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

11. Blatt Praktikum Protokolldesign WS 07/08

Aufgabe 1: (100 Punkte) **Verläßlicher Datentransfer: RDT 2.2 über UDP:**

Schreibe einen Sender, der bitfehlerresistente Kommunikation über das UDP-Protokoll realisiert. Implementiere dazu das RDT-2.2-Protokoll aus dem Buch von *Kurose und Ross: Computer Networking.*[1] Dein Programm liest Eingaben von der Tastatur und überträgt sie an den von uns geschriebenen Empfänger.

Der Empfänger liegt im Verzeichnis `/afs/net.t-labs.tu-berlin.de/home/praktikum/daten/11.uebung/` und wird von Dir mit `/afs/net.t-labs.tu-berlin.de/home/praktikum/daten/11.uebung/empfaenger rdt22` gestartet, woraufhin er in den Zustand *wait for 0* übergeht. Der Empfänger generiert automatisch Bitfehler sowohl für ein- als auch für ausgehende Pakete, verliert aber keine Pakete. Die Bedeutung zusätzlicher Commandline-Parameter (z. B. andere Wahl des UDP-Ports) erfährst Du mit `/afs/net.t-labs.tu-berlin.de/home/praktikum/daten/11.uebung/empfaenger -help`.

Wie gesagt, soll Dein Programm Tastatureingaben einlesen – was den Empfang von ACKs natürlich nicht blockieren darf! – und sie dann per RDT 2.2 in Form von UDP-Paketen an unseren Empfänger übertragen, der sie dann ausgibt. Du darfst Tastatureingaben verwerfen, wenn Du noch auf ein ausstehendes ACK wartest.

Dein Programm muss die Möglichkeit bieten zu **jedem** gesendeten **und** empfangenen Paket auszugeben:

- ob es gesendet oder empfangen wurde
- (bei Versand:) warum es gesendet wurde (z.B. neues Paket, Retransmit)
- (bei Empfang:) was es bedeutet, dass wir es empfangen haben, und unsere geplante Reaktion
- (bei Empfang:) ob es corrupt ist
- welche Sequence Number / ACK-Nummer es trägt
- wieviele Bytes an Nutzdaten es überträgt, also ohne die ganzen UDP- und RDT 2.2-Headerinformationen.
- die enthaltenen Nutzdaten, falls vorhanden

Pakete sollen über einen UDP-Socket versendet und Empfangen werden und sehen folgendermaßen aus:

- 2 Bytes Sequence Number (bei Paketen, die Du versendest) bzw. ACK-Nummer (bei ACK-Paketen, die der Empfänger versendet)
- 2 Bytes Checksumme
- 2 Bytes Länge der anschließenden Nutzdaten in Bytes
- Rest Nutzdaten (evtl. leer)

Die Headerdaten sind dabei in Netzerkbyteorder (erst Hi-Byte, dann Lo-Byte) gespeichert. ACK-Pakete, die vom Empfänger zurückkommen, tragen keine zusätzlichen Nutzdaten. Die Checksummenfunktion ist schnell erklärt: Ein ‚heiles' Paket hat als ‚Checksumme' immer und unabhängig von seinem Inhalt `0x4f4b`, ein defektes Paket dagegen irgendeinen anderen Wert[2]. `:-)`

[1] Online verfügbar unter `http://www.net.t-labs.tu-berlin.de/` — Login: `student`; Passwort: `quarter24`

[2] Das ist natürlich keine ‚echte' Checksummen-Funktion. Ihr sollt auf diesem Blatt aber ein Transportprotokoll implementieren und keinen Prüfalgorithmus

Abzugeben sind:

- Dein Programm (sowohl Quelltext als auch ggf. compiliert im Falle von C, C++ und Java)
- Die Konsolenausgabe Deines Programms im ‚geschwätzigen' Modus bei einer kurzen Beispielsession. Es sollte daraus hervorgehen, dass Dein Programm mit Corruption umgehen kann.
- Die zugehörige Konsolenausgabe des Servers von der gleichen Session, ebenfalls im ‚geschwätzigen' Modus[3]
- Ggf. eine Beschreibung, was funktionieren sollte, dies leider aber immer noch nicht tut...

Details zur Abgabe der Aufgaben: siehe FAQ (unterhalb `http://www.net.t-labs.tu-berlin.de/teaching/ws0708/PD_labcourse/`**)**
Abgabe: bis Dienstag, 29. Januar 2008, 11:59 h s. t.

[3] Dazu den Server mit
`/afs/net.t-labs.tu-berlin.de/home/praktikum/daten/11.uebung/empfaenger --verbose rdt2.2` starten