INSTANT MESSAGING SERVICE

Final Project Report

Nirmal Karia (01008723)

Team Members: Sameer Nandu

Kewal Shah

Date: - 5/17/2015

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**ABSTRACT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Instant Messaging service mainly refers to the transfer of messages between users in real-time. Used in a conversational mode. Transfer of messages back and forth is fast enough for participants to maintain an interactive conversation.

In this project we have implemented a multi user messaging service using socket programming in C. Our main aim was to make multi-client supporting server allowing communication between any two different clients after proper authentication. Clients are also provided with an option of group chat or private chat.

The server generates a new numeric pin for each client connecting to it. A client will be recognized by its pin. Any client wanting to communicate with another will require this pin to search the client in this messaging service. The server supports a friend list for clients connecting to it and store a database for each client.

Redundancy and error handling are also performed. So if an original server fails a new server will activate the connection and carry out the communication. After a proper connection is established, the clients will be able to chat and send instant messages in real time.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_INTRODUCTION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

In this project the basic scenario was to create a server which supports multiple client successfully from a single server. Chat Server implemented is a multithreading, concurrent server which handles each client separately by creating threads for each client. Server maintains a separate data base for each connecting client and also verifies each incoming client for an existing client and notifies the user about its existence by verifying the data base.

Server also maintains a separate pin or a password for each connection and then accepts the client connection. After verification server allows the user to choose the type of chat he wants to have with other clients. It offers private or group chat as the choices for communication. Then the client is then directed to other client with whom he wants to connect depending upon the type of communication he prefers.

After, verification and authentication client is then directed for communication and transfer of messages occur using send and receive buffer. Proper concurrency and availability is maintained throughout the session of messaging service. Server is also scalable as clients can be created and friend-list can be increased as and when required. Error handling is maintained for each socket calls and at both the client and server side and client are also checked for each wrong entry for pin and user name ID.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_PURPOSE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

A Chat server enables the client to store its data to the server database. For the purpose of security server verifies and authenticates the client username and password. Server offers client two type of messaging services that is private and group chat. The client request will be serviced at all times with the implementation of High Availability. Even if one server is unavailable the client can communicate with the other server. The server also maintains a list users available online.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MOTIVATION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The project ‘Internet Messaging Service’ aims at designing and the implementation of a synchronized multi-threaded concurrent server-client instant messaging service. The services within different clients will be provided by the server which is concurrent and multi-threaded. Concurrency is said to be provided if server can handle more than one request at the same time. If the server is non-concurrent, then the blockage of resources can happen which is not ideal for a server client scenario and is non-efficient. The server is designed to support more than one users and the main motivation behind the project is the communication between different clients via the server.

The concurrency feature ensures that the number of clients at any point is not an issue at the server. Multi-thread provides a separate thread for each of the client request, so that, whatever the client does will be served by that thread only. This helps in achieving efficiency and also delivers the server in a timely manner.

**\_\_\_\_\_\_\_\_\_\_\_Application and Protocol Design\_\_\_\_\_\_\_\_\_\_\_\_**

**TCP:**

TCP is byte oriented protocol. The Transmission Control Protocol provides a communication service at an intermediate level between an application program and the Internet Protocol. It provides host-to-host connectivity at the [Transport Layer](http://en.wikipedia.org/wiki/Transport_Layer) of the Internet model. An application does not need to know the particular mechanisms for sending data via a link to another host, such as the required packet fragmentation on the transmission medium. At the transport layer, the protocol handles all handshaking and transmission details and presents an abstraction of the network connection to the application.

The protocol used at server is TCP protocol which maintains a reliable, connection-oriented connection between the client and server. TCP protocol provides connection-oriented and reliable service for communication and data transfers. As messaging service has an obvious need for guaranteed delivery as well as reliability, we choose to use TCP protocol in our implementation. For authentication and message transmission, we develop a protocol specific for this application.

