Chapter 6

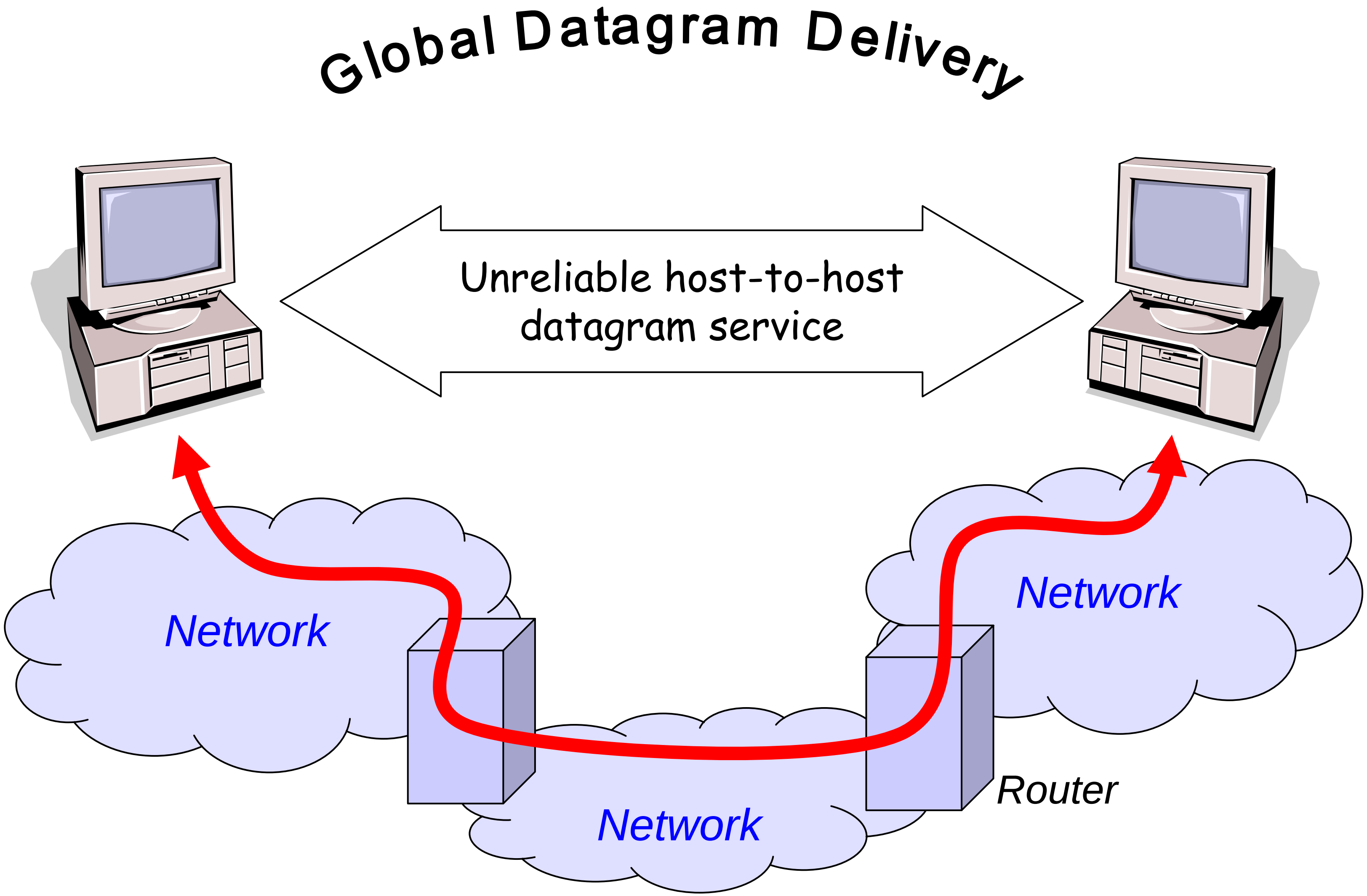
Advanced computer networking

Internet Protocols

Thomas Fuhrmann

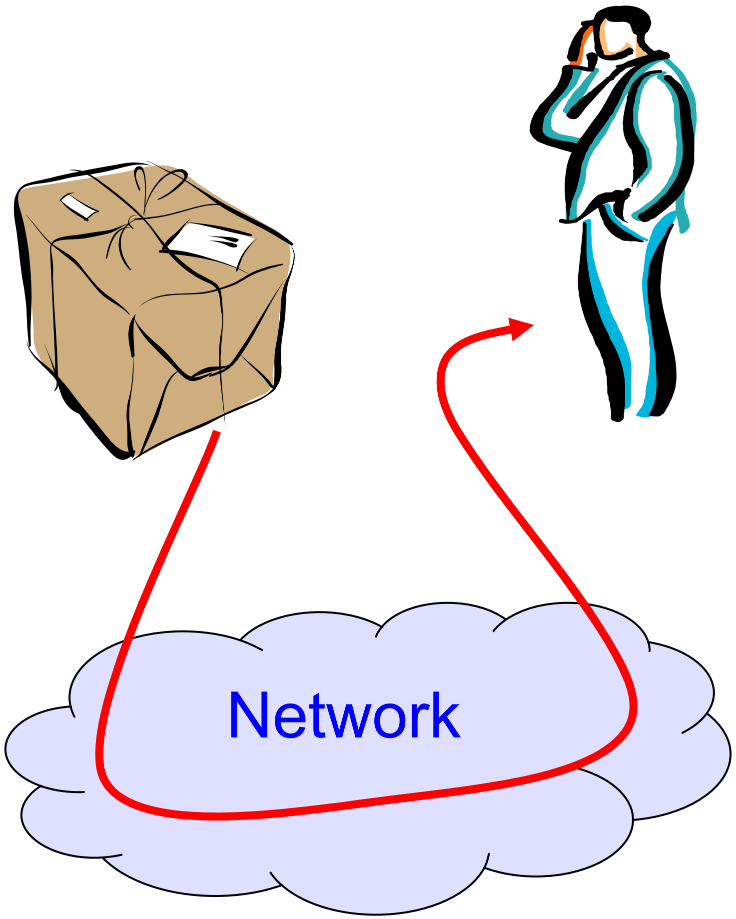
Network Architectures
Computer Science Department
Technical University Munich

The Network Layer – Summary

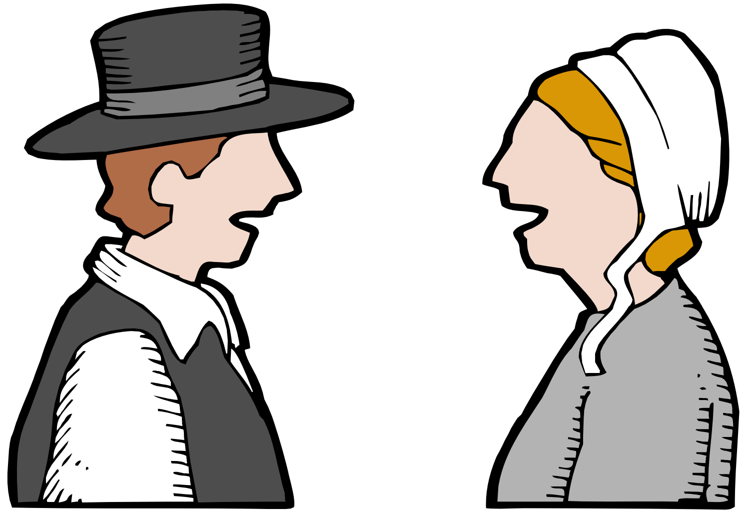


The Transport Layer – Overview

Connection less

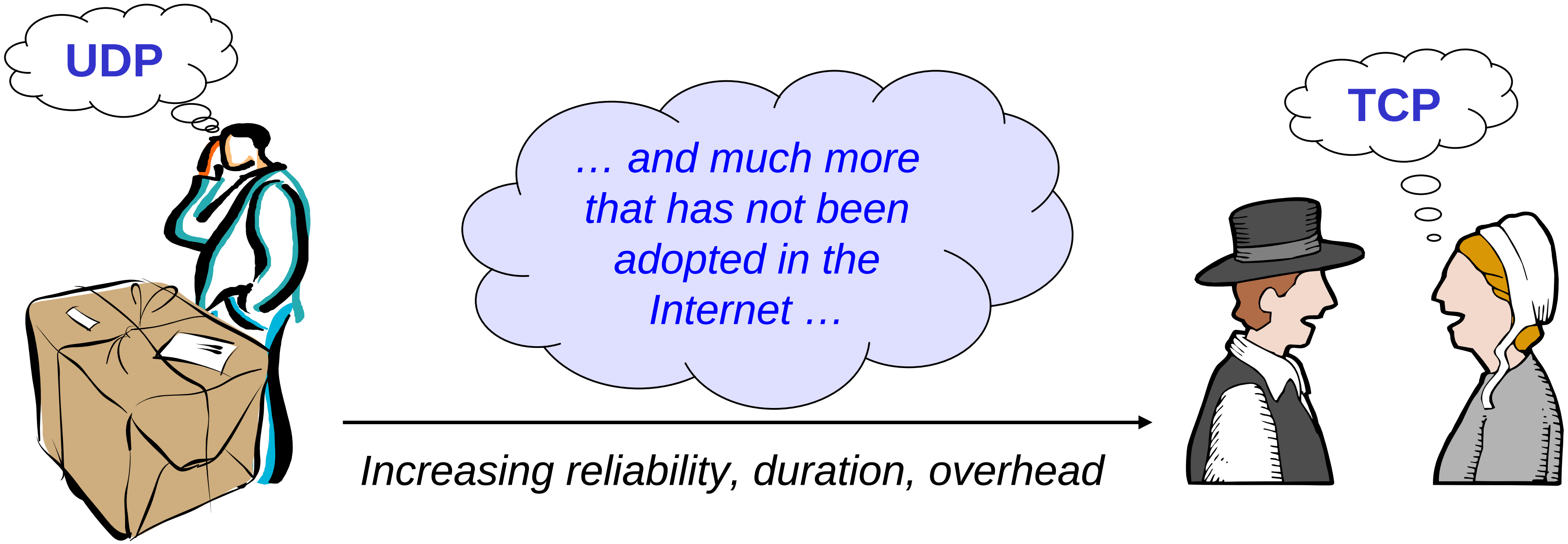


Connection oriented



- Connection with non-zero duration
- Bidirectional
- Data stream with rate control
- Reliable

Transport Protocols: UDP & TCP



The Internet's two transport protocols

- Unreliable, connection less datagram service (UDP), and
- Reliable, connection oriented, data stream service (TCP)

are only two extremes. Many more transport semantics are possible.

However, in the Internet, only these extremes are widely used.

