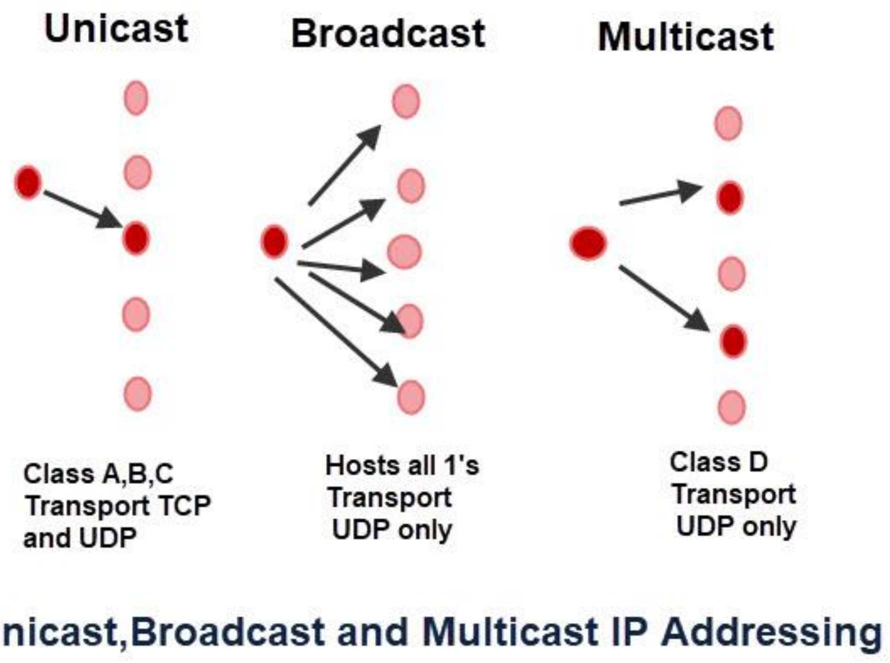
Internet Radio: Multicasting Multimedia over IP

Project Report

Introduction

The purpose of this project was to explore the IP multicast and the nature of multimedia traffic. We developed an internet TV/radio. We used our experience of sending data over a TCP connection, sending multimedia over a UDP connection, and sending data in a structured manner from the previous labs. The Any Source Multicast (ASM) model was used for multicasting.



Ideas for further development

This project can potentially be used for internet radio and live streaming of videos. Currently, this code is meant only for Linux systems but modifications in socket programming can make it viable for the windows platform too.

Implementation and design

The code basically follows the default design. It uses multiple processes and not multi-threading. Termination of a station is done by killing the process.

The application contains a menu which has options for Pause, Reset, Change of Station and Close.

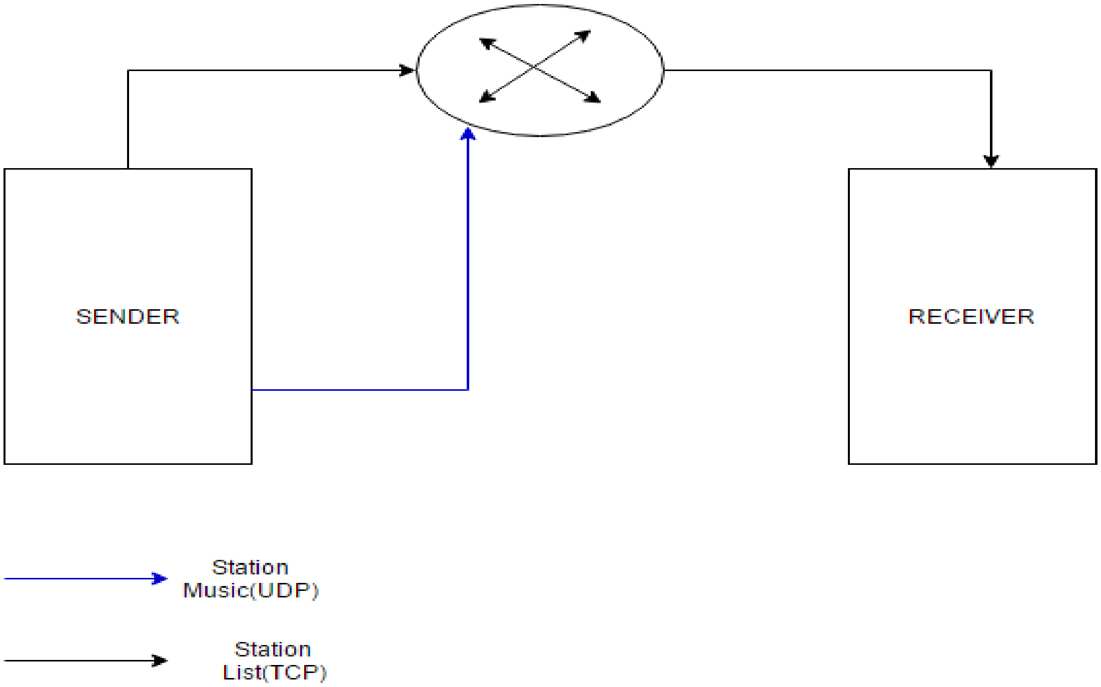
The flow of events is as follows:

- Client connects to the server
- Server sends channel information
- Client connects to the desired channel.
- Client starts receiving packets in a buffer.

System Design

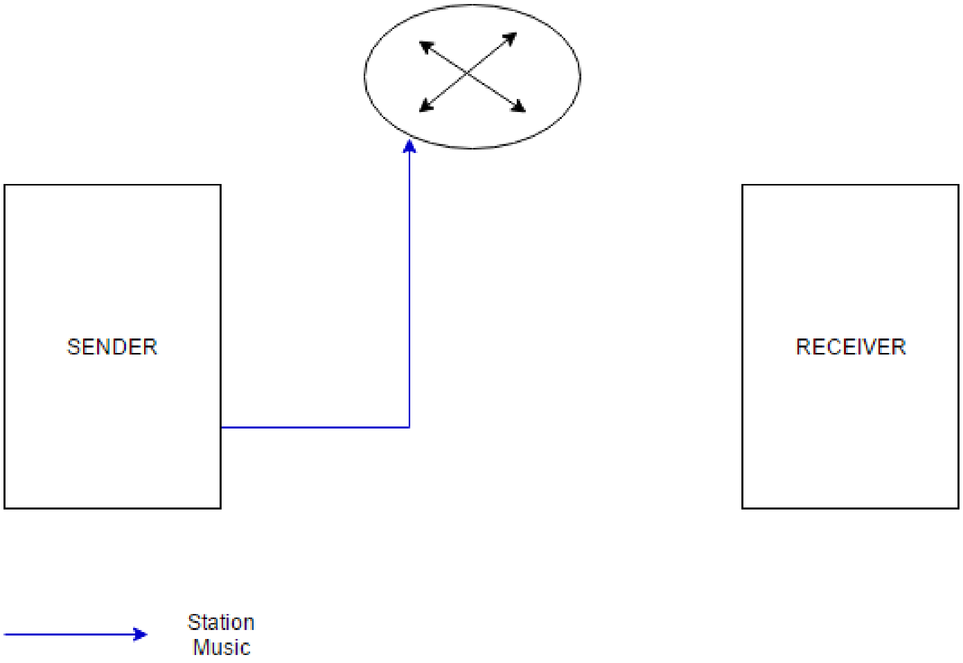
Phase 1

Sender sends TCP packet to receiver with station information.



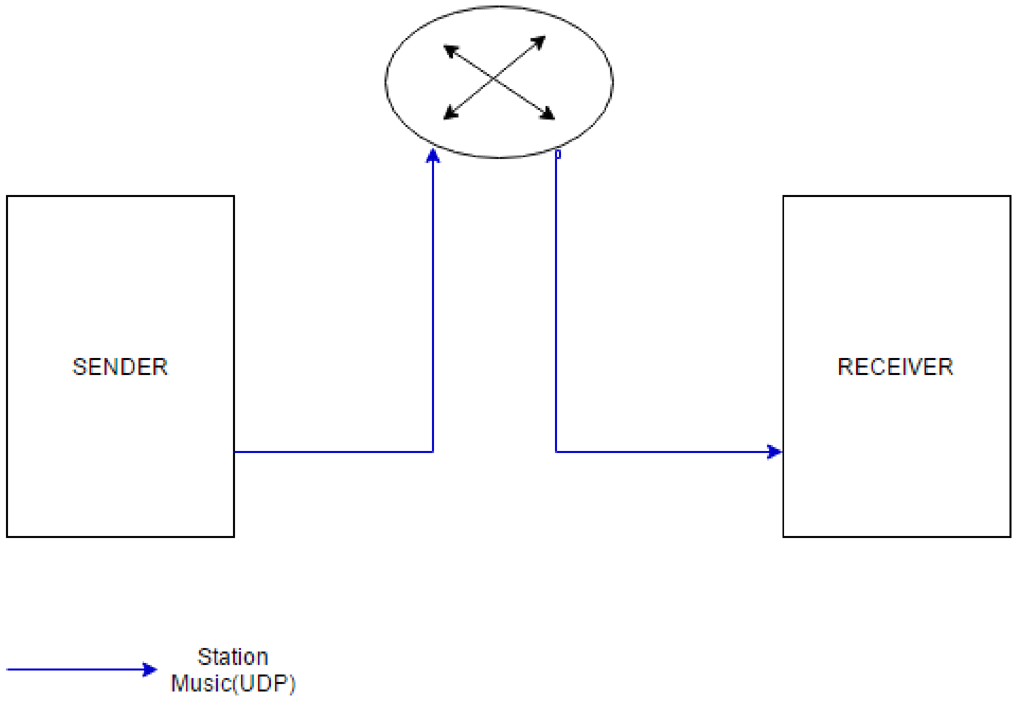
Phase 2

Receiver selects preferred channel.



Phase 3

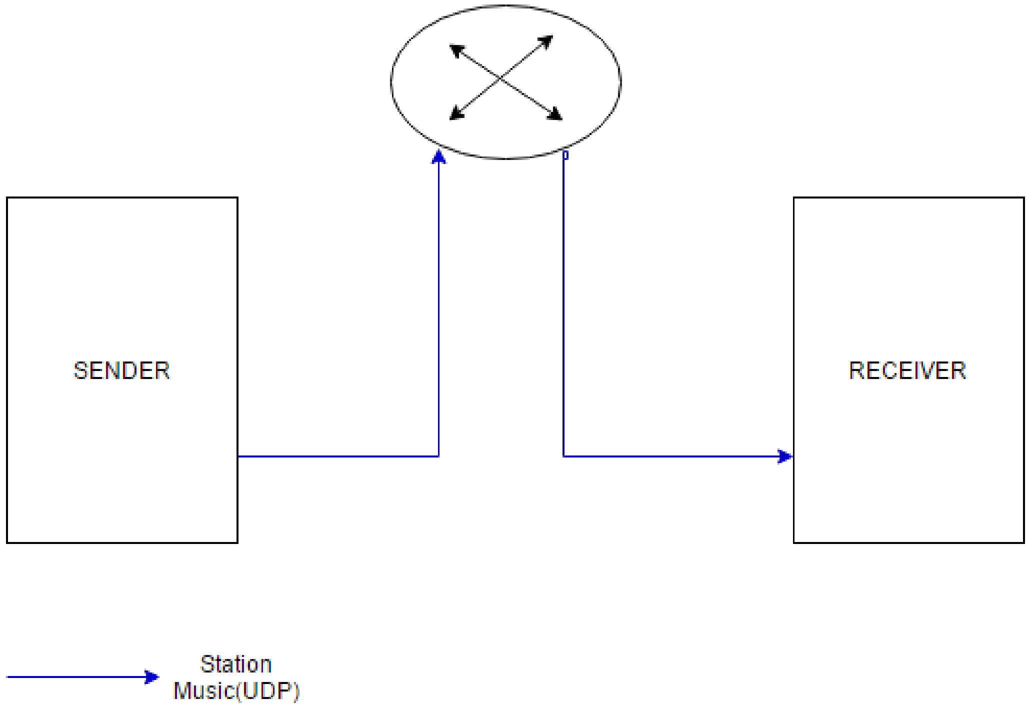
Receiver starts receiving the station music.



