

Internet Relay Chat Project

- Bhavani Chanda
- Sai Sindhu Muppaneni

Table of Contents

1. Introduction:	3
1.1 Server and Client:.....	3
1.2 Features	3
2. Specifications	3
2.1 Communication:.....	3
2.2 Listing rooms.....	3
2.3 Creating Room	3
2.4 Joining a room.....	3
2.5 Leave room	3
2.6 Messages	4
2.7 Replies.....	4
3. Infrastructure	4
3.1 Client can connect to a server.....	4
3.2 Client can create a room	4
3.3 Client can list all rooms	4
3.4 Client can join a room	5
3.5 Client can leave a room.....	6
3.6 Client can list members of a room	6
3.7 Switch to another room	7
3.8 Multiple clients can connect to a server	7
3.9 Client can send messages to a room	8
3.10 Client can join multiple (selected) rooms	8
3.11 Client can send distinct messages to multiple (selected) rooms	9
3.12 Client can disconnect from a server.....	9
3.13 Server can disconnect from clients	9
3.14 Server can gracefully handle client crashes	10
3.15 Client can gracefully handle server crashes	10
4. Conclusion and Future work.....	10

1. Introduction:

The IRC (Internet Relay Chat) project aims to provide multi-client communication through protocols that allow users to create, join, and leave rooms. It also enables private messaging between clients. In a room, every client can see all the messages exchanged when multiple clients are present. The server and clients act as the application's central components, with the server handling message distribution and client management.

1.1 Server and Client:

The server and client components form the core of the application. A single server is employed, which can accommodate any number of clients. Each client has a unique name consisting of alphanumeric and underscore characters.

1.2 Features:

The primary function of the application is to facilitate client-to-client communication through the server. Users can create and join rooms, as well as perform additional actions such as listing available rooms, exiting a room, and sending private messages.

2. Specifications:

2.1 Communication:

Rooms and one-on-one chats serve as the primary means of communication in the application. When a user sends a message in a room, the server receives it and distributes it to all other clients in the room, except the sender.

2.2 Listing rooms:

The application includes a command, "\$list_room" that provides a list of all available rooms. Initially, when no rooms have been created, the response will be "no rooms available."

2.3 Creating Room:

Initially, no rooms were available. The first person to start the software can create a room using the "\$join_room" command. Once the room is created, other users can join and start conversing in the newly created room using its name.

2.4 Joining a room:

Any client can join an available room by using the "\$join_room" command followed by the desired room name. Once the client has joined, all other members in the room become aware of their presence, enabling them to engage in conversations.

2.5 Leave room:

Each member of a room can leave at any time using the "\$leave_room" command. Upon leaving, the client will no longer receive messages from that specific room.

These features allow users to participate in multi-client communication, create and join rooms, and engage in private conversations, enhancing the collaborative and interactive nature of the application.

2.6 Messages:

Messages in the IRC project consist of three crucial components: name, transaction ID, and payload. These components are separated by a delimiter, which is a space in this case.

- The name field represents the message's identifier or label, providing information about the purpose or type of the message.
- The transaction ID is an unsigned integer that increases by 1 each time a client sends a message. However, for server-to-client messages, the transaction ID remains at 0.
- The payload field contains the actual content of the message, which can vary depending on the specific purpose of the message. It could include text, commands, data, or any other relevant information.

2.7 Replies:

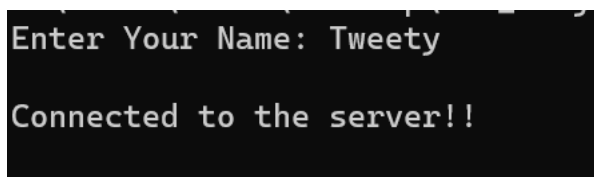
When a message is sent from a client to the server, the server responds with a numerical response. This response includes the transaction ID and indicates the status of the request. A value of 0 typically signifies a successful request, while a non-zero value indicates that some problem or error occurred during the processing of the request.

These message components and the server's numerical response help facilitate communication and provide a structured way to exchange information between clients and the server in the IRC project.

3. Infrastructure :

3.1 Client can connect to a server:

Multiple clients can establish connections to a single server, with each client having its unique name. The names of the clients can contain both underscore and alphanumeric characters.

A terminal window with a black background and white text. The first line shows a prompt 'Enter Your Name: ' followed by the input 'Tweety'. The second line shows the output 'Connected to the server!!'.

