Anonymous Multi-Chat Application

MD. AFRIDI BAIG[1], B. MURALI[2]
[1]*PG Student, Quba College of Engineering & Technology*
[2]*Assistant professor, Quba College of Engineering & Technology*

***Abstract* — The Internet has revolutionized how we communicate and transact, making the world more connected. This project focuses on developing a chat application using Java multithreading and networking concepts. The application supports private and public messaging, as well as file sharing, and is designed for reliability and security.**

Chat applications have become integral to daily life, with popular examples like WhatsApp, Facebook, and Instagram. These platforms continuously evolve to offer new features and ensure data security, addressing the growing concern of data theft. Our chat system aims to provide a secure and efficient communication tool for organizations such as colleges and IT parks.

***Index Terms* — Chat application, data security, Java multithreading, network communication**

I. INTRODUCTION

Chat applications have become essential in our daily lives, offering various features that enhance communication. Popular apps like WhatsApp, Facebook, and Instagram have billions of users and continuously evolve to stay competitive. These platforms prioritize security to protect user data, addressing the growing issue of data theft.

Objectives of the Chat Application:

- Implement a chat system for private networks or organizations.
- Ensure the security of messages and private data shared over the network.
- Store confidential data securely.
- Develop a two-way communication system.
- Add features that surpass traditional systems available in the market.
- Support both group and private chats.
- Enable easy and fast communication between people.
- Allow unlimited data transfer without size restrictions.
- Connect people anytime, anywhere.
- Provide unlimited storage for message data.

This project aims to create a reliable and secure chat application using Java multithreading and networking concepts, suitable for organizations like colleges and IT parks.

II. REQUIREMENT SPECIFICATION

Hardware Requirements

-Processor: 2.4 GHz Clock Speed

-RAM: 1 GB

-Hard Disk: 500 MB (Minimum free space)

Software Requirements

-Operating System: Windows 7 or above

-Platform: Java

-Special Tools: Visual Studio CODE

III. ARCHITECTURE & DESIGN

This application has been implemented based on the client/server model.

A. SERVER

A server may be a computer dedicated to running a server application. Organizations have dedicated computers for server applications, which need to be maintained periodically and monitored continuously for traffic loads to prevent downtime that could affect the company’s revenue. Most organizations have a separate monitoring system to detect server downtime before it impacts clients. These server computers accept client requests over network connections and respond accordingly.

There are various types of server applications based on their dedicated functions. Some handle requests and perform tasks like business application servers, while others, like proxy servers, simply pass requests through. These server computers must have a faster CPU, more RAM, and larger hard drives. Additional distinctions include redundancy in power supplies, network connections, RAID, and modular design.

B. CLIENT

A client is a software application or system that requests services from a server. These clients do not need to be connected to the server through wired communication; wireless communication is also possible. A client with a network connection can send requests to the server.

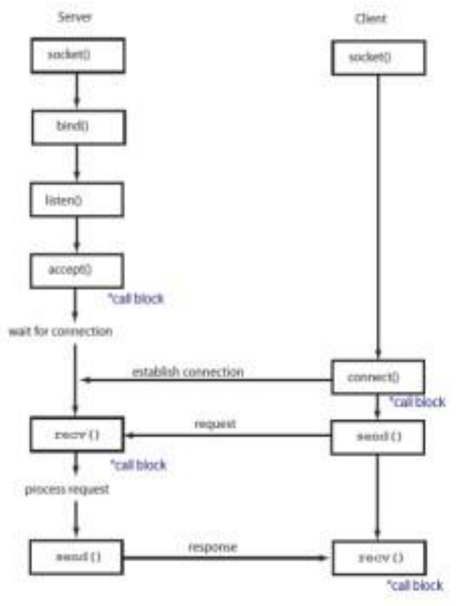


Fig. 1 Block Diagram

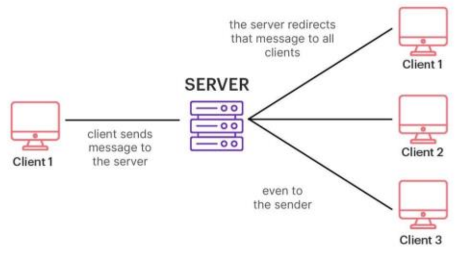

the server redirects that message to all clients
SERVER
Client 1
client sends message to the server
Client 1
Client 2
even to the sender
Client 3

Fig. 2 Explanatory Diagram

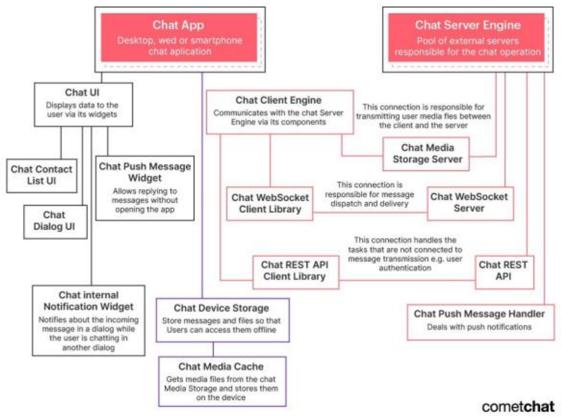

Chat App
Desktop, wed or smartphone chat aplication
Chat Server Engine
Pool of external servers responsible for the chat operation
Chat UI
Displays data to the user via its widgets
Chat Client Engine
Communicates with the chat Server Engine via its components
This connection is responsible for transmitting user media fles between the client and the server
Chat Media Storage Server
Chat Contact List UI
Chat Push Message Widget
Allows replying to messages without opening the app
Chat Dialog UI
Chat WebSocket Client Library
This connection is responsible for message dispatch and delivery
Chat WebSocket Server
This connection handles the tasks that are not connected to message transmission e.g. user authentication
Chat REST API Client Library
Chat REST API
Chat internal Notification Widget
Notifies about the incoming message in a dialog while the user is chatting in another dialog
Chat Device Storage
Store messages and files so that Users can access them offline
Chat Push Message Handler
Deals with push notifications
Chat Media Cache
Gets media files from the chat Media Storage and stores them on the device
cometchat

