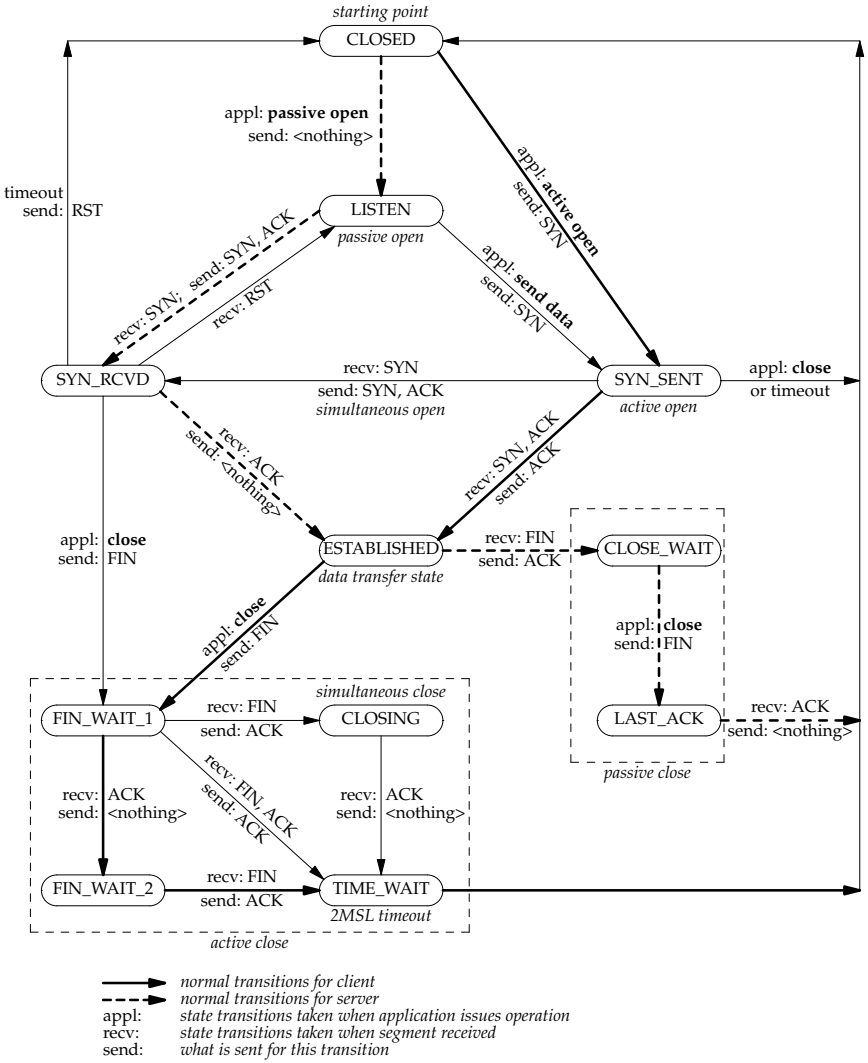
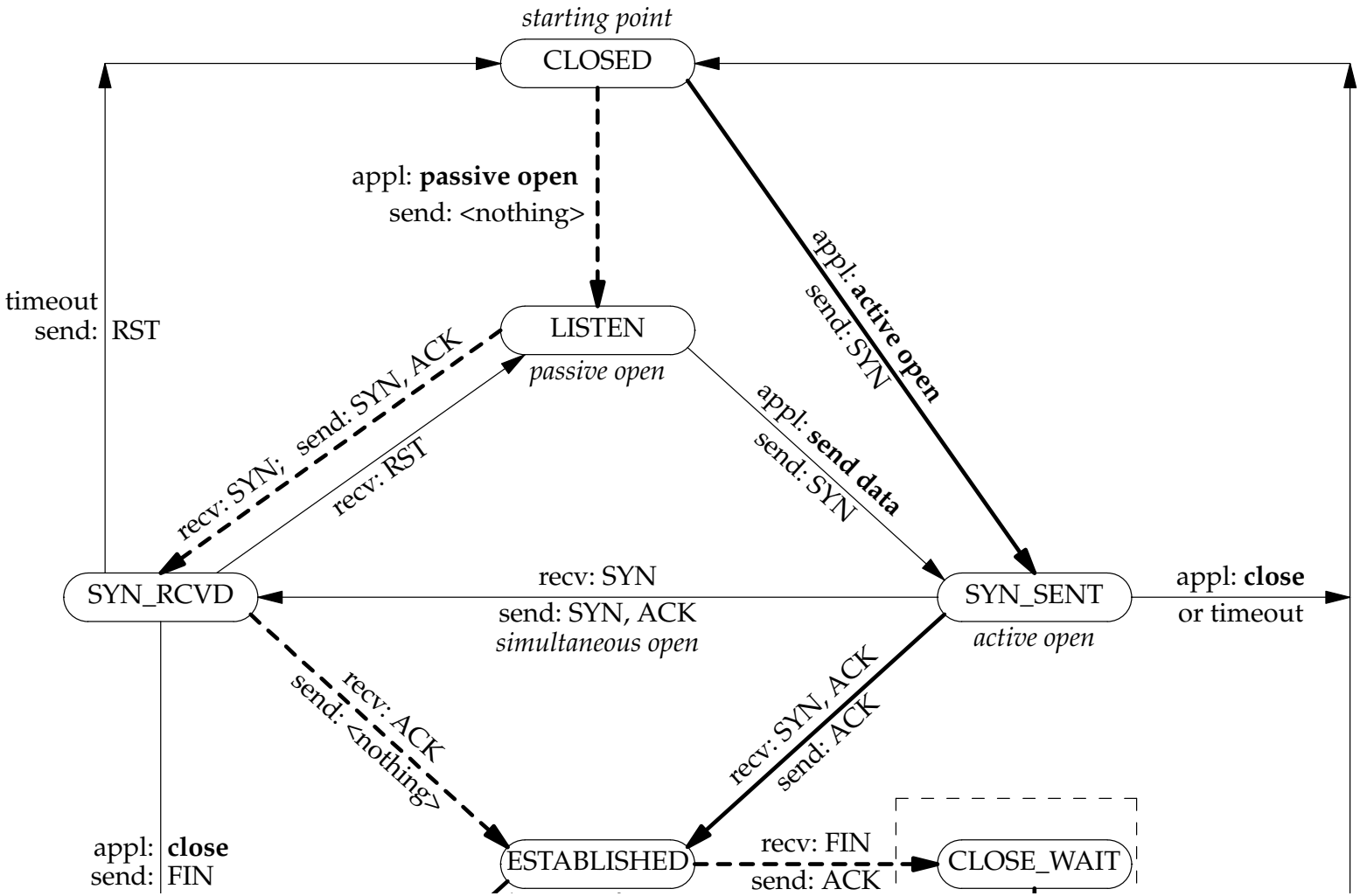
Transportschicht
(Fortsetzung TCP)

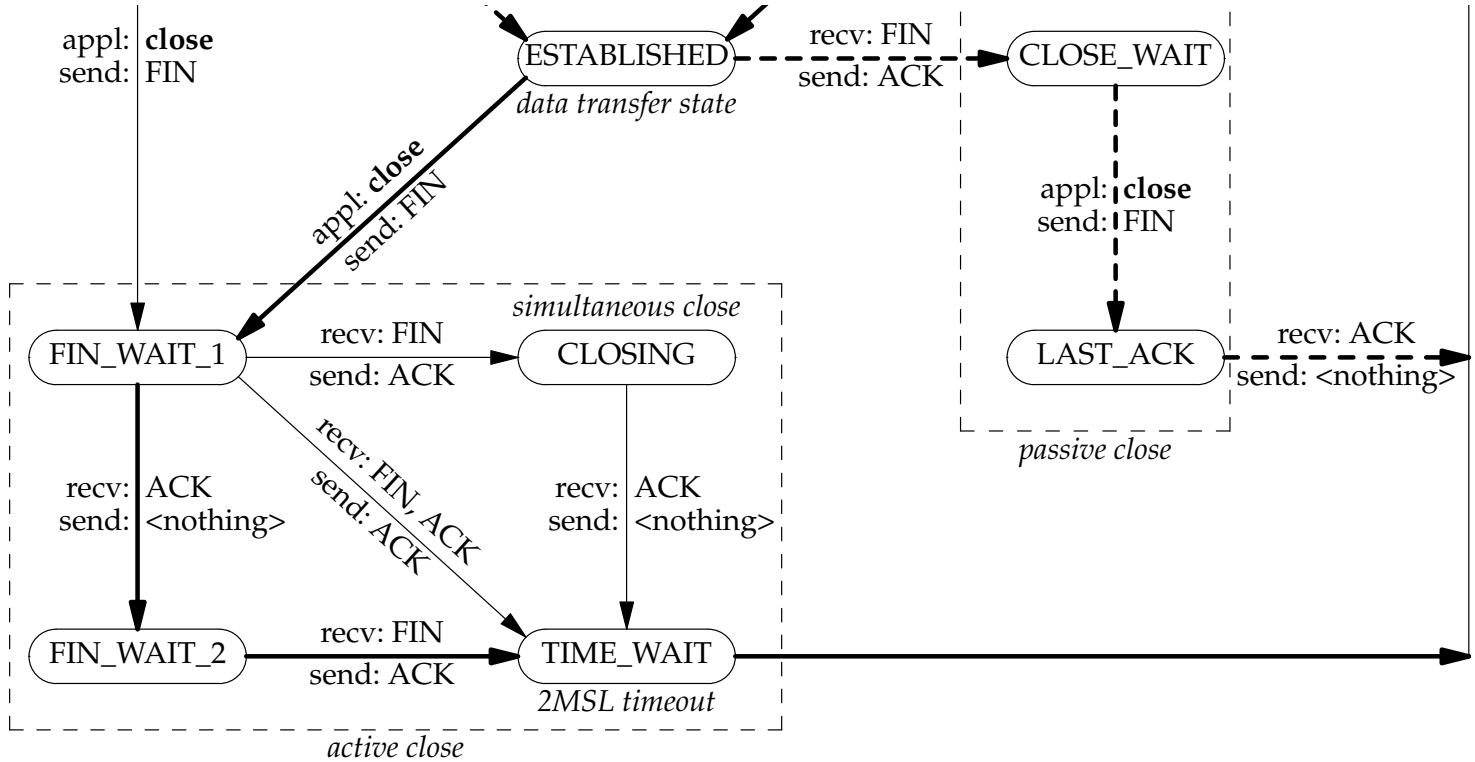
TCP Zustandsmodell



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:

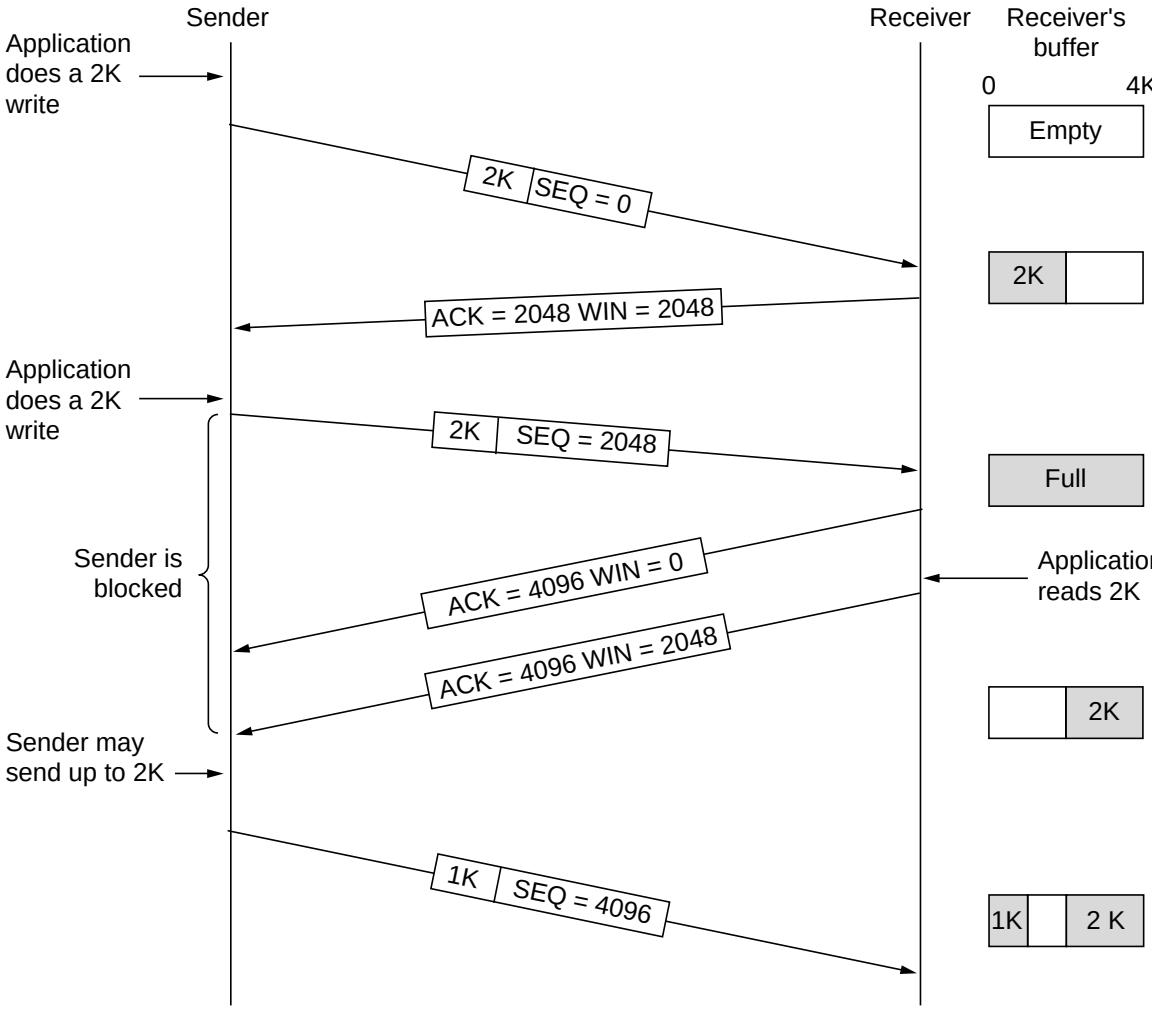
```
# tcpdump -n -i lo port 12345
127.0.0.1.51375 > 127.0.0.1.12345: S 235190854:235190854(0)
127.0.0.1.12345 > 127.0.0.1.51375: S 244140407:244140407(0)
    ack 235190855
127.0.0.1.51375 > 127.0.0.1.12345: . ack 1
127.0.0.1.51375 > 127.0.0.1.12345: P 1:7(6) ack 1
127.0.0.1.12345 > 127.0.0.1.51375: .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

Nagle Algorithmus

TCP dient zur Übertragung eines kontinuierlichen Datenstroms. Werden die Zeichen von der Anwendungsschicht langsam eingeliefert, führt eine einfache Segmentierung zu Segmenten mit genau einem Byte Nutzdaten bei mindestens 40 Byte Header. Dies kann mit dem Nagle Algorithmus verhindert werden:

- Sind alle Daten im Sendefenster bestätigt, sende sofort das nächste Segment.
