Internet Protokoll Version 6 (IPv6)

Zustand und Probleme von IPv4

- Router im Internet haben > 200000 Einträge in der Routingtabelle
- IP Adressen sind eine extrem knappe Resource, siehe z.B. http://www.potaroo.net/tools/ipv4/index.html (mit 32 Bit sind $2^{32} = 4294967296 \approx 4 \cdot 10^9$ Adressen möglich, aber durch den hierarchischen Aufbau, reserviert Adressbereiche,... ist der Nutzungsgrad nicht hoch)
- Temporäre Adressvergabe mittels DHCP, private Adressbereiche und NAT helfen, die vorhandenen Adressen effizient zu nutzen.
- Viele Dienste sind nur mit Hilfe neuer und komplizierter Protokolle möglich, z.B.:
  - Multimedia Messaging Service (MMS) (vgl. www.openmobilealliance.org)
  - Mobile E-Mail
  - ...

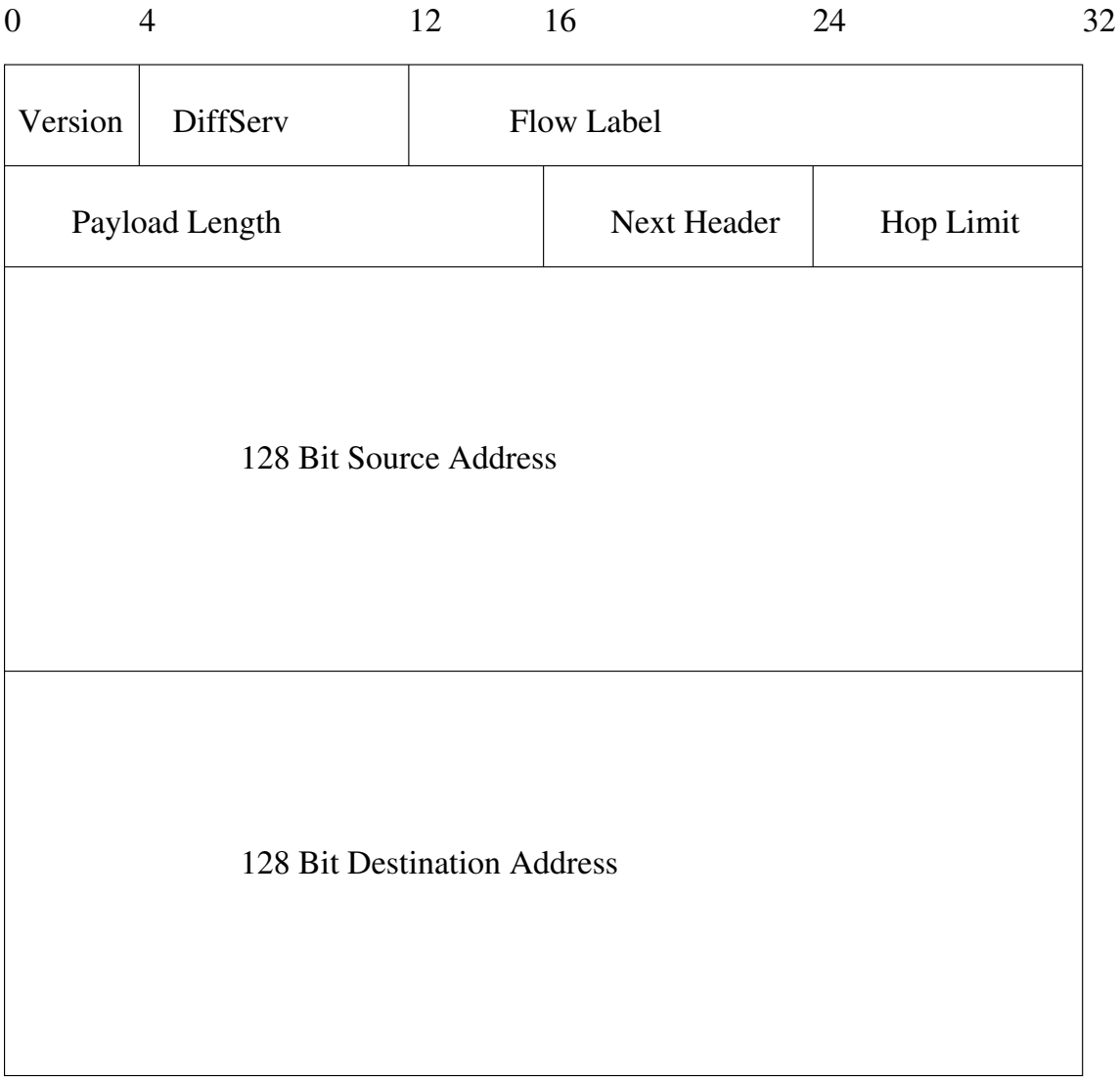
Ziele und Neuerungen von IPv6

- Massive Vergrößerung des Adressraums, sodass selbst bei ineffizienter Nutzung immer genug Adressen vorhanden sind.
  - → 128Bit für Adressen, nicht 32Bit
  - → 128Bit ermöglicht $2^{128} \approx 3 \cdot 10^{38}$ Adressen
  - → $\approx 7 \cdot 10^{23}$ Adressen pro Quadratmeter Erdoberfläche
  - → Bei "ineffizientester" Nutzung $\approx 1000/m^2$ (vgl. RFC3194)
- Verkleinerung der Routingtabellen
  - → Hierarchische Netzstruktur durch gezielte Adresszuweisung
- Vereinfachung des Protokolls um Verarbeitung in Routern zu beschleunigen
  - → Minimaler Basisheader
  - → Keine Checksum mehr im Header
- Koexistenz mit IPv4 und zukünftige Erweiterbarkeit
  - → Flexible Erweiterungsmöglichkeiten des Basisheaders

Ziele und Neuerungen von IPv6 (2)

- Besser Anpassungen an verschiedene Serviceklassen (Real-Time, ...)
  - → Differentiated Services sind Standard
- Automatische Konfiguration ist ohne Zusatzprotokolle möglich.
- Host sollen ihren Standort verändern können ohne Adressänderung (Roaming)
  - → Dienste für mobile Terminals werden unterstützt
- Verbesserung bei der Sicherheit (Authentifizierung und Vertraulichkeit)
  - → Sicherheitsdienste werden von der Vermittlungsschicht transparent angeboten.
- Verbesserte Integration von Multicast Diensten

IPv6 Header


0
4
12
16
24
32
Version
DiffServ
Flow Label
Payload Length
Next Header
Hop Limit
128 Bit Source Address
128 Bit Destination Address

IPv6 Header

- **Version:** 4 Bit Version, 6 für IPv6
- **DiffServ:** Differentiated Services Kennung, entspricht IPv4
- **Flow Label:** Kennung zusammengehöriger Pakete, vgl. RFC3697
- **Payload Length:** 16 Bit Länge der Daten in Byte (d.h. exklusive des 40 Byte Basisheaders)
- **Next Header:** Typ des nächsten Headers, entweder Protocol (RFC1700) wie in IPv4, oder einer der standardisierten Erweiterungsheader
- **Hop Limit:** Max. Anzahl Hops, vgl. IPv4 TTL
- **Source Address:** 128 Bit Quelladresse
- **Destination Address:** 128 Bit Zieladresse

IPv6 Adressen (vgl. RFC3513)

Adressen identifizieren Schnittstellen (Interfaces, vgl. RFC2460, Section 2), mit denen Knoten mit dem Netzwerk verbunden sind.

In IPv6 werden drei Typen von Adressen unterschieden:

- **Unicast:** Adresse einer Schnittstelle, Pakete werden zu genau dieser Schnittstelle weitergeleitet.
- **Anycast:** Adresse für eine Gruppe von Schnittstellen, ein Paket wird zu einer dieser Schnittstellen weitergeleitet.
- **Multicast:** Adresse für eine Gruppe von Schnittstellen, Pakete werden zu allen Schnittstellen der Gruppe ausgeliefert.