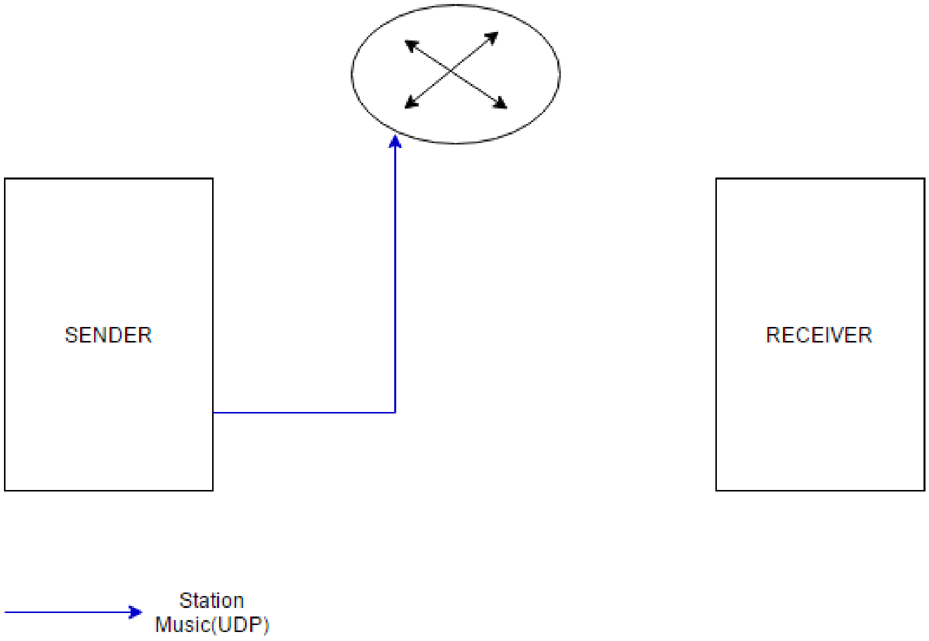
Phase 4

(Reset, Pause, Change Station, Close)

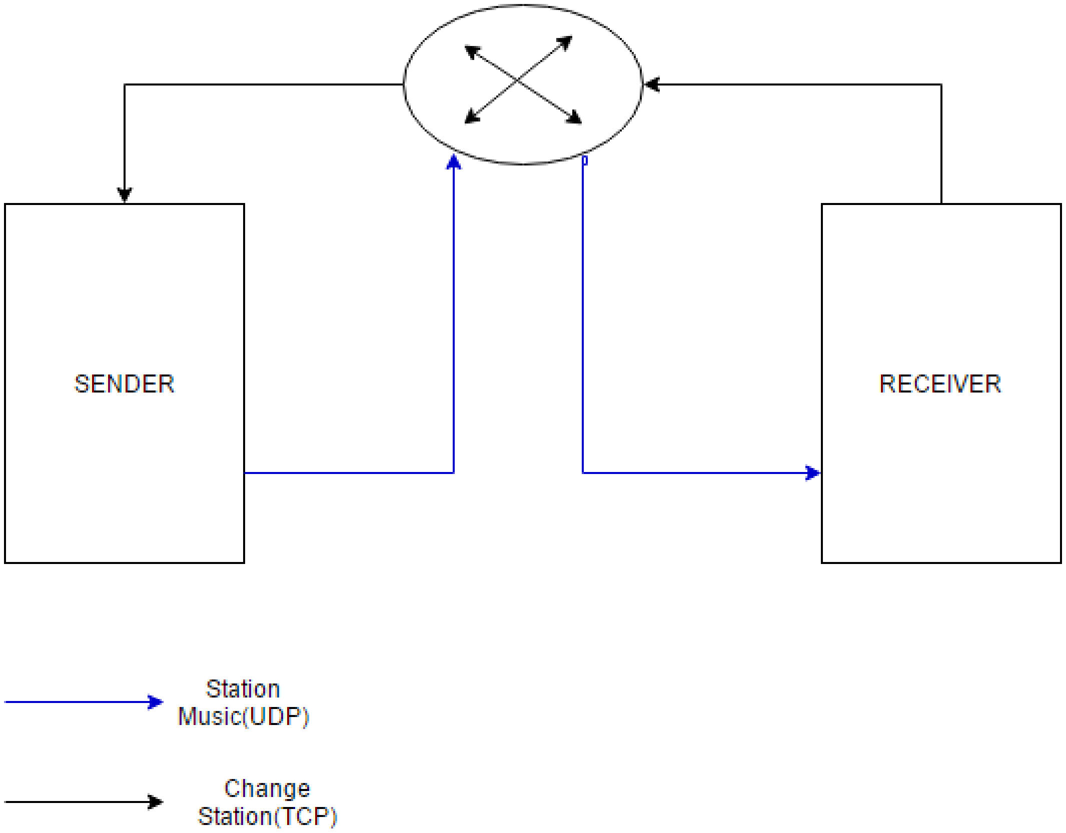
- *Reset*



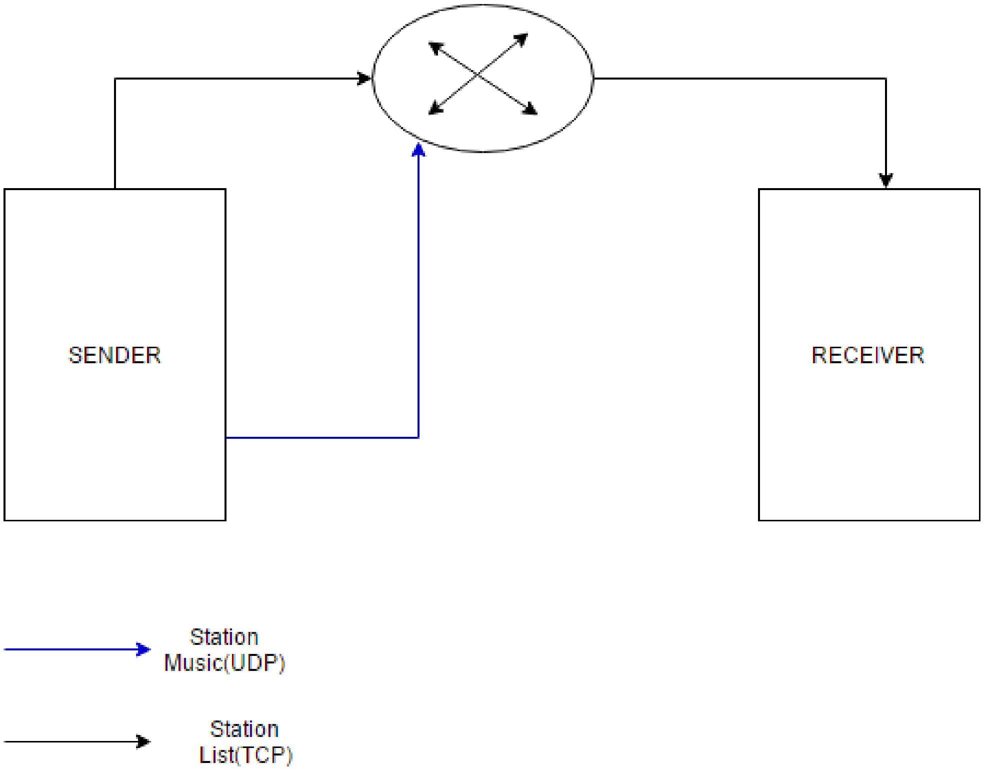
- *Pause*



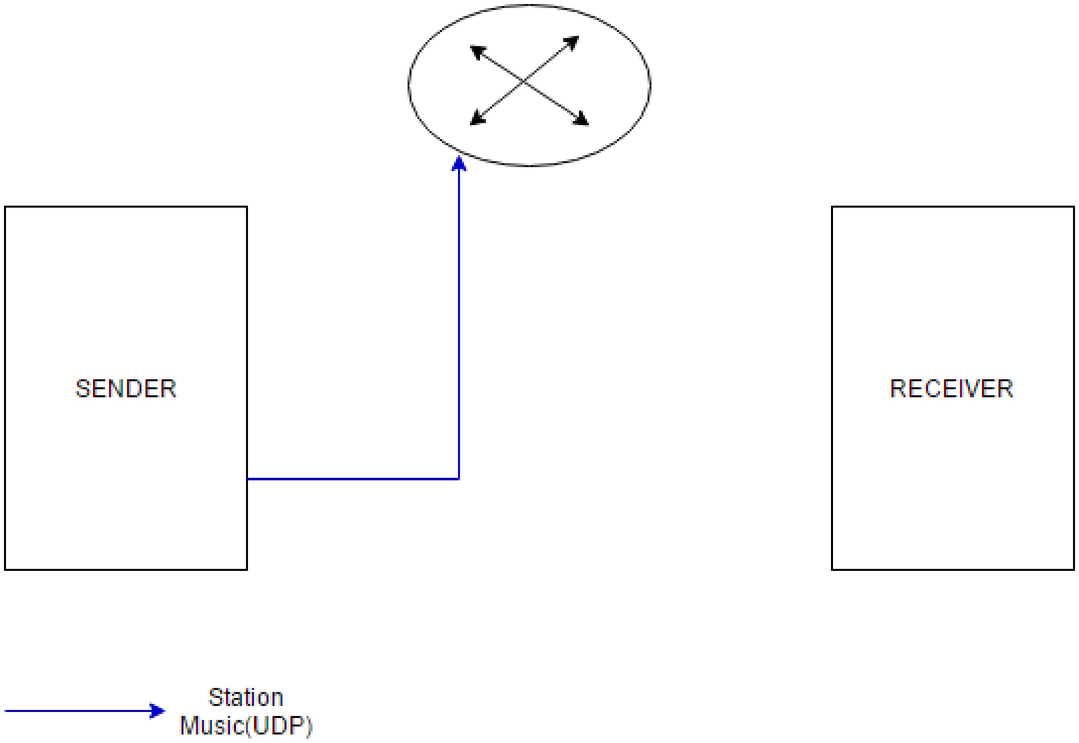
- *Change Station*
    - Receiver requests for a change in station by sending a TCP packet.



