Aufgabe 3a: Programmierung eines Dateitransfer-Clients

In dieser Aufgabe sollen Sie einen Dateitransfer-Client entwickeln, der an einen vorgegebenen Dateitransfer-Server (programmiert in Java) einzelne Dateiobjekte transferiert und hierzu das UDP-Transportprotokoll verwendet. Da UDP kein zuverlässiges Transportprotokoll ist, müssen Sie sich bei der Programmierung des Clients selbständig um die richtige Reihenfolge, Flusskontrolle und mögliche Paket-Verluste kümmern. Als Verfahren hierfür soll das auch in der Vorlesung behandelte „Selective-Repeat“ (Foliensatz RNAI-05-Layer_4_v03.pdf, Seiten 86f., deutet nur die Lösung an, aber mit dem Stichwort kommen Sie weiter) verwendet werden.

Die Realisierung kann in Java oder anderen Sprachen erfolgen (allerdings dürfte dies eher umfangreich werden, wenn Sie nicht den mitgelieferten Rumpf einer Java-Entwicklung nutzen). Eine bessere GUI als die, die im mitgelieferten Rumpf realisiert ist, ist nicht erforderlich, da es sich um ein reines Copy-Utility handelt. Bei der Ausgabe (siehe auch Aufgabe 3b) sollten Sie daran denken, dass beliebige Freitexte auch beliebig komplex zu parsen sind. Konzentrieren Sie sich bei der Ausgabe von Werten auf einfache CSV-Formate, so dass Sie die Werte z.B. direkt in eine Tabellenkalkulation übernehmen können, um mit den üblichen Office-Werkzeugen aussagekräftige Grafiken zu erzeugen.

Client:

Dem Dateitransfer-Client werden beim Aufruf vom Benutzer folgende Parameter als Argumente übergeben, einige können natürlich auch als Initialisierung clever hinterlegt werden:

1. Hostname oder IP-Adresse des Dateitransfer-Servers
2. Portnummer des Dateitransfer-Servers
3. Quellpfad inkl. Dateiname der zu sendenden Datei auf dem lokalen System
4. Zielpfad inkl. Dateiname der zu empfangenden Datei auf dem Dateitransfer-Server
5. Window-Größe N (> 0, ganzzahlig)
6. Fehlerrate ERROR_RATE (>= 0, ganzzahlig) zur Übergabe an den Dateitransfer-Server

Bevor mit der Übertragung der Quelldatei (Parameter 3) an den Dateitransfer-Server begonnen wird, sendet der Client zunächst als erstes Paket mit der Sequenznummer 0 die Steuerdaten gemäß folgender Tabelle:

Datum	Datentyp	Beschreibung
Sequenznummer	8 Byte ohne Vorzeichen	Das erste Paket hat immer die Sequenz-nummer 0, danach wird diese je Paket hochgezählt
Zieldateipfad	String in UTF-8-Codierung mit beliebiger Länge	Pfad und Dateiname der Zieldatei auf dem Server ohne das Zeichen „;“ (falls bereits vorhanden, wird die Datei überschrieben)
Trennzeichen	1 Char	„;“ (Semikolon) als Trennzeichen
Window-Größe	1 Int	Die Angabe der Window-Größe gemäß behandeltem Verfahren aus der Vorlesung
Trennzeichen	1 Char	„;“ (Semikolon) als Trennzeichen
Fehlerrate	1 Long	Die Angabe der Fehlerrate, die der Server simulieren soll

Anschließend überträgt der Dateitransfer-Client die Datei an der Dateitransfer-Server, die dort gespeichert wird. Der Client muss dafür sorgen, dass verlorengegangene Pakete von ihm erkannt und gemäß dem **Selective-Repeat-Verfahren** sollen nur diese Pakete erneut übertragen werden. Zur dynamischen Berechnung der **Timeoutzeit** soll der Algorithmus des TCP-Verfahrens verwendet werden (Formeln gemäß den Folien der Vorlesung, FCdata.pdf siehe Seite 56 ;)

Server:

Ein neuer Dateitransfer wird beim Eintreffen des ersten UDP-Pakets gestartet, wobei gleichzeitig der Client durch IP-Adresse und Quellport eindeutig festgelegt wird. Der Server ist nicht multi-threadfähig, es kann daher nur ein Kopierauftrag zur Zeit verarbeitet werden. Evtl. eintreffende UDP-Pakete von anderen Absendern werden ignoriert, solange ein begonnener Kopierauftrag noch nicht beendet ist. Der UDP-Port, auf dem der Dateitransfer-Server Aufträge erwartet, muss beim Start des Servers als Parameter angegeben werden können.

