CS494 Mohammed Mchichou
Internet Draft Portland State University
Intended status: IRC Class Project Specification June 10, 2016
Expires: June 2016

Internet Relay Chat Class Project
draft-irc-pdx-cs494-0000.txt

Status of this Memo

Copyright Notice

Abstract

This project consist of implementing the communication protocol of an IRC-style client-server system using Python for the Internetworking Protocols class at Portland State University. This IRC will consist of a command line based client/server that can handles multiple users and multiple channels.

Table of Contents

1. Introduction:

This document describes a project to implement an IRC protocol allowing users to communicate with each other. In this system, we will have a central server that relays messages coming from a connected user to another connected user.

Users can join rooms or create rooms where they can exchange messages: one/many to one/many users.

Communication between the server and clients will be conducted over TCP/IP socket connection Using the socket class and libraries of Python.

2. IRC Server:

   The IRC Server will be responsible of managing sockets and broadcasting messages from a client to another one.

   The IRC Server will be responsible of monitoring all the socket connection and executing commands provided by the client.

   The IRC Server will handle client crashes and list the appropriate messages.

3. IRC Client:

   The client can connect to the server by providing by entering the host and the port number provided by the server.

   The client has to provide a nickname that is less than 10 characters.

   The client can only talk to another client via the server.

   The menu of all the option would be sent as a message from the server to the client and the selected option will be sent back as message to the server.

   Each message sent or received will then be parsed by the server to decide/perform a response/action.

   The IRC Client will establish a TCP socket connection to the IRC Server and once connected, the client should be able to:

   - Create a room
   - List all rooms
   - Join a room

- Leave a room
- List all members of a room
- Send messages to a room
- Send private messages to specific users
- Send a file
- Disconnect from the server

The IRC Client will handle server crashes and list the appropriate message.

4. Create a room:

Once connected, the IRC menu will be displayed.

To create a room, the user has to type the command "/create", then he will be prompted to enter the room's name.

The user does not have to worry about '#' before the channels name as the server will add that.

The server will create the room if the name was not used before otherwise, the user will receive a message that the name has been taken.

The server will display the #<room> and give the user the option to go back to the menu by entering "/0"

5. Display rooms:

To display a list of all available rooms on the server, the user can enter "/rooms".

The server receive the message and go through the list of rooms created, the forward this list a message to the user.

#<room1>

#<room2>

…

The server will then give the user the option to go back to the menu by entering “/0”.

6. Join a room:

The user can join one or multiple rooms by entering “/join”.

The server will send a message asking the user to input the room he wants to join.

For joining multiple channels this process can be repeated.

The server will display an error message if the room does not exist.

If room found, the server will link the nickname to this room and send the client a message that he is in the room and asking him to start exchanging messages inside the room.

Messages entered in the room will be seen by other used as following:

#<room> →<nickname> : <message>

Only users who share the same room can see these messages.

The user can go back to the menu by entering “/0”.

7. Leave a room:

The user can at any time leave a room by entering “/leave”.

Users of the same room will be notified by the server that this user has left.

The server will remove the room from list of rooms joined by this user.

8. List all users in a room:

   The user may need to see who is joining a room.

   The server provides this feature using the command “/list”

   The user has to specify what room to list and the server will search that list and display all nicknames that are joining that room.

   The server will then give the user the option to go back to the menu by entering “/0”.

9. Private message:

   User can send to one/many users a private message weather they are in the same room or not by asking the server to direct the message only to him/them.

   The command to use is “/msg:<nickname>:<message>”

   The server parse this long message and extract the command, the nickname and the private message.

   IF the receiver’s nickname is found, the message will be delivered as following:

   <sender_nickname>→<message>

   The receiver can then replay the same way if he choose to.

   Once the message is sent, the sender is notified that his message was successfully delivered and in case of failure the appropriate message will be displayed.

   The user can go back to the menu by entering “/0”.

10. Send a File :

   You can send a file by using the command:

   /send:<nickname>:<fileName>

The server will receive your file and look for the nickname of the receiver. If Found, the file will be transferred, preserving the extension and the format.

The transfer will happen in raw data, therefore we can send any type of file.

11. Disconnect from the server:

The user may choose to disconnect from the server at any moment by entering “/q”.

The server will then close the client’s socket, notify the other users and delete the user’s saved data such as nickname, joined rooms.

12. Missing futures:

This is a list of option that I was hoping to include in my project but was not able to due time constraint:

File transfer.

GUI interface

Protocol encryption

Cloud connected Server

Audio/video support.

I may come later to work on these feature for my own pleasure and enjoyments.

Maybe Med-IRC will be one day seen at apple/google store.

13. Security Considerations:

Messages sent or received by the server or any of the clients were not implemented to be secure. Private messages are only hidden from other clients but can still be intercepted by any of the many network-attacks techniques. Therefore, if someone wish to use this IRC for secure communication he/ she may need to use/implement their own encryption protocol.

14. IANA Considerations:

None

15. Normative References:

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

16. Acknowledgments:

Many thanks to the http://www.bogotobogo.com/ for their extremely helpful tutorial that helped me build the IRC project.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Mohammed Mchichou
Portland State University Computer Science
1825 SW Broadway, Portland, OR 97201

Email: mchichou@pdx.edu