SAN JOSÉ STATE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

SPRING 2016

CMPE207

NETWORK PROGRAMING AND APPLICATIONS

PROJECT REPORT

ON

Multi-Media Streaming

Under Guidance
Of
Dr. Younghee Park

Project team #13
Project team Members:
Viral Baitule (SJSU ID: 010710960 Class ID:02)
Abhinav Prasad (SJSU ID: 010690459 Class ID:54)
Aniket Tayade (SJSU ID: 010720034 Class ID:74)
Nimish Unde (SJSU ID: 010698493 Class ID:76)

1. Abstract

This project is designed on the Client-Server implementation, where the clients make requests to the server deployed on the Amazon Web Services for streaming files. The server responds to each requests concurrently. The server is deployed on the Amazon web service. Amazon provides free deployment of server and offers various flavor of operating system images pre-installed in them. An amazon EC2 server can be added with additional storage when required and AWS also provides scheduled backup options for all the servers. The clients are given choices on the types of file it wants to access. Upon specific request from the clients, the server executes and provides the requested file to the clients. The transmission of the file is done through the Transmission Control Protocol. Both the Client and the Server are implemented using C socket based programing in Linux environment.

2. Table of Contents

3. Application Overview

The “Multimedia streaming” project offers the clients an independence of accessing files from any where over the internet. The server responds to each client concurrently and fetch each with their choice of file.

This project is implemented with the concepts gained during the course on Network programing and applications. The Client Server communication is implemented using TCP. The server is featured with concurrency using multithreading. The server listens to each of the clients on the master thread and creates new slave threads to serve new requests from each client.

The project is implemented using a fat server and a thin client architecture where majority of processing is handled by the server. The system architecture consists of one server and multiple clients. The server provides redundancy, reliability and backup services. Multiple clients can access server and make their desired request at the same time. Multiple clients can even access a single file at the same time.

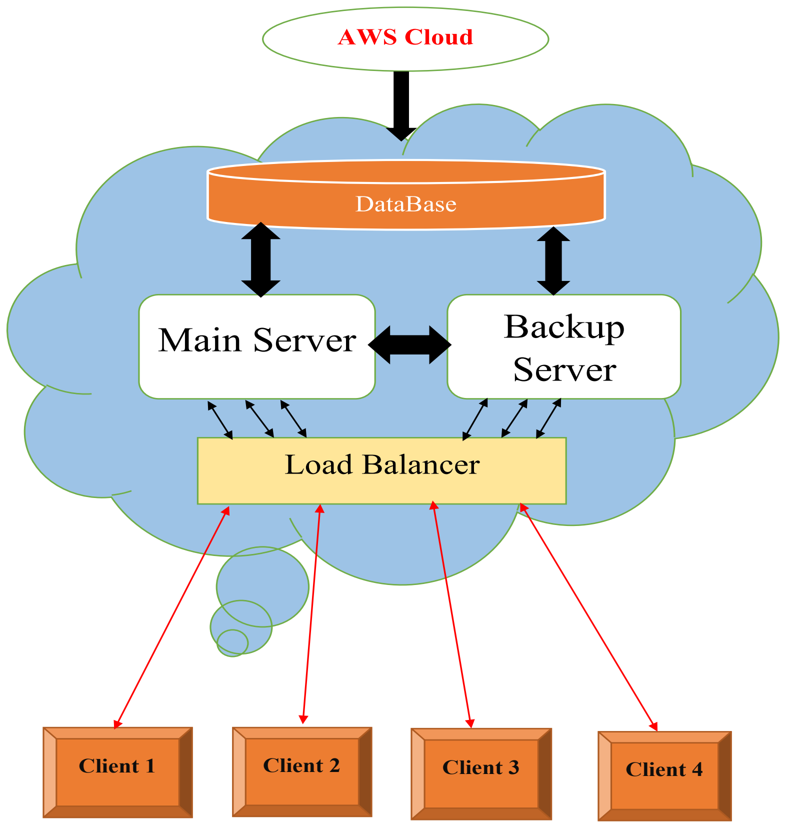
The clients of this server can be located anywhere and access the server irrespective of their geographical location. This service is provided by deploying the server on the Amazon web services.

3.1 Amazon Web Servers

Amazon Web Services (AWS), is a collection of cloud computing services that makes up the on-demand computing platform offered by Amazon.com.
The most central and best-known of this services Amazon Elastic Compute Cloud, also known as "EC2". [1]

Amazon Elastic Compute Cloud (EC2) forms a central part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), by allowing users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the hour for active servers - hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy. [2]

4. Network Model



4.1 Architecture

The Architecture consist of one main Server and one Backup Server connected to the Data Base.

A Load Balancer distributes the Number of clients accessing the servers at the same time.

5. Programing Concept

5.1 Network Protocols

5.1.1 Application Level

The System Architecture consists of a main server, a backup server, a Load balancer and multiple clients. The server constitutes the major component of the application. The server is implemented for fast processing and the client is made as simple as possible to make the end user easy access. The server is designed to handle as many clients as possible providing a high level of transparency. This makes the client believe that the server is providing stand-alone service only to the requesting clients at the specific time.

5.1.2 Thin client

The thin client architecture is implemented to make the application very simple and to provide an easy access. The clients in this project is given limited functionality, limiting to sending desired input to the server, reading the server responses and displaying it to the user.

5.1.3 Fat Server

All the processing for this application is done at the server. The multithreading to handle concurrency among clients, handling the data base and proving files on requests, are the main features of the server.

5.1.2 Transport Level

The communication between the server and the clients are handled by the Transmission Control Protocol (TCP). TCP provides reliable, ordered, and error-checked delivery of a stream between applications running on server communicating over an IP network. TCP is a connection oriented point to point protocol where it enables two hosts to connect each other through 3- way handshaking. The data is sent/received as streams of data.

