COMPUTER NETWORKS
P2P Message and File Sharing App

Niket Dixit - 150448
Meet Kumar Jigar Sanghvi - 150400
Shashank Shekhar - 150662
Saket Harsh - 150617

1 Goal of the Project

The goal of the project is to create a Messaging app which also allows File Sharing between the connected peers. Initially we set about to create a messaging app only with features of private chat and broadcasting, but soon we felt that file-sharing was also an essential part of any messaging app, hence we tried to provide this functionality too in the app. The app is a completely terminal based application developed using Python.

2 Features of the App

The following points mention the salient features of the Application:

1. **Signup**: Being a new user, one can signup to the app by choosing a valid username (one that is not already taken) and password.
2. **Encrypted Password**: The passwords are stored in a 'passwords.txt' file in an md5 encrypted format. Hence even if someone is able to get the 'passwords.txt' file, one can't still access anybody else's account. The encryption provides an added layer of security.
3. **Active Users list**: Whenever a user wants to have a private chat session with someone else, he is displayed the current list of active users. He/She can chose the person to chat with from this list.
4. **Private Chat**: One has the option to start a private chat session with anyone of the active clients at that time.
5. **Broadcast**: Any active user has the added functionality to go into broadcast mode. When in broadcast mode, all the users sitting idle or already in broadcast mode would engage in a community broadcast chat session. Thus anyone in this session would be able to receive and send messages to all such users.
6. **File Sharing**: We have provided file sharing options with several drive like features. Users can store files at a centralized location and provide access to other users as and when required. The following user options are implemented as part of the file sharing feature:
    - **List Files**: Shows all the files that are uploaded by the particular user to the centralized location. Also all other files that are shared with the user by other users can be seen using this option.
    - **Upload file**: This option can be used to upload a file by a user to the centralized location. This file can be chosen from the users file system (i.e. the users local storage)
    - **Download File**: This option allows the user to download any file uploaded by him or any file that is shared with him. If the particular file requested for download doesn't exist than the user is notified.
    - **Delete File**: This option can be used by the user to delete any file that is uploaded by him.
    - **Give Access**: This option can be used to share a particular file uploaded by the user with any other user. The user that he has given the access to can now check the file out.
    - **Revoke Access**: This option allows the user to revoke the access of a particular file provided to another user, if that file has been previously shared with that user.
    - **List Shared**: This option lists all the files shared by the user with any other user. It shows the mapping of files shared with the corresponding users with whom the file is shared.

3 Implementation

The application is a terminal based application written completely in Python 3 from scratch. We have used basic socket and threading libraries for the implementation. We have also used the hashlib library for password encryptions.

4 How to run

- Unzip the file.
- The server's name is server.py and client's name is client.py
- Distribute the client.py and supporting files to all computers you want to connect to the server
- Run server.py with the IP address of the computer (on which the server.py file is running)
- Run client.py file on other computers you want to connect to the server.
- Note that all computers must be on the same local network.
- Provide the ip address of the server to connect.
- You will now be guided by the program.

5 Limitations

- Once a client is in the idle state, it is not possible to quit if all other are on broadcasting.
- When two peers get involved in PrivateChat, the user initially in Idle mode would still be getting the broadcast messages.
- The password is first sent to the server in plain-text, and then the password is encrypted.
- The history of the chats is not saved.
- A user cannot send message to someone who is offline.
- Note that the last two points can be resolved by maintaining files to store chats. However, we have not implemented that.