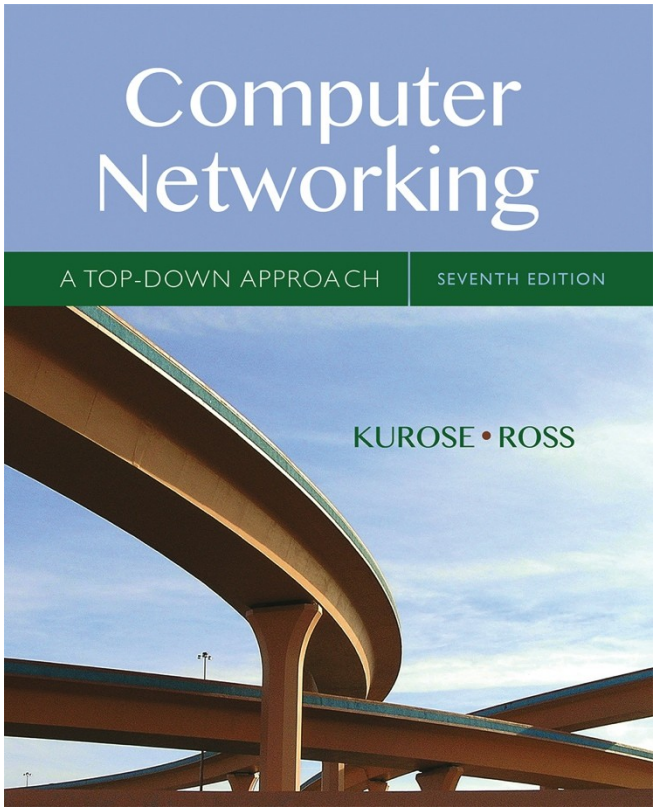
beachte auch: http://gaia.cs.umass.edu/kurose_ross

Chapter 3 Transport Layer



A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

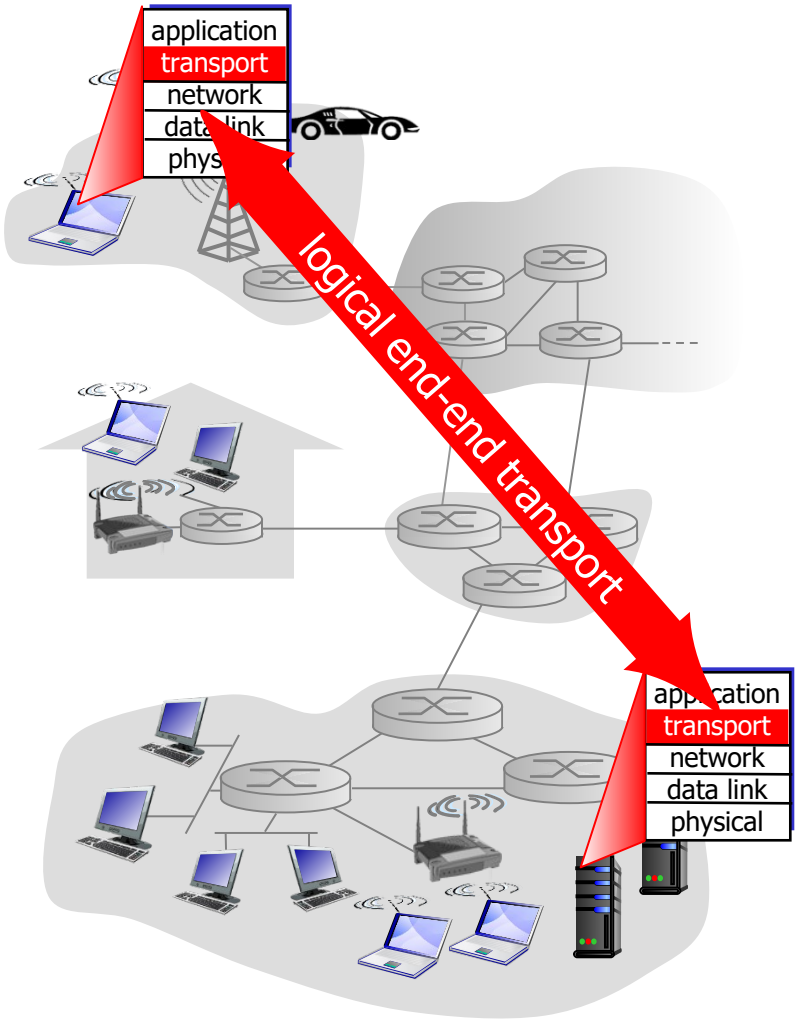
Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Transportschicht: Dienste und Protokolle