The Transmission Control Protocol provides a communication at an intermediate level between an application program and the Internet Protocol. It provides host-to-host connectivity at the Transport Layer of the Internet model. An application does not need to know the particular mechanisms for sending data via a link to another host, such as the required packet fragmentation on the transmission medium. At the transport layer, the protocol handles all handshaking and transmission details and presents an abstraction of the network connection to the application.

5.2 Concurrency

This is a feature enabled in the server to be able to serve multiple clients at a same time. Concurrency is enabled using multithreading. Whenever a client accesses the server a new thread will be created to handle the request. So, all the requests from new clients will be handled by new threads enabling the main thread to accept further accept new connection. This allows for parallel execution of the concurrent units, which significantly improve the overall speed of execution.

Concurrency in the server application is implemented with POSIX Threads, referred to as Pthreads. It allows server application to control multiple different flows from client applications that overlap in time. Each flow of work is referred to as a thread, and creation and control over these flows is achieved by making calls to the POSIX Threads API. API are used for thread creation and synchronization. [3]

5.3 Load Balancing

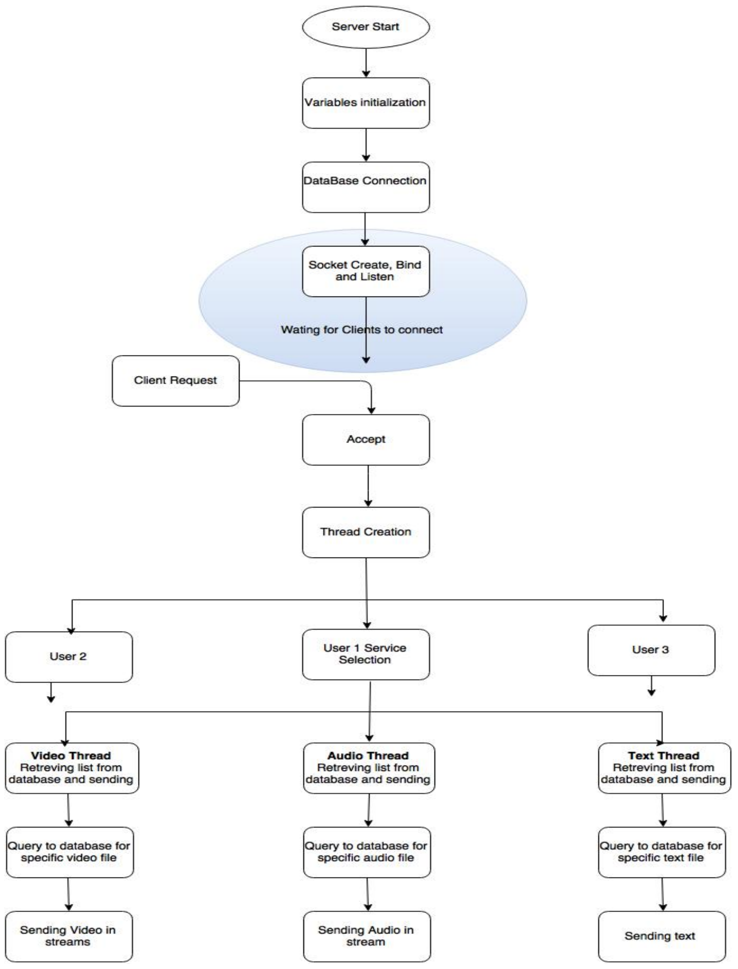
In computing, load balancing distributes workloads across multiple computing resources. In this project, the Load Balancer is connected to the main server and the backup server. Load from the clients accessing the database gets distributed among the main server and the backup server.

Using multiple components with load balancing increases the reliability and availability through redundancy.

Load Balancing is implemented in the server application with MySQL. MySQL is an open source relational database management system.

