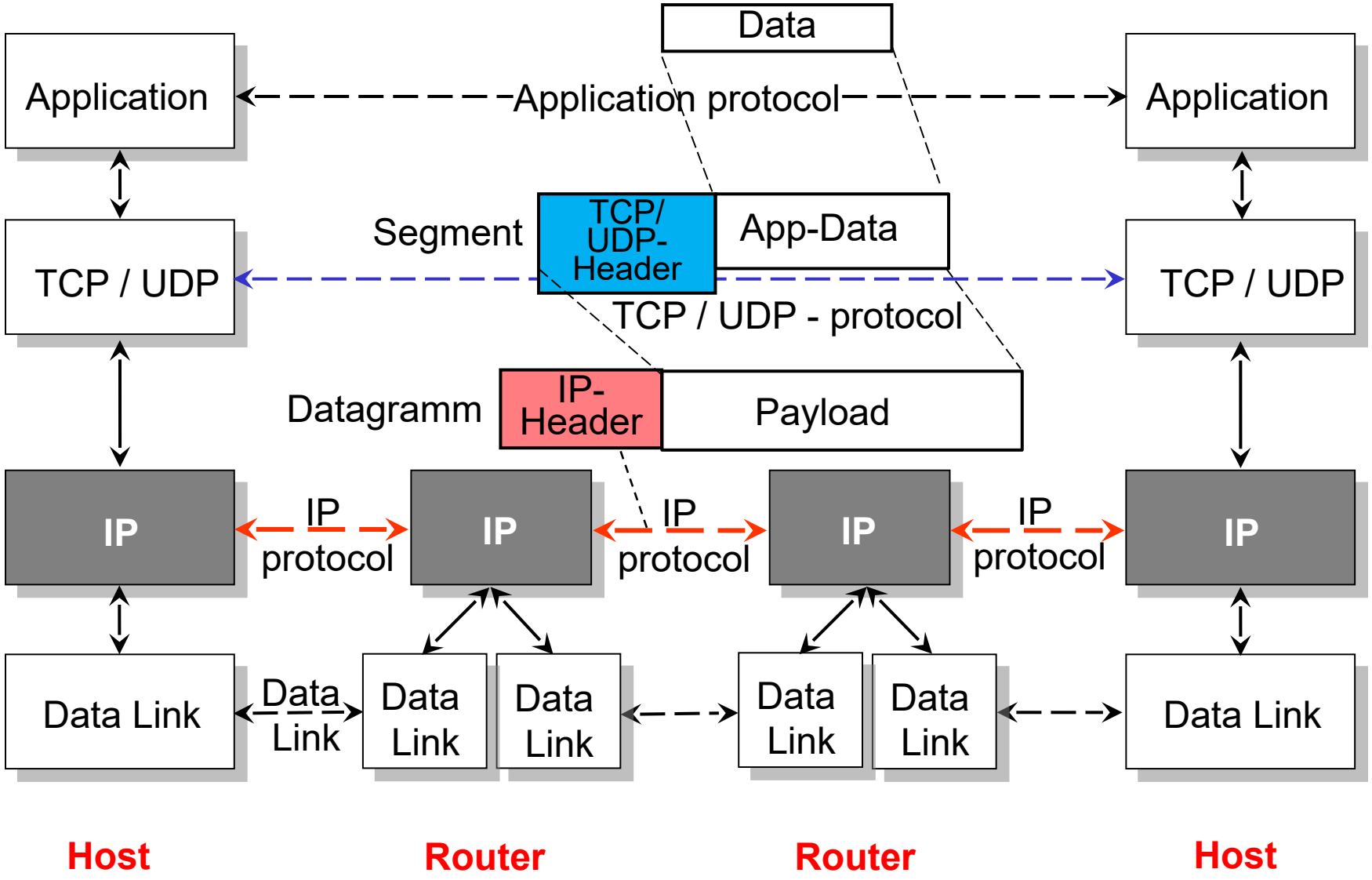
Einführung und Übersicht

Internet Protokoll – Stack in Hosts und Routern



Transmission Control Protocol (TCP)

- Verbindungsorientierter Dienst zwischen Endsystemen
  - „Handshake“ Verfahren zum Aufbau einer Verbindung vor dem Datentransfer
- zuverlässiger, reihenfolgerichtiger bidirektionaler Datentransfer
- TCP Beispiele
  - http/s (Web), smtp (Mail), ftp (File Transfer), ssh (login), …

User Datagram Protocol (UDP)

