COMPUTER NETWORKS LAB (PROJECT REPORT)

Instructor - Shashi Prabh

PROJECT TOPIC (TEAM - 14)

MULTICASTING MULTIMEDIA OVER IP (INTERNET RADIO)

Project Background:

The project is an application of Internet Radio which involves multicasting of data in the form of multimedia over IP. Since, in a radio any number of users can be tuned in to a particular station out of all the available stations at any time; to support the same scenario the model used in our project is Any Source Multicast model (ASM). In this model, multicast group address will help to identify the multicast messages and multiple senders can be part of the same group. Just like a user can change and exit the stations in an actual radio, our client and server provides the similar functionality. The connection of sockets is initially done using TCP which establishes control channel and later the multimedia is sent using UDP socket since it is more efficient for multicast type of transmission. Hence, in all it implements one-to-many multicast several times.

Design Decisions:

1. **Use of threads** - For a server to process multiple requests of clients in parallel in order to support multicast, we decided to create separate threads for each client for the same.

2. **New structure** in server for storing information related to the station number, path and port number of the respective station.

3. We have created **separate functions** for each of the individual tasks like storing the details of the stations, sending structures to the client, calculating bit rate, receiving structures and so on in order to support modularity.

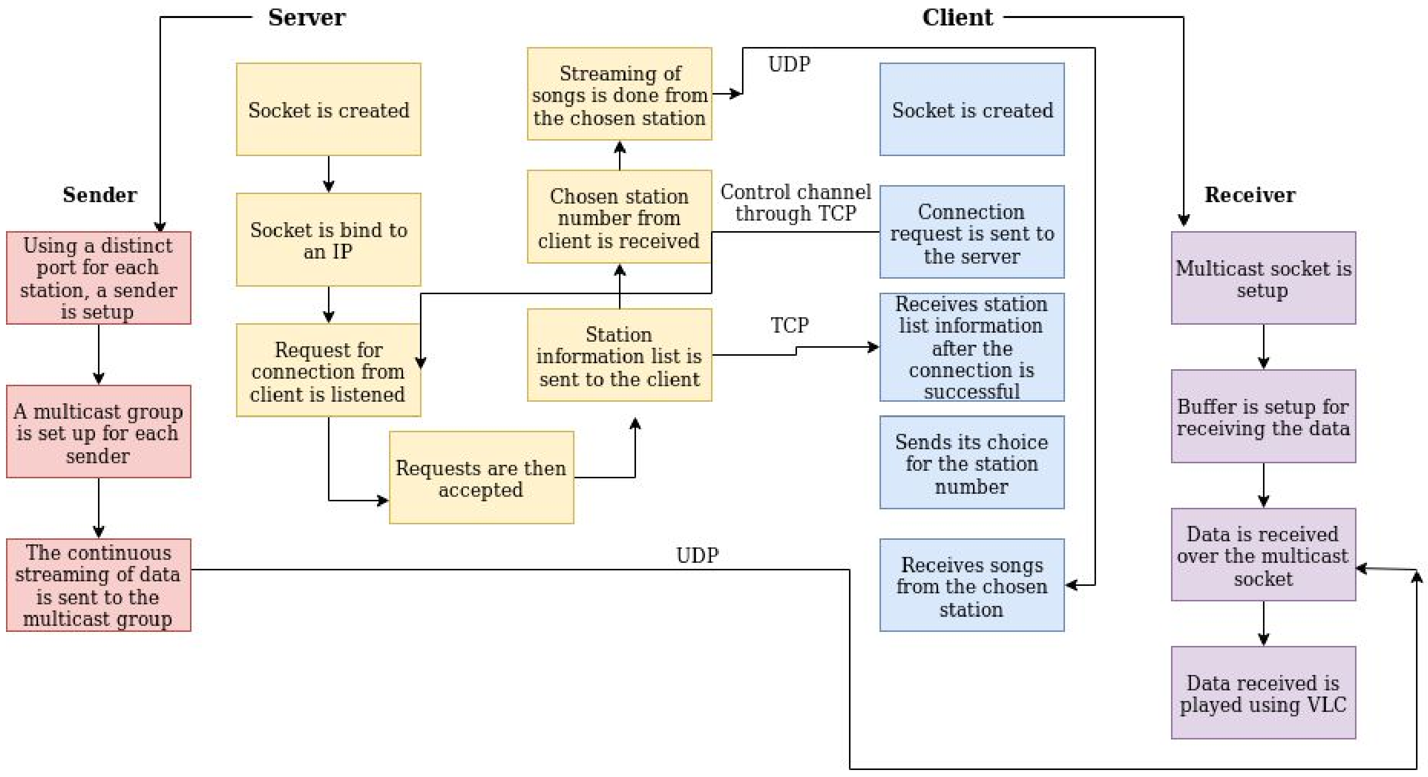
4. **GUI Interface** - We have implemented GUI interface which is user-friendly for the ease of the user to operate. It displays the essential 4 buttons - run, pause, change station and stop radio and station list containing the number of stations.

