CS594 Radha Natesan
Internet Draft
Intended status: IRC Class Project Specification October 21, 18
Expires: December 2018

Internet Relay Chat Class Project
draft-irc-pdx-cs594-00.txt

Status of this Memo

Copyright Notice

Abstract

This memo describes the format of the messages that the client and server will exchange in order to implement the IRC application. The central server relays the messages sent to it to all other connected users.

Table of Contents

1. Introduction

This document specifies the protocol for Internet Relay Chat (IRC) application that lets multiple users exchange group or private messages. Users are given the privilege to create a room, join a room, leave a room, list available rooms, send messages to a room which will be delivered to all members in that room and send private messages to any member in a room.

2. Conventions used in this document

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 0.

3. Basic Information

The client and server processes communicate via port 8000 using a persistent TCP connection.

4. Message Infrastructure

4.1. Message Format

```
IRC-PKT = {'Header': {'OP-CODE': <Operation Code>,
                      'Length': <Payload size>},
           'Payload': <Data>}
```

4.1.1. Field Definitions:

An IRC packet comprises the packet header and payload.

- The packet header has the operation code to indicate the action requested and Length specifies the size of the user data.
- The packet Payload contains the actual user data.

4.1.2. Operation Codes

```
IRC_OPCODE_CONNECT          = 1
IRC_OPCODE_OK               = 2
IRC_OPCODE_JOIN             = 3
IRC_OPCODE_SEND_MSG         = 4
IRC_OPCODE_SHOW_MSG         = 5
IRC_OPCODE_LIST_ROOM        = 6
IRC_OPCODE_LIST_ROOM_RESP   = 7
IRC_OPCODE_LEAVE            = 8
IRC_OPCODE_LIST_USERS       = 9
IRC_OPCODE_LIST_USER_RESP   = 10
IRC_OPCODE_QUIT             = 11
IRC_OPCODE_SEND_GROUP_MSG   = 12
IRC_OPCODE_SEND_PRV_MSG     = 13
IRC_OPCODE_SERVER_SHUTDOWN  = 14
```

4.2. Error Message Format

```
IRC-PKT-ERROR = {'Header': {'OP-CODE': <Error opcode>,
                            'Length': <Payload size>},
                'Payload': <Error code>}
```

4.2.1. Usage:

Error Messages are sent whenever there is a connection failure or to report any invalid operation or error encountered by the server.

4.2.2. Field Definitions:

- The 'OP-CODE' field is set to Error opcode
- The Payload field contains one of the predefined error codes.

4.2.3. Error Codes:

```
IRC_ERROR_ILLEGAL_NAME          = 201
IRC_ERROR_NAME_ALREADY_EXISTS   = 202
IRC_ERROR_USER_LIMIT_REACHED    = 203
IRC_ERROR_ROOM_LIMIT_REACHED    = 204
IRC_ERROR_MALFORMED_REQUEST     = 205
IRC_ERROR_ILLEGAL_REQUEST       = 206
IRC_ERROR_ROOM_NOT_FOUND        = 207
IRC_ERROR_USER_NOT_FOUND        = 208
```

4.3. Keep Alive Messages

```
IRC-PKT-KEEP-ALIVE = {'Header': {'OP-CODE': IRC-OPCODE-KEEP-ALIVE,
                        'Length': 0}}
```

4.3.1. Usage:

The KEEP-ALIVE messages are sent by both the client and server to let each other know that they are still connected. They SHOULD send a KEEP-ALIVE message every 5 seconds and if the message is not received after a pre defined time, say 15 seconds, the connection is terminated.

5. User/Room Name Semantics

- The User/Room Name SHOULD have a minimum of 1 character and maximum of 20 characters.
- The name MUST be of readable ASCII character values.
- If any of the naming semantics are not met, an error code IRC_ERROR_ILLEGAL_NAME will be sent.

6. Client Messages

6.1. Connection request to the server

```
IRC-PKT-CONNECT = {'Header': {'OP-CODE': IRC_OPCODE_CONNECT,
                        'Length': <Payload size>},
             'Payload': {<User name>}}
```

6.1.1. Usage:

This is the first message the client sends to the server. The client should choose a user name that conforms to the User name semantics.