**IRC:**

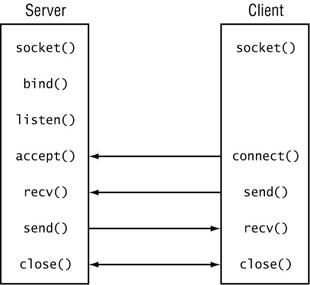
Internet Relay Chat (IRC) is an [application layer](http://en.wikipedia.org/wiki/Application_layer) protocol that facilitates the transfer of messages in the form of text. The chat process works on a client/server networking model. IRC clients are computer programs that a user can install on their system. These clients communicate with chat servers to transfer messages to other clients. IRC is mainly designed for [group communication](http://en.wikipedia.org/wiki/Many-to-many) called [channels](http://en.wikipedia.org/wiki/Internet_Relay_Chat#Channels), but also allows one-on-one communication via [private messages](http://en.wikipedia.org/wiki/Instant_messaging) as well as [chat and data transfer](http://en.wikipedia.org/wiki/Direct_Client-to-Client), including [file sharing](http://en.wikipedia.org/wiki/File_sharing).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Project Modules\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Socket Framework:-**

For the end point communication between Client and Server, basic socket programming framework has been implemented. A socket is an end point of communication between two systems, it is a combination of IP address and port number. It allows communication to take place on the system.

<sys/socket.h> is the socket API which has been used for endpoint communication



**There are** four tasks to perform before a server can transfer data with a client connection:

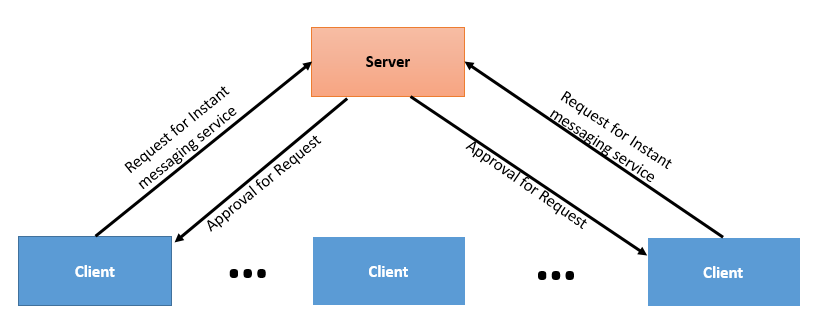
1. Create a socket.
2. Bind the socket to a local IPEndPoint.
3. Place the socket in listen mode.
4. Accept an incoming connection on the socket.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_SERVERDESIGN\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The server design is the main and crucial part for Instant messaging service. The services offered by the server are as follows:-

* Registering The Users(User Login)
* Password Authentication
* Users Available Online
* Personal Chat messenger
* Group Chat-Room
* Error Handling
* Redundancy

**USER REGISTRATION AND PASSWORD AUTHENTICATION:-**



After, a proper connection is established between the client and server after necessary socket calls clients will be able to chat and send messages in real time. After client connection, server ask the client if he is an existing user or a new user. If he is a new user server ask for user ID and then ask to enter the passwords and proceeds. For an existing user server ask the client for a username and verifies this username in its database and proceeds further. Server also maintains a list of records of all clients who are registered. This will help to check for any authorization and will also provide the list of existing clients to future clients who wish to add/chat to any respective client ids.

For each incoming client server maintains a separate database that is server creates a separate folder for each client with username of the client as the folder name and stores the username and password entered by the client in the folder. The password and socket descriptors of the incoming connection in a separate text file are stored in a separate text file.

For each incoming client server maintains a common file where all the username of the connected clients are stored. Security is also maintained here as for each attempt server verifies the data entered by the client and for each invalid input server tells the client to enter the username and password invalid please try again.