5. We have used **TCP socket for the control channel** to ensure reliable transmission of the information of the stations. However, implementing **UDP socket for the transmission of multicast messages (streaming of songs)** would suffice as the songs received at the client side can bear some packet loss. This increases performance and ensures efficient utilization of different channels which helps in reduction of the hardware cost.

Determining Data Rate and calculating Delay:

- Bit-rate for each individual song is determined uniquely.
- In-built system call “mediainfo” is used to obtain the same.
- Obtained data rate is then used to calculate the ideal sleep time which basically causes delay in the execution of thread in order to receive data with appropriate pace and reduce data loss.
- Formula employed to calculate ideal sleep time: ((size of buffer * 8) /bitrate) * 500000000
- The obtained value is then assigned to tv_nsec of timespec structure.
- Ultimately, a nanosleep function is called which utilises the calculated value of delay obtained above.

Working flow:


Server
Client
Sender
Using a distinct port for each station, a sender is setup
A multicast group is set up for each sender
The continuous streaming of data is sent to the multicast group
Socket is created
Socket is bind to an IP
Request for connection from client is listened
Requests are then accepted
Streaming of songs is done from the chosen station
Chosen station number from client is received
Station information list is sent to the client
UDP
Control channel through TCP
TCP
Socket is created
Connection request is sent to the server
Receives station list information after the connection is successful
Sends its choice for the station number
Receives songs from the chosen station
UDP
Receiver
Multicast socket is setup
Buffer is setup for receiving the data
Data is received over the multicast socket
Data received is played using VLC

Features of server:

1. **Function StationDetails()**
    - Fills the details of the station in station_info structure.
    - The details involve the station type, station number, station name, data port, station id and the station path.
    - It includes assignment of port numbers to all 3 stations and paths to all of them.
    - It uses memcpy() function for copying paths and info to structure variables.
    - Function bzero() is used to clear the memory or clear the address field.

2. **Function startStationListServer()**
    - It is used to send structures of station list to the client.
    - It includes major socket function calls and condition checks for bind,accept and listen functions for TCP connection.
    - On successful connection with client it starts to send the structures of information to all the 3 stations.

3. **Function BR_calc()** - It calculates the bit rate for the number of songs

4. **Function startStation()** -
    - The function checks for the song file in the directory by parsing all the directories mentioned.
    - It maintains a counter for the songs with .mp3 extension.
    - There are two 2-D arrays defined for storing the path of songs and their size.
    - Socket connection for multicast addresses is done here.
    - It multicasts information about the current song by sending the structure for song_info.
    - This function uses **nanosleep()** which suspends the execution of the calling thread for the specified time or till the time it receives any signal.

5. **Function main() -** It initialises and creates threads for each individual station.

Features of client:

1. **Function runRadio()** - A thread is created and joined to receive songs from the server.

2. **Function checkAndCloseVLC()**
    - When a user presses the ‘pause’ button, or ‘stop radio’ button, this function is called which in effect kills the current process running by making a “system()” command to make an OS call for pausing the streaming.

3. **Function receiveAndPrintStationList()**
    - It creates a TCP socket for the control channel which is used to display information about the available stations.
    - It receives the number of stations from the server and prints the details of each of the stations which includes station number, station name and station multicast port.

4. **Function setupAndRecvSongs()**
    - It creates a UDP socket for receiving multicast messages from server.
    - It joins the multicast group specified on which the server is binded and is constantly sending messages.
    - It assigns the port according to the user specified station so as to receive songs streaming on that particular station.
    - Hence, it streams the current song playing on that station and also displays the current song name and the next song name extracted from song_info structure sent by server.

