COMPUTER NETWORKS LAB4 (PROJECT REPORT)

Instructor - Shashi Prabh

PROJECT (GROUP – 25)

MULTICASTING MULTIMEDIA OVER IP (INTERNET RADIO)

Project Background:

In this project it includes multicasting of data in the form of multimedia over internet protocol. It is an application for internet radio. As we know in radio a large number of users tune in to particular station out of all the available stations at time and similarly in the project there is Any Source Multicast model (ASM) where to identify multiple messages from multiple sender, multiple group address is used and so more than one sender can be a part of same group. The other point is like in radio the user can exit or change the stations in an actual radio at any time, our client and server provides the similar functionality to the user as the connection of sockets is initially done using transmission control protocol (TCP) which establishes the control channel and then later multimedia is sent using datagram protocol (UDP) socket as it is more efficient for the multicast type of transmission. Therefore, in all it implements one to many multicasts several times.

Design Decisions:

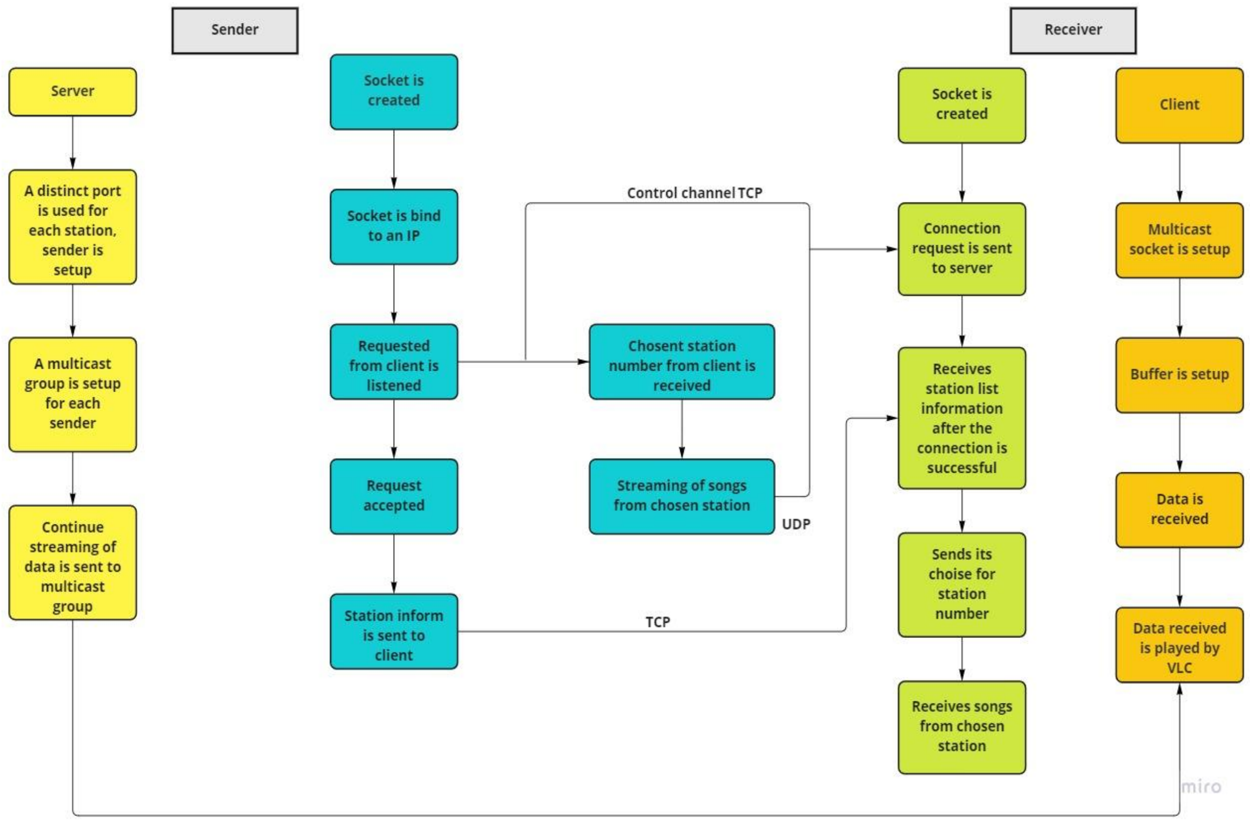
1. **Use of threads** - To process multiple requests in parallel with clients in order to support multicast we decided to create separate threads for each client for the same.
2. **New structure is used for storing information on the server**. The information is related to station number, path and port number of respective stations.

