

50 Jahre Internet – Internetworking 2019
TCP – Fluss- und Überlastkontrolle

Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut
Universität Potsdam

TCP regelt Datenfluss über einer Verbindung mit Hilfe eines Schiebefenster-Protokolls → **Sliding Window Protocol**

- Schiebefenster-Protokoll arbeitet mit adaptiven, lastabhängigen Fenstern – **Windows**, wobei Quittierung und Zuweisung eines Fensters voneinander entkoppelt sind

Beispiel:

Nachricht der Länge 2.500 Byte soll von A nach B über eine TCP-Verbindung gesendet werden. B hat einen Eingangspuffer (Fenstergröße) von 1.500 Byte ($F = 1.500$)

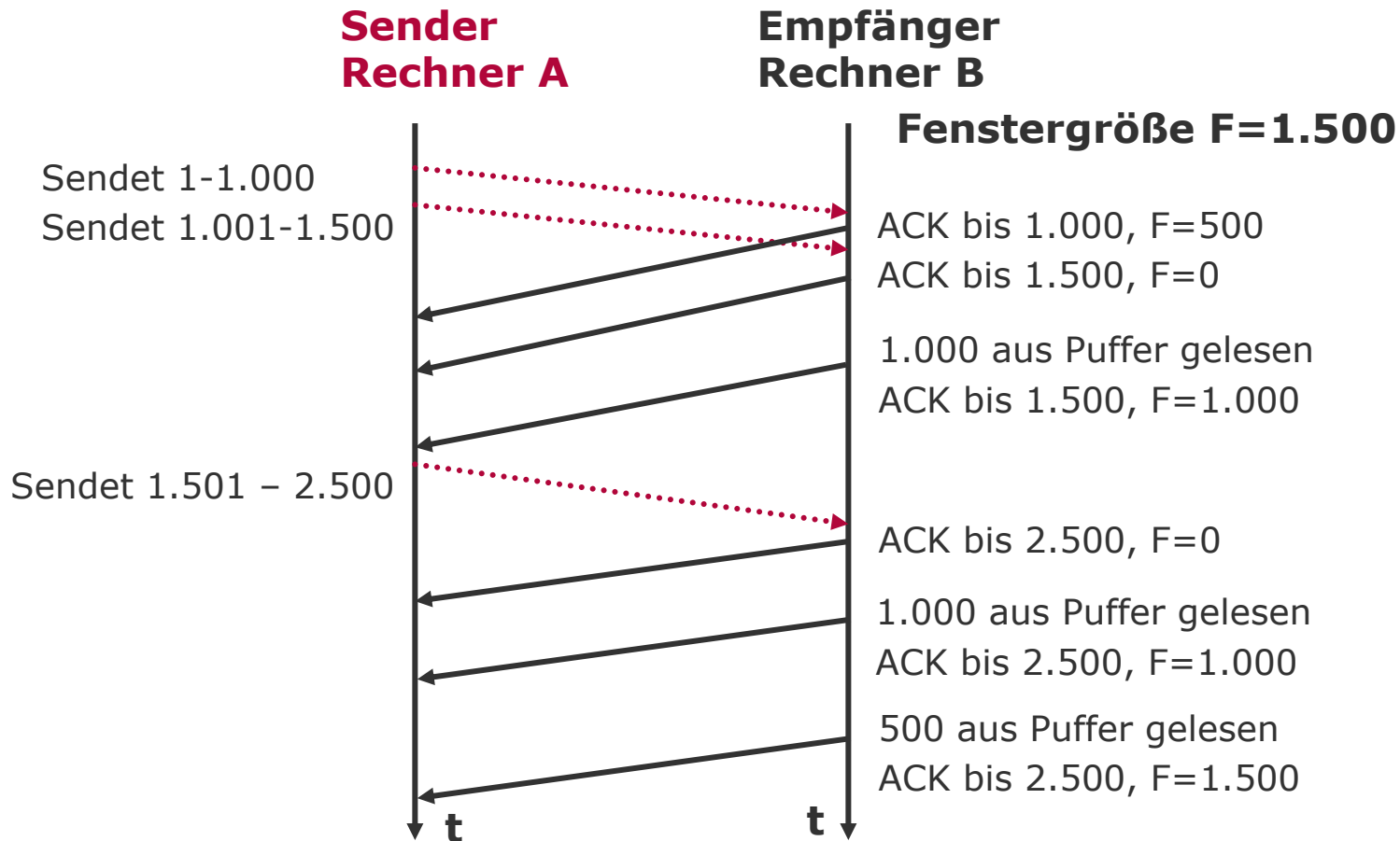
- A sendet erste 1.000 Byte
- B nimmt diese entgegen, schreibt sie in Eingangspuffer, setzt Fenstergröße auf $1.500 - 1.000 = 500$ ($F=500$) und bestätigt Annahme der ersten 1.000 Byte ($ACK=1.000$) und Fenstergröße ($F=500$)