Further Transport Protocol Semantics

UDP reflects the unreliable service of IP. Other connection less semantics could be useful:

- *Maybe:* Best effort service as provided by IP. Messages may be lost in the network without further notice.
- *At-Least-Once:* The transport layer repeats the message as long as its reception has not been confirmed. This may result in duplicate deliveries. (Note that the implementation typically uses timeouts.)
- *At-Most-Once:* The transport layer keeps track of duplicates, so that each message is received at most once. This book-keeping does not guarantee reception. (Note that the implementation typically uses unique identifiers such as sequence numbers.)
- *Exactly-Once:* Combination of ‘at least once’ and ‘at most once’: The reception of exactly one copy of each message is guaranteed if the network is operational at all.

These semantics have to be considered in distributed systems (cf. respective lecture).

Aufgaben der Transportschicht

Gesicherte Übertragung von Datenströmen über ein ungesichertes paketvermitteltes Netz erfordert:

- Erkennen verschiedener Verbindungen durch spezifische Kennungen (***Multiplexen***)

TCP & UDP

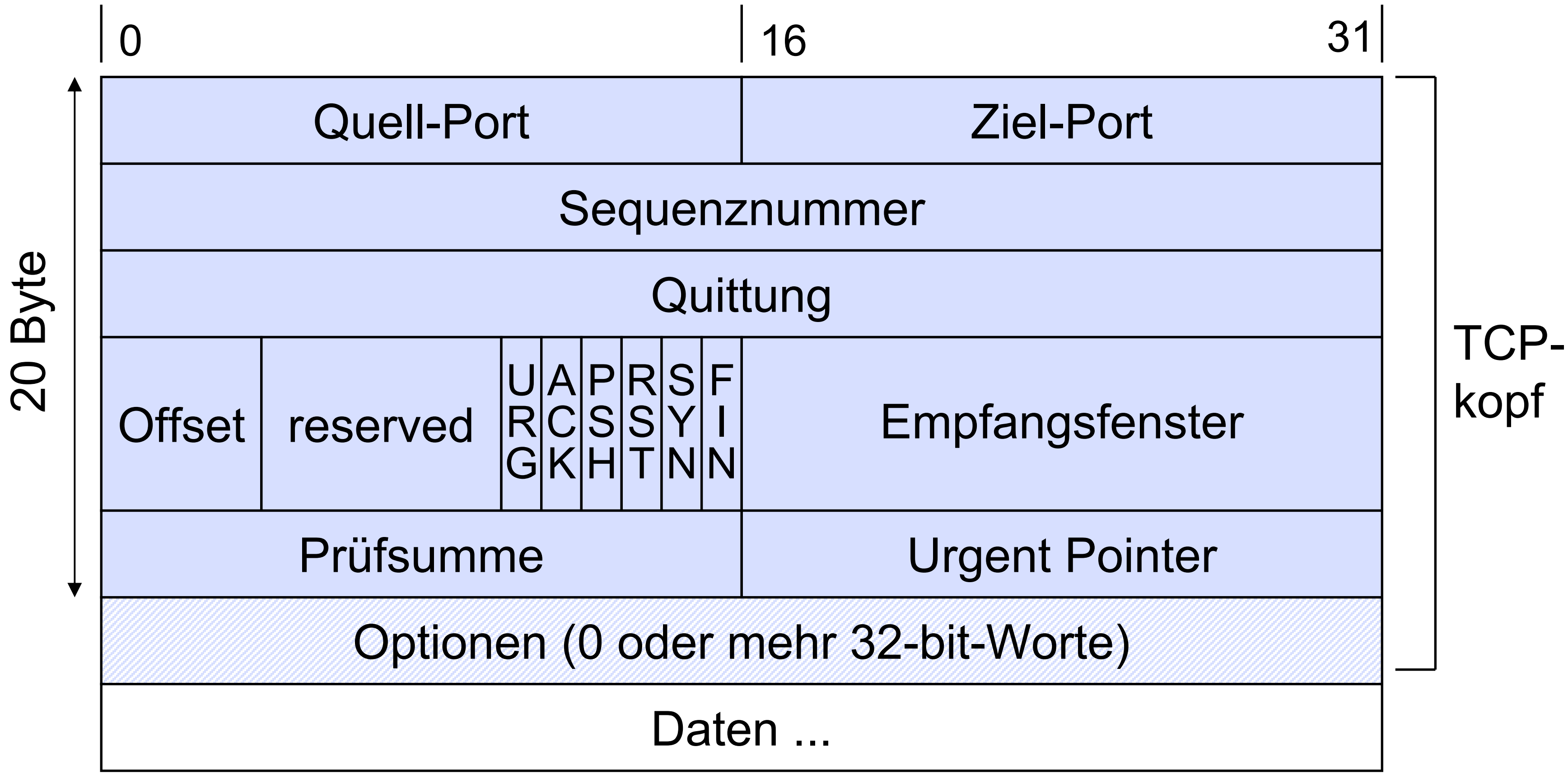
- Datenströme werden in Pakete unterteilt (***Segmentierung***)
- Pakete werden wiederholt, falls sie im Netz verloren gegangen sind (***Übertragungswiederholung***). Dazu müssen entweder empfangene Pakete bestätigt werden oder fehlende angemahnt werden.
- Senderate wird an Leistungsfähigkeit der Empfänger angepasst (***Flusskontrolle***)
- Senderate wird an Kapazität des Netzwerks angepasst (***Staukontrolle***)

Nur TCP

auch verbindungslos

verbindungsorientiert

TCP Header


0
16
31
20 Byte
Quell-Port
Ziel-Port
Sequenznummer
Quittung
Offset
reserved
URG
ACK
PSH
RST
SYN
FIN
Empfangsfenster
Prüfsumme
Urgent Pointer
Optionen (0 oder mehr 32-bit-Worte)
Daten ...
TCP-
kopf

TCP Header

- ***Quell- und Ziel-Port*** identifizieren die Endpunkte der TCP-Verbindung (→ Multiplexing).
- ***Sequenznummer*** (gemessen in Byte) zur vollständigen, reihenfolgetreuen Auslieferung des Datenstroms (siehe unten).
- ***Quittung***: Die nächste vom Empfänger erwartete Sequenznummer.
- ***Offset***: Anzahl der 32-Bit-Wörter im TCP-Kopf. Ermöglicht variable Anzahl von Optionen im TCP-Kopf.
- ***Empfangsfenster*** (für die Flusskontrolle, siehe unten): Anzahl der Bytes, die nach dem höchsten bestätigten Byte gesendet werden dürfen.
- ***Prüfsumme*** über TCP-Kopf und Daten.
- ***Urgent-Zeiger*** zeigt (relativ zum Anfang dieses Segments) auf „wichtige" Daten, die die Anwendung bevorzugt bearbeiten soll.
  - Beispiel: Interaktive Sitzung, bei der der Anwender eine große Datenmenge abgeschickt hat, deren Bearbeitung länger dauert, die aber jetzt abgebrochen werden soll.
- Flags:
  - ***URG*** wird auf 1 gesetzt, falls der Urgent Pointer verwendet wird.
  - ***SYN*** wird beim Verbindungsaufbau verwendet (siehe unten).
  - ***ACK*** signalisiert die Gültigkeit des Acknowledgement-Feldes bzw. unterscheidet bei gesetztem SYN-Bit einen Connection Request vom Connection Confirm.
  - ***FIN*** gibt an, dass der Sender keine Daten mehr senden möchte.
  - ***RST*** wird im Fehlerfall benutzt, um eine Verbindung zurückzusetzen.
  - ***PSH*** signalisiert, dass die übergebenen Daten sofort weitergeleitet werden sollen. Gilt sowohl für den Sender als auch für den Empfänger. (Wird meist sowieso immer gesetzt und deshalb von den typ. Implementierungen nicht beachtet.)

UDP Header

8 Byte

0	16	31
Quell-Port	Ziel-Port	
Länge	Prüfsumme	
Daten ...		

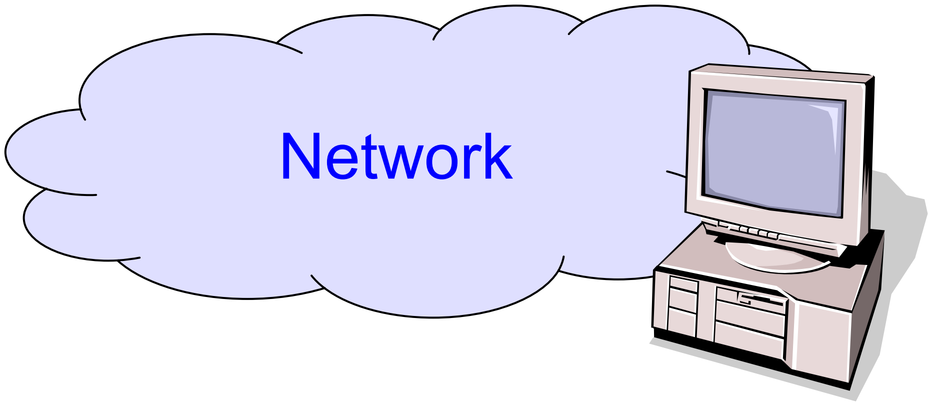
- Prüfsumme sichert UDP-Kopf und UDP-Nutzdaten, sowie Teile des IP-Kopfs (Adressen, Protokolltyp und Länge)
- Prüfsumme kann Null gesetzt werden, d.h. sie muss nicht berechnet werden
- UDP-Längenfeld ist unnötig*, die Länge ist ja durch das IP-Paket vorgegeben!

