Grundlagen: Rechnernetze und verteilte Systeme

Zehnte Woche: 25./27. Juni 2018

TCP

Leo Glavinić
netze@eo.gl
eo.gl/netze

Inhalt

1. TCP: Fluss- und Staukontrolle

2. TCP But It Gets Slower When Data Is Sent

1. Fluss- und Staukontrolle bei TCP

a. Unterschied zwischen Fluss- und Staukontrolle

Flusskontrolle: Verhinderung von Überlastsituation beim Empfänger

Staukontrolle: Reaktion von Überlastsituation im Netz

1. Fluss- und Staukontrolle bei TCP

b. Zuordnung der Begriffe zu TCP-Fluss- bzw. Staukontrolle

Flusskontrolle: Empfangsfenster (Empf. kann dem Sender mitteilen, wie viele Daten er maximal gleichzeitig senden darf

Staukontrolle: Slow-Start, Congestion-Avoidance (Staukontrollphasen einer TCP-Verbindung), Multiplicative-Decrease (Halbierung des Staukontrollfensters bei Segmentverlust)

1. Fluss- und Staukontrolle bei TCP

Im Folgenden: Betrachtung einer zusammenhängenden Übertragung mit bereits abgeschlossener Slow-Start-Phase

Sendefenster W_s, $|W_s| = w_s$

Empfangsfenster W_r, $|W_r| = w_r$

Staukontrollfenster W_c, $|W_c| = w_c$ (Verkleinerung bei Segmentverlust, Vergrößerung bei erfolgreicher Übertragung)

1. Fluss- und Staukontrolle bei TCP

Empfangsfenster sei beliebig groß → Sendefenster = Staukontrollfenster

Keine Verluste, solange Sendefenster kleiner als Maximalwert x ist

Vergrößerung des aktuellen Fensters um genau 1 *MSS* (Maximum Segment Size) bei ACK eines vollständigen Sendefensters

Verlust eines TCP-Segments bei Erreichen von x (Erkennung durch mehrfach erhaltene ACK-Nummer)

1. Fluss- und Staukontrolle bei TCP

Halbierung des Staukontrollfensters bei Verlust, aber Verbleib in der Congestion-Avoidance-Phase; kein erneuter Slow-Start!