Wenn sich der Client für 3 Sekunden nicht gemeldet hat, beendet der Server den Kopierauftrag und schließt die Speicherung der Dateidaten ab. Evtl. verzögert oder später eintreffende UDP-Pakete von diesem Client werden ignoriert. Der Dateitransfer-Server steht sofort wieder für einen neuen Auftrag zur Verfügung.

Der Server arbeitet als Empfänger der Datenpakete ebenfalls nach dem Selective-Repeat-Verfahren und versendet entsprechende Bestätigungspakete als positive Quittung (sogenanntes ACK). Diese enthalten jeweils nur eine Sequenznummer. Jedes ACK wird allerdings mit einer zusätzlichen Verzögerung von 10 ms zur Simulation einer hohen Ausbreitungsverzögerung gesendet.

Der Server ist in der Lage, zur Simulation von Paketverlusten, jeweils das n-te empfangene Pakete als fehlend zu melden (Parameter ERROR_RATE = n mit n>=0). Die zu simulierende Fehlerrate wird vom Dateitransfer-Client mit dem ersten Paket (Sequenznummer 0) als Teil der Steuerdaten an den Server übermittelt.

FSM-Spezifikationen:

Client:

- Datenstrukturen:
  - Sendepuffer mit N Plätzen (N: Window-Größe)
  - sendbase (Sequenznummer des ältesten Pakets im Sendepuffer)
  - nextSeqNum (Sequenznummer des nächsten zu sendenden Pakets)
- Ereignisse und Aktionen:
  - Das nächste Paket der zu übertragenden Datei ist gelesen:
    - Lege das Paket im Sendepuffer ab, falls der Puffer nicht voll ist
    - Sende das Paket mit Sequenznummer nextSeqNum
    - Starte Timer für das Paket
    - Erhöhe nextSeqNum
  - Timeout von Timer für Paket mit Sequenznummer n:
    - Wiederhole das Senden von Paket n
    - Timer für Paket n neu starten
  - ACK(n) empfangen und Paket n ist im Sendepuffer
    - Paket n als quittiert markieren
    - Timer für Paket n stoppen
    - Timeoutwert mit gemessener RTT für Paket n neu berechnen
    - Wenn n = sendbase, dann lösche ab n alle Pakete, bis ein noch nicht quittiertes Paket im Sendepuffer erreicht ist, und setze sendbase auf dessen Sequenznummer

Server:

- Datenstrukturen:
  - Empfangspuffer mit N Plätzen (N: Window-Größe)
  - rcvbase (die Sequenznummer des nächsten Pakets, das bei Reihenfolgeerhaltung ausgeliefert werden muss)
- Ereignisse und Aktionen:
  - Paket mit Sequenznummer n korrekt empfangen und n in [rcvbase, rcvbase+n-1]
    - Sende ACK(n)
    - Sortiere Paket n anhand der Sequenznummer in den Empfangspuffer ein (falls dort noch nicht vorhanden)
    - Liefere - beginnend bei rcvbase - alle Pakete aus, die sich in lückenlos aufsteigender Reihenfolge im Empfangspuffer befinden und lösche diese aus dem Empfangspuffer
    - Setze rcvbase auf die Sequenznummer des letzten ausgelieferten Pakets n + 1
  - Paket mit Sequenznummer n korrekt empfangen und Paket n in [rcvbase-N,rcvbase-1]
    - Sende ACK(n)
  - Paket mit Sequenznummer n korrekt empfangen und (n < rcvbase-n oder n ≥ rcvbase+n):
    - keine Aktion

Vorgaben für die Implementierung

Durch den vorgegebenen Rahmen brauchen Sie sich nicht selbst um viele Aspekte der Robustheit und Zuverlässigkeit zu kümmern. Beachten Sie jedoch die oben gemachten Aussagen und vermeiden Sie es, gleich mit sehr großen Dateien zu testen. Fangen Sie zunächst mit kleinen und einfachen Tests an. Hierzu gehört auch, dass es durchaus sinnvoll und auch erlaubt ist, sowohl Server als auch Client zunächst auf demselben Rechner zu betreiben.

