Internet Relay Chat Protocol Specification

Yiming Lin

May 30th, 2020

1. Introduction

This specification describes a simple Internet Relay Chat (IRC) protocol by which clients can communication with each other. The system employs a centralized server that relay clients' messages to other clients.

User can join multiple rooms and leave rooms, send distinct messages to different rooms, send messages to selected users. All the users joined in a room can receive all the joined users' messages that are sent to the joined room.

2. Basic Information

All the communication described in this protocol take over TCP/IP with the server listening for connection on port number specified by server administrator. Clients connect to this port will maintain persistent TCP connection to the server,

The client and server are sending messages asynchronously. The client and server can send messages at any time and receive message at any time.

The server and client may terminate connection at any time with any reason. They may send message to indicate disconnection status.

3. Message Infrastructure

3.1. Operation Codes

Registration	= 00001
Join	= 00002
Room_Message	= 00003
Private_Message	= 00004
Leave_Room	= 00005
List_Room_Users	= 00006
List_Rooms	= 00007
User_Disconnect	= 00010

3.2. Status Codes

200	Success
400	Bad command
401	Duplicated client registration
402	Duplicated username in registration

403	Invalid format for username and room name
420	Client at given address has not registered
450	Client leave a not existed room
451	Client leave a room that is not in
461	Username to disconnect not found
462	Client address given not found when disconnecting
496	Private message receiver not found
497	Room to receive message not found
498	Duplicated joining a room
499	Username does not exist when joining room

4. Label Semantics

Identify users and rooms involving sending and receiving labels, all the labels have the same rule. During user registration, and room creation and joining, the username and room names sending to server MUST be validated as follows:

- The length of username and room name MUST always be 20 characters.
- The username and room name MUST not contain '$', '#', or '&'.
- If any of the above two rules is broken, the server will return corresponding 4xx code to indicate invalid format, and user registration, room creation and joining MUST not success.
- If server receives invalid format of username or room name when client sending message, leaving room, or disconnecting from server, a 4xx code corresponding to "room not found" or "username not found" MUST return to indicate username or room name does not exist in server.

5. Client Messages

This section describes messages that client composes to send to server. Each message is described by a C-styled structure and a "message" bytes array that is the actual message format that MUST be sent to server.

The bytes array is denoted by **b''** where strings are placed inside quotes and concatenated by "+".

5.1. Client Registration

```
struct registration {
        string username[20];
        string command_code = Registration;
}
message = b'$ + command_code + username + $'
```

5.1.1. Usage

Before subsequent message can be sent to server, client MUST send a registration message to server. The server MUST send a message back to client to indicate whether registration success. If success, client can send other kinds of messages. If not success, client MUST send a registration message again until server send back a message that indicate successfully registered.

Once registration successes, server insert client's username and address into database.

Registration message SHOULD NOT be sent once server has already notified client that registration is success. If client sends registration messages after successful registration, the server MUST reply to it by sending back a 403 code if username provided is invalid, or a 402 if username already been registered, or 401 if client's address has already in server's database.

5.1.2. Field Definition

- username is the client identifier in the system, client CAN choose whatever name they wish, but MUST follow label semantics.
- command_code identifies the type of command in server.

5.2. Joining and Creating a room

```
struct join {
        string room_name[20];
        string username[20];
        string command_code = Join;
}
message = b'$ + command_code + room_name + username + $'
```

5.2.1.Usage

Send a message to server to create a room or join a room for a given room name in the message. If room does not exist, server creates a room with the given name. Otherwise, server add client to the room and client SHALL receive all the messages sent to this room.

When the format of the room is invalid (as described in label semantics), a 403 code is sent back to client. When the username does not exist in server database, a 499 code is sent back to client. If 403, 420, or 499 code is sent back to client by server, client MUST not be added to any room. Otherwise, a 200 code is sent back to client, and client is added to room provided.

5.2.2.Field Definition

- Room name is the name of the room to join or create, which MUST follow label semantics.

- username is the name of the user to join the room, which MUST follow label semantics.
- command_code identifies the type of command in server

5.3. Sending Message to Rooms

```
struct room_message {
        list room_name;
        string message;
        string room_num[2];
        string command_code = Room_Message;
}
message = b'$ + command_code + room_num + room_name[0] + room_name[1] + … + room_name[len(room_name) - 1] + message + $'
```

5.3.1. Usage

Send a message to given list of rooms. The length of the room_name MUST be less than 100. All the room name provided in room_name list must be valid and existed in server database. Once there is a room_name that does not exist in server, the message MUST not be sent to any of the rooms in room_name successfully, and server MUST send a 497 code back to indicate room not found.

