Internet Relay Chat Protocol

Sophia Weeks

11/29/2020

1. Introduction

This document describes a simplified Internet Relay Chat protocol based on RFC 1459. It is implemented using the TCP/IP network protocol. The IRC is a teleconferencing system, which can run on multiple machines in a distributed fashion. The setup includes a single server process which multiple client processes may connect to in order to deliver messages to one another.

1.1 Server

The server provides a central point to which clients may connect in order to talk. This protocol allows only one single server process.

1.2 Clients

Multiple clients may connect to a single server. Each client has a unique nickname with a maximum length of nine characters.

1.3 Rooms

A room is a named group of one or more clients which will all receive messages addressed to that channel. The room is created by request from a client and ceases to exist when all clients leave it. If a client attempts to create a room whose name is already in use by another room, the attempt will fail. While the room exists, any client can reference it using its name.

Room names are strings of length up to 200 characters. Room names may not contain any spaces (‘ ‘) or a comma (‘,’). To become part of an existing channel, a user must request to join the channel. This will succeed if the room exists.

2. The IRC Specification

2.1 Messages

Servers and clients send each other messages which may or may not generate a reply. If the message contains a valid command, the client should expect a reply as specified but should not wait forever for the reply, since server-client communication is asynchronous.

Each IRC message may consist of up to two parts: the command, and the command parameters. The command and all parameters are separated by one (or more) ASCII space character (s). Text will be inferred as a sequence of characters between quotes. Nicknames are inferred by the server from the connection messages are received from. The command must either be a valid IRC command or a three-digit number represented in ASCII text.

IRC messages are always lines of characters terminated with a CR_LF (Carriage Return – Line Feed) pair, and these messages shall not exceed 512 characters in length, counting all characters including the trailing CR_LF.

2.1.1 Message format

CR and LF are used to separate contiguous messages. The extracted message is parsed into the components <command> and list of parameters matched either by <middle> or <trailing> components.

The BNF representation is:

<message> ::= <command> <params> <crlf>
<command> ::= <letter> { <letter> } | <number> <number> <number>
<SPACE> ::= ' ' { ' ' }
<params> ::= <SPACE> [':' <trailing> | <middle> <params>]
<middle> := <Any *non-empty* sequence of characters not including SPACE or NUL or CR of LF, the first of which may not be ':'>
<trailing> := <Any possibly *empty* sequence of characters no including NUL or CR or LF>
<crlf> := CR LF

2.2 Numeric replies

Most of the messages sent to the server generate a reply. The most common reply is the numeric reply, used for both errors and normal replies. A numeric reply is not allowed to originate from a client.

3. IRC Concepts

Each client communicates only with the server. The server communicates with multiple clients. To manage the many connections, the server keeps a queue of received messages. Upon receiving a message from a client, the message will be placed in the queue. On a separate thread, the server parses the messages, carries out a command if requested, and returns a response to the originating client.

4. Message details

This section contains descriptions of each message recognized by the IRC server and client. The serer must implement all commands described here. The server must parse a message from a client and return appropriate errors.

4.1 Connection Registration

The commands described here are used to register a connection with an IRC server as a user as well as correctly disconnect. In order to register, the client must send a Nick message with a unique nickname as argument.

4.1.1 Nick message

Command: NICK
Parameters: <nickname>

NICK message is used to give a user a nickname. Nicknames cannot be changed during a session. If the server receives a NICK from a client that is already in use, it will issue an ERR_NICKNAMEINUSE to the

client, and the registration of the client will fail. A user's nickname will be forgotten once they unregister.

Numeric replies:

ERR_ALREADYREGISTERED
ERR_NICKNAMEINUSE
ERR_ERRONEUSNICKNAME
ERR_NEEDMOREPARAMS
RPL_REGISTERSUCCEEDED

Example:

NICK Wiz ; Introducing new nick "Wiz".

4.1.2 Quit message

Command: QUIT
Parameters:

A client session is ended with a QUIT message. The server must close the connection a client that sends this message. If the server detects that a client connection has closed without issuing a QUIT command (if a message fails to send, for example), it will unregister the client, and remove it from any rooms it had joined.

Numeric Replies:

RPL_QUITSUCCEEDED

Example:

QUIT ; "Wiz" has quit.

4.2 Room operations

This group of messages is concerned with manipulating rooms.

4.2.1 Create message

Command: CREATE
Parameters: <room name>

The CREATE command is used by a client to create a room. If a room with the requested name already exists, the server will issue a ERR_ROOMNAMEINUSE error and the room will not be created. If the command succeeds, the creator will automatically join the room.

Numeric replies:

ERR_NOTREGISTERED
ERR_NEEDMOREPARAMS
ERR_ROOMNAMEINUSE
ERR_ERONEOUSROOMNAME
ERR_TOOMANYROOMS

RPL_CREATESUCCEEDED
RPL_JOINSUCCEEDED

Examples:
CREATE myroom ; room “myroom” has been created.

4.2.2 Join message

Command: JOIN
Parameters: <room name>

The JOIN command is used by a client to start listening to messages from a specific room. As long as the room has been created and is not full, the user will be allowed to join. Once a user has joined a room, they receive notice about all commands the server receives which affect the room. This includes JOIN, PART, QUIT, and MSG.