6. Application Flow

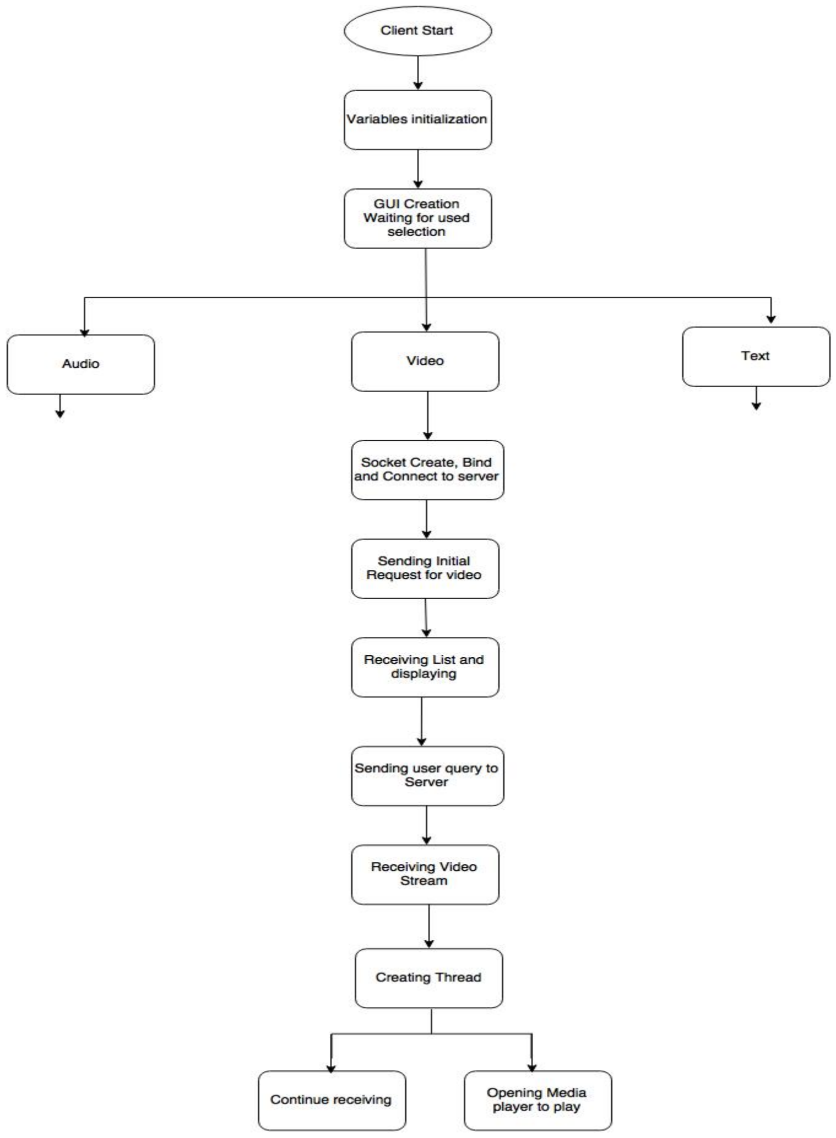
6.1 Server Application


Server Start
Variables initialization
DataBase Connection
Socket Create, Bind and Listen
Wating for Clients to connect
Client Request
Accept
Thread Creation
User 2
User 1 Service Selection
User 3
Video Thread Retreving list from database and sending
Audio Thread Retreving list from database and sending
Text Thread Retreving list from database and sending
Query to database for specific video file
Query to database for specific audio file
Query to database for specific text file
Sending Video in streams
Sending Audio in stream
Sending text

The Server application starts with initializing the variables, creating the Database. The server creates a socket and binds to an IP address. The server waits for a connection from any client in the listening mode.

Once a client request arrives, the server accepts the connection. Threads are created to implement concurrency. The clients make a choice among the three options of Video, Audio or Text file. Upon specific request from each connected client, the server process the request successfully.

6.2 Client Application


Client Start
Variables initialization
GUI Creation Waiting for used selection
Audio
Video
Text
Socket Create, Bind and Connect to server
Sending Initial Request for video
Receiving List and displaying
Sending user query to Server
Receiving Video Stream
Creating Thread
Continue receiving
Opening Media player to play

The client application starts with initialization of the variables. GUI for the user interface. This gives the user a choice to select either a video, audio or text file. The client creates a socket, binds to a IP address and connects to the server application. The client makes a request of any of the file from the three options and receives list of files present. The client can select any of the file from the given list and start viewing. A new thread is created to continue receiving the file and start viewing simultaneously.

7. Source Code

7.1 Server Code Snippets

```
static char *host="localhost";
static char *user="root";
static char * pass="htc";
static char * dbname="warehouse";
static unsigned int port=NULL;
static char * unix_socket=NULL;
static unsigned int flag=0;
static int initial_input[5];
MYSQL_RES *res;
MYSQL_ROW row;
MYSQL_ROW row1;
MYSQL * conn;
```

The parameters required for connecting to the database in MySQL. In the above code, the parameters are defined to connect to the database.

```
conn=mysql_init(NULL);
if(!(mysql_real_connect(conn,host,user,pass,dbname,port,unix_socket,flag))).
```

This function in main is used for connecting to the database. This function basically initializes the various global variables required by MySQL. It further calls mysql_thread_init(). Then in the if () condition checks whether connection to the database is established or not.
The function mysql_real_connect() tries to connect to the MySQL database that is running on the server. It is important that this function works successfully before executing any API functions to access the MySQL database.

```
mysql_query(conn, "SELECT * from video");
res=mysql_store_result(conn);
```

mysql_query() is used to query the database. This function transmits unique query to the database on the server which is active. This function returns zero if successful else a non zero value if there is any error.

num_fields = mysql_num_fields(res);

The mysql_num_fields() function is used to determine the number of columns in the database. It returns an unsigned integer value which indicates the number of columns.

while ((row = mysql_fetch_row(res)) != NULL)

The function mysql_fetch_row() retrieves next row. It returns NULL if the row does not exist.
res=mysql_store_result(conn);

The function mysql_store_result() reads the entire result of a query to the client, allocates a MYSQL_RES structure, and places the result into this structure.

void sendfile(char resultstr[], int sockdes)

The sendfile function is used to send the multli-media file requested by the user. This function can transfer any file requested by the user. The file can be of any format. It make a video, audio or a text file that a client has requested.

FILE *fp = fopen(resultstr,"rb");

The fopen() is used to open a file which is pointed by the pointer passed as an parameter. It also requires another parameter which mentions the specific mode in which the file should open. In the above function the parameter resultstr points out to the file name while the rb mentions the mode (to update that is to read or write). It returns a pointer to the file.

int nread = fread(buff,1,256,fp);

The fread() performs the function of reading the data from a specific stream pointed (fp) and then copies it into an array pointed by buff. It returns the number of elements read.

if (feof(fp))

The function feof() checks whether a given stream indicated by fp has reached its end or not. It returns a non zero value when the End-Of-File indicator for the given stream is set else it returns zero.

7.2 Client Code Snippets

Main Function

GtkWidget *window;
GtkWidget *fixed;
GtkWidget *btn1;
GtkWidget *btn2;
GtkWidget *btn3;

These are the variables required for creating GUI in GTK+. GtkWidget is the base class that is used to manage the widget. It is used by all the widgets.

gtk_init(&argc, &argv);

This function used for initializing the GTK+. It should be called before any other GTK+ functions for creating GUI.

window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

The gtk_window_new() is used for creating a new GTK window. GTK_WINDOW_TOPLEVEL defines the window type. This means that the window will contain all other widgets in it.

gtk_window_set_title(GTK_WINDOW(window), "GRAPHICAL USER INTERFACE");
gtk_window_set_default_size(GTK_WINDOW(window), 300, 30);
gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);

gtk_window_set_title() is used to define the the title of the window. gtk_window_set_default_size() is used to define the basic size of the window. Here we are keeping the size of the window as 300X300.

gtk_window_set_position() is used to define the position of the window that will appear on the screen. By passing the parameter GTK_WIN_POS_CENTER we are setting the window to appear at the center of the screen.