3. We also have **separate functions** for each of individual tasks like sending structure to clients, calculating bit rate , storing details of stations, receiving structures and other to support modularity.
4. **GUI Interface** – In GUI interface we have a user friendly one for the ease of the user to understand and operate. The important buttons are run , pause , change station and stop radio and also station list containing the number of stations.
5. **In the TCP socket** for the control channel is to have reliable transmission of information of stations. But for the transmission of multicast messages like streaming of songs **UDP** is chosen as songs received at the client side can bear some loss of packets and also it increases performance and ensures efficient utilization of different channels which helps in reduction of hardware cost as well.

Determining Data Rate and calculating Delay:

- For each individual song the bit rate is determined uniquely.
- In-built systems, also known as mediainfo, are used to obtain the same.
- Then to calculate the ideal sleep time, the obtained data rate is used. The ideal sleep time causes delay in the execution of thread in order to receive data with appropriate pace and reduce the data loss.
- The formula which is used to calculate the ideal sleep time is ((size of buffer * 8)/bit rate) * 500000000
- The obtained value is then assigned to tv_nsec of timespec structure.
- Lastly a nanosleep function is called which uses the calculated value of delay obtained in above steps.

Working flow:


Sender
Receiver
Server
A distinct port is used for each station, sender is setup
A multicast group is setup for each sender
Continue streaming of data is sent to multicast group
Socket is created
Socket is bind to an IP
Requested from client is listened
Request accepted
Station inform is sent to client
Chosent station number from client is received
Streaming of songs from chosen station
Control channel TCP
UDP
TCP
Socket is created
Connection request is sent to server
Receives station list information after the connection is successful
Sends its choise for station number
Receives songs from chosen station
Client
Multicast socket is setup
Buffer is setup
Data is received
Data received is played by VLC
miro

Features of server:

1. Function Station_Details()
    - Fills the station info structure with information about the station.
    - The details are the station type, station number, station name, data port, station id, and station path.
    - It includes the assignment of port numbers and pathways to all three stations.
    - The memcpy() function transfers paths and information to structure variables.
    - The bzero() function is used to clear memory or the address field.

2. Function start_Station_ListServer()
    - It delivers station list structures to the client and significant socket function calls and condition checks for TCP connection bind, accept and listen to functions.
    - After establishing a successful connection with the client, it begins sending information structures to all three stations.

3. Function BR_Calculation - It works out the bit rate based on the amount of songs. \

4. Function start_Station() -
    - The function looks for the song file in the directory by processing all the folders mentioned.
    - It keeps track of the songs that end in.mp3 format.
    - For storing the path of songs and their sizes, two 2-D arrays are defined.
    - This is where multicast socket connections are made.
    - It utilizes nanosleep() to pause the calling thread's execution for the specified period or until it gets any signal.

5. Function main() - It sets up and produces threads for each station individually.

Features of client:

1. Function run_Radio() - To receive songs from the server.

2. Function checkAnd_CloseVLC() - This function is invoked when a user presses the pause or stop radio buttons, effectively killing the current process by utilizing the "system()" command to initiate an OS call to suspend the streaming.

3. Function receive_AndPrint_StationList()
   - It generates a TCP socket for the control channel, which shows information about the available stations.
   - It receives the number of stations from the server and outputs the information, including the station number, name, and multicast port.