5. **Function main()**
    - It displays a GUI window containing four options for the user to choose - 1. Run 2. Pause 3. Change station and 4.exit.
    - On first clicking the ‘Run’ button, it by default plays songs from the default station (1st station).
    - Furthermore, on clicking ‘pause’ the song pauses.
    - On clicking ‘Change Station’ button, another GUI window appears that contains the stations available in the radio.
    - On clicking one of the stations, run button plays the songs streaming on the respective station.
    - Furthermore, Station detailed information is displayed on the client terminal.

Attempt For Content Management:

- First in content management we have to take the live feed using the webcam(Cheese : For Linux).
- The first issue with taking the live feed is that the file creation at the designated folder for videos create a file name according to time stamp. But this can be handled using system calls.
- The video transmission of live camera feed was achieved but there were errors when it was to be started directly from client code.
- The main error lies for the right file name to be accessed in the code.
- Also, the file of video feed is stored at client side and server side for working of video transmission, which is a drawback.

Assumptions:

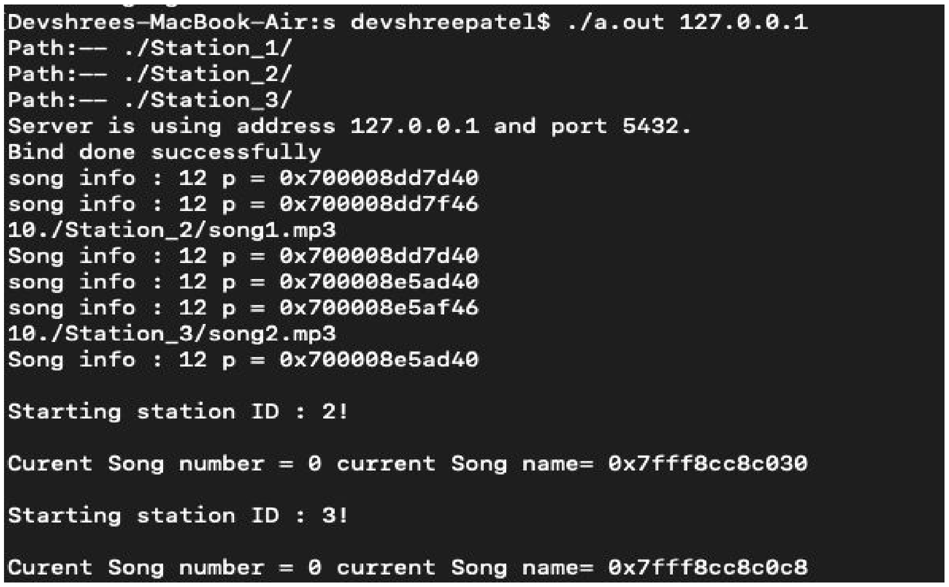
- Station 1 will be the default station running whenever a user starts receiving the songs.
- Number of stations is pre-defined to 3.
- Each station is limited to contain only 3 songs.

Limitations:

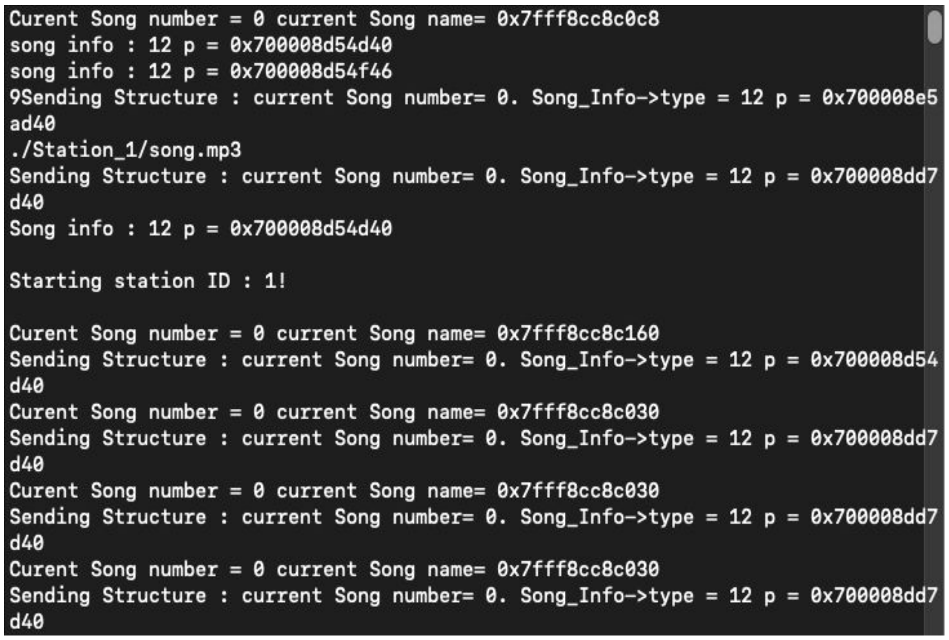
- The user cannot change the station as soon as one joins. User will have to select the run option for the first time only after that would one be able to change the station.
- The functionality of pause and change station will only be available after one station has been streamed completely.
- Code is restricted to streaming only audio formats.
- Input of songs to the stations is not done in real-time.

