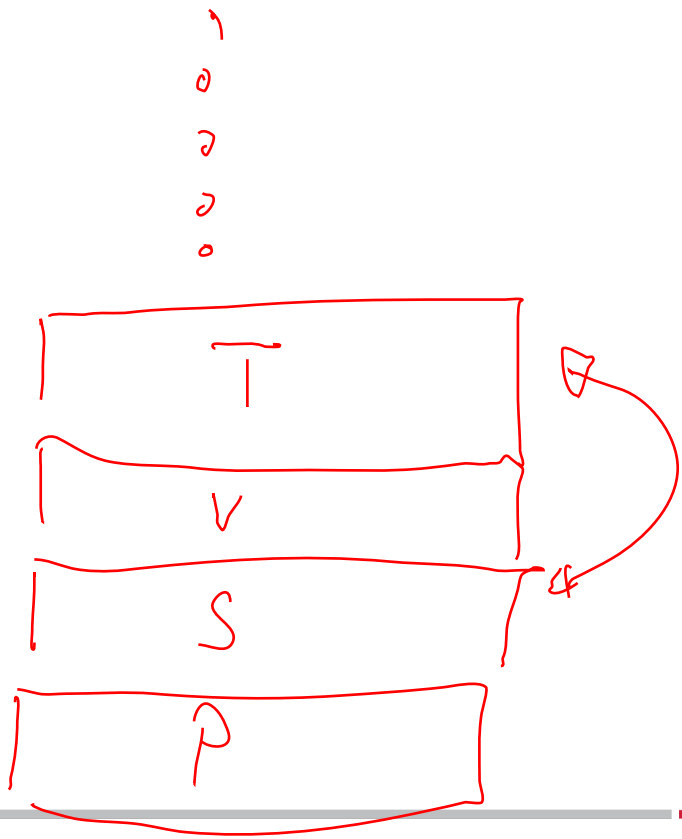
CoNe Freiburg

IIF INSTITUT FÜR INFORMATIK FREIBURG

Systeme II

5. Die Transportschicht

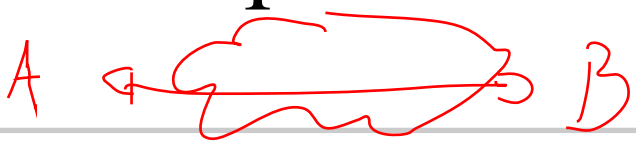
Christian Schindelhauer
Technische Fakultät
Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg
Version 16.06.2014



UNI FREIBURG

CoNe Freiburg

Dienste der Transportschicht

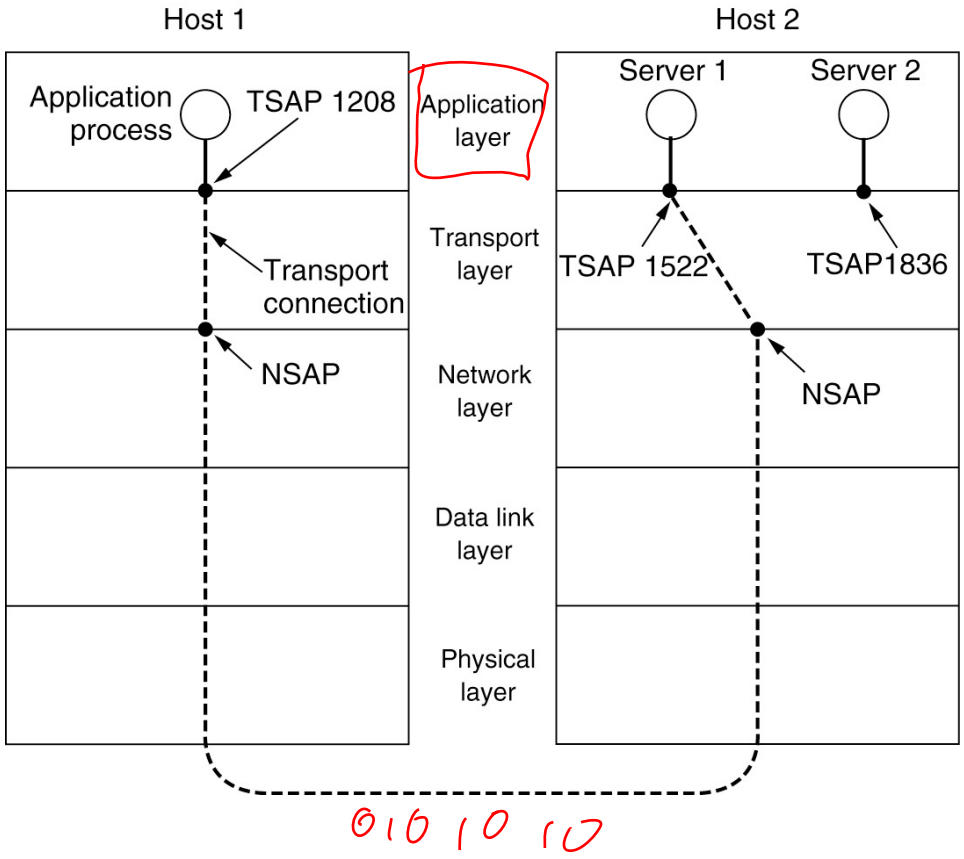


- Verbindungslos oder Verbindungsorientert
  - Beachte: Sitzungsschicht im ISO/OSI-Protokoll
- Zuverlässig oder unzuverlässig
  - Best effort oder Quality of Service
  - Fehlerkontrolle
- Mit oder ohne Congestion Control
- Möglichkeit verschiedener Punkt-zu-Punktverbindungen
  - Stichwort: Demultiplexen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

