Transmission Control Protocol (TCP)

Netzwerkgrundlagen (NWG2)

Markus Zeilinger[1]

[1]FH Oberösterreich
Department Sichere Informationssysteme

Sommersemester 2023

FH OBERÖSTERREICH
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Wichtiger Hinweis

Alle Materialien, die im Rahmen dieser LVA durch den LVA-Leiter zur Verfügung gestellt werden, wie zum Beispiel Foliensätze, Audio-Aufnahmen, Übungszettel, Musterlösungen, ... dürfen ohne explizite Genehmigung durch den LVA-Leiter NICHT weitergegeben werden!

Inhalt

Transmission Control Protocol (TCP)

- Transmission Control Protocol (TCP) - RFC 793
- Verbindungsorientierte und zuverlässige Übertragung eines Byte-Stroms (Stream) zwischen Prozessen auf (nicht notwendigerweise) unterschiedlichen Systemen über ein unzuverlässiges Netzwerk.
- TCP betrachtet von der Anwendung kommende Daten (via `write()` Call) als prinzipiell unendlichen Byte-Strom (Stream-Orientierung) und kümmert sich selbst um die Einteilung der Daten in Segmente.

Transmission Control Protocol (TCP)

Schlüsselaspekte

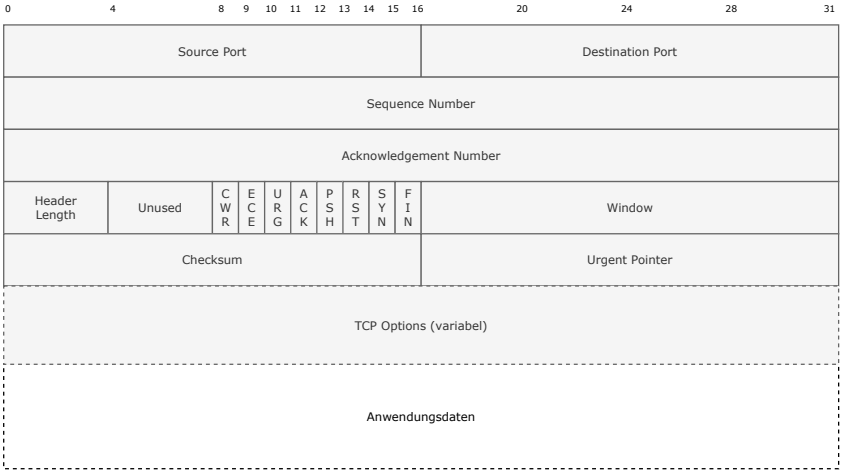
- Adressierung über Ports (lokaler und remote Socket bilden die Kommunikationsendpunkte).
- Verbindungsorientierung: Aufbau, Verwaltung und Abbau von bidirektionalen Verbindungen (TCP Finite State Machine [FSM]) (Analogie: Telefonanruf).
- Paketierung (in Segmente) des Byte-Stroms (stream-orientiert) und Weiterleitung an IP.
- Full-Duplex, d. h. in einer TCP-Verbindung können gleichzeitig Daten in beide Richtungen fliessen.
- Zuverlässigkeit (Reliability) durch Positive Acknowledgment with Retransmission (PAR).
- Flusskontrolle (Flow Control) und Überlastungsüberwachung (congestion control).

TCP RFCs (Auszug der wichtigsten)

- RFC 7414, A Roadmap for Transmission Control Protocol (TCP) Specification Documents

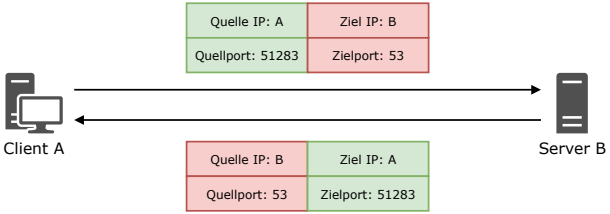
RFC	Jahr	Titel	Beschreibung
793	1981	Transmission Control Protocol	Basisstandard
1122	1989	Requirements for Internet Hosts	Implementierungsdetails zu u. a. TCP
2018	1996	TCP Selective Acknowledgment Options	Selektive Angabe von TCP Segmenten, die neuerlich übertragen werden müssen.
5681	2009	TCP Congestion Control	Beschreibung von vier TCP Congestion Control Algorithmen: Slow Start, Congestion Avoidance, Fast Retransmit und Fast Recovery
6298	2011	Computing TCP's Retransmission Timer	Algorithmus zur Berechnung des Retransmission Timers.
6691	2012	TCP Options and Maximum Segment Size (MSS)	TCP Maximum Segment Size Option
7323	2014	TCP Extensions for High Performance	Mechanismen für TCP auf High-Speed Links (z. B. TCP Window Scale Option).