**Server Database**: - **User 1 User id: Nirmal**  **Password** – **12345**

**SERVER Existing User (Nirmal)**

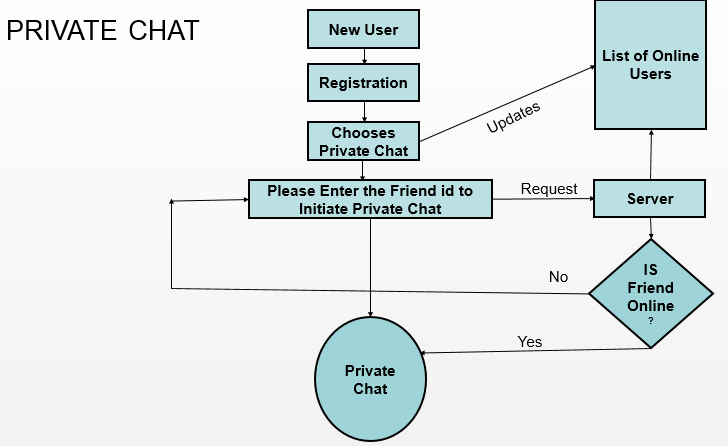
**Invalid Password..! Enter your Password**

**Pin: 12345**

**Enter your Password**

**SECURITY**

**PERSONAL CHAT MESSENGER:-**

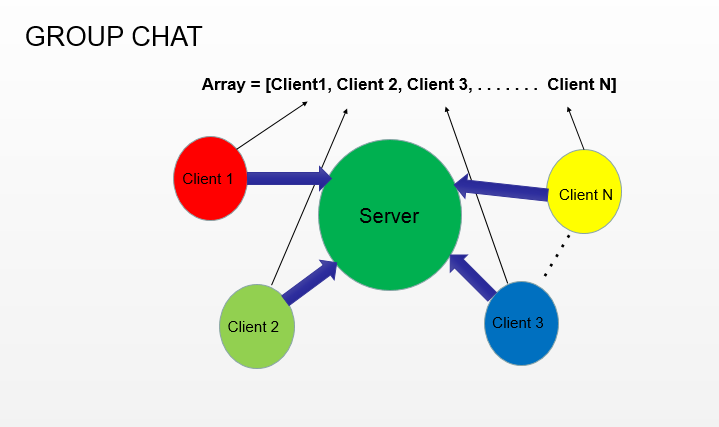


When the clients connects to the server after performing the necessary socket calls server request the client to enter the username and password and verifies this information in the database and if a valid match is found then directs the client to select the messaging service. Server offers private and group chat as the type of messaging services.

So when the client selects private chat as the option server ask the client to enter the friend id or the username of the client with whom he wants to chat. Server also updates his database and enters the username of the newly connected client in the list of online users. Server along with maintaining the database for each client where he stores the password and socket descriptor in a separate text file in a folder he also maintains a separate list for users available online and ready for communication.

After, updating and verifying the database if server finds a valid match for the username entered by the client then a private session is established between the clients if both the clients are willing to communicate. If there is no valid match found in the database the server ask the clients to re- enter the username of the client with whom he wants to connect.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_GROUP CHAT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



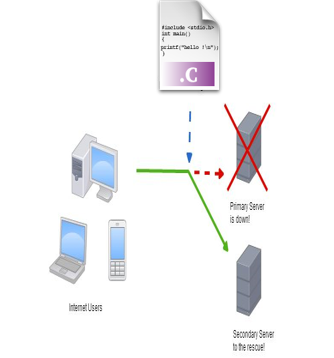
In group chat a chat room is created when clients select this service by selecting the option group chat as directed by the server. Whenever any client selects this option socket descriptors of that particular client is stored in an array.

This array is declared globally in the program. Array defined here contains a list of socket descriptors of all connected clients. After, that a select statement is used to active that socket descriptor and to see if that descriptors is ready to read or write. Select is mainly used to check if the descriptors is active and can be used to read or write.

After, reading a function send\_to\_all is called and this function takes the socket descriptors from the array one by one and message is sent to each client except from the one who entered the message. Each client is notified about the newly joined clients in the chat room.

Similarly, if a client disconnects its descriptors is removed from the array and other members are also notified about the client leaving the chat room.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_REDUNDANCY\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

1. Redundancy is basically extra hardware or software that can be used as backup if the main hardware or software fails.
2. A higher layer of redundancy is achieved when the backup device is completely separate from the primary device.
3. Through automatic detection, an error on your primary server can be detected and traffic will automatically be sent to a backup server.

