CS 594 — Sowmya Sri Indukuri & Nishna Reddy Aleti
Internet Draft — Portland State University
Interworking Protocols – IRC Project Specification — May 31, 2020

Internet Relay Chat

Status of this Memo

Copyright Notice,

Abstract

In this memo we will go through the implementation details of Internet Relay Chat (IRC) which uses client-server architecture. This document describes about the working of this application and IRC protocol.

Table of Contents

1. Introduction

Internet Relay Chat (IRC) is an application layer protocol that provides a way of communicating with people from all over the world in the real time. The IRC Protocol is based on the client-server model, and is well suited to running on many machines in a distributed fashion. In this project, the setup involves a single server forming a central point for clients and perform the required message delivery and other functions. When the IRC Clients are connected to the server they can perform few actions like creating room, leaving a room, joining a room, private messages etc. The means of communication to a group of users connected to a room is through channels. Channels are the virtual rooms for communication. Channels on a network can be displayed using the IRC command listrooms. While the server captures the conversations on each channel and transfer the message to every other client connected to that chat room.

2. Conventions used in this document

Values in this code piece presented in this document, that are dynamically allocated or hard-coded will be enclosed within < >.

3. Basic Information:

The main components of IRC protocol are client, server, channel.

3.1 Server:

The server forms the backbone of IRC as it is the only component of the protocol which is able to link all the other components together. Each server is uniquely defined by a name. Each server knows every other server. The server, when it receives message identifies its source using the prefix. It offers the client to connect each other and a point where other servers can connect. Servers have client to client, server to server and server to server connections. And when the server quits or crashes, clients can no longer communicate with each other.

3.2 Client:

Any number of clients can connect to the IRC sever via socket on a specified port number. Program generated unique random names for client. Here the communication between clients and server is asynchronous. Each client has opportunity to perform actions as mentioned above in introduction section.

3.3 Channel

The channel is a group of users that gets messages intended to that particular channel. The name and its current members are describing factors of a channel. The channels on the network can be known using command LIST. There are various modes of a channel like secret and private channels. The channel modes define the properties of each channel and modes could be manipulated by the channel members.

4. Commands:

When the client makes a connection with server, server prints a welcome message in the client with the client name and a message in the server saying the a particular client is connected

4.1 CREATEROOM - For creating a room, CREATE command is used

Command: createroom <ROOMNAME/>

With the above command client creates room and the server responds with message “Room <ROOMNAME/>created” to the client who created the room. At server, “Room <ROOMNAME/> created by <CLIENTNAME/>” will be printed. When client tries to create an existing room, then error will be thrown as “Room already exits”

4.2 JOINROOM – To allow the client to join in a room

Command: joinroom <ROOMNAME/>

For above command, server returns a message “joined to room <ROOMNAME/>” to the client who joined and with message “<CLIENTNAME/> joined the room <ROOMNAME/>” to clients already in the room. At server, “<CLIENTNAME/> joined room <ROOMNAME/>” will be seen. When client tries to join any non-existing rooms, the error will be thrown to client as “Room <ROOMNAME/> doesn’t exist”. Client can join in any number of rooms.

4.3 MESSAGEROOM – Client can send message to room which will be broadcasted to all clients in that room.

Command: messageroom <ROOMNAME/> <MESSAGE/>

A client can even send messages to different rooms using <ROOMNAME/> in command in which he has joined in. The server responds with message “Message sent” to client and at server the message “<CLIENTNAME/> broadcasted the message” will be seen. If client tries to broadcast to message to room in which he is not member of, then error will be thrown to client as “You are not member of this room”.

4.4 LISTMEMBERS - List members of any room.

Command: listmembers <ROOMNAME/>

The server responds with who are members of the room with <CLIENTNAMES/>. If the client attempts to list any non-existing room members, the error will be thrown at the client as "Room < ROOMNAME/ > does not exist."

4.5 LISTROOMS: A client can request for list of existing rooms.

Command: listrooms

The server responds with list of <ROOMNAMES/>. In case if command is not provided properly, the error “Invalid Command! Please enter proper command” will be thrown.

4.6 LEAVEROOM: The client can leave any room that they are the members of.

Command: leaveroom <ROOMNAME/>

The server responds with message “you left the room <ROOMNAME/>” to the client who left the room and also sends the message “<CLIENTNAME/> left the room <ROOMNAME/>” to other client members of the same room. When client tries to leave the room in which is not member of, then error “You are not member of room <ROOMNAME/>” will be thrown.

4.7 PRIVATE: A client can send private messages to another client.

Command: private <CLIENTNAME/> <MESSAGE/>

The client who sent private message to another client receives message as “Message Delivered”. If the other client whom the client is sending private message does not exist, the error “Client <CLIENTNAME/> does not exist”.

4.8 QUIT: Client can disconnect from the server with quit command.

Command: quit

The client is disconnected from all the rooms is present and receives a message "See you! The customer receives < CLIENTNAME/ > "and the message" < CLENTNAME/ > Left "will be displayed at the server.

4.9 Server:

QUIT: The server can disconnect with all the clients.

Command: quit

“Server OFF! Try later” message is received by all the clients connected to the server.

The server crash is handled through exceptions and all clients connected to server will receive message “Server crashed. Existing to handle server crash gracefully”. “Client Disconnected!” message is received in the server when the client crashes.

5. Error Handling:

a) A client cannot communicate with a client which does not exist.

b) If a client is not a part of a chat room, then it cannot send message to that chat room.

c) The error may occur when connecting the socket link has ended or when connections to clients or servers are lost.

d) The server and client must be able to track messages on a continuous basis. When the server senses the lack of client contact, all clients in the chat room they have joined must be deleted.

e) Also, if the client detects it has lost contact to the server, it MUST know it is disconnected and can try again. Should the correct command not be given, the "Invalid Command. Please give correct command” will be thrown in all above cases.

6. Security Considerations

There are few security issues in IRC, since messages sent through connections are typically not encrypted. It is necessary to have a careful security policy that is not vulnerable to attacks. Another problem is that all messages will be seen by the server. Some third parties can easily hack even private message mode.

7. Conclusions & Future Work:

This specification offers a brief overview of different IRC modules, such as servers, clients and channels, where multiple clients may interact by directing to a service via text. To build a network, the server will communicate with one another. The communication can be terminated by different client server and channel-related commands to test the link features. Security issues arise because there is no authentication for messages sent, and the user can change and design his protocol that will encrypt the message or accomplish any more function of the user. Image and file sharing functionalities can also be added. It can be used to transfer large files of data in a secure connection using cryptographic transport protocols. Further, Transport Layer Security (TLS) protocols and Secure socket connection are exploited.

9. References:

- Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.
- 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.
- StackOverflow.com, GeeksforGeeks.com, quora.com, irchelp.org

10. Acknowledgments

To prepare this document, RFC 1459 is taken as reference.