Transportschicht

Dienste der Transportschicht

Die Transportschicht bietet einen verbindungsorientierten und einen verbindungslosen Dienst, unabhängig von den Diensten der zugrunde liegenden Vermittlungsschicht.

Im verbindungsorientierten Dienst bietet die Transportschicht einen gesicherten, bestätigten Datenstrom zwischen den Endpunkten.

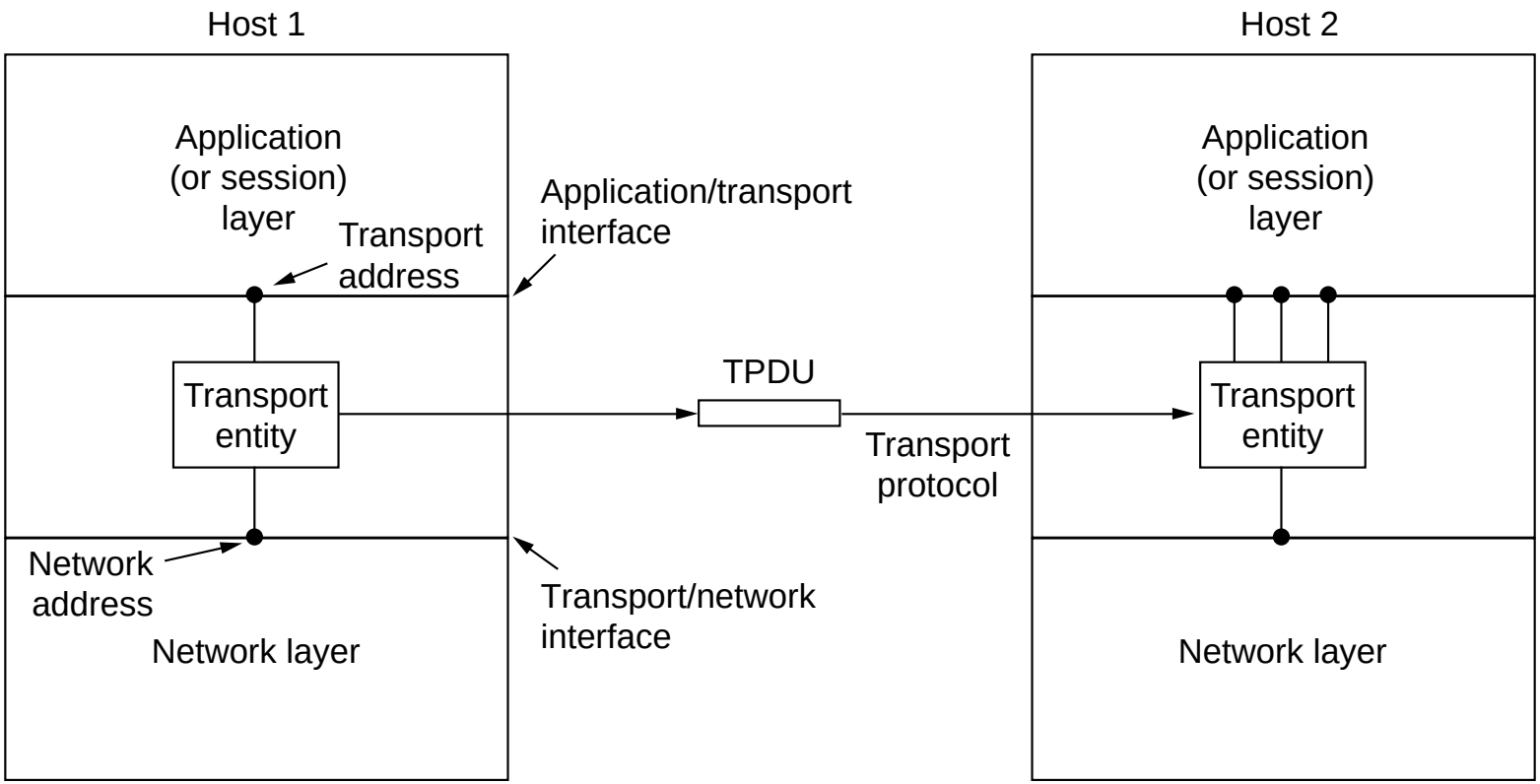
Endpunkte werden durch Transportadressen beschrieben. Im Internet z.B. sind die Endpunkte Prozesse, die durch Transportadressen identifiziert werden.

Der verbindungslose Dienst regelt nur die Adressierung.

Funktionen der Transportschicht

- Adressierung
- Segmentierung
- Verbindungsaufbau / Verbindungsabbau
- Fehlererkennung / Fehlerbehebung
- Flußkontrolle

Nachbarschichten der Transportschicht



(c) Tanenbaum, Computer Networks

Vergleich Transportschicht und Netzwerkschicht

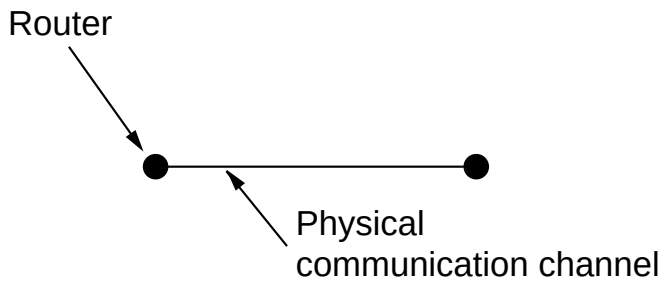
Gemeinsamkeiten

- Beide bieten verbindungsorientierte und -lose Dienste
- Adressierung und Flußkontrolle vergleichbar
- Verbindungen bestehen aus drei Phasen: Aufbau, Datenübertragung, Abbau

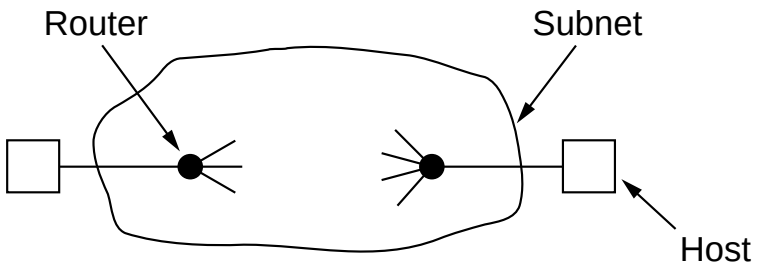
Unterschiede

- Transportschicht liegt nur beim Nutzer/Endgerät, Netzwerkschicht auch im Netz (Router, Switche,...)
- Transportschicht ermöglicht Unabhängigkeit des Nutzers von der eigentlichen Übertragung, d.h. Nutzer sieht nicht welches Netz, Nutzer sieht Änderungen im Netz nicht,...

