Project -3

Instant Messaging System

by
Amruta Chavan
Anjali Pachpute

Under Guidance of
Prof. Hosking

- **Message Format**

  The message format used for communication between client and server is as follows

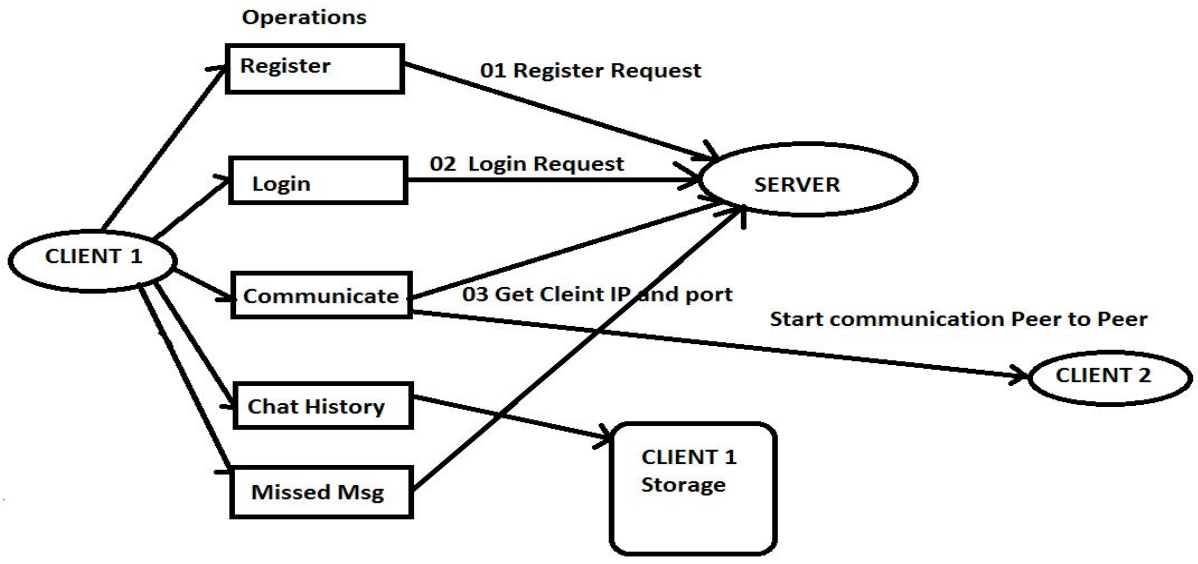
msgId	This field is used to store the message Id. Each message has a message ID which will help the server or client to identify the type of message received.
userName	userName field is used to store the client's userName
passWord	passWord field is used to store the client's password
IP	IP field is used to store the client's IP address
port	port is used to store the client's port
ErrMsg	This field is basically a string field which is used to store the messages that are to be exchanged between the client and the server.

- **Protocol Used for Communication:**

In order to provide reliability we have used ***TCP*** for communication between client and server as well as for communication between two clients. Multithreading is used so that multiple clients can communicate with the server and with each other.We have used client server approach as well as peer-to-peer approach for communication.

