**RFC Internet Relay Chat Internet Relay Chat Protocol Ahmed Abd.**

**Nov. 2015**

Internet Relay Chat Protocol

Status of This Memo

This is an experimental protocol specifically designed for the internet community. Please refer to the current edition of the “IAB Official Protocol Standards” for the status and standardization state of this protocol. Suggestions and discussions to improve this experimental protocol are requested.

Abstract

The Internet Relay Chat (IRC) is a text-based protocol that transfers messages in Form of text. The chat process works on a client-server networking model. Clients are able to communicate with each other through servers. Basically, IRC clients are computer programs that anybody can install on their computer. IRC is designed for group-based communication, one-to-one communication, data transfer, and file sharing.

Table of Contents

1. INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . . . 2
   1. Servers . . . . . . . . . . . . . . . . . . . . . . . . . . 2
   2. Clients . . . . . . . . . . . . . . . . . . . . . . . . . . 2
   3. Channels . . . . . . . . . . . . . . . . . . . . . . . . . . 2
      1. Channel Operators . . . . . . . . . . . . . . . . . . . 2
      2. Channel Name . . . . . . . . . . . . . . . . . . . . . . 2
2. THE IRC SPECIFICATION . . . . . . . . . . . . . . . . . . . . . . 2
   1. Character Codes . . . . . . . . . . . . . . . . . . . . . . . 2
   2. Messages . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   3. Numeric Replies . . . . . . . . . . . . . . . . . . . . . . . 3
3. IRC Concepts . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   1. One-to-Many-Channel Communication . . . . . . . . . . . . . . 3
   2. One-to-Many-Clients Communication . . . . . . . . . . . . . . 3
4. MESSAGE DETAILS
   1. Server Message . . . . . . . . . . . . . . . . . . . . . . . 3
   2. Quit Message . . . . . . . . . . . . . . . . . . . . . . . . 4
   3. Nick Message . . . . . . . . . . . . . . . . . . . . . . . . 4
   4. Join Message . . . . . . . . . . . . . . . . . . . . . . . . 4
   5. Part Message . . . . . . . . . . . . . . . . . . . . . . . . 5
   6. User List Message . . . . . . . . . . . . . . . . . . . . . 5
   7. List Message . . . . . . . . . . . . . . . . . . . . . . . . 6
   8. Msg Message . . . . . . . . . . . . . . . . . . . . . . . . 6
   9. Ping Message . . . . . . . . . . . . . . . . . . . . . . 6
   10. Pong Message . . . . . . . . . . . . . . . . . . . . . . 6
5. REPLIES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
   1. Command Responses . . . . . . . . . . . . . . . . . . . . . . 7
   2. Error Replies . . . . . . . . . . . . . . . . . . . . . . . . 8
6. IMPLEMENTATION
   1. Network protocol (TCP) . . . . . . . . . . . . . . . . . . . 8
   2. Command Parsing . . . . . . . . . . . . . . . . . . . . . . . 9
   3. Message Delivery . . . . . . . . . . . . . . . . . . . . . . 9
   4. Connection ‘Liveness’ . . . . . . . . . . . . . . . . . . . . 9
   5. Establishing a server-client connection . . . . . . . . . . . 9
   6. Terminating server-client connections . . . . . . . . . . . . 9

**1. INTRODUCTION**

The IRC has been mainly designed for one-to-one, group communications, and file transferring. An IRC involves one thing, the server. The server forms a central point for clients to connect, send file, or messages to. This document will describe the IRC protocol.

* 1. **Servers**

A server is basically a central point where the clients can connect and talk to each other. Servers have unique names to be identified by.

* 1. **Clients**

A client is essentially a software where users install on their computers to connect to an IRC server. Clients can be identified by name. Each one has a unique name with a maximum length of nine characters.

* 1. **Channels**

A channel is a room with one or more groups of clients where all clients receive messages that are sent to that specific channel. A channel gets created when a client joins it. It’s destroyed as soon as the last client leaves it.

To create a new client, the user must join as specified above. The user must join the channel by their names. When a user creates a channel, they become the channel operator. Users can also join existing channels. Clients can be in multiple channels at the same time if they wish.

* + 1. **Channel Operators**

A channel operator is the user who creates the channel. They act as the manager of the channel. Two commands used by the channel operator are given below:

TOPIC: Change the channel topic

KICK: Kick a client out

* + 1. **Channel Name**

Channel names are strings that begin with a “#”. A channel name can’t have spaces, control G or a comma.

**2. The IRC Specifications**

In this section, the protocol described is for use with client to

Server connections.

**2.1 Character Codes**

8-bit protocol will be used to generate a text (message). Control codes

will be used by special characters. They’ll act as message delimiters.

**2.2 Messages**

A message is used between servers and clients. They both generate

messages which may or may not be a reply. When a client sends a message

with a command to a server, the client will receive a reply from the

server. Messages commonly include a command and its parameters

separated by space.