CoNe Freiburg

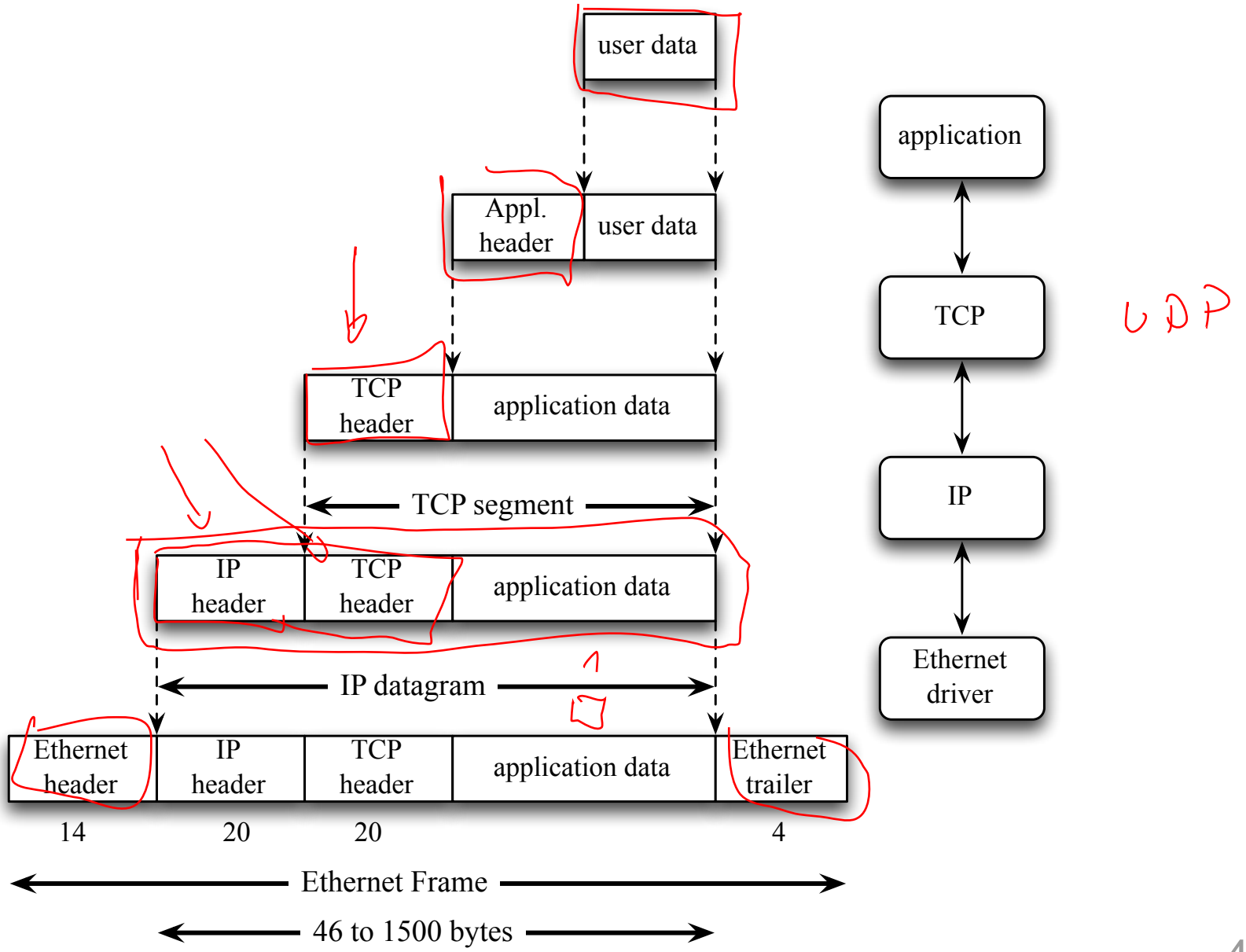
Multiplex in der Transportschicht

- Die Netzwerkschicht leitet Daten an die Transportschicht unkontrolliert weiter
- Die Transportschicht muss sie den verschiedenen Anwendungen zuordnen:
  - z.B. Web, Mail, FTP, ssh, ...
  - In TCP/UDP durch Port-Nummern
  - z.B. Port 80 für Web-Server



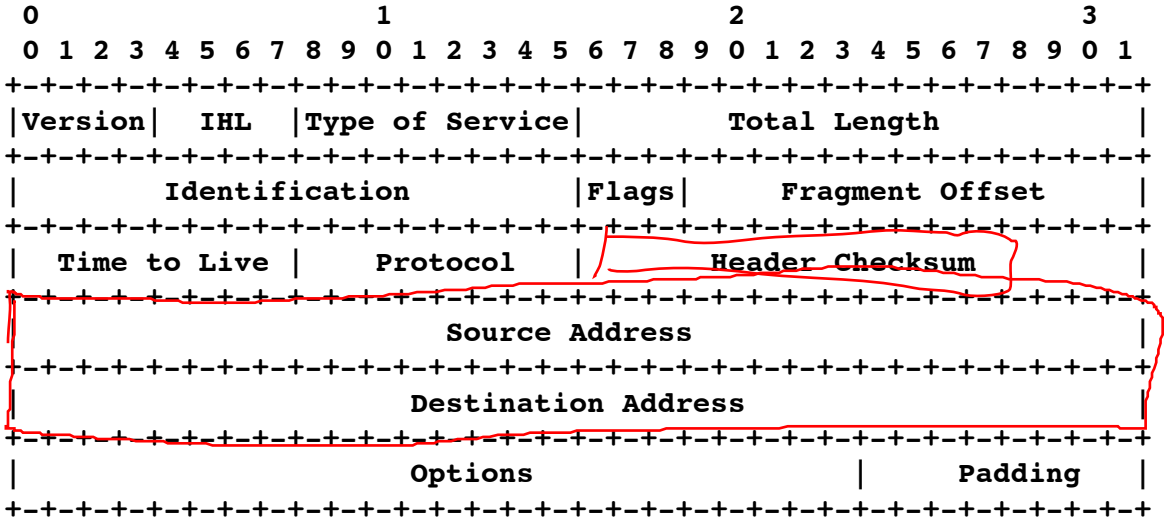
CoNe Freiburg

Datenkapselung



IP-Header (RFC 791)

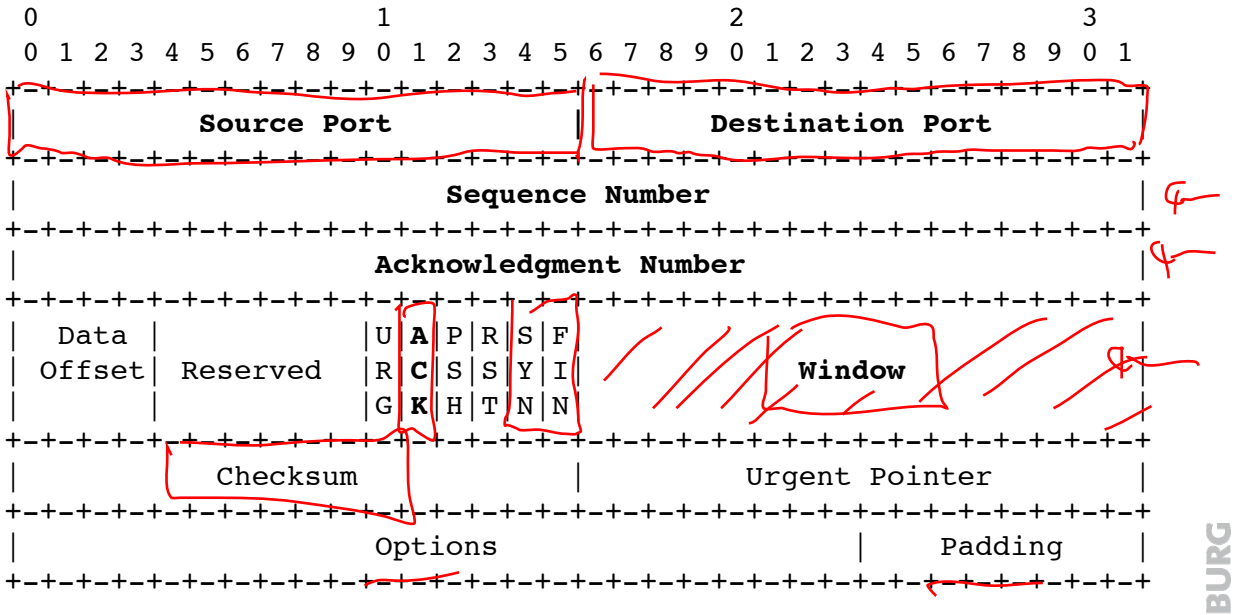
- Version: 4 = IPv4
- IHL: Headerlänge
  - in 32 Bit-Wörter (>5)
- Type of Service
  - Optimiere delay, throughput, reliability, monetary cost



