JAVA BASED MULTI USER CHAT APPLICATION

Submitted To-

Prof. Sivakumar P

School of Computer Science and Engineering

DECLARATION

I/We hereby declare that the project entitled "**JAVA BASED MULTI USER CHATAPPLICATION"** submitted by us to the School of Computer Science and Engineering, VIT University, Vellore - 632014 in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bona fide work carried out by me/us under the supervision of **Prof Shivakumar P ,** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

Arpit Khurana (15BCE0353)

Signature

Apoorv Raizada (15BCE2072)

School of Computer Science and Engineering

CERTIFICATE

The project report entitled "**JAVA BASED MULTI USER CHATAPPLICATION"** is prepared and submitted by **Candidates Apoorv Raizada (15BCE2072) and Arpit Khurana(15BCE0353).** It has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** in VIT University, India.

Guide

(Name & Signature)

Internal Examiner

Signature)

External Examiner (Name &

(Name & Signature)

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my Faculty (Sivakumar P) as well as our DEAN who gave me the golden opportunity to do this wonderful project on the topic (Interactive Club/chapter based application for VIT students), which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them. Secondly i would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Sivakumar P.

Professor, SCOPE

VIT University

1) Introduction

Communication over a network is one field where this tool finds wide ranging application. Chat application establishes a connection between two or more systems connected over an intranet or ad-hoc. This tool can be used for large scale communication and conferencing in an organization or campus of vast size, thus increasing the standard of co-operation. In addition it converts the complex concept of sockets to a user friendly environment. This software can have further potentials, such as file transfer and voice chatting options that can be worked upon later.

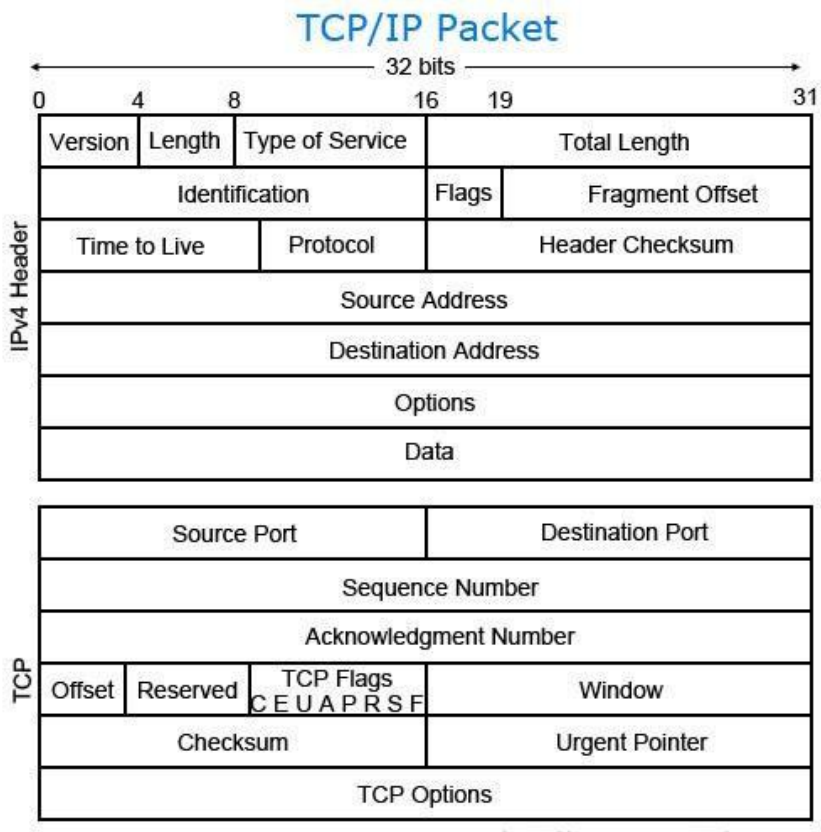
1.1) Relation to External Environment

This tool helps in two major aspects -

- Resolving the names of all the system connected in a network and enlisting them.
- Used for communication between multiple systems enlisted in the resolved list.