- Kommunikation zwischen *Prozessen* (Anwendungen, z.B. Webbrowser)
- Sender
  - teilt Nachricht in *Segmente*
  - übergibt diese an die Vermittlungsschicht
- Empfänger
  - erzeugt Nachricht aus *Segmenten*
  - übergibt an diese an die Anwendungsschicht
- mehr als ein Transportprotokoll verfügbar: TCP and UDP



Transport- vs. Vermittlungsschicht

- *Transportschicht:* Kommunikation zwischen Prozessen
  - nutzt Dienste der Vermittlungsschicht und baut auf diesen auf
- *Vermittlungsschicht:* Kommunikation zwischen Endgeräten

schlechte Nachricht:

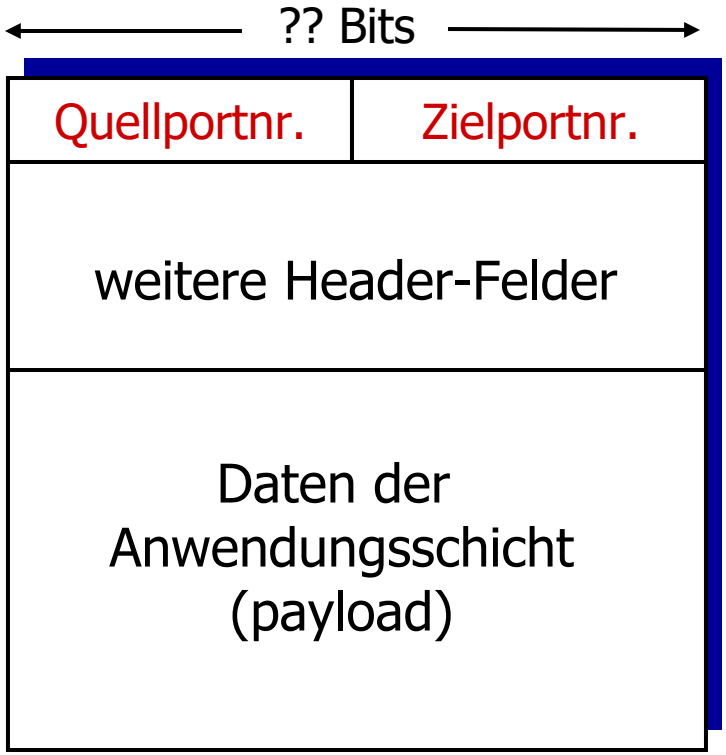
Das Internet (IP) tut sein Bestes ("best-effort"), aber es gibt keinerlei Garantien für eine schnelle und/oder vollständige Zustellung.

Protokolle der Transportschicht

- UDP
  - verbindungslos
  - unzuverlässig
  - (fast) ohne Schnickschnack
- TCP
  - verbindungsorientiert
  - zuverlässig
    - Daten kommen fehlerfrei und in der korrekten Reihenfolge an
  - mit Schnickschnack
    - Fluss- und Überlastkontrolle
    - Verbindungsmanagement

Portnummern: Adressen der Transportschicht

- Host empfängt IP-Paket (Vermittlungsschicht!)
  - jedes Paket hat
    - Quell-IP-Adresse
    - Ziel-IP-Adresse
    - und enthält ein Segment der Transportschicht
  - jedes Segment hat
    - Quell-Portnummer
    - Ziel-Portnummer