- Checksum (nur für IP-Header)
- Source and destination IP-address
- Protocol, identifiziert passendes Protokoll
  - Z.B. TCP, UDP, ICMP, IGMP
- Time to Live:
  - maximale Anzahl Hops

TCP-Header

- 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:
- Port-Adressen
  - Für parallele TCP-Verbindungen
  - Ziel-Port-Nr.
  - Absender-Port
- Headerlänge
  - data offset
- Prüfsumme
  - Für Header und Daten



CoNe Freiburg

Transportschicht (transport layer)

- 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

CoNe Freiburg

TCP (I)

- 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

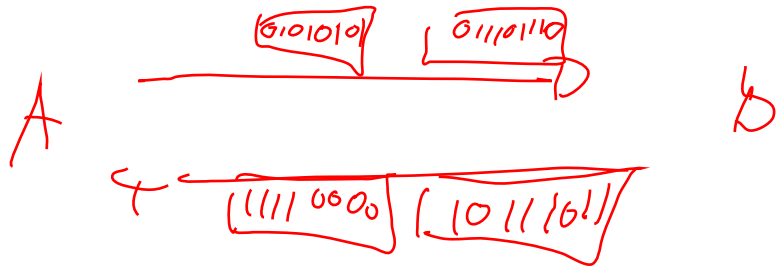
CoNe Freiburg

TCP (II)

- 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 ↪ nicht sicher!
  - TCP nummeriert Pakete und sortiert beim Empfänger
  - Löscht duplizierte Pakete

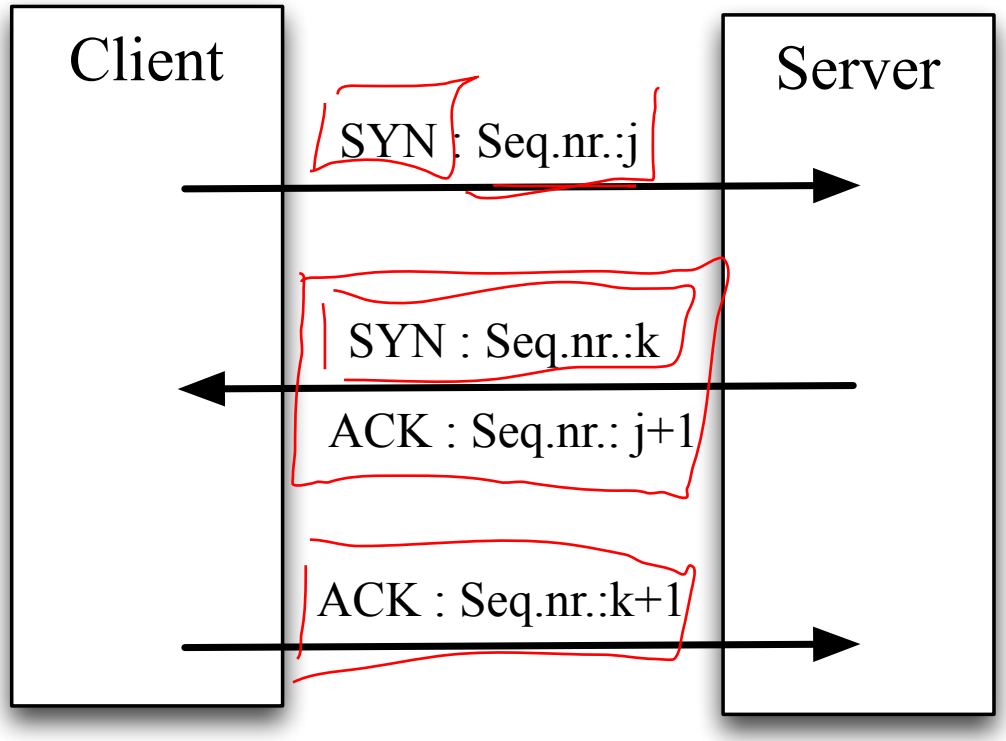
TCP (III)



- 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 → Nagle
  - Versucht Übertragungsmedium effizient zu nutzen
    - = wenig Pakete

