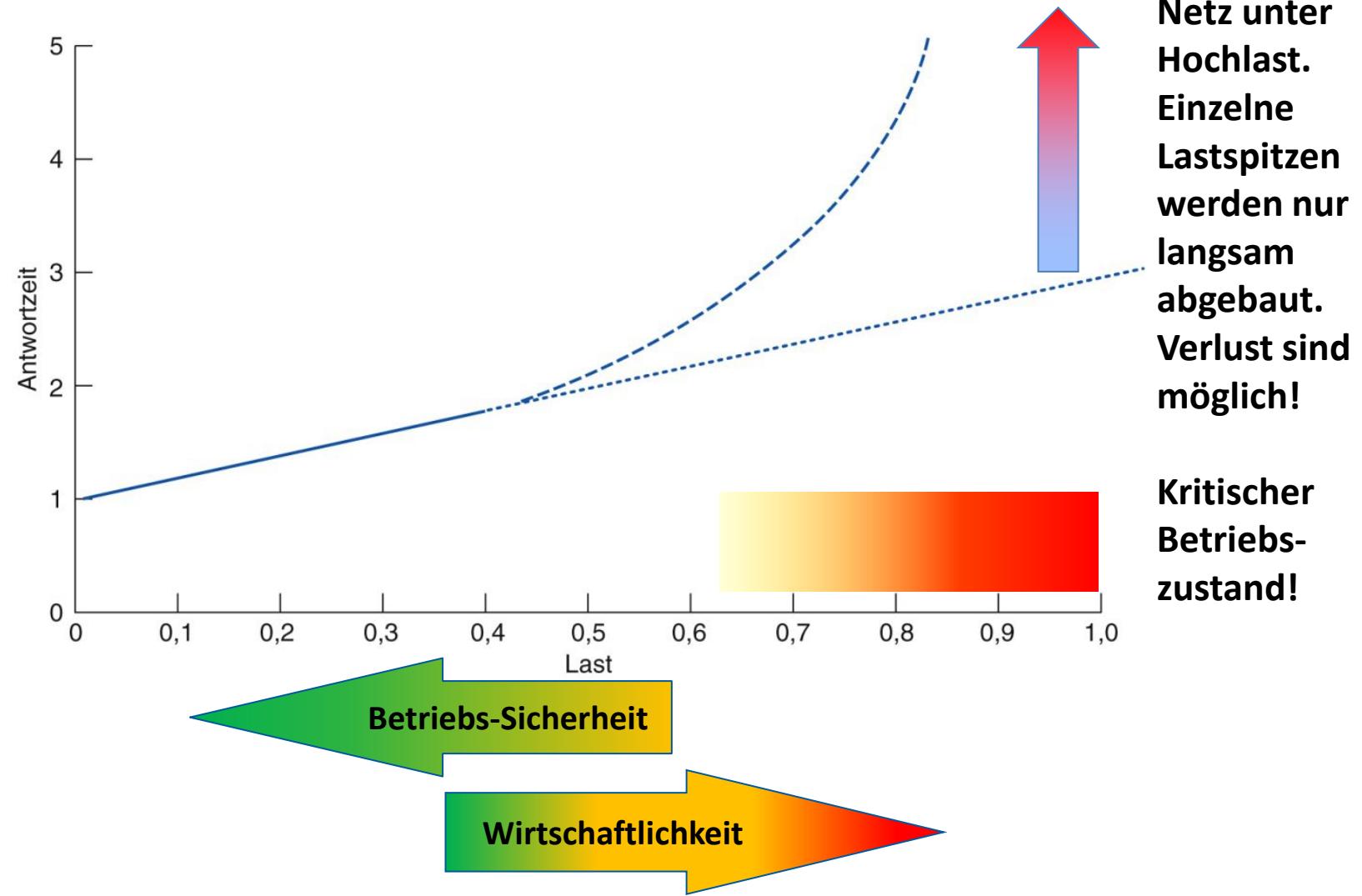
- **$d_{\text{Verarbeitung}}$ = Verarbeitungsverzögerung**
 - Üblicherweise wenige Mikrosekunden oder weniger
 - Oft Abhängig von der Auslastung des Servers / Rechners
- **d_{Warten} = Wartezeit in Puffern**
 - Abhängig von der aktuellen Überlastsituation
- **$d_{\text{Übertragung}}$ = Übertragungsverzögerung (Übertragungszeit)**
 - = L/R , signifikant wenn R klein ist
(L = Paketgröße in Bit, R = Bandbreite der Leitung (Bit/s))
- **$d_{\text{Ausbreitung}}$ = Ausbreitungsverzögerung (Laufzeit)**
 - Wenige Mikrosekunden bis einige hundert Millisekunden
 - = d / s ; Länge der Leitung / Ausbreitungsgeschwindigkeit des Mediums ($\sim 2 \dots 3 \cdot 10^8$ m/s)

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

Verkehrswert = La/R

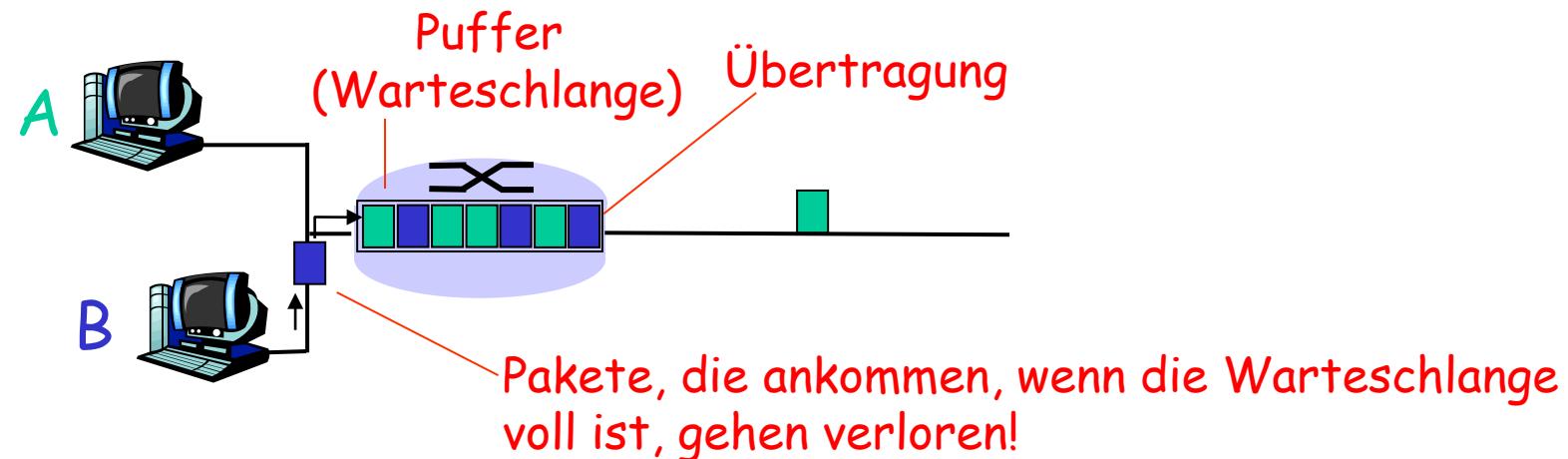
- $La/R < 0,6$: Wartezeit gering
- $La/R \rightarrow 1$: Wartezeit steigt stark an
- $La/R > 1$: durchschnittliche Wartezeit ist unendlich!

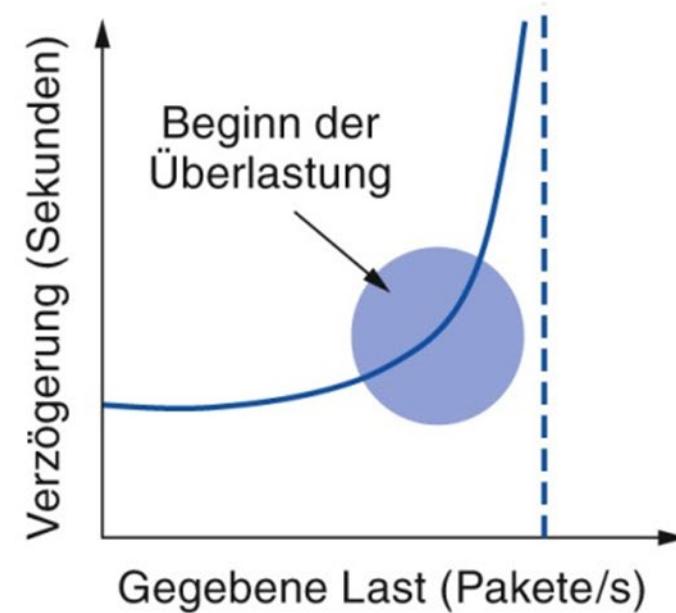
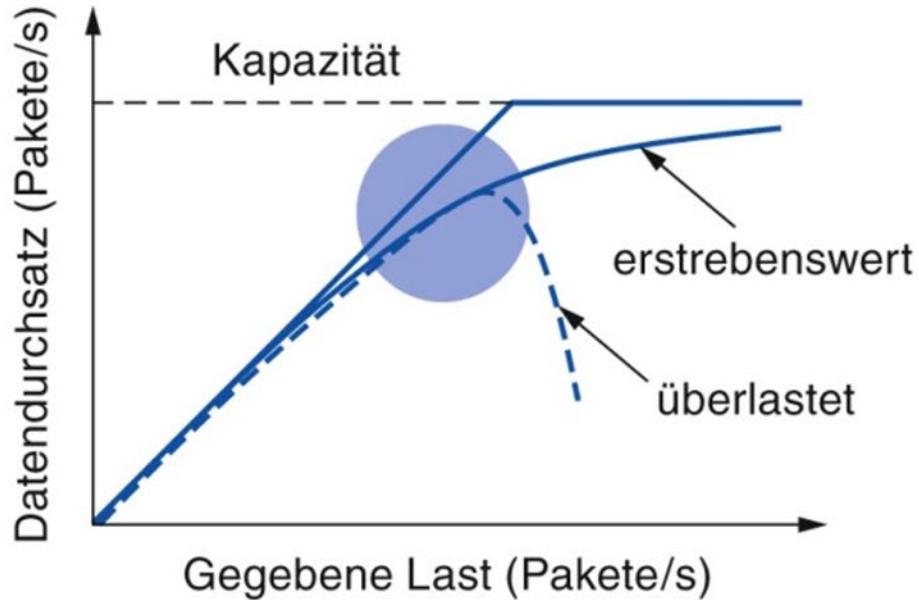




- **Warteschlange** vor einer Ausgangsleitung hat nur endlich viele **Pufferplätze**
- Wenn alle Pufferplätze belegt sind, werden neu ankommende Pakete verworfen:
→ **Paketverlust!**
- Verlorene Pakete können vom vorangegangenen Knoten oder vom Sender erneut übertragen werden – oder auch nicht!

Das ist Aufgabe der entsprechenden Protokolle





Viele Angriffe versuchen Teile des Netz in die Überlast zu treiben (z.B. Denial of Service Attacken)

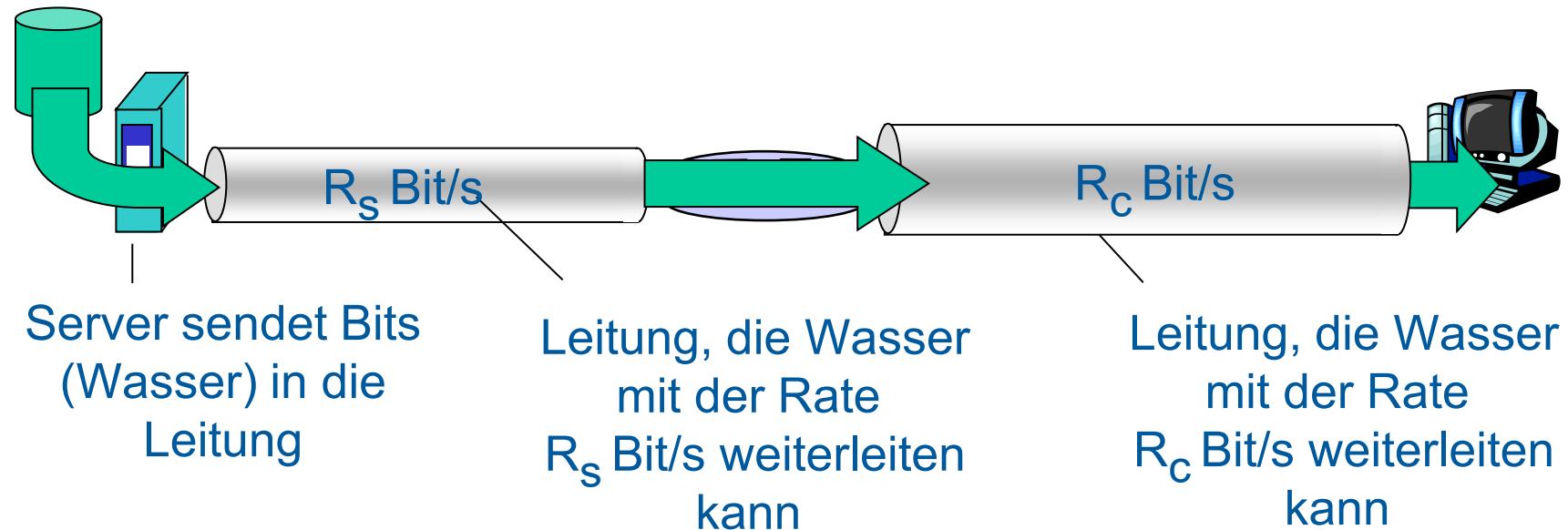
Und damit den Verkehr stark zu behindern

Wichtig: gute Überlastkontrolle in den Knoten („erstrebenswertes Verhalten“)

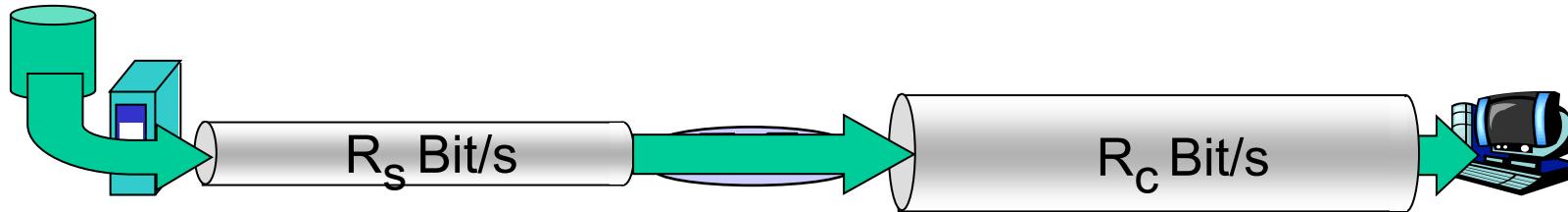
Problem: Bei hohen Verzögerungen laufen die Überwachungstimer ab und Pakete werden wiederholt => Die Überlast verstärkt sich!

[Tanenbaum, 2012]

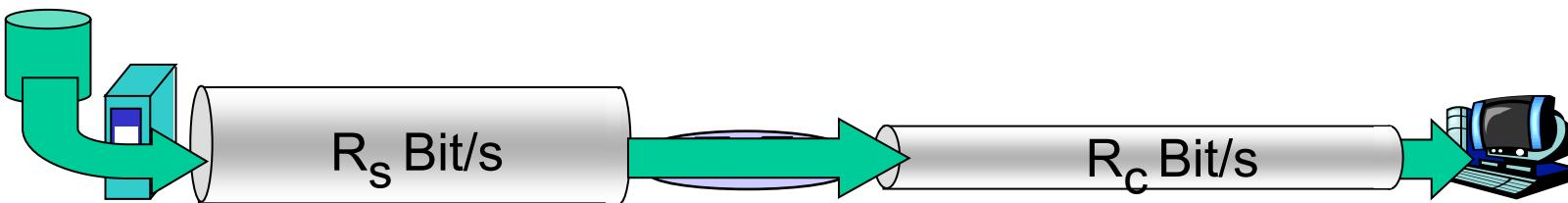
- **Durchsatz:** Rate (Bit/Zeiteinheit), mit der Daten zwischen Sender und Empfänger ausgetauscht werden
 - Unmittelbar: Rate zu einem gegebenen Zeitpunkt
 - Durchschnittlich: Rate über einen längeren Zeitraum
 - Analogie: Wasserleitung



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



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



- **Engpass einer Verbindung:**

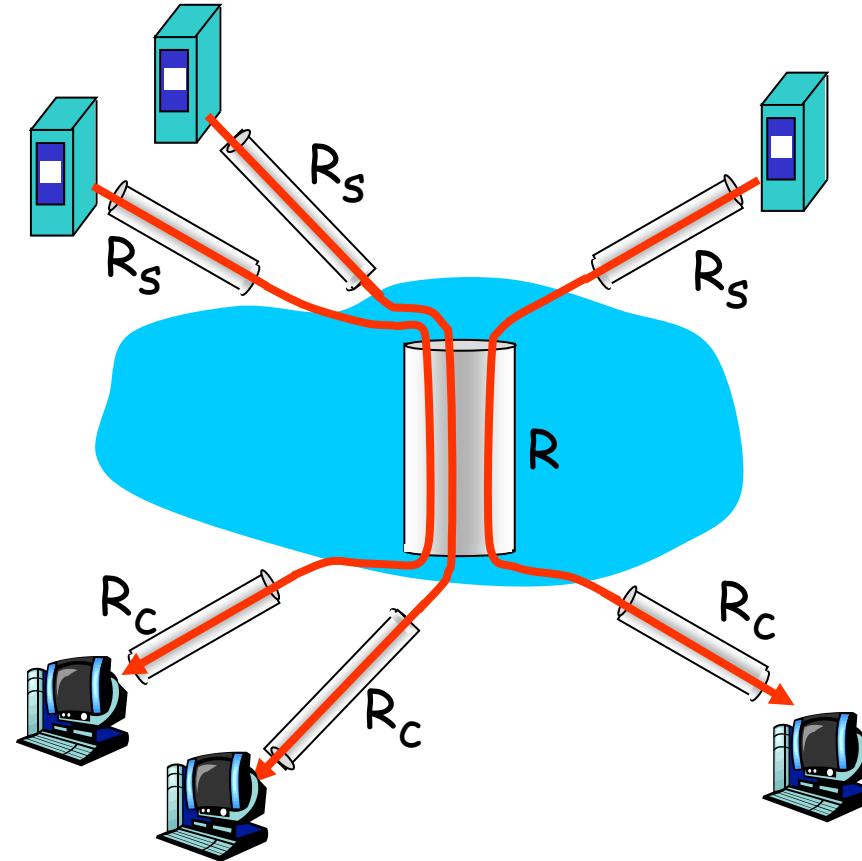
Der Abschnitt auf dem Ende-zu-Ende Pfad mit dem geringsten Durchsatz, der also den Ende-zu-Ende Durchsatz begrenzt

[Quelle: Kurose, J.; Ross, K.: Computernetzwerke : der Top-Down-Ansatz]

- Durchsatz für Ende-zu-Ende-Verbindungen:

$$\min(R_c, R_s, R/10)$$

- In der Realität:
Häufig sind R_c oder R_s die Engpässe
(Zugangsnetz / Access)

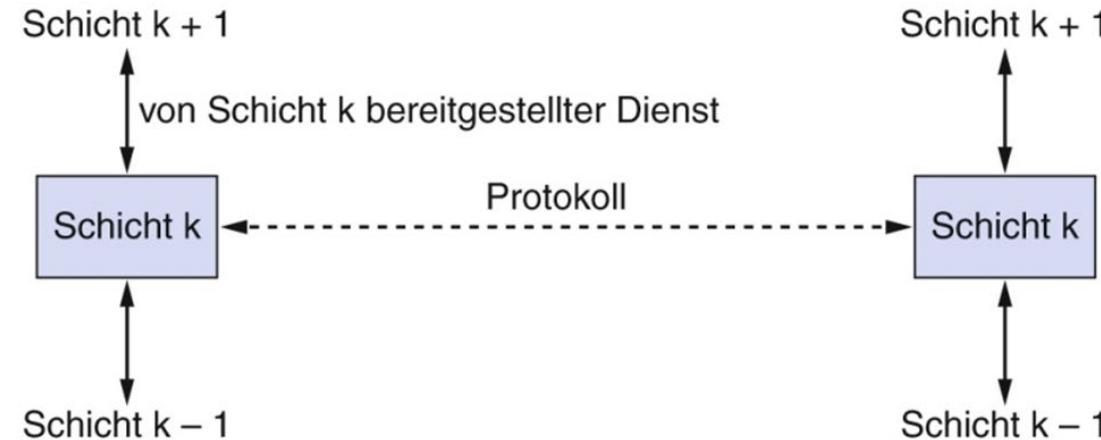
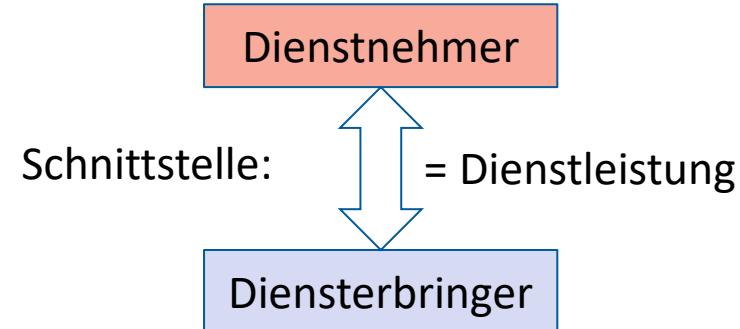


10 Verbindungen teilen sich den Engpass des Backbone-Netzwerkes

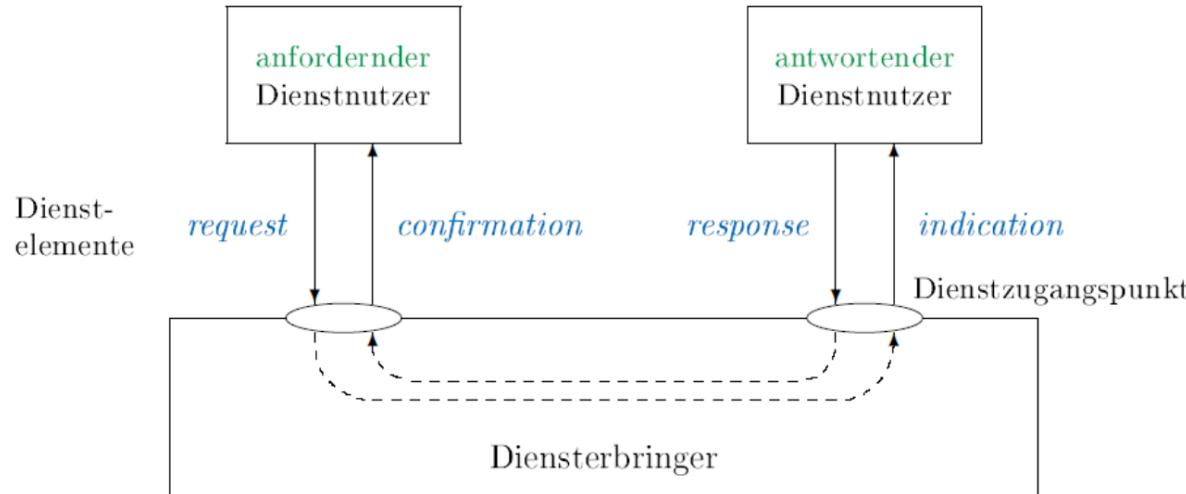
Internet = Paketvermittlung:
→ *Die Kommunikation braucht
feste Regeln*

Fundamentales Konzept:

- Dienste
- Schnittstellen
- Protokolle



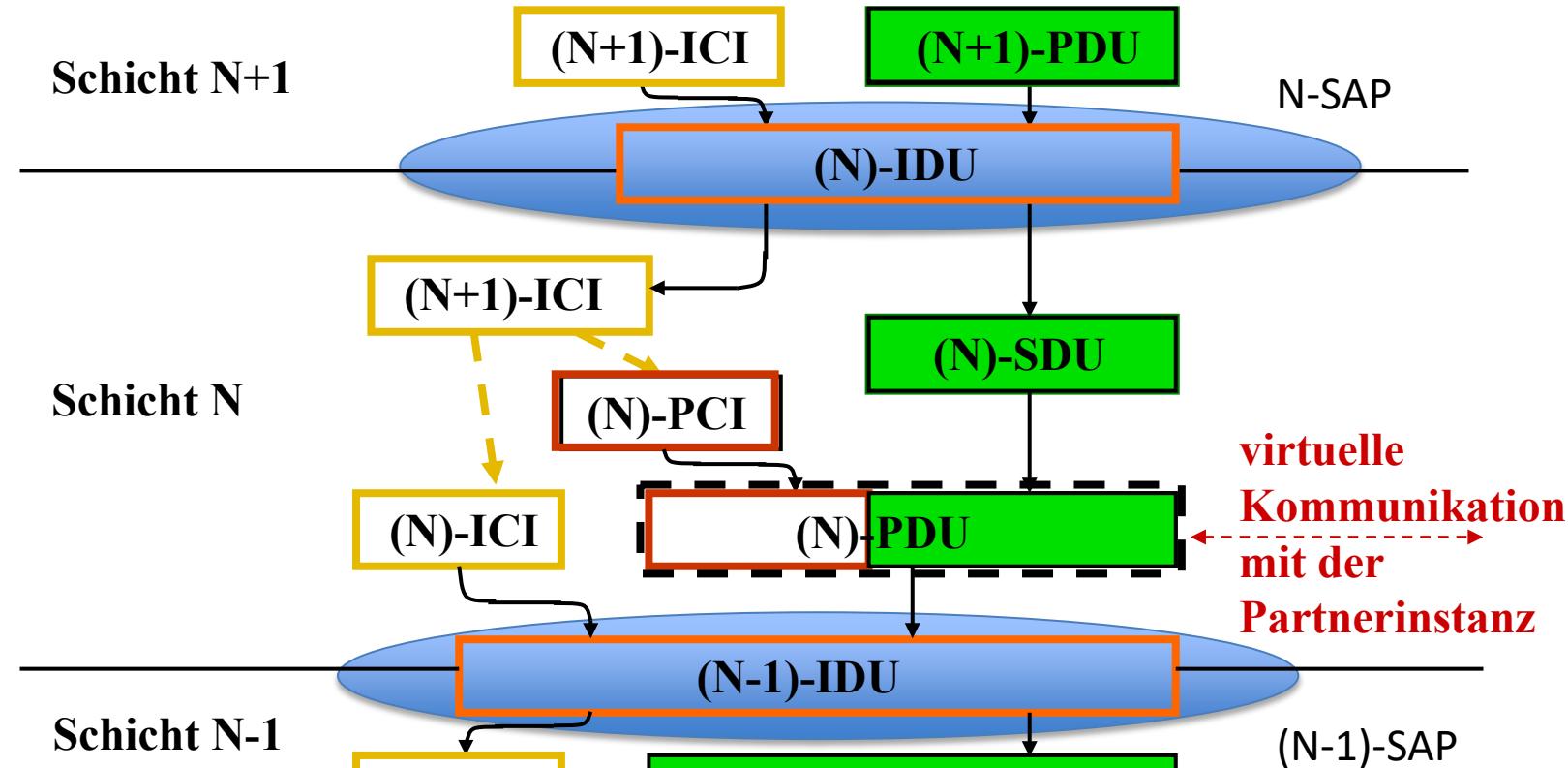
Bedeutung der Dienst-Primitive



| Primitiv | Bedeutung |
|---------------------|---|
| request | vom Dienstnutzer ausgelöst, um einen Dienst anzufordern und die Parameter zu übergeben, die diesen Dienst spezifizieren |
| indication | vom Diensterbringer ausgelöst, <ol style="list-style-type: none">um anzuzeigen, dass ein entsprechender Dienst von der Partnerinstanz des Dienstnutzers angefordert wurde und um die spezifizierenden Parameter zu übergeben oderum den Dienstnutzer von einer vom Diensterbringer selbst ausgelösten Aktion in Kenntnis zu setzen |
| response | vom Dienstnutzer ausgelöst, um eine per indication angezeigte Diensterbringung zu beantworten oder zu beenden |
| confirmation | vom Diensterbringer ausgelöst, um eine per request vom Dienstnutzer ausgelöste Dienstanforderung zu bestätigen oder zu beenden |

