Systeme II

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Christian Schindelhauer

Sommersemester 2007

10. Vorlesungswoche

25.06.-29.06.2007

schindel@informatik.uni-freiburg.de

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Die Transportschicht

Transportschicht (transport layer)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **TCP (transmission control protocol)**
  - Erzeugt zuverlässigen Datenfluß zwischen zwei Rechnern
  - Unterteilt Datenströme aus Anwendungsschicht in Pakete
  - Gegenseite schickt Empfangsbestätigungen (Acknowledgments)
- **UDP (user datagram protocol)**
  - Einfacher unzuverlässiger Dienst zum Versand von einzelnen Päckchen
  - Wandelt Eingabe in ein Datagramm um
  - Anwendungsschicht bestimmt Paketgröße
- **Versand durch Netzwerkschicht**
- **Kein Routing: End-to-End-Protokolle**

TCP (I)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **TCP ist ein verbindungsorientierter, zuverlässiger Dienst für bidirektionale Byteströme**

- **TCP ist verbindungsorientiert**
  - Zwei Parteien identifiziert durch Socket: IP-Adresse und Port (TCP-Verbindung eindeutig identifiziert durch Socketpaar)
  - Kein Broadcast oder Multicast
  - Verbindungsaufbau und Ende notwendig
  - Solange Verbindung nicht (ordentlich) beendet, ist Verbindung noch aktiv

TCP (II)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **TCP ist ein verbindungsorientierter, zuverlässiger Dienst für bidirektionale Byteströme**

- **TCP ist zuverlässig**
  - Jedes Datenpaket wird bestätigt (acknowledgment)
  - Erneutes Senden von unbestätigten Datenpakete
  - Checksum für TCP-Header und Daten
  - TCP nummeriert Pakete und sortiert beim Empfänger
  - Löscht duplizierte Pakete

TCP (III)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

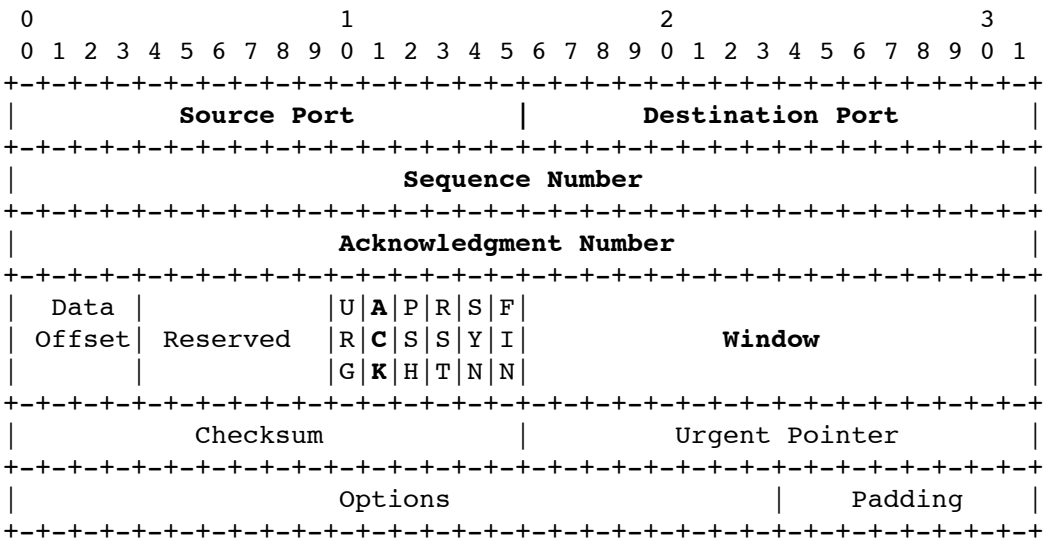
- **TCP ist ein verbindungsorientierter, zuverlässiger Dienst für bidirektionale Byteströme**

- **TCP ist ein Dienst für bidirektionale Byteströme**
  - Daten sind zwei gegenläufige Folgen aus einzelnen Bytes (=8 Bits)
  - Inhalt wird nicht interpretiert
  - Zeitverhalten der Datenfolgen kann verändert werden
  - Versucht zeitnahe Auslieferung jedes einzelnen Datenbytes
  - Versucht Übertragungsmedium effizient zu nutzen
    - = wenig Pakete

TCP-Header

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Sequenznummer**
  - Nummer des ersten Bytes im Segment
  - Jedes Datenbyte ist nummeriert modulo 2^{32}
- **Bestätigungsnummer**
  - Aktiviert durch ACK-Flag
  - Nummer des nächsten noch nicht bearbeiteten Datenbytes
    - = letzte Sequenznummer + letzte Datenmenge
- **Sonstiges:**
  - Port-Adressen
    - Für parallele TCP-Verbindungen
    - Ziel-Port-Nr.
    - Absender-Port
  - Headerlänge
    - data offset
  - Prüfsumme
    - Für Header und Daten



TCP-Verbindungsaufbau

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

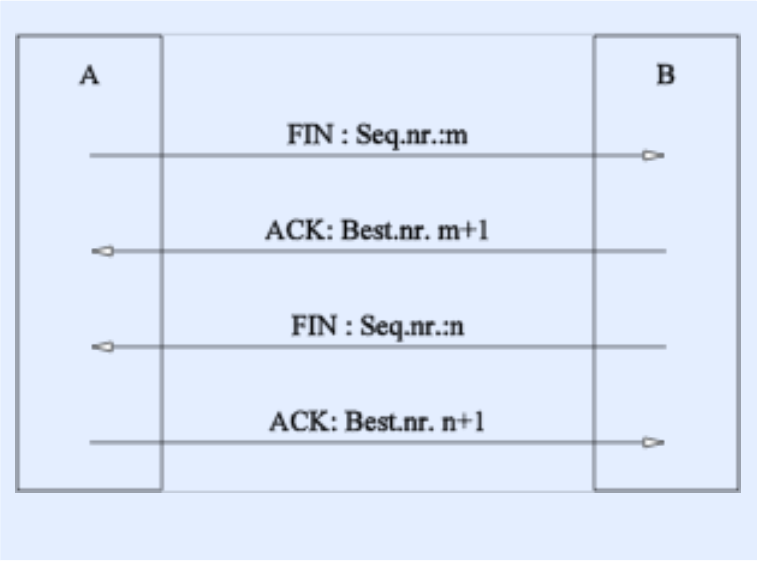
- **In der Regel Client-Server-Verbindungen**
  - Dann Aufbau mit drei TCP-Pakete (=Segmente)
  - Mit ersten SYN-Segment auch Übermittlung der MSS (maximum segment size)

TCP-Verbindungssende

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Half-Close**
  - –Sender kündigt Ende mit FIN-Segment an und wartet auf Bestätigung
  - –In Gegenrichtung kann weitergesendet werden

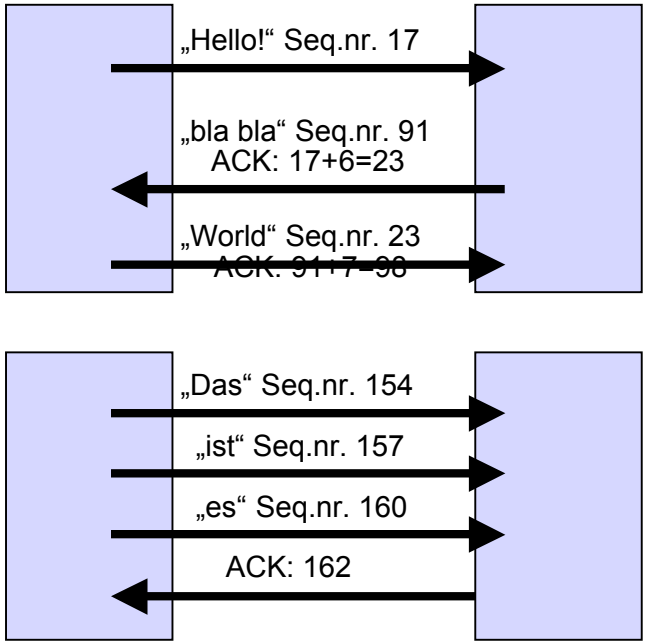
- **2 Half-Close beenden TCP-Verbindung**