fixed = gtk_fixed_new();
gtk_container_add(GTK_CONTAINER(window), fixed);

gtk_fixed_new() is used to create a container that has a fixed position and sizes of all the children containers. gtk_container_add() is used to add a container.

btn1 = gtk_button_new_with_label("VIDEO FILE");

gtk_button_new_with_label() is used for generating GTK button with the label. In this case the label is VIDEO FILE. Thus it creates a button with label VIDEO FILE.

gtk_fixed_put(GTK_FIXED(fixed), btn1, 0, 0);
gtk_widget_set_size_request(btn1, 300, 100);
g_signal_connect(G_OBJECT(btn1), "clicked", G_CALLBACK(video), NULL);

gtk_fixed_put() is used for adding a GTK fixed container. Gtk_widget_set_size_request is used for defining the smallest size of the button. g_signal_connect() is used for connecting the object to the video function. This video function is called when the button is clicked.

gtk_widget_show_all(window);
g_signal_connect(window, "destroy",G_CALLBACK(gtk_main_quit), NULL);
gtk_main();

The above code will display the entire window. Further whenever we click on the close button in the GUI, it will completely destroy the window by calling the gtk_main_quit function.

7.3 Video Function

This function is called when we click on the video button in GUI. This function establishes a TCP connection with the server deployed on AWS. It sends a request to the server indicating that the server should send the list of the video files in its database. After the server has transmitted the list to the client, the client can choose the video file which it wants to stream. After sending the specific video the client will receive now the packets in the size of 256 bytes continuously. Also it will start playing the video as soon as it receives the packet in VLC media player.

```
void *Thread_connection_handler(void * filename)
{
	char cmdbuf[256];
	snprintf(cmdbuf, sizeof(cmdbuf), "vlc --rate=1.00 --playlist-autostart -vvv %s --tcp-
caching=10", (char*)filename);
	system(cmdbuf);

	pthread_exit(NULL);
}
```

We are calling the thread to handle the packets that we are receiving at the client side. This thread is used to open VLC media file using the system call. We are using the character array cmdbuf[] to store the string required to open VLC media player using system() call.

Similarly, the **Audio and Text functio**n works.

7.4 Text Function

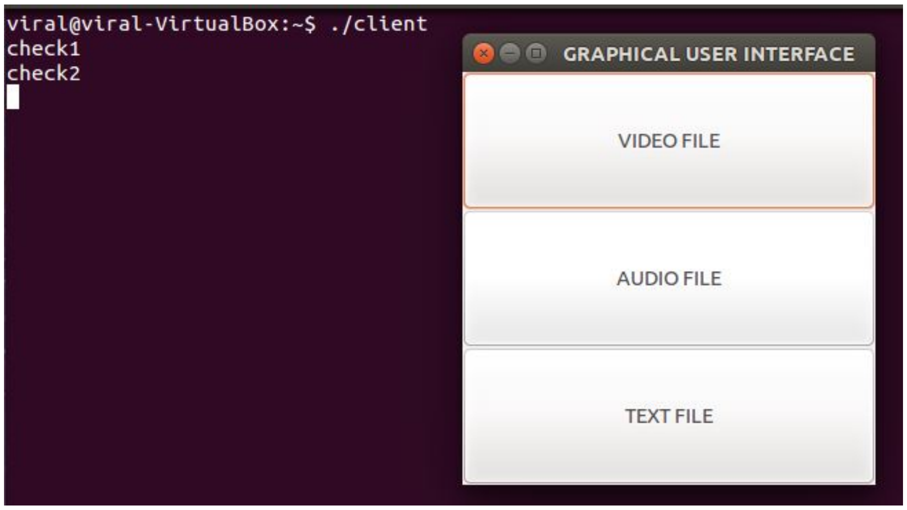
```
char cmdbuf[256];
snprintf(cmdbuf, sizeof(cmdbuf), "xdg-open %s ", (char*)filename);
system(cmdbuf);
```

The above code is used to open the text file that is received by the client. We are using xdg-open command for this purpose. This command is stored in cmdbuf[] and then used in system().

8. Application Execution

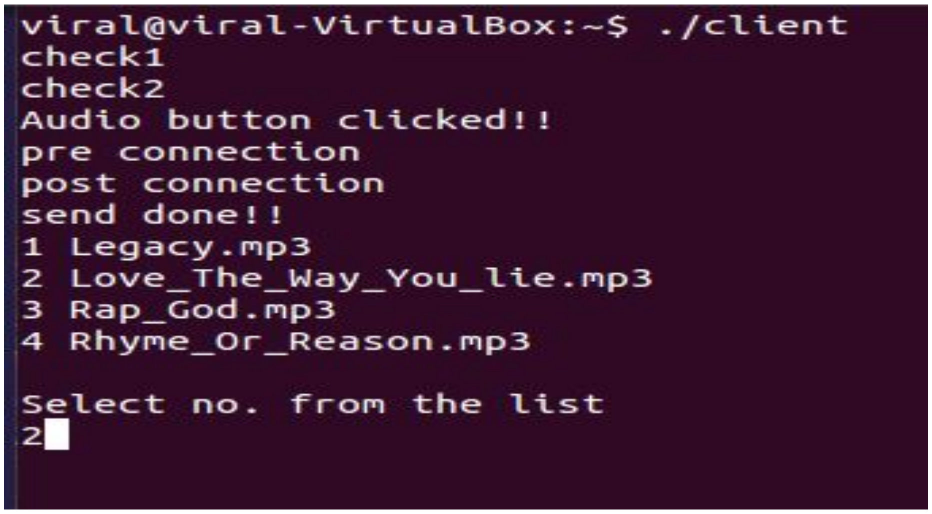
6.1 Application Interface

The client application can be executed from anywhere over the internet with the access to the Internet Protocol (IP) Address of the continuous running server.



```
viral@viral-VirtualBox:~$ ./client
check1
check2
```

