Programming in Java

Coursework assignment 5 — TCP and UDP

2014-15

1 Objectives

The goal of this coursework assignment is to write a simple client-server application where multiple clients connect to a server. The first client to connect will send a looping audio recording (in chunks) to the server. The server will relay the audio stream to all the clients who connect after this.

Even though we haven't studied the TCP and UCP protocols directly you should have enough information, together with appropriate online sources, to tackle this (interesting) problem. It is more representative of the commercial usage of Java networking than is RMI, which is usually *hidden* under other layers.

2 The problem description

The first client to connect to the server sends audio the audio. Clients that connect to the server after this will receive the audio and play it back.

2.1 Protocol usage

TCP — connection is used for signaling

UDP — connection is used for streaming data

2.2 Suggested architecture/program flow

Server main

1. Server starts listening for clients (TCP)
2. Client connects to server
3. Server places client connection and further handling of client in a separate thread
4. go back to 1

Client

1. Client connects to server (TCP)
2. Asks for *Unique ID*
3. Asks if its the first to connect
4. Open UDP connection to server
   (a) If first client: send audio (in chunks) to server
   (b) If not first client: listen for audio chunks coming in on UDP and play audio

Server Client handling

1. Send unique ID
2. Indicate to client if it is a sender or receiver process
3. Listen for UDP connection
4. Tell client to connect over UDP
5. Relay Audio data

In the case where the sender client stops its execution the server has to decide which of the other clients has to take over the sending task.

3 The deliverables

The assignment must be pushed to a project called `UDP` on your GitHub account; it will be automatically cloned on the due date. You should upload your classes, documentation, and unit tests, as part of your submission. You should include full javadoc for your classes, and a `README` file detailing what the project is about and how to run your system (with an example). You are encouraged to have everything ready well in advance – both the programming and pushing it to your GitHub account – to avoid last-minute problems (e.g., GitHub may be down for maintenance).

The assignment will be graded according to its ability to fulfil the above scenario; the simplicity, clarity, and generality of the code (including succinct but illustrative comments and JavaDoc); and the compliance with good practices of coding, and version control as outlined during the module (e.g., committing often and in small pieces, use of descriptive commit messages, committing only source code and not binary or `.class` files).

Plagiarism

Regardless of the times you choose to push your changes to GitHub, you should commit early and often. In case of suspected plagiarism, your version control history will be used as additional evidence to judge the case. It is in your best interest to commit very often (and to use adequate commit messages) to make it clear that the process of creation is entirely your own.

4 Credits

Thank you to Olivier Van Acker and Ryan Durling for the basis for this coursework assignment.