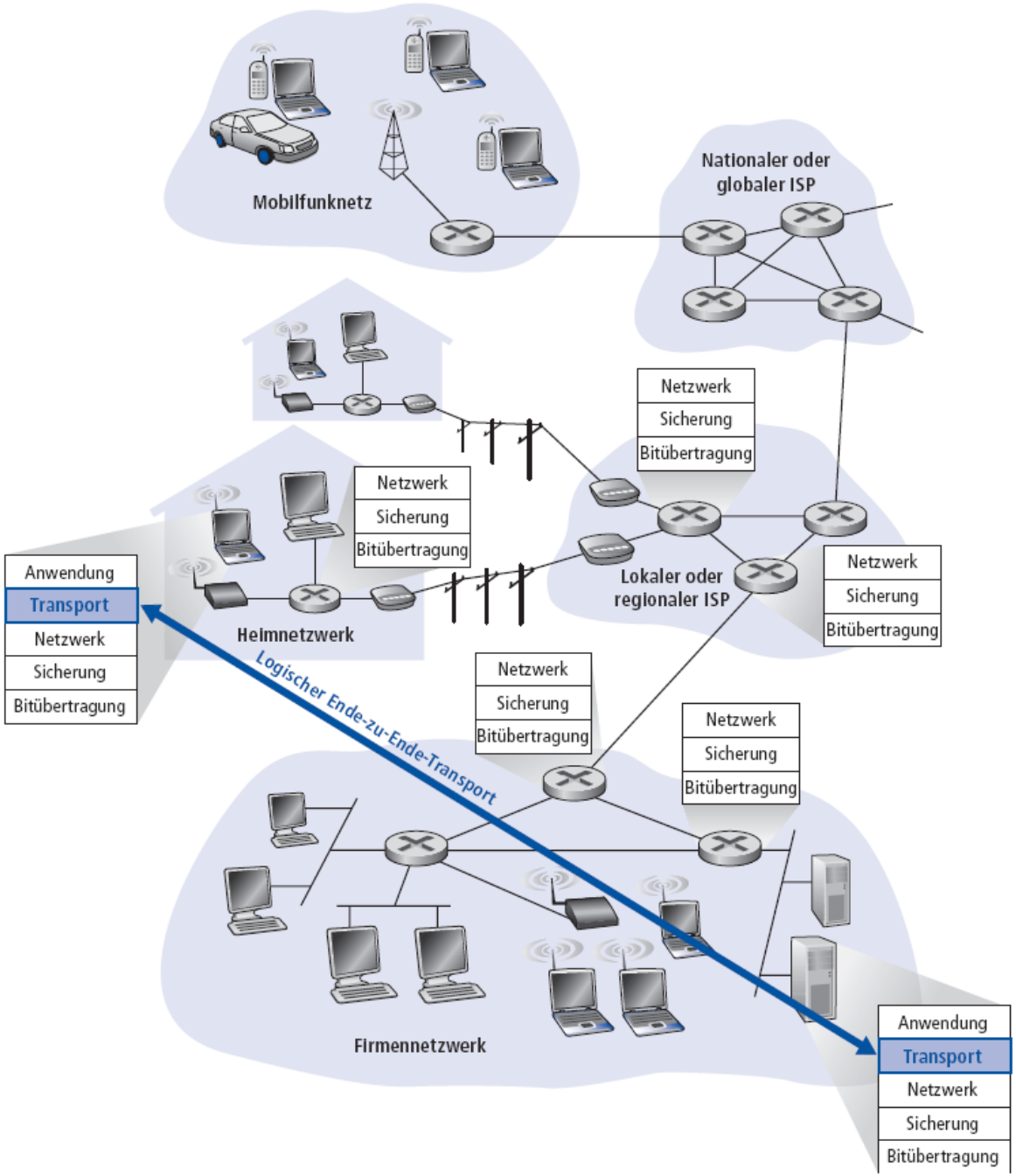
Protokollstapel TCP/IP

- Anwendungsschicht: Unterstützung von Netzwerkanwendungen
  - FTP, SMTP, HTTP
- Transportschicht: Datentransfer zwischen Prozessen
  - TCP, UDP
- Netzwerkschicht (auch Vermittlungsschicht): Weiterleiten der Daten von einem Sender zu einem Empfänger
  - IP, Routing-Protokolle
- Sicherungsschicht: Datentransfer zwischen benachbarten Netzwerksystemen
  - PPP, Ethernet
- Bitübertragungsschicht: Bits auf der Leitung

We reject kings, presidents and voting. We believe in rough consensus and running code (Dave Clark)

T-Shirt IETF

Transport-dienste und -protokolle



UDP: User Datagram Protocol [RFC 768]

- Minimales Internet-Transportprotokoll
- “Best-Effort”-Dienst, UDP-Segmente können:
  - verloren gehen
  - in der falschen Reihen-folge an die Anwendung ausgeliefert werden
- *Verbindungslos:*
  - kein Handshake zum Verbindungsaufbau
  - jedes UDP-Segment wird unabhängig von allen anderen behandelt

UDP-Prüfsumme

Ziel: Fehler im übertragenen Segment erkennen (z.B. verfälschte Bits)

