Beleg UDP-Dateitransfer für Rechnernetze/Kommunikationssysteme

Tilman Ischner

15. Januar 2023

Inhaltsverzeichnis

1 Aufgabenstellung

Erstellen Sie ein Programm (client + server) zur Übertragung beliebiger Dateien zwischen zwei Rechnern, basierend auf dem UDP-Protokoll. Das Programm soll mit der Sprache JAVA erstellt werden und im Labor S311 unter Linux lauffähig sein und dort vorgeführt werden. Folgende Punkte sind umzusetzen:

- Aufruf des Clients (Quelle) auf der Konsole mit den Parametern: Zieladresse (IP oder Hostname) + Portnummer + Dateiname + Protokoll (sw—gbn) (Bsp.: filetransfer client is311p1 3333 test.gif gbn)
- Aufruf des Servers (Ziel) mit den Parametern: Portnummer (Bsp.: filetransfer server 3333). Um die Aufrufe für Client und Server so zu realisieren ist ein kleines Bash-script notwendig (z.B.: java Clientklasse $1 $2 $3)
- Auf dem Zielrechner (Server) ist die Datei unter Verwendung des korrekten Dateinamens im Pfad des Servers abzuspeichern. Ist die Datei bereits vorhanden, soll an den Basisnamen der neuen Datei das Zeichen „1“ angehängt werden. Client und Server sollten auch auf dem selben Rechner im selben Pfad funktionieren.
- Messen Sie bei der Übertragung die Datenrate und zeigen Sie am Client periodisch (z.B. jede Sekunde) den aktuellen Wert und am Ende den Gesamtwert an. Sie können sich bei der Anzeige am Konsolenprogramm wget orientieren.
- Implementieren Sie exakt das im Dokument Beleg-Protokoll vorgegebene Übertragungsprotokoll. Damit soll gewährleistet werden, dass Ihr Programm auch mit einem beliebigen anderen Programm funktioniert, welches dieses Protokoll implementiert.
- Gehen Sie davon aus, dass im Labor Pakete fehlerhaft übertragen werden können, verloren gehen können oder in ihrer Reihenfolge vertauscht werden können (beide Richtungen!). Implementieren Sie eine entsprechende Fehlerkorrektur.

- Testen Sie Ihr Programm ausgiebig, hierzu sind Debug-Ausgaben sinnvoll. Entwerfen Sie eine Testumgebung als Bestandteil des Servers, mittels derer Sie eine bestimmte Paketverlustwahrscheinlichkeit und Paketverzögerung für beide Übertragungsrichtungen simulieren können. Sinnvollerweise sollten diese Parameter über die Konsole konfigurierbar sein, z.B: filetransfer server 3333 0.1 150 für 10 Prozent Paketverluste und 150 ms mittlere Verzögerung für beide Richtungen. Bei der Vorführung im Labor werden wir einen Netzsimulator nutzen, welcher eine entsprechende Netzqualität simuliert.

- Bestimmen Sie den theoretisch max. erzielbaren Durchsatz bei 10% Paketverlust und 10 ms Verzögerung mit dem SW-Protokoll und vergleichen diesen mit Ihrem Programm. Begründen Sie die Unterschiede.

- Erstellung eines Lernportfolios (Dokumentation Ihrer Entwicklungsschritte, des Lernfortschritts, der Misserfolge, etc.)

- Dokumentieren Sie die Funktion Ihres Programms unter Nutzung von Latex. Notwendig ist mindestens ein Zustandsdiagramm für Client und Server. Geben Sie Probleme/Limitierungen/Verbesserungsvorschläge für die Belegaufgabe und das verwendete Protokoll an.

- Der Abgabetermin ist auf der Website des Fachs zu finden, die Vorführung der Aufgabe findet dann zu den angekündigten Praktikumszeiten statt. Die Abgabe des Belegs erfolgt als tar-Archiv mit einem vorgegebenen Aufbau, Informationen hierzu werden im Dokument Beleg-Abgabeformat bereitgestellt. Plagiate werden mit Note 5 bewertet!

- Sie können zur Programmierung einen beliebigen Editor / Entwicklungsumgebung verwenden. Empfohlen wird die Entwicklungsumgebung IntelliJ IDEA welche für Studenten kostenlos erhältlich ist.

- Die Note 1 können Sie nur erhalten, wenn beide Protokolle (SW und GBN) korrekt implementiert sind.

- Optional:
  - Umsetzung des Clients in C und Nachweis der Funktionsfähigkeit durch Datenübertragung zum Java-Server.
  - Bei Interesse können Sie einzelne Klassen oder den Beleg auch in Kotlin programmieren

2 theoretischer Durchsatz

Berechnung des theoretisch max. erzielbaren Durchsatz bei 10% Paketverlust und 10 ms Verzögerung mit dem SW-Protokoll

- Durchsatz bei fehlerhaftem Hin- und Rückkanal

$\eta_{SW}\& = \frac{T_P}{T_P+T_W}(1-P_{sc})(1-P_{cs})R$

- Annahmen

  $P_{sc} = P_{cs} = 0,1$ (Kanalfehlerrate für Hinkanal P_{sc} und Rückkanal P_{cs} gleich)

  $T_P = L/r_b = 11\,224Bit/r_b$

  $L = Header + Daten = (16+8)Bit + 1400Byte/8 = 11\,224Bit$

  $T_W = 2Ta + T_{ack} = 20ms + 24Bit/r_b$

  $Ta = 10ms$

  $T_{ack} = \frac{16Bit+8Bit}{r_b} = 24Bit/r_b$

  $R = 1$ (Zur Vereinfachung der Rechnung wird der Protokoll-Overhead nicht beachtet)

- Rechnung

  $\eta_{SW} = \frac{T_P}{T_P+T_W}(1-P_{sc})(1-P_{cs})R$

  $\eta_{SW} = \frac{11\,224Bit/r_b}{11\,224Bit/r_b+24Bit/r_b+20ms}0,9^2$

  $\eta_{SW} = \frac{11\,224Bit/r_b}{11\,248Bit/r_b+20ms}0,9^2$

3 Implementierung

Implementierung mit Kotlin unter Nutzung der bereitgestellten Klassen. Dabei wurde sich auf die Implementierung des SW-Protokolls beschränkt. Es wurden folgende Funktionen implementiert:

- Client

  Client wird mit den Parametern **Übertragungsmodus** also hier client,**Zielipadresse**, **Portnummer**, **Dateiname** der zusendenden Datei sowie **Übertragungsprotokoll**(abgekürzt mit sw und gbn für die jeweiligen Protokolle, jedoch nur Stop-and-Wait implementiert) aufgerufen.

  Client zeigt Übertragungsfortschritt sowie Übertragungsrate an (wird mit jedem gesendeten Paket aktualisiert).

  Client zeigt am Ende einer erfolgreichen Übertragung die Gesamtdauer und durchschn. Geschwindigkeit an.

- Server

  Server wird mit den Parametern **Portnummer** sowie ggf. mit den Parametern **Wahrscheinlichkeit für Packetverlust** und **Verzögerung** aufgerufen.

Server speichert Datei in seinem Ausführordner ab.

Ist die übertragene Datei bereits beim Server vorhanden, wird diese mit einer 1 am Ende des Dateinamens abgespeichert.

4 Zustandsdiagramme

4.1 Client

4.2 Server