Vergleich Transportschicht und Sicherungsschicht



(a)



(b)

(c) Tanenbaum, Computer Networks

a) Sicherungsschicht
b) Transportschicht

Vergleich Transportschicht und Sicherungsschicht

Gemeinsamkeiten

- Beide bieten Flußkontrolle, Fragmentierung, Fehlererkennung

Unterschiede

- Sicherungsschicht hat eine eindeutige physikalisch Verbindung, Transportschicht sieht ein Netzwerk
- Sicherungsschicht braucht im Gegensatz zur Transportschicht keine Adressierung
- Sicherungsschicht benötigt keinen Verbindungsaufbau, Gegenstelle ist immer da
- Im Netz der Transportschicht können Verzögerungen durch Speicher, Buffer und Jitter auftreten, d.h. Pakete können mehrere Sekunden verspätet eintreffen

Verbindungsaufbau

Probleme

- Probleme durch verlorene, verspätete oder doppelte Pakete
- Probleme wenn ein Host rebootet (d.h. seinen Speicher verliert)

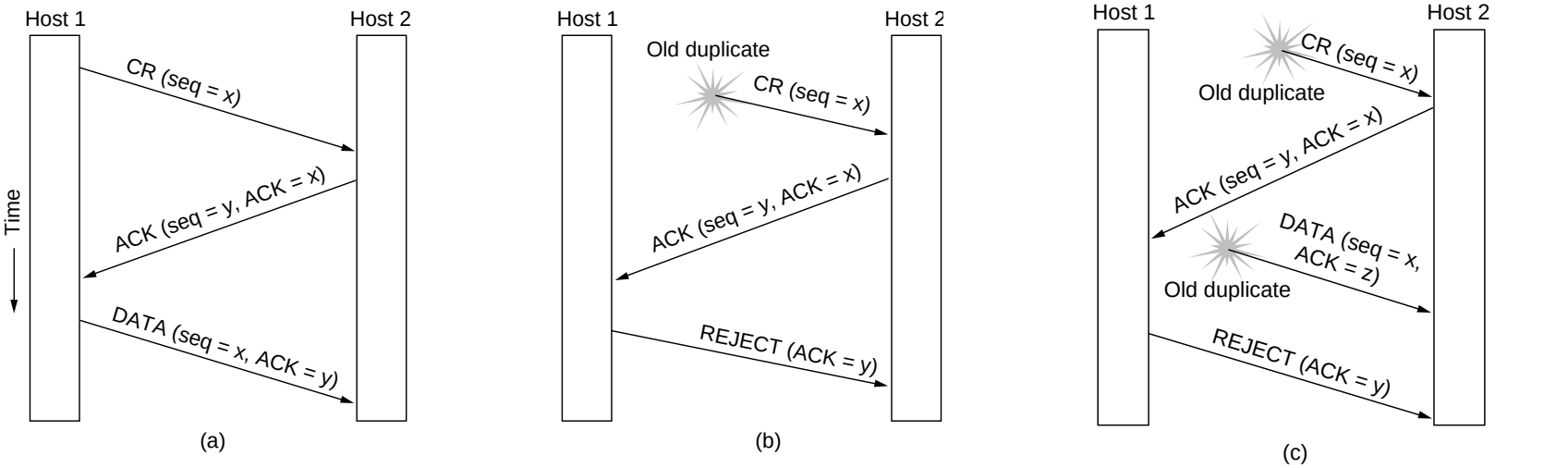
Lösungsansätze

- Eindeutige Sequenznummern für Pakete
- Begrenzte Lebensdauer von Pakete

Beispiel: “three-way handshake”

- Beide Hosts senden ein ACK mit der empfangenen Sequenznummer des anderen Hosts

Verbindungsaufbau mit “three-way handshake”



(c) Tanenbaum, Computer Networks

a) Normalfall

b) Veraltete Verbindungsanforderung (connection request CR)

c) Veralteter CR und veraltetes ACK

Verbindungsabbau

- Asymmetrischer Abbau: Ein Host kann die duplex Verbindung komplett beenden
- Symmetrischer Abbau: Getrennter Abbau der beiden Richtungen

Probleme

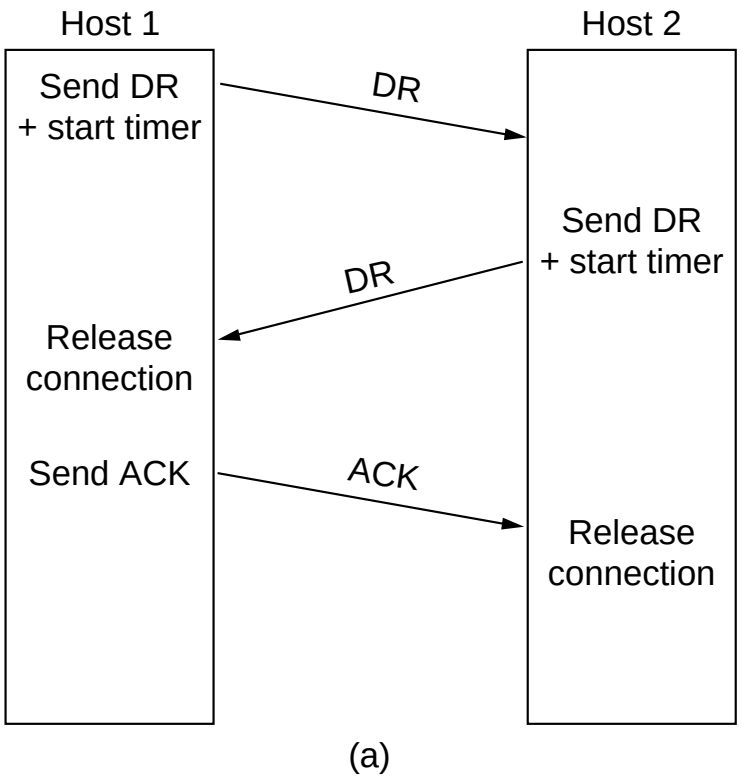
- Zuverlässiger Verbindungsabbau bei möglichen Paketverlusten
- siehe z.B. “Two-Army Problem” bzw. “Coordinated Attack Problem”