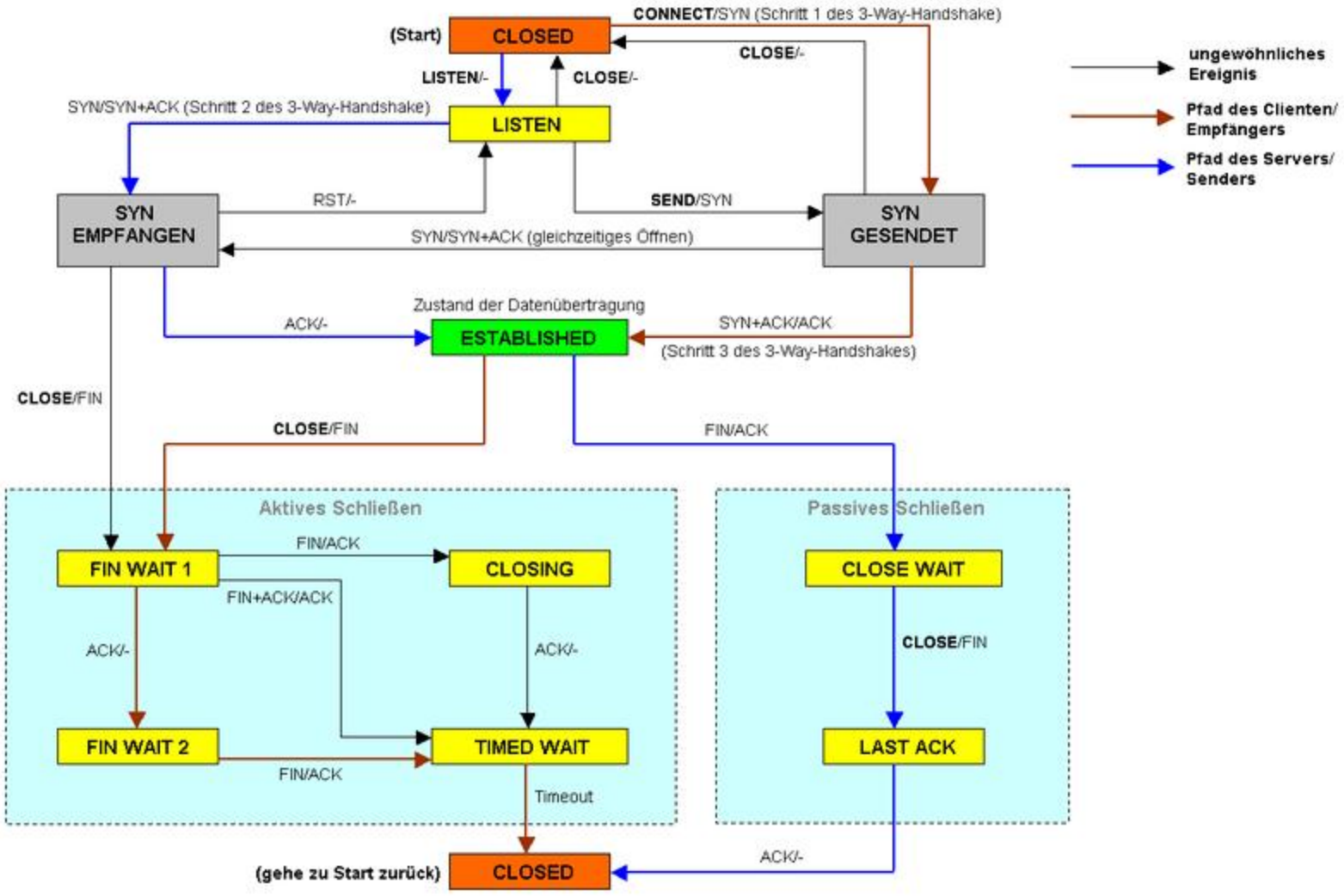
Sender:

- Betrachte Segment als Folge von 16-Bit-Integer-Werten
- Prüfsumme: Addition (1er-Komplement-Summe) dieser Werte
- Sender legt das invertierte Resultat im UDP-Prüfsummenfeld ab

Empfänger:

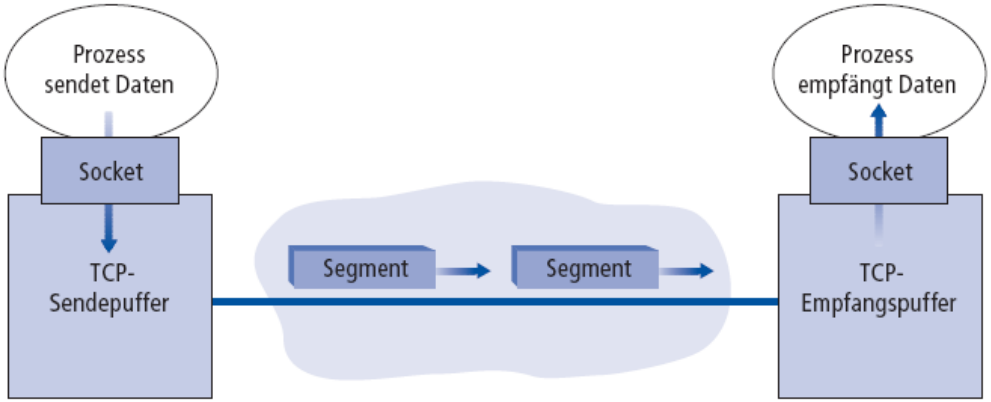
- Berechne die Prüfsumme des empfangenen Segments inkl. des Prüfsummenfeldes
- Sind im Resultat alle Bits 1?
  - NEIN – Fehler erkannt
  - JA – kein Fehler erkannt
    Vielleicht liegt trotzdem ein Fehler vor? Mehr dazu später
    …