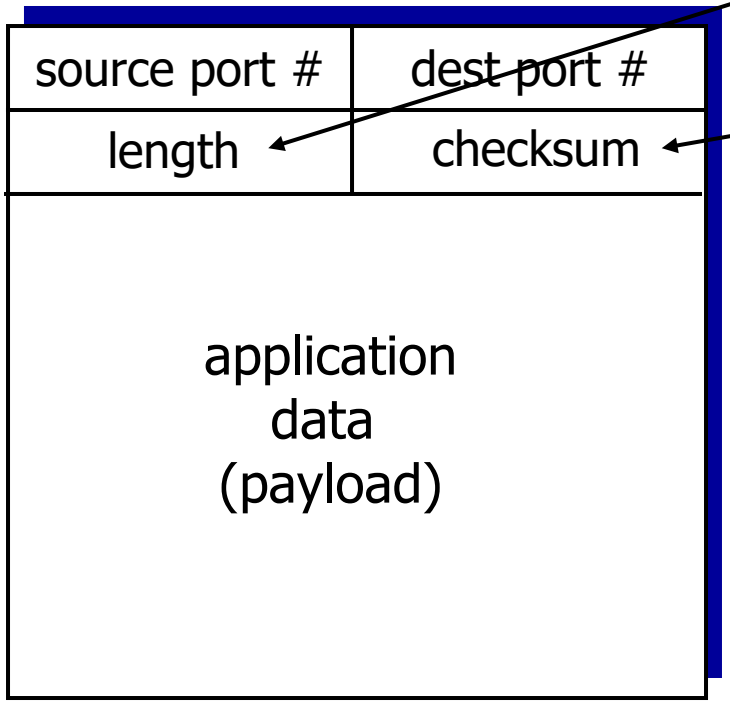


TCP/UDP Segment

Wireshark: UDP

https://gaia.cs.umass.edu/kurose_ross/wireshark.htm

UDP Segment



UDP segment format

Länge des UDP Segments in Bytes, inkl. Header

Prüfsumme zum Erkennen von Übertragungsfehlern

Warum UDP?

- kein Verbindungsaufbau (Verzögerung!)
- es ist (sehr) einfach
- kleiner Header für relativ viel Nutzlast
- keine Überlastkontrolle: UDP kann einfach drauflosfeuern, so schnell wie möglich

UDP: User Datagram Protocol [RFC 768]

