Name: Minh Khai Tran

zID: z5168080

Comp3331 Assignment report

Language: Python3

Features implemented:

- Successful log in and log out for single client and multiple clients.
- Blocking user for specified interval after 3 unsuccessful attempts (even from different IP).
- Successful log in for multiple clients.
- Implementation of Download_tempID.
- Implementation of Contact log checking.
- Peer to peer communications including removing beacons older than 3 minutes.

Problems:

- In the authentication part, when a client logs in, it will block other clients log in if these clients come after the client until the client finishing log in or disconnect from the server.

Data structure:

- Mainly dictionary because it takes $O(1)$ to access the key-value pair.

Application Layer Protocol:

- Every message is sent from server to clients or clients to server will follow the same format.
    - Packet includes header and follow by data.
        - Header is a number to indicate the length of the data.
        - Data is a byte stream.
- Every message is sent from client to client (p2p protocol):
    - Packet is mainly data part (a byte stream)
- The client needs to login with correct password and username in the credentials.txt file.
- After successful login:
    - The client can communication with the server.
    - The client can communication with other online clients using UDP protocol. (after receiving temporary ID from the server)

How my program works:

- Turn on the server: $python3\ server.py < port >< block\ duration >$
- The run one or many clients: $python3\ client.py < serverIP >< serverPort >< UDP\ client\ port >$
- Example:
    - If client 1 runs the command before the client 2 runs the command:
        - The client 2 must wait the client 1 finishes the authentication (i.e multiple attempts, block, or successful log in) then the client 2 can log in
        - After the 2 clients login, those clients can freely use any other available commands without interrupting.
        - The clients can enter multiple enters without disconnecting from server.
    - To use Beacon command, the client has to command Download_tempID to get the tempID to its local machine first.
        - When the client user Beacon <destIP> <desPort>, the destination client will immediately see the beacon message including tempID(20 bytes), starting_time(19 bytes), expiry_time(19 bytes), BlueTrace protocol version(1 byte) (the BlueTrace protocol is not displayed on the client's terminal).
        - The destination client will check whether it is valid or not:
            - If it is valid, the destination will add this beacon into its log file (log file will be created if it hasn't existed yet in format of <zID>_contactlog.txt)

    - When the client has a log file:
        - It can use command Upload_contact_log: the client can send the message including tempID(20 bytes), starting_time(19 bytes), expiry_time(19 bytes).
        - After the server receives, it immediately displays the received contact log and contact log checking
    - The client can use logout command any time it would like to use, and the process will ends immediately.
    - If the client enters a random command, it will display "Invalid command" prompt on the client's terminal.
    - If the client enters many enter, it does nothing.
- I used three threads in client.py:
    - First one is for the communication between the client and the server (authentication, server-client commands)
    - Second one is used to listen to any messages from other clients (UDP protocol)
    - Third one is used to constantly check the time duration of each log in the log file.
        - If a beacon is over 3 minutes, it will remove this log out of the log file.

References:

- Socket Chat application from YouTube.
    - https://www.youtube.com/watch?v=CV7_stUWvBQ
- Sample code from the course:
    - https://webcms3.cse.unsw.edu.au/COMP3331/20T2/resources/45154