→ vereinfachtes TCP-Reno

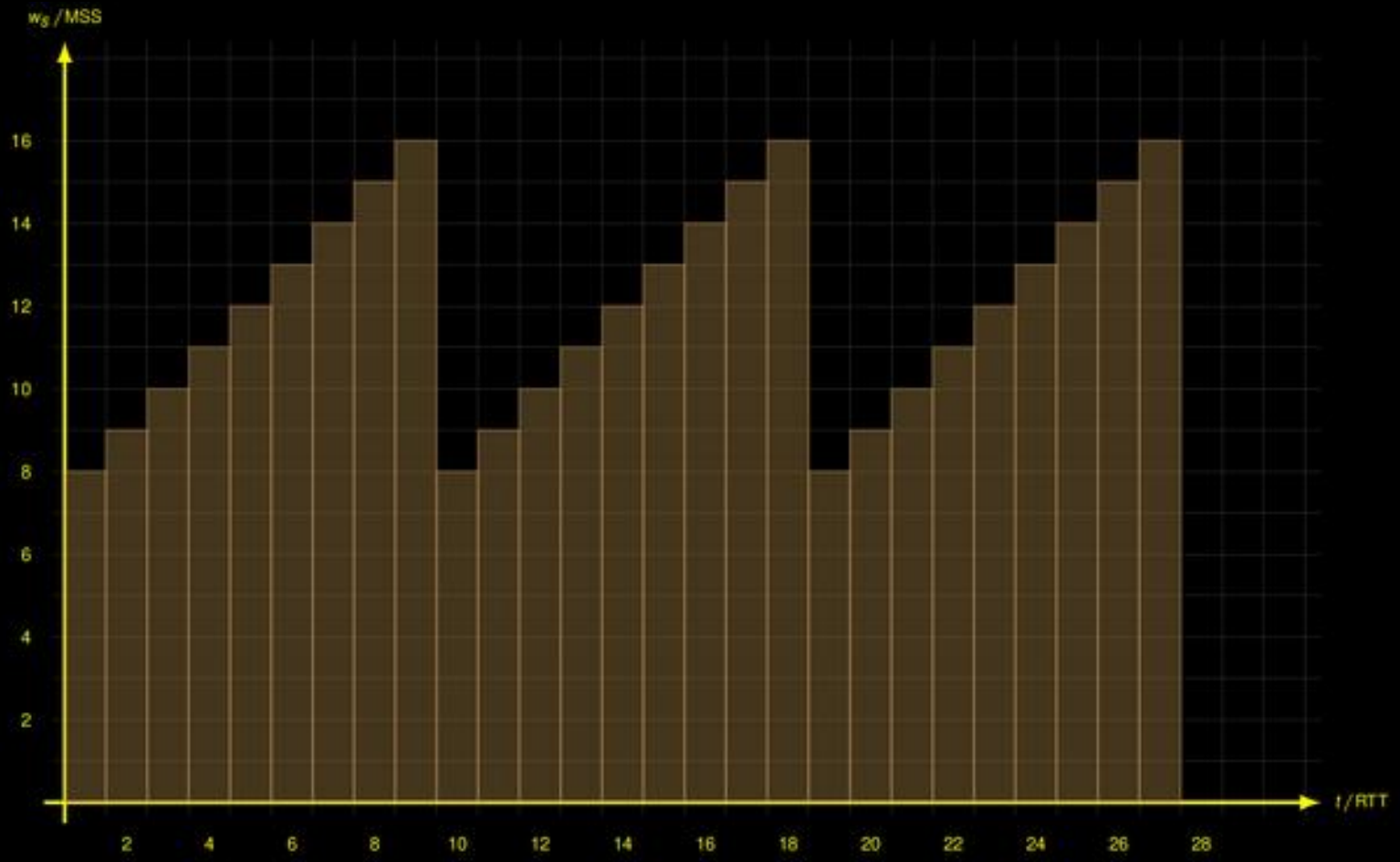
Vereinbarung: MSS=1460 B; RTT=200 ms

Gegenüber Verzögerung vernachlässigbar kleine Serialisierungszeit

x=16 MSS

1. Fluss- und Staukontrolle bei TCP

c. Schaubild von w_s in *MSS* über t in *RTT*; bei t_0=0 s gelte $w_s = x/2$; Zeitintervall t={0, ..., 27}



1. Fluss- und Staukontrolle bei TCP

d. Zeit zwischen zwei Segmentverlusten

Reduktion von w_c auf $x/2$ nach Segmentverlust;
Vergrößerung pro vollständig bestätigtem Fenster um *MSS*

Vernachlässigbar kleine Serialisierungszeit → zu einem Zeitpunkt t_0 können w_c Segmente gesendet werden, die um t_0+RTT bestätigt werden

$T=(x/2+1)\cdot RTT=9\cdot 200$ ms$=1{,}8$ s

1. Fluss- und Staukontrolle bei TCP

e. Durchschnittliche Verlustrate θ pro Periode

Zahl n der übertragenen Segmente pro Periode

$$n = \sum_{i=x/2}^{x} i = \sum_{i=1}^{x} i - \sum_{i=1}^{x/2-1} i = \frac{x \cdot (x+1)}{2} - \frac{\left(\frac{x}{2} - 1\right) \cdot \frac{x}{2}}{2}$$

$$= \frac{x^2 + x}{2} - \frac{x^2}{8} + \frac{x}{4}$$

$$= \frac{3}{8}x^2 + \frac{3}{4}x$$

$$\overset{x=16}{=} 108$$

Ein verlorenes Segment pro Periode → θ=1/108

1. Fluss- und Staukontrolle bei TCP

f. Durchschnittliche Übertragungsrate

$$
\begin{aligned}
r_{TCP} &= \frac{n}{T} \cdot (1 - \theta) \\
&= \frac{108 \cdot 1460\,\mathrm{B}}{1{,}8\,\mathrm{s}} \cdot \frac{107}{108} \\
&= \frac{107 \cdot 1460\,\mathrm{B}}{1{,}8\,\mathrm{s}} \\
&= \frac{1562200}{18}\,\mathrm{B/s} \\
&\approx \frac{1562}{18}\,\mathrm{kB/s} \approx 87\,\mathrm{kB}
\end{aligned}
$$

1. Fluss- und Staukontrolle bei TCP

g. Maximale Übertragungsrate für UDP ohne Stauung (UDP-Header 12 B kleiner als TCP-Header)

Verlässliche Übertragung von 15 *MSS*; Segment trägt bei UDP 12 B mehr Nutzdaten

$$r_{UDP} = \frac{15 \cdot (\mathrm{MSS} + 12\,\mathrm{B})}{\mathrm{RTT}}$$

$$= \frac{15 \cdot (1460\,\mathrm{B} + 12\,\mathrm{B})}{0{,}2\,\mathrm{s}}$$

$$= \frac{15 \cdot 1472\,\mathrm{B}}{0{,}2\,\mathrm{s}}$$

$$= 15 \cdot 7{,}36\,\mathrm{kB/s} \approx 110\,\mathrm{kB/s}$$

2. TCP und Long Fat Networks

Long Fat Networks: Verbindungen mit hoher Übertragungsrate und hoher Verzögerung (z.B. Satellitenverbindungen)

a. TCP-Sendefenster wird in Abhängigkeit von Empfangsfenster und Staukontrollfenster gewählt

$w_s = \min\{w_r, w_c\}$

2. TCP und Long Fat Networks

Anbindung zweier Nutzer über geostationären Satelliten; RTT=800 ms, r=24 Mbit/s

b. Größe des Sendefensters in B für kontinuierliche Sendung

Eintreffen des ersten ACK frühestens nach einer RTT

$w_s \geq RTT \cdot r = 800 \cdot 10^{-3}\ \text{s} \cdot 24 \cdot 10^6\ \text{bit/s} = 100 \cdot 24 \cdot 10^3\ \text{B/s}$

$= 2{,}4\ \text{MB}$

2. TCP und Long Fat Networks

c. Warum ist dies ein Problem für die Flusskontrolle?