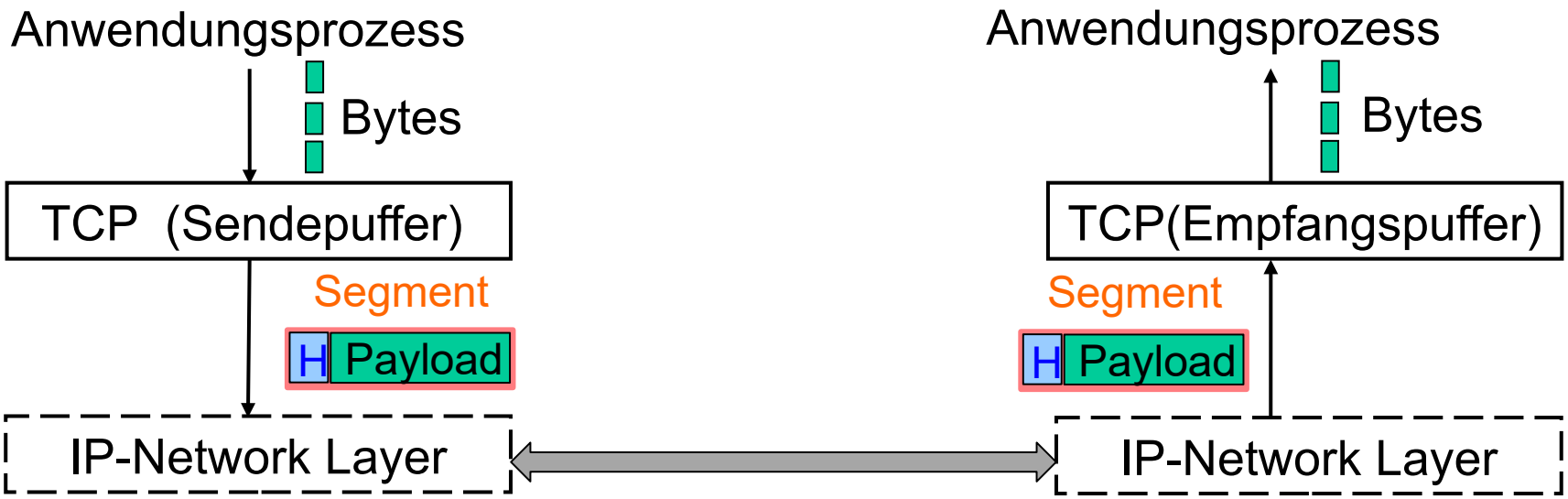
- Verbindungsloser Dienst zwischen Endsystemen
  - kein Verbindungsaufbau vor dem Datentransfer
- unzuverlässiger, nicht reihen-folgetreuer Datentransfer
- UDP Beispiele
  - Multimedia Anwendungen, Video Streaming, VoIP, DNS