- **Architecture:**

The basic architecture of application is as follows



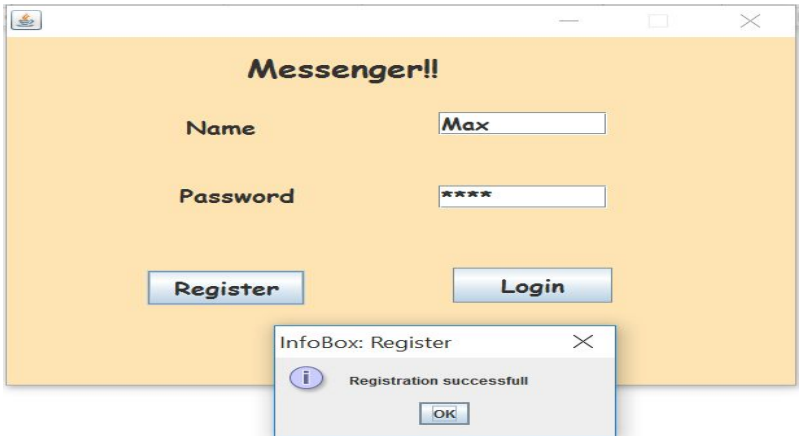
- **The basic flow of our application is as follows:**
  1. Client has to register with the server. Server uses a hashmap *Users* to store the username and password of each client. Server also maintains a hashmap *hmIP* to store the IP for each username.
  2. If the client is already registered with the server then he should login with his username and password. The server verifies the client's username and password against the value in the hashmap *Users*. The server also updates the user's IP in the *hmIP* hashmap because the same user can login from different systems each time.
  3. If the *client1* wants to communicate with *client2,* then *client1* first sends this request to the server. The server replies with the IP and the port for *client2. Client1* then directly pings c*lient2* with its message.
  4. Each conversation between two clients is stored is stored in a file at each client. For example if client ***a*** and client ***b*** are communicating then the chat will be stored in a file named ***a_b.txt*** at a's end and at b's end it will be stored in a file named ***b_a.txt.*** So a can view the chat it had with b using the a_b.txt.
  5. If client1 wants to communicate with client2 and if client2 is not online then the messages sent by client1 to client2 are buffered at the server in a file name ***Failure-client1_client2.txt***.

- **The detailed flow for each operation is as follows:**

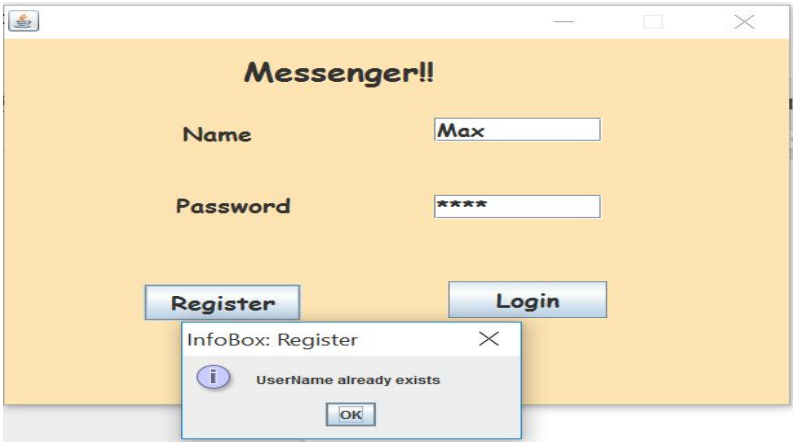
1. Client Register:

Client sends a request message with msgId=01, username, password, IP and port. If the registration is successful the server replies with msgId=010 and a registration successful message. If the registration is not successful the server replies with msgId=011 and registration not successful message.

Below is the snapshot that shows successful client registration:



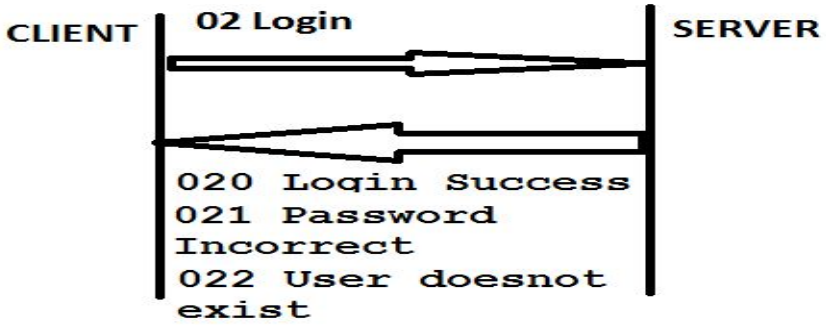
If the username already exists then the user is prompted with an error message