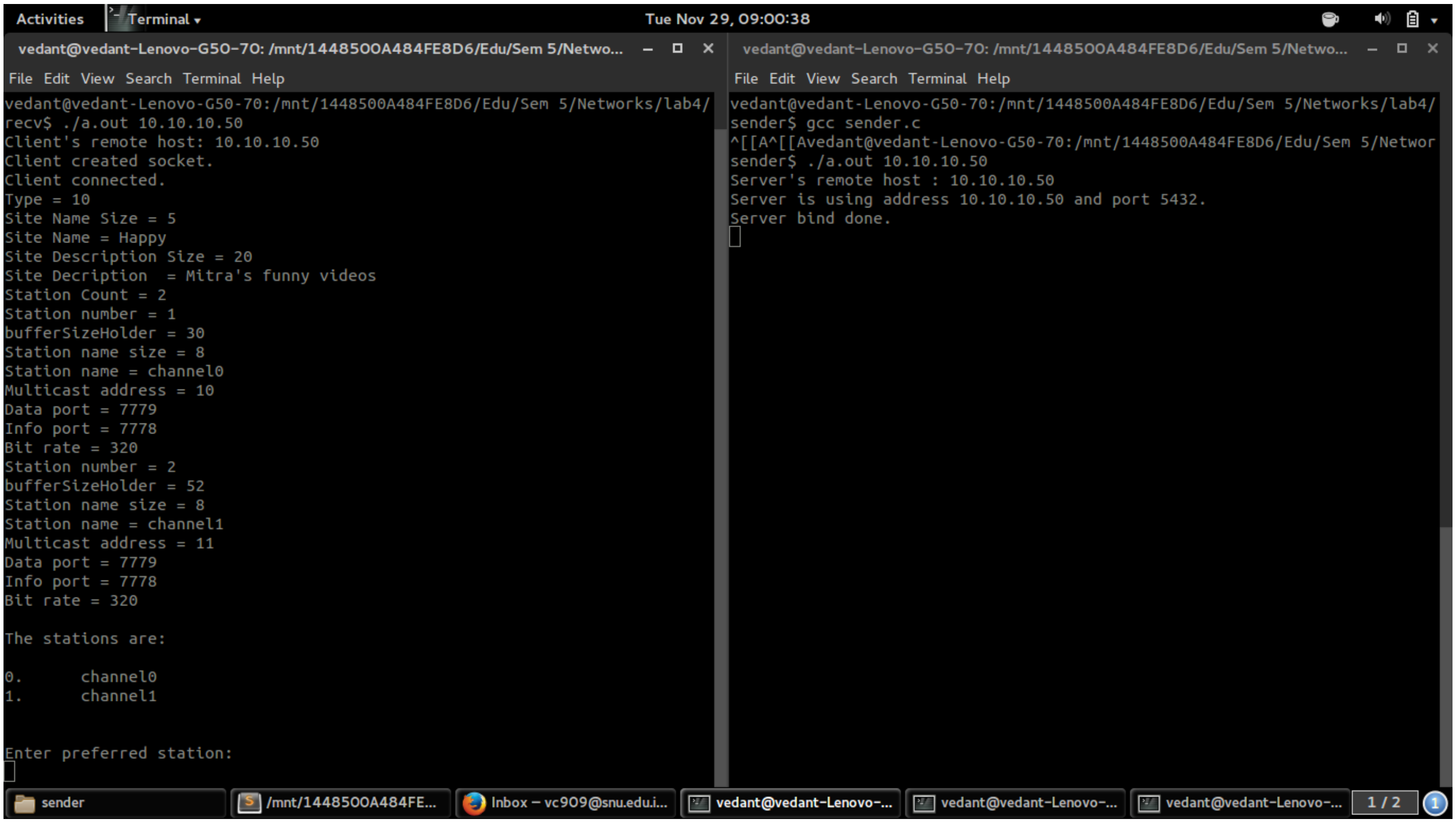
    - Sender sends a TCP packet with the station list.



- *Close Station*

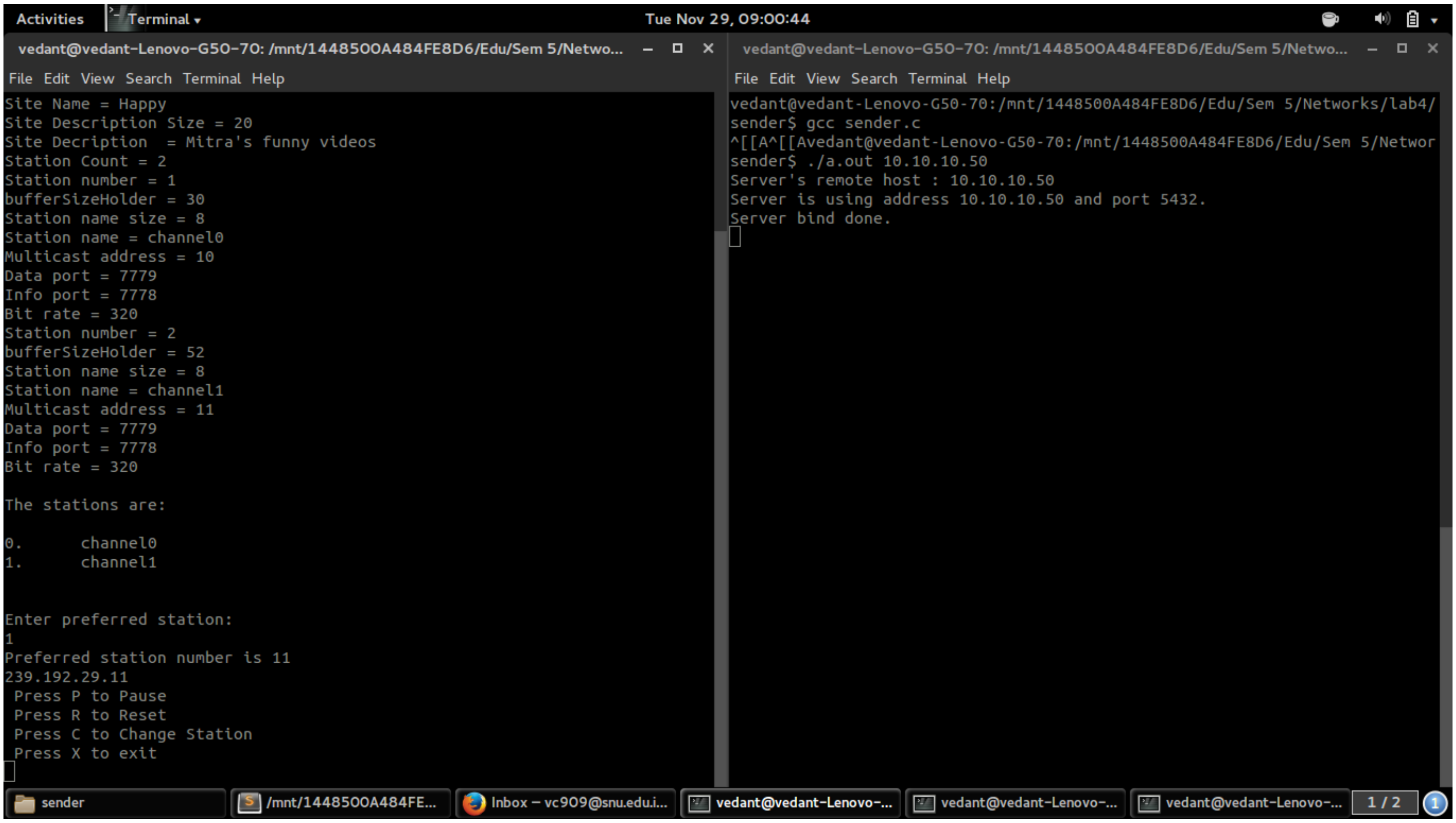


Screenshots


Activities
Terminal
Tue Nov 29, 09:00:38
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
sender
/mnt/1448500A484FE...
Inbox – vc909@snu.edu.i...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
1 / 2

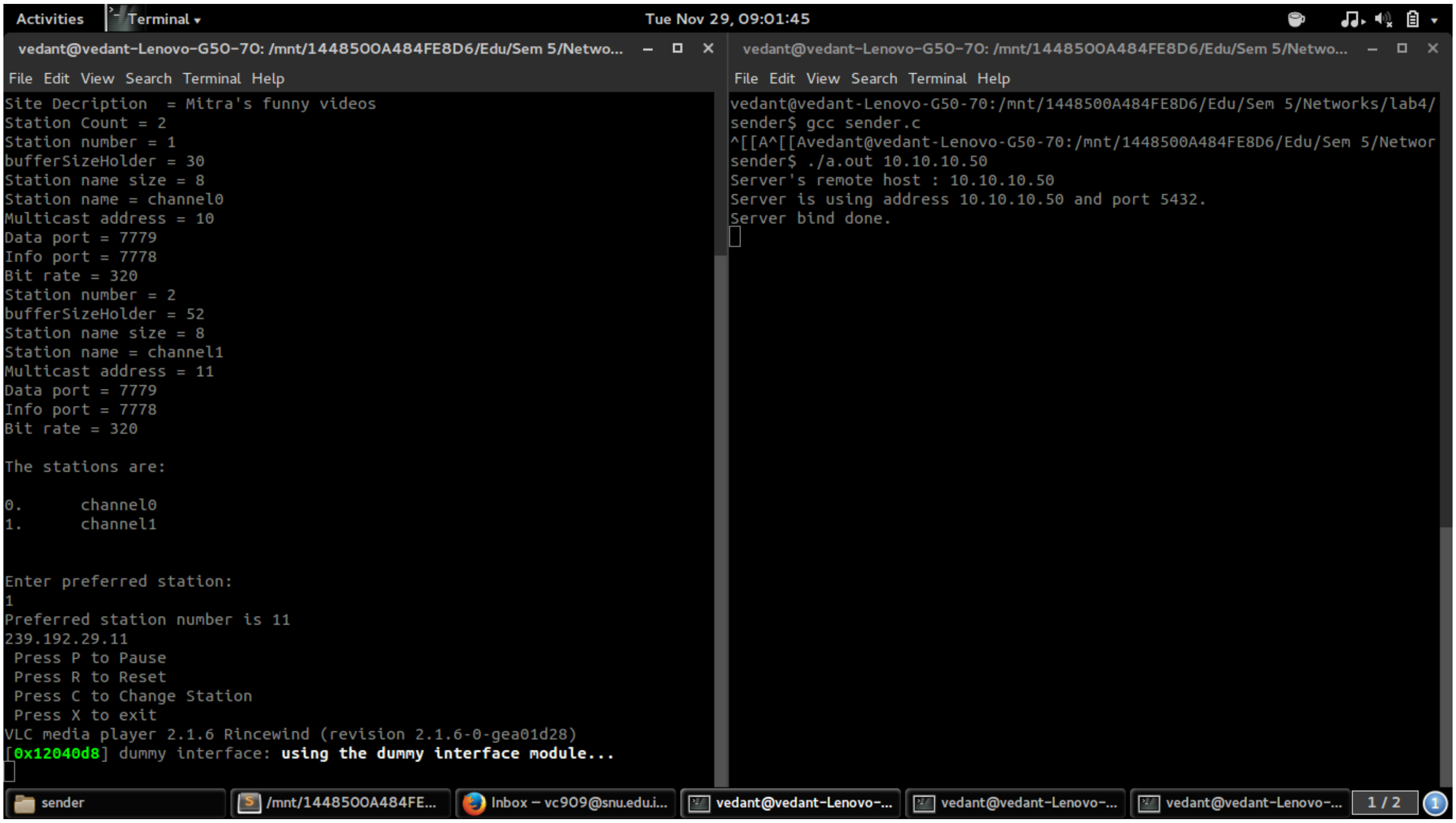
```
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
recv$ ./a.out 10.10.10.50
Client's remote host: 10.10.10.50
Client created socket.
Client connected.
Type = 10
Site Name Size = 5
Site Name = Happy
Site Description Size = 20
Site Decription  = Mitra's funny videos
Station Count = 2
Station number = 1
bufferSizeHolder = 30
Station name size = 8
Station name = channel0
Multicast address = 10
Data port = 7779
Info port = 7778
Bit rate = 320
Station number = 2
bufferSizeHolder = 52
Station name size = 8
Station name = channel1
Multicast address = 11
Data port = 7779
Info port = 7778
Bit rate = 320

The stations are:

0.      channel0
1.      channel1


Enter preferred station:
```

```
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
sender$ gcc sender.c
^[[A^[[Avedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networ
sender$ ./a.out 10.10.10.50
Server's remote host : 10.10.10.50
Server is using address 10.10.10.50 and port 5432.
Server bind done.
```