Beide Protokolle garantieren keine Dienstgüte

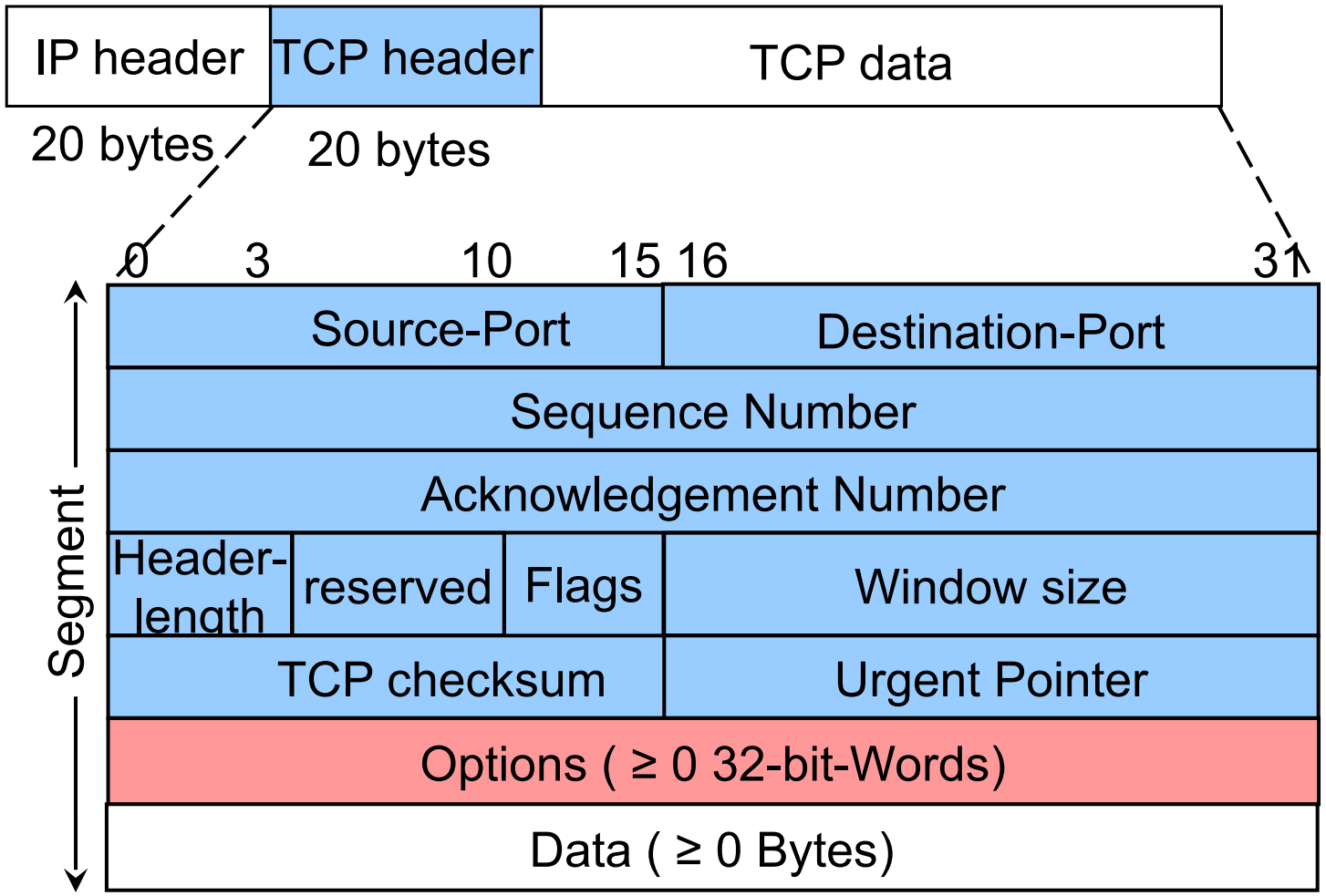
- keine garantierte Verzögerungszeit
- keine garantierte Bandbreite

TCP - Bytestream Services

- Nimmt von der Quell- Anwendung über Socket einen Bytestrom entgegen
- Liefert der Ziel-Anwendung über Socket einen Bytestrom aus
- TCP übergibt IP-Protokoll Segmente als Dateneinheiten
  - Segmentgröße = Nutzdaten + TCP-Header
  - Maximum Segment Size (MSS): 1500, 536 und 512 Byte
    - entspricht der maximalen Anzahl von Nutzdaten im TCP Segment
  - TCP nummeriert einzelne Bytes



Format von TCP - Segmenten

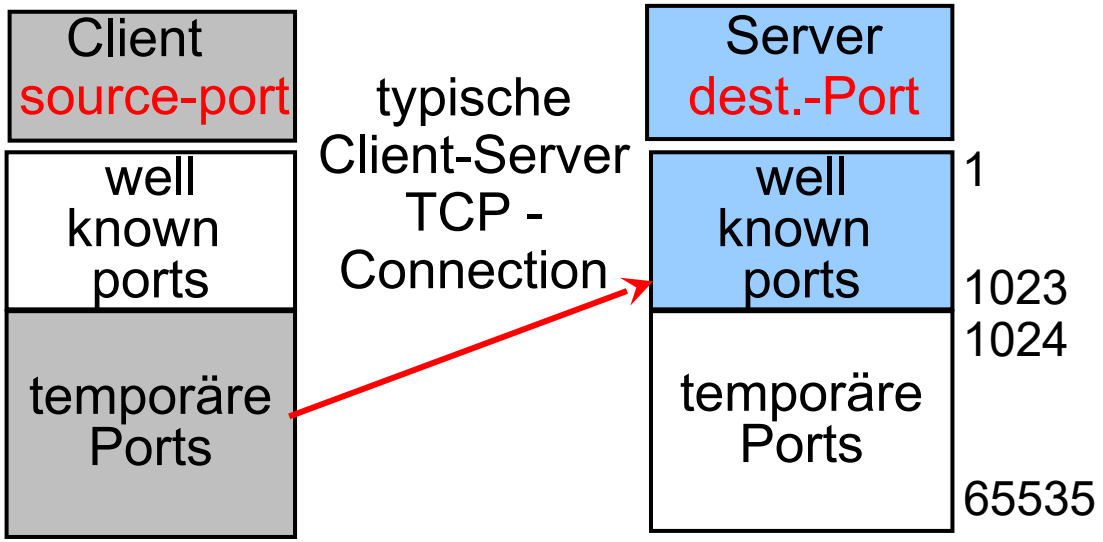


Well Known Ports

Portnummern für spezielle Anwendungen sind in RFC 793 festgelegt

ftp	20,21	/ tcp	file transfer protocol
ssh	22	/ tcp	remote terminal
smtp	25	/ tcp	simple mail transfer protocol
dns	53	/ udp	domain name system
tftp	69	/ udp	trivial file transfer protocol
http	80	/ tcp	hyper text transfer protocol
rlogin	513	/ tcp	remote login