- Sind genug Daten zu senden (MSS erreicht), sende sofort ein weiteres Segment.
- Andernfalls warte, bis weitere Daten zu senden sind oder alle bereits gesendeten Segmente bestätigt sind.

Bemerkung: Benutzt die Gegenseite Delayed ACKs, kann der Nagle Algorithmus zu merklichen Verzögerungen der Übertragung führen!

TCP Optionen

Die Kodierung von TCP Optionen entspricht dem Verfahren im IP Header. Entsprechend sind sie wieder mit folgenden zwei Ausnahmen Type-Länge-Wert kodiert:

0 End of Options (1 Byte, kein Längenfeld)

1 No Operation (1 Byte, kein Längenfeld)

2 Maximum Segment Size (Länge 4, 2 Byte MSS)

3 Window Scale Factor, RFC1323 (Länge 3, 1 Byte Shift)

4 Selective ACK Permitted, RFC2018 (Länge 2)

5 Selective ACK, RFC2018

8 Timestamp, RFC1323 (Länge 10, 8 Byte für 2 Zähler)

Beispiel

SYN Paket eines Linux Hosts (nur TCP Rahmen):

Bytes	Interpretation
e65c 3039	Quellport 58972, Zielport 12345
d651 05e7	Initiale Sequenznummer 3595634151
0000 0000	ACK Sequenznummer 0
a002	Header Länge 40 Bytes, SYN
16d0	Window Size 5840 (mal Faktor)
4135 0000	Prüfsumme, URG nicht gesetzt
0204 05b4	MSS 1460 Bytes
0402	Selective ACK
080a 000e 715c 0000 0000	Timestamp TSval 946524
01	No Option
03 0306	Window Scale, Shift 6 (d.h. Faktor 64)

MSS Option, RFC793

Die MSS Option dient dazu, die Wahrscheinlichkeit zu senken, daß ein TCP Segment von der Vermittlungsschicht fragmentiert werden muß.

- Die MSS Option ist nur im SYN / SYN-ACK erlaubt.
- Üblicherweise berechnen Hosts die MSS als MTU des Interfaces, über das die Verbindung aufgebaut werden soll, abzüglich der minimalen Headerlänge (40 Bytes bei IPv4).
- Wird MSS nicht gesendet, wird der Default 536 angenommen.
- Beide Seiten verwenden das Minimum der eigenen MSS und der MSS der Gegenseite zur Segmentierung.
- Die Segmentierung kann bei bestehender Verbindung durch ICMP Destination Unreachable, Fragmentation Needed verkleinert werden (Path MTU Discovery).

Timestamp Option, RFC1323

- Die bisherige Zeit bis zur Antwort auf Segmente durch die Gegenseite ist die wichtigste Größe bei der Entscheidung, ob ein Segment erneut übertragen werden muß oder noch eine Antwort zu erwarten ist.
- Da es bei TCP kein einfaches Request-Response Modell gibt (vgl. ICMP Echo Request), sondern ein ACK mehrere Segmente bestätigen kann, werden geschätzte Antwortzeiten bei großen Empfangsfenstern schnell ungenau.
- Die Timestamp Option wird benutzt, wenn im SYN und SYN-ACK die Option gesetzt ist, also beide Hosts die Option verstehen.

Timestamp Algorithmus

- Die Option enthält 4 Byte TS Value (TSval) und 4 Byte TS Echo Reply (TSecr)
- TSecr ist nur gesetzt, wenn das ACK Flag gesetzt ist, sonst 0

Folgender Algorithmus wird von TCP verwendet, um TSecr zu bestimmen:

1. TS.Recent speichert den zu sendenden TSval, Last.ACK.sent speichert die ACK Sequenznummer des letzten gesendeten Segmentes.
2. Ist Last.ACK.sent innerhalb eines empfangenen Segmentes, setze TS.Recent auf den Zeitstempel im Segment.
3. Wird ein Segment mit Timestamp Option gesendet, setzte TSecr auf TS.Recent.

Window Scale Option, RFC1323

Bei schnellen Anbindungen mit hoher Latenzzeit ist das 64KByte Window nicht ausreichend, um hohen Datendurchsatz zu erzielen.

- Der 1 Byte Shift Count der Window Scale Option gibt an, um wie viele Bit der Wert im Window Feld des TCP Rahmens nach links verschoben werden muß, um den Wert in Byte zu erhalten.
- Die Window Scale Option wird nur genutzt, wenn beide Hosts im SYN bzw. SYN-ACK ihren Shift Count angeben.
