TKN Telecommunication Networks Group

Praktikum
Rechnernetze und Verteilte Systeme

Block 2

— Verbindungsorientierte und verbindungslose Datenübertragung —

Termin: 29.10.-2.11.2018 & 5.-9.11.2018

1 Theoretische Vorbereitungsaufgaben

Die folgenden Aufgaben sollen Ihnen helfen, sich auf den **Vorbereitungstest** vorzubereiten:
Aufgabe 1:
Machen Sie sich mit den Unterschieden zwischen verbindungsorientierter und verbindungsloser Datenübertragung vertraut und beantworten Sie jeweils für ”Datagram-Service” und ”zuverlässigen Byte-Strom” die folgenden Fragen:

- Ist garantiert, dass Pakete der gleichen Route folgen?
- Wird ein Verbindungsaufbau benötigt?
- Kommen Pakete korrekt an?
- Kommen Pakete in der richtigen Reihenfolge an?
- Was ist der Unterschied zwischen einem Datagram und einem Stream?

Aufgabe 2:
Machen Sie sich mit der Berkeley Socket API vertraut und beantworten Sie die folgenden Fragen:

- Geben Sie jeweils für die folgenden API-Aufrufe an, was diese genau bewirken:
  - socket
  - bind
  - listen
  - accept
  - connect
  - send
  - recv
  - close
- Skizzieren Sie den Ablauf der API-Aufrufe einer Stream Socket Kommunikation zwischen Client und Server unter Verwendung von Berkeley Sockets im fehlerfreien Fall. Wiederholen Sie die Aufgabe ebenfalls für Datagram Sockets.

Aufgabe 3:
Wozu werden Portnummern verwendet?

2 Praktische Aufgaben

Die praktischen Aufgaben sind in Kleingruppen von i. d. R. 4 Personen zu lösen. Die Ergebnisse des ersten Termins führen Sie im zweiten Termin dem Tutor vor. Im zweiten Termin werden vertiefende praktische Aufgaben gestellt, während der Tutor Lösungen des ersten Termins abnimmt. Reichen Sie bitte den Quelltext bzw. Lösungen dieser Aufgaben bis Sonntag vor dem nächsten Termin 23:55 Uhr per ISIS ein. Es besteht in beiden Terminen grundsätzlich Anwesenheitspflicht.

2.1 Im ersten Termin zu lösen

Auf der ISIS-Seite zur Veranstaltung finden Sie zusätzliche Literatur und Hilfen für die Bearbeitung der Aufgaben! Zusätzlich können wir den kostenlosen "Beej's Guide to Network Programming Using Internet Sockets"[1] empfehlen. Es gibt für die Aufgaben keinen Vorgabe-Quelltext, aber es finden sich im verlinkten Guide und auch in zahlreichen weiteren Quellen genug Beispiele. Sie sollen bei der Aufgabe auch lernen, sich selbstständig in ein Thema einzuarbeiten.

Aufgabe 4:
Zuerst sollen Sie einen Client für das im RFC 865 beschriebene "Quote of the Day Protocol"[2] programmieren. Zunächst soll ihr Client einen zuverlässigen Bytestrom benutzen, um die Daten abzurufen. In modernen Netzen, z.B. im Internet, benutzt man als Datagram-Service UDP (User Datagram Protocol) und als zuverlässigen Byte-Strom TCP (Transmission Control Protocol). Auch wenn im RFC angegeben ist, dass der Server nicht mit mehr als 512 Zeichen antworten sollte, muss Ihr Client auch mit größeren Antworten umgehen können.

Ihr Client soll (wie `netcat`) vom Benutzer über die Kommandozeile die IP Adresse bzw. den DNS-Namen als ersten Parameter und die Port-Nummer des Servers als zweiten Parameter übergeben bekommen. Der Client soll auf der Konsole die Antwort des Servers ausgeben. Achten Sie besonders auf Spezialfälle wie Zeilenumbrüche! **Fügen Sie selbst keine weiteren Zeichen (z.B. Zeilenumbrüche) hinzu!** Sie können ihren fertigen Client mit einem der folgenden öffentlichen Server testen:

DNS-Name	Port
alpha.mike-r.com	17
djxmmx.net	17

Aufgabe 5:
Im zweiten Teil dieser Aufgabe sollen Sie einen **einfachen** Stream-Socket Server implementieren, der das im RFC 865 beschriebene "Quote of the Day Protocol" implementiert. Es genügt, wenn der Server mit einem Client gleichzeitig kommunizieren kann. Der Server soll über die Kommandozeile den Port und den Dateinamen einer Textdatei übergeben bekommen und jedem Client zufällig ein Zeile aus dieser Datei zurückgeben. Sie können sich bei Ihrer Textdatei im Internet inspirieren lassen[3].