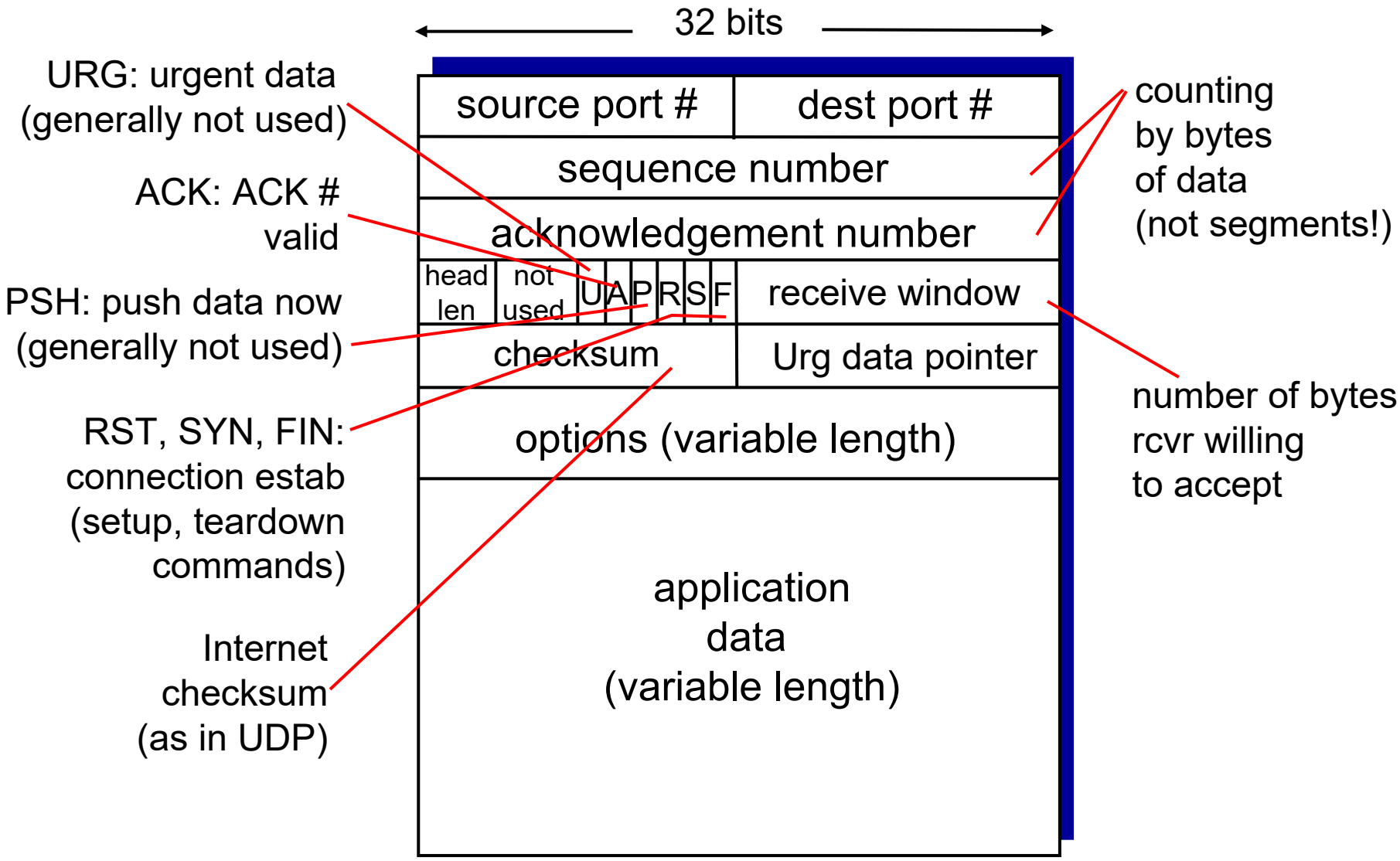
- kein Schnickschnack
- "best effort"-Dienst (wie IP)
- UDP Segment können
  - verloren gehen
  - out-of-order geliefert werden
- *verbindungslos*
  - kein Handshake (Verbindungsaufbau)
  - erzeuge Socket mit lokaler Quell-Portnummer
  - verschicke Daten mit Ziel-IP-Adresse und Ziel-Portnummer

- Verwendung
  - Multimedia Streaming
  - DNS
  - SNMP
  - …
- Zuverlässigkeit
  - ist Problem der Anwendungsschicht

