Übersicht – Übung 7

Wiederholung: ARP

TCP

- Allgemeine Informationen
- Ports
- Verbindungs- und Zustandsmanagement
- Zuverlässige Übertragung
- Fluss- und Verstopfungskontrolle

Wiederholung: ARP

- Grundprinzip:
  - Host A und Host B im lokalen Netzwerk (Ethernet)
  - A kennt IP-Adresse von B und benötigt MAC-Adresse
  - A sendet ARP-Request als Broadcast (MAC: FF FF FF FF FF FF):

    → Sender-MAC: MAC-Adresse von A

    → Sender-IP: IP-Adresse von A

    → Ziel-MAC: 00 00 00 00 00 00

    → Ziel-IP: IP-Adresse von B

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Wiederholung: ARP

- Grundprinzip:
  - Host A und Host B im lokalen Netzwerk (Ethernet)
  - A kennt IP-Adresse von B und benötigt MAC-Adresse
  - B sendet ARP-Response an MAC-Adresse von A:

    → Sender-MAC: MAC-Adresse von B

    → Sender-IP: IP-Adresse von B

    → Ziel-MAC: MAC-Adresse von A

    → Ziel-IP: IP-Adresse von A

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Wiederholung: ARP

- *ARP-Probe*:
  - Rechner kommt neu ins Netzwerk und möchte testen, ob seine favorisierte IP-Adresse schon vergeben ist
  - Sender-IP auf 0.0.0.0
- *Gratuitous-ARP*:
  - Mitteilung an die anderen Rechner im Netz, um deren ARP-Cache zu aktualisieren
  - Sender-IP = Ziel-IP = eigene IP
  - Geänderte MAC-Adresse
  - Szenario: *Mobile-IP* (siehe Vorlesung)

Wiederholung: ARP

- *ARP-Spoofing*:
  - Feindliche Übernahme fremder IP-Adressen
  - Host A meldet, über die IP-Adresse von Host B zu verfügen
  - Mitlesen möglich
  - Doppeltes ARP-Spoofing + Weiterleitung → *Man-in-the-Middle*

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Allgemeine Informationen

- *Auf welcher OSI-Schicht ist das Transmission Control Protocol (TCP) angesiedelt?*
  - Schicht 4 (Transportschicht)
  - RFC 793 (1981)
- *Nennen Sie die fünf Hauptaufgaben des TCP.*
  - Auf- und Abbau von Verbindungen

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Allgemeine Informationen

- *Auf welcher OSI-Schicht ist das Transmission Control Protocol (TCP) angesiedelt?*
  - Schicht 4 (Transportschicht)
  - RFC 793 (1981)
- *Nennen Sie die fünf Hauptaufgaben des TCP.*
  - Auf- und Abbau von Verbindungen

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Allgemeine Informationen

- *Auf welcher OSI-Schicht ist das Transmission Control Protocol (TCP) angesiedelt?*
  - Schicht 4 (Transportschicht)
  - RFC 793 (1981)
- *Nennen Sie die fünf Hauptaufgaben des TCP.*
  - Auf- und Abbau von Verbindungen
  - Stromorientierte Vollduplex-Übertragung

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Allgemeine Informationen

- *Auf welcher OSI-Schicht ist das Transmission Control Protocol (TCP) angesiedelt?*
  - Schicht 4 (Transportschicht)
  - RFC 793 (1981)
- *Nennen Sie die fünf Hauptaufgaben des TCP.*
  - Auf- und Abbau von Verbindungen
  - Stromorientierte Vollduplex-Übertragung

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Allgemeine Informationen

- *Auf welcher OSI-Schicht ist das Transmission Control Protocol (TCP) angesiedelt?*
  - Schicht 4 (Transportschicht)
  - RFC 793 (1981)
- *Nennen Sie die fünf Hauptaufgaben des TCP.*
  - Auf- und Abbau von Verbindungen
  - Stromorientierte Vollduplex-Übertragung