Schreibweise für IPv6 Adressen

- Die 16 Byte einer Adresse in Network Byte Order werden geschrieben als 8 Segmente von je 2 Byte in hexadezimaler Darstellung, durch Doppelpunkte getrennt.
- Führende Nullen in Segmenten können weggelassen werden.
  Beispiel: 00 01 02 03 04 05 06 07 18 19 1A 1B 1C 1D 1E 1F
  geschrieben: 1:203:405:607:1819:1A1B:1C1D:1E1F
- Eine Gruppe von aufeinander folgenden, leeren Segmenten kann durch zwei Doppelpunkte abgekürzt werden:
  Beispiel: 00 01 00 00 00 00 00 00 00 00 1A 1B 1C 1D 1E 1F
  geschrieben: 1::1A1B:1C1D:1E1F
- Alternative Schreibweise: Die letzten 4 Byte können als “Dotted Notation” geschrieben werden:
  Beispiel: 00 01 00 00 00 00 00 00 00 00 1A 1B 01 02 03 04
  geschrieben: 1::1A1B:1.2.3.4

Schreibweise für Netzwerke

- Netzwerkpräfixe werden in CIDR Notation (Adresse/Länge) geschrieben.
- Segmente, die außerhalb der Maske liegen, brauchen nicht aufgeführt zu werden.
  Beispiel: das 60 Bit Präfix `12AB 0000 0000 CD3` kann geschrieben werden als:
  - 12AB:0:0:CD30/60
  - 12AB:0:0:CD30:0:0:0:0/60
  - 12AB:0:0:CD30::/60
- Alle möglichen Mehrdeutigkeiten sind **nicht** erlaubt, z.B.:
  - 12AB:0:0:CD3/60, Segment vier kann auch 0CD3 sein.
  - 12AB::CD30/60, Würde interpretiert als 12AB 0000 0000 000

Adresstypen

Die unterschiedlichen Adresstypen und Adressbereiche sind Subnetze des IPv6 Adressraumes:

Type	Prefix
Unspecified	::/128
Loopback	::1/128
Multicast/Anycast	FF00::/8
Link-local unicast	FE80::/10
Site-local unicast	FEC0::/10
Global unicast	everything else

Aufbau von Global Unicast Adressen

Adressen, die nicht mit ::/12 beginnen, enden in einer 64 Bit Schnittstellenadresse gemäß IEEE EUI-64.

Beispiel: Die 48 Bit Ethernetadresse einer Schnittstelle wird zu einer 64 Bit Schnittstellenadresse expandiert, indem Bit 1 des ersten Bytes auf 1 gesetzt wird (ist in der Ethernetadresse immer 0, da OUI), Byte 2 und 3 unverändert übernommen werden, dann wird 0xFFFE eingefügt, dann folgen die letzten 3 Byte der Ethernetadresse:

Beispiel: Aus der Ethernetadresse 08:00:46:9E:92:0E wird . . .:0A00:46FF:FE9E:920E

Für andere Interfacetypen finden sich entsprechende Abbildungsregeln in den RFCs, die die Übertragung von IPv6 über den Link spezifizieren.

Mapping von IPv4 Adressen

Schnittstellen, die sowohl IPv4 als auf IPv6 bedienen können, können spezielle IPv6 Adressen bekommen (**IPv4 compatible address**), die die IPv4 Adresse als letzte 4 Bytes enthalten. Diese Adressen werden heute kaum noch verwendet:

Beispiel: IPv6 fähige Schnittstelle mit IPv4 Adresse 1.2.3.4 hat IPv6 Adresse ::1.2.3.4

Soll eine Anwendung sowohl mit IPv4 als auch IPv6 funktionieren, wird oft IPv6 als Obermenge verwendet. Adressen einer Schnittstelle werden dann als sogenannte **IPv4 mapped address** vom Stack geliefert, wenn es sich um eine IPv4 Adresse handelt.

Beispiel: Schnittstelle mit IPv4 Adresse 1.2.3.4 hat IPv4 mapped address ::FFFF:1.2.3.4

Diese Adressen unterliegen den IPv4 Einschränkungen.

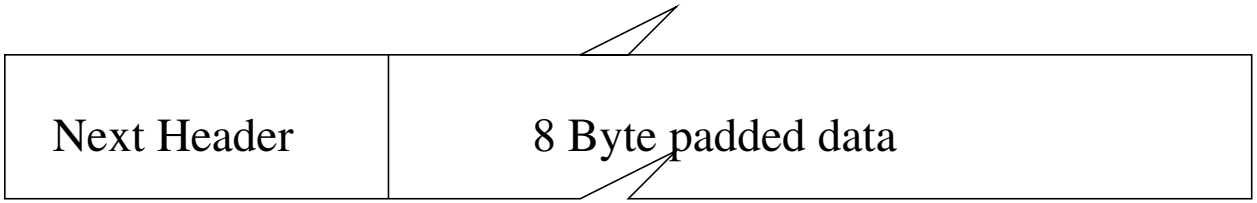
IPv6 Erweiterungsheader

Erweiterungsheader beginnen auf 8 Byte Grenzen, die folgende Reihenfolge muß eingehalten werden:

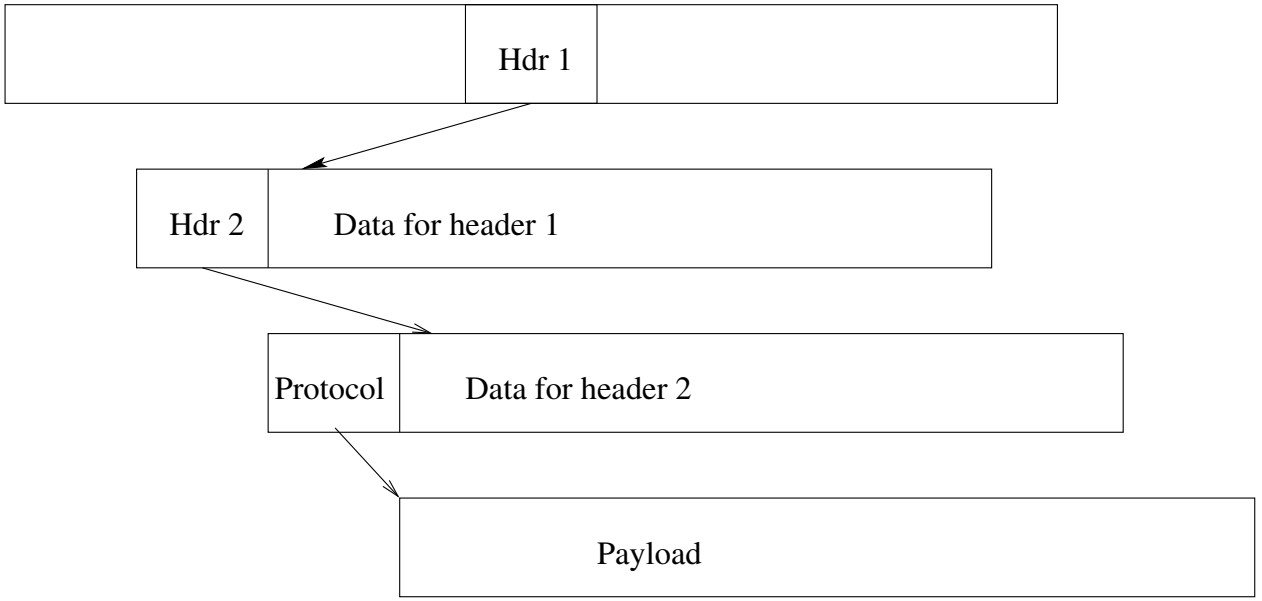
0	Hop-By-Hop Option (RFC1883)
60	Destination Option
43	Routing Header
44	Fragment Header
51	Authentication Header (RFC2402)
50	Encapsulating Security Payload (RFC2406)
59	No Next Header
60	Destination Option
135	Mobility Header

Erweiterungsheader: Format und Verkettung

Format eines Erweiterungsheaders:



Headerfolge, ähnlich IPv4 Optionen:



Hop-By-Hop Option

- Alle Optionen, die von jedem Router ausgewertet werden müssen, werden in Hop-By-Hop Headern übertragen.
- Der Header hat immer die Felder
  1. Next Header
  2. Length, gezählt in Bytes ab Byte 8
  3. Parameter
- Parameter sind Typ/Länge/Wert kodiert, bis auf Typ 0, der keine Parameter besitzt.
- Type 0 und 1 dienen als Füller
- Typ 194 (Jumbo Payload) signalisiert Pakete bis 4GByte (sonst bis 64kByte)
- Weitere Typen sind in der Standardisierung

Destination Option

- Die Destination Option ist von jedem Router auszuwerten, sofern sie vor dem Routing Header auftritt, sonst nur vom Zielhost.
- Der Aufbau entspricht den Hop-By-Hop Headern.
- Einige Optionen sind standardisiert, z.B.:
  - ein Header, der die Zahl geschachtelter IPv6 Tunnel limitiert.
  - Home Address bei Mobile IP
- Optionen können aus Anwendungen gesetzt werden.