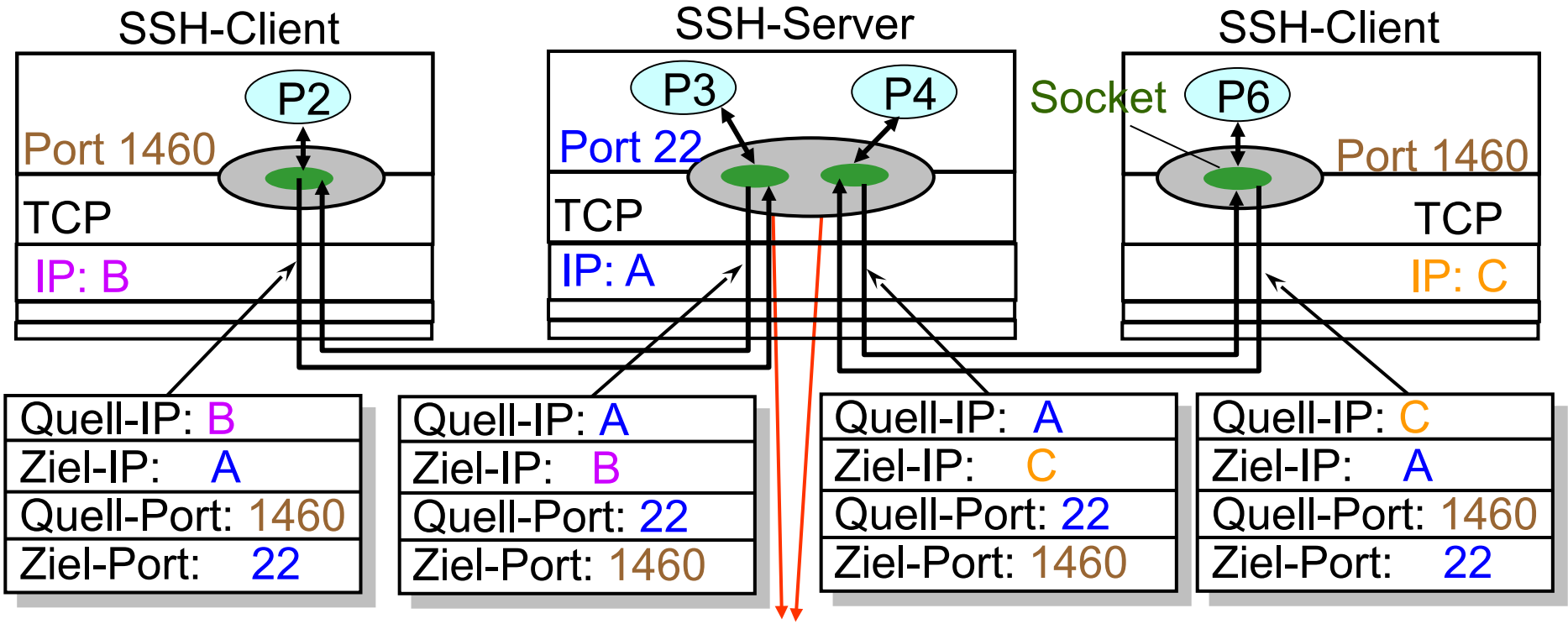
- Die well known ports sind i.d.R. den Server-Prozessen vorbehalten
- Typische Anwendung:
  - Server horcht auf „well known ports“



Adressierung von verbindungsorientierten Services

Beispiel: TCP Client/Server-Anwendung

- Auf einem Server kommunizieren zwei Prozesse (P) der gleichen Anwendung mit verschiedenen Clients
- Beide Clients benutzen zufällig den gleichen Quell-Port

