Project 5: File Sharing Program

Contents

1. Introduction

For this project you are required to implement a peer-to-peer (P2P) file sharing program similar to programs like uTorrent. Your program will not have all the functionality that is provided by something like uTorrent, yet the principles used will give you insight into the problems faced when implementing anonymous P2P software.

2. General Information

Please note the following points when implementing your solution:

1. You may code in any programming language, however we recommend Java. The assistants might not know your chosen language and would be unable to help you if you run into trouble.
2. You are to work in allocated groups for this project.
3. This project is to be accompanied by a report.
4. The use of virtual networking tools such as Hamachi for Linux or ZeroTier for Mac must be used when communicating between the clients and the server.
5. All project source code and reports will be checked for plagiarism, including being checked against solutions from previous years. If you are found guilty there will be serious consequences, including receiving a mark of zero for the project.

3. Project Specifications

You will implement both a client and a server for this project.

Server

As with the chat program, the server is a minimal program that orchestrates the actions and interactions of the clients. It does not need a GUI, but system messages such as connection

requests, disconnection requests and search requests, etc., should be displayed either in the command-line or a GUI. The following aspects must be implemented:

1. Clients must remain anonymous and only be known by a username.
2. The server does not store the list of files shared.
3. The server handles all requests from clients.
4. The clients only talk to each other after the server has 'introduced' them.
5. Duplicate usernames must be handled when clients connect to the server.

Client

The client provides an interface to the user. You must implement a minimal GUI, and the following must be implemented:

1. Multiple clients connect to a single server.
2. A list of online users MUST NOT be present, as the users are anonymous.
3. Clients choose a unique username at connection time.
4. Client connections/disconnections must not disrupt the server or other clients.
5. Clients must be able to search for files.
6. Searches must return exact matches and also close or substring matches.
7. Searches may not be broadcast. Searches must go via the server.
8. Provide a file download/upload progress indicator. More marks are allocated for the use of a GUI progress indicator.
9. Pause and resume functionality for downloads only.
10. The client does not have to be able to handle concurrent download or upload streams. One download/upload at a time is sufficient. It must however be able to handle one download and one upload at the same time. You can do this by using different ports for download and upload.

Interaction

When a client requests a file download, the following procedure should be followed:

1. The client sends a download request to the server with a randomly generated key referred to as the 'message-key'.
2. The server passes the originating client's address, request and message-key to the target client.
3. If the target client is accepting download requests, it contacts the originating client with the address and message-key provided by server.
4. The originating client then accepts and opens the stream if the message-keys match.

Security

You may use any form of security, and the number of marks will be based on the level of security it brings. The most common use would be encrypting the messages, especially the message-keys, exchanged between the clients and server. Remember that encryption uses a *different* key for encryption. This different key can be referred to as the encryption-key.

4. Repos and submission

Each group shall be given a GitLab repo to work in. The contents of the repo must contain a ***README.md*** to explain how to compile and run your program, and a ***report.pdf*** to be read by your

markers. For submission, compress your repo into a single tar or zip file and only one person from the group submits it to SUNLearn.

5. Guidelines

Report

The page limit for the report is strictly 8 pages. Fit the following sections into this page limit:

1. Title
   Your document must have a title which reflects that of the project you are implementing. Furthermore, you must indicate the names, student email addresses and roles of all group members.

2. Introduction / Overview
   Explain what the purpose of the project was and give an overview of the main sections in the document in a short paragraph, not a table of contents.

3. Features
   Since you will implement most features, indicate which of the required features you did not implement. Also indicate what additional features you implemented.

4. File Descriptions
   Provide concise descriptions of the source files present in your project. You may ignore files that have minimal functionality, for example a file containing only two member fields.

5. Program Description
   Provide a description of your program flow. We suggest that you tailor this according to the design of your implementation. Since most implementations will be event based / reactive, we recommend that you structure your program flow description as follows. Start with a birds-eye view of what components your implementation comprises of, how these components interact with one another, and what purpose of each is.
   Finally, explain the program flow that takes place for all features as specified in this document. Example of features include client disconnection handling, client interaction, and message routing.

6. Experiments
   The report should have at least three experiments done, where each experiment follows the scientific method which includes a question and/or hypothesis, experimentation, and results/conclusion. Make sure you provide the data that you have gathered, as well as identify the dependent and independent variables for each experiment.

7. Issues Encountered
   Give a brief description of any problems or issues that you encountered.

8. Design
   Mention any interesting design decisions that you made in your implementation.

Source Code Documentation

1. Class level
   You must use Javadoc format to provide a short summary of the program description from the report, i.e. a birds-eye overview of the main functionality for each class.
2. Method/Function level
   Again, use Javadoc format. Generally, methods/functions should be documented with the following:
   - A single line describing the main utility
   - A paragraph describing the algorithm
   - Inputs, outputs and return value
   - Caveats and limitations
3. In-line level
   Use comments sparingly inside method bodies; for intricate code only. Don't state the obvious. Also, provide comments that explain why you are doing something rather than what you are doing, since what you are doing should be obvious from the method name.

6. Marking Scheme

File Sharing	**15**
Searching	4
Check matching message-keys	1
Progress indicator	4
Indicate incremental progress	2
Files received correctly	4
Transfer	**10**
Uploading and downloading	5
Pause and resume	5
Server	**6**
List of files	2
Peer-to-peer	2
Searches go through server	2
Stability	**6**
Client	3
Server	3
General	**9**
Client GUI	2
Security	5
No displayed user list	1
Unique usernames	1
Report	**16**
Include all sections specified in report guidelines	8
Language, grammar and spelling	3
Project description	2
Experimental hypotheses and results	3
Documentation	**8**
Class level	2
Method level	2
In-line level	2
Readme present	2
General	**5**
Presentation quality during demo	2
Hamachi/ZeroTier	3
TOTAL	**75**