Task 3: File transmission protocol with throughput control
SKJ (2017)

Introduction

Transfer of files is a common network activity. This proces must be reliable. File transfer potocols usually try to obtain the highest transfer speed, but it is not the only possible attitude. In this project, we are going to build a file transfer protocol using UDP. The protocol must be reliable and include mechanisms for data throughput control.

Specification

The main aim of the project is design of the data transfer protocol:

1. As a transmission protocol in layer 4, the UDP protocol must be used. It imples, that for data sending/receiving **only** the `DatagramSocket` and `DatagramPacket` classes and their methods may be used.
2. The protocol must be reliable (which is not provided by UDP). In paticular, it must control the consistency of the transfered data, it must provide retransmissions in case of data loss and must preserve their ordering.
3. The protocol allows for specification of an average data sending speed (concerns only the file data, does not include any additional information needed by the protocol), which must be obeyed by the process sending the data. The average must be preserved in every next 1s interval (in particular, it is not allowed to send the whole file and then wait to obtain proper average speed, unless a file is smaller than the volume of data to be sent within 1s interval). The transfer speed parameter may be adjusted during data transfer many times and the sending process must adjust to it.
4. Verifies the data consistency by computing and comparing the MD5 sum of the transferred data of both initial and the target file. In order to determine MD5 sums, standard Java clases may be used.

A designed protocol may be implemented as a pair of application: the **client** and the **server**

Client

The **client** is a process, which sends the data. It accepts the following parameters at runtime (in any order):

- `-server <address>` – specifies an address, at which the server application works.
- `-port <port numer>` – specifies a UDP port number, at which the server application works.
- `-file <file name>` – a the name of a file for transmission.

After the client starts, it connects to the specified server. After the connection is established, the client sends to the server the name of the file (and additional data if needed, according to the protocol), receives from the server the initial transfer speed value and starts to send the file specified as the parameter. During the transmission it must react to the server's responses – request of transfer speed change, messages about succesfull/unsuccessfull delivery, etc. After the whole file is delivered and its contents are verified, the client sends to the server a message about success/failure and finishes its work.

During tranfer, the client prints out the following information about the transmission:

- te number of delivered bytes and the number of bytes left,
- current average speed computed from the beginning of the transfer,
- current average speed computed during recent 10s,
- current average speed computed during recent 1s r information about data transfer errors, if they appear.

After the transmission, the client shuld print aggregated information about the transfer: the number of bytes sent, transmission time, average transmission speed, verified MD5 sum.

Server

The server's goal is to accept a connection from a client and to receive a file delivered by this client. At execution, the server accepts the following parameters (in any order):

- `-port <port numer>` – specifies a UDP port number, at which the server application works.
- `-speed <initial speed>` – initial data transfer speed expressed in KB/s (1KB=1024B).

After execution, the server waits for a client's connection. After the client connects, the server receives a name of a target file (and additional data required by the protocol), it sends back the value of initial transfer speed and starts receiving the data, according to the designed protocol. The server must control correctness of the delivered data and

reaction to any spotted errors (data not delivered, error in transmission, etc.). After the transfer, the server sends to the client, the MD5 sum of the file and awaits for a message about success/failure of the transfer. The file should be stored in the directory, in which the server works (if a file with the same name already exists, its contents should be erased and overwritten).

During work, the sever may read from the keyboard a new data transfer speed value (in KB/s). If such value is read, the server sends it to the client, which is obliged to adjust to the new value. Such changes can be done at any moment and many times during the transfer.

Grading and requirements

1. Fully and correctly implemented project is worth **7 points**. Implementing each of the following functionalities is rewarded up with the specified number of points:

   (a) The project of a correct protocol providing functionalities described in the task is worth **5 points**. The project desription must contain information about:
   - connection establishment: what messages are sent, what comes back as a reply (no authorization is neede, just the data required for transmission must be exchanged) (1p.),
   - data transmission: what additional data are sent, how do they provide proper ordering of data, how reliability is granted, what is the reaction to faulty delivery or no data delivered (2p.),
   - method of data consistency verification during and/or at the end of transfer (1p.),
   - how the data transfer speed is controlled: how the client adjusts to requested limits (1p.)

   (b) Implementation of client and server processes, according to the protocol: **2 points**.

   (c) Additional 1 extra point can be obtained if the designed protocol is resistant to such data transfer disorders like reception of data (by both client and server) sent from a machine not taking part in the transfer.

2. The program should be written in Java, in compliance with Java 8 standard (JDK 1.8). Only basic TCP socket classes can be used for implementing the network functionality. A different language can be used with permission of teaching assistant.

3. The projects should be saved in appropriate directories of EDUX system not later than on 28.01.2018 (the deadline can be moved by the teaching assistant).

4. The archived project file should contain:

- File *Documentation(indexNo)Task3.pdf*, describing the designed protocol, what has been implemented, what hasn't, what were the difficulties and what errors are still present.
- Source files (JDK 1.8) (together with all the libraries used by the author that are not the part of Java standard library). The application should compile and run on PJA lab's computers.

NOTICE: THE DOCUMENTATION FILE IS NECESSARY. PROJECTS WITHOUT DOCUMENTATION WILL NOT BE GRADED.

5. Solutions are evaluated with respect to the correctness and compliance with the specification. The quality of code and following software engineering rules can in uence the final grade.

6. UNLESS SPECIFIED OTHERWISE, ALL THE AMBIGUITIES THAT MIGHT ARISE SHOULD BE DISCUSSED WITH TEACHING ASSISTANT. INCORRECT INTERPRETATION MAY LEAD TO REDUCING THE GRADE OR REJECTING THE SOLUTION.