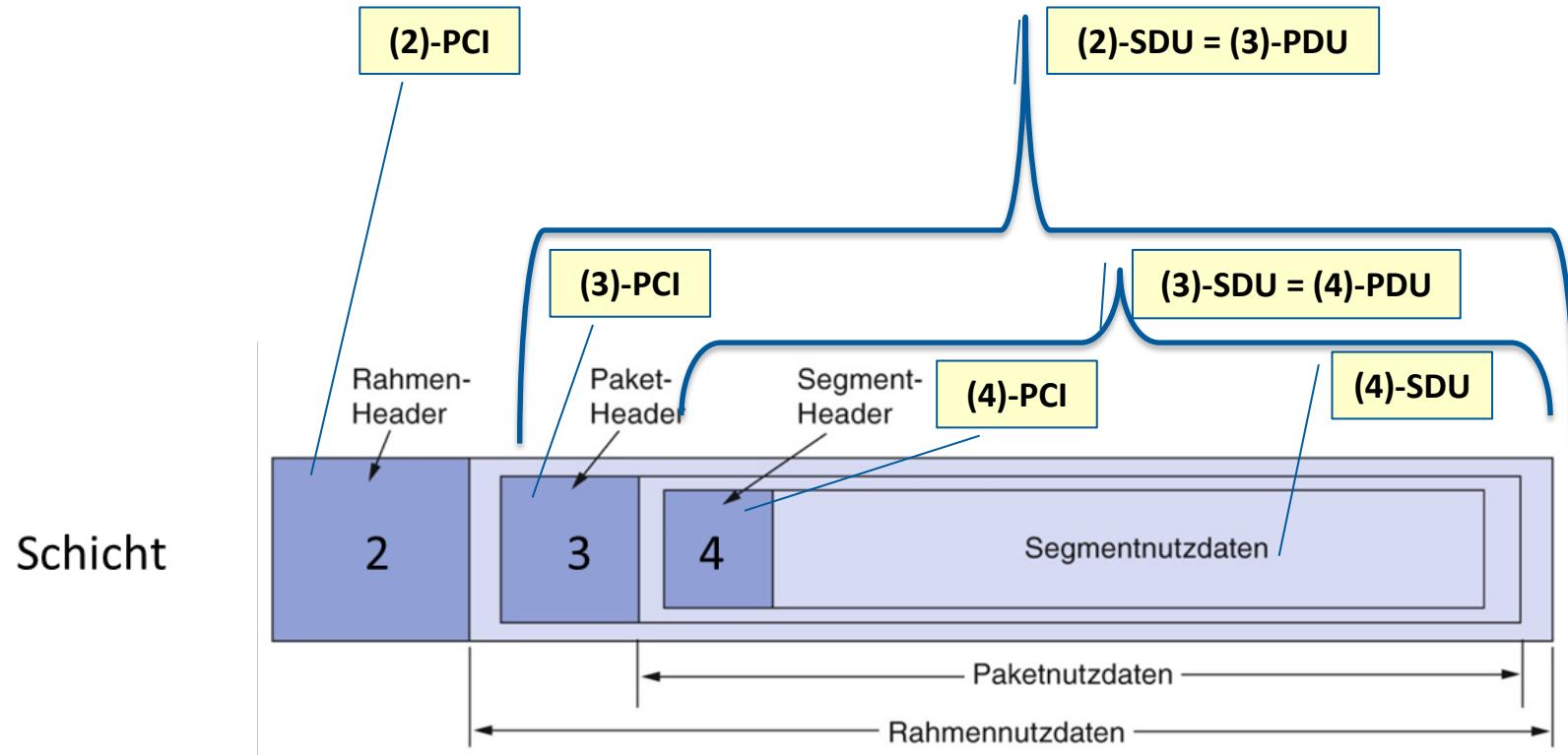
- **Klar definierte Schnittstellen:**
 - Dienst, der dem Nutzer erbracht wird (Schicht N erbringt Schicht N+1 einen Dienst)
 - Dienst, der von dieser Schicht benutzt wird (Schicht N benutzt den Dienst der Schicht N-1)
 - Protokoll der Schicht N (Zwischen den beiden Instanzen der Schicht N)
- **Funktionen:**
 - Das Protokoll ist für die Funktion der Schicht N verantwortlich.
 - Steuerinformation der Schicht N wird im Header der Schicht N übertragen (z.B. Adressen)
(N-PCI, Protocol Control Information der Schicht N)
- **Daten:**
 - Die Nutzdaten der Schicht N: N-SDU (Service Data Unit der Schicht N)
 - Dazu kommt die Steuerinformation der Schicht N: N-PCI (Protocol Control Information der Schicht N)
 - N-PCI + N-SDU => N-PDU
- **Übertragung:**
 - N-PDU wird logisch an die andere Instanz (Empfänger) gesendet
 - Dazu wird die N-PDU der darunterliegenden Schicht N-1 übergeben als (N-1)-SDU

Bildung der Nachrichtenpakete (Data Unit)



ICI: Interface Control Information
IDU Interface Data Unit

SDU: Service Data Unit
PDU: Protocol Data Unit
PCI: Protocol Control Information



- Prinzip: „Einwickeln“ der Daten der Schicht (N) mit Header in Schicht (N-1)
- (N)-PDU \rightarrow (N-1)- SDU
- (N)- SDU + (N)- PCI \rightarrow (N)-PDU
- Nutzdaten + Header \rightarrow „Paket“

SDU: Service Data Unit
 PDU: Protocol Data Unit
 PCI: Protocol Control Information

Internet = Paketvermittlung:
→ *Die Kommunikation braucht
feste Regeln...*
→ *... und eine feste Struktur*

- **Anwendungsschicht:**

Unterstützung von Netzwerkanwendungen

- FTP, SMTP, HTTP, ...

- **Transportschicht:**

Datentransfer zwischen Prozessen (E2E)

- TCP, UDP

- **Netzwerkschicht (auch Vermittlungsschicht):**

Weiterleiten der Daten von einem **Sender**

zu einem **Empfänger (Netzweit)**

- IP, Routing-Protokolle

- **Sicherungsschicht:**

Datentransfer zwischen benachbarten

Netzwerksystemen

- PPP, Ethernet

- **Bitübertragungsschicht: Bits auf der Leitung**



■ Darstellungsschicht:

Ermöglicht es Anwendungen, die Bedeutung von Daten zu interpretieren,

z.B. Verschlüsselung, Kompression, Vermeidung systemspezifischer
Datendarstellung

■ Kommunikationssteuerungsschicht:

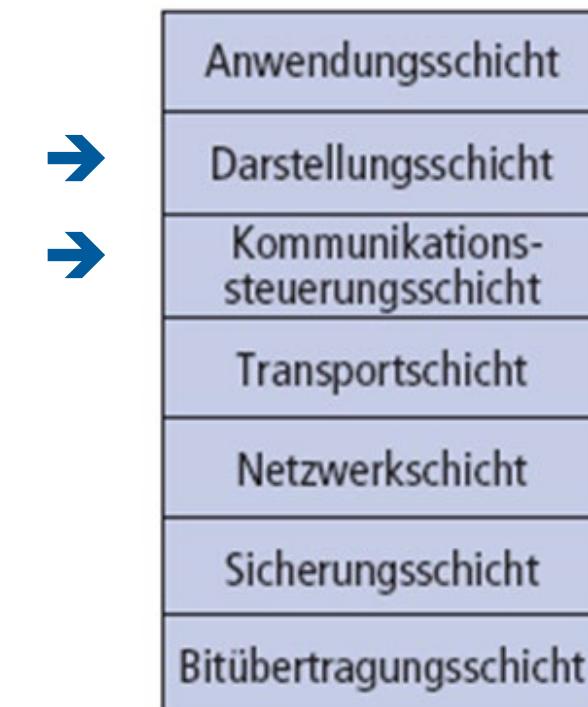
Synchronisation, Setzen von Wiederherstellungspunkten

■ Das ISO/OSI-Referenzmodell wurde 1984 standardisiert und sollte alle
Funktion der Kommunikation abdecken.

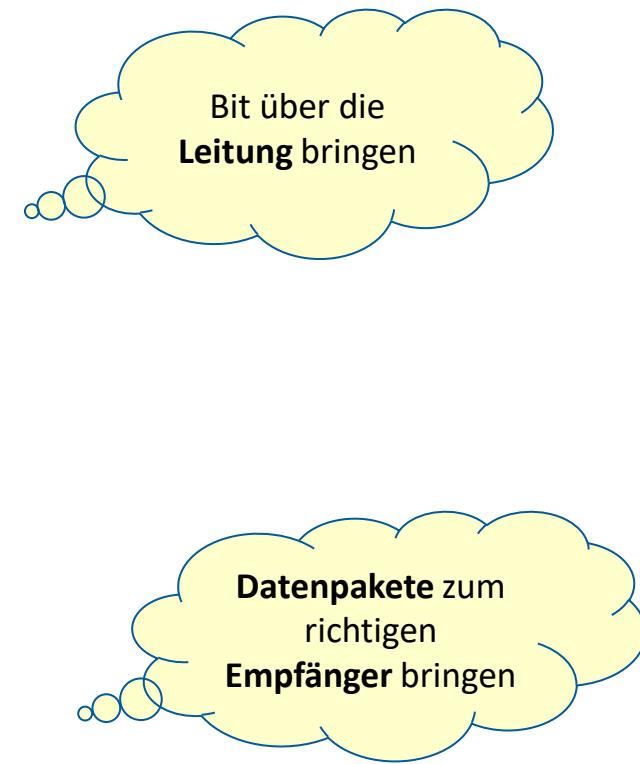
■ Der Protokollstapel des Internets bietet diese Funktionalitäten nicht!

- Wenn benötigt, müssen sie von der Anwendung implementiert werden
- Werden sie wirklich benötigt, bzw. müssen Sie international standardisiert
sein?

→ Wir betrachten normalerweise das Modell
des Internet (TCP/IP- Schichtenmodell)



- **Schicht 1: Bitübertragungsschicht (Physical Layer)**
 - Übertragung von Bits
 - Verwendung von Leitungscodes, Modulation, Übertragungstechniken etc.
 - Keine Pufferung
 - Keine Zuverlässigkeit
 - Problem: mögliche Störungen der Übertragung
 - Ziel: feste Übertragungsqualität und Datenrate
 - Zusätzlich möglich: Taktübertragung, Taksynchronisation,...
- **Schicht 2: Sicherungsschicht (Data Link Layer)**
 - Erweitert nachrichtentechnischen Kanal (Bitübertragung) zum abstrakten „gesicherten Kanal“
 - Erkennung und Behebung von Fehlern der Bitübertragungsschicht möglich
 - Übertragung von Daten zwischen direkt verbundener Geräte
 - Gliederung eines Bitstroms in Dateneinheiten (Frames => „Pakete“)
 - Pufferung sowohl beim Sender als auch beim Empfänger
 - Regelung des Kanalzugangs (Bei LAN / WLAN: Media Access Control, MAC)



■ Schicht 3: Vermittlungsschicht (Network Layer)

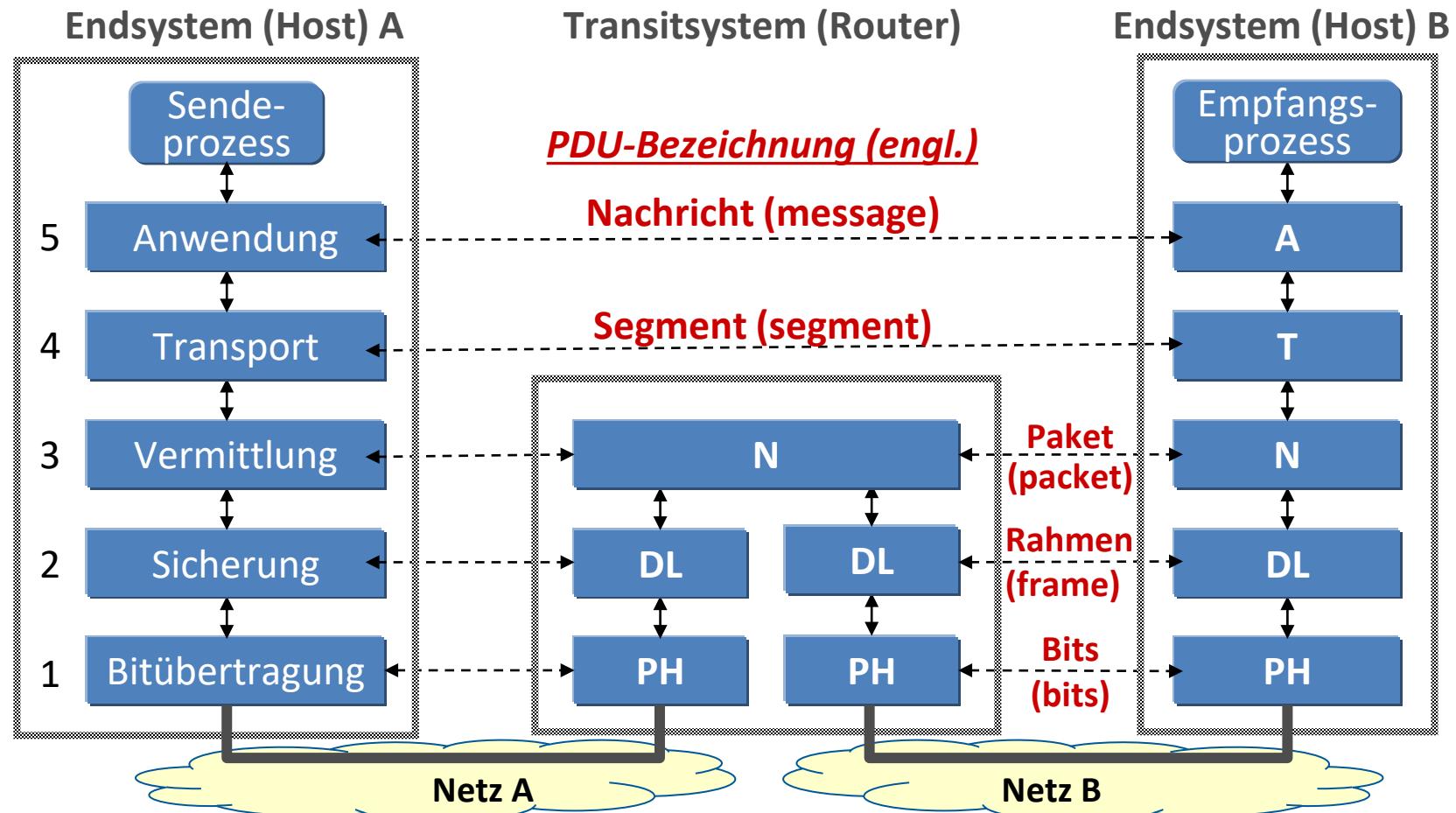
- Findet den Kommunikationspartner im weltweiten Netz
- Verknüpft einzelne Übertragungsabschnitte zu Ende-zu-Ende-Strecken
- Wegeauswahl im Kommunikationsnetz und finden geeigneter Wege für die Ende-zu-Ende-Datenübertragung
- **Adressierung der Geräte (Quelle / Ziel)**
- Multiplexen



■ Schicht 4: Transportschicht (Transport Layer)

- Übertragung von Daten zwischen **Anwendungen** auf den Endsystemen (PC, Server, Hanau)
- Adressierung der Applikationen (Das sind die Transportdienst-Benutzer)
- Abstrahiert von Diensten der Vermittlungsschicht (Der Applikation bleiben sämtliche Aspekte der Datenübertragung und Wegesuche im weltweiten Netz verborgen)
- Fehlererkennung und -behebung
- Pufferung
- Flusssteuerung
- Multiplexen





PH = Physical, DL = Data Link, N = Network

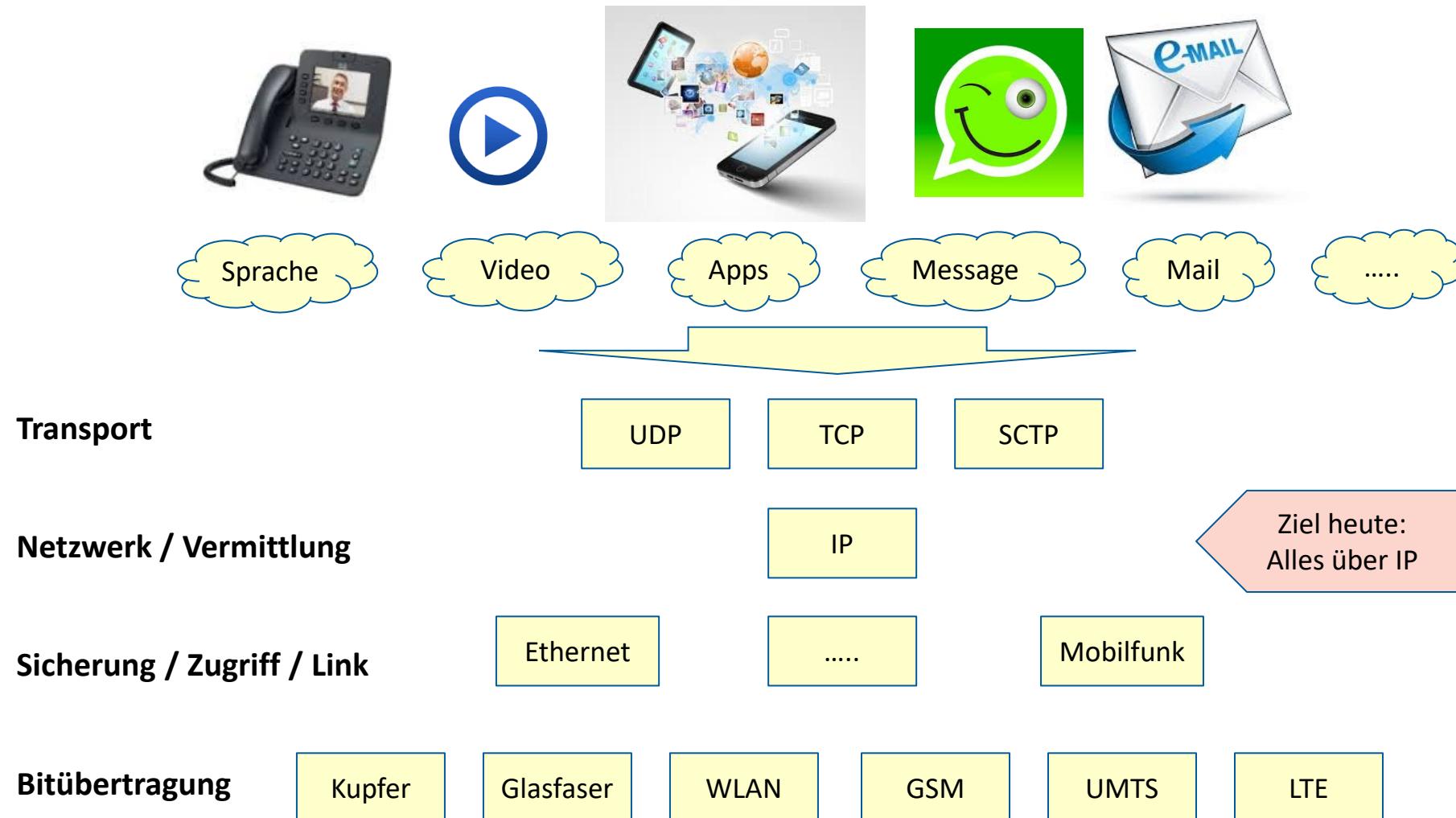
Struktur eines Ethernet-Frames mit TCP-IP



| Aufbau eines Ethernet-Pakets mit IPv4- / TCP-Daten | | | | | | | | | | | |
|--|---------------|----------------------|----------------------------|------------------|--------------|--------------------------|---------------|----------------------------|---------------------------|------------------------|-------------------------|
| Schicht 4: TCP-Segment | | | | | | | | TCP-Header | Nutzlast (≤1460 bytes) | | |
| Schicht 3: IP- Paket | | | | | | | | IP-Header | Nutzlast (≤1480 bytes) | | |
| Schicht 2: Ethernet- Frame | | | | MAC - Empf | MAC- Abs. | 802.1 Q-Tag (opt.) | Ether Type | Nutzlast (64 - 1500 bytes) | | | Frame Check Sequ. |
| Schicht 1: Ethernet- Paket+IPG | Prä- ambel | Start of Frame | Nutzlast (1518/1522 bytes) | | | | | | | Inter packet Gap | |
| Oktette | 7 | 1 | 6 | 6 | (4) | 2 | 20 | 20 | 6-1460 | 4 | 12 |

- Prinzip: „Einwickeln“ der Daten der Schicht (N) mit Header in Schicht (N-1)
- PDU (N) → SDU (N-1)
- SDU (N-1) + PCI (N-1) → PDU (N-1)

Beispiel für die einzelnen Schichten



Internet = TCP / IP
→ *Details zu TCP*

- **Ziel:**

Simpler Datentransfer zwischen Endsystemen (mit gleichem Ziel)

- **UDP – User Datagram Protocol**

[RFC 768]:

- Verbindungslos
- Unzuverlässige Datenübertragung
- Keine Flusskontrolle
- Keine Überlastkontrolle

- **Anwendung für UDP:**

- Streaming von Audio/Video,
 - Telekonferenzen,
 - IP-Telefonie
- (hier kann / braucht nicht wiederholt werden)
- Abfrage-Diensten (DNS,...)
 - Dienste, die eine einfache Implementierung erfordern
(UDP ist viel simpler als TCP)

- **Ziel:**

Datentransfer zwischen Endsystemen

- **Verbindungsaufbau mit Handshaking:**

- die Datenübertragung wird vorbereitet
- Hallo? Hallo!

Protokoll zwischen Menschen

- Es wird ein **Zustand** in den miteinander kommunizierenden Hosts erzeugt

- **TCP – Transmission Control Protocol**

- **Der zuverlässige Datentransport im Internet**

- **TCP-Dienste [RFC 793]**

- **Zuverlässige, Reihenfolge-erhaltende Übertragung eines Bytestroms:**

- Behandlung von Verlusten: Bestätigungen und Übertragungswiederholungen

- **Flusskontrolle:**

- Ein langsamer Empfänger kann einen zu schnellen Sender drosseln.

- **Überlastkontrolle:**

- Sender werden gebremst, wenn das Netzwerk überlastet ist.

- **Anwendung:**

- HTTP (Web),
- FTP (Dateiübertragung),
- Telnet (remote login),
- SMTP (E-Mail)
-

■ Punkt-zu-Punkt:

- Ein Sender, ein Empfänger

■ Zuverlässiger, Reihenfolge-erhaltender Byte-Strom:

- Keine "Nachrichtengrenzen"

■ Pipelining

- TCP-Überlast- und Flusskontrolle verändern die Größe des Fensters

■ Sender- & Empf

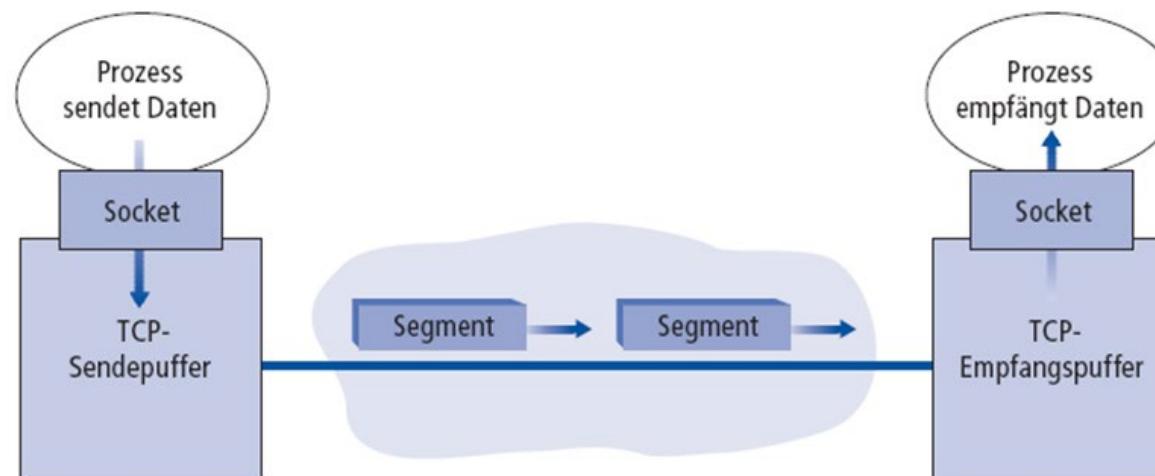
■ Vollduplex:

- Daten fließen in beide Richtungen

- MSS: Maximum Segment Size

■ Flusskontrolle:

- Empfänger kann den



Internet = TCP / IP

→ *Details zu TCP*

→ *Ports*

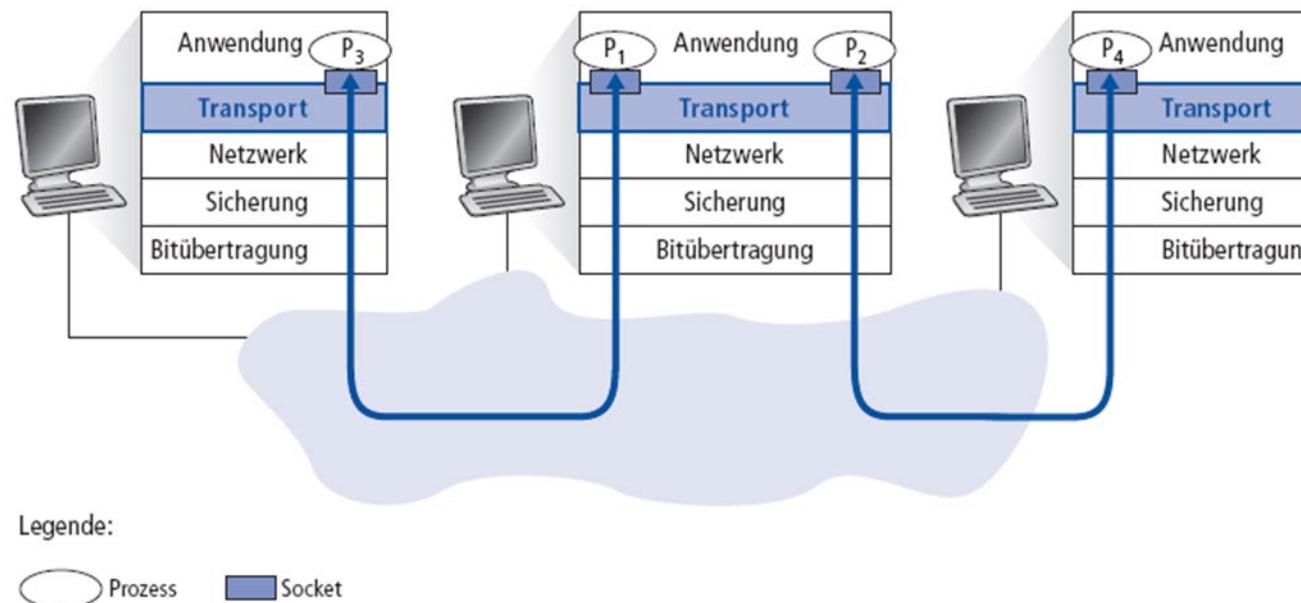
Adressierung der Transport-Schicht

Multiplexing beim Sender:

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

Demultiplexing beim Empfänger:

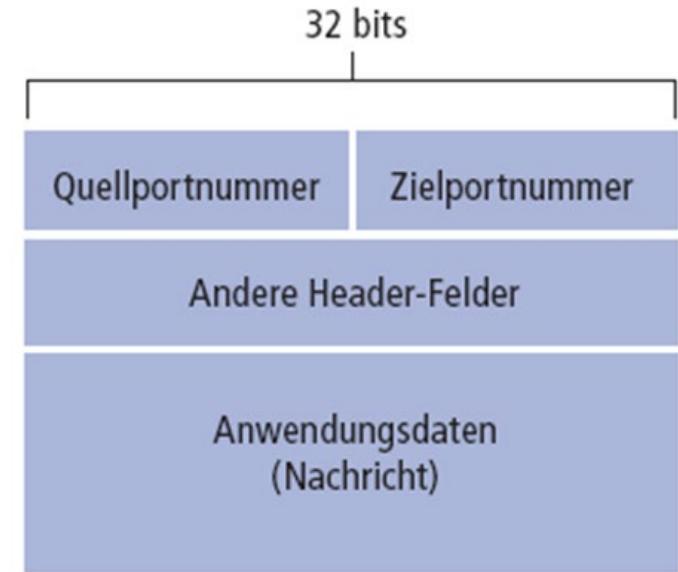
- Empfangene Segmente am richtigen Socket (Prozess) abliefern



Wie funktioniert Demultiplexing?