Beispiel:

Nachricht der Länge 2.500 Byte wird per TCP-Verbindung mit Fenstergröße 1.500 Byte von A nach B übertragen

- ...
- A überträgt gemäß erlaubter Fenstergröße nächste 500 Byte
- B nimmt diese entgegen, setzt Fenstergröße auf 0 und bestätigt Empfang und neue Fenstergröße
- A muss nun warten, bis Anwendung auf B aus Eingangspuffer 1.000 Byte ausgelesen hat, Fenstergröße wieder auf 1.000 Byte gesetzt und entsprechende Mitteilung geschickt (ACK=1.500, F=1.000) ist
- A kann nun den Rest der Nachricht (1.000 Byte) schicken

TCP Flusskontrolle: Sliding Window Protocol – Beispiel



- Ohne zusätzliche Vorkehrungen würde sich bei schnellen Verbindungen die Fenstergröße nahe 0 einpegeln und die Übertragungsstrecke nur unzureichend genutzt
- Empfänger gibt deshalb Rückmeldung über verfügbare Fenstergröße erst dann, wenn wieder mindestens 50% der maximalen Fenstergröße zur Verfügung stehen
- Sender sorgt dafür, dass die verfügbare Fenstergröße nicht vollständig ausgenutzt wird

Überlaststeuerung – **Congestion Control** – ist eines der schwierigsten Probleme für TCP, da TCP Überlastsituationen nicht direkt erkennen kann:

- Datenpakete werden per IP auf unterschiedlichen Wegen durch das Internet befördert, sodass keine direkten Rückschlüsse auf Überlast bestimmter Zwischensysteme gezogen werden können
- Verschiedene TCP-Instanzen können nicht miteinander kooperieren

Idee: TCP interpretiert Zahl der verlorenen (also nicht bestätigten) Pakete als Maß für Überlast und Parameter für Bestimmung des Congestion Windows zur Regelung der Überlast

TCP Überlastkontrolle: Slow-Start und Congestion-Avoidance Algorithmus

Slow-Start-Algorithmus – zur schnellen Anpassung des Congestion Windows:

- Verbindungen werden mit kleiner Fenstergröße gestartet, die dann so lange exponentiell vergrößert wird, bis Pakete verloren gehen...

Congestion-Avoidance-Algorithmus – zum Abbau von Überlastsituationen:

- Häuft sich der Paketverlust – d.h. Quittungen erreichen Sender nicht innerhalb der festgesetzten Zeitspanne – senkt TCP die Sende-Datenrate, um eine Überlastung des Netzwerkes nicht mit ständiger Wiederholung von Sendungen zu verschärfen

Nagle-Algorithmus (RFC 896 wurde 2016 durch RFC 7805 ersetzt)

Ziel: Einschränkung des Overheads beim Senden:

- Sind TCP-Pakete nur von kurzer Länge, machen Steuer- und Kontrollinformationen im Header Großteil der zu übertragenden Daten aus
- Quittierung eines Pakets wird deshalb so lange wie möglich verzögert, um diese z.B. mit weiteren Nutzdaten zu senden
- Sendung eigener Nutzdaten wird solange wie möglich verzögert, um diese erst zusammen mit ausstehenden Quittierungen zu versenden
- Nagle-Algorithmus sammelt alle Nutzdaten bis zur nächstfälligen Quittung und überträgt diese dann gemeinsam

RTT – Round Trip Time – wird für jede Aussendung berechnet:
Zeitdauer zwischen Aussendung eines Pakets und seiner Quittierung

- **Originalimplementierung:** RTT wird für jedes gesendete Paket ermittelt und dann ein gewichtetes Mittel errechnet
- **Karn/Partridge-Algorithmus:**
 - Bei Originalimplementierung kann tatsächlicher Wert unterschätzt werden, da fälschlich auch Zeit zwischen Neuaussendung eines verlorenen geglaubten Pakets und seiner verspätet eingetroffenen Quittierung gemessen werden kann
 - Karn/Partridge-Algorithmus zur RTT-Berechnung basiert nur auf Messungen der tatsächlich vom Empfänger quittierten Pakete
 - Zusätzlich wird Zeitschranke nach jedem erfolgreichen Versand angehoben

RTT – Round Trip Time – wird für jede Aussendung berechnet:
Zeitdauer zwischen Aussendung eines Pakets und seiner Quittierung

■ **Karek/Jacobson-Algorithmus**

- Verfeinerung des Karn/Partridge-Algorithmus durch Gewichtung der Schwankungen der gemessenen Netzumlaufzeit
- Zusätzlich wird Berechnung des Timeouts angepasst