Activities
Terminal
Tue Nov 29, 09:00:44
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
sender
/mnt/1448500A484FE...
Inbox – vc909@snu.edu.i...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
1 / 2

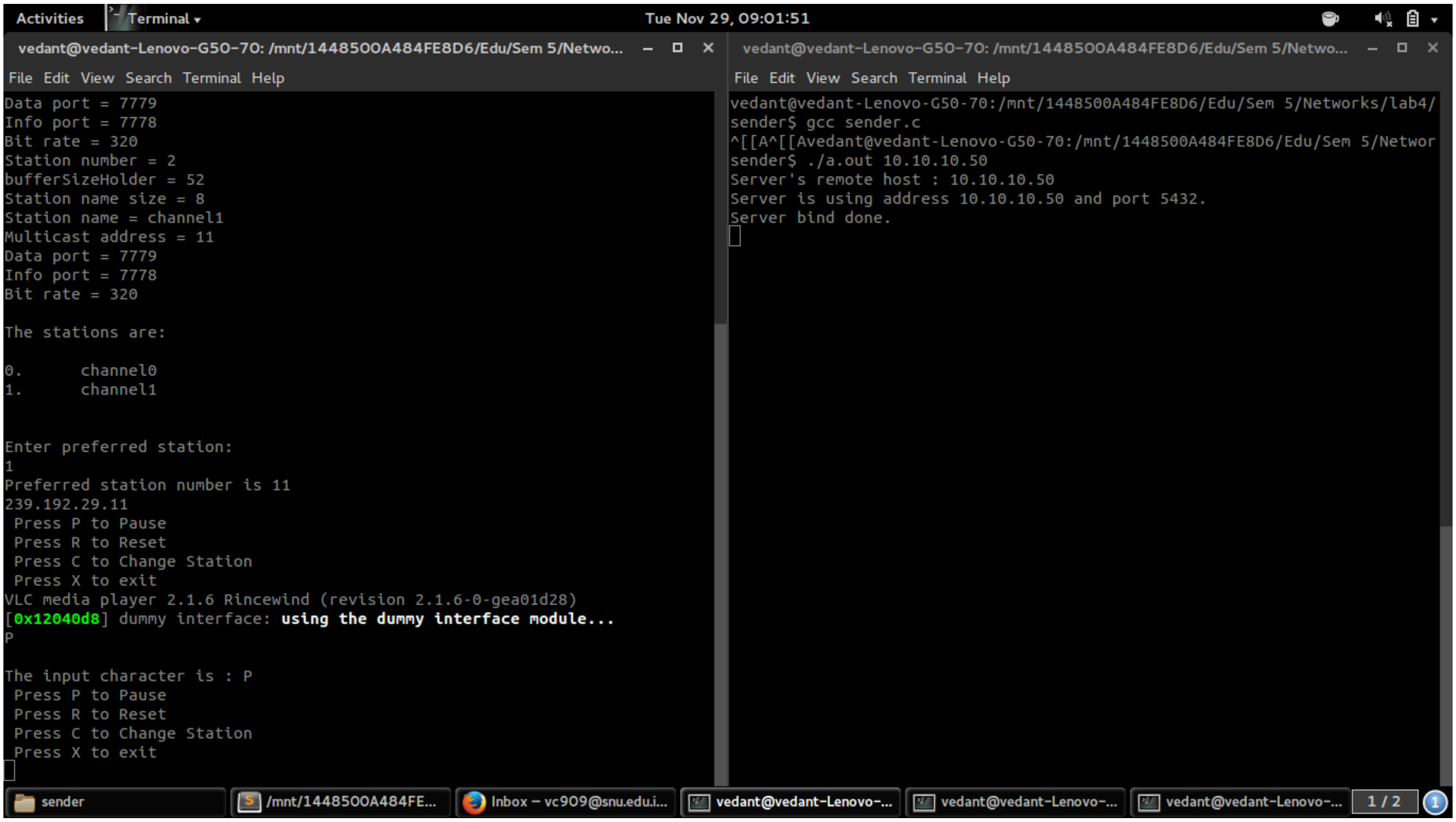
```
Site Name = Happy
Site Description Size = 20
Site Decription  = Mitra's funny videos
Station Count = 2
Station number = 1
bufferSizeHolder = 30
Station name size = 8
Station name = channel0
Multicast address = 10
Data port = 7779
Info port = 7778
Bit rate = 320
Station number = 2
bufferSizeHolder = 52
Station name size = 8
Station name = channel1
Multicast address = 11
Data port = 7779
Info port = 7778
Bit rate = 320

The stations are:

0.      channel0
1.      channel1


Enter preferred station:
1
Preferred station number is 11
239.192.29.11
 Press P to Pause
 Press R to Reset
 Press C to Change Station
 Press X to exit
```

```
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
sender$ gcc sender.c
^[[A^[[Avedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networ
sender$ ./a.out 10.10.10.50
Server's remote host : 10.10.10.50
Server is using address 10.10.10.50 and port 5432.
Server bind done.
```


Activities
Terminal ▾
Tue Nov 29, 09:01:45
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
sender
/mnt/1448500A484FE...
Inbox – vc909@snu.edu.i...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
1 / 2

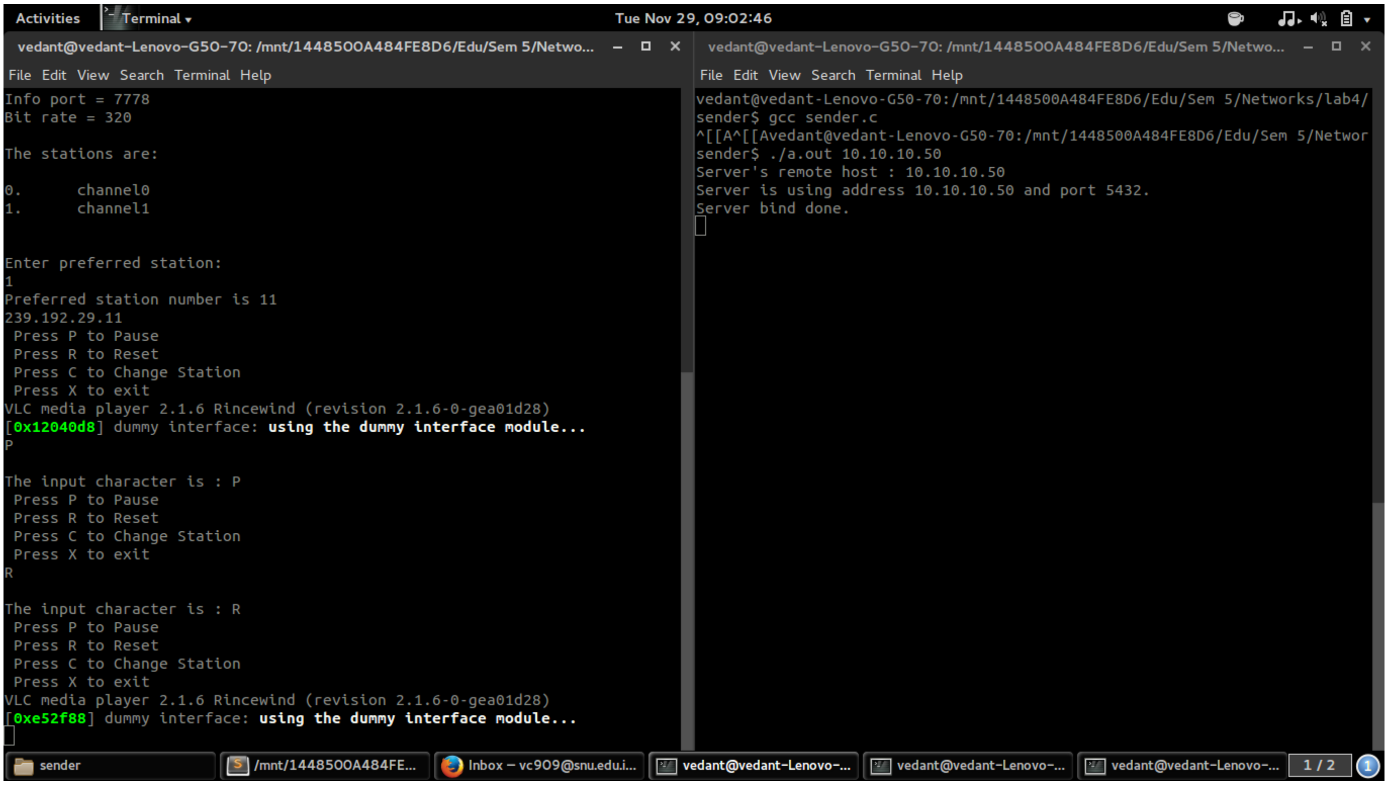
```
Site Decription  = Mitra's funny videos
Station Count = 2
Station number = 1
bufferSizeHolder = 30
Station name size = 8
Station name = channel0
Multicast address = 10
Data port = 7779
Info port = 7778
Bit rate = 320
Station number = 2
bufferSizeHolder = 52
Station name size = 8
Station name = channel1
Multicast address = 11
Data port = 7779
Info port = 7778
Bit rate = 320

The stations are:

0.      channel0
1.      channel1


Enter preferred station:
1
Preferred station number is 11
239.192.29.11
 Press P to Pause
 Press R to Reset
 Press C to Change Station
 Press X to exit
VLC media player 2.1.6 Rincewind (revision 2.1.6-0-gea01d28)
[0x12040d8] dummy interface: using the dummy interface module...
```

```
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
sender$ gcc sender.c
^[[A^[[Avedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networ
sender$ ./a.out 10.10.10.50
Server's remote host : 10.10.10.50
Server is using address 10.10.10.50 and port 5432.
Server bind done.
```


