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

2nd Work Sheet Praktikum Protokolldesign WS 07/08

Question 1: (60 points) *Simple Client and Server*

In this exercise we start with network programming using the sockets interface. Therefore, for this one excercise, the use of high-level socket classes and methods is not allowed. Only system-call-like low level methods are allowed[1]. This means that Java, the Perl class `IO::Socket` and the corresponding abstractions in other script languages are not allowed for this one exercise!

The goal for this exercise is to write a file transfer client and server. Both client and server should be a single program (executable, script) that acts both as client or server depending on the command line parameters.

(a) The client should have the following properties:

- One should be able to use it with the following command line:
  `transfer <server> <port> <file>`
- It should connect to the given host on the given port and send the following request to the server:
  `get <file>\r\n`
- The server should answer with
  `200 <count> Bytes\r\n`
  and then send the file if it exists. If the file doesn't exist, the server should answer with
  `550 No such file or directory\r\n`
  instead.
- The client should output the file to standard out if no error occured, otherwise the client should print the error message to standard error output. Then the client should exit with an appropriate exit code.

(b) The server should have the following properties:

- One should be able to use it with the following command line:
  `transfer -listen <port>`
- It should listen on the given port and wait for connections.
- If a client sends a request as specified above, the server should answer accordingly if the file is in the *current directory*.
- If the filename contains a `/`, the server should answer with
  `551 Current directory only\r\n`
- If the server doesn't find the file, it should answer with
  `550 No such file or directory\r\n`

Deliverables:

- source code of your program

Question 2: (40 points) *Simple proxy for the file transfer program*

Now we want to enhance the file transfer program so that it can also act as a proxy. A client can use this proxy to handle file transfers.

In order to do this you should add the following modifications:

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

- Enhance the server functionality so that a server also acts as a proxy.
- The client can be started with `transfer <proxy> <proxy-port> <server> <port> <file>` and it should behave like the client from Question 1, except that it connects to the proxy instead of the server.
- If a proxy receives a command in the form of
  `get <hostname> <port> <other string>\r\n`
  it opens a client connection to `hostname` on port `port` and sends
  `get <other string>\r\n`
  to `hostname`. The proxy forwards the arriving data to the original client.
- If the proxy doesn't know (cannot resolve) a server with the given name, it responds with
  `553 <hostname1>: <hostname> not known\r\n`
  where `<hostname1>` is the hostname of the proxy machine.

Deliverables:

- sourcecode of your prorgram

Submission details:
see FAQ (`http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`)
Due: Tuesday, October 30, 2007, 11:59 h s. t.