Routing Header

- Der Routing Header hat im wesentlichen dieselbe Funktion, wie Loose Source Routing bei IPv4, allerdings ohne deren Limitierungen.
- Ein Äquivalent zu Strict Source Routing wird in IPv6 bisher nicht angeboten.
- Der Header besteht wie bei IPv4 aus einer Länge (8 Bit gezählt in 8 Byte Einheiten), einer Liste von Adressen und einem Zeiger in diese Liste, damit die Router das nächste Ziel erkennen können.
- Ein 8 Bit Typ Feld sowie 4 Byte Padding sichern Erweiterbarkeit.

Fragment Header

- Bei IPv6 fragmentiert nur der sendende Endpunkt einer Kommunikationsbeziehung, keine Router im Pfad.
- Die MTU wird mit dem Path MTU Discovery Protocol (RFC1981) bestimmt.
- Der Fragment Header entspricht weitgehend den IPv4 Headerfeldern, die zur Fragmentierung genutzt werden, d.h. er enthält:
  - 13 Bit Offset in 8 Byte Einheiten
  - 32 Bit Identification
  - 1 Bit More Fragments
  - Mit 8 Bit Next Header und 10 Bit Padding erhält man 64 Bit
- Daten des IP Headers bis zum Fragment Header können nicht fragmentiert werden.

Authentication Header (AH)

- AH ist eines der Protokolle, die unter dem Namen IPSec transparente Sicherheitsdienste auf Ebene der Vermittlungsschicht anbieten.
- IPSec ist integraler Bestandteil von IPv6.
- Es gibt inzwischen viele Implementationen von IPSec auf Basis von IPv4.
- Authentizität wird von AH durch gegenseitige Authentifizierung gesichert.
- Integrität wird durch Signatur der Header und Daten sichergestellt.
- Sicherung gegen Mehrfacheinspielung (Replay Attack) von Daten ist optional.
- Vertraulichkeit ist **nicht** Bestandteil von AH

Encapsulating Security Payload (ESP)

- ESP bietet gegenüber AH erweiterte Sicherheitsdienste
- ESP kann in Verbindung mit AH verwendet werden.
- Es werden Tunnel Modus und Transport Modus unterschieden.
- Vertraulichkeit wird durch Verschlüsselung der Daten und im Tunnel Modus durch Aggregierung von Datenströmen erreicht.
- Gegenseitige Authentifizierung der Endpunkte (Host oder Router/Security Gateway) ist möglich.
- Verhinderung von Mehrfacheinspielung ist möglich.

Mobility Header

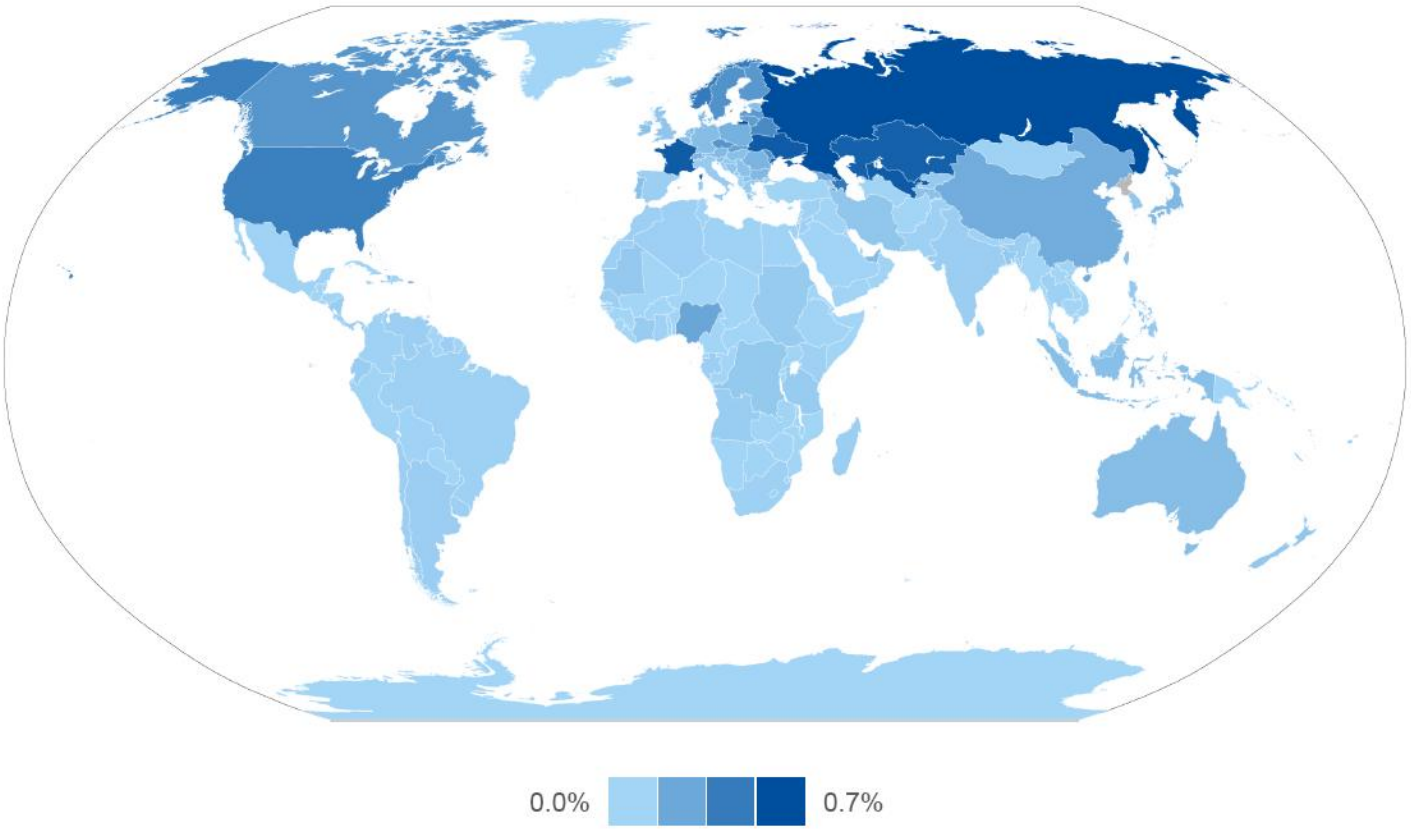
- Der Mobility Header wird benutzt, um Assoziationen zwischen mobilem Endgerät und Home-Agent herzustellen, aufzulösen oder zu testen.
- Folgende Nachrichtentypen sind spezifiziert:
  1. Binding Refresh Request Message, erzeugt Assoziation im Home-Agent
  2. Home Test Init Message, Care-of Test Init Message: Teil der “Return Routability Procedure”, die die Erreichbarkeit des Endgerätes sicherstellt
  3. Home Test Message: Antwort zur Home Test Init Message
  4. Care-of Test Message: Antwort zur Care-of Test Init Message
  5. Binding Update Message: Setzen einer neuen Care-Of-Address
  6. Binding Acknowledgement Message: Antwort auf den Binding Update Message

Migration zu IPv6

- Aktuelle Betriebssysteme (Windows, Unix Derivate, Symbian, IOS, ...) unterstützen IPv6
- Server und Infrastruktur im Internet sind inzwischen auf IPv6 eingerichtet, Protokolle angepasst, z.B.:
  - RIPng
  - BGP4+
  - DNS
- Viele Programme sind nicht fähig, IPv6 zu nutzen und werden es niemals sein.
- Know How für IPv6 ist kaum verfügbar, Erfahrungen gibt es kaum.
- Die “IPng Transition (ngtrans) working group” erklärt am 14.8.2002: **v6 considered operational**
- vgl. D.J. Bernstein: The IPv6 Mess, `http://cr.yp.to/djbdns/ipv6mess.html`