4. Function setupAndRecvSongs()
   - It establishes a UDP socket with the server to receive multicast messages.
   - It joins the multicast group to which the server is bound and sends messages all of the time.
   - It assigns the port to the user-specified station to receive songs streaming from that station.
   - As a result, it streams the current song on that station and shows the current song name and the next song name derived from the server's song info structure.

5. Function main()
   - It shows a GUI window with four options to pick from 1. Run 2. Pause 3. Switch stations and exit.
   - When you initially press the 'Run' button, it starts playing music from the default station (1st station).
   - Furthermore, pressing 'pause' causes the song to pause.
   - When you click the 'Change Station' button, a new GUI window appears with the available radio stations.
   - When you click one of the stations, the run button starts playing the songs currently streaming on that station.
   - Additionally, complete station information is shown on the client terminal.

Attempt For Content Management:

- First, we must use the webcam to capture a live stream (Cheese: For Linux).
- The first problem with recording the live feed is that the timestamp is used to name the file created in the selected video folder. However, system calls can be used to handle this.
- The live camera feed video transmission was successful, but there were issues when it was attempted directly from the client code.
- The critical mistake is not using the correct file name to access the code.
- The video feed file is stored on both the client and server sides for working video transmission, which is a disadvantage.

Assumptions:

- Station 1 will be the default station when a user first starts receiving songs.
- Each station is only allowed to have three pieces.
- The number of stations is fixed at three.

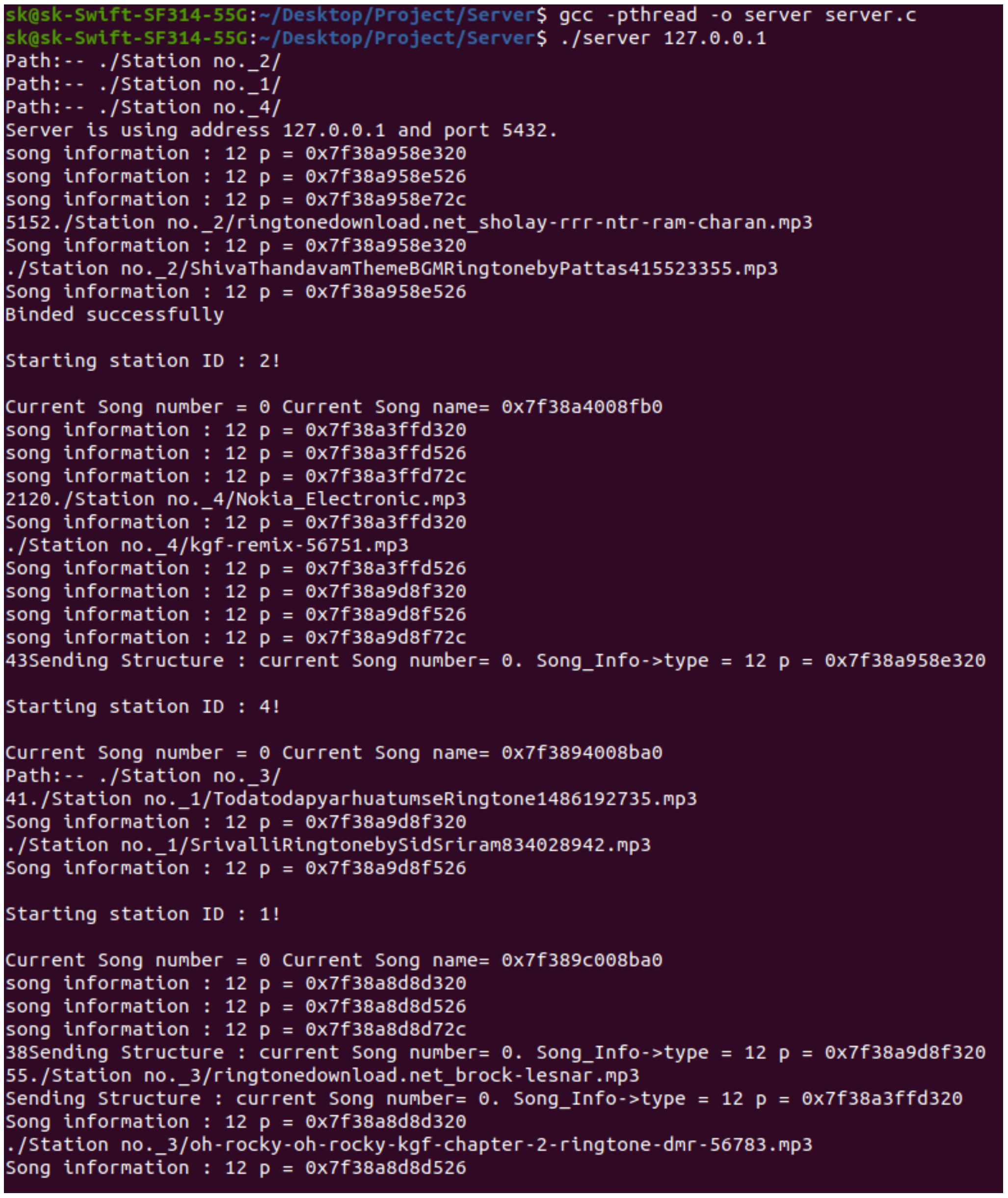
Limitations:

- As soon as a person joins, they are unable to change stations. For the first time, the user must choose the run option. Then and only then would one be able to change stations.
- The code is limited to only streaming audio formats, and song input to the stations is not done in real-time.
- Only once one station has been wholly streamed will the pause and change station capabilities be enabled.

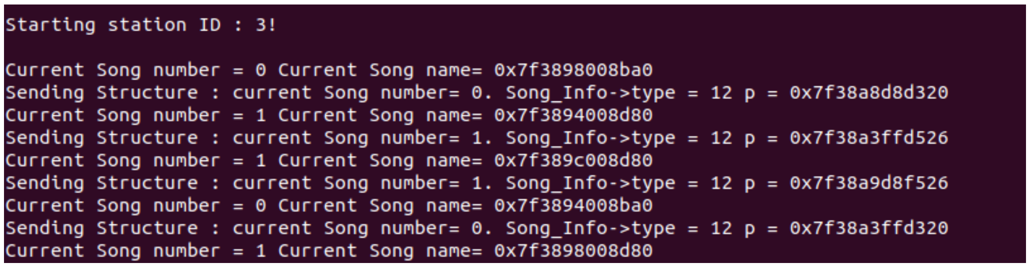
Screenshots representing terminal output:

1) Server

1.1 Running Server code :

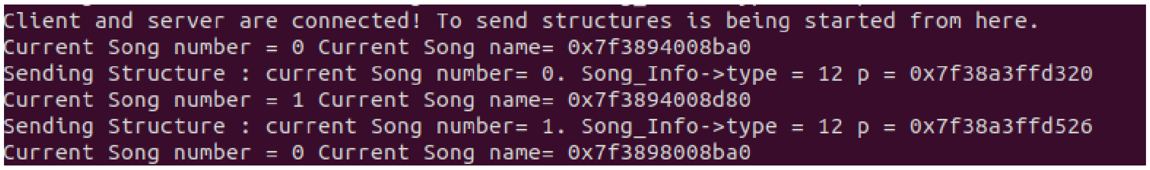


```
sk@sk-Swift-SF314-55G:~/Desktop/Project/Server$ gcc -pthread -o server server.c
sk@sk-Swift-SF314-55G:~/Desktop/Project/Server$ ./server 127.0.0.1
Path:-- ./Station no._2/
Path:-- ./Station no._1/
Path:-- ./Station no._4/
Server is using address 127.0.0.1 and port 5432.
song information : 12 p = 0x7f38a958e320
song information : 12 p = 0x7f38a958e526
song information : 12 p = 0x7f38a958e72c
5152./Station no._2/ringtonedownload.net_sholay-rrr-ntr-ram-charan.mp3
Song information : 12 p = 0x7f38a958e320
./Station no._2/ShivaThandavamThemeBGMRingtonebyPattas415523355.mp3
Song information : 12 p = 0x7f38a958e526
Binded successfully

Starting station ID : 2!

Current Song number = 0 Current Song name= 0x7f38a4008fb0
song information : 12 p = 0x7f38a3ffd320
song information : 12 p = 0x7f38a3ffd526
song information : 12 p = 0x7f38a3ffd72c
2120./Station no._4/Nokia_Electronic.mp3
Song information : 12 p = 0x7f38a3ffd320
./Station no._4/kgf-remix-56751.mp3
Song information : 12 p = 0x7f38a3ffd526
song information : 12 p = 0x7f38a9d8f320
song information : 12 p = 0x7f38a9d8f526
song information : 12 p = 0x7f38a9d8f72c
43Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x7f38a958e320

Starting station ID : 4!

Current Song number = 0 Current Song name= 0x7f3894008ba0
Path:-- ./Station no._3/