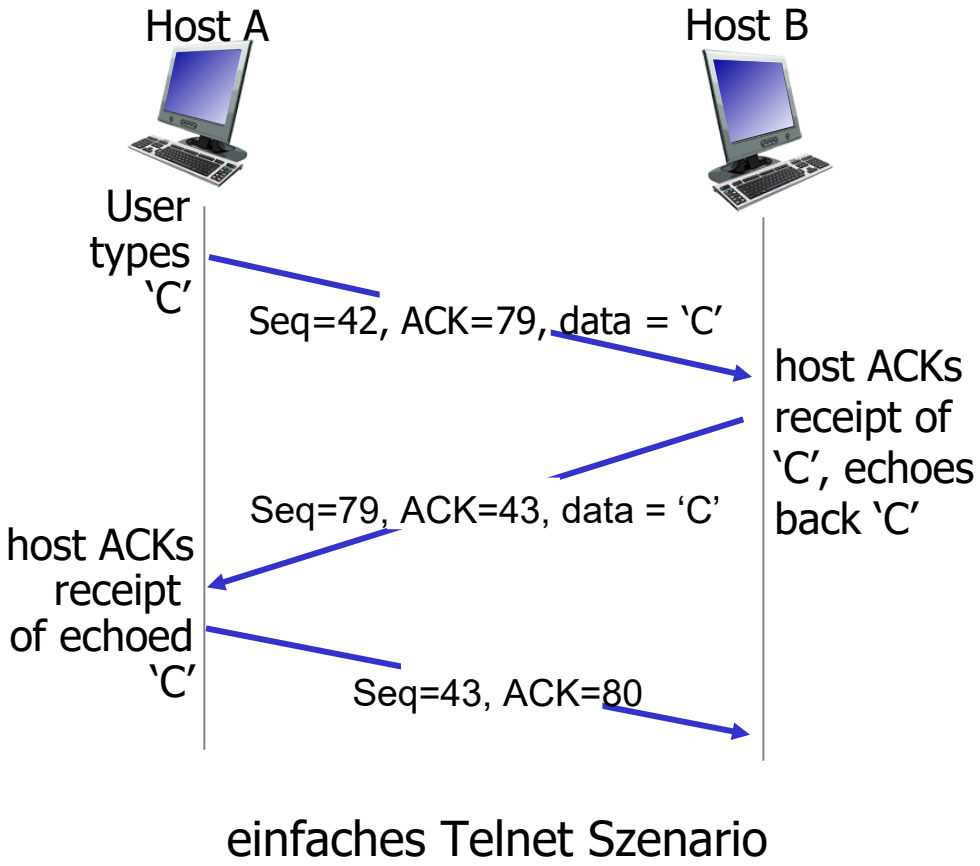
Wireshark: TCP

https://gaia.cs.umass.edu/kurose_ross/wireshark.htm

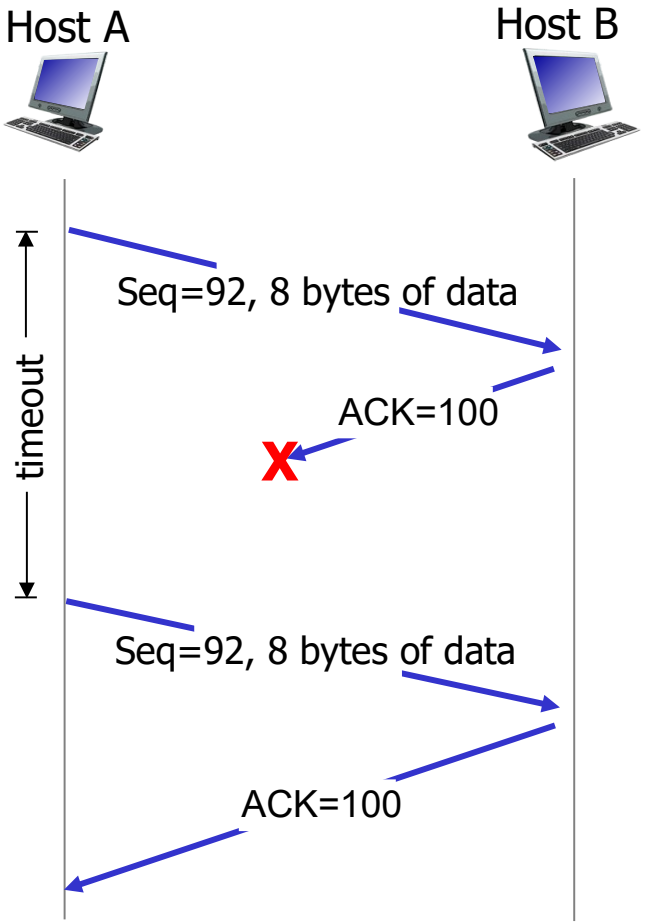
TCP Segment


32 bits
URG: urgent data
(generally not used)
ACK: ACK #
valid
PSH: push data now
(generally not used)
RST, SYN, FIN:
connection estab
(setup, teardown
commands)
Internet
checksum
(as in UDP)
source port #
dest port #
sequence number
acknowledgement number
head len
not used
U
A
P
R
S
F
receive window
checksum
Urg data pointer
options (variable length)
application
data
(variable length)
counting
by bytes
of data
(not segments!)
number of bytes
rcvr willing
to accept

