Technische Universität München
Informatik VIII
Netzarchitekturen und Netzdienste
Prof. Dr.-Ing. Georg Carle

Heiko Niedermayer, Marc Fouquet

Übungen zur Vorlesung Rechnernetze und Verteilte Systeme

Übungsblatt 8, SS 2010

Abgabe: 28. Juni 2010 (vor der Vorlesung bis 14:10 Uhr)

Aufgabe 22 - Programmieraufgabe: TCP-Kommunikation (18 Punkte)

In dieser Aufgabe sollen Sie einen TCP-Client programmieren. Dazu steht Ihnen bereits ein Grundgerüst in Java zur Verfügung. Sie finden es auf der Übungswebsite.

Als Gegenstelle Ihres Programms haben wir einen TCP-Server aufgesetzt. Den Sourcecode des Servers finden Sie ebenfalls auf der Website.

Der Server ist erreichbar unter der Adresse ilab.net.in.tum.de:2342.

Sie können sich per „telnet ilab.net.in.tum.de 2342“ mit dem Rechner verbinden und das Protokoll ausprobieren. Ein Protokollbeispielablauf ist nachfolgend abgedruckt:

```
GrnvsTcpServer: Verbindung 9 von IP /131.159.14.62:56745
--> HELLO
Welcome!
Please answer the following challenge by summing up the numbers:
5428
1145
2363
--> 8936
Please send your Teamletter...
--> B
Please send your Name formatted as [Surname] [Prename]...
--> Pahl Marc-Oliver
Submitted data:
B,Pahl Marc-Oliver,/131.159.14.62:56745,Fri Jun 20 00:19:17 GMT+01:00 2008
Your data was saved.
Thank you!
Bye...
```

Die Eingaben des Client sind jeweils durch einen Pfeil gekennzeichnet.

Ziel der Aufgabe ist es, dass Ihr Client Ihre Gruppe und Ihren Namen an unseren Server überträgt. Da dies prinzipiell auch per Telnet möglich ist, haben wir eine kleine Challenge eingebaut, die darin besteht, die drei zufällig vom Server gewählten Zahlen zu addieren und das Ergebnis innerhalb von maximal 5 Sekunden zurückzusenden. Ihr Programm sollte dies in weniger als einer Sekunde durchführen können. Als menschliche Gegenstelle werden Sie es allerdings nicht schaffen, die Antwort in weniger als 5 Sekunden zur Verfügung zu stellen.

War Ihre Eingabe zu langsam, so weist Sie der Server darauf hin:

```
Your data was *NOT SAVED* since you answered too slowly...
(You took 38 seconds, maximum allowed time is 5 seconds.)
Pease try again!
```

Der Server ist so programmiert, dass er bei falschen Eingaben die Verbindung beendet. Sie können dieses Verhalten vorab in einer Telnet-Sitzung ausprobieren.

Die Liste der bisher erfolgreichen Übermittlungen finden Sie auf der passwortgeschützten Website „https://ilab.net.in.tum.de/extern/grnvs“. Sie können sich dort mit folgenden Daten einloggen:

```
username: grnvs10
passwort: socket
```

a) Programmieren Sie basierend auf dem Grundgerüst einen Client in Java, der es Ihnen ermöglicht, die gewünschten Daten innerhalb der 5 Sekunden an unseren Server zu übertragen.

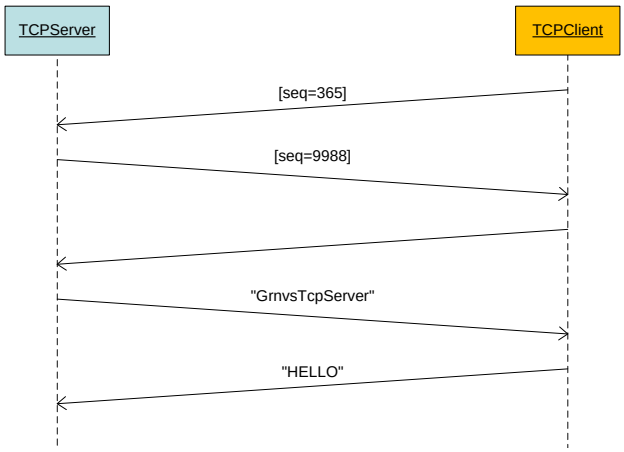
b) Übermitteln Sie Ihre Daten (Gruppennummer und Namen) in dem vom Server gewünschten Format (siehe Servertexte in obigem Kommunikationsprotokoll). Bitte übermitteln Sie für jeden Teamteilnehmer einzeln die Daten.
Auf der oben angegebenen Website können Sie überprüfen, ob Ihre Abgabe erfolgreich war.

c) Drucken Sie Ihren Client aus und geben Sie ihn mit Ihren Lösungen ab.

