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

13. Blatt: Network Protocols and Architectures, WS 14/15

Dieses Arbeitsblatt enthält eine Programmieraufgabe. Als Abgabe wird ein `.zip`-Archiv erwartet, das sowohl eine `.pdf`-Datei mit den Antworten zu den Textaufgaben, als auch Dein Programm enthält.

Aufgabe 1: $(15 + 5 + 60 + 10 + 10 = 100$ Punkte) *Zustand: Das Cubby-Hole-Protocol*

Ein *cubby hole* ist ein kleines Versteck, in dem sich Dinge verbergen können. Das Cubby-Hole-Protokoll ermöglicht es dem Nutzer einzeilige Nachrichten auf einem Server abzulegen. Da unser Versteck wirklich sehr klein und eng ist, passt immer nur eine Nachricht in unser *cubby hole*, welche aber über mehrere Verbindungen hinweg erhalten bleibt. Wird eine neue Nachricht abgelegt, so geht die alte verloren.

Das Cubby-Hole-Protokoll ist als einfaches textbasiertes Protokoll auf Basis von TCP realisiert. Jeder Befehl im Protokoll besteht aus einem Wort (bei dem Groß- und Kleinschreibung ignoriert wird), optional gefolgt von beliebigem Text, und wird durch einen Zeilenumbruch beendet. Folgende Befehle werden unterstützt:

PUT $<$ *message* $>$ Legt eine neue Nachricht $<$ *message* $>$ im Versteck ab.

GET Holt die Nachricht aus dem Versteck hervor und zeigt sie an.

LOOK Zeigt die Nachricht im Versteck an. Sie bleibt aber im Unterschied zu **GET** dabei im Versteck liegen.

DROP Verwirft die Nachricht aus dem Versteck ohne sie anzuzeigen.

HELP Erklärt, wie der Server zu verwenden ist.

QUIT Beendet die Verbindung zum Server.

Der Server heißt neue Clients mit der Nachricht `!hello: <text>` willkommen. Jeder Befehl wird entweder mittels `!<command>: ok` oder `!<command>: <text>` quittiert. Zum Beispiel wird der Befehl `pUt hello world` mit `!PUT: ok` beantwortet und auf `get` wird mit `!GET: hello world` geantwortet.

Um etwas mehr Gefühl für das Protokoll zu bekommen, kannst du Dich mit `telnet` oder `netcat` an unseren Demo-Server auf *teach-and-test.inet.tu-berlin.de*, port *9876* verbinden und etwas damit herumspielen.

(a) Stelle ein Zustandsdiagramm für den Server auf. Dafür darfst Du davon ausgehen, dass immer nur ein Client gleichzeitig mit dem Server verbunden ist.

(b) Benutzt das Protokoll Soft- oder Hard-State? Begründe.

(c) Implementiere einen einfachen Server für das Cubby-Hole-Protokoll. Neben den oben angegebenen Befehlen muss der Server mit Clients umgehen können, die ihre Verbindung beenden ohne vorher ein **QUIT** zu senden.

(d) Stelle sicher, dass der Server mit mehreren Clients gleichzeitig umgehen kann.

(e) Stelle sicher, dass der Server nicht blockiert, falls Clients unvollständige Anfragen senden. Er sollte auch mit mehreren Befehlen in einem TCP-Segement umgehen können.

Deine Abgabe muss den vollständigen Quellcode Deines Programms, sowie ein Script namens `run.sh` enthalten, das (falls nötig) das Programm übersetzt und es so startet, dass es auf TCP Port 9876

Verbindungen entgegen nimmt. Der Server darf in einer Programmiersprache Deiner Wahl geschrieben werden, muss aber ohne zusätzliche Bibliotheken zu installieren auf den Linux-Rechnern des IRB [1] laufen. Abgaben, die diese Anforderungen nicht erfüllen werden nicht bewertet!

Du darfst jegliche TCP-Server Beispiele verwenden, die Du im Internet findest, musst aber die Quelle angeben.

Wir empfehlen das gegenseite Testen der Implementierungen mit anderen Kursteilnehmern.

Abgabe bis Mittwoch, den 4. Februar 2015 nur bis 14:00 h s. t.

- **Als PDF-Dateien (keine MS-Office- oder OpenOffice-Dateien):** Mittels ISIS hochladen (`https://www.isis.tu-berlin.de/2.0/course/view.php?id=2560`)
- Gib auf deiner Lösung deinen Namen, deine Matrikelnummer **und** den Namen deines Tutors an.

[1] Eine Liste der IRB-Maschinen findest Du unter `http://wiki.freitagsrunde.org/SSH#Liste_der_Server_im_CS-Netz`.