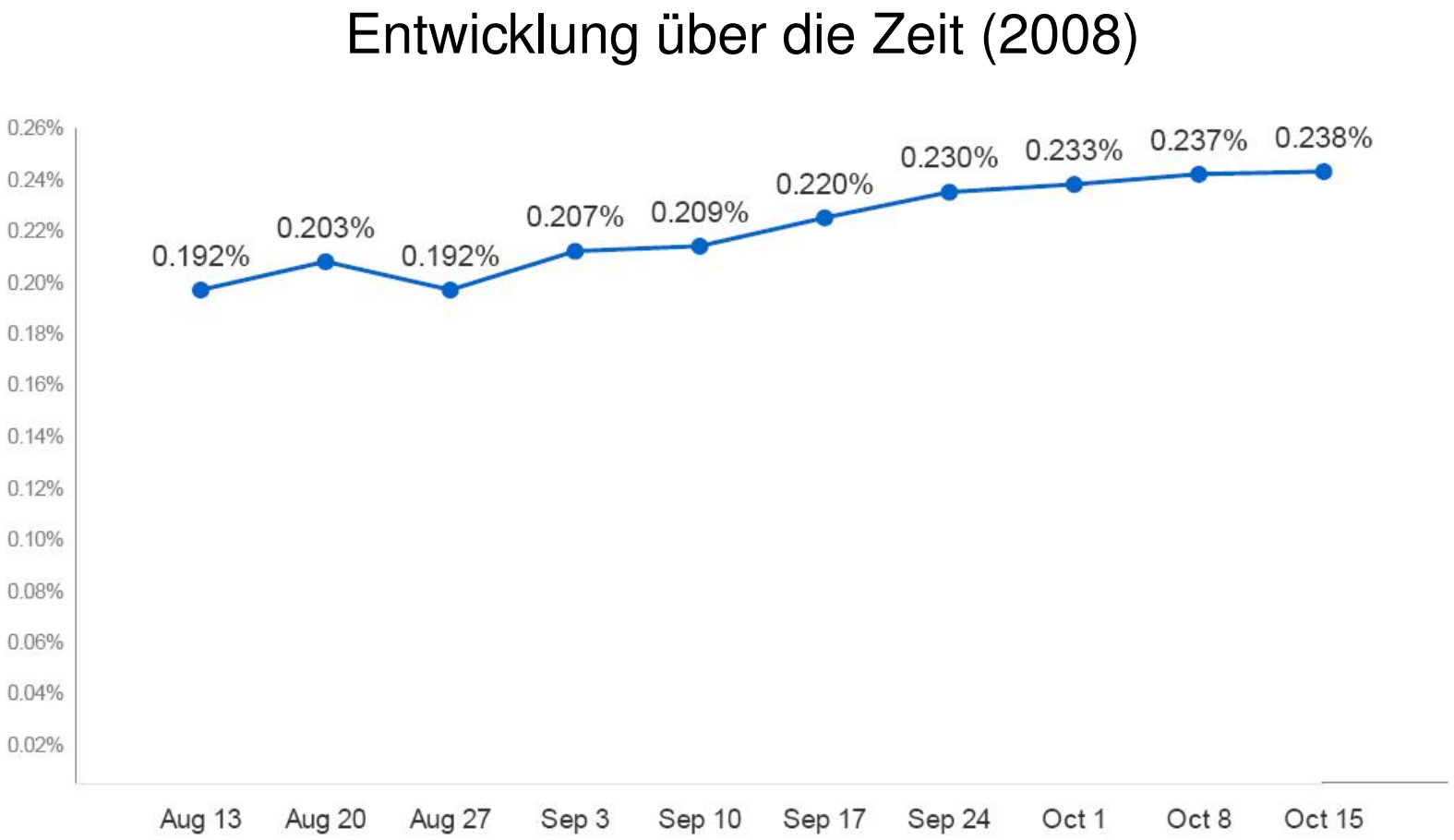
IPv6 Verbreitung - einige Statistiken von 2008

Verbreitung nach Ländern



Quelle: Global IPv6 statistics, S. H. Gunderson, Google, RIPE57, Dubai 2008

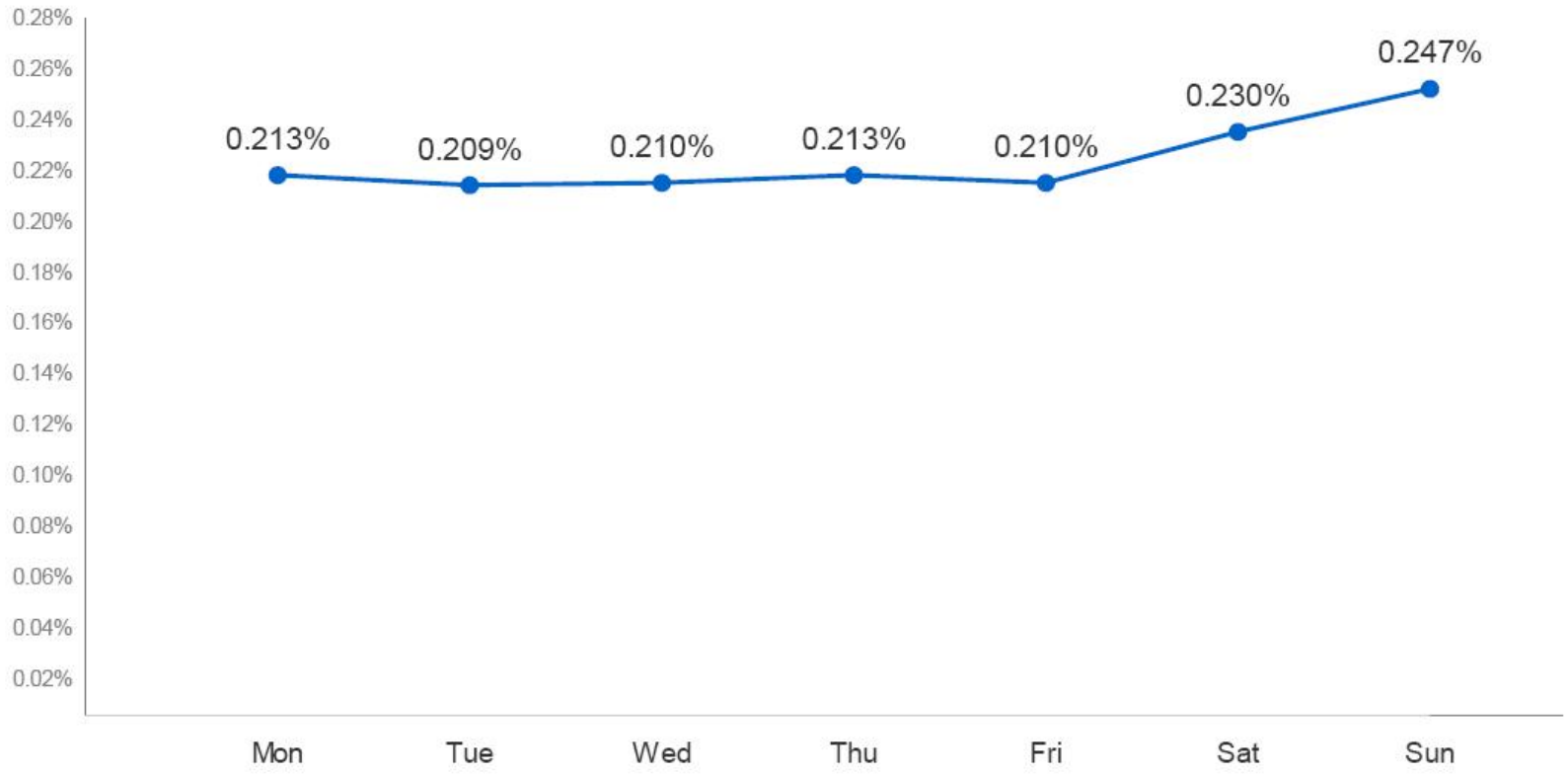
IPv6 Verbreitung - einige Statistiken von 2008



Quelle: Global IPv6 statistics, S. H. Gunderson, Google, RIPE57, Dubai 2008

IPv6 Verbreitung - einige Statistiken von 2008

Aufteilung auf Wochentage



Quelle: Global IPv6 statistics, S. H. Gunderson, Google, RIPE57, Dubai 2008

IPv6 Verbreitung - einige Statistiken von 2008

Aufteilung auf Betriebssysteme

IPv6 penetration and connectivity type by operating system
Ranked by overall IPv6 penetration

Operating system	IPv6 penetration	Native/other proportion	6to4 proportion	Teredo/ISATAP proportion
Mac OS	**2.44%**	**9%**	**91%**	**0%**
Linux	0.93%	86%	13%	1%
Windows Vista	0.32%	55%	43%	2%
Windows Server 2003	0.07%	–	–	–
Windows XP	0.03%	50%	30%	20%
Windows 2000	<0.01%	–	–	–

52% of all IPv6 hits are from Macs with 6to4

97% of all Teredo users are on Windows (even undercounting Vista)

Quelle: Global IPv6 statistics, S. H. Gunderson, Google, RIPE57, Dubai 2008

IPv6 Verbreitung - einige Statistiken von 2008

Fazit

- Große Unterschiede zwischen Ländern
- Starke regionale Abhängigkeit von einzelnen Faktoren, z.B. ISP free.fr in Frankreich
- Starke Abhängigkeit vom Betriebssystem
- Großteil des IPv6-Verkehrs wird über das 6to4 Protkoll über IPv4 getunnelt

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.