TCP-State-Machine



TCP: Überblick RFCs: 793, 1122, 1323, 2018, 2581

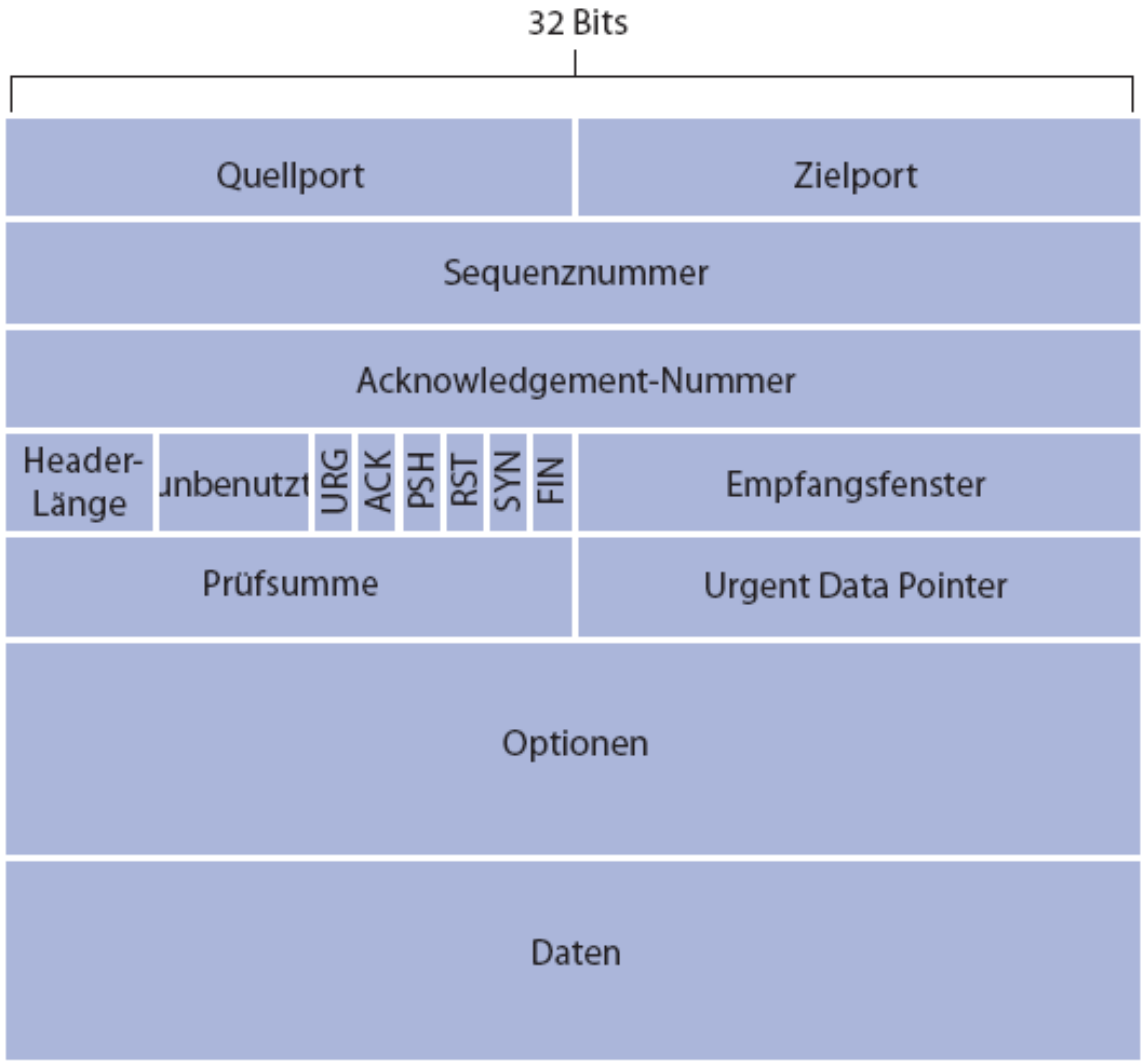
- Punkt-zu-Punkt:
  - Ein Sender, ein Empfänger
- Zuverlässiger, reihenfolgeerhaltender Byte-Strom:
  - Keine “Nachrichtengrenzen”
- Pipelining:
  - TCP-Überlast- und –Fluss-kontrolle verändern die Größe des Fensters
- *Sender- & Empfängerfenster*



