Universität
Paderborn
Fachgebiet
Rechnernetze

Wintersemester 2019/2020

Rechnernetze

Übung 10:

2020-01-15

---

1. **TCP Verständnisfragen allgemein**

   (a) TCP ermöglicht einem Empfänger, die maximum segment size (MSS), die der Empfänger akzeptieren möchte, an den Sender zu übermitteln. Warum? Wie verhält sich das zum advertised receiver window?

   (b) Führt ein verloren gegangenes TCP ACK stets zu einer Übertragungswiederholung?

   (c) Kann ein TCP-Zustandsautomat beliebig lange im Zustand FIN WAIT 2 verbleiben?

   (d) Das Dateiübertragungsprotokoll FTP benutzt für Daten und für Kommandos separate TCP-Verbindungen. Warum?

2. TCP-Fairness: Bekanntermaßen stellt additive increase multiplicative decrease (AIMD) Fairness zwischen TCP-Flüssen her.

   Nehmen Sie nun an, zwei Benutzer A und B teilen sich dieselbe Leitung mit einer Kapazität von 6 Mbit/s. Beide wollen jeweils eine große Datei von 36 GBit Größe über TCP herunterladen. Benutzer A benutzt einen herkömmlichen Download-Client, der *einen* TCP-Fluss benutzt. Benutzer B benutzt einen anderen Download-Client, der die Datei in fünf gleichgroße Teile aufteilt und *gleichzeitig fünf* TCP-Flüsse benutzt.

   Vereinfachungen: Nehmen Sie an, der Server wäre schnell genug, um die Leitung zu sättigen. Nehmen Sie an, TCP könnte hellsehen und bräuchte keine Einschwingzeiten für Fairness, Schätzung der verfügbaren Datenrate und Fenstergröße, sodass konstant 6 MBit/s auf der Leitung fließen, solange mindestens ein TCP-Fluss besteht.

   Wie viele Sekunden dauert der Download für Benutzer A, wie lange für Benutzer B?

3. **Token bucket** Eine Möglichkeit der Flußkontrolle ist die Senderatenkontrolle mittels eines so genannten “token bucket”. In diesen “Eimer” werden regelmäßig mit der Rate ρ Tokens gelegt, die jeweils die Erlaubnis zum Senden einer bestimmten Datenmenge d repräsentieren. Der Eimer hat nur eine endliche Kapazität σ. Tokens, die in einen vollen Eimer gelegt werden sollen, werden ignoriert, d.h. der Eimer läuft über. Ein Datenpaket kann nur gesendet werden, wenn mindestens soviele Token im Eimer vorhanden sind, wie der Größe des Pakets entsprechen.

   (a) Was ist die maximale Datenmenge, die ein mit einem solchen token bucket Algorithmus regulierter Sender innerhalb einer Zeitspanne t maximal senden kann?

   (b) Nehmen Sie an, dass $\rho = 1\ \mathrm{s}^{-1}$, $d = 200$ Bytes, $\sigma = 1000$ Bytes. Wie lange muss ein Paket der Größe 800 Bytes mindestens und höchstens warten, wenn bei seinem Eintreffen an der Flußkontrolle Tokens für 100 Bytes vorhanden sind und wenn sonst keine anderen Pakete zur Übertragung anstehen?

   (c) Stellen Sie sich vor, Sie möchten über einen solchen Kanal ein komprimiertes Video abspielen. Da Sie für höhere Werte von ρ und σ bezahlen müssen, sind Sie an möglichst “kleinen” Werten hierfür interessiert. Wenn Sie die durchschnittliche Senderate A des Videos sowie die durchschnittliche Spitzenrate P kennen, wie wählen Sie dann (qualitativ!) die Werte ρ und σ sinnvollerweise?

   (d) Nehmen Sie an, ein Router wird von drei Flows benutzt. Für jeden Flow gibt es einen eigenen token Bucket, mit Parametern wie in Tabelle 1 gezeigt (jeweils in Paketen bzw. Paketen pro Sekunde angegeben; alle Pakete gleich groß). Alle Flows benutzen den gleichen outgoing link; dieser Link kann alle 100 ms ein Paket weiterleiten. Der Link wird durch eine FIFO-Warteschlange gespeist, in der Paket nach passieren des Token Buckets eingereiht werden.

Flow number	Rate ρ	Bucket σ
1	1	10
2	2	4
3	4	1

Tabelle 1: Parameter der Token Buckets

   Zeigen Sie grafisch, in einem Diagramm mit Anzahl Paketen aufgetragen über einer Zeitachse:

   – Die für jeden Flow maximal ankommenden Pakete; nehmen Sie an, dass jeder Flow zu $t = 0$ seine Bucket voll ausnutzt (maximaler Burst)

- Die insgesamt maximal ankommenden Pakete
- Die durch den Router maximal abgearbeiteten Pakete

Überlegen Sie sich, die horizontale Abstand (auf der Zeitachse) und der vertikale Abstand zwischen der Abarbeitungskurve und der maximalen-Ankunfs-Kurve zu interpretieren sind.