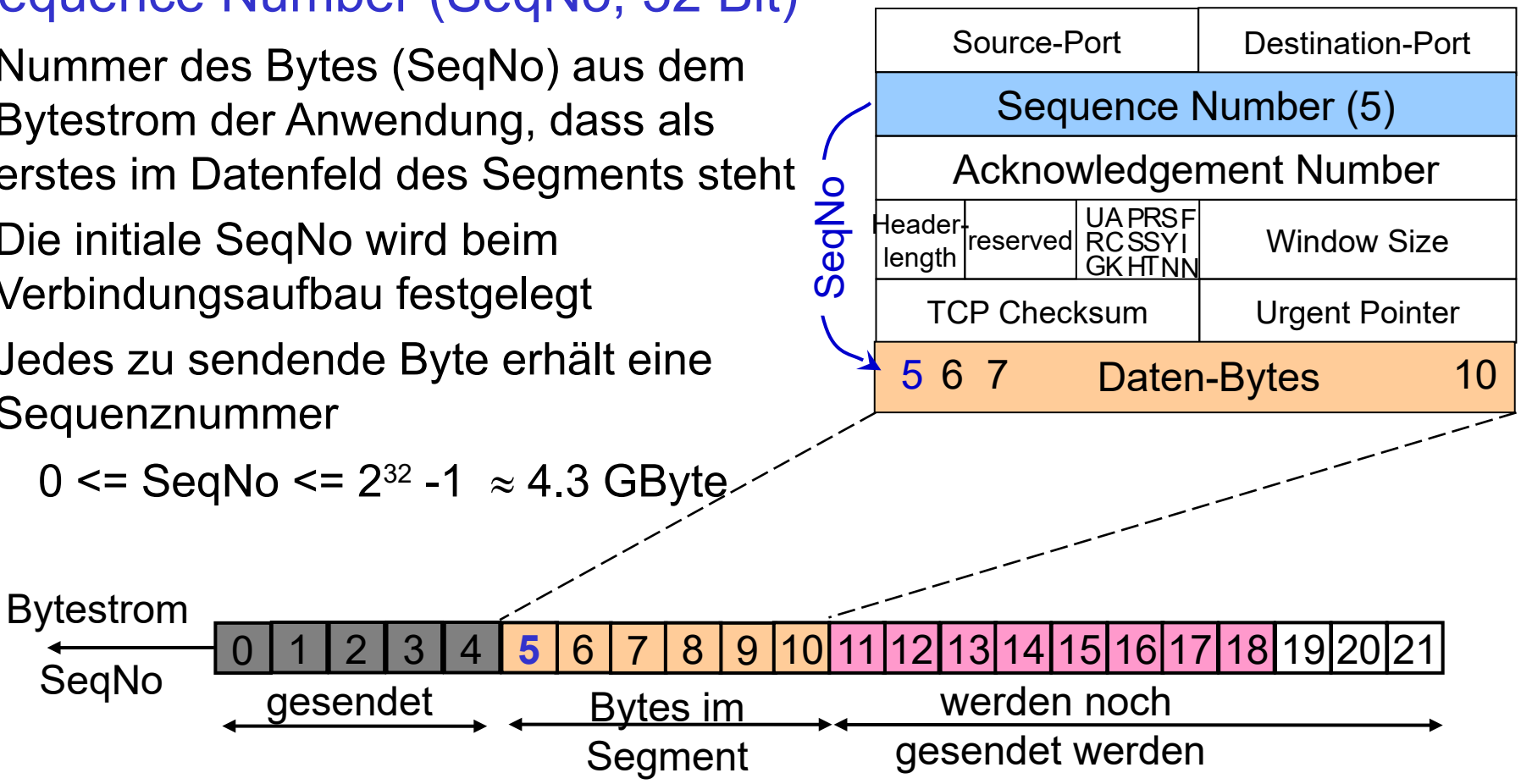


Die verschiedenen Anwendungsadressen (4-Tuple <Client: IP-Adresse, Port> und <Server: IP-Adresse, Port>) unterscheiden beide Prozesskommunikationen

Sequence Number (SeqNo, 32 Bit)