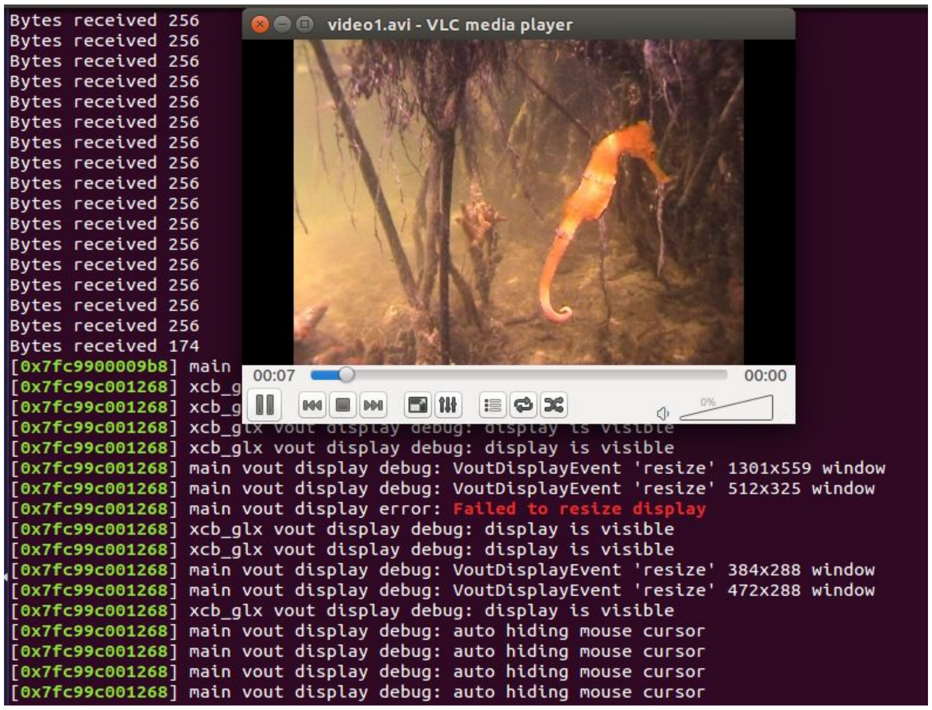
The above snapshot is taken from one of the client accessing the server. This is the first Graphical User Interface the user access on executing the client application.



```
viral@viral-VirtualBox:~$ ./client
check1
check2
Audio button clicked!!
pre connection
post connection
send done!!
1 Legacy.mp3
2 Love_The_Way_You_lie.mp3
3 Rap_God.mp3
4 Rhyme_Or_Reason.mp3

Select no. from the list
2
```

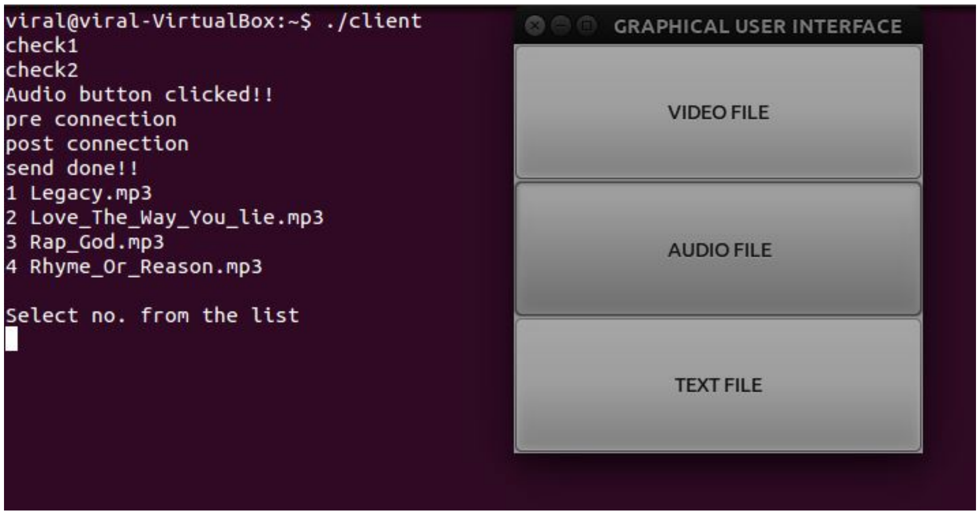
After selection among the Video, Audio or Text File, the client can make a choice among the three options.

Here the client accesses video file and makes a selection of 2nd   video to view.

