Internet Relay Chat

Portland State University
Internetworking Protocols CS 494P

Ram Bhattarai
Sohan Tamang
June 2018

Abstract

This memo defines the communication protocol for IRC style client/server system. In this protocol, a client being any socket program is capable of connecting to the server. IRC protocol is a text based protocol.

Table of Contents

Project Description **3**

Architecture **3**

Server 3

Client 3

Channels 3

Services **4**

Join and Create a Channel 4

leave a Channel 4

List Channels 5

List Members 5

Send Message 6

Send Distinct Message 6

Private messaging 7

Quit the Chat 7

Error Handling **8**

Insufficient Commands 8

Invalid Channel 8

No Permission to Members List 8

Nickname Being Used 8

Invalid Nickname 9

Duplicate Join Request 9

Not a Member of Channel 9

Tools **9**

Socket Library 9

Chat Server 9

Chat Client 10

Thread Library 10

Server Thread 10

Client Thread 10

Time Library 10

Conclusion and Future Work **10**

Chat Application Optimization 10

Data Structures 10

Secure Messaging 10

File Transfer 11

GUI Interface 11

References **11**

1. Project Description

A simple internet relay chat(IRC)- application layer protocol that provides text based chat service. In this application, there is one central server which is responsible for handling all message requests from clients. A client can send messages to a channel or send messages privately. Clients are are allowed to send messages to various channels, join a channel, leave a channel at any time, list channels available,list members available and quit chat.

2. Architecture

IRC protocol follows client-server architecture model where server is active all the time waiting for clients. It uses TCP/IP protocol to communicate. The server listens to a port.

2.1. Server

Server establishes a connection with a port number and listens for incoming requests. When a server receives a request from client, it validates a requests and performs accordingly. Server is the backbone of IRC chat application.

2.2. Client

Client is the user of the application. A client is connected to the server using the designated port number. The client has an ability to send messages one-one or one-to many.

2.3. Channels

Named group consisting of one or more clients. The channel is created when the user supplies the name and the name of the channel they intend to join. If the channel does not exist, new channel is created. Channel names are string beginning with '#'. User will supply the nickname and it will take the user to a default channel. From the default channel, the user is able to join a channel. For this chat program, minimum of 10 users are allowed to join.

3. Services

3.1. Join and create a Channel

Command: /join
Parameters: <channel>{ <channel>}

Clients can join a channel by sending /join request to the server. The server creates a new channel if there is no channel available with that name.Server adds the user to the list of members in the channel. All the channel users are notified that a new user has joined the channel. The chat window will start displaying messages as soon as a user joins the channel. User can start typing to join the conversation.

3.1.1 Usage

The client sends a join request to the server to join a channel. If the channel does not exist new, channel is created. Upon joining a channel, an alert message is broadcasted in that particular channel letting the members know that new member has joined the channel. At any time the user leaves or joins, the members should get a notice. Client has the capacity to join multiple channels at a time.

Examples:
If the channel does not exists it creates the channel. Channels with same name are not possible in this approach

```
/join #foo                ; join/create channel foo.
/join bar                 ; join/create channel bar.
/join #foo #bar                ; join/create channels foo and bar.
/join foo bar             ; join/create channels foo and bar.
```

3.2. Leave a Channel

Command: /leave
Parameters: <channel>

Users can leave a channel at any time. Leave command is used by user to leave a channel. If a user leaves a channel, the server removes the user from the list of members in that channel. If the user wants to join the chat session again, it needs to send a request to the server. If all the users leave a channel, then the channel is removed from the channel list.

3.2.1. Usage

Client send a leave request to the server intending to leave a channel. Upon receiving the request, the server should remove the user from the appropriate channel and send an alert message to entire list of users. Client is ignored if the client is not a member of the particular channel.

Example:

```
/leave #foo                    ; leave channel foo
```

3.3. List Channels

Command: /list

Parameters: empty

List available channels during the chat session.

3.3.1. Usage

Client sends a request to server to get a list of channels that are available to join. Upon receiving the request, the server sends a reply to the client with the list of available channels.

Examples:

```
/list                          ; list all the channels available to join.
```

3.4. List Members

Command: /list
Parameters: <channel>

Lists available users of a channel.

3.4.1. Usage

Client sends a request to server with the channel name to know the members of the channel. Upon receiving the request, server verifies and checks if the user who is requesting is a member of that channel. If the client is not a member of the channel, then it rejects the requests and gives it a failure request indicating that the client is not a member of the channel.

Example:

```
/list #foo                    ;lists all members in channel foo
```

3.5. Send Message

Command: /channel
Parameters: <channel> [message]

User can send a message to a channel using /channel command.

3.5.1. Usage

User sends a message request to the server. Upon receiving the request, the server verifies the request to see if the message is coming from a valid user. Server also verifies that the user in the member of the channel before sending messages to its members.

Example:

```
/channel #foo hello world.        ; sends "hello world." message to all
                                    the members Of foo channel if the user
                                    is in the member list.
```

3.6. Send Distinct Message

Command: /channel
Parameters: <channel>{ <channel>} [message]

User has the ability to send distinct messages to multiple channels.