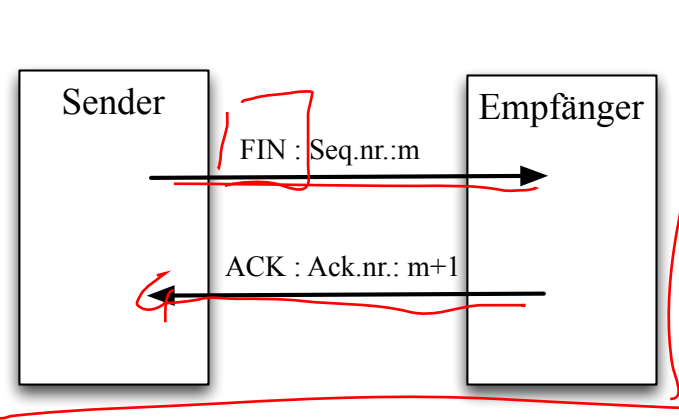
TCP-Verbindungsaufbau

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

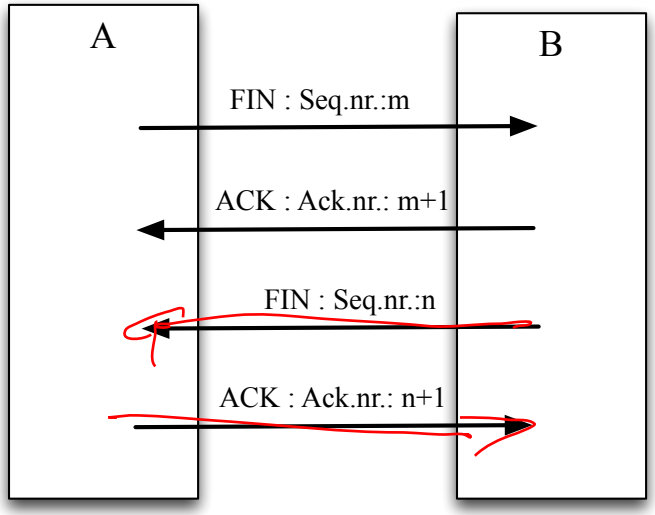


TCP-Verbindungssende

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



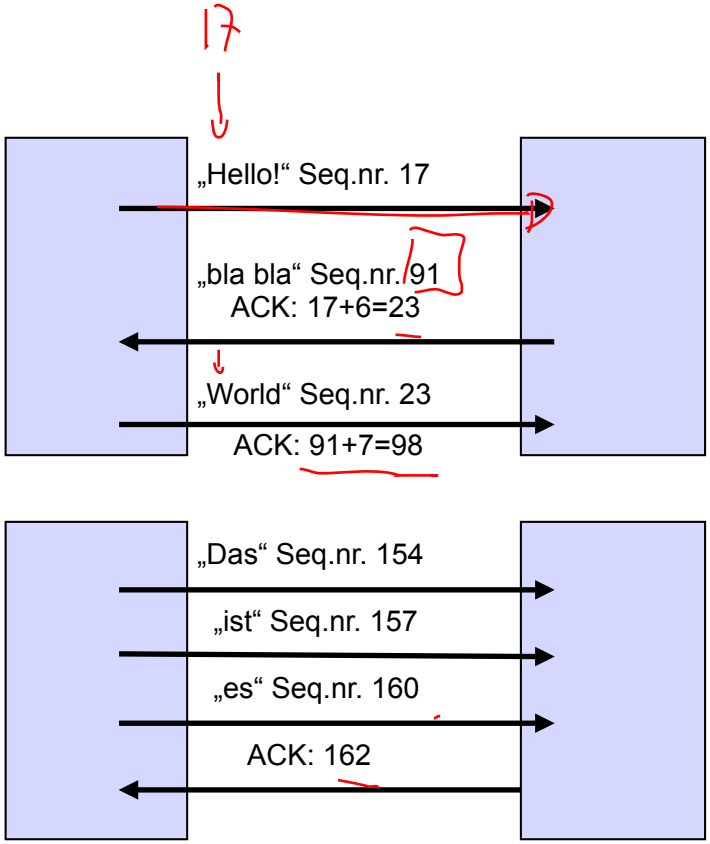
- 2 Half-Close beenden TCP-Verbindung



CoNe Freiburg

Bestätigungen

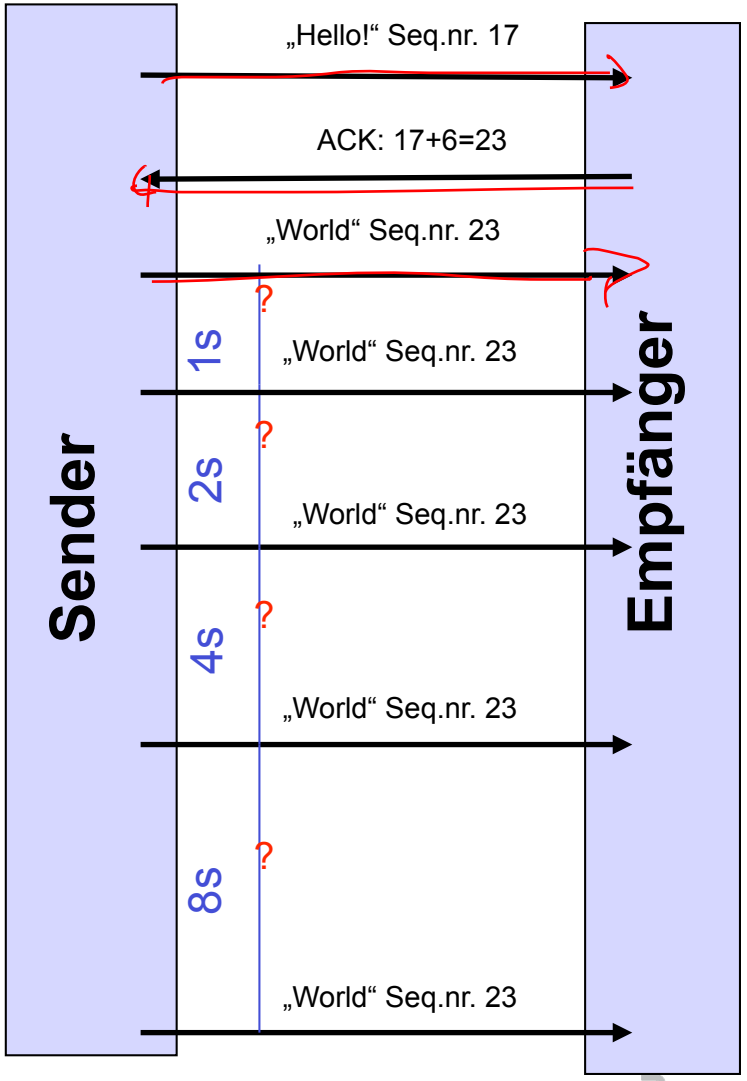
- Huckepack-Technik
  - Bestätigungen „reiten“ auf den Datenpaket der Gegenrichtung
- Eine Bestätigungssegment kann viele Segmente bestätigen
  - Liegen keine Daten an, werden Acks verzögert



UNI FREIBURG

Exponentielles Zurückweichen

- 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



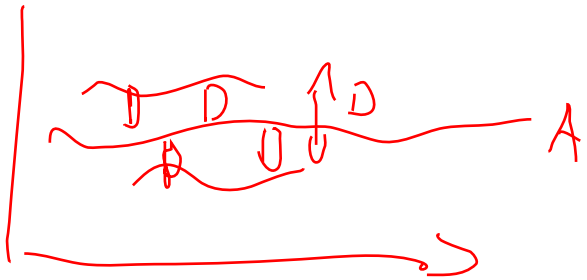
CoNe Freiburg

Schätzung der Umlaufzeit (RTT/Round Trip Time)

- 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 $h = 1/4$
    - $RTO \leftarrow A + 4D$

$R \leftarrow 0{,}9 \cdot R + 0{,}1 \cdot 1$



- Aktualisierung nicht bei mehrfach versandten Pakete

$$A' = A + g(M - A)$$

$$A = A(1-g) + g \cdot M$$

$$g = \frac{1}{\zeta}$$

$$P[X \geq k \cdot E(X)] \leq \frac{1}{k} \qquad \text{Markov}$$

$$P[X \geq 2E[X]] \leq \frac{1}{2}$$

~~Aaaa~~ !