Bestätigungen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

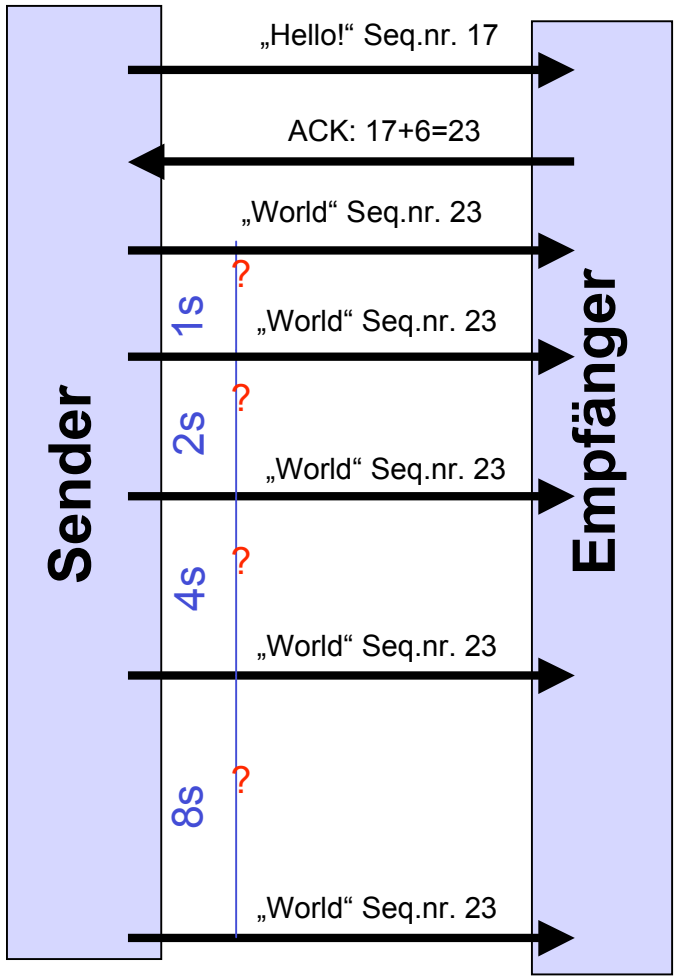
- **Huckepack-Technik**
  - Bestätigungen „reiten“ auf den Datenpaket der Gegenrichtung
- **Eine Bestätigungssegment kann viele Segmente bestätigen**
  - Liegen keine Daten an, werden Acks verzögert



Exponentielles Zurückweichen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Retransmission Timout (RTO)**
  - regelt Zeitraum zwischen Senden von Datenduplikaten, falls Bestätigung ausbleibt
- **Wann wird ein TCP-Paket nicht bestätigt?**
  - Wenn die Bestätigung wesentlich länger benötigt, als die durchschnittliche Umlaufzeit (RTT/round trip time)
    - 1. Problem: Messung der RTT
    - 2. Problem: Bestätigung kommt, nur spät
  - Sender
    - Wartet Zeitraum gemäß RTO
    - Sendet Paket nochmal und setzt
    - RTO ← 2 RTO (bis RTO = 64 Sek.)
- **Neuberechnung von RTO, wenn Pakete bestätigt werden**



TCP - Algorithmus von Nagle

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Wie kann man sicherstellen,**
  - dass kleine Pakete zeitnah ausgeliefert werden
  - und bei vielen Daten große Pakete bevorzugt werden?

- **Algorithmus von Nagle:**
  - Kleine Pakete werden nicht versendet, solange Bestätigungen noch ausstehen.
    - Paket ist klein, wenn Datenlänge < MSS
  - Trifft die Bestätigung des zuvor gesendeten Pakets ein, so wird das nächste verschickt.
- **Beispiel:**
  - Telnet versus ftp
- **Eigenschaften**
  - Selbst-taktend: Schnelle Verbindung = viele kleine Pakete

Schätzung der Umlaufzeit (RTT/Round Trip Time)

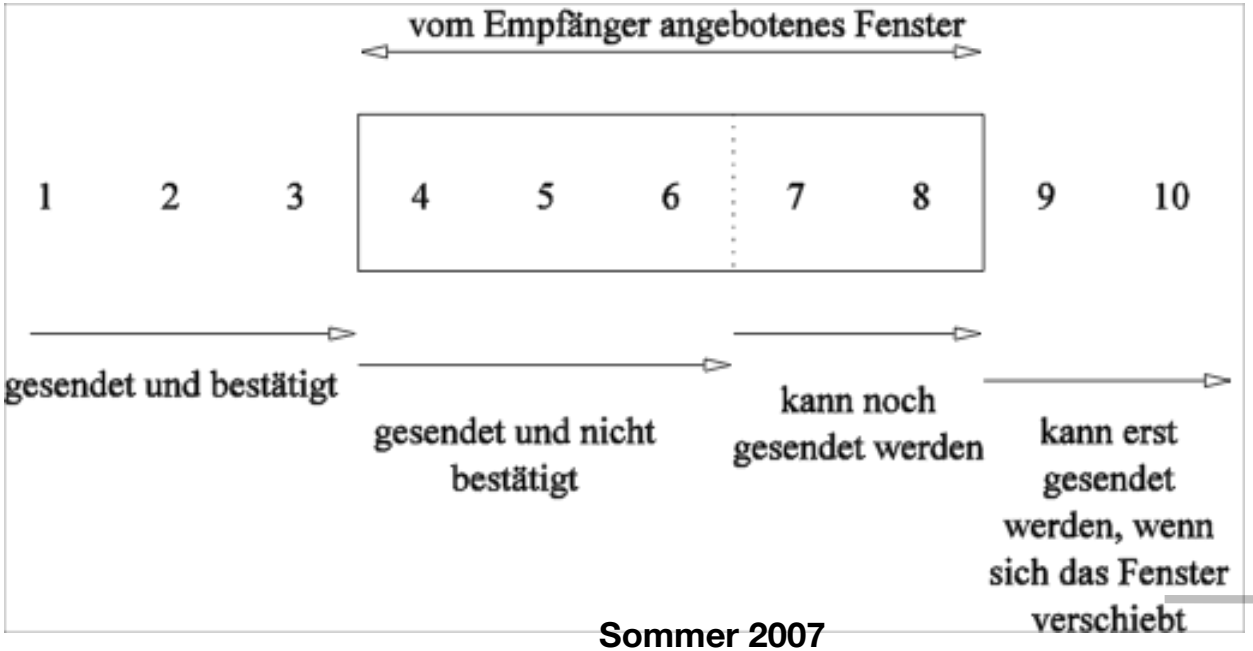
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **TCP-Paket gilt als nicht bestätigt, wenn Bestätigung „wesentlich“ länger dauert als RTO**
  - RTT nicht on-line berechenbar (nur rückblickend)
  - RTT schwankt stark
- **Daher: Retransmission Timeout Value aus großzügiger Schätzung:**
  - RFC 793: (M := letzte gemessene RTT)
    - $R \leftarrow \alpha R + (1- \alpha) M$, wobei $\alpha = 0{,}9$
    - $RTO \leftarrow \beta R$, wobei $\beta = 2$
  - Jacobson 88: Schätzung nicht robust genug, daher
    - $A \leftarrow A + g (M - A)$, wobei $g = 1/8$
    - $D \leftarrow D + h (|M - A| - D)$, wobei $h = 1/4$
    - $RTO \leftarrow A + 4D$
- **Aktualisierung nicht bei mehrfach versandten Pakete**

Gleitende Fenster (sliding windows)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

