COMP3331 22T2 Assignment Report

By Zheng Luo(z5206267)

Program Design

The program is designed into two parts, client and server, and both are written in Python 3.

- server.py:
  - Provides TCP connection with potential multiple clients, interact with clients based on given commands.
- client.py
  - Provides user interface, TCP connection with server, as well as the ability to upload and download files to or from other peers using UPD connection.
- helper.py
  - Provides helper functions for server.
- credentials.txt
  - Acted as a username database, storing all existing usernames and passwords, to authentical potential login request from client through server.

There are several assumptions other than the ones mentioned in the specification:

1. The username must be different in the credential.txt, so there will not have the same username and different password existed.
2. The user cannot login again while itself has active already.
3. There must not have empty line in the credential.txt, which will cause the authentication function to be crashed.

The information about active user, available rooms, and the chats inside the room have mainly been recorded inside the list in server, and the log files are changing based on the list.

The list formats are:

1. Active user list:

```
[
    {
        'username':username,
        'address':self.clientAddress,
        'UDPPortNumber':UDPportNumber,
        'activeTime': loginInTime
    }
]
```

2. User blocked list for blocking the user to login within 10 second:

```
[
    {
        "username": username,
        "blockedTime": printCurrentTime()
    }
]
```

3. Available room list and the messages in each room:

Application Layer Message

The input from the terminal will be sent to client, which will perform preliminary checks against the required format and passes the requests as string with encoding to the server. The server will decode and extract the information from string into the required format, and perform corresponding actions based on the request. Then pass the resulted information as String back to the client.

System Overview

User first need to login to their account on the client portal while server is activating, the account will be blocked for 10 seconds if the number of fail attempts have been reached. After successfully logged in, the command prompt will be appeared to guide the user for inputting one of the seven available commands. These commands are:

1. BCM: broadcasting message to everyone with 1 argument of message as input.
2. ATU: downloading the information about active users in the server to client, with no argument as input.
3. SRB: creating chat room for desired users, arguments are the username that client wish to chat with.
4. SRM: sending message to the specified chat room, at least 2 arguments are required, the ID of the chat room, and the messages.
5. RDM: downloading the messages from either public channel or chat room since given time frame, the message type and time stamp are required for the input arguments.
6. OUT: logging out from the client and discontinue the connection with server.
7. UPD: uploading the files to other peer, the username for the recipient and file are required as arguments.

Design Trade-off

In the UDP section, the small file such as example1.mp4 can be transferred perfectly, however there was the situation of losing packets during transmission for larger file such as example2.mp4. After debugging, I found out the presenter was sending the packets too quick, and the audience cannot process these packets in the same speed, causing the issue of losing packets. Hence, I have set up the 0.0001 second of sleeping time on the presenter

side, to give extra time for receiver to process the packets. However, this will decrease the performance of UDP transmission, which is critical since UDP is focusing on the transmission speed. The potential improvements can be reduced the sleeping time from 0.0001 to a smaller level, so the performance can be improved. Similarly, I have set up 1 second waiting time on the audience side, to ensure the transmission has been finished. The 1 second waiting time can also be reduced dynamically if working on public network, and improve the performance correspondingly.