- **Host empfängt IP-Pakete (Datagramme)**
 - IP-Adresse wird in Netzwerkschicht verwendet für das Routing zum Ziel-Host
 - Jedes Datagramm hat
 - eine **Absender-IP-Adresse** und
 - eine **Empfänger-IP-Adresse**
 - Jedes IP-Paket enthält ein Transportschichtsegment (Nutzdaten)
 - Jedes Segment hat
 - eine **Absender- und**
 - eine **Empfänger-Portnummer**
- Hosts nutzen **IP-Adressen und Portnummern**, um Segmente an den richtigen Socket (Service Access Point) weiterzuleiten



TCP/UDP-Segmentformat

- **Sockets mit Portnummer anlegen:**

```
DatagramSocket mySocket1 = new DatagramSocket(12534);
```

```
DatagramSocket mySocket2 = new DatagramSocket(12535);
```

- **UDP-Socket wird durch ein 2-Tupel identifiziert:**

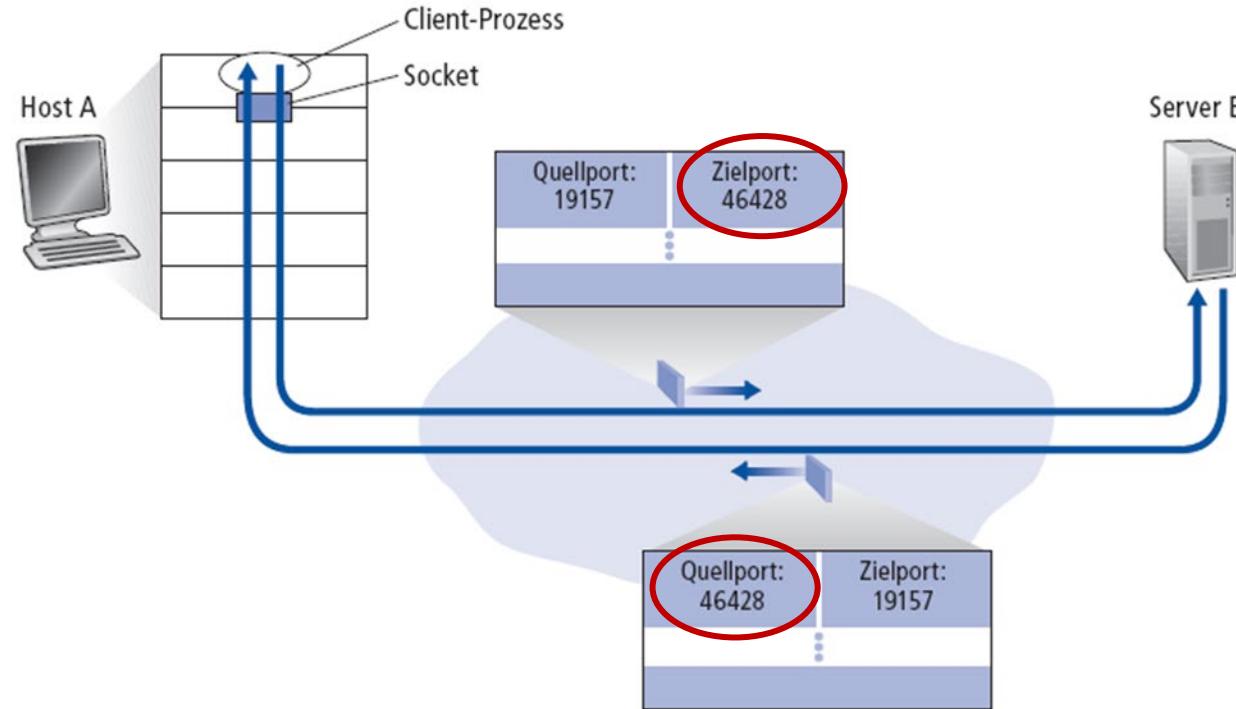
(Empfänger-IP-Adresse, Empfänger-Portnummer)

- **Wenn ein Host ein UDP-Segment empfängt:**

- Lese Empfänger-Portnummer
- Das UDP-Segment wird an den UDP-Socket mit dieser Portnummer weitergeleitet

- **IP-Datagramme mit anderer Absender-IP-Adresse oder anderer Absender-Portnummer werden an denselben Socket ausgeliefert**

Verbindungsloses Demultiplexing

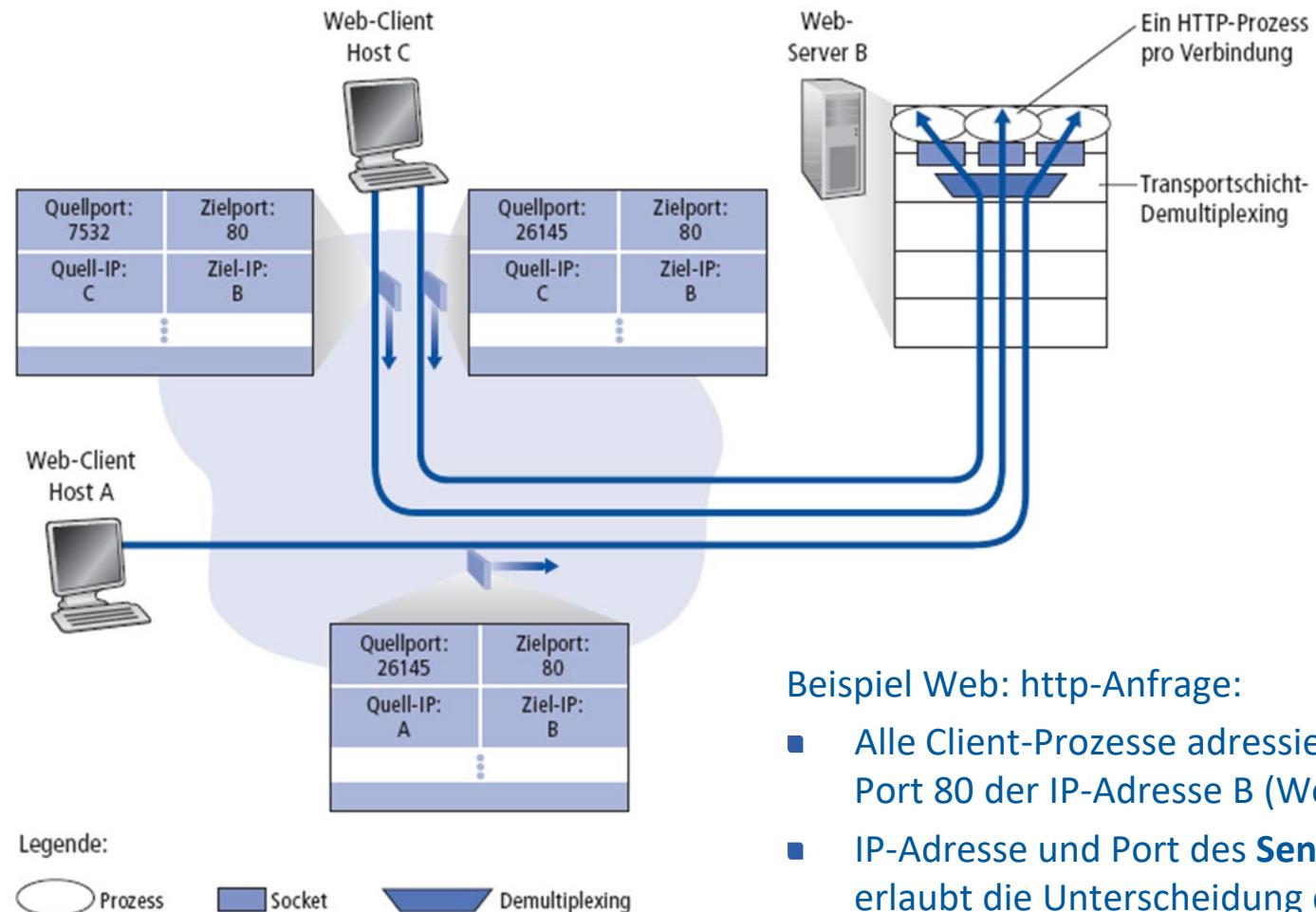


Quellport ist der Port, an den geantwortet werden soll

- 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

- 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 nichtpersistenter HTTP wird jede Anfrage über einen eigenen Socket beantwortet (dieser wird nach jeder Anfrage wieder geschlossen)

Verbindungsorientiertes Demultiplexing



Beispiel Web: http-Anfrage:

- Alle Client-Prozesse adressieren Port 80 der IP-Adresse B (Web-Server)
- IP-Adresse und Port des **Senders** erlaubt die Unterscheidung der TCP-Verbindungen.

- **TCP/UDP-Ports**, vergeben / reserviert von IANA :
 - 16-Bit-Zahlen für die Portnummern → 0 bis 65535.
 - von **0 - 1023** sind **reserviert** und vordefiniert („well known ports“)
 - „Klassische Dienste“, Bsp: 80: http, 25: smtp, 53: dns, 110: pop3,...
 - Von **1024 - 49151** **registrierte Port**: “vendors use for applications”
 - Bsp: Web-basierte Dienste von Firmen, Spiele,
 - Von **49152 - 65535** sind frei benutzbar („dynamic / private ports“), auch („anonymous ports“)



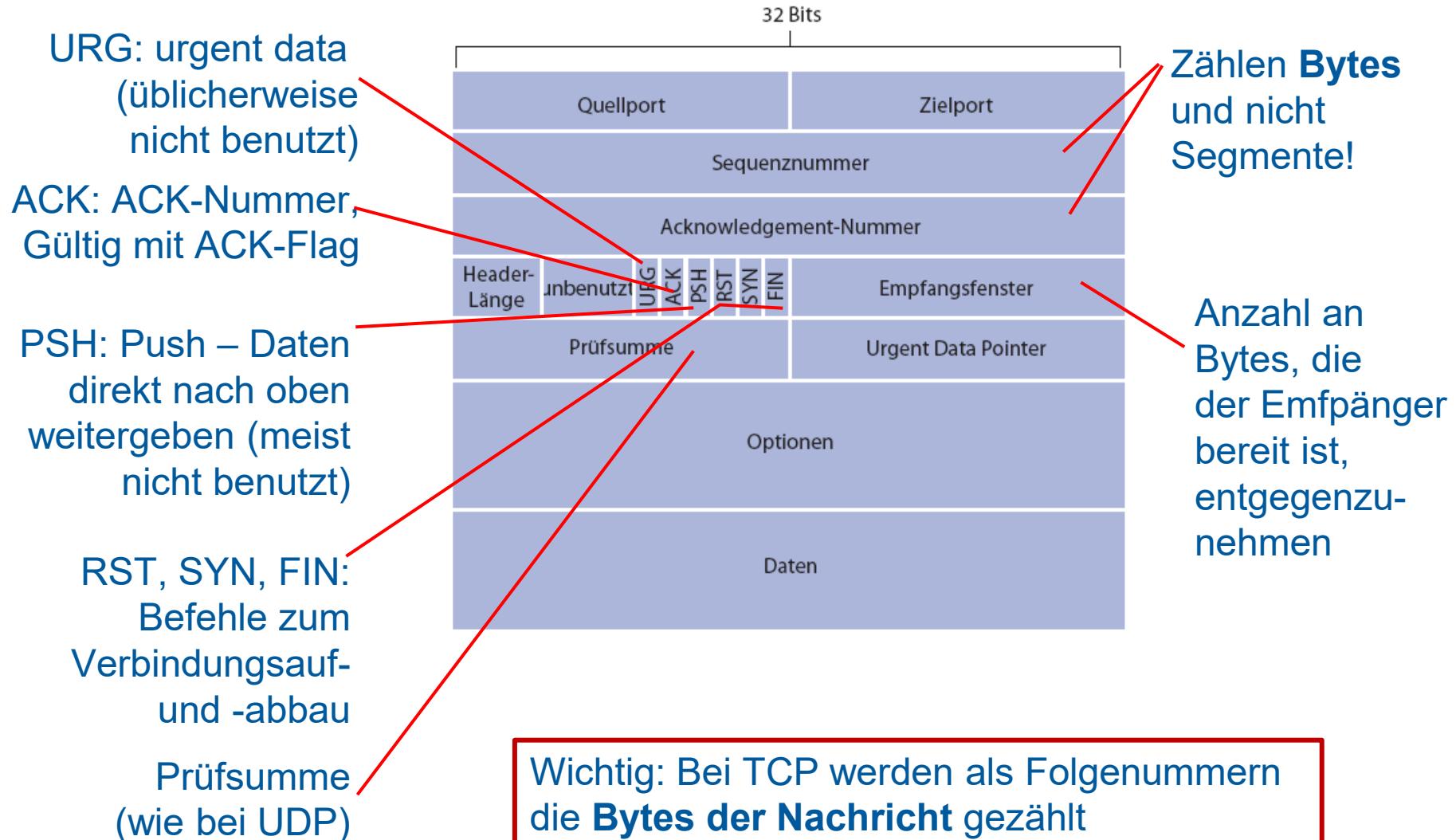
Liste der registrierten und reservierten Ports:

http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Internet = TCP / IP

→ *Details zu TCP*

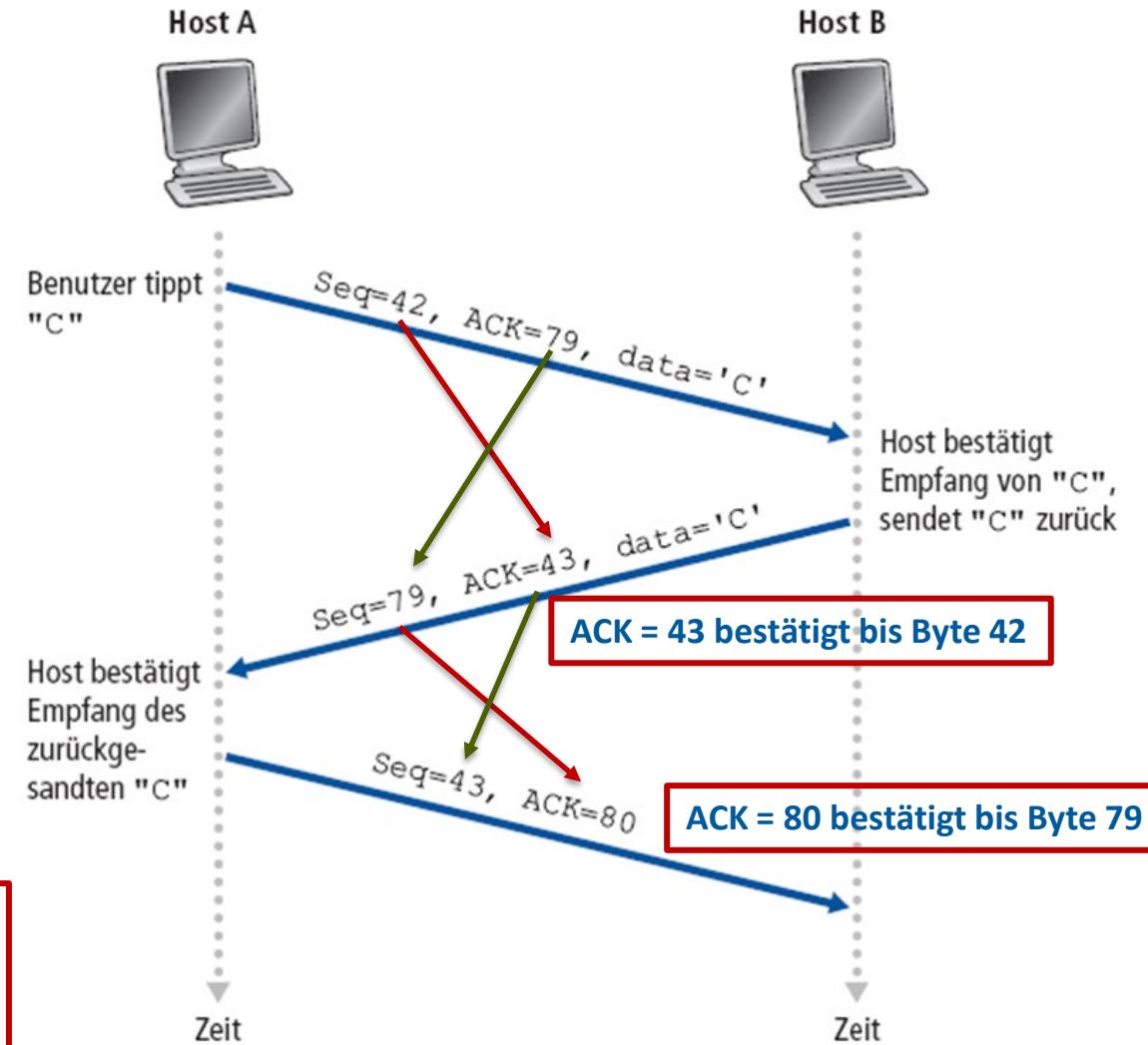
→ *Quittierung und Wiederholung*



Sequenznummern:

- Nummer des ersten Byte im Datenteil
- ACKs:
 - Sequenznummer des **nächsten Byte**, das von der Gegenseite erwartet wird
 - Kumulative ACKs

Merke:
ACK gibt die Nummer des **nächsten erwarteten** Bytes der Nachricht an.



Beispiel einer Datenübertragung (Web-Seite)



Filter für http-Nachrichten

| http | | | | | | | |
|------|-----------|--------|-------------|----------|--------|---|--|
| No. | Time | Source | Destination | Protocol | Length | Info | |
| → 4 | 0.002423 | Client | Server | HTTP | 474 | POST /data.lua HTTP/1.1 (application/x-www-form-urlencoded) | |
| ← 21 | 1.768539 | Server | Client | HTTP | 283 | HTTP/1.1 200 OK (text/html) | |
| • 23 | 2.794663 | Client | Server | HTTP | 474 | POST /data.lua HTTP/1.1 (application/x-www-form-urlencoded) | |
| • 39 | 4.576435 | Server | Client | HTTP | 283 | HTTP/1.1 200 OK (text/html) | |
| 41 | 5.614013 | Client | Server | HTTP | 474 | POST /data.lua HTTP/1.1 (application/x-www-form-urlencoded) | |
| 57 | 7.584856 | Server | Client | HTTP | 284 | HTTP/1.1 200 OK (text/html) | |
| 59 | 8.597267 | Client | Server | HTTP | 474 | POST /data.lua HTTP/1.1 (application/x-www-form-urlencoded) | |
| 73 | 10.333236 | Server | Client | HTTP | 1397 | HTTP/1.1 200 OK (text/html) | |
| 75 | 11.346226 | Client | Server | HTTP | 474 | POST /data.lua HTTP/1.1 (application/x-www-form-urlencoded) | |
| 89 | 13.241475 | Server | Client | HTTP | 1398 | HTTP/1.1 200 OK (text/html) | |
| 91 | 14.256429 | Client | Server | HTTP | 474 | POST /data.lua HTTP/1.1 (application/x-www-form-urlencoded) | |
| 1... | 16.014090 | Server | Client | HTTP | 1397 | HTTP/1.1 200 OK (text/html) | |

Frame 21: 283 bytes on wire (2264 bits), 283 bytes captured (2264 bits) on interface 0
Ethernet II, Src: Avm_3b:fb:82 (c0:25:06:3b:fb:82), Dst: Giga-Byt_08:15:61 (6c:f0:49:08:15:61)
Internet Protocol Version 4, Src: Server (192.168.178.1), Dst: Client (192.168.178.36)
Transmission Control Protocol, Src Port: 80, Dst Port: 55802, Seq: 14601, Ack: 421, Len: 229
[11 Reassembled TCP Segments (14829 bytes): #6(1460), #7(1460), #9(1460), #10(1460), #11(1460), #13(1460), #15(1460), #17(1460), #19(1460), #21(1460), #23(1460)]
Hypertext Transfer Protocol
Line-based text data: text/html

Anwendungssicht http:

- http-POST => Anforderung von Daten vom Web-Server (Port 80).
- http-OK => Die Daten werden in mehreren TCP-Segmenten übertragen, dann wieder zusammengefügt (reassembled) und dem Browser zu Darstellung als html-Text übergeben (im Bsp. 14829 Byte)
- http wird in TCP übertragen

Beispiel einer Datenübertragung (Web-Seite)

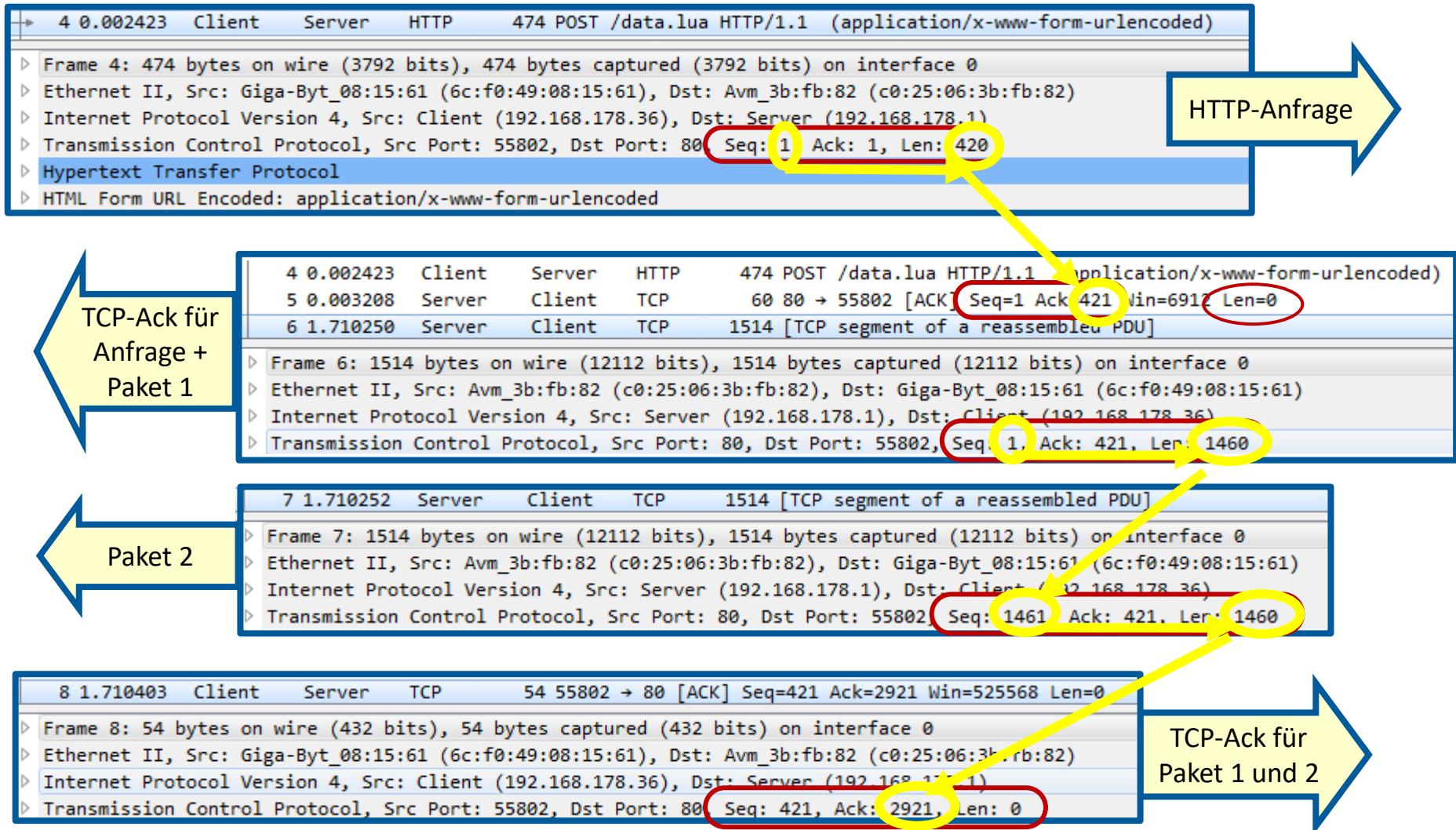


| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|--------|-------------|----------|--------|--|
| 1 | 0.000000 | Client | Server | TCP | 66 | 55802 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 2 | 0.000458 | Server | Client | TCP | 66 | 80 → 55802 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=4 |
| 3 | 0.000626 | Client | Server | TCP | 54 | 55802 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TCP-Verbindungsauftbau |
| 4 | 0.002423 | Client | Server | HTTP | 474 | POST /data.lua HTTP/1.1 (application/x-www-form-urlencoded) |
| 5 | 0.003208 | Server | Client | TCP | 60 | 80 → 55802 [ACK] Seq=1 Ack=421 Win=6912 Len=0 |
| 6 | 1.710250 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 7 | 1.710252 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 8 | 1.710403 | Client | Server | TCP | 54 | 55802 → 80 [ACK] Seq=421 Ack=2921 Win=525568 Len=0 |
| 9 | 1.711265 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 10 | 1.711269 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 11 | 1.711270 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 12 | 1.711465 | Client | Server | TCP | 54 | 55802 → 80 [ACK] Seq=421 Ack=7301 Win=525568 Len=0 |
| 13 | 1.712890 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 14 | 1.712979 | Client | Server | TCP | 54 | 55802 → 80 [ACK] Seq=421 Ack=8761 Win=525568 Len=0 |
| 15 | 1.713228 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 16 | 1.713230 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 17 | 1.713233 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 18 | 1.713327 | Client | Server | TCP | 54 | 55802 → 80 [ACK] Seq=421 Ack=13141 Win=525568 Len=0 |
| 19 | 1.713523 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 20 | 1.713567 | Client | Server | TCP | 54 | 55802 → 80 [ACK] Seq=421 Ack=14601 Win=525568 Len=0 |
| 21 | 1.768539 | Server | Client | HTTP | 283 | HTTP/1.1 200 OK (text/html) |
| 22 | 1.818423 | Client | Server | TCP | 54 | 55802 → 80 [ACK] Seq=421 Ack=14830 Win=525312 Len=0 |
| 23 | 2.794663 | Client | Server | HTTP | 474 | POST /data.lua HTTP/1.1 (application/x-www-form-urlencoded) |
| 24 | 2.795104 | Server | Client | TCP | 60 | 80 → 55802 [ACK] Seq=14830 Ack=841 Win=7984 Len=0 |
| 25 | 4.517560 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 26 | 4.517563 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 27 | 4.517564 | Server | Client | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 28 | 4.517747 | Client | Server | TCP | 54 | 55802 → 80 [ACK] Seq=841 Ack=19210 Win=525568 Len=0 |