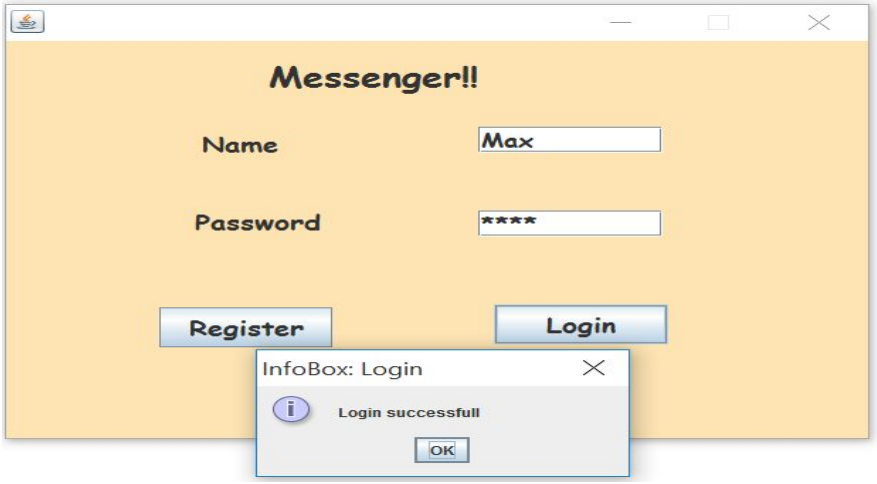
2. Client Login:

Client sends a request message with msgId=02, username, password,its current IP address and port. If the login is successful the server replies with msgId=020 and a login successful message. If the login is not successful the server replies with msgId=022 and login not successful message.

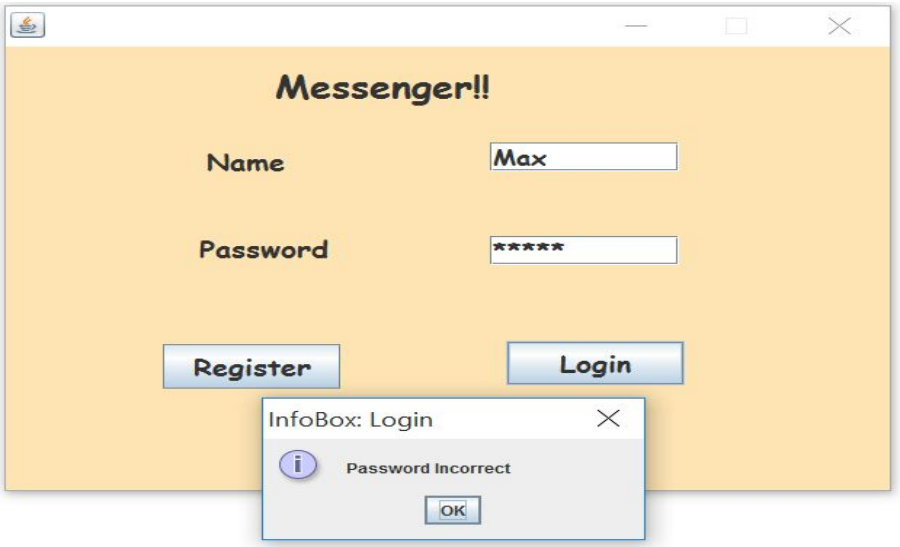
If the login is successful then the server will update that client's IP address and port in the *hmIP* hashmap because the same client can login from different systems.