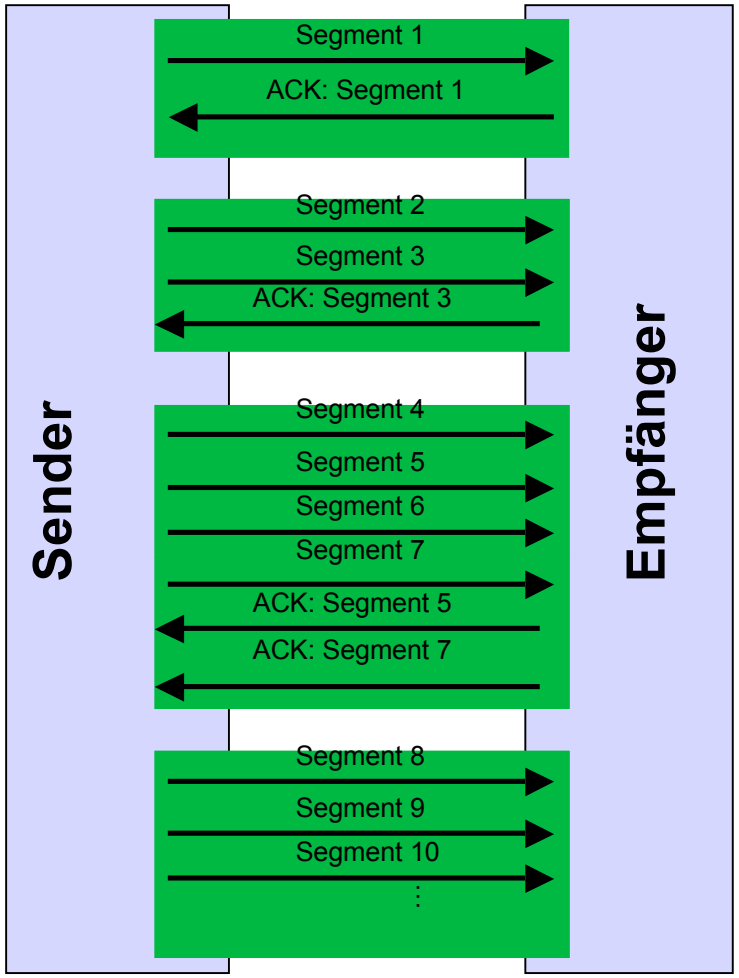
- **Datenratenanpassung durch Fenster**
  - Empfänger bestimmt Fenstergröße (wnd) im TCP-Header der ACK-Segmente
  - Ist Empfangspuffer des Empfängers voll, sendet er wnd=0
  - Andernfalls sendet Empfänger wnd>0
- **Sender beachtet:**
  - Anzahl unbestätigter gesender Daten ≤ Fenstergröße



Slow Start Congestion Fenster

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Sender darf vom Empfänger angebotene Fenstergröße nicht von Anfang wahrnehmen**
- **2. Fenster: Congestion-Fenster (cwnd/Congestion window)**
  - Von Sender gewählt (FSK)
  - Sendefenster: min {wnd,cwnd}
  - S: Segmentgröße
  - Am Anfang:
    - cwnd ← S
  - Für jede empfangene Bestätigung:
    - cwnd ← cwnd + S
  - Solange bis einmal Bestätigung ausbleibt
- **„Slow Start“ = Exponentielles Wachstum**



Stauvermeidung in TCP Tahoe

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

x: Anzahl Pakete pro RTT

- **Jacobson 88:**
  - Parameter: cwnd und Slow-Start-Schwellwert (ssthresh=slow start threshold)
  - S = Datensegmentgröße = maximale Segmentgröße
- **Verbindungsaufbau:**
  cwnd ← S **ssthresh ← 65535**

  x ← 1
  y ← max
- **Bei Paketverlust, d.h. Bestätigungsdauer > RTO,**
  - multiplicatively decreasing
    cwnd ← S ssthresh ← $\max\left\{2{\times}S, \frac{\min\{cwnd, wnd\}}{2}\right\}$

  x ← 1
  y ← x/2
- **Werden Segmente bestätigt und cwnd ≤ ssthresh, dann**
  - slow start:
    cwnd ← cwnd + S

  x ← 2·x, bis x = y
- **Werden Segmente bestätigt und cwnd > ssthresh, dann additively increasing**
  cwnd ← cwnd + S $\frac{S}{cwnd}$

  x ← x +1

TCP Tahoe

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

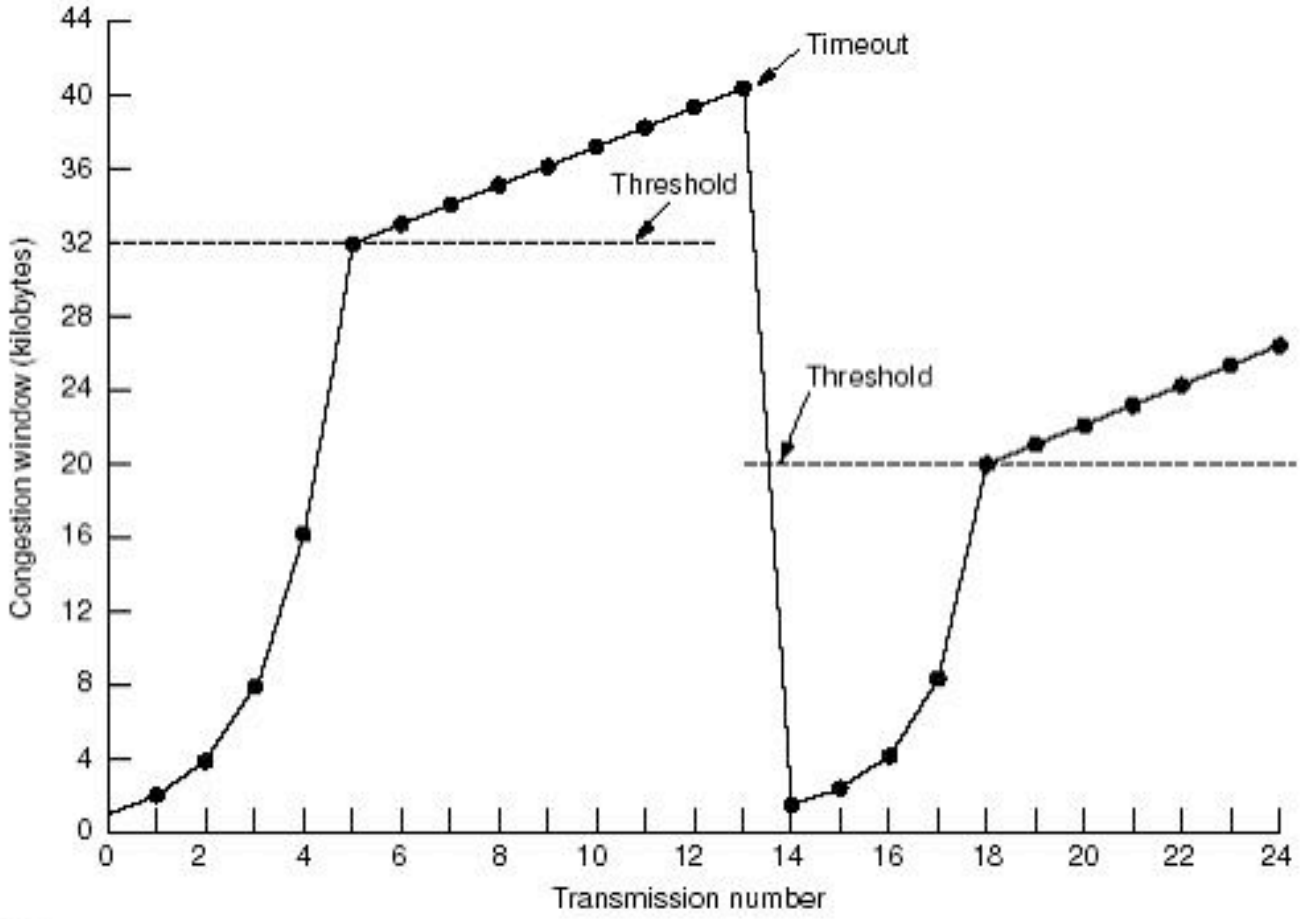


Fig3

pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

Fast Retransmit und Fast Recovery

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **TCP Tahoe [Jacobson 1988]:**
  - Geht nur ein Paket verloren, dann
    - Wiederversand Paket + Restfenster
    - Und gleichzeitig Slow Start
  - Fast retransmit
    - Nach drei Bestätigungen desselben Pakets (triple duplicate ACK),
    - sende Paket nochmal, starte mit Slow Start
- **TCP Reno [Stevens 1994]**
  - Nach Fast retransmit:
    - ssthresh ← min(wnd,cwnd)/2
    - cwnd ← ssthresh + 3 S
  - Fast recovery nach Fast retransmit
    - Erhöhe Paketrate mit jeder weiteren Bestätigung
    - cwnd ← cwnd + S
  - Congestion avoidance: Trifft Bestätigung von P+x ein:
    - cwnd ← ssthresh

