Project background

Program is run by python3.
Application consists of one server to handle multiple clients communicating concurrently. Functionalities mainly include one-to-one private messaging, group messaging and video transfer. All message-related functions are communicated using TCP and video transferring is communicated using UDP. All functions will be briefly explained in the report.

Program design

Server Side:
It is a TCP server. Server will start with following command.

```
python3 Server.py SERVER_PORT number_of_consecutive_failed_attempts
```

Server-side program designs to listen and accept client to build connection by receiving message from client and returning the responding message to client afterward in every turns. It handles multiple clients' connection by using multithread to create a new thread for every new successful connection. In the server side, all share files use lock function to lock the write process, preventing multiple users access the same file and modify it concurrently.

Client side:
Client-side program designs to communicate with server for login process at beginning. After authentication complete, program starts two threads, one for TCP message and one for UDP video. Main program asks input and determine which thread should be in use.

```
python3 TCPClient3.py SERVER_IP SERVER_PORT UDP_PORT
```

Message Commands

/msgto
This function is to send message from one user to another active user. To send message, user is expected to follow the command format, the first argument is identified as username and the remaining is the message content.
`/msgto` *`USERNAME MESSAGE_CONTENT`*
Server returns error and prompts to input command again if command violets any cases
Case1: Incorrect command format
Case2: USERNAME not active
After all, all messages send to the USERNAME and are saved in the messagelog.txt.

/activeuser
This function returns all the information of active user except the user execute the command from server to user.

Groupchat
/creategroup
User is expected to follow the command format to execute creategroup. Note that group name cannot consist of any space, otherwise first argument is treated as group name and remaining arguments are considered username.
`/creategroup groupname username1 username2 ..`
Server returns error and prompts to input command again if command violets any cases
Case1: Incorrect command format

Case2: usernameX is not active (X represents the number 1, 2 ..)
Case3: Exist group name
After all, group is created, server generates GROUPNAME_messagelog.txt for recording.

/joingroup
This function allows user to join a group that any users did add his or her name when creating a specific group. The format is expected as:
`/joingroup groupname`
Server returns error and prompts to input command again if command violets any cases
Case1: Incorrect command format
Case2: joining non-exist group
Case3: re-joining group already joined before
Case4: Joining exist group but username is not in the list when creating group
After all, user is joining the group and allows to send message in the group. Note that user does not join group will not receive any messages from the group.

/groupmsg
This function allows group member send message in the group after execute /joingroup command successfully. The format is expected as:
`/groupmsg groupname message`
Server returns error and prompts to input command again if command violets any cases
Case1: Incorrect command format
Case2: message to non-exist group
Case3: message to exist group but not group member
Case4: message to exist group but not yet joined group
After all, user send message to group. Message is saved in GROUPNAME_messagelog.txt and people who created the group or already joined group are able to receive message.

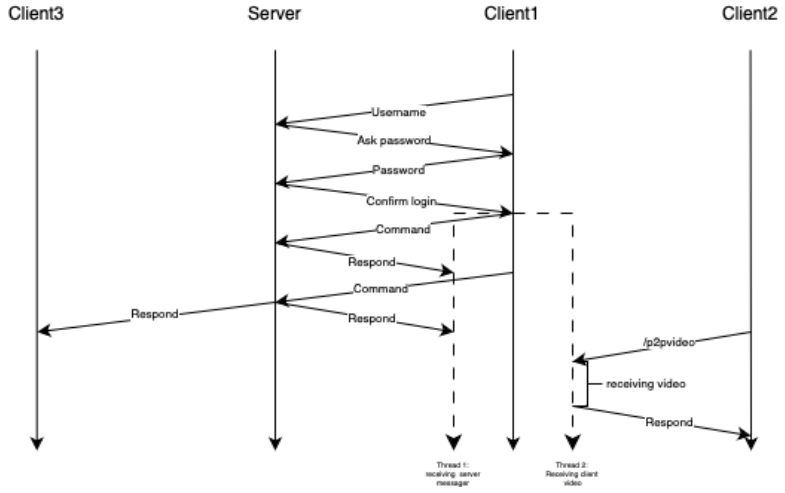
/Logout
After user execute this command, user is no longer active and will remove from both active user list that save in server and userlog.txt, sequence number of active user in userlog.txt remains in order.

/p2pvideo
This function transfers video file through UDP. User should execute /activeuser to get all active users UDP Port before executing /p2pvideo. If not, video file would not send to target receiver. User is expected to follow this command format.
`/p2pvideo username filename`
Server returns error and prompts to input command again if command violets any cases
Case1: Incorrect command format
Case2: username not exist
Case3: filename not exist

System work

After server starts and clients connects to server, server and client communicate as follow. Server and clients interchange message on username and password. Once login, client create two threads, one receives TCP message, communicating with server; another receives

video peer-to-peer. During TCP related communication, server sends response message to at least one client based on what command is received.



Design Tradeoff

In the application, it expects user to send video file through UDP, to prevent buffer overflow, program sleeps 0.1s to allow the processing time, so the tradeoff would be program takes longer time to complete the transfer.

Possible improvements and extensions

One possible improvement is file transfer in group channel. Program only supports peer-to-peer file transfer now. File transfer in group could be implemented in later stage. One way to implement the function is using TCP to send file to all active user in group.
Another improvement can allow user sends message to more than one user. Currently, program support user send message to another one. To make this improvement, program can change the command like this, to indicate how many users to send at single time.
`/msgto NUMBER_OF_USER` *`USERNAME1 USERNAME2 … MESSAGE_CONTENT`*