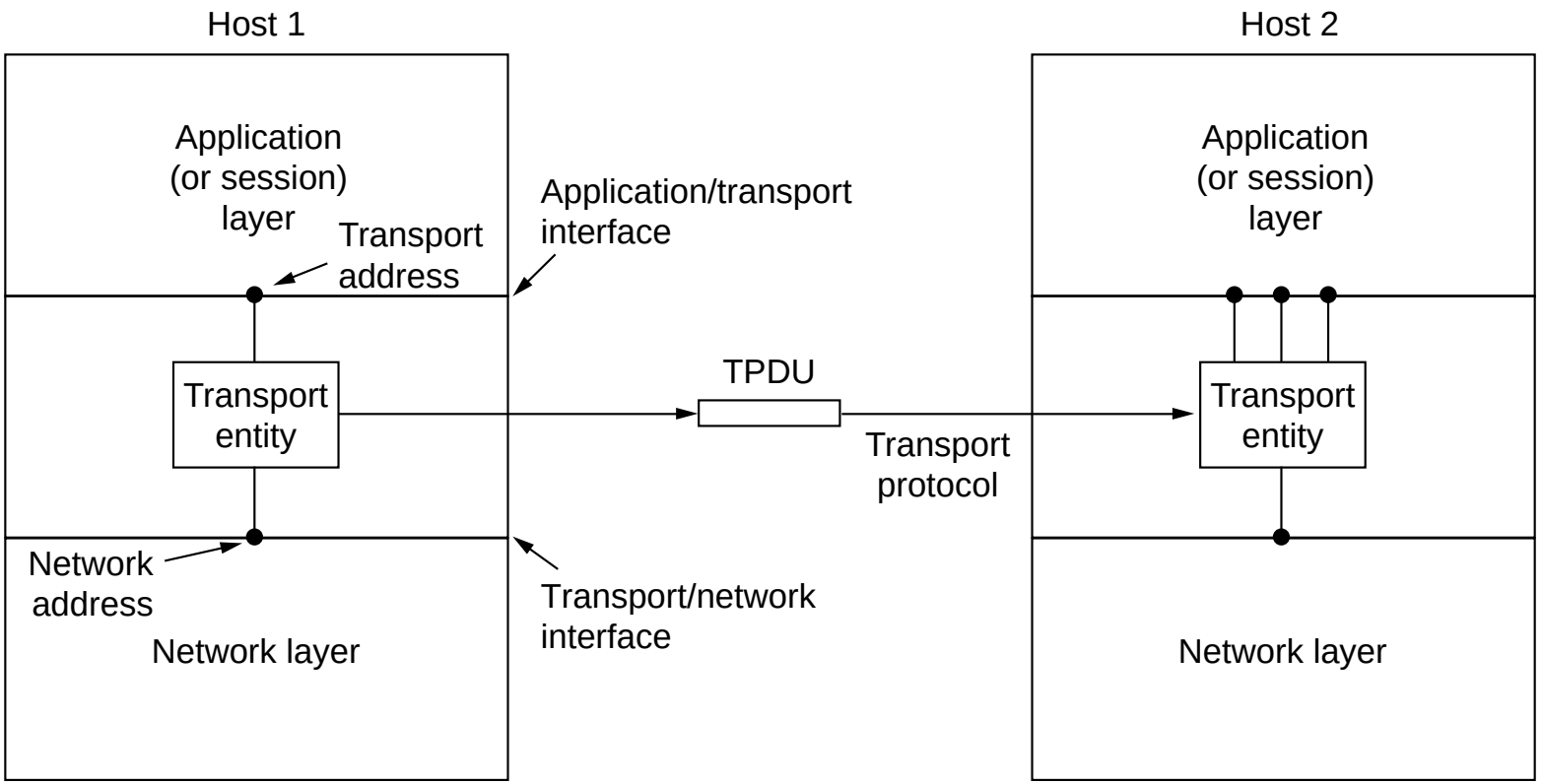
Endpunkte werden durch Transportadressen beschrieben. Im Internet z.B. sind die Endpunkte Prozesse, die durch Transportadressen identifiziert werden.

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

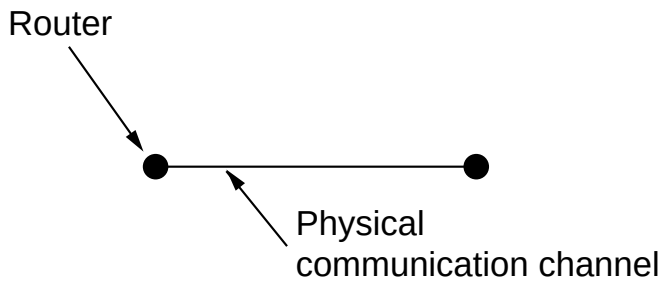
Gemeinsamkeiten

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

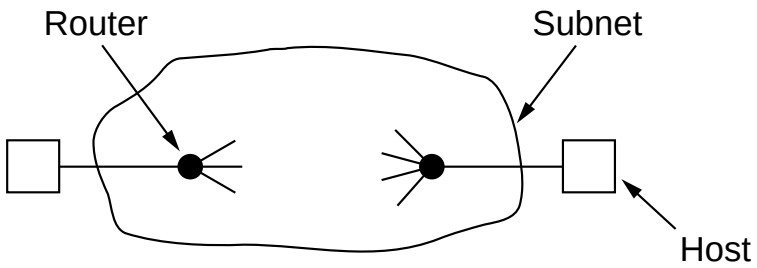
Unterschiede

- 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)

