Computer Network Project2—CNLine

B00902114 李沛庭　　B00902123 楊筑媛

Protocol Specification

- System built on top of TCP sockets, with a single server and multiple clients, one for each online user.
- create a socket list (CONNECTION_LIST) to keep track of the sockets connecting to the server. Initially, only the listening socket from the server is in the list, then, using select, probe for any sockets ready to be read (has incoming data/requests). If the socket ready to be read is the listening socket, a new socket connection is required, the server accepts the connection request and adds that socket to the list.

User & Operator Guide

- For first-time users, type in new and press enter. Then type in desired username and password to register. You will then be logged in automatically. For those that have already signed up, simply type in username(enter), and password(enter).
- To pass a message to another user, you should type in their username first, a space, and then the message.
- A message from another user will be in the form [sender]: [message]
- To check past conversations, type in “Query”.
- If a message is directed to an offline user, it will be sent to the user once they log on, and is connected to the system.
- To logout, type in “LOGOUT”.

System & Program Design

- For login and registration, an “info.txt” file is kept at the server side, keeping track of all registered users and their passwords. (separated by a blank space) We either open the file to append for registration, or we search through the file looking for the username that desires to login, and check the password.
- When a user correctly signs in (or successfully registrates), the client will request a TCP connection with the server, using a free port on the client side. Once the connection is accepted by the server, the server appends the new connection socket into a socket list of online users.
- For a message between two users, the server is responsible for receiving the message from the sending side’s socket, and directing the message to the socket for the receiving side.

    - This is accomplished by linking a socket with a corresponding index when the connection is first accepted by the server. This allows a socket to be associated with a username.
- A history of a client's previous conversations is kept as a file named as the username, kept at the client side. Any incoming and outgoing messages are appended into the username.txt file. When the user types in "Query", print out the file.
- For offline messages: When an online user sends a message, and the message arrives at the server, the server checks the first token in the message, which should be the receiver of the message. It searches through the "log.txt" which keeps track of who is currently online and which index to use to access its socket. When we do not find the receiver in the list, we open a "offline.txt", which (in order) we write in the sender, receiver, and the message itself.
    - When a client first logs into the system, the server checks into the "offline.txt", and searches through it with the client's username, seeing if there are any offline messages for him. If there is, we send the line of message through the socket to the client as a regular message.
- Data encryption is implemented into the system. A message between two users are encrypted at the client side, sent to the server, decrypted and directed to the right socket, encrypted again, sent to the receiving user, and finally decrypted so it can be read by the user. In fact, any message flow in the system is encrypted before sent, and decrypted at its destination.
- When a user desires to logout, it sends a message to the server saying "LOGOUT". The server then sets out to delete the user from "log.txt", the "online-users-list", and adjust the socket indexes, since the tcp connection between the server and the user's client is going to be teared down, which means the socket will be gone as well.