Fig. 3 Sequence Diagram

IV. IMPLEMENTATION

1. A static Server socket is created at the beginning and is then bound to a host and port.
2. After the server instantiates the socket on a particular host, it begins to listen on the specified port. The server then accepts requests from clients through this port.
3. Once the server is started, it can accept requests from clients.
4. On the client side, a socket is instantiated to connect to the server.
5. A new Server Thread using the socket is created to handle requests from multiple clients.
6. After accepting a request, both read and write operations occur simultaneously, allowing clients to communicate with each other and share resources.
7. Once communication is finished, the socket is closed on both the client and server sides.

V. CODE

```
// Client.java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;


public class Client {
  public static void main(String[] args) {
    // Default host and port
    String host = "localhost";
    int port = 1222;


    if (args.length != 2) {
      System.err.println("Considering localhost
and Port 1222");
      // System.exit(1);
    } else {
      host = args[0];
      port = Integer.parseInt(args[1]);
```

```
        }


        try {
            Socket socket = new Socket(host, port);
            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));


            String inp;
            while ((inp = stdIn.readLine()) != null) {
                out.println(inp);
                System.out.println("Received from Server: " + reverseString(in.readLine()));
            }


            in.close();
            out.close();
            socket.close();
        } catch (UnknownHostException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        } catch (IOException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }


    public static String reverseString(String s) {
        StringBuilder rev = new StringBuilder(s.length() + 1);
        String[] words = s.split(" ");
        for (int i = words.length - 1; i >= 0; i--) {
            rev.append(words[i]);
            rev.append(" ");
        }
        rev.setLength(rev.length() - 1);
        return rev.toString();
    }
}
```

Compile the code using the following lines:

```
$ javac server.java
$ javac client.java
```

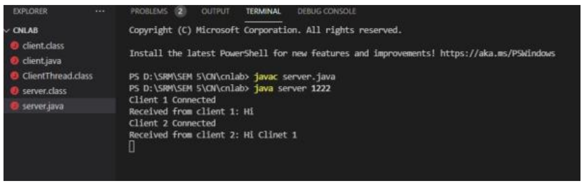
Run the code:

```
$ java server <PORT NUMBER>
$ java client localhost <PORT NUMBER>
```

VI. EXPERIMENT RESULTS AND ANALYSIS

Results

Server

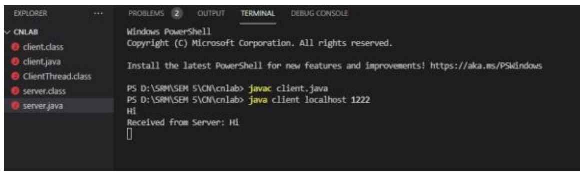


```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\SRM\SEM 5\CN\cnlab> javac server.java
PS D:\SRM\SEM 5\CN\cnlab> java server 1222
Client 1 Connected
Received from client 1: Hi
Client 2 Connected
Received from client 2: Hi Clinet 1
```

Client 1

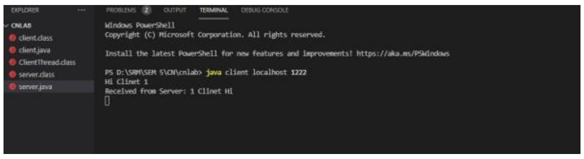


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\SRM\SEM 5\CN\cnlab> javac client.java
PS D:\SRM\SEM 5\CN\cnlab> java client localhost 1222
Hi
Received from Server: Hi
```

Client 2



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\SRM\SEM 5\CN\cnlab> java client localhost 1222
Hi Clinet 1
Received from Server: 1 Clinet Hi
```

VII. RESULT ANALYSIS

The above figures show the output of the multi-user chat application. When any user sends a message, it is sent to all the receivers along with an anonymous name. The maximum user count is set, determining how many users can connect. Once the client count exceeds this limit, the server intimates that it is busy, and the client will wait until the server becomes idle. Any user can leave the chat box after sending a message. Once the sender leaves, they will be removed from the chat, and all other users will be informed. When a user leaves the chat box, other users waiting for the server to become idle will get a chance to communicate. Instead of sending a message to all users in the chat box, messages can also be forwarded to a particular user.

VIII. CONCLUSION AND FUTURE WORKS

The chat application provides a better and more flexible system for chatting. It is developed with recent advanced technologies to ensure reliability. The main advantages of the system include instant messaging, real-world connectivity, enhanced security, and group chat capabilities. This application is well-suited for organizations seeking private communication solutions. Additional features, such as conference calls, video chat, and location sharing, will be included based on public demand.

Future Works:

Further enhancements will focus on improving security, enabling video calls, supporting large file transfers, and adding other features necessary to stay competitive. Additionally, efforts will be made to implement the system in private networks.

IX. REFERENCES

[1] https://www.geeksforgeeks.org/multi-threaded-chatapplication-set-1/
[2] https://gyawaliamit.medium.com/multi-client-chatserver-using-sockets-and-threads-in-java2d0b64cad4a7