y ← x/2

x ← y + 3

Stauvermeidungsprinzip: AIMD

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Kombination von TCP und Fast Recovery verhält sich im wesentlichen wie folgt:**

- **Verbindungsaufbau:**

$$x \leftarrow 1$$

- **Bei Paketverlust, MD:multiplicative decreasing**

$$x \leftarrow x/2$$

- **Werden Segmente bestätigt, AD: additive increasing**

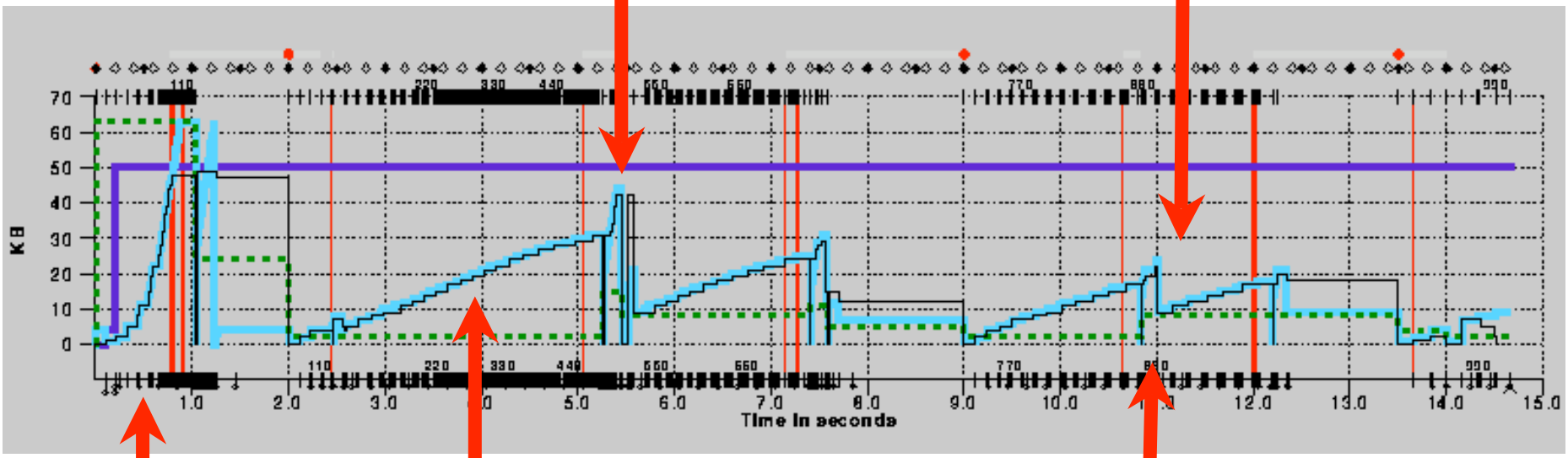
$$x \leftarrow x + 1$$

Beispiel: TCP Reno in Aktion

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Fast Retransmit

Fast Recovery



Additively Increase

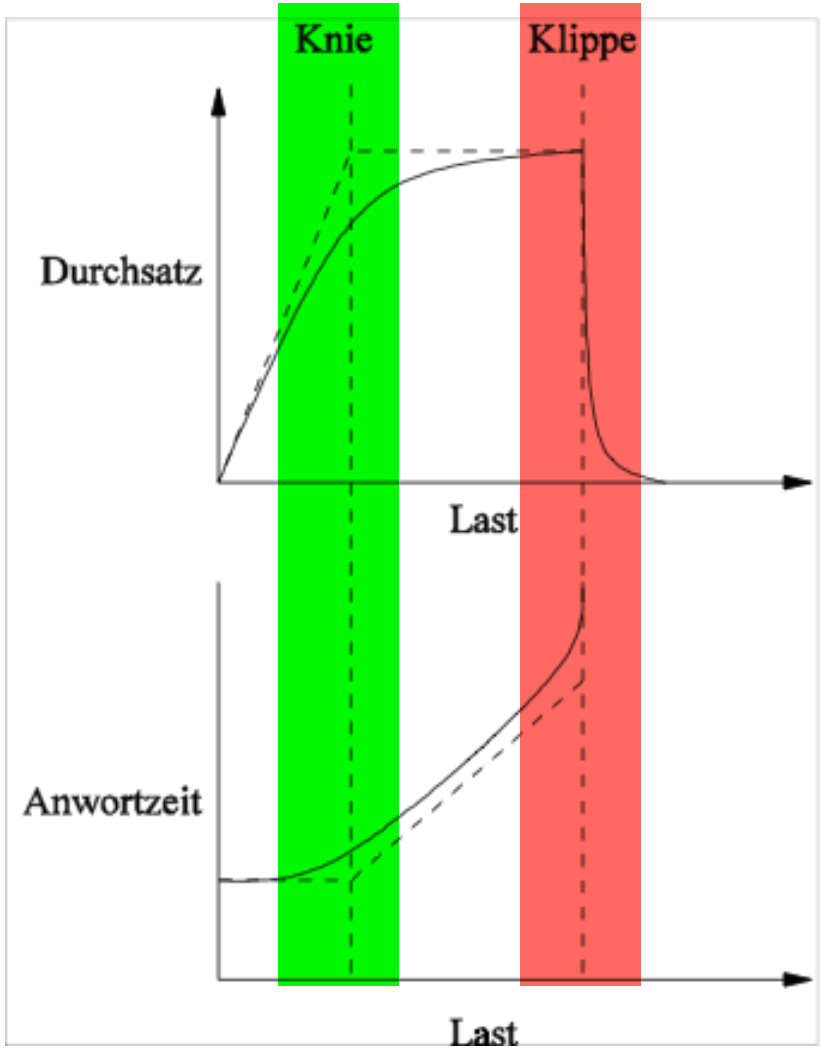
Slow Start

Multiplicatively Decrease

Durchsatz und Antwortzeit

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Klippe:**
  - Hohe Last
  - Geringer Durchsatz
  - Praktisch alle Daten gehen verloren
- **Knie:**
  - Hohe Last
  - Hoher Durchsatz
  - Einzelne Daten gehen verloren



Ein einfaches Datenratenmodell

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **n Teilnehmer, Rundenmodell**
  - Teilnehmer i hat Datenrate xi(t)
  - Anfangsdatenrate $x_1(0), \ldots, x_n(0)$ gegeben

- **Feedback nach Runde t:**
  - y(t) = 0, falls $\sum_{i=1}^{n} x_i(t) \leq K$
  - y(t) = 1, falls $\sum_{i=1}^{n} x_i(t) > K$
  - wobei K ist Knielast
- **Jeder Teilnehmer aktualisiert in Runde t+1:**
  - $x_i(t+1) = f(x_i(t),y(t))$
  - Increase-Strategie $f_0(x) = f(x,0)$
  - Decrease-Strategie $f_1(x) = f(x,1)$

- **Wir betrachten lineare Funktionen:**

$$f_0(x) = a_I + b_I x \qquad \text{und} \qquad f_1(x) = a_D + b_D x \ .$$

Lineare Datenratenanpassung

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Interessante Spezialfälle:**
  - AIAD: Additive Increase Additive Decrease

$$f_0(x) = a_I + x \quad \text{und} \quad f_1(x) = a_D + x \ ,$$

wobei $a_I > 0$ und $a_D < 0$.