Activities
Terminal ▾
Tue Nov 29, 09:01:51
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
sender
/mnt/1448500A484FE...
Inbox – vc909@snu.edu.i...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
1 / 2

```
Data port = 7779
Info port = 7778
Bit rate = 320
Station number = 2
bufferSizeHolder = 52
Station name size = 8
Station name = channel1
Multicast address = 11
Data port = 7779
Info port = 7778
Bit rate = 320

The stations are:

0.      channel0
1.      channel1


Enter preferred station:
1
Preferred station number is 11
239.192.29.11
 Press P to Pause
 Press R to Reset
 Press C to Change Station
 Press X to exit
VLC media player 2.1.6 Rincewind (revision 2.1.6-0-gea01d28)
[0x12040d8] dummy interface: using the dummy interface module...
P

The input character is : P
 Press P to Pause
 Press R to Reset
 Press C to Change Station
 Press X to exit
```

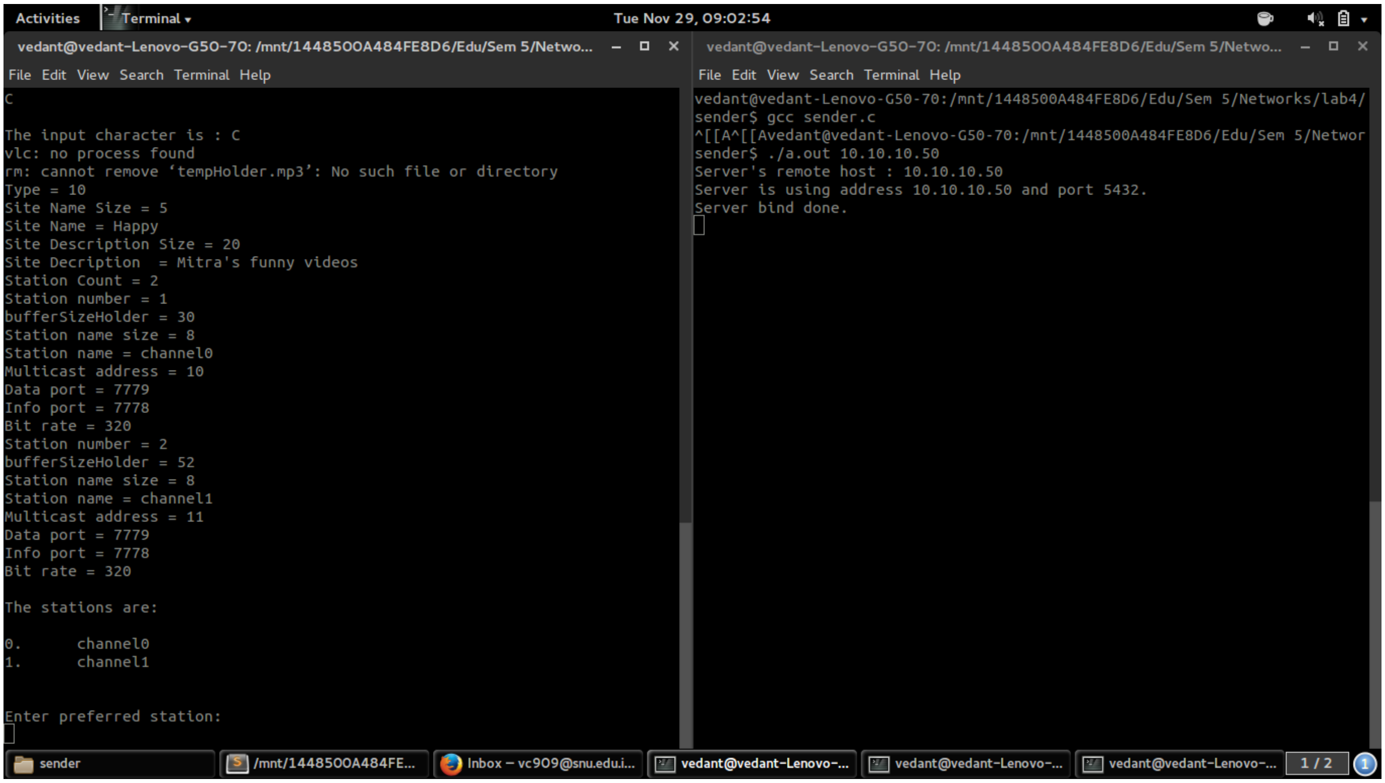
```
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
sender$ gcc sender.c
^[[A^[[Avedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networ
sender$ ./a.out 10.10.10.50
Server's remote host : 10.10.10.50
Server is using address 10.10.10.50 and port 5432.
Server bind done.
```



Activities
Terminal ▾
Tue Nov 29, 09:02:46
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
sender
/mnt/1448500A484FE...
Inbox – vc909@snu.edu.i...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
1 / 2

```
Info port = 7778
Bit rate = 320

The stations are:

0.      channel0
1.      channel1


Enter preferred station:
1
Preferred station number is 11
239.192.29.11
 Press P to Pause
 Press R to Reset
 Press C to Change Station
 Press X to exit
VLC media player 2.1.6 Rincewind (revision 2.1.6-0-gea01d28)
[0x12040d8] dummy interface: using the dummy interface module...
P

The input character is : P
 Press P to Pause
 Press R to Reset
 Press C to Change Station
 Press X to exit
R

The input character is : R
 Press P to Pause
 Press R to Reset
 Press C to Change Station
 Press X to exit
VLC media player 2.1.6 Rincewind (revision 2.1.6-0-gea01d28)
[0xe52f88] dummy interface: using the dummy interface module...
```

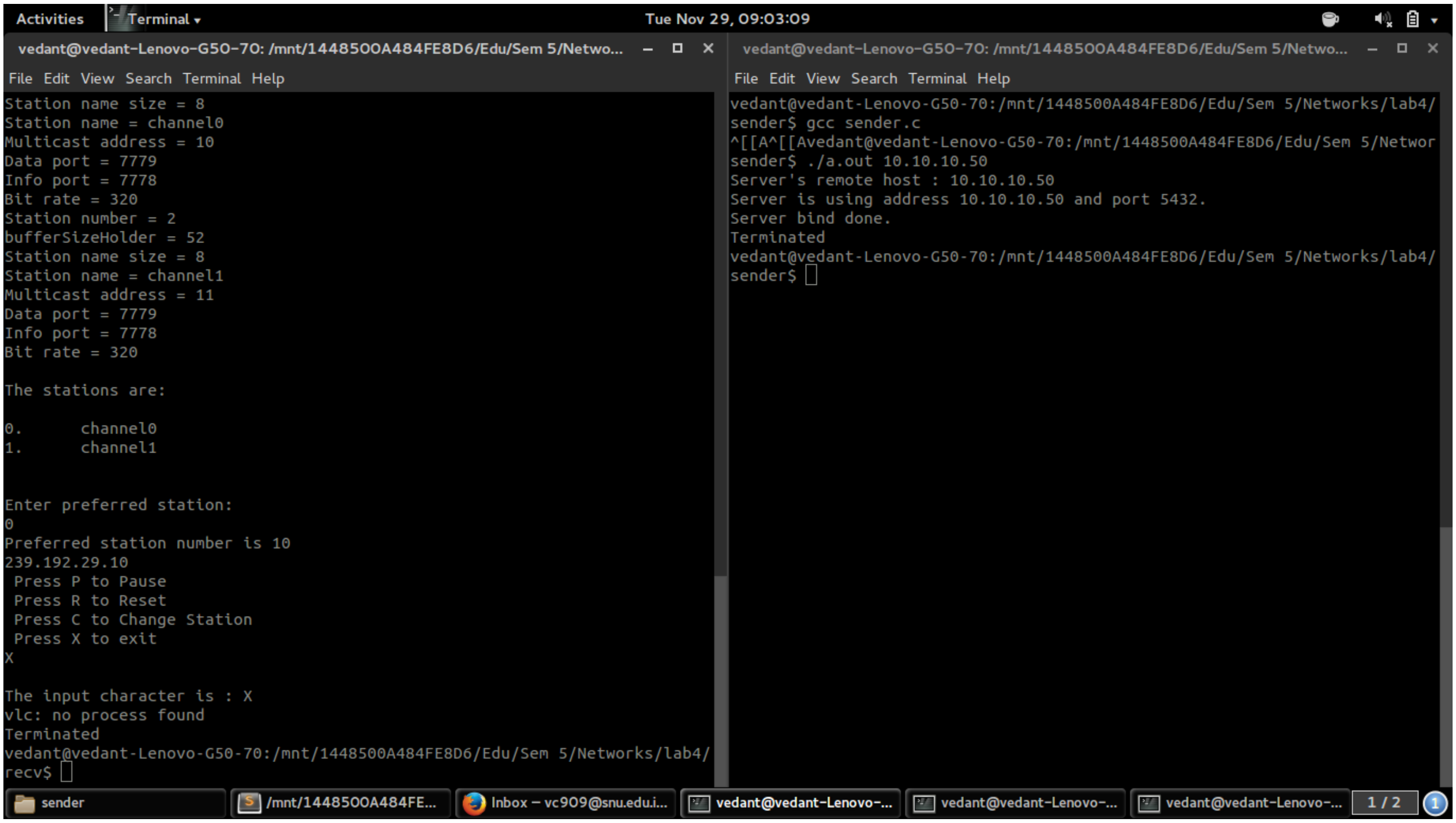
```
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
sender$ gcc sender.c
^[[A^[[Avedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networ
sender$ ./a.out 10.10.10.50
Server's remote host : 10.10.10.50
Server is using address 10.10.10.50 and port 5432.
Server bind done.
```



Activities
Terminal ▾
Tue Nov 29, 09:02:54
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
sender
/mnt/1448500A484FE...
Inbox – vc909@snu.edu.i...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
1 / 2

```
C

The input character is : C
vlc: no process found
rm: cannot remove ‘tempHolder.mp3’: No such file or directory
Type = 10
Site Name Size = 5
Site Name = Happy
Site Description Size = 20
Site Decription  = Mitra's funny videos
Station Count = 2
Station number = 1
bufferSizeHolder = 30
Station name size = 8
Station name = channel0
Multicast address = 10
Data port = 7779
Info port = 7778
Bit rate = 320
Station number = 2
bufferSizeHolder = 52
Station name size = 8
Station name = channel1
Multicast address = 11
Data port = 7779
Info port = 7778
Bit rate = 320

The stations are:

0.      channel0
1.      channel1


Enter preferred station:
```

```
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
sender$ gcc sender.c
^[[A^[[Avedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networ
sender$ ./a.out 10.10.10.50
Server's remote host : 10.10.10.50
Server is using address 10.10.10.50 and port 5432.
Server bind done.
```


Activities
Terminal
Tue Nov 29, 09:03:09
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
vedant@vedant-Lenovo-G50-70: /mnt/1448500A484FE8D6/Edu/Sem 5/Netwo...
File Edit View Search Terminal Help
sender
/mnt/1448500A484FE...
Inbox – vc909@snu.edu.i...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
vedant@vedant-Lenovo-...
1 / 2
1

```
Station name size = 8
Station name = channel0
Multicast address = 10
Data port = 7779
Info port = 7778
Bit rate = 320
Station number = 2
bufferSizeHolder = 52
Station name size = 8
Station name = channel1
Multicast address = 11
Data port = 7779
Info port = 7778
Bit rate = 320

The stations are:

0.      channel0
1.      channel1


Enter preferred station:
0
Preferred station number is 10
239.192.29.10
 Press P to Pause
 Press R to Reset
 Press C to Change Station
 Press X to exit
X

The input character is : X
vlc: no process found
Terminated
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
recv$ 
```

```
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
sender$ gcc sender.c
^[[A^[[Avedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networ
sender$ ./a.out 10.10.10.50
Server's remote host : 10.10.10.50
Server is using address 10.10.10.50 and port 5432.
Server bind done.
Terminated
vedant@vedant-Lenovo-G50-70:/mnt/1448500A484FE8D6/Edu/Sem 5/Networks/lab4/
sender$ 
```

Break up of individual contribution

Team Number: 29

S.No.	Team Member Name	Contribution Detail
1	Prasanna Natarajan	Client and Server code. Content management.
2	Siddharth Mitra	Report
3	Sridhar Ramanujam	Report
4	Vedant Chakravarthy	Client and Server code. Content management.