In the back –up server a text file is created name count\_file.text which maintains the time the server is activated. A terminal is opened where the server file is executed and connection from the client is accepted and client is directed to further steps. It record the count in the text file and stores the 0 count and then comparison is done with the new recorded count.

Then goes to sleep for 20 sec and records the count again and if both count match which means main is server is down and the count is reduced to 0 and then executes the server file again saying that that connection is down initiating new connection which is the back -up server and client gets client get connected to the new server and the server then again ask the client for verification and authenticates that client.

If the count do not match then it again proceeds with the counting. The back-up server is executes only when the count matches to be 0 which is an indication that the server is down and there is a need to start the back- up server to proceed with the connection.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Error Handling\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

During the course of this project there were several cases that required error handling. Some of them have been covered in this paper.

1. The connection between the server and client may be lost because the server is no longer available. In this case the client connects to the other server and since the other server has files synchronized with the server that just became unavailable there would be no mismatch of data and also no downtime.

2. Both servers are unavailable when the client tries to connect to either of them. In this scenario a proper error message is provided to the client informing about this situation.

3. When client enters an incorrect information server verifies the database and if the there is no match for the entered information server request the client to re- enter the details.

4. Error handling is also performed at each socket call so whenever there is issue in any of the socket call it is returned with -1 indicating about the error occurred in calls or accepting a connection.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_PERFORMANCE EVALUATION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Server implemented here is a concurrent – supports multiple clients, highly available – services are provided as long as the server is online, redundant – a back server activates and carries the communication if, the original server fails.

CONCURRENCY

* Concurrency is established using multithreading. On accepting new connection from clients, the main thread in the server creates a new thread to service that specific client.
* The server is free to accept new clients. Two communicating threads interacts using shared memory of the server and send the messages back to their respective clients

AVAILABILITY

* Messaging services are available as long as the server is online and clients are connected to it. If a client disconnects, it is able to reestablish the connection to a desired client as the client’s friend list are stored on server.
* The client can look up to this data base and establish a connection.

SCALABILITY

* The messaging services are highly scalable and support communication between multiple pair of clients as multithreading is implemented in the server.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_TESTING AND VERIFICATION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Testing Plan:**

Some of the test cases considered during the final testing phase of the project are:

**1**. testing the basic scenario of creating multiple clients successfully form single server

**2**. To Test whether Client is able to request server to avail for instant messaging service.

**3**. To Test whether server handles the request from client and provides a valid and unique id to clients.

**4**. To test the creation of database in a text file by server, having all the details of all the clients that have already requested for the messaging service.

**5**. Create a list for all clients which shows their current friend list.

**6**. To ensure that any client is not able to message any other client until they are friends.

**7**. To check whether proper concurrency and availability is maintained throughout the session of messaging service.

**8**. To ensure scalability that is multiple clients can be created and friend-list can be increased as and when required.

**9**. To check whether there is proper error handling.

**10**. To check this feature uses the desired protocols and services efficiently.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ENVIRONMENTALTOOLS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |
| --- | --- | --- |
| **TOOLS** | **SOURCE** | **DESCRIPTION** |
| **GCC** | **UBUNTU** | **Used for code compilation** |
| **GEDIT** | **UBUNTU** | **Used for coding** |
| **Microsoft office 2013** | **MICROSOFT** | **Microsoft Used for documentation, Flow charts** |
| **Google Drive** | **https://drive.google.com/#my-**  **drive** | **Microsoft Used for documentation, Flow charts** |
| **Google Scholar** | **http://scholar.google.com/** | **Used for research** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CONCLUSION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The Instant messaging chat server was developed to enable client to store its data on the server database. For providing the service of High Availability we have also implemented a back-up server. The Client to Server and then messaging service offered by the server takes place by our defined request and reply protocols. In the testing phase, we made sure to cover all the scenarios of file updating, deletion and modification.

**Contribution: -** The implementation of group chat and a part of redundancy was my contribution in this project.