ChatChamber

Gaurav Giri, Iza kc, Ajaya Chaudhary, Narayan Poudel
Department of Computer and Electronics Engineering
Kantipur Engineering College, IOE
Dhapakhel,lalitpur,Nepal
kan077bct034@kec.edu.np
Kan077bct039@kec.edu.np
Kan077bct007@kec.edu.np
Kan077bct049@kec.edu.np

***Abstract* - The objective of the ChatChamber is to create a chat application which provides a bidirectional communication between the clients.The ChatChamber Program will enable the user to chat with the logged users in the chat room with an active server. It provides a very simple, fast, multithreaded platform with an independent web socket and client library that is used for the connection between client and server. ChatChamber achieves a simple, reliable, multi-user in-terminal chatting service that supports sending and receiving of texts.**

Keywords - In-terminal application, IP and Ports, star topology, TCP Connection, web sockets, server, clients

I.Introduction

ChatChamber is a client-server system where the users exchange text messages between the system's users with the help of a central server. The user of the system is defined as client-server. Chat system is a distributed programming which consists of two distributed components,a chat server and a chat client. Chat client supports all communication including requesting chat server location information from a location server and displaying received chat messages. Chat server will conduct chat sessions and manage all chat clients. Basically, a chat client starts the chat session by requesting the communication parameter (server name and port number). In a topological way, this system is working on star topology because if the server is down, then the connection is over. Chat System is a form of communication that utilizes computer programs that allow for two-way conversations between users in real time (events that occur in cyberspace at the same speed that they would occur in real life). Typically the users will connect to a chat server using a chat client and meet in a chat room. Once the users are in the same chat room, they can converse with one another by typing messages into a window where all of the other users in the chat room can see the message. The user can also see all of the messages entered by the other users. Conversations are then carried on by reading the messages entered by the other users in the chat room and responding to them.

It is an open source application so the users themselves can gain insight on the code of the program and can contribute to help better the program itself. The application provides encryption for secure message sharing. As it is a terminal based application no resources are required to render GUI and takes up minimal resources in the regards of memory and CPU uses.

II. Features

A. Password Protection

Password protection is an access control technique that helps keep important data safe from hackers by ensuring it can only be accessed with the right credentials[1].Password protection is one of the most common data security tools available to users. ChatChamber requires a registered login and password to use the chatting service which adds security to the application itself and to provides privacy to the user

B. *Reliable and Secure Messaging*

ChatChamber is an easy to use terminal application. The use of a primitive GUI may look daunting at the first sight but the learning curve for using ChatChamber is not steep. Once the user is accustomed with the GUI of ChatChamber which is easy to navigate with a few simple steps, the application provides a reliable and secure chatting service.ChatChamber has SSL integrated into it to provide secure messaging.

C. Opensource

ChatChamber is an open source application that can easily run on LINUX and Windows based systems

.

D. Free of cost messaging

ChatChamber uses the internet to send and receive messages so it does not require any additional messaging charges.

III. Whereabouts of the Program

The program is a terminal based application written in C. It works upon the primitive sockets where there are TCP(Transmission Control Protocol)[2] connections between clients and a server. ChatChamber will have a hosting server and clients will be able to join the server and share text data among the clients and some user[1]defined functions for the manipulation of the clients. The server will manage clients using a Data Structure called LinkedList and the processes for them will be handled by threads. So, every time a client joins the server, a thread and a node in linked list is created. The concepts and libraries used are described below

A.*Transmission Control Protocol*

TCP is a standard that defines how to establish and maintain a network conversation by which applications can exchange data. [2]TCP works with IP, which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules that define the internet.TCP performs the following actions:

- Determines how to break application data into packets that network can deliver
- Sends packets to and accepts packets from the network layer
- Manages flow control
- Handles retransmission of dropped or garbled packets
- provides error-free data transmissions
- Acknowledges all network packets that arrives [2]

Network packets are basic units of data that's grouped together.

B.*Server*:

A server is a software or hardware device that accepts the requests made over a network. This is the computer or a system which will accept or listen to the connection of other devices or systems to it. Servers are the main root of the network. If the server is down, the devices connected to it will lose their connection. Server handles and manages the devices connected to it and sets up various rules and protocols for the devices connected to it to make the network more secure, efficient and reliable[3].

C. *Clients:*

These are the devices or computers which are connected to a server. They will be limited to their functions that are assigned from the server. If the server rejects the clients data that violates the server's rule or protocol then the data will not be processed further[3].

D.*Sockets*

A socket is one endpoint of a two way communication link between two programs running on the network. The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication takes place. A socket connecting to the network is created at each end of the communication. Each socket has a specific address. This address is composed of an IP address and a port number.

Sockets are generally employed in client server applications[4]. The server creates a socket, attaches it to a network port address then waits for the client to contact it. The client creates a socket and then attempts to connect to the server socket. When the connection is established, transfer of data takes place.