41./Station no._1/TodatodapyarhuatumseRingtone1486192735.mp3
Song information : 12 p = 0x7f38a9d8f320
./Station no._1/SrivalliRingtonebySidSriram834028942.mp3
Song information : 12 p = 0x7f38a9d8f526

Starting station ID : 1!

Current Song number = 0 Current Song name= 0x7f389c008ba0
song information : 12 p = 0x7f38a8d8d320
song information : 12 p = 0x7f38a8d8d526
song information : 12 p = 0x7f38a8d8d72c
38Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x7f38a9d8f320
55./Station no._3/ringtonedownload.net_brock-lesnar.mp3
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x7f38a3ffd320
Song information : 12 p = 0x7f38a8d8d320
./Station no._3/oh-rocky-oh-rocky-kgf-chapter-2-ringtone-dmr-56783.mp3
Song information : 12 p = 0x7f38a8d8d526
```



```
Starting station ID : 3!

Current Song number = 0 Current Song name= 0x7f3898008ba0
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x7f38a8d8d320
Current Song number = 1 Current Song name= 0x7f3894008d80
Sending Structure : current Song number= 1. Song_Info->type = 12 p = 0x7f38a3ffd526
Current Song number = 1 Current Song name= 0x7f389c008d80
Sending Structure : current Song number= 1. Song_Info->type = 12 p = 0x7f38a9d8f526
Current Song number = 0 Current Song name= 0x7f3894008ba0
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x7f38a3ffd320
Current Song number = 1 Current Song name= 0x7f3898008d80
```

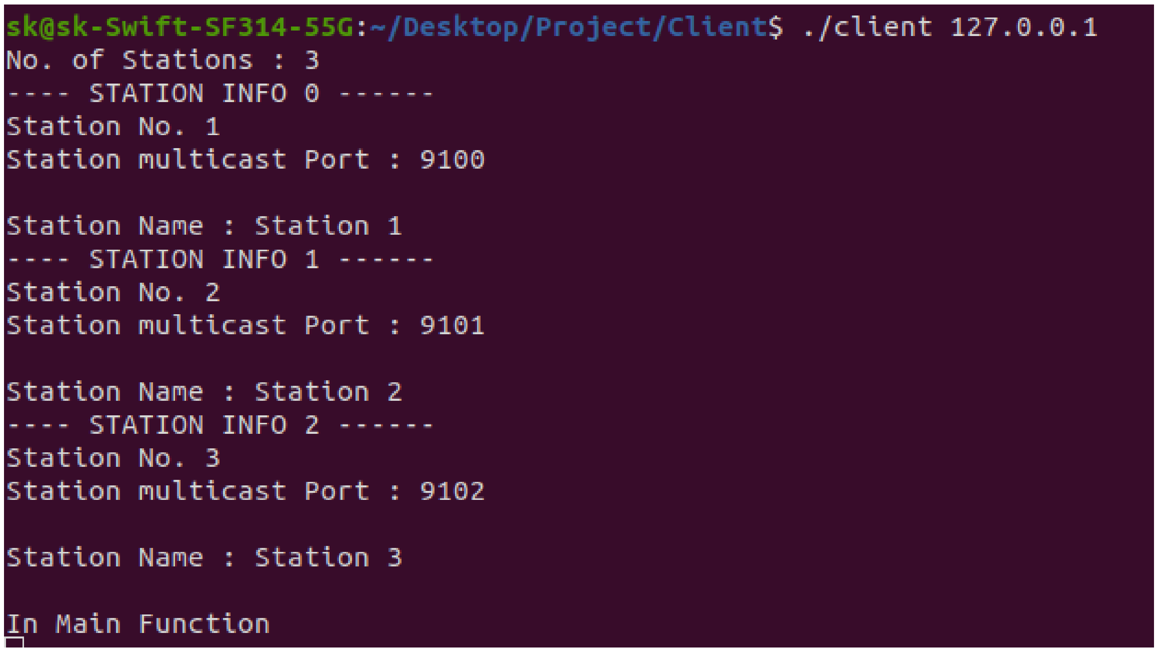
1.2 Output when server connected with Client :



```
Client and server are connected! To send structures is being started from here.
Current Song number = 0 Current Song name= 0x7f3894008ba0
Sending Structure : current Song number= 0. Song_Info->type = 12 p = 0x7f38a3ffd320
Current Song number = 1 Current Song name= 0x7f3894008d80
Sending Structure : current Song number= 1. Song_Info->type = 12 p = 0x7f38a3ffd526
Current Song number = 0 Current Song name= 0x7f3898008ba0
```

