CS594 C.Bao,D.Lee
Internet Draft Portland State University
Intended status: IRC Class Project Specification June 1,2019
Expires: December 2019

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

Status of this Memo

Copyright Notice

Abstract

This memo describes the communication protocol for an IRC-style client/server system project for the CS594:Internetworking Protocols Course at Portland State University.

Table of Contents

1. Introduction

This specification describes a simple Internet Relay Chat (IRC) protocol in which clients can communicate with each other with text-based communications. This is central based system where the server relays messages from active client to active client through the server.

Clients can join rooms, with groups of users as well as create rooms for others to join. The client can also leave rooms and the server. The messaging system also has a private function which allows user to send messages that cannot be viewed by other users except as designated. A user can also send a message to room even if the user is not within with the built-in client functionality.

For every user that connects to the server they are greeted from the server their first command of ‘\help’ which allows the user to learn the commands of the system easily.

2. Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

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 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

In this document, the characters ">>" preceding an indented line(s) indicates a statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the portions of this RFC covered by these keywords.

3. Basic Information

All communication described in this protocol takes place over TCP/IP with the server listening for connections specified by the server. Clients connect to this port and maintain this persistent connection to the server. The client can send messages and requests to the server over this open channel, and the server can reply via the same. This messaging protocol is inherently asynchronous – the client is free to send messages to the server at any time, and the server may asynchronously send messages back to the client.

As is described in [4.2], both the server and client may terminate the connection at any time for any reason. They May choose to send an error message to the other party informing them of the reason for connection termination.

The server may choose to allow only a finite number of users and rooms, depending on the implementation and resources of the host system. Error codes are available to notify connecting clients that there is currently a high volume of users or groups accessing the server.

4. Message Infrastructure

4.1. Generic Message Format

The generic message format is the username and character string.

4.1.1. Field Definitions:

- user_name – specifies the username of the client to be used for the duration of the application until the user exits the client.
- IP_address – Specifies client IP address from command line at client runtime.
- Port – Stores and specifies the port number the client is connected to and must match the one of the server.

5. Client Messages

5.1. First message sent to the server

```
Server.send("name_coming")

time.sleep(0.5)
```

server.send(user_name)

5.1.1. Usage

Before subsequent messages can be sent, a connecting client MUST provide a unique chat name on the server. The first message sent to the server is “name_coming”.

The server MUST associate the client’s chat_name with the socket connection of the user. This message SHOULD be sent only once.

5.1.2. Field definitions

user_name – MUST not be the same name provided by any other currently connected client. If the name already exists, the server MUST return the error “The user name has already been used.” The successful connection of the client will yield the message “user_name has connected to the server” followed by “Welcome to the chatroom! Type \help or \? for commands”.

5.2. Creating Rooms

message == “\create”

5.2.1. Usage

With the “\create” command the user creates chat room for users to join.

5.2.2. Field Definitions

- message – Used to send the command over to the server to respond to the create command.

5.3. Listing Rooms

message == “\listroom”

5.3.1. Usage

This command sends a message to the server that recognizes the command as “\listroom” which will print out “Heres’s the list of available rooms: “ and will then print out the available rooms on the server by name in sequential order.

5.3.2. Field Definitions

message – Used to send the command over to the server to respond to the "\listroom" command.

5.4. Joining Rooms

message == "\join"

5.4.1. Usage

This sends a "\join" command to the server which allows a user to join a room that is active on the server. This sends the user_name to verify the client's eligibility to join the room.

5.4.2. Field Definitions

- message – Used to send the command over to the server to respond to the "\join" command.
- user_name – The name the user has chosen to be identified by for the duration of their time on the server. This is used to check if the user is already in the chat room or not.
- to_join – Used to store the name of the room the user wants to join.

5.5. Leave Rooms

message == "\leave"

5.5.1. Usage

This sends a "\leave" command to the server which allows a user to leave a room that is active on the server. This sends the user_name to verify the client's eligibility to leave the designated room.

5.5.2. Field Definitions

- message – Used to send the command over to the server to respond to the "\leave" command.
- user_name – Used to verify if the user is in the room.
- to_leave – Used to store the name of the room name the user has intent to leave to be sent over to the server for verification.

5.6. Member List

message == “\memberlist”

5.6.1. Usage

This sends a “\memberlist” command to the server which allows a user to display all users in the specified room. The user is prompted with the name of the room the user wants to display.

5.6.2. Field Definitions

- message – Used to send the command over to the server to respond to the “\memberlist” command.
- to_list – Used to store the name of the room name the user has intent to display members of to be sent over to the server for processing.

5.7. Select Room Private Messaging

message == "\selectroom”

5.7.1. Usage

This command prompts the user to be able to send a private message to a designated chat room.

5.7.2. Field Definitions

- message – Used to send the command over to the server to respond to the “\selectroom” command.
- to_select – Used to store the name of the room name the user has intent to send a message to.