Screenshots representing terminal output and GUI:

Server side Screenshots

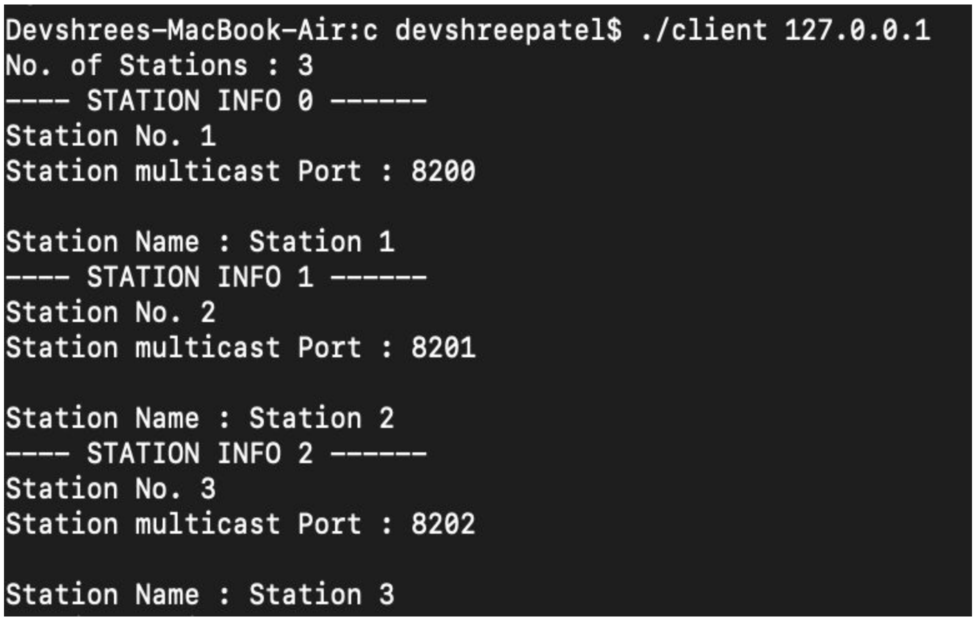


```
Devshrees-MacBook-Air:s devshreepatel$ ./a.out 127.0.0.1
Path:-- ./Station_1/
Path:-- ./Station_2/
Path:-- ./Station_3/
Server is using address 127.0.0.1 and port 5432.
Bind done successfully
song info : 12 p = 0x700008dd7d40
song info : 12 p = 0x700008dd7f46
10./Station_2/song1.mp3
Song info : 12 p = 0x700008dd7d40
song info : 12 p = 0x700008e5ad40
song info : 12 p = 0x700008e5af46
10./Station_3/song2.mp3
Song info : 12 p = 0x700008e5ad40

Starting station ID : 2!

Curent Song number = 0 current Song name= 0x7fff8cc8c030

Starting station ID : 3!

Curent Song number = 0 current Song name= 0x7fff8cc8c0c8
```