6.1.2. Field Definitions:

- The 'OP-CODE' field is set to IRC-OPCODE-CONNECT
- The Payload field contains the user name chosen by the client.

6.1.3. Response:

The server validates the <user name> field and responds with IRC_ERROR_ILLEGAL_NAME if the naming semantics are not met or IRC_ERROR_NAME_ALREADY_EXISTS if the <user name> is unavailable. In both cases, the client has to provide a different user name. If the request is successful, IRC_OPCODE_OK response is sent.

6.2. Listing rooms

```
IRC-PKT-LIST = {'Header': {'OP-CODE': IRC_OPCODE_LIST_ROOM,
                           'Length': 0}}
```

6.2.1. Usage:

This request is made to get a list of all the rooms that are currently active on the server.

6.2.2. Response:

The server responds with the operation code IRC_OPCODE_LIST_ROOM_RESP with the list of the rooms.

6.3. Joining and Creating rooms

```
IRC-PKT-JOIN = {'Header': {'OP-CODE': IRC_OPCODE_JOIN,
                           'Length': <Payload size>},
               'Payload': {'Room-Name': <Room name/Room list>,
                           'User-Name': <User name>}}
```

6.3.1. Usage:

This request is made by the client to join a room. The client specifies the <Room name> he wants to join. The client can also specify a list of rooms to join.

6.3.2. Response:

The server validates the <room name> field and responds with IRC_ERROR_ILLEGAL_NAME if the naming semantics are not met. If the Maximum room limit is already reached, then the server sends a IRC_ERROR_ROOM_LIMIT_REACHED response. If the room is too full to join, then the server sends a IRC_ERROR_USER_LIMIT_REACHED response.

Then, the client has to provide a different room name. If the request is successful, IRC_OPCODE_OK response is sent.

6.4. List users

```
IRC-PKT-LIST-USER = {'Header': {'OP-CODE': IRC_OPCODE_LIST_USERS,
                        'Length': <Payload size>},
            'Payload': {'Room-Name': <Room name>}}
```

6.4.1. Usage:

This request is made by the client to get the list of users in the specified room.

6.4.2. Response:

The server validates the room name and sends IRC_OPCODE_LIST_USER_RESP response to all users in that room. If the room name specified by the user is not found, then the server responds with IRC_ERROR_ROOM_NOT_FOUND error message. In case of error responses, the client has to rectify the error and send a new request.

6.5. Leave a room

```
IRC-PKT-LEAVE = {'Header': {'OP-CODE': IRC_OPCODE_LEAVE,
                        'Length': <Payload size>},
            'Payload': {'Room-Name': <Room name/Room list>,
                        'User-Name': <User name>}}
```

6.5.1. Usage:

This request is made by the client to leave a room he is part of.

6.5.2. Response:

If the room name specified by the user is not found, then the server responds with IRC_ERROR_ROOM_NOT_FOUND error message. In case of error responses, the client has to rectify the error and send a new request. If the request is successful, IRC_OPCODE_OK response is sent.

6.6. Send message

```
IRC-PKT-SEND-MSG = {'Header': {'OP-CODE':
IRC_OPCODE_SEND_MSG/IRC_OPCODE_SEND_GROUP_MSG,
                        'Length': <Payload size>},
```

```
                'Payload': {'Room-Name': <Room name/Room List>,
                            'User-Name': <User name>,
                            'Message': <Message>}}
```

6.6.1. Usage:

This request is made by the client to send a message to all users in a specific chat room.

6.6.2. Response:

The server validates the room name and sends a IRC_OPCODE_SHOW_MSG message to all the users of that specific chat room. If the room name specified by the user is invalid, then the server responds with IRC_ERROR_ROOM_NOT_FOUND error message. In case of error responses, the client has to rectify the error and send a new request.

6.7. Send private message

```
IRC-PKT-SEND-PRIV-MSG = {'Header': {'OP-CODE':
IRC_OPCODE_SEND_PRV_MSG,
                            'Length': <Payload size>},
                'Payload': {'User-Name': <user name>,
                            'To-User': <target user name>,
                            'Message': <Message>}}
```

6.7.1. Usage:

This request is made by the client to send a private message to a user.