Numeric replies:
ERR_NOTREGISTERED
ERR_NEEDMOREPARAMS
ERR_ALREADYJOINED
ERR_NOSUCHROOM
ERR_ROOMISFULL
RPL_JOINSUCCEEDED

Examples:
JOIN myroom ; join room “myroom”

4.2.3 Part message

Command: PART
Parameters: <channel>

The PART message causes the client sending the message to be removed from the list of users for the room in the parameters string.

Numeric replies:
ERR_NOTREGISTERED
ERR_NEEDMOREPARAMS
ERR_NOSUCHROOM
ERR_NOTINROOM
RPL_PARTSUCCEEDED

Examples:
PART myroom ; leave room “myroom”

4.2.4 Names message

Command: NAMES
Parameters: <room>

Using the NAMES command, a user can list all nicknames of users that are in the room given by the parameter.

Numeric replies:
ERR_NEEDMOREPARAMS
ERR_NOSUCHROOM
RPL_NAMESSTART
RPL_NAMES
RPL_NAMESEND

Example:
NAMES myroom ; list all users currently in “myroom”

4.2.5 List message

Command: LIST
Parameters:

The list message is used to list all existing rooms.

Numeric replies:
RPL_LISTSTART
RPL_LIST
RPL_LISTEND

Example:
LIST ; List all rooms

4.3 Sending messages

The purpose of the IRC protocol is to allow clients to communicate with each other. MSG is the only message available which performs delivery of a message from one client to another.

4.3.1 Msg message

Command: MSG
Parameters: <room> <text>

MSG is used to send messages to all the users in the room listed in the parameters.

Numeric replies:
ERR_NOTREGISTERED

ERR_NOSUCHROOM
ERR_NEEDMOREPARAMS
ERR_NOTINROOM
RPL_MSGSUCCEEDED

Example:
MSG myroom :Hello everyone! ; Message to everyone in room “myroom”

5. Replies

The following is a list of numeric replies which are generated in response to the commands given above. Each numeric is given with its number, name and reply string. Each numeric can have one or more arguments, which add additional information about what happened at the server.

5.1 Error replies

402 ERR_NOSUCHROOM
- Indicates that a command failed because the indicated room did not exist
- The first argument provides the command name
- The second argument provides the requested room name

403 ERR_NOTINROOM
- Indicates that a command failed because the client is not in the indicated room
- The first argument provides the command name
- The second argument provides the requested room name

404 ERR_ALREADYJOINED
- Indicates that the join request failed because the client is already in the indicated room
- The first argument provides the requested room name

405 ERR_TOOMANYROOMS
- Indicates that the create request failed because the server has reached its room limit
- The first argument provides the requested room name

406 ERR_ROOMNAMEINUSE
- Indicates that the create request failed because a room with that name already exists
- The first argument provides the requested room name

407 ERR_ERONEOUSROOMNAME
- Indicates that the create request failed because the room name was not allowed
- The first argument provides the requested room name

408 ERR_UNKNOWNCOMMAND
- Indicates that the command sent by the client was not recognized by the server
- The first argument provides the requested command

409 ERR_ERRONEOUSNICKNAME
- Indicates that the register request failed because the requested nickname was not allowed
- The first argument provides the requested nickname

410 ERR_NICKNAMEINUSE
- Indicates that the register request failed because the requested nickname was not allowed

- Indicates that the register request failed because a client with that nickname is already registered

411 ERR_NOTREGISTERED
- Indicates that the command failed because the client is not registered
- The first argument indicates the command

412 ERR_NEEDMOREPARAMS
- Indicates that the command failed because not enough parameters were provided
- The first argument indicates the command

413 ERR_ALREADYREGISTERED
- Indicates that the register request failed because the client was already registered
- The first argument provides the registered nickname

414 ERR_ROOMISFULL
- Indicates that the join request failed because the room has reached its member limit
- The first argument provides the room name

5.2 Command responses

300 RPL_LISTSTART
- Indicates that the server will start sending room names

301 RPL_LIST
- Indicates a room name in the list of rooms.
- The first argument provides the room name

302 RPL_LISTEND
- Indicates that the server has finished sending room names

303 RPL_NAMESSTART
- Indicates that the server will start sending client nicknames
- The first argument provides the room name

304 RPL_NAMES
- Indicates a client nickname in the specified room
- The first argument provides the room name
- The second argument provides the client nickname

305 RPL_ENDOFNAMES
- Indicates that the server has finished sending client nicknames
- The first argument specifies the room name

306 RPL_REGISTERSUCCEEDED
- Indicates that the register request succeeded
- The first argument provides the nickname

307 RPL_CREATESUCCEEDED
- Indicates that the create request succeeded
- The first argument provides the room name

308 RPL_JOINSUCCEEDED
- Indicates that the join request succeeded
- The first argument provides the room name

309 RPL_PARTSUCCEEDED
- If one argument is sent:
    - Indicates that that the part request succeeded
    - The first argument provides the room name

- If two arguments are sent:
    - Indicates that a different client parted from a room
    - The first argument provides the room name
    - The second argument provides the client nickname

310 RPL_QUITSUCCEEDED
- Indicates that the quit request succeeded

311 RPL_MSGSUCCEEDED
- Indicates that a client sent a message to a room
    - The first argument provides the client nickname
    - The second argument provides the room name
    - The thirst argument provides the message

312 RPL_SERVERSHUTDOWN
- Indicates that the server has shut down