CS 594 FALL 2022

Nov 29, 2022

Portland State University

CS 594: Internetworking Protocol
Internet Relay Chat Project

By: Shrikrishna Bhat and Vishrut Sharma

TABLE OF CONTENTS

1. Introduction

This RFC represents a protocol in which multiple clients communicate with each other. There is a server that sends and receives messages from different users and there are users who can create rooms, join rooms, leave rooms, and can also send personal messages to the other users. If a group of users is a part of a particular room, any message sent by one user is received by all the users in the room.

1.1 Server

Although the IRC defined in RFC 1459 has multiple server architecture, to provide a common point for all the clients to connect, a single server is used as the pillar for this IRC. This is done to achieve simplicity.

1.2 Client

As mentioned above, this is a single-server architecture. Therefore, all the users connect to a single server that relays messages. The client can have a name length of up to 20 alphanumeric characters along with an underscore which is used for the identification of the client.

1.3 Rooms:

A user can either create a room or join a particular room. It is created by the user who joins the channel first. A room is typically a group where multiple users can communicate with each other. A user can join a room just by referring to its room name. If all the users in the room leave, the room no longer exists.

The basic functionality of a room is that when a user tries to join a room using the "**<join> room_name**" command. If the room with the name mentioned is not available, then a new room is created with that name. Now, this room is visible or available for all the other users to join. Other users can see the list of rooms available using the **<list>** command.

2. Specification

2.1 Mode of Communication

This application supports One to One, Many, and One to All modes of communication through rooms and personal chats. When a user sends a message in the room, the server receives it and forwards it to all the users who are part of that room excluding the sender. This resembles dynamic Multicasting.

2.2 Character Codes

Each message may consist of any number of characters from the ASCII character set. Here the space acts as a delimiter.

2.3 Messages

The essential parts of any message are the command's name, transaction ID, and payload.
All these parts are separated by the delimiter which is the space. Here is the transaction ID which is an unsigned integer increment by 1 whenever a message is sent by the client. However, it remains 0 for the server-to-client messages. This ID might be useful for tracing the client's steps.

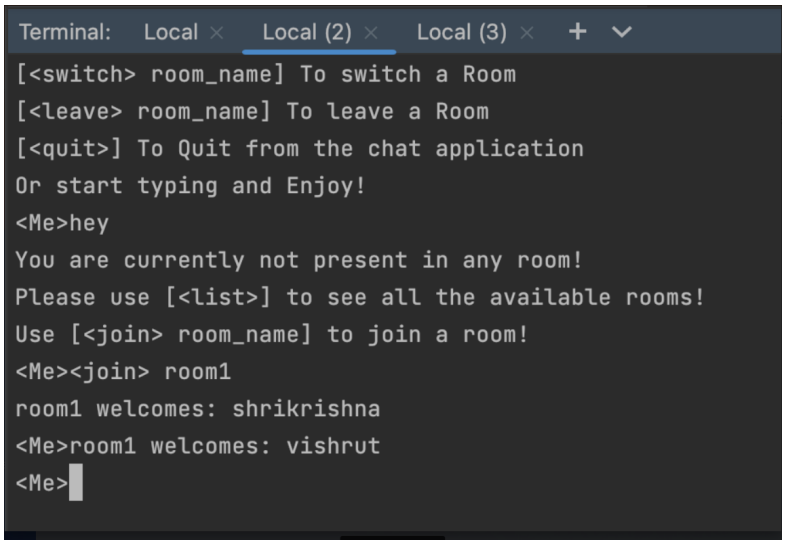
2.4 Replies

Almost all the messages sent to the server generate a reply. The numeric reply is the most common. It consists of the transaction ID and status of the request. It represents a 0 for a successful request and a non-zero for any errors. A client cannot generate this request.
Any such messages received by a server will generate an error response.