Aufgezeichnete Folge der übertragenen Datenpakete Client ↔ Server

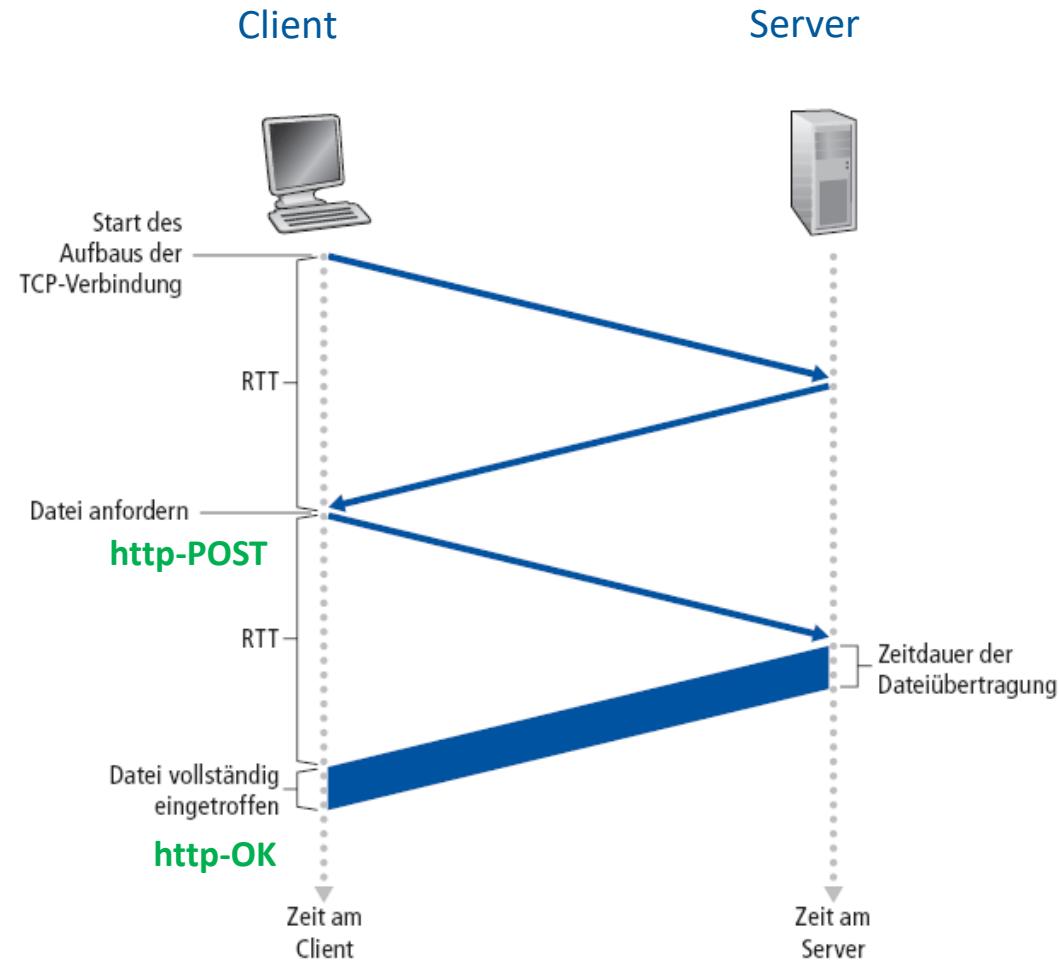
TCP-Verbindung mit Übertragung von Nutzdaten (Webseite)

Beispiel einer Datenübertragung (Web-Seite)



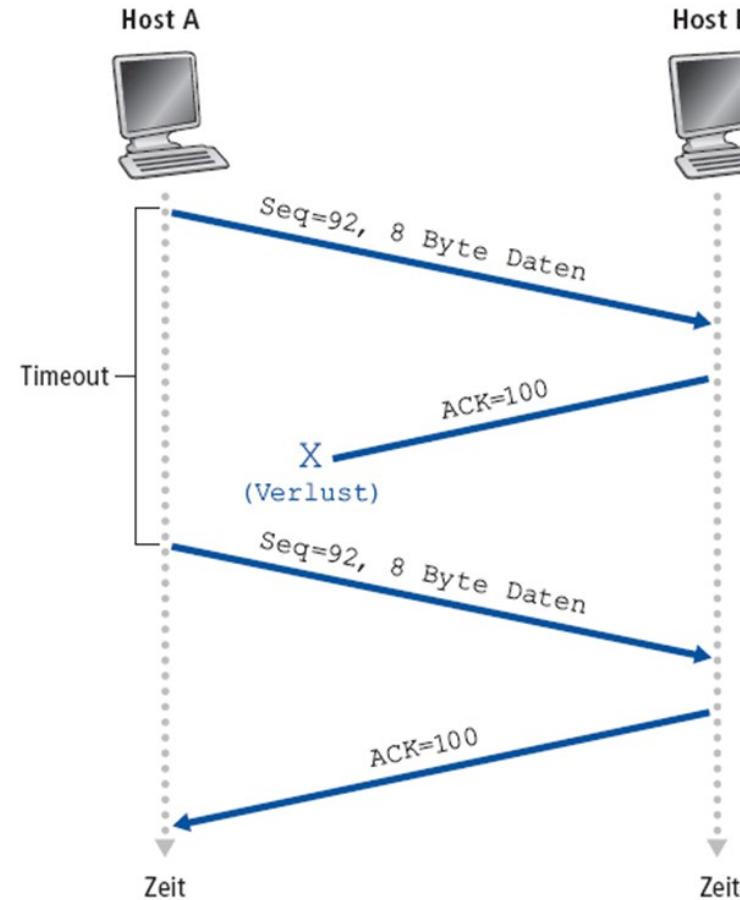
Verzögerung:

- 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 + Übertragungsverzögerungen
- RTT = Round Trip Time = Laufzeit Client → Server + Laufzeit Server → Client



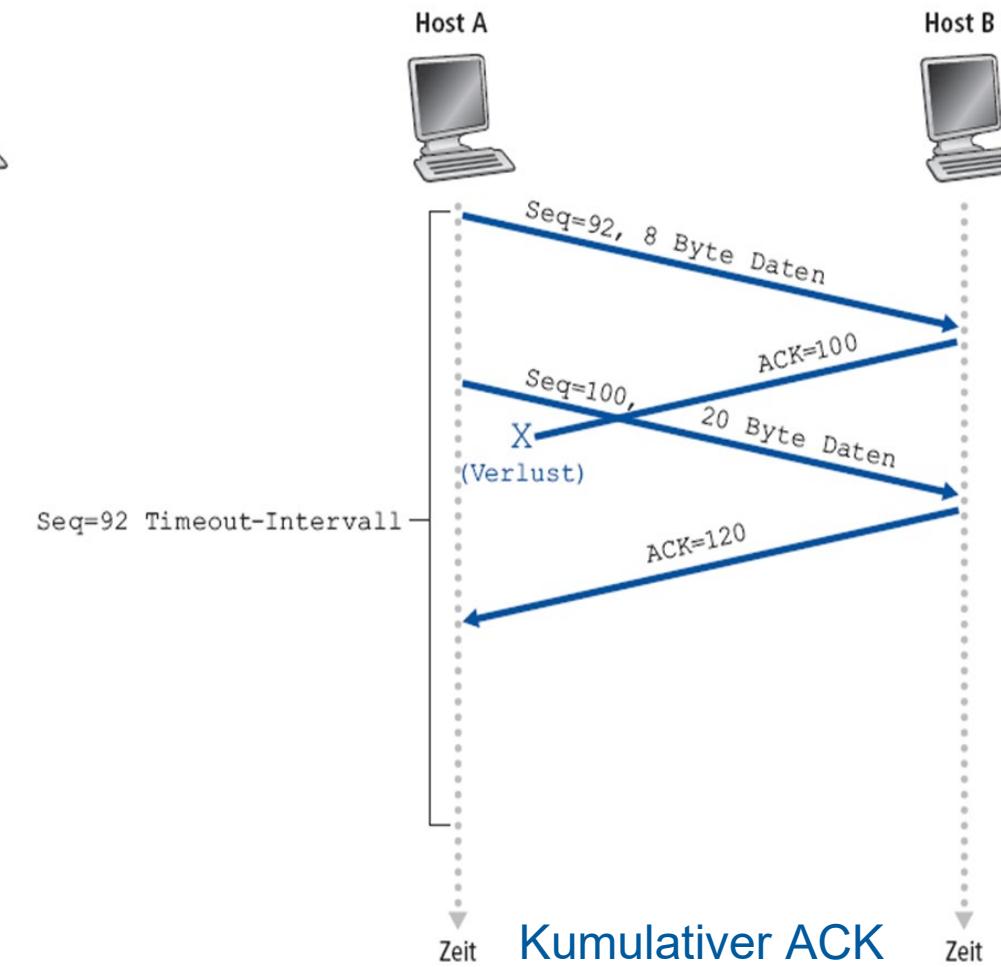
vgl. Kurose, Abb. 2.7

TCP Beispiele – Time out und Kumulativer ACK



Wiederholung nach Time-out

Problem: Dauer für Time-out => Estimated RTT + „Sicherheitsabstand“
Time out ist oft sehr lang => Wiederholung kommt sehr spät!



Kumulativer ACK

- TCP benutzt Timer: u.a. Retransmission-Timer für erneute Übertragungen (RT-Time Out)
- Festlegung der Zeitschranke für das Eintreffen einer Bestätigung auf Schicht 4 (Laufzeit durch ein Netz mit vielen Knoten) wesentlich schwieriger als auf Schicht 2 (1 Abschnitt)
- Grund: Höhere Varianz der Round Trip Time (RTT)

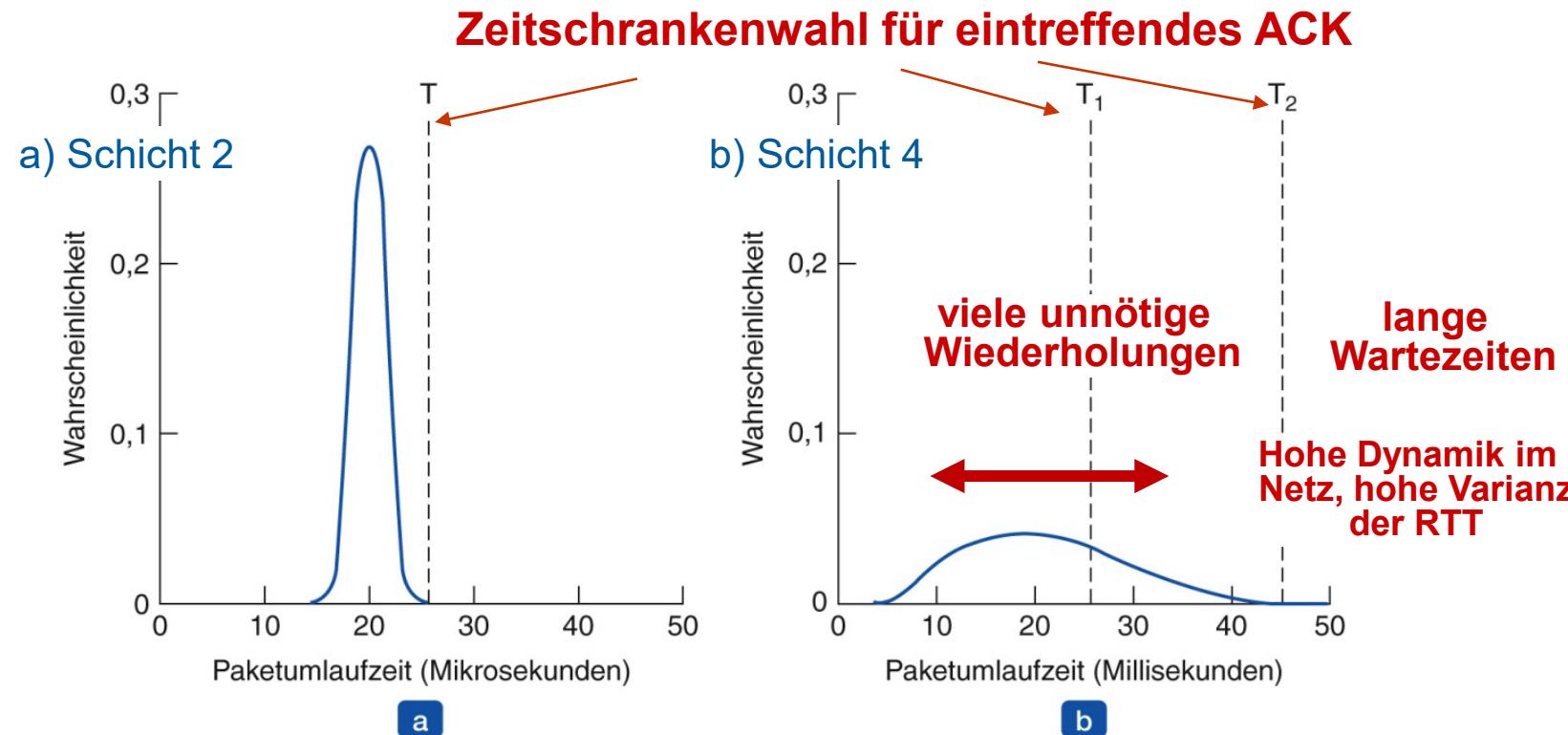
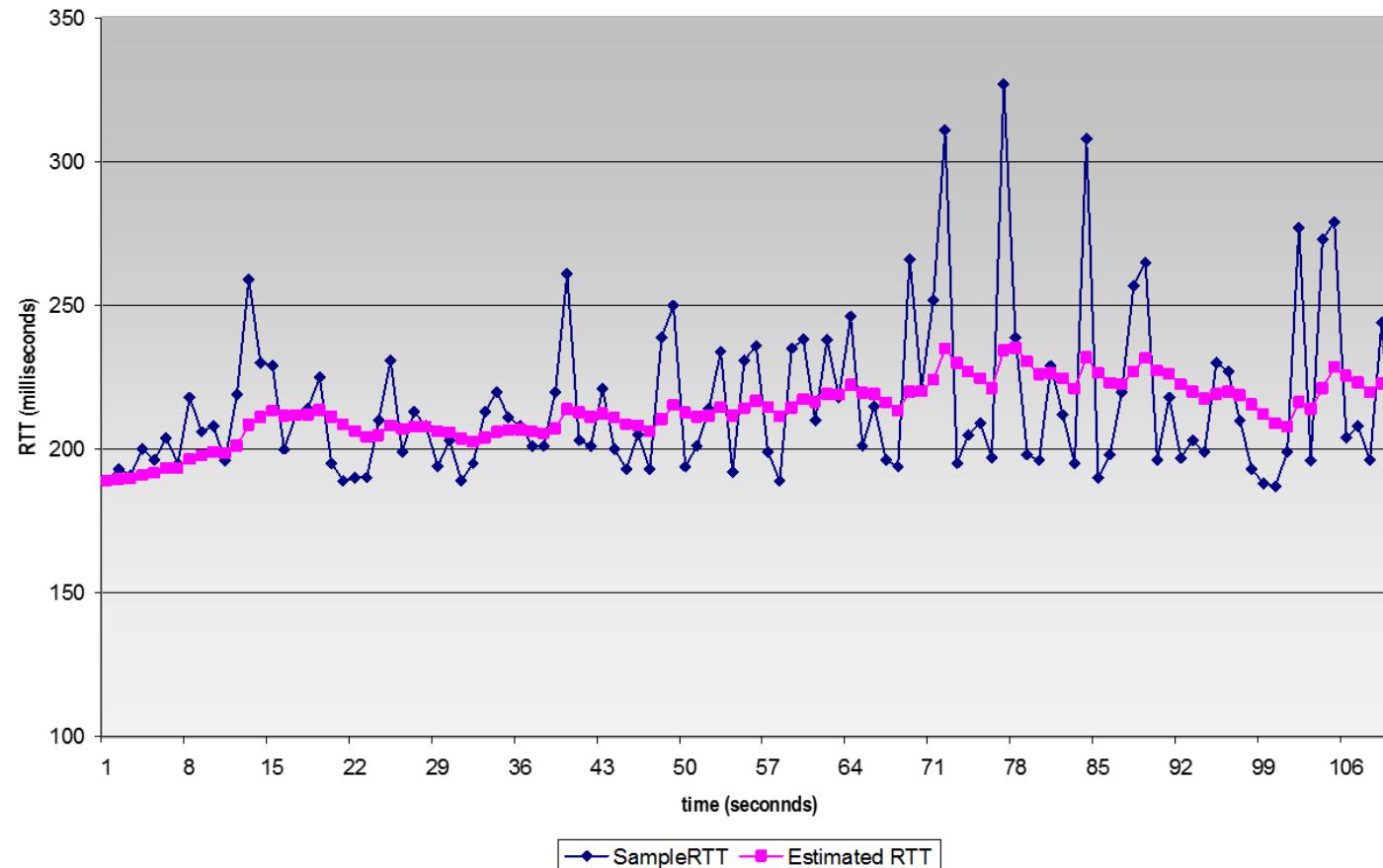


Abbildung 6.42: (a) Wahrscheinlichkeitsdichte der Ankunftszeiten von Bestätigungen auf der Sicherungsschicht.
 (b) Wahrscheinlichkeitsdichte der Ankunftszeiten von Bestätigungen für TCP.

Beispiel für die RTT-Bestimmung / Messung



- **Problem: RTT variiert sehr stark, wenn Netz hoch belastet**
 - ➔ Timeout löst Wiederholungen aus, welche die Netzlast noch weiter erhöhen
 - ➔ Überlast wird noch weiter verstärkt 💣 ➔ Netzstabilität ???

TCP-Zeitschrankenfestlegung RTT -1-



- Algorithmus von **Van Jacobson** (1988, verbessert 1990), (siehe RFC 6298, früher 2988):
- Für jede Verbindung verwaltet TCP die **Variable SRTT** (Smoothed Round-Trip Time):

$$SRTT = \alpha * SRTT + (1 - \alpha) R$$

mit Glättungsfaktor $\alpha = 7/8$ (typisch), R = letzter Messwert von RTT

(=> Exponentiell gewichteter gleitender Durchschnitt; Exponentially Weighted Moving Average, EWMA)

- Wahl der **Zeitschranke RTO** abhängig von RTT: (Δt = zusätzliche Wartezeit)

$$RTO = SRTT + \Delta t$$

- Bestimmung von Δt aus der Varianz der RTT (**RTTVar**):

$$RTTVar = \beta * RTTVar + (1 - \beta) * |SRTT - R|$$

Differenz zwischen
erwartetem Wert und
gemessenem Wert

mit Glättungsfaktor $\beta = 3/4$ (typisch), R = letzter Messwert von RTT

- Zeitschranke für eine Wiederholung:

$$RTO = SRTT + 4 * RTTVar$$

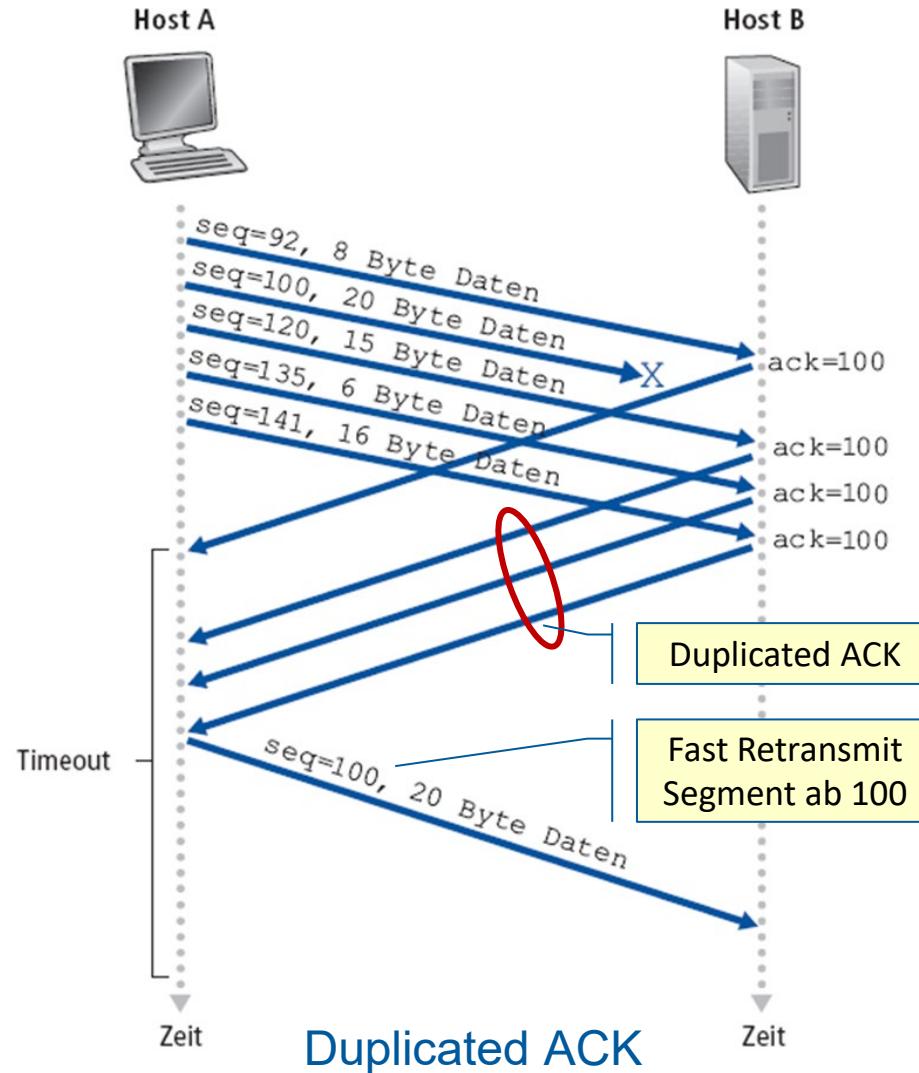
- Startwert: **RTO: 1 sec** (RFC 6298)

Anm: Die Formeln entsprechen RFC 2988, und Tanenbaum 2012.
Im neuen RFC 6298 sind die Formeln umgestellt $\alpha := 1 - \alpha$

TCP-Zeitschrankenfestlegung RTT -2-



- Problem: RTT variiert sehr stark, wenn Netz hoch belastet
 - ➔ Timeout löst Wiederholungen aus, welche die Netzlast noch weiter erhöhen
 - ➔ Überlast wird noch weiter verstärkt ⚡ ➔ Netzstabilität ???
- Bei Segment-Wiederholungen: Problem der Zuordnung von der Antwort zu ursprünglichen oder zum wiederholten Segment => welche RTT?
- Messung im Fall einer Segmentwiederholung nach Timeout
 - Variante von Karn (Karn-Algorithmus, 1991):
 - Messwert **R** wird nicht verwendet zur Aktualisierung von **RTT** und **RTTVar**
 - Nach Segmentwiederholung: Verdopplung der Zeitschranke
$$RTO := 2 * RTO$$
 - So lange bis ein Segment wieder innerhalb Zeitschranke bestätigt wurde
 - Danach wieder Algorithmus nach Jacobson:
$$RTO = SRTT + \Delta t$$



- Zeit für Timeout ist häufig sehr lang:
 - Große Verzögerung vor einer Neuübertragung
 - Erkennen von Paketverlusten durch doppelte ACKs
- (Duplicated ACK):**
- 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**
 - Segment erneut schicken, **bevor** der Timer ausläuft

Internet = TCP / IP

→ *Details zu TCP*

→ *Verbindungsmanagement*

Zur Erinnerung:

- TCP-Sender und TCP-Empfänger bauen eine Verbindung auf, bevor sie Daten austauschen

- Initialisieren der TCP-Variablen:

- Sequenznummern,
 - Informationen für Flusskontrolle (z.B. **RcvWindow**)

- **Client: Initiator**

```
Socket clientSocket = new  
Socket("hostname", "port number");
```

- **Server: vom Client kontaktiert**

```
Socket connectionSocket =  
welcomeSocket.accept();
```

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 SYN+ACK

- Server legt Puffer an
- Initiale Sequenznummer
(Server->Client)

Schritt 3: Client empfängt SYN+ACK und antwortet mit einem ACK; dieses Segment darf bereits Daten beinhalten

■ Verbindungsauftbau im 3-way-handshake

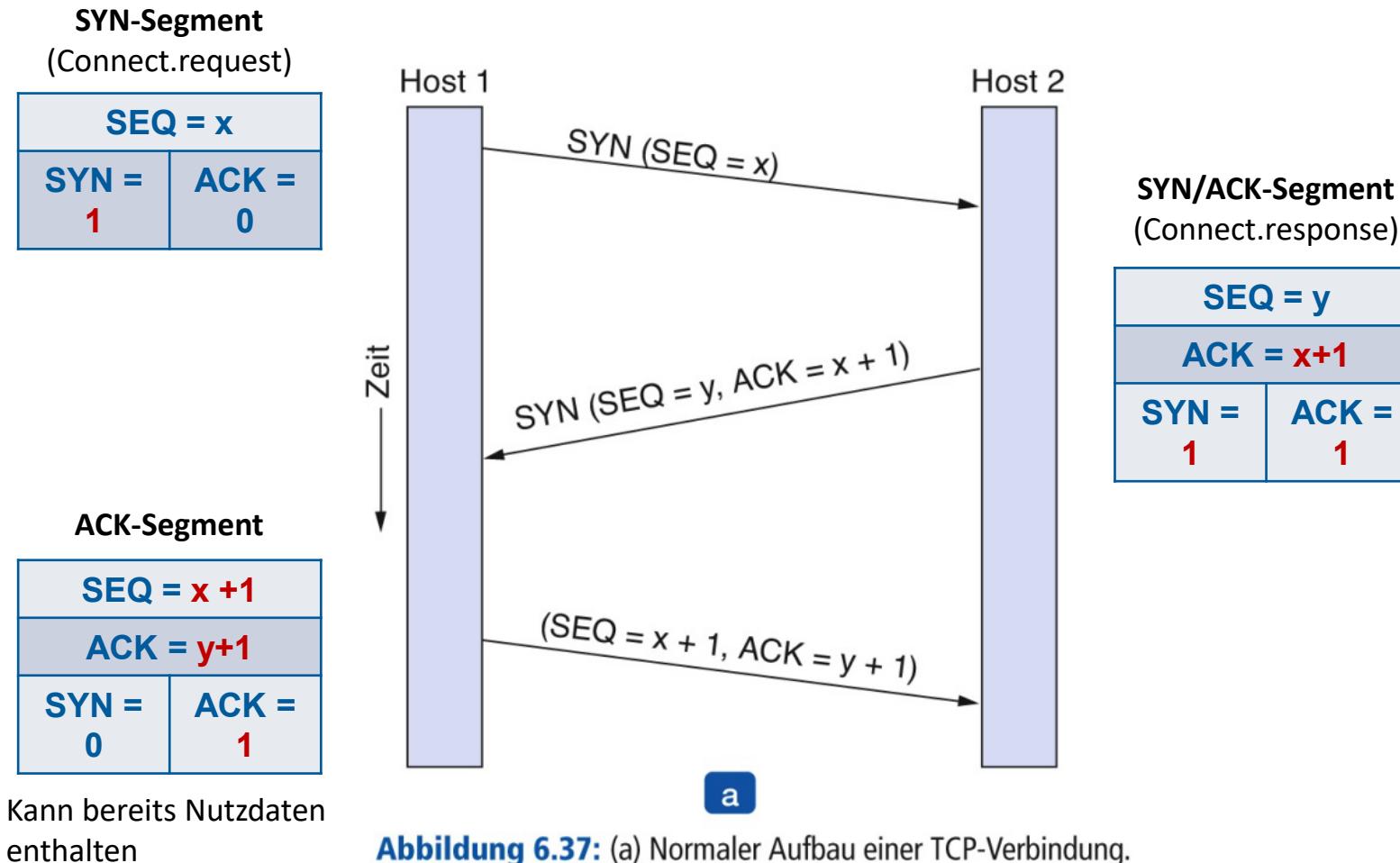
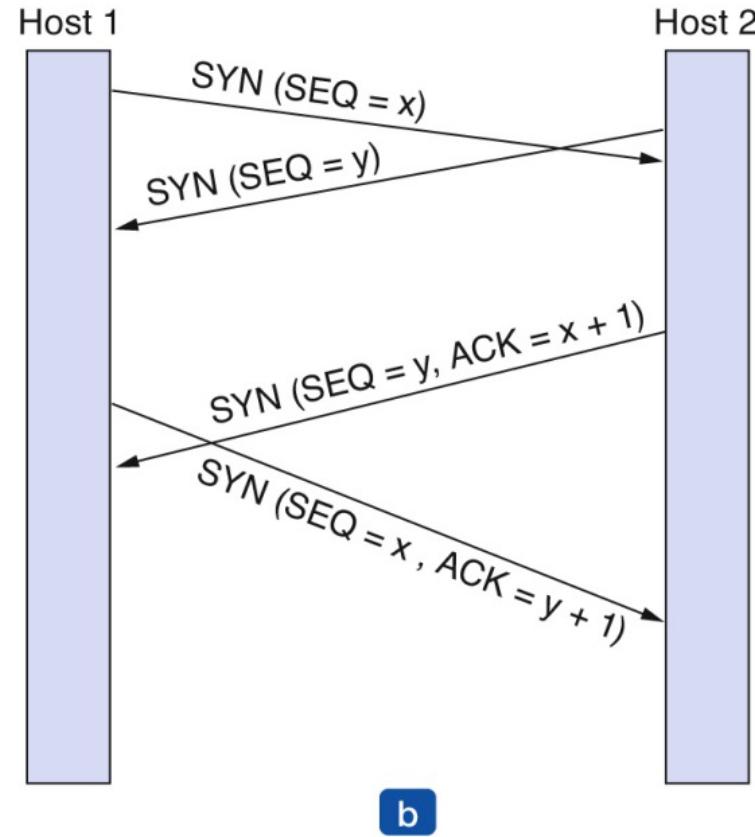


Abbildung 6.37: (a) Normaler Aufbau einer TCP-Verbindung.