E.*Threads*

A thread is an independent set of values for the processor registers (for a single core). Since this includes the Instruction Pointer (Program Counter), it controls what executes in what order. It also includes the Stack Pointer, which had better point to a unique area of memory for each thread or else they will interfere with each other. Threads are the software unit affected by control flow (function call, loop, goto), because those instructions operate on the Instruction Pointer, and that belongs to a particular thread. Threads are often scheduled according to some prioritization scheme (although it's possible to design a system with one thread per processor core, in which case every thread is always running and no scheduling is needed).

F. *Libraries used*

1.**<stdio.h> :**

This is the C-standard library that contains functions for input and output. Functions like: printf(), sprintf(), scanf(), fgets() etc… are present inside this header file.

2.**<stdlib.h>:**

stdlib.h is the header of the general purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions and others.

3.**<string.h>:**

This header file contains the functions that are used for manipulating strings. Functions for string copying, concatenation of two strings etc are present inside this header file.

4.**<signal.h>:**

It is the C Standard Library for signal processing that defines how a program handles various signals while it executes. A signal can report some exceptional behavior within the program, or a signal can report some asynchronous event outside the program.

5.**<unistd.h>:**

the header file that provides access to the POSIX operating system API. It is defined by the POSIX. 1 standard, the base of the Single Unix Specification, and should therefore be available in any POSIX compliant operating system and compiler. Sleep() function is present inside this header file.

6.**<sys/types.h>:** The sys/types.h header file defines a collection of typedef symbols and structures. For example: u_char → unsigned char, u_int → unsigned int, pthread_t → identify a thread etc.

7. **<sys/socket.h>:**

This header file contains all the functions and structures that are used for creating sockets. Function

like socket() to create a socket, bind() to bind an address in server, listen() to listen to the connection, connect() to connect to a socket, and the structure of sockaddr is present inside socket.h. This library helps to initiate the socket.

8.**<netinet/in.h>:**

The netinet/in.h header file contains definitions for the internet protocol family. It contains some functions like htons() that converts the given host short port number to network short bytes.

9.**<arpa/inet.h>:**

The `arpa/inet.h` header file contains definitions for internet operations. Functions like inet_ntoa() , inet_aton() are present that converts the given address to network bytes or vice-versa.

10. **<pthread.h>:**

This header files contains the functions and macros that are needed to implement a thread and manipulate them. This is POSIX-Standard library. Data types like pthread_t , functions like pthread_create(), pthread_join(), pthread_mutex() are present inside this file to work with threads.

IV.Requirements

- A Unix-based operating system (Windows users can use the Windows Subsystem for Linux (WSL))
- GCC, the GNU Compiler Collection
- Make
- Docker (if you want to run the application in a container)

V.Insights and using ChatChamber

When the application is first opened the user is prompted with a window that requires server IP and port. Then the user is prompted with a login and registration menu.user can either login with the registered username and password or can choose to create a new username to register that does not exist in the database of the ChatChamber application. while registering the user requires a valid server invitation code.If the valid code is entered then the user is prompted with a window that requires the user to create a new username, enter the password and confirm it.After the user has entered the username, password, ip, and port successfully the user can now use the chatting application.

First of all, a server will host a ChatChamber by binding its ip (say XXX.XXX.XXX.XXX) and port (YYYY) and listening to the clients that will join it. The clients will then enter the ip and the port no of the server to connect the server. Then the server will prompt them 2 options, one will be for login and another will be for registering an account. Then User will choose accordingly, the verification is done in the server by reading the user's log and a new account will be appended to the data file if the user registers a new account. After that, the clients that are already present in the chatroom will get notified about the information of new client joining. And then the message one client sends will be sent to all the other clients through the server. If a client leaves, then the signal is caught in the server and then all the clients will get notified. All the chats will get stored in a log file and if any user wants to read the log file, they are freely allowed to see them.

VI.Acknowledgements

This chat application was developed with the help of several open-source libraries and technologies. We would like to extend our sincere gratitude to the following:

The creators and contributors of the ncurses library, which was used to provide a user-friendly interface for the application.

The creators and contributors of the sqlite3 library, which was used to store the application's data.

The creators and contributors of the openssl library, which was used to provide secure communications through SSL.

The creators and contributors of the pthreads library, which was used to implement multi-threading in the application.

The creators and contributors of the Docker technology, which was used to deploy the application in a containerized environment.

Without the tireless efforts of these individuals and groups, this application would not have been possible. Thank you for your contributions to the open-source community.

We would also like to thank the IEEE Pulchowk and logpoint for providing us the opportunity to work on this project.

VI. References

[1] Microsoft. “What is password protection.” https://www.microsoft.com/en-ww/security/business/security-101/what-is-password-protection.

[2] Lutkevich, Ben. “What Is Transmission Control Protocol (TCP)? Definition from Searchnetworking.” *Networking*, TechTarget, 4 Oct. 2021, https://www.techtarget.com/searchnetworking/definition/TCP

[3] “What Is a Server?” *Computer Hope*, 31 Dec. 2022,https://www.computerhope.com/jargon/s/server.html

[4]“How Sockets Work.” *IBM*, https://www.ibm.com/docs/en/i/7.1?topic=programming-how-sockets-work.