Code:

Sender Side:

```
/* CSD 304 Computer Networks, Fall 2016
   multicast receiver
   Team: Prasanna Natarajan
                        Siddhart Mitra
                        Sridhar Renga Ramanujam
                        Vedant Chakravarthy
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <net/if.h>
#include <netdb.h>
#include <sys/ioctl.h>
#include <signal.h>

#define TCP_PORT 5432
#define MC_PORT 7779
#define BUF_SIZE 64480
#define DEBUG 0
#define DEBUG2 0


typedef struct station_info_request {
    uint8_t type;
}
station_info_request_t;

typedef struct song_info {
    uint8_t type;
    uint8_t song_name_size;
    char song_name;
    uint16_t remaining_time_in_sec;
    uint8_t next_song_name_size;
    char next_song_name;
}
song_info_t;

/*function prototypes*/
void Serialize_station_info_req(station_info_request_t * buf, char * x);

/*Helper Functions*/
void Serialize_station_info_req(station_info_request_t * buf, char * x) {
    int i;
    char buffer[1024];
    memcpy(buffer, & (buf -> type), 1);

    if (DEBUG == 1) {
        puts(">>Print memcpy buffer");
        for (i = 0; i < 1; i++) {
            printf("%d", buffer[i]);
        }
    }
    memcpy(x, buffer, sizeof(buffer));
    if (DEBUG == 1)
        puts("End Serialize_station_info_req");
}

int main(int argc, char * argv[]) {

    int s, s_tcp; /* socket descriptor */
    struct sockaddr_in sin, sin_tcp; /* socket struct */
    char * if_name; /* name of interface */
    struct ifreq ifr; /* interface struct */
    char buf[BUF_SIZE];
    int len;
    /* Multicast specific */
    // char * mcast_addr; /* multicast address */
    struct ip_mreq mcast_req; /* multicast join struct */
    struct sockaddr_in mcast_saddr; /* multicast sender*/
    socklen_t mcast_saddr_len;

    /*TCP specific*/
    char * tcp_addr; /*tcp ip address for control codes*/
    struct hostent * hp;

    /* Add code to take port number from user */
    if ((argc == 2) || (argc == 3)) {
        tcp_addr = argv[1];
    }
    else {
        fprintf(stderr, "usage:(sudo) receiver tcp_address [interface_name (optional)]\n");
        exit(1);
    }

    if (argc == 3) {
        if_name = argv[2];
    }
    else
        if_name = "wlan0";

    /* translate host name into peer's IP address */
    hp = gethostbyname(tcp_addr);
    if (!hp) {
        fprintf(stderr, "simplex-talk: unknown host: %s\n", tcp_addr);
        exit(1);
    }
    else
                                    printf("Client's remote host: %s\n", argv[1]);
    /* build address data structure */
    bzero((char * ) &sin_tcp, sizeof(sin_tcp));
    sin_tcp.sin_family = AF_INET;
    bcopy(hp -> h_addr, (char * ) & sin_tcp.sin_addr, hp -> h_length);
    sin_tcp.sin_port = htons(TCP_PORT);
    /* active open */
    if ((s_tcp = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        perror("simplex-talk: socket");
        exit(1);
    }
    else
        printf("Client created socket.\n");
```

```
if (connect(s_tcp, (struct sockaddr *)&sin_tcp, sizeof(sin_tcp)) < 0) {
    perror("simplex-talk: connect");
    close(s);
    exit(1);
}
else
    printf("Client connected.\n");
// puts("I beg of you, mister gobolan");
station_info_request_t * req;
req = malloc(sizeof(struct station_info_request) * sizeof(char));
req -> type = 1;

// puts("Creating station_info_request_t");

char buf_tosend[1024];
char receivedBuffer[20000];

uint8_t decoded_type;
uint8_t decoded_site_name_size;
char * decoded_site_name;
uint8_t decoded_site_desc_size;
char * decoded_site_desc;
uint8_t decoded_station_count;
int bufferSizeHolder = 0;

uint8_t * decoded_station_name_size;
char * * decoded_station_name;
uint32_t * decoded_multicast_address;
uint16_t * decoded_data_port;
uint16_t * decoded_info_port;
uint32_t * decoded_bit_rate;

char buffer_recv[BUF_SIZE];

int tempx;

              char *default_mcast_addr;
              default_mcast_addr = (char*)malloc(sizeof(char)*30);
              int preferred_station_number;

pid_t forkResult;
              int file_counter = -1;
              int scanfCounter = -1;
              while(1)
              {
                                CHANGE:file_counter++;
                                // puts("before send loop starts");
                                bufferSizeHolder = 0;
                  Serialize_station_info_req(req, buf_tosend);
                  send(s_tcp, buf_tosend, 1024, 0);
                  if(DEBUG == 1)
                                puts("before recv");
                  recv(s_tcp, receivedBuffer, sizeof(receivedBuffer), 0);
                  if(DEBUG == 1)
                                puts("After recv");
                  /*
                  1) Decode the site_info struct and put it into correponding variables
                    uint8_t type;
                    uint8_t site_name_size;
```

```
  char * site_name;
  uint8_t site_desc_size;
  char * site_desc;
  uint8_t station_count;*/
//Assuming everything is in a thing called char* receivedBuffer

memcpy( & decoded_type, receivedBuffer + bufferSizeHolder, 1);
bufferSizeHolder += 1;
printf("Type = %d\n",decoded_type );

memcpy( & decoded_site_name_size, receivedBuffer + bufferSizeHolder, 1);
bufferSizeHolder += 1;
printf("Site Name Size = %d\n",decoded_site_name_size );

decoded_site_name = (char*)malloc(sizeof(char)*decoded_site_name_size);
memcpy(decoded_site_name, receivedBuffer + bufferSizeHolder, decoded_site_name_size);
bufferSizeHolder += decoded_site_name_size;
printf("Site Name = %s\n",decoded_site_name );

memcpy( & decoded_site_desc_size, receivedBuffer + bufferSizeHolder, 1);
bufferSizeHolder += 1;
printf("Site Description Size = %d\n",decoded_site_desc_size );

decoded_site_desc = (char*)malloc(decoded_site_desc_size);
memcpy(decoded_site_desc, receivedBuffer + bufferSizeHolder, decoded_site_desc_size);
bufferSizeHolder += decoded_site_desc_size;
printf("Site Decription  = %s\n", decoded_site_desc);

memcpy( & decoded_station_count, receivedBuffer + bufferSizeHolder, 1);
bufferSizeHolder += 1;
printf("Station Count = %d\n", decoded_station_count);

decoded_station_name_size =(uint8_t*) malloc(sizeof(uint8_t)*decoded_station_count);
decoded_station_name = (char **) malloc(decoded_station_count * sizeof(char*));
decoded_multicast_address = (uint32_t *) malloc(decoded_station_count * sizeof(uint32_t));
decoded_data_port = (uint16_t *) malloc(decoded_station_count * sizeof(uint16_t));
decoded_info_port = (uint16_t *) malloc(decoded_station_count * sizeof(uint16_t));
decoded_bit_rate = (uint32_t *) malloc(decoded_station_count * sizeof(uint32_t));


for (tempx = 0; tempx < decoded_station_count; tempx++) {
        printf("Station number = %d\n",tempx+1);fflush(stdout);
        bufferSizeHolder += 1;
        printf("bufferSizeHolder = %d\n",bufferSizeHolder);fflush(stdout);
  memcpy( & decoded_station_name_size[tempx], receivedBuffer + bufferSizeHolder, 1);
  bufferSizeHolder += 1;
  printf("Station name size = %d\n",decoded_station_name_size[tempx]);fflush(stdout);
  // printf("Station name size = %d\n",*(receivedBuffer + bufferSizeHolder-1));fflush(stdout);
  // decoded_station_name[tempx] = (char*)malloc(sizeof(char)*decoded_station_name_size);

                  decoded_station_name[tempx] = (char*)malloc(sizeof(char)*decoded_station_name_size[tempx]);
  memcpy(decoded_station_name[tempx], receivedBuffer + bufferSizeHolder, decoded_station_name_size[tempx]);
  bufferSizeHolder += decoded_station_name_size[tempx];
  printf("Station name = %s\n",decoded_station_name[tempx] );fflush(stdout);
  // printf("Station name = %c\n", *(receivedBuffer + bufferSizeHolder - decoded_station_name_size[tempx]));

  memcpy( & decoded_multicast_address[tempx], receivedBuffer + bufferSizeHolder, 4);
  bufferSizeHolder += 4;
  decoded_multicast_address[tempx] = ntohs(decoded_multicast_address[tempx]);
```

```
    printf("Multicast address = %d\n",decoded_multicast_address[tempx] );

    memcpy( & decoded_data_port[tempx], receivedBuffer + bufferSizeHolder, 2);
    bufferSizeHolder += 2;
    decoded_data_port[tempx] = ntohs(decoded_data_port[tempx]);
    printf("Data port = %d\n",decoded_data_port[tempx] );