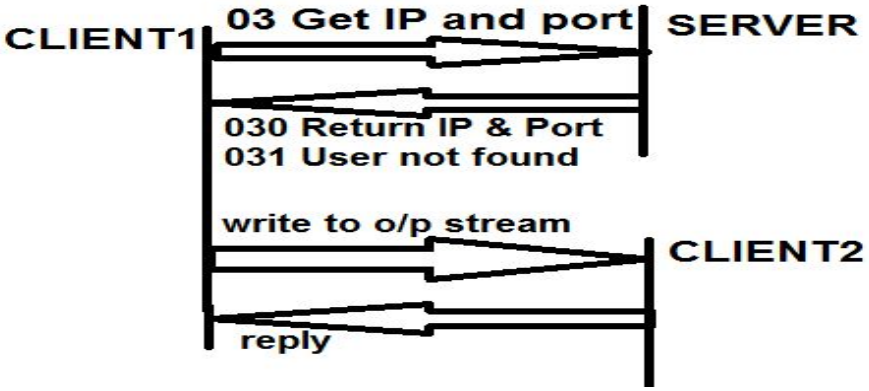
Below is the snapshot that shows successful client login:



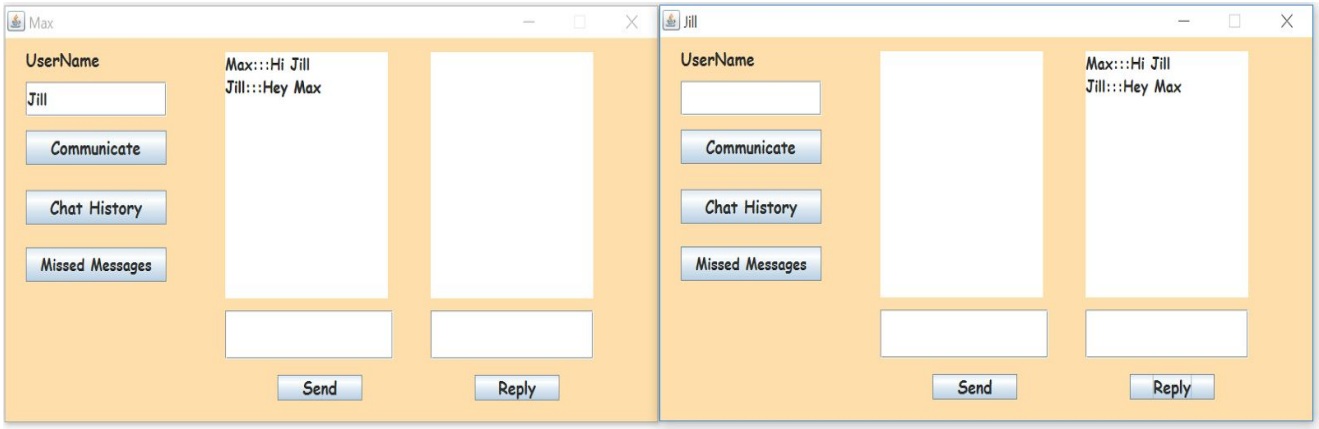
If the password is incorrect then the user is prompted with an error message



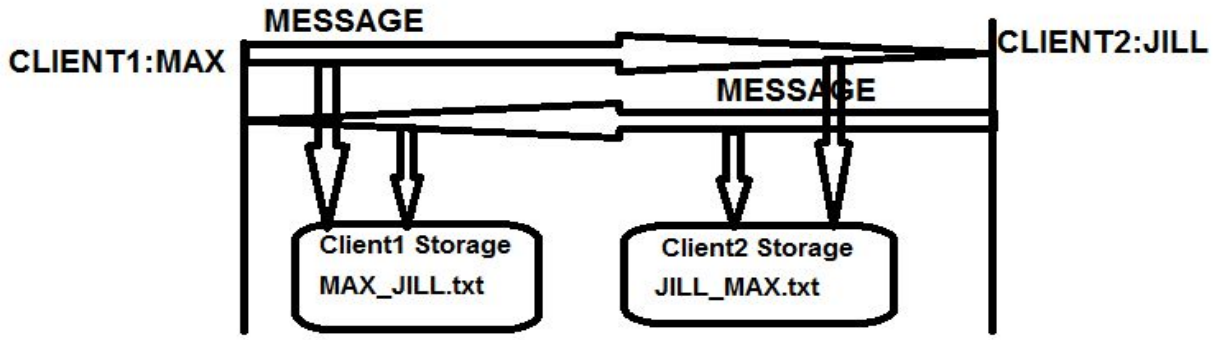
3.Communicate: If a client wants to communicate with other client then, the user first contacts server to get the IP address and port of the second client.The server replies with the IP and port of the second client.The client can then start a communication in a peer to peer mode.