TCP Header I



TCP Header II

- Source (Quell-) Port: Port des sendenden Prozesses am Quellsystem.
  - Client → Server: Dynamic/Private Port > 49152
  - Server → Client: Well-Known oder Registered Port des Services
- Destination (Ziel-) Port: Port des empfangenden Prozesses am Zielsystem.
  - Client → Server: Well-Known oder Registered Port des Services
  - Server → Client: Dynamic/Private Port > 49152



TCP Header III

Zusammenhang Bytestrom, Sequence Number und Acknowledgement Number

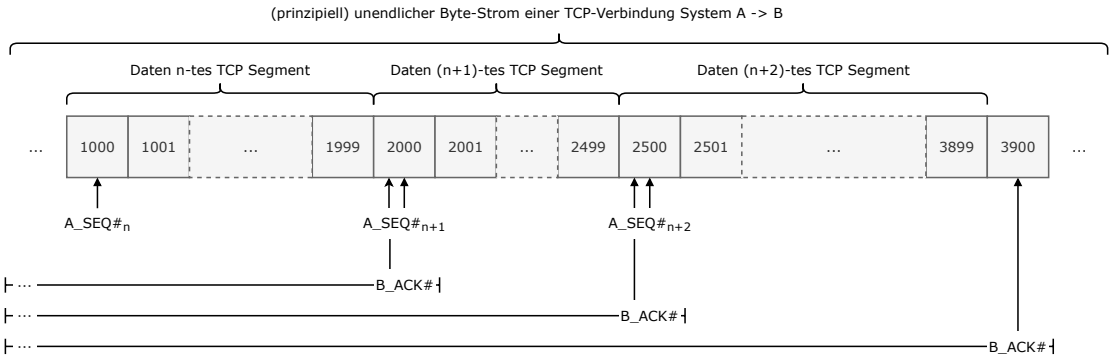


Abbildung 1: Zusammenhang Bytestrom, Sequence Number, Acknowledgement Number (basierend auf [1])

TCP Header IV

- Sequence Number (Sequenznummer, SEQ#)
  - Nummer des ersten im Segment übertragenen Bytes.
- Acknowledgement Number (Bestätigungsnummer, ACK#)
  - Nummer des nächsten erwarteten Bytes, d. h. bis inklusive $(ACK\# - 1)$ wurde erfolgreich empfangen.
  - Cumulative Acknowledgement = TCP bestätigt immer alles bis zum letzten korrekt empfangenen Byte, d. h. auch schon alle zuvor bestätigten.
- Window (Window Size, Fenstergröße)
  - Anzahl an Bytes, die ein System A senden darf, bevor es längstens auf eine Bestätigung von seinem Gegenüber B warten muss.
  - Wird für die Flusskontrolle (Flow Control) verwendet, B steuert A in dessen Übertragungsgeschwindigkeit.

TCP Header V

- SYN (Synchronize): Signalisiert den Wunsch eines Systems A zum Verbindungsaufbau mit System B.
- FIN (Finish): Signalisiert den Wunsch eines Systems A zum Verbindungsabbau mit System B.
- RST (Reset): Zurücksetzung (Abbruch) der TCP Verbindung.
- ACK (Acknowledgement): Das Segment enthält eine Bestätigung, d. h. der Wert im ACK#-Feld ist gültig.
- PSH (Push): Der Empfänger muss die Daten im Segment sofort an die Anwendung weitergeben.

TCP Header VI

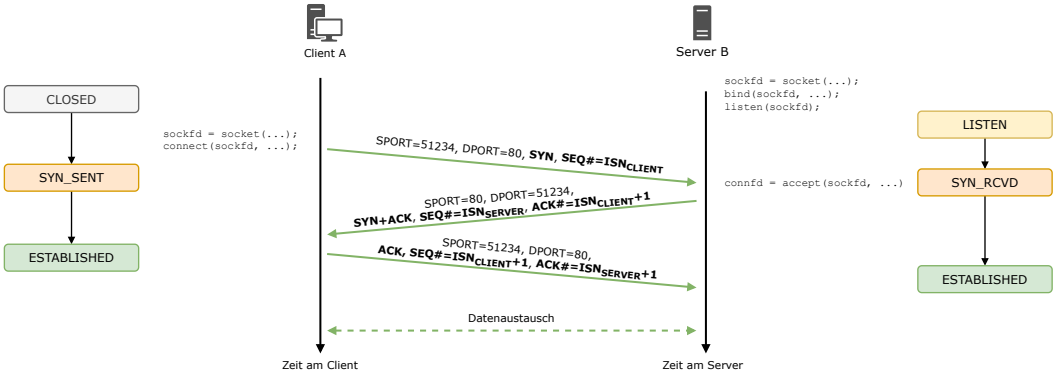
- TCP Optionen
  - Auf den TCP Header kann eine variable Anzahl (0..n) von Optionen folgen, z. B.
  - TCP Maximum Segment Size (MSS) (RFC 6691)
  - TCP Window Scale (RFC 7323)
  - TCP Selective Acknowledgement (SACK) (RFC 2018)
  - TCP Timestamps (RFC 7323)
  - TCP Fast Open (RFC 7413)
- Header Length: Länge des TCP Headers in 32-Bit Wörtern (notwendig weil auf einen TCP Header eine variable Anzahl von Optionen folgen kann).