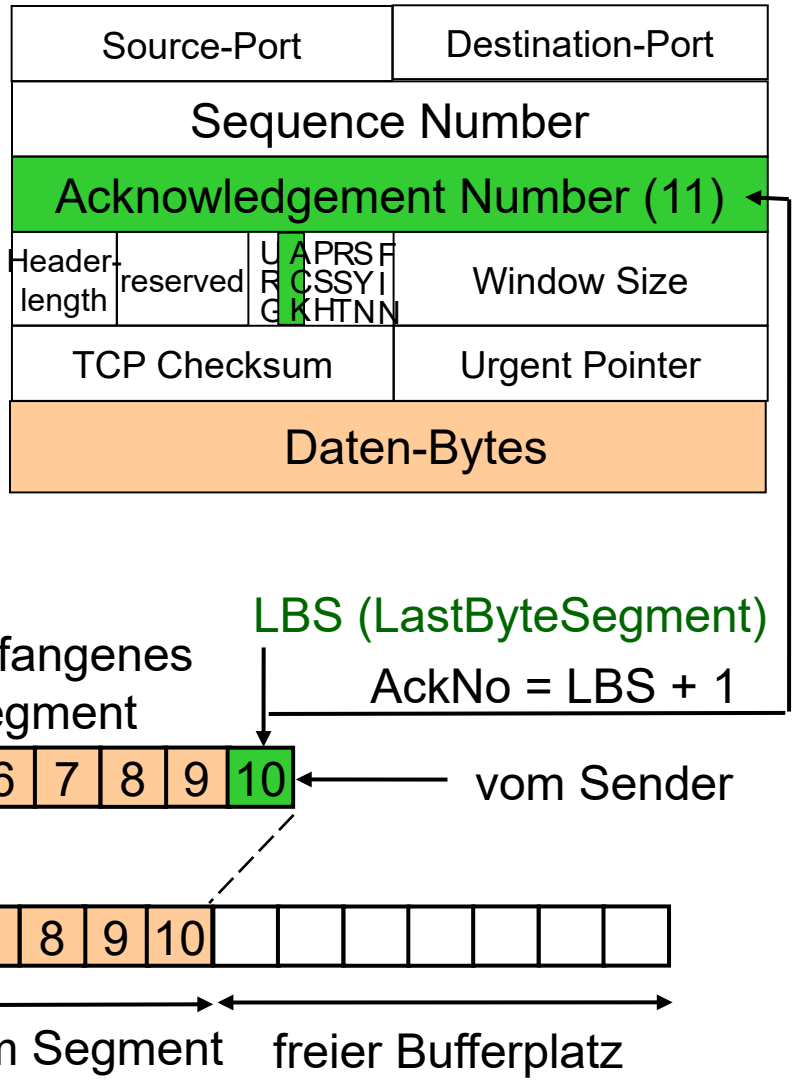
- Nummer des Bytes (SeqNo) aus dem Bytestrom der Anwendung, dass als erstes im Datenfeld des Segments steht
- Die initiale SeqNo wird beim Verbindungsaufbau festgelegt
- Jedes zu sendende Byte erhält eine Sequenznummer