■ Verbindungsauftbau im 3-way-handshake



(b) Gleichzeitiger Verbindungsauftbau von beiden Seiten.

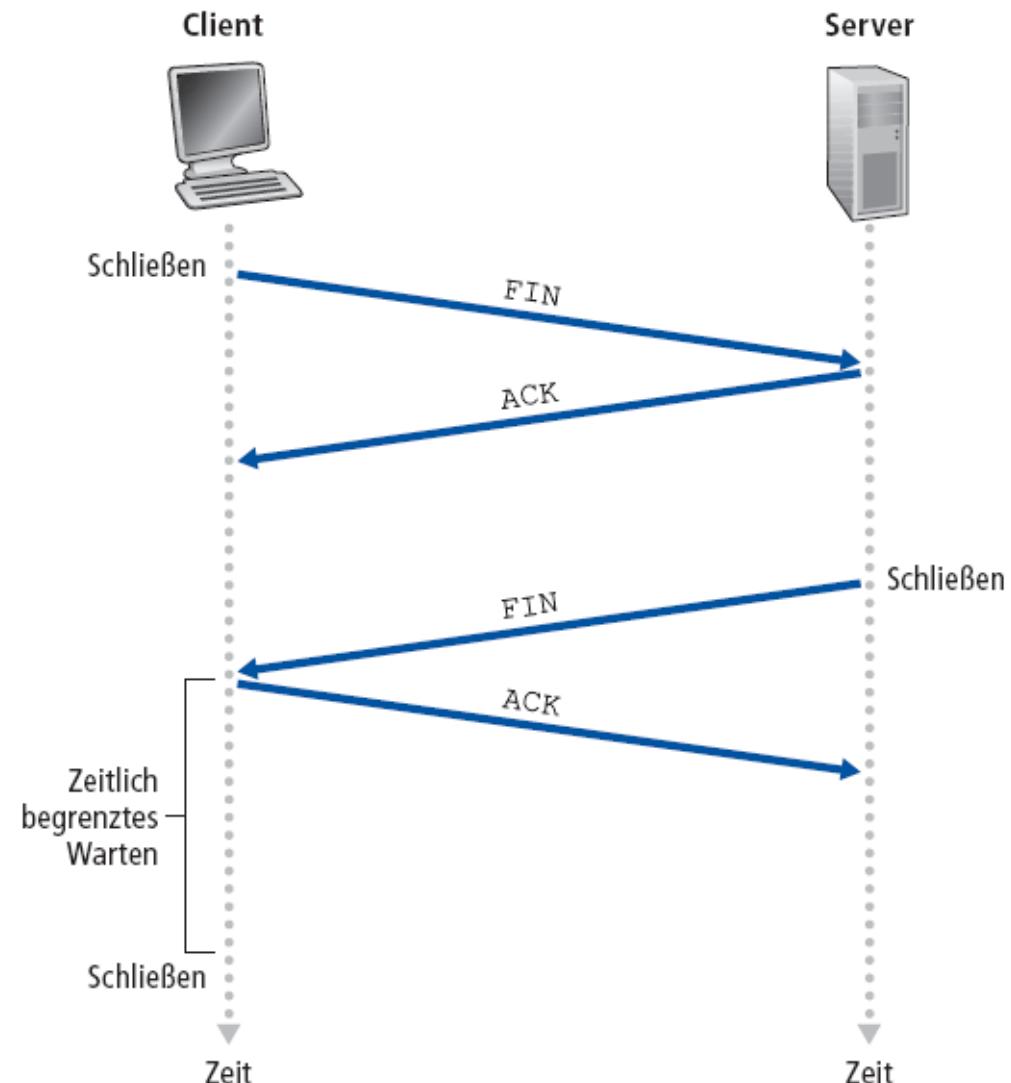
Schließen einer Verbindung

Client schließt socket:

```
clientSocket.close();
```

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)

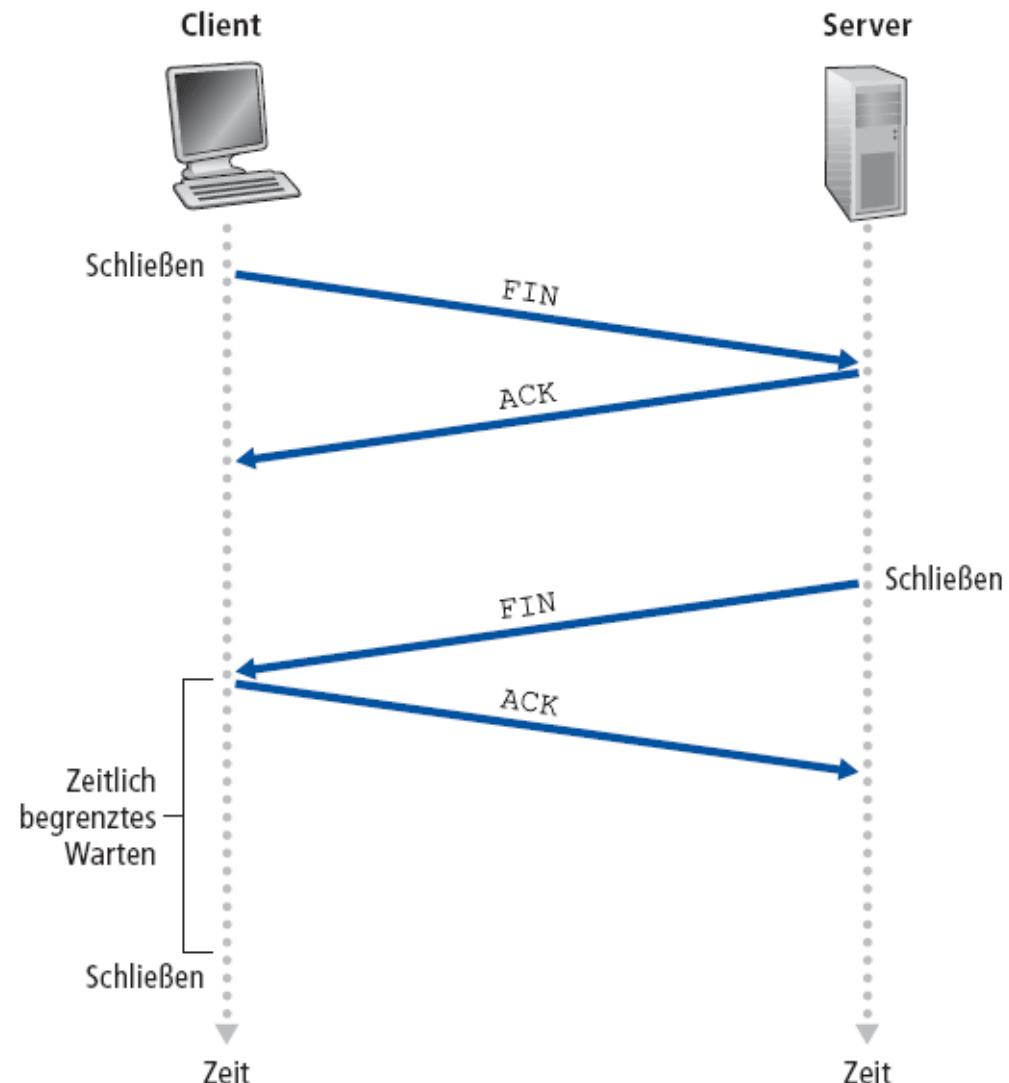


Schritt 3: Client empfängt FIN und antwortet mit ACK

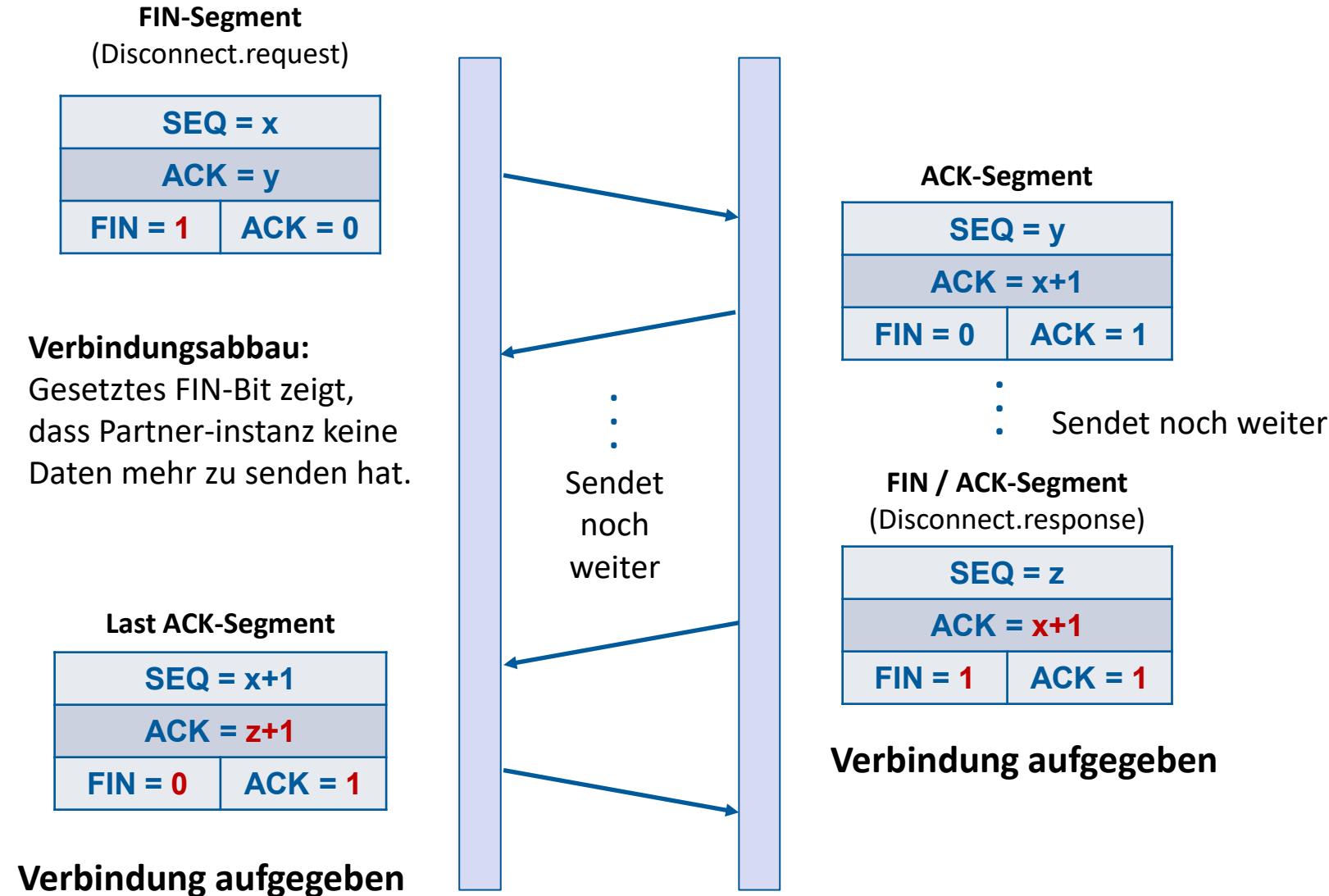
- Beginnt einen “Timed-Wait”-Zustand – er antwortet auf Sendewiederholungen des Servers mit ACK

Schritt 4: Server, empfängt ACK und schließt Verbindung

Anmerkung: Mit kleinen Änderungen können so auch gleichzeitig abgeschickte FINs behandelt werden



TCP-Verbindungsabbau



- **Ordnungsgemäßer Abbau**
 - Beide Anwendungsprozesse müssen unabhängig voneinander ihre „Hälften“ der Verbindung schließen
 - Wurde lediglich eine Richtung geschlossen, so können in der anderen Richtung noch Daten gesendet werden
 - „Halb geschlossene Verbindungen“
 - Damit ist noch ein unidirektionaler Datenfluss möglich
 - In „geschlossener“ Richtung können noch Kontroll- Dateneinheiten gesendet werden
- **Abbruch einer Verbindung**
 - Verbindung kann mit Reset abgebrochen werden:
 - Dateneinheit mit gesetztem RST-Bit
- Anm.: Halb **offene** Verbindung => andere Seite ist abgestürzt und reagiert nicht mehr

TCP-Verbindungs-Management

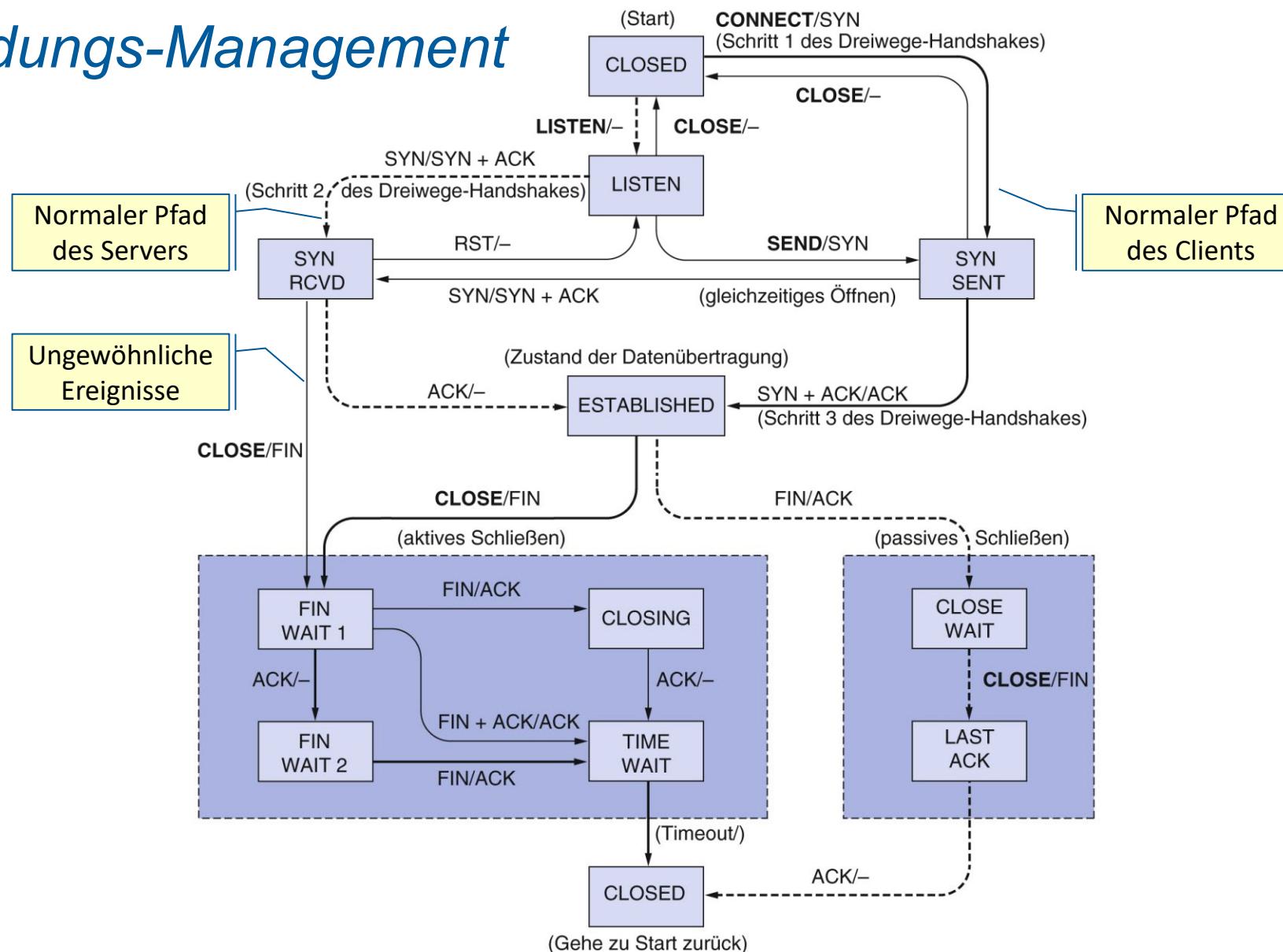


Abbildung 6.39: TCP-Verbindungsmanagement als Zustandsautomat (fette Linie = normaler Pfad des Clients; fette gestrichelte Linie = normaler Pfad des Servers; feine Linien = ungewöhnliche Ereignisse). Jeder Übergang wird durch das Ereignis bezeichnet, das ihn verursacht, sowie durch die daraus resultierenden Aktionen (getrennt durch einen Schrägstrich).

[Tanenbaum, 2012]

Zustände der TCP-Verbindungssteuerung



| Zustand | Beschreibung |
|-------------|---|
| CLOSED | Keine Verbindung aktiv oder anstehend. |
| LISTEN | Der Server wartet auf eine ankommende Verbindung. |
| SYN RCVD | Eine Verbindungsanfrage ist angekommen. Warten auf Bestätigung. |
| SYN SENT | Die Anwendung hat begonnen, eine Verbindung zu öffnen. |
| ESTABLISHED | Zustand der normalen Datenübertragung. |
| FIN WAIT 1 | Die Anwendung möchte die Übertragung beenden. |
| FIN WAIT 2 | Die andere Seite ist einverstanden, die Verbindung abzubauen. |
| TIME WAIT | Warten, bis keine Pakete mehr kommen. |
| CLOSING | Beide Seiten haben gleichzeitig versucht, zu beenden. |
| CLOSE WAIT | Die Gegenseite hat die Freigabe eingeleitet. |
| LAST ACK | Warten, bis keine Pakete mehr kommen. |

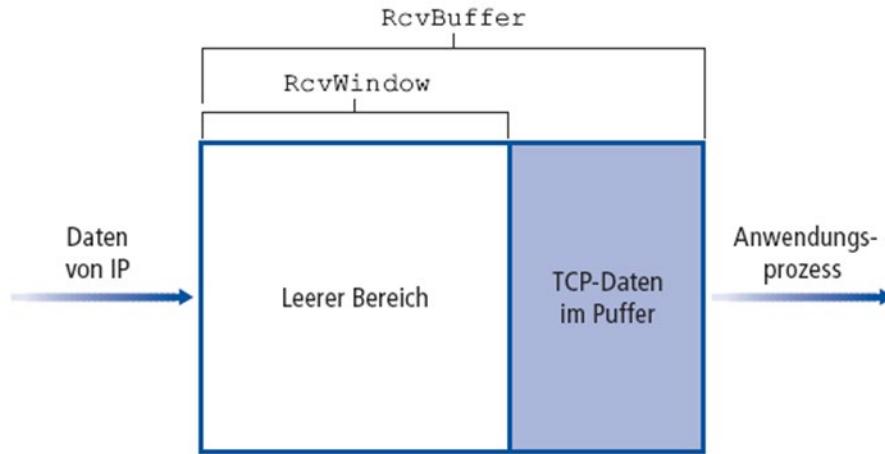
Abbildung 6.38: Zustände des endlichen Automaten für die Verwaltung von TCP-Verbindungen.

Internet = TCP / IP

→ *Details zu TCP*

→ *Flusskontrolle*

- Empfängerseite von TCP hat einen **Empfängerpuffer**:



- Die Anwendung kommt unter Umständen nicht mit dem Lesen hinterher, daher:
- Empfangsfenster (Receive Window, RcvWindow)
- RCVWindow = **Credit** für Sender

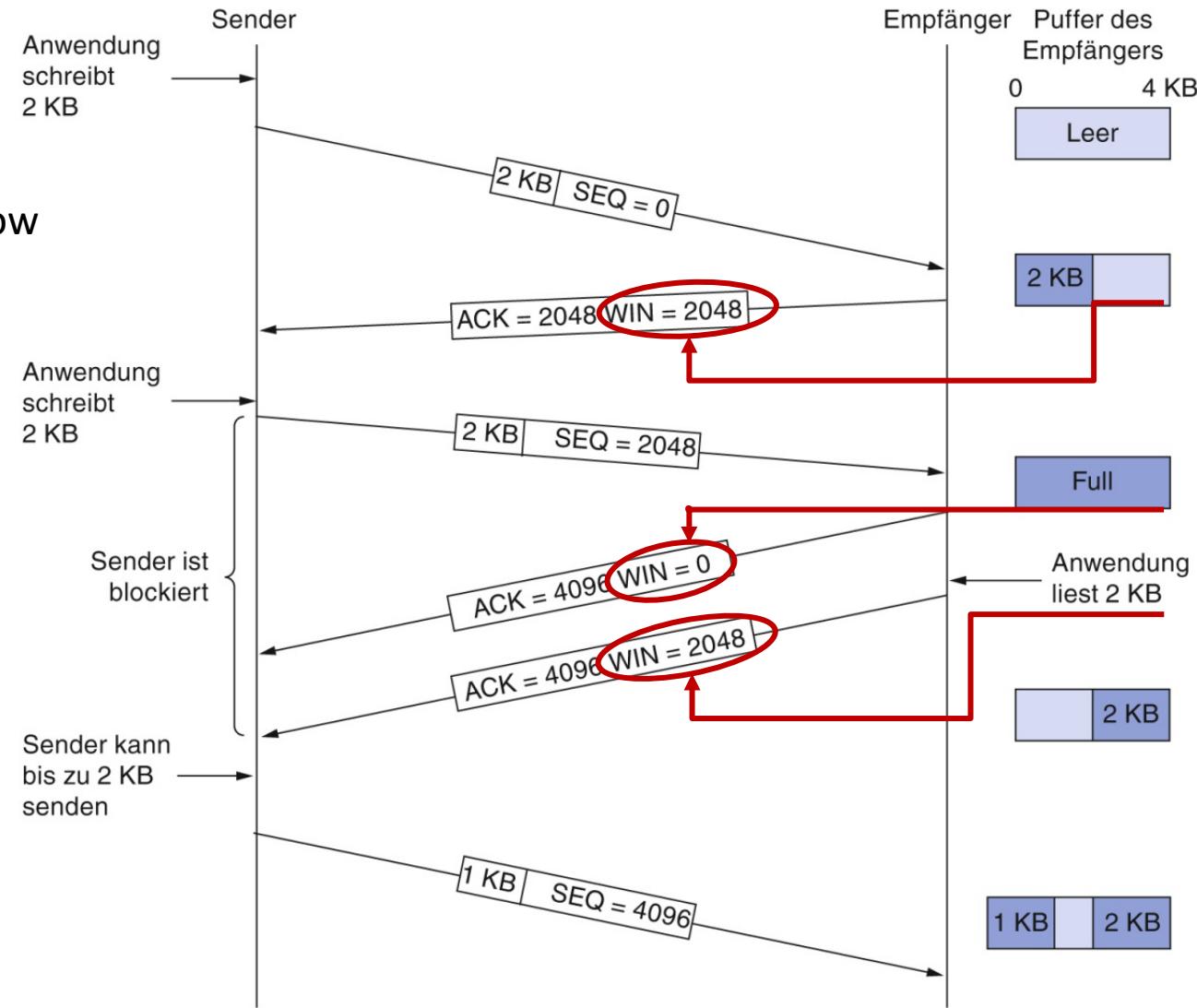
- Dient zum Angleichen der Geschwindigkeiten:
Senderate wird an die **Verarbeitungsrate der Anwendung** auf dem Empfänger **angepasst**
- Empfänger kündigt den Platz durch den **aktuellen Wert des RcvWindow** im TCP-Header der Antworten an
- Sender begrenzt seine unbestätigten gesendeten Daten auf RcvWindow

- Dann ist garantiert, dass der Puffer im Empfänger nicht überläuft

Flusskontrolle

Sender schickt nicht mehr Daten, als der Empfänger in seinem Puffer speichern kann

ACK bestätigt,
WIN = RcvWindow
gibt den noch
freien Puffer an



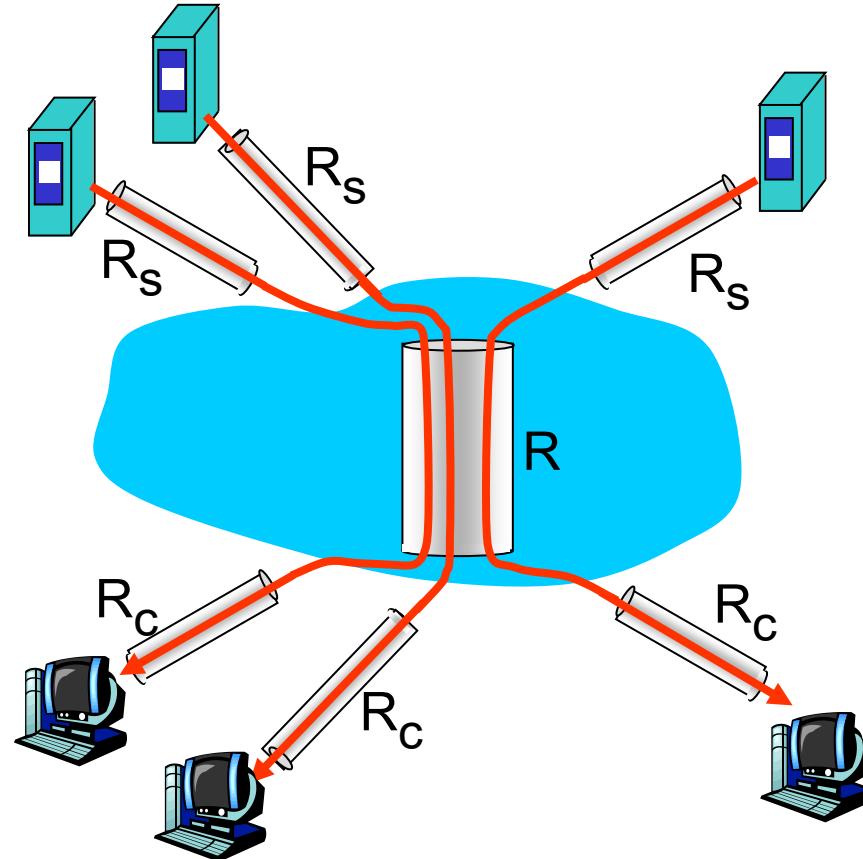
Internet = TCP / IP

→ *Details zu TCP*

→ *Überlastkontrolle*

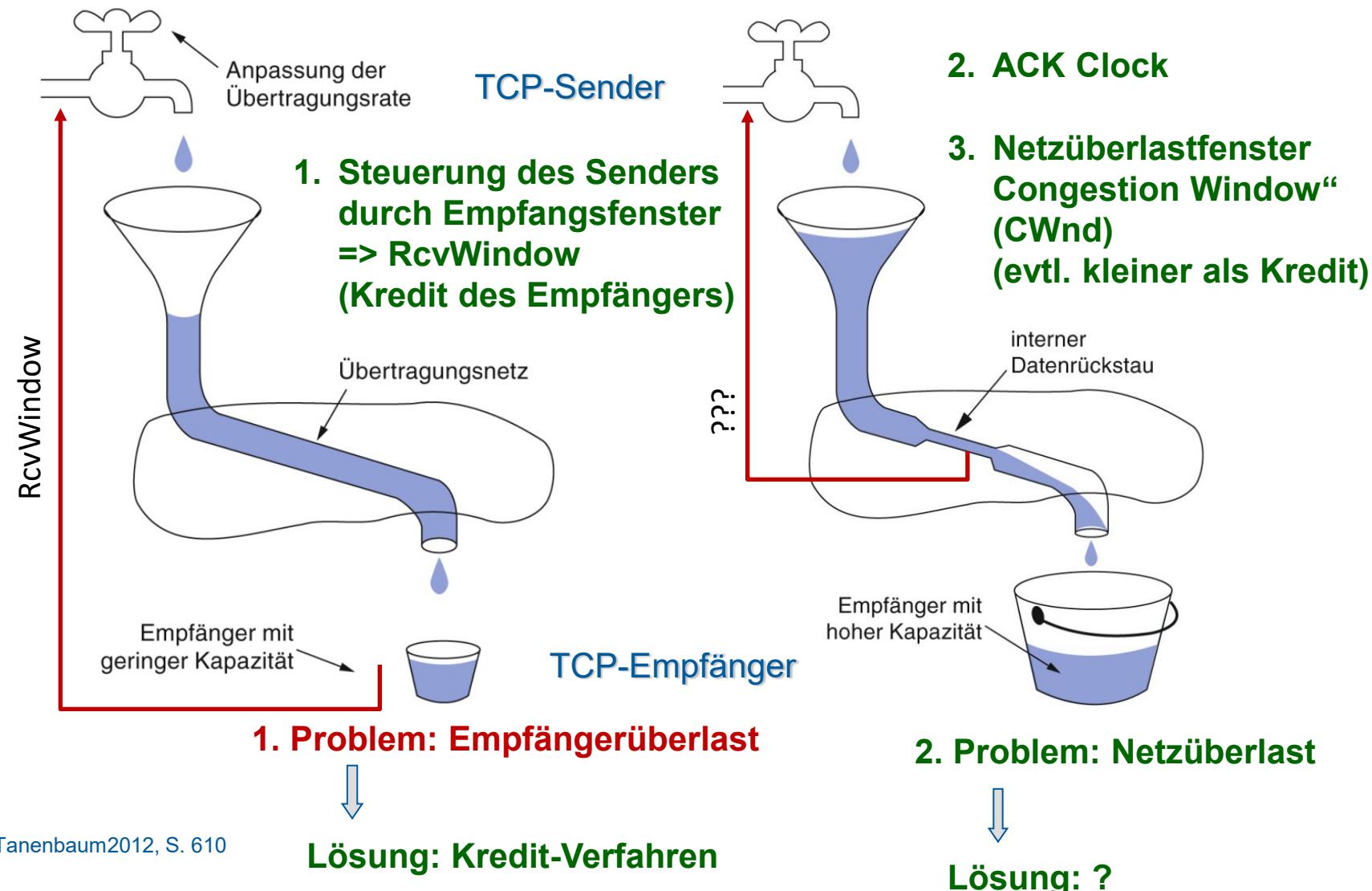
Überlast:

- Zu viele Systeme senden zu viele Daten, das Netzwerk kann nicht mehr alles transportieren
- Verschiedene von Flusskontrolle!
- Erkennungsmerkmale:
 - Paketverluste (Pufferüberlauf in den Routern)
 - Lange Verzögerungen (Warten in den Routern)
- Eines der zentralen Probleme in Computernetzwerken!



