TKN Telecommunication Networks Group

Praktikum
Technische Grundlagen der Informatik IV

Aufgabenblatt Pr1

— Berkeley Socket API —

Abgabe: 29./30.4.2013

Organisatorisches

Die praktischen Termine finden im Terminalraum statt.

Die praktischen Aufgaben sind in Kleingruppen von i. d. R. 3 Personen zu lösen. Die Ergebnisse führt Ihr im praktischen Termin Eurem Tutor vor. Reicht bitte des weiteren den Quellcode bis Sonntag vor dem Termin 23:55 Uhr per ISIS ein. Es besteht Anwesenheitspflicht.

Auf der ISIS-Seite zur Veranstaltung findet Ihr zusätzliche Literatur und Hilfen für die Bearbeitung der Aufgaben.

Aufgabe 1:

Zuerst sollt Ihr für einen fertigen UDP und TCP Server zwei entsprechende Clients programmieren. Einen bereits kompilierten Server stellen wir Euch auf der ISIS-Seite zur Verfügung (Aufruf: `tcpUdpServer <tcpPort> <udpPort>`).

Weiterhin wird euch ein Grundgerüst für den TCP Client gestellt welches ihr mit wenigen Änderungen auch für den UDP Client nutzen könnt. Für die ersten Tests erhaltet ihr auch kompilierte Versionen der fertigen Clients.

Zum Kompilieren eurer Clients unter Linux verwendet den Befehl:

`gcc -lnsl -lresolv <dateiname.c> -o <ausgabe>`

Eure Clients sollen vom Benutzer über die Kommandozeile zwei natürliche Zahlen entgegennehmen, sowie die IP Adresse und Port-Nummer des Servers. Die beiden Zahlen sollen an den Server gesendet werden. Dieser berechnet den größten gemeinsamen Teiler und gibt das Ergebnis über die Standardausgabe aus. Die korrekte Funktionsweise Eurer Clients soll für Euch anhand von Debug-Informationen nachvollziehbar sein.

Um die beiden Zahlen aus den empfangenen Daten zu extrahieren, verwendet unser Server die folgende Funktion. Passt Eure Clients also entsprechend an:

```
void unpackData(unsigned char *buffer, unsigned int *a, unsigned int *b)
{
        *a = (buffer[0]<<8) | buffer[1];
        *b = (buffer[2]<<8) | buffer[3];
}
```

Was passiert, wenn Euer TCP Client mehrere Zahlen schnell hintereinander an den Server sendet? Überprüft dies, indem Ihr z.B. den Befehl send() mehrmals in einer Schleife ausführen lasst. Wie kommt es zu dem Ergebnis?

Aufgabe 2:
Im zweiten Teil dieser Aufgabe sollt Ihr einen einzigen UDP/TCP Server implementieren, der alle Funktionalitäten besitzt, die Ihr in der ersten Aufgabe vorgegeben bekommen habt. Der Server soll mit unterschiedlichen Clients gleichzeitig kommunizieren können. Dafür muss er zwischen neuen TCP Verbindungsanfragen, Daten über bestehende TCP Verbindungen und gesendeten UDP Daten unterscheiden können.

Hinweis: Siehe Abschnitt 4.4 der "Hinweise zu Pr1": Es ist **nicht** *nötig, mit parallelen Prozessen/Threads zu arbeiten.*