5.8. Quit

message == "\quit"

5.8.1. Usage

This command sends a message to the server of the client's intent to disconnect from the server with the "\quit" message.

5.8.2. Field Definitions

- message – Used to send the command over to the server to respond to the "\quit" command.
- user_name – Used to store as reference for user that wants to leave the server.
- message – Used to store any message that comes from the client.

5.9. Help Definition

5.9.1. Usage

This command allows users unfamiliar with the client to read all the commands that is available to the user. The user is given the command prompts that are available to them if they choose to use the command.

5.9.2. Field Definition

- message – Used to store any message that comes from the client.

5.10. Private Message

message == "\private"

5.10.1. Usage

This command is used to send a message to the server the intent of the user to send a private message to a specific user with the "\private" command. The server will send a response back prompting for the username of the target user. An invalid entry for the target user will yield the following message: "There's no such user"

5.10.2. Field Definitions

- message – Used to store any message that comes from the client.

5.11. Print All

message == "\printall"

5.11.1. Usage

This command is used to send a message to the server to respond with a full list of all the rooms with a listing of users within for each room.

5.11.2. Field Definition

- message - Used to store any message that comes from the client.

5.12. Server Side Shutdown

message == "\shutdown"

5.12.1. Usage

This command allows a client which SHOULD be an admin to shut down the server. The "\shutdown" command prompts the server to ask the client for a password in which only trusted users SHOULD know.

5.12.2. Field Definitions

- message - Used to send the command over to the server to respond to commands.
- password - Used to compare to password stored on server side.

6. Server Messages

6.1. First message received by the server

"name_coming"

6.1.1. Usage

Before subsequent messages can be received, a connecting client MUST provide a unique chat name on the server. The first message sent to the client is if the name already exists, the server MUST return the error "The user name has already been used." The successful connection of the client will yield the message "user_name has

connected to the server" followed by "Welcome to the chatroom! Type "\help" or "\?" for commands".

6.1.2. Field Definitions

- Message – Used to store any message that comes from the client.
- client_conn – Used to store valid connection information of clients connecting to the server.
- name_tag – Used by the server to determine that the next subsequent message is name prepared by the client.
- close() – Used to terminate connection if the user_name has already been in use.

6.2. Creating Rooms Response

message == "\create"

6.2.1. Field Definitions

- list_of_rooms – Used to keep record of active rooms inside the server.
- room_num – Used to name the room by assigning a number in naming it which is sequential based on the length of the list_of_rooms.
- room_name – Used to generate and store the name of the chat room on the chat server to be appended to the list_of rooms.

6.3. Listing Rooms Response

message == "\listroom"

6.3.1. Usage

User to interpret the "\listroom" command from the client to then list all the rooms currently available on the server as a message back to the client.

6.3.2. Field Definitions

- list_of_rooms – Used to keep record of active rooms inside the server.
- message – Used to store any message that comes from the client.

6.4. Joining Rooms Response

message == "\join"

6.4.1. Usage

This command interprets the client's intent to join an active chat room on the server. This verifies the client's user_name and the validity of the room the client wants to join before allowing the client to join the designated chat room. If the client is already in the room the server will send the following message to the client as an error message: "You are already in that room." The user is allowed to join multiple rooms using this command.

If the client is able to join the room then the server will send the following message to the client: "You have joined 'room name' 'user name' has been added to 'room name'".

6.4.2. Field Definitions

- join_label - Used by the server to show the client has intent to use the join command and further information is required from the client to be processed and the current stage in the process.
- name_to_join – Used to store the client's user name for the join command.
- list_of_rooms – Used to reference if the room the client wants to join is a valid room.
- message - Used to store any message that comes from the client.
- room – Used to store the room being compared to in question within the list of rooms.

6.5. Leave Rooms Response

message == "\leave"

6.5.1. Usage

This command interprets the client's intent to leave an active chat room on the server. This verifies the client's user_name and the validity of the room the client wants to leave before allowing the client to leave the designated chat room. If the client is not in the room the server will send the following message to the client as an error message: "You are not in that room. Can't leave."

If the client is able to join the room then the server will send the following message to the client: "You have left 'room name' 'user name' has left 'room name'".

6.5.2. Field Definitions

- leave_label - Used by the server to show the client has intent to use the leave command and further information is required from the client to be processed and the current stage in the process.
- name_to_leave – Used to store the client's user name for the join command.
- list_of_rooms – Used to reference if the room the client wants to leave is a valid room.
- message - Used to store any message that comes from the client.
- room – Used to store the room being compared to in question within the list of rooms.

6.6. Member List Response

message == "\memberlist"

6.6.1. Usage

This command is used to interpret the "\memberlist" command from the client to then list all the members in an active room designated by the user.

6.6.2. Field Definitions

- member_label - Used by the server to show the client has intent to view members in a room and further information is required

from the client to be processed and used to reference the current stage in the process.