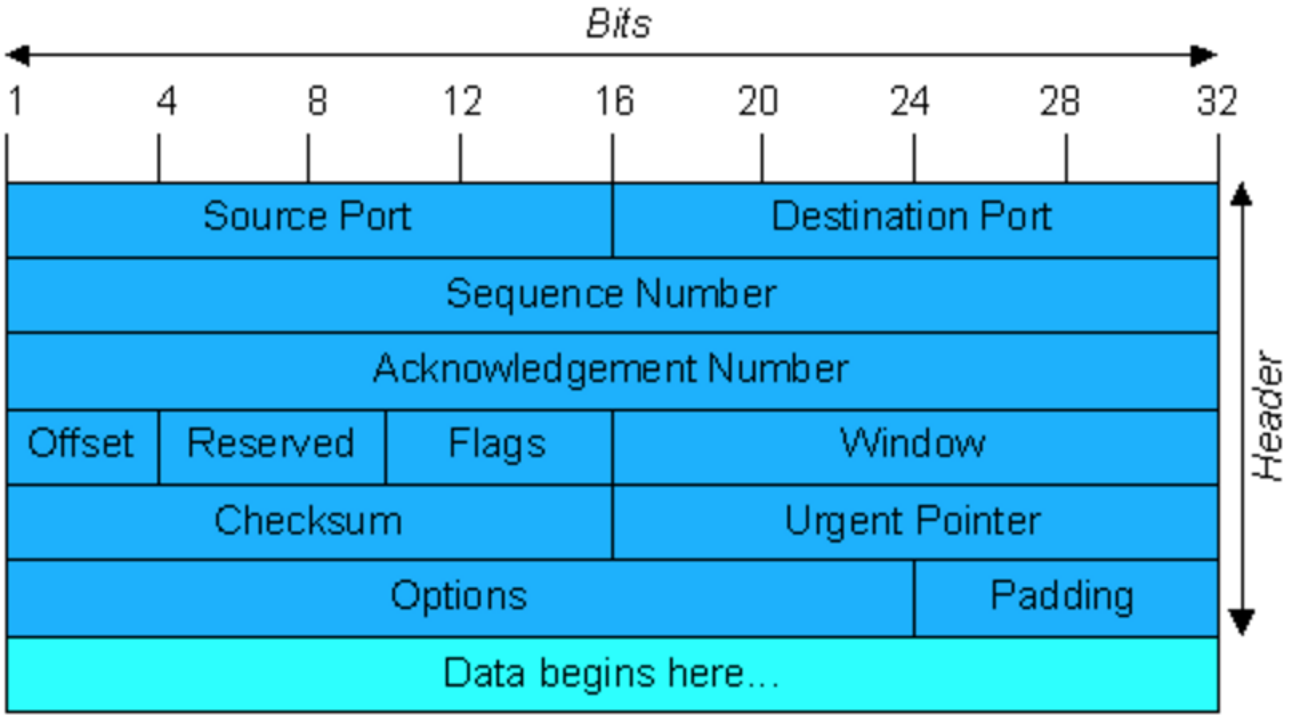
Allgemeine Informationen

- *Auf welcher OSI-Schicht ist das Transmission Control Protocol (TCP) angesiedelt?*
  - Schicht 4 (Transportschicht)
  - RFC 793 (1981)
- *Nennen Sie die fünf Hauptaufgaben des TCP.*
  - Auf- und Abbau von Verbindungen
  - Stromorientierte Vollduplex-Übertragung
  - Zuverlässige Übertragung
  - Flusskontrolle
  - Verstopfungskontrolle

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Allgemeine Informationen

- *Erläutern Sie die Felder im TCP-Header.*



http://www.rvs.uni-bielefeld.de/~heiko/tcpip/tcpip_html_alt/abbildungen/tcp_header.gif

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Ports

- *Wozu werden **Ports** benötigt? Handelt es sich um ein soft- oder hardwareseitiges Konzept?*
  - Softwareseitig
  - Adressierung von *Prozessen* bzw. *Diensten* auf dem Zielhost
- *Durch welches **5-Tupel** lassen sich Schicht-4-Verbindungen eindeutig identifizieren?*
  - (Quell-IP, Quell-Port, Ziel-IP, Ziel-Port, Protokoll)
  - Beispiel: (192.168.2.1, 12345, 139.20.64.219, 443, TCP)