1.2) Principle Used: TCP/IP

Short for **Transmission Control Protocol/Internet Protocol**, **TCP/IP** is a set of rules (protocols) governing communications among all computers on the Internet. More specifically, TCP/IP dictates how information should be packaged (turned into bundles of information called packets), sent, and received, as well as how to get to its destination. TCP/IP was developed in 1978 and driven by Bob Kahn and Vint Cerf.



1.3) How does TCP/IP work?

As the name implies, TCP/IP is a combination of two separate protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP). The Internet Protocol standard dictates the logistics of packets sent out over networks; it tells packets where to go and how to get there. IP has a method that lets any computer on the Internet forward a packet to another computer that is one or more intervals closer to the packet's recipient. You can think of it like workers in a line passing boulders from a quarry to a mining cart.

The Transmission Control Protocol is responsible for ensuring the reliable transmission of data across Internet-connected networks. TCP checks packets for errors and submits requests for re-transmissions if any are found.

Three of the most common TCP/IP protocols

- **HTTP** - Used between a web client and a web server, for *non-secure* data transmissions. A web client (i.e. Internet browser on a computer) sends a request to a web server to view a web page. The web server receives that request and sends the web page information back to the web client.
- **HTTPS** - Used between a web client and a web server, for *secure* data transmissions. Often used for sending credit card transaction data or other private data from a web client (i.e. Internet browser on a computer) to a web server.
- **FTP** - Used between two or more computers. One computer sends data to or receives data from another computer directly.

1.4) Domain names and TCP/IP addresses

The TCP/IP address for a website or web server is typically not easy to remember. To remedy this issue, a domain name is used instead. For example, **45.79.151.23** is the IP address for the Computer Hope website and **computerhope.com** is the domain name. Using this method, instead of a set of numbers, makes it much easier for users to remember Computer Hope's web address.

2) Literature Survey

2.1) Socket Overview

A socket is an object that represents a low level access point to the IP stack. This socket can be opened or closed or one of a set number of intermediate states. A socket can send and receive data down disconnection. Data is generally sent in blocks of few kilobytes at a time for efficiency; each of these block are called *a packet*.

All packets that travel on the internet must use the Internet Protocol. This means that the source IP address, destination address must be included in the packet. Most packets also contain a port number. A port is simply a number between 1 and 65,535 that is used to differentiate higher protocols. Ports are important when it comes to programming your own network applications because no two applications can use the same port.

Packets that contain port numbers come in two flavors: UDP and TCP/IP. UDP has lower latency than TCP/IP, especially on startup. Where data integrity is not of the utmost concerned, UDP can prove easier to use than TCP, but it should never be used where data integrity is more important than performance; however, data sent by UDP can sometimes arrive in the wrong order and be effectively useless to the receiver. TCP/IP is more complex than UDP and has generally longer latencies, but it does guarantee that data does not become corrupted when travelling over the internet. TCP is ideal for file transfer, where a corrupt file is more unacceptable than a slow download; however, it is unsuited to internet radio, where the odd sound out of place is more acceptable than long gaps of silence.

2.2) UDP Ports

The User Datagram Protocol is an unreliable, connectionless oriented protocol that uses an IP address for the destination host and a port number to identify the destination application.

The UDP port number is distinct from any physical port on a computer such as a COM port or an I/O port address. The UDP port is a 16-bit address that exists only for the purpose of passing

certain types of datagram information to the correct location above the transport layer of the protocol stack.

A UDP datagram header consists of four (4) fields of two bytes each

1. source port number
2. destination port number
3. datagram size
4. checksum

2.3) Using UDP Sockets

In order to use a UDP socket for network programming one has to follow the following steps as shown in figure.

End point is a combination of IP address and port number. Endpoint objects allow you to easily establish and communicate over TCP/IP network connections between client and server processes, possibly residing on different hosts. The Endpoint class follows a telephone-like model of networking: clients "call" servers and servers "answer" clients. Once a network connection is established between a client and a server, the two can "talk" to each other by reading from and writing to the connection.