- list_of_rooms – Used to reference if the room the client wants to view members of is a valid room.

- message - Used to store any message that comes from the client.

- room – Used to store the room being compared to in question within the member list.

6.7. Select Room Private Message Response

message == "\selectroom "

6.7.1. Usage

This command interprets the client's intent to send a message to a designated chat room. If the room does not exist then the following will be sent as a message from server to client: "Room name doesn't exists"

6.7.2. Field Definitions

- message – Used to store any message that comes from the client.

- client_conn – Used to store valid connection information of clients connecting to the server.

- client – Used to store and look for the client's name in the client list.

- select label – Used by the server to determine that the next subsequent message prepared by the client in the "\selectroom" command.

- select_room – Used to store room name to be compared in the select cast function.

- name – Used to store the room being compared to in question within the member list.

- list_of_rooms – Used to reference if the room the client wants to view members of is a valid room.
- roomname – Used to store a room name to compare to the room list.
- conn – Holds connection information on the current client.

6.8. Quit Response

message == "\quit”

6.8.1. Usage

This command interprets the client’s intent to leave the server by looking for a “\quit” command.

6.8.2. Field Definition

- quit_label - Used by the server to determine the next step in the quit command response on the server.
- client_name – Holds client’s current name as reference.
- Connection – Holds connection information on the current client.
- message – Used to store any message that comes from the client.
- client_conn – Used to store valid connection information of clients connecting to the server.
- client – Used to store and look for the client’s name in the client list.
- select label – Used by the server to determine that the next subsequent message prepared by the client in the “\selectroom” command.
- list_of_rooms – Used to reference if the room the client wants to remove self from is a valid room.

- roomname – Used to store a room name to compare to the room list.
- conn – Holds connection information on the current client.

6.9. Help Definitions Response

message == “”help” or message == “\?”

6.9.1. Usage

A help command that prints all the usable commands the client can use within the chat client on the server.

6.9.2. Field Definition

message – Used to store any message that comes from the client.

6.10. Private Message Response

message == “\private”

6.10.1. Usage

A command that allows a user to send a private message to another user where other users won’t be able to read the message sent in between. If the target user does not exist then the following message is sent to the client user: “There’s no such user”

6.10.2. Field Definition

- message – Used to store any message that comes from the client.
- private_label – Used by the server to determine the state of the current process for the “\private” command.
- client_name – Holds client’s current name as reference.
- temp – Hold’s the current position of the client list within the server when looking for the verified user.
- such_user – Holds the target user’s user_name from the client user to compare to the user list on the server.

- to_send – Holds the message the user wants to send to target user.

6.11. Print All Response

message == "\printall"

6.11.1. Usage

This command prints all rooms with all the users within the designated rooms with the "printall" command.

6.11.2. Field Definitions

- message - Used to store any message that comes from the client.
- room – Used to store the room name variable to cycle through the list of rooms.
- list_of_rooms – Stores all the rooms that are currently active on the server.
- name - Used to store the user name variable to be cycled through the list of users in the room.

6.12. Server Side Shutdown Response

6.12.1. Usage

This command responds to the "\shutdown" command that comes from the client side. The "\shutdown" command prompts the server to ask the client for a password in which only trusted users SHOULD know.

6.12.2. Field Definitions

- message – Used to send the command over to the server to respond to commands.
- password – Used to compare to password stored on server side.
- shutdown_label – Used by the server to determine the state of the current process for the \shutdown" command.

7. Error Handling

Both server and client SHOULD detect when the socket connection linking them is terminated. This is done by actively sending message. If the server detects the client that the client has been lost, the server SHOULD remove the client from all rooms to which they have joined. If the client detects that the connection to the server has been lost, it MUST consider itself disconnected and MAY choose to reconnect.

The client side MUST check for error inputs from the client and handle them as error messages to let the client know of the error of the command.

8. “EXTRA” Features Supported

Note that private messaging is supported in addition to the project criteria.

Note that help commands is supported in addition to the project criteria.

Note that print all is supported in addition to the project criteria.

9. Conclusions & Future Work

This specification provides a generic message passing framework for multiple clients to communicate with each other via a central forwarding server.

Without any modifications to this specification, it is possible for clients to devise their own protocols that rely on text-passing system provided described here.

For example, transfer of arbitrary binary data can be achieved through transcoding to base64. Such infrastructure could be used to transfer arbitrarily large files, or to establish secure connections using cryptographic transport protocols such as Transport Layer Security(TLS).

Another example would be to scramble messages between the client and server with a key to implement secure messaging.

10. Security Considerations

Messages sent using this system have no protection against inspection, tampering or outright forgery. The server sees all messages that are sent through the use of this service. ‘Private’ messaging maybe easily intercepted by a 3rd party that is able to capture network traffic. Users wishing to use this system for secure communication should implement their own user-to-user encryption protocol.

11. Normative References

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

[2] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

12. Acknowledgments

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