Viele Verbindungen teilen sich das Backbone-Netzwerkes

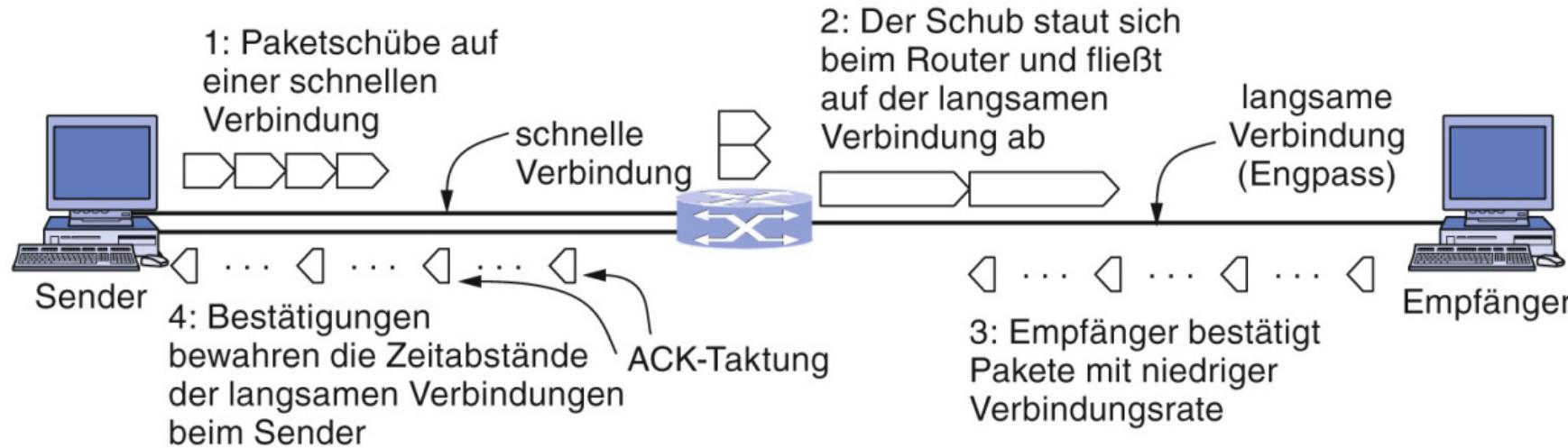
TCP-(Netz-) Überlastkontrolle



ACK Clock: Quittungen steuern den Datenfluss



Bsp.: Schnelle Verbindung \Leftrightarrow langsame Verbindung



- Langsames Eintreffen der ACK (durch die langsamere Leitung)
bremst das Senden neuer Segmente
- ➔ Glättung des Verkehrs
- „ACK Clock“ oder „ACK Taktung“

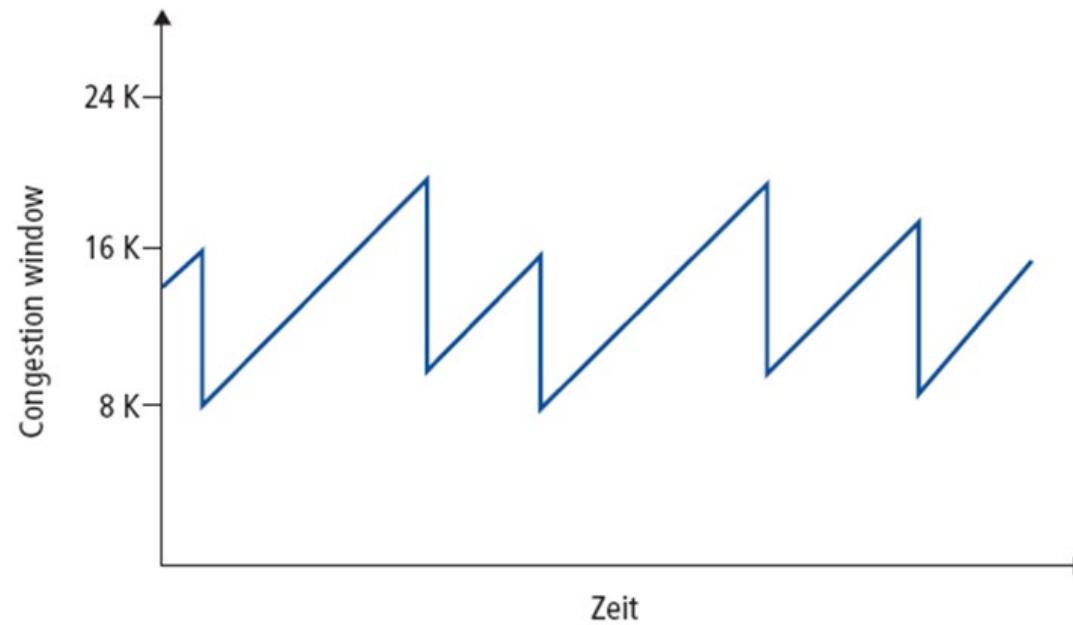
- **Flusskontrolle** regelt den Datenfluss **zwischen den Endsystemen**
 - Bei TCP: Fenstermechanismus mit Kreditvergabe (RcvWindow)
- **Staukontrolle** befasst sich mit Stau-Situationen **in den Zwischensystemen**
 - Bei TCP ist „congestion collapse“ möglich
 - Stau in Zwischensystemen führt meistens dazu, dass Transportprotokolle nach einer gewissen Zeit (aufgrund von Timeouts) **Dateneinheiten wiederholen**
→ **Verstärkung der Stausituation!**
- **Einführung eines Staukontrollfensters: „Congestion Window“ (CWnd)**

Zusätzlich zum Fenster für die Flusskontrolle

 - CWnd entspricht der Anzahl Bytes, die ein Sender jederzeit im Netz haben kann
 - Gesendet werden darf maximal bis zum Minimum beider Fenstergrößen
 - es muss somit immer gelten:
 - **LastByteSent – LastByteAcked \leq min { CWnd, RcvWindow }**

- Ansatz: Erhöhe die Übertragungsrate (Fenstergröße), um nach überschüssiger Bandbreite zu suchen, bis ein Verlust eintritt
 - Additive Increase: Erhöhe CWnd um **eine** MSS pro RTT, bis ein Verlust erkannt wird
 - Multiplicative Decrease: **Halbiere** CWnd, wenn ein Verlust erkannt wird
 - Ziel: Congestion Avoidance!

Das Ergebnis:
Ein Sägezahnverlauf:
Suchen nach
nutzbarer Bandbreite
durch die Steuerung
von Congestion
Window (CWnd)



CWnd: Congestion Window, MSS: Maximum Segment Size;
RTT: Round Trip Time

- Sender begrenzt die Übertragung:

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{CWnd}$$

Allgemein gilt:

$$\text{Rate} = \frac{\text{CWnd}}{\text{RTT}} \text{ Byte/s}$$

- CWnd ist dynamisch, hängt von der wahrgenommenen Netzwerklast ab
- **CWnd steuert den Durchsatz der Verbindung!**

CWnd: Congestion Window

- Wie erkennt der Sender Überlast?

- Verlustereignis =
 - Timeout oder
 - drei doppelte ACKs
- TCP-Sender verringert seine Rate (CWnd) nach einem Verlustereignis
- **Drei Mechanismen:**
 - AIMD
(Additive Increase, Multiplicative Decrease)
 - Slow Start
 - Vorsichtiges Verhalten nach einem **Timeout** => erneuter Slow Start

- Bei Verbindungsbeginn:

CWnd = 1 MSS

- Beispiel:

MSS = 500 Byte & RTT = 200 ms

- Initiale Rate = 20 kbit/s

- **Verfügbare Bandbreite kann viel größer als MSS/RTT sein**

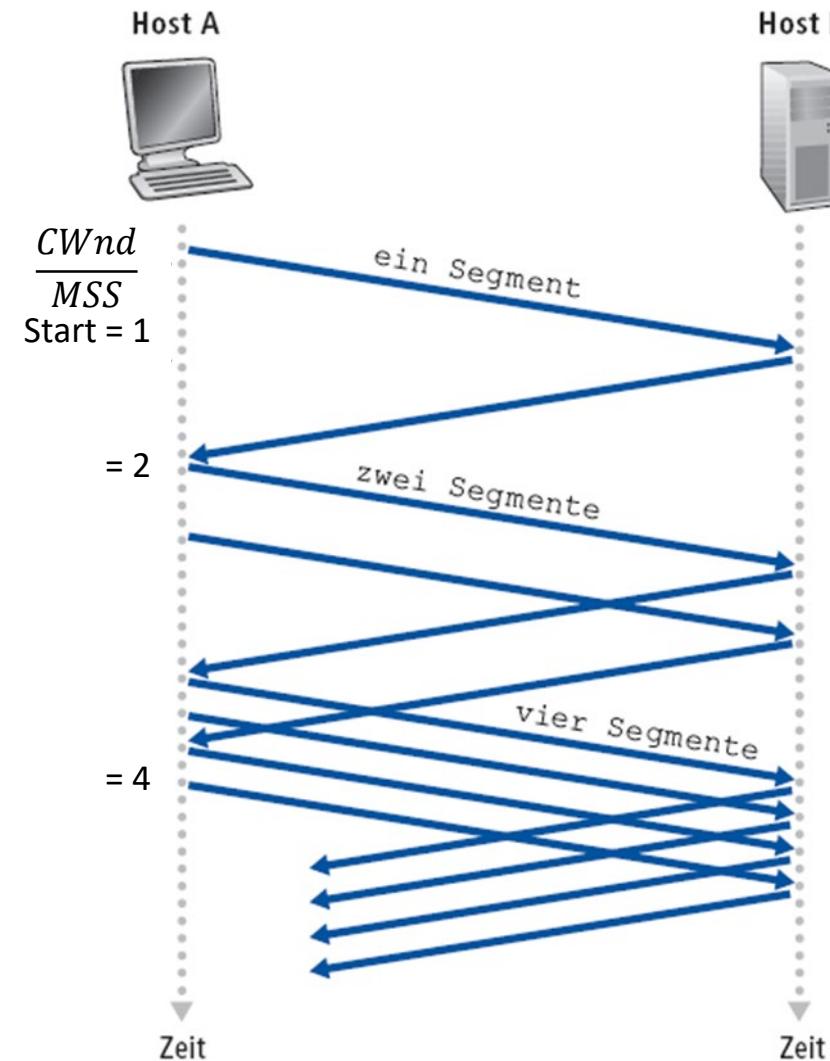
- **Die Rate sollte sich schnell der verfügbaren Rate anpassen**

- Bei Verbindungsbeginn:

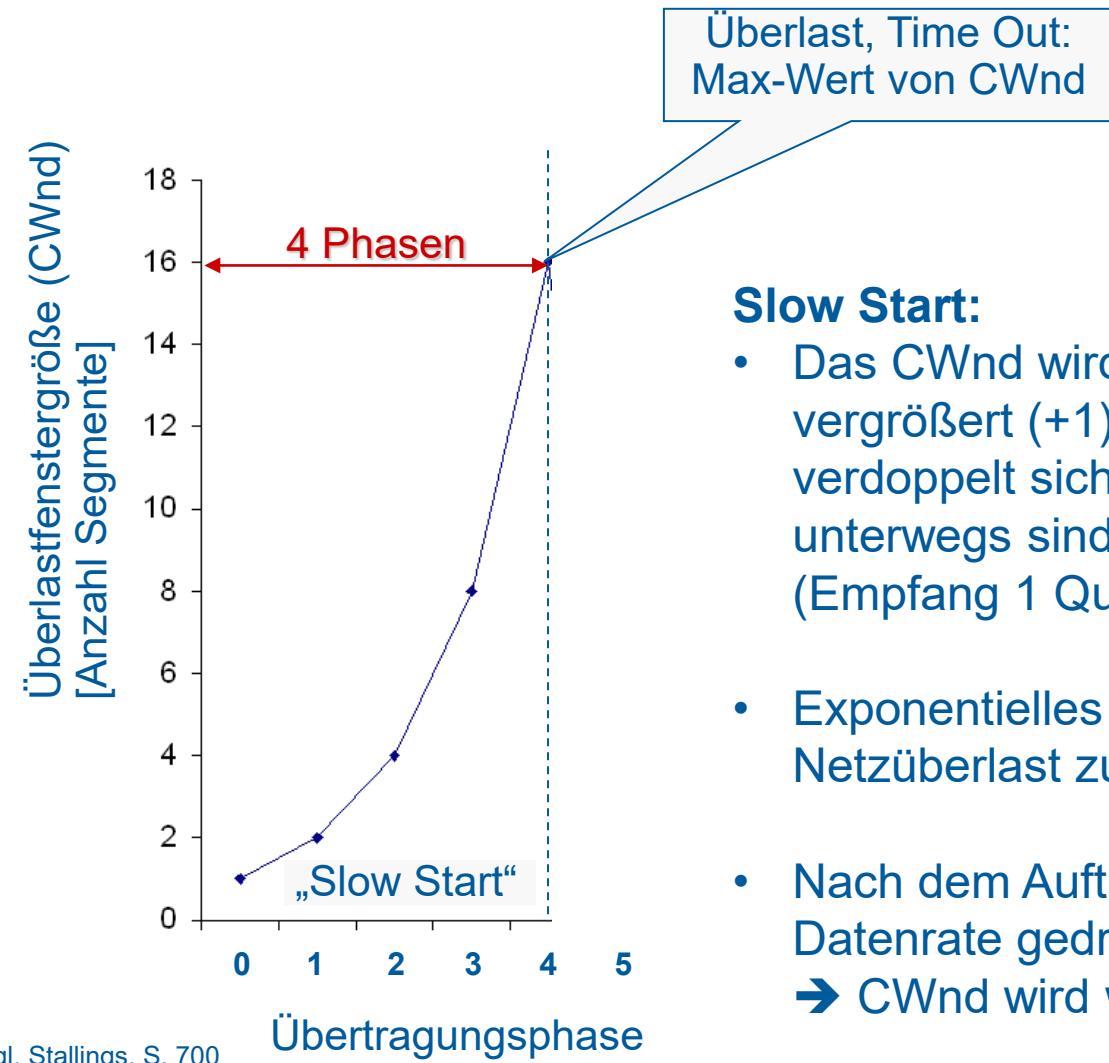
Erhöhe die Rate exponentiell schnell bis zum ersten Verlustereignis

- Verdoppeln von CWnd in jeder RTT

- Realisiert: CWnd um 1 für jedes erhaltene ACK erhöhen



Ergebnis: Initiale Rate ist gering, wächst aber exponentiell schnell



Slow Start:

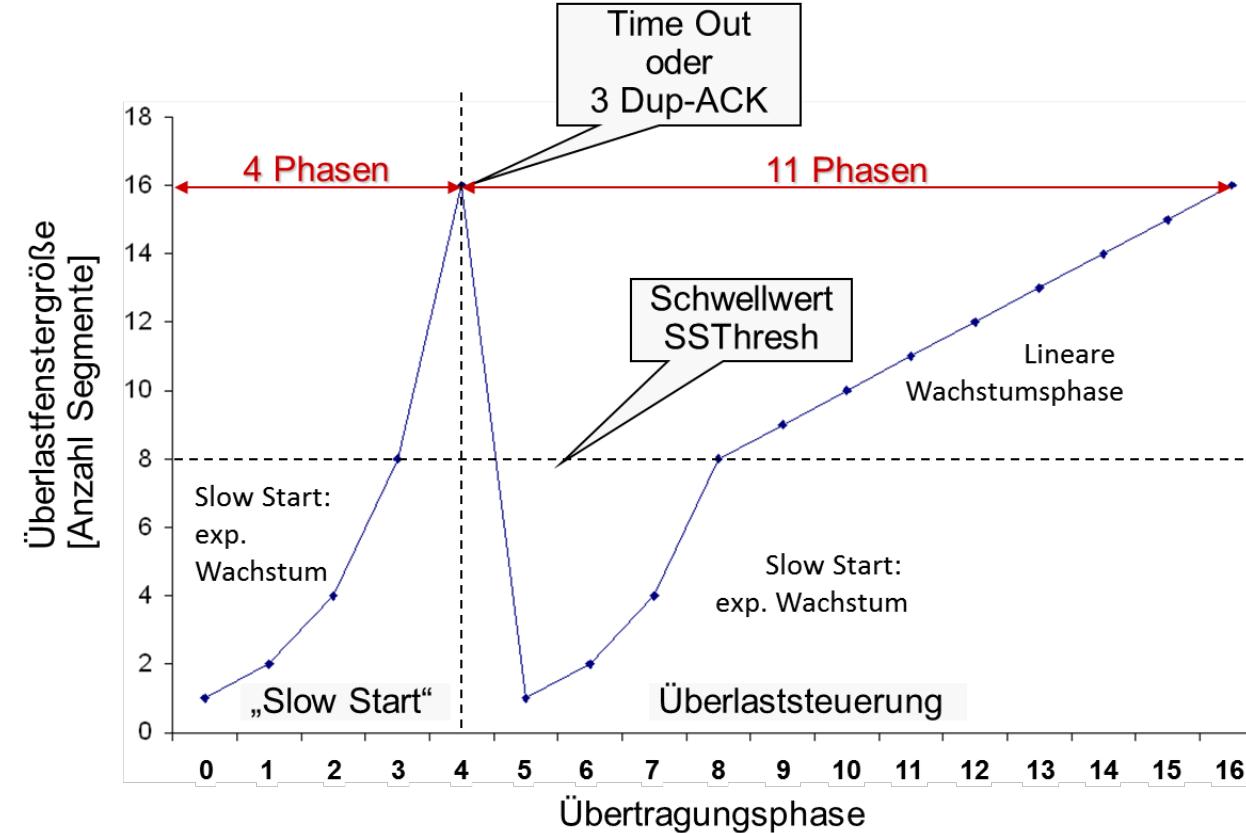
- Das CWnd wird nach jeder Bestätigung vergrößert (+1), nach jedem Umlauf verdoppelt sich die Anzahl der Pakete, die unterwegs sind (Empfang 1 Quittung => aussenden 2 Pakete)
- Exponentielles Wachstum der Paketrate bis Netzüberlast zum Time-Out führt
- Nach dem Auftreten der Überlast wird die Datenrate gedrosselt.
→ CWnd wird wieder auf 1 MSS gesetzt

- **Frage:** Wann soll vom exponentiellen Wachstum zum linearen Wachstum übergegangen werden?

- **Antwort:** Wenn CWnd die Hälfte des Wertes vor dem letzten Verlust-ereignis erreicht hat

Implementierung:

- Variable **Threshold (SSThresh)**
- Bei einem Verlustereignis wird Threshold auf die Hälfte von CWnd vor dem Verlustereignis gesetzt



Überlastalgorithmus nach TCP (1988)

Vers. Tahoe

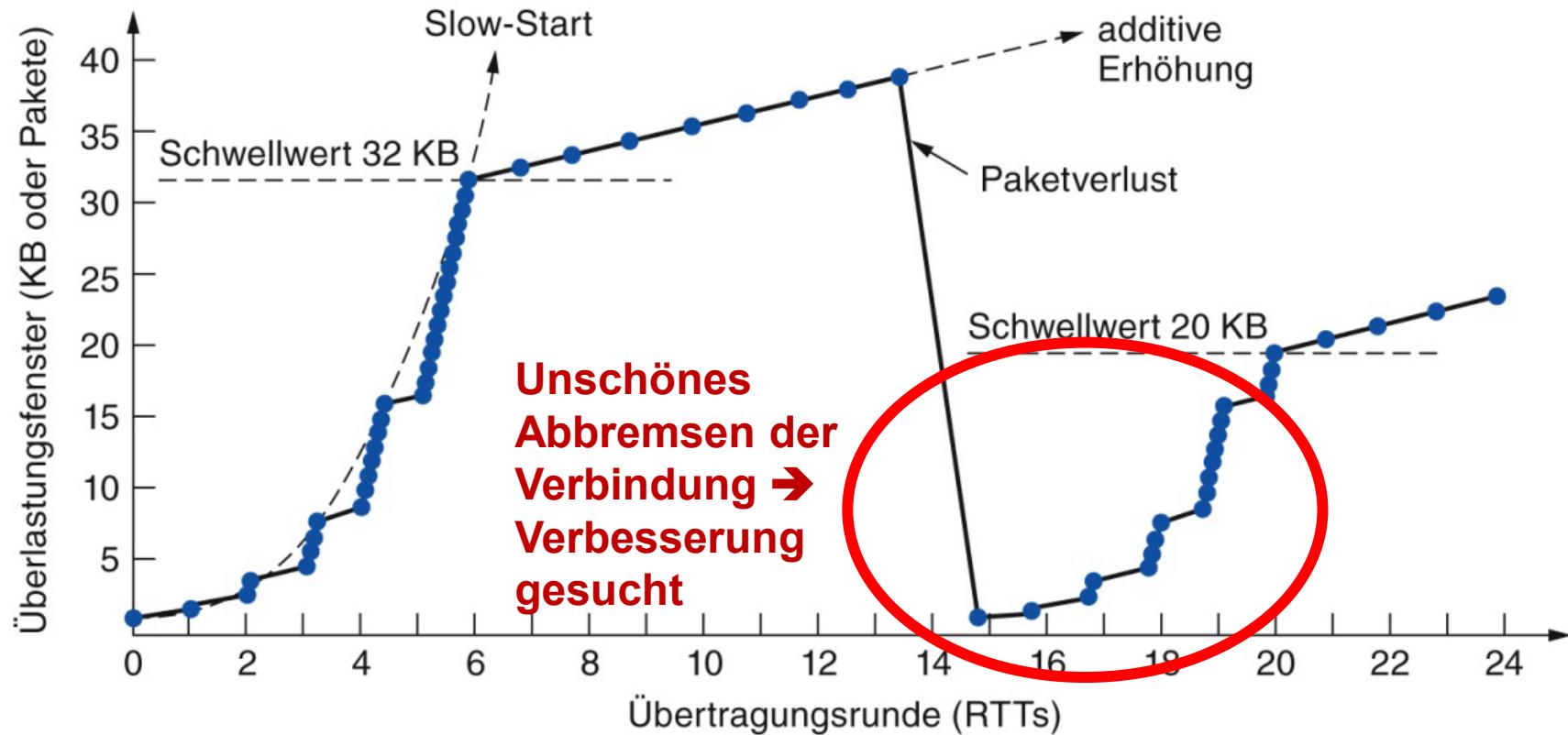
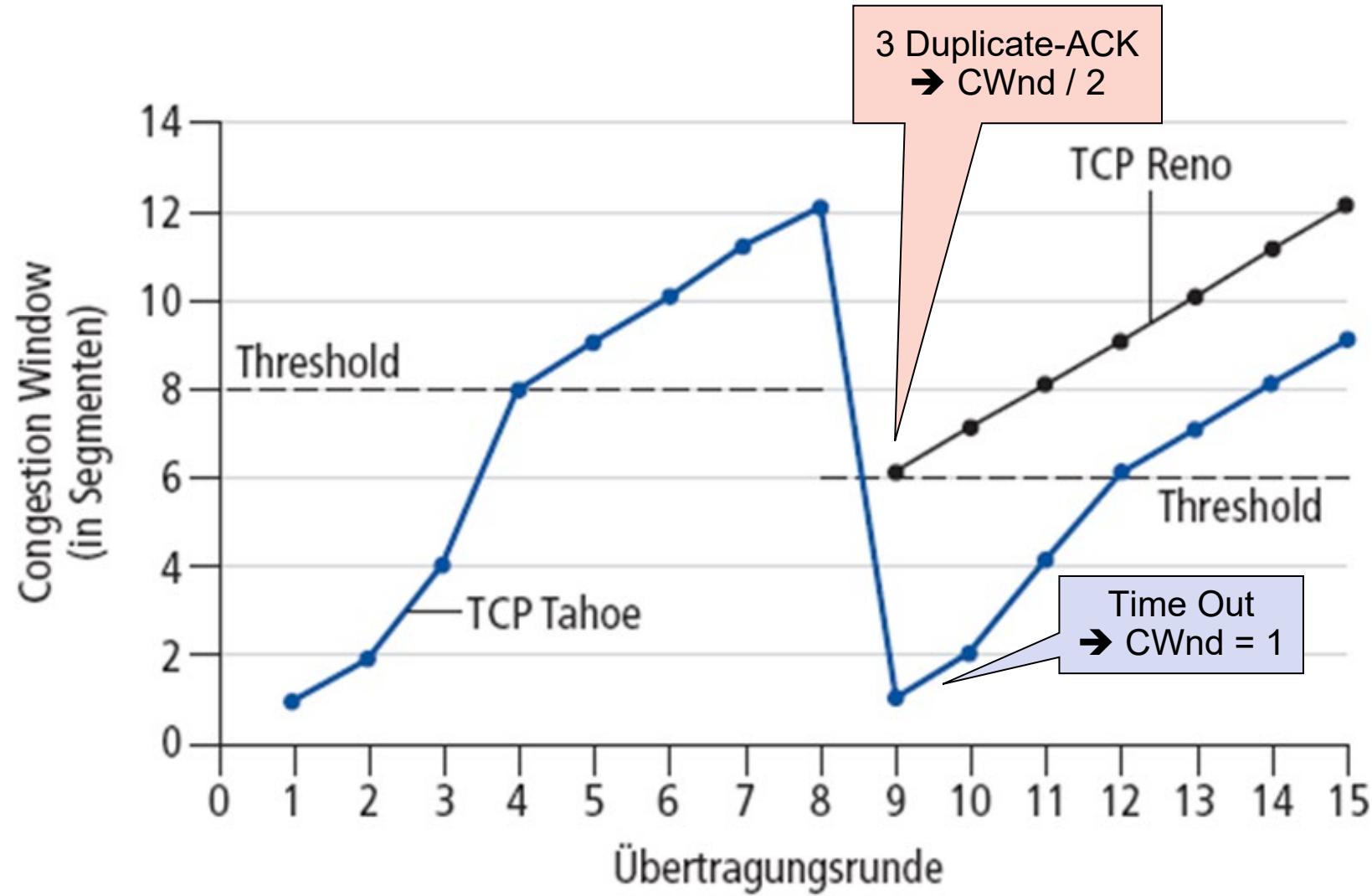


Abbildung 6.46: Slow-Start gefolgt von einer additiven Erhöhung in TCP Tahoe.