- Der maximale Shift Count ist 14, d.h. das Window ist maximal $65535 \cdot 2^{14}$ Byte groß.

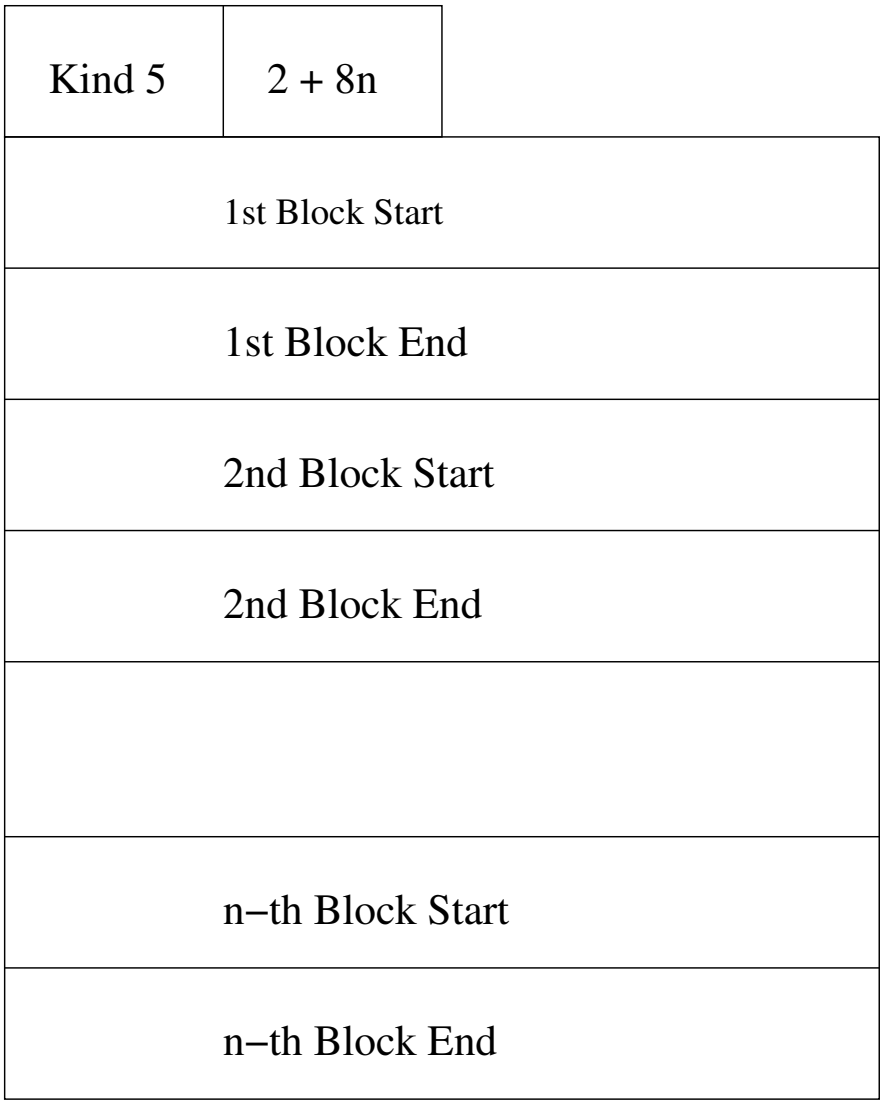
Selective ACK (SACK), RFC2018

Geht ein Segment bei der Übertragung verloren, müssen in der Regel die Daten vom letzten bestätigten Byte (ACK Sequenznummer) startend ggfs. in anderer Segmentierung erneut übertragen werden (vgl. Go-Back-n ARQ).

Die TCP Selective ACK Option erlaubt es, Bereiche im Datenstrom zu bestätigen und damit implizit fehlende Bereiche erneut anzufordern (vgl. Selective-Repeat ARQ).

SACK wird nur verwendet, wenn beide Hosts es im SYN / SYN-ACK anbieten, d.h. Option 4 SACK Permitted senden.

Selective ACK Format



Selective ACK Felder

- Im Selective ACK werden erfolgreich empfangene Segmente bestätigt.
- Jeder Block im Selective ACK ist zusammenhängend und isoliert, d.h. das Byte vor dem übertragenen Anfang des Blocks ist noch nicht bestätigt, genauso wie das Byte, das durch das Ende des Blocks identifiziert wird (halboffenes Intervall).
- Die maximale Anzahl Blöcke ist 4, falls keine weiteren Optionen (speziell Timestamp) verwendet werden.
- Selective ACK ist weit verbreitet (Windows98, Solaris 8).

Duplicate Selective ACK (DSACK, vgl. RFC2883)

Um dem Sender Rückmeldung über unnötig wiederholte Segmente zu geben, ist der Selective ACK in RFC2883 erweitert worden. Die Optionen und deren Format ist aber unverändert.

- Nach RFC2018 müssen die Blöcke im SACK disjunkt sein und die Sequenznummern hinter ($mod 2^{32}$) der ACK Sequenznummer liegen. Diese Beschränkung wird aufgehoben.