    memcpy( & decoded_info_port[tempx], receivedBuffer + bufferSizeHolder, 2);
    bufferSizeHolder += 2;
    decoded_info_port[tempx] = ntohs(decoded_info_port[tempx]);
    printf("Info port = %d\n",decoded_info_port[tempx] );

    memcpy( & decoded_bit_rate[tempx], receivedBuffer + bufferSizeHolder, 4);
    bufferSizeHolder += 4;
    decoded_bit_rate[tempx] = ntohs(decoded_bit_rate[tempx]);
    printf("Bit rate = %d\n",decoded_bit_rate[tempx] );
}

// printf("decoded_station_count = %d\n",decoded_station_count);
/*2) Display recv_buf->station_list[] and get user input and set up a multicast recv socket(s)*/
puts("");
puts("The stations are:");
for (tempx = 0; tempx < decoded_station_count; tempx++) {
    printf("\n%d.\t%s", tempx, decoded_station_name[tempx]);
}
puts("");
puts("");
puts("");
fflush(stdout);
// if(scanfCounter == -1){
//          puts("Enter preferred station :");
//          fflush(stdout);
//          scanf(" %d",&preferred_station_number);
//          scanfCounter = 1;
// }



//scanf(" %d",&preferred_station_number);
//char ch[2];
//printf("Press Y to continue:\n");
        /*do{
                fflush(stdout);
                // printf("Press Y to continue:");fflush(stdout);fflush(stdin);
                // scanf(" %c",&ch);
                ch[0] = fgetc(stdin);
                fputc(ch[0],stdout);
                fflush(stdout);
                ch[1] = fgetc(stdin);
                fputc(ch[1],stdout);
                fflush(stdout);

                if(ch[0]=='Y'||ch[1]=='Y')
                        break;
                // ch = fgetc(stdin);

                // printf("This is the fucking character = %d",ch);fflush(stdout);

                fflush(stdin);
        }while(ch[0] !='Y' || ch[1] != 'Y');
        ch[0] = 'N';
*/// while((ch = fgetc(stdin)) != 'Y');
        puts("Enter preferred station:");
scanf(" %d",&preferred_station_number);

printf("Preferred station number is %d\n",preferred_station_number+10);
sprintf(default_mcast_addr,"239.192.29.%d",preferred_station_number+10);
puts(default_mcast_addr);


        FORK:forkResult = fork();


        if(DEBUG2 == 1)
                        printf("after forking = %d",forkResult);
if(forkResult == 0){
        if(DEBUG2 == 1)
                        puts("inside fork child");
        if(DEBUG2 == 1)
                        printf("pid = %d\n",getpid());
        /* create socket */
            // puts("sending Multicast join request");
            if ((s = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {
                perror("receiver: socket");
                exit(1);
            }

            /* build address data structure */
            memset((char * ) & sin, 0, sizeof(sin));
            sin.sin_family = AF_INET;
            sin.sin_addr.s_addr = htonl(INADDR_ANY);
            sin.sin_port = htons(MC_PORT);

            /*Use the interface specified */
            memset( & ifr, 0, sizeof(ifr));
            strncpy(ifr.ifr_name, if_name, sizeof(if_name) - 1);

            /*if ((setsockopt(s, SOL_SOCKET, SO_BINDTODEVICE, (void * ) & ifr,
                    sizeof(ifr))) < 0) {
                perror("receiver: setsockopt() error");
                close(s);
                exit(1);
            }*/

            /* bind the socket (MULTICAST)*/
            if ((bind(s, (struct sockaddr * ) & sin, sizeof(sin))) < 0) {
                perror("receiver: bind()");
                exit(1);
            }
            if(DEBUG2 == 1)
                        puts("after bind to multicast");
            /* Multicast specific code follows */

            /* build IGMP join message structure */
            mcast_req.imr_multiaddr.s_addr = inet_addr(default_mcast_addr); //default_mcast_addr
            mcast_req.imr_interface.s_addr = htonl(INADDR_ANY);

            /* send multicast join message */
            if ((setsockopt(s, IPPROTO_IP, IP_ADD_MEMBERSHIP,
           (void * ) & mcast_req, sizeof(mcast_req))) < 0) {
         perror("mcast join receive: setsockopt()");
         exit(1);
        }

        if(DEBUG2 == 1)
               puts("before file handling");
        // char* temp_filename = malloc(sizeof(char)*10);
        // sprintf(temp_filename,"%d.mp3",file_counter);

        FILE *fp_recv = fopen("tempHolder.mp3","wb");
        if(fp_recv == NULL){
             if(DEBUG2 == 1)
                    puts("file pointer is null");
        }

        int once=0,l=0;
        /* receive multicast messages */
        while(1){
               /* reset sender struct */
               memset(&mcast_saddr, 0, sizeof(mcast_saddr));
               mcast_saddr_len = sizeof(mcast_saddr);

               /* clear buffer and receive */
               memset(buffer_recv, 0, sizeof(buffer_recv));
               // add file handling instead of printing the buf and also unpack the correct struct
               if(DEBUG2 == 1)
                    puts("waiting on recv");
               if ((len = recvfrom(s, buffer_recv, BUF_SIZE, 0, (struct sockaddr*)&mcast_saddr,
             &mcast_saddr_len)) < 0) {
                puts(buffer_recv);
                perror("receiver: recvfrom()");
                exit(1);
               }
                    if(DEBUG2 == 1)
                      printf("pid = %d\n",getpid());
                    if(DEBUG2 == 1)
                           printf("Length : %d",len);
               // puts(buffer_recv);
               // puts("inside while after recv");
               // fputs(buffer_recv, fp_recv);
               once++;
               // printf("once = %d\n",once);
               fflush(stdout);
               for(l=0;l<len;l++){
                    //puts("inside writing");
                    //fputc(buffer_recv[l],stdout);
fflush(fp_recv);
                    fputc(buffer_recv[l],fp_recv);
fflush(fp_recv);
               }
               if(once == 15){
                    if(DEBUG2 == 1)
                           puts("once==25");
                           system("ffmpeg -loglevel fatal -i tempHolder.mp3 -f mp2 - | ffplay - &");

               }
               fflush(stdout);
               fflush(fp_recv);
        }

}
else
{
    char ch;
    // printf("\n pid before switch = %d\n ",forkResult);
    while(ch!='X'){
            printf(" Press P to Pause \n");
            printf(" Press R to Reset \n");
            printf(" Press C to Change Station \n");
            printf(" Press X to exit \n");
            fflush(stdin);
            //scanf("%c",&ch);
            scanf(" %c",&ch);
            /*do{
                ch = fgetc(stdin);
            }while(ch=='\n');*/
            // while((ch = fgetc(stdin)) == '\n');
            fflush(stdin);
            ch = toupper(ch);

            printf("\nThe input character is : %c\n",ch);
            fflush(stdout);

            switch(ch){
            case 'P':
                kill(forkResult, SIGTERM);
                system("killall ffplay");
                system("rm tempHolder.mp3");
                // system("clear");
                //prepare penetration
                break; //sigkill child
            case 'R':
                // kill(forkResult, SIGCONT);
                // system("clear");
                goto FORK;
                //set variable to penetrate
                break; //fork
            case 'C':

                system("killall ffplay");
                system("rm tempHolder.mp3");

                kill(forkResult, SIGTERM);
                // system("clear");
                // puts("Enter preferred station :\n");
//          scanf(" %d",&preferred_station_number);
            goto CHANGE;
                break;
            case 'X':
                system("killall ffplay");
                system("killall a.out");
                // system("clear");
                exit(0);
                break;
            default :
                // puts("in default");
                break;
```

```
                                                }
                                                fflush(stdout);
                                                fflush(stdin);
                            }

                    }

                    close(s);
                  }
    return 0;
}
```

