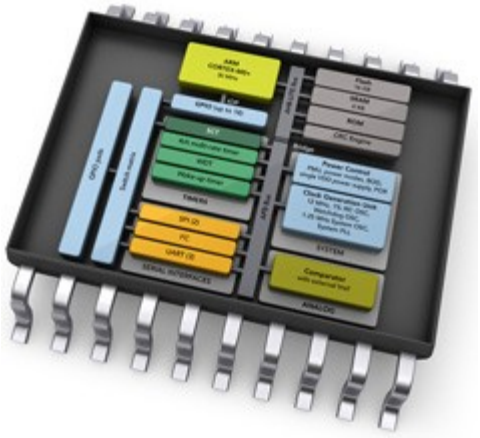
FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



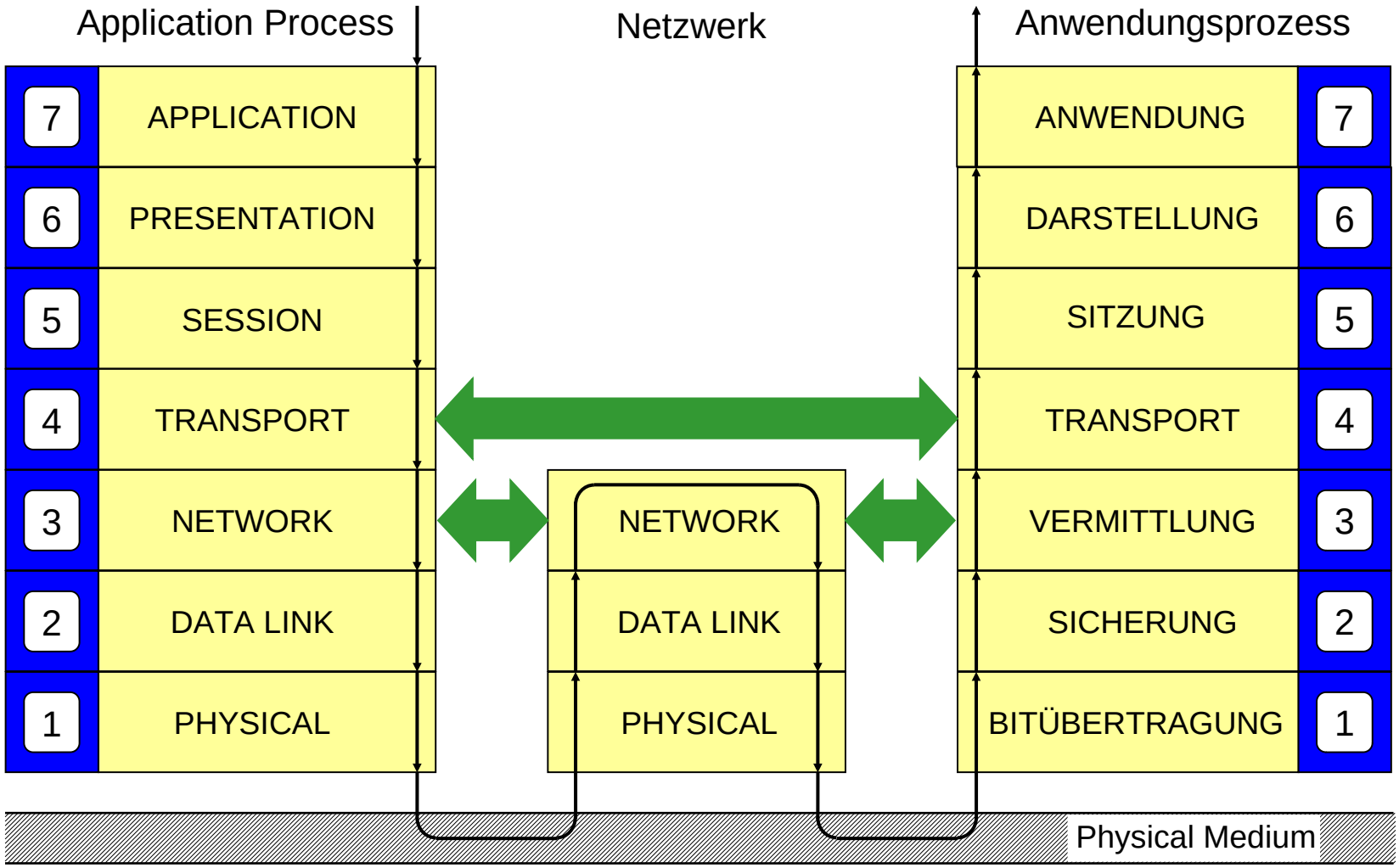
Arbeitsweise der Transportschicht

Transport- und Vermittlungsschicht

- Vermittlungsschicht: logische Kommunikation zwischen *Rechnern in einem Netz aus Netzen*
  - Anbindung von Rechnern in die Netze
  - Regelung des netzübergreifenden Verkehrs:
    - > Kopplung von lokalen Netzen
    - > Globale Adressierung
    - > Routing (Weiterleitung über Zwischenstationen) von Datenpaketen

- Transportschicht: logische Kommunikation zwischen *Prozessen*
  - Benutzt und erweitert die Dienste der Vermittlungsschicht
  - Das erzielbare Dienstangebot wird daher von den Fähigkeiten der Vermittlungsschicht beeinflusst!