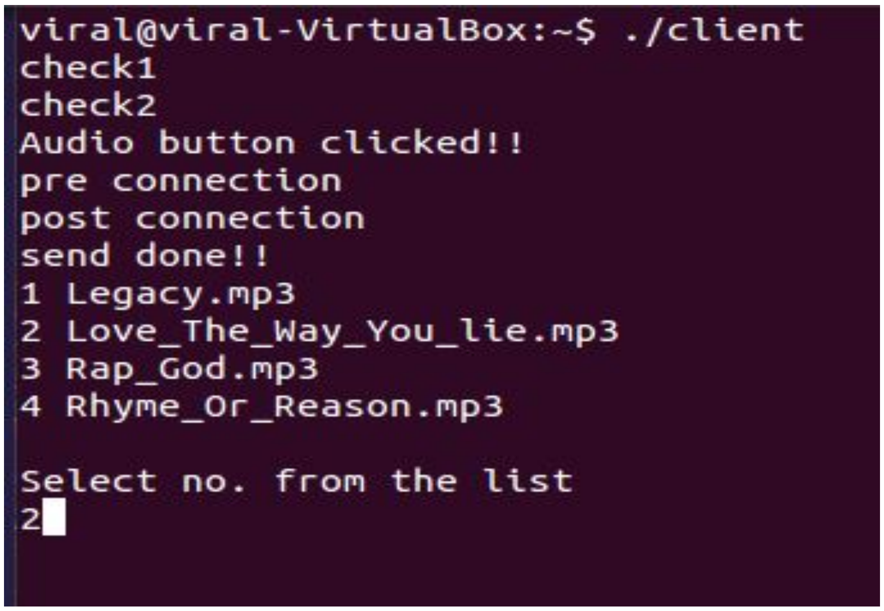


```
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 174
[0x7fc9900009b8] main
[0x7fc99c001268] xcb_g
[0x7fc99c001268] xcb_g
[0x7fc99c001268] xcb_glx vout display debug: display is visible
[0x7fc99c001268] xcb_glx vout display debug: display is visible
[0x7fc99c001268] main vout display debug: VoutDisplayEvent 'resize' 1301x559 window
[0x7fc99c001268] main vout display debug: VoutDisplayEvent 'resize' 512x325 window
[0x7fc99c001268] main vout display error: Failed to resize display
[0x7fc99c001268] xcb_glx vout display debug: display is visible
[0x7fc99c001268] xcb_glx vout display debug: display is visible
[0x7fc99c001268] main vout display debug: VoutDisplayEvent 'resize' 384x288 window
[0x7fc99c001268] main vout display debug: VoutDisplayEvent 'resize' 472x288 window
[0x7fc99c001268] xcb_glx vout display debug: display is visible
[0x7fc99c001268] main vout display debug: auto hiding mouse cursor
[0x7fc99c001268] main vout display debug: auto hiding mouse cursor
[0x7fc99c001268] main vout display debug: auto hiding mouse cursor
[0x7fc99c001268] main vout display debug: auto hiding mouse cursor
```

Upon section of the desired file, the client application is able to retrieve the video file and play the same in the VLC video player.

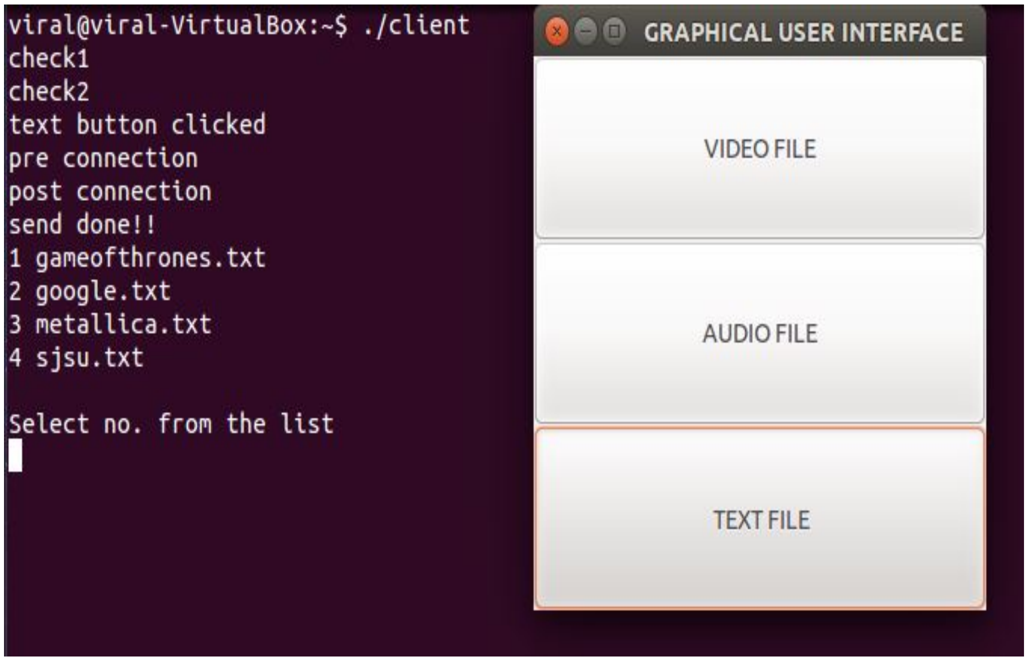


```
viral@viral-VirtualBox:~$ ./client
check1
check2
Audio button clicked!!
pre connection
post connection
send done!!
1 Legacy.mp3
2 Love_The_Way_You_lie.mp3
3 Rap_God.mp3
4 Rhyme_Or_Reason.mp3

Select no. from the list
```



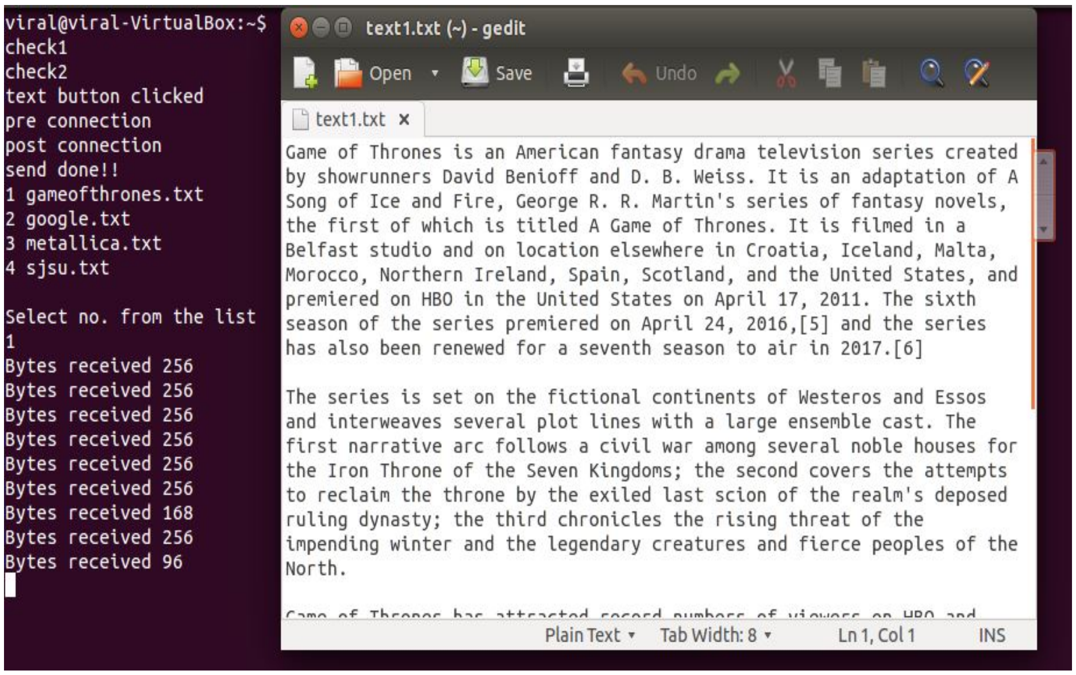
```
viral@viral-VirtualBox:~$ ./client
check1
check2
Audio button clicked!!
pre connection
post connection
send done!!
1 Legacy.mp3
2 Love_The_Way_You_lie.mp3
3 Rap_God.mp3
4 Rhyme_Or_Reason.mp3

Select no. from the list
2
```