TCP Connection Management I

- TCP ist verbindungsorientiert → vor der Datenübertragung muss eine Verbindung errichtet und nach dem Ende der Datenübertragung wieder abgebaut werden.
- Die Verwaltung des Verbindungszustands wird durch die TCP Finite State Machine (TCP FSM, [2, S. 22]) durchgeführt.
- Änderungen am Verbindungszustand ergeben sich aus TCP Segmenten mit gesetzten SYN, ACK, FIN und/oder RST Flags.
- Der TCP Verbindungsaufbau wird als TCP 3-Way-Handshake bezeichnet.
  - Zweck: Start der Verbindung, Aushandlung diverser Parameter (z. B. Initial Sequence Number, MSS Option, SACK Option, Window Scale Option, ...).

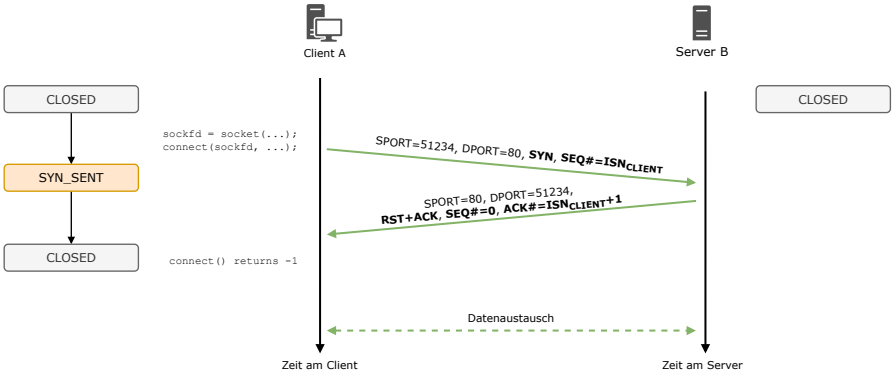
TCP Connection Management II

TCP Verbindungsaufbau I



TCP Connection Management III

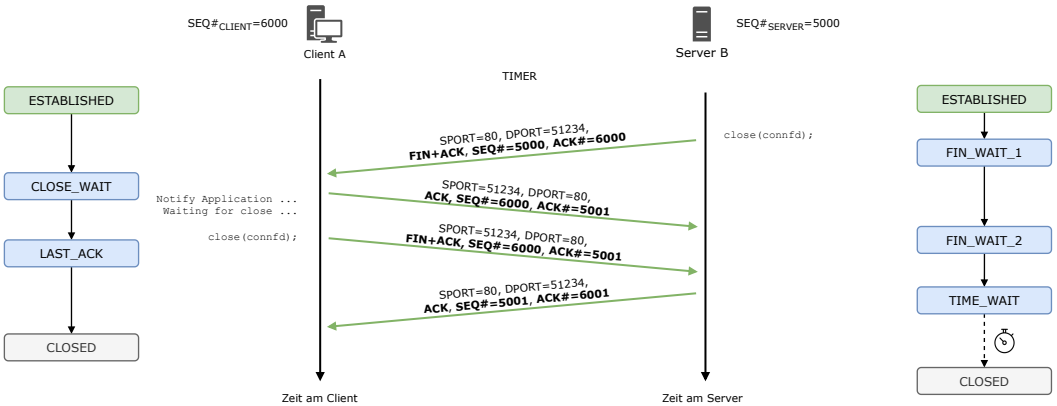
TCP Verbindungsaufbau II



- Initial Sequence Number (ISN) = Sequence Number von der ausgehend alle Bytes einer TCP Verbindung nummeriert werden.
- Nachdem eine TCP Verbindung full-duplex ist, d. h. beide Teilnehmer gleichzeitig Sender und Empfänger sind, wählt jeder Teilnehmer seine eigene ISN (ISN_{CLIENT} und ISN_{SERVER}).
- ISNs müssen immer echt zufällig gewählt werden (32-Bit Zufallswert), da sonst die Gefahr besteht, dass ein Angreifer eine korrekt Sequence Number "erraten" und Datenverkehr in eine TCP Verbindung einschleusen könnte (TCP Sequence Number Guessing, TCP Session Hijacking)!

