INTERNET RELAY CHAT

By

Name: MOUNIKA JAVVAJI (956734921)

TEAM

HARSHITHA DEPURU (934332550)

VELGONDA LAASYA (957262603)

FALL 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PORTLAND STATE UNIVERSITY

1825, SW Broadway

Portland,

OREGON, 97201

ABSTRACT

The project depends on the concept of Internetworking Relay Chat. Through this project we make an attempt to achieve communication between server and client. It involves a user using the functionalities listed below:

1. Creating a room
2. Joining a room
3. Exit a room
4. List of members in a room
5. List of rooms created
6. Chat in a room
7. Get Help
8. Error Display

It enables multiple clients to communicate with each other over a single server. For instance, let us consider the online game Game 'Ludo'. The players first enter a room with a code or create a room which generates a code in order to start a game. It displays all the members joined into that room. During the game, it enables the users to chat via a chat box which can be read by all the users in that particular room. If a user's internet is down, it throws a message 'Unable to connect to the server'.

Similarly, we try to achieve the same concept in this project.

PROJECT GRADING CRITERIA

	IRC Grading Sheet	Maximum Points	Actual Points Scored
1	RFC Document	20	
2	Server Process	3	
3	Client can connect to a server	3	
4	Client can create a room	3	
5	Client can list all rooms	3	
6	Client can join a room	3	
7	Client can leave a room	2	
8	Client can list members of a room	3	
9	Multiple clients can connect to a server	5	
10	Client can send messages to a room	5	
11	Client can join multiple (selected) rooms	10	
12	Client can send distinct messages to multiple (selected) rooms	10	
13	Client can disconnect from a server	5	
14	Server can disconnect from clients	5	
15	Server can gracefully handle client crashes	5	
16	Client can gracefully handle server crashes	5	
17	Programming Style	10	
	Extra Credit Features		
	Private or Ephemeral Messaging	5	
	Secure messaging	5	
	File transfer	5	
	Cloud connected server	5	
	TOTAL	120	

TABLE OF CONTENTS

1. Introduction

The primary aim of Internet Relay Chat project is to build a game which involves chat functionality protocol which enables the users to communicate with each other. It comprises of a central server which relays messages that are sent to it to the other users connected. Users would be able to create rooms, join rooms, chat with one another, exit the room and Get Help.

2. Basic Information

The server is open for connections on port 31425, and so this protocol only enables TCP/IP for all communication. This persistent connection is maintained by clients connecting to this port. relationship with the server. Over this, the client can interact with the server by sending messages and requests. open channel, and the server can respond in a same manner. It's inherent in this transmission protocol Asynchronous means that both the client and the server are free to receive and transmit messages at any time. back to the client asynchronously with messages. The client as well as the server may both end Any time, for any cause, a connection can be made. The two could decide to communicate an error to one another. party informing them of the connection's termination's cause. Depending on the configuration and resources of the host system, the server may decide to limit the number of users and rooms that can be created. The number of client connections is set to six for the present project.

3. Message Infrastructure

The client delivers the orders to be executed in the format command argument1 argument2, where command is a four-letter word that can be written in any case (uppercase, lowercase, or a combination of both). When a user's addition is inaccurate, errors are taken into consideration in the front of the response message. But no errorcode is needed for this straightforward system. The server only sends the client a string representation of the error message or response.

4. Label Semantics

- Sending and receiving labels is necessary for identifying users and rooms. All label rules must be validated in conformity with the following criteria:
    - Must only be composed of readable UTF character values.
    - The input size must be at least 1 character and no greater than 100k.

5. Client Messages

On the client side, the user will send particular commands with the arguments to the server.

5.1 Welcome client

Usage: person will enter his username

Response: welcome user

5.2 List all rooms

Usage: person will enter command “lor”.

Response: list of all rooms

5.3 List all room members

Usage: person will enter command “lime roomem”.

Response: list of all members in the room.

5.4 Create a room

Usage: person will enter command “cor”.

Response: room with given name will be created.

5.5 Join a room

Usage: person will enter command “jor”.

Response: client will join the room with a given name.

5.6 Leave a room

Usage: person will enter command “eor”.

Response: client will leave the given name room.

5.7 Send message in a room

Usage: person will enter command “chatmsg”.

Response: client will receive the confirmation message that message to given roomname is delivered.

5.8 Get help

Usage: person will enter command “help”.

Response: client will receive the text describing all the available commands.

5.9 Exit the running client

Usage: person will enter command “exit”

Response: person will stop the execution of the current client process and he will be prompted with the new terminal command instance.

5.10 Error responses

- Some examples of the error response

```
INVALID_ROOM_NAME = "invalid room name"
ROOM_DOES_NOT_EXIST = "room does not exist"
NOT_MEMBER = "you are not member"
INVALID_MESSAGE_FORMAT = "invalid message format"
```

6. Server Messages

According to the client command or error message, a server will respond. An "invalid username" message will be shown for instance, if the username is less than two characters. The server application will stop running if exit is typed in the command line terminal of the server.
As an instance, this section contains all server messages.

```
SERVER_STARTED = "server started: ctl + c to exit or exit to exit"
SERVER_STOPPED = "server stopped"
NEW_CLIENT = "new client"
WELCOME_CLIENT = "welcome user \n"
CLIENT_INPUT = "client input received"
CLIENT_INVALID = "client invalid"
CLIENT_DISCONNECT = "client disconnected"
CLIENT_DISCONNECT_ERR = "client disconnect error"
CLIENT_ALREADY_IN = "you are already in the system"
CLIENT_EXISTS = "name exists"
INPUT_INVALID = "input invalid"
UNKNOWN_COMMAND = "command invalid Type - HELP"
ROOMS_AVAILABLE_TITLE = "Available rooms \n"
NO_ROOMS_TITLE = "Sorry, no rooms "
INVALID_ROOM_NAME = "invalid room name"
ROOM_DOES_NOT_EXIST = "room does not exist"
ROOM_MEMBERS = "room members \n"
```

ROOM_ADDED = "new room added"
ALREADY_MEMBER = "you are already member"
NEW_MEMBER_JOINED = "new member joined in room"
MEMBER_LEFT = "one member left the room"
YOU_LEFT_ROOM = "you left the room"
MEMBERSHIP_GRANTED = "Membership granted to the room"
NOT_MEMBER = "you are not member"
INVALID_MESSAGE_FORMAT = "invalid message format"
EXIT_SUCCESSFUL = "'\n exit successful"

7. Manager Module

This module controls the handling of various command executions.

SCRIPTS

- def authenticate(client)
- def room_creation(name, client)
- def user_creration(name, client)
- def disconnection (stream_server, client)
- def room_join (name, client)
- def exit_room (name, client)
- def list_of_members (arg, client)
- def list_of_rooms(arg, client)
- def chat (arg, client)
- def transmit (room, note)
- def check(name)
- def welcome(client)

8. Conclusion & Future Work

The framework for generic message passing provided by this specification enables numerous clients to connect with one another via a unified forwarding server. Future developments can include the completion of private messaging, chat room, file transmission, and cloud server deployment.

9. Security Considerations

No defence exists against scrutiny, tampering, or blatant forgery for messages sent over this mechanism. All messages sent through the use of this service are not accessible to the server.

10. IANA Considerations

None

10.1 Normative References

- [1] 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.