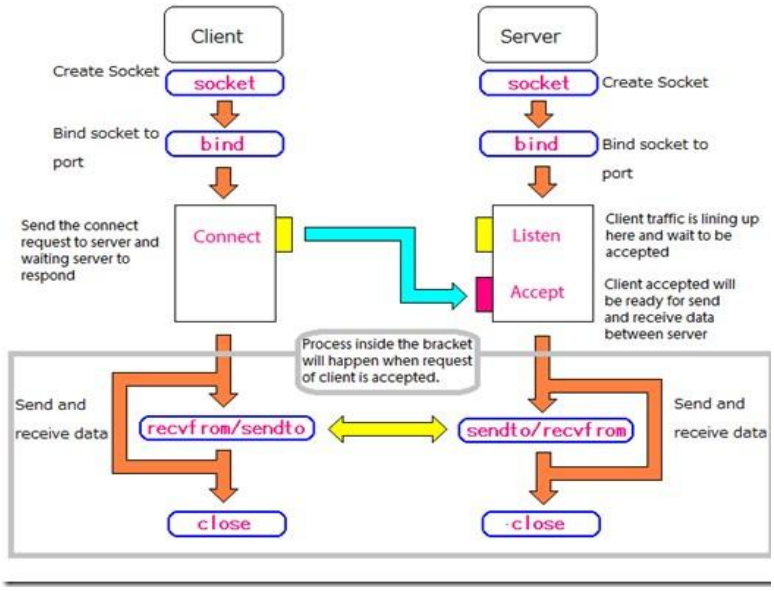
3) System Architecture

In this article we are demonstrating a chat application which can handle multiple users at the same time. It also supports file transfer.

It is entirely based on Java and consists of two parts:

- jMessenger (client application)
- JServer (server application).

3.1) Block Diagram:-



3.2) Features

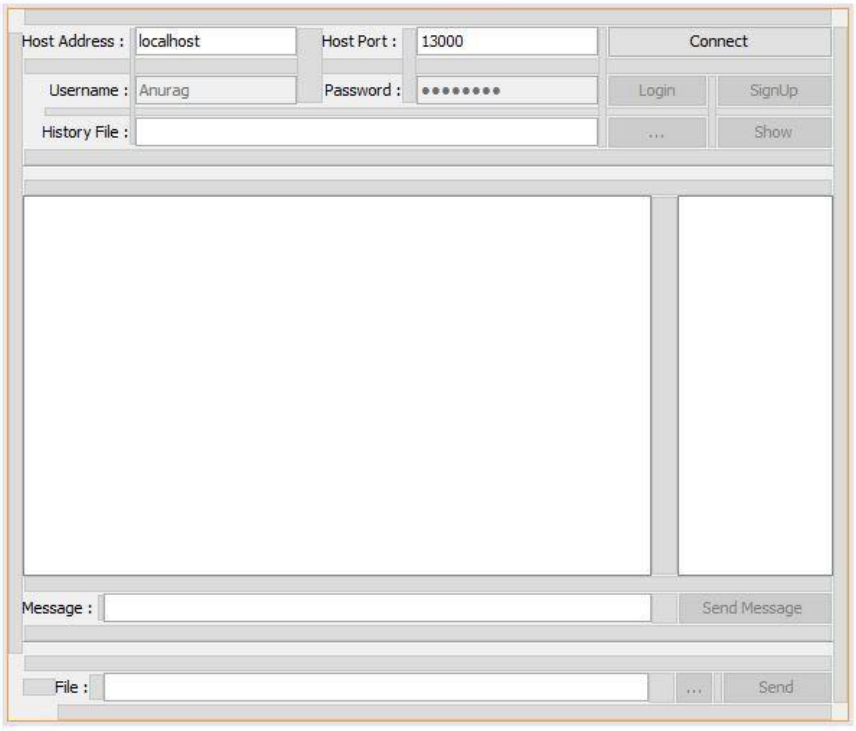
1. Handles multiple users at the same time
2. Support for both public and private messages
3. User signup and login available
4. Support for file transfer