Transport- und Vermittlungsschicht



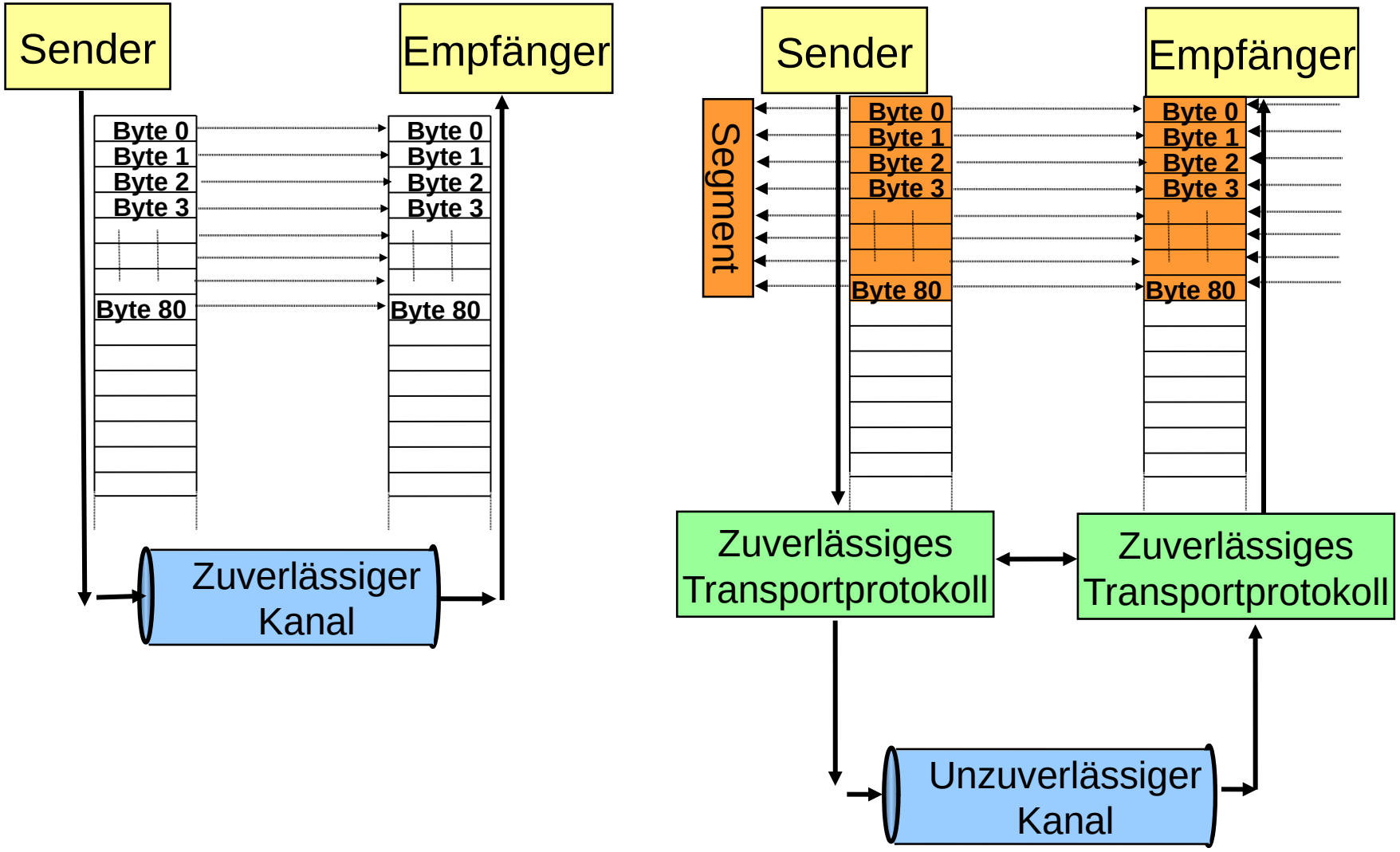
Im Internet wird eine zuverlässige Kommunikation zwischen Prozessen entweder auf Transport- oder auf Anwendungsebene realisiert

Viele heute besprochene Sicherungsmechanismen können Sie in anderen Netzen aber auch in der Sicherungsschicht finden!

Internet-Protokolle – TCP und UDP

- Transport- und Vermittlungsschicht im Internet
  - Verbindungsorientierte Kommunikation
  - Verbindungslose Kommunikation
- Transmission Control Protocol (TCP) -> zuverlässig
  - Kommunikationsprimitive, Sockets, Ports
  - Virtuelle Verbindungsorientierung: Modell zweier Streams die die Prozesse miteinander verbinden; „Open“-“Close“ der Streams sind die virtuellen Verbindungen
  - Flusskontrolle, Staukontrolle
- User Datagram Protocol (UDP) -> unzuverlässig
  - Kommunikationsprimitive, Sockets, Ports
  - Postkartenmodell: Ein Empfangspunkt (Port) für alle Nachrichten, versenden ohne Absprache mit dem Empfänger

Zuverlässiges Transportprotokoll



Ausgangspunkt

- Wie können eigentlich zuverlässige Protokolle über dem unsicheren IP implementiert werden?
  - Pakete können verloren gehen
  - Die Reihenfolge der Pakete kann sich ändern
  - Pakete können fehlerhaft empfangen werden

- Anmerkung
  - Im folgenden gehen wir von einer existierenden „Verbindung“ zwischen Sender und Empfänger aus
  - Sender und Empfänger haben also bereits Nachrichten ausgetauscht und wissen voneinander
  - Es gibt keine Verzögerung bei der Beschaffung oder dem Zustellen neuer Nachrichten an höhere Schichten