Variance

$$Var(X) = \boxed{E\left[(X - E(X))^2\right]}$$

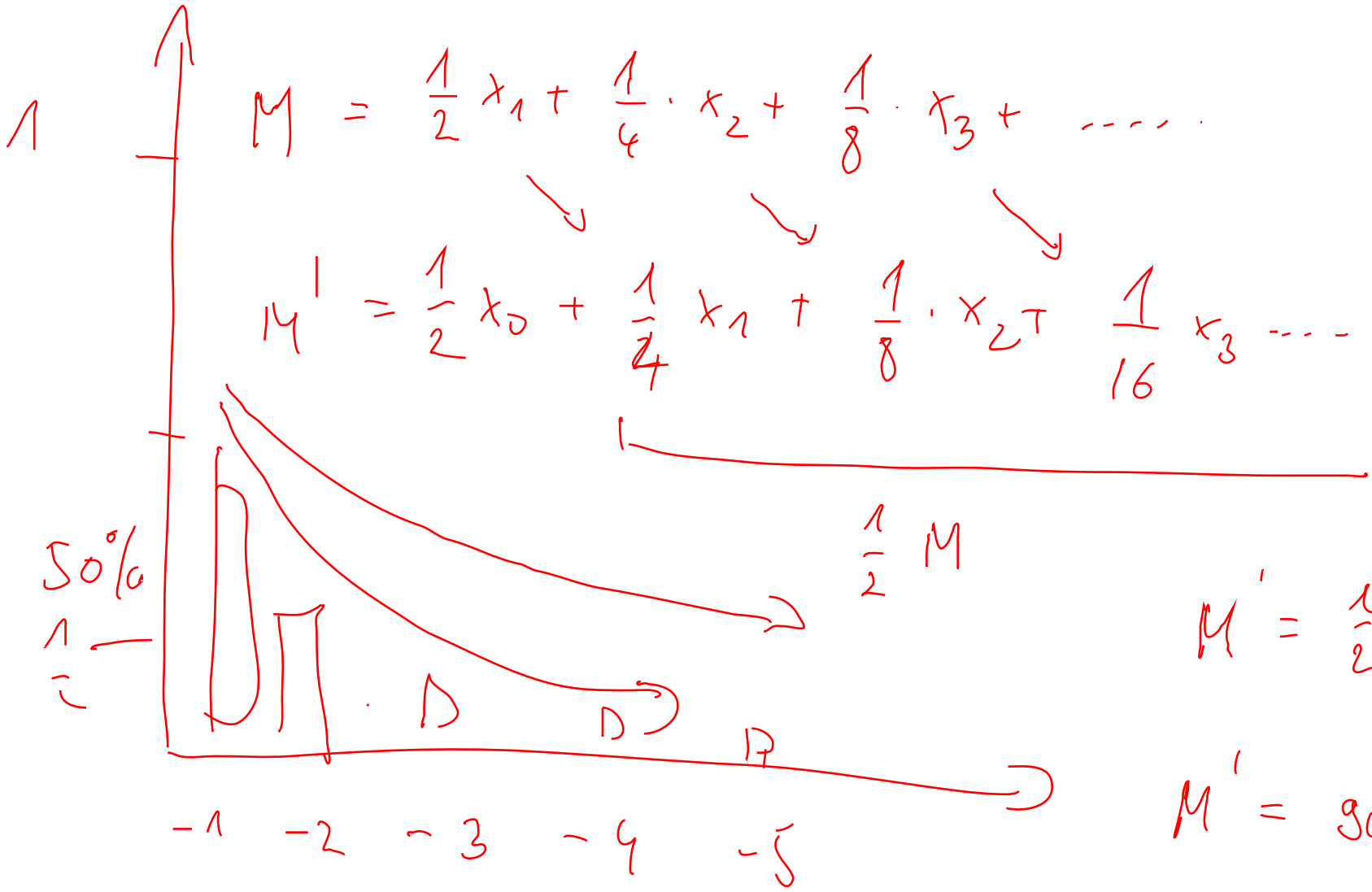
$$= E(X^2) - E(X)^2$$

$$E\left[|X - E(X)|\right] \Leftarrow$$

$x_1 \; x_2 \; x_3 \; x_4 \; \ldots\ldots$

$M = \frac{1}{2} x_1 + \frac{1}{4} \cdot x_2 + \frac{1}{8} \cdot x_3 + \ldots\ldots$

$M' = \frac{1}{2} x_0 + \frac{1}{4} x_1 + \frac{1}{8} \cdot x_2 + \frac{1}{16} x_3 \ldots\ldots$