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Ports

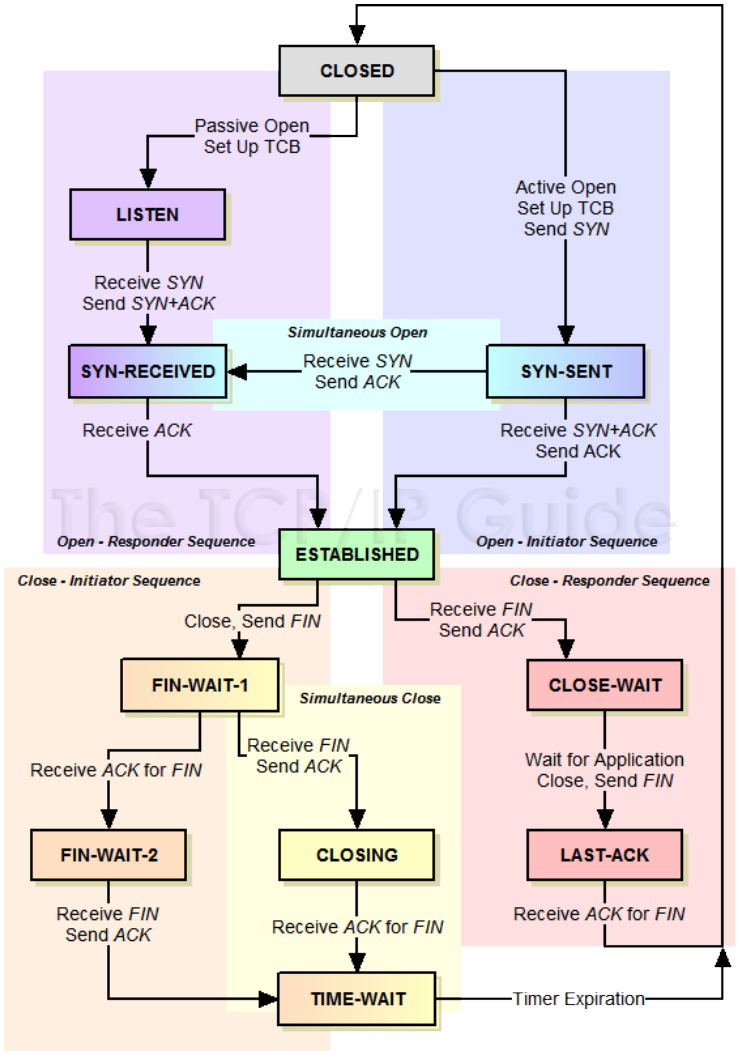
- *Welchen Protokollen sind die folgenden **Well-known-Ports** durch die IANA (Internet Assigned Numbers Authority) per Standard zugeordnet? Welche Funktionen übernehmen sie?*
  - 21: FTP
  - 22: SSH
  - 23: Telnet
  - 25: SMTP
  - 53: DNS
  - 80: HTTP
  - 123: NTP
  - 143: IMAP
  - 443: HTTPS

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Verbindungs- und Zustandsmanagement

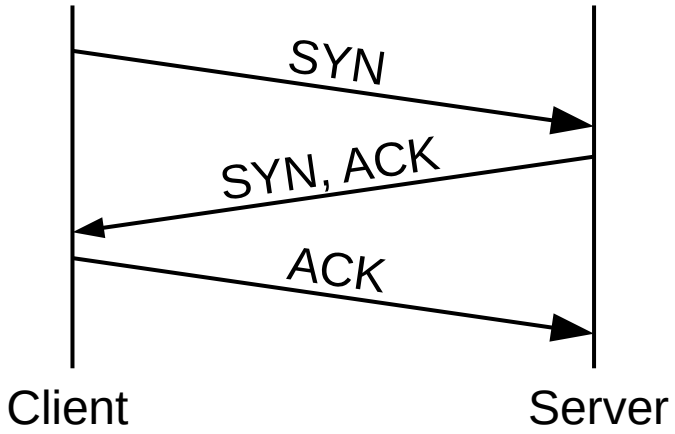
- *Zeichnen Sie die **TCP-State-Machine**.*

