DESKTOP INSTANT MESSENGER

For Private Network-LAN

Course Name: Advanced Networking Project Report

Student Name: HAQ IJAZ UL

Student ID: 2820150066

Table of Contents

List of Figures

1 Introduction

IM-Instant Messenger is a real time online chat application that offers transfer of messages between machines over internet or local LAN. Unlike email, where one sends a mail to the recipient and waits for the reply till the recipient is online, in Instant messenger you can get a reply instantly when recipient is online. The basic working principals are sockets and stream. Network sockets and streams enable machines to communication one another over network. A Push Technology is used to achieve real-time transfer of messages character by character. Some advance chat application involves file sharing, hyperlinks. VoIP and video chat.

1.1 Purpose

In our project I are intended to make a desktop based chat application for a private network. This application will be environment dependent. I will be doing socket programming in JAVA. In socket programming comprises of two kinds of processes. One Server process and other Client process both running on different machines. Client process sends connection request to the Server. Server process defines sockets and ports on which it can listen to client request. When client sends request to Server, server process binds to client process via socket and initiates listening to the port defined by Server process. This application will be connection oriented. For this TCP-transfer control protocol will be used. Users can connect to other via there user name or machine IP. Server will be given a unique IP. This application will provide a group chat too where multiple can communicate.

1.2 Project Scope

Instant messenger is the platform used to manage user contact list, groups, save profile, hold conversation and also perform data and file transfer to and from the contacts all over the world. Moreover users can also do video chat, voice chat, screen sharing, free texting, add emotions and set privacy and can view notifications. It also maintains previous call and message history for later use.

2 High Level System Architecture

2.1 System Flow Chart

Before going into High Level System Architecture, let us understand system flow.

Instant Messenger starts with the setting the Server up. As Server is the main of the system it has to be in running state all the time. Server then starts session with creating a SOCKET for its Client to communicate. Then it defines PORT for that socket on which Server can listen to its Clients. Server then WAITS for its Clients to request a connection. Server waits till there is no client. When a Client sends a connection request to a Server, Server BINDS the Client process. Server accepts client request for defined SCOKET and PORT. When the connection is accepted Client and Server can now transfer messages with each other. With every send and receive message Server checks the connection. With the communication is done Server and Client kill their sessions and destroy their SOCKET. The Flow chart is shown in figure- 1;