3. Message Infrastructure

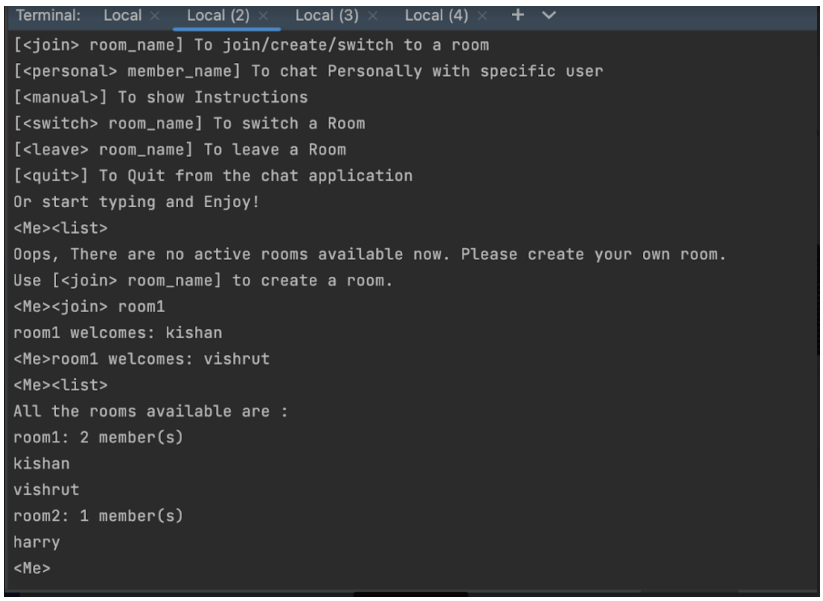
3.1 Create room

Initially, there are no rooms available. The first user to start this application can create a room using the "**<join> room_name**" command. This created room will now be visible to all the other users to join and start communicating as seen in the image below.



3.2 Join room

A user can see the list of rooms available to join using the "**<list>**" command which displays all the rooms created by other users. Users can join any of the listed rooms using the "**<join> room_name**" command. We also get the list of people in that room using the <list> command.



3.3 Leave room

A user can leave the room at any moment of the time using the "**<leave> room_name**" command. The room will still be available as there are other users in the room. Otherwise, if there are no users in the room, the room does not exist anymore.

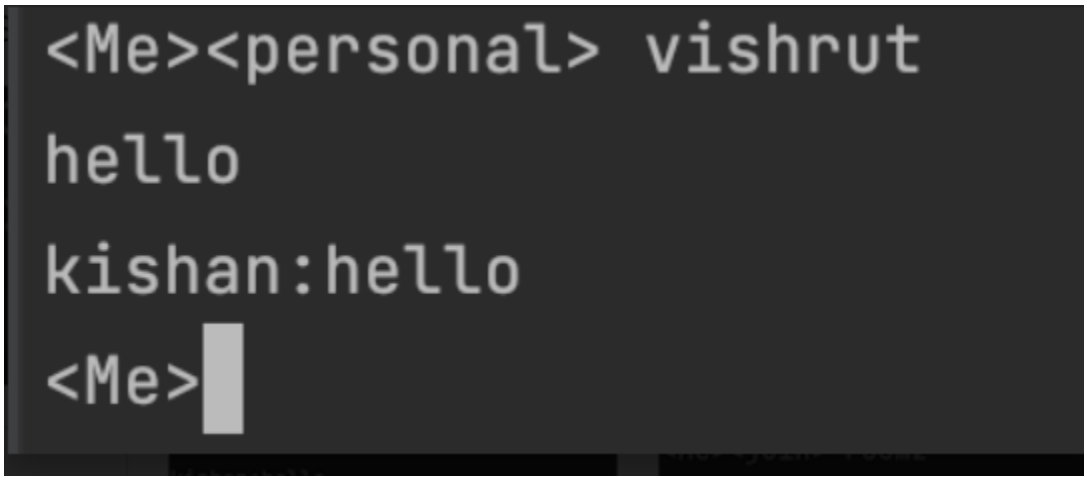
```
<Me>hello
kishan:hello
<Me><leave> room1

You are currently not present in any room!
Please use [<list>] to see all the available rooms!
Use [<join> room_name] to join a room!
<Me>
```

3.4 Switch a room

A user if joined 2 rooms can switch between the rooms and can message only to a particular room using the "**<switch> room_name"** command

3.5 Personal Message

A user can send a personal message to a specific person outside of the chatroom using the "**<personal> user_name**" command.