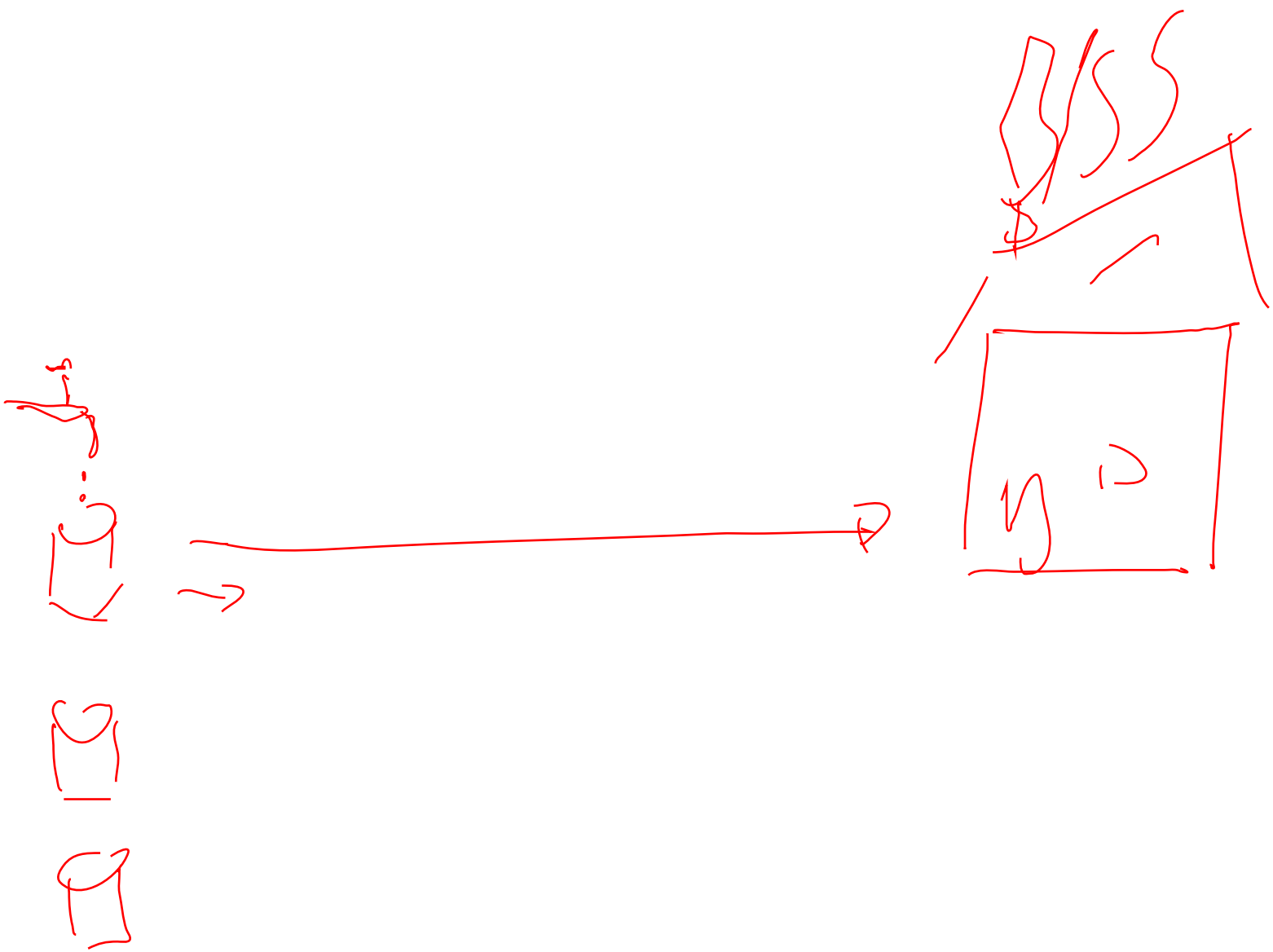
$M' = \frac{1}{2} M + \frac{1}{2} x_0$

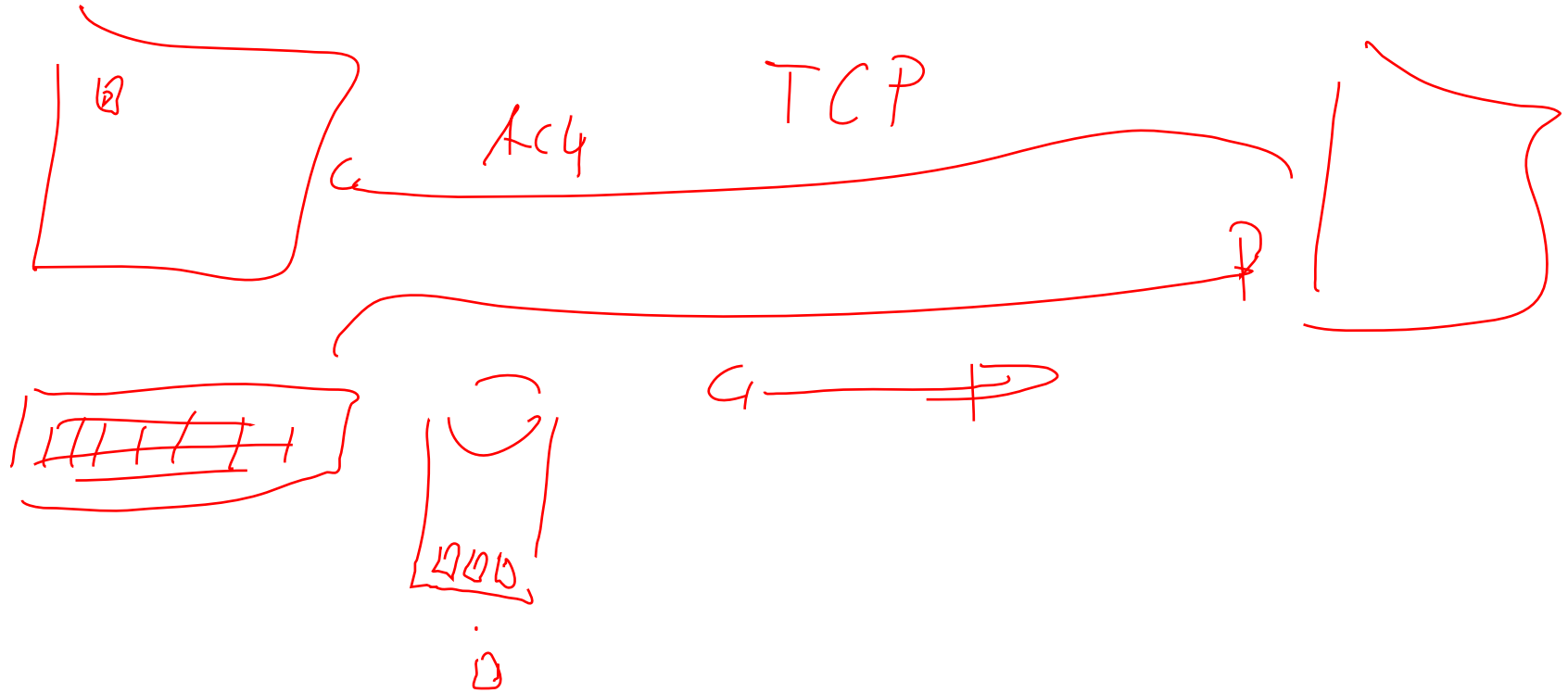
$M' = 90\% \, M + 10\% \, x_0$

CoNe Freiburg

TCP - Algorithmus von Nagle

- 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




TCP
Ack

CoNe Freiburg

Flusskontrolle

- Problem: Schneller Sender und langsamer Empfänger
  - Der Sender lässt den Empfangspuffer des Empfängers überlaufen
  - Übertragungsbandweite wird durch sinnlosen Mehrfachversand (nach Fehlerkontrolle) verschwendet
- Anpassung der Frame-Sende-Rate an dem Empfänger notwendig

