Praktikum Netzwerke

Socketprogrammierung mit Visual Studio

1. Programmieren eines Client zur Dateianforderung von einem Web-Server

Die Aufgabe ist es, von einem Web-Server mit der IP Adresse 192.168.10.113 eine Datei anzufordern. Der Dateiname lautet „hallo.txt“.

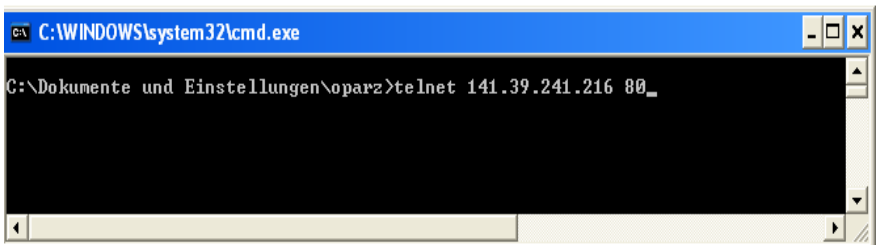
Der http- Befehl zur Anforderung einer Datei lautet:

GET	/hallo.txt	HTTP/1.1	\r\n	Host:	PC	\r\n\r\n
1	2	2	4	5	6	7

Die einzelnen Bestandteile haben dabei die folgende Bedeutung:
1 = GET Kommando (= eine Datei anfordern)
2 = Name der gewünschten Datei: hallo.txt
3 = Version des http-Protokolls: 1.1
4 = neue Zeile
5 = Parameter-Kennung für den Host-Namen (wird immer benötigt!)
6 = Name des Host (frei wählbar): PC
7 = neue Zeile und Leerzeile (darf nicht fehlen!)

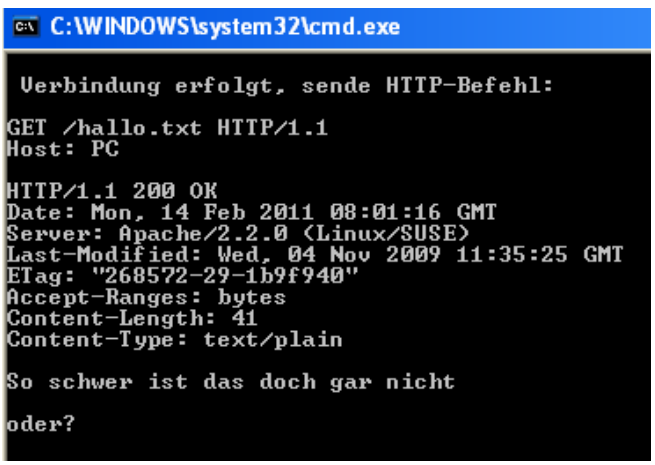
1.1. Telnet

Einfach testen können Sie diese Anforderung mittels des Programms telnet. Dazu geben Sie auf der Kommandozeile folgendes ein:



Frage: Was bedeutet der erste und der zweite Parameter dieser telnet Abfrage?

Nach drücken der Return-Taste müssten Sie eine ähnliche der abgebildeten Ausgabe erhalten:



1.2. Socket Client mit C++

Diese Aufgabenstellung sollen Sie nun als C++ Programm schreiben. Ein Muster mit einigen Fehlern ist hier abgedruckt:

```
// aufgabe2.cpp : Datei von einem Webserver abholen.
//

#include "stdafx.h"

#include "socket.h"
#include <iostream>
#include <cstring>

int main(int argc, char * argv[])
{
	bool ergebnis;
	struct sockaddr_in adresse;

	string empf_zeichen;

	Socket sock;

	unsigned short int portnummer = 55;
	char ip_adresse[] = "141.39.241.215";
	char befehl[] = "GET /hallo.txt HTTP/1.1\r\nHost: PC\r\n\r\n";

	sock.create();

	ergebnis = sock.connect(ip_adresse, portnummer);

	/* fuegen Sie eine Ausgabe ein mit der IP Nummer und der Port-Adresse */

	if (ergebnis == true)
	{
		perror(" Keine Verbindung erfolgt: ");
	}
	else
	{
		cout << "Verbindung erfolgt" << endl;
		cout << ", sende HTTP-Befehl:\n\n“ << befehl << endl;

		sock.send(befehl);

		sock.recv(empf_zeichen);

	/*
	fuegen Sie eine Ausgabe mit dem String empf_zeichen
	und der Anzahl der Zeichen an
	*/

	}
	sock.close();
	cout << "\n Programm beendet\n\n";
}
```

Vervollständigen Sie das Programm und schauen Sie sich die Antwort des Servers an. Wenn alles richtig funktioniert, müsste Ihnen die Antwort aus den Wireshark Übungen bekannt vorkommen.