** Typisches Beispiel von schlampigem Protokollentwurf*

Connections …

… Set-Up & Tear Down

Multiplexing

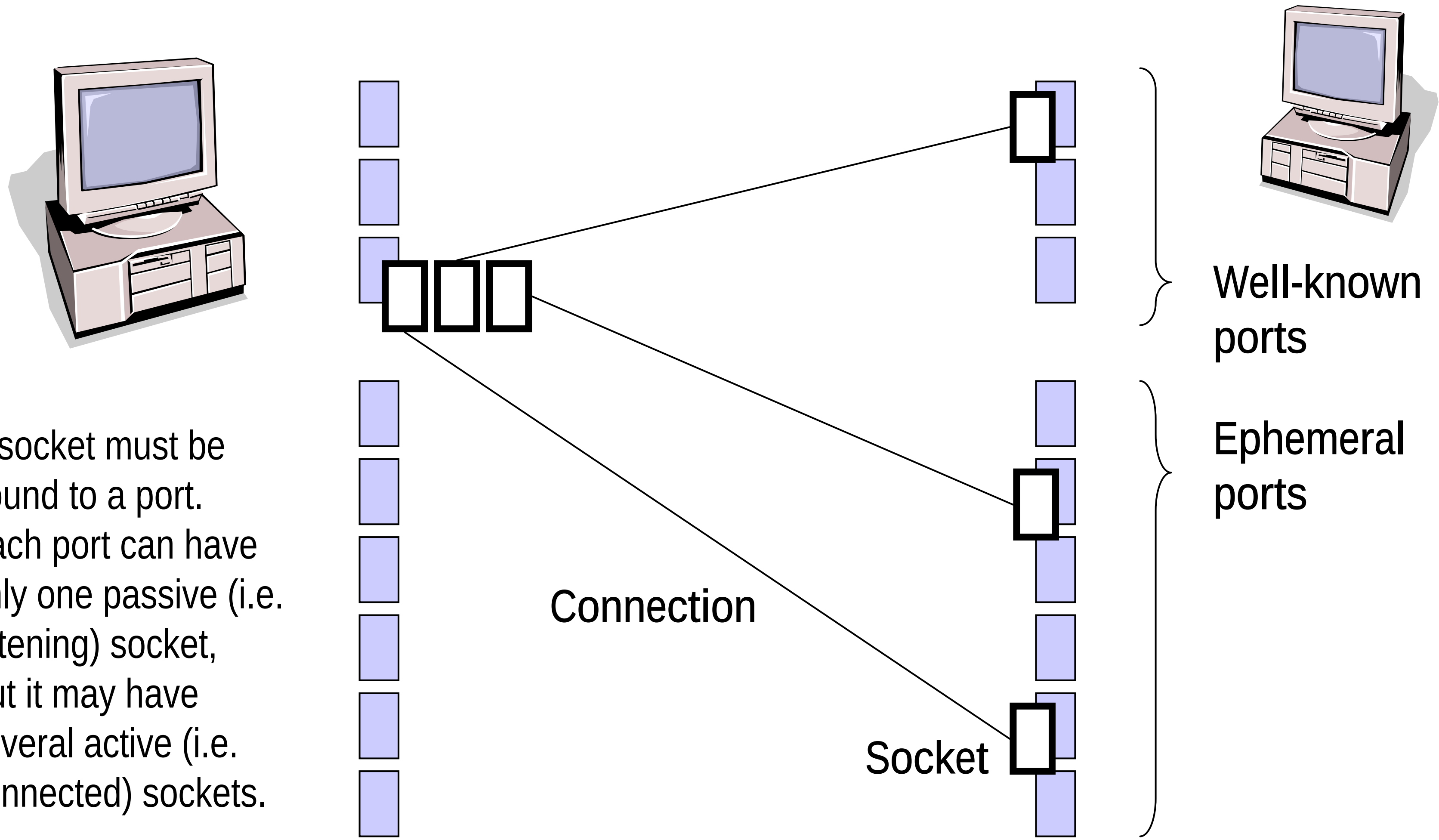


What is the communication end point?

- Not the host, because one machine may run multiple applications.
- Not the application process, because one application may use multiple communication subsystems.

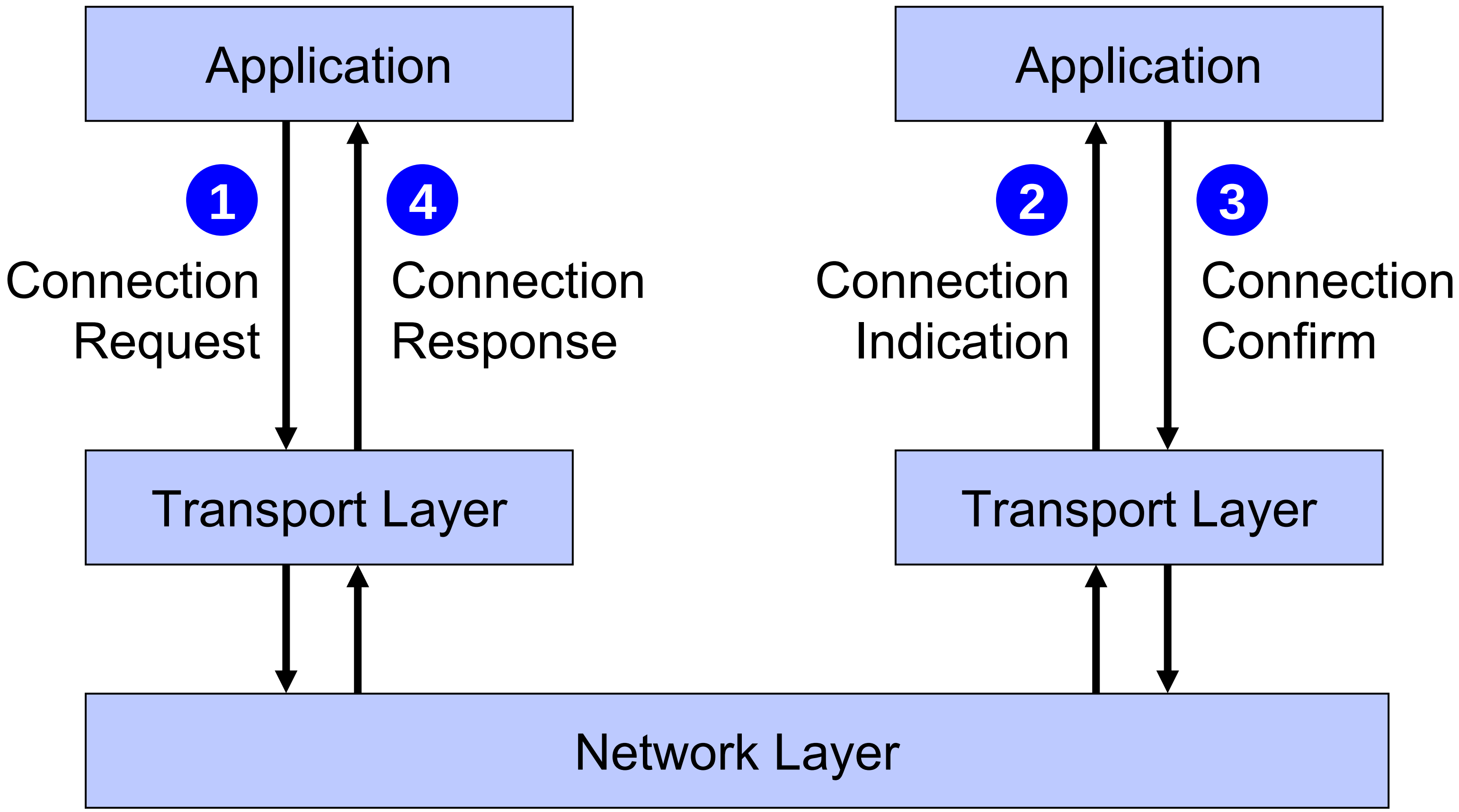
→ ***Socket*** as end point abstraction

What‘s a TCP connection?

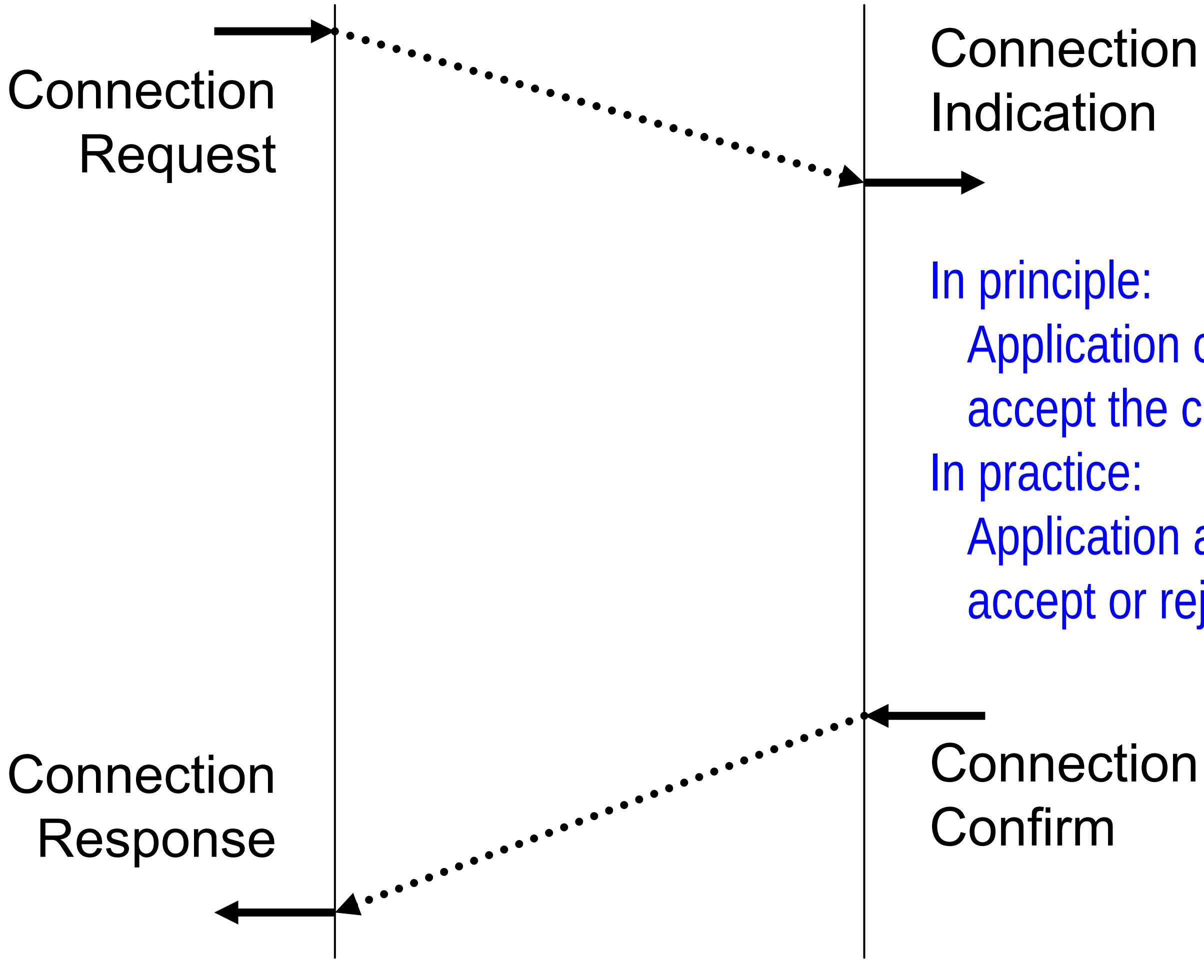


- A socket must be bound to a port.
- Each port can have only one passive (i.e. listening) socket,
- But it may have several active (i.e. connected) sockets.

