CS 594 Belén Bustamante
Internet Draft Portland State University
Intended Status: IRC Class Project Specification November 30, 2018
Expires: April 2019

Internet Relay Chat Class Project
Protocol Draft

Status of this Memo

Copyright Notice

Abstract

This memo describes the communication protocol for an IRC-style client/server system for the Internetworking Protocols class at Portland State University.

Table of Contents

1. Introduction

This specification describes a simple Internet Relay Chat (IRC) protocol by which clients can communicate with each other. This protocol allows multiple users to communicate via text messages with each other in common “virtual” rooms.

This system utilizes a main server as a central hub for clients to connect to. The server in turn will relay a client’s messages to other users connected to the same room.

In the initial implementation the functionality of the IRC client will be limited to the following: clients will be able to create a room, join a room, leave a room, and list rooms available. Additional features may be added in the future and this document will be updated as necessary.

1.1 Server

A single server will be hosted to allow for clients to communicate with one another. It will serve as the central point of contact and the initial recipient of all messages being sent through the application. All clients will connect to the server when the application is launched. Any time a client sends a message to all users in a room, it will be sent to the server first, who will be in charge of forwarding it to anyone else connected to that particular room.

1.2 Clients

A client is any other computer connected to a server. To distinguish clients from one another, a unique nickname will be set during the initial connection set up. This nickname will appear anytime a client types a message so that other clients connected to a room can identify who sent a particular message.

1.3 Rooms

Clients will have the ability to connect to rooms. These will be specified by a unique name and are public to all clients. A room will be comprised of one or more clients. Any client to connected to a specific room will be able to send and receive messages to and from any other clients also connected to that room.

2. IRC Concepts

This section will include a description of actual concepts behind the IRC protocol and how the current implementations deliver messages to clients.

2.1 One-to-one Communication

All one-to-one communication will be between clients. This is the case since there is only one server available to IRC system.

Example:

A message from client A to client B is seen by server S, which sends it directly to client B.

2.2 One-to-many Communication

The main goal of the IRC system is to provide the ability for multiple clients to communicate with one another. In this implementation, the server will relay the messages from one client to any other clients that are connected to the same room.

Example:

Clients A, B, C are connected to room 1. Client D is connected to room 2. A message sent by client A is only received by clients B and C.

3. Message Details

There will be different types of client messages: some to be recognized and processed only by the server, and others to be forwarded on to other clients. Those that are to be seen only the server will be known as “server commands” and are to be in capitalized letters. Those that are to be forwarded to other clients that are members of the same room as the client originating the message, are to be known as “client messages”. These have a regular text form and can include capitalized or lowercase letters, and must be preceded by the SEND command.

Example:

Server command: COMMAND
Client message: SEND recipient this is a client message

3.1 Setting up a Connection

To set up a connection, the client first has to use the terminal and the telnet application in order to connect to a server. Though this will initiate a connection between the client and the central relaying server, the client will not be able to send messages until they join a room or select a specific recipient also connected to the server. Connections to the server will be handled through port 6667, which is customary for IRC connections.

A list of available commands will be displayed to the client after a successful connection to the server has been made.

Example:
telnet *server_ip port_number*
telnet 192.168.1.7 6667

3.2 Nickname

After a connection has been established with the server, the client will then be prompted for their nickname. This is the identifying name that will be visible to all other clients participating in a chat next to their messages.

Example:
<nickname_here> : this is a message from nickname_here

3.3 Create/Join a Room

When a user indicates they'd like to join a room, if one by that name does not yet exist, it will be created. Otherwise, they will be added to the list of users in that room. To join a room, a client must enter the server command JOIN along with a @room_name. The server will display a message confirming the client has been successfully added to their specified room.

Example:
<nickname_here> : JOIN @room1
<server> : You have joined "@room1".

3.5 Leave a Room

To leave a room after a client has joined it, all the client has to do is type the command LEAVE @room_name. This will remove the client from the specified room and they will then be unable to send or receive messages to the clients still connected to that room. To be able to send messages again, the client would need to join a room again.

Example:
<nickname_here> : LEAVE @room1
<server> : You have left @room1.

3.6 List all Rooms

To view all existing rooms, a client can type the command ROOMS. A list of any rooms that have been created will be displayed.

Example:
<nickname_here> : ROOMS
<server> : [@room1, @room2]

If no rooms have been created, the server will output the following message:
<server> : “There are no rooms”

3.7 Send a Private Message

To send a private message, a client must first establish a connection to the server. Clients can specify the recipient through the SEND command by adding their nickname. The server will take care of distributing the message to the intended recipient.

Example:
<nickname_here> : SEND *recipient_nick message*
<nickname_here> : SEND nick1 hello

3.8 Send a Public Message

To send a public message, a client must first establish a connection to the server and join a room. Clients can join an existing room or can create a new room if they would like. To send a public message to all users in a room once they’ve joined it, a client can specify the room through the SEND command by adding the room name preceded by the “@” symbol. The server will take care of distributing the message to any other clients in the room.

Example:
<nickname_here> : SEND *@recipient_room message*
<nickname_here> : SEND @room1 hello

3.9 Show Commands

To see all available commands, the user can use the help command. By doing so, the server will display all commands and their proper usage.

Example:
<nickname_here> : HELP

3.10 List Users in a Room

To see all users that have connected to a room, a client can utilize the USERS command followed by the room name. In doing so, the server will display all users that have joined that room, or if none have done so display a message indicating that the room is empty.

Example:
<nickname_here> : USERS @room1

3.11 End a Connection

To terminate a connection to the server, a client may at any point type the command QUIT. This command will immediately end that client's connection to the central server and leave any room they have joined. As a result, this client will no longer be able to send and receive messages to other clients.

Example:
<nickname_here> : QUIT

4 Error Handling

The application has been configured to deal with errors when entering commands. If a user enters an invalid command, the server will display a message indicating an unknown command was entered. If a user enters a command that expects a certain number of arguments, the server will display an error message.

Example:
<nickname_here> : SEND nickname
<server> : Please enter a message

<nickname_here> : JOIN
<server> : Please enter a room name

5 Crash Handling

The application has also been configured to handle system crashes. If the server crashes, a message is displayed to any connected clients stating that the connection has been closed.

Alternatively, if a client's connection is spontaneously closed, this does not affect the server directly. The server just takes care of checking for any

closed connections every time a new client tries to connect, and removes it accordingly from its list of connected clients.