TCP: Sequenznummern, ACKs



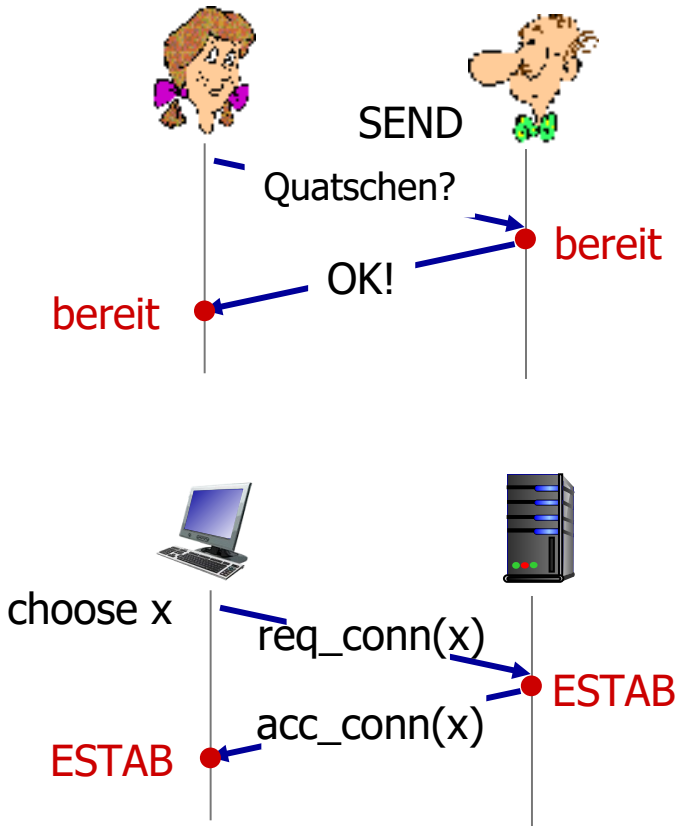
einfaches Telnet Szenario



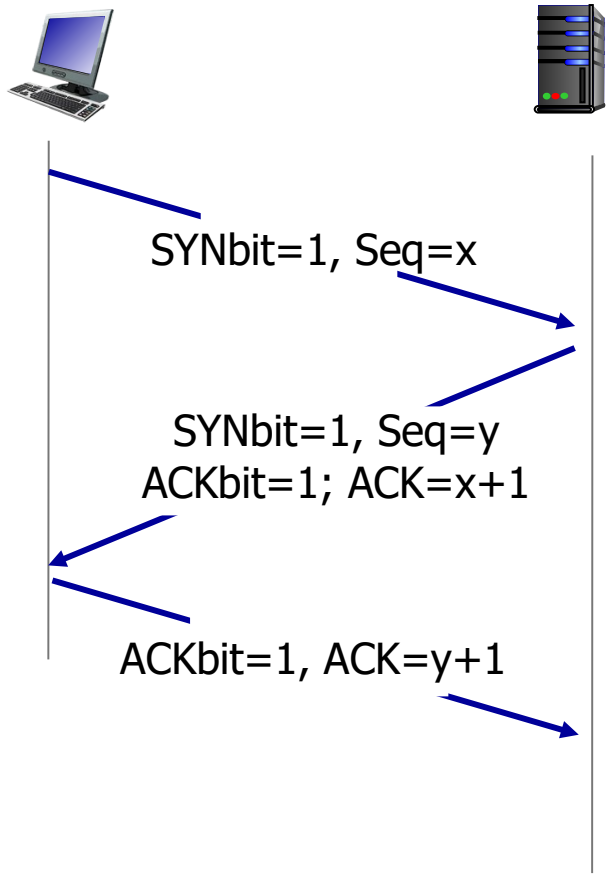
verlorenes ACK

Verbindungsaufbau

2-Wege-Handshake



TCP: 3-Wege-Handshake



TCP: Transmission Control Protocol

RFCs: 793,1122,1323, 2018, 2581

- mit Schnickschnack
- TCP Segmente erreichen ihr Ziel
  - fehlerfrei und in der richtigen Reihenfolge (Seq, ACK)
- verbindungsorientiert
  - 3-Wege-Handshake (SYNbit, ACKbit)
  - erzeuge Socket mit
    - Quell-IP-Adresse
    - Quell-Portnummer
    - Ziel-IP-Adresse
    - Ziel-portnummer

- Verwendung
  - Web
  - Email
  - Dateitransfer
  - etc.
  - etc.