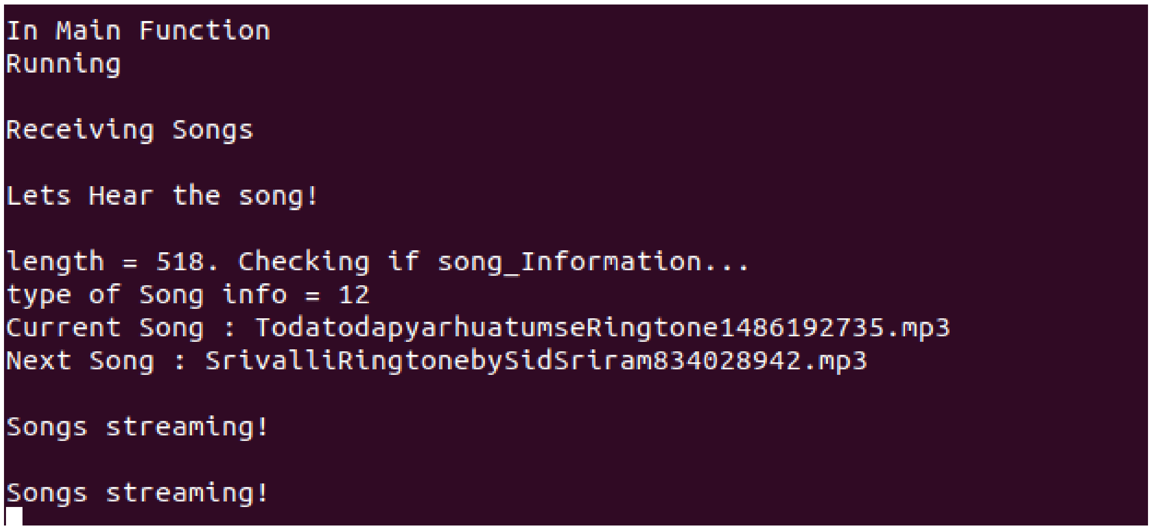
2) Client

2.1 Running Client code:



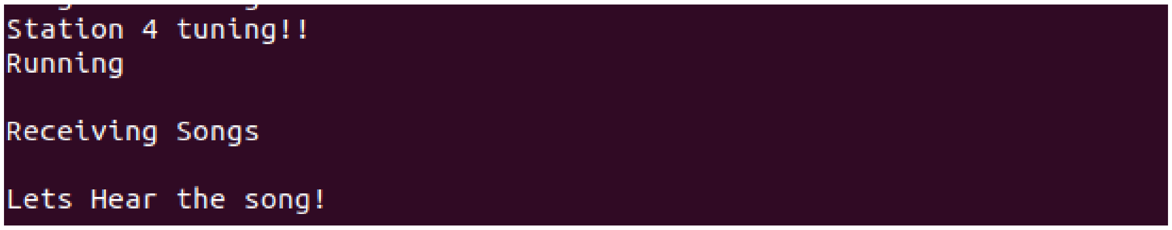
```
sk@sk-Swift-SF314-55G:~/Desktop/Project/Client$ ./client 127.0.0.1
No. of Stations : 3
---- STATION INFO 0 ------
Station No. 1
Station multicast Port : 9100

Station Name : Station 1
---- STATION INFO 1 ------
Station No. 2
Station multicast Port : 9101

Station Name : Station 2
---- STATION INFO 2 ------
Station No. 3