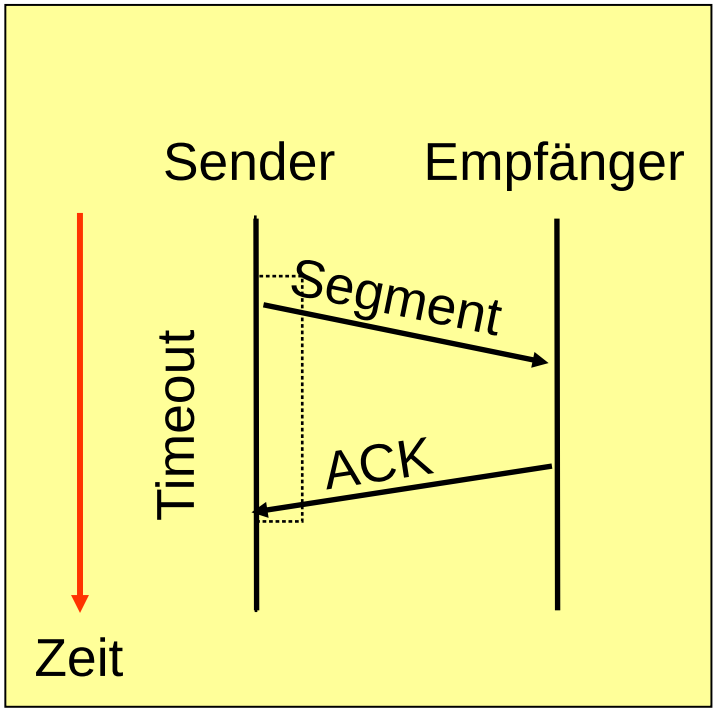
Zuverlässige Transportprotokolle

- Ausgangspunkt:
  - Übertragung eines kontinuierlichen Datenstroms in Nachrichteneinheiten (Segmente), die identifizierbar sind
  - Sender und Empfänger wissen voneinander (verbindungsorientiert)
- Erforderliche Mechanismen
  1. Fehlererkennung und ggf. -behebung
     > Der Empfänger muss Bitfehler in der Übertragung erkennen
     > Der Empfänger muss das Fehlen (kann aber ggf. später noch ankommen) eines Segmentes erkennen und darauf geeignet reagieren
  2. Empfänger-Feedback
     > Quittierungsnachrichten zwischen Empfänger und Sender
     > Positive **ACK**nowledgment (**ACK**)
       – Empfänger bestätigt den korrekten Erhalt eines Segments mit einem ACK
     > **Optional: N**egative **A**c**K**nowledgment (**NAK**)
       – Empfänger informiert Sender über „Lücken“ oder Fehler
  3. Neuübertragung
     > Sender überträgt fehlerhafte Segmente erneut
     > Timer- und/oder NAK-gesteuert

Das “Send and Wait”-Protokoll

(gelegentlich auch als “Stop-and-Wait” bezeichnet)

- Automatic Repeat-reQuest (ARQ)
  - Empfänger verschickt Bestätigungen (**ACK**s) bei korrektem Empfang
  - Fehlerhafte Übertragungen werden durch ausbleiben der Bestätigung erkannt und vom Sender durch Wiederholung behoben
- Einfachstes ARQ-Protokoll
  - Segmentweises Vorgehen
  - Timer zur Fehleridentifikation
    - Verschicke ein Segment und warte auf den Empfang des ACKs
    - Bei Empfang eines ACKs wird das nächste Segment verschickt, andernfalls wird die Übertragung wiederholt



Ist dieses Protokoll sicher?

Schlechter Nutzungsgrad der Verbindungskapazität

- Jeder Datenstrom wird in Segmente unterteilt, z.B. 1460 Bytes Nutzdaten
- Nach jedem gesendeten Segment wird auf eine Bestätigung gewartet; während der Wartezeit werden keine weiteren Segmente übertragen

  ⇨ Begrenzung auf ein Paket pro Zyklus (Round-Trip)!!!!

- **Nutzungsgrad:** Verhältnis von genutzter Übertragungszeit zu gesamter Übertragungsdauer (inkl. der Wartezeit)