- Ein Block kann nach RFC2883 wie in RFC2018 einen noch nicht bestätigten Bereich im Datenstrom beschreiben.
- Liegt ein Block vor der ACK Sequenznummer oder innerhalb eines anderen Blocks, bezeichnet er einen mehrfach empfangenen Abschnitt.

TCP Flußkontrolle (vgl. RFC3390)

Bei großen Sende- und Empfangsfenstern kann es leicht zur Überlastung des Netzes kommen. Deshalb werden vier Algorithmen zur Flußkontrolle angewendet (Van Jacobson):

1. TCP Slow Start
2. Congestion Avoidance
3. Fast Retransmits
4. Fast Recovery

Algorithmen 1 und 2 sind bei TCP verbindlich.

Prinzipiell existieren zwei möglich Engpässe

- Kapazität des Netzwerks
- Kapazität des Empfängers

TCP Slow Start / Congestion Avoidance

Sei *MSS* die Maximum Segment Size der Gegenseite, *WINS* deren Windowsize und *FS* die Anzahl nicht bestätigter Bytes.

- Der Kontext einer Verbindung hält die Werte Congestion Window (*CWND*) und Slow Start Threshold (*S*), er sendet maximal min(*CWND*, *WINS*) Bytes ohne ACK.
- Anfangs ist *CWND* maximal so groß wie 4 *MSS* (oft $CWND = MSS$), *S* z.B. *WINS*
- Ist $CWND \leq S$ und wird ein Segment bestätigt, setze $CWDN+ = MSS$ (Slow Start)
- Ist $CWND > S$, inkrementiere für jede Round Trip Time *CMWD* um maximal *MSS* (Congestion Avoidance)
