Internet Relay Chat Protocol

Internet Draft Harrison
Huynh
CS494p PSU
RFC Document 3/16/2024

Memo Status

Abstract

This project, assigned within the scope of the CS494 Network Protocol course, aims to develop and showcase a functional IRC (Internet Relay Chat) protocol implementation, encompassing both IRC server and client components. The implementation utilizes any programming language supporting socket API, with a specific focus on Python in this instance.

IRC operates as a message-based protocol facilitating communication among multiple IRC clients by connecting to a local host IRC server via socket connections. The implementation of this IRC protocol includes support for various chatting functionalities, including creating a room, joining a room, sending a message to room. It also supports joining multiple rooms, sending multiple messages, and

Table of Contents

1. Introduction

This specification outlines a straightforward Internet Relay Chat (IRC) protocol enabling communication between clients. The system operates with a central server acting as an intermediary, facilitating message transmission among connected users.

Users/clients possess the capability to create, list, join, and leave rooms. Messages dispatched to a room are related to all users currently present in that room.

2. Basic Information

This code implements a rudimentary Internet Relay Chat (IRC) system, featuring both server and client functionalities. The server component initializes with local host and port settings (host='127.0.0.1', port=6667), capable of handling up to 5 client connections concurrently through threading. Clients can execute various commands, including joining, leaving, and creating rooms within the chat system. Additionally, clients have the ability to send messages to rooms, access message threads within a room, and view the list of members currently present in a room.

The server incorporates a graceful shutdown mechanism to manage interruptions seamlessly. On the client side, users establish connections to the server using designated host and port configurations. They can then interact with the chat system by issuing commands such as creating, joining, and leaving rooms, as well as sending and viewing messages. A help command is available to provide users with information about the available commands. The client component effectively handles server responses and timeouts, ensuring smooth communication within the IRC environment.

3. Message Infrastructure

3.1 Message Format

Messages exchanged between the server and clients adhere to a specific format to ensure seamless communication. The format includes:

Command: Specifies the action to be executed by the recipient.
Parameters: Additional information required for executing the command, such as room names, or message content.
Delimiter: A designated character or sequence used to separate different components of the message, facilitating parsing.
Example: JOIN room1

3.2 Error Messages

In cases where an error occurs during message processing or command execution, the system employs error messages to indicate the nature of the issue. Each error messages corresponds to a specific error condition, aiding in troubleshooting and resolution. Some common error codes include:

Confirmation: Indicates successful execution of the requested action, providing relevant details or acknowledgments.
Notification: Informs the client about specific events or changes within the system.
Error Notification: Alerts the client about encountered errors, including error codes and descriptions to facilitate understanding and resolution.
Example (Confirmation): You have successfully joined the room: room1

Example (Error Notification): Room not found

3.3 Response Message

Upon receiving and processing a command, the server generates response messages to communicate outcomes or provide feedback to the client. Response messages include:

Confirmation: Indicates successful execution of the requested action, providing relevant details or acknowledgments.
Notification: Informs the client about specific events or changes within the system.
Error Notification: Alerts the client about encountered errors, including error messages and descriptions to facilitate understanding and resolution.
Example (Confirmation): You have successfully joined the room: room1

Example (Error Notification): Room not found

4. Client Messages:

4.1 Client Connect

Upon establishing a connection with the server, the client will receive a confirmation of successful connection. Subsequently, it will prompt the user to provide their username to associate the client socket with the username and their outgoing messages. The client will continuously prompt the user for commands to execute, and in the event of an incorrect command, it will notify the user accordingly.

4.2 Help Message

Command HELP:

The command will print out all the commands that are available for the user to use.

4.3 Create Rooms

Command CREA:

The "CREA" command in the IRC protocol implementation allows users to create a new chat room within the system. When a user issues the "CREA" command followed by a room name, the server processes the request and generates a new room with the specified name. If the room name is unique and not already in use, the server creates the room and notifies the user of successful creation. However, if the room name is already in use, the server returns an error message indicating that the room already exists. This command provides users with the capability to establish new communication spaces tailored to their needs within the IRC network.

Example: CREA room1

Error messages: Room already exists

4.4 Join a Room

Command: JOIN

User inputs the "JOIN" command followed by a room name, the client will send this command to the server. If the room exists, the client will join the specified room and receive confirmation. However, if the room does not exist or the command format is incorrect, the client will notify the user accordingly. It can also handle joining multiple rooms if the user inputs different room names in the command line:

Example: JOIN room1 room2

Error Messages: You are already in the room: {room_name}, Room doesn't exist: {room_name}

4.5 Listing all Rooms

Command: LIST

User inputs the "LIST" command, the client will send this command to the server. Upon receiving the command, the server will provide a list of available rooms to the client. However, if no rooms exist or if there is an issue with the command format, the client will inform the user accordingly.

Error Message: No rooms have been created

4.6 Sending a Message to a Room

Command: MESG

User inputs the "MESG" command followed by a room name and a message, the client will send this command along with the room name and message to the server. If the room exists and the message format is correct, the client will receive confirmation of successful message transmission. However, if the room does not exist or there is an issue with the command format, the client will notify the user accordingly. Also if the user is not in the room it will ask them to join first.

Example: MESG room1 user msg

Error Messages: You are not in the room {room_name}. Cannot send message, Room doesn't exist or incorrect format

4.7 Receiving Messages

Command: MSGB

Command: MSGB

The client will send this command to the server with a room name, which will respond with the message history of the specified room. The client will display this message history to the user. If the input doesn’t match any of the rooms then the server will notify the client with an error message

Example: MSGB room1

Error Messages: No messages in this room, Room doesn't exist: {room_name}

4.8 Sending Distinct Messages to Multiple Rooms

Command: DMSG

The client will ask the user for each room they want to send a message to. Then it will loop and ask the user for each message they want to send to each different room. The client will send the parameters using the MESG command. However, if the specified room does not exist, there is an issue with the command format, or the user is not in the room the client will notify the user accordingly.

Example: First DSMG, then MESG room1 user msg

Error Messages: You are not in the room {room_name}. Cannot send message, Room doesn't exist or incorrect format

4.9 List Members in a Room

Command: MEMB

If the user inputs the "MEMB" command followed by a room name, the client will send this command to the server. The server will respond with a list of members present in the specified room or will respond with “No members in this room”. The client will then display this list to the user. However, if the specified room does not exist or there is an issue with the command format, the client will notify the user accordingly.

Example: MEMB room1

Error Messages: Room doesn't exist

4.10 Leaving a Room

Command PART

If the user inputs the "PART" command followed by a room name, the client will send this command to the server, indicating the user's intention to leave the specified room. Upon receiving confirmation from the server, the client will notify the user of successful departure. However, if the specified room does not exist or if the user is not in the room, the client will notify the user accordingly.

Example: PART room1

Error messages: You are not in the room: {room_name}, Room doesn't exist: {room_name}

4.11 Disconnecting from Server

Command DISC

If the client sends “DISC” then it will disconnect from the server leaving all rooms they were originally in. Quit the program.

Error Messages: Client not found

5. Server Messages

5.1 Bad Command

If a command was sent that was not in the available commands then it will send the user “Bad command”

5.2 Server Quit

Just close the terminal to quit the server

6. Error Handling

The client can detect if the server crashes if it does it exits the program. If the client crashes from the server. The server will catch the error and remove the client from any rooms they were in.

7. Acknowledgments

This document was prepared using Google Doc by Harrison Huynh
Email: huynhhar@pdx.edu