University of Dublin

TRINITY COLLEGE

Better way to AirPlay

Vincent BODIOU
B.A.(Mod.) Computer Science
Final Year Project April 2013
Supervisor: Dr Jonathan Dukes

School of Computer Science and Statistics

O'Reilly Institute, Trinity College, Dublin 2, Ireland
DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

______________________________________________        _____

Name                    Date
Acknowledgements

I would like to thank my supervisor, Dr. Jonathan Dukes, for being constantly available to impart his excellent advice and guidance throughout this project.
Table of contents

Declaration ........................................................................................................................................... 2
1. Introduction .................................................................................................................................... 5
   1.1. Aim .......................................................................................................................................... 5
   1.2. Motivation ............................................................................................................................... 5
2. Background ..................................................................................................................................... 6
   2.1. VLC .......................................................................................................................................... 6
       Interfaces ....................................................................................................................................... 6
       Transcoding in VLC .................................................................................................................. 7
       Streaming .................................................................................................................................... 7
       VoD ............................................................................................................................................ 9
       VLM .......................................................................................................................................... 9
       VLC on android ......................................................................................................................... 10
       Conclusion................................................................................................................................ 11
   2.2. Similar work ............................................................................................................................ 12
       VLC HDR .................................................................................................................................. 12
       VLC DIRECT PRO ................................................................................................................... 15
   2.3. Conclusion ................................................................................................................................ 16
3. Implementation .............................................................................................................................. 17
   3.1. Server ...................................................................................................................................... 17
       Detail of the script .................................................................................................................... 17
   3.2. Architecture VLC on android ................................................................................................. 22
       View Part ................................................................................................................................... 23
       Model ......................................................................................................................................... 27
       CONTROLLER ......................................................................................................................... 30
4. Test & result ................................................................................................................................... 35
5. Conclusion ...................................................................................................................................... 42
   5.1 Result ....................................................................................................................................... 42
   5.2. Achievements .......................................................................................................................... 42
   5.3. Future work ............................................................................................................................. 42
Bibliography ...................................................................................................................................... 43
Chapter 1

Introduction

1.1. Aim

My initial aim was to develop an android application which involves multimedia service. Special emphasis would be placed on such area as API android or streaming protocol.

The purpose of this project is to make an application which provides a multimedia player, stream media files from the network on an android device. Display this media on another screen (TV for example) and remote it.
In a second time, instead of stream from the server, the smartphone will be able to become a server itself and send his contents to the TV.

1.2. Motivation

The motivation behind this project was to utilize the programming and problem solving skills I have acquired during my three last year of school in computer science in France to tackle to a real problem.
This is also to learn the API android and improve my knowledge about streaming video.
Chapter 2

Background

The goal of this chapter is to provide a brief background to the project, including a quick introduction to VLC, transcoding, similar work that has been conducted and the approaches take.

2.1. VLC

VLC (VideoLanClient) is primarily a media player multi-platform system. It is also capable of outputting media to files, streaming over multiple protocols, including UDP/RTP/ MMS/HTTP unicast or multicast. It is able to stream over computer network and to transcode multimedia files. VLC is able to read more than hundred different formats of video or audio.

2.1.1. Interfaces

VLC provides several interfaces for controlling its operation. Their interfaces are suited to remote control from another program. To access of these interfaces, two solutions are possible:

- In command line : vlc --extraintf <interface>
- Via the graphics interfaces : Tools > Preferences > All > Main Interface

  a. Telnet

  This interface open the port 4212 by default on the server, VLC can be control using command line. The command “–help” display all the commands, among them there for example the basic control for a media as “play”, “pause”, “next”, ...

  It is possible to configure the host and the port using the option –telnet-host and –telnet-port.

  b. HTTP

  VLC provide also a HTTP interface which allows the user to control the player via a web interface or with some HTTP request. With this interface, it is possible to create easily a web page and control the player. This is an example of request to play:
It is possible to configure the host and the port using the option –http-host and –http-port.

### 2.1.2. Transcoding in VLC

Today, the new android device has double core around 1Ghz or more, but if you use an old android device and you receive a media from the network, you will have some problem to read the media. If you transcoding the media before sending it, it will be most efficient, because you reduce the quality to the android device so you reduce the size of the media.

To transcode a media on VLC, it must be say which kind of codec (audio and video) you wants to use, audio and video bitrate. Ex:

```bash
vlc -vvv input_stream --sout #transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}
```

### 2.1.3. Streaming

VLC allows the streaming via multiple protocol, in this document, we will describe the most important HTTP, and RTSP protocol.

#### a. HTTP streaming

HTTP Live Streaming is a streaming protocol based on HTTP. This protocol was develop by Apple for the system QuickTime X and the embedeed player iOS. It works in split the media flow in little file(ts). All this little part are list in a playlist file (m3u8). This protocol is able to cross the firewalls which allows HTTP standard traffic.

To enabled it in VLC:

- In the server:

  ```bash
  vlc -vvv input_stream --sout
  '#standard{access=http,mux=ogg,dst=server.example.org:8080}"
  ```

- In the client:

  ```bash
  ```
b. **RTSP**

To ensure the most correct streaming possible, the stream is split into packets whose size is adapted to the available bandwidth between the client and the server. When the client receives enough packets, the client application begins to play a package, unpacks another and receives a third one. Thus, the user can have a media stream without having to download the entire file. However, there is a delay due to buffering.