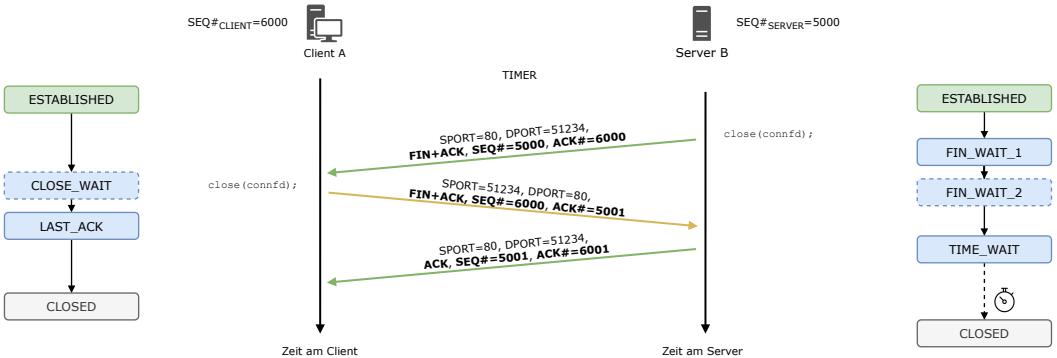
TCP Connection Management V

TCP Verbindungsabbau I



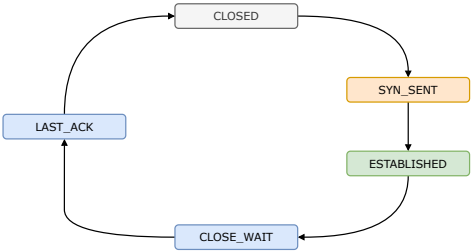
TCP Connection Management V

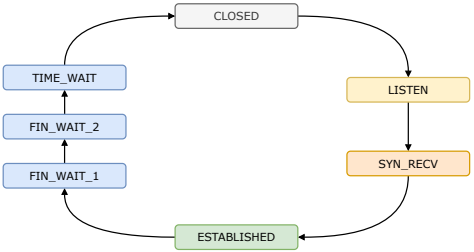
TCP Verbindungsabbau II



TCP Connection Management VI

- Typische TCP Zustandsabfolge für Clients (links) und Server (rechts) (basierend auf [1]):





TCP Zuverlässigkeit (Reliability) I

- Zuverlässigkeit (Reliability) wird bei TCP über Positive Acknowledgement with Retransmission (PAR) realisiert.
- Positive Acknowledgement
  - Der Empfänger gibt dem Sender positive Rückmeldung über erfolgreich empfangene Bytes (Acknowledgement Number, ACK#).
  - Cumulative Acknowledgement = TCP bestätigt immer alles bis zum letzten korrekt empfangenen Byte, d. h. auch schon alle zuvor bestätigten.
- Retransmission
  - Erhält ein Sender innerhalb einer definierten Frist (Retransmission Timeout, RTO) keine Bestätigung für in einem Segment enthaltene Daten, schickt er das Segment erneut.

TCP Zuverlässigkeit (Reliability) II

Retransmission Timeout (RTO)

- Die Berechnung des TCP Retransmission Timeouts (RTO) ist in RFC 6298 [3] beschrieben.
- Problemfeld der RTO Berechnung: RTO zu kurz $\rightarrow$ unnötige Retransmissions, RTO zu lang $\rightarrow$ unnötig hohe Latenzen
- Die Basis bilden laufende RTT Messungen (`SampleRTT, R'`).
  - TCP RTT = Zeit vom Start der Sendeoperation für ein Segment bis zum Empfang der Bestätigung dafür.
- Startwert für RTO = 1 Sekunde, Erreichen des RTOs $\rightarrow$ Verdopplung des RTOs .
- Retransmission Timer läuft nicht für jedes Segment einzeln (zu großer Overhead), sondern immer für eines, für das noch eine Bestätigung ausständig ist.