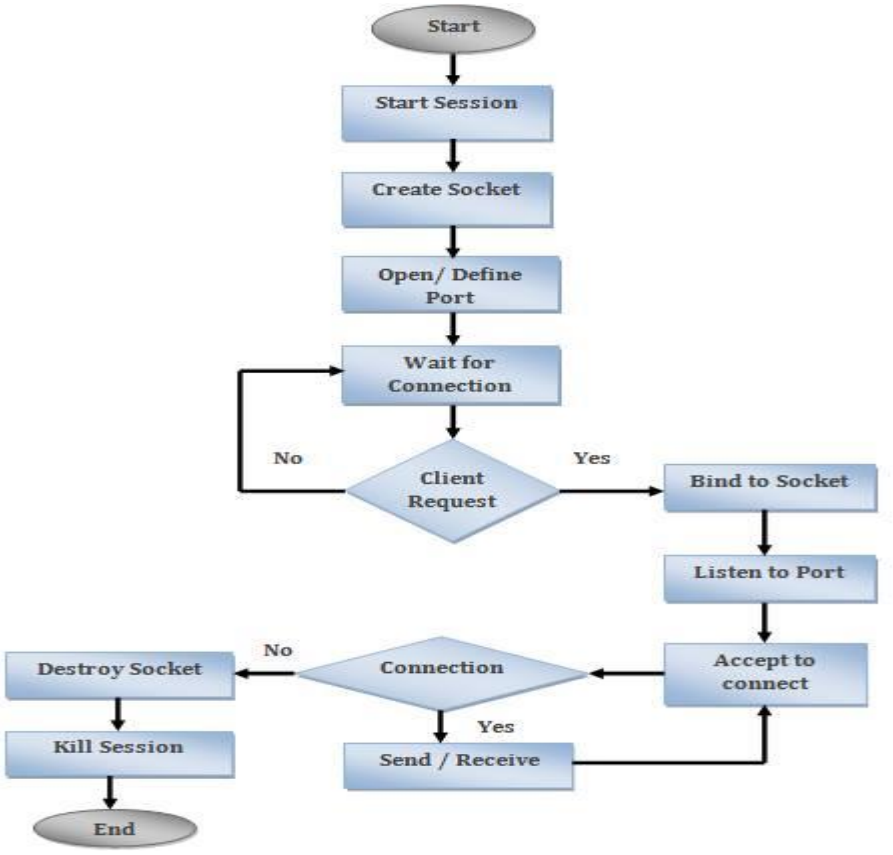

Figure-1: IM System Flow Chart.

2.2 Class Diagram

Class diagram is used to define system structure. It shows classes and objects defined in system, their attributes, methods of the classes and relationship between them. Attributes defines the data types and methods describes the behavior of the classes. As for efficient system design classes should be loosely couple and highly cohesive, these properties are show in Class diagram via class relationship.

The possible classes for desktop based Instant Chat Messenger will be:

1. Session.
2. Instant Messenger.
3. Server_chat_window.
4. Client_chat_window.
5. Server_Process.
6. Socket.
7. Port.
8. Client_Process.
9. Message.
10. Connection_info.
11. IOException.
12. Request.

Class Diagram for IM is shown in figure-2:

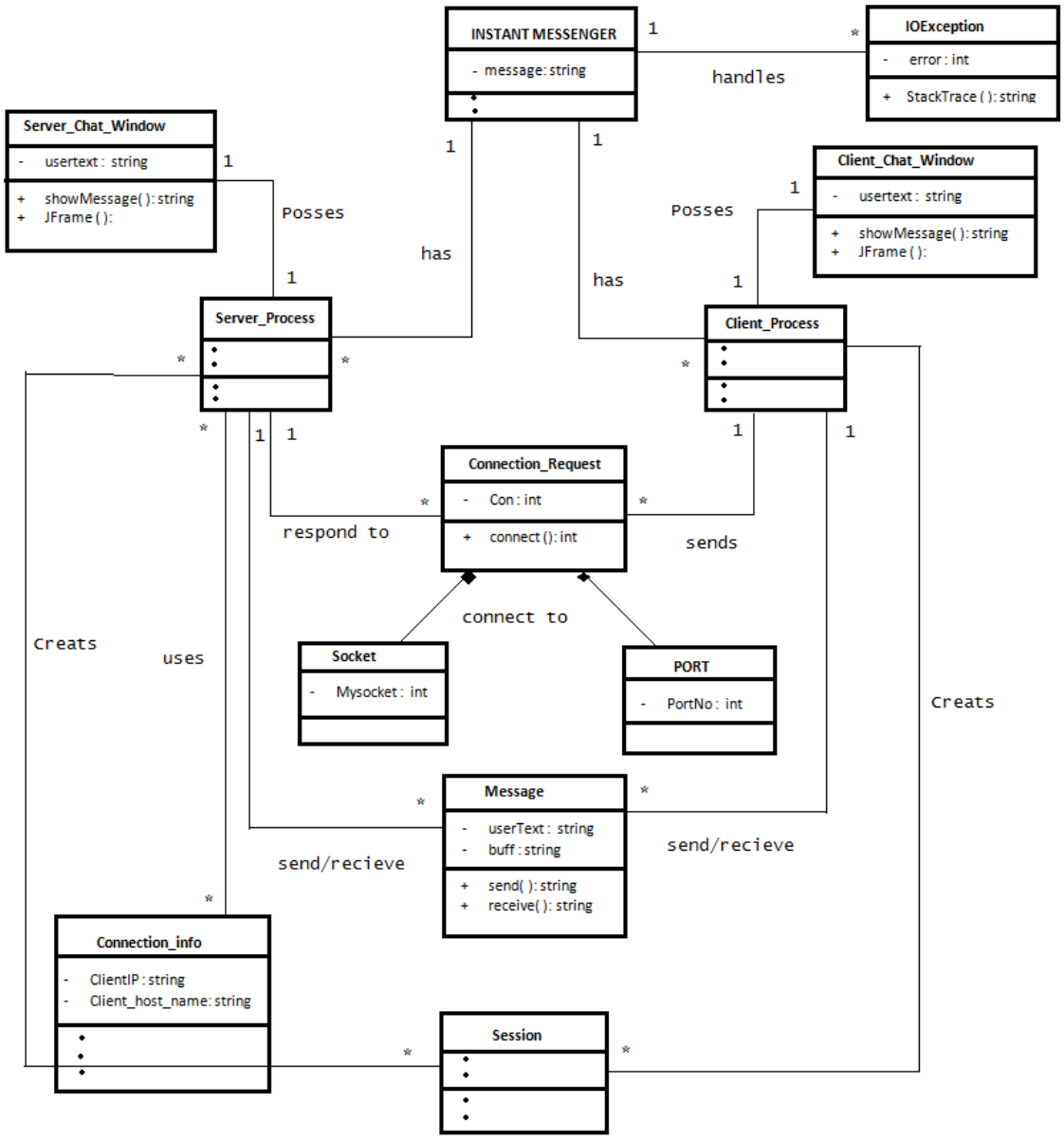

Figure-2: Class Diagram for IM Application.

2.2.1 Server Class

Sever and Client classes are the main classes which defines the system. Following are the attributes and methods of the server class.

Attributes:

- Message: is a of type string, contain the text from Server to client.
- Buffer: of type string, contain text from client.
- Mysocket: of type integer, containing socket descriptor.
- Sock_addr_in: a structure, containing IP version and server address.
- portNo: of type integer that has port for listening.

Methods:

- socket( ): defines port to socket.
- Bind( ): binds client to defined socket.
- Listen( ): start listening to the port on client want to connect.
- Accept( ): accepts clients request.
- Send( ): send messages.
- Receive( ): receive client messages.
- Close_connect( ): closes connection when communication is done.
- Close_socket( ): destroy socket.

Server Class is shown in figure-3 :

