TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze und Management verteilter Systeme
Prof. Anja Feldmann, Ph.D.
Gregor Maier, Jörg Wallerich, Andi Wundsam

2. Blatt Praktikum Protokolldesign WS 07/08

Aufgabe 1: (60 Punkte) *Einfacher Client und einfacher Server*

Auf diesem Aufgabenblatt beginnen wir mit Netzwerkprogrammierung mit Hilfe der Socket-Schnittstelle. Deshalb ist es für dieses eine Aufgabenblatt *nicht* erlaubt, abstrahiernede Socket-Klassen zu verwenden. Es sind lediglich die Funktionen erlaubt, die den direkten Systemaufrufen zur Socket-Schnittstelle erlauben[1]. Aus diesem Grund ist auch die Verwendung von Java, der Perl-Klasse `IO::Socket` und den entsprechenden Abstraktionen anderer Scriptsprachen für dieses eine Aufgabenblatt nicht gestattet!

Das Ziel dieses Aufgabenblattes besteht in der Programmierung eines Dateitransferklienten und eines Dateitransferservers. Klient und der Server sollten ein gemeinsames Programm sein, das anhand von Kommandozeilenparametern unterscheidet, was es tun soll.

(a) Der Klient soll folgende Eigenschaften haben:

- Man soll ihn mittels folgender Kommandozeile bedienen können:
  `transfer <server> <port> <datei>`.
- Er soll sich dann mit dem genannten Rechner und Port verbinden und mittels
  `get <datei>\r\n`
  die Datei anfordern.
- Der Server wird
  `200 <Anzahl> Bytes\r\n`
  antworten, und danach die Datei schicken, sofern sie da ist, oder
  `550 No such file or directory\r\n`
  sofern nicht.
- Der Klient soll die Datei auf die Standardausgabe ausgeben, sofern kein Fehler auftritt, oder eine passende Fehlermeldung auf die Fehlerausgabe schreiben. Anschließend soll er sich mit einem passenden Exitcode beenden.

(b) Der Server soll folgende Eigenschaften haben:

- Man soll ihn mittels folgender Kommandozeile bedienen können:
  `transfer -listen <port>`.
- Er soll dann auf dem genannten Port auf Verbindungen warten.
- Wenn ein Klient eine Anfrage, wie oben beschrieben schickt, soll er sie entsprechend beantworten, sofern eine entsprechende Datei im aktuellen Verzeichnis liegt.
- Wenn der Dateiname ein '/' enthält, soll er
  `551 Current directory only\r\n`
  ausgeben.
- Wenn er keine passende Datei findet, soll er
  `550 No such file or directory\r\n`
  ausgeben.

Abzugegen ist:

- der Quellcode Deines Programmes

[1] socket(), bind(), listen(), connect(), close(), sysread(), syswrite(), send(), recv(), print(), <HANDLE>, etc.

Aufgabe 2: (40 Punkte) *Einfacher Proxy für das Dateitransferprogramm*

Jetzt soll das Dateitransferprogramm so erweitert werden, dass es auch als Proxy fungieren kann, mit dem sich ein Client verbinden kann, um den Dateitransfer über ihn abzuwickeln.

Zu diesem Zweck führen wir folgende Modifikationen ein:

- Erweitere die Funktionalität des Servers, so dass er auch als Proxy arbeiten kann.
- Der Klient kann mit `transfer <proxy> <proxy-port> <server> <port> <datei>` gestartet werden, und verhält sich wie ein Client aus Aufgabe 1, ausser dass er den Proxy kontaktieren soll statt des Servers.
- Wenn ein Proxy ein Kommando der Form
  `get <hostname> <port> <weiterer String>\r\n`
  erhält, öffnet er eine Klientenverbindung zu `<hostname>` und gibt ein
  `get <weiterer String>\r\n`
  an `<hostname>` weiter. Die eintreffenden Daten oder Fehlermeldungen werden dann vom Proxy durchgereicht.
- Wenn der Proxy keinen Server des angegebenen Namens kennt, antwortet er mit
  `553 <hostname1>: <hostname> not known\r\n`
  wobei `<hostname1>` sein eigener Hostname ist.

Abzugegen ist:

- der Quellcode Deines Programmes

Details zur Abgabe der Aufgaben:
siehe FAQ (`http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`)
Abgabe: bis Dienstag, den 30. Oktober 2007, 11:59 h s. t.