TCP Zuverlässigkeit (Reliability) III

Acknowledgement Strategien

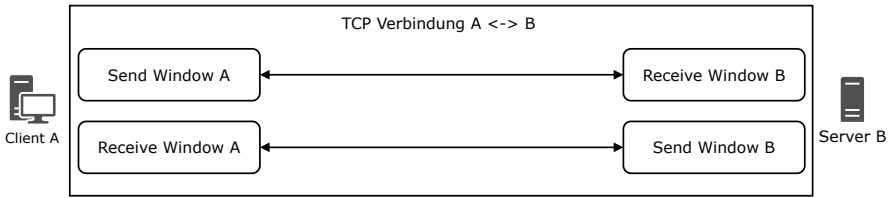
- Eine Bestätigung kann Daten eines oder auch mehrere Segmente umfassen.
- Mögliche Fälle und Strategien (u. a. RFC 5681 [4]):
  1. Segment mit erwarteter SEQ# trifft ein + alle Daten davor sind schon bestätigt → Delayed Acknowledgement = Bestätigung darf um 500 ms verzögert werden.
  2. Segment mit erwarteter SEQ# trifft ein + Daten eines weiteren Segments sind noch nicht bestätigt → sofortiges Senden einer Bestätigung für damit zwei Segmente.
  3. Segment mit unerwarteter SEQ# trifft ein (Lücke) → Duplicate Acknowledgement, d. h. dreimaliges Bestätigen bis zum Byte unmittelbar vor der Lücke (bewirkt Fast Retransmit ohne Abwarten des RTOs).
  4. Segment mit Daten in einer Lücke trifft ein → sofortiges Senden einer Bestätigung soweit als möglich.

TCP Flusskontrolle I

- Flusskontrolle ermöglicht es einem Empfänger einen Sender in dessen Übertragungsgeschwindigkeit zu steuern.
  - Ziel: Ein schneller Sender soll einen langsamen Empfänger nicht überlasten.
- TCP verwendet dafür einen Sliding-Window-Ansatz.
- Der Empfänger signalisiert dem Sender über das Window-Feld im TCP Header wie viele Bytes der Sender senden darf (in einer beliebigen Anzahl von Segmenten), bevor er längstens auf eine Bestätigung durch den Empfänger warten muss.
- Der Empfänger verwaltet dafür eine Receive Window, der Sender ein Send Window.
- TCP ist Full-Duplex $\rightarrow$ beide Teilnehmer sind Sender und Empfänger gleichzeitig $\rightarrow$ beide haben je ein Receive und ein Send Window.

TCP Flusskontrolle II

- Server B bestimmt die Größe seines Receive Windows und damit die Größe des Send Windows von Client A.
- Client A bestimmt die Größe seines Receive Windows und damit die Größe des Send Windows von Server B.

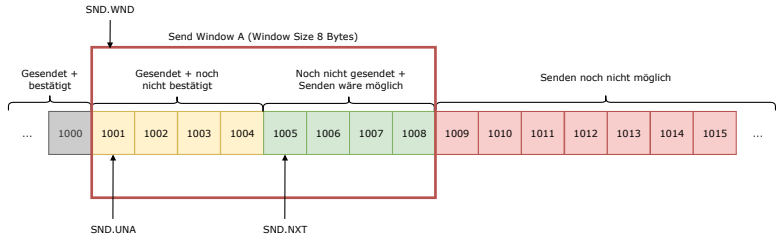


TCP Flusskontrolle III

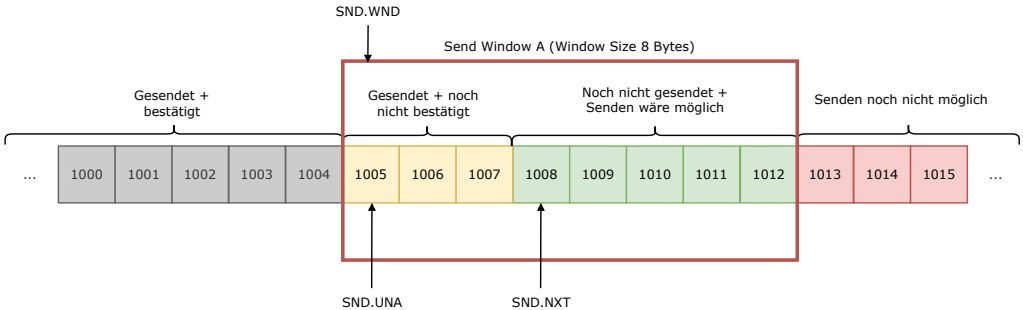
- Der TCP Standard (RFC 793, Section 3.2) definierte eine Reihe von Variablen zur Verwaltung einer TCP Verbindung in einem Transmission Control Block (TCB).
- Für die Flusskontrolle sind dabei u. a. relevant:
  - SND.WND = Größe des Send Windows in Bytes.
  - SND.UNA = Nummer des ersten schon gesendeten aber noch nicht bestätigten Bytes (Send Unacknowledged).
  - SND.NXT = Nummer des nächsten zu sendenden Bytes.
  - RCV.WND = Größe des Receive Windows in Bytes.
  - RCV.NXT = Nummer des nächsten erwarteten Bytes.