Lösungsansätze:

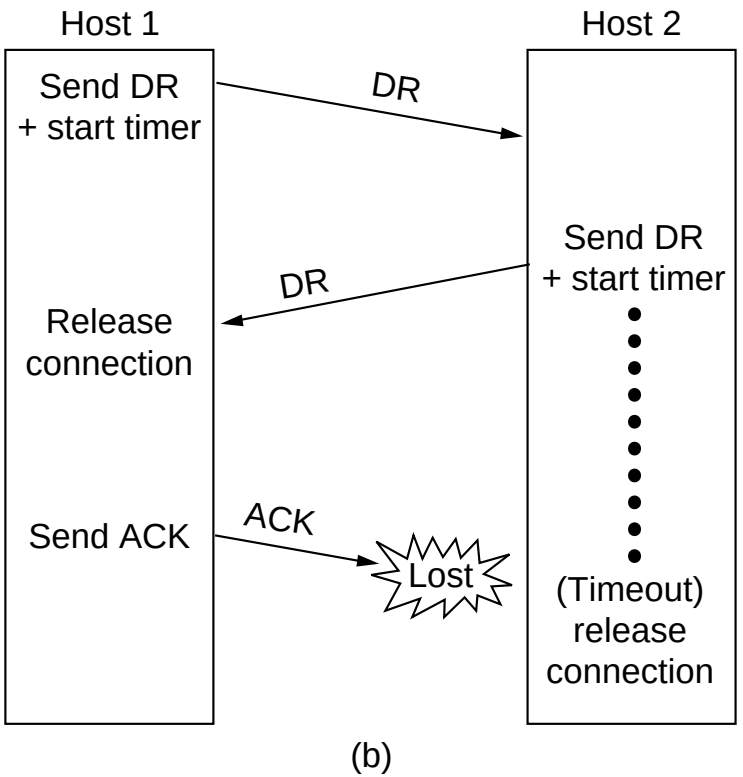
- Nutzung von Timern um verlorene Pakete zu detektieren
- z.B. Verbindungsabbau mit “three-way handshake”

Verbindungsabbau mit “three-way handshake” (1)

Mögliche Szenarien



(a)

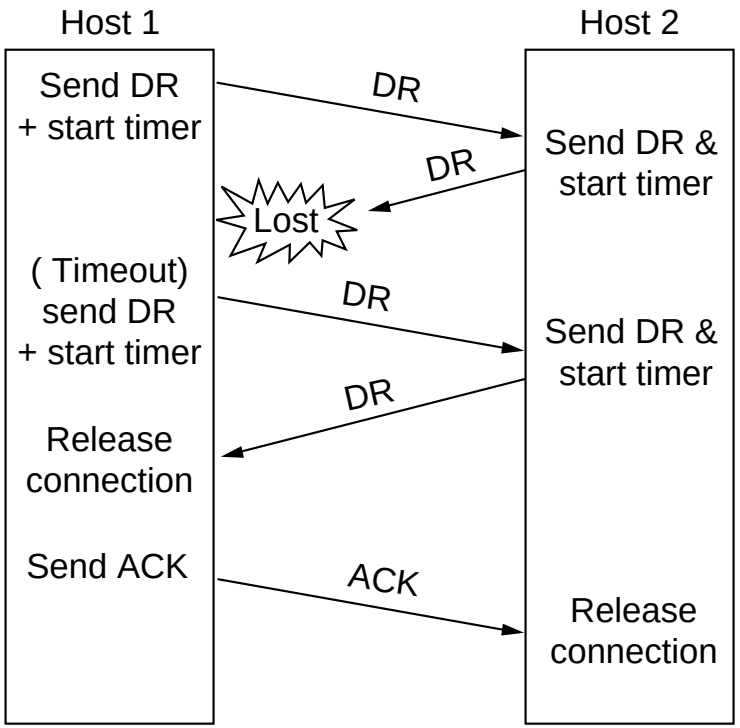


(b)

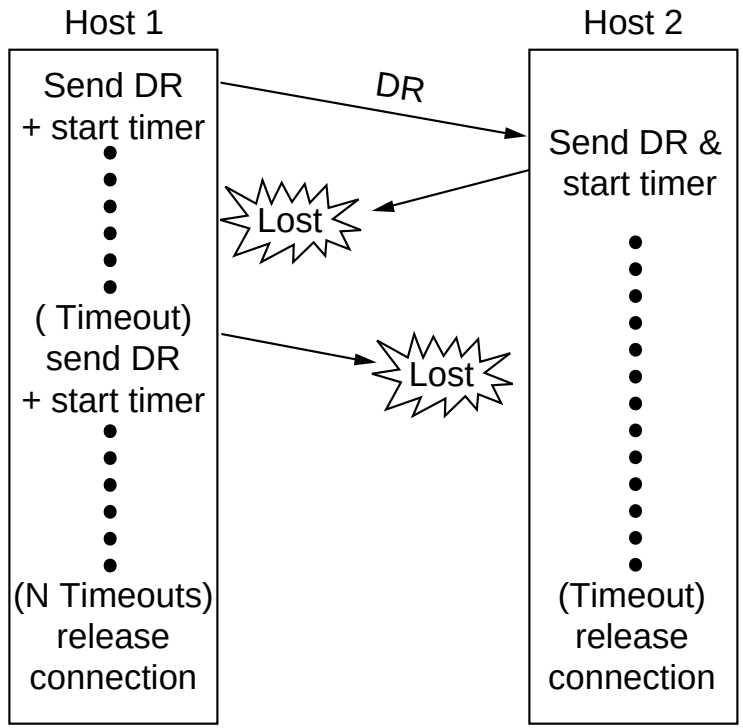
(c) Tanenbaum, Computer Networks

Verbindungsabbau mit “three-way handshake” (2)

