COMP3331 Assignment Report: BlueTrace

Haozhe Chen z5142012

1. Login authentication

In order to keep track of assignment specification, I divided login part into username authentication and password authentication. When server starts, it will save all accounts' login (from credentials.txt) info into a HashMap called Credentials.

a. Username

If user inputs a existed username (existed in credentials.txt), the program will automatically jump into next authentication section which is password section. At the meantime, to prevent some users keeping trying password, user will not able to pass username authentication with same username on the other devices. I used a HashMap called alreadyLogin in Server.java to track whether certain account has already login somewhere. Once receiving different username, server will return a different message correspond to different scenario respectively which are

"username_validation: userID existed"
"username_validation: already login"
"username_validation: user block".

b. password

If user input a matched password with username, client will receive a TCP message

"password_validation: password collect" ,

then the login authentication will finished. If the password is mismatched, the server will send a integer to client which indicates the available attempt time for current user. The available attempt time also stored in a HashMap called loginAttempt in Server.java, so that both client and server can keep track of certain user's available attempt time in this way.

c. block

User will be blocked once he/she has already tried 3 times. However, there are several scenarios which need to be considered about.

i. <u>User was blocked in window1, try again in window2 (could with different IP) without closing window1.</u>

Like 1.a.Username said, user will not pass the username authentication in other window once he/she passed the username authentication in certain window without terminated or logout.

ii. <u>User was blocked in window1, try again in window2 (could with different IP) with closing window1 (ctrl + c).</u>

Since using ctrl + c to terminate program before login successfully will be treated as unusual logout, the HashMap of loginAttempt and alreadyLogin in Server.java will not be reset. Server can track this even when user logins in with another different IP so user will be blocked again in the new window.

iii. <u>User was not blocked in window1, try again in another window (could be different IP) with closing window1 (ctrl-c).</u>

As part ii said, Server.java can track user's login status and available attempt times, so even user try to login in another window, the attempt timer will not be reset. For example, userA login failed one time in window1, close window1, open window2, he/she will has only two available attempt times inherits from window1.

2. **tempID**

   Considerate of tempID’ s uniqueness, tempID will be generated by

   3 bytes random int + 13 bytes timestamp (time of generation) + 4 bytes random int.

   a. Download_tempID

   After client receiving Download_tempID command, client will send a message “command: Download_tempID” to server, after that, server will generate a new tempID for this user and send back to client.

3. **P2P communication**

   In this assignment, peer will communicate to another peer through UDP. So each client device will run a UDPRcv thread to keep listening UDP packet from other client device.

   a. Beacon

   The content of beacon message is:

   tempID,start_time,expiry_time,version (set as 1 in my program)

   Since client need to know the tempID information for current user, it will send a TCP message to server in order to get all tempID information which need in beacon message. Once receiving beacon message from other device, client will verify the tempID with current time to make sure tempID is not expired. After that, client will save beacon message to z5142012_contactlog.txt.

   b. *Auto deleting expired beacon record*

   Each client device will run a thread which design to delete expired beacon message stored in z5142012_contactlog.txt. To make program easier to verify whether certain beacon message is expired, Client.java will save every message’ s received time at the tail of each message. So thread can just compare each message’ s received time with current time.

4. **Upload_contact_log**

   In this part, client will send the record stored in z5142012_contactlog.txt to server. Server then compare the record’ s tempID and userID with information in tempIDs.txt to obtain the corresponding userID of tempID in contact log record.

5. **logout**

   User can logout of the service by either pressing ctrl + C or logout command. Since ctrl + C will disturb the threads which run in backend, so this logout method will trigger an exception but still works.

   a. Client.java terminate

   Since there are several threads run in backend of Client.java so we need to stop different threads with different methods. With tempIDUpdate and contactLog thread, they will be controlled by a ThreadFlag of Client class. With UDPRcv thread, since it will get stuck at DatagramSocket.receive, so when it need to be terminate, it will receive a UDP message contain “stop” to tell this thread best time to stop.

b. Server.java terminate

There is only one thread in Server.java. Since in TCP connection, one client will correspond to one socket on server side, so when client logout, server will receive a TCP message contain “stop” message, at that time server will break the loop inside of thread to terminate socket thread. At last, server will update the account’ s login HashMap as logout status.

6. Connection Establishment and Communication

Client will establish TCP connection with server though ServerSocket.accept(), after connect establishment, they can communicate with DataStream which are:

DataInputStream(Socket.getInputStream())
DataOutputStream(Socket.getOutputStream())

There is no need to establish UDP connection between peer and peer. So they just send a DatagramPacket through DatagramSocket to another peer to communicate with each other.

7. What can be improve?

From my perspective, I still can have further understanding of multiple thread in Java in the future. So that I will be able to run more threads to improve efficient of both client and server side.

8. Extension

a. The first extension will be the registration part. Server should make sure there is no duplicate accounts by checking tempIDs.txt. This function should be easy to implement although this assignment has no requirement about it.
b. The second extension can be the auto calling or auto email when certain account is confirmed as potential infected person, this function will need to register a phone number or email server for BlueTrace. After the server check the beacon record uploaded by certain client, it should send a TCP message to client which are mentioned in record.
c. The third extension is client program should be able to update user’ s tempID automatically in real life. Actually this function has been implemented in my code, but I still deprecated this part of code at the tail of Client.java after confirmed with lecturer. However, this function should be implemented by mutli-thread in real BlueTrace.