The above screenshot shows the client's selection of an audio file. The list of all audio files can be viewed from the database.



```
viral@viral-VirtualBox:~$ ./client
check1
check2
text button clicked
pre connection
post connection
send done!!
1 gameofthrones.txt
2 google.txt
3 metallica.txt
4 sjsu.txt

Select no. from the list
```

The client application accesses the audio file from the server.

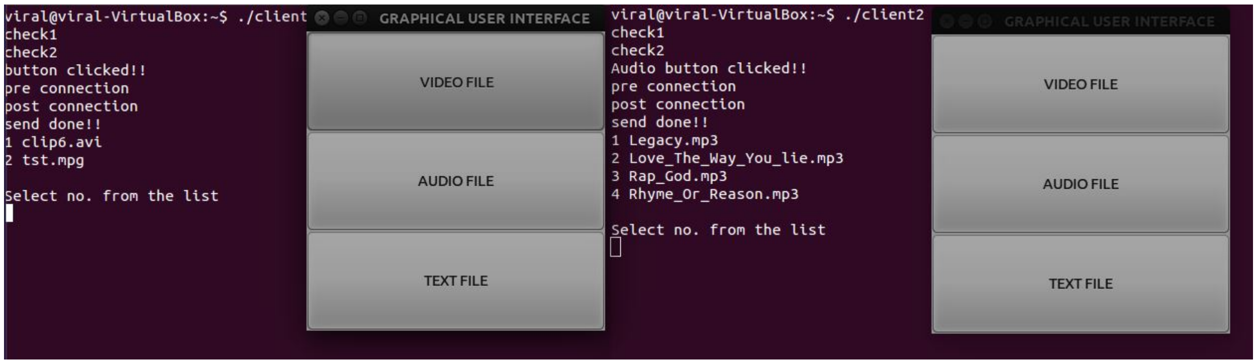


```
viral@viral-VirtualBox:~$
check1
check2
text button clicked
pre connection
post connection
send done!!
1 gameofthrones.txt
2 google.txt
3 metallica.txt
4 sjsu.txt

Select no. from the list
1
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 256
Bytes received 168
Bytes received 256
Bytes received 96
```

text1.txt (~) - gedit

```
Game of Thrones is an American fantasy drama television series created
by showrunners David Benioff and D. B. Weiss. It is an adaptation of A
Song of Ice and Fire, George R. R. Martin's series of fantasy novels,
the first of which is titled A Game of Thrones. It is filmed in a
Belfast studio and on location elsewhere in Croatia, Iceland, Malta,
Morocco, Northern Ireland, Spain, Scotland, and the United States, and
premiered on HBO in the United States on April 17, 2011. The sixth
season of the series premiered on April 24, 2016,[5] and the series
has also been renewed for a seventh season to air in 2017.[6]

The series is set on the fictional continents of Westeros and Essos
and interweaves several plot lines with a large ensemble cast. The
first narrative arc follows a civil war among several noble houses for
the Iron Throne of the Seven Kingdoms; the second covers the attempts
to reclaim the throne by the exiled last scion of the realm's deposed
ruling dynasty; the third chronicles the rising threat of the
impending winter and the legendary creatures and fierce peoples of the
North.
```

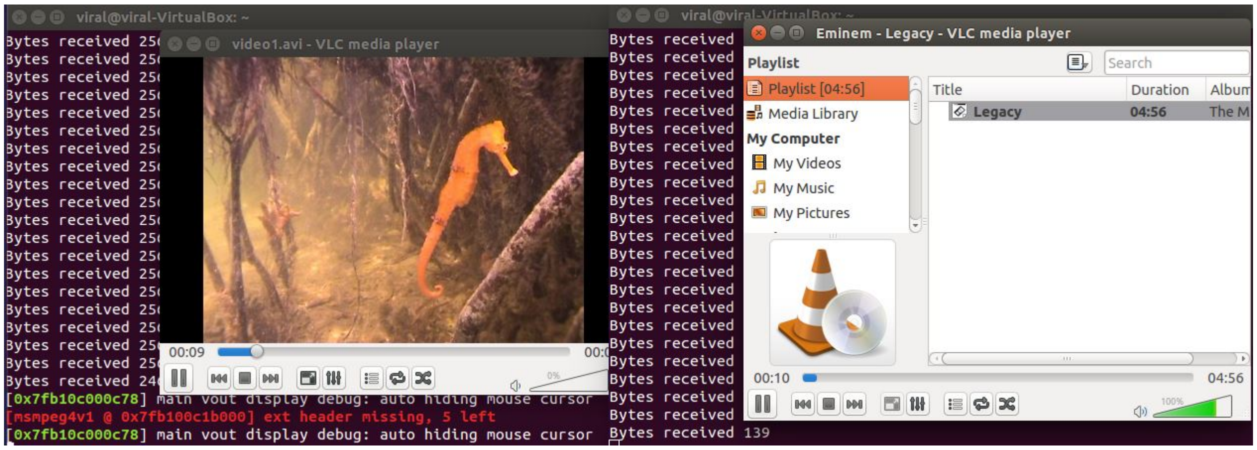
The Text file transfer from the server to the requested client.



```
viral@viral-VirtualBox:~$ ./client
check1
check2
button clicked!!
pre connection
post connection
send done!!
1 clip6.avi
2 tst.mpg

Select no. from the list
```

```
viral@viral-VirtualBox:~$ ./client2
check1
check2
Audio button clicked!!
pre connection
post connection
send done!!
1 Legacy.mp3
2 Love_The_Way_You_lie.mp3
3 Rap_God.mp3
4 Rhyme_Or_Reason.mp3

Select no. from the list
```

Concurrency handled by the server. Two clients simultaneously access the server and make request for different files.