Mögliche Szenarien



(c)



(d)

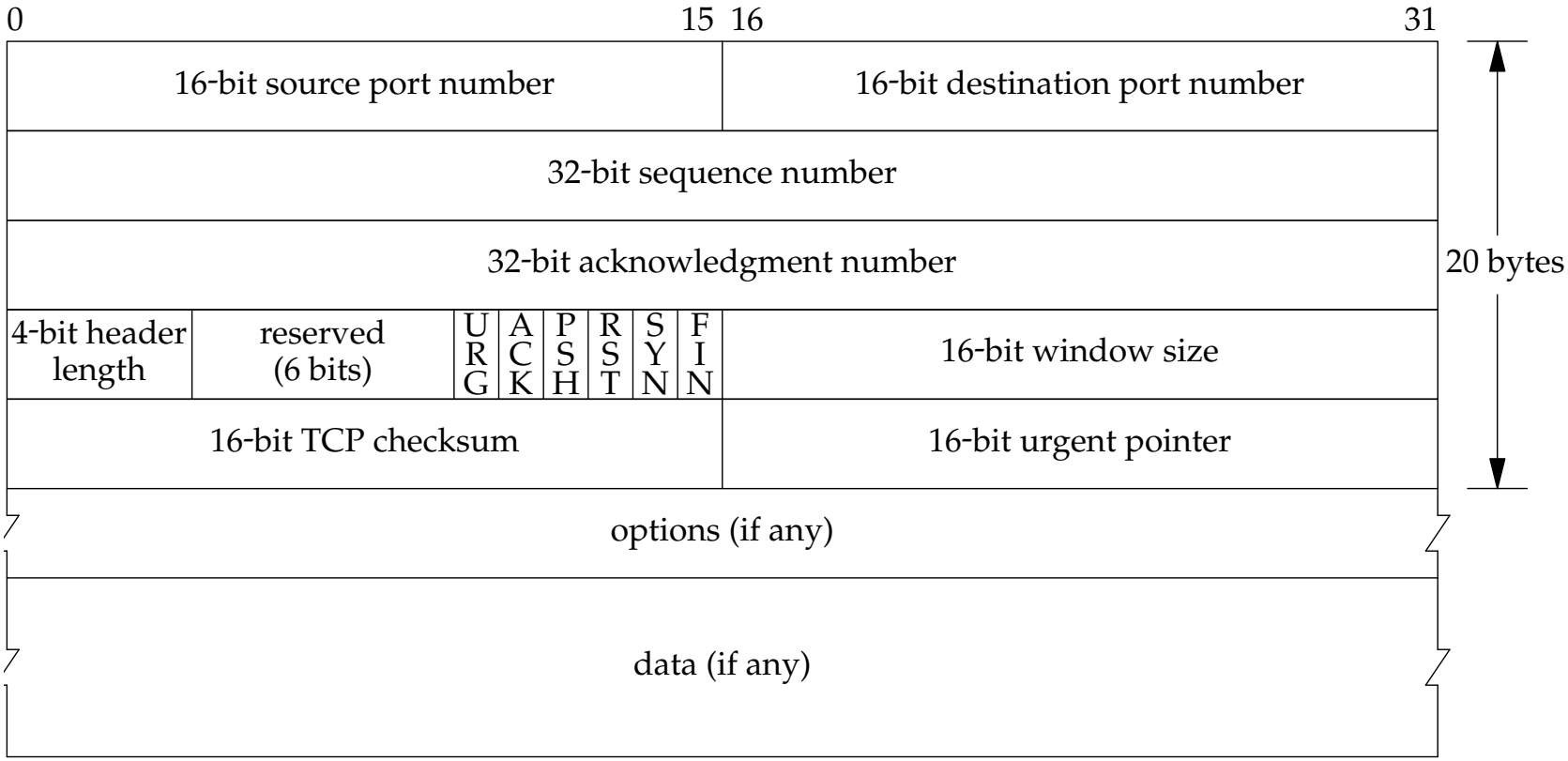
(c) Tanenbaum, Computer Networks

Transmission Control Protocol (TCP)

Grundlagen

- Wert des Protokoll Feldes im IP Header ist 6 (vgl. RFC1700)
- Gesicherter, verbindungsorientierter Transport
- TCP Verbindungen haben immer genau zwei Endpunkte
- Daten sind bei TCP immer eine Folge von 8 Bit Zeichen
- Jedes Byte hat eigene Sequenznummer

TCP Rahmen


0
15
16
31
16-bit source port number
16-bit destination port number
32-bit sequence number
32-bit acknowledgment number
20 bytes
4-bit header length
reserved (6 bits)
URG
ACK
PSH
RST
SYN
FIN
16-bit window size
16-bit TCP checksum
16-bit urgent pointer
options (if any)
data (if any)

Felder im TCP Header

- **Source Port:** Nummer, mit der die sendende Applikation vom Host identifiziert werden kann.
- **Destination Port:** Nummer, mit der die empfangende Applikation vom Host identifiziert werden kann.
- **Sequence Number:** Startposition der Daten im TCP Fenster des Senders.
- **ACK Sequence Number:** Sequenznummer der ersten freien Position im TCP Fenster des Empfängers. Bis hier sind alle Daten korrekt empfangen worden.
- **Header Length:** Länge des TCP Headers inklusive Optionen in 4Byte Einheiten.