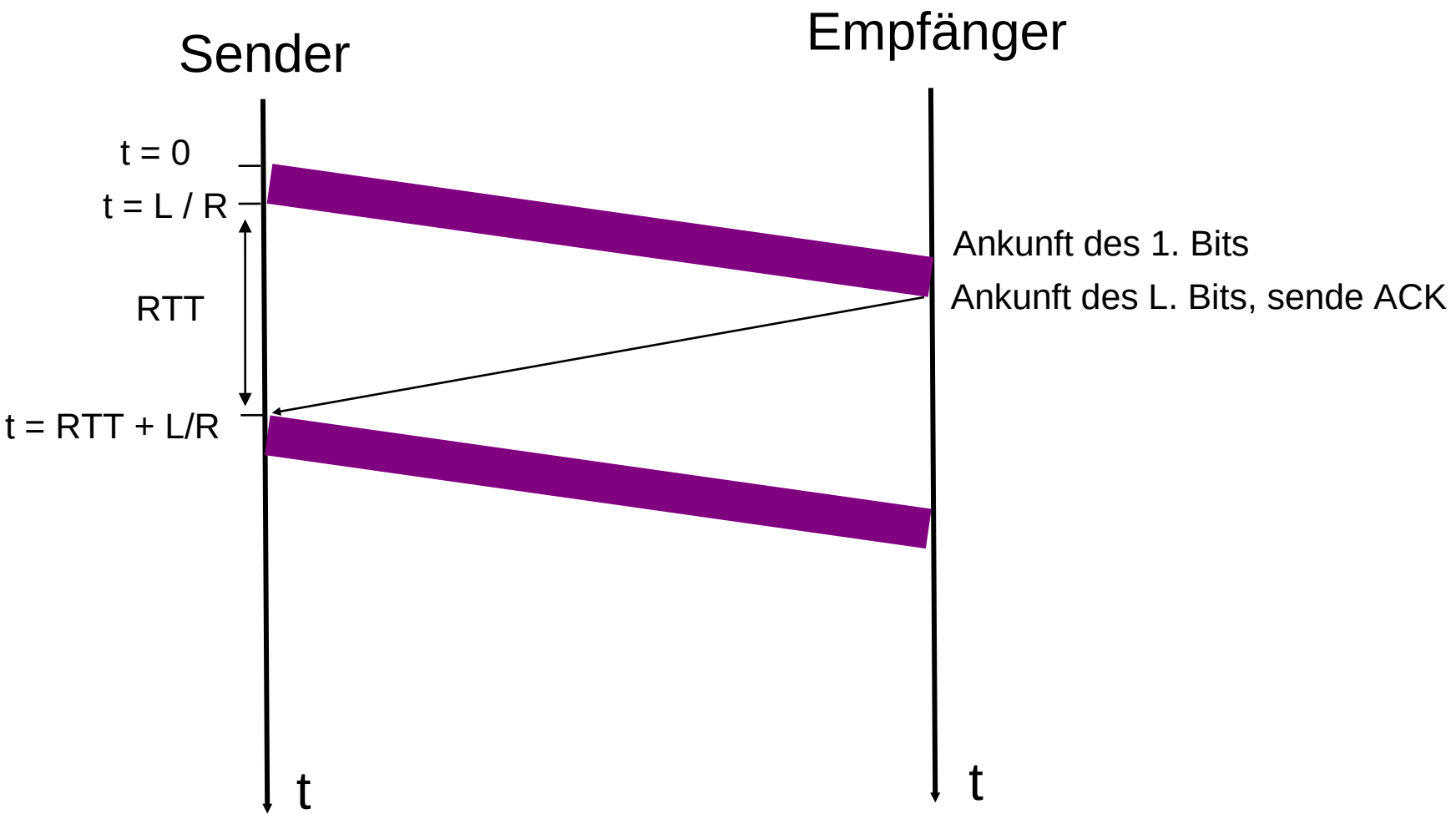
ρ = (genutzte Übertragungszeit) / (genutzte Übertragungszeit + Wartezeit)

Sei L die Länge eines Segmentes [bits]

Sei R die Übertragungsrate des Kanals [bits/s]

→ L / R Zeit zur Übertragung des Segments [s]

