Übung 10

Betriebs- und Kommunikationssysteme
Sommersemester 2021
Prof. Dr.-Ing. Jochen Schiller

Abgabe 02.07.2021, 10:00

Aufgabe 1: Transportschicht

Geben Sie ggf. Ihre genutzten Quellen an.

1. Erläutern Sie den Unterschied zwischen TCP und UDP. Welche anderen IP-basierten Protokolle gibt es?

2. Was passiert, wenn ein Paket mit einer TTL abgesendet wird, die nicht ausreicht um das Ziel zu erreichen? Was für Fehlerinformationen bekommen wir und wie?

   Angenommen wir senden bspw. UDP-Pakete zu einem Ziel los, und fangen dabei mit einer TTL von eins an. Wenn das Paket nicht ankommt, senden wir nochmal mit einer höheren TTL – bis das Paket ankommt. Warum könnte man das machen wollen außer um schlicht herauszufinden wie „weit“ es zum Ziel ist?

3. Wie sieht ein typischer Verbindungsauf- und -abbau bei TCP aus? Warum gibt es einen Handshake „in beide Richtungen“? Was soll ein **SYN** signalisieren, was **FIN**?

4. Angenommen Sie wollen sich von der KVV-Seite den neuesten Übungsbogen herunterladen und verfolgen zeitgleich einen Livestream. Weiterhin bestehen keinerlei technische Probleme in Ihrer Verbindung zum FU-Server. Wie kann es dennoch passieren, dass Sie den Übungsbogen nicht von der KVV-Seite herunterladen können?

5. Nennen Sie sowohl für UDP als auch TCP je mindestens zwei Beispielanwendungen, die dieses Protokoll verwenden. Begründen Sie, warum dies für diese Anwendung sinnvoll ist.

Aufgabe 2: httpd

In dieser Aufgabe sollen Sie ihren Fileserver zu einem Webserver erweitern. Dieser soll auf Port 8080 hören und dort Verbindungen entgegennehmen. Der Server soll mehrere simultane Verbindungen verarbeiten können. Sie brauchen zum Lösen dieser Aufgabe nicht das HTTP-Protokoll zu verstehen! Wichtig ist nur, dass Sie aus der ersten Zeile der Anfrage (des Browsers) den Dateinamen parsen, den Rest können Sie komplett ignorieren. Der Server soll immer mit folgendem Header vor den eigentlichen Daten (`<CONTENT>`) antworten:

```
HTTP/1.0 <STATUS>\r\n
Content-Type: <MIME>\r\n
Connection: close\r\n
Content-Length: <LEN>\r\n
\r\n
<CONTENT>\r\n
```

Dabei ist `<STATUS>` entweder `200 OK` für ausgelieferte Dokumente oder `404 Not Found` für nicht gefundene Dokumente.

`<MIME>` ist der Typ der Rückgabe, also `text/html` für Webseiten bzw. `image/jpeg` oder `image/gif` für Grafiken. Andere Typen braucht der Server nicht zu unterstützen – es soll hier reichen, schlicht die Dateiendung anzugucken.

`<LEN>` ist die Länge des zurückgelieferten Contents, also exklusive des Headers. Nach dem Header senden Sie eine Leerzeile `\r\n` und dann den Dateiinhalt bzw. die Fehlerseite im Fehlerfall. Danach schließen Sie den Socket.

Vorgeschlagene Vorgehensweise

1. Öffnen Sie einen Server-Socket.
2. Versetzen Sie den Socket in den Listen-Status und merken Sie sich die Rückgabe (den Deskriptor).
3. Nutzen Sie `select()` oder `fork()`
4. Programmieren Sie eine Endlosschleife mit folgendem Inhalt:
   a) Falls es eine Anfrage gibt, akzeptieren Sie diese und lesen Sie mittels `read()` von dem erhalten Descriptor die Anfrage ein.
   b) Parsen Sie den Dateinamen aus der Anfrage und geben Sie eine entsprechende Antwort (Grafik, Page oder Fehler), inkl. Header zurück.
   c) Schliessen Sie die Verbindung!

Testen und dokumentieren Sie Ihren Webserver mit einer einfachen HTML-Seite die mindestens 4 Bilder enthält, indem Sie auf die URL `http://localhost:8080/foo.html` navigieren – angenommen der Server horcht auf Port 8080 und Ihre HTML-Datei heißt `foo.html`. Sie werden sehen, dass Ihr Browser 4 Verbindungen zu Ihrem Server gleichzeitig aufbaut um die Bilder zu erhalten.

Wenn Ihr Server läuft, haben Sie einen performanten, robusten (hoffen wir!) HTTP-Server programmiert, den Sie so auch in eigenen Projekten einsetzen können. *Herzlichen Glückwunsch!*