$0 <= SeqNo <= 2^{32} - 1 \approx 4.3$ GByte

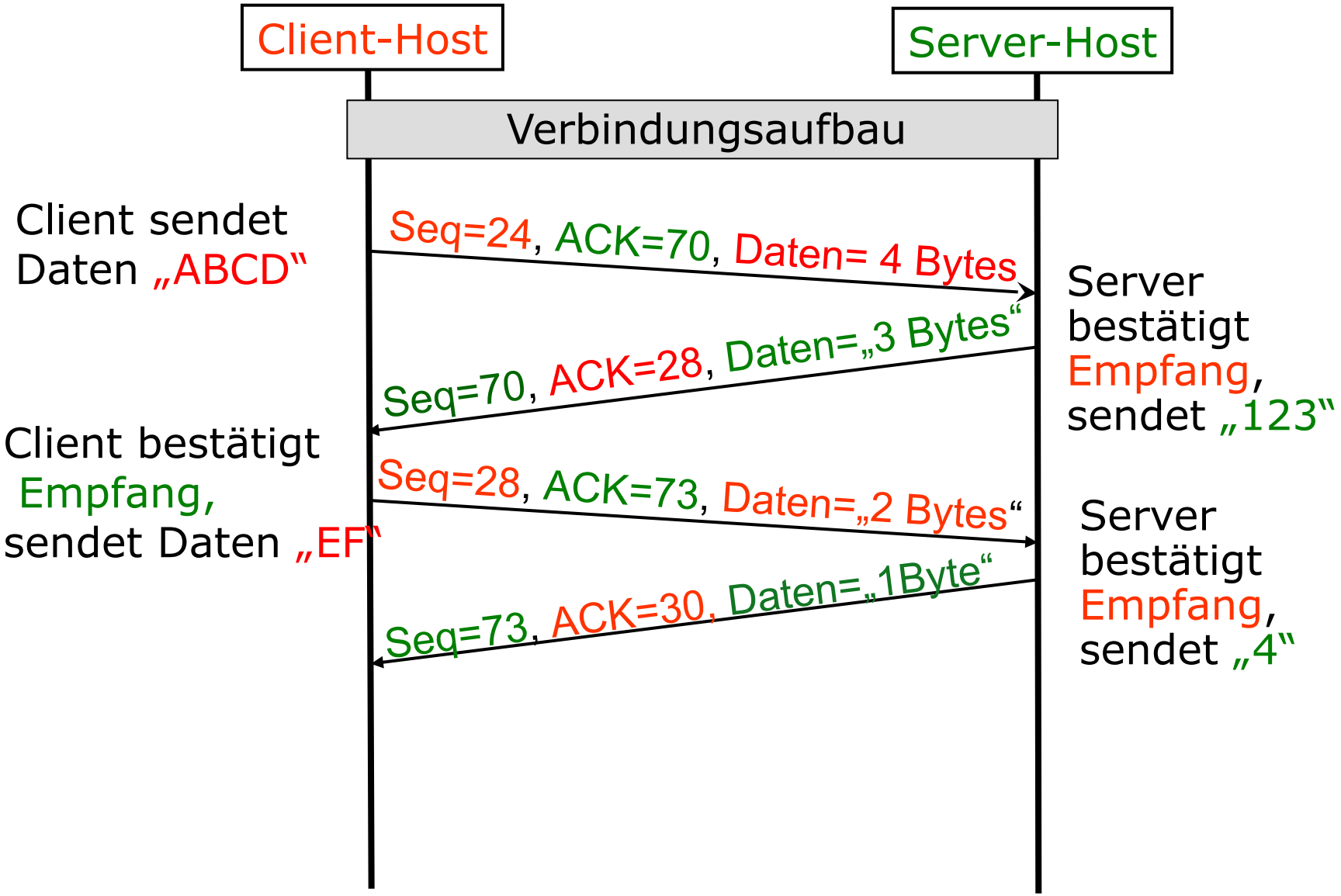


Acknowledgement Number (AckNo)

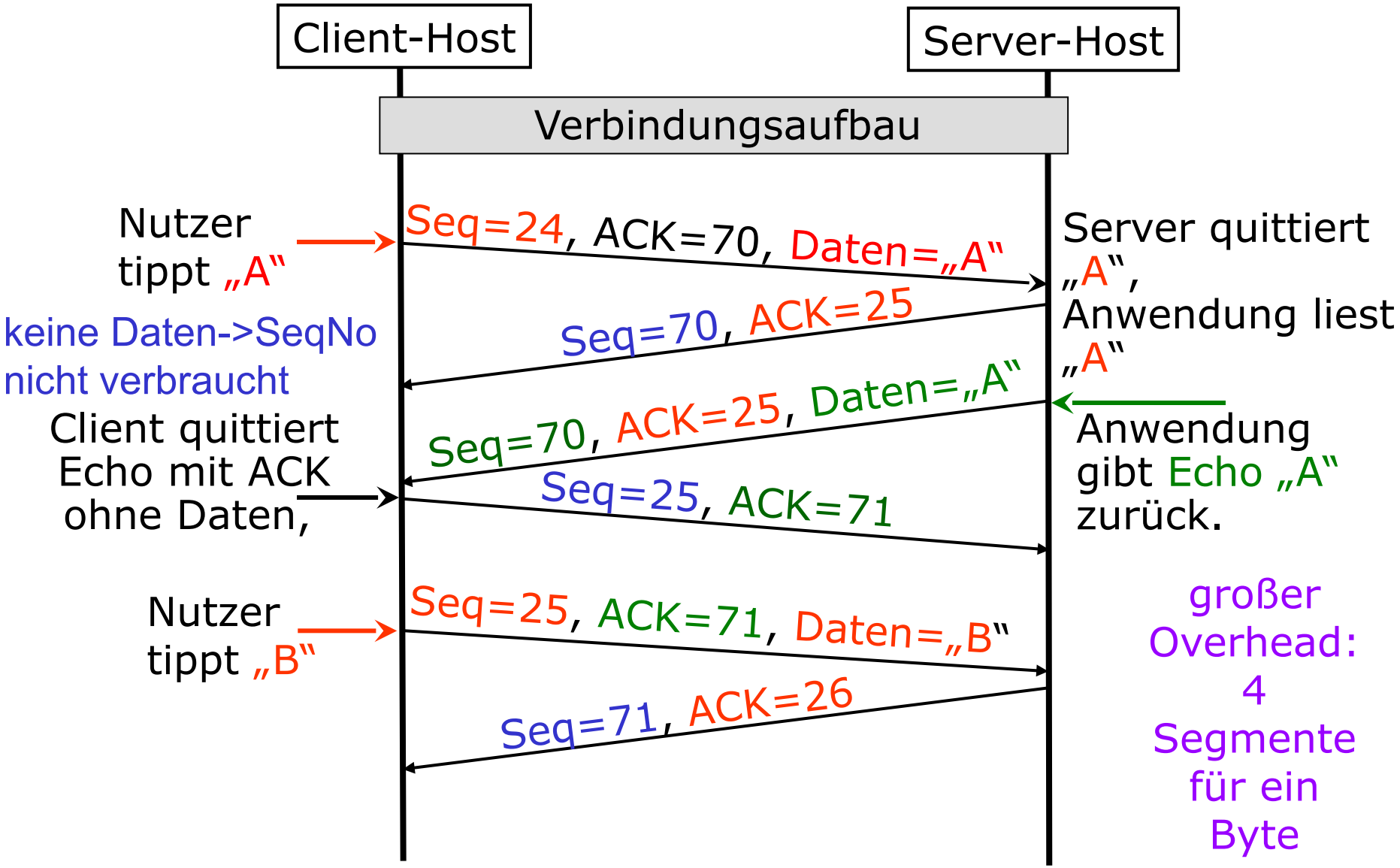
- Nutzt der TCP-Empfänger zum quittieren empfangener Daten
- Wird die AckNo (32 Bit) verwendet, muss das ACK-Flag gesetzt werden
- AckNo gibt die Sequenznummer des als nächstes erwarteten Bytes an.
- Ack quittiert alle SeqNo die kleiner als AckNo sind → kumulatives Ack
- Beispiel: TCP-Empfänger empfängt ein neues Segment