6.7.2. Response:

The server validates the target user name and sends a IRC_OPCODE_SHOW_MSG message to the target user. If the user name specified by the user is invalid, then the server responds with IRC_ERROR_USER_NOT_FOUND error message. In case of error responses, the client has to rectify the error and send a new request.

6.8. Quit IRC

```
IRC-PKT-QUIT = {'Header': {'OP-CODE': IRC_OPCODE_QUIT,
                            'Length': <Payload size>},
                'Payload': {'User-Name': <user name>}}
```

6.8.1. Usage:

This request is made by the client to quit from IRC application.

6.8.2. Response:

The server sends a IRC_OPCODE_OK message to the user and deletes all the user details from the server.

7. Server Messages

7.1. List Response

```
IRC-PKT-LIST-RESP = {'Header': {'OP-CODE': IRC_OPCODE_LIST_ROOM_RESP
|| IRC_OPCODE_LIST_USER_RESP,
                        'Length': <Payload size>},
            'Payload': {'Message': <List of rooms or users>}}
```

7.1.1. Usage:

This response is sent when the client makes a IRC_OPCODE_LIST_ROOM_RESP or IRC_OPCODE_LIST_USER_RESP request.

7.1.2. Field Definitions:

- The 'OP-CODE' field is set to IRC_OPCODE_LIST_ROOM_RESP or IRC_OPCODE_LIST_USER_RESP
- The Payload field contains the array of users or room depending on the request received.

7.2. Forward message to user

```
IRC-PKT-SHOW-MSG = {'Header': {'OP-CODE': IRC_OPCODE_SHOW_MSG,
                        'Length': <Payload size>},
            'Payload': {'User-Name': <Target user>,
                        'From-User': <from user name>,
                        'To-Room': <To room>,
                        'Message': <Message sent by the user>,
                        'Type': <Reserved for later use>,}}
```

7.2.1. Usage:

This response is sent when the client makes a IRC_OPCODE_SEND_MSG/IRC_OPCODE_SEND_GROUP_MSG/ IRC_OPCODE_SEND_PRV_MSG request.

7.2.2. Field Definitions:

- The 'OP-CODE' field is set to IRC_OPCODE_SHOW_MSG
- The 'From-Name' is set to the name of the user originally sending the message.
- The 'User-Name' is set to the target user

- The 'To-Room' is to specify the target room in case of group message.
- The 'Message' field contains the message that has to be forwarded to a room or user.

7.3. Server shutdown message

```
IRC-PKT-LIST-RESP = {'Header': {'OP-CODE':
IRC_OPCODE_SERVER_SHUTDOWN,
                        'Length': <Payload size>},
            'Payload': {' User-Name': <Target user>}}
```

7.3.1. Usage:

This message is sent when the server decides to shut down its service.

8. Error handling

Whenever there is a connection error, both the server and the client SHOULD detect it (using KEEP-ALIVE messages) and act upon it. If server identifies that a client is not responding, then it should remove the user from all the participating chat rooms/private message session. If a client loses connection with the server, then it MAY consider reconnecting later.

9. Crash handling

The server handles client crashes by identifying the connection that is not responding using the select (timeout) method of the selector module, which waits for I/O readiness notification by polling.

The client handles server crashes by handling the 'ConnectionRefusedError' exception thrown while trying to read from the port with no server listening.

10. Extra Credits

10.1. Private Messages

Private messaging is already supported in the design.

10.2. Secure Messaging

Secure Messaging is achieved by encrypting the data on the client side using AES encryption in python and decrypting on the server end.

10.3. File Transfer

The file transfer is done by, breaking the data into multiple segments of size 1024 and sending it over a TCP connection. The server can reassemble the segments received to get the entire file.

10.4. Cloud Connected Server

The server is configured to use the Amazon AWS Platform to store and retrieve chat information

11. Security Considerations

<Add any security considerations>

12. IANA Considerations

None

12.1. Normative References

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

13. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.