Flags im TCP Header

LSB zuerst:

- **FIN** Der Sender teilt dem Empfänger mit, daß keine weiteren Daten folgen.
- **SYN** Sender teilt dem Empfänger die initiale Sequenznummer mit.
- **RST** Reset der Verbindung, jeglicher Kontext wird verworfen.
- **PSH** Der Empfänger soll diese Daten sofort an die Anwendungsschicht weiterreichen.
- **ACK** Daten im Feld ACK Sequence Number sind gesetzt.
- **URG** Daten im Urgent Pointer sind gesetzt.
- **ECE** ECN-Echo (RFC3168), Bit 9 vor URG
- **CWR** Congestion Window Reduced (RFC3168), Bit 8 vor ECE
- **NS** Nonce Sum (RFC3540), experimentelle Erweiterung zu RFC3168, Bit 7 vor CWR

Felder im TCP Header

- **Window Size:** Menge an Daten, die der Sender des Paketes beginnend mit der ACK Sequence Number beim Empfang noch zwischenspeichern kann. Die Einheit hängt von TCP Optionen ab (RFC1323)
- **TCP Checksum:** Prüfsumme über erweiterten Header und Daten.
- **Urgent Pointer:** Wenn das URG-Flag gesetzt ist, zeigt der Urgent Pointer hinter das letzte Byte der Urgent Daten im Segment.
- **TCP Length:** Pseudo Header, Länge von TCP Header und Daten (Verwendung bei Prüfsummenberechnung)

TCP Prüfsummenberechnung

Verwendet wird das vom IP Header bekannte Verfahren. Pakete mit ungerader Anzahl Bytes werden mit einem 0 Byte aufgefüllt. Das Datagram wird um einige Felder des IP Headers erweitert:

<table>
<tr><td colspan="3">Source IP Address</td></tr>
<tr><td colspan="3">Destination IP Address</td></tr>
<tr><td>0</td><td>Protocol</td><td>TCP Length</td></tr>
<tr><td colspan="3">TCP Header</td></tr>
<tr><td colspan="3">(Padded) TCP Data</td></tr>
</table>

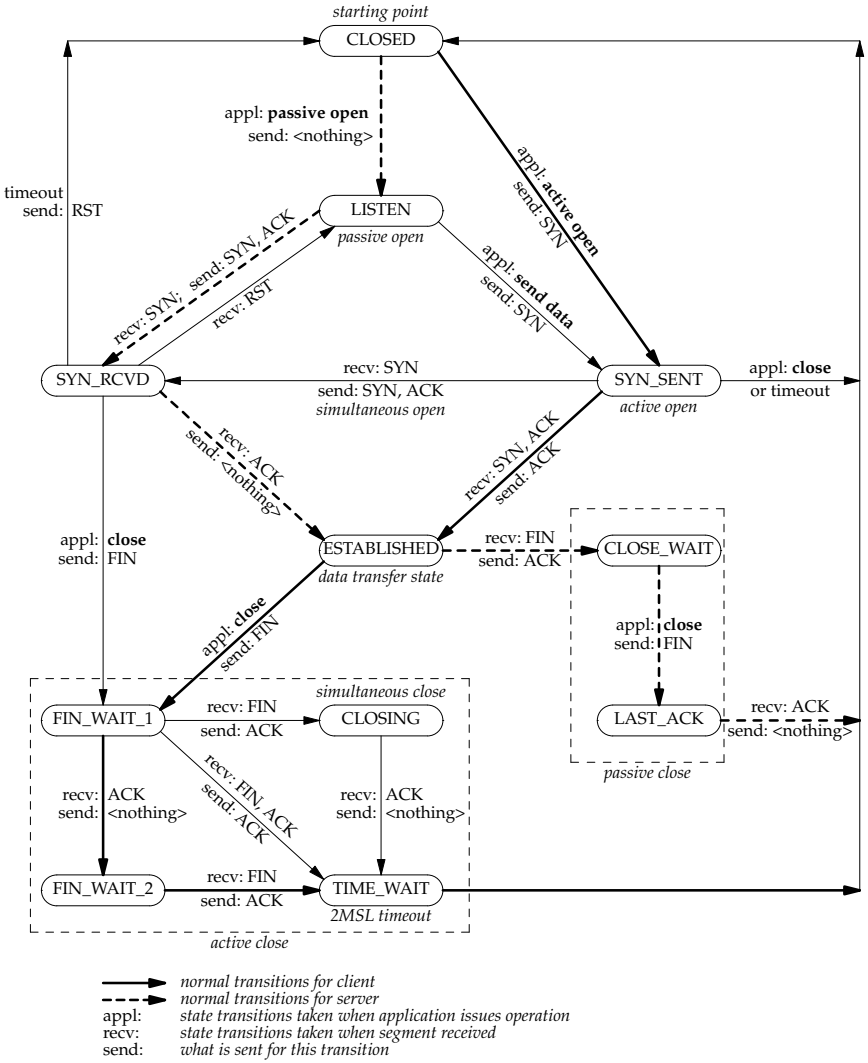
Bezeichnungen / Sprechweisen

- **Segment**: TCP Rahmen mit Daten
- **MSL (Maximum Segment Lifetime)**: Zeit(!), die ein Segment maximal im Netz verbringen kann (2 min nach RFC793)
- **SYN,FIN,ACK,RST**: TCP Rahmen, in dem das entsprechende Flag gesetzt ist; Kombinationen sind üblich, z.B. SYN-ACK
- **MSS (Maximum Segment Size)**: Maximale Datenmenge in einem Segment