3.6 Manual

The **<manual>** command is present to show the instructions which help the users to know the commands available.

```
<Me><manual>
Instructions:
[<list>] To list all the available rooms
[<join> room_name] To join/create/switch to a room
[<personal> member_name] To chat Personally with specific user
[<manual>] To show Instructions
[<switch> room_name] To switch a Room
[<leave> room_name] To leave a Room
[<quit>] To Quit from the chat application
Or start typing and Enjoy!
<Me>
```

3.7 Quit the chat application

The **<quit>** command is used to exit from the chat application and this is used to directly exit from the server. In this scenario, all the other users are notified when the user exits the chat application. Even the server gets to know when the user leaves the chat application.

```
<Me><quit>
Exiting the Server
(base) shrikrishnabhat@MacBook-Pro IPProjectModified %
```

4. IRC Concepts

4.1 One-to-one communication

Here the communication takes place only between two hosts. This might be in the form of a personal chat where the chat is not visible to any other client even if the sender and receiver are a part of the room or the client's request to a server. The client first sends its message to the server which then redirects the message to the client that is supposed to receive that message. This is achieved by using the "$personal user_name" command

4.2 One-to-many communication

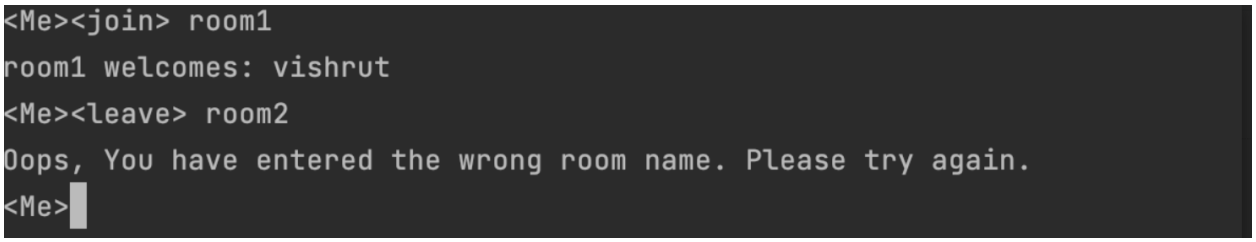
This communication takes place when a room is created and multiple users join that room and start communicating. When a user of that room sends a message, that message is viewed by all the other users who are also a part of that room.

4.3 One-to-all communication

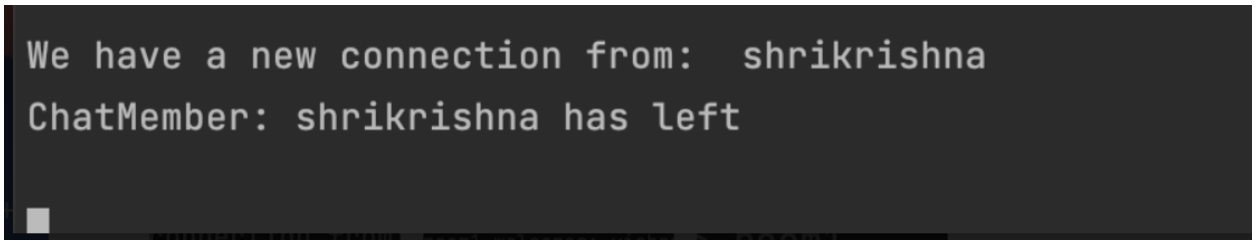
A client can send a broadcast message that will be delivered to every client and server.