Philosophie:

- Drei doppelte ACKs zeigen an, dass das Netzwerk in der Lage ist, Pakete auszuliefern
 - Timeout ist “schlimmer”, weil gar keine Rückmeldung mehr erfolgt
-
- Nach drei doppelten ACKs:
 - CWnd halbieren,
 - Threshold auf diesen Wert setzen
 - Fenster wächst dann linear
 - Nach Timeout:
 - CWnd auf **eine MSS** setzen, Threshold auf die Hälfte des alten CWnd setzen
 - Fenster wächst dann exponentiell, bis Threshold erreicht ist
 - ... danach linear
-
- Wenn CWnd kleiner als Threshold ist, befindet sich der Sender in der **Slow-Start-Phase**, das Fenster wächst **exponentiell**.
 - Wenn CWnd größer als Threshold ist, befindet sich der Sender in der **Congestion-Avoidance-Phase**, das Fenster wächst **linear**.
 - Wenn **drei doppelte ACKs** für dasselbe Segment eintreffen, wird Threshold auf **CWnd/2** gesetzt und dann CWnd auf Threshold.
(=> **Fast Recovery, TCP Reno**)
 - Wenn ein **Timeout** auftritt, werden Threshold auf **CWnd/2** und CWnd auf **eine 1 MSS** gesetzt.

Vergleich der Reaktionen auf Paketverlust



SACK: Selektive Quittung bei TCP

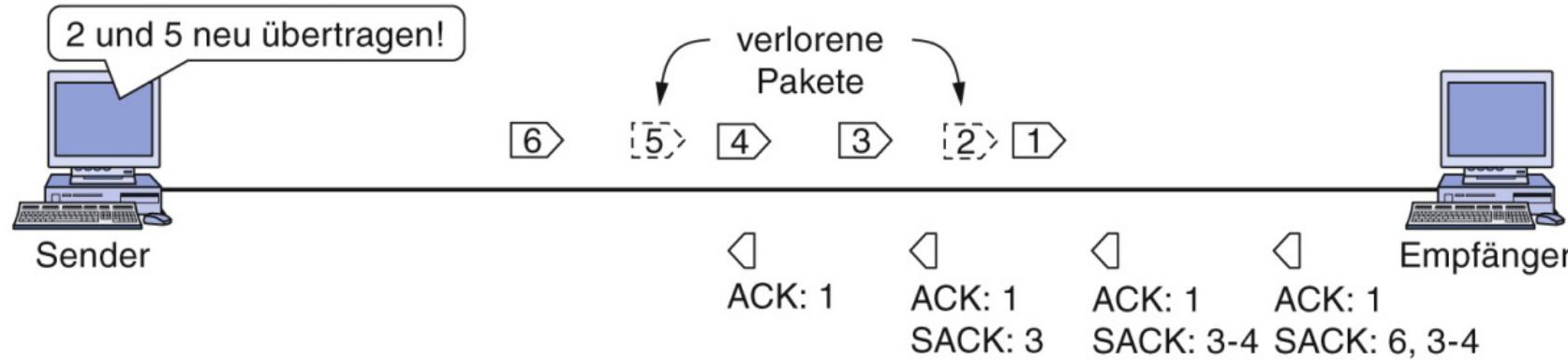


Abbildung 6.48: Selektive Bestätigungen.

- Im Optionsfeld von SACK wird genau angegeben, welche Bereiche (Segmente) fehlen
- SACK ist heute standardmäßig aktiviert; es werden also nur die fehlenden Segmente wiederholt und der Empfänger muss wieder sortieren.

| Zustand | Ereignis | Reaktion der TCP-Überlastkontrolle | Kommentar |
|---------------------------|---|--|---|
| Slow Start (SS) | ACK für zuvor unbestätigte Daten empfangen | $CongWin = CongWin + MSS$, Wenn ($CongWin > \text{Threshold}$), setze Zustand auf „Congestion Avoidance“ | Führt zu einer Verdopplung von $CongWin$ in jeder RTT. |
| Congestion Avoidance (CA) | ACK für zuvor unbestätigte Daten empfangen | $CongWin = CongWin + MSS \cdot (MSS / CongWin)$ | Additive Increase, resultiert in einer Zunahme von $CongWin$ um 1 MSS jede RTT. |
| SS oder CA | Verlustereignis entdeckt durch drei doppelte ACKs | $\text{Threshold} = CongWin / 2$, $CongWin = \text{Threshold}$, setze Zustand auf „Congestion Avoidance“ | Fast Recovery, implementiert Multiplicative Decrease. $CongWin$ kann nicht unter 1 MSS fallen. |
| SS oder CA | Timeout | $\text{Threshold} = CongWin / 2$, $CongWin = 1 MSS$, setze Zustand auf „Slow Start“ | Erneute Slow-Start-Phase |
| SS oder CA | Doppeltes ACK empfangen | Erhöhe den Zähler für doppelte ACKs für das bestätigte Segment | $CongWin$ und Threshold werden nicht verändert. |

Internet = TCP / IP

→ *Details zu TCP*

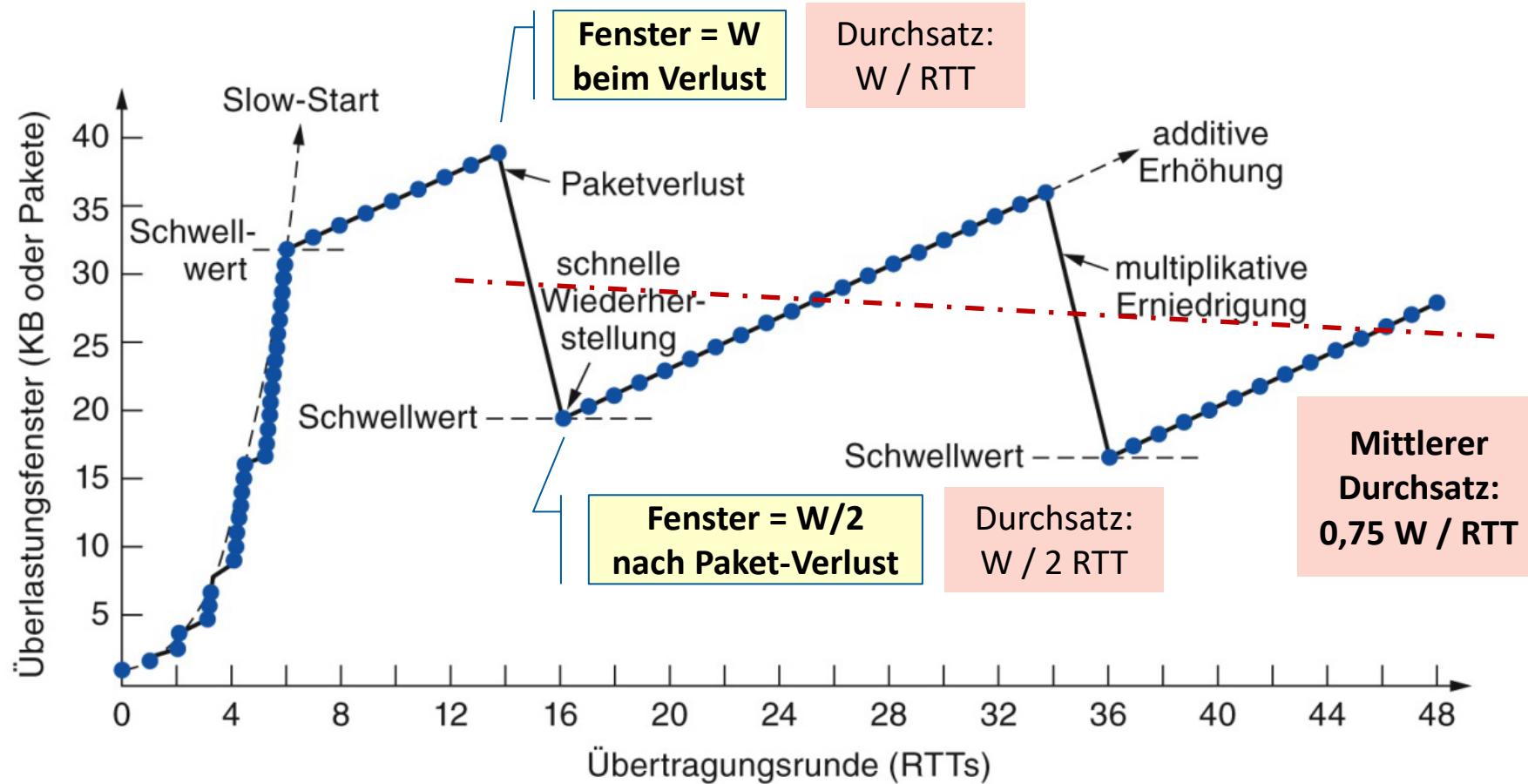
→ *Durchsatz*

- Was ist der durchschnittliche TCP-Durchsatz bei gegebener Fenstergröße W und RTT?
 - Vernachlässigen von Slow Start
- Sei W die Fenstergröße, wenn das Verlustereignis eintritt
- Wenn das Fenster W groß ist, dann ist der Durchsatz W/RTT
- Direkt nach dem Verlust verringert sich das Fenster auf $W/2$ und damit der Durchsatz auf $W/2RTT$
- Durchschnittlicher Durchsatz: $0.75 W/RTT$

Überlastalgorithmus TCP, Durchsatz

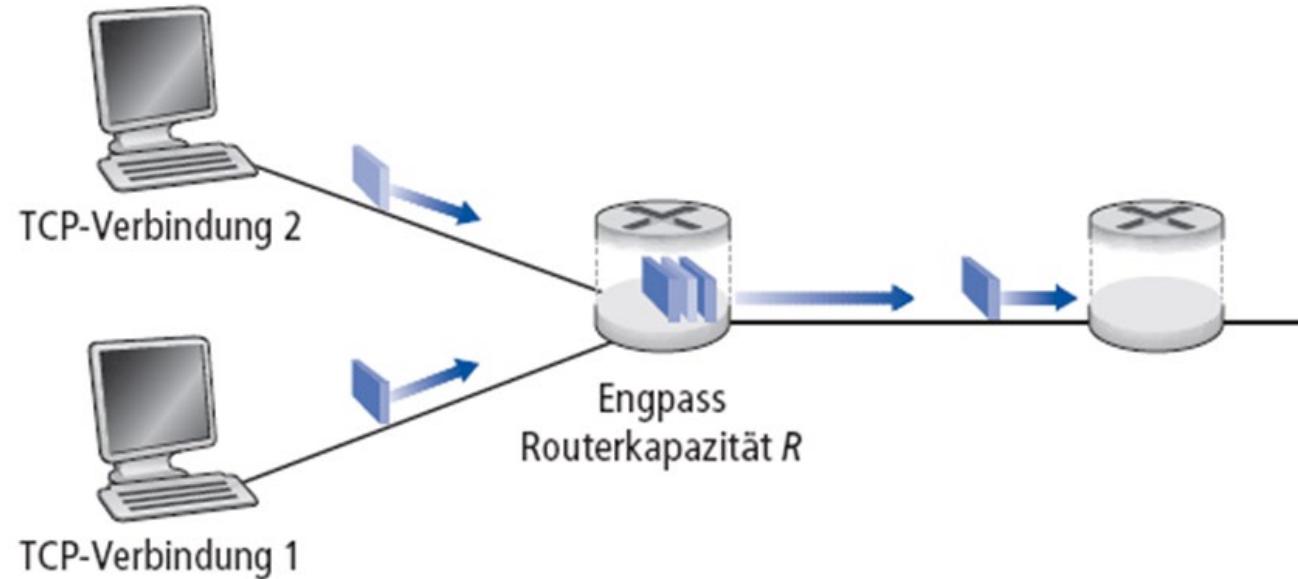


■ Vers. Reno (1990, bis heute aktuell)



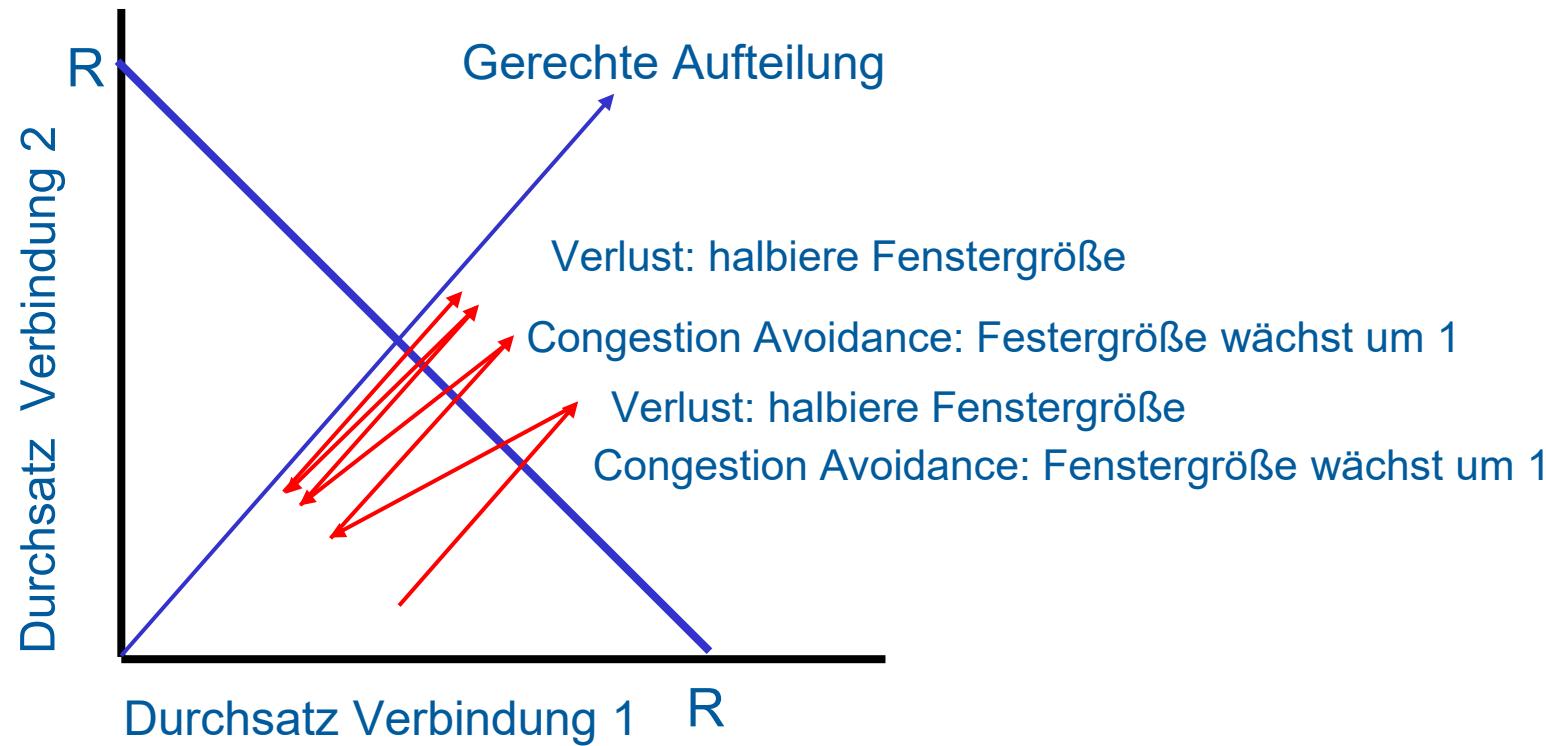
Beispiel mit Erkennen des Paketverlusts mit Duplicated ACK

- **Ziel:** Wenn K TCP-Verbindungen sich denselben Engpass mit Bandbreite R teilen, dann sollte jede eine durchschnittliche Rate von R/K erhalten
- Es soll keine Verbindung benachteiligt sein



Zwei Verbindungen im Wettbewerb:

- Additive Increase führt zu einer Steigung von 1, wenn der Durchsatz wächst
- Multiplicative Decrease reduziert den Durchsatz proportional



Fairness und UDP

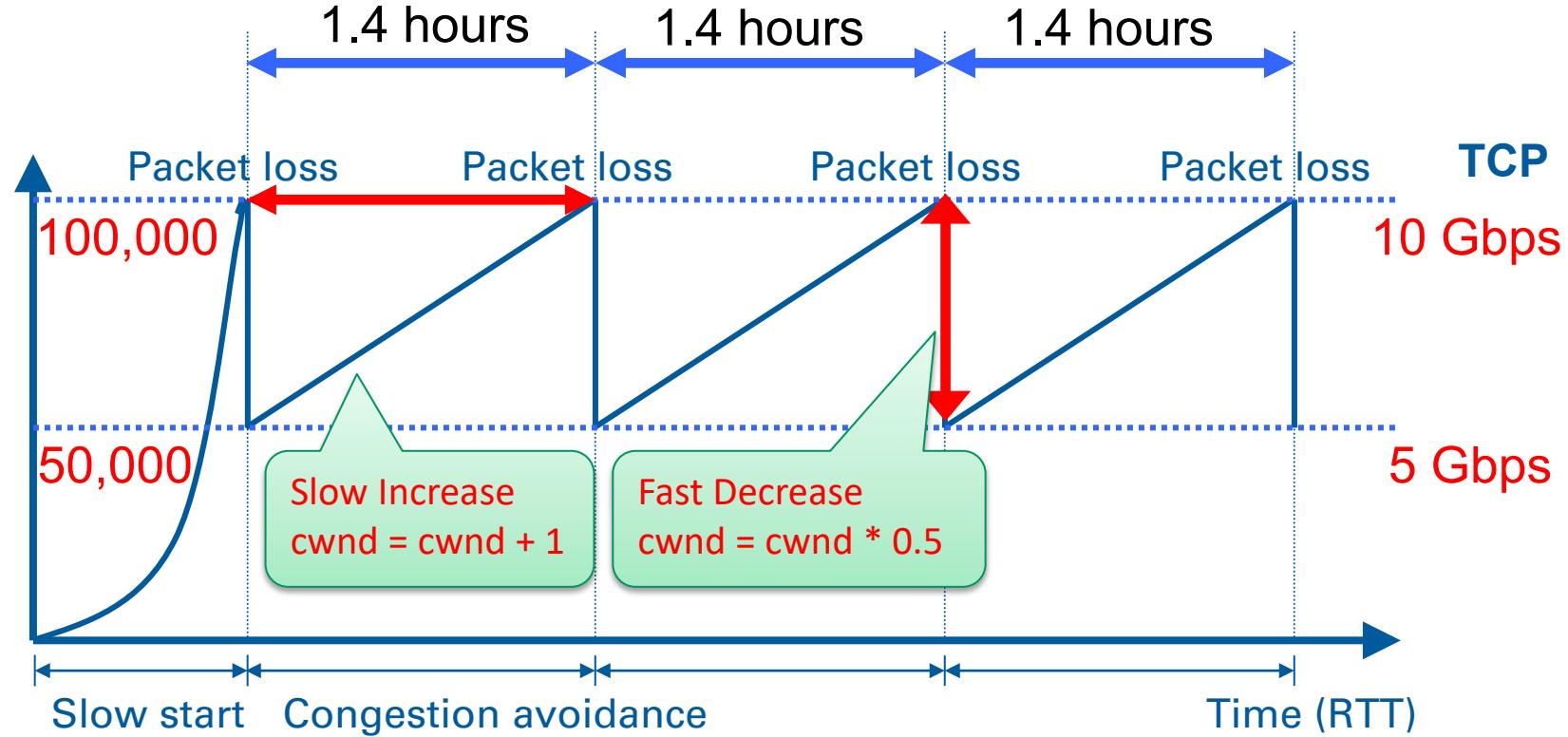
- **Viele Multimedia-Anwendungen verwenden kein TCP**
 - Wollen nicht, dass die Rate durch Überlast-Kontrolle reduziert wird
- **Stattdessen: Einsatz von UDP**
 - Audio-/Videodaten mit konstanter Rate ins Netz leiten, Verlust hinnehmen
- **Forschungsgebiet: TCP-Friendliness**
- **Neue Protokolle müssen sich an das Verhalten von TCP anpassen!**

Fairness und parallele TCP-Verbindungen:

- **Eine Anwendung kann zwei oder mehr parallele TCP-Verbindungen öffnen**
- **Webbrowser machen dies häufig**
- **Beispiel:**
 - Engpass hat eine Rate von R , bisher existieren neun Verbindungen
 - Neue Anwendung legt eine neue TCP-Verbindung an und erhält die Rate $R/10$
 - Neue Anwendung legt elf neue TCP-Verbindungen an und erhält mehr als $R/2$!

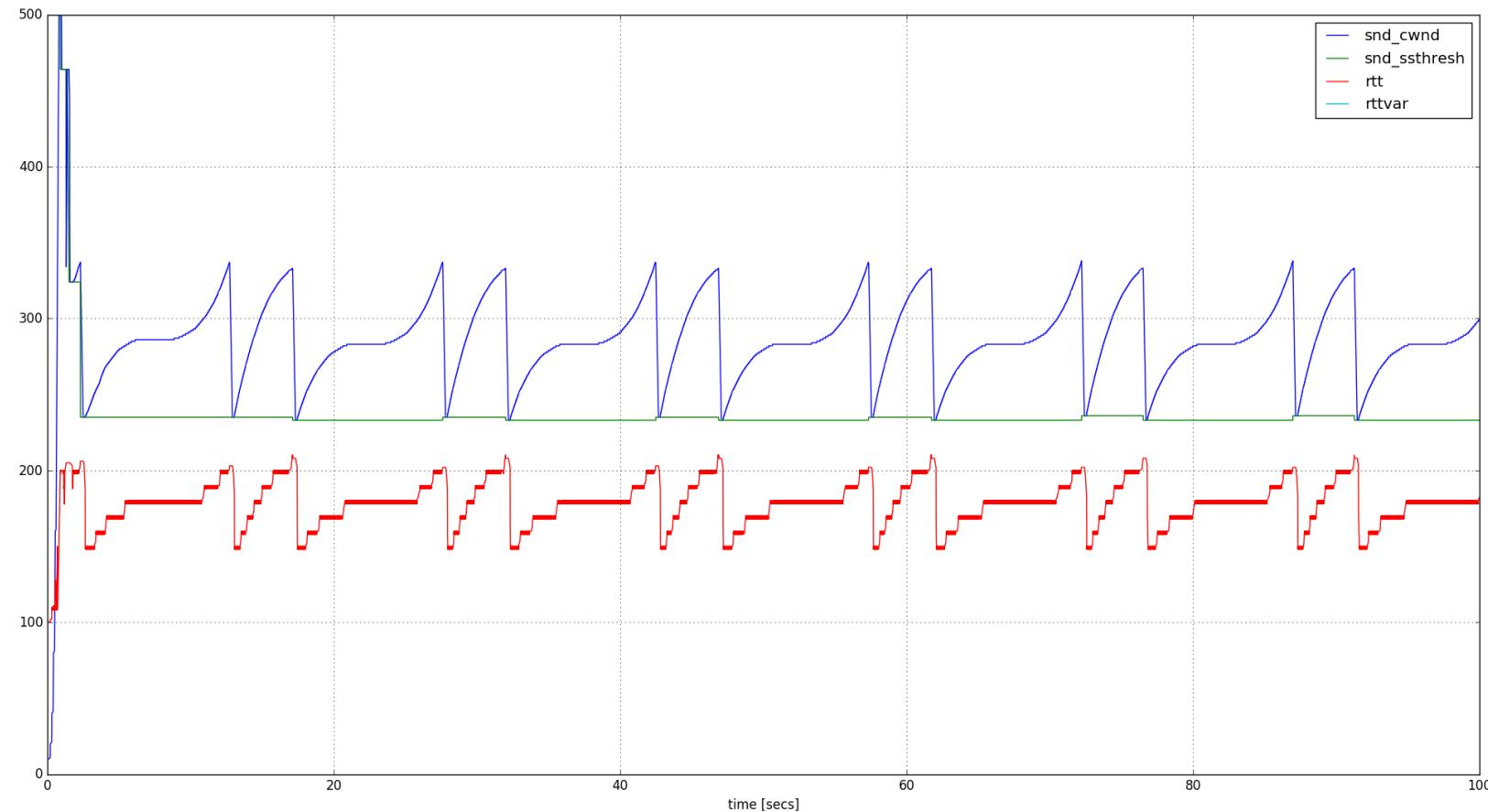
- **Problem:** Netze mit sehr hohen Datenraten und langen Laufzeiten

→ Hohes Bandwidth*Delay Product (BDP)



Presentation: "Congestion Control on High-Speed Networks", Injong Rhee, Lisong Xu, Slide 7

Meist wird vom Linux heute der Cubic-Algorithmus zur TCP-Congestion Control verwendet.



- ACK received

$$cwnd = C \cdot (t - K)^3 + W_{\max}$$

- **C** is a scaling factor
- **t** is the elapsed time from the **last window reduction**
- **Wmax** is the window size just before the last window reduction
- **K** is updated at the time of **last lost event**

- Recovery

- Update **K** with:

$$K = \sqrt[3]{\beta \cdot W_{\max} / C}$$

- Update **Wmax** with:

$$W_{\max} = \beta \cdot W_{\max}$$

- **β** is a constant multiplication decrease factor

➔ Cubic wird durch Verluste gesteuert (wie Reno)

■ ACK received

$$cwnd = C \cdot (t - K)^3 + W_{\max}$$

- **C** is a scaling factor
- **t** is the elapsed time from the last window reduction
- **Wmax** is the size of the last window

$$cwnd = \beta \cdot W_{\max} + 3 \cdot \frac{1-\beta}{1+\beta} \cdot \frac{t}{RTT}$$

as to keep the growth rate the same as standard TCP in **short RTT** networks.