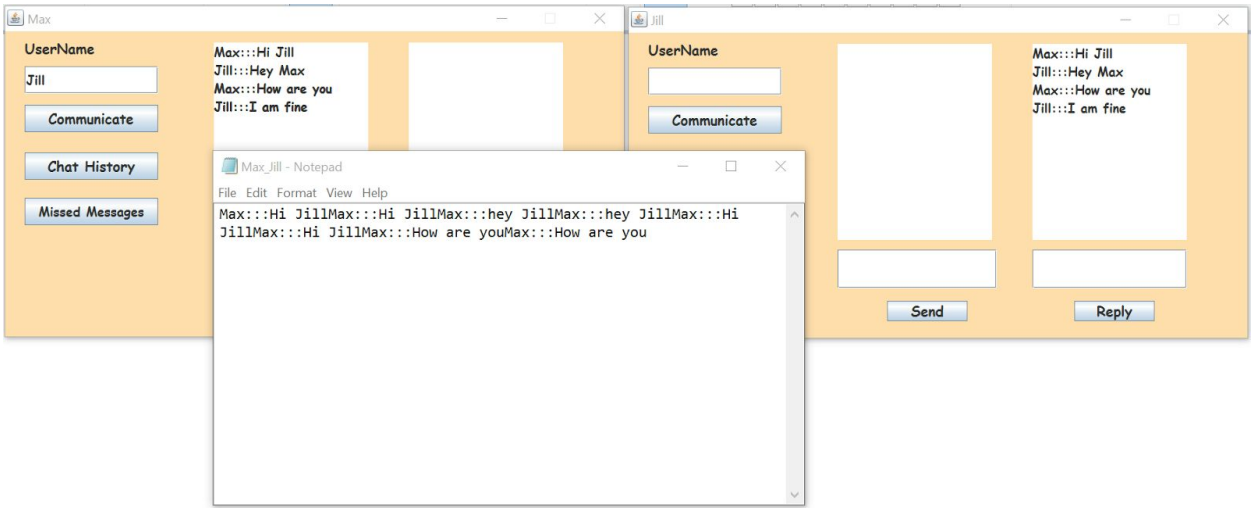
Below snapshot shows the communication between two clients



4. Chat History: Client has an option to view chat history. When the client clicks the Chat History button the client's chat history with the user in username text box is opened. For example the chat history between Max and Jill is stored in a file named Max_Jill.txt at Max end and at Jill's end it is stored in a file named Jill_Max.txt. This text file i.e chat history is displayed for the username present in the text box.

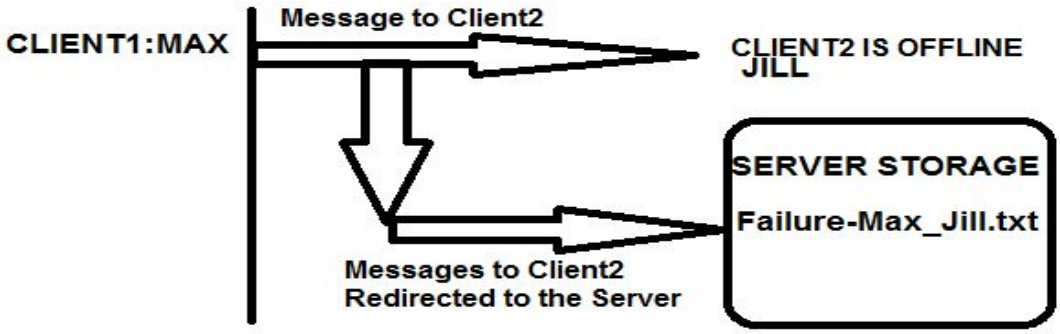


The snapshot below shows the result when Max wants to see his chat record with Jill