5. Error Handling

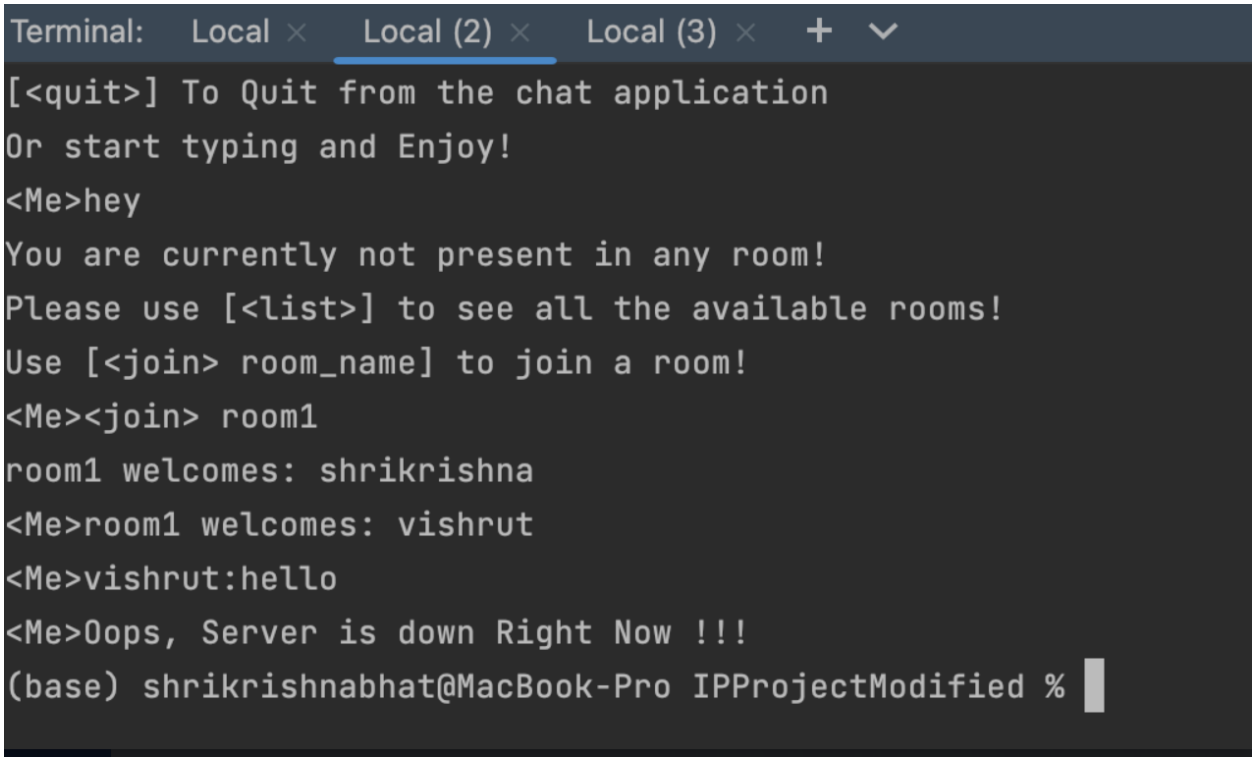
a) If a user tries to leave a room that he is not a part of, the “Oops, You have entered the wrong room name. Please try again.” message will be displayed.



b) Suppose a client crashes, then servers, rooms, and the client are notified with the message “user_name has left the chat”

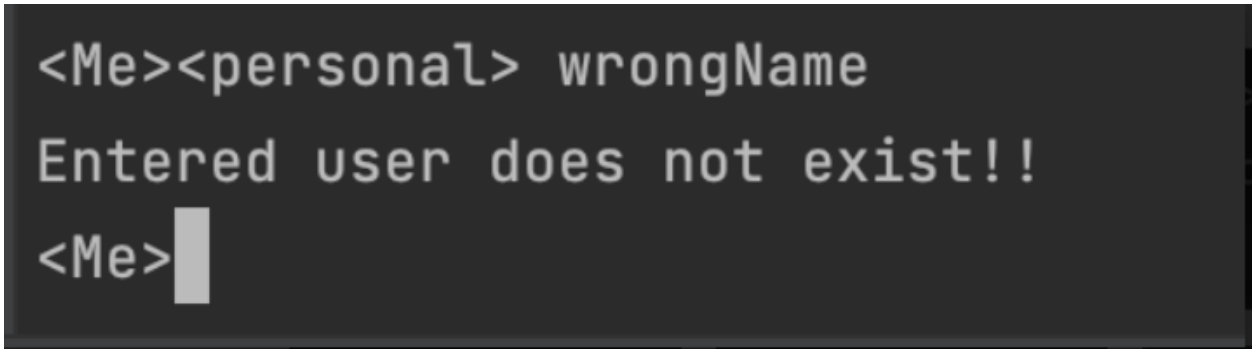


c) If the server crashes, the clients are logged out and will be notified with an “Oops, Server is down Right Now” message.

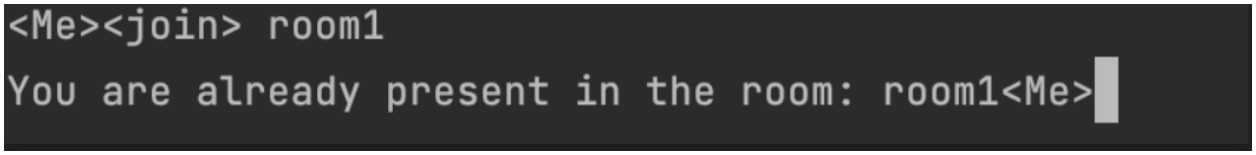


d) Users who are not a part of a particular room cannot send messages in that room.