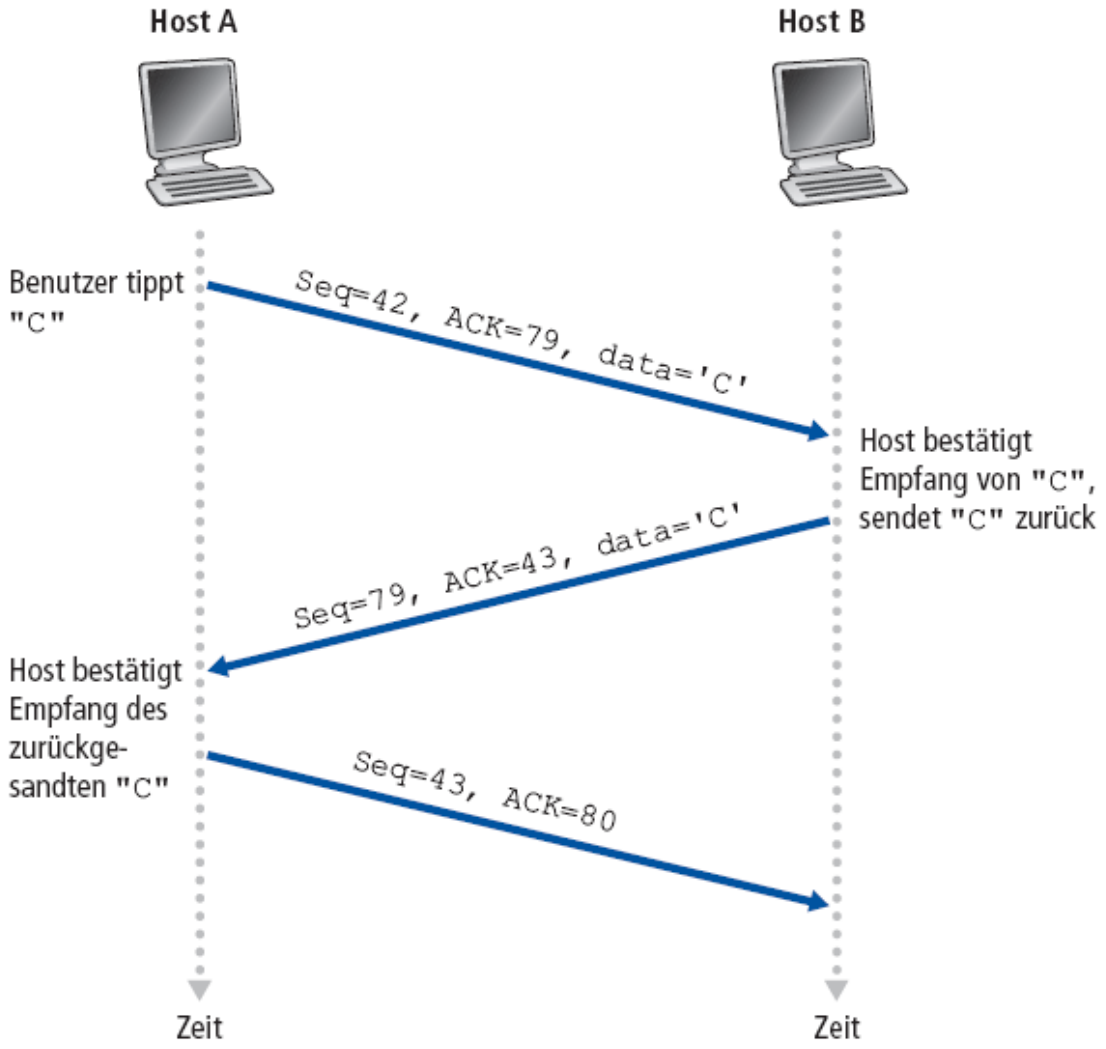
- Vollduplex:
  - Daten fließen in beide Richtungen
  - MSS: Maximum Segment Size
- Verbindungsorientiert:
  - Handshaking (Austausch von Kontrollnachrichten) initialisiert den Zustand im Sender und Empfänger, bevor Daten ausgetauscht werden
- Flusskontrolle:
  - Sender überfordert den Empfänger nicht

TCP-Segmentaufbau

Fakultät für Feinwerk- und Mikrotechnik, Physikalische Technik
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN · FH MÜNCHEN