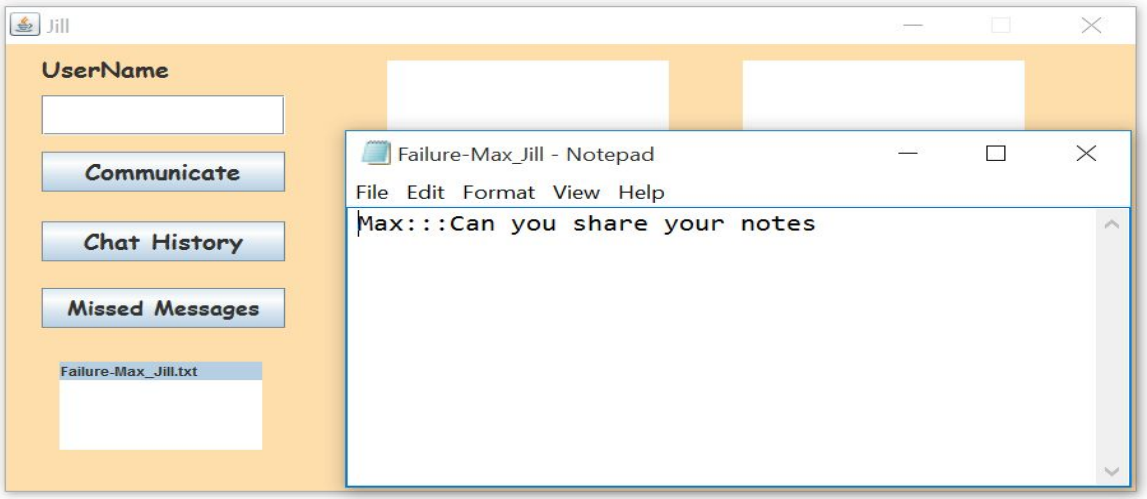
5. Missed Messages:

This feature is to handle failures. The messages for a client who is not online are buffered at the server. The client can then get those messages when he comes online. For ex. if Max texts Jill but Jill is not available then the messages are directed to the server and saved in a file named *Failure-Max_Jill.txt*. Jill can get these messages from the server on the clicking the Missed Messages button.



Below is the snapshot of the message that Max is displayed when Jill is not online or if there is a failure at Jill's end. The messages are forwarded to the server with **msgId 05**, Jill's username and the message.

When Jill comes online she can view all the messages she missed by clicking the Missed Messages button. The list of all the missed messages files will be displayed in a list. The file will be opened when the filename is double clicked.



- **Steps to run the application:**
  1. Run the Server.java
  2. Run the ClientApp.java on the client system. Give the username and password.
  3. To Communicate, input the username of the other client in the username textbox and click Communicate button. Now write the message in the first textbox and click send.
  4. To reply to a message, write the message in the textbox above reply button and clickreply.
  5. To view chat history, input the second client name in the username box and click the Chat History button.

6. To view Missed messages, click the Missed Messages button. A list of missed messages file will be displayed. Double click the filename you want to view.

- **Difficulties and challenges faced:**
    - Instant Messaging has a lot of features in any commercial application. Deciding the features that we wanted to include in our application was a task.
    - Also initially we were planning on not having a GUI but as the application was multithreaded it was difficult to display all the messages on a single console.

- **Approach Taken to Solve the Problems:**
    - To handle the multithreading and console messages display problem we implemented a simple GUI using Java Swings.
    - For features we decided to implement chatting, message history, failure handling.

- **Conclusion:** We successfully implemented an instant messaging system with additional features including failure handling. This application can be used by multiple users at the same time as the application is multithreaded.