**2.3 Numeric Replies**

Numeric replies are used for replies, such as an error reply or even

just a regular reply. A numeric reply consists of three digits followed

by a message related to the three digits. Numeric replies can only be

sent from servers to clients. Messages with numeric replies from

clients will be ignored by the server.

**3. IRC Concepts**

This section describes how the IRC delivers messages. Mainly, deliver

Type of one-to-many communication.

**3.1 One-to-Many Channel Communication**

Clients can send a message to the channels in which they’re currently

In. When a message is sent, it’s sent to all the channels and delivered

to all the clients in these channels.

**3.2 One-to-Many Client Communication**

Clients can send messages to all clients that are in the same channel

**4. Message Details**

This section describes the types of recognized messages by the IRC

server and client.

A server must parse the messages sent from clients that are connected

to it. Any errors encountered by the server must be reported back to

the client with an appropriate message. Parsing is terminated when an

error is encountered. The type of errors are shown below:

Not enough parameters

Incorrect command

Incorrect privileges

**4.1 Server Message**

A server message looks like the following:

Command: SERVER

Parameters: [server name]

The message above is used by a client when the client wants to connect

To a server. When a client connects successfully, the server will send

a message indicating that the connection is successful. However, when

a client tries to connect to an unknown server, the message “unknown

server” will be returned to the client. When a client is successfully

connected to a server, it will be given a unique nickname. These names

can be changed by the client by using Nick messages.

**4.2 Quit Message**

A quit message looks like the following:

Command: /exit

Parameters: None

As shown, they don’t contain and parameters. When a server receives a

quit message from a client, it must close the connection with that

client. When a client crashes, the server will generate this message

on its own.

Example /exit

**4.6 Join Message**

A join message looks like the following:

Command: JOIN

Parameters: [channel], {,[channel]}

When a client wants to join one or more channel(s), it sends this type

of message. When joining multiple channels, the client must separate

the name by a “,”. If a join fails, the server will send an error

numeric reply. If a join succeeds, the server will send “successfully

join [channel name]”.

Numeric replies examples:

ERR\_CHANNELFULL

ERR\_TOOMANYCHANNEL

Examples:

JOIN #2 join channel 2

JOIN #4, #5, #7 join channel 4, 5, and 7

**4.7 Part Message**

Command: /leave

Parameters: [channel] {,[channel]}

Part messages are used by clients to remove themselves from the active

user list. When successful, the client leaves the channel. When not

successful, an error message is returned.

Example: /leave #7 Leave channel 7

Example: /leave #7, #8 Leave channel 7, 8

Error ex.:

ERR\_NOTINCHANNEL

**4.8 User List Message**

A user list message looks like the following:

Command: /who

Parameters: [channel]

This type of message is used by clients to get a list of the current

users in the channel.

Example: /who #channel\_1

Error example:

ERR\_NOTINCHANNEL

**4.9 List Message**

A list message looks like the following:

Command: /list

Parameters: none

This type of message is used by clients to get a list of all active

Channels.

Example: /list

**4.10 Channels Message**

Command: /message\_channels

Parameters: [channel, {channel2…}], [[message]]

This type of message is used to send a message to all selected

channels.

Examples:

/message\_channels #5 hi send “hi” to channel 5

/message\_channels #5, #6 hi send “hi” to channel 5 and 6

**5. REPLIES**

This section describes a list of numeric replies in response to the

commands above.

**5.1 Command Responses**

300 RPL\_NONE Not used (dummy)

301 RPL\_LIST “[channel]: [topic]”

302 RPL\_LISTEND “:End of /LIST”

306 RPL\_USERS “NickName”

307 RPL\_ENDOFUSERS “:End of users”

**5.2 Error Replies**

401 ERR\_NOSUCHSERVER “[server name] :No such server”

402 ERR\_NOSUCHCHANNEL “[channel name] :No such channel”

403 ERR\_TOOMANYCHANNELS “[channel name] :You have joined too many

channels”

1. R\_UNKNOWNCOMMAND “[command] :Unknown command”
2. R\_NEEDMOREPARAMS “[command] :Not enough parameters”

**6. Implementation**

This section describes the implementation.

**6.1 Network Protocol (TCP)**

IRC is implemented using TCP protocol. This is done because of the

Reliability, error correction, and data flow that TCP provides.

**6.2 Command Parsing**

As shown earlier, messages contain commands and parameters. These

messages are parsed based on these two items. Then, conditional

statements are used to determine the meaning of each item.

**6.3 Message Delivery**

A queue is used when delivering messages. This queue is specially

useful when sending a large amount of data. Initially, the server

parses all data, and queues them. After that, queued data is sent.

**6.4 Connection ‘Liveness’**

Servers must ping each of its clients (connections) to make sure

they’re alive when they don’t respond in a certain amount of time.

If a client doesn’t respond in time, the connection is closed with

That specific client.

**6.5 Establishing a Server-Client Connection**

When a client connects to a server successfully, the server sends

A welcome message (MOTD message, Message of the Day).

**6.6Terminating Server-Client Connection**

When the connection of a client closes, a quit message is generated

by the server (on behalf of the client)