3.3) User Interface

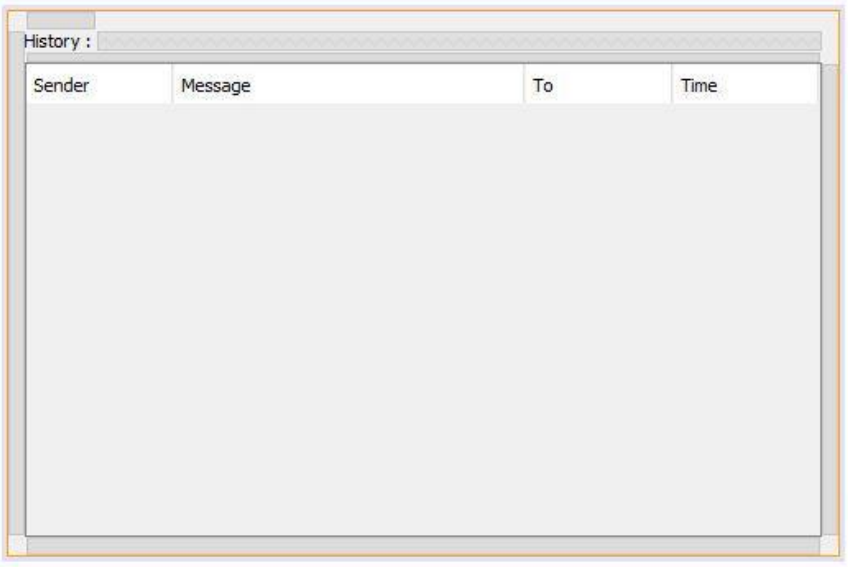
User interface has three major components:

- Chat frame
- History interface
- Server frame

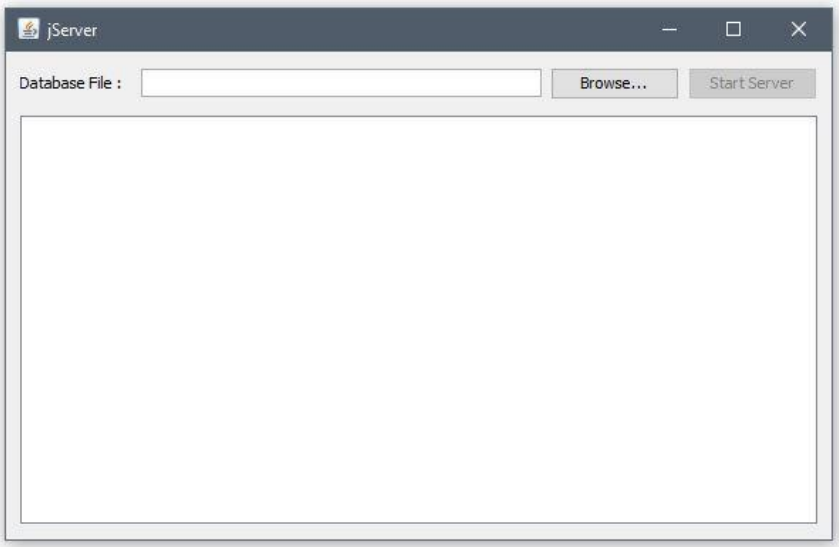
3.3.1) Chat Frame



3.3.2) History Frame



3.3.3) Server Frame



3.4) Message structure

Each message in jMessenger has four fields:

- **Type**: This can be set to message, login, new user, etc.
- **Sender**: The username of sender
- **Content**: Actual content of the message
- **Recipient**: Username of recipient of the message

3.5) Modules implemented

3.5.1) jServer

There are two main classes in **jServer** for handling connections and messages. On startup the SocketServer runs in a separate thread.

The job of SocketServer is to wait for connections and for each connection start a new thread ServerThread.

Once the connection is established, ServerThread will listen for any messages and hand it over to SocketServer to process. Also it will forward messages from other users to the connected user.

3.5.2) Part of the Code To get the Incoming Message

```
// In ServerThread read the incoming message and hand it to SocketServer

Message msg = (Message) streamIn.readObject();
server.handle(ID, msg);
.......


// In SocketServer process the messages based on their type

public synchronized void handle(int ID, Message msg){
        if(msg.type.equals("login")){
                ....
        }
        else id(msg.type.equals("message")){
                if(msg.recipient.equals("All")){ Announce("message", msg.sender,
msg.content); }
                else{
                        // Find the thread of recipient and forward it to him
                }
        }
.......
```

3.5.3) jMessenger

jMessenger first connects to the jServer, specified by its IP-address and port number. Arriving messages are then displayed on message board along with their senders.