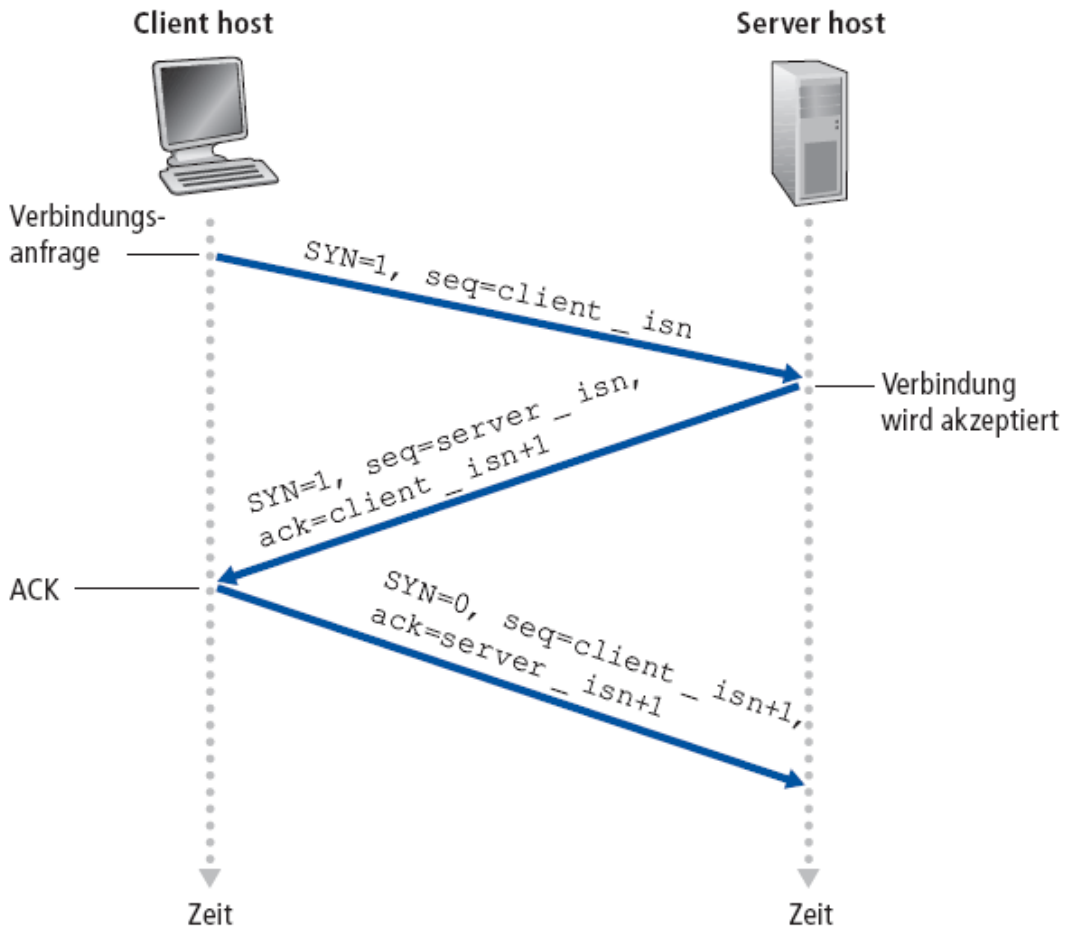


TCP-Sequenznummern und -ACKs



TCP: Drei-Wege-Handshake

Fakultät für Feinwerk- und Mikrotechnik, Physikalische Technik
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN · FH MÜNCHEN



TCP: zuverlässiger Datentransfer

- TCP stellt einen zuverlässigen Datentransfer über den unzuverlässigen Datentransfer von IP zur Verfügung
- Pipelining von Segmenten
- Kumulative ACKs
- TCP verwendet einen einzigen Timer für Übertragungswieder-holungen
- Übertragungswieder-holungen werden ausgelöst durch:
  - Timeout
  - Doppelte ACKs

TCP-Sender (vereinfacht)

```
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum

loop (forever) {
  switch(event)

  event: Daten von der Anwendung
      erzeuge TCP Segment mit Sequenznummer NextSeqNum
      if (Timer läuft nicht)
          starte Timer
      gib Segment an IP weiter
      NextSeqNum = NextSeqNum + length(data)

  event: Timeout
      übertrage das unbestätigte Segment mit der kleinsten
          Sequenznummer
      starte Timer

  event: ACK empfangen für Sequenznummer y
      if (y > SendBase) {
          SendBase = y
          if (es gibt noch unbestätigte Segmente)
              starte Timer
          }

} /* end of loop forever */
```

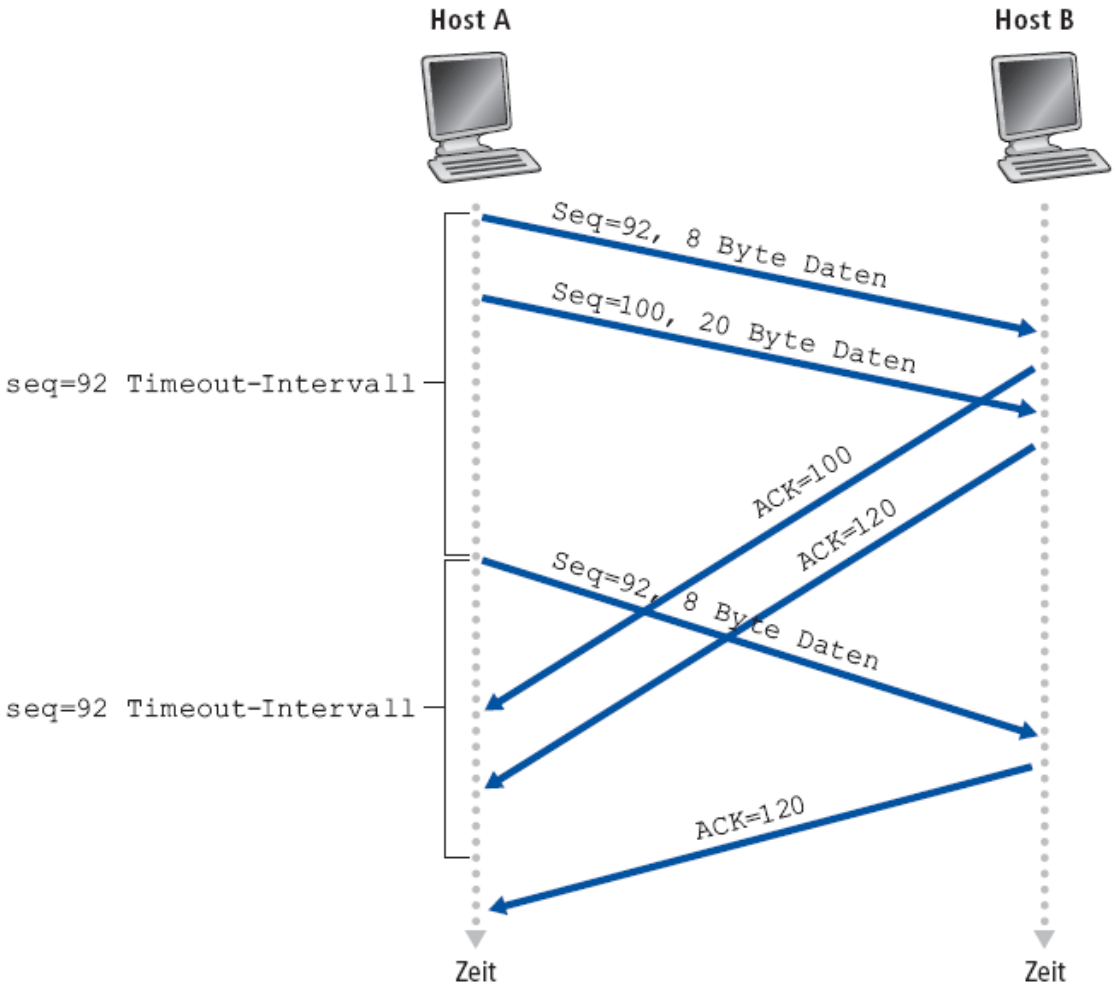
Anmerkungen:

- SendBase-1: das letzte erfolgreich bestätigte Byte

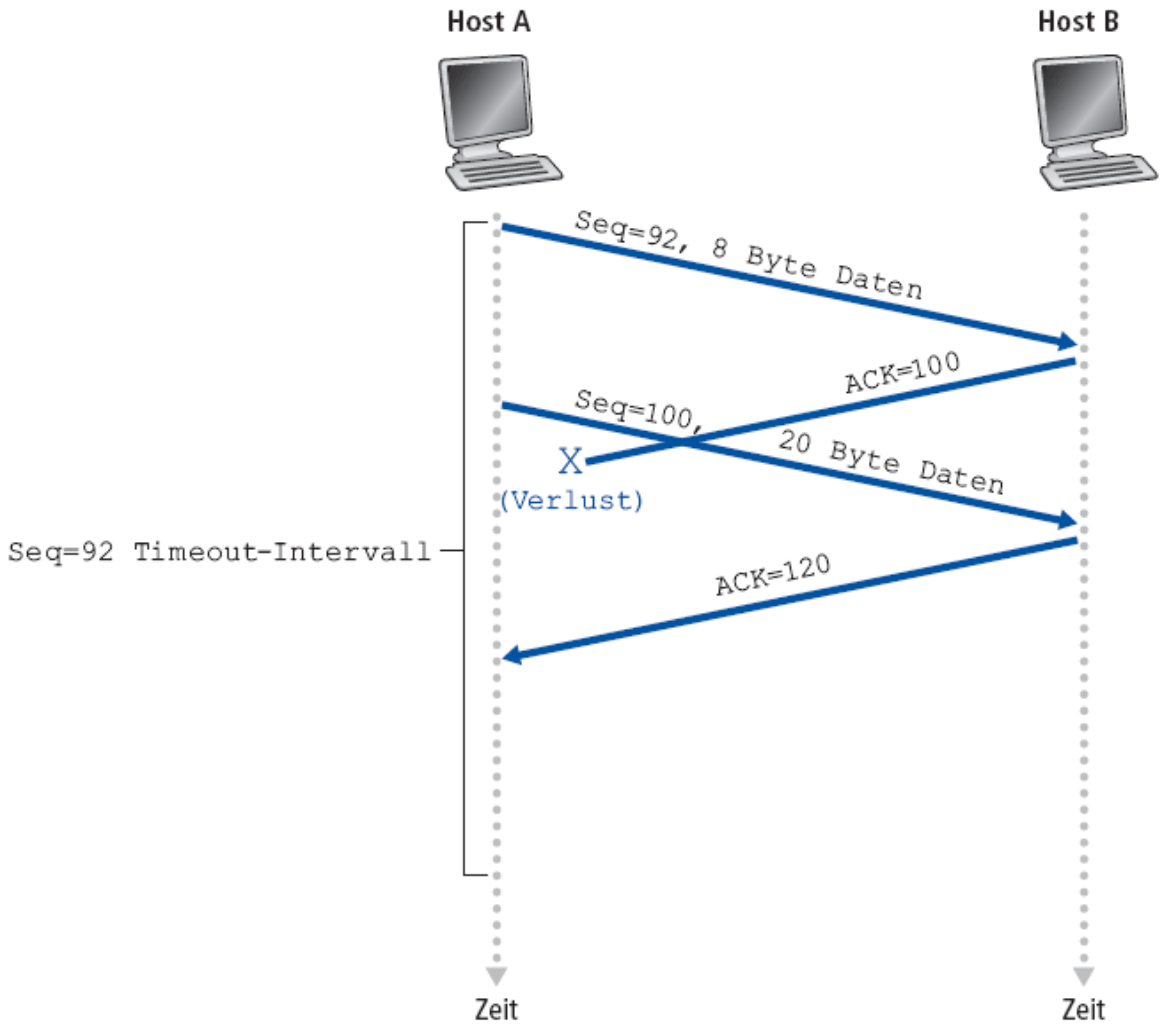
Beispiel:

- SendBase-1 = 71; y= 73, d.h. Empfänger erwartet 73+ ; y > SendBase, d.h. das ACK ist neu

TCP: Beispiele für Übertragungs-wiederholungen – verfrühter Timeout



TCP: Beispiele für Übertragungs-wiederholungen – kumulative ACKs



TCP-ACK-Erzeugung [RFC 1122, RFC 2581]

Ereignis	Aktion des TCP-Empfängers
Ankunft des Segmentes in der richtigen Reihenfolge mit der erwarteten Sequenznummer. Alle Daten bis zur erwarteten Sequenznummer sind bereits bestätigt.	Verzögertes ACK. Wartet bis zu 500 ms auf die Ankunft eines anderen Segmentes in richtiger Reihenfolge. Wenn das nächste Segment nicht in diesem Zeitintervall eintrifft, wird ein ACK gesendet.
Ankunft eines Segmentes in der richtigen Reihenfolge mit erwarteter Sequenznummer. Ein anderes Segment in der korrekten Reihenfolge wartet auf die ACK-Übertragung.	Sendet sofort ein einzelnes kumulatives ACK, bestätigt beide in richtiger Reihenfolge eingetroffene Segmente.
Ankunft eines Segmentes außerhalb der Reihenfolge mit einer Sequenznummer, die größer ist als erwartet. Lücke im Bytestrom aufgetreten.	Sendet sofort ein doppeltes ACK, in dem er die Sequenznummer des nächsten erwarteten Bytes angibt.
Ankunft eines Segmentes, das die Lücke in den erhaltenen Daten ganz oder teilweise ausfüllt.	Sendet sofort ein ACK, vorausgesetzt, das Segment beginnt mit der Sequenznummer des nächsten erwarteten Bytes. Bestätigt alle nun lückenlos vorliegenden Bytes.

Fast Retransmit

- Zeit für Timeout ist häufig sehr lang:
  - Große Verzögerung vor einer Neuübertragung
- Erkennen von Paketverlusten durch doppelte ACKs:
  - Sender schickt häufig viele Segmente direkt hintereinander
  - Wenn ein Segment verloren geht, führt dies zu vielen doppelten ACKs

- Wenn der Sender 3 Duplikate eines ACK erhält, dann nimmt er an, dass das Segment verloren gegangen ist:
  - Fast Retransmit (schnelle Sende-wiederholung): Segment erneut schicken, bevor der Timer ausläuft

TCP-Verbindungsmanagement

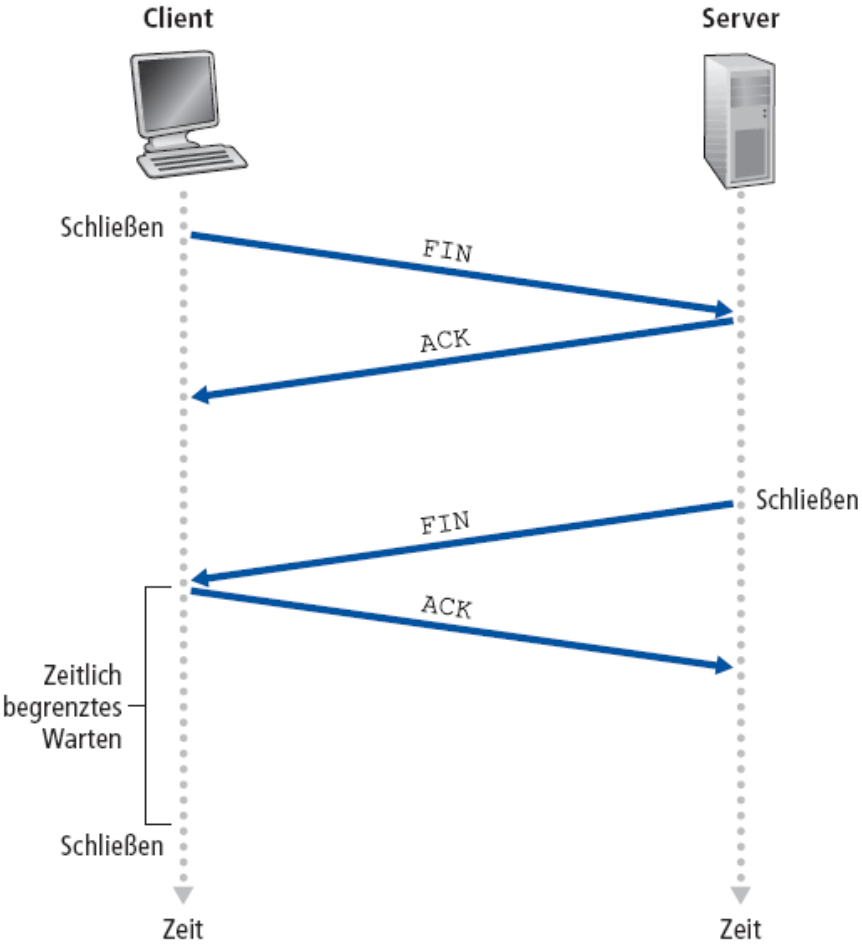
Schließen einer Verbindung

Client schließt socket:

```
clientSocket.close();
```

Schritt 1: Client sendet ein TCP-FIN-Segment an den Server

Schritt 2: Server empfängt FIN, antwortet mit ACK; dann sendet er ein FIN (kann im gleichen Segment erfolgen)