Station multicast Port : 9102

Station Name : Station 3

In Main Function
```

2.2 After pressing Play Radio Button



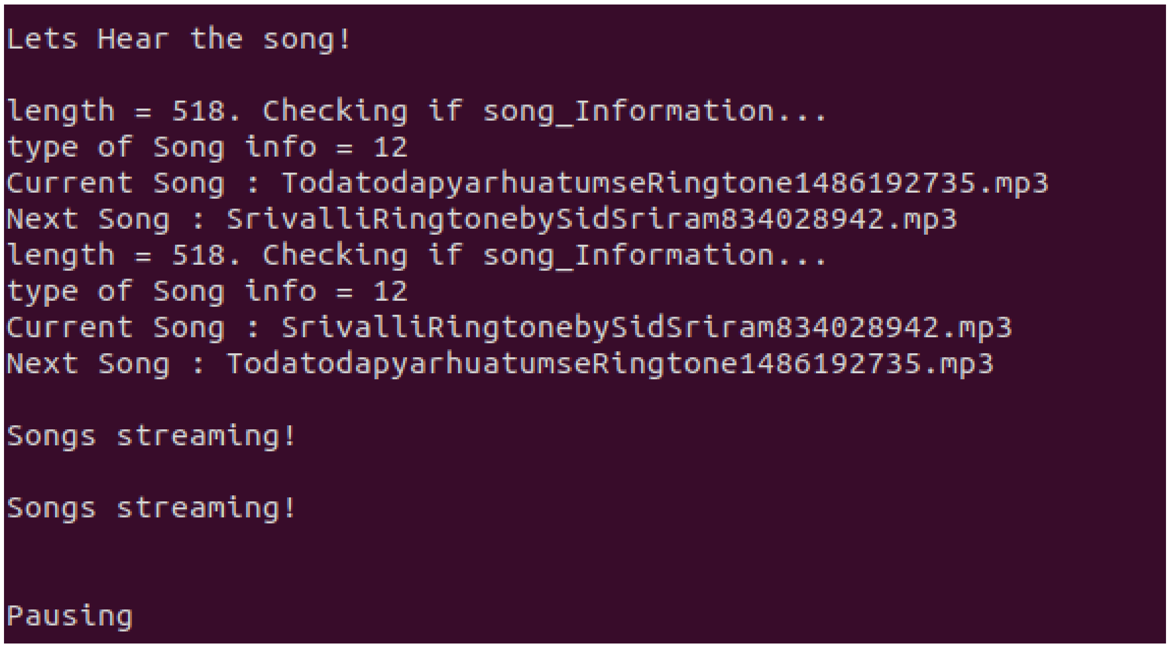
```
In Main Function
Running

Receiving Songs

Lets Hear the song!

length = 518. Checking if song_Information...
type of Song info = 12
Current Song : TodatodapyarhuatumseRingtone1486192735.mp3
Next Song : SrivalliRingtonebySidSriram834028942.mp3

Songs streaming!

Songs streaming!
```