Horizontal: Wartezeit. Vertikal: Warteschlangenlänge.

4. **OSPF Routing unter Linux**

   In dieser Aufgabe ersetzen wir die statisch konfigurierten Routen aus der vorherigen Aufgabe durch Routen, die von OSPF-basierten Routing Daemons automatisch bestimmt werden. Dazu benutzen wir eine etwas kompliziertere Version der Topologie aus der vorherigen Aufgabe.

   Gehen Sie wie folgt vor (Sie brauchen für praktisch alles hier root-Rechte): :

   (a) Betrachten Sie das Vagrantfile unter `c_three_routers`. Erstellen Sie eine Skizze, welche Rechner und Router mit welchen Netzen verbunden sind und welche IP-Adressen dort jeweils benutzt werden.

   (b) Installieren Sie auf den Routern die `Quagga` Routing-Suite `http://www.nongnu.org/quagga/`. Dies gelingt am einfachsten mit den Paketen `quagga` und `quagga-doc`.

   (c) Konfigurieren Sie auf den Routern die Quagga-Daemonen so, dass `zebra` und `ospfd` laufen. Hinweis: `/etc/quagga/daemons`.

   (d) Kopieren Sie die Beispiel-Konfigurationsdateien für `zebra` und `ospfd` von `/usr/share/doc/quagga/examples/` nach `/etc/quagga/` (entfernen Sie dabei natürlich das .sample aus dem Dateinamen). Die Dateien sollten dem Nutzer `quagga` und der Gruppe `quaggavty` gehören und Zugriffsrechte 640 haben. (Hinweis: `chown` und `chmod`).

   (e) Die Quagga-Daemonen können Sie über Systemdienste starten: `service quagga start|stop|restart|status`.

   (f) Sie müssen dann den OSPF-Daemon konfigurieren (in `/etc/quagga/ospfd.cont`): Fügen Sie Beschreibungen der Interfaces hinzu und eine Beschreibung des OSPF-Router-Verhaltens. Darin müssen Sie Netz-Präfixe aufzählen und diese einem Area zuordnen. Siehe `http://www.nongnu.org/quagga/docs/docs-info.html#Configuring-ospfd` (insbesondere: OSPF Command: network).

   (g) Danach müssen Sie `zebra` konfigurieren, in `/etc/quagga/zebra.conf`. Darin müssen Sie jedes Interface aufführen, über das Routing geschehen soll. Für ein Interface müssen Sie die IP-Address samt Netzmaske angeben. Siehe `http://www.nongnu.org/quagga/docs/docs-info.html#`

`Interface-Commands`. Zusätzlichen können Sie hier auch direkt IP forwarding aktivieren

(h) Starten Sie auf den Routern die Quagga-Daemonen neu (`service quagga restart`; ein `service quagga status` kann nie schaden).

(i) Können Sie jetzt schon einen Ping zwischen verschiedenen PCs senden? Welche Einstellung auf den PCs fehlt (vergleiche vorherige Aufgabe).

(j) Über das Werkzeug `vtysh` können Sie sich Konfigurationsstände anzeigen lassen. Siehe `http://www.nongnu.org/quagga/docs/docs-info.html#Virtual-Terminal-Interfaces`. (Ein `export VTYSH_PAGER=more` vor dem Aufruf kann hilfreich sein, um einen Bug zu umgehen.)

(k) Schauen Sie sich neben ping auch den Pfad an, den die Pakete von einem PC zum nächsten nehmen (`tracepath`).

Hinweise: Das Booten kann bei langsameren PCs recht lange dauern. Erhöhen Sie ggf. den Timeout-Wert im Vagrantfile.

5. **Zusammenhang Portnummer und Prozess-ID**

   Typischerweise laufen in einem System wie Linux einige Prozesse, die als Server auf ankommende Pakete an einem Port warten. Suchen Sie nach einem Werkzeug, mit dem Sie unter Linux alle solche Prozesse aufzeigen lassen können. Wie können Sie damit herausfinden, welche Prozesse (Prozess-ID) an welchem Port wartet?

6. **tcpdump**

   Benutzen Sie `tcpdump`, um eine TCP-Verbindung anzeigen zu lassen (z.B. `tcpdump host www.uni-paderborn.de`). (Schauen Sie auch Option -S an!) Benutzen Sie beispielsweise `wget`, um kontrolliert Datenübertragungen zu initiieren. Stellen Sie die Folge von Sequenznummern für Daten und ACKs als Funktion der Zeit da (z.B. mittels gnuplot o.ä.).

   Was können Sie aus dem Plot ablesen? Können Sie z.B. Rückschlüsse auf Ihre TCP-Version ziehen?