  - MIMD: Multiplicative Increase/Multiplicative Decrease

$$f_0(x) = b_I x \quad \text{und} \quad f_1(x) = b_D x \ ,$$

wobei $b_I > 1$ und $b_D < 1$.

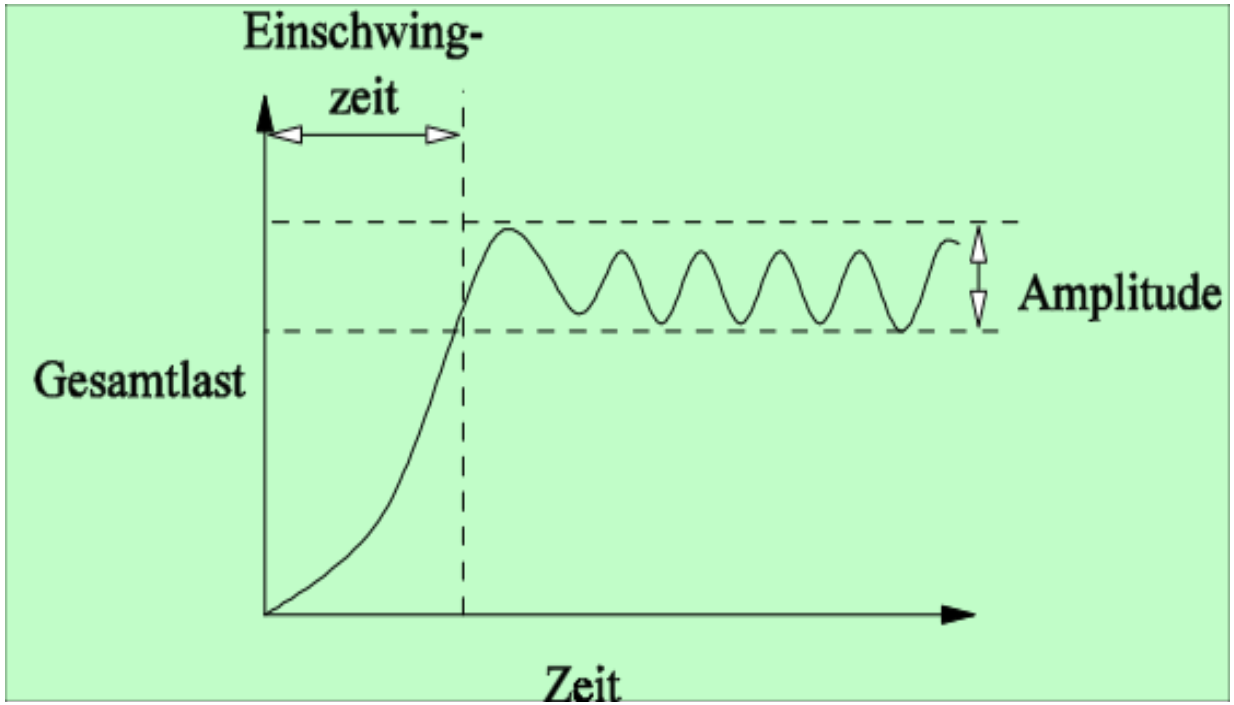
  - AIMD: Additive Increase Multiplicative Decrease

$$f_0(x) = a_I + x \quad \text{und} \quad f_1(x) = b_D x \ ,$$

wobei $a_I > 0$ und $b_D < 1$.

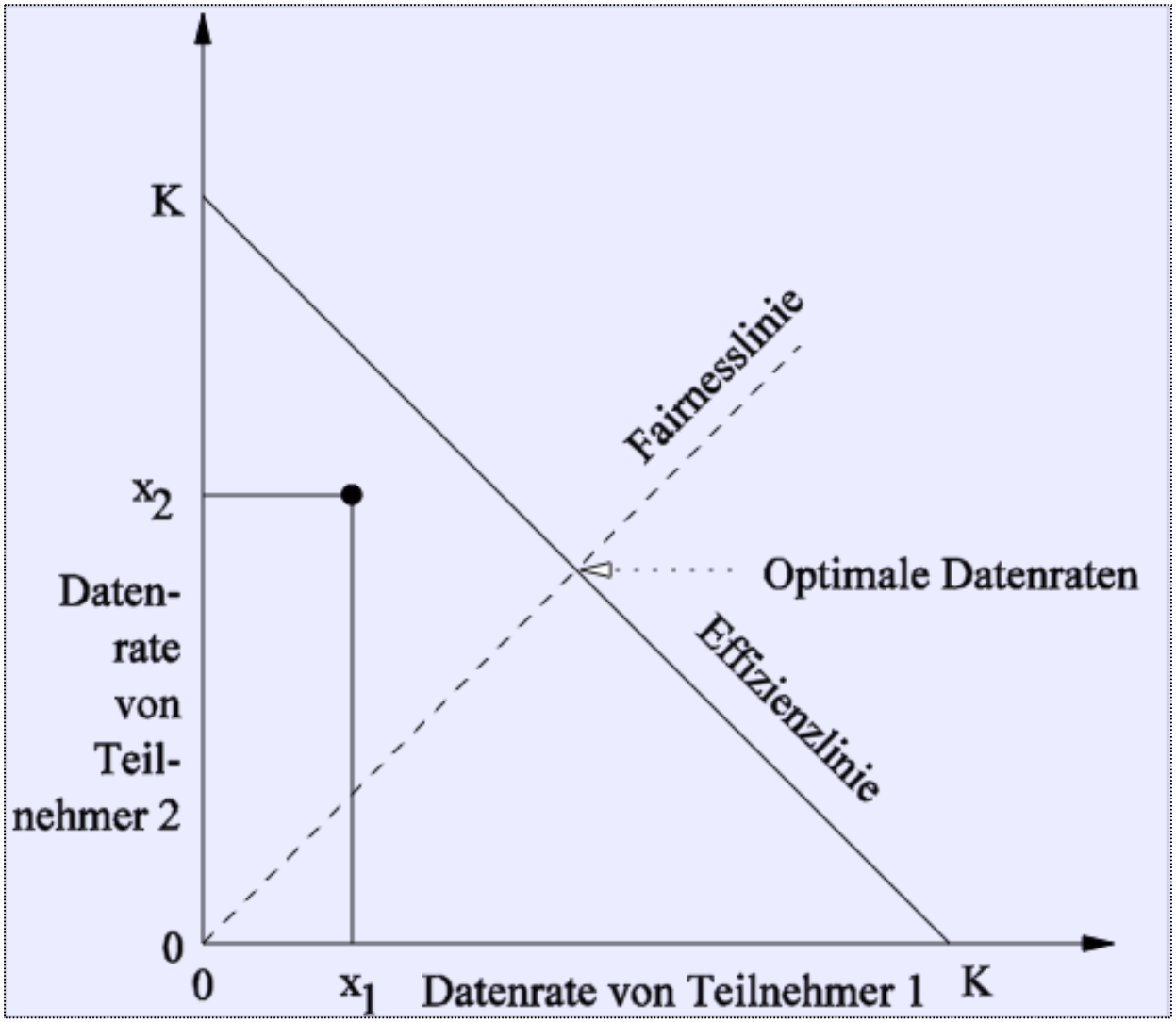
Konvergenz

- **Konvergenz unmöglich**
- **Bestenfalls Oszillation um Optimalwert**
  - Oszillationsamplitude A
  - Einschwingzeit T



