COMP3331 ASSIGNMENT REPORT

Program Design:

Language: Python3.7

Platform: CSE vlab environment

How does the system work?

Server.py

- Every socket has their own thread
- In every thread, there are two functions which are for user authentication and after login phases respectively. After login phase will be run after successful user authentication.
- A successful user authentication requires a correct password of a user that is not currently online. There are some helper functions in server.py that checks if a user is currently online and then if the password input matches the correct password.
- After-login phase receives the command from the client and have different functions to give the output depends on different commands. Then, will send the output that the client wants to the client.

Client.py

- Every client has two threads
- One is for receiving message from server and one is for receiving message from p2p.
- Same as server.py, client.py also consists of two phases in the main program which are user authentication and after login phase.
- Client.py counts how many times the user had unsuccessful attempts of inputting password and will inform server if there are 3 failed attempts.
- After successful user authentication, the program enters after login phase. There is a while loop which the users can keep inputting the command until they logout.
- In this while loop, the two threads of receiving messages from server and p2p will be started.

Trade-off

While implementing the inactivity timeout, I decided to get server to count the inactivity timeout rather than both server and client do this together. First, there is an extra step to determine whether the command input is valid or not. Because countdown will only be reset to the given time if the command is valid. Besides, the starting time of countdown time may be a bit different due to two programs running separately. For example, if client runs the countdown first, then close the socket and exit the program first. It is possible that the server comes up with error about sending a timeout message to this client. To better avoid the potential error, I decided to let server to do timeout functionality.

Feature Implementations

Commands supported by the client: message, broadcast, whoelse, whoelsesince, block, unblock, logout

Commands supported by p2p: startprivate, private, stopprivate

The functionalities of the above commands are specified in the spec.

Other functionality:

Timeout

- Automatically logout the user after a period of inactivity
- set timeout in server.py for every socket which timeout is given when server.py is initially run
- If server has not received any message from client for a period of given time, timeout will be triggered, inform the client about the timeout and logout the user in server.py
- Meanwhile, client receives message about timeout from server, then the client will automatically logout.

Data Structure Design

Most of the data are stored in list and dictionary.

Server.py

All the information of individual user will be stores in a dictionary and this dictionary will be store in a list called users.

A dictionary of a user is structured as below. Note that <type> represents the type of the value corresponds to each key:

User = {

‘username’: <str>,

‘unblocked_time’: <int>,

‘logged_in_time’: <int>,

‘last_active’: <int>,

‘is_online’: <bool>,

‘blacklist’: <list>,

‘socket’: <socket>,

‘address’: <socket_address>,

‘offline_message’: <list>,

‘p2p_port’: <int>

}

Besides, there is a list called list_socket to store the sockets that are connecting to the server's socket. The socket will be removed if the user who using this socket logs out.

Client.py

There are dictionaries for p2p_server and p2p_client. In each dictionary, they will have the key of socket and address.

Application Layer Message Format

In user authentication stage,

The server has its own helper function to check if the input username is a new user or not, if the username input is currently active and if the password input is valid or not.

Client counts the number of attempts of inputting invalid password. So, client sends 'blocked' to the server.

Server sends 'blocked' to client if the client is currently blocked by server due to 3 unsuccessful attempts of inputting password. So the user is not able to go further to input password.

Server sends 'Using' to client if the user is currently used by other clients. So the user is not able to go further to input password and will be asked to input another username.

Server sends 'Confirmed' to client if the client inputs correct password of a user that is currently not used by other clients. So the user can now move to after login phase.

In after login phase,

The client sends the command to the server and then the server sends the output to the client, or the other online users depends on different functionalities.

- Server sends 'timeout_logout' to the client If server has not received any message from client for a period of given time. So the client will logout user, close all the sockets and exit the program.