Schlechter Nutzungsgrad der Verbindungskapazität



$$\rho = (L/R) / (L/R + RTT)$$

Schlechter Nutzungsgrad der Verbindungskapazität

Beispiel: FastEthernet im LAN

R = 100 Mbit/s

L = 1460 Byte L/R = 116,8 µs

RTT = 3 ms

$\rho = (L/R) / (L/R + RTT) = 0{,}03747 = 3{,}74\%$

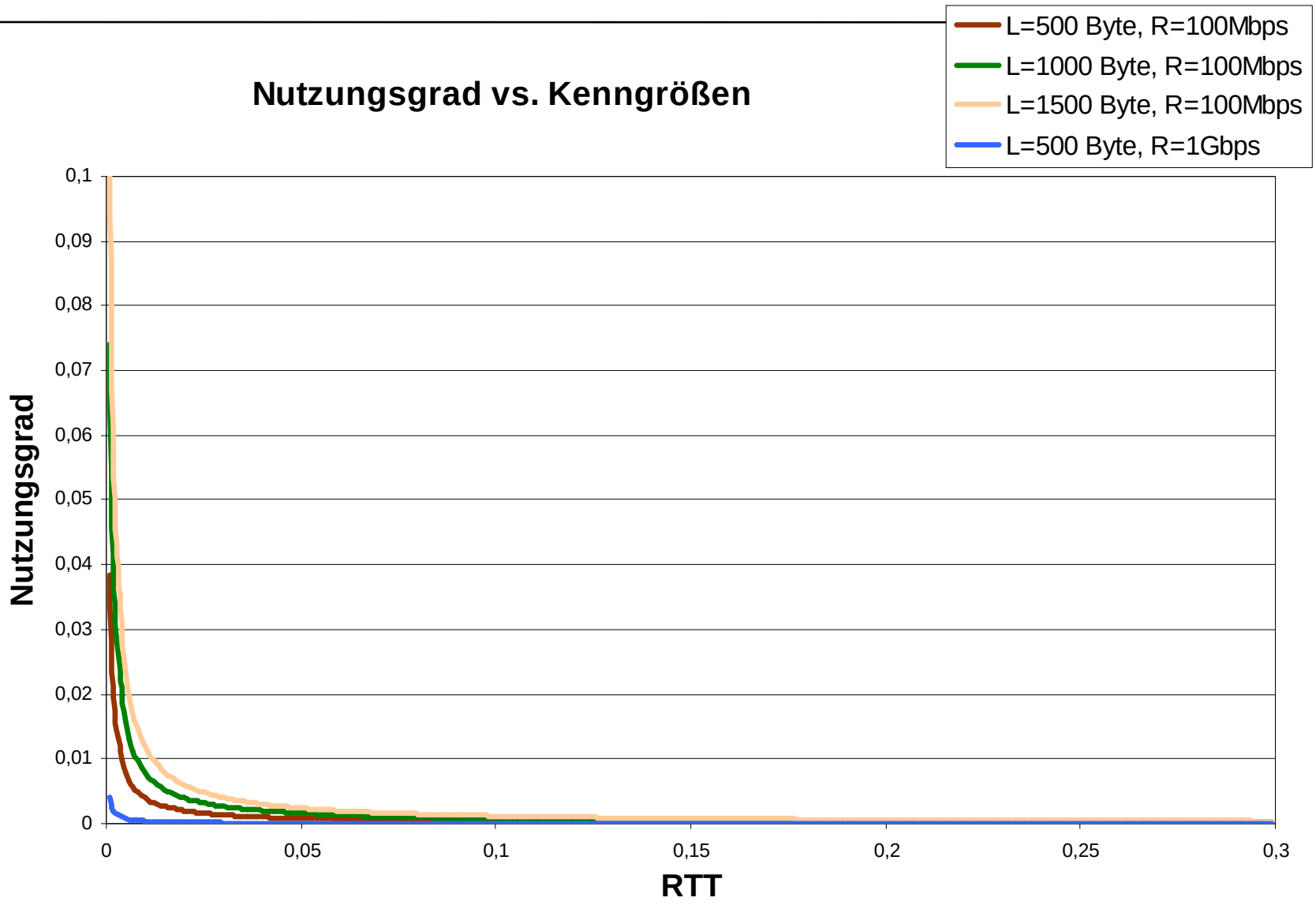
Beispiel: Kommunikation mit Kalifornien

R = 100 Mbit/s

L = 1460 Byte L/R = 116,8 µs

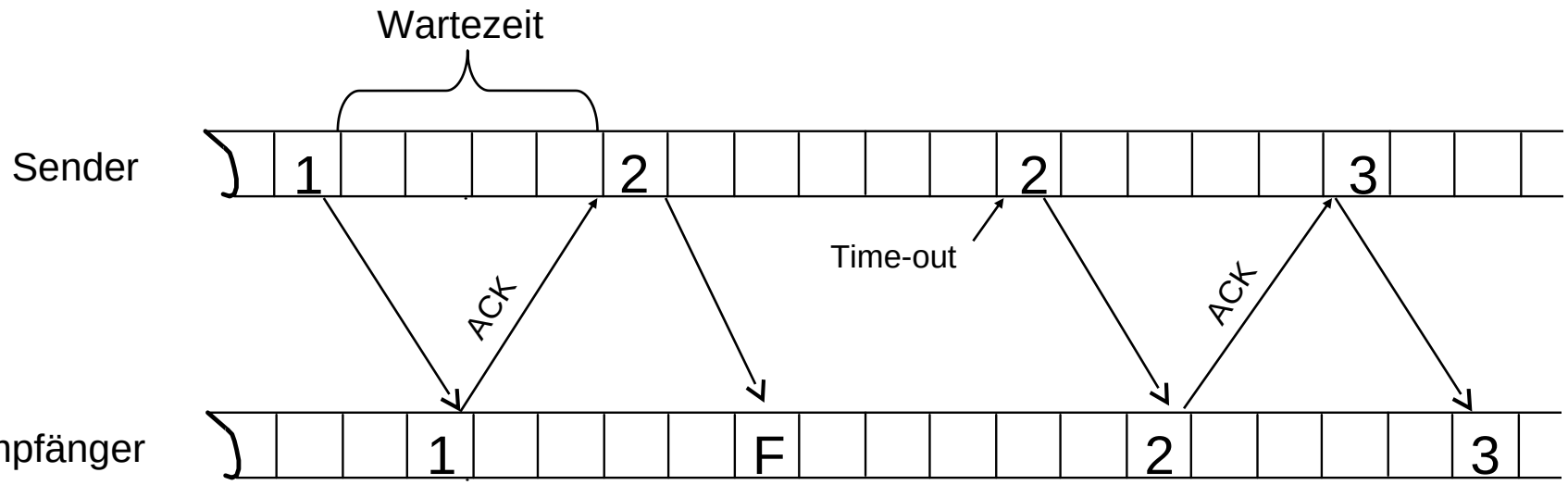
RTT = 200 ms $\rho = (L/R) / (L/R + RTT) = 0{,}05837\%$

Schlechter Nutzungsgrad der Verbindungskapazität



Eins nach dem anderen: Send-and-Wait

- Sender überträgt ein Segment und startet einen Timer
  - Trifft eine Quittung ein: sende das nächste Segment
  - Tritt ein Timeout ein (d.h. läuft der Timer ab): wiederhole das vorherige Segment
- Lange Wartezeiten zwischen den Segmenten! Dadurch Verschwendung von Übertragungskapazität!