TCP Flusskontrolle IV

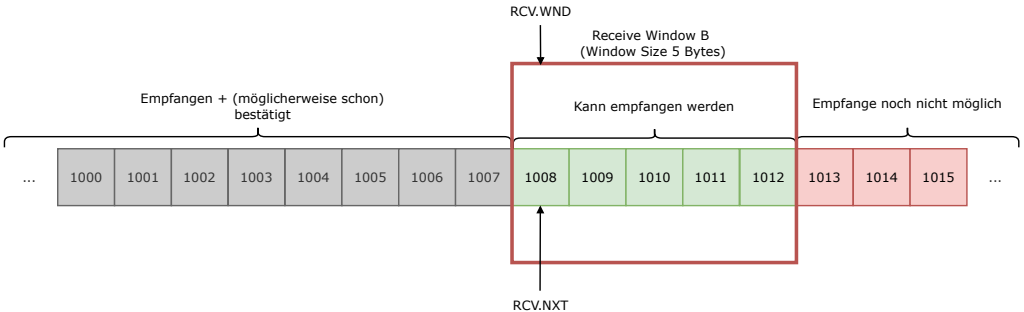
Beispiel I - Ausgangslage TCP Verbindung zw. Client A und Server B



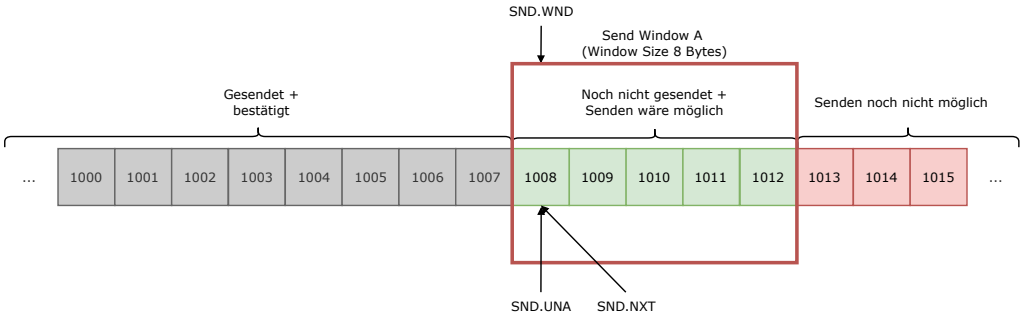
- Server B schickt eine Bestätigung bis inkl. Byte 1004 + Client A empfängt diese + Client A sendet 3 Byte Daten.



- Server B empfängt diese Daten + schickt eine Bestätigung bis inkl. Byte 1007 + schickt eine Aktualisierung der Fenstergröße auf 5 Bytes.



- Client A empfängt die Bestätigung und die Aktualisierung der Fenstergröße.



TCP Segmentation I

- TCP betrachtet die von der Anwendung kommenden Daten als quasi unendlichen Bytestrom und kümmert sich selbst um die Paketierung in Segmente.
- Problemfeld zur Paketierung:
  - Idealfall volles Segment → optimales Overhead-Nutzdaten-Verhältnis
  - Zuwarten auf Daten von der Anwendung → unnötig hohe Latenzen
- Nagle Algorithmus (RFC 1122 [5]):
  1. Volles Segment steht bereit → senden!
  2. Gibt es nicht bestätigte Daten → puffern aller Daten bis alle Daten bestätigt sind oder ein volles Segment bereit steht.
- Der Nagle Algorithmus verbessert die Effizienz (besseres Overhead-Nutzdaten-Verhältnis) führt aber zu größeren Latenzen (Problem bei z. B. Spielen).
- → TCP Implementierungen müssen eine Möglichkeit zur Deaktivierung des Nagle Algorithmus bereitstellen (`TCP_NODELAY` Socket-Option).

TCP Segmentation II

- Wann ist eine TCP Segment "voll"?
  - TCP orientiert sich dabei daran, was sein Gegenüber maximal empfangen kann.
- TCP Maximum Segment Size (MSS) Option (RFC 6691)
  - Die Teilnehmer einer TCP Verbindung signalisieren sich im TCP 3-Wege-Handshake die maximale Größe eines für sie empfangbaren TCP Segments.
  - Diese Größe hängt von der verwendeten Netzwerktechnologie (z. B. Ethernet, WLAN, ...) ab.
  - Beispiel: Ethernet + IPv4 → MSS 1460 Bytes (Ethernet MTU[1] 1500 Bytes - 20 TCP Header - 20 Byte IPv4 Header = 1460 Byte)

[1] MTU = Maximum Transmission Unit = größt mögliche Datenmenge, die über eine Netzwerktechnologie übertragen werden kann (z. B. Ethernet 1500 Bytes) → VO Teil zu Ethernet