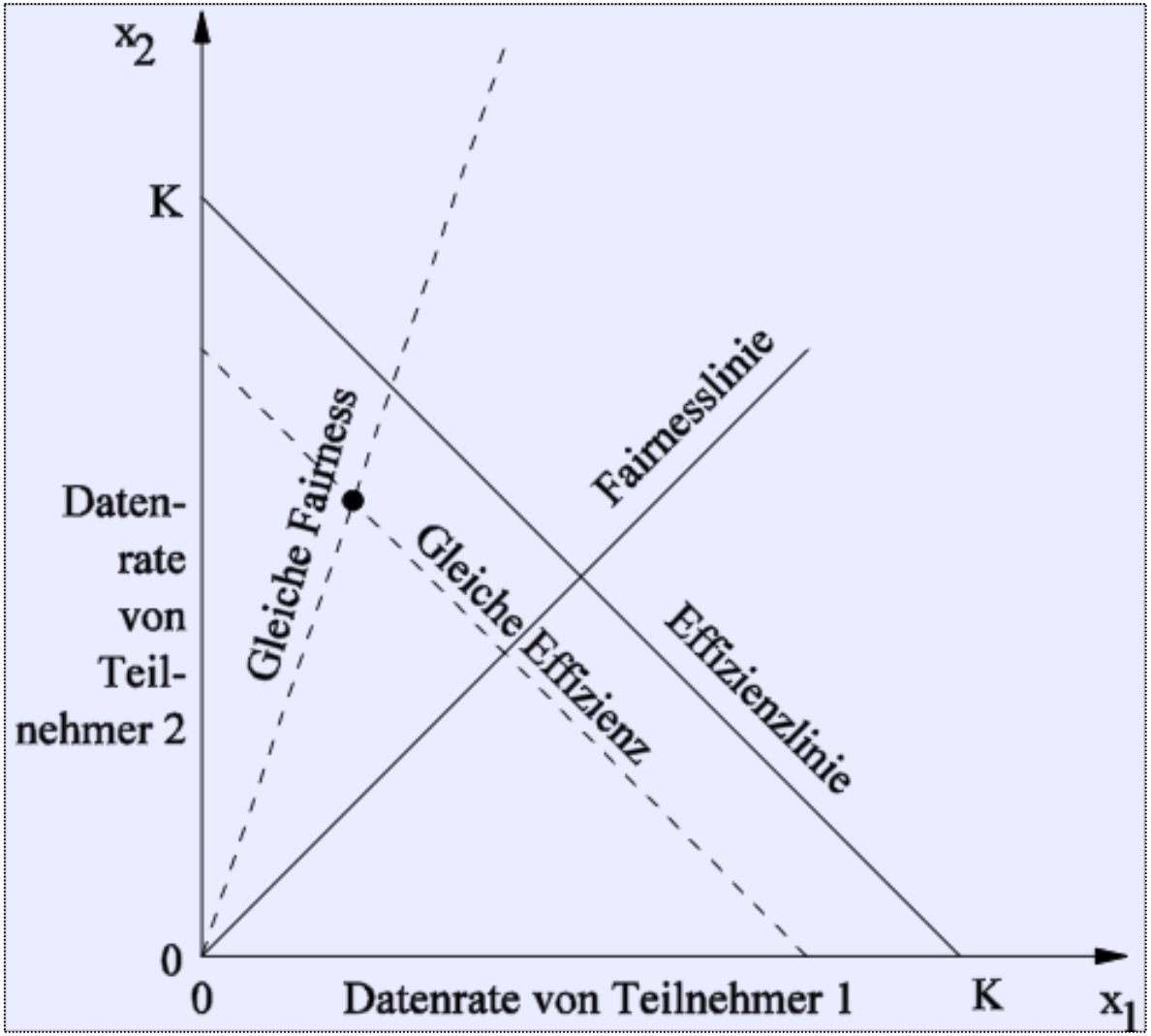
Vektordarstellung (I)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



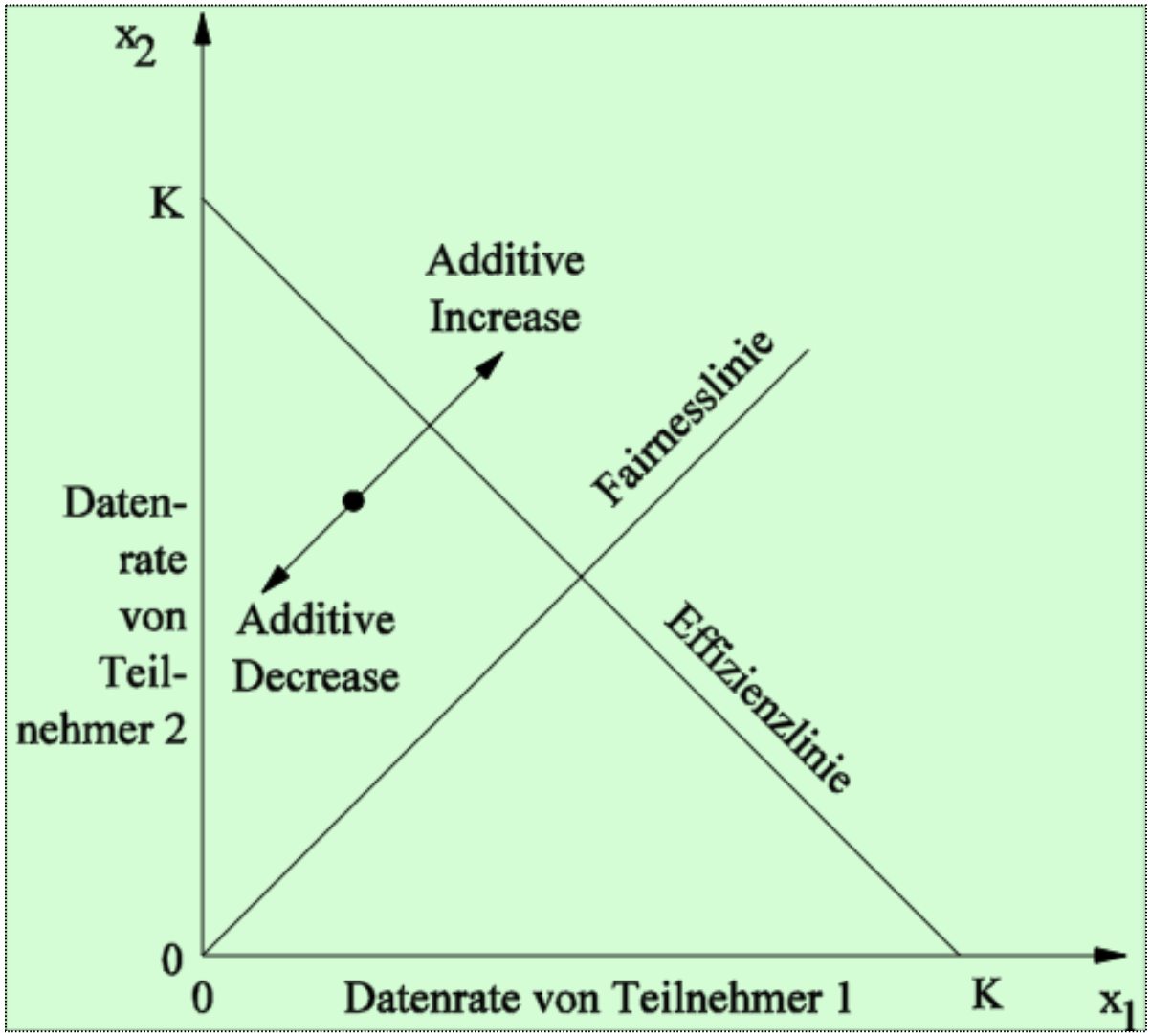
Vektordarstellung (II)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