When a user wants to send a file, first his request is sent via a message of type upload_req. The recipient then does the following:

1. The recipient side sends its reply in a message of type upload_res
2. If request is accepted then the recipient opens a new port
3. For positive reply, recipient's IP address and port number is sent back
4. The sender, on receiving positive reply connects to this socket and starts file upload

An advantage of this approach is that the clients can chat and transfer files at the same time. Unlike messages, files do not go through jServer.

3.5.4) Part of the Code for reciepient:

```
// On recipient side, start a new thread for download

Download dwn = new Download(....);
Thread t = new Thread(dwn);
t.start();
```

```
send(new Message("upload_res", ui.username, dwn.port, msg.sender));
// Reply to sender with IP address and port number
.....

// On sender side, start a new thread for file upload

// Connect to the port specified in reply
Upload upl = new Upload(addr, port, ui.file, ui);
Thread t = new Thread(upl);
t.start();
```

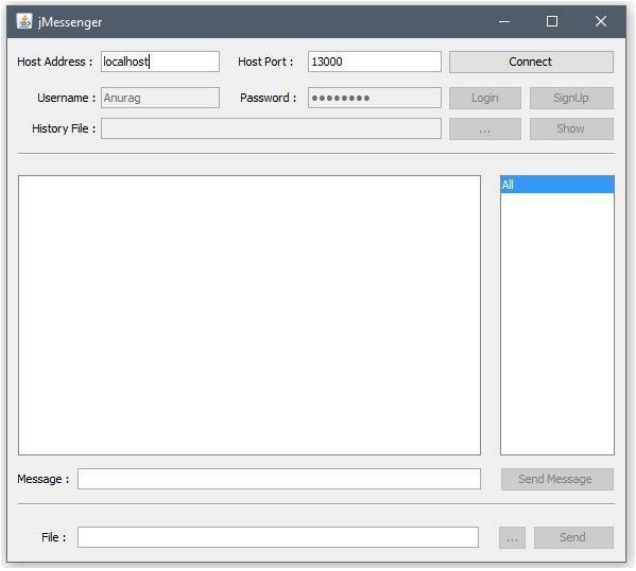
4) Implementation

Run the files *jMessenger.jar* and *jServer.jar* and do the following:

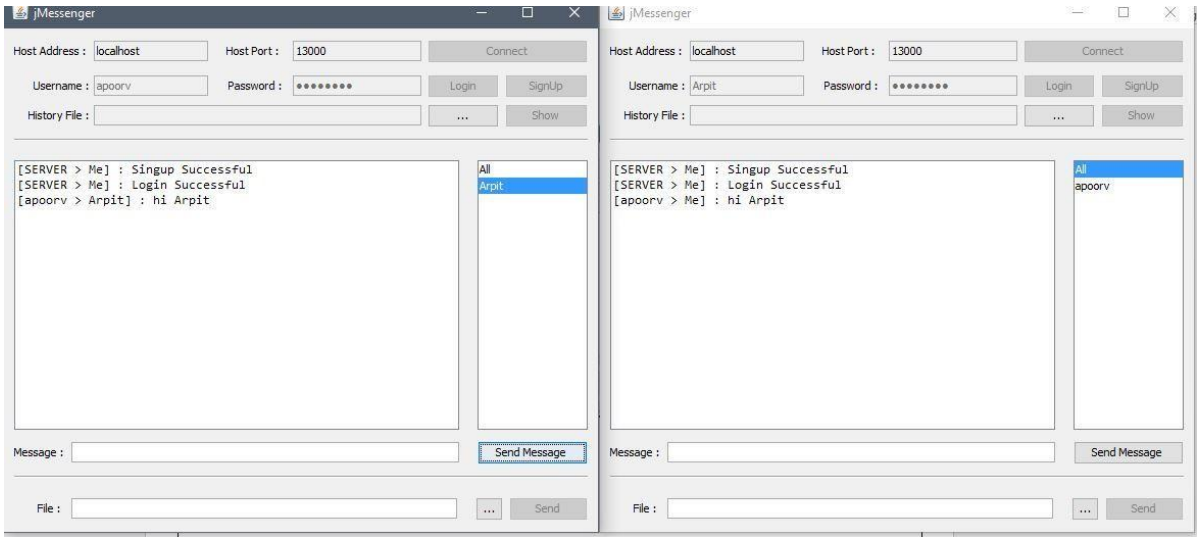
1. On jServer select "*data.xml*" as database file. This file contains usernames and passwords.
2. On jMessen'ger select "*History.xml*" as history file. This file is used to save chat history.
3. In many cases, if jMessenger cannot find the server then adjust firewall to give it network access.