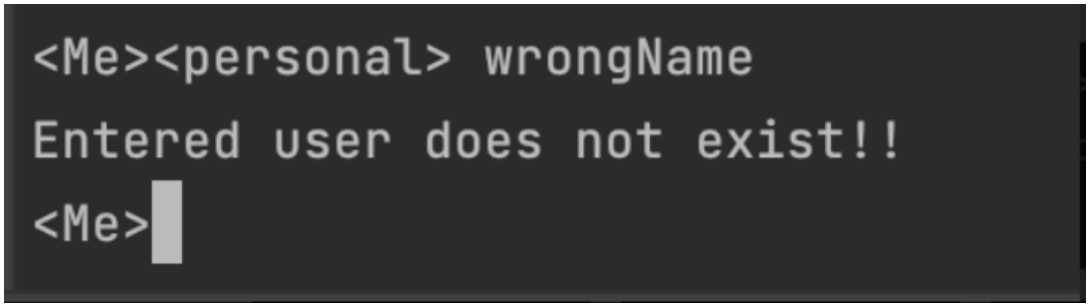
e) A client cannot communicate with a client who does not exist.



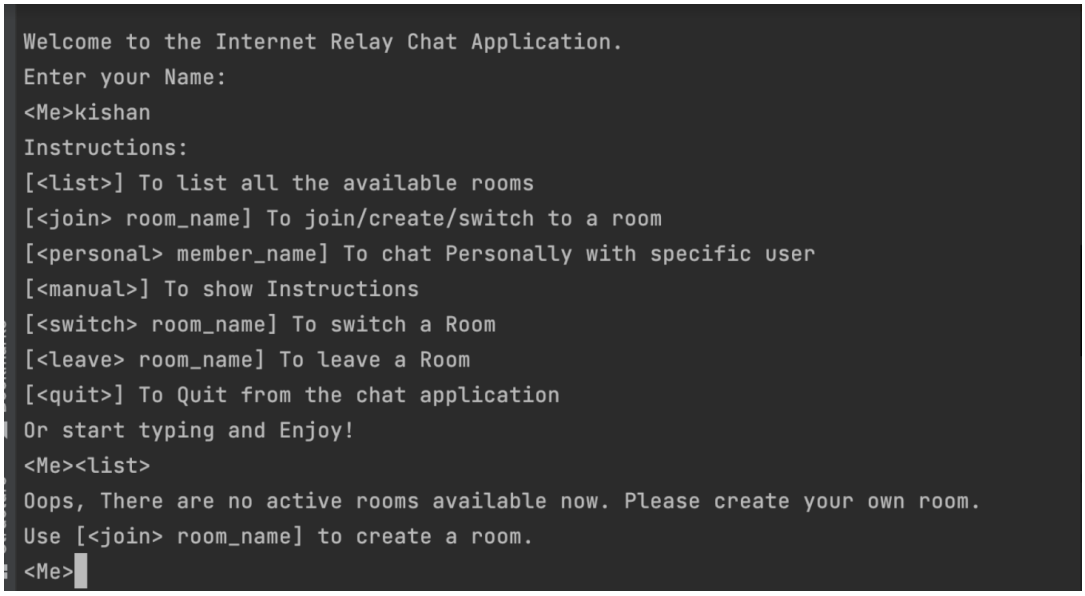
f) If a user is present in a room and tries to join the same room, the error message “You are already present in the room” is displayed.



g) When sending a personal message to a specific user using <personal> command and the user is not present the error message - “Entered user does not exist!!” is displayed.



f) When a user checks the list of available rooms when no active rooms are available an error is thrown as seen in the image below.



6. Conclusion & Future Work

Message sharing has been extended to single servers and multiple clients either in the form of broadcast messages or private messages. The interaction between different rooms has been implemented. This application can be furthermore extended by implementing a few security features and media-sharing options.

Security features include eliminating the redundancy of the user’s name and passwords for the rooms. Media sharing can be possible with cryptographic transport protocols.