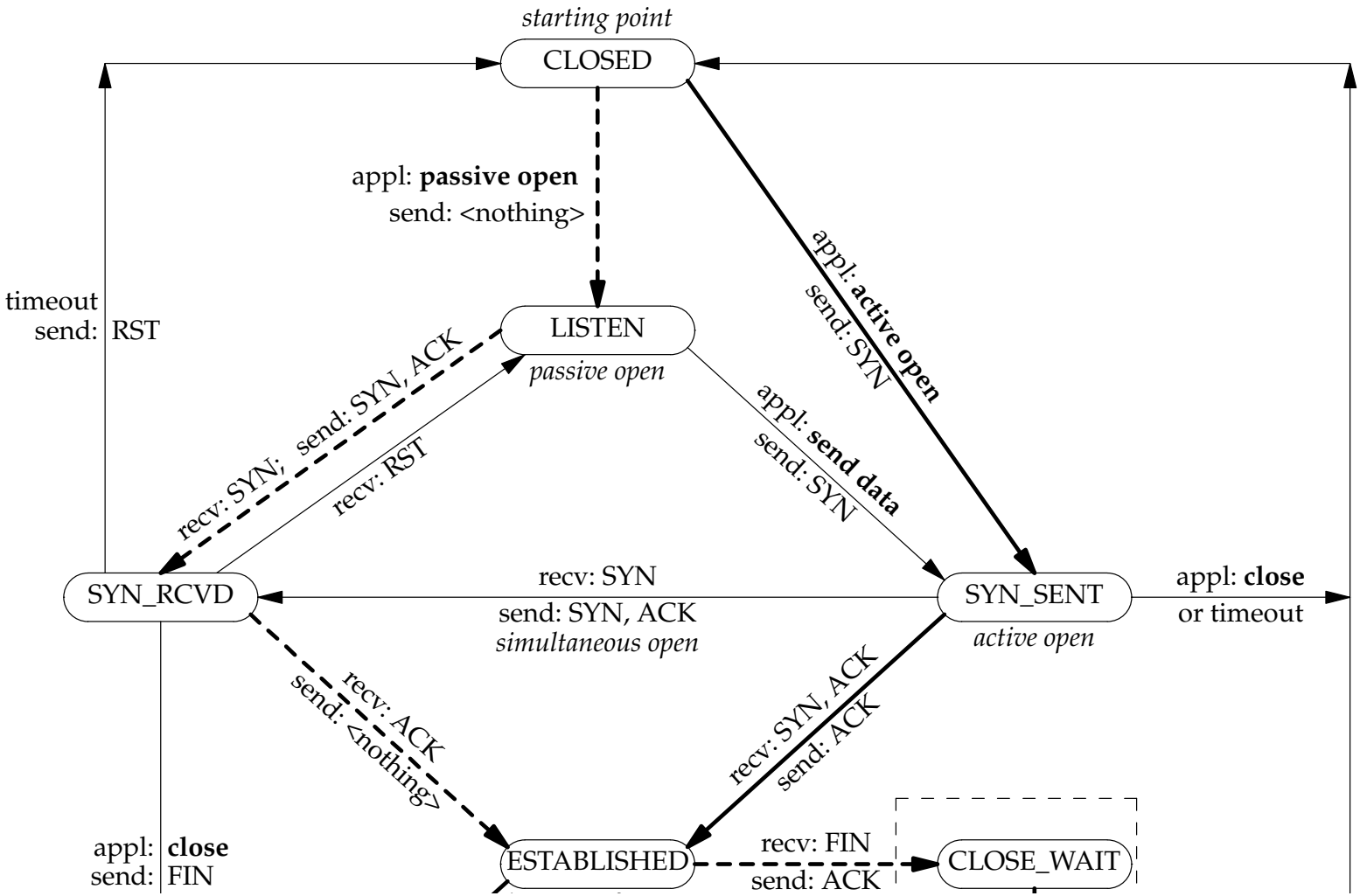
Sockets / Connections

- **Socket**: Ein **Socket** ist das Paar aus Port und Internet Adresse und beschreibt eindeutig einen Endpunkt einer Verbindung.
- **Connection**: Alle Daten, die eine Kommunikationsbeziehung (zwischen zwei Prozessen) beschreiben, z.B.:
  - Endpunkt(e) bzw. Socket(s)
  - Sequenznummern
  - Windowsize
  - TCP Optionen
  - Aktuelle Sequenznummern und Fenstergrößen

