TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Intelligente Netze
Prof. Anja Feldmann
Thomas Krenc, Felix Poloczek, Srivatsan Ravi and Philipp Schmidt

13th Assignment: Network Protocols and Architectures, WS 14/15

As this worksheet contains a programming task, please provide a `.zip` archive containing a `.pdf` document with the answers as well as your program.

Question 1: $(15 + 5 + 60 + 10 + 10 = 100$ points) *State: The Cubby Hole Protokoll*

A cubby hole is a small hiding place where one can hide things. We now define the cubby hole protocol that allows users to store one line messages on a server. As the hole is really small, the server will only store one message at a time, but keeps and shares it across different connections. If a new message is put in the cubby hole, the old message is lost.

We realize the cubby hole protocol as simple, TCP based text protocol. Each command consists of a single word (casing does not matter) that might be followed by a space and an arbitrary text and is terminated with a newline. The following commands should be supported:

PUT $<$ *message* $>$ Places a new message in the cubby hole.

GET Takes the message out of the cubby hole and displays it.

LOOK Displays message without taking it out of the cubby hole.

DROP Takes the message out of the cubby hole without displaying it.

HELP Displays some help message.

QUIT Terminates the connection.

The server greets any new client with `!hello: <text>`. After that, the server answers each command with `!<command>: ok` or `!<command>: <text>`. e.g. the command `pUt hello world` is answered with `!PUT: ok` and `get` is answered with `!GET: hello world`.

To get a feeling for the protocol, you can use `telnet` or `netcat` to connect to our demo server on *teach-and-test.inet.tu-berlin.de*, port *9876* and play around.

(a) Draw a state diagram of the server. You may assume that only one client is connected at a time.

(b) Is the protocol using soft-state or hard-state? Explain.

(c) Implement a simple server for the Cubby Hole protocol. Besides implementing all commands, it has to be able to handle clients that terminate the connection without sending the **QUIT** command.

(d) Make sure that the server can handle multiple clients at a time.

(e) Make sure that the server does not block, and therefore continues serving other client, when a client sends an incomplete line. It should also be able to handle multiple commands in a single TCP segment.

Your submission has to include the full source of the program as well as a script called `run.sh` that builds your server (if necessary) and starts it in a way that it will listen on TCP port 9876. The server can be written in a language of your choice, but has to work without additional libraries on the IRB Linux machines.[1] If you fail these conditions, we won't give you any points in task c to e!

You can use any skeleton for a TCP server you find on the Internet, but please credit the source in a comment.

We recommend to do peer-testing with your fellow students.

[1] See `http://wiki.freitagsrunde.org/SSH#Liste_der_Server_im_CS-Netz` for a list of IRB machines you can use.

Due Date: Wednesday, February, 4th 2015 only until 14:00 h s. t.

- **As PDF files (no MS Office or OpenOffice files):** Uploaded via ISIS (`https://www.isis.tu-berlin.de/2.0/course/view.php?id=2560`)
- Put your name, StudentID number (Matrikelnummer) **and** the name of your tutor on your solution.