Network Working Group
Request for Comments: nnnn
Category: Informational

Michael Hughes
December 2012

Internet Relay Chat Protocol

Status of this Memo

RFC editor to fill in this portion.

Abstract

This document specifies requirements for an Internet relay chat protocol (IRC). This protocol uses text based chat rooms for Internet conferencing using servers to relay messages between clients. Internet protocols are specified and structure for client and server. This document assumes that the reader is familiar with IRC Architecture.

Table of Contents

1. INTRODUCTION

The purpose of this document is to specify how to implement an Internet Relay Chat service (IRC). This document will be useful for anyone trying to develop his or her own IRC. It is assumed that the user knows what a chat room is and how to implement networking protocols. This document focuses on the simplest IRC and that is a single server IRC.

1.1 Client-Server Relationship

The Client-Server relationship is very simple. The client sends requests to the server and the server responds with a message to inform the client that the action has taken place. The client will also send all messages to the server and the server will distribute the messages to their respective chat rooms.

1.2 Client-Client Relationship

The Client-Client relationship is limited. Clients have no ability to directly communicate with each other. Clients respective servers handle all communication between each other.

2. CONNECTIONS

This section specifies how connections for the Client-Server relationship. The IRC relies on a TCP/IP connection. One port number is used to keep the connection process as simple as possible.

2.1 Client-Server Connection

The Client-Server connection is fairly straightforward. The server creates a socket and the client connects to the socket. The programmer is to decide if the user should input the address of the server or if it should be hardcoded in (for usability purposes hardcoded is recommended). Once a connection is created the server sends the client an "OK" to confirm the connection. The client then sends the server the command "C" to inform the server that a client is connecting. Next the

server will need to create a new process or thread for the connection so the server can listen from new users in the main thread. The new client sends the server the user name of the individual who just joined the server. That way the server knows who is in the IRC and can list current peers if a request is made. The connection is now set up and the server waits for commands from the client.

3. PROTOCOL SPECIFICATIONS

Below are specifications for services provided by the IRC. Some of the commands below will expect an “OK” message returned by the server. It is important to keep note of which of these commands will require receiving an “OK” message.

3.1 Client Protocol

The client protocol works specifically on a Client-Server relationship and does not work on a Client-Client relationship. This protocol is how all communication for the client occurs.

3.1.1 List Rooms

The list rooms function checks the server for a list of the available chat rooms. The command to get a list of rooms is “LR”. Once the server receives the command it will immediately return a string of the rooms available. If no rooms are available the only message the sever should return is a message informing the client that no chat rooms are currently available. The server DOES NOT need to respond with an “OK” in this instance.

3.1.2 List Peers

The list peers function is very similar to the list rooms function except list peers returns a list of people currently online. The command for list peers is “USE”. Once the server receives the list peers command the server will send a string back to the client of peers currently online. Once again the server DOES NOT send an “OK” message for this function. There will always be at least one peer available and that is the client that sent the command itself. It is up the programmer to decided if he or she wants to send a no peers online message or just the current clients name (If only the sender of the command is online it is recommended to send no peers online).

3.1.3 Create Room

The create room command is used to create a chat room. The command to start a chat room is “CR”. Once CR is sent the server will immediately send an “OK” to the client so the client knows the message was received successfully. Next the server will wait for the client to send the name of the chat room. If the room already exists the server sends the “ER” message to inform the client that the room can not be created. If the chat room has not been created the server places the client (referred to as the owner) who created the chat room in that room. If the owner of the chat room leaves the room then the room closes and all peers currently connected to the room are informed that the owner is closing the room by receiving the message “Q”. Once a “Q” is received by the clients connected to the room they can take the appropriate steps to disconnect from the chat room. The reason that the room closes once the owner leaves is to prevent the server from filling up with a large amount of empty chat rooms.

3.1.4 Join Room

The join room function is used so a client can join a room if it exists. The command is “JR”. If the command has been sent an “OK” will be returned from the server. The server will now be waiting for the client to send the name of the room that he or she wishes to join. Once the name of the room is sent if no room exists with the clients specified name the server should respond with the message “ER”. If an “ER” is returned the client should notify the user that the room does not exist. If the room does exist the server should send an “OK” and the client should enter the room. Once in the room the client has to make itself know by sending the message “username has entered the chat room.” To leave the room the client sends the message “Q”, that way the server knows the client no longer wishes to participate in the chat room and removes the individual. Before the client disconnects the message “username has left the chat room” should be displayed.

3.1.5 Command Mode

The purpose of command mode is to left privileged users close the server. To close a server the user MUST know the password for that server. If he or she does not they will not be allowed to terminate the server. First the command “CMD” is sent to the server. Next the server will respond with an “OK” message and be waiting for the user to send the password of that server. If the password is correct an “OK” message is sent, otherwise an “ER” message is sent. To finalize the quit server option the client must send the “SQUIT” command. The user in command mode will immediately disconnect from the server and the server will stop

listening for new connections. Other users will remain connected to the server until they to decide to disconnect from the server. This prevents their current chat rooms from getting disrupted.

3.1.6 Quit IRC

If the client wants to disconnect from the server the client must send the command “QUIT”. Once the command “QUIT” is received by the server the server responds with the message “OK” informing the client that a successful disconnection occurred. Anything else would be considered an error.

3.1.7 Client Design Considerations

This IRC protocol assumes that the client is in one room at a time. Therefore the use of multiple processes or threads is unnecessary in the client.

3.2 Server Protocol

The commands the server sends to the clients are covered in the client protocols section (section 3.1) however there are a few design concepts to keep in mind.

3.2.1 Initializing the Server

When initializing the server a password MUST be set. This password is used to allow a user to enter command mode and terminate the server. If no password is entered a default value will need to be specified.

3.2.2 Server Design Considerations

When creating a server using this IRC protocol it is important to ensure that multiple clients can connect at once. This will need to be accomplished through the use of either multiple processes or threads. The use of multiple processes or threads is left to the programmer to choose.

4. SCALEABILITY

This protocol is not meant for very large IRC services due to its use of a single server. The way to make this protocol better for large IRC services would be to create a network of several Server-Server connections.

5. SECURITY

Security has not been implemented in this protocol for simplicity with the exception of the server password. However since none of the connections are using any encryption it is assumed that this IRC is not meant for secure message transmission.

6. ACKNOWLEDGMENTS

This IRC protocol was developed through research in RFC 2813, RFC 2812, and RFC 2810.

7. REFERENCES

s+1. Informative References

[IRC-ARC] Kalt, C., "Internet Relay Chat: Architecture" , RFC 2810 April 2000.

[IRC-CLIENT] Kalt, C., "Internet Relay Chat: Client Protocol" , RFC 2812 April 2000

[IRC-SERVER] Kalt, C., "Internet Relay Chat: Server Protocol" , RFC 2813, April 2000.

8. AUTHOR'S ADDRESS

Michael Hughes
Longview, WA 98632
USA

michael.r.hughes@email.wsu.edu

9. FULL COPYRIGHT STATEMENT