2.3 After pressing Change Station to 4 Button



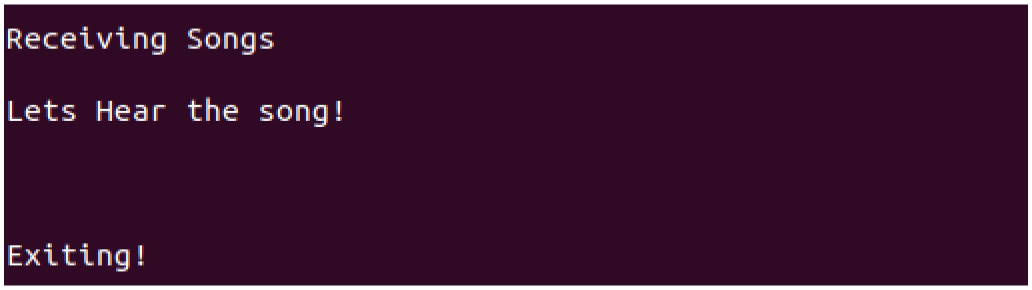
```
Station 4 tuning!!
Running

Receiving Songs

Lets Hear the song!
```

2.4 After pressing Pause Button



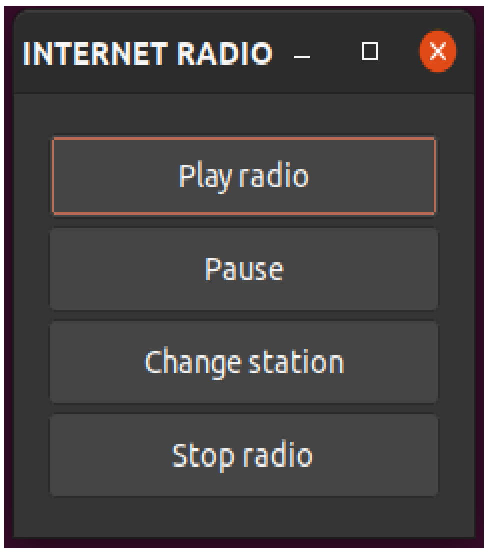
```
Lets Hear the song!

length = 518. Checking if song_Information...
type of Song info = 12
Current Song : TodatodapyarhuatumseRingtone1486192735.mp3
Next Song : SrivalliRingtonebySidSriram834028942.mp3
length = 518. Checking if song_Information...
type of Song info = 12
Current Song : SrivalliRingtonebySidSriram834028942.mp3
Next Song : TodatodapyarhuatumseRingtone1486192735.mp3

Songs streaming!

Songs streaming!


Pausing
```

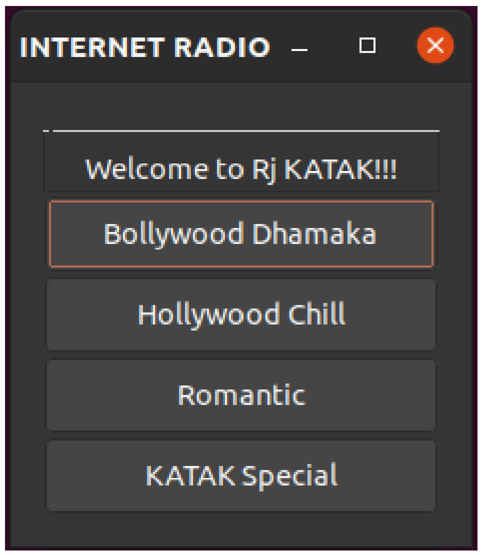
2.5 After pressing Stop radio Button



```
Receiving Songs

Lets Hear the song!



Exiting!
```

GUI Screenshots:





Contribution of each member :

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

Krunal Savaj (AU1940271)	Krina khakhariya (AU1940208)	Axay Ghoghari (AU1940269)	Tirth Kanani (AU1920144)	Arti Singhal (AU1940181)
GUI- Scratch Code and Run Radio Part	GUI-Pause Part	GUI-change Stations Part	Attempt for color grading in GUI	Attempt for TV station
Server Part In Report	Content Management In Report	Flow chart for report	Client part in Report	Background and bit rate calculations