IRC Protocol

Azadi Bogolubov

1 November 2014

Abstract

The IRC protocol was developed in 1988 by Jarkko Oikarinen as a way for individuals to chat online.

This IRC protocol is a text-based protocol, which makes use of one or more client sockets connecting to a server. It is a minimal implementation of that which is detailed in RFC 1459.

Table of Contents

1. Introduction

The IRC protocol is a popular chat client, and is used heavily by the student body of Portland State University. In this document, you will find the protocol designed for this particular IRC protocol implementation.

This IRC protocol uses the TCP/IP Client/Server network protocol.

The server acts as the central hub for clients to connect to, using a socket selector to handle multiple clients simultaneously.

1.1 Server

The server is the central point of communication to which the clients connect. The server may be cloud hosted or a local machine. In this particular use case, a client is anyone who wishes to connect to the IRC server to talk with another client. This model employs only one server, which can handle multiple clients simultaneously.

1.2 Clients

A client is anyone who connects to the server in order to carry on communication with another client. It is presumed that a client is a human being, though there is no formalized check (e.g. CAPTCHA) to ensure that the client is indeed a human and not a bot. A client may have a nickname which is up to 16 characters long. Valid nicknames are detailed in the next section. Additionally, the server will keep a listing of the IP address for each connected client.

1.3 Nicknames

A nickname may consist of any valid combination of the groups given below in pseudo BNF format. {group} means one or more instances of the group, [group] means an optional group, and { [group] } means repeat an optional group.

```
<nickname> := {alpha} { [number | special] }
<alpha> := {A-Za-z}
<number> := {0-9}
<special> := {_,&}
```

1.4 Channels

A public channel is a group of one or more clients. All clients who have joined the channel will receive all messages which are posted to the channel in real-time.

The creation of a public channel occurs when the first client joins it, and will be destroyed when the last client leaves it. The length of a channel name shall not exceed 16 characters.

Public channel names will adhere to the following grammar:

```
<name> := # {alpha} [ {numeric | alpha } ]
<alpha> := [A-Za-z]
<numeric> := [0-9]
```

Additionally, there are private channels, where 2 clients can privately message each other.

These channels are created when the initiating client joins it and specifies the nickname of the client to whom they wish to private message with. The nomenclature of a private channel follows a similar convention, given below:

```
<name> := ! {alpha} [ {numeric | alpha } ]
<alpha> := [A-Za-z]
<numeric> := [0-9]
```

2. IRC Concepts

This section will describe how clients can interact with each other by means of the IRC server

2.1 One-to-one communication

All one-to-one communication shall be between clients only, since this protocol is designed for one server and many clients.

2.2 One-to-many communication

All one-to-many communication shall be between clients only, since this protocol is designed for one server and many clients.

2.3 Many-to-many communication

Many-to-many communication shall be comprised of in-channel activity, where there may be as few as one client sending a message, up to many clients simultaneously sending messages amongst themselves.

3. Message Details

A server message is a message which will have an inherent meaning to the server, and will be processed before being displayed, if displayed at all, to the clients. Listed in the following subsections are the message types which receive special handling by the server, as well as the grammars which are used to invoke them.

3.1 Set Nick

Command: /NICK
Parameters: <nickname>
In order for the client to use the IRC service, they must have a unique nick. This is done using the NICK message. Upon request to connect, a socket will

be created and will attempt to connect to the server. The nick will be tied to the IP address and port number of the client to whom it belongs.

3.2 Create Channel

Command: /JOIN
Parameters: <channelname>

In the case that a channel does not exist, on the invocation of the message JOIN, a channel will be created with the name <channelname> and the client who initiated the message will become the first member of that channel.

3.3 Join Channel

Command: /JOIN
Parameters: <channelname>

For clients to be able to message using IRC, they will need to join channels. In order to join a channel, the client will need to use the JOIN message, along with the name of the channel they wish to join. Channel names are case-sensitive. If the channel does not exist, it will be created with the first user who joined.

3.4 Leave Channel

Command: /PART
Parameters: <channelname>

A client may wish to unsubscribe from channel updates. To do so, they will use the PART message, along with the name of the channel which they wish to unsubscribe from. Channel names are case-sensitive.

3.5 List All Channels

Command: /LIST
Parameters: none

A client can list all available channels by entering LIST. This will present all the current channels in the format of one channel per line.

3.6 List All Members in a Channel

Command: /MEMBERS
Parameters: <channelname>

A client can list all available members in a channel by sending the MEMBERS message. This will present all the current members in a channel in the format of one member per line.

3.7 Private Message

Commmand: /MSG
Parameters: <nickname>; <message>

There may be a time in which a client desires to have a private conversation with another client. In order to have such a conversation, the client will use the MSG message along with the name of the client they wish to converse with.

Example: MSG <name_of_client>; Hello, this is a private message.

3.8 Send Message To Multiple Channels

Command: /MULT
Parameters: [<channelName>,{<channelName>}]; <message>

A client can send a single message and have it be sent to multiple rooms using the MULT message.

Example: /MULT [CS594,CS510,MTH561] Hello, this is a message to multiple rooms.

3.9 Quit IRC

Command: /QUIT
Parameters: none

At some point, a client will want to end their IRC session. For this purpose, a client will use the QUIT message. Upon disconnection the server, the client's nick will be freed, and their IP address removed from the pool of connections.