4.1) Starting the Server

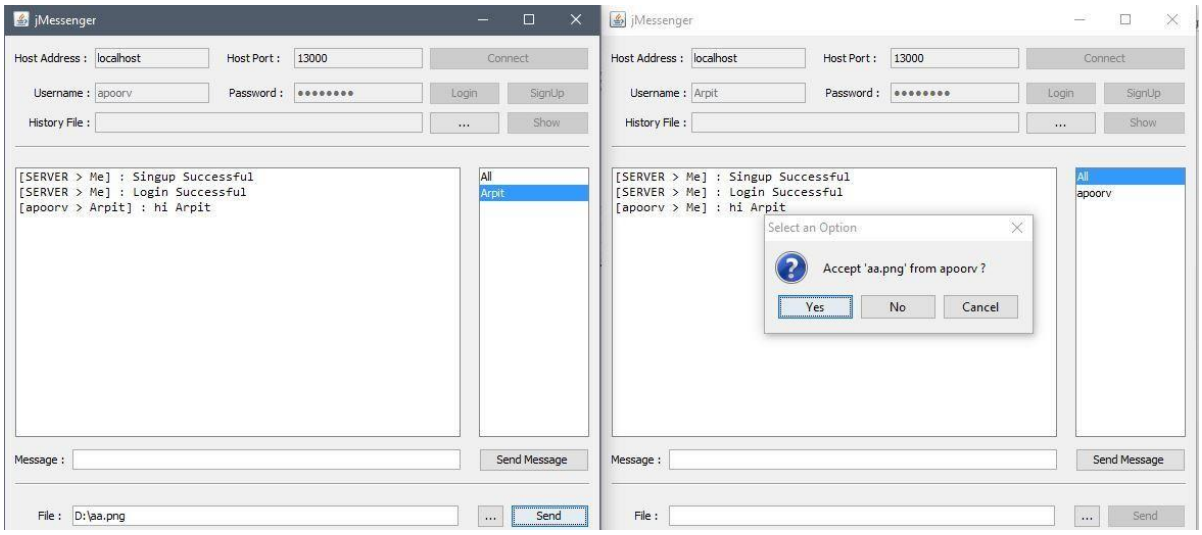
4.2) Starting the Messenger



4.3) Chat Implementation between two Users:



4.4) File Transfer:



5) Conclusion

Our Chat application establishes a connection between two or more systems connected over an intranet or ad-hoc. This tool can be used for large scale communication and conferencing in an organization or campus of vast size, thus increasing the standard of co-operation. In addition it converts the complex concept of sockets to a user friendly environment. This software can have further potentials, such as file transfer and group chat. So this application is very useful and can be used in real world.

6) References:

1. Lam, Edmund Y.and Joseph W. Goodman. "Discrete Cosine Transform domain restoration of defocussed images." Applied optics. 37, 6213-6218 (1998).
2. Ken Kabeen and Peter Gent. "Image Compression and Discrete Cosine Transform" College of Redwoods.
3. Predictive Text Entry using Syntax and Semantics - Sebastian Ganslandt, Jakob Jörwall, Pierre Nugues
4. http://www.java2s.com/Code/Java/JSTL/SimpleChatApplication.htm
5. http://www.codeproject.com/KB/java/java_applet_chat_with_gui.aspx
6. http://oreilly.com/catalog/javmesser/chapter/ch02.html