- Gehen Segmente verloren (Timeout oder doppelter ACK), dann setze $S := \max(2 \cdot MSS, FS/2)$ und $CWND := MSS$.

TCP Slow Start / Congestion Avoidance (2)

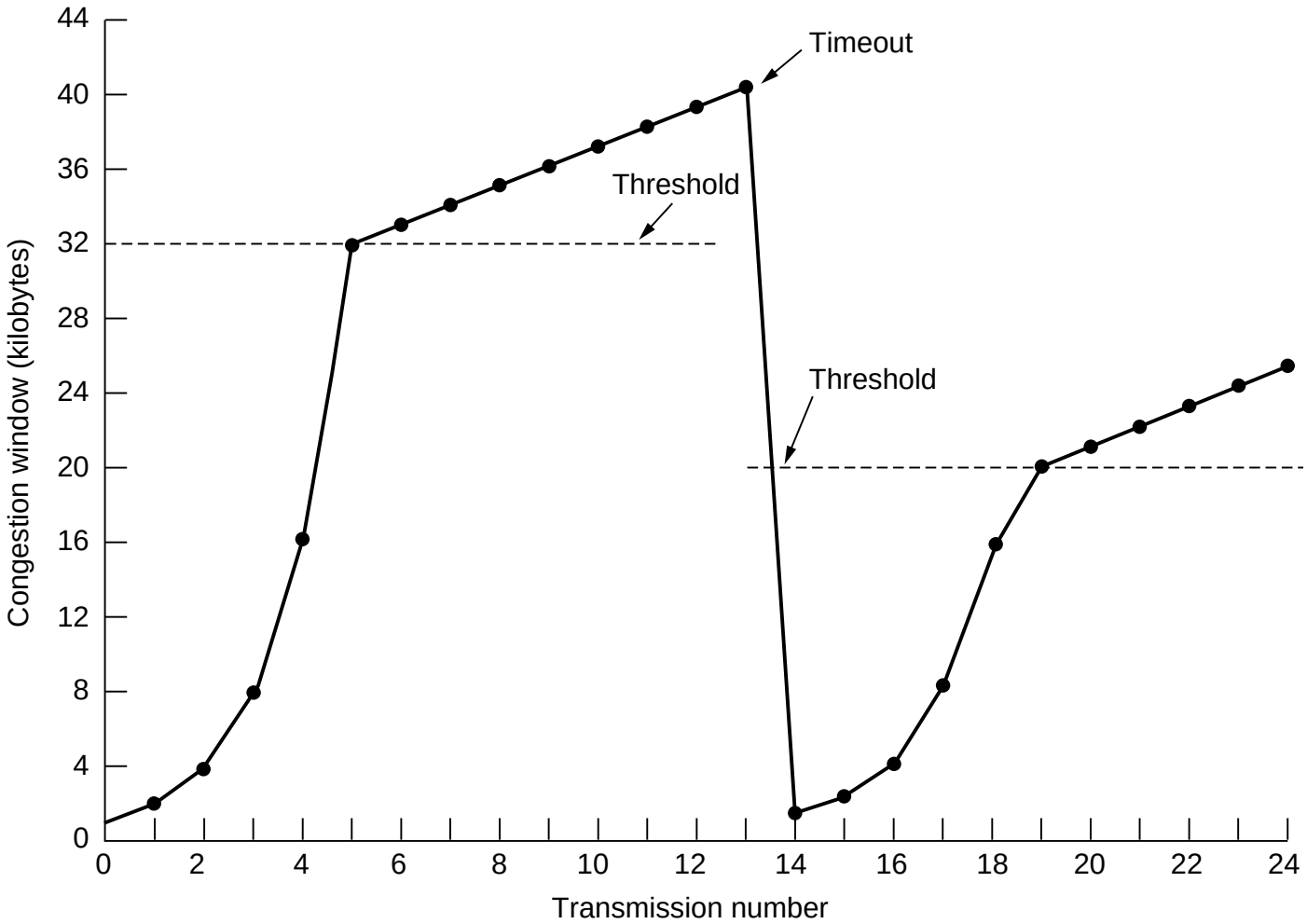
Slow Start

- Start mit sehr kleinem *CWND*
- exponentielles Wachstum von *CWND*, Verdopplung bei jeder Übertragung
- *CWND* wächst bis Slow Start Threshold *S* erreicht

Congestion Avoidance

- Wenn Threshold *S* mit Slow Start erreicht wurde, langsames lineares Wachstum bis entweder
- die Window Size *WINS* des Empfängers erreicht wurde
  $\rightarrow$ Empfängerkapazität erreicht, *CWND* bleibt konstant
- oder Fehler auftreten (Timeout, doppeltes ACK)
  $\rightarrow$ Netzwerkkapazität erreicht, *CWND* wird zurückgesetzt

TCP Slow Start / Congestion Avoidance (3)



(c) Tanenbaum, Computer Networks

Fast Retransmit / Fast Recovery

Bei großen Empfangsfenstern wird der Sender in der Regel durch doppelte ACK auf verlorene Segmente aufmerksam. Diese können auch durch Umordnung der Segmente entstanden sein.

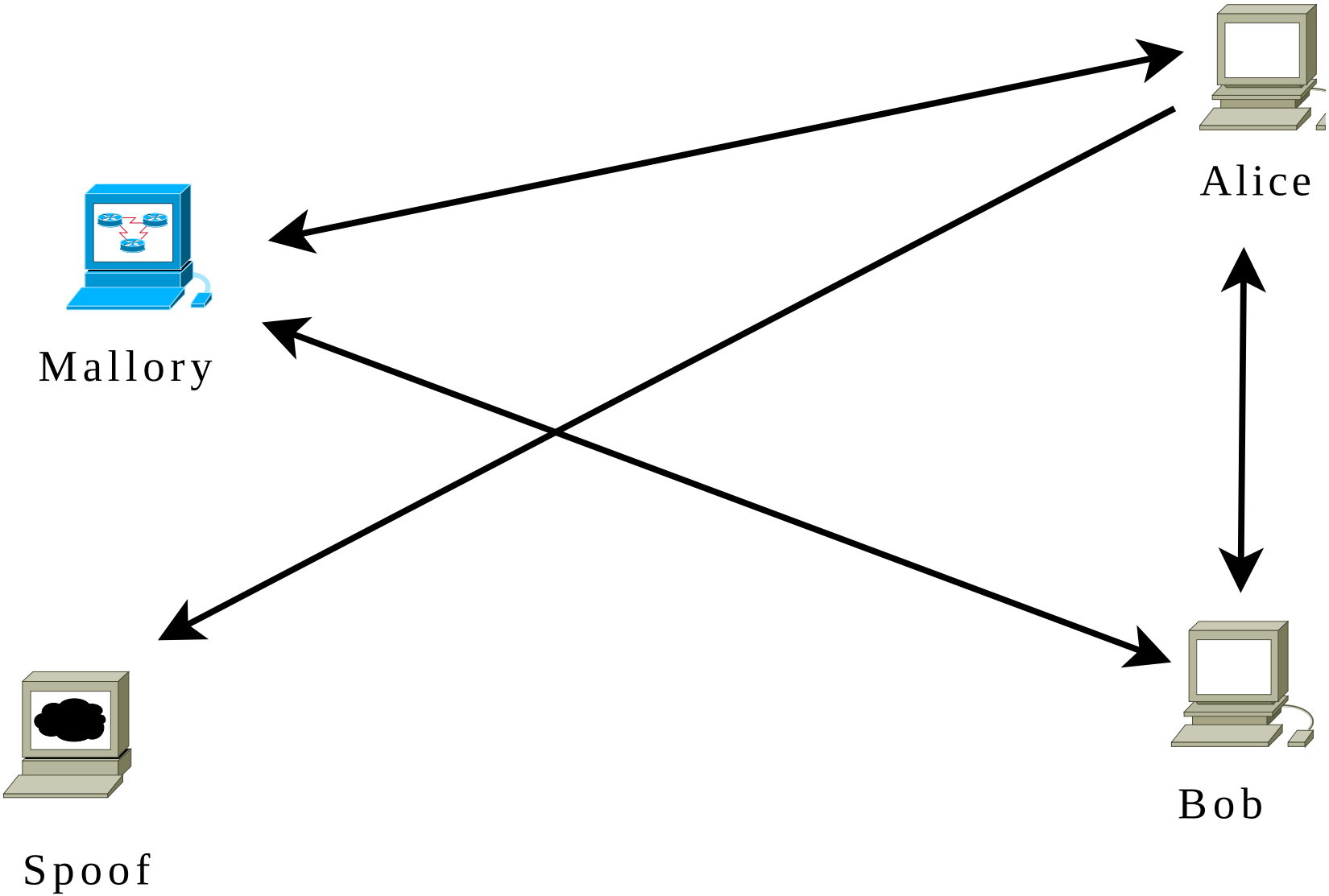
- Wird der dritte ACK mit gleicher ACK Sequenznummer empfangen, setze $S := \min(WINS, CWND)/2$
  Sende das (vermutlich) fehlende Segment
  Setze $CWDN := S + 3 \cdot MSS$
- Für jedes weitere ACK mit derselben Sequenznummer, setze $CWND+ = 1$ und sende ein weiteres Segment.