3.6.1. Usage

User sends a distinct message with list of channel names. Upon receiving the request, the server will check and see if the client who is requesting is a member of that channel. If the client is not a member of the channel, the error message is displayed on client screen indicating that the client

is not a member of the channel. If the request is valid, the message is broadcasted over to multiple channels.

Examples:

```
/channel #foo #bar hello world.          ; sends "hello world." message to
                                           the members of foo and bar.
/channel #foo #bar #psu hi               ; sends "hi" message to all the
                                           Members of foo, bar and psu
                                           Channels.
```

3.7. Private Messaging

Command: /prvmsg
Parameters: <nickname> [message]

User can send private message to any valid user in the server. /prvmsg command is used to send a private message.

3.7.1. Usage

User sends a private message request to server. Server receives the request and validates if the receiver nickname exists in the server. Upon receiving valid nickname, server sends the message to the receiver.

Examples:

```
/prvmsg bill hi                          ; sends "hi" message to bill.
/prvmsg joe how are you?                 ; sends "how are you?" message to
joe.
```

3.8. Quit the Chat

Command: /quit
Parameters: empty

User can quit anytime anytime using /quit command. Server removes the user from all the channels and also from its connections list.

3.8.1. Usage

When user disconnects from a server, "user disconnected" message is send to all the channels that the user was connected to. This process is done only after removing the user form the channels first.

Example:

```
/quit                          ; user quits the chat.
```

4. Error Handling

4.1. Insufficient Commands

An error - Insufficient commands - is emitted on client screen if the client does not provide enough parameters for commands.

Example: Channel name not given when sending message.

```
   /channel                    ;"Insufficient commands" error.
```

4.2. Invalid Channel

An error- Invalid channel- is displayed on client screen if the client provides invalid channel name when sending a message

Example: Channel #foo does not exists.

```
   /channel #foo hello         ; "Channel #foo does not exists" error.
```

4.3. No Permission to Members List

An error- No permission to view list of users- is displayed if the client does not belong to the channel.

Example: User in not a member of channel #foo.

```
   /list #foo             ; "No permission to view list of users"
                                 Error.
```

4.4. Nickname Being Used

An error- Nickname being used - is shown to client screen if the nickname provided is already taken by some other user.

Example: Nickname bill is already taken by other client.

```
   /nick bill             ; "Nickname being used" error.
```

4.5. Invalid Nickname

An error -No such nickname- is shown to the user screen if the nickname provided does not exists during private messaging.

Example: User with nickname “harry” does not exists.

```
/prvmsg harry hi          ; “No suck nickname” error.
```

4.6. Duplicate Join Request

An error -Already joined channel- is shown to the user if they is tries to send duplicates join request to a same channel.

Example: A user have already joined #foo channel.

```
/join #foo          ; “You have already joined #foo” error.
```

4.7. Not a Member of Channel

An error -Not a member of channel- is shown to the user if they try to send message to a channel which they have not joined yet.

Example: A user is not a member of channel #foo.

```
/channel #foo hi guys.              ; “You are not a member of #foo”
                                      Error.
```

5. Tools

The program is written in Python. Unix socket library is used to interface two nodes on a network. Server forms a listener socket and client reaches out to the server.

5.1. Socket Library

A built in socket Library (socket) is a medium for sending and receiving data between network endpoints.

5.1.1. Chat Server

A server creates a socket, binds the socket and listens for any incoming request.

5.1.2. Chat Client

A chat server creates a socket and connects to it and starts sending request to server.

5.2. Thread Library

This Library provides low-level primitives for working with multiple threads.

5.2.1. Server Thread

One thread listens for incoming messages and the another thread handles the new client coming in.

5.2.2. Client Thread

One thread sends messages and the other thread receives messages from server.

5.3. Time Library:

Time Library is used to print out the current time. Every communication happening in a chat program will have time. It is used to timestamp all the incoming messages.

6. Conclusion & Future work

6.1. Chat Application Optimization

The chat application could be optimized to improve the run time efficiency. Since this is a small chat program and does not involve lot of overhead, it seems like the program is running fast. In an environment where there are millions of users, optimization is needed.

6.2. Data Structure

Data structure such as Hash Map could be used to store all the connection information. Anytime you need to access the user information or channel information, you can get it instantly.

6.3. Secure Messaging

Currently, the communication is in plaintext which is extremely insecure. Encrypted messaging system could be applied to all the messages for secure messaging.

6.4. File transfer

Sometimes sending messages is not sufficient, so having a file transfer is better.

6.5. Graphical User Interface (GUI) chat

Currently, to see who is a member of channel and what channels are available, you have to type it in command line to see. Graphical interface gives you an advantage of separate and easily switchable channels.

7. References

https://docs.python.org/2/howto/sockets.html
https://piazza.com/pdx/spring2018/cs494594/resources
https://tools.ietf.org/html/rfc1459