Sendefenster: Minimum aus Empfangs- und Staukontrollfenster

Mitteilung des Empfangsfensters über Receive-Window-Feld (16 bit) → maximale Sendefenstergröße (2^{16}-1) B=65535 B

Aufg. b: Sendefenster der Größe $2{,}4 \cdot 10^6$ B

2. TCP und Long Fat Networks

d. Lösung für dieses Problem in TCP

Option für TCP-Window-Scaling: Skalierung des Empfangsfensters mit 2^x

Angabe von x in Feld „shift.cnt“ der Option

2. TCP und Long Fat Networks

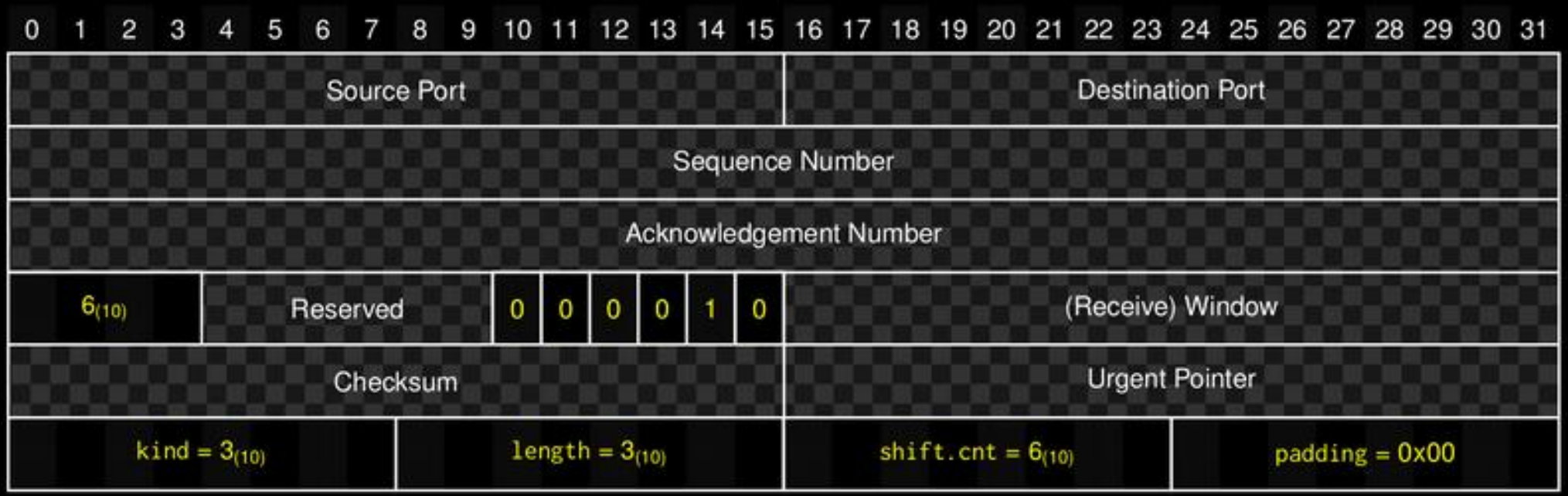
e. Minimaler Wert für shift.cnt

Kleinster Exponent x, sodass das maximale Receive-Window größer als 2,4 MB ist

$$
\begin{aligned}
(2^{16} - 1) \cdot 2^x &\geq 2,4 \cdot 10^6 \\
x &\geq \mathrm{ld}\left(\frac{2.4 \cdot 10^6}{2^{16} - 1}\right) \\
&\approx \mathrm{ld}\left(\frac{2.4 \cdot 2^{20}}{2^{16}}\right) \\
&= \mathrm{ld}\left(2.4 \cdot 2^4\right) \\
&\approx 5{,}26 \\
&\Rightarrow x = 6
\end{aligned}
$$

2. TCP und Long Fat Networks

f. Header des ersten TCP-SYN-Pakets



2. TCP und Long Fat Networks

Größe des Staukontrollfensters: 1,2 MB; *MSS*=1200 B; momentan: Congestion-Avoidance-Phase

g+h. Dauer, bis das Fenster die Leitung komplett ausnutzen kann (evtl. Problem?)

Vergrößerung des Fensters pro *RTT* um 1 *MSS*

T=(1,2·10^6 B/1200 B)·0,8 s=96·10^2/12 s=800 s

Mehr als zehn Minuten! → TCP sucks

3. Code Demos

„Hausaufgabe“

Codekenntnis kommt auch in der Klausur dran!

„man [syscall]“ in Bash oder Google eingeben

udpchat, tcpchat sind auch Stoff!