RTSP (Real Time Streaming Protocol) allows to control the distribution of multimedia over an IP network. This is an application-level protocol designed to run on protocols such as RTP / RTCP. So it uses the protocols previously seen.

With this protocol, the flow may come from a real-time source (camera, microphone). In addition to the actual data transfer, RTSP maintains a clock associated with each media objects transmitted. The clock is used to determine if the proportion of data to be transmitted has already passed the limit of time, in this case it will need to send data to the client who in any case will reject the packet.

RTSP can be used in applications as well as unicast multicast.

RTSP can control and synchronize multiple audio or video stream. It does not itself provide the flow that is to support other protocols such as RTP.

RTSP is similar in syntax and functionality, HTTP. It uses URLs and HTTP complementary mechanisms may be used.
There is no concept of connection in RTSP, although the server maintains a session with an identifier. An RTSP session does not correspond to a transport connection as TCP. During a session, RTSP can open and close several transport connections for each request. It is also possible to use a transport protocol such as UDP. Each presentation and each media stream is identified by an RTSP URL as for example rtsp://127.0.0.1:554/aud. For more details about the RTSP protocol, cf rfc2326.

In VLC, the rtsp protocol is use to do some VideoOnDemand(VoD) with the VideoLanManager (VLM).

2.1.4. VoD

With Video On Demand (VOD), the user can start the video when he wants, make pauses, go forward and back in the video. It is of course the best in video streaming and the dream for every user. VOD is a very big consumer of resources for the server and the network. VOD is unicast, not multicast : this means that the network and server resources needed are directly proportional to the number of clients.

2.1.5. VLM

VideoLAN Manager is a small media management tool designed to control multiple streams with only one instance of VLC. This allows multiple streaming and video on demand (VoD). It is possible to control by Telnet and HTTP interfaces. For this project, we will use the Telnet interface because it is less heavy to use telnet than HTTP to control the VLM.

a. VLM elements

A Media is composed with a list of inputs (the video and audio streams you want to stream), an output (how and where you want to stream them) and some options.

There are two types of medias:

- *vod*: A vod media is commonly used for Video on Demand. It will be launched only if a vod client asks for it.
- *broadcast*: A broadcast media is very close to a TV program or channel. It is launched, stopped or paused by the administrator and may be repeated several times. The client has no control over this media.

b. Example of VLM command

To add a new VoD media in the VLM, the syntax is :
To access to the flow:

```
vlc rtsp://serveur:554/myVideo
```

### 2.1.6. VLC on android

Before enter in the code of VLC, and write an app of a good quality, it is important to understand all the components and how their components are linked via the manifest android. This is a short description of them who will use in the following chapter.

#### a. Component of android app

- **Activity**

  This is the presentation of the app. Every screen is an extension of the class Activity. Activity use views or fragments to display information and answer to actions of the user.

- **Services**

  This component is running in background (without user interface). This is useful to update database, or play some music for example...

- **Intents**

  Framework of communication inter-application by passing some message. This is very used by android. This is use for example to start or stop activity, service or asking actions on particular data.

- **Manifest**

  The manifest is a XML file allows to define structure and metadata of the application, their components, prerequisite.

  Android implement the pattern MVC. The Model is define with the class as services, asynchrone task ... the View with the XML file ( layout ) and the Controller with the class extends Activity. Android provide a complete and very good documentation.


\textit{b. Compile VLC}

VLC is open Source, the source code is available here:

\texttt{git clone \url{git://git.videolan.org/vlc-ports/android.git}}

It is absolutely primordial to follow all the steps with the tutorial on the wikipedia page (http://wiki.videolan.org/AndroidCompile) to compile VLC on android with Eclipse.

\textbf{2.1.7. Conclusion}

For this project, VLC is a very good base; it actually allows to stream a media from the network via the VLM. We will use the telnet interface to control the VLM and the protocol RTSP to stream media file.
2.2. Similar work

The applications describe below are available on the Google play store and **FREE download**. These following tests are on a Samsung GT-P3110 (Galaxy tab 2), and 2 “computer”.

- The server running on Linux, Ubuntu 32bits v12.04 LTS, CPU i5 @2.67Ghz * 4 with 3,7Go RAM available.
- The client running on Windows 7 Familial version 64bits, CPU i5 @2,67Ghz *4 with 3,68Go RAM available.

The server running a HTTP and Telnet server via VLC.

First there is a description about the developer, the date and what doing the application and secondly the test did by myself.

2.2.1. VLC HDR

**Develop by :** Codejugglers

**Date:** March 2012

**Description:**
Allow streaming from VLC to android device
Allow streaming from android device to VLC

**Test:**
To configure, the user can inserts one or more IP address of computer. The switching between from one computer to another is easy. No more configurations are needed by default.
There are some bug on the audio player integrate in this application, difficult to know why without the code. However, that’s works with the video even if sometime the video is not running very well and poor quality due to transcoding option in poor quality and not adapt for tablet. We will describe below why:
As we can see, the output is specific for the “vb=320” means the video bitrate. We will now analyze these data to the original data of this video:

![Screenshot 1: VLM with video file](image1)

![Screenshot 2: information about the same video with mediainfo](image2)
Given that the original bit rate was 1319, that’s explain the poor quality. But this application has a settings menu for set all this information and adapt in function of the android device. The user needs to have some knowledge for set correctly. Volume can be changed in the control panel via a seekbar. The user interface need a little bit of practice for understand every possibility.