Langsamer Empfänger

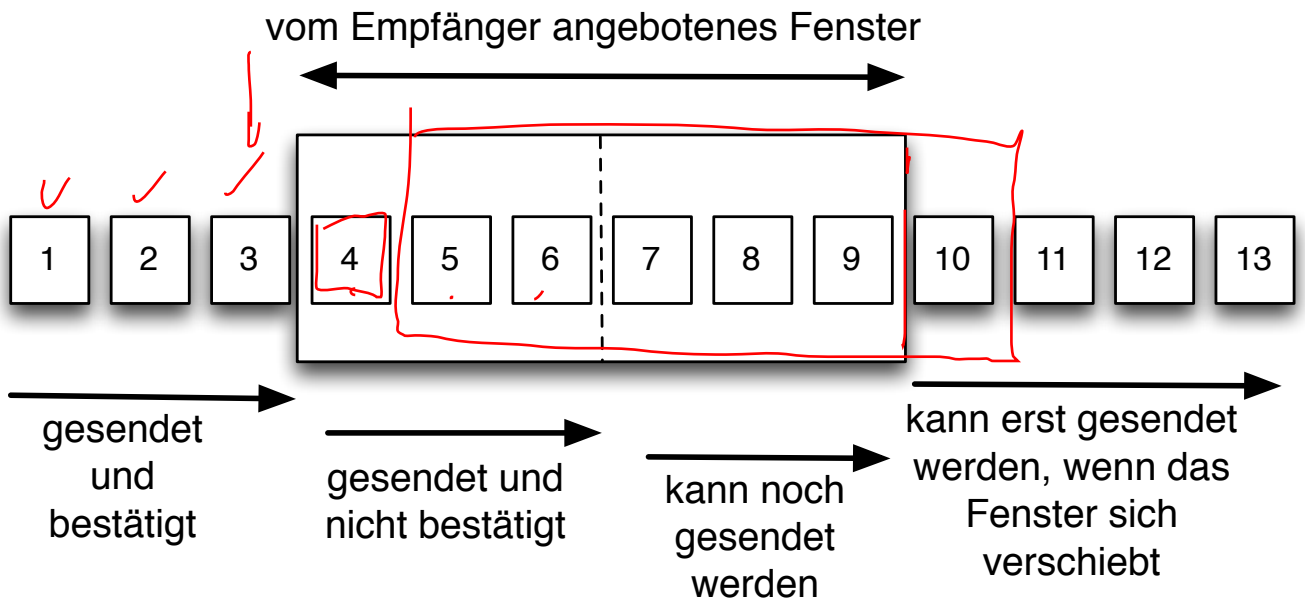
Schneller Sender

UNI FREIBURG

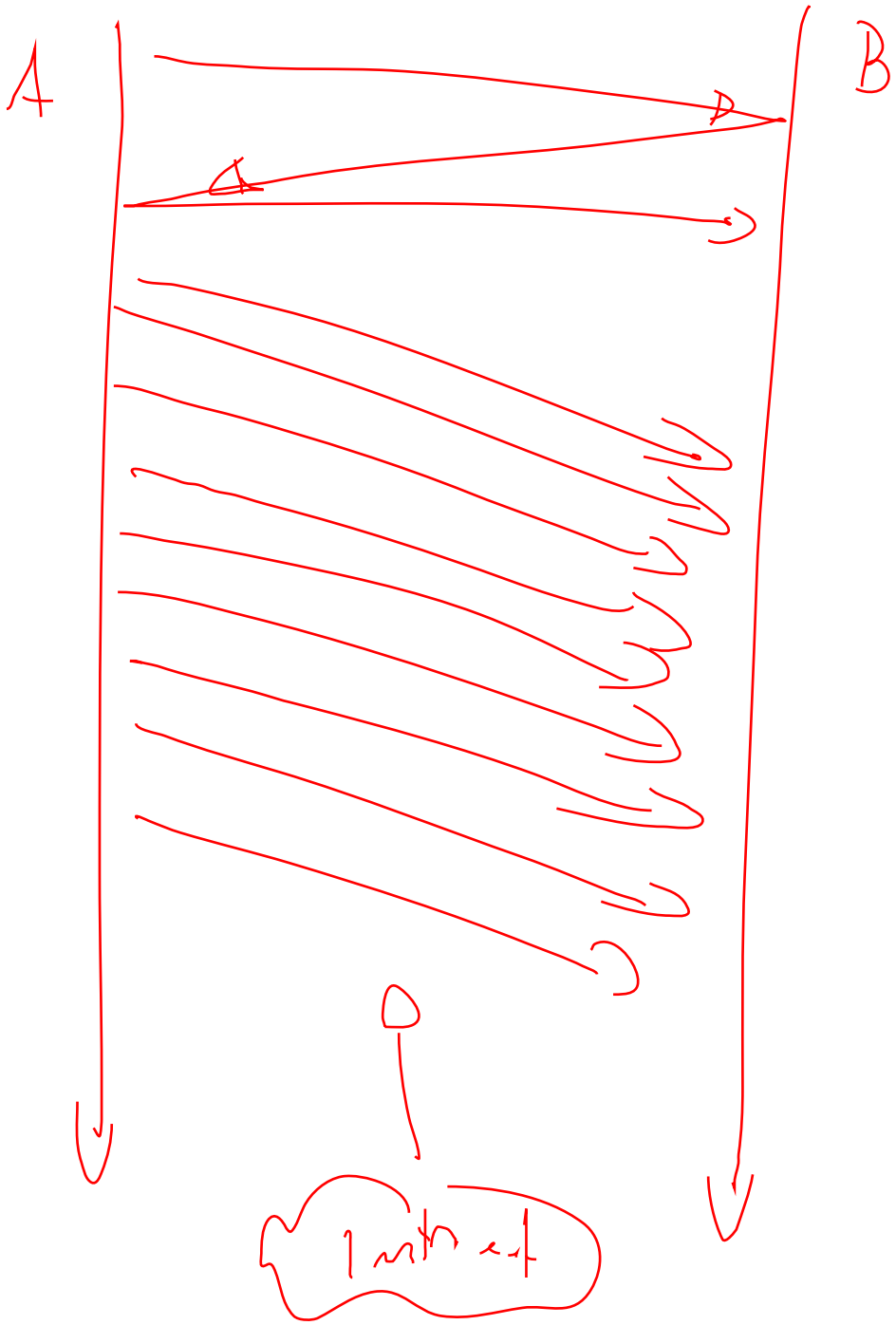
CoNe Freiburg

Gleitende Fenster (sliding windows)

