CS494P Project RFC
Internet Draft
Intended status: IRC Class Project Specification

Lily Tran
Portland State University
December 5, 2019

Abstract

The goal of this project is to develop an IRC protocol by implementing an IRC client and server. IRC is a text based protocol that allows IRC clients to send messages to other clients via a socket connection. This implementation, done in python, supports multiple clients and chat rooms. This memo is for the Internetworking Protocols class at Portland State University.

Table of Contents

1. INTRODUCTION

This specification describes a simple Internet Relay Chat (IRC) protocol that provides a means of communication between clients via a central model in a client-server model. 'Relay" messages are sent to the central server, which are sent to other connected users. This protocol will operate over the TCP/IP protocols to connect clients and servers over TCP socket connections. Users can join create and join rooms. Any message sent to that room is broadcasted to all users currently joined to that room.

2. BACKGROUND INFORMATION

Client/server computing is very important because the internet services are organized according to the client/server architecture like file transfer remote login and the web. The client must make a request to the server, and the server responds to the requests with the service requested. They can establish connections and communicate via sockets. Connections are communication links that are created over the Internet using TCP. Some client/server applications are also built around the connectionless UDP. All of these communicate via sockets with IP address of the host and port address that represents the communication port. Clients create client sockets and connect them to server socket and then exchange data over the connection. We use TCP because it supports reliable communication such as dealing with packet loss, unlike UDP.

3. SOCKET IMPLEMENTATION IN CLIENT AND SERVER

In order to support multiple clients, I had to use non-blocking I/O. All socket methods are originally blocking, and to serve more clients, I set the socket to non-blocking.
I had figured out a way for the operating system to notify the program when there is something to read from the socket or when it is ready for writing. I used python's select.select() method. This asks the OS whether the sockets provided are readable, writable, or erraneous. This blocks the program until a socket is ready. If there are sockets, operations are performed! If the socket is the main server socket, the one being used to listen for connections, then it is ready to accept another incoming connection. Python's select() function is a direct interface to the underlying operating system implementation. It monitors sockets, open files, and pipes (anything with a fileno() method that returns a valid file descriptor) until they become readable or writable, or a communication error occurs. Select() makes it easier to monitor multiple connections at the same time. That is the purpose of class Peer and why it returns fileno.
"If s is readable" means that there is a socket

3. CLIENT

Usage

The IRC client will need to establish a TCP socket connection to the IRC server and forward stdin to the IRC server. The client will display messages from the server on stdout. Clients can log in using a nickname and view all logged in users and chat rooms. Clients will be able to join, create, and leave rooms. They can also send messages to all other clients within that same room, and clients will be able to exit the server gracefully. Before subsequent messages can be sent, a connecting client must provide a nickname. The server associates the member with the socket connection of the user.

Field Definitions

- Variable: 'Server' is creating a socket by calling socket.socket. setsockopt() is then called to allow to reuse of the socket before it times out. The socket then connects to server with address localhost and port 50000.

- Variable: ‘Inputs’ represents a list of clients that has arrived
- “If s is server” means that it is a new client, and we are ready to receive the client’s message.

4. SERVER

The server creates a socket and creates variables for sockets from which we expect to read and write to. The server accepts connections from multiple clients and stores data of the new clients before it begins to read from input. If there is no message, we can remove the member from the input list. The server is responsible for sending useful information back to the clients via socket.sendall() method.

5. LISTING ROOMS AND ITS MEMBERS

The server checks to see if there are any number of rooms. If variable room is null, that means that there are no rooms, and the server will display to the user to create one. However, if there is at least 1 room available, the server will print all available rooms and its members using a nested for loop.

6. JOINING AND CREATING A ROOM

The makes a check to see if a member is currently part of an existing room. If the client is not, it will proceed to call Room class that initializes fields for a room.

7. SWITCHING TO A NEW ROOM

8. LEAVING A ROOM

This has not been implemented yet.

References

https://pymotw.com/2/select/
https://steelkiwi.com/blog/working-tcp-sockets/