3.2 Client can create a room:

At the start, there were no available rooms. The first user can create a room using a specific command. Subsequently, other users can join this newly created room and start communicating using the room's designated name.

```

Enter Your Name: tweety

Connected to the server!!

Application Menu:
1.menu
2.create_room
3.join_room
4.switch_room
5.leave_room
6.list_room
7.personal message
8.exit_room

create_room IP1
IP1 created

Hello!
[IP1] tweety: Hello!
[IP1] ash joined the room
[IP1] ash Welcome to the room
[IP1] ash: Hello! tweety

```

3.3 Client can list all rooms:

By using the "list_room" command, a user can access a list of rooms. To join a specific room, the user can enter the room's name with the "join_room" command. If there are available rooms, the list of rooms will be displayed. In case there are no rooms available, the message "Sorry, no rooms available to join" will be shown.

If no rooms available:

```

Enter Your Name: tweety

Connected to the server!!

Application Menu:
1.menu
2.create_room
3.join_room
4.switch_room
5.leave_room
6.list_room
7.personal message
8.exit_room

list_room
There are no rooms left; you cannot join

```

List of rooms available and members in a room:

```

list_room
The List of available rooms and members are:

IP1

['tweety', 'ash']

```

3.4 Client can join a room:

Once rooms become available, any client can see the list of available rooms and use the "join_room" command to enter a room of their preference. Once the client successfully joins the room, all other clients in the room will be notified about the new member, enabling them to start conversing together.

```
Enter Your Name: ash
Connected to the server!!

Application Menu:
1.menu
2.create_room
3.join_room
4.switch_room
5.leave_room
6.list_room
7.personal message
8.exit_room

join_room IP1
[IP1] ash joined the room
[IP1] ash Welcome to the room
Hello! tweety
[IP1] ash: Hello! tweety
create IP3
IP3 created

join IP3
[IP3] ash joined the room
[IP3] ash joined the room[IP3] ash Welcome to the room[IP3] ash Welcome to the room
switch_room IP1
You are successfully switched to IP1
```

3.5 Client can leave a room:

At any time, a user can leave a room by using the "leave_room" command. As long as there are other users present, the room remains functional and accessible. However, once the last user leaves the room, the room is deleted as there are no longer any users within it.

```
join IP2
[IP2] tweety joined the room
[IP2] tweety joined the room[IP2]
come to the room
[IP1] ash: Leave_room IP1
[IP1] ash: Leave_room
[IP1] ash left the room!
```

3.6 Client can list members of a room:

If there is at least one user in a room, the members of the room can be listed. When a new user joins the room, all existing users in the room are notified. If there are no users in the room, it is indicated that no one is available.

```
switch_room IP1
You are successfully switched to IP1

list_room
The List of available rooms and members are:

IP1
['tweety']

IP2
['tweety', 'tweety']

IP3
['ash', 'ash']
```

3.7 Switch to another room:

A user can switch from one room to another room using the switch_room command.

```
switch_room IP1
You are successfully switched to IP1

list_room
The List of available rooms and members are:

IP1
['tweety']

IP2
['tweety', 'tweety']

IP3
['ash', 'ash']

switch_room IP1
You are already a member of the room; select an alternative room to switch.

switch_room IP2
You are successfully switched to IP2
```

3.8 Multiple clients can connect to a server:

A server can connect to multiple clients ('n' number of clients). Once the rooms are ready for joining, any client can see the list of available rooms. Whenever a new user joins a room, all clients in the room receive notifications about the new user's arrival.

```

Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sindh>cd C:\Users\sindh\Desktop\IRC_Project
C:\Users\sindh\Desktop\IRC_Project>python main.py
C:\Users\sindh\Desktop\IRC_Project>python server.py
Connection established
Server Started
Got connection with ('127.0.0.1', 59581)

<socket.socket fd=312, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 64526), raddr=('127.0.0.1', 59581)>
Name of the client is tweety

Got connection with ('127.0.0.1', 59582)

<socket.socket fd=412, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 64526), raddr=('127.0.0.1', 59582)>
Name of the client is ash

1
1
IP1
['tweety', 'ash']
2
3
3
IP1
['tweety']
IP2
['tweety', 'tweety']
IP3
['ash', 'ash']

Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sindh>cd C:\Users\sindh\Desktop\IRC_Project
C:\Users\sindh\Desktop\IRC_Project>python client.py
Enter Your Name: tweety

Connected to the server!!

Application Menu:
1.menu
2.create_room
3.join_room
4.switch_room
5.leave_room
6.list_room
7.personal message
8.exit_room

create_room IP1

Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sindh>cd C:\Users\sindh\Desktop\IRC_Project
C:\Users\sindh\Desktop\IRC_Project>python client.py
Enter Your Name: ash

Connected to the server!!

Application Menu:
1.menu
2.create_room
3.join_room
4.switch_room
5.leave_room
6.list_room
7.personal message
8.exit_room

join_room IP1