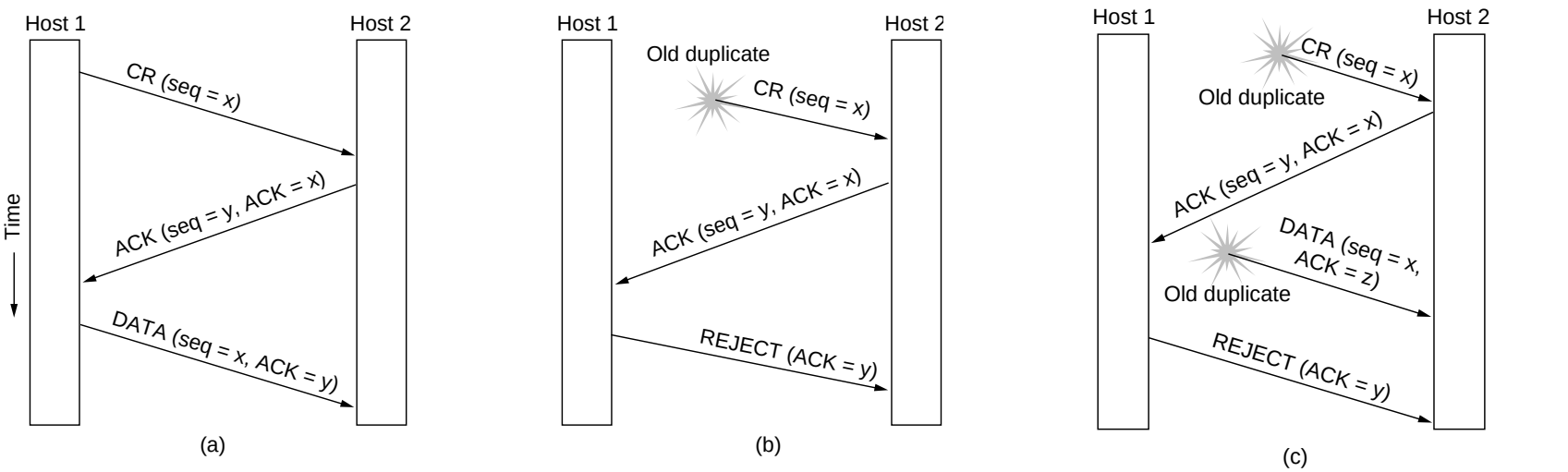
Lösungsansätze

- 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

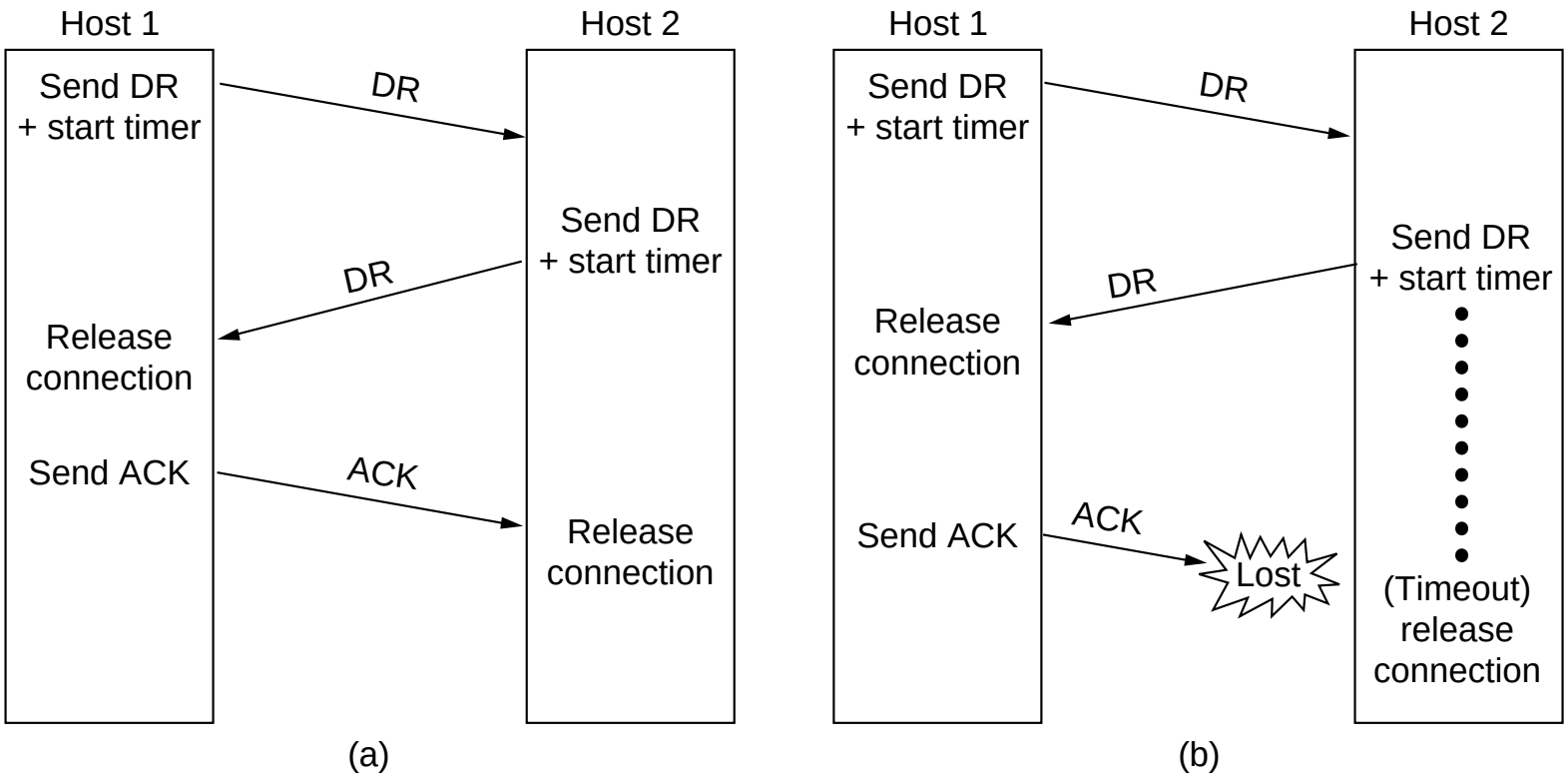
- Zuverlässiger Verbindungsabbau bei möglichen Paketverlusten
- siehe z.B. “Two-Army Problem” bzw. “Coordinated Attack Problem”

Lösungsansätze:

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

Verbindungsabbau mit “three-way handshake” (1)

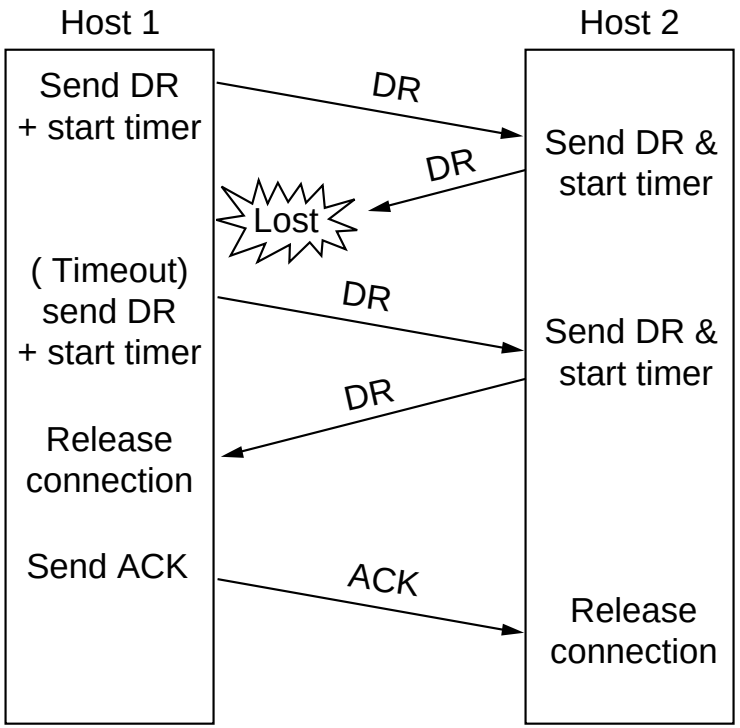
Mögliche Szenarien



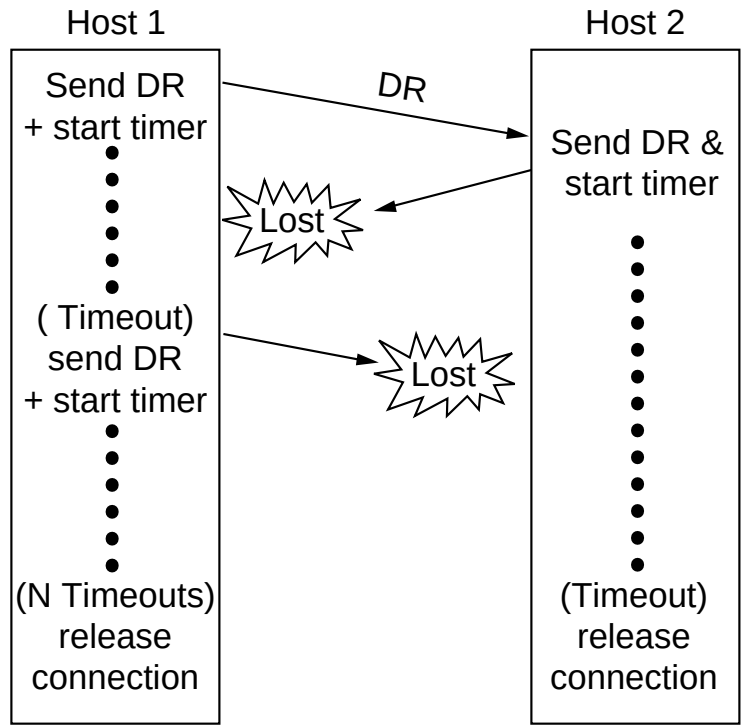
(c) Tanenbaum, Computer Networks

Verbindungsabbau mit “three-way handshake” (2)

Mögliche Szenarien



(c)



(d)