■ Recovery

- Update **K** with:

$$K = \sqrt[3]{\beta \cdot W_{\max} / C}$$

- Update **Wmax** with:

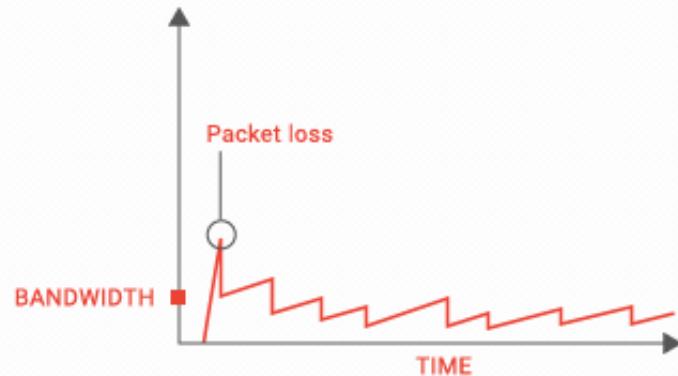
$$W_{\max} = \beta \cdot W_{\max}$$

- **β** is a constant multiplication decrease factor

- **Problem:** Netze mit sehr hohen Datenraten und langen Laufzeiten
→ Hohes Bandwidth*Delay Product (BDP)
 - Die bekannten Congestion Control Verfahren (Reno, Cubic) reagieren auf **Verluste** und drosseln dann den Sender.
 - Bei Strecken mit hohen BDP (lang + sehr schnell) sind aber sehr viele Bytes „unterwegs“ (Bytes in flight)
Bsp: 10 Gbit/s Leitung, 20 ms Laufzeit (4000 km) => 25 Mbyte in flight
 - Google hat 2017 einen neuen Ansatz vorgestellt, das versucht den Durchsatz von Streaming-Anwendungen zu erhöhen (Google Cloud Platform, → Youtube)
 - BBR ("Bottleneck Bandwidth and Round-trip propagation time"): Algorithmus sucht nach Min. RTT und max. Bandbreite durch "ausprobieren".
- Präsentation vom IETF Nov 2016 dazu

TCP before BBR

Today's Internet is not moving data as well as it should. TCP sends data at lower bandwidth because the 1980s-era algorithm assumes that packet loss means network congestion.

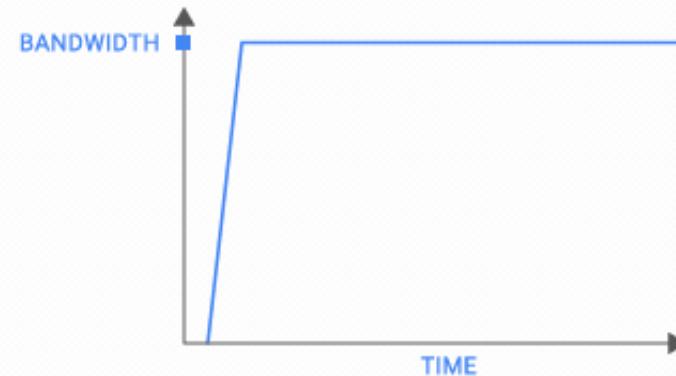


[Quelle: <https://cloudplatform.googleblog.com/2017/07/TCP-BBR-congestion-control-comes-to-GCP-your-Internet-just-got-faster.html>]

→ Unklar: Wie verhalten Sie gemischte Verkehre in einem Netz?
(BBR und Cubic und Reno)

TCP BBR

BBR models the network to send as fast as the available bandwidth and is 2700x faster than previous TCPs on a 10Gb, 100ms link with 1% loss. BBR powers google.com, youtube.com, and apps using Google Cloud Platform services.



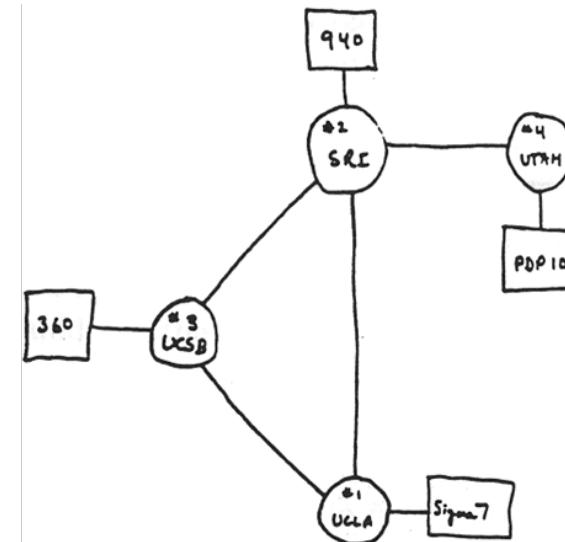


Internet

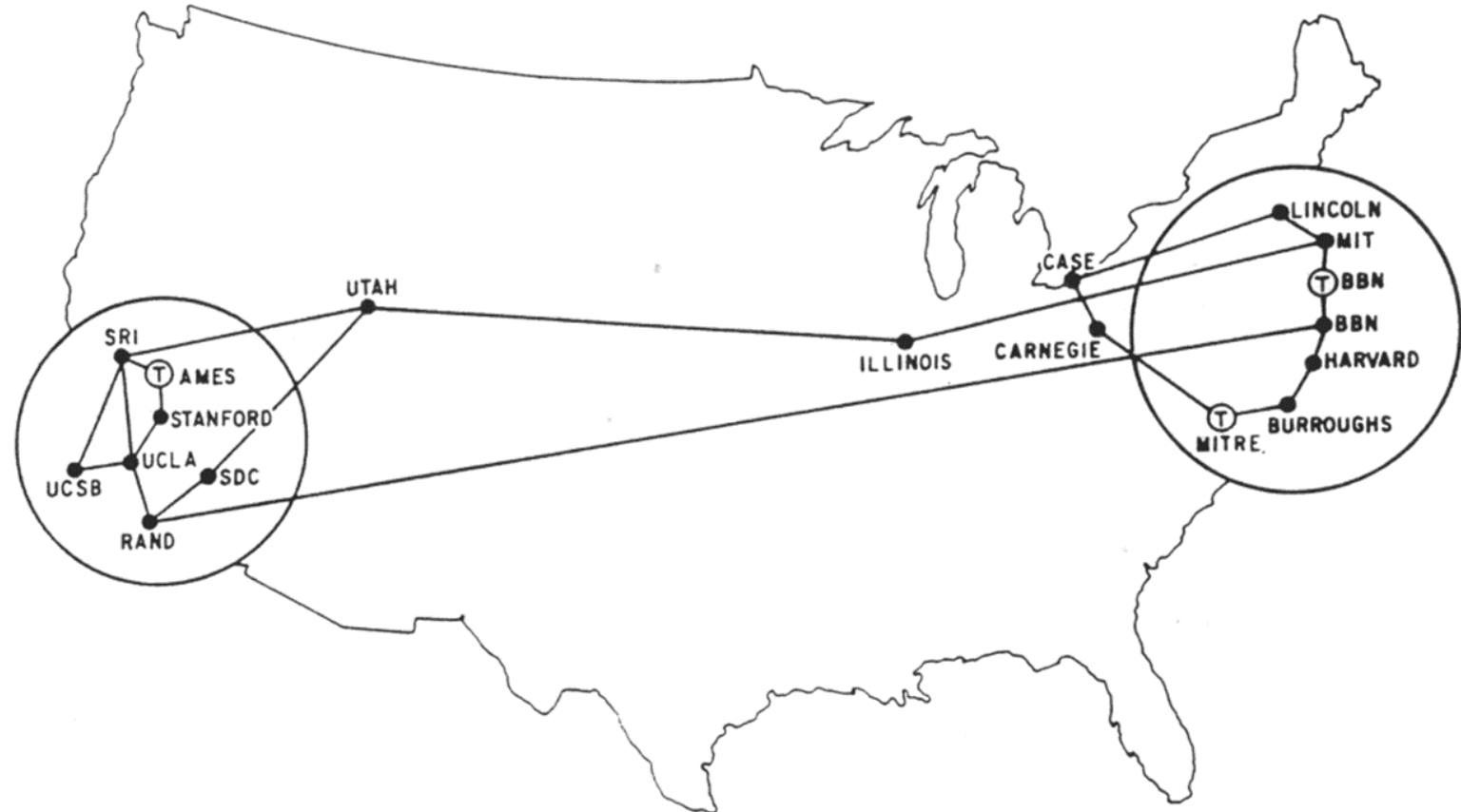
→ *Geschichte des Netz*

1961–1972: Paketvermittlung

- 1961: Leonard Kleinrock – Warteschlangentheorie zeigt die Effizienz der Paketvermittlung
- 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



DEC 1969



MAP 4 September 1971

Rechnernetz für Forschung,
Start mit 50 kbit/s Übertragungsrate über Standleitungen

1972–1980: Netzwerk von Netzwerken

- 1970: ALOHAnet Satellitennetzwerk auf Hawaii
- 1973 erste internationale ARPANET Knoten in England und Norwegen (insgesamt 37 Hosts)
- 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

ARPA: Advanced Research Projects Agency

Cerf und Kahn definieren die 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

1980–1990: Neue Protokolle, Ausbreitung des Netzwerkes

- 1982: Definition des SMTP-E-Mail-Protokolls
- 1983: Einführung von TCP/IP als netzwerkübergreifendes Protokoll
(Im Netz sind 562 Host)
- 1983: Definition von DNS zur Übersetzung von Namen auf IP-Adressen
- 1985: Definition von ftp
- 1988: Überlastkontrolle in TCP
- 1984 Aufspaltung:
 - militärischer Teil → MILNET,
 - ziviler Teil → Internet
- 1988 1,5 Mbit/s Backbone der NSF (National Science Foundation);
(insgesamt 56000 Hosts)
- 1990 NSFNet auf 45 Mbit/s Links aufgerüstet;
(313 000 Hosts)

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

200X - heute: Video und Mobilfunk

2007:

- ~ 500 Millionen Endsysteme
- Voice, Video over IP
- P2P-Anwendungen: BitTorrent (Filesharing) Skype (VoIP), PPLive (Video)
- Mehr Anwendungen:
 - YouTube (Start 2005),
 - Gaming (allein World of Warcraft hat 9 Millionen Kunden)
- Mobilkommunikation
 - Mit dem iPhone beginnt die Zeit der mobilen Datenkommunikation
 - Smartphones

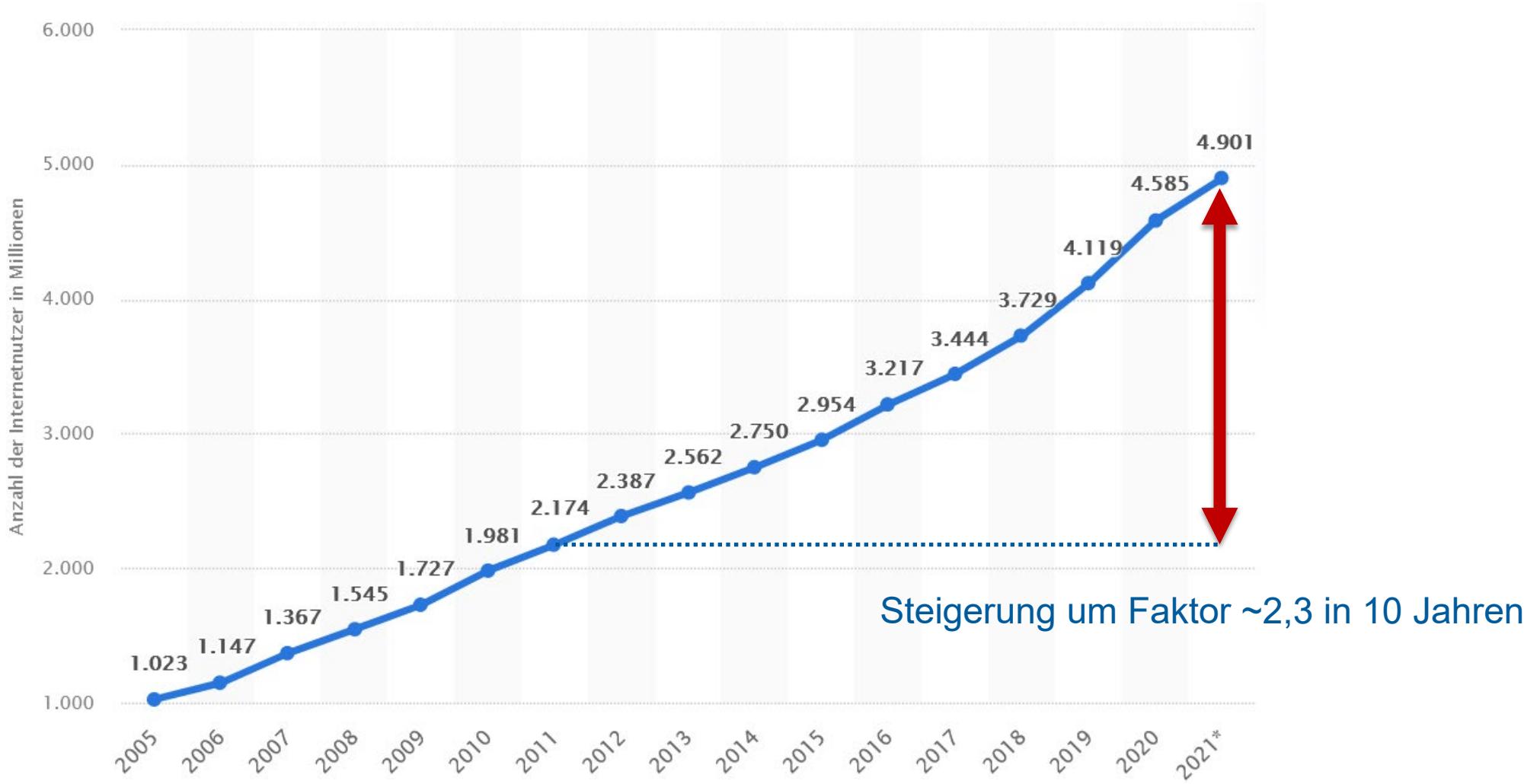
Seit 2008 Breitband überall:

- Weiter ungebremstes Wachstum
- Applikationen (Video) brauchen höhere Datenraten
- Smartphones werden das Medium zur Internet-Nutzung
- Breitband zum Haus (DSL, Kabel)
- Mobil im Haus: WLAN / WiFi
- Mobilfunk für die Smartphones

- Alle Kommunikationsnetze übertragen IP (All-IP)
- Mobil: LTE ist nur IP
- Telefon: seit 2018 nur IP

Sicherheit!!

Anzahl der Internetnutzer weltweit 2005 bis 2021 (Mio)



[Quelle: ITU, Statistica, 2021 geschätzt]

2018+: Was sehen wir heute

- **Mobiles Internet wird weiter stark wachsen, speziell in den Ländern ohne gute Infrastruktur.**
- **Nach der Vernetzung der Menschen kommt nun die Vernetzung der „Dinge“**
 - IoT: Internet of Things
 - Smart
 - Car 2 X
 - Industrie 4.0
- **Nach der Kommunikation wird die Bearbeitung der Daten der „Dinge“ entscheidend:**
 - Big Data
 - Datenbanken
 - Analytics
- **Alle Kommunikationsdienste werden über 1 IP-Netz geführt werden**
 - Sprache (Telefonie)
 - TV / Video
 - Mobilfunk (LTE / 5G)
- **Die Sicherheit und Stabilität dieses Netzes ist entscheidend für uns.**
- **Leider hat auch die „dunkle Seite“ das Internet entdeckt und kann es benutzen**
 - Attacken werden weiter zunehmen
 - Verwundbarkeit der Gesellschaft im Internet nimmt zu
 - Kriminelle wenden immer ausgefeilte Techniken an
 - Der Wettlauf um ein sicheres Netz geht weiter
=> Halten Sie Ihre Software auf den aktuellsten Stand!!
=> Sind Sie vorsichtig und skeptisch!



Internet

→ *Aspekte der Sicherheit*

Bekannte Angriffe auf die Infrastruktur des Internets sind:

- Kompromittieren/Angreifen von Endsystemen: Malware, Spyware, Würmer, unberechtigter Zugriff (Diebstahl von Daten und Accounts)
- Denial of Service: den Zugang zu Ressourcen verhindern (Server, Bandbreite)

Das Internet wurde nicht mit dem Ziel Sicherheit entworfen

- Vision beim Entwurf des Internet:

“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 Protokollschichten untersucht!
 - Zahlreiche Ergänzungen und Verbesserungen um die Sicherheit zu erhöhen
- ➔ Steter Wettlauf zwischen den Angreifern und den Sicherheitsmaßnahmen / Verbesserungen

■ Spyware:

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

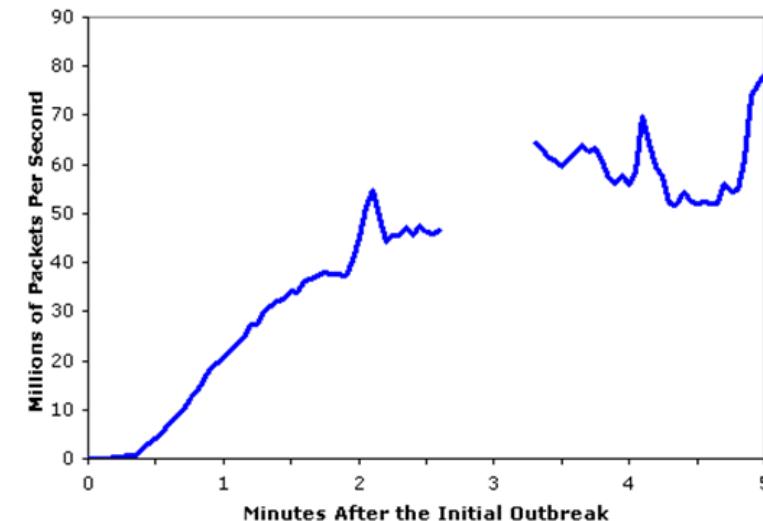
■ Virus

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

■ Würmer:

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

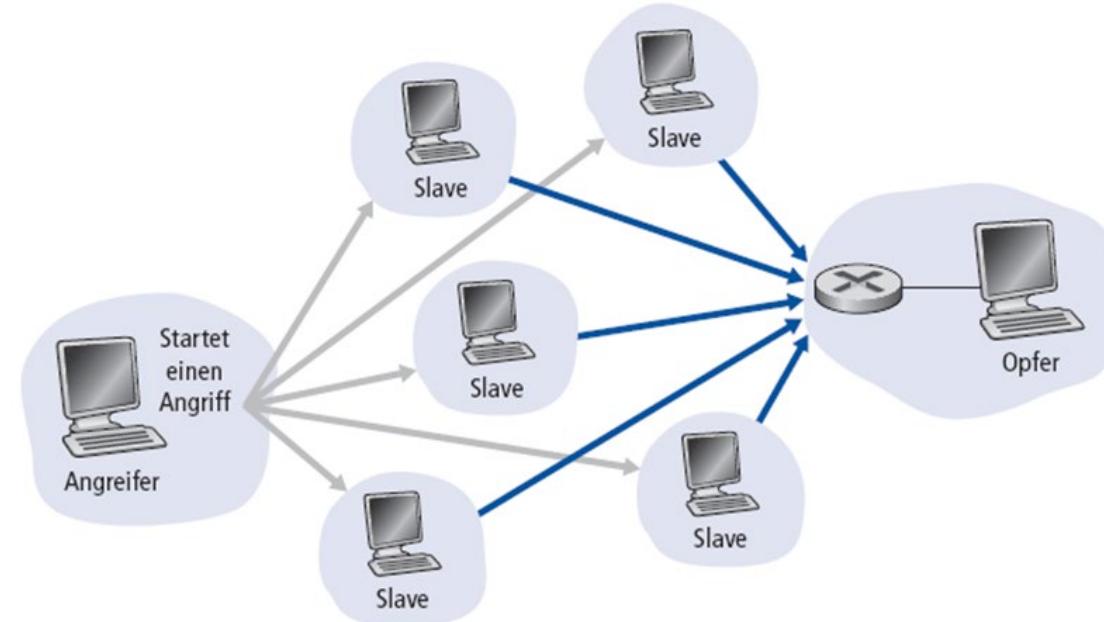
Aggregate Scans/Second in the first 5 minutes based on Incoming Connections To the WAIL Tarpit



- Angreifer verhindern den Zugriff von Benutzern auf Ressourcen (Server, Bandbreite), indem diese durch den Angreifer belegt werden

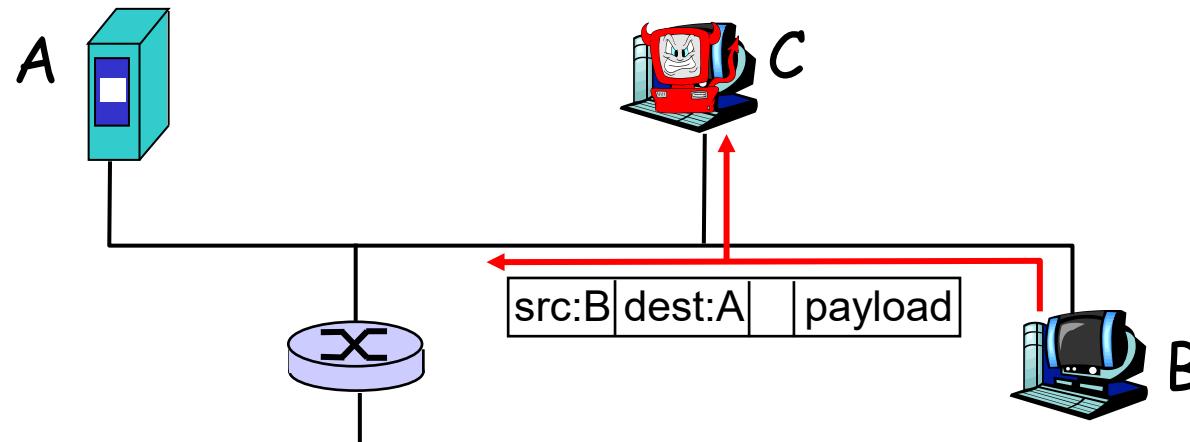
- Vorgehen:

1. Wähle ein Ziel
2. Kompromittiere andere Systeme
3. Sende viele Pakete von den kompromittierten Systemen an das Ziel



■ Mithören von Paketen:

- Broadcast-Medien (Ethernet, WLAN)
- Netzwerkkarten im Promiscuous Mode zeichnen **alle!** Pakete auf, die sie hören können
- Dafür kann z.B. Wireshark verwendet werden



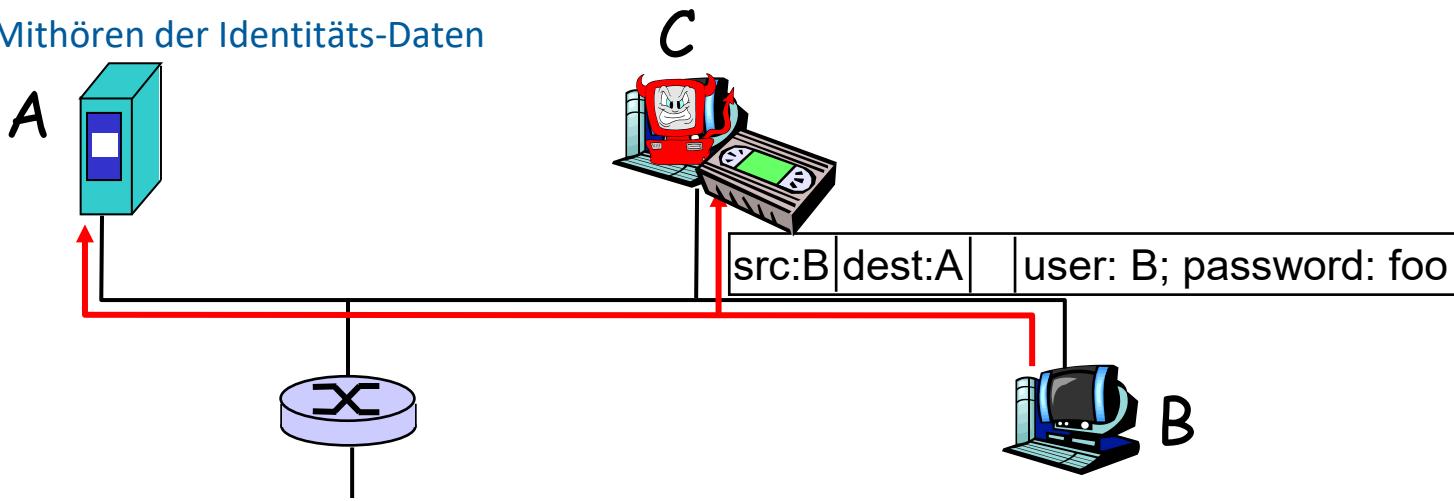
Identität-Diebstahl: Sich als jemand anderes ausgeben

■ Aufzeichnen und Abspielen:

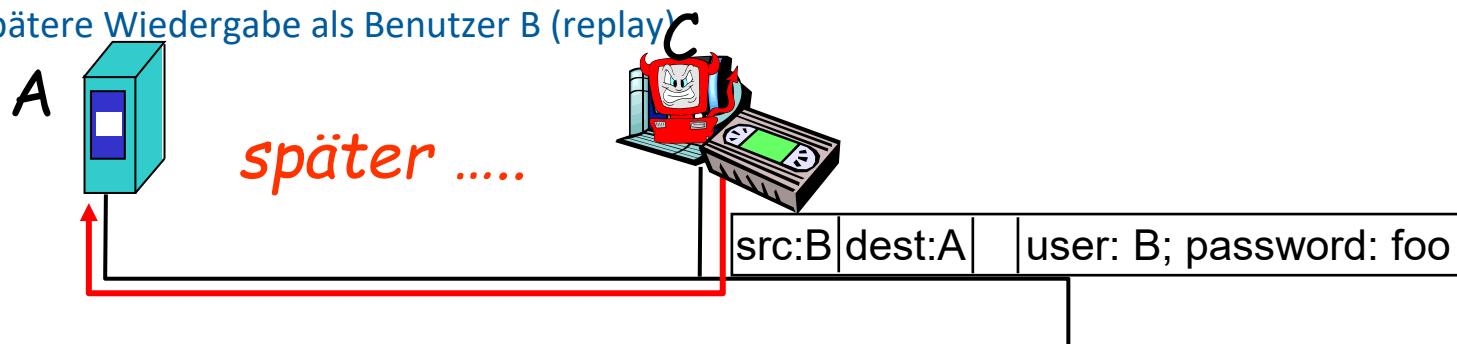
Sicherheitsrelevante Informationen (z.B. Passwort) mithören und später verwenden

- Aus Sicht des Systems ist jemand, der das Passwort eines Benutzers kennt, dieser Benutzer!

1. Mithören der Identitäts-Daten



2. Spätere Wiedergabe als Benutzer B (replay)



Weitere Aspekte für die Sicherheit (Safety) der Infrastruktur des Internets sind:

- **Technische Störungen in den Komponenten des Netzes:**
 - Unterbrechungen von Leitungen (Glasfaser, Kupfer, ...)
 - Störungen der Funkstrecken (Fading, Überlagerungen, Wetter,...)
 - Ausfälle von Komponenten (Laser, Halbleiter, Spannungswandler im System, Fan,...)
 - Ausfälle von gesamten Knoten (Router, WDM-Multiplexer,...)
- **Konzepte für den sicheren Dienst trotz Störungen in den Komponenten:**
 - Redundanz innerhalb einzelner Netzkomponenten
 - Netzweite Redundanz
 - Fehlertolerantes Design im Knoten
 - Fehlertolerantes Netz-Design
 - Protokolle zur Automatischen Behebung von Störungen / Ausfällen
 - Vorsorgliche Wartung, Vorhersage von Ausfällen

- **Grundlagen des Internet wiederholt**
- **Paketvermittlung und Verhalten von Paketverkehr**
- **Schichtenmodell von TCP/IP wiederholt**
- **Flusskontrolle und Überlastkontrolle bei TCP wiederholt**
- **Ein erster Blick auf Sicherheitsfragen im Internet**
- **Geschichte des Internets**