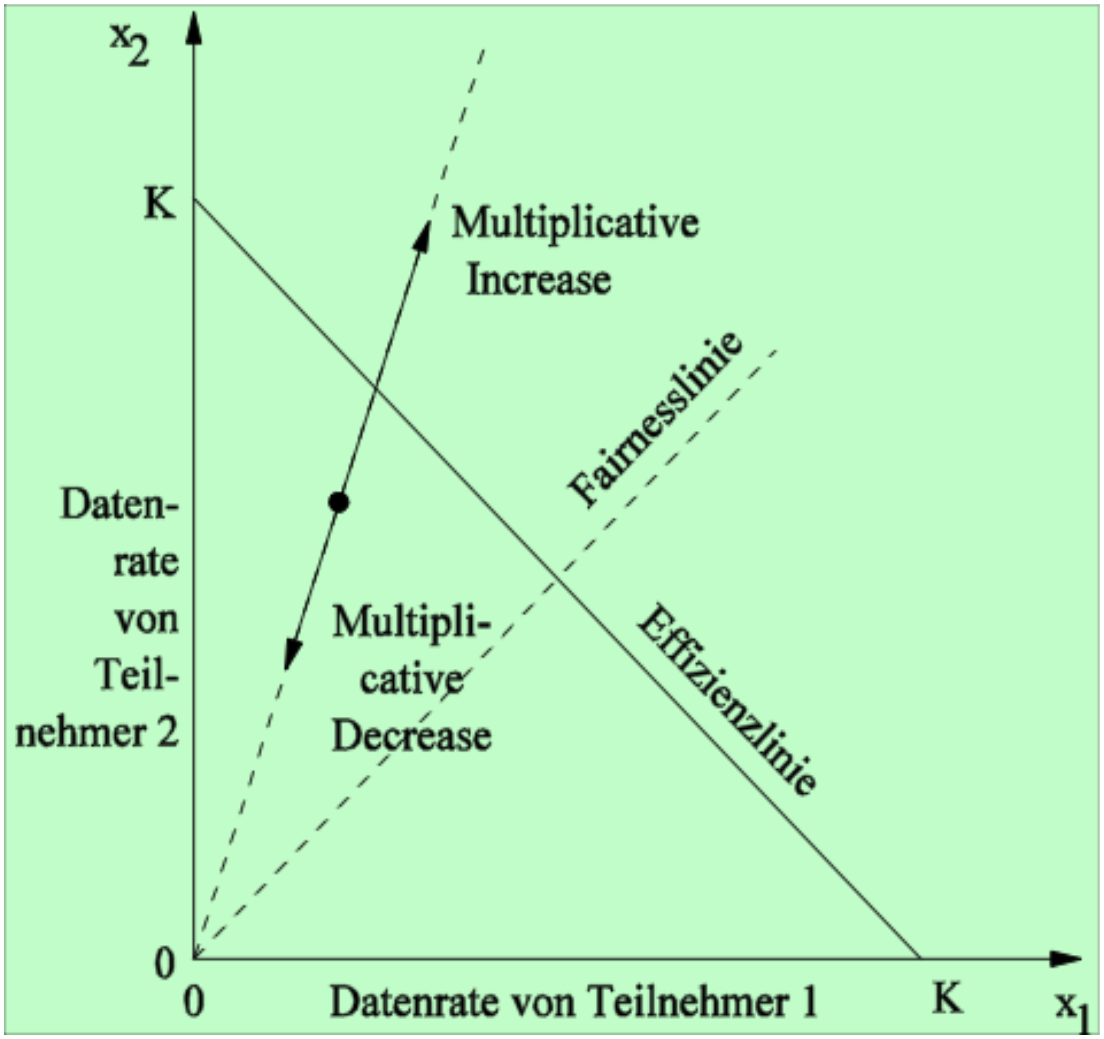
AIAD Additive Increase/ Additive Decrease

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



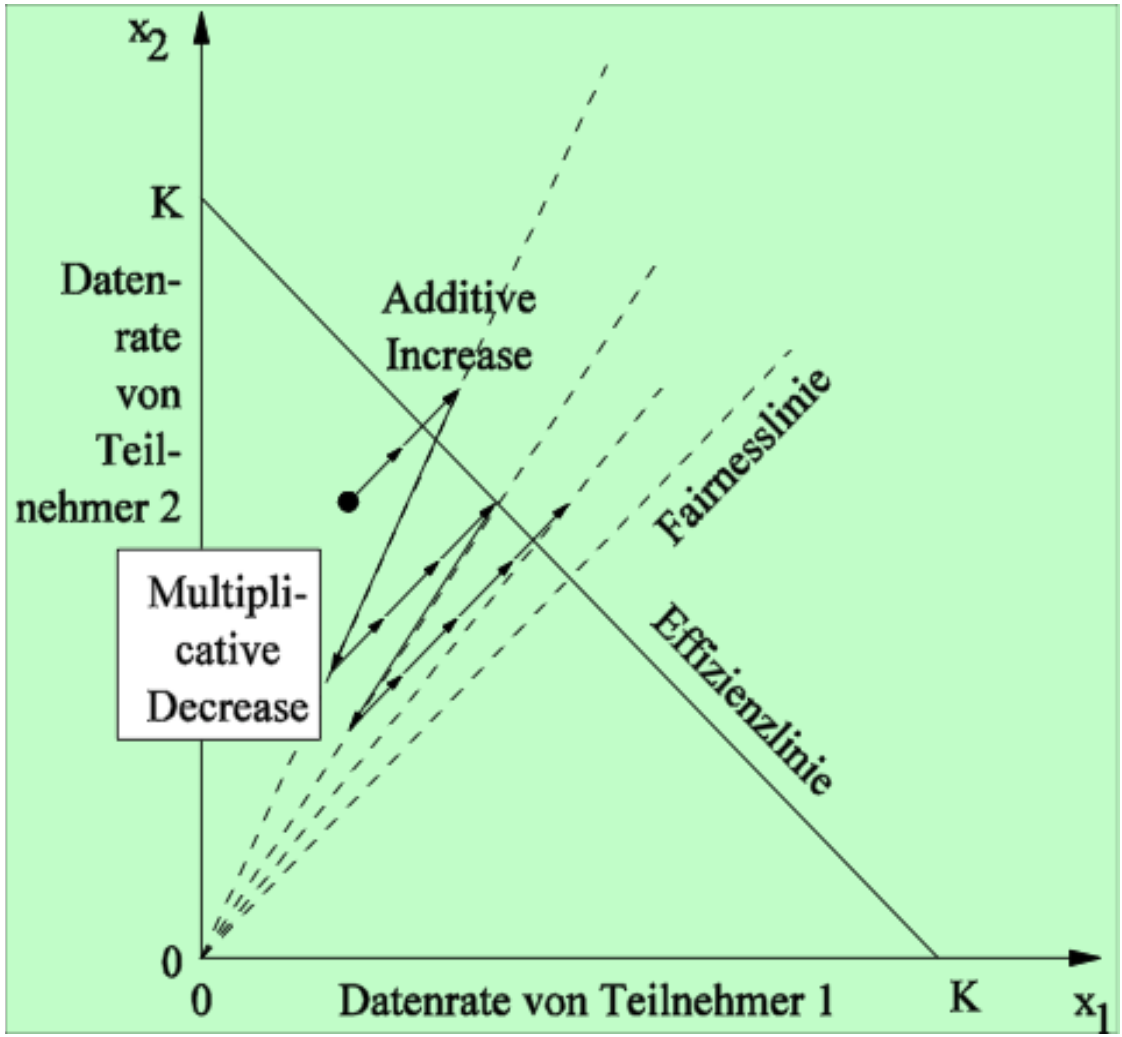
MIMD: Multiplicative Incr./ Multiplicative Decrease

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



AIMD: Additively Increase/ Multiplicatively Decrease

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



TCP Zusammenfassung

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **TCP erzeugt zuverlässigen Byte-Strom**
  - Fehlerkontrolle durch “GoBack-N”
- **Congestion control**
  - Fensterbasiert
  - AIMD, Slow start, *Congestion Threshold*
  - Flusskontrolle durch *Window*
  - Verbindungsaufbau
  - Algorithmus von Nagle

