CS494
Request for Comments

William Harrington
Portland State University
May 2015

CS494 Programming Project: Internet Relay Chat (IRC) Protocol

Status of this Memo

This memo defines an Experimental Protocol for the Internet community; it is not an Internet standard of any kind. This document has been published for examination, experimental implementation, and evaluation. Discusssion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Abstract

The purpose of this Request for Comments (RFC) is to outline an experimental protocol for a programming project that will be used for developing an Internet Relay Chat (IRC) Server and Client application. The intended result is to mimic some very basic IRC functionality with the protocol.

The protocol defines the responsibilities of the server and the client as well as how they will communicate. The protocol is heavily weighted towards the server. The server must keep track of all the users and all the channels as well as what users are in what channels. It must also manage incoming connections and messages and reply in a timely manner. The server is also required to give information about the clients it is managing to any client that asks but it is required to do so in a manner that will not pose any security risks. The client is only required to issue valid commands when interacting with the server.

Table of Contents

1 Introduction

This Request for Comments (RFC) provides an outline for the protocol that will be used to develop an Internet Relay Chat (IRC) Server and Client program. This document covers the responsibilities of the server, the client, and how their interaction must take place in order to properly implement an IRC application.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.1 Servers

A server must comply with the client-server model. Connections between the server and the client must be established via the TCP protocol, using a socket application programming interface (API). Furthermore, the server shall be responsible for managing multiple connections at any one time. This means that multiple clients should be able to connect to the server. The server should have a minimum backlog of five to account for possible congestion.

When the server begins operation, it must start storing specific information about clients that connect such as a username. The server shall also have the responsibility of operating on this stored information, but shall only do so when requested by the client. All the information stored on the server will be used to identify a client and their activities. More about client information stored in the server is discussed in sections 1.2 and 1.2.1.

The server shall also be responsible for keeping a list of users (clients) and channels. It may also keep a list of socket objects.

1.2 Clients

A client is anything connecting to a server that is not another server. When a client establishes a connection with the server, the client must provide information to the server that can be used to identify it. Each client must be distinguished from other clients by a unique username. The maximum length for a username should be (9) characters. At a minimum the client should provide a username. If the client does not supply a username, then the server must assign them one.

Information collected about each client should be organized by socket, but it may be organized differently as long as the server is able to create unique accounts for each client. Additional information is optional as specified in section 1.2.1.

1.2.1 Accounts

The information that the server holds about a particular client is referred to as an account. An account should be associated with a client’s socket. An account is made up of the following pieces of information: username, ip, channels, current.

The username is a name associated with the user of a particular client. This is a required piece of information which means the client must provide it or the server must assign it. It must be unique which means no one else should be currently using it as their username.

The channels are simply the channel(s) that a user is a part of and current being the channel that a user currently has selected.

Account information can be modified through client to server communication known as messages which is explained in section 2.

1.3 Channels

A channel is a named group of one or more clients which all all receive messages addressed to that channel. The channel is created implicitly when the first client joins it, and the channel ceases to exist when the last client leaves it. While a channel exists, any client can reference the channel using the name of the channel.

Channel names are strings of length up to 50 characters, and must begin with a ’&’ or ’#’. They must not contain any spaces, a control G, or a comma.

To create a new channel or become part of an existing channel, a user is required to join the channel. If the channel doesn’t exist prior to joining, the channel is created. Otherwise, the user is added to that channel. A user may be a part of up to 10 channels at once.

2 Messages

Servers and clients send each other messages which may or may not generate a reply. If the message contains a valid command, as described in later sections, the client should expect a reply as specified but it is not advised to wait forever for the reply; client to server communication is essentially asynchronous in nature.

Each IRC message may consist of up to two parts: the command and the command parameters.

Messages are always lines of characters terminated with a Carriage Return - Line Feed pair (CR-LF). They shall not exceed 512 characters in length, counting all characters including the trailing CR-LF. Thus, there are 510 characters maximum allowed for the command and its parameters.

2.1 Message Details

The following sections are descriptions of each message recognized by the IRC server. All commands described in this section must be implemented by any server for this protocol. The server to which a client is connected must parse the complete message, returning any appropriate errors. If a full set of parameters is presented, then each must be checked for validity and the appropriate response must be sent back to the client.

2.1.1 Help Message

/help <command>

The help message is used to provide the client information on how to properly send commands to the server. If a help message arrives to a server with out the <command> parameter, it shall send the user who sent the message a list of valid commands. If a help message arrives to a server with the <command> parameter, it shall send the user who sent the message more info on how to properly send that command to the server. If a help message arrives to a server with the <command> parameter and the <command> parameter is invalid, it shall send the user an error.

2.1.2 Nickname Message

/nick <username>

The nickname message is used to specify a username or change it from what it previously was.

If a nick message arrives at a server which already knows about an identical username for another client, it shall refuse the nick message by sending an error to the client. If the client has no username, the server will request that another nick message be sent. If no nick message is sent within a specified amount of time, the server may assign one. Otherwise, the server should close the connection. If the client already has a username, then it shall remain unchanged.

If a nick message arrives at a server that doesn't have an identical username for another client, it shall change the client's username to the <username> parameter. If a nick message arrives at a server without the <username> parameter, the server shall echo back the user's current username.

2.1.2 Join Message

/join <channel>

The join message is used by a client to start listening to <channel> . If the channel does not exist, it is created by the server and added to the list of channels. If the channel name is not valid, an error message is sent to the client and the channel is not joined nor created.

If a join is successful, it is added to the client's list of channels on their account. They will begin to see messages sent to the channel from other users when the channel is selected as their current channel. They will also be able to send messages to the channel when it is selected as their current channel.

2.1.3 Part Message

/leave <channel>

The part message is used by a client to leave <channel>. If the channel name specified is not valid, an error message is sent to the client and nothing else is done. If the user is not in the channel specified, an error message is sent to the client and nothing else is done. If the user is in the channel specified, that channel is removed from their list of channels. If the channel to leave is a client's current channel, their current channel is changed to any other channel in their list of channels or none if there is none.

If the user leaves the channel and was the last person in the channel, it is removed from the list of channels tracked by the server.

2.1.4 Quit

/exit

The quit command is used by a client to leave all channels and disconnect from the server. Upon a successful quit, each channel that a client was in is notified that they have left. All client information stored in the server account is removed after the connection has been closed.

3 User based query

User queries are commands which are concerned with supplying details to a user when requested. The result of these commands when successful yield important information to the user. None of the information returned from these commands should be sensitive in that it could be used to compromise another user.

3.1 Who query

/who <channel>

The who message is used by a client to generate a query which returns a list of information which matches the <channel> parameter given by the client. In the absence of the <channel> parameter, all the clients currently connected to the server are listed by their username. When the <channel> parameter is present, all the clients currently connected to the server that are in that channel are listed. If the <channel> parameter is invalid, an error message is sent to the client and nothing else is done. If the channel does not exist, an error message is sent to the client and nothing else is done.

3.2 Whois query

/whois <username>

The whois message is used by a client to generate a query which returns information about the user whose username matches the <username> parameter given by the client. In the absence of the <username> parameter, an error message is sent to the client and nothing else is done. If there is no user whose username matches the <username> parameter, an error message is sent to the client and nothing else is done. If there is a user whose username matches the <username> parameter, then the server shall send information to the client who send the message about the user whose username matches the <username> parameter. This information should include a list of the user’s current channels, and may include other information like IP address.