COMP3331 Report

Quynh Boi Phan - z5205690

Program Design

Client

The client utilizes two files:

- `client.py`: client entry file
- `client_manager.py`: helper functions for client.py

Server

The server utilizes three files:

- `server.py`: client entry file
- `server_manager.py`: helper functions for client.py
- `credentials.txt`: users credentials

Application Layer Message Format

Message is formatted as the following

- Header: encoded length of encoded content
- Data: encoded content

Overall the message is binded as:

```
message = f"{len(data):<{HEADER_SIZE}}" + data
```

For data to be read, the message needs to be decoded.

During the authentication process, the data to be sent to the server is given as:

```
data = f"{username},{password},{UDP_port}"
```

Where username and password are the client's credentials and UDP_port is the client's UDP port number. From here the data is binded with the header and encoded to send to the server.
When the data is received and decoded by the server, the server will send the login status back to the client. If an error occurred, such as the password is incorrect, a message will be given to the client about the error. Otherwise the user will successfully log in and is able to use the system.

After authentication, the data to be sent to the server is the command layout as given in the specifications. The data is then binded with the header and encoded to send to the server.
When the data is received and decoded by the server, the server will send the command status back to the client. If an error occurred, such as an invalid use of the command, a message will be given to the client about the error. Otherwise the command is executed successfully by the server.

Using the Program

Server terminal:
`python3.7 server.py [server_port] [number_of_consecutive_failed_attempts]`

Client terminal:
`python3.7 client.py [server_IP] [server_port] [udp_server_port]`

The server must be runned before the client.

On server starts, the server listens to every new connection and every incoming data requested by the client. If a new connection from a client is detected by the server, the client must either log in successfully or exit the program before another new connection can be processed by the server.
Upon login in successfully, multiple clients can execute the command normally and the server will process the data.

On client starts, the client establishes a TCP connection to the server and goes through the authentication process before being able to execute commands. Note that the UPD command does not work as it is not yet implemented.

Design Tradeoffs

Multithreading was not implemented in this program. Though the program still executes most commands required in the specifications, the UPD command could not be properly implemented. A couple of bugs were also found in the authentication process:

- Multiple clients can log in and use the program successfully, however the first client that runs client.py will have to log in successfully or disconnect for other client's data to be processed.
- If a client is already logged in and another client is attempting to log in, the other client has to log in successfully or disconnect before the first client's command is processed.

The message format being an encoded string made it simple for the server to read the client's messages, however the code can become messy. For example, to process the EDT command, the server has to join the timestamps together before it can compare if it is a valid timestamp. Furthermore, after ensuring that the command sent is valid, the server has to remove every part of the command except for the message itself so that it can save the client's message.

Improvements

The main improvement is to implement multithreading as that will eliminate the authentication bugs as mentioned above. This will also allow easier implementation of the UPD command.

Another improvement to be made is to remove the client from the online list if the client abruptly exits the server (using CTRL-C). In the current design, the server does not update this information, showing that the user is still online though they have exited the program.

In terms of code design, more functions can be defined to improve neatness of the code as it was found that many code was repeated and reduce the neatness of the code. Many if-else statements were used as well, making the code hard to read.

Extensions

An extension is the implementation of the UPD command.
Another extension that can be used for the program is the process of creating a new account, rather than reading credentials.txt for available credentials.

Reference

Socket programming:
https://pythonprogramming.net/server-chatroom-sockets-tutorial-python-3/
https://docs.python.org/3/howto/sockets.html

UDP implementation for file transfers:
Assignment multi-threaded code
https://wiki.python.org/moin/UdpCommunication#Using_UDP_for_e.g._File_Transfers
https://www.bogotobogo.com/python/python_network_programming_server_client_file_transfer.php
Note that the UDP websites were used in attempt to implement the UPD function