TCP Window Scale Option (RFC 7323)

- Das Window-Feld im TCP Header hat eine Länge von 16 Bit $\rightarrow$ die maximale Fenstergröße ist damit $2^{16} = 65535$ Bytes.
- Für moderne Netzwerke (z. B. mit Datenraten > 1 Gbps) ist das klein und ineffizient.
- Die TCP Window Scale Option (RFC 7323) ermöglicht es, den Wert im Window-Feld um einen Faktor 2^x mit $x \leq 14$ zu skalieren.
- Die maximal mögliche Fenstergröße ist somit $(2^{16} - 1) \cdot 2^{14} \approx 1$ GB.
- Die Verfügbarkeit der Option und der Skalierungsfaktor werden im SYN-Segment des TCP 3-Wege-Handshakes signalisert.

TCP - Was noch fehlt ... unter anderem ...

- TCP Selective Acknowledgement Option (RFC 2018)
- TCP Congestion Control (RFC 5681)
- TCP Windows Size Management (Initialgröße, Anpassung, Silly Window Syndrom)
- TCP Fast Open (RFC 7413)
- TCP Timers (behandelt: RTO, Time Wait Timer, weiters: Persistent Timer, Keep Alive Timter).


Fragen
Ceisteanna
Domande
Froen
Vragen
Spørgsmål
Preguntas
Pitanja
Quaestiones
Kérdések
Vprašanja
Frågor
Questions
Kysymykset
Cwestiynau
Spurningar
Galderak
Mistoqsijiet

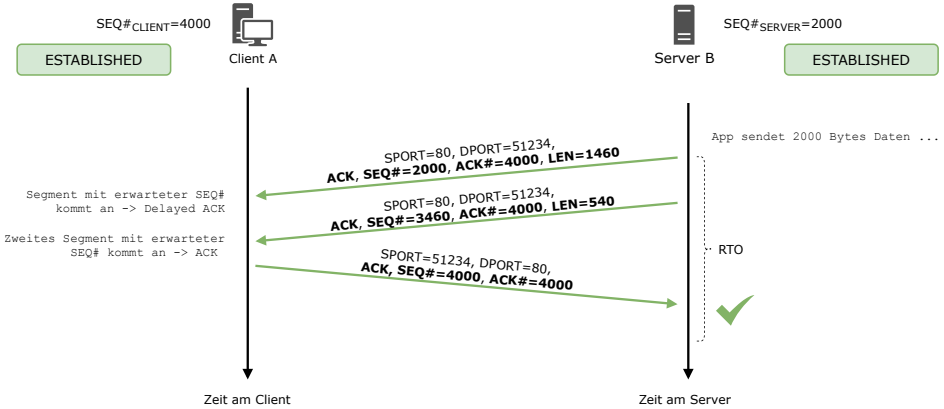
Robustness Principle (John Postel [2])

[...] TCP implementations will follow a general principle of robustness: be conservative in what you do, be liberal in what you accept from others. [...]

- Was bedeutet das Robustness Principle?
- Was ist aus heutiger Sicht davon zu halten (s. [6])?