- 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

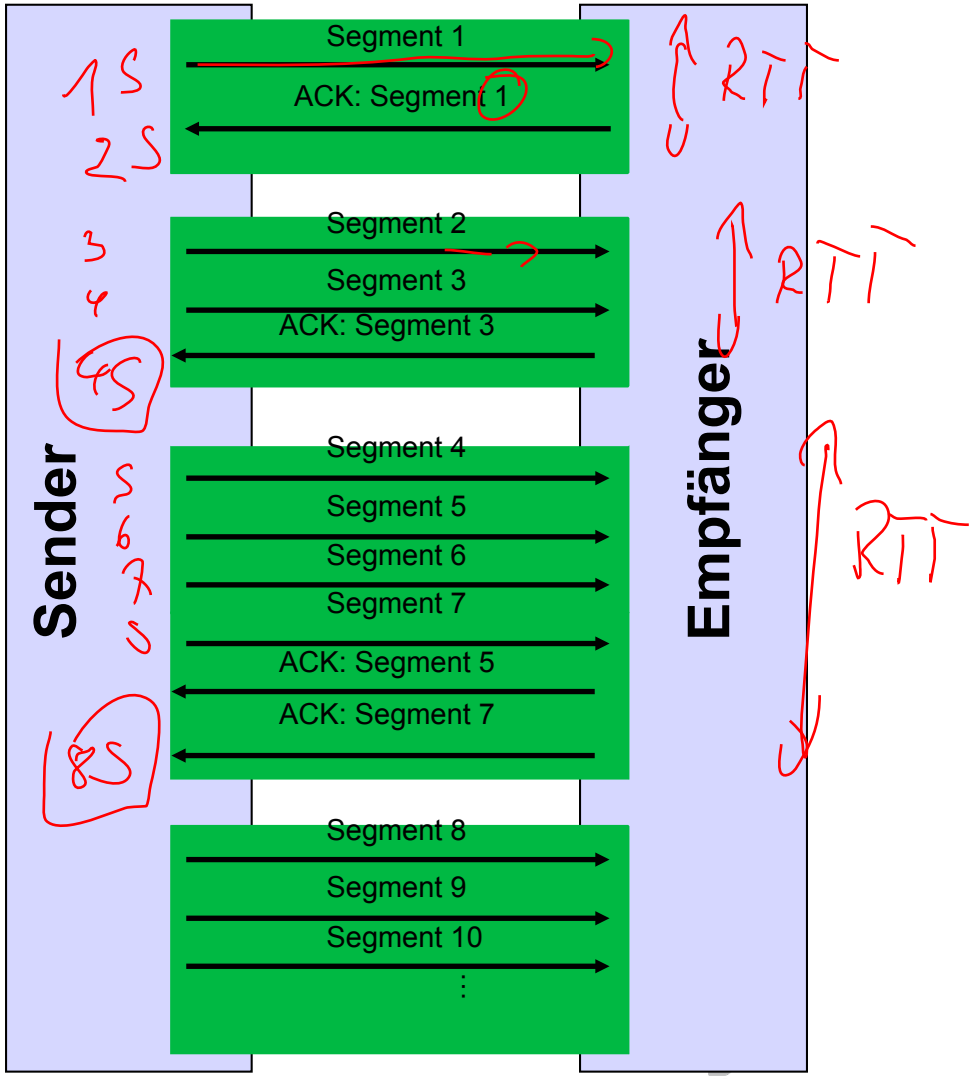


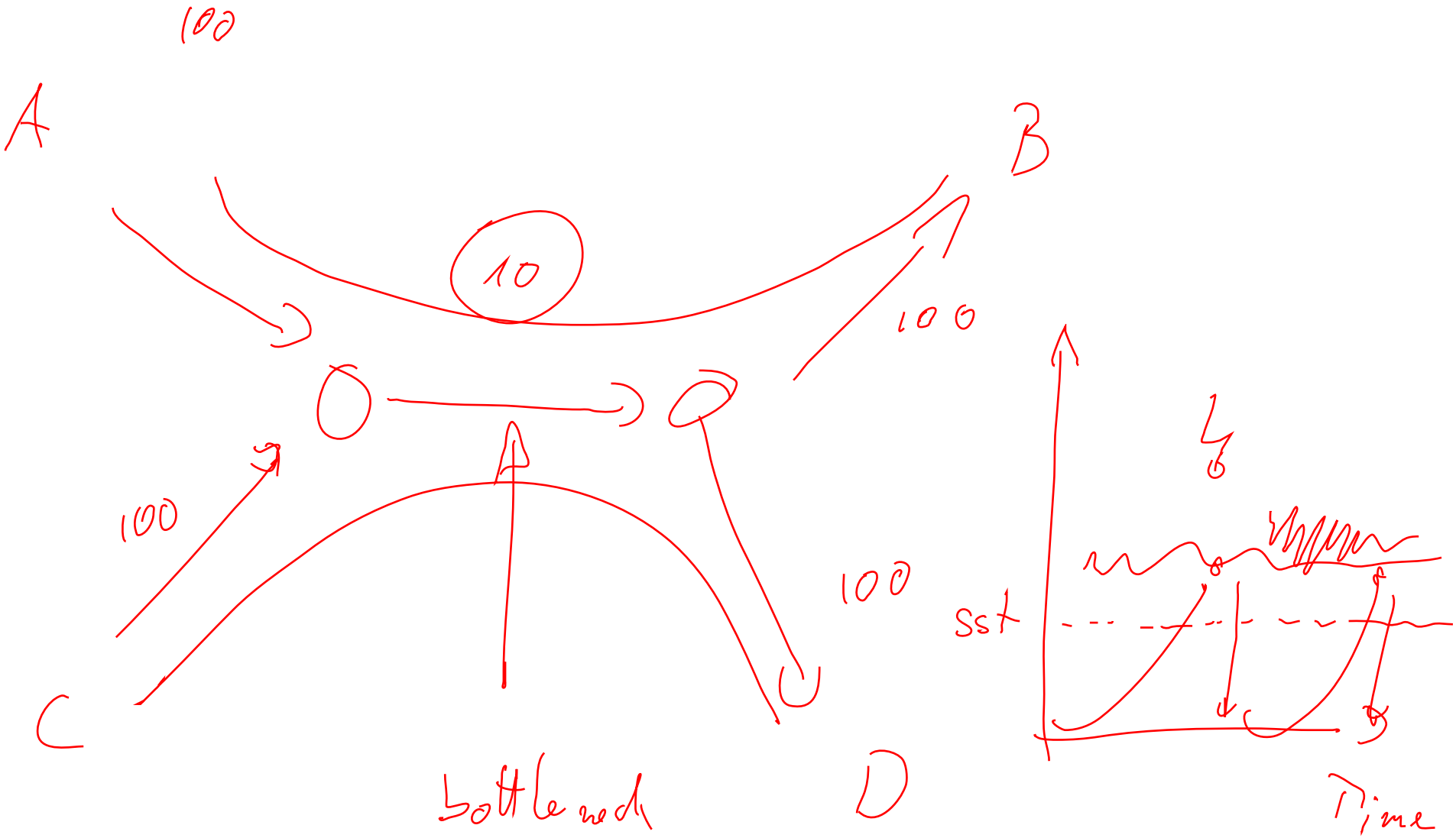
UNI FREIBURG


A
B

Slow Start
Congestion Fenster

- 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




100
A
B
10
100
100
100
C
bottleneck
D
sst
Time

CoNe Freiburg

Stauvermeidung in TCP Tahoe

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**

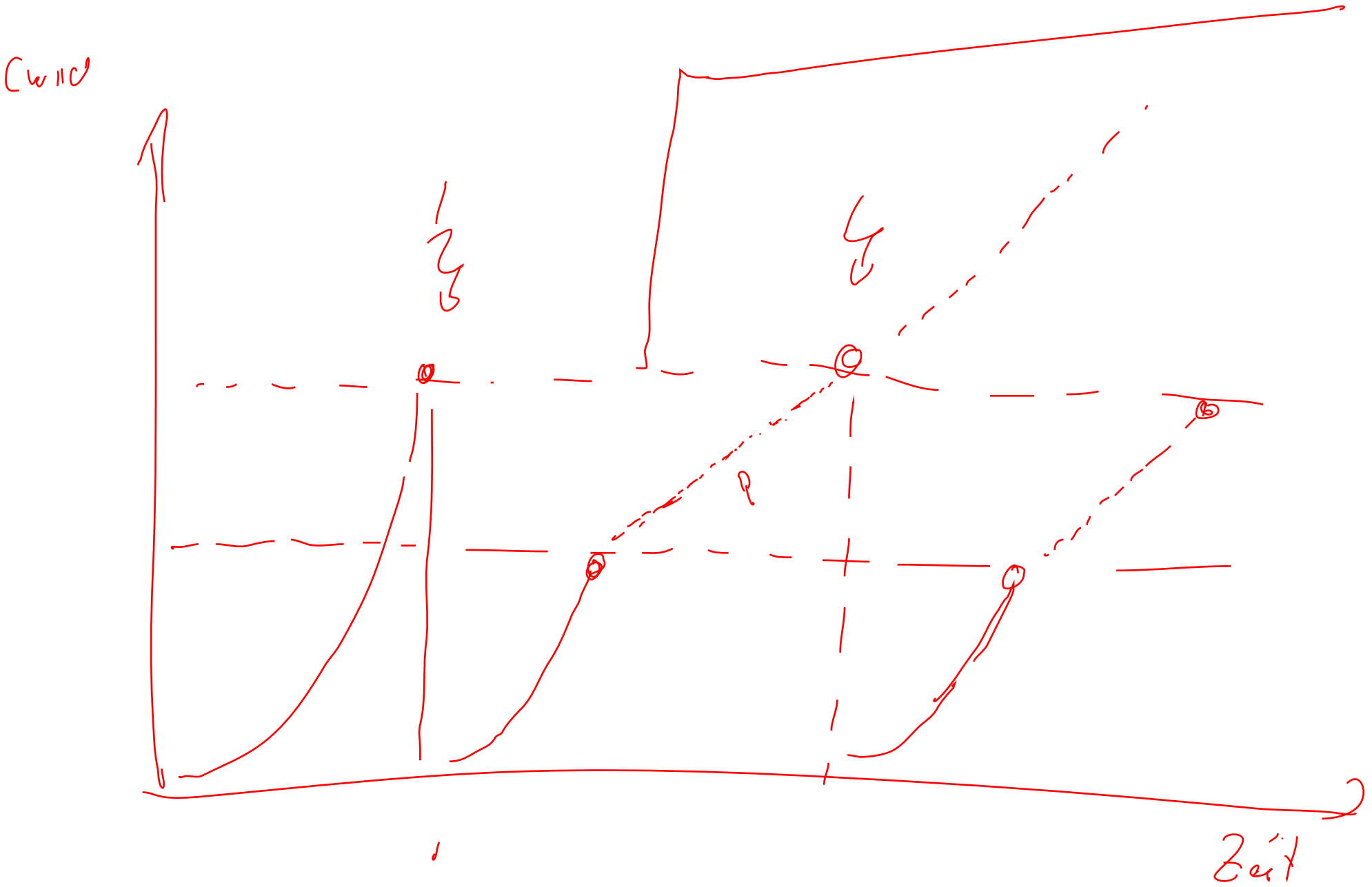
- Werden Segmente bestätigt und cwnd ≤ ssthresh, dann
  - slow start: cwnd ← cwnd + S

x ← 2·x, bis x = y

- Werden Segmente bestätigt und cwnd > ssthresh, dann additively increasing

cwnd ← cwnd + S $\frac{S}{cwnd}$

x ← x +1


Zeit

CoNe Freiburg

IIF INSTITUT FÜR INFORMATIK FREIBURG

Systeme II

5. Die Transportschicht

Christian Schindelhauer
Technische Fakultät
Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg

UNI FREIBURG