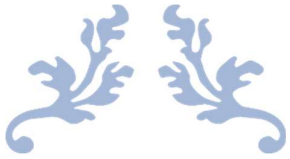

FILE SHARING PROGRAM

Project 5

OCTOBER 15, 2022

Markus Sass (22548890@sun.ac.za)
Chris Langeveldt (23632135@sun.ac.za)

Table of Contents

Overview

The aim of this project was to implement a peer-to-peer file sharing program. The program consists of a server and clients, where the server is quite minimal because file sharing happens client-to-client. Our project builds on from the previous project , however, added is the function of secure files being transferred.

Required Features Implemented

Client GUI:

1. Implemented

Stability:

1. Client is stable
2. Server is stable

Searching:

3. File lists are not stored on server
4. Searches return exact matches
5. Searches return substring and close matches
6. Searches does not disrupt the system

File Transfer

7. Encrypting of key for file sharing
8. Single connection stream works
9. Double connection stream works
10. File downloads correctly
11. File uploads correctly
12. Pause/Resume functionality
13. Progress indicators

Unimplemented Features

Most project required features are implemented, however security features were kept to a minimum. Upload does not pause when download pauses.

Additional Features Implemented

- Chatrooms
- Server GUI

- Concurrent download or upload streams
- '/myfiles' functionality
- Can pause multiple times in one download
- '/help' functionality
- Whispering to any amount of clients
- Intricate client GUI
- List of online users
- download based on index, not long file name

Description of files

Server.java
The server with GUI that handles connections.

ClientHandler.java
This class function as the threads of the server class that handles each client connection. The class is responsible for all communication to and from the server class with the different clients.

ClientListenerThread.java
This class receives all messages and files from other clients. This is run on its own thread to allow for simultaneous transfers.

TableRow.java
Each TableRow object is an entry in the NAT-table. It stores the relevant information needed in the table.

Client.java
The Client class is the platform for communication and sending of files to each other.

Message.java
This class is an object for the messages.

SendThread.java
This class is responsible for sending files securely over tcp on a separate thread. Updates the upload progress bar.

ReceiveThread.java
This class receives the file and updates the progress bar, it also allows to pausing and resuming of files sent.

Program Description

The initiated with the server being started. To create a client, compile client.java and run client, whereafter the connection is reflected within the server's GUI. This process can be repeated multiple times to allow for a variety of connections.

Once connected, clients can do all project one outcomes, plus a few more, mentioned in the spec above. However, now the ability to transfer files to one another through a local hosting exists. This can be called through the relevant commands mentioned in "/help". Once the file is 'uploaded', another client can request to download the files. Upon the request the user can accept it, should they want it.

Before sending the file, a secret key is generated and securely checked with the receiving client. If they match then the file shares, transferring the bytes, while the progress bar updates dynamically and allows for the transferring process to be paused at any given time. This process can be repeated or simultaneously done with uploads and downloads.

Experiments

File sending/receving correctly
- **Hypothesis:** The file is sent correctly and will match the hashfile.
- **Variables used:** The file (dependent)
- **Method:** $ certutil -hashfile <file> MD5
- **Results:** c04a2b0d42a20773da2be97c997a9edd
- **Conclusion:** The results match the file's hashfile, hence, it can be concluded that hypothesis is true and that the file sends correctly.

File search correctly:

- **Hypothesis:** The file is found when a substring within the name is searched.
- **Variables used:** File name (dependent)
- **Method:** The file is uploaded and then search for using the command /search <filename>.
- **Results:** The file does correctly appear when the matching letters are entered.
- **Conclusion:** The hypothesis holds. The searching returns substring, ie: not only exact matches but also keywords, which returns the closest match of that substring.

Security keys send/receive:

- **Hypothesis:** The security key is sent and received by both clients.
- **Variables used:** key (dependent)
- **Method:** Requesting to send a file, and then printing the key on both sides.
- **Results:** The keys match - 17d709436c3852c1 and file is sent and received.
- **Conclusion:** The hypothesis holds.

Progress bar:

- **Hypothesis:** The progress bar will show a live transfer rate as well as being a able to pause the download. The upload pausing will not.
- **Variables used:** The pausing (independent), the progress bar (dependent).
- **Method:** Observe the change of the bars respective to the size of the folder.
- **Results:** the bars change concurrently and consistently.
- **Conclusion:** The hypothesis holds.

Conclusion

From these tests we concluded that our implementation acts as we expected it to act with regards to transferring files correctly.

Issues Encountered

During the demo we encountered a few issues that were not present during extensive demo testing, such as the upload bar not pausing after pause was pressed.

Compilation

Server
javac Server.java

Client
Javac Client.java

Execution

Server
java Server

Client
java Client

Libraries and Packages

- java.io
- java.net
- java.util
- java.awt
- javax.swing