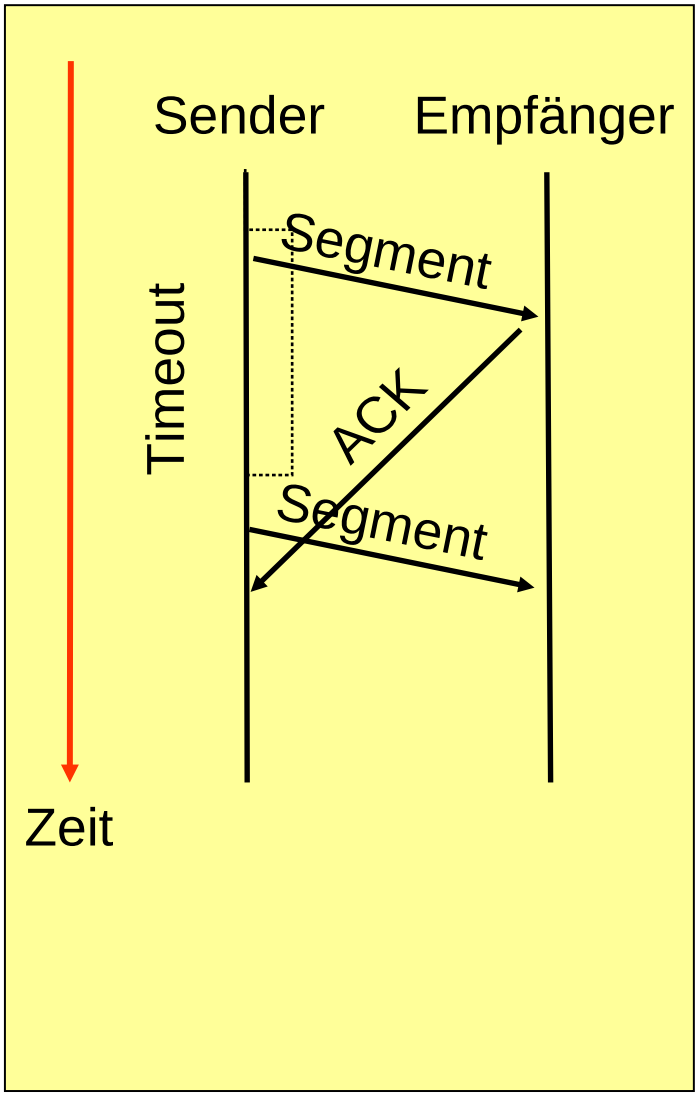


Verfahren bei Fehler/Sonderfällen

Welche Ereignisse können den geschilderten Ablauf stören?

Verfahren bei Fehler/Sonderfällen

Welche Bestätigung haben wir hier empfangen?



Welches Segment haben wir hier empfangen?

Konsequenz

- Wir benötigen eine eindeutige Identifikation der Nachricht
- Damit müssen wir die Segmente und die dazugehörigen Bestätigungen mit einer eindeutigen Nummer versehen

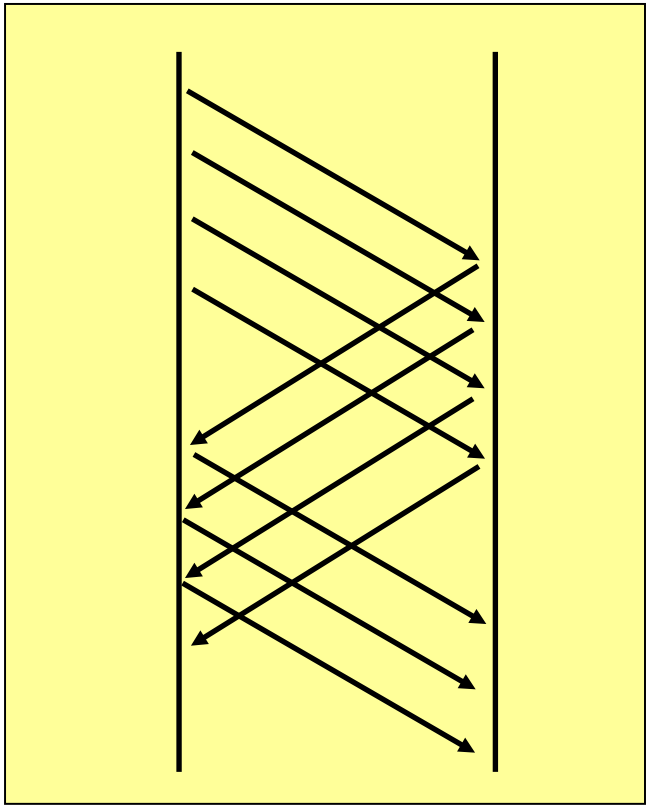
Lösungsansätze:

- Wir verwenden den Byte-Offset im Stream in der Art, dass für das aktuelle Segment angezeigt wird, an welcher Position das 1. Byte liegt, oder wir nummerieren die Nachrichten durch
- Zyklisch, da Zahlenraum begrenzt
- Bestätigungen verweisen entweder auf das **nächste zu erwartende** Byte oder auf die **empfangene** oder als nächste zu erwatende Nummer