**Conclusion:**

Good application even if sometime it crash because it’s not responding.

User interface not easy to use at the begin.

Configuration OK for informed users.

Multiple server works
2.2.2. VLC DIRECT PRO

Develop by: Video & Streaming

Date: Free since March 2013

Description:

Remote control VLC from phone/tablet (read, play, pause, seek, fullscreen, volume)
Streaming from android to computer
Subtitles authorized

Test:

To configure the server, the computer need to download a script which run VLC with the web interface and passes it some parameters to find codecs and rtsp data.

To configure the tablet, I don’t need to precise the IP address/port of my server, it detect it automatically by scanning the network. However, it’s possible to define manually in the settings menu.

To switch from the tablet to the computer is possible but there is some problems when the user wants to come back on the tablet, the player on the tablet can’t read it ! And even if the user wants to switch again on the computer, it is not working ! It is difficult to understand exactly why without the source code, the log reveal just the usage of a watchdog so it is possible than one method on the tablet can’t synchronize with the computer.
This app creates a new flow RTSP on the tablet when the user plays the flow from the tablet to the computer, when he decides to seek, the computer crashes because the tablet looks like not be ready for stream this part of the video (cf RTSP protocol).

For the change the volume, this application uses a seekbar, and that’s work well. It looks like the application ask before the volume on the computer, adapt it on the tablet and then even if the user change the volume on the computer, the app receive an URL with the new percentage of volume.

Only one computer is authorized.

**Conclusion:**
- Good interface
- Seek and switching computer/android problems mainly.
- Settings ok

### 2.3. Conclusion

The features we want to add (streaming, control interface) can be done by VLC that’s why we take VLC as a base. The next step will be implement it.
Chapter 3
Implementation

This chapter will describe first, the script to prepare the server. In a second time, we will explain the interface of the existent application VLC for android and how it is possible visually to add our feature. Finally, we will give some information about how works the core of VLC and how implement our features in the code.

3.1. Server

The server is simply a computer running on Linux, where a script is execute and launch the program to search all the media available in a folder define by the user. I choose to do that instead of going directly from the tablet to the computer with the view to have multiple servers.

The aim is to launch from this script all the servers, and make just one playlist with all the Medias available on the network.

As we saw previously in the background, some multiple server application needs to switch all the time between the server for play some music, with this solution the user needs only to download one file and then he will have access at all the media without switching between servers.

Then when the playlist is done, the script launch the server and load the VLM with all the media available.

3.1.1. Detail of the script

In entry, the script needs one parameter: a path. It will look for media in this path and his subdirectory.

The script will compile a C program but before that, I need to check if the entire libraries are installing in a machine. I'm doing that with this command

```
PKG_OK=$(dpkg-query -W --showformat='${Status}\n' libavcodec-dev|grep "install ok installed")
```

dpkg is a tool use to display the information on the packet in the database of dpkg. « -W --showformat » display the packets who corresponding to the motif. Showformat is to define an output format, here if the packet is find, PKG_OK return « install ok installed ».  

17
I need actually three library, libavcodec, libavformat, mediainfo. The two first are vital to run VLC.

The next step of this script is to search the path of media file with the C program searchMedia.c.
Foremost, we need to create two different file, one with the path of the video and the other one for the path of the audio. I put in different file their path because I will need after to use a command (mediainfo) with different parameters depending on the type of file. It is explain in detail later.

When the files are created, I call the recursive method printdir which open the directory passed in parameter. Then, check all the elements, if it is a video/audio file, it's write in the right file and if it is a directory, printdir is called recursively. Imagine, if we have a depth in the hierarchy of 50, that can be dangerous for the stack.

Reminder about compilation design:

When a method is called recursively, it is called in the stack and in this case, if you try to execute the program from the root, the program can call a multiple of time the recursive function.

Effectively, from the root a error as "*** stack smashing detected ***" appear and stop the execution of the program.

This problem come from the system doesn’t have enough place in the stack.

As we now, more we call deep in the hierarchy, more we save return address, ex pointer of stack and local variable.

This is because in the first version (just for try), we use a buffer of 1000 octets for make a copy of the new path concatenate with the name of file.

To allow the program to access to more space in the stack, it is possible when you had the option -fno-stack-protector but it is actually very bad conceptually.

My second and final version was simply to avoid the local variable, manipulate directly.

This point was just to remind of importance of the stack when we use recursive function.
Finally when the program find a media:

- audio: write the url in list_urlAudio
- video: write the url in list_urlVideo

Each URL is separate with the character \n. It also write the file to load the VLM in VLC (named vod_conf). It define a new element and the path of it. Example:

```
new A1 vod enabled
setup A1 input /home/vincent/Desktop/media/music.mp3
```

**Question of time:**

- 13 min 34 sec in a hard disk of 204Go with 759 videos and 2275 audio files with a depth of 22 directory.
- 37 sec in a hard disk of 81Go with 240 videos and 676 audio files with a depth of 7 directory.

This time can be improve in using multithread, create different threads for a large repertory, and write it in a file with the help of a mutex.

It can be possible to reduce this time if we had some “black list”, a list where this folder don’t need to be because it is sure this is no media file interesting there like all the system file for example.