- Beim nächsten neuen ACK setze $CWND := S$

Die Mitnick Attacke

Einleitung

- Angriff von Kevin Mitnick auf Rechner von Tsutomu Shimomura Weihnachten 1994.
- Tsutomu Shimomura war Spezialist für Rechnersicherheit bei SUN Microsystems
- John Markoff, damals Journalist der New York Times hat den Angriff bekannt gemacht.
- Die Details zum Angriff sind z.B. hier dokumentiert `http://www.gulker.com/ra/hack/`

Beteiligte Systeme



Finger und Stealth SYN Scan

- **mallory**# finger -l **@alice**
  Liste der Benutzer auf **alice**
- **mallory**# finger -l **@bob**
  Liste der Benutzer auf **bob**
- **mallory**# finger -l [2]**root@bob**
  Details einer Session des Benutzers **root** auf **bob**, z.B.:
  Last login Thu Aug 5 09:55 (PDT) on ttyx1 from **alice**
- 1 mallory.51916 > alice.login: S
  2 alice.login > mallory.51916: S ack
  Test, ob auf **alice** der `rlogin` Dienst aktiviert ist

TCP Sequenznummern

- 1 mallory.1000 > bob.shell: S 1382726990(0)
  2 bob.shell > mallory.1000: S 2021824000(0) ack 1382726991
  3 mallory.1000 > bob.shell: R 1382726991(0)
- 1 mallory.999 > bob.shell: S 1382726991(0)
  2 bob.shell > mallory.999: S 2021952000(0) ack 1382726992
  3 mallory.999 > bob.shell: R 1382726992(0)
- 1 mallory.998 > bob.shell: S 1382726992(0)
  2 bob.shell > mallory.998: S 2022080000(0) ack 1382726993
  3 mallory.998 > bob.shell: R 1382726993(0)
- Die initialen Sequenznummern von **bob** wachsen pro Verbindungsaufbau um 128000

SYN Flood Angriff auf **alice.login**

- 1 spoof.600 > alice.login: S 1382726960(0)
  2 alice.login > spoof.600: S 1022085000(0) ack 1382726961
  3 spoof.601 > alice.login: S 1382726961(0)
  4 alice.login > spoof.601: S 1022086000(0) ack 1382726962
  ...
  15 spoof.607 > alice.login: S 1382726967
  16 alice.login > spoof.607: S 1022092000(0) ack 1382726968
  Bis hier wurden von **alice** SYN ACK erzeugt
- 17 spoof.608 > alice.login: S 1382726968(0)
  18 spoof.609 > alice.login: S 1382726969(0)

  Die Listen-Queue von **alice** ist voll, SYN ACK Pakete werden verworfen, sofern sie nicht ein gespeichertes SYN beantworten.

IP Spoofing Angriff auf **bob.login**

- **mallory** schickt ein SYN Paket mit falscher Quell-IP an **bob**
  1 alice.login > bob.shell: S 1382727010:1382727010(0)
- **bob** antwortet **alice** 2 bob.shell > alice.login: S 2022208000(0) ack 1382727011
- **alice** verwirft den SYN ACK, sendet kein RST
- **mallory** sieht den SYN ACK nicht, kann den Inhalt aber vorhersagen
- **mallory** sendet ACK, damit ist die Verbindung ESTABLISHED
  3 alice.login > bob.shell: . ack 2022208001
- **mallory** sendet Schadcode
  Daten: ..root.root.echo + + » /.rhosts
  5 alice.login > bob.shell: F 1382727044(0) ack 2022208001

Konsequenzen

- rlogin -l root bob funktioniert ohne Authentifizierung aus dem Internet
- Kevin Mitnick wurde später zu 46 Monaten Haft verurteilt
- Moderne Betriebssysteme benutzen zufällige initiale Sequenznummern
- rlogin/rsh/rexec sind durch ssh weitgehend abgelöst
- SYN Flooding bleibt weiter ein Problem, SYN Cookies nur ein Behelf