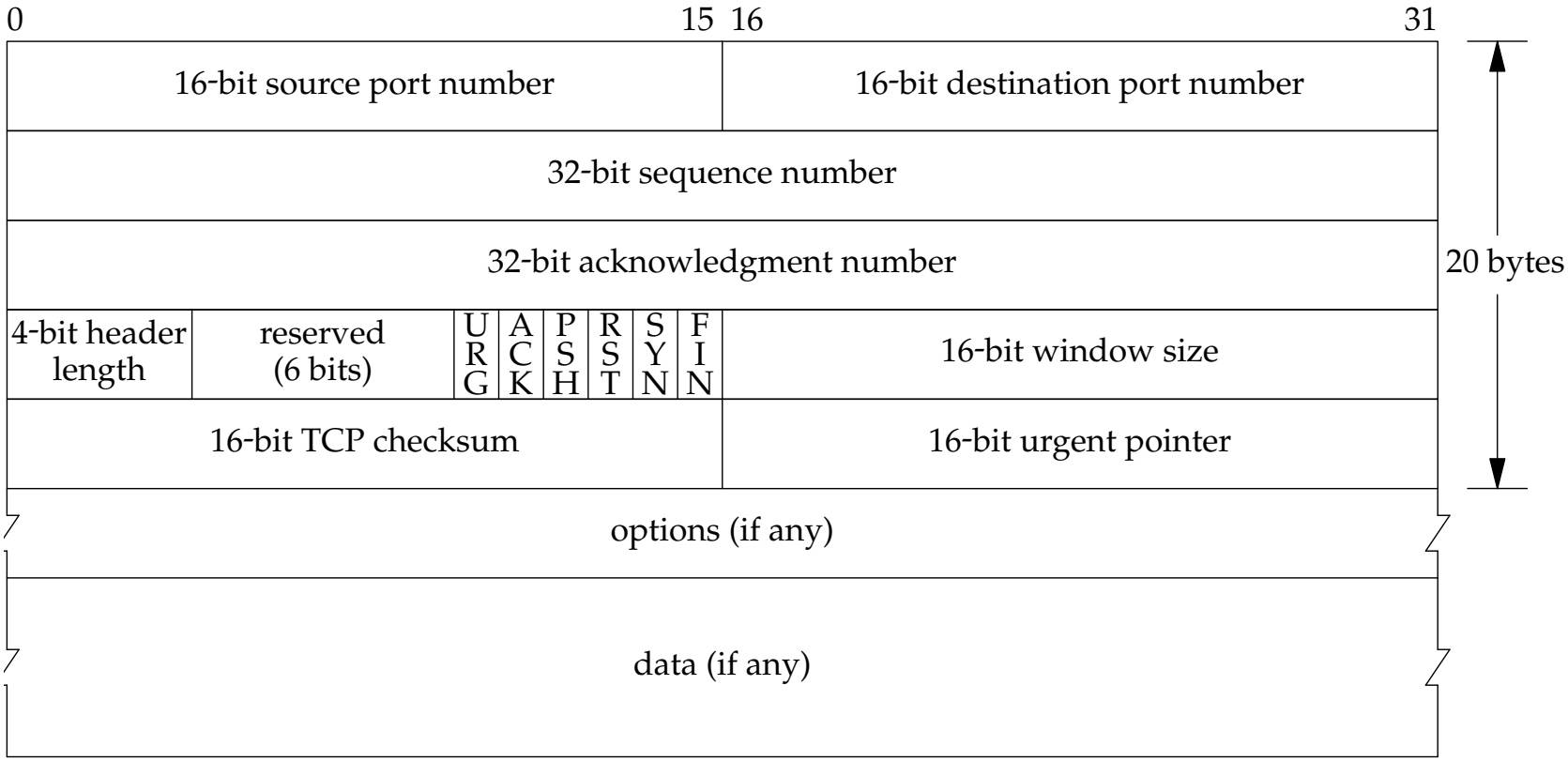
(c) Tanenbaum, Computer Networks

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>

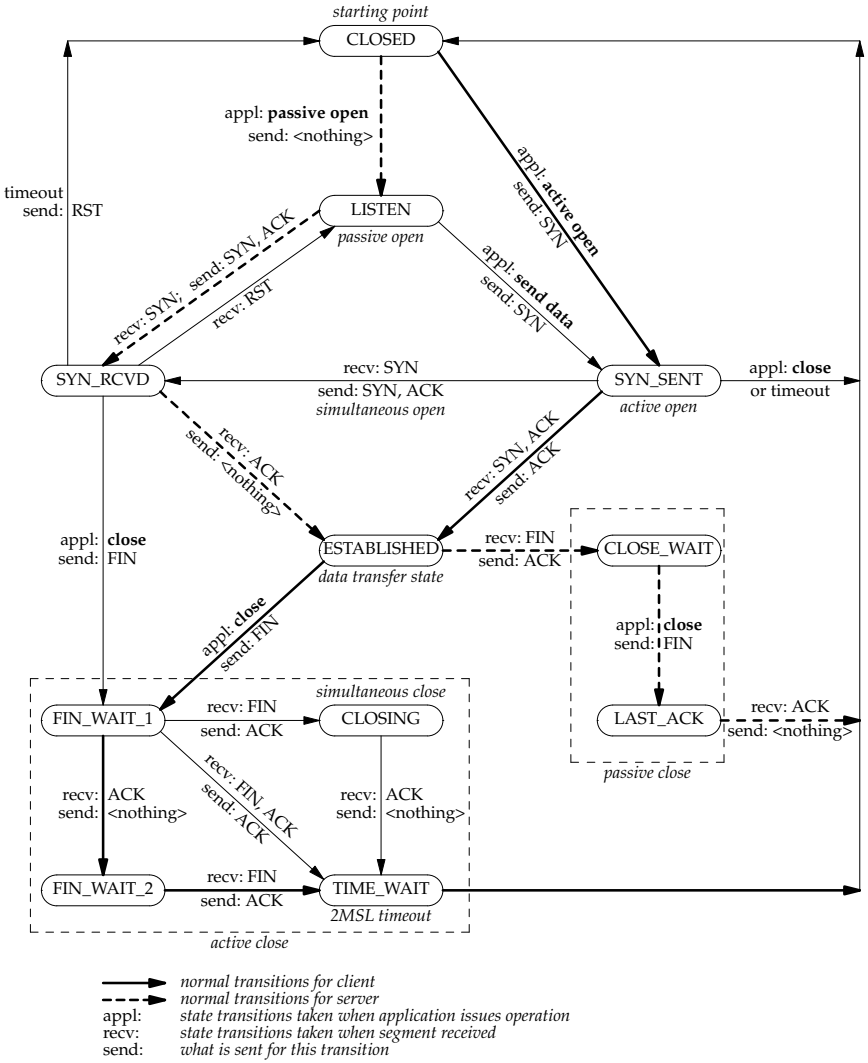
Bezeichnungen / Sprechweisen

- **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

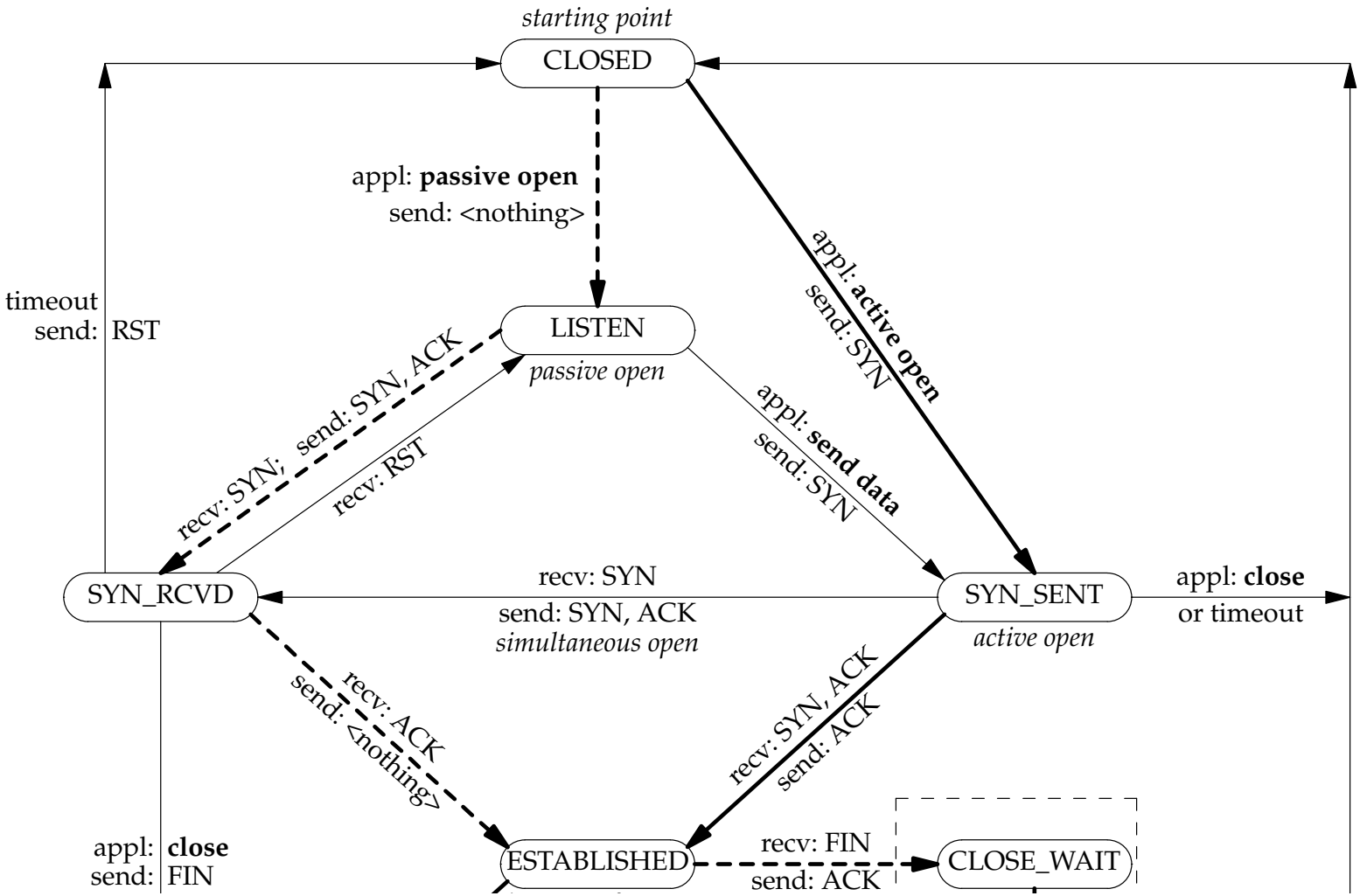
Sockets / Connections

- **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

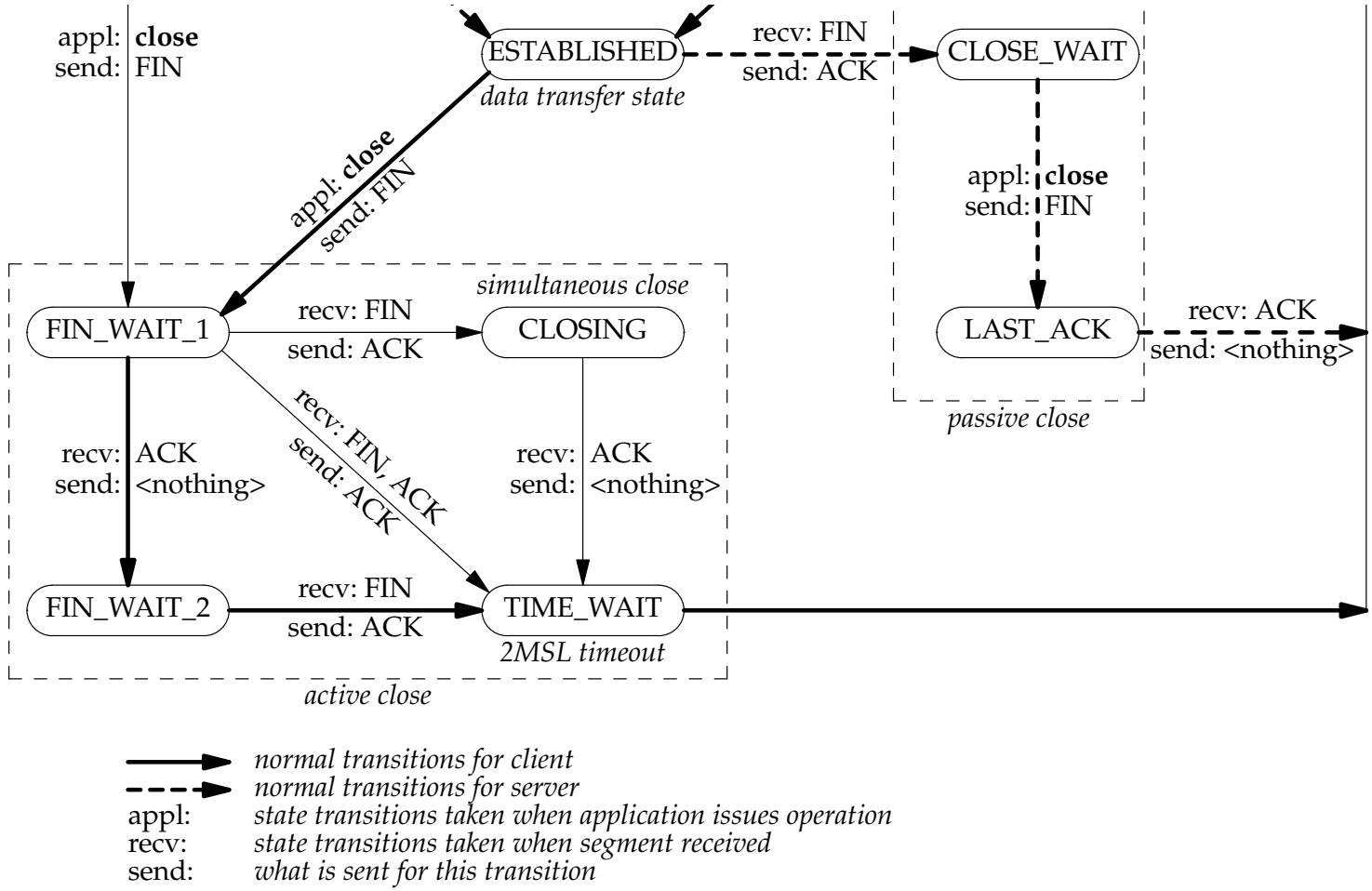
TCP Zustandsmodell