Last point about this program, I decide to do it in C because it was easier to me but in bash script, could be faster maybe. The time is not critical as an embedded system that’s why I allow me to do it.

In a second time when we have all the path about the media, now we need to read all the information about it and generate the playlist.

The audio and video file have different information, I need to call a function with different parameter depending on the type. Here we can actually understand why I make two different file in the C program. It is to avoid overloading the path with a tag for example for say if it is a video or audio.

It will be easier to explain the format with a example:

```
<AudioFile>
  <id::A65>
  <url::rtsp://192.168.0.12:5554/A65>
  <artiste::Emancipator>
  <album::Safe In The Steep Cliffs>
  <track::Ares>
  <genre::Electronica/Downtempo/Trip-Hop>
  <duration::>
</AudioFile>
```
Each file need to have a unique ID, I decide to choose to put A<number> for audio and V<number>. For each field, I use “::” as separator, will be useful to parse the field.

The URL is adapt here with the address of the server followed by the ID of the media.

In order to obtain all this informations, I use the packet “mediainfo”.

Mediainfo is a convenient unified display of the most relevant technical and tag data for video and audio files. It take in parameter the path of the media and display all the informations about it. In addition to this, it is possible to change the format of the answer using the option –Inform and select the field seeks with '%'. Example:

```
mediainfo --Inform= "General; <VideoFile>
    <id::V5>
    <url::rtsp://192.168.0.12:554/V5>
    <movie::%Movie%>
    <genre::%Genre%>
    <duration::%Duration%>
" /home/vincent/Bureau/video.avi
```

return :

```
General; <VideoFile>
    <id::V5>
    <url::rtsp://192.168.0.12:554/V5>
    <movie::Batman>
    <genre::Policier/Action>
    <duration::24525243>
/>```
MediaInfo analytical include:

- Container: MPEG-4, QuickTime, Matroska, AVI, MPEG-PS (including unprotected DVD), MPEG-TS (including unprotected Blu-ray), MXF, GXF, LXF, WMV, FLV, Real...
- Tags: Id3v1, Id3v2, Vorbis comments, APE tags...
- Video: MPEG-1/2 Video, H.263, MPEG-4 Visual (including DivX, XviD), H.264/AVC, Dirac...
- Audio: MPEG Audio (including MP3), AC3, DTS, AAC, Dolby E, AES3, FLAC...
- Subtitles: CEA-608, CEA-708, DTVCC, SCTE-20, SCTE-128, ATSC/53, CDP, DVB Subtitle, Teletext, SRT, SSA, ASS, SAMI...

It is not all the extension possible to read with VLC but the principal are here. If one media is not readable, mediainfo return the template with just the field “id” and “url” filled.

After all, the VLC is launch with all the option needed, the HTTP server, VoD server, Telnet and rtsp port open, and the VLM from the file vod_config.
3.2. Architecture VLC on android

This part will explain first how the application works, explain the variation in the UI, and will finish by detail how it is implement in the code.

Image 3: Pattern MVC of VLC on android with the main class usefull for this project
3.2.1. View Part

The View part is the user interface, there is one for the video/audio player and the menu. VLC decide to structure his application with a unique mainActivity (in red) with different layouts (yellow and blue), all of them extend Fragment class.

The yellow part is named sideBar controlled by the class «sideBarAdaptater». This is actually the menu of the application. The trick is when the user wants to make it disappear, in reality, it's just make in state invisible and then the blue part can hold all the screen.

The blue part match each item in the menu. here we can see the audio browser, called:

Audio browser : « AudioBrowerFragment »

Video browser : « VideoBrowerFragment »

Directories : « DirectoryViewFragment »

History : « HistoryFragment »

Screenshot 7 : original layout of VLC android
At the left, we can see the video player activity, with a sidebar at the bottom with the control panel and at the right, the audio player activity with one part to control and the second one to display the information of the media (cover, title, album, ...)

In order to save logic of the application, in this project we will save the same system of Fragment. I decide to create two news fragments, one to write all the information about the server and the client, and another one to read the playlist available on the server. I will start to describe the interface and after how it works.

First, here opposite the fragment « NetworkFragment » correspond to the new item in the menu (cf next screen), where the user put all the IP of the server, the path to find the playlist, the IP of the client and the telnet port of the client. All the user interface is Android are made with XML.

PS : the port of the server is define by 4212 for the server. The path by default for the playlist is « http://ipServer:8080/playlist »

When all the fields are filled, a control will verifie if all the field are syntactically correct, if yes, the playlist is ready to download.
When the user come back in the menu, can now going into the fragment « Network ».

Currently, the menu always allow the user to go in the item “Network” even if the network is not define. This is a improvement for the future task.
This new fragment is dividing in audio and video where the user can select with a horizontal scroll. This part will read the playlist file, and add in the database all the media available on the server.

Currently, in this first version, I have only two tabs, Audio and Video. In another version, it could be good to have the tab album, genre and artist.

Player remain the same design except one detail. A Toggle button appears, by default the media is read on the android device, but when this button is enabled, the media stop on the tablet and is launch in the client configured. From this moment, the android device become a remote control.

When the user decide to disabled the toggle button, the player vlc come back at the same point in the video/audio where it was play on the client.
3.2.2. Model

This section describes how to collect the data from the UI, how it is represented, and which class needs to be created to integrate the network.

In order to store the network data in the application, I created the class “Network”. The attributes of this class are:
- IP server
- Port HTTP Server
- The path of the playlist on the web server
- IP client
- Port Telnet Client
- The password for the telnet connection

All this information will be completed by the interface “Network”. The format of the IP address is checked with a regex for IPv4.

Then, we need to download the playlist file; we will use the different methods of the class.

The methods are:
- Network: constructor
- downloadPlaylist: Download the playlist
- testTelnet: Test the telnet connection
- sendTelnet: Send an order
- endTask: report the answer of the order and ask to update the UI

To download the file, the downloadPlaylist method is called with different parameters:
- progressDialog: to inform the user of the progress
- textview: to inform the user in case of failure
- context: to access to the good user interface

DownloadPlaylist is a class that extends AsyncTask because we will use this class to have access to the network, which means the possibility of freezing on the network. That's why it is necessary to use a thread.

AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.

In our case, Params will be String (the url of the playlist), Progress a integer (like a percentage) and Result will be a String (result of the download).
OnPreExecute, onProgressUpdate and onPostExecute are callback function and can be executed only by the thread UI. It is very important because, imagine the thread which download the file can access to the UI, if this thread crash, the application crash because after 5 sec of freeze the dalvikvm (process virtual machine (VM) in Google's Android operating system) considers the application are not responding and crash. If the developers try to execute some network task in the UI thread, the dalvikvm will throws an exception anyway.

Thread in java use the method Run for make the treatment. With the asyncTask, this is the method doInBackground. Here it is define a classic way to download a file via URLConnection. This method will write the playlist in the directory vlc/playlist which is concretely on the hard disk of the android device. Each time the loop read something, it updates the progress bar. Then when the treatment is over, the thread UI is called to execute the method onPostExecute, this is when the information message can be write to inform the user via the textview passed in parameter.

So now, the android device has downloaded the playlist file, it will be interesting to know how it will interpret it. In VLC, one media is representing by the existent class Media. The attributes of this class are the Title, Artist, Genre, Album, etc ... and the main method the constructors and “extractTrackInfo”, both of them will initialize all the data of the media.

MediaLibrary is a class of VLC also who can initially load only file from the hard disk of the android device. Now, it will need to load media from the custom playlist file. For this we need another thread, always for the same reason, not block the UI thread. GetMediaItemsPlaylistRunnable extend AsynchTask will read the playlist and add each media in the ArrayList<Media> attribute. To decode the playlist, the class parsePlaylist is defined. It parse a first time with the regex '/>' to select one media, parse with the regex '<*>' to select one field and finally with regex "::" to select the key and the value.

For example :

**Entry :**

```xml
<AudioFile>
  <id::A1>
  <url::rtsp://192.168.1.11:5554/A1>
  <artiste::Irma>
  <album::Letter To The Lord>
  <track::Your Guide>
  <genre::>
  <duration::188238>
</>

<VideoFile>
  <id::V1>
  <url::rtsp://192.168.1.11:5554/V1>
  <movie::James Bond - Skyfall>
  <genre::VOSTFR by MiND>
  <duration::2912256>
</>
```
Applying split("/>") :

```xml
<Applying split("/>") :

<AudioFile>
  <id::A1>
  <url::rtsp://192.168.1.11:5554/A1>
  <artiste::Irma>
  <album::Letter To The Lord>
  <track::Your Guide>
  <genre::>
  <duration::188238>
</AudioFile>
```

Applying split("<*>") :

```xml
Applying split("<*>") :

<AudioFile id::A1 url::rtsp://192.168.1.11:5554/A1 artiste::Irma album::Letter To The Lord track::Your Guide Genre::
duration::188238
```

Applying split ("::") and select the right part :

```xml
Applying split ("::") and select the right part :

  A1
  rtsp://192.168.1.11:5554/A1
  Irma
  Letter To The Lord
  Your Guide
```

When we have all this information, we can create a new TrackList (class existant in VLC) for each media and precise the type. Now, the aim is just to complete the database in create a new constructor of Media with a TrackList in parameter.
3.2.3. CONTROLLER

The controller manages the event to update the view or the model. In VLC, the main class who do that is libVLC. libVLC is the core engine and the interface to the multimedia framework on which VLC media player is based. It allows developers to create a wide range of multimedia applications using the VLC features.

The software and the application for Android VLC works in the almost the same model. There is VLC which generate all the events.

LibVLC manage all these events, and call the method in libVLC core in order to execute really the action. It use all the different module use in VLC because indeed the software VLC uses a modular system, which allows to add easily new format and functions.

Libvlc is a singleton. For each application wrapped around libvlc, only one libvlc instance should be running. In order to create an instance of libvlc, you must call

If we come back in the Android application, libVLC exist and call some functions coded in C, C++ via the JNI (Java Native Interface).

To begin, we can describe how the player can read an audio file, video file. These two modules are needed by VLC to render decoded images on the screen and to output sound. Finally we will describe how, in this project can we manage all of those features.

a. Audio output

VLC android offers 3 different ways to read a audio, that can be configure in the preference.

- Audio track in Java
- Audio track Native
- openSL ES

Image 4: Architecture VLC
OpenSL ES, is a hardware-accelerated audio API tuned for embedded systems. Works on the low level audio API android. OpenSL is use just by the android device from Android 2.3 that's why they decide to create a another audio output module that might work on any android device regardless the version named AudioTrack. AudioTrack read audio flow directly from the hardware buffer. To instant a AudioTrack, you need to precise the frequency, channel Configuration, the type of encoding and the length.
The method write is use to add data in the buffer and play for start the asynchronous reading.
If we write before in the buffer, the reading will start immediately, in the other case, the reading will start only when something is written in the buffer.

( PS : The Galaxy tab 2 GT P3110, VLC with openSL mysteriously crash ).

How it is implement ?

The view part ( audioPlayerActivity ) called the functions of the controller for the audio ( audioServiceController). This class implement IaudioPlayerController, this interface define all the basic method needed to run an audio file as play, pause, next ... All of these functions are define using “handy utility function” to reduce the code duplication as precise the comment in the code, all of these method call the same function and precise in parameter :

- **instance** The instance of IAudioService to call, usually mAudioServiceBinder
- **returnType** Return type of the method being called
- **defaultValue** Default value to return in case of null or exception
- **functionName** The function name to call, e.g. "stop"
- **parameterTypes** List of parameter types. Pass null if none.
- **parameters** List of parameters. Must be in same order as parameterTypes. Pass null if none.

And return the results of the RPC or default value if error.

This handy function create automatically a file IaudioService who extend Iinterface. Iinterface is the implementation in android of the volume service manager.
AudioServiceController is bind to the service AudioService on the method bindAudioService().

AudioService has an argument “serviceReceiver” of type BroadcastReceiver. The method onRecieve of the class BroadcastReceiver has intent in parameter, that inform the type of order the audioService receive. Depending of the type, audioservice use the reference to libVLC to launch the native function corresponding.

For this project, I will need to control the audio player, for example stop the player when the user wants to send the audio in the client. For this I will use a boolean “bNetwork” in the activity in order to know if the activity is currently playing in the android device or if the flows is send to a client and in this case, the control panel become a remote control.
In the method update(), we will check this boolean, if it is “true”, we disabled the visibility of the control as shuffle, random, next, previous and keep only the seek bar, play/pause.

Now for send a command, we will need a different new class:

- **Telnet.java**
- **message_telnet.java**

First, message_telnet is a class to represent a message to send into the network.

```java
public class message_telnet {
    public LinkedHashMap<String,String> msg;
    public enum TypeCommand {
        COME_BACK, GO_CLIENT, PLAY, NEXT, TEST, SEEK, PAUSE, STOP
    }
    public TypeCommand type;
    public message_telnet(String[] s, TypeCommand type) {
        this.type = type;
        msg = new LinkedHashMap<String, String>();
        if(s!=null) {
            for(int i =0; i< s.length;i++) {
                msg.put(s[i], "");
            }
        }
    }
}
```

A message telnet represent in fact, a sequence of command. All the command are passed in parameter by an array of String. Thereafter, those are represented by a LinkedHashMap<String,String>.

For be organized, it is better to use the system <key,value> of the map, key for the command, and the value is the answer. In return, it is not possible to send twice the same command in the same message but I have not found any utility to do it.

In Java, Hashmap doesn’t have order in the key, in our case the order is critical because if you send the sequence “add rtsp://192.168.0.12:5554/toto.mp3, seek 40” and you send “seek 40” first, the VLM will not understand. That’s why it is better to use LinkedHashMap, which is a HashMap with order.

When the message is initialized, the next step is to send it : Telnet.java.

This class extend AsynchTask, because it is a network task, it use the existant class TelnetClient to connect to the server, to send a command we just need to call the method println of PrintWriter.
For each command there is always an answer. All the answer finish by the caracter “>”. To read the answer, you just need to take everything before this character. (The telnet interface uses this character just for finish in the answer).

The answer is catch in doing a split() and take the left part and store it in the LinkedHashMap with the right key.

If the answer is empty, one exception is throws and the answer is an empty string. In case of error in the command the answer starts by “Error”.

By experience, the server is not fast enough so the thread must wait during at least 300ms between two commands.

In the server doesn't answer, the function return the error code otherwise the successcode.

The synchronization between the “network thread” and the “thread UI” is done via the method onPostExecute and call the method endTask(). For a future version that could be done by a handler it is better conceptually.

The different commands used in this project are:
GO_CLIENT : sequence of command (clear, add <path of the video>, seek <current _time>.
COME_BACK : sequence of command (status, get_title, stop)
Play : sequence of command (s_playing,get_title,pause)
Pause : sequence of command (s_playing,get_title,pause)
Stop : “stop”
Test : use for test the connexion
Seek <time> : “seek <time>”

About the Seek command, this is actually tricky because in order to have the same seekbar for the player AND for the remote. Actually, the seekbar is bind to the player. During the time where the android device is use as a remote, the seekbar is not moving forward.

In the future work, it will be possible to do it with a timer, who change every second the time on the seekbar and send a command “get_time” every 5 seconds for synchronize with the real time.

To interpret the answer, it just need to looking at the type of the message and act accordingly.

b. Video output

VideoPlayerActivity use directly the libVLC class because to play a video it doesn’t need to be on a Service. In other terms, if the user quit the application or switch the screen, the player must stop and not continue as the audio player.

But in the video, there is an audio part so VLC use the existent audioController as describe in the last paragraph but stop it when the active activity is not VideoPlayerActivity.

The user interface is break down in two parts. The video is display using a surface, and the control panel is a layout. In this project, there is no need to change anything about the video player.
To start a video, the method load (already existant is called), first this method recovers the path of the video, and then bind the audioservice with this media and launch the native fonction play() in the libVLC.

When the user pause the video, it just need to call the good function in libVLC, the handler of the sound is done in this method. But when the user stop the video, it is necessary to unbind the audioservice.

The aim of the control panel becomes the same than the audio when the android device becomes a remote. The same command are used to control the client.

There is the same problem with the seekbar then the audioplayer, and about the volume, it could be possible to increase or lower the volume on the client. It's not done in this version of the project but this can be done easily in asking to the client with the command “volume”, make a percentage and to adapt in function.
To proceed to a verification of the respect of the feature in this project, this is ten test about the main topic.

<table>
<thead>
<tr>
<th>TEST 1 : Configure VLM</th>
<th>PASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Purpose</td>
<td></td>
</tr>
<tr>
<td><em>Load correctly the VideoLanManager.</em></td>
<td></td>
</tr>
<tr>
<td>2. Methodology</td>
<td></td>
</tr>
<tr>
<td><em>Open the bash script Configure with in parameter the path of the library, write the right IP’s and port. VLM needs to be launch with all the data in vod_config (for try this, connect in telnet with the server and write « show »).</em></td>
<td></td>
</tr>
<tr>
<td>3. Expected result</td>
<td></td>
</tr>
<tr>
<td><em>The script generate list_urlAudio, list_urlVideo, playlist and vod_config file with the good value. Then launch the VLM and load all the media.</em></td>
<td></td>
</tr>
<tr>
<td>4. Actual result</td>
<td></td>
</tr>
<tr>
<td><em>Same as Expected result</em></td>
<td></td>
</tr>
<tr>
<td>5. Comments</td>
<td></td>
</tr>
<tr>
<td><em>One crash (core aborded cf screen below) appear once when the VLM launch more than 3000 files but was ok after. Error in VLC ?!</em></td>
<td></td>
</tr>
</tbody>
</table>
## TEST 2 : Generate playlist

<table>
<thead>
<tr>
<th>1. Purpose</th>
<th>PASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate a playlist file with the good format and good data.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Methodology</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the function mediainfo and look at the playlist file generate and compare to the real data of the video/audio</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Expected result</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The playlist file are generate with success, the data are correct for video and audio file.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Actual result</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Same as Expected result</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Comments</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## TEST 3 : Download playlist

<table>
<thead>
<tr>
<th>1. Purpose</th>
<th>FAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the file of playlist, Test of the HTTP class , good behavior of the User Interface.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Methodology</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill the form in the network settings, click on the button « Download Playlist », read file and check the right data.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Expected result</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Put right and fake data to try. File identical then in the server. Error reported in case of wrong value of data or unreachable host.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Actual result</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>With the right data : OK</td>
<td></td>
</tr>
<tr>
<td>With wrong value : error of behaviour of the IU when the data are incorrect</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Comments</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Just need to check more closely the answer of the thread DownloadFile</td>
<td></td>
</tr>
</tbody>
</table>
## TEST 4: Read/Send audio file

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Purpose</strong></td>
<td>Test of the behaviour of the UI in the player activity, all the network class, behaviour of the android device as a remote control.</td>
</tr>
<tr>
<td><strong>2. Methodology</strong></td>
<td>Setup the server, download the playlist, click to send the flow in the TV and listen if it is works.</td>
</tr>
<tr>
<td><strong>3. Expected result</strong></td>
<td>ToggleButton visible just when it is a media rstp. Stop the player when is enabled and start at the same point in the TV.</td>
</tr>
<tr>
<td><strong>4. Actual result</strong></td>
<td>Same as Expected result</td>
</tr>
<tr>
<td><strong>5. Comments</strong></td>
<td></td>
</tr>
</tbody>
</table>

## TEST 5: Read/Send video file

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Purpose</strong></td>
<td>Test of the behaviour of the UI in the player activity, all the network class, behaviour of the android device as a remote control.</td>
</tr>
<tr>
<td><strong>2. Methodology</strong></td>
<td>Setup the server, download the playlist, click to send the flow in the TV and listen if it is works.</td>
</tr>
<tr>
<td><strong>3. Expected result</strong></td>
<td>ToggleButton visible just when it is a media rstp. Stop the player when is enabled and start at the same point in the TV.</td>
</tr>
<tr>
<td><strong>4. Actual result</strong></td>
<td>Same as Expected result exept ToggleButton always visible</td>
</tr>
<tr>
<td><strong>5. Comments</strong></td>
<td>Because of time</td>
</tr>
</tbody>
</table>
**TEST 6 : Change progress audio**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Purpose</strong></td>
<td>FAIL</td>
</tr>
<tr>
<td>When the android device is use as a remote, test the synchronisation of the seekbar with the player of the client, possibility to change the progress.</td>
<td></td>
</tr>
<tr>
<td><strong>2. Methodology</strong></td>
<td></td>
</tr>
<tr>
<td>Launch a audio on the client, watch if the progressbar move along the reading of the audio file. Change the progress of the reading on the TV from the android device.</td>
<td></td>
</tr>
<tr>
<td><strong>3. Expected result</strong></td>
<td></td>
</tr>
</tbody>
</table>
| Seekbar synchronize with the audio player  
Seek the audio player on the client when the user change it from android |
| **4. Actual result** |   |
| Seekbar synchronize with the audio player : not done ( explain in the actual document )  
Seek the audio player on the client when the user change it from android : OK |
| **5. Comments** |   |
| *Need to implement a way to synchronize ( mix of Timer and command « is_playing » « get_time »)* |   |
### TEST 7 : Change progress video

<table>
<thead>
<tr>
<th>1. Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>When the android device is use as a remote, test the synchronisation of the seekbar with the player of the client, possibility to change the progress.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch a video on the client, watch if the progressbar move along the reading of the audio file. Change the progress of the reading on the TV from the android device.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seekbar synchronize with the video player</td>
</tr>
<tr>
<td>Seek the video player on the client when the user change it from android</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Actual result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seekbar synchronize with the audio player : not done ( explain in the actual document )</td>
</tr>
<tr>
<td>Seek the video player on the client when the user change it from android : OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need to implement a way to synchronize ( mix of Timer and command « is_playing » « get_time » )</td>
</tr>
</tbody>
</table>

### TEST 8 : Come back from TV (audio)

<table>
<thead>
<tr>
<th>1. Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the return on the android device when the TV is playing a audio file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the audio file on the TV, click on the toggleButton, see if the android player start again and at the same time than in the TV.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>The player on android takes over exactly where the progress was on the TV and the TV is stop.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Actual result</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need to check the title of the current file on the server. Because of time.</td>
</tr>
</tbody>
</table>
### TEST 9: Come back from TV (video)

<table>
<thead>
<tr>
<th>1. Purpose</th>
<th>FAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the return on the android device when the TV is playing a video file.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the video file on the TV, click on the toggleButton, see if the android player start again and at the same time than in the TV.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>The player on android takes over exactly where the progress was on the TV and the TV is stop.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Actual result</th>
</tr>
</thead>
<tbody>
<tr>
<td>The player on android come back but not at the same progress.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not implement. Because of time</td>
</tr>
</tbody>
</table>

### TEST 10: Remote Volume

<table>
<thead>
<tr>
<th>1. Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the volume when the android device is use as a remote control.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scroll on the screen to the high/down to inscrease/lower the volume</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume change on the client.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Actual result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not implement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not implement</td>
</tr>
</tbody>
</table>
This is a resume of all these tests:

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.Configure VLM</td>
<td>PASS</td>
<td></td>
</tr>
<tr>
<td>2.Generate playlist</td>
<td>PASS</td>
<td></td>
</tr>
<tr>
<td>3.Download playlist</td>
<td>FAIL</td>
<td>Crash if not good information, need to catch exception</td>
</tr>
<tr>
<td>4.Read/Send audio file</td>
<td>PASS</td>
<td></td>
</tr>
<tr>
<td>5.Read/Send video file</td>
<td>PASS</td>
<td></td>
</tr>
<tr>
<td>6.Change progress audio</td>
<td>FAIL</td>
<td>SeekBar synchronization to implement and test</td>
</tr>
<tr>
<td>7.Change progress video</td>
<td>FAIL</td>
<td>SeekBar synchronization to implement and test</td>
</tr>
<tr>
<td>8.Come back from TV (audio)</td>
<td>PASS</td>
<td></td>
</tr>
<tr>
<td>9.Come back from TV (video)</td>
<td>FAIL</td>
<td>Not finish to implement (Time)</td>
</tr>
<tr>
<td>10.Remote Volume</td>
<td>FAIL</td>
<td>Not implement</td>
</tr>
</tbody>
</table>

In general, the time is the main reason of the FAIL status.
The class download playlist(3) need to catch more exception, just need time.
The seekbar (6-7) in theory should work with the solution proposed previously.
Come back from TV video (9), use the same logical than for the audio player.
Remote Volume(10) just not implement but should be easy.
Chapter 5
Conclusion

5.1. Result
The main feature of this project works, even if there is some little bug to fix. The application of VLC can setup a network, read a playlist, and interpret it. The application can also be used as a remote control with another computer.

Unfortunately, I could not start to make the android device as a server.

5.2. Achievements
The first goal for me was to make an Android app with a large overview of all the technical resources of the API.
I gained a good knowledge about how works Android because I was not user, even less developer on Android four months ago.
This app still need some work to be really robust.

I learn also a lot about the possibility of VLC, all the side streaming I didn’t know before. And the integration in the code of the application on Android was not easy because of the lack of comments and documentation.

Finally, point of view linguistic for an Erasmus student, this project gave me the opportunity to improve my English. (Although it is not perfect yet)

5.3. Future work
The future work planned for android is firstly finish to fix the bug about the behavior of the User Interface on the NetworkFragment and the VideoPlayer.

Secondly, start the development for making the phone a server, as AirPlay on iOS.
Bibliography

Android API

VLC
http://www.videolan.org/vlc/


http://wiki.videolan.org/Hacker_Guide/Core (architecture VLC)

http://uv.utbm.free.fr/tutoriels/VOD_Streaming_avec_VLC/VoD_avec_VLC_v1.0.pdf (VLC on Android)


Java
http://www.java2s.com/Code/Java/Network-Protocol/ExampleofuseofTelnetClient.htm (Telnet)


Streaming

http://www.remlab.net/op/vod.shtml (RTSP by a VLC developer)