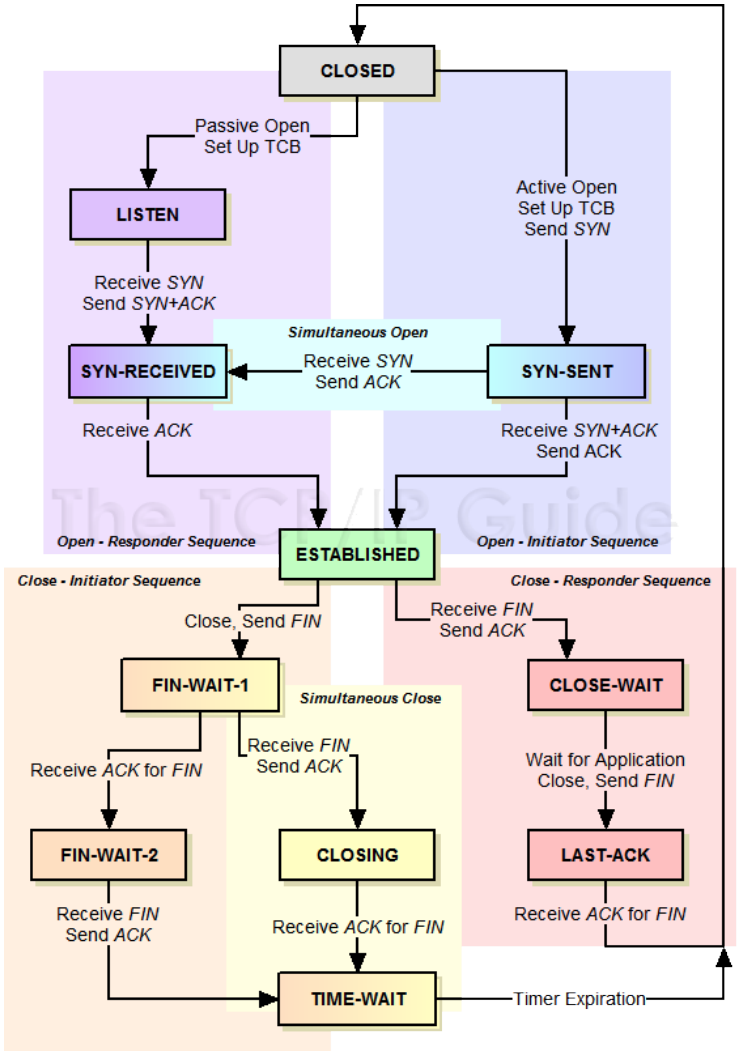
Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>



http://tcpipguide.com/free/diagrams/tcpfsm.png

- *Wie findet der Verbindungsaufbau (aktiv / passiv / simultan) statt? Erläutern Sie in diesem Zuge auch den **Three-Way-Handshake**.*