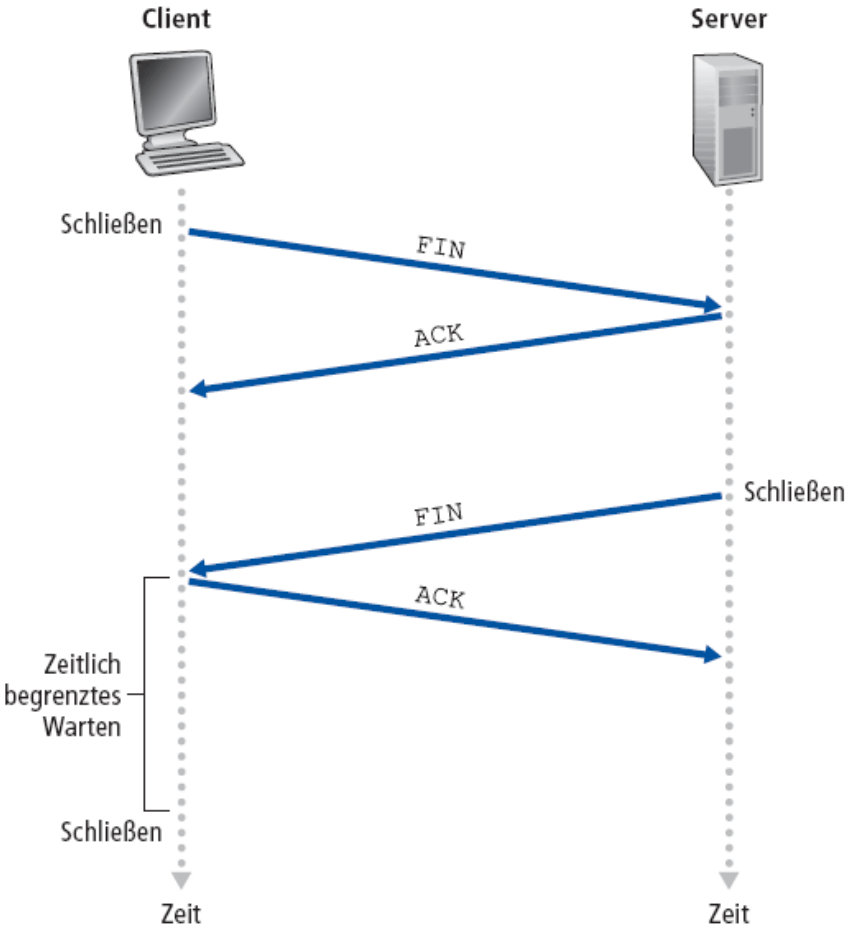
TCP-Verbindungsmanagement

Schritt 3: Client empfängt FIN und antwortet mit ACK

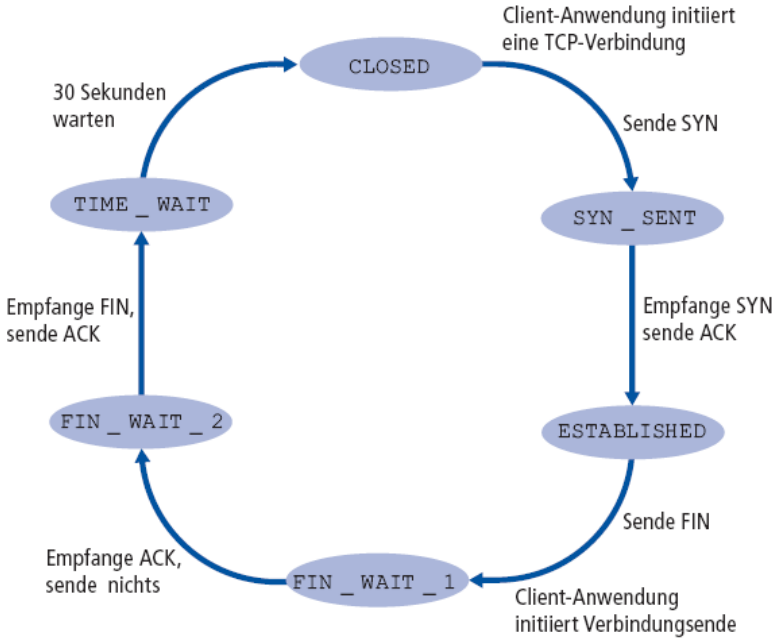
- Beginnt einen "Timed- Wait"-Zustand – er antwortet auf Sende-wiederholungen des Servers mit ACK

Schritt 4: Server, empfängt ACK und schließt Verbindung

Anmerkung: Mit kleinen Änderungen können so auch gleichzeitig abgeschickte FINs behandelt werden



TCP-Verbindungsmanagement



TCP-Client-Lebenszyklus

TCP-Server-Lebenszyklus