```

3.9 Client can send messages to a room:

Users create and enter a room where they can engage in interactions by exchanging messages. These messages are visible to all other users present in the room, allowing multiple members of the group to receive and view the messages sent by an individual user.

```

create_room IP1
IP1 created

hello
[IP1] tweety: hello
[IP1] ash joined the room
[IP1] ash Welcome to the room
[personal message] ash: hello

```

```

create_room IP2
IP2 created

join_room IP1
[IP1] ash joined the room
[IP1] ash Welcome to the room
personal tweety hello
You can start sending messages
[personal message] ash: hello

```


3.10 Client can join multiple (selected) rooms:

Clients have the flexibility to join multiple created rooms, provided that each room has at least one user present. These rooms can accommodate multiple clients simultaneously, allowing for a potentially large number of clients to be present in a room at any given time.

```
list_room
The List of available rooms and members are:

IP1
['tweety', 'ash']

IP2
['ash', 'tweety']
```

3.11 Client can send distinct messages to multiple (selected) rooms:

A single client can send messages to different rooms. The clients present in each respective room will receive the messages sent by the client to those specific rooms. The content of the messages sent by the client may vary depending on the selected rooms, as different rooms may have different message exchanges.

```
IP2
['ash', 'tweety']

switch_room IP1
You are successfully switched to IP1

hello
[IP1] tweety: hello
```

3.12 Client disconnects from server:

By utilizing the "exit_room" command, a client can terminate their connection to the server. This action essentially shuts down the client terminal. If the client is a member of any rooms, they will receive a notification. Additionally, the server is also aware of this forced disconnection.

```
User Name is tweety
Got connection with ('127.0.0.1', 52739)

<socket.socket fd=972, family=2, type=1, proto=0, laddr=('127.0.0.1', 64526), raddr=('127.0.0.1', 52739)>
Name of the client is tweety

exception occurred [WinError 10054] An existing connection was forcibly closed by the remote host
User Name is ash
```

3.13 Server can disconnect from clients:

After five minutes of inactivity, the server automatically disconnects its connection with the clients. This disconnection from the server results in a timed-out error.

```
fd, addr = self.accept()
TimeoutError: timed out
```

3.14 Server can gracefully handle client crashes:

When a client experiences a crash, the server, rooms, and other clients are notified. To handle this situation, error handling mechanisms are implemented, and the host will close the existing connection affected by the client crash.

```
Got connection with ('127.0.0.1', 59966)
<socket.socket fd=396, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 64526), raddr=('127.0.0.1', 59966)>
Name of the client is tweety

Ip1
[]
[]
exception occured [WinError 10038] An operation was attempted on something that is not a socket
User Name is tweety
Ip1
Exception in thread Thread-3 (handle):
Traceback (most recent call last):
  File "C:\Users\sindh\Desktop\IRC_Project\server.py", line 138, in handle
    userMessage = client.recv(1024).decode('utf-8')
OSError: [WinError 10038] An operation was attempted on something that is not a socket

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "C:\Users\sindh\AppData\Local\Programs\Python\Python310\lib\threading.py", line 1009, in _bootstrap_inner
    self.run()
  File "C:\Users\sindh\AppData\Local\Programs\Python\Python310\lib\threading.py", line 946, in run
    self._target(*self._args, **self._kwargs)
  File "C:\Users\sindh\Desktop\IRC_Project\server.py", line 174, in handle
    removeClientFunc(nameOfUser)
  File "C:\Users\sindh\Desktop\IRC_Project\server.py", line 126, in removeClientFunc
    room.peoples.remove(client)
ValueError: list.remove(x): x not in list
```

3.15 Client can gracefully handle server crashes:

In the event of a server crash, all clients will be logged out automatically and receive a message indicating that the server is unavailable. The specific message displayed to the clients will state "Server not responding."

```
IP1
['ash']

IP2
['ash']

hello
You are not member of any room

SERVER NOT RESPONDING
```

4. Conclusion and Future work:

In conclusion, future improvements can be made to the IRC system. These include addressing redundancy in room and client names, implementing cryptographic transport protocols for secure media sharing, enabling communication between different chat rooms, enhancing security features, and expanding message sharing capabilities. This can involve the addition of broadcast messages and private messages, allowing for communication between multiple clients connected to a single server. By incorporating these enhancements, the IRC system can provide a more robust and feature-rich instant messaging experience.