Beispiel: TCP SeqNo und AckNo



Beispiel: Telnet über TCP



Delayed Acknowledgement im Empfänger

Ziel:

Es sollen keine ACKs ohne Daten gesendet werden

Lösung:

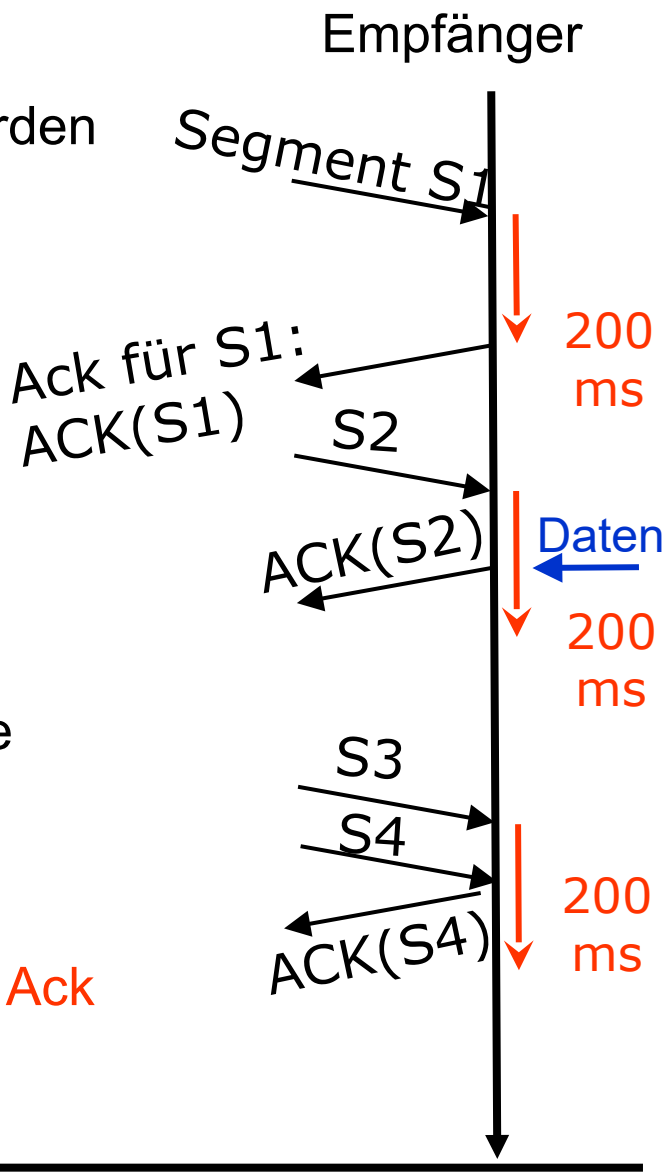
- ACKs können bis zu einem maximalen Wert (typisch < 200 ms) verzögert werden
- oder bis Daten zu senden sind.

Ausnahmen:

- Spätestens das Zweite in richtiger Reihenfolge empfangene Segment wird sofort quittiert.

➢ kumulatives ACK für S3 und S4

➢ generell: bei kumulativen ACKs kein Delayed Ack



Delayed Acknowledgements am Beispiel von Telnet