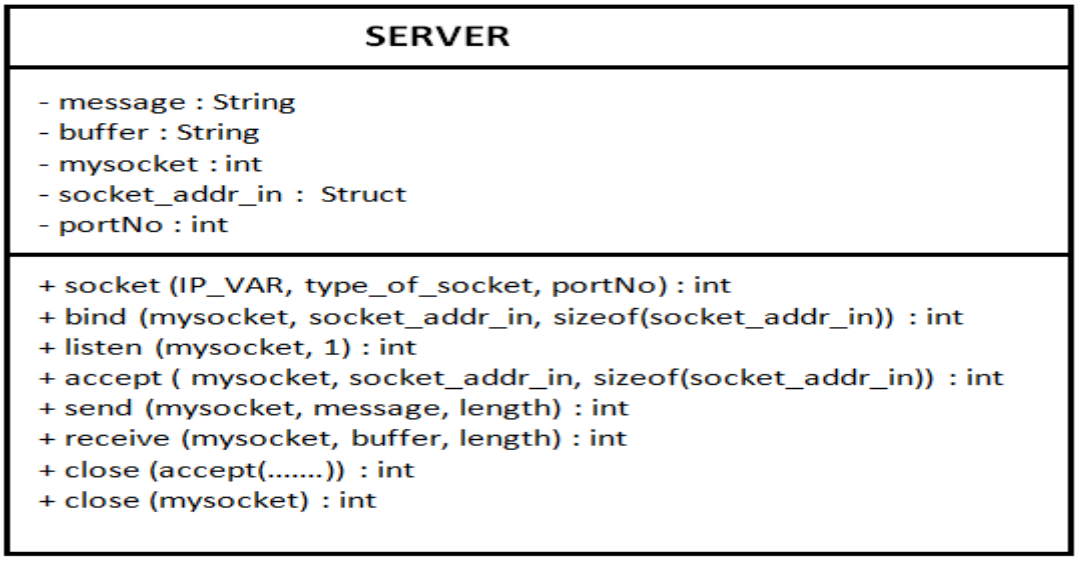

Figure-3: Server Class.

2.2.2 Client Class

Client class is almost same as Server Class but with slightly different functionality:

Attributes:

- Message: is a of type string, contain the text from client to server.
- Buffer: of type string, contain text from server.
- Mysocket: of type integer, containing socket descriptor.
- Sock_addr_in: a structure, containing IP version and client address.
- portNo: of type integer that has port for listening.

Methods:

- socket( ): defines port to socket.
- connect( ): connects to the server via its IP.
- Send( ): send messages.
- Receive( ): receive messages.
- Close_socket( ): destroy socket.

Client Class is shown in figure-4:

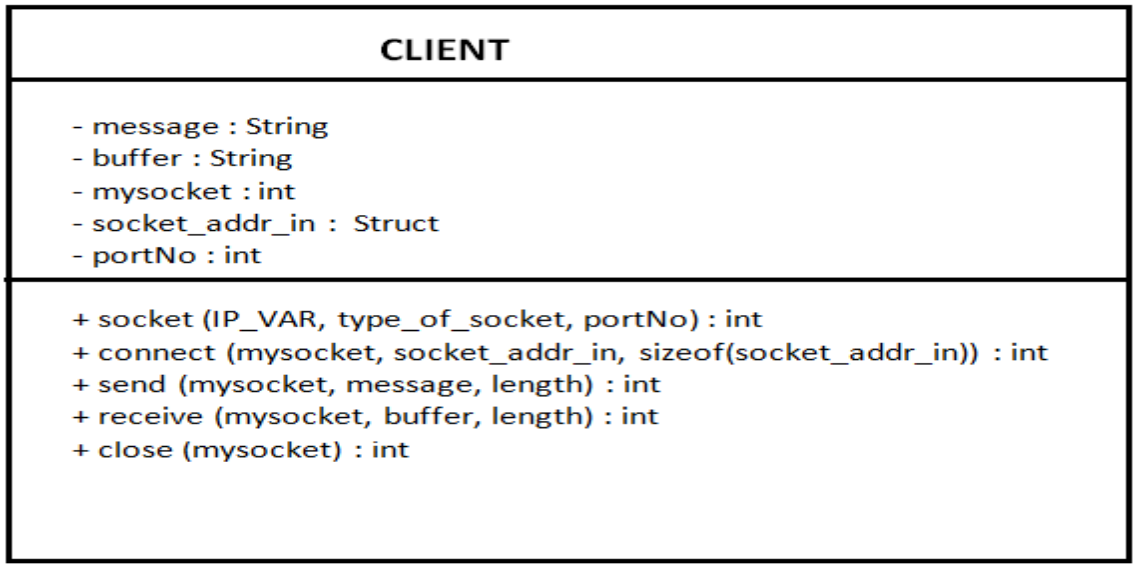

Figure-4: Client Class.

2.3 State Diagram

Figure-5 show the State Diagram. It describes the System Sates at different levels. Initially system starts up with setting up Server, defined as Start Session state. Then it defines socket and port, this state is called Creating Socket. Then comes the Wait state where it waits for connection form the client. When client send a request system goes to Bind for binding socket/port and then to Accept state which defines connection acceptation. Then it moves to Send/Receive state where actual communication occurs. When communication is done, system moves to Destroy Socket state for clearing all the socket and port.