The room_num indicates how many room names are provided in room_name list. The room_num is a string with length 2, if the number of room names is less than 10, it MUST pad a character '0' before the number. For example, if the length of room_name is 5, then room_num = '05'.

If room_num provided is not equal to the actual length of room_name, then a code 410 is send back to client, which indicates inconsistent argument length and actual item number.

If all the room names provided in room_name list exist in server, also the room_num is same as the actual length of the room_name list, the message MUST send to all rooms provided in room_name list.

If the sender client does not join the room it sends message to, the client cannot receive a message from server that indicates message is sent to rooms successfully. The sender client SHALL receive server message with code 410, and 497.

5.3.2. Field Definition

- room_name is a list of room names that receive message

- message is the content of the message that rooms to receive
- room_num is the length of room_name list represented by a string with length 2
- the command_code identifies the command type in server

5.4. Sending Message to Users

```
struct private_message {
        list usernames;
        string message;
        string num_users[2];
        string command_code  = Private_Message;
}
message = b’$ + command_code + num_users + usernames[0] + & + usernames[1] + & + … + & + usernames[len(usernames) – 1] + # + message + $’
```

5.4.1. Usage

Send a private message to given list of users. The length of the usernames MUST be less than 100. All the room name provided in usernames list must be valid and existed in server database. Once there is a username that does not exist in server, the message MUST not be sent to any of the users in username successfully, and server MUST send a 496 code back to indicate username not found.

The num_users indicates how many usernames are provided in usernames list. The num_users is a string with length 2, if the number of usernames is less than 10, it MUST pad a character ‘0’ before the number. For example, if the length of usernames is 5, then num_users = ‘05’.

If room_num provided is not equal to the actual length of room_name, then a code 410 is send back to client, which indicates inconsistent argument length and actual item number.

If all the usernames provided in usernames list exist in server, also the num_users is same as the actual length of the usernames list, the message MUST send to all users provided in usernames list.

The sender shall receive a 200 code sent from server to indicate private message sending successfully or code 496, and 410 to indicate corresponding errors.

5.4.2.Field Definition

- usernames is a list of usernames that receive message
- message is the content of the message that users to receive

- num_users is the length of usernames list represented by a string with length 2
- the command_code identifies the command type in server

5.5. Leaving a Room

```
struct leave {
        string room_name[20];
        string username[20];
        string command_code = Leave_Room;
}
message = b'$ + command_code + room_name + username + $'
```

5.5.1. Usage

Send a message to server to leave a room a user joined. The room name and username MUST exist in server database.

If username does not exist in room with name room_name, a 451 code is sent back to client; if room name does not exist in server database, a 450 code is sent back to client to indicate room does not exist.

Otherwise, if a user successfully leaves a room, all the users joined that room SHALL receive a message to notify that the user has already left the room.

5.5.2. Field Definition

- room_name is the name of the room to leave
- username is the name of the user to leave
- command_code identifies the type of command in server

5.6. Listing Users in a Room

```
struct list_room_users {
        string room_name[20];
        string command_code = List_Room_users;
}
message = b'$ + command_code + room_name + $'
```

5.6.1. Usage

Send a message to user to get a list of users joined in a given room. If room name does not exist in server database, a code 451 is sent back to client. Otherwise, a list of usernames in that room is sent back to client.

5.6.2. Field Definition

- room_name is the name of room to get a list of users
- command_code identifies the type of command in server

5.7. List Rooms Existed in Server

```
struct list_rooms {
        string command_code = List_Rooms;
}
message = b'$ + command_code + $'
```

5.7.1. Usage

Send a message to server to get a list of rooms existed in server database.

5.7.2. Field Definition

- command_code identifies the type of command in server

5.8. Disconnection

```
struct disconnect {
        string username[20];
        string command_code = User_Disconnect;
}
message = b'$ + command_code + username + $'
```

5.8.1. Usage

Send a message to server to disconnect user given username. If username does not exist in server database, a 461 code is sent back to client and MUST not disconnect any client. Otherwise, the name of user to disconnect will be notified to all the room it has already joined and the user SHALL disconnect from server.

5.8.2. Field Definition

- username is the name of client to disconnect
- command_code identifies the type of command in server

6. Server Messages

6.1. Message Forwarding

```
struct message {
        string code[3];
        string message;
        string to_room[1];
        string sender[20];
        string room[20];
        string receiver[20];
        string data;
        string command_code[5];
}
message = b'$ + code + command_code + to_room + sender + # + room + # data
+ # message + $'
```

6.1.1. Usage

Server forwards a message to client. Server MUST specifies whether the message is sent to a room or to a user. If sends to room, to_room MUST set to '1', and receiver MAY be an empty string or something else ; otherwise, to_room MUST set to '0', and room MAY be an empty string or something else.