Reichen Sie bitte ihre Lösungen bis Sonntag vor dem zweiten Termin 23:55 Uhr per ISIS ein. Die Lösungen werden **automatisch überprüft** und müssen ohne Ausnahme in einer Datei mit dem Namen **Block2a.TXXGYY.tar.gz** (XX = Praktikumstermin, YY = Gruppennummer) eingereicht werden. Darin enthalten erwarten wir in einem Ordner mit dem Namen Block2a.TXXGYY:

- Makefile
- client.c
- server.c

[1] `http://beej.us/guide/bgnet/`
[2] `https://tools.ietf.org/html/rfc865`
[3] `https://github.com/HashanP/qotd-server/blob/master/qotd.txt`

Wir werden ihre Abgabe mit dem Makefile kompilieren und dann mit den zwei Parametern aufrufen und die Ausgabe überprüfen. Das Referenzsystem ist das Ubuntu 18.04 auf den Poolrechnern. Beispielaufrufe sind:

```
./client djxmmx.net 17
./server 1717 qotd.txt
```

2.2 Im zweiten Termin zu lösen

Aufgabe 6:
Mit einigen wenigen Änderungen sollen Sie Ihren Client so anpassen, dass er Datagram-Sockets benutzt. Benutzen Sie die Funktion clock_gettime (`man clock_gettime`) um innerhalb Ihrer beiden Clients (Stream- und Datagram-Socket Client) Zeitstempel zu setzen, um zu messen, wie lange der Datenaustausch mit dem Server **insgesamt** (inkl. Verbindungsaufbau) dauert. Geben Sie die Zeitdifferenz zwischen den Stempeln jeweils auf der Konsole aus. Sie sollen nun zum Testen `netcat` als Quote of The Day Server verwenden (vergleich letztes Blatt); jeweils einmal als TCP und einmal als UDP Server mit dem gleichen Rückgabestring. Wählen Sie für Ihren Server jeweils einen Port über 1024, da auf UNIX-artigen Betriebssystemen das Betrieben von Diensten auf Ports unter 1024 Root-Rechte voraussetzt. Was stellen Sie beim Vergleich zwischen Stream- und Datagram-Socket Client fest? Haben Sie eine Erklärung?

Reichen Sie ihre Lösung auf ISIS ein. Abzugeben sind:

- Die neue Quelltexte Ihrer Clients
- Eine kurze Begründung (max. 3 Sätze) zur Erklärung der Differenz zwischen Stream- und Datagram-Socket Client

Aufgabe 7:
Sie sollen nun Ihren Stream-Socket Client so umschreiben, dass er Daten von HTTP-Servern abrufen kann. Wie Sie beim Beispiel mit netcat gesehen haben, ist HTTP ein einfaches Text-basiertes Protokoll, bei dem der Server über TCP-Port 80 angesprochen wird.

a) Machen Sie eine Kopie Ihres Streaming-Socket Clients und schreiben Sie ihn dahingehend um, dass er eine über die Kommandozeile übergebene URL abruft. Sie können davon ausgehen, dass nach der Zeichenkette `//` der Hostname folgt und ab dem dritten `/` der angeforderte Pfad anfängt. Ein Beispiel für den Abruf der URL `http://httpbin.org/ip` mit HTTP wäre, dem Server mit dem DNS-Namen `httpbin.org` über TCP-Sockets folgende Zeilen zu schicken:

   GET /ip HTTP/1.1\r\n
   HOST: httpbin.org\r\n
   \r\n

b) Geben Sie die Antwort der Server auf der Konsole aus, allerdings ohne die HTTP-Header. Beobachten oder schlagen Sie nach, wie HTTP-Header und Daten getrennt sind. Rufen Sie ein möglichst lustiges Bild über HTTP ab und leiten Sie die Ausgabe in eine Datei um. Die Datei sollte dabei zwischen 10kb und 200kb groß sein. Beispiel:
   `./client http://i.imgur.com/Et2gFH3.jpg > image.jpg`

c) Reichen Sie ihre Lösung auf ISIS ein. Abzugeben sind: Wir erwarten die Datei Block2b.TXXGYY.tar.gz mit dem Inhalt:

   - Makefile
   - client.c

- clientudp.c
- clienthttp.c
- README

Dabei soll das Makefile die Binärdateien `client`, `clientudp` und `clienthttp` erstellen.

3 Vertiefungsaufgaben

Diese Aufgaben sind zu Ihrere eigenen Vertiefung in Hinblick auf die Klausurvorbereitung gedacht:

Aufgabe 8:
Machen Sie sich mit Naming & Addressing innerhalb des Internet-Stacks vertraut und beantworten Sie die folgende Frage:

- Was ist der Unterschied zwischen Name und Adresse?

Aufgabe 9:
Nennen Sie 4 falsche Annahmen an ein Netzwerk, die bei der Entwicklung eines verteilten Systems zu Problemen führen können. Welche Probleme können jeweils in der Realität auftreten?

Aufgabe 10:
Beantworten Sie für jeden Punkt gesondert die Frage: Welche Gründe sprechen dafür und dagegen für diese Anwendung ”Datagram-Service” oder ”zuverlässigen Byte-Strom”“” zu nutzen?

- Sprachübertragung
- Dateiübertragung
- Remote-Login
- Multicast-Kommunikation (an mehrere Gegenstellen gleichzeitig senden)

Aufgabe 11:
Beantworten Sie folgende Fragen rund um den TCP/IP Stack:

- Welche Layer gibt es im TCP/IP Modell?
- Angenommen Sie öffnen einen Browser und senden eine Web-Anfrage (HTTP) über TCP über IP über Ethernet. In welcher Reihenfolge werden die Header gesendet?
- Angenommen Sie haben zwei Browser gleichzeitig geöffnet und rufen vom gleichen Server gleichzeitig über HTTP das selbe Dokument ab. Wie unterscheidet der Server zwischen den beiden Anwendungen?
- Was bringt das ”Hourglass Model” zum Ausdruck?