11 Client-Server-Interaktion mit Sockets

1 Thema des Praktikums

Im folgenden Praktikum werden die Server- und Client-Programme aus der Theorie in Aktion betrachtet. Insbesondere wird die Verwendung der Ports und der Port-Nummern genauer untersucht. Weiter soll gezeigt werden, dass Dienste unabhängig von Betriebssystem und Client-Programmen verwendet werden können.

Im zweiten Teil sollen die Server- und Client-Programme erweitert werden, so dass sie mit einem normalen HTTP-Client oder HTTP-Server kommunizieren können. Die Wirkungsweise textbasierter Applikationsprotokolle und im speziellen von HTTP (Hyper Text Transport Protocol) kann so einfach untersucht werden.

Die Schwerpunkte des Praktikums sind:

- Verwendung von Port-Nummern
- Interaktion verschiedener Client-/ Server-Programme
- Erstellen eines einfachen HTTP-Clients / HTTP-Servers

2 Vorbereitung

- Studieren Sie das Kapitel Socket-Programmierung und die zu diesem Praktikum benötigten Programme server.c und client.c.
  Sie finden diese auf OLAT / KT / Themen und Unterlagen / Socket-Schnittstelle / unter Praktikum oder im ZHAW-Netz: http://160.85.20.60/ines_labor/002_kt/download-kt-content/11_socket/

Im Gegensatz zum vorliegenden Client sendet ein HTTP-Client immer zuerst einen Request, bevor der HTTP-Server die Antwort sendet. Ein solcher HTTP-Request-Header kann ziemlich kompliziert aufgebaut sein. Im Folgenden verwenden wir eine Minimalvariante.

Erweitern Sie den Client (client.c) so, dass dieser als erstes einen HTTP-Request sendet. Dazu fügen Sie dem Client nach dem Verbindungsaufbau eine Befehlsfolge ein, die den folgenden Request sendet:

```
"GET / http/1.0\nAccept: */*\nUser-Agent: client_www\n\n"
```

- Speichern Sie das Programm für das Praktikum zugänglich als client_www.c.
- Erweitern Sie den Server (server.c) für die Entgegennahme und Anzeige eines HTTP-Client-Request. Dazu fügt man im Server eine Befehlsfolge ein, die nach dem Verbindungsaufbau zunächst alle Daten vom Client liest und ausgibt.
  Hinweis: Einen entsprechenden Codeblock finden Sie im Programm client.c
- Speichern Sie das Programm für das Praktikum zugänglich als server_www.c.
- Beantworten Sie folgende Fragen:

Was müssen Sie ändern, damit in den obigen Programmen 8080 als Default-Port verwendet wird?

DefaultPortNumber auf 8080 anpassen

- Führen diese Änderungen durch.

Der Server zeigt die IP-Adresse und das Port des Clients an. Woher kennt er diese?

siehe nächste Seite

- Zeigen Sie die Vorbereitungen dem Laborbetreuer.

```
/* Gewünschte Eigenschaften des Sockets angeben                    */
ClearMemory(hints);
hints.ai_family = AF_INET;       // nur IPv4
hints.ai_socktype = SOCK_STREAM; // TCP Stream Sockets
hints.ai_flags = AI_PASSIVE;     // eigene IP-Adresse verwenden
Status = getaddrinfo(NULL, PortNumber, &hints, &servinfo);
ExitOnError(Status, "getaddrinfo fehlgeschlagen");
```

3 Versuchdurchführung

- Verbinden Sie die Rechner werden über eth1 mit einen Hub. Starten Sie Rechner A und B mit Linux und C mit Windows (Abbildung 1). Das Ethernet-Interface eth0 bleibt mit dem ZHAW-Netz verbunden.

