19. März 2014

Steffen Giersch & Maria Lüdemann

HAW Hamburg

Rechnernetze Praktikum 1

Werkzeuge zur Arbeit mit Netzwerken & Netzwerkprogrammiereung

Inhaltsverzeichnis

[2 Überschrift 2](#_Toc383001563)

Teil 1

# Werkzeuge zur Arbeit mit Netzwerken

1. Welche Informationen erhält man mit dem Programm /sbin/ifconfig

Sbin/config = Liefert Informationen zu den Netzwerkgeräten z.B iP Adresse, Netzwerkmaske, broadcast, ethernetmaske, Packets Informationen, Device Informationen

1. Messen Sie die Round Trip Time für ein 1000-Datenbyte-Paket nach www.google.de, nach

www.haw-hamburg.de und zum Rechner lab22 jeweils als Durchschnittswerte von 10

Stichproben.

Round Trip Time: (RTT) ping –c 10 –s 1000 [www.google.de](http://www.google.de)

173.194.113.183 10 packets transmitted, 0 recived, 100% packet loss, time 9010ms

(im HAW Netz machte das anpingen von Google Probleme)

1. Finden Sie heraus, welches Programm auf Ihrem Rechner den Port 9400 benutzt. Welches Protokoll setzt dieses Programm auf der Transportschicht ein? In welchem Systemaufruf "hängt" das Programm? (Für letzteres brauchen Sie noch das Programm strace mit der Option -p <pid>)

Ping lab22 geht wenn man sich physikalisch im Raum befindet (Vermutlich 7. Stock)

1. Stellen Sie eine TCP-Socketverbindung zwischen der Shell (Standardein-/ausgabe) und dem Server www.bsh.de auf dessen Port 80 her. Senden Sie die Zeichenkette „GET“ und Sie die Antwort des Servers. Um welches Protokoll handelt es sich hier?
2. Loggen Sie sich von Ihrem Arbeitsplatzrechner aus auf dem zugehörigen Gegenstellenrechner (siehe Tabelle) ein und lassen Sie sich dort mit dem Dateimanager (Befehl konqueror von der Kommandozeile) den Inhalt des dortigen /tmp-Verzeichnisses zeigen.

Teil 2

# Netzwerkprogrammierung

Programmierung einer Server ~ Client Anwendung

Beschreibung des Vorgehens:

Die Aufgabe wurde in drei Klassen gelöst, Client, Server und ServerThread die als Server und Client agieren. Es gibt den Clienten der versucht eine Verbindung zu einem Server aufzubauen dessen Verbindungsdaten direkt vom Nutzer abgefragt werden. Der Server nutzt eine WelcomeSocket um verbindungs Versuche anzunehmen, dann überprüft er ob er noch weitere Clienten annehmen möchte, ist dies der Fall öffnet er einen neuen Thread gibt ihm einen neuen Socket auf dem er fortan mit dem Clienten kommuniziert.

Funktionalität Erfolgsfall:

Der Client hat einige Möglichkeiten Befehle an den Server zu senden.

* UPPERCASE <message>
* LOWERCASE <message>
* REVERSE <message>
* BYE
* SHUTDOWN <password>

Für alle Befehl <message> Konstrukte gilt, sie dürfen nicht länger als 255 Bytes sein.

**Uppercase** nimmt den <message> String und wandelt alle kleinen Buchstaben in große um

Bsp. Hallo -> HALLO

**Lowercase** nimmt den <message> String und wandelt alle großen Buchstaben in kleine um

Bsp. Hallo -> hallo

**Reverse:**  nimmt den <message> String und dreht ihn um

Bsp. Hallo -> ollah

**Bye:** wird vom Server erwiedert, danach beendet sich der Client

**Shutdown:** nimmt einen String den er als Passwort übergibt. Der Server überprüft ob das Passwort stimmt, hört wenn dies der Fall ist auf neue Verbindungen anzunehmen und wartet darauf dass sich alle Threads beenden. Dann beendet sich auch der Server.

Funktionalität Misserfolg/Fehler:

Wenn Eingaben länger sind als 255 Bytes wird ein Fehler geworfen und die gesamte Eingabe verworfen und auf eine neue Eingabe gewartet.

ERROR Message to long

Dabei wird der gesamte Buffer vorgespult, sodass alle nachfolgenden Befehle nicht mehr bearbeitet werden.

Bei unbekannten Befehlen oder fehlender <message> wird ein

Unknown command

Zurück gegeben und auf eine neue Eingabe gewartet.

Ein falsches Passwort erzeugt die Rückgabe:

ERROR Wrong Password

Danach ist der Server wieder bereit Befehle entgegen zu nehmen

Client Konsole einfach schließen

Connection lost

Keine Verbindung bekommen

Thread Sicherheit:

Damit der Server beim Shutdown auf alle Threads warten kann nutzen wir eine Variable die mitzählt wie viele Threads ein Server geöffnet hat. Beendet sich ein Client ruft er eine Funktion auf die diese Variable dekrementiert. Dies ist bei Multithreaded Anwendungen immer eine Gefahr. Somit haben wir die Variable auf voltaire gesetzt um sicherzugehen, dass sich die Threads nicht gegenseitig in die Quere kommen können.