Marek Kubica kubica@in.tum.de, Michael Kerscher kerschem@in.tum.de

Punkteverteilung:

24	25	26	Σ

Aufgabe (24)

(a) Flusssteuerung soll präventiv einen Stau im Netz verhindern, Staukontrolle soll reaktiv einen entstandenen Stau auflösen. Zusammen soll der Durchsatz des Netzes optimiert werden.

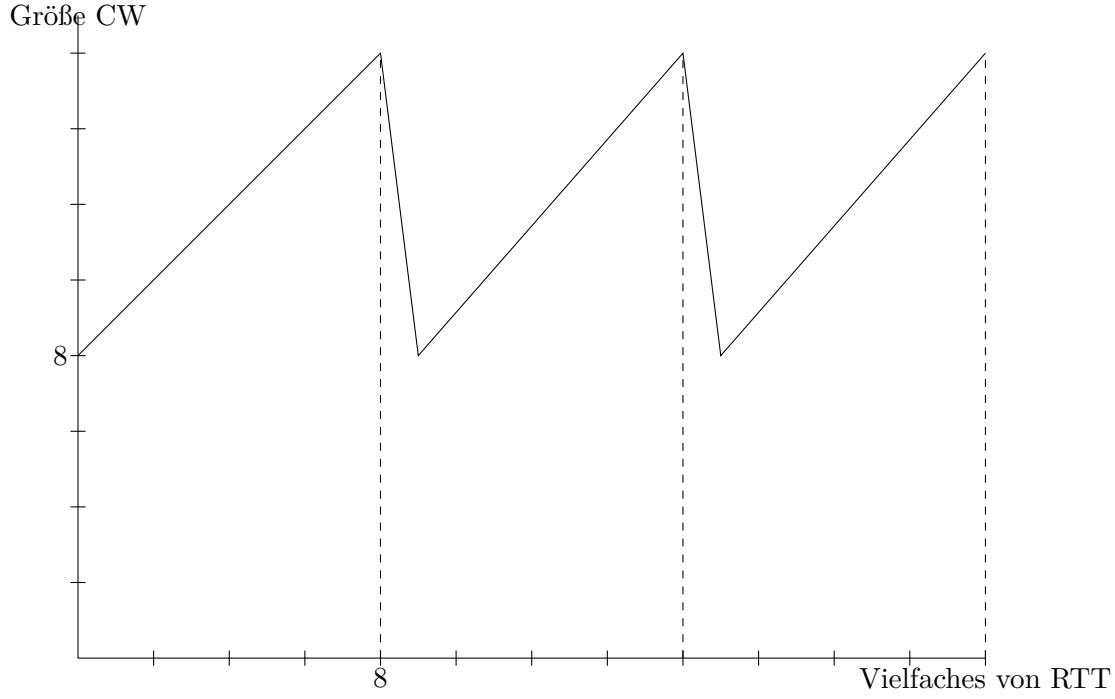
(b) Slow-Start (Staukontrolle): Bei Staus (Paketverlust) wird das Receiver-Window auf 1 gesetzt, so dass zusätzliche Stauung vermieden wird.

Receiver-Window (Flusssteuerung): Der Empfänger teilt dem Sender mit, wie viele Bytes er momentan noch empfangen will/kann und hilft so Staus vorzubeugen, indem er sichergeht dass nicht zu viele Daten verschickt werden.

Congestion-Avoidance (Staukontrolle): Nachdem der Schwellwert des Receiver-Windows überschritten wurde, wird das Fenster pro quittiertem, empfangenem Paket nur noch um 1 erhöht. Dadurch soll bei Staus sichergestellt werden, dass nicht noch mehr Daten erzeugt werden, die den Stau vergrößern.

Multiplicative Decrease (Staukontrolle): Der Schwellwert des Receiver-Windows wird bei Paketverlust halbiert, wodurch die Anzahl der Daten verringert wird und der Stau nicht verstärkt wird.

(c)



Aufgabe (25)

(a) Man verwendet das kleinere der beiden Fenster, da dieses den Flaschenhals darstellt.

(b) Die RTT in der Verbindung beträgt 800ms, deshalb wählt man das Sendefenster so, dass die 800ms mit voller Bandbreite gesendet werden kann.

$$\text{Sendefenster} = 24\frac{\text{MBit}}{s} \cdot 0.8s = 19.2\text{MBit} = 2.4\text{MB}$$

(c) Im TCP-Header stehen nur 16 Bit für die Windowsize zur Verfügung, deshalb sind nur Fenstergrößen bis max. 65535 Byte möglich. Der in b) berechnete Wert ist jedoch größer, deshalb gibt es hier ein Problem.

(d) In RFC1323 ist eine Möglichkeit definiert, wie man Fenstergrößen mit einer Option skalieren kann. Damit sind Fenstergrößen bis 2^{30} (2^{16} um 14 Bits geschiftet).

Als konkrete Zahlen kann man als Windowsize 37500 nutzen und als TCP Window Scale Option (WSopt) verwendet man Kind = 3, Len = 3, shift.cnt = 6.

(e) Bei einer Fenstergröße von 1.2MB können $\frac{1.2MB}{1500Byte} = 800$ Segmente in einer Phase mit der Dauer 0.8s verschickt werden. Dann wird nach jeder Phase ein Segment mehr verschickt, bis das Maximum von 1600 Segmenten erreicht ist. Dies geschieht nach $800 \cdot 0.8s = 640s$. Das ist ein Problem, da die volle Auslastung erst nach recht langer Zeit erreicht wird und der Kanal anfangs nur zur Hälfte genutzt werden kann.

(f) Linux hat seit 2.6.19 die Variante TCP-Cubic. Hierbei wird im Vergleich zu TCP-Reno die Fairness verbessert (bei kleinerer Round-Trip-Time und langsamen Verbindungen). Die Fenstergröße wird hierbei durch eine kubische Funktion bestimmt.

Aufgabe (26)

(a)

(b)
a) DNS-Anfrage: who is server.i8.tum?
Identification: 0x0001, Flags: 0, Question: server.i8.tum
b) DNS-Anfrage: who is server.i8.tum?
Identification: 0x0002, Flags: 0, Question: server.i8.tum
c) DNS-Antwort: query ns.tum!
Identification: 0x0002, Flags: 1, Authority: name: tum, data: ns.tum
d) DNS-Anfrage: who is server.i8.tum?
Identification: 0x0003, Flags: 0, Question: server.i8.tum
e) DNS-Antwort: query ns.i8.tum!
Identification: 0x0002, Flags: 1, Authority: name: i8.tum, data: ns.i8.tum
f) DNS-Anfrage: who is server.i8.tum?
Identification: 0x0003, Flags: 0, Question: server.i8.tum
g) DNS-Antwort: server.i8.tum is ...
Identification: 0x0003, Flags: 1, Answer: name: server.i8.tum, date: <ip> Authority: name: i8.tum, date: ns.i8.tum
h) DNS-Antwort: server.i8.tum is ...
Identification: 0x0003, Flags: 1, Answer: name: server.i8.tum, data: <ip> Authority: name: i8.tum, data: ns.i8.tum
i) Client kennt nun die IP und kann mit dem Server kommunizieren.