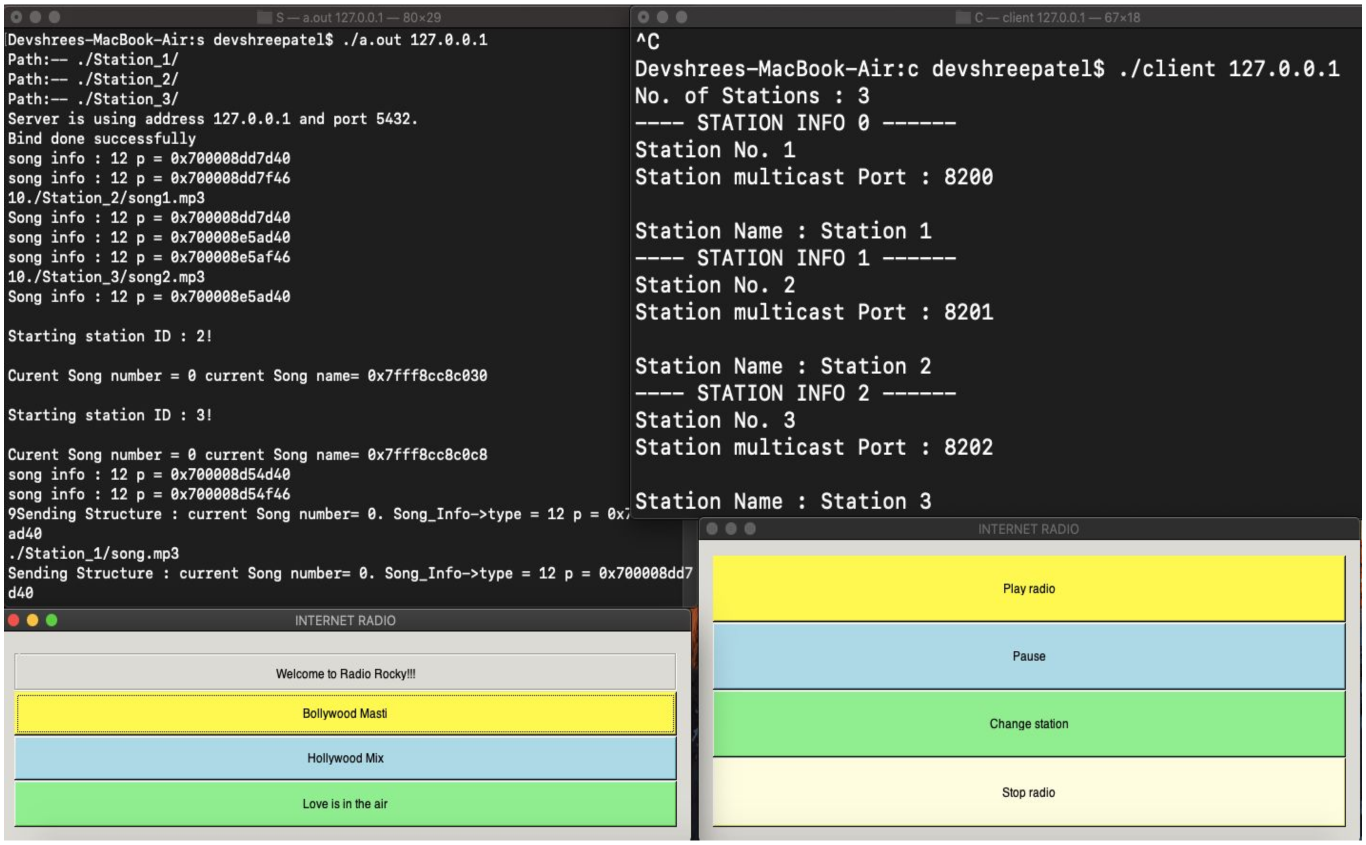
```
Curent Song number = 0 current Song name= 0x7fff8cc8c0c8
song info : 12 p = 0x700008d54d40
song info : 12 p = 0x700008d54f46
9Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x700008e5
ad40
./Station_1/song.mp3
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x700008dd7
d40
Song info : 12 p = 0x700008d54d40

Starting station ID : 1!

Curent Song number = 0 current Song name= 0x7fff8cc8c160
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x700008d54
d40
Curent Song number = 0 current Song name= 0x7fff8cc8c030
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x700008dd7
d40
Curent Song number = 0 current Song name= 0x7fff8cc8c030
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x700008dd7
d40
Curent Song number = 0 current Song name= 0x7fff8cc8c030
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x700008dd7
d40
```

Client side screenshots



```
Devshrees-MacBook-Air:c devshreepatel$ ./client 127.0.0.1
No. of Stations : 3
---- STATION INFO 0 ------
Station No. 1
Station multicast Port : 8200

Station Name : Station 1
---- STATION INFO 1 ------
Station No. 2
Station multicast Port : 8201

Station Name : Station 2
---- STATION INFO 2 ------
Station No. 3
Station multicast Port : 8202

Station Name : Station 3
```

Final output screenshot:

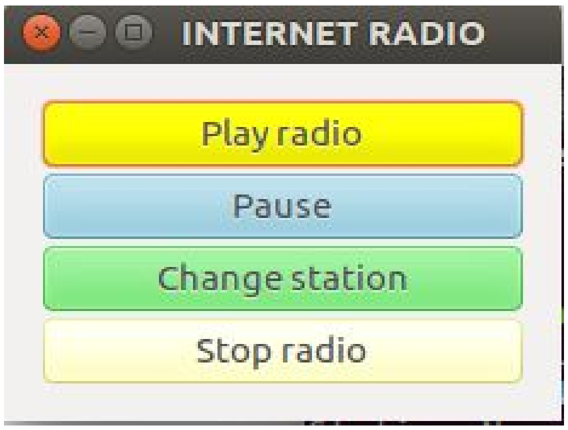


```
Devshrees-MacBook-Air:s devshreepatel$ ./a.out 127.0.0.1
Path:-- ./Station_1/
Path:-- ./Station_2/
Path:-- ./Station_3/
Server is using address 127.0.0.1 and port 5432.
Bind done successfully
song info : 12 p = 0x700008dd7d40
song info : 12 p = 0x700008dd7f46
10./Station_2/song1.mp3
Song info : 12 p = 0x700008dd7d40
song info : 12 p = 0x700008e5ad40
song info : 12 p = 0x700008e5af46
10./Station_3/song2.mp3
Song info : 12 p = 0x700008e5ad40

Starting station ID : 2!

Curent Song number = 0 current Song name= 0x7fff8cc8c030

Starting station ID : 3!

Curent Song number = 0 current Song name= 0x7fff8cc8c0c8
song info : 12 p = 0x700008d54d40
song info : 12 p = 0x700008d54f46
9Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x7
ad40
./Station_1/song.mp3
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x700008dd7
d40
```

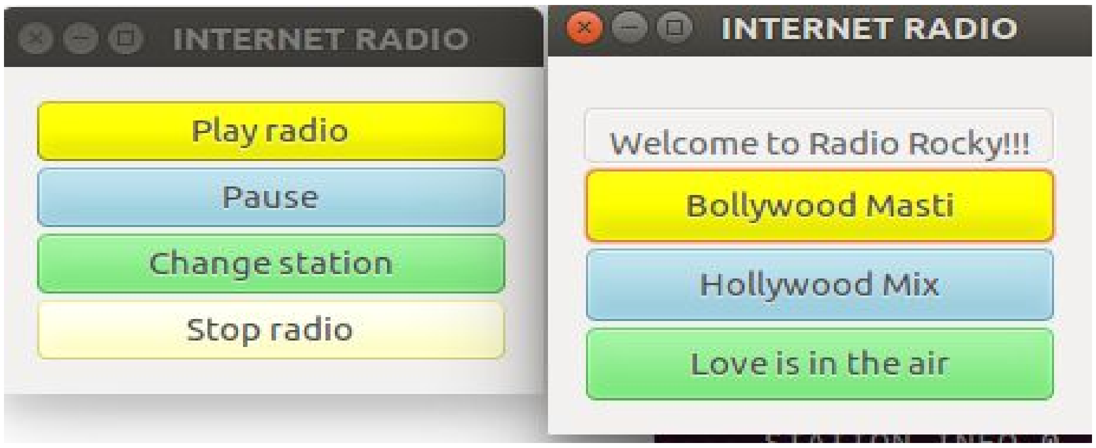
```
^C
Devshrees-MacBook-Air:c devshreepatel$ ./client 127.0.0.1
No. of Stations : 3
---- STATION INFO 0 ------
Station No. 1
Station multicast Port : 8200

Station Name : Station 1
---- STATION INFO 1 ------
Station No. 2
Station multicast Port : 8201

Station Name : Station 2
---- STATION INFO 2 ------
Station No. 3
Station multicast Port : 8202

Station Name : Station 3
```

GUI screenshots:

- **First window:**



- **Upon clicking on change station:**



Contributions of team members:

Devshree Patel (AU1741075)	**Ratnam Parikh** (AU1741036)	**Muskan Matwani** (AU1741027)	**Yesha Shastri** (AU1741035)
GUI- Scratch Code and Run Radio Part	GUI-Pause Part	GUI-Stations Part	GUI-colours and formatting
Server Comments	Attempt for Content Management	Client Comments	Flow chart for report
Server Part In Report	Content Management In Report	Client part in Report	Background and data rate calculations

Note: Main Client and Server code was done by all. Assumptions, limitations and design decisions were made by all.