Wir werden nun die tatsächlich auf der Leitung gesendeten Pakete beobachten. Üblicherweise verwendet man dazu das Open Source Tool “Wireshark”, auf http://www.wireshark.org/ gibt es Installationsdateien für Windows und Mac. Sollten Sie Linux verwenden, so ist Wireshark vermutlich in den Paket Repositories Ihrer Distribution enthalten (z.B. Debian/Ubuntu installiert man es per `sudo apt-get install wireshark`).

Installieren Sie Wireshark und spielen Sie zunächst etwas damit rum, um ein Gefuhl für das Programm zu bekommen (Anregungen: Beobachten Sie den Traffic verschiedener Anwendungen, z.B. Web-Browser, Mailclient, “ping”,. . .).

In folgendem Sequenzdiagramm sehen Sie beispielhaft den Beginn einer TCP-Sitzung zwischen Ihrem Client und dem Server:



d) Zeichnen Sie ein Sequenzdiagramm änlich dem obigen für alle TCP-Pakete bis zum “HELLO” inklusive der Flags, Sequenznummern (SEQ) sowie bestätigten Sequenznummern (ACK). Schalten Sie dazu in Wireshark unter “Edit ⇒ Preferences ⇒ Protocols ⇒ TCP” die Anzeige relativer Sequenznummern ab.

e) Die MAC-Adressen welcher Geräte kommen als Quell- und Zieladresse im Ethernet-Header von

Paket 1 vor, wenn Sie das Paket auf ihrem eigenen Rechner (also auf Clientseite) beobachten? Schlagen Sie die Hersteller der Geräte auf http://www.coffer.com/mac find/ nach.

f) Wie ist die TTL der Pakete vom Server? Wie hoch war der TTL-Wert vermutlich, als die Pakete gesendet wurden?

g) Rechnen Sie nach, ob die Sequenznummern mit der gesendeten Datenmenge übereinstimmen. Siehe Bild oben.

h) Welche TCP-Flags sind im "GrnvsTcpServer"-Paket gesetzt? Was ist die Bedeutung des PSH-Flags? Hatte dieses Flag Einfluß auf Ihr Programm?

Aufgabe 23 - IPv6 (6 Punkte)

In dieser Aufgabe geht es um das Protokoll IPv6, dem Nachfolger des heute verbreiteten IPv4.

Next Header (Dezimalwert)	Protokoll
4	IPv4
6	TCP
17	UDP
41	IPv6
43	Routing Extension Header
44	Fragmentation Extension Header
58	ICMPv6
59	No Next Header

Tabelle 1: Werte für das Feld Next Header in IPv6

Feld	Größe in Bits	Bedeutung
Next Header	8	wie bei IPv6-Header
Reserved	8	nicht benutzt, wird auf 0 gesetzt
Fragment Offset	13	wie bei IPv4
Reserved	2	nicht benutzt, wird auf 0 gesetzt
More Fragments Flag	1	wie bei IPv4
Identification	32	wie bei IPv4, hier aber 4 Bytes lang

Tabelle 2: IPv6 Fragment Extension Header

a) Wie groß ist der Adressraum bei IPv6? Was sind Link-Local-Adressen?

b) Geben Sie den IPv6-Header für ein Paket von FE80::0207:0006:3152 zu FE80::0217:11A6:0052 mit 1000 Byte TCP als Payload an. Ignorieren Sie dabei die Felder Traffic Class und Flow-Label. Setzen Sie Hop Limit auf 64. Es reicht bei entsprechenden Felder Dezimalwerte anzugeben. Welche Bedeutung hat das Feld Hop Limit?

c) Zur Fragmentierung verwendet IPv6 einen Erweiterungs-Header (siehe Tabelle). Bei IPv6 darf dabei jeweils nur der Sender eines IP-Paketes dieses fragmentieren und daher entfallen einzelne Flags wie DF, die es bei IPv4 gibt. Schreiben Sie nun den IPv6-Header und den entsprechenden Erweiterungsheader für das **erste** der Fragmente hin, wenn das Fragment 500 Bytes (Payload) lang ist. Identification können Sie auf einen beliebigen Wert setzen (zufällig).

d) Woran erkennt IPv6, dass Pakete Multicast-Pakete sind?