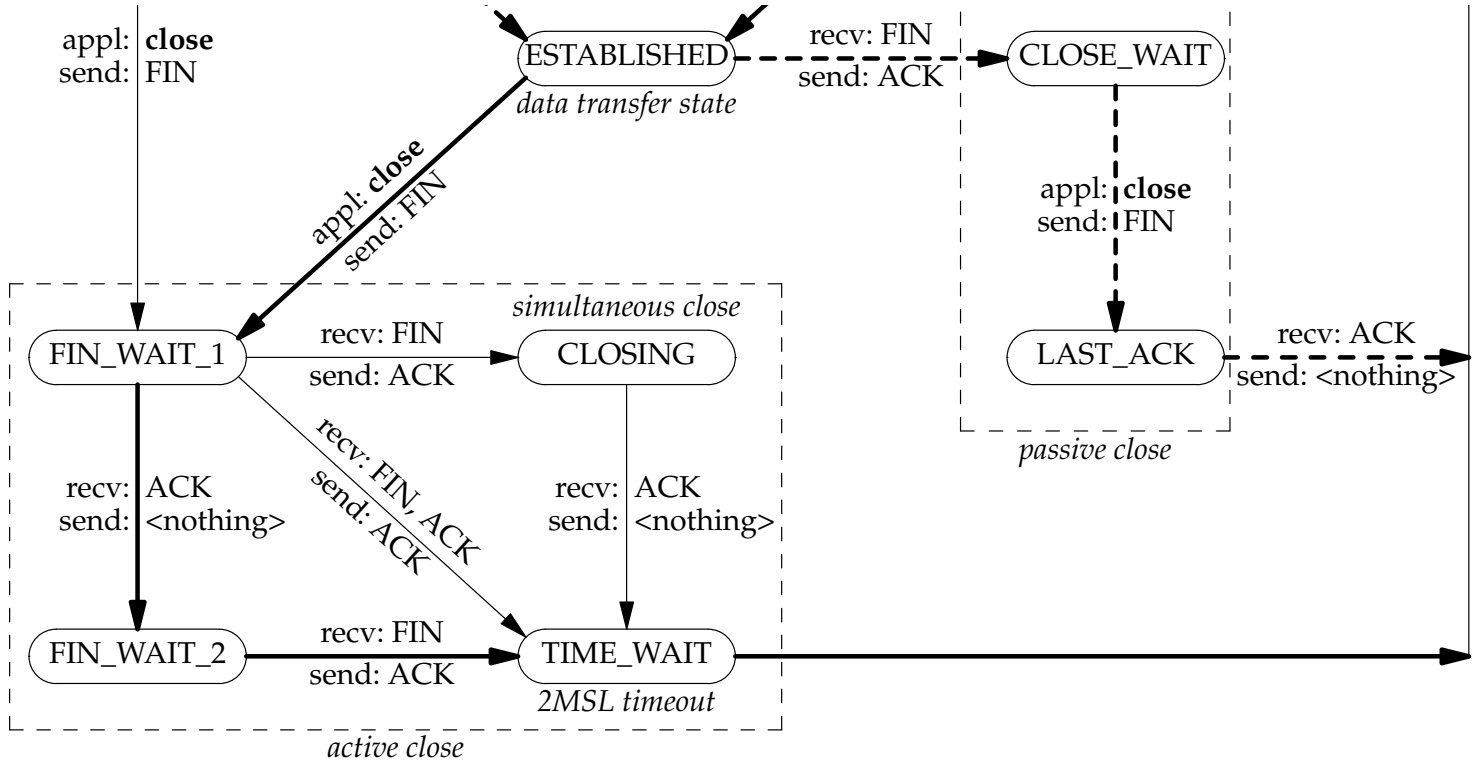
TCP Zustandsmodell



TCP Zustandsmodell



TCP Zustandsmodell



Verbindungsaufbau

Regeln zur Sequenznummerberechnung

- Segmente mit SYN oder FIN Flag erhöhen die Sequenznummer um 1 (sonst lässt sich ein ACK nicht zuordnen). Die Segmente enthalten keine weiteren Daten.
- Segmente mit Datenbytes erhöhen die Sequenznummer um deren Anzahl.

Verbindungsaufbau:

1. Client sendet SYN mit initialer Sequenznummer S_{client}
2. Server sendet SYN-ACK mit initialer Sequenznummer S_{server} und bestätigt $S_{client} + 1$.
3. Client bestätigt $S_{server} + 1$

Verbindungsabbau

Will eine Seite der anderen mitteilen, daß keine weiteren Daten folgen, sendet sie:

- FIN mit ihrer aktuellen Sequenznummer S, die letzte Sequenznummer der Gegenseite wird bestätigt.
- Die Gegenseite bestätigt $S + 1$.

Für den vollständigen Verbindungsabbau muß dieser Ablauf in beiden Richtungen stattfinden.

Will eine Seite die Kommunikation sofort vollständig beenden, kann sie anstatt des FIN ein RST schicken. Dies führt dazu, daß die Gegenseite jeden Kontext der Verbindung sofort verwirft. Der Empfang eines RST wird nicht bestätigt.

Datenübertragung mit TCP

TCP implementiert ein Schiebefensterprotokoll mit variablen Segmentgrößen.

- Der Datenstrom wird bei TCP in Segmente unterteilt.
- Wird der Empfang eines Segmentes nicht rechtzeitig bestätigt, wird es erneut übertragen.
- Wird ein korrekt übertragenes Segment empfangen, sendet der Empfänger ein ACK. Startet das Segment am Fensteranfang, ist ein **Delayed ACK** möglich.
- Ist die Prüfsumme eines Segments falsch, wird das Segment verworfen und auf erneute Übertragung gewartet.
- Mehrfach empfangene Daten werden verworfen.
- Über die Windows Size steuert der Empfänger die maximale Geschwindigkeit des Senders.
- Abhängig von Optionen beim Verbindungsaufbau werden NACK unterstützt (vgl. Selective Repeat ARQ).

Beispiel

Übertragung von 6 Bytes über TCP:

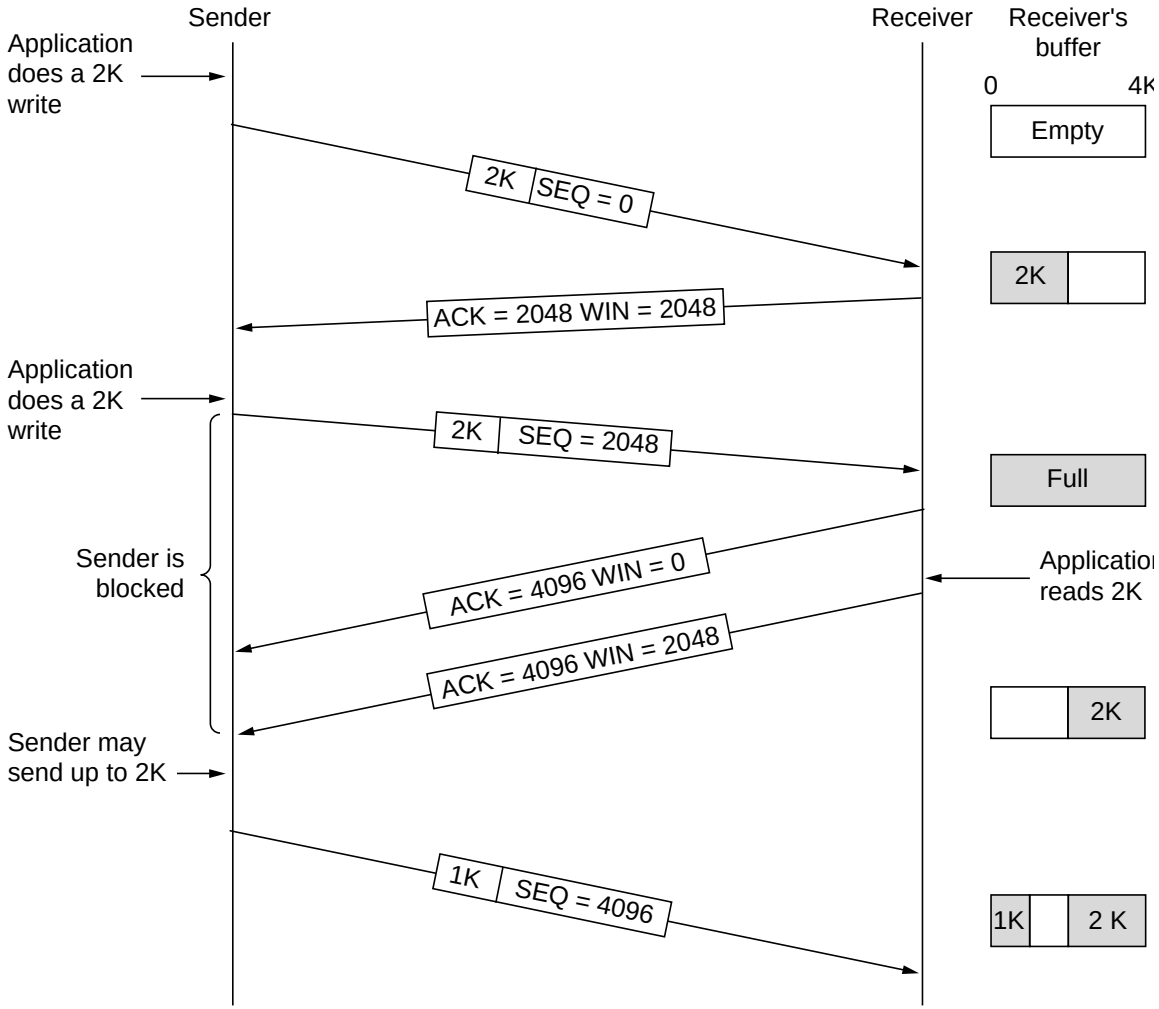
```
# tcpdump -n -i lo port 12345
127.0.0.1.51375 > 127.0.0.1.12345: S 235190854:235190854(0)
127.0.0.1.12345 > 127.0.0.1.51375: S 244140407:244140407(0)
    ack 235190855
127.0.0.1.51375 > 127.0.0.1.12345: . ack 1
127.0.0.1.51375 > 127.0.0.1.12345: P 1:7(6) ack 1
127.0.0.1.12345 > 127.0.0.1.51375: . ack 7
127.0.0.1.51375 > 127.0.0.1.12345: F 7:7(0) ack 1
127.0.0.1.12345 > 127.0.0.1.51375: F 1:1(0) ack 8
127.0.0.1.51375 > 127.0.0.1.12345: . ack 2
```

Detail siehe `http://www.tcpdump.org/tcpdump_man.html`

Wer den ersten FIN schickt, bleibt im `TIME_WAIT`

```
# netstat -an
tcp 0 0 127.0.0.1:51375 127.0.0.1:12345 TIME_WAIT
```

TCP Window Management



(c) Tanenbaum, Computer Networks

RST Generierung

1. Wenn ein Segment eintrifft, das nicht zu einer Verbindung gehört, z.B. SYN mit Zielport, der nicht zu einer Verbindung im Zustand LISTEN oder SYN_SENT gehört.
2. Ist die Verbindung in einem der Zustände LISTEN, SYN_SENT, SYN_RCVD und das empfangene Segment ist ein ACK mit ungültiger Sequenznummer.
3. Wird in anderen Zuständen der Verbindung ein Segment empfangen, das eine ungültige Sequenznummer oder ACK Sequenznummer trägt, wird kein RST generiert, nur ein ACK mit den aktuellen Sequenznummern.

Ein RST ist ein TCP Segment ohne Daten.

- Ist der RST Antwort auf ein ACK, ist die Sequenznummer die ACK Sequenznummer des Ausgangssegmentes.
- Andernfalls ist die Sequenznummer 0 und die ACK Sequenznummer berechnet sich wie üblich (RST-ACK).

RST Verarbeitung

- In jedem Zustand außer SYN_SENT muß die Sequenznummer des RST im Sendefenster liegen, sonst wird der RST verworfen.
- Im Zustand SYN_SENT wird der RST-ACK verworfen, wenn die ACK Sequenznummer die Sequenznummer im SYN nicht bestätigt.

Wird der RST verarbeitet, ergeben sich folgende Zustandsänderungen:

- Im Zustand LISTEN wird der RST verworfen.
- Im Zustand SYN_RCVD kehrt der Server zum Zustand LISTEN zurück, sofern er vorher in LISTEN war.
- In allen anderen Fällen geht er in der Zustand CLOSED über.

Zeitverhalten

- Wird der initiale SYN nicht beantwortet, so unternimmt der Sender eine (konfigurierbare) Anzahl von Versuchen, bevor er der Anwendungsschicht eine Fehlermeldung schickt.
- Wird der SYN-ACK nicht beantwortet, so unternimmt der Sender eine (konfigurierbare) Anzahl von Versuchen, bevor er den Kontext wieder freigibt.
- Die Zeit zwischen erneuten Übertragungen nicht bestätigter Segmente ist implementationsabhängig, RFC793 gibt ein Beispiel an, das die Laufzeit der Daten bis zur Gegenseite berücksichtigt.
- Gibt ein Segment eine Window Size von 0 Byte an, fragt die Gegenstelle periodisch, ob wieder Daten gesendet werden können (Persist Timer).

TCP Keepalive (vgl. RFC1122)

Ist eine Connection im Zustand ESTABLISHED, fließen keine Segmente zwischen den Endpunkten, wenn keine Daten zu übertragen sind. Dies kann zu Problemen führen:

- Wird die Verbindung zwischen den Endpunkten getrennt, behalten beide Knoten den Verbindungskontext auf unbestimmte Zeit.
- Wird einer der Knoten ausgeschaltet und neu gestartet, wird die Gegenseite davon nichts merken, bis sie wieder Daten übertragen muß.

Daher bieten TCP Implementationen einen Keepalive Timer.

- Nach 2 Stunden Inaktivität wird ein ACK mit den aktuellen Sequenznummern geschickt und im Normalfall mit einem entsprechenden ACK beantwortet
- Wird der ACK nicht beantwortet, werden weitere Tests gesendet.