Wenn die Funktion sichergestellt ist, machen Sie die Tests und die Testumgebung immer etwas komplizierter. Dies gilt auch für den Einsatz der Parameter z.B. zur Fehlerrate. Erstmal

müssen Sie sicherstellen, dass es eben bei einem Wert von 0 korrekt funktioniert, danach geht es weiter mit komplizierteren Fällen.
Zur Verarbeitung von Informationen über alle verwalteten Objekte sind jeweils eigene Datenstrukturen (Klassen) zur Beschreibung hilfreich. Integrieren Sie eine Sequenznummer in jedes UDP-Datenpaket (als die ersten 8 Byte des Daten-Bytearrays). Benutzen Sie dafür die Methoden der mitgegebenen Klasse `FCpacket` und speichern Sie im Sendepuffer nur Pakete vom Typ `FCpacket`, damit Sie auch weitere Zusatzinformationen für ein gesendetes Paket ablegen können!

Implementieren Sie den Empfang der Quittungen beim Sender (Client) als einen weiteren Thread und nutzen Sie den Sendepuffer, der die unbestätigten Pakete speichert, zur Synchronisation der Threads.

Die Software sollte auf der gelieferten Vorgabe aufsetzen. Ein Entwurfsdokument muss mit üblichen Office-Werkzeugen erstellt und beim Praktikumstermin ausgedruckt vorlegt werden. Es muss folgendes beinhalten:

a) Sequenzdiagramm (vergleichbar des Beispiels für Aufgabe 1a)

b) Liste der unterstützten Aufrufparameter für den Client

Für die Abnahme während des Praktikums müssen Sie nachweisen, dass:

- Der Server übertragene Dateien korrekt und ohne Fehler kopiert – auch bei unterschiedlichen Window- bzw. Fehlerraten
- Der Server eine Übertragung nach 3 Sekunden ohne neue Daten abbricht

Wenn Sie sich auf die Abnahme vorbereiten, müssen Sie die obigen Tests selbst durchgeführt haben. D.h. keine Abnahme ohne die Erklärung, dass Sie sich selbst überzeugt haben, dass der Client macht, was er machen soll!

Aufgabe 3b: Analyse von Dateitransfers

Testen Sie Ihre Anwendung mit der Datei **FCData.pdf** und verschiedenen Werten sowohl für die Fenstergröße (WINDOW_SIZE: 1, 10, 100) als auch für die Fehlerrate (ERROR_RATE: 10, 100 und 1000).

Geben Sie dazu durch den Client direkt nach einer Übertragung folgende Angaben aus:

a) Gesamtübertragungszeit für die gerade übertragene Datei

b) Anzahl an Timerabläufen und wiederholten Übertragungen

c) Anzahl an empfangenen Bestätigungen

d) der ermittelte Mittelwert für die RTT aller ACKs

Verwenden Sie für die Auswertungsläufe immer **zwei unterschiedliche Rechner** als Client bzw. Server sowie jeweils **lokale Verzeichnisse** für die zu übertragenden Dateien, um die Messung nicht durch zusätzlichen Netzwerkverkehr zu verfälschen.

Verwenden Sie die dieselbe Datei für alle Tests und lassen Sie diese Datei für jede Parameterkonfiguration jeweils fünf (5) mal übertragen. Der Server muss natürlich bereit sein für eine erneute Übertragung, bevor der nächste Test beginnen darf. Notieren sie alle Ergebnisse dieser Auswertungsläufe bzw. erleichtern Sie sich die Arbeit, indem Sie durch geschickte Parameterwahl und etwas Skripting den gesamten Testablauf automatisieren.

Wenn Sie – nach entsprechenden Überprüfungen – von der korrekten Funktion überzeugt auf die Ausgabe unnötiger Angaben verzichten, können Sie auch die Sammlung der Daten automatisieren. Dabei werden Sie allerdings auch die verwendete Parameterkonfiguration mit protokollieren müssen, um später zusammengehörende Tests zu erkennen.

Für das Protokoll dokumentieren Sie bitte:

- die Ausgaben des Clients sowie die für Fenstergröße und Fehlerrate verwendeten Angaben, sowie
- eine kurze Analyse der erhobenen Daten mit Begründung, wodurch die Veränderungen bei den unterschiedlichen Werten zu erklären wären.

Für die Abnahme während des Praktikums müssen Sie erklären können, wie sich die Zusammenhänge zwischen Window-Größe, Fehlerrate und Ihren ermittelte Werte ergeben.