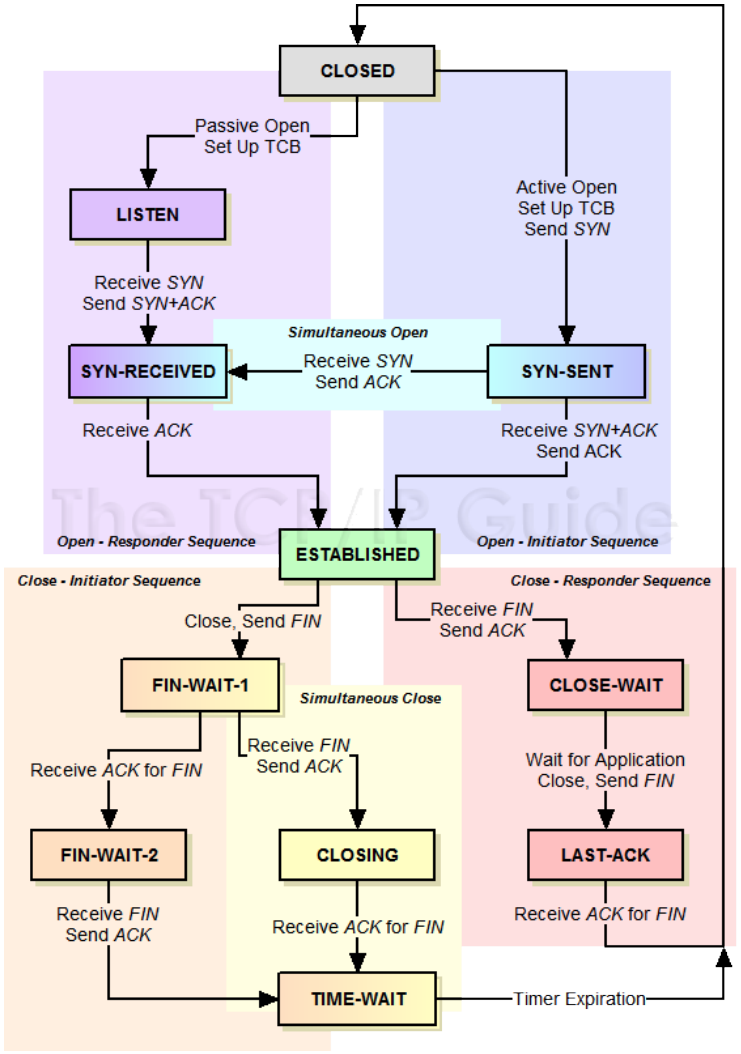




http://tcpipguide.com/free/diagrams/tcpfsm.png

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

- *Wie wird die Verbindung wieder abgebaut? Warum ist der **TIME-WAIT**-Zustand in der TCP-State-Machine notwendig?*



http://tcpipguide.com/free/diagrams/tcpfsm.png

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Zuverlässige Übertragung

- *Wie vergibt das TCP die Sequenznummern für die Segmente?*
  - Eine Sequenznummer für jedes Byte des Stromes
  - Bestätigt wird mit der als nächstes zu erwartenden Sequenznummer!
- *Setzt TCP in den Grundeinstellungen bei verlorenen Segmenten auf **Go-back-N** oder auf **Selective Repeat** (siehe Übung 3)? Wie lässt sich dieses Verhalten modifizieren?*
  - Grundsätzlich: *Go-back-N* (wenn ein Segment fehlt, werden folgende Segmente verworfen, solange es nicht eintrifft)
  - RFC 2018 (1996): *TCP-SACK* (*Selective Acknowledgement)* in den Options

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

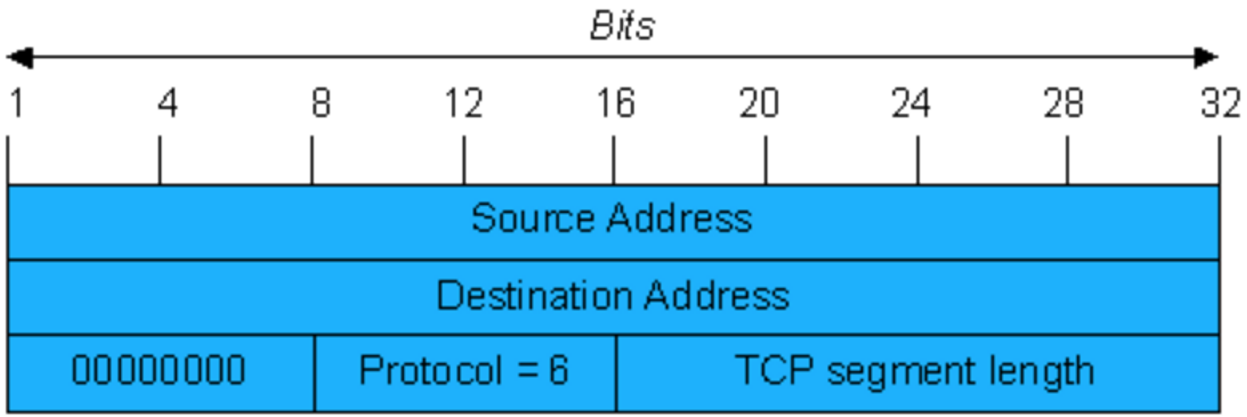
Zuverlässige Übertragung

- *Wie wird sichergestellt, dass (evtl. datenlose) Segmente mit gesetztem **SYN**- resp. **FIN**-Flag zuverlässig und in der richtigen Reihenfolge übertragen werden?*
  - SYN und FIN zählen im Sinne der Sequenznummer als jeweils ein Byte.
  - Beispiel: Three-Way-Handshake (Wireshark)