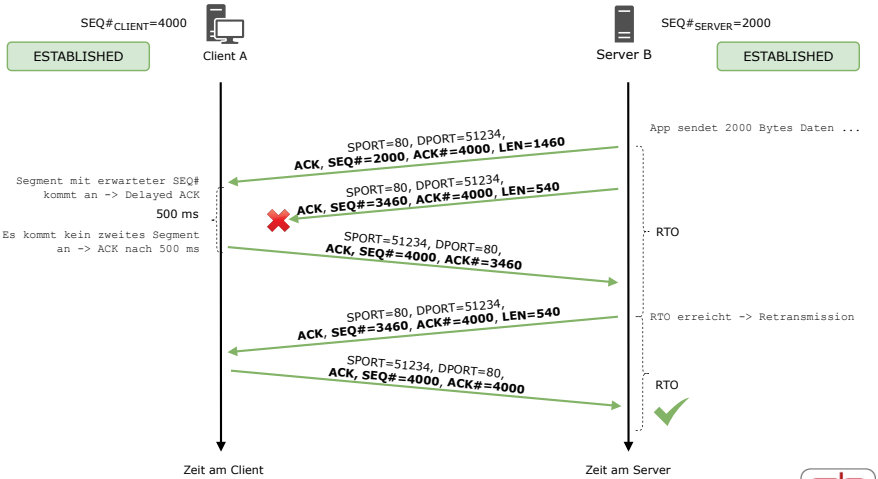
TCP Zuverlässigkeit (Reliability) IV

Beispiel 1 - Alles normal



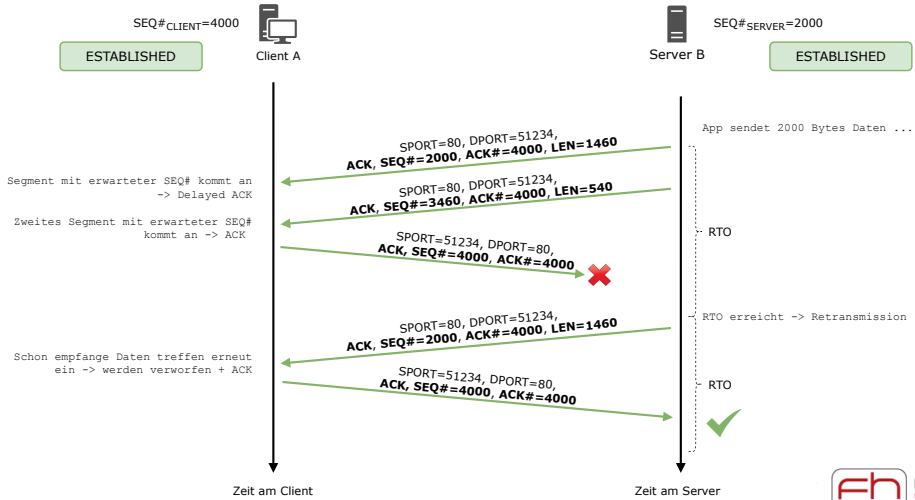
TCP Zuverlässigkeit (Reliability) V

Beispiel 2 - Data lost



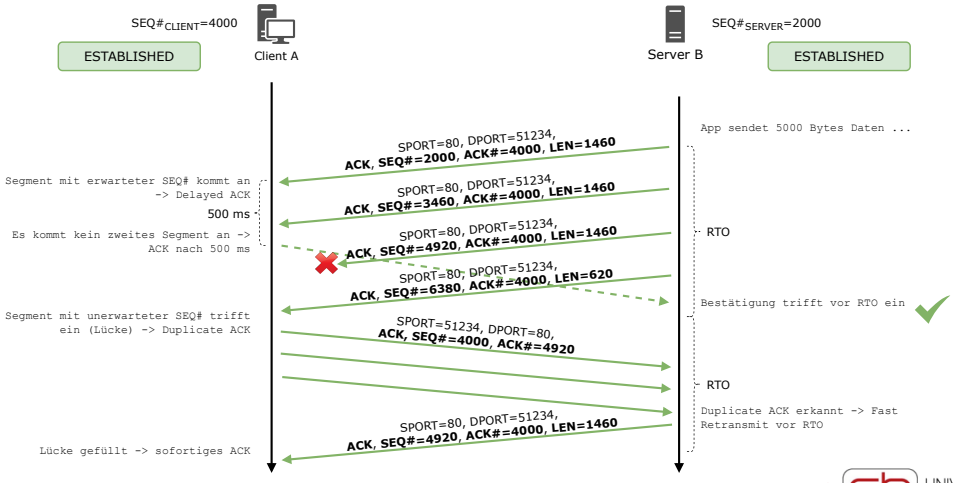
TCP Zuverlässigkeit (Reliability) VI

Beispiel 3 - ACK lost



TCP Zuverlässigkeit (Reliability) VII

Beispiel 4 - Duplicate ACK



Referenzen I

[1] J. Kurose und K. Ross, Computer Networking - A Top-Down Apporach, 7. Aufl. Pearson Education, 2017, S. 852, ISBN: 1-292-15359-8.

[2] J. Postel, „Internet Protocol," Sep. 1981, S. 45. DOI: 10.17487/rfc0791. Adresse: https://www.rfc-editor.org/info/rfc0791.

[3] M. Sargent, J. Chu, D. V. Paxson und M. Allman, Computing TCP's Retransmission Timer, RFC 6298, 2011. DOI: 10.17487/RFC6298. Adresse: https://rfc-editor.org/rfc/rfc6298.txt.

[4] E. Blanton, D. V. Paxson und M. Allman, TCP Congestion Control, RFC 5681, 2009. DOI: 10.17487/RFC5681. Adresse: https://rfc-editor.org/rfc/rfc5681.txt.

Referenzen II

[5] R. T. Braden, Requirements for Internet Hosts - Communication Layers, RFC 1122, 1989. DOI: 10.17487/RFC1122. Adresse: https://rfc-editor.org/rfc/rfc1122.txt.

[6] E. Allman, „The Robustness Principle Reconsidered,“ Commun. ACM, Jg. 54, Nr. 8, S. 40–45, Aug. 2011, ISSN: 0001-0782. DOI: 10.1145/1978542.1978557. Adresse: https://doi.org/10.1145/1978542.1978557.