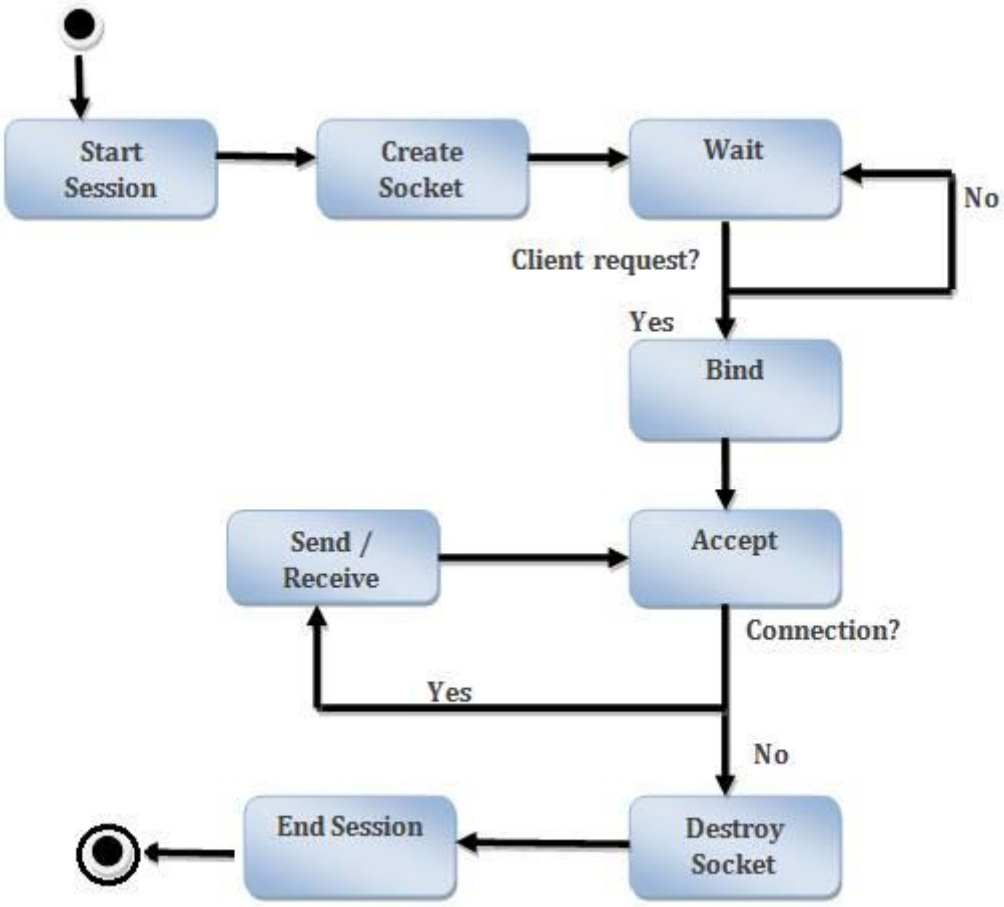

Figure-5: State Diagram of IM Application.

2.4 Sequence Diagram

A diagram which shows how processes interact with each other in order to complete a task is called Sequence diagram. Its built on Message sequence chart. Sequence diagram shows objects and their interactions mapped on time. It shows how objects and classes interact in a certain task by exchanging messages to fulfill the functionality that is to be achieved by that particular task. These diagrams characteristically are related to use case comprehensions in the rational view of the system that is being developed. They can also be called event diagrams or event scenarios.

Sequence Diagram of IM application is shown in Figure-6.