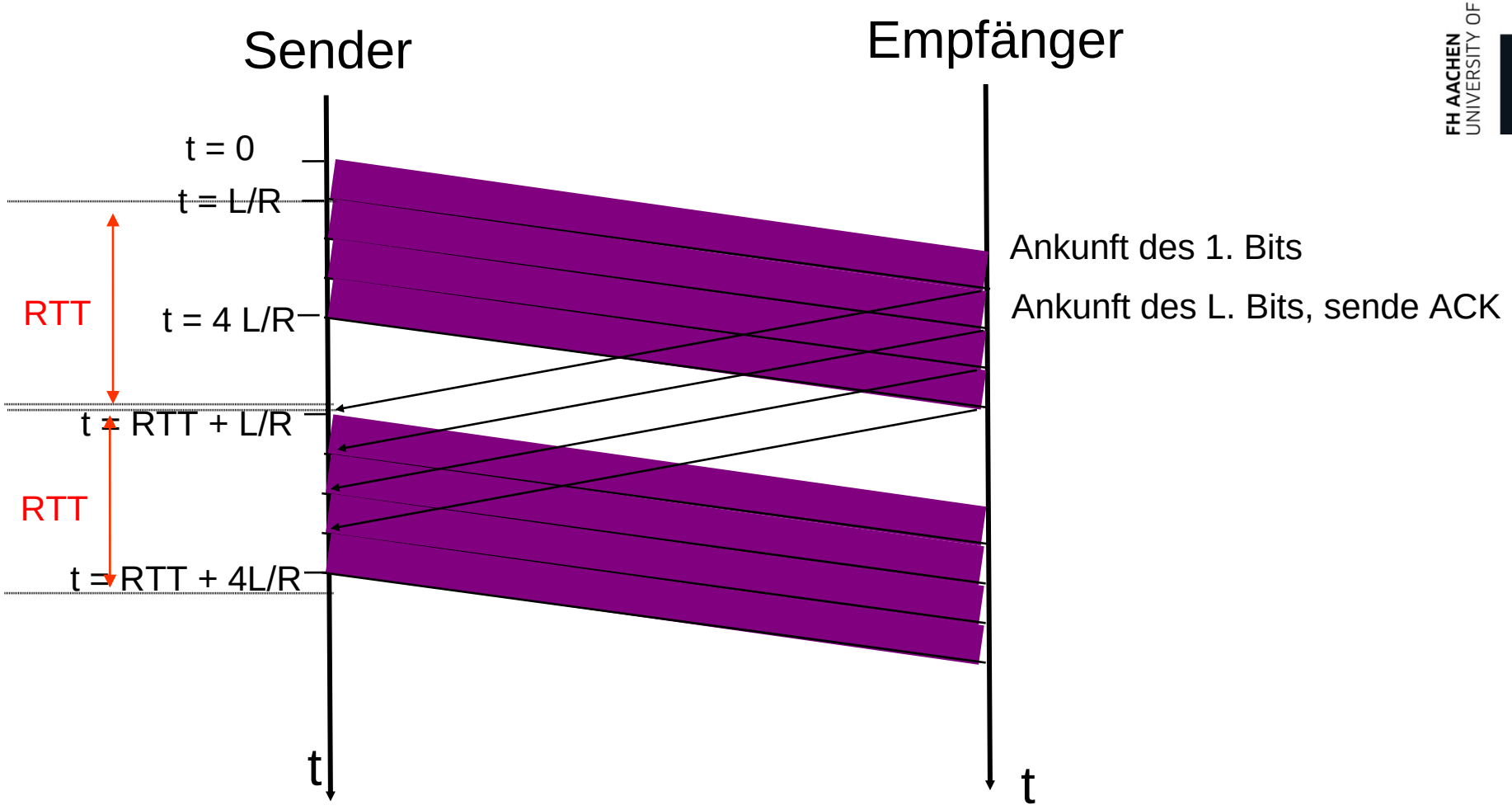
Wie kann man das Netz besser auslasten?

Pipelining:

- Senden ohne vorheriges ACK
- Fensterbreite beim Sender: Anzahl der Segmente, die ohne Bestätigung gesendet werden
- Fensterbreite beim Empfänger: Anzahl der Segmente, die auch bei Lücken oder Fehlern zwischengespeichert werden