Connection Set-up (1)



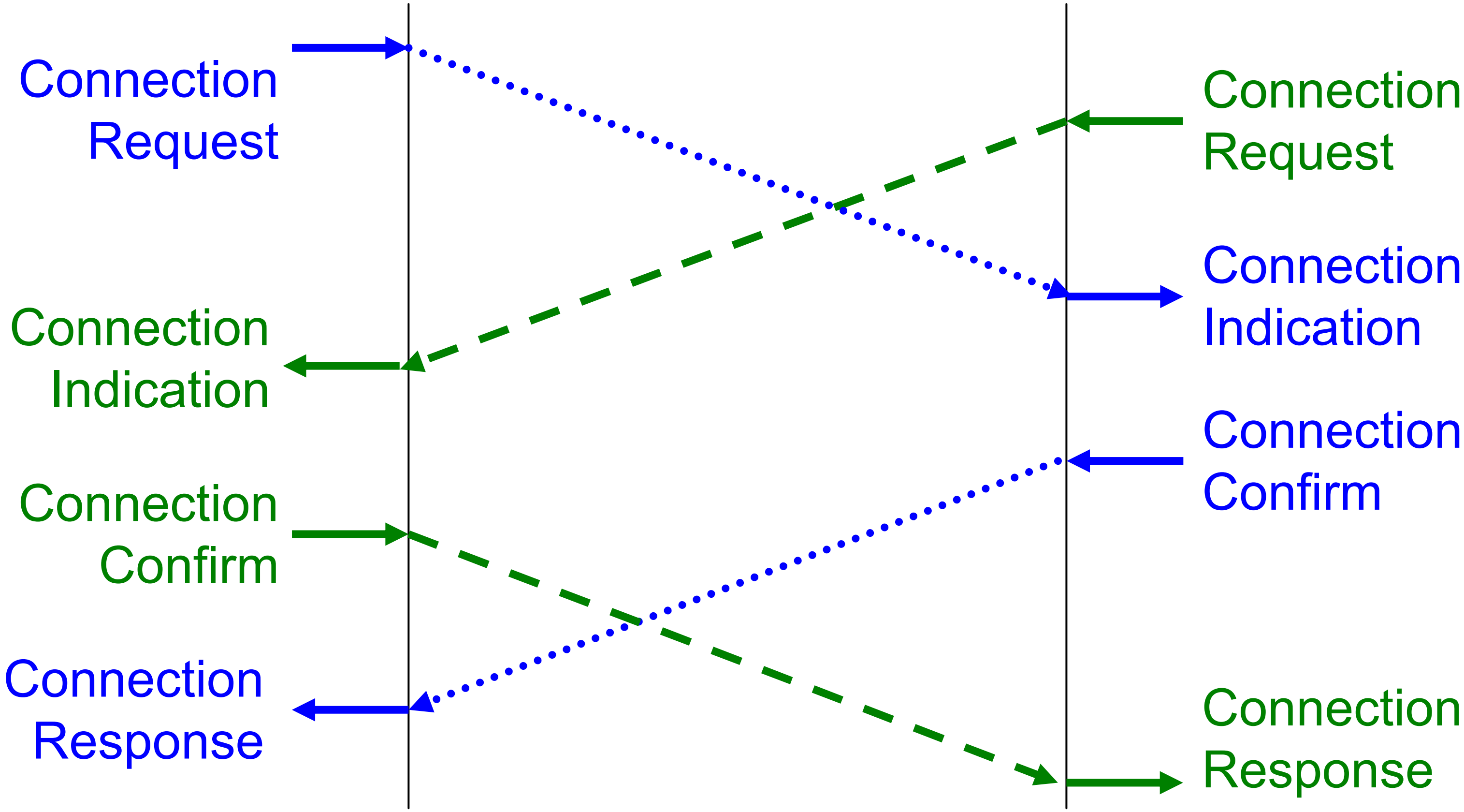
Connection Set-up (2)



In principle:
- Application can decide if it wants to accept the connection.

In practice:
- Application authorizes the OS kernel to accept or reject connections.

Connection Set-up (3)

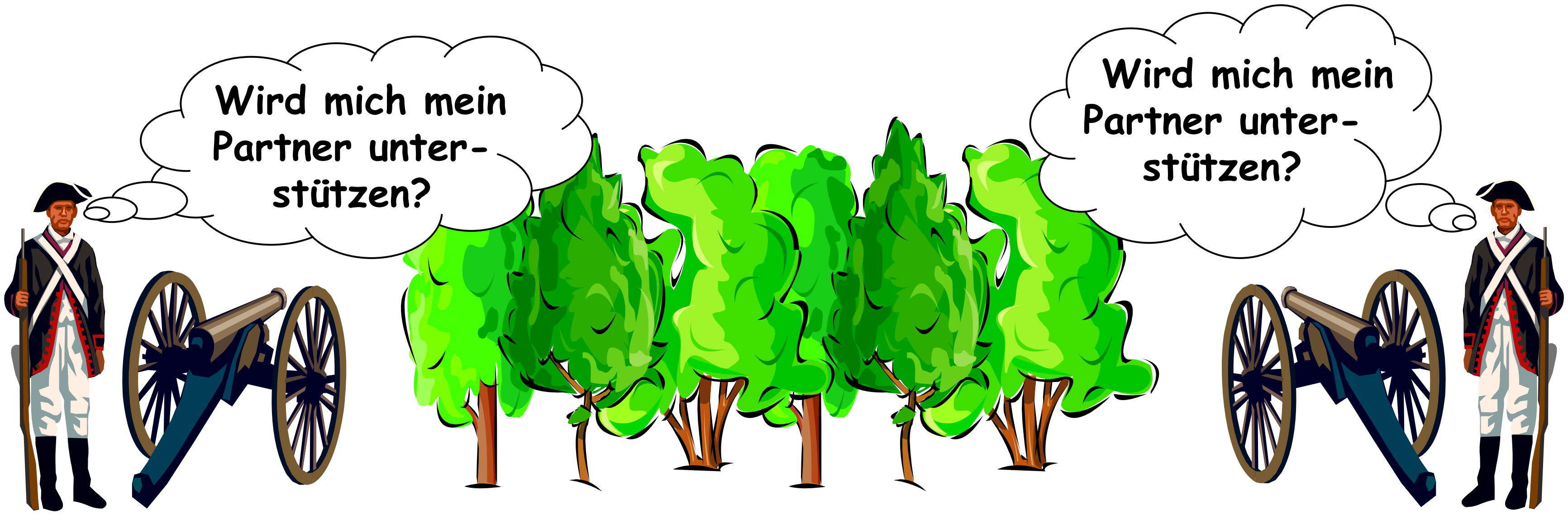


Depending on the specification of the transport protocol, this situation results in one or two connections. (TCP creates one connection.)

Connection Tear-Down (1)

Zwei alternative Semantiken:

- Einseitiger Verbindungsabbruch
  - Eine Station beschließt „aufzulegen“ und bricht damit die Verbindung
  - Ggf. noch im Netz befindliche Daten gehen verloren
- Gegenseitige Übereinkunft, Verbindung zu beenden
  - Erforderliche Abstimmung nicht mit absoluter Sicherheit zu erreichen (→***Problem der zwei Armeen***)



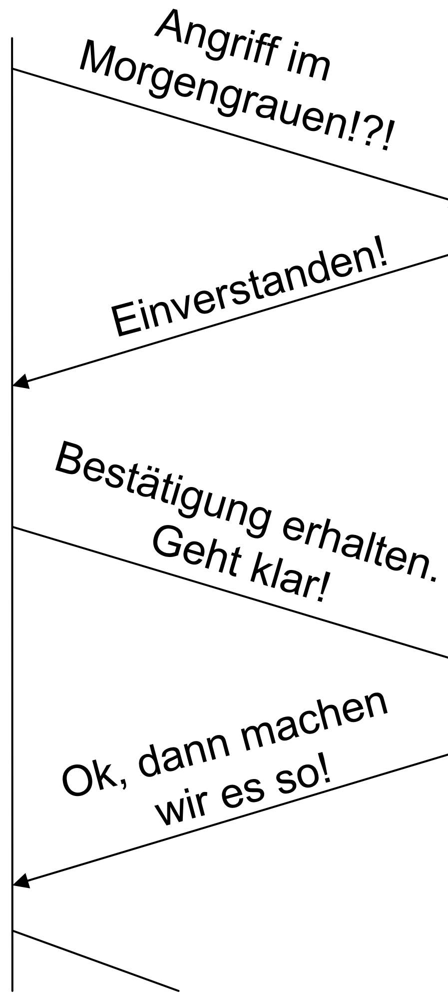
Connection Tear-Down (2)

Das Problem der zwei Armeen:

- Nur wenn beide Armeen gleichzeitig den Feind angreifen, können sie erfolgreich sein.
- Wie können die beiden Partner einen Angriffszeitpunkt ausmachen und *sicher* sein, dass nicht die entscheidende Nachricht verloren gegangen ist?

Das Problem ist in dieser Form unlösbar!

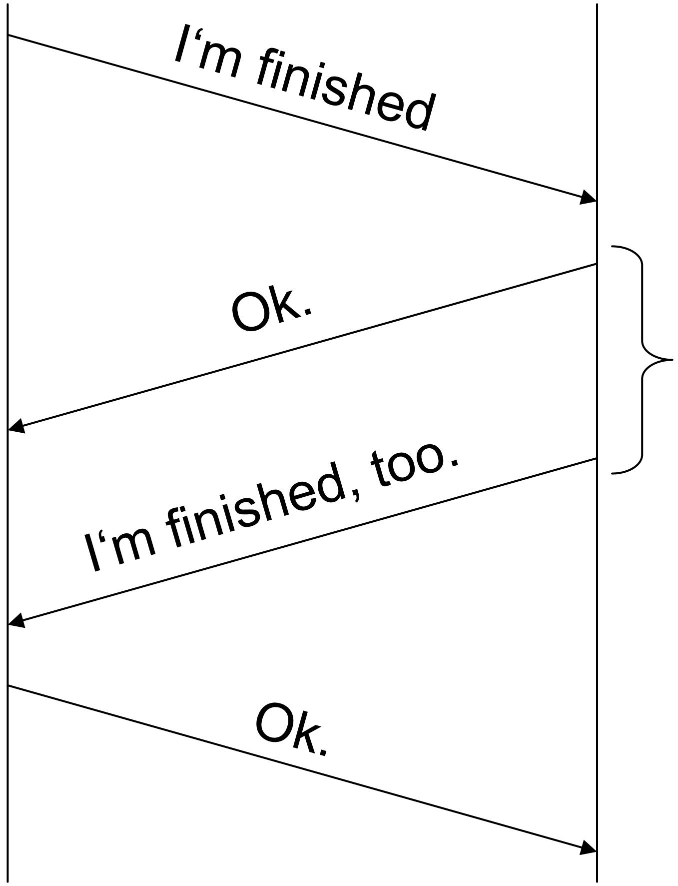
- Angenommen es gibt eine Nachricht, die über Angriff oder nicht Angriff entscheidet, dann ist der Verlust dieser Nachricht katastrophal.
- Andernfalls, könnte das Protokoll auf die Nachricht sowieso verzichten.
- Vollständige Induktion führt zum Widerspruch.