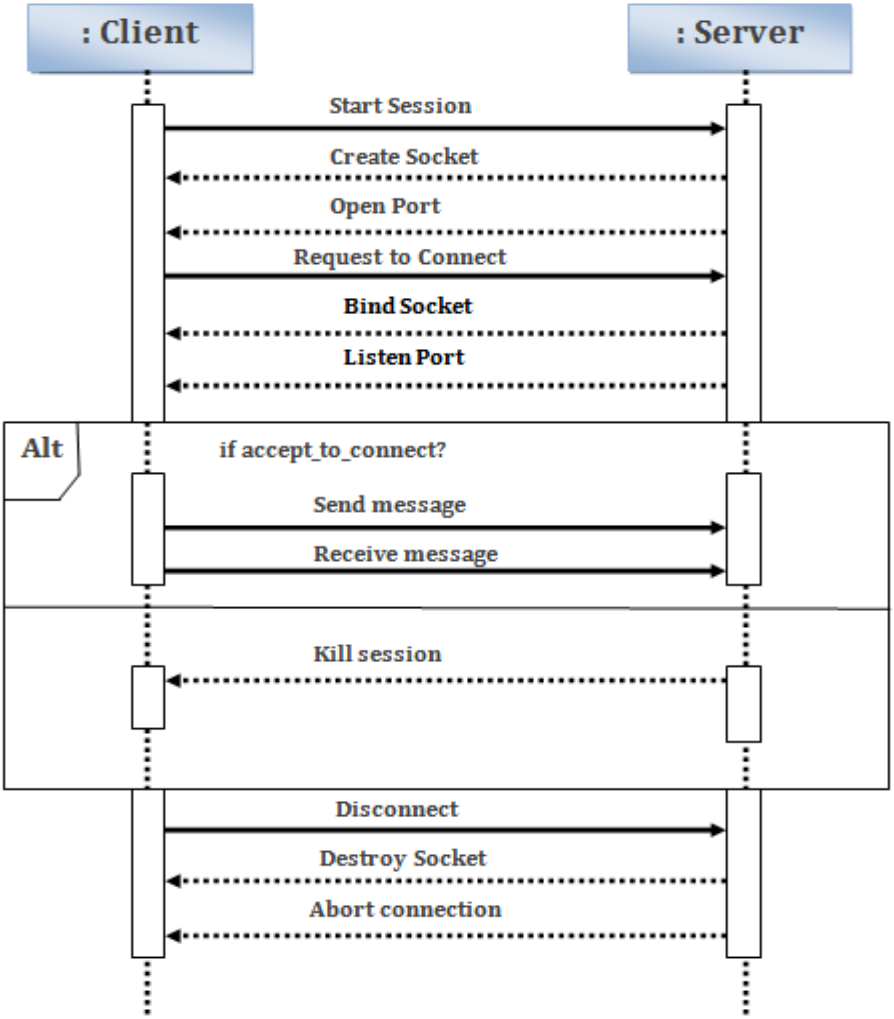

Figure-6: Sequence Diagram of IM Application.

3 Conclusion

With the increasing growth of internet and communication over network, developers are interested in making application that provides services like transfer of massages, files and images over internet or computer network. Team RedBandits has the same objective. Using socket programming I are making a desktop application for instant messaging. This application will provide people to communicate with one another and in group. Socket and streams provides an interface in which two machines can transfer messages with each other. This application will be connection oriented using TCP-Transfer Control Protocol, in which user request for connection before sending some information.

4 Task Management

- **Requirements Gathering**
    - Amber and Zunaira
- **Design and Architecture**
    - Ali and Kamran
- **Implementation**
    - Munir and Shehreyer
- **Testing**
    - Amber, Zunaira, Kamran and Ali

5 Future Scope:

- **Login session information**: I can include a feature in this application that will keep a record of the user's login session.
- **Offline messages**: User can send messages to friends even when they are offline.
- **File transferring and sharing**: User can transfer one or more files to other users. A file can also be shared between two or more users.
- **Profile Database**: The profile information of every user can be stored at the server.
- **Login Timeout**: This feature allows the user to be logged in only for a specific time. After this time span ends, the user is automatically logged out