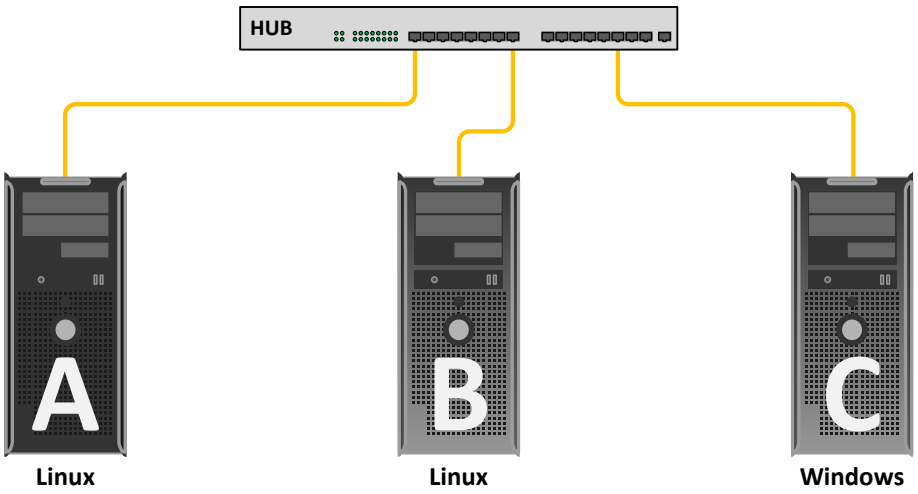

Abbildung 1

- Kopieren Sie alle Client- und Server-Programme auf die Festplatte. Öffnen Sie dazu eine Konsole und geben Sie **`download-kt`** ein. Das Script kopiert die benötigten Files in das Verzeichnis /home/ktlabor/praktika.
- Am Ende des Praktikums führen Sie das Scrip **`reset-kt-home`** aus. Das Script löscht das Home-Verzeichnis des Users ktlabor und erstellt es neu.
- Um die Client-/Server-Synchronisation aufzuzeigen, verwenden wir die Programme server.c und client.c aus dem Unterverzeichnis `/home/ktlabor/praktika/11_socket`. Übersetzen Sie diese vorbereiteten Server- und Client-Programme auf beiden Linux-Rechnern.

```
gcc -Wall -o server server.c
gcc -Wall -o client client.c
```

Konfiguration von Diensten

Der Daytime-Service wartet auf einen Client (Connect) und liefert eine Zeile mit dem aktuellen Datum und der Uhrzeit zurück.

- Die Datei /etc/services enthält eine Auflistung der bekannten Dienste und deren Portnummern.

Welches Well Known Port ist für den Daytime-Service reserviert?

- Mit dem Befehl ss erhalten Sie unter Linux Informationen über die IPv4-Dienste, die auf dem Rechner momentan gestartet sind.

```
S -a -4
```

- Unter Windows starten Sie dazu **services.msc** (im Run/Search-Fenster eingeben). Suchen Sie den Dienst «Einfache TCP/IP-Dienste»).

Ist der Daytime-Service unter Linux/Windows gestartet?

- Der Daytime-Service ist heute oft abgeschaltet, da er angeblich für Denial-of-Service-Attacken benutzt werden kann. Die Konfiguration unter Linux erfolgt im File **/etc/inetd.conf** und unter Windows mit **services.msc**. Stimmen die Angaben mit den obigen Beobachtungen überein?

Im Folgenden versuchen Sie von verschiedenen Clients auf Daytime-Dienste verschiedener Server zuzugreifen. Wir haben einen zusätzlichen Server aufgesetzt, der den Daytime-Service garantiert anbietet sollte: **srv-lab-t-667.zhaw.ch** (160.85.20.60).

- Testen Sie nun den Zugriff mit Hilfe der Programme **telnet** und **client** (von Linux) sowie **telnet** (von Windows) aus. Verfolgen Sie die Kommunikation mit Wireshark und erklären Sie, warum es geht resp. nicht geht.

  `telnet` *`server port`* oder `./client` *`server port`*

Funktionieren die Zugriffe auf Dienste des eigenen Rechners (Linux / Windows)?

---

*Funktionieren die Zugriffe den Server **srv-lab-t-667.zhaw.ch***

Funktionieren die Zugriffe auf andere Rechner (A-C, C-A etc.)

- Schalten Sie unter Windows die Firewall aus und testen Sie den Zugriff erneut mit Port 17.

  `./client` *`windowsserver`* `17`

Was macht der Dienst?

Verwendung von Ports

- Zunächst soll die Grundfunktion überprüft werden. Dazu starten Sie auf dem einen Rechner 160.85.x.x das Server-Programm, das am Port 4711 auf eine Verbindung wartet.

  `./server 4711`

  Erstellen Sie von einem anderen Rechner 160.85.y.y aus eine telnet-Verbindung oder nutzen Sie dazu das client-Programm:

  `telnet 160.85.x.x 4711` oder `client 160.85.x.x 4711`

  Betrachten Sie den generierten Datenverkehr mit Wireshark und bestimmen Sie die verwendeten Port-Nummern.