TCP fairness & TCP friendliness

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **TCP**
  - **reagiert dynamisch auf die zur Verfügung stehende Bandweite**
  - Faire Aufteilung der Bandweite
    - Im Idealfall: n TCP-Verbindungen erhalten einen Anteil von 1/n
- **Zusammenspiel mit anderen Protokollen**
  - Reaktion hängt von der Last anderer Transportprotokolle ab
    - z.B. UDP hat keine Congestion Control
  - Andere Protokolle können jeder Zeit eingesetzt werden
  - UDP und andere Protokoll können TCP Verbindungen unterdrücken
- **Schlussfolgerung**
  - Transport-Protokolle müssen TCP-kompatibel sein (TCP friendly)

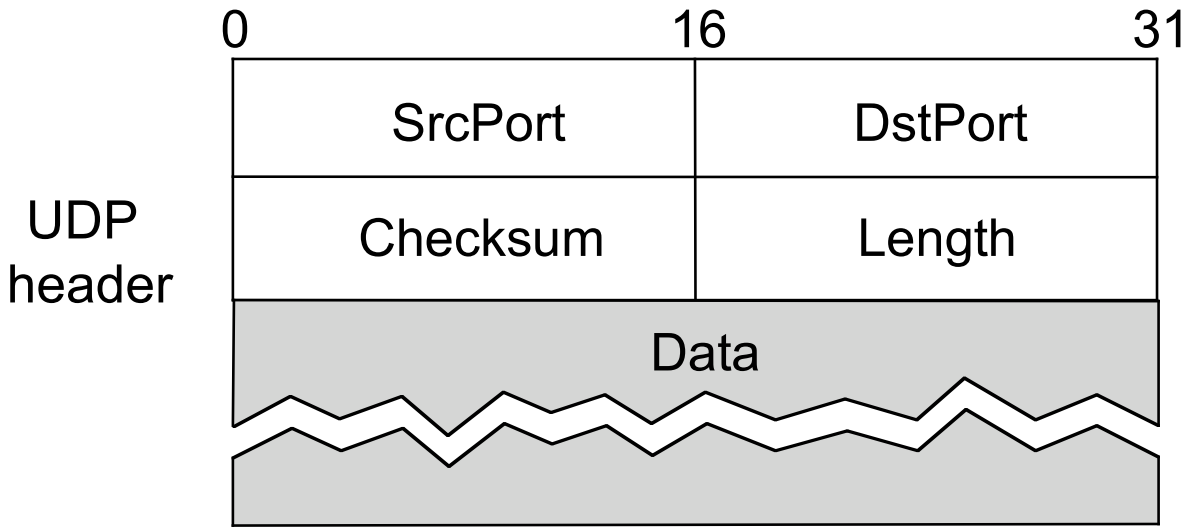
Dienste der Transport-Schicht

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Verbindungslos oder Verbindungsorientert**
  - Beachte: Sitzungsschicht im ISO/OSI-Protokoll
- **Verlässlich oder Unverlässlich**
  - Best effort oder Quality of Service
  - Fehlerkontrolle
- **Mit oder ohne Congestion Control**
- **Möglichkeit verschiedener Punkt-zu-Punktverbindungen**
  - Stichwort: Demultiplexen
- **Interationsmodelle**
  - Byte-Strom, Nachrichten, „Remote Procedure Call“

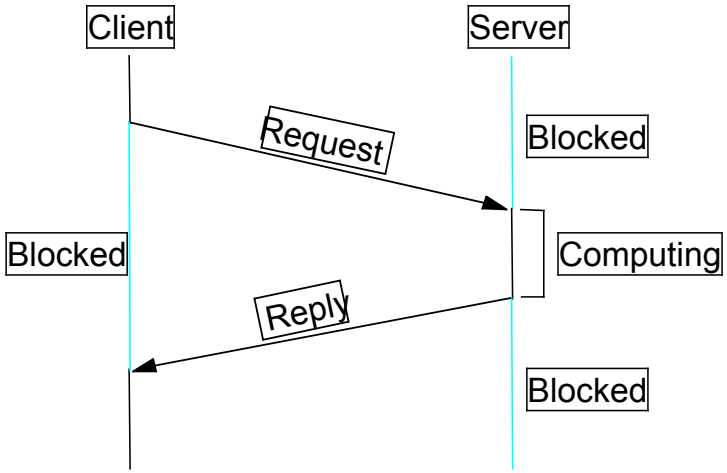
UDP

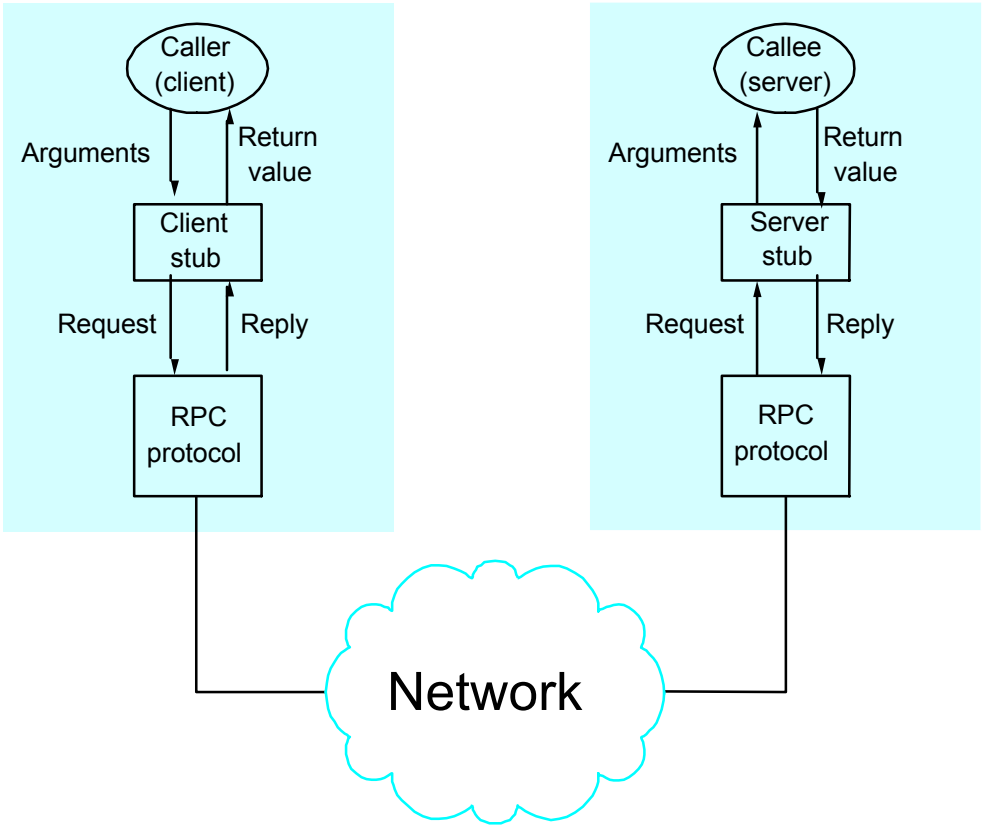
- **User Datagram Protocol (UDP)**
  - ist ein unzuverlässiges, verbindungsloses Transportprotokoll für Pakete
- **Hauptfunktion:**
  - Demultiplexing von Paketen aus der Vermittlungsschicht
- **Zusätzlich (optional):**
  - Checksum aus UDP Header + Daten



Remote procedure call – Struktur

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer





- **Für komplexere Interaktion:**
  - Aufruf einer Funktion auf einen anderen Rechner
- **Ziel: Transparentes Protokoll für den Aufrufer/Aufgerufenen**

Ende der 10. Vorlesungswoche

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Systeme II

Christian Schindelhauer
schindel@informatik.uni-freiburg.de