Assignment 2: Chat Application

Rushil Gupta
2017CS10487

Shreya Sharma
2017CS50493

September 2019

1 Chat Application: Potential Protocol Extensions

1.1 Step 1: Protocol Explanation

1. Initially when the script is run, a popup appears which asks the user to enter the mode in which he wants the application to run. The possible modes of communication are unencrypted, encrypted or encrypted with message signatures. Based on the selection of the user, the relevant client and server files are compiled and executed.
2. Initially, a message pops on the client screen to enter the username followed by a popup to enter the server address.
3. After entering the above fields, the client application establishes two TCP Connections i.e. two sockets to the server port 6789 via the formal registration messages.
4. For the latter two modes, the public key of the client is sent to the server for storing in a global database in an encoded form within the registration request for receiving socket.
5. On receiving registration requests, the server checks the validity of the username. If it is found to be valid, then a success message is sent on the input streams of both the sockets otherwise an error is returned which is flashed on client screen as well.
6. If error is flashed, then you have an option to enter either Y,N or E.
   - Y means the client continues into the application with wrong username and no functionality.
   - N implies that client will enter the username again and try to register.
   - E implies that the user exits from the application.
7. Once the registration processes are over, the client application then waits for the client to enter a message to send on the screen which is specified in the format @USERNAME MESSAGE
8. The Client Application encapsulates the message in a packet and the recipient information in the packet header. For the encrypted and encrypted with signatures mode, the message is properly encrypted before sending via the public key of the recipient. After the message is sent to the server, the client waits for a successful delivery ACK from the server.
9. The server modifies the packet header before forwarding the message to the concerned recipient.
10. The message is received by the receiving socket of the concerned client. If the recipient is not registered on network, then an error message is returned to the sender.
11. On receiving the message, the receiving socket, in case of unencrypted messages checks only if the content length specified in the header matches the length of the message received. However, for the case of encrypted messages, it first decrypts the data using its own private key and then compares the length of the message to the length specified in the header. For message signatures, a second sanity check is applied which confirms if the digested encrypted message is same as the decrypted hash.
12. If all the sanity checks are passed, the successful receiving ACK is sent to the server and the server then sends a successful delivery ACK to the sender of the message. The message is also displayed on the receiving client's screen.

1.2 Extensions :

- **DEREGISTER:** If a user enters "DEREGISTER" as input, then he/she is deregistered from the network and the client application is exited. The entries corresponding to that user are removed from the global storages at the server also.

- **User exit via Ctrl+ C:** To handle this situation, we can use the concept of ping and timeouts. The server can issue pings in a round robin manner to all the clients registered with it and wait for a response from the user for a time duration x. If time to receive ACK is more than x, then the user is supposed to be deregistered from the network and all the corresponding entries are removed from the global storages at the server. This way we can ensure that all the clients present in storage lists of server are well connected to the server.

- **Offline Users:** To deal with the offline users, we can use an approach very similar to the one that occurs in email. We can maintain a message box for each client as a storage for messages which should have been forwarded to the corresponding client if he/she was online.
  On Client side, every time the client comes into the network and registers with the server, the receiver socket first pulls all the messages stored in the message box and sends the second ACK to the sending client corresponding to those messages. The first ACK is sent by the message box to the server which then sends it to relevant client. If the user was online at the time of message being sent to him, then the client application sends a combined or equivalent ACK to the sender. This way we can deal with this situation in a simple way.