CS594 Gauri Kasar
Request For Comments Shruti Deshmukh
Internet Relay Chat Project Specification Portland State University
29 November 2022

Internet Relay Chat CS594 class project

Abstract

This document is written to define a IRC protocol used to perform text based communication between client/server. It is designed as a part of a final project for Internetworking protocol class. The two important components within this protocol are client and server. The server resides at the centre while multiple clients are connected to each other via server. The client and server need some commands in order to communicate with each other.

Status of this memo

This is a project for the Internetworking Protocol subject. ALl the practices or messages used in this project used here are based on understanding of concepts in this course.This document is published for examination and review purpose.

Table of contents

1 Introduction

1.1 Servers

1.2 Clients

1.3 Rooms

2 Messaging structure

2.1 Label Semantics

3 Commands to explore application features

3.1 Server Connection

3.2 Client Connection

3.3 Create room

3.4 Join room

3.5 Leave room

3.6 Get list of all room members

3.7 Get list of all the available rooms

3.8 Get list of all available users

3.9 Send a common message to all room members

3.10 Send a private message to a client

3.11 Send a broadcast message to all the clients

3.12 Send a secure message to a client

3.13 Decrypt an encrypted message

3.14 Server Disconnection

3.15 Get Help

3.16 Exit out of the application

4 Server Messages

5 Error Handling

6 Security

7 IANA Considerations

1 Introduction

This specification has a detailed description of the IRC protocol that is used for text based communication. This is developed based on TCP/IP concepts. There is a connection between multiple clients through a server. They can create a room or send secure messages or private messages and many other functions.

1.1 Servers

This provides connection to multiple clients hosted on a particular port. It has an option to start port on user defined port or it gets started on the default port 1002.

Whenever a client connects the port as mentioned it gets listed on the server window. Also when it creates a room server keeps track of all commands that connected clients execute.

1.2 Clients

Multiple or single threads that try to connect to a port by either specifying it same as server, if they don't mention anything the port starts at default 1002. Once they start the connection these clients can enter their desired username and now this is used to keep a track of messages and threads.

1.3 Rooms

Groups can be created of more than one client where messages can be broadcasted in this room. Clients can create a new room. Client creating the room will automatically be a part of that room. Other clients are able to view all the available rooms and can join any of the rooms specified by using a command. Clients can also leave a room, enlist the members of the room and also send a common message to all the members of the room.

2 Messaging structure

This application is of asynchronous type. The message structure consists of three parts <command> some prefix (optional depending on the command) and message parameters. We make use of a simple format to create this communication. For example if a client wants to join room 1 he would simply type “room join 1” in the command line and he can join that room.

2.1 Label Semantics

Everytime user enters a command it is processed by a client thread using a set of conditions written for each operation and it performs those tasks. There is also error handling done in case of server crash or if a peer leaves the group/ room.

Eg: broadcast <message>

Everytime client starts running on the same port as the server it basically starts a thread. This thread takes care of reading the message and then it uses steams to write the message and send it to the rest of the clients(in case of this broadcast message). Now here in this example it is broadcast so it will read this message. Server keeps track of total clients connected through an array that has a number of clients connected at that instance.Server takes the part of notifying every connected client about the newly joined client. Every time a new client joins it starts executing the code on that thread. It reads this message and takes the content of the message and writes in the output stream of all the connected client threads. And this is how all the connected clients get the broadcasted message.

3 Commands to explore application features

3.1 Server Connection

To initiate a connection between client and server we need to start the server. This can be done either using an IDE or through the command line. For IDE, simply run the server. For the command line we need to first compile and then run the server file by using “javac ChatServer.java” and “java ChatServer” respectively.

3.2 Client Connection

Similar to a server to establish client connection using the command line we need to use the commands, “javac ChatServer.java” and “java ChatServer” respectively.

3.3 Create room

After the connection is established, the client can create a new room. The command to create a new room is “room create

<room_name>”. The client who creates a room becomes a part of the room and does not have to join the room for it.

3.4 Join room

Once the room is created, other clients can join the room. The user needs to type in the command “room join <room_name>” to join the room. The client who joined the room and other clients who are already the members of that room receive the confirmation message.

3.5 Leave room

The client which has entered the room or rooms can leave a particular room by using the command “room leave <room_name>”. The client who left the room and other room members receive the message of the client leaving the room.

3.6 Get list of all room members

The client can enlist the list of all the room members of a particular room. To enlist the room members the client has to use the command “room members <room_name>”.

3.7 Get list of all the available rooms

The client can see all the rooms that are available by using the command “room names”. This helps clients to decide and join a required room.

3.8 Get list of all available users

The client can also enlist all the clients who are connected to the IRC application. The client has to type in the command “enlist_users” to get the list of all the users.

3.9 Send a common message to all room members

Clients can send a message to all the room members of the room of which the client is a part of. The command to send a message is “room chat <room_name> <type message>. The message along with the sender’s name is received by all the room members.

3.10 Send a private message to a client

The client can send a personal message to a client. The client has to use the command “pvt_msg @clientname <type message> to send a private message. On receiving the message also knows from whom the message was sent.

3.11 Send a broadcast message to all the clients

To send a broadcast message a client has to type in the command “pub_msg <type message>”. All the clients connected to the server will receive the message and also will know which client has sent the message.

3.12 Send a secure message to a client

To send a secured message the client has to send a message along with the key to decrypt the message. The message sent by the client is encrypted and then sent to the other client. The

command to send the secured message is “secure <key> @clientname <type message>. The receiver receives an encrypted message.

3.13 Decrypt an encrypted message

The client who has received an encrypted message needs to know the key to decrypt the message. The client can use the key and decrypt the message by using the command “decrypt <key>”. The client will then be able to see a decryption. If the client again tries to decrypt the message or if the client who did not receive an encrypted message tries to decrypt the message will receive a message saying “No message available to decrypt”.

3.14 Server Disconnection

If the server disconnects for some reason each client gets a message saying “Server is not responding please try after sometime or close the application with ctrl c”. After the server disconnection the client cannot communicate with other clients.

3.15 Get Help

The client at any point can type in the command “help” to know about the command and their usage.

3.16 Exit out of the application

The client can exit the IRC application by typing in the command “\quit”.

4 Server Messages

On the server side messages related to which client joins the room, total active connected clients on the port . Once the client enters /quit and bye the server displays on the commandline that “<username> has left the chatroom”. Other connected clients are also notified.

5 Error Handling

Unexpected server crash or server not responding exception is handled in this project. Every client gets notified that the server is not responding and to connect after some time. Similarly, client crash is also handled here. If the client unexpectedly closes the application it notifies other clients.For eg. If users gauri , shruti, John ,Josh are connected and lets say they all have joined a room and John’s application crashes because of some reason, then all other users get a message saying John has left the chat room. The List of errors handled are specified below:

Error messages:

- LIMITED_USERS : When the number of clients connected are more than 20.
- SERVER_NOT_RESPONDING : Server is not responding please try after sometime.
- CLIENT_CRASH : Closing everything...

6 Security

If the client specifies sending a message as “secure <password> @clientName <message>” this would send a secure message to the designated client. The client at the receiving end would decrypt the message using decrypt <password>. This mechanism is implemented using the AES-SHA-256 algorithm. For the other messages there is no such security implementation for message encryption. Every message that client sends is read by the server. There is no such separate implementation for message encryption.

7 IANA Considerations

None.