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Zuverlässige Übertragung

- *Wie wird die TCP-Prüfsumme gebildet?*
  - Pseudoheader:



http://www.rvs.uni-bielefeld.de/~heiko/tcpip/tcpip_html_alt/abbildungen/pseudo_header.gif

- *Stellen Sie den Vergleich zur IP-Prüfsumme her. Gehen Sie auf Redundanzen in Bezug auf das OSI-Modell ein.*

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Fluss- und Verstopfungskontrolle

- *Wie ist das aus Übung 3 bekannte **Sliding Window** im TCP implementiert?*
  - 16 Bit, freie Bytes im Empfangspuffer
  - Maximal 65.535 Bytes, heute z. T. zu wenig
  - TCP Window Scale Option (RFC 1323, von 1992): Multiplikationsfaktor von bis zu 2^{14}
- *Wie hängen die Größen **Durchsatz**, **Fenstergröße** und **Round-Trip-Time (RTT)** zusammen?*
  - Idealfall: Ein komplettes Fenster wird geschickt, nach einer RTT trifft das ACK ein → Durchsatz ist Fenstergröße pro RTT

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Fluss- und Verstopfungskontrolle

- *Erläutern Sie den Begriff der **Maximum Segment Size (MSS)**.*
  - Zunächst: *Maximum Transmission Unit* (*MTU*): Maximalgröße eines Schicht-3-Pakets (IP) inklusive Header, das als Payload von Schicht 2 (Ethernet) übertragen wird.
  - IP: Eigentlich 65.535 Bytes möglich, aber Ethernet kann nur 1500 Bytes Nutzdaten transportieren.

    → MTU für IP ist hier 1500 Bytes.
  - MSS: Maximalgröße der Nutzdaten eines TCP-Segments, ohne dass IP-Fragmentierung auftritt

    → 1500 Bytes (IP-MTU) – 20 Bytes (IP-Header) – 20 Bytes (TCP-Header) = 1460 Bytes MSS
  - Sinnvollerweise: Maximale Fenstergröße ist n * MSS

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Fluss- und Verstopfungskontrolle

- *Was ist das **Silly Window Syndrome (SWS)**? Erläutern Sie Problemlösungen auf Sender- und Empfängerseite.*
  - SWS auf Senderseite: Anwendung generiert Bytes nur stückweise
  - SWS auf Empfängerseite: Anwendung liest Bytes nur stückweise
  - Wenig Nutzdaten in einem Paket → ineffizient
  - Lösung für die Senderseite: *Nagle's algorithm* (solange Segmente unbestätigt sind: neue Daten bis auf eine MSS puffern und erst dann senden)
  - Lösung für die Empfängerseite: *Clark's algorithm* (Fenster geschlossen halten, bis eine MSS oder der halbe Empfangspuffer frei ist)

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Fluss- und Verstopfungskontrolle

- Verstopfungskontrolle:
  - *Congestion window* (*CWND*) als zusätzliche Größe
  - Interne Berechnung (CWND tritt nie explizit im TCP-Header auf!)
  - Aus Sicht des Senders: Fenstergröße = min(WND, CWND)

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Fluss- und Verstopfungskontrolle

- **Slow Start:**
  - Zu Beginn: CWND = 1 (oder anderer kleiner Wert wie 2 oder 10)
  - Wenn ACK empfangen: Erhöhe CWND um 1.

    → exponentielles Wachstum
  - Limitierung: Irgendwann überschreitet CWND den Wert von WND
  - Bei Timeout: Setze CWND wieder auf 1 (oder 2 oder 10) und beginne von vorn.