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:

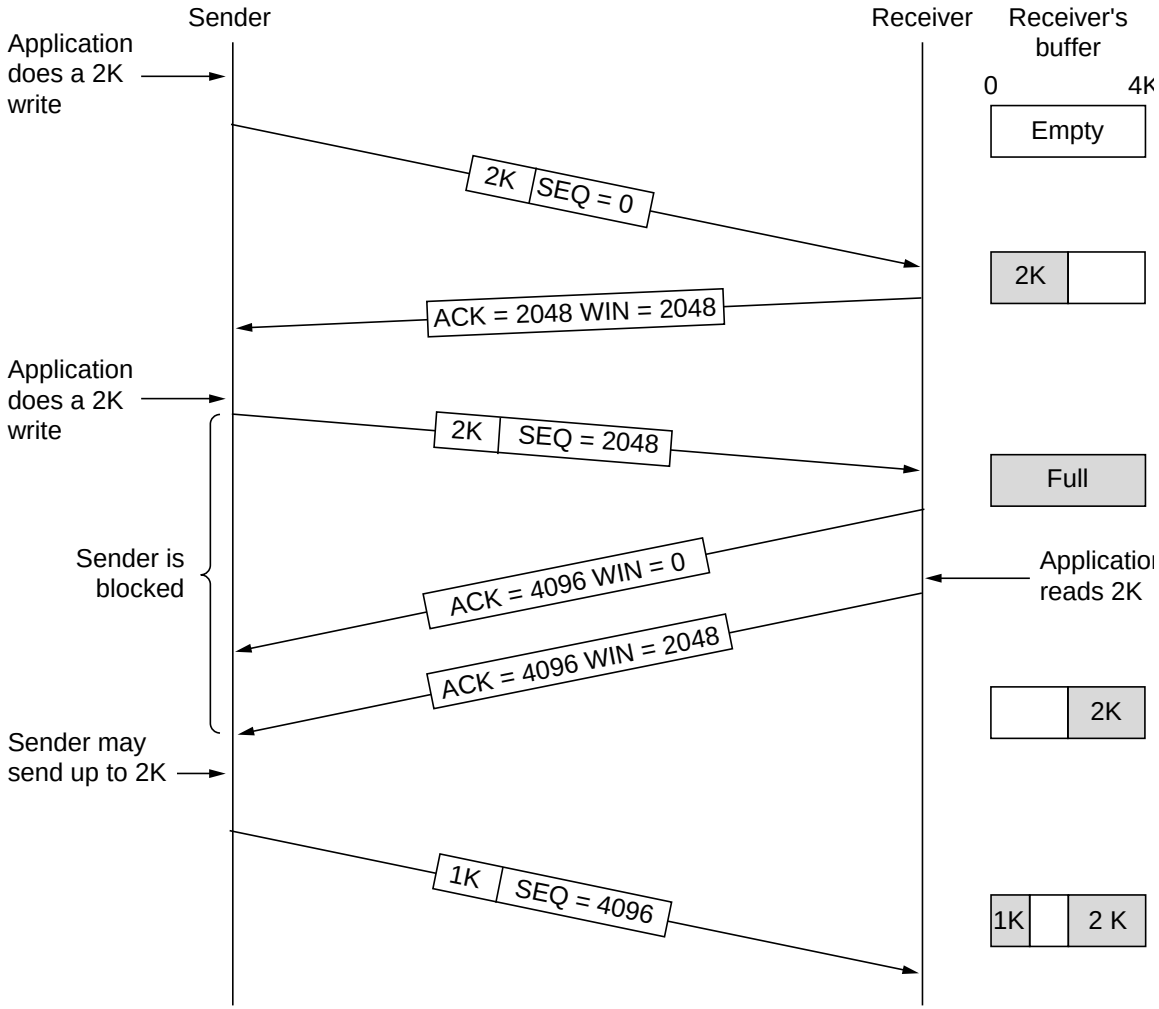
```
# tcpdump -n -i lo port 12345
127.0.0.1.51375 > 127.0.0.1.12345: S 235190854:235190854(0)
127.0.0.1.12345 > 127.0.0.1.51375: S 244140407:244140407(0)
    ack 235190855
127.0.0.1.51375 > 127.0.0.1.12345: . ack 1
127.0.0.1.51375 > 127.0.0.1.12345: P 1:7(6) ack 1
127.0.0.1.12345 > 127.0.0.1.51375: .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