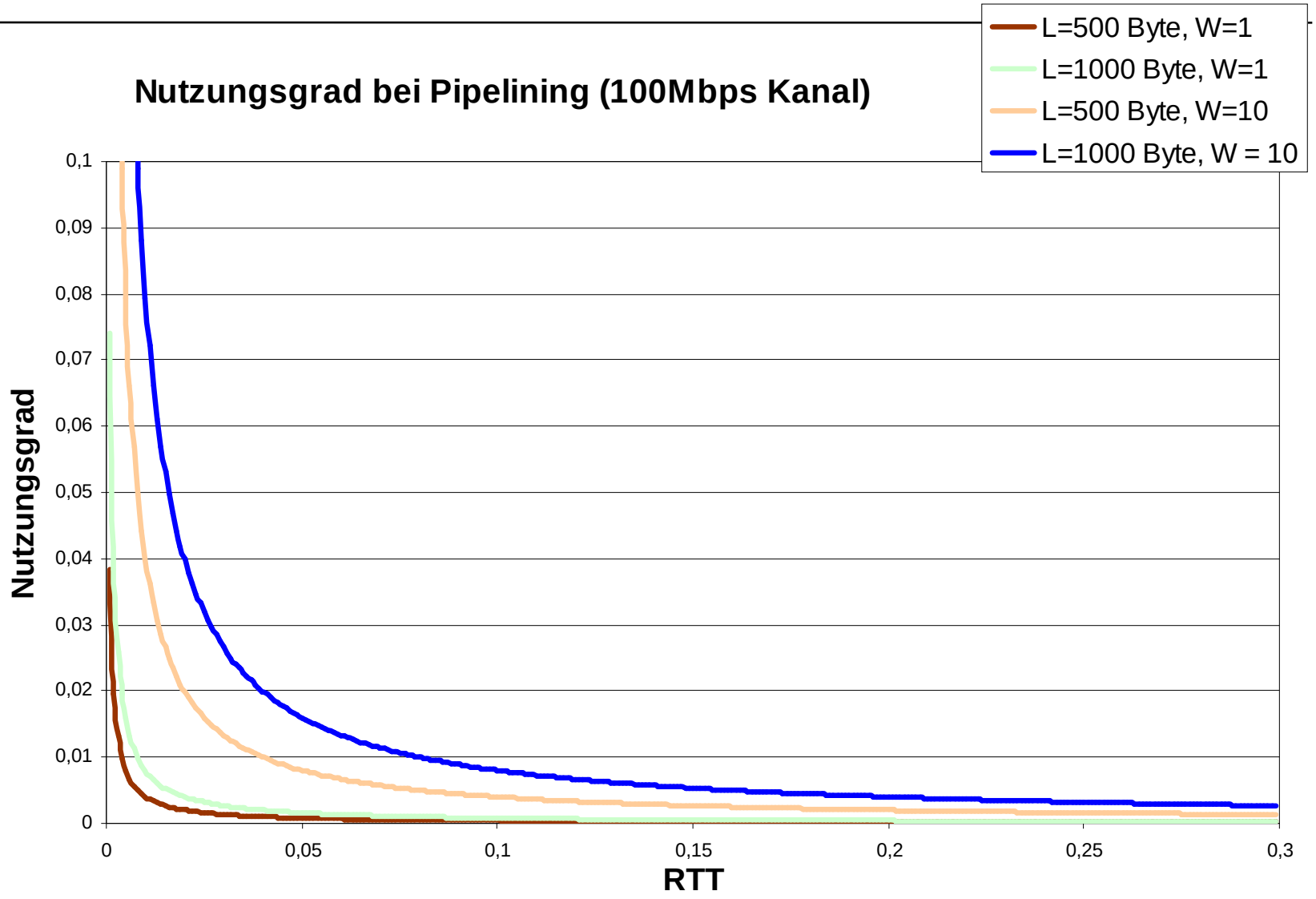


Flusskontrolle folgt dem Pipelining-Prinzip



$$\rho = (4L/R) / (4L/R + (RTT + L/R - 4L/R)) = (4\ L/R) / (L/R + RTT)$$

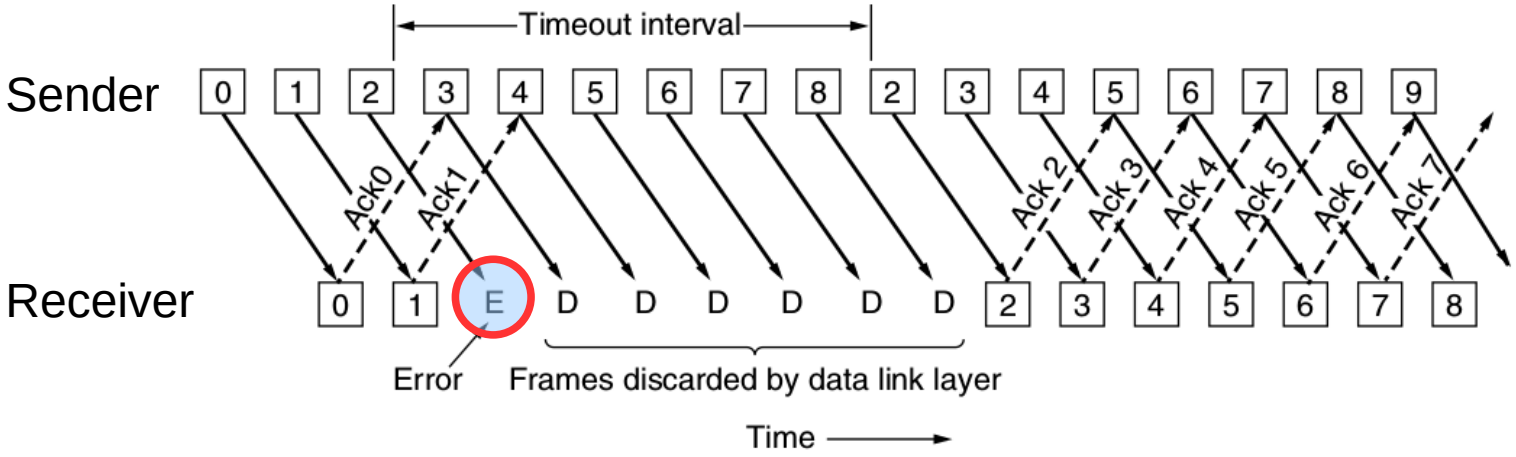
Schlechter Nutzungsgrad der Verbindungskapazität



Wir erhöhen hier aber die Komplexität im Fehlerfall

Einfachste Lösung: **Go-Back-N**

Go-Back-N



- Der Empfänger sendet nach dem Fehler kein ACK mehr
- Der Sender läuft dadurch in ein Timeout und fängt wieder ein Segment nach dem letzten bestätigten Segment wieder an
- Alle direkt nach dem Fehler versendeten Segmente werden verworfen
- → Der Sender hat eine „Fensterbreite“ > 1
- → Der Empfänger hat eine „Fensterbreite“ von 1

FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de