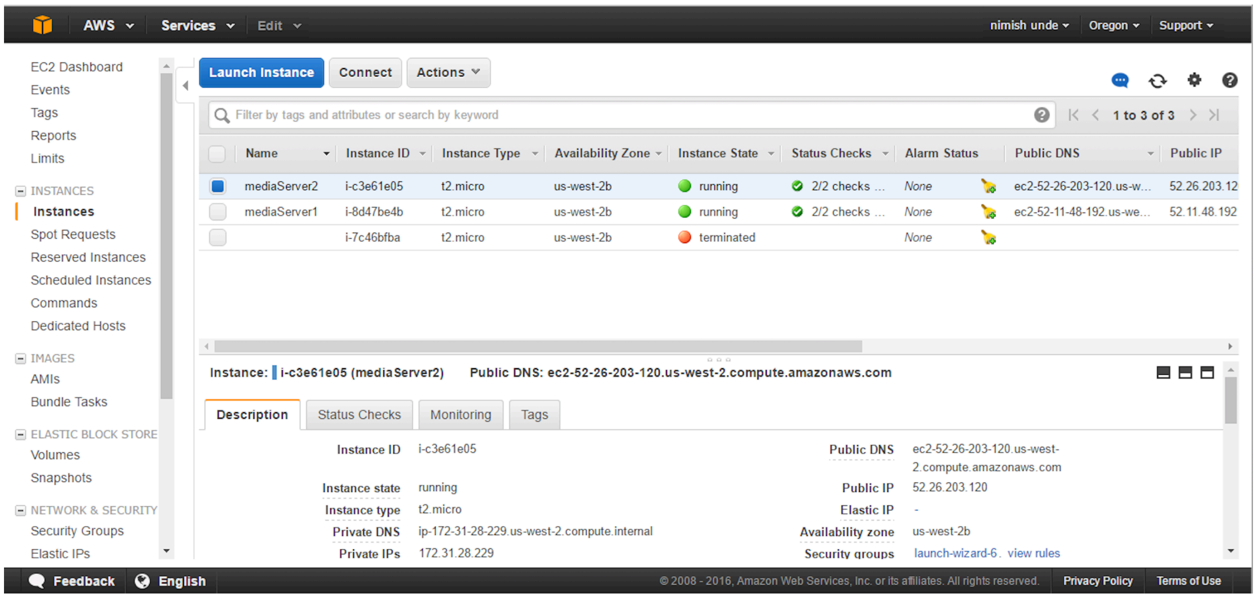


```
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 25
Bytes received 24
[0x7fb10c000c78] main vout display debug: auto hiding mouse cursor
[msmpeg4v1 @ 0x7fb100c1b000] ext header missing, 5 left
[0x7fb10c000c78] main vout display debug: auto hiding mouse cursor
```

```
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received
Bytes received 139
```

Concurrency among two clients. One client makes an access to one of the video file form the server's database while the other client makes an access to an audio file from the server's database.

The running Instance of deployed server code on the Amazon EC2 server. One main server and one Backup server.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
mediaServer2	i-c3e61e05	t2.micro	us-west-2b	running	2/2 checks ...	None	ec2-52-26-203-120.us-w...	52.26.203.12
mediaServer1	i-8d47be4b	t2.micro	us-west-2b	running	2/2 checks ...	None	ec2-52-11-48-192.us-we...	52.11.48.192
	i-7c46bfba	t2.micro	us-west-2b	terminated		None		

Instance: i-c3e61e05 (mediaServer2)   Public DNS: ec2-52-26-203-120.us-west-2.compute.amazonaws.com

Instance ID	i-c3e61e05	Public DNS	ec2-52-26-203-120.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	52.26.203.120
Instance type	t2.micro	Elastic IP	-
Private DNS	ip-172-31-28-229.us-west-2.compute.internal	Availability zone	us-west-2b
Private IPs	172.31.28.229	Security groups	launch-wizard-6. view rules

9. Test Plan

Client Application

STEPS	SCENARIO	OUTPUT
1 Running the Client program	Client program running successfully	GUI appears on the screen with buttons to select the file type for video, audio & text.
2 Clicking the button on GUI	One of the three types of file can be requested using GUI.	TCP connection is established successfully and the list of files is displayed on the terminal window.
3 Sending the number associated with the file name.	Different numbers are associated with different file names. So one of the number can be selected to request that specific file.	Starts receiving file of 256 bytes size. VLC media player will start playing the audio or video file or Text file will open automatically.

10. Conclusion

The Multi-Media Streaming is implemented successfully with the Server application deployed on the Amazon EC2 web server. The server is continuously running, accepting clients over specific IP address. Multiple clients are able to connect to the server concurrently and make request for a desired file from the list concurrently.

11. Reference

1. https://en.wikipedia.org/wiki/Amazon_Web_Services
2. https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud
3. https://en.wikipedia.org/wiki/POSIX_Threads
4. https://en.wikipedia.org/wiki/MySQL
5. http://zetcode.com/gui/gtk2/
6. http://www.gtk-server.org/gtk1/gtk/index.html
7. http://dev.mysql.com/doc/refman/5.7/en/c-api-function-overview.html