Internetworking Protocols, Fall 2022

Internet Relay Chat

Team Members

Pratyusha Sanapathi (PSU ID: 949974649)

Soham Nandi (PSU ID: 930090452)

Vuddagiri Naga Venkata Sai Tejesh (PSU ID: 948734291)

Abstract

In this project, we will implement the internet relay chat. A single IRC server is created, and people connect to it using an IRC client. This protocol allows users to set a username on the server and communicate in private or group chats over various IRC channels. This is our first semester, and we want to start small, so the focus is chat functionality. The ambitious future of this project is chat and voice functionality with secure messaging.

Table of Contents

1. Introduction

This RFC describes a protocol for multiple clients to communicate with each other. I have a server that sends and receives messages from various users, and there are users who can create rooms, join rooms, leave rooms, and send personal messages to other users. If a group of users are part of a particular room, messages sent by one user will be received by all users in the room.

1.1 Sever

IRC, as defined in FC 1459, has a multi-server architecture, but a single server is used as the pillar of this IRC, providing a common connection point for all clients. This is done for simplicity.

1.2 Client

As mentioned earlier, this is a single server architecture. Therefore, all users connect to a single server that relays messages. Customer names can be up to 20 alphanumeric characters plus underscores, which are used to identify the customer.

1.3 Room Hall

Users can create rooms or join specific rooms. Created by the first user to join the channel. A room is typically a group where multiple users can communicate with each other. Users can join a room by simply referencing their room name. A room ceases to exist when all users have left it.

The basic functionality of rooms is when a user tries to join a room with the $join room_name command. If no room with the specified name is available, a new room with that name will be created. This room is now visible or available to all other users. Other users can use the $list command to see the list of available rooms.

2. Basic Information

All communication described in this protocol is over TCP/IP and the server listens for connections. Clients connect to this port to maintain a persistent connection to the server. Clients can send messages and requests to the server through this open channel, and the server can respond through the same channel. This messaging protocol is asynchronousin nature. The client can send messages to the server at any time, and the server can asynchronously return messages to the client. Both the server and client can terminate the connection at any time for any reason. You can choose to send an error message to the other party informing them of the reason for the disconnection. Depending on the implementation and the resources of the host system, the server may decide to allow only a limited number of users and rooms. In my current project, the client connection limit is 6.

3. Message Infrastructure

The client sends the command to execute in the form command argument1 argument2.where command is her four-letter word in any format (uppercase, lowercase, or a combination of both). In case of error, the beginning of the reply message is consideredwhen adding the user is not perfect. However, this simple system does not require an error code. The server just sends the error her message or response as a string back to the client. For each message they send the client and server sides, buffer sides will havea maximum size of 1024 bytes.

4. Label Semantics

- User and room identification includes sending and receiving labels. All label rules are the same and should be validated as follows:
- Must be at least 1 character, and at most in 100k input size.
- Must consist entirely of readable UTF (Unicode Transformation Format) character values.

5. Client Messages

On the client side, the user will submit certain commands to the server along with the arguments.

5.1 Welcome client

At the beginning, user needs to give his user ID to enter. Then the user will be welcomed.

5.2 Create a room

There are no free rooms at first. The first user to start this application can create a room using the $join room name command. This created room is now available for all other users to join and initiate communication.

5.3 Join a room

A user can view the list of available rooms using the "$list" command, which shows all rooms created by other users, A user can join any of the listed rooms using the “$join room_name” command, also get a list of people in that room.

5.4 Leave a room

A user can leave a room at any time using the $leave room_name command. The room is still available because there are other users in the room. Otherwise, if the room has no users, the room ceases to exist.

6. Specification

6.1 Mode of Communication

The application supports one-to-one, one-to-many, and one-to-all modes of communication via rooms and personal chats. When a user sends a message in a room, the server receives it and forwards it to all users who are part of that room except the sender. This is like dynamic multicast.

6.2 Character Codes

Each message can consist of any number of characters from the ASCII character set. Spaces act as delimiter here.

6.3 Messages

The important parts of each message are the command's name, transaction ID, and payload. All these parts are separated by a delimiter which is a space. Here, the transaction ID, an unsigned integer, is incremented by 1 for each message sent by the client. However, it remains 0 for server-to-client messages. This ID helps track customer steps.

6.3.1. Intents

1.$listroom :- To list all the rooms
2.$listusers :- To list all the users
3.$quit :- To quit
4.$help :- To list all the commands
5.$leave :-To leave the room
6.$join roomname :- To join or create the room
7.$switch roomname :- To switch room
8.$personal name message :- To send personal message

6.4 Replies

Almost all messages sent to the server generate a response. Numerical answers are the most common. Consists of a transaction ID and the status of the request. This represents O if the request was successful, and all errors are non-zero. This request cannot be generated by the client. When the server receives such a message, it generates an error response.

7. IRC Concepts

7.1 One-to-one Communication

Here, communication is only between two hosts. This can take the form of face-to-face chat where the sender and recipient are part of a room or even a client request to the server, where the chat is not visible to other clients. The client first sends that message to the server, and the server forwards it to the client that needs to receive it. This is done using the $personal user_name command

7.2 One-to-many Communication

This communication occurs when a room is created, and multiple users join the room and initiate communication. When a user of this room sends a message in this room, this message will be visible to all other users who are also part of this room.

7.3 One-to-all Communication

Clients can send broadcast messages that are delivered to individual clients and servers.

8. Server Messages

As per the client input server will respond accordingly. Suppose, client have put the room ID wrong, then server will show output/popup/ notification as " invalid roomID".
Some sample server messages are given below:

Invalid_roomID -> " Invalid roomID, please check"
Invalid_userID -> " user doesn't exist"
Room_Created -> "new room got created"
Client_Left -> " This member left the rrom"

9. Error Handling

The first error we might encounter when user will try to join using same username. At that time it will not allow and throw a error as “The username already exists”.

After that if there is any space in username, it will not allow to register and the response will be “ Username should not contain any spaces”.

Moreover is any username consist of more than 10 characters, it will deny the registration and throw the error mentioning “Username can’t have maximum length greater than 10 characters”.

10. Future Work

This specification provides a generic message-passing framework for allowing multiple clients to communicate with each other through a central transport server. In addition to that private messaging, secure messaging, file transfer, and implementation on cloud servers can be completed in the future.

11. Security Considerations

Messages sent using these systems are not protected from inspection, manipulation, or outright forgery. The sever does not acknowledge all messages sent using this service.

12. IANA Considerations

None.

9.1 Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

13. Conclusion

Message sharing extends to a single server and multiple clients in the form of either broadcast or private messages. Interaction between different rooms has been implemented. This application can be further enhanced by implementing several security features and media sharing options. Security features include eliminating room username and password redundancy. Media sharing may be enabled using encrypted transport protocols.