Receiver Side:

```
/* CSD 304 Computer Networks, Fall 2016
          Sender
          Team: Prasanna Natarajan
                    Siddhart Mitra
                    Sridhar Renga Ramanujam
                    Vedant Chakravarthy
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <time.h>

#define MC_PORT 7779
#define BUF_SIZE 64480
#define TCP_PORT 5432
#define MAX_PENDING 10
#define MAX_NUM_OF_STATIONS 2
#define MAX_FILES_IN_A_STATION 1
#define DEBUG 0


/*Global Variables*/
          int tempSizeHolder = 0;

          typedef struct station_info {
                    uint8_t station_number;
                    uint8_t station_name_size;
                    char * station_name;
                    uint32_t multicast_address;
                    uint16_t data_port;
                    uint16_t info_port;
                    uint32_t bit_rate;
          }
          station_info_t;


          void serialize_station_info_t(station_info_t *buf,char *x){
         char buffer[2048];    //ARBITRARY SIZE INITIALIZATION
```

```
        if(DEBUG == 1)
                puts("Inside serialize_station_info_t");
        memcpy(buffer           ,&(buf->station_number)   ,1);    //1

        memcpy(buffer+1         ,&(buf->station_name_size)  ,1);  //2

        memcpy(buffer+2         ,(buf->station_name)    ,buf->station_name_size); //3

        uint32_t multicast_address = htons(buf->multicast_address);   //4
        memcpy(buffer+2+buf->station_name_size      ,&multicast_address ,4);  //5

        if(DEBUG == 1)
                puts("5");

        uint16_t data_port = htons(buf->data_port);                  //6
        memcpy(buffer+2+buf->station_name_size+4    ,&data_port     ,2);  //7
        if(DEBUG == 1)
                puts("6");
        uint16_t info_port = htons(buf->info_port);                  //8
        memcpy(buffer+2+buf->station_name_size+4+2    ,&info_port     ,2);//9
        if(DEBUG == 1)
                puts("9");
        uint32_t bit_rate = htons(buf->bit_rate);                    //10
        memcpy(buffer+2+buf->station_name_size+4+2+2  ,&bit_rate      ,4);//11
        if(DEBUG == 1)
                puts("11");

        tempSizeHolder = 2+buf->station_name_size+4+2+2+4;           //12
        if(DEBUG == 1)
                puts("12");

        memcpy(x,buffer,tempSizeHolder);
        if(DEBUG == 1)
                puts("End of serialize_station_info_t");
}

typedef struct site_info {
        uint8_t type;
        uint8_t site_name_size;
        char * site_name;
        uint8_t site_desc_size;
        char * site_desc;
        uint8_t station_count;
        station_info_t * station_list;
}
site_info_t;


void serialize_site_info_t(site_info_t *buf,char *x){
        char buffer[1024];    //ARBITRARY SIZE INITIALIZATION

        memcpy(buffer           ,&(buf->type)                        ,1);

        memcpy(buffer+1         ,&(buf->site_name_size)          ,1);

        memcpy(buffer+2         ,(buf->site_name)                                         ,buf->site_name_size);

        memcpy(buffer+2+buf->site_name_size   ,&(buf->site_desc_size) ,1);
        memcpy(buffer+2+buf->site_name_size+1 ,(buf->site_desc)         ,buf->site_desc_size);

        memcpy(buffer+2+buf->site_name_size+1+buf->site_desc_size,&(buf->station_count),1);

        char* otherStruct;
        otherStruct = (char*)malloc(20000);

        if(DEBUG == 1)
                puts("In the middle of serialize_site_info_t");
// FATAL ERROR
        int sumSizes = 0;
        int tempx;
        for(tempx = 0; tempx < MAX_NUM_OF_STATIONS; tempx++){ // station count has been replaced by
MAX_NUM_Of_STATIONS
                if(DEBUG == 1)
                        puts("Before serialize_site_info_t inside serialize_site_info_t");
                serialize_station_info_t(&(buf->station_list[tempx]),otherStruct);
                if(DEBUG == 1)
                        puts("After serialize_site_info_t inside serialize_site_info_t");
                sumSizes+=tempSizeHolder;
                memcpy(buffer+2+buf->site_name_size+1+buf->site_desc_size+1+sumSizes,otherStruct,tempSizeHolder);
        }
        if(DEBUG == 1)
                puts("End of serialize_site_info_t");

        memcpy(x,buffer,sizeof(buffer));
}

typedef struct station_not_found {
        uint8_t type;
        uint8_t station_number;
}
station_not_found_t;

void serialize_station_not_found_t(station_not_found_t *buf,char *x){
        char buffer[5];   //ARBITRARY SIZE INITIALIZATION

        memcpy(buffer            ,&(buf->type)     ,1);

        memcpy(buffer+1          ,&(buf->station_number)   ,1);

        memcpy(x,buffer,sizeof(buffer));
}

typedef struct song_info {
        uint8_t type;
        uint8_t song_name_size;
        char* song_name;
        uint16_t remaining_time_in_sec;
        uint8_t next_song_name_size;
        char* next_song_name;
}
song_info_t;

void serialize_song_info_t(song_info_t *buf,char *x){
        char buffer[1024];    //ARBITRARY SIZE INITIALIZATION

        memcpy(buffer            ,&(buf->type)     ,1);
        memcpy(buffer+1         ,&(buf->song_name_size)   ,1);

        memcpy(buffer+2         ,(buf->song_name)     ,buf->song_name_size);

        uint16_t remaining_time_in_sec = htons(buf->remaining_time_in_sec);
        memcpy(buffer+2+buf->song_name_size   ,&remaining_time_in_sec   ,2);

        memcpy(buffer+2+buf->song_name_size+2 ,&(buf->next_song_name_size),1);

        memcpy(buffer+2+buf->song_name_size+2+1 ,(buf->next_song_name)    ,buf->next_song_name_size);

        memcpy(x,buffer,sizeof(buffer));
}

/*Global Variables*/
char* all_mcast_addresses[MAX_FILES_IN_A_STATION];

int main(int argc, char * argv[]) {

        int s, s_mcast ,new_s; /* socket descriptor */
        struct sockaddr_in sin; /* socket struct for TCP*/
        struct sockaddr_in sin_mcast; /*socket struct for multi cast*/
        char buf[BUF_SIZE];
        int len;
        pid_t pid;
        /* Multicast specific */
        char * mcast_addr; /* multicast address */
        char * team_multicast_address = "239.192.29.10";
        struct timespec tim,tim2;
        tim.tv_sec = 1;
        tim.tv_nsec = 0;
  int length = 0;
  int looper = 0;

        /* Add code to take port number from user */
        if (argc == 2) {
                mcast_addr = argv[1];

        } else {
                fprintf(stderr, "usage: sender multicast_address\n");
                exit(1);
        }

        struct hostent *hp;
        hp = gethostbyname(mcast_addr);
        if(!hp){
                perror("simplex-talk: Unknown Host");
                exit(1);
        }
        else{
                printf("Server's remote host : %s\n",mcast_addr);
        }
        /*Filling up the mcast_addresses*/
        int j;
        for (j = 0; j < MAX_NUM_OF_STATIONS; ++j)
        {
                all_mcast_addresses[j] = malloc(sizeof(char)*15);
                char ip[3];
                sprintf(ip,"%d",10+j);
```

```
            strcat(all_mcast_addresses[j],"239.192.29.");
            strcat(all_mcast_addresses[j],ip);
            puts(all_mcast_addresses[j]);
        }

        pid = fork();
        if (pid == 0) {

            /* Send multicast messages */
            /* Warning: This implementation sends strings ONLY */
            /* You need to change it for sending A/V files */
            memset(buf, 0, sizeof(buf));
            char str[INET_ADDRSTRLEN];
            /* setup passive open */
            /* build address data structure */
            memset((char * ) & sin, 0, sizeof(sin));
            sin.sin_family = AF_INET;
            sin.sin_addr.s_addr = inet_addr(mcast_addr);    //This is correct, naming is misleading
            sin.sin_port = htons(TCP_PORT);

            if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
                perror("simplex-talk: socket");
                exit(1);
            }

            inet_ntop(AF_INET, & (sin.sin_addr), str, INET_ADDRSTRLEN);
            printf("Server is using address %s and port %d.\n", str, TCP_PORT);
    /*
            int yes=1;
            //char yes='1'; // Solaris people use this

            // lose the pesky "Address already in use" error message
            if (setsockopt(s,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof yes) == -1) {
                perror("setsockopt");
                exit(1);
            } */


            int yes=1;
            //char yes='1'; // Solaris people use this

            // lose the pesky "Address already in use" error message
            if (setsockopt(s,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof (yes)) == -1) {
                perror("setsockopt");
                exit(1);
            }

            if ((bind(s, (struct sockaddr * ) &sin, sizeof(sin))) < 0) {
                perror("simplex-talk: bind");
                exit(1);
            } else
            printf("Server bind done.\n");



            listen(s, MAX_PENDING);


            if ((new_s = accept(s, (struct sockaddr*) &sin, &len)) < 0) {
```

```
            perror("simplex-talk: accept");
            exit(1);
        }
        else{
            puts("inside else");
        }

        puts("after listen");
        while (1) {
            if(DEBUG == 1)
                puts("inside while");

                /*
                        check station list and then send current station list and break

                        3) fill the struct
                        4) serialize the struct
                        5) send it to the reciever using syntax
send(new_s,buff,strlen(buff)+1,0);

                */
                        if(DEBUG == 1)
                            puts("Begin things");
                        char* station_list_array[MAX_NUM_OF_STATIONS];
                        int j;
                        for(j=0;j<MAX_NUM_OF_STATIONS;j++){
                            station_list_array[j] = malloc(sizeof(char)*20);
                            strcat(station_list_array[j],"channel");
                            char j_char[33];
                            sprintf(j_char,"%d",j);
                            strcat(station_list_array[j],j_char);
                        }
                        if(DEBUG == 1)
                            puts("Intermediate places");
                        site_info_t *first_send = malloc(sizeof(first_send)*sizeof(char));
                        first_send->type = 10;
                        first_send->site_name_size= 5;
                        first_send->site_name = malloc(sizeof(char)*first_send-
>site_name_size);
                        strcpy(first_send->site_name, "Mitra");
                        first_send->site_desc_size = 20;
                        first_send->site_desc = malloc(sizeof(char)*first_send-
>site_desc_size);
                        strcpy(first_send->site_desc,"Mitra's funny videos");
                        first_send->station_count = MAX_NUM_OF_STATIONS;

                        station_info_t stations[MAX_NUM_OF_STATIONS];

                        // puts("Populate station list");
                        for(j=0;j<MAX_NUM_OF_STATIONS;j++){
                // stations[j] = (station_info_t)malloc(sizeof(station_info_t)*sizeof(char));
                            stations[j].station_number = j;
                            stations[j].station_name_size =
strlen(station_list_array[j]);
                            stations[j].station_name =
malloc(sizeof(char)*strlen(station_list_array[j]));
                            stations[j].station_name = station_list_array[j];

                            stations[j].multicast_address = j;
                            stations[j].data_port = 5555+j;
                            stations[j].info_port = 6666+j;
                            stations[j].bit_rate = 126;

                        }
                        if(DEBUG == 1)
                            puts("Final spaces");
                        fflush(stdout);
                        first_send->station_list = stations;
                        if(DEBUG == 1)
                            puts("Before malloc");
                        char* x = malloc(sizeof(char)*20000);
                        if(DEBUG == 1)
                            puts("after malloc");
                        serialize_site_info_t(first_send,x);
                        if(DEBUG == 1){
                            puts("The serialized output is");
                            puts(x);
                            puts("type : ");
                            printf("%d",x[0]);
                            puts("site_name_size : ");
                            printf("%d",x[1]);
                        }
                        if(DEBUG == 1)
                            puts("Sending thing");
                        send(new_s,x,strlen(x)+1,0);
                        if(DEBUG == 1)
                            puts("Successful send");
                    }
                } else {
        // while(1)
                    if(DEBUG == 1)
                        puts("inside first else");
                    int station_number;
                    pid_t pid1;
                    int xx;
                    for(station_number =
0;station_number<MAX_NUM_OF_STATIONS;station_number++){
                        pid1 = fork();
                        if(pid1==0){

                            if(DEBUG == 1)
                                puts("inside fork");
                            if(DEBUG == 1)
                                printf("fork number =
%d",station_number);

                            fflush(stdout);


                            if ((s_mcast = socket(PF_INET, SOCK_DGRAM, 0))
< 0) {

                                perror("server: socket");
                                exit(1);
                            }

      /* build address data structure */
                            memset((char * ) & sin, 0, sizeof(sin_mcast));
                            sin_mcast.sin_family = AF_INET;
      sin_mcast.sin_addr.s_addr = inet_addr(all_mcast_addresses[station_number]); //mcast_addr
      sin_mcast.sin_port = htons(MC_PORT);
```

```
printf("the all_mcast_addresses[station_number] = %s",all_mcast_addresses[station_number]);
fflush(stdout);

char* filenames_in_a_station[MAX_FILES_IN_A_STATION]; // each string will contain all the names of that
particular staion

int i;

char command_ls[20] = "ls channel";
char i_char[33];
sprintf(i_char,"%d/",station_number);
strcat(command_ls,i_char);
//printf("%s",command_ls);
strcat(command_ls," | grep .mp3");
//printf("%s\n",command_ls);
FILE * temp_fp = popen(command_ls,"r");

for (i = 0; i < MAX_FILES_IN_A_STATION; ++i)
{
    filenames_in_a_station[i] = malloc(sizeof(char)*100);

}
char * temp_filename = malloc(sizeof(char)*20);
char * channel_name = malloc(sizeof(char)*20);
sprintf(channel_name,"./channel%d/",station_number);
i=0;
while(fgets(temp_filename,20,temp_fp)!=NULL){
    if(temp_filename[strlen(temp_filename)-1]=='\n'){
        temp_filename[strlen(temp_filename)-1]='\0';
        if(DEBUG == 1)
            puts("inside if");
    }
    strcat(channel_name,temp_filename);
    strcat(filenames_in_a_station[i],channel_name);
    puts(filenames_in_a_station[i]);
    i++;
}
char buffer_to_send_multcast[BUF_SIZE];
FILE *fp_array[MAX_FILES_IN_A_STATION];
FILE *fp_write = fopen("temp.mp3","w");
puts("just before sending multicast");
for(i=0;i< MAX_FILES_IN_A_STATION;i++){
    printf("inside for in sending multicast = %d",i);
    fflush(stdout);
    fp_array[i] = fopen(filenames_in_a_station[i],"r");
    puts("after creating file pointer");
    if(fp_array[i] == NULL){
        printf("\n%s\n",filenames_in_a_station[i]);
        fflush(stdout);
        puts("no file");
        break;
    }

    fflush(stdout);
    fflush(stdin);
    fseek(fp_array[i], 0, SEEK_END);
    length = ftell(fp_array[i]);
    fseek(fp_array[i], 0, 0);
    fflush(fp_array[i]);
```

```
                 for(looper = 0;looper<length;looper++){
                    buffer_to_send_multcast[looper%BUF_SIZE] = fgetc(fp_array[i]);
                   if((looper%BUF_SIZE == BUF_SIZE-1)){
                     fwrite(buffer_to_send_multcast,BUF_SIZE,1,fp_write);
                    if ((len = sendto(s_mcast, buffer_to_send_multcast, BUF_SIZE, 0,(struct sockaddr *)&sin_mcast,
sizeof(sin_mcast))) == -1) {
                     perror("sender: sendto");
                     exit(1);
                    }
                    puts("Printed");

                    nanosleep(&tim,&tim2);
                   }

                 }
                 i=(i+1)%MAX_FILES_IN_A_STATION;

               }



                                       }else{
                                                 // int status;
                                                 // pid1 = wait(&status);
                                       }
                             }

                             pid = wait();
                   }

                   close(s);
                   return 0;
          }
```