Welche Ports werden vom Client und vom Server verwendet?

Unterscheidet sich die Kommunikation zwischen dem telnet- und dem client-Programm resp. zwischen Linux und Windows?

- Führen Sie die folgenden Schritte aus, notieren Sie die Beobachtungen und versuchen Sie anschliessend dieses Verhalten dem Laborleiter zu erklären.
- Erstellen Sie eine Verbindung, stoppen Sie anschliessend den Server mit der Tastenkombination *CTRL-C* und versuchen Sie sofort ihn neu zu starten.

- Betrachten Sie den Zustand des Ports mit

```
Ss -a -4 -n | grep 4711
```

  Versuchen Sie den Server mit einem anderen Port zu starten (z.B. ./server 4712)

- Warten Sie ca. 2 Minuten und versuchen Sie es erneut mit dem ersten Port (4711).

Erklärung?

- Vergleichen Sie die angezeigten Ports des Servers, mit den Angaben im Wireshark.

Woher stammt die Abweichungen?

- Korrigieren Sie den Code.

- Zeigen Sie die Resultate dem Laborbetreuer!

4 Client-/ Server-Kommunikation nach dem Request/Response-Schema

Erweiterung des Clients

- Laden und übersetzen Sie Ihre vorbereiteten Programme www_client.c und www_server.c.
- Testen Sie den Client mit einem HTTP-Server, wobei als Port der HTTP-Port 80 angegeben wird. Auf allen Linux-Systemen im KT-Labor ist ein Apache-Server installiert. Starten Sie diesen:

```
sudo apache2ctl start
```

  Testen Sie den Apache-Server mit einem normalen Browser.

- Nutzen Sie Ihr HTTP-Client-Programm, um die Antwort des Servers anzuzeigen:

```
./client_www 160.85.xx.xx 80
```

- Studieren Sie den HTTP-Response-Header, der etwa wie folgt aussehen sollte:

```
HTTP/1.1 200 OK
Date: Wed, 24 Aug 2099 13:34:03 GMT
Server: Apache/2.2.16
Last-Modified: Tue, 16 Aug 2099 12:14:18 GMT
…
Content-Type: text/html
```

Von diesem Header sind vor allem die erste und die letzte Zeile interessant. Der Rest bezieht sich auf Informationen zum angeforderten Dokument und zum Server.

- Die erste Zeile besagt, dass der Server HTTP Version 1.1 versteht und dass der Request erfolgreich war ("200 OK").
- Die letzte Zeile gibt den MIME-Typ der nun folgenden Informationen an. In diesem Header ist es ein HTML-Dokument.
- Eine Leerzeile schliesst den Header ab.

Was ändert sich am Response-Header, wenn Sie im Get-Request eine unbekannte URL anfordern.

Erweiterung des Servers

- Starten Sie das übersetzte HTTP-Server-Programm.

```
./server_www 8080
```

- Testen Sie den HTTP-Server mit Ihrem HTTP-Client-Programm.
- Testen Sie den HTTP-Server, indem Sie mit einem normalen Browser (z.B. Iceweasel) einen Request erzeugen. Dazu wird die URL (Universal Resource Locater), die IP-Adresse des Servers und der Port 8080 angegeben.

```
http://160.85.xx.xx:8080/index.html
```

- Studieren Sie den Request-Header des Browsers, der etwa wie folgt aussehen sollte:

```
GET /index.html HTTP/1.1
Host: 160.85.36.117:4711
User-Agent: Mozilla/5.0
Accept: text/html,application/xhtml+xml,application/xml,*/*
Accept-Language: de-de,en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive: 300
Connection: keep-alive
Cache-Control: max-age=0
```

  - Die erste Zeile besagt, dass der Client das Dokument /index.html anfordert.
  - Der Rest ist Information über den Browser. Die Accept-Zeile gibt z.B. an, welche MIME-Typen dieser darstellen kann. Der Header wird durch eine Leerzeile abgeschlossen.

- Geben Sie in der URL statt der IP-Adresse den Rechnernamen an und vergleichen Sie die Ausgabe.

Worauf bezieht sich die Information der 2. Zeile? (Meint Host: den Client oder den Server?)

Wozu könnte man diese Information benützen?

- Zeigen Sie die Resultate dem Laborbetreuer!

5 Zusatzaufgabe

Modifizieren Sie den Server, so dass er abhängig vom URL unterschiedliche Antworten liefert.