If the sender, room and receiver exist in server database, the code SHALL be '200' to indicate success; if not, the corresponding 4xx code is set, and corresponding reasons for 4xx code is placed in message field. The actual message to send is placed in data field.

6.1.2.Field Definition

- code is to indicate status; the code options are described in section 3.2.
- message is to store reasoning message for code field
- to_room indicates whether this message is a room message or private message
- sender is the client that sends the message
- room is the room name that is going to receive the room message
- receiver is the username that is going to receive the private message
- data is the actual message that the sender sends
- command_code identifies the type of command in server

6.2. List Users in Room

```
struct list_users {
        string code[3];
        string roomName[20];
        list users;
        string message;
        string command_code[5];
}
message = b'$ + code + command_code + roomName + user[0] + & + user[1]
+ … + & + user[len(users) – 1] + # + message + $'
```

6.2.1. Usage

Server send a message to client contains a list of users in a given room.

6.2.2. Field Definition

- code is to indicate status; the code options are described in section 3.2.
- message is to store reasoning message for code field
- roomName is the room name that is going to list users in the room
- users is a list of username that in the roomName
- command_code identifies the type of command in server

6.3. List Rooms

```
struct list_rooms {
        string code[3];
        list rooms;
        string message;
        string command_code[5];
}
message = b'$ + code + command_code + rooms[0] + & + room[1] + & + … + &
+ room[len(rooms) – 1] + # + message + $ '
```

6.3.1. Usage

Server send a message to client contains a list of room names in server database.

6.3.2. Field Definition

- code is to indicate status; the code options are described in section 3.2.
- message is to store reasoning message for code field
- rooms is a list of room names that exist in database
- command_code identifies the type of command in server

6.4. Join Room

```
struct join {
        string code[3];
        string message;
        string isCreation[1];
        string username[20];
        string roomName[20];
        string command_code[5];
}
message = b'$ + code + command_code + isCreation + roomName + username
+ # + message + $'
```

6.4.1. Usage

Server send a message to notify a client is joining a room. If the room is being created, the message is sent to creator; otherwise, the message is sent to all the users in that room. If the room is being created, the isCreation field is set to '1'; otherwise, it is set to '0'.

6.4.2. Field Definition

- code is to indicate the status; the code options are described in section 3.2.
- message is to store reasoning message for code field
- roomName is the room being created or being joined
- username is the name of the client that is joining or creating the room
- command_code identifies the type of command in server

6.5. Leave Room

```
struct leave {
        string code[3];
        string message;
        string roomName[20];
        string username[20];
        string command_code[5];
}
message = b'$ + code + command_code + roomName + username + # + message
+ $'
```

6.5.1. Usage

Server send a message to notify a client is leaving a room.

6.5.2. Field Definition

- code is to indicate the status; the code options are described in section 3.2.
- message is to store reasoning message for code field
- roomName is the room the client is leaving
- username is the name of the client that is leaving the room
- command_code identifies the type of command in server

6.6. Client Disconnection

```
struct disconnect {
        string code[3];
        string message;
        string username[20];
        string roomName[20];
        string command_code[5];
        tuple address;
}
message = b'$ + code + command_code + username + # + addr + # + roomName
+ # +message + $'
```

6.6.1. Usage

Server send a message to notify a client is disconnecting. When a client is disconnecting from server, the client SHALL send a message with command_code User_Disconnect and the server SHALL send a message to all the rooms the client is joining to notify that client is disconnecting. The address field MAY be not used, this field corresponds to 462 code. In current version, when server cannot find a client with given address when disconnecting, the server does nothing. When server cannot find a given address in server database, it is an internal error of server program rather than the error of application, since in order to send a message to server except registration the client must register a username, and that username as well as client's address MUST be inserted into server database.

When the client's address is found, the server MUST remove the username from all joined rooms, then remove username and corresponding address information from server before disconnecting. After that, this disconnect message MUST send to all the rooms the disconnecting user previously in to notify the rooms that this client is disconnecting.

When the client crashes, the server detects it and clear all the information about this client as described above. The server MUST send a message to rooms that the crashed client previously in to notify that the client is disconnecting.

6.6.2. Field Definition

- code is to indicate the status; the code options are described in section 3.2.
- message is to store reasoning message for code field
- roomName is the room the message is sending to
- username is the name of the client that is disconnecting
- command_code identifies the type of command in server

- address is the address and port number of the client

7. Error Handling

When client detects server is crashed, client SHALL close the connection. When server detects client is crashed, server MUST remove all the data related to the crashed client and close the connection.

8. Conclusion

This specification provides a message sending and receiving system for clients to communication with each other through a server.