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

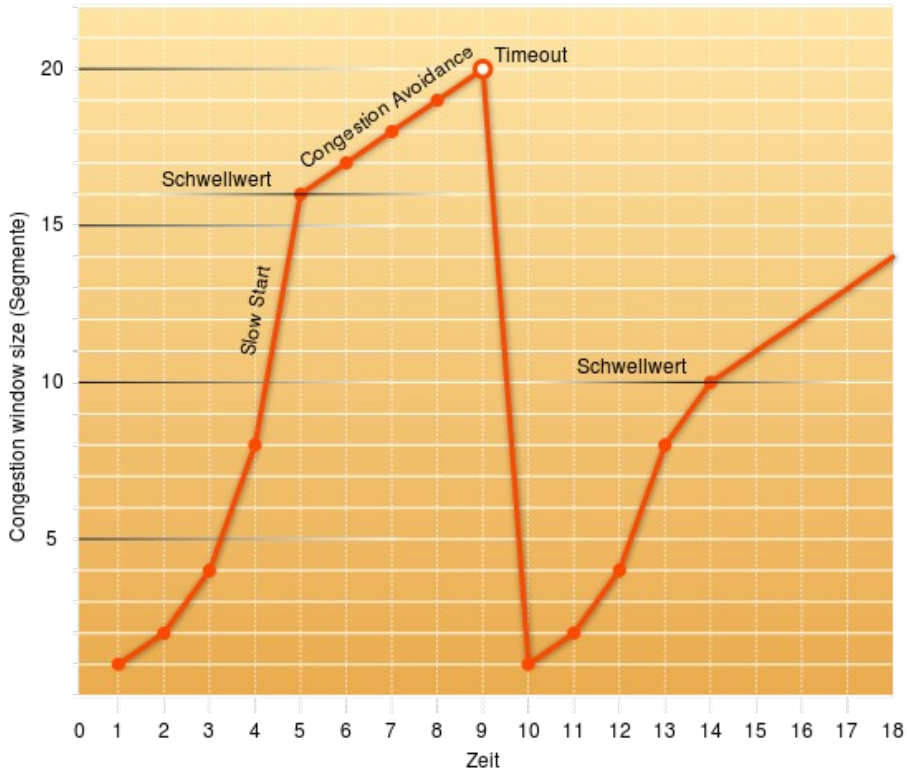
Fluss- und Verstopfungskontrolle

- **Congestion Avoidance:**
  - Problem: *Slow Start* führt meistens zu Timeout → „Sägezahn-Verhalten“
  - Einführung eines Schwellwertes: *ssthresh*
  - Initialisierung: „sinnvoller“ Wert unterhalb von WND
  - Wenn CWND >= ssthresh: Inkrementiere CWND bei jedem ACK nur noch um 1 / CWND

    → lineares Wachstum ab Schwellwert
  - Bei Timeout: Korrigiere ssthresh auf max(2, FlightSize* / 2), setze CWND auf 1 und beginne wieder mit *Slow Start*.

*FlightSize: tatsächlich genutzter Teil des Fensters (gesendete, aber unbestätigte Bytes)

Fluss- und Verstopfungskontrolle



https://de.wikipedia.org/wiki/Transmission_Control_Protocol#/media/File:TCPSlowStartundCongestionAvoidance.svg

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Fluss- und Verstopfungskontrolle

- **Fast Retransmit:**
  - Zu kleine Timeouts führen zu häufiger Neuübertragung → generell groß wählen
  - Können wir verlorene Segmente schon früher erkennen (Tipp: *Go-back-N*) ?
  - Mehrfache Bestätigung eines Segments spricht für Verlust, bevor der Timeout abgelaufen ist: *Dupe-ACKs*
  - Daher: Sofortige Neuübertragung, wenn Dupe-ACKs auftreten

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>

Fluss- und Verstopfungskontrolle

- **Fast Recovery:**
  - Dupe-ACKs sind ein Zeichen für Verstopfung → durch *Fast Retransmit* ignoriert
  - Dupe-ACKs als „Pseudo-Timeout“ behandeln und, wie bisher auch, ssthresh auf CWND / 2 setzen
  - Aber: CWND nicht auf 1, sondern ebenfalls auf CWND / 2 setzen

    → Im Gegensatz zum „echten“ Timeout wird *Slow Start* übersprungen und *Congestion Avoidance* kommt unmittelbar zum Tragen.

Ben Lorenz <lorenzb@tu-freiberg.de>
Jonas Treumer <treumer@tu-freiberg.de>