Connection Tear-Down (3)

Lösung für die Praxis: „Halbseitiges Schließen“

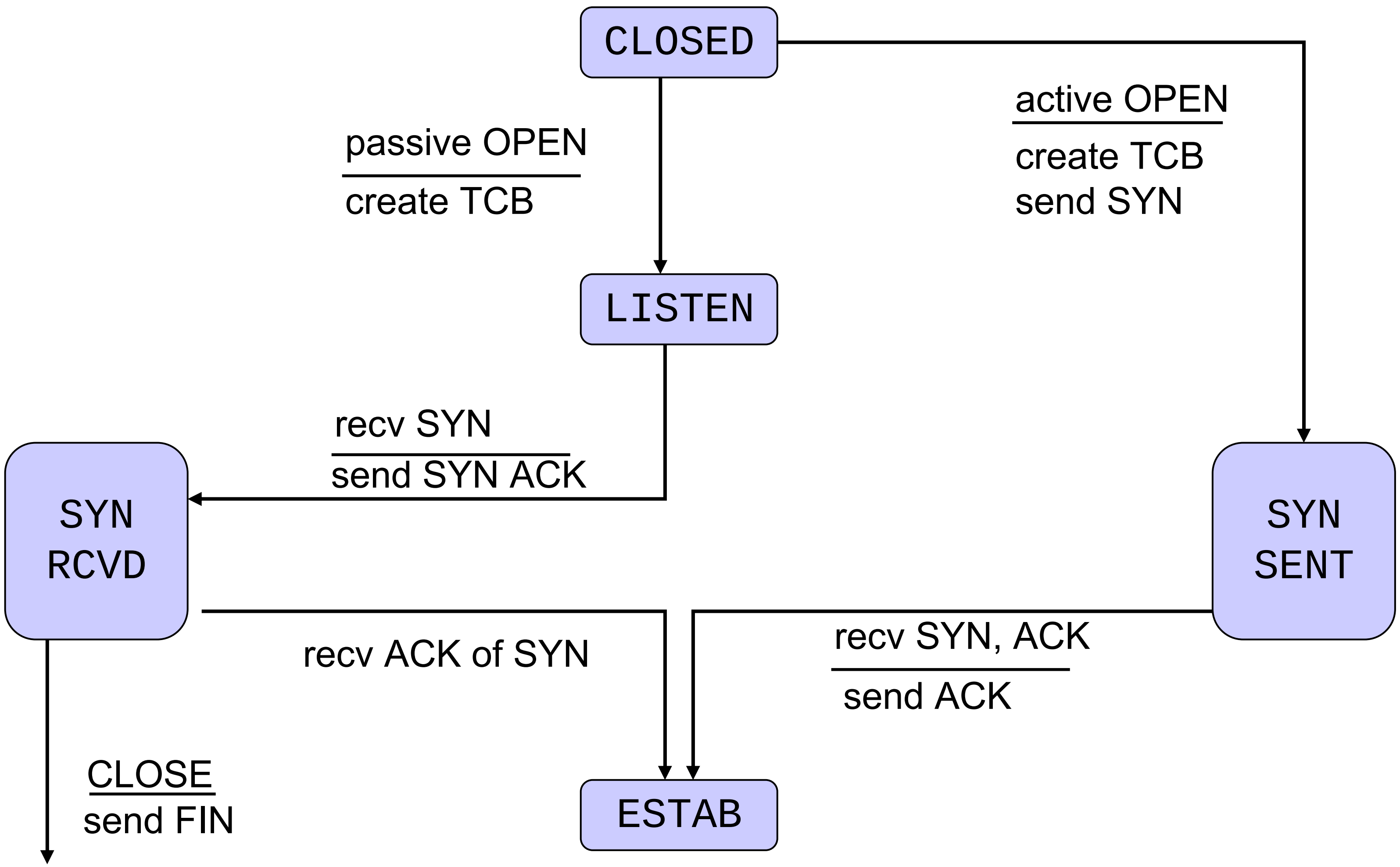
– Eine Seite sendet: „Das war‘s – ich werde nichts mehr senden.“
– Nach Empfang dieser Nachricht weiß die Anwendung der Gegenseite, dass keine weiteren Daten mehr kommen.
– Ebenso für die Gegenrichtung.



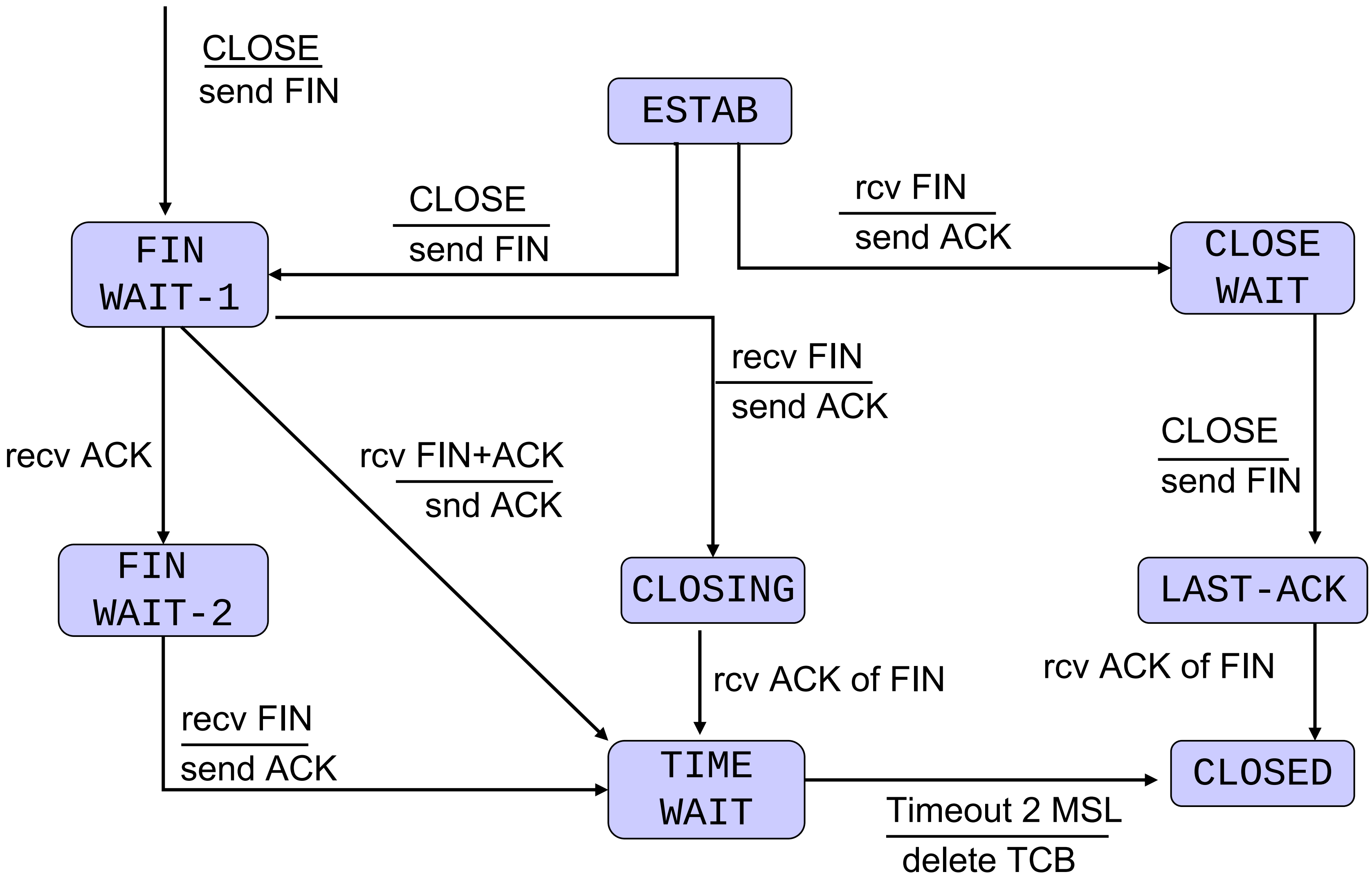
Können auch in eine Nachricht zusammen-gefaßt werden.

Jede Seite macht eine Aussage über das eigene Sendeverhalten, ohne sicher zu sein, dass diese von der Gegenseite empfangen wurde.

State Diagram: Establish TCP Connection



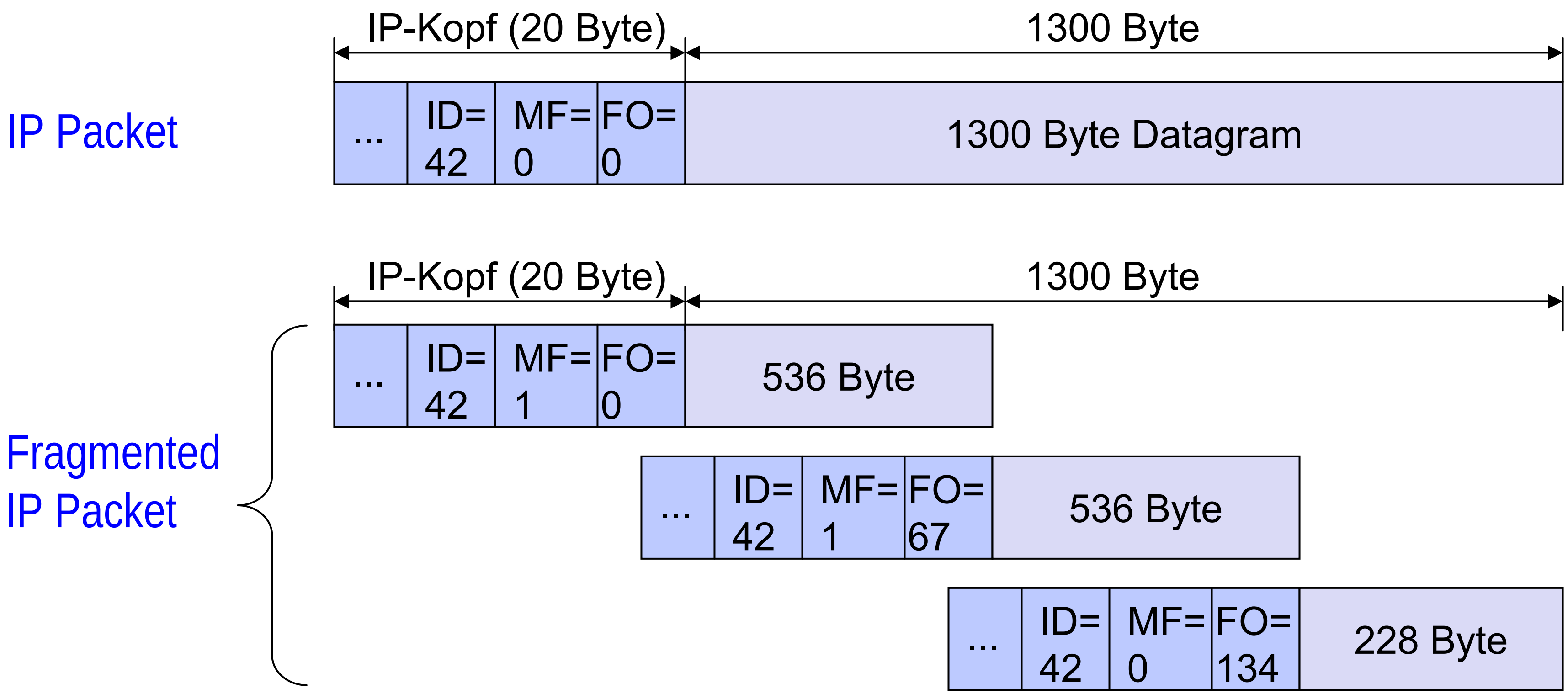
State Diagram: Tear down TCP Connection



Segmentation …

… of a continuous data stream
into datagrams

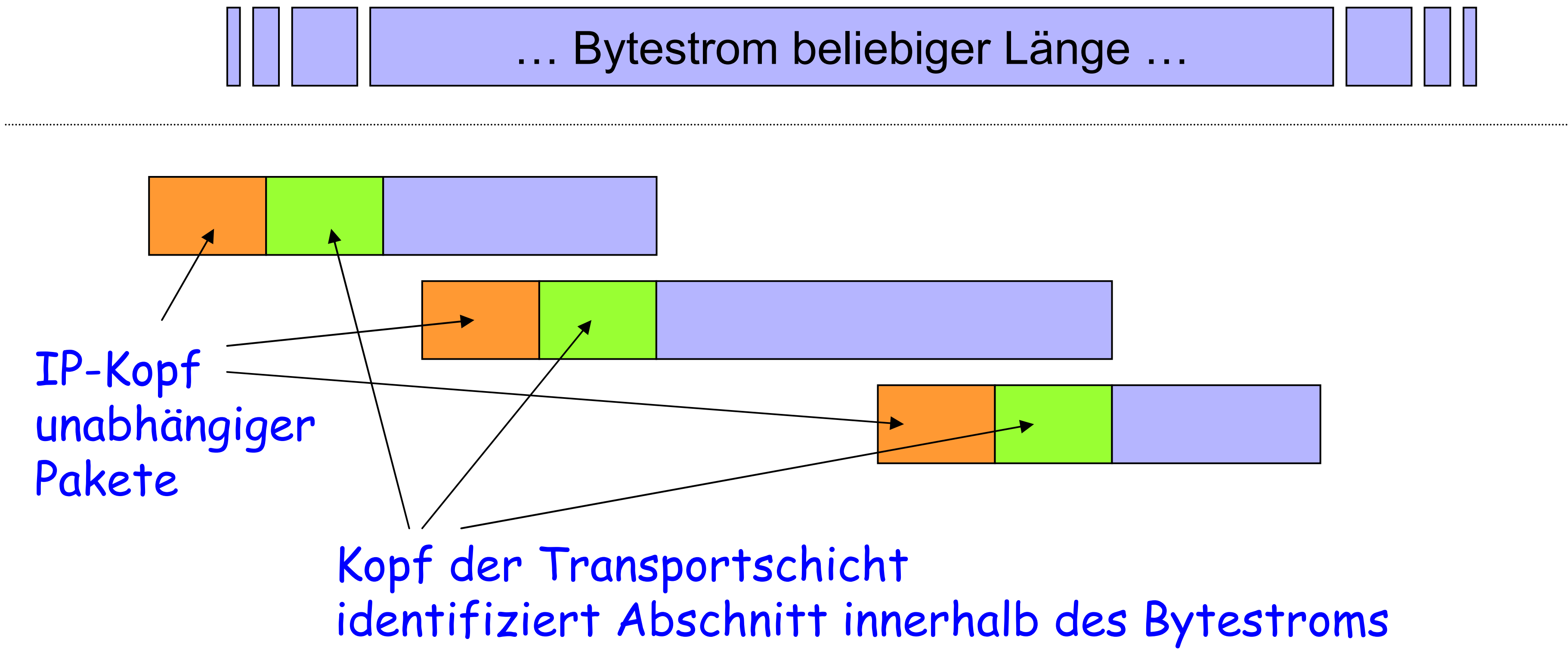
Fragmentation



The network layer can fragment packets to allow for small MTUs. If one of the fragments is lost, the whole packet is lost. Fragmentation inside the network is painful and thus discouraged.

Segmentierung

Zur Übertragung muss ein Bytestrom in IP-Pakete zerlegt werden. TCP zerlegt die ihm übergebenen Daten in Segmente unterschiedlicher Größe:



Zuverlässigkeit

- TCP definiert einen zuverlässigen Bytestrom-Transport
- IP bietet aber nur einen ungesicherten Datagrammdienst
  - Verlust einzelner oder mehrerer Pakete zulässig
  - Mehrfaches Eintreffen des selben Pakets zulässig
  - Reihenfolgevertauschung zulässig
- Zur *Behebung* dieser Paketfehler werden die folgenden Techniken verwendet
  - Quittungen (Acknowledgements)
  - Sendewiederholungen (Retransmissions)
- Dazu werden folgende Mechanismen verwendet
  - Sequenznummern (Sequence number)
  - Zeitgeber (Timer)

Sequenznummern

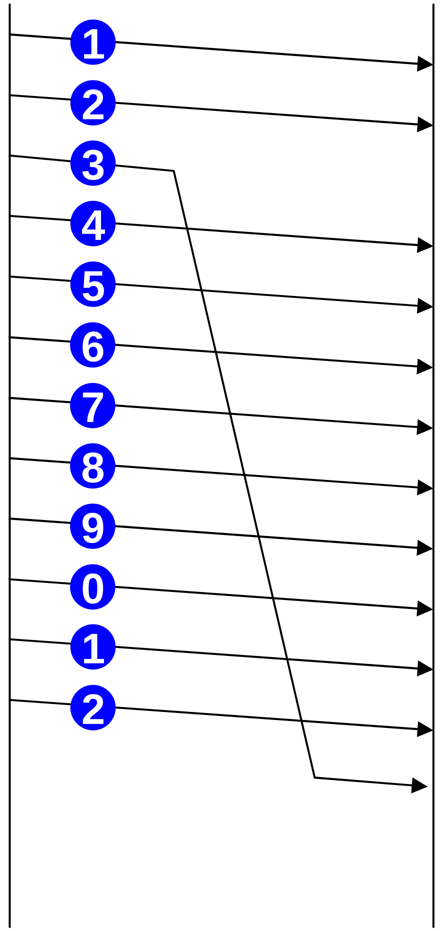
Sequenznummern erlauben es einer Station zu erkennen, dass

- die Dateneinheiten in der richtigen Reihenfolge ankommen
- keine Duplikate enthalten sind
- keine Dateneinheiten fehlen

Der *Sequenznummernraum* ist eine Teilmenge der natürlichen Zahlen

- Overhead im Paketkopf hängt ab von Größe des Sequenznummernraums, d.h. Sequenznummernraum sollte möglichst klein sein.
- Um (beliebig) lange Datenströme zu ermöglichen müssen Sequenznummern daher wieder verwendet werden.
- Damit der Empfänger ein verspätetes Segment als solches erkennt, darf eine Nummer so lange nicht wieder verwendet werden, wie das Segment noch im Netz unterwegs sein könnte!

Größe des Sequenznummernraums



Bestimmung der Größe L des Sequenznummernraums:

- Maximale Paket „Lebensdauer“ (MPL) [Sekunden]
- Maximale Zeit TR in der eine Sendewiederholung durchgeführt wird [Sekunden]
- Maximale Zeit TA bevor Empfänger nach Erhalt der Daten eine Quittung sendet [Sekunden]
- Maximale Übertragungsrate R des Senders [Pakete bzw. Byte pro Sekunde]
- Dann ergibt sich die untere Schranke: $L \geq (2\ MPL + TR + TA) * R$

Initiale Sequenznummer (1)

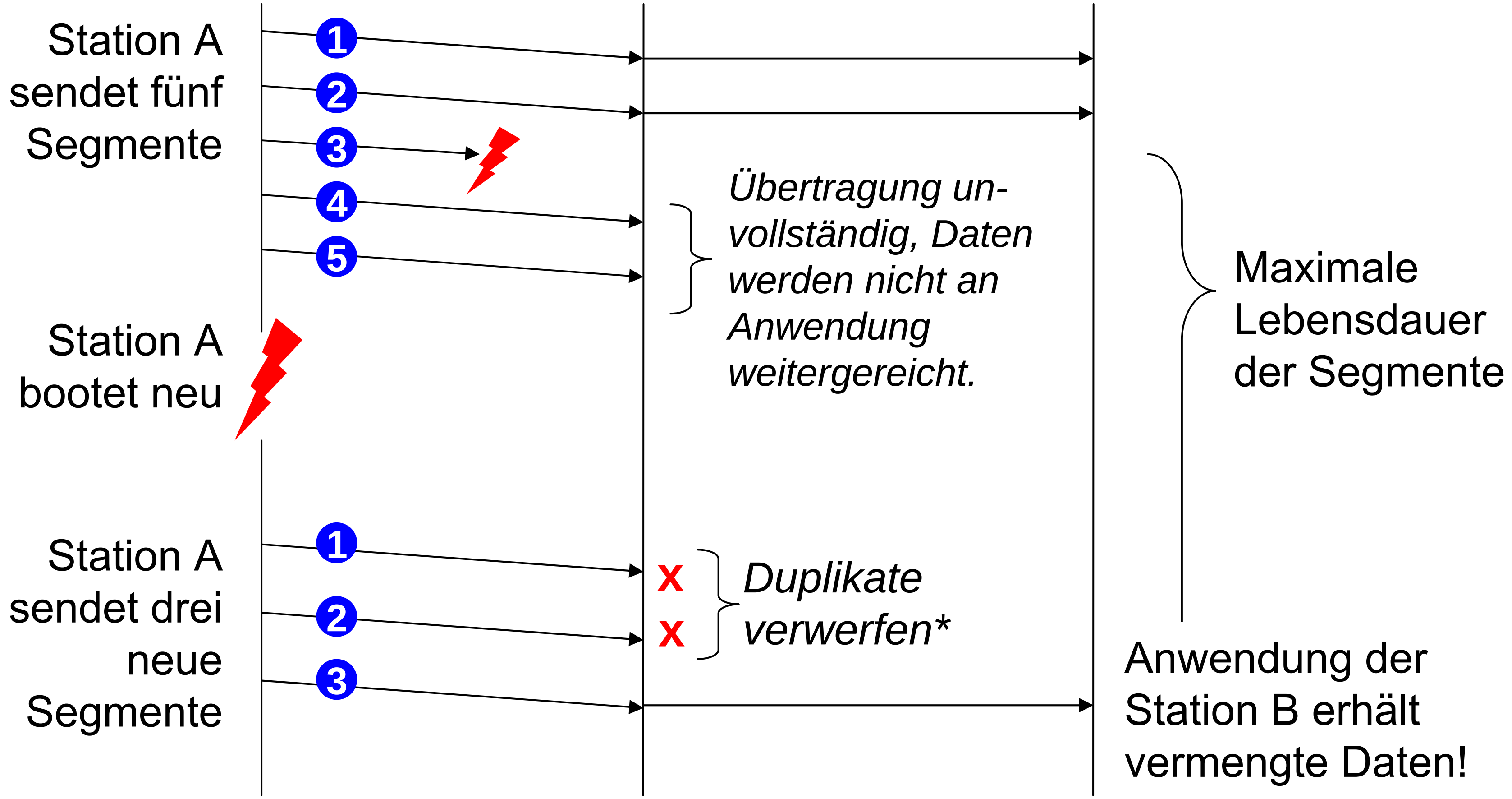
Problem:

- Station A beginnt eine Verbindung mit Segment Nummer 1
- Station A stürzt ab und bootet neu
- Station A beginnt eine neue Verbindung (mit anderen Inhalten) wieder mit Segment Nummer 1

Lösung:

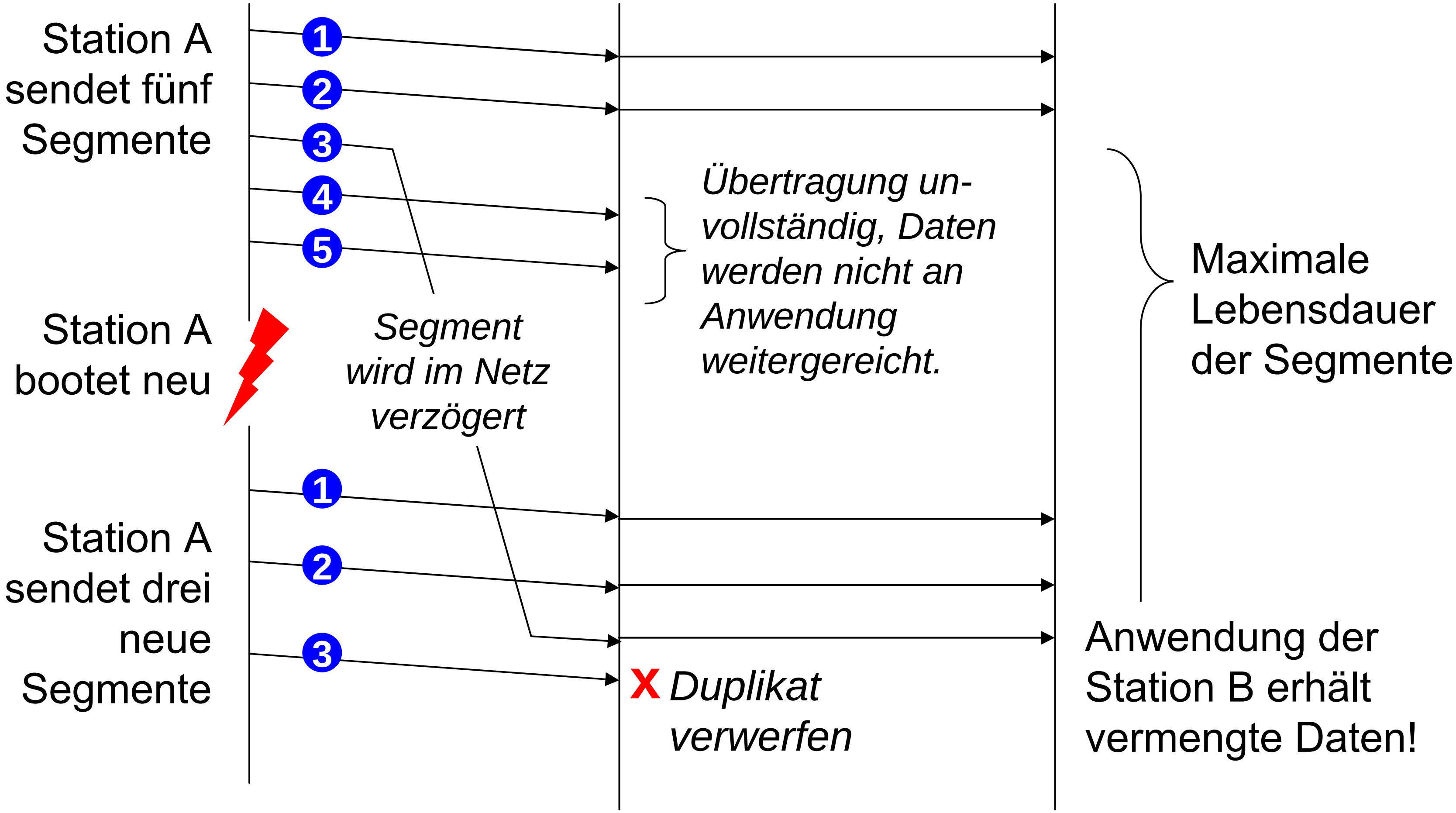
- Station B muss wissen wie lange Segmente maximal im Netz unterwegs sein können, d.h. ab wann kein Duplikat, sondern ein Segment einer neuen Verbindung angenommen wird (siehe oben).
- Während dieser Zeitspanne darf Station A die Segmentnummer nicht erneut verwenden, *selbst dann nicht, wenn A neu gebootet*.

Initiale Sequenznummer (2)



* Die Transportschicht kann hier nur anhand der Sequenznummern Duplikate erkennen. Andernfalls müßten ja Daten der „alten“ Segmente gepuffert werden.

Initiale Sequenznummer (3)



Initiale Sequenznummer (4)

Lösung 1

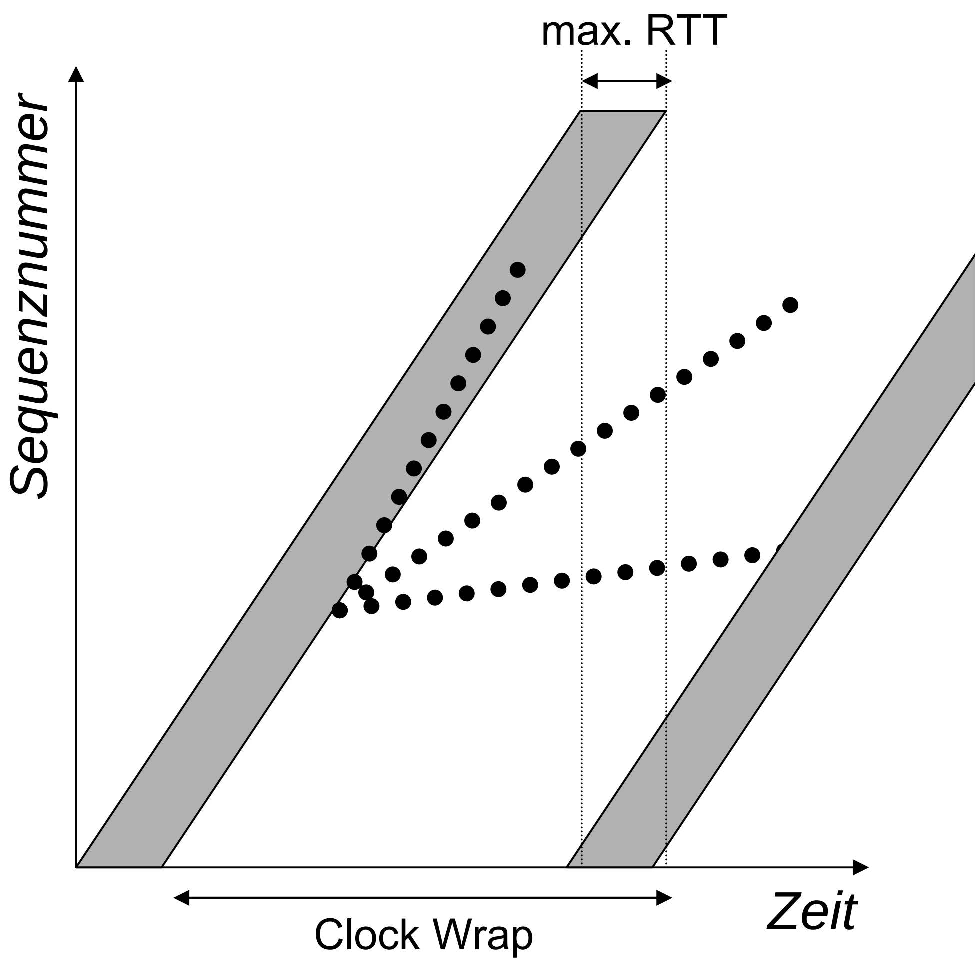
- Eine Station wartet nach dem Booten* zweimal die maximale Paketlebensdauer ab.
- Dann weiß jeder Empfänger, dass die neuen Segmente nicht zu alten Verbindungen gehören können.
- Die Station weiß, dass Quittungen garantiert zur neuen Verbindung gehören (deshalb zweimal die Maximaldauer).

Lösung 2: Clock-Tick-Methode (nach Tomlinson)

- Die initiale Sequenznummer einer jeden Verbindung wird aus einer Uhr gewonnen, die auch über das Booten hinweg weiterläuft.
- Die Uhr muss schneller laufen, als die Segmente gesendet werden.
- Ein Überlauf der Uhr muss seltener sein, als zweimal die maximale Lebensdauer der Pakete

* Gleiches gilt für das wieder verwenden einer Portnummer. Es müssen daher Daten auch über beendete Verbindungen noch einige Zeit gespeichert werden. Bei TCP typisch 30 Sekunden.

Clock-Tick-Methode

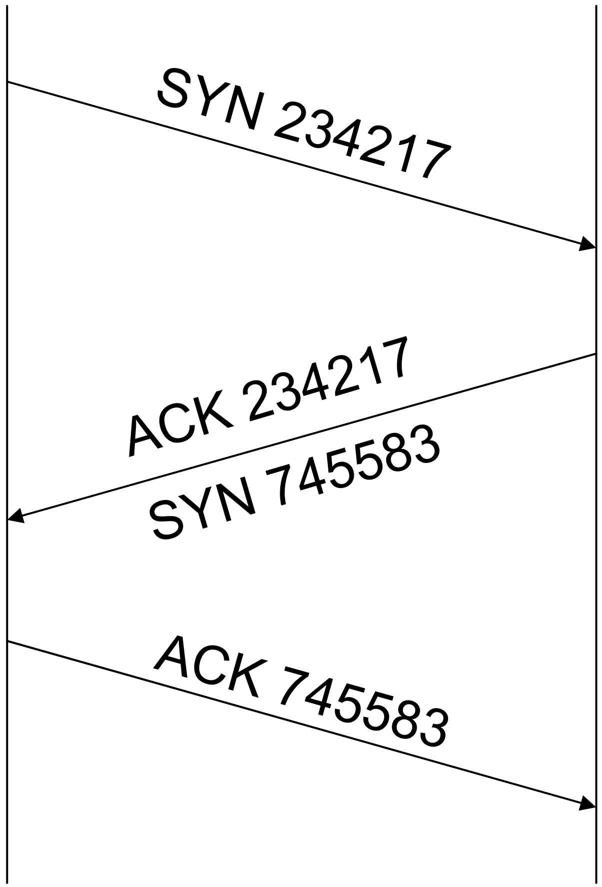


Verwendung der Clock-Tick-Methode führt dazu, dass

- … zu Beginn einer Verbindung die initiale Sequenznummer dem Partner übermittelt werden muss.
- … die durch die Clock-Ticks gegebene Senderate nicht überschritten werden darf.
- … bei lang dauernden Verbindungen (in Abhängigkeit von der Senderate) nach einiger Zeit die Sequenznummern angepasst werden muss.

Letzteres stellt einen ggf. erheblichen Protokollaufwand dar!

Drei-Wege-Handshake



Nicht nur bei der Clock-Tick-Methode, auch bei zufällig gewählten initialen Sequenznummern muss dem Kommunikationspartner die gewählte initiale Sequenznummer übermittelt werden.

Beim so genannten Drei-Wege-Handshake werden einschließlich der erforderlichen Quittungen drei Nachrichten ausgetauscht.

Frage: SYN-Flood Angriffe

Dieser Drei-Wege-Handshake erzwingt das Speichern von Sequenznummern beim Aushandeln für eine gewisse Zeit*. Dies ermöglicht z.B. den so genannten SYN-Flood-Angriff.

- Erläutern Sie wie ein SYN-Flood Angriff funktioniert!
- Schlagen Sie eine Gegenmaßnahme gegen SYN-Floods vor! – Berücksichtigen Sie, dass Angriffe auch verteilt von verschiedenen Rechnern ausgehen können.

Begriffe:

Denial of Service (DoS) Angriff

Distributed Denial of Service (DDoS) Angriff

* Beispiel: Linux bot 1996, d.h. zur Zeit als dieser Angriff „populär“ wurde, Platz für 256 gleichzeitig auszuhandelnde Verbindungen. Jeder Eintrag wurde für ca. drei Minuten gespeichert.

Lösung gegen SYN-Floods

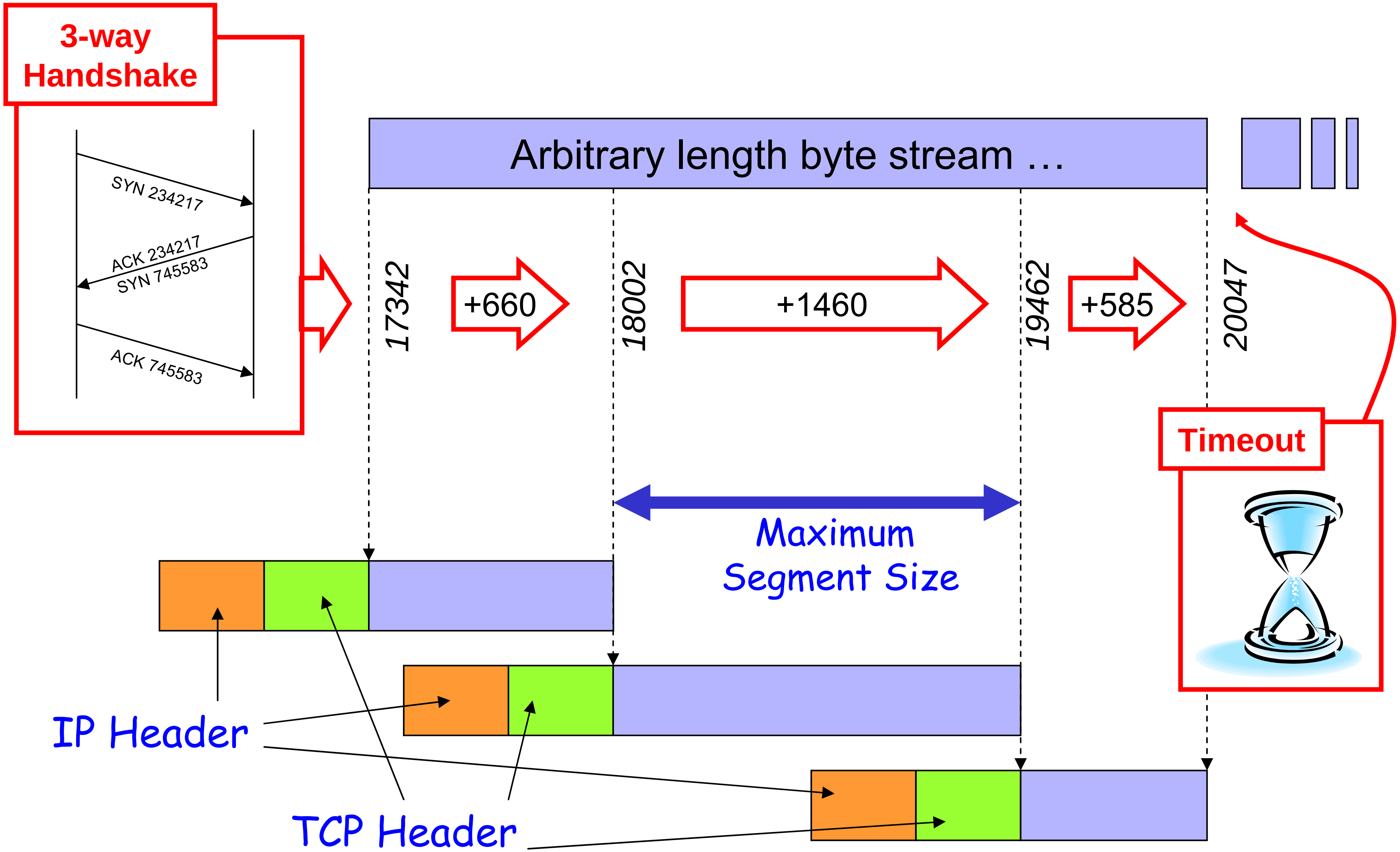
- Einfache Lösung:
  - Viel mehr Speicher für Zustand (Cisco: 128MByte für die gesamte Zustandhaltung inkl. SYN-Backlog)
  - Viel kürzerer Timeout für SYN ohne ACK (Paketverlust verhindert Zustandekommen einer Verbindung, aber der Anwender kann es ja nochmal versuchen)
- Dan Bernstein und Eric Schenk (1996): SYN Cookies
  siehe http://cr.yp.to/syncookies/archive
  - Keinerlei Zustandshaltung auf dem Zielrechner bis Gegenstelle den Drei-Wege-Handshake abgeschlossen hat

Sequence Number Wrap Around

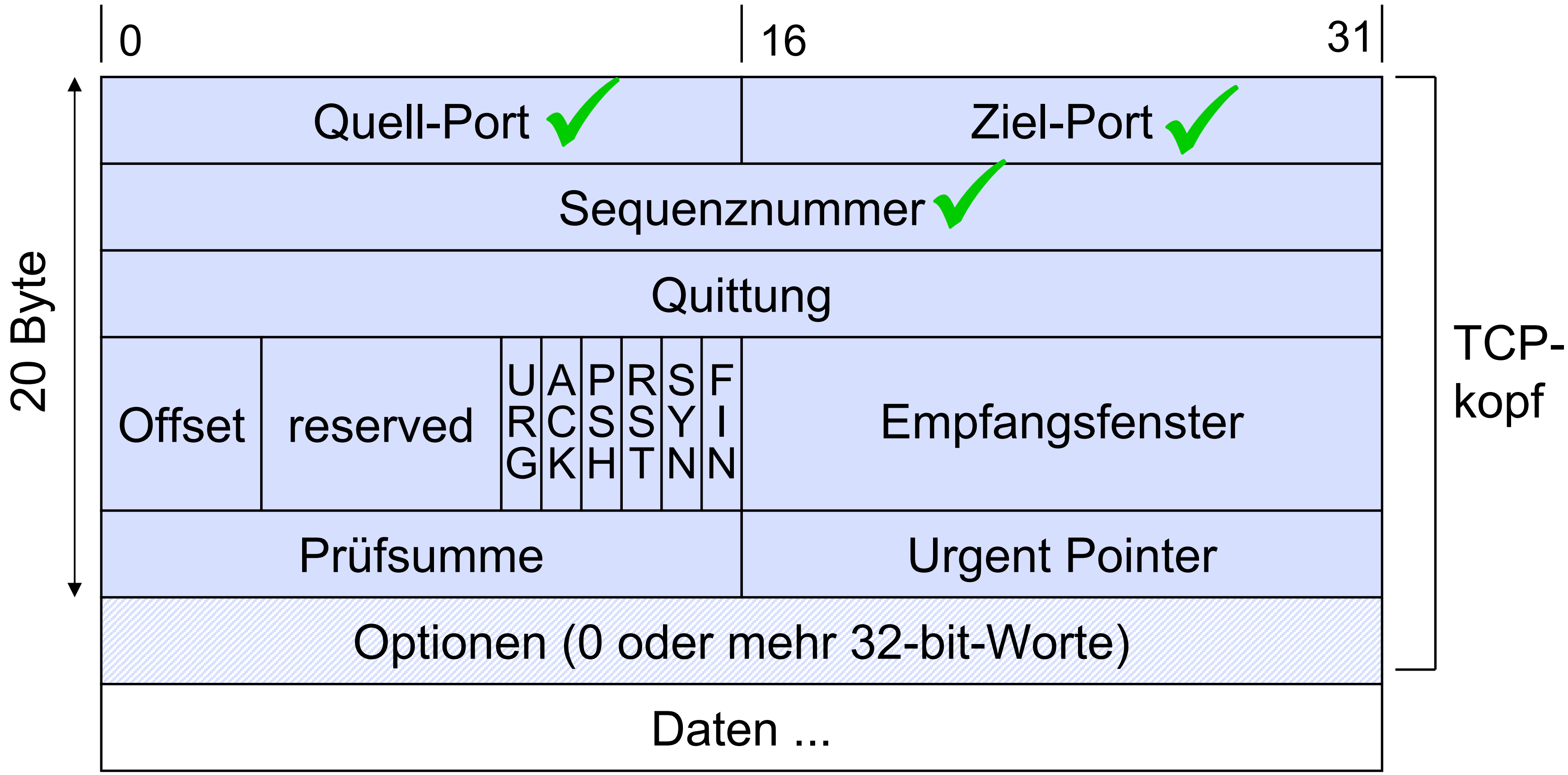
- RFC 1323 (Jacobson, Braden, and Borman, TCP Extensions for High Performance, May 1992) addresses the problem of sequence wrap around (as well as window size limitations, see later) in high speed networks.
- RFC 793 (Postel ed., Transmission Control Protocol, Sep 1981) assumes a maximum segment lifetime (MSL) of 2 minutes.
- Moving to Gigabit networks poses a problem:

ARPANET	56kbps	~3.6 days
DS1	1.5Mbps	~3 hours
Ethernet	10Mbps	~30 min
FDDI	100Mbps	170 sec
Gigabit	1Gbps	17 sec

Transport Protocol TCP – Segmentation



TCP Header


0
16
31
20 Byte
Quell-Port
Ziel-Port
Sequenznummer
Quittung
Offset
reserved
URG
ACK
PSH
RST
SYN
FIN
Empfangsfenster
Prüfsumme
Urgent Pointer
Optionen (0 oder mehr 32-bit-Worte)
Daten ...
TCP-kopf

Questions?

Thomas Fuhrmann

Department of Informatics
Self